Analyzing PI System Data

Version 2022

OSIsoft, LLC
1600 Alvarado Street
San Leandro, CA 94577

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of OSIsoft, LLC.

OSIsoft, the OSIsoft logo and logotype, Managed PI, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, OSIsoft EDS, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developers Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC Driver, PI Manual Logger, PI Notifications, PI ODBC Driver, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC DA Server, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, RLINK and RtReports are all trademarks of OSIsoft, LLC.

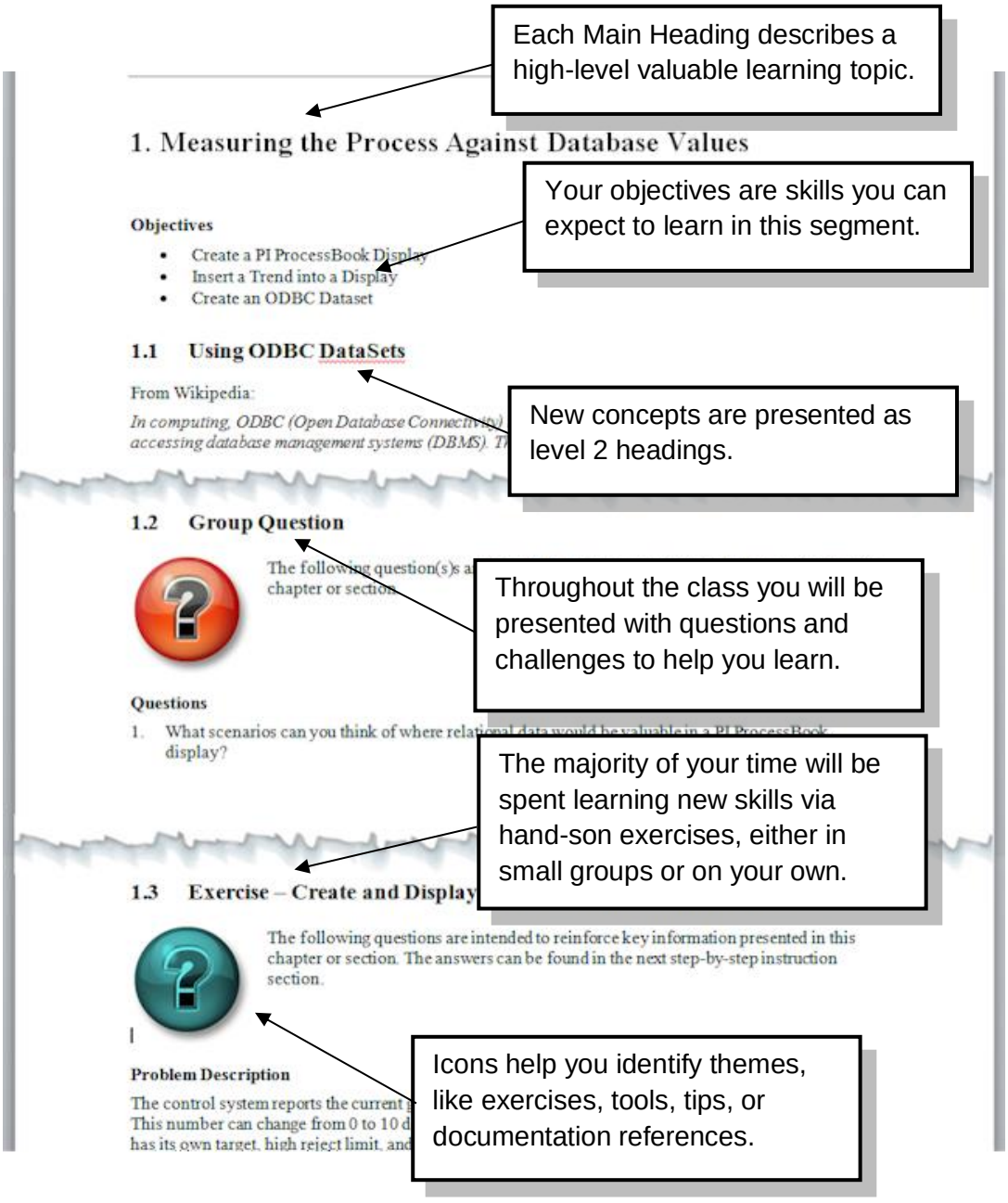
All other trademarks or trade names used herein are the property of their respective owners.

U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the US Government is subject to restrictions set forth in the OSIsoft, LLC license agreement and/or as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12-212, FAR 52.227-19, or their successors, as applicable.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording or otherwise, without the written permission of OSIsoft, LLC.

How to Use this Workbook



User manuals, Learning workbooks, and other materials used in class can be downloaded from http://techsupport.osisoft.com. Login to an OSIsoft technical support account is required.

Software Versions Used in this Document

The list below describes the software versions used in this version of the course.

Software	Version
PI DataLink	2019 SP1 Patch 1
Microsoft Office	2016
PI ODBC Driver	2016 R2
PI SQL Client	2018 R2
PI Integrator for Business Analytics Advanced Edition	2020 R2 Patch 1
PI OLEDB Enterprise	2019
Microsoft SQL Server	2016
PI Data Archive	2018 SP3 Patch 2
PI Asset Framework	2018 SP3 Patch 3
PI Vision	2021 Patch 1

Contents

1 Welcome

Welcome to the Analyzing PI System Data Course!

Since you are attending this class, you should have some experience with OSIsoft Client Tools: PI ProcessBook, PI DataLink, PI System Explorer, and PI Vision. Experience might include using displays, reports, BI tools, web applications, programming code to analyze your data, or it might include creating these applications so that others in your organization have access to all the powerful data that resides in the Data Archive and data external to the PI System.

The basic tasks within these tools are assumed to be understood. What you will experience here can be seen as a factory of ideas, a space for Aveva customers to experience how powerful existing data can be when analyzed with the advanced options of native and third party tools.

Hope you enjoy!

1.1 Course Environment

The environment for this course is being hosted with Azure. The environment has 3 VMs with the following software installed:

- PIDC – Domain Controller
- PISRV01 – The server environment
  - Microsoft SQL Server Standard 2016
  - PI Server 2018 SP3 Patch 2 (Includes PI AF Client, PI AF Server, PI Analysis Service, and PI SQL DAS RTPQ Engine)
  - PI Vision 2021 Patch 1
  - PI Integrator for Business Analytics 2020 R2 Patch 1 Advanced Edition
- PICLIENT01 – This is the primary working environment.
  - PI System Explorer 2018 SP3 Patch 2
  - Microsoft Office Professional Plus 2016 (64-bit)
  - Microsoft PowerBI Desktop 2.79.5768.721
  - Google Chrome 80.0.3987.149

The userid for each student is **pischool\student01**, the password will be provided by the instructor.

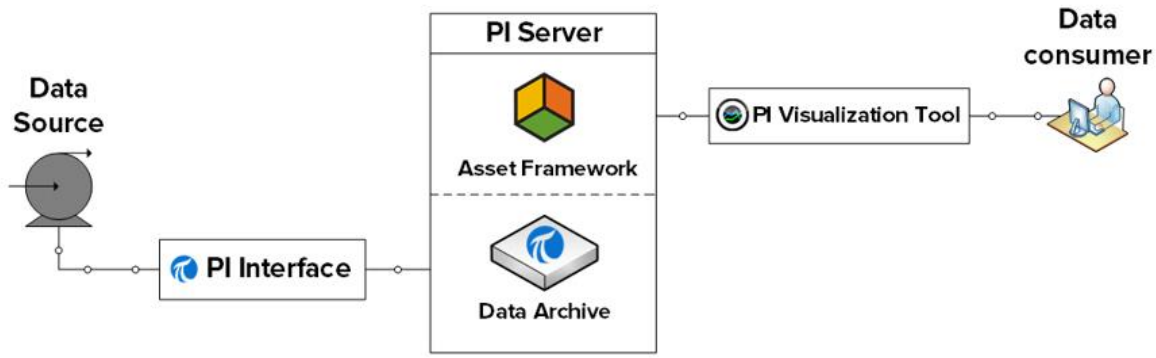
1.2 Review PI System Architecture

Objectives:

- Define the components of a PI System
- Draw a diagram of the architecture of a PI System

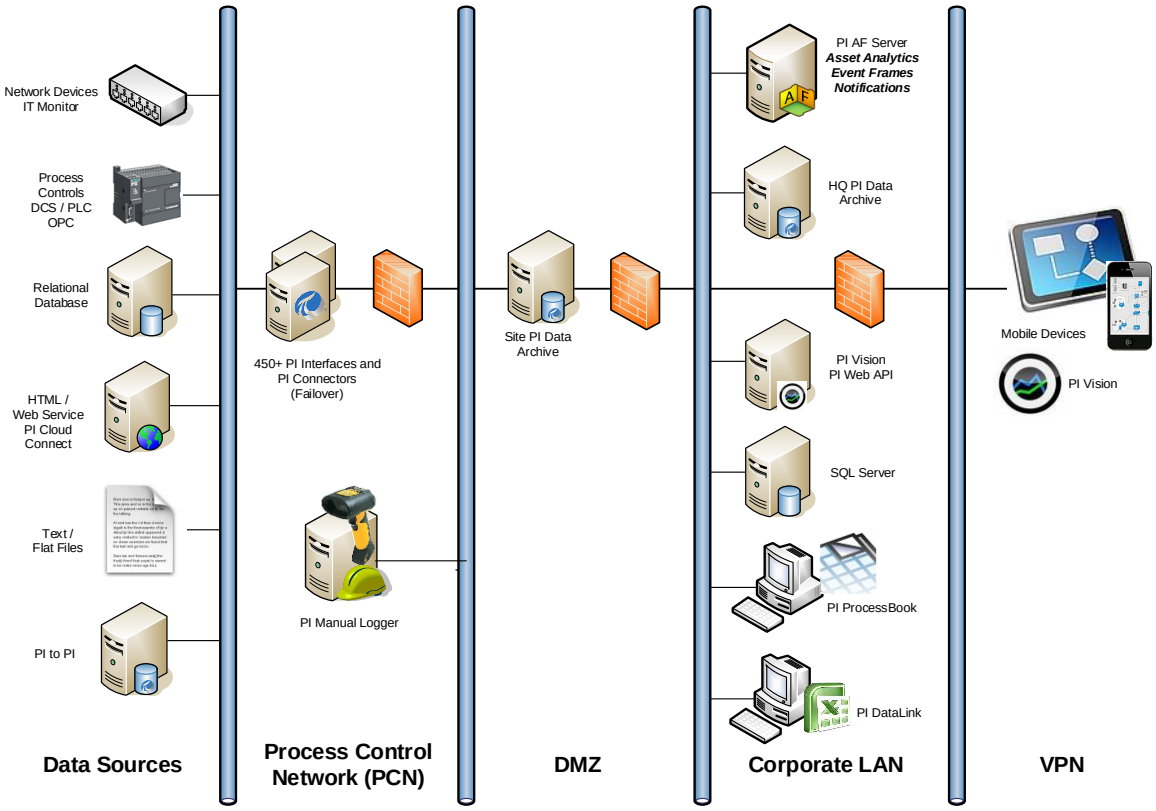
1.2.1 The PI System Described

The PI System collects, stores, and manages data from your plant or process. Products called PI Interfaces and PI Connectors read data from your data sources (control systems, instrumentation, etc), timestamp it, and write it to tags on the PI Data Archive. The data is organized and given context using PI Asset Framework. Users get data from the Data Archive and Asset Framework and work with it using a variety of client tools, such as PI Vision and PI DataLink.



1.2.2 Architecture of a Typical PI System

Sometimes the architecture can be very simple. Some customers have as few as one or two interfaces feeding data to a single Data Archive. Access to data is through the single Data Archive. In a large enterprise however, the architecture becomes more complex.



There are often several Data Archives in an organization, aggregating data from lower levels. Some corporations have Data Archives dedicated to servicing their clients with restricted company data.

1.3 Assets and Tags – The Basic Building Blocks in the PI System

Objectives:

- Define an AF Asset with its components element and attributes.
- Define four attribute types: Static (None), PI Point, Formula, and Table Lookup.
- Define a Data Archive Tag with the attributes Tag Name, Descriptor, and Point Source.
- Define the different data types that can be stored in Data Archive Tags.

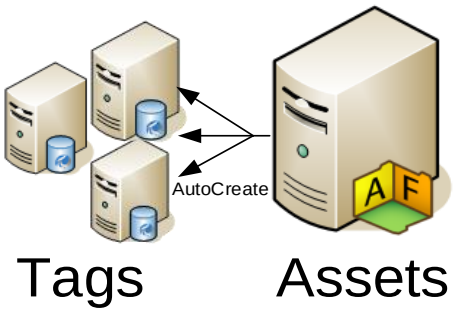


Figure 3: Assets and Tags

1.3.1 What is an Asset?

The AF Server is a part of the PI System. It contains asset or "metadata" usually organized according to the assets containing the attributes being monitored. AF can be helpful to users of the Data Archive who know the assets, but are not familiar with attribute nomenclature. With assets, data can be located without understanding the technical details of each piece of equipment. Organized assets help find all of the attributes associated with a specific piece of equipment.

1.3.2 What is an AF Attribute?

Attributes represent a unique property associated with an asset. The attribute maybe a constant, a value from an internal AF table, a value from an external database or a storage point for data in the Data Archive. An AF attribute is simply a single point of measurement. The point has been the traditional storage method of data in the Data Archive. The AF Server can automatically generate points as assets are created.

1.3.3 Some Basic Properties and Why They Are Important to You

AF attributes and Data Archive points have a set of properties that define them. Some common properties used in client tools are for display or informational purposes.

Attribute name

The attribute name is similar in concept to the point description. A detailed name for the attribute may help the user identify the source of the information.

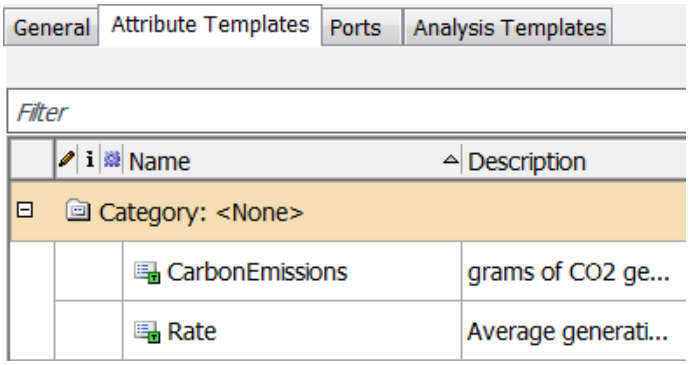


Figure 6: Attribute Name

Tag name

Unique name is used to create points for storage in the Data Archive. Points for data attributes storage can be built through AF templates using substitution parameters for ***local naming convention*** **or can be searched for on the Data Archive.** Creating points through templates, lends consistency in nomenclature making searches easier for PI Administrators. For example, which might be easier to locate in a search?

Point: M03_E1P1_MOTDRV1202_RUNSTAT

Attribute: Machine3 Enclosure 1 Panel 1 Motor Drive 1202 Run Status

Substitution parameters are variables placed in attribute templates for PI point and PI point array data references representing portions of the AF hierarchy.

For example, %Element% is a substitution parameter that represents the element name. After you create an element based on that template, you tell AF to create the data reference. When AF creates the reference, it substitutes the current element name wherever %Element% is present.

Descriptor

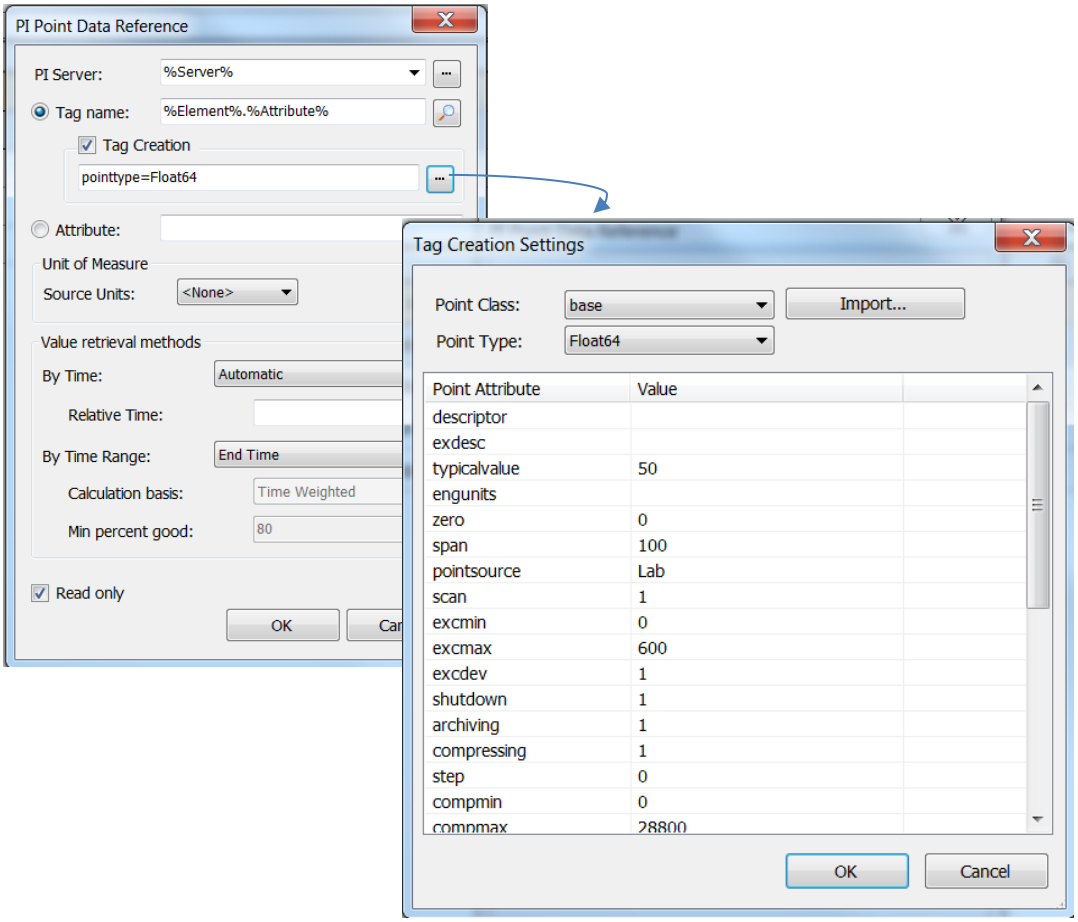
This is the human-friendly description of the Data Archive Point, similar to the attribute. The descriptor is often a **search criterion** since the point name is not always intuitive. Often the point name is some sort of abbreviated convention and the descriptor captures the "full name."

Point source

Points can be related to their interfaces that collect the data by a point attribute called ***pointsource***. Grouping by point source allows all of points associated with a particular device to be identified by searching for all points of a certain **point source**. This assumes that the user knows the point sources in use and that will not be true in most situations.

Point type

The PI point attribute that specifies the data type for the values that a point stores. The possible point types include int16, int32, float16, float32, float64, digital, string, BLOB, and timestamp.

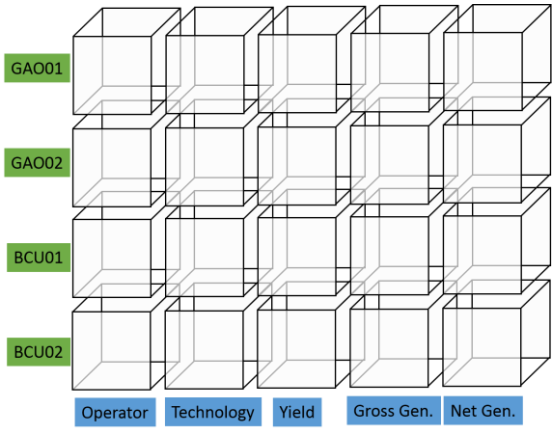


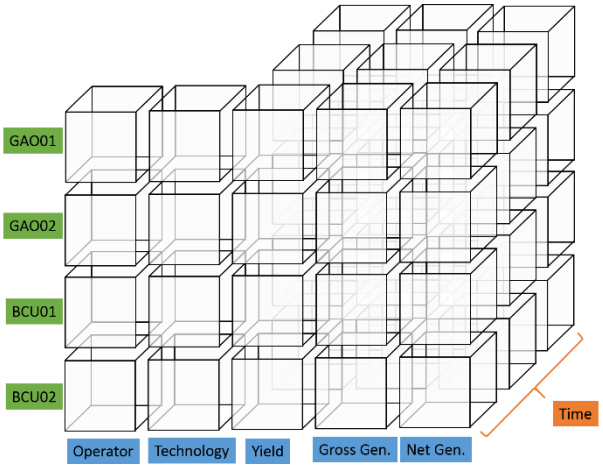
2 Business Intelligence

Business intelligence (BI) tools offer solutions to quickly analyze raw, un-normalized, multidimensional data. Values from the PI Data Archive, external metadata, and calculations from Asset Framework can be transformed by business intelligence tools into actionable analysis and interactive reports in order to gain insight into business and operational processes.

Later on in the course, we will explore the process of preparing the Asset Framework model to add additional dimensions of information to our AF database. The next step is extracting desired information (process data, metadata, and event frame data) from the PI System through PI Data Access tools. This data will be incorporated into a BI cube and used to develop interactive reports that allow us to "slice and dice" our data and bring meaning to our multidimensional data cube.

The Distribution Network and Fleet Generation databases have a comprehensive amount of information including a hierarchy of substations and metadata for each asset. The figure to the right depicts a data cube that captures metadata and real-time data of generating units.





Inclusion of additional attributes through table lookups and analytics on existing attributes allow for the expansion of additional columns (or dimensions) to the data cube above.

Further, historical data, interpolated or compressed, add an additional dimension of information that bring more meaning in Business Intelligence reports.

In the next several chapters in the course, we will be using a pair of AF databases to expose meaningful data that will help management and engineers make better, more informed decisions. Specifically, we will add value through the following:

1. Expose the database in a simpler structure for data processing.
2. Develop analytics within PI AF and PI Integrator for Business Analytics
3. Develop advanced analytics using Python and create forecast data
4. Import the data into Microsoft Power BI and prepare it for predictive analytics
5. Draw actionable conclusions from the resulting data sets in our reports

2.1 Intro to Power BI

Power BI is a business analytics service and client provided by Microsoft. It provides interactive visualizations with self-service business intelligence capabilities where end users can create reports and dashboards by themselves without having to depend on information technology staff or database administrators.

Some of the benefits of Power BI:

- Less work than Excel for more complex analysis and visuals
- Can solve problems that are simply too large for Excel and PI DataLink
- Cheap – Free download or $9.99 / month per user for Power BI Pro
- Live reporting and centralized web-based dashboards in Office 365 and Power BI Server
- Slick visuals including 3rd Party Visuals in Microsoft AppSource

3 Analyzing PI System Data with Power BI Reports and PI Integrator for BA (Asset View)

This course will be broken down into three main sets of exercises. In Part 1, we'll use PI Integrator for BA asset view to publish data from PI and spend a lot of time configuring Power BI. In Part 2, we'll get to know event frame and streaming view in PI Integrator for BA, test some python code to predict the values. In Part 3, we'll make modifications to a PI AF hierarchy and then use prebuilt SQL tables to create a report in Power BI.

In Part 1, we will be working with a data set for a fleet generation company, which includes different KPI characteristics for 30 units like gas and steam turbines and other. The source data will be published in a data-science ready format using PI Integrator for BA. Once this is done, we'll configure an array of Power BI visuals and integrate the results with PI Asset Framework and PI Vision.

Fleet generation database is not complete. We are going to add more analytics in it later in Part 3. For the current exercise this information is enough to get a good understanding of the data.

3.1 Directed Activity – PI AF Hierarchy and Data Set

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

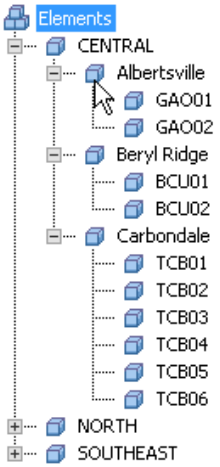
Objective:

- Better understand the data set used in the following chapters

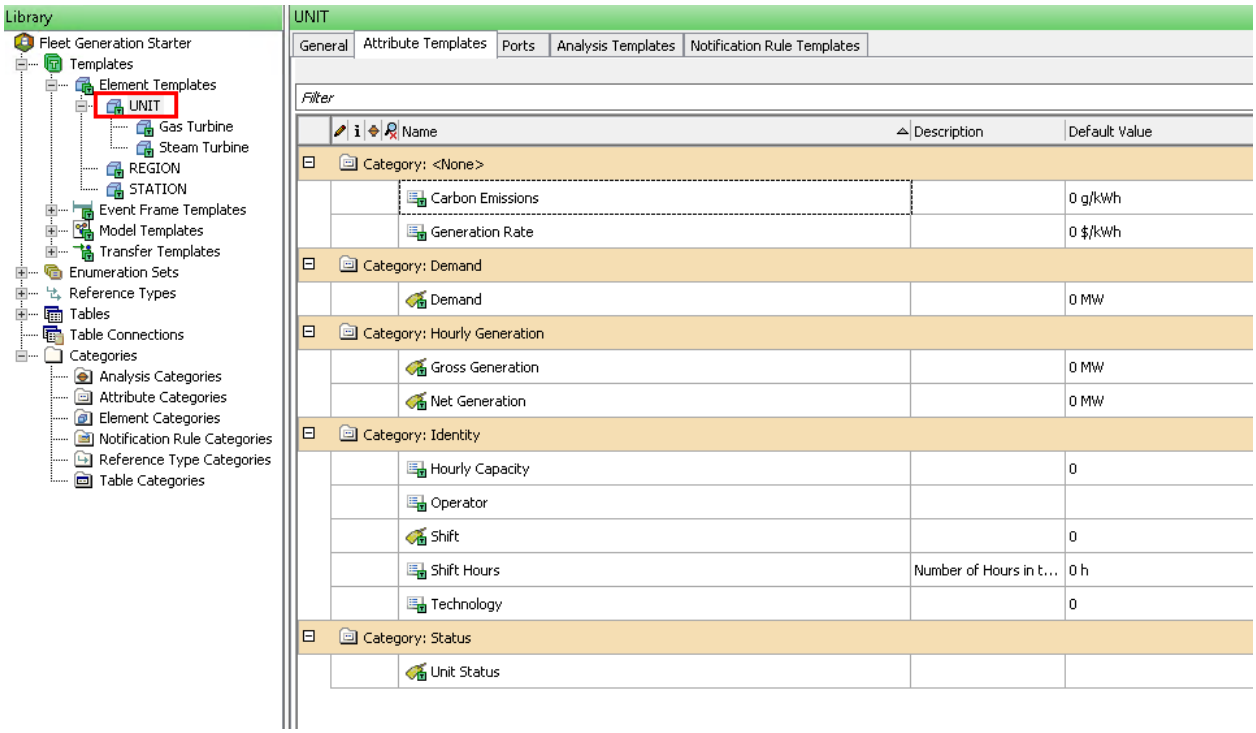
We will take a few minutes to understand where the data set came from and relate the sample Power BI report back to the PI System. We are working with a data set for a fictitious fleet generation company. They have built a PI AF Hierarchy for their units serving a number of geographical areas. In this course, we will focus on analyzing these units, which generate energy using different technologies.

Open PI System Explorer and head to the **Fleet Generation AF database**. Drill down to a level with unit (names starting with GAO_, BCU_, etc.) and inspect the available attributes. We will be using a sub-set of these attributes for all of our analysis, in addition to leveraging the AF hierarchy.

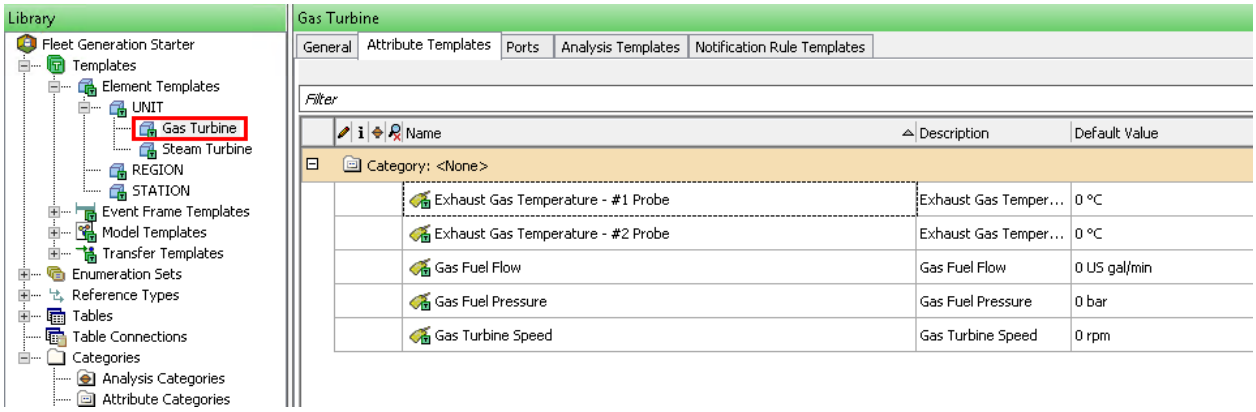
Browse the hierarchy, which is organized into Region, Station, and Unit.



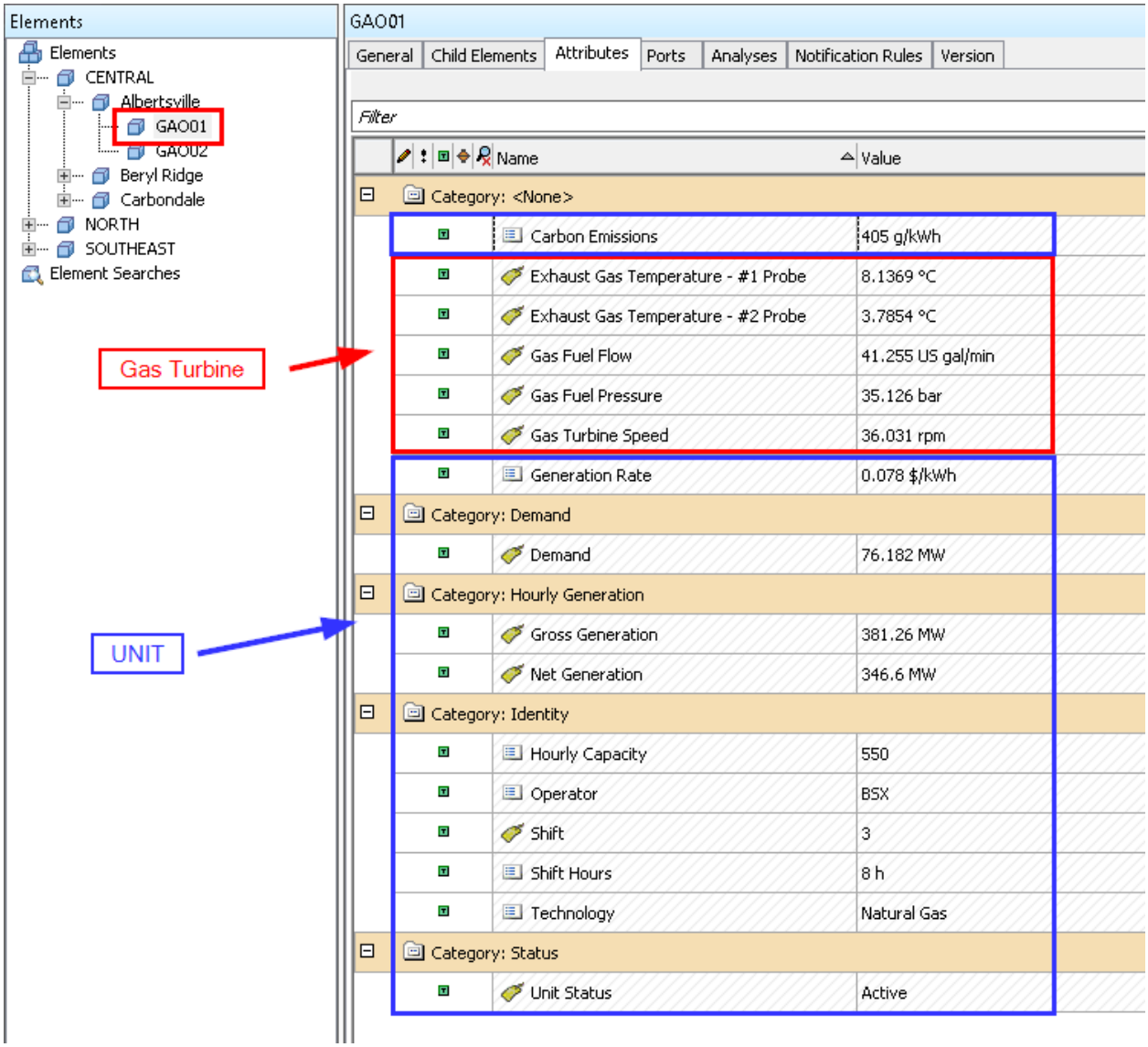
Most of the child elements are based on the generic **Unit template**.



Those in the CENTRAL region are based on the **Gas Turbine template**, which is derived from the UNIT template and has additional attributes.



Gas Turbines have all the attributes from the Gas Turbine template, but also inherit those from the UNIT Template:



Data from this PI AF hierarchy will be published for use in a Power BI report in a later exercise.

4 PI Integrator for Business Analytics

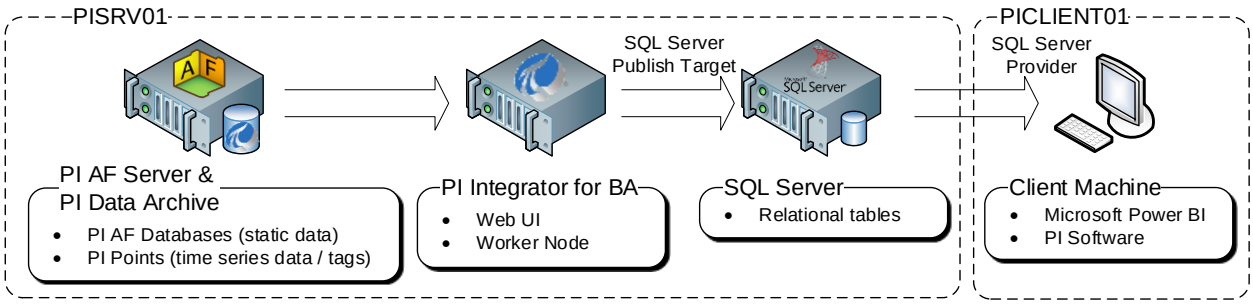
Getting the data out of the AF structure and into the client tools requires the use of integration software such as the PI Integrator for Business Analytics or PI System Access software such as PI SQL Client. Another option is custom application, which is using AFSDK calls for example to get AF data to a 3rd party client. This chapter will discuss the former method of extracting the data.

The PI Integrators join your Business Intelligence (BI) infrastructure with OSIsoft's PI System, allowing you to combine high-value Operation Technology (OT) data from the PI System with Information Technology (IT) data for reporting, analytics, and application integrations. The integration of data from OT systems, such as automation and control systems and internet-enabled devices, with data from IT systems, such as transactional and business process systems, increases situational awareness, adds transparency into industrial operations and business processes, and makes it possible to anticipate problems and identify opportunities for process improvements.

4.1 Architecture Used in Class

In this course, all server roles including PI Data Archive, PI AF Server, SQL Server, and PI Integrator for BA are all installed on PISRV01. In a production grade architecture, each role would typically have its own dedicated server.

There are many configuration options, which we will touch on later in the course, but in this course we will only configure PI Integrator for BA to publish data to a SQL Server and then use the native SQL Server provider to import the data into Microsoft Power BI.



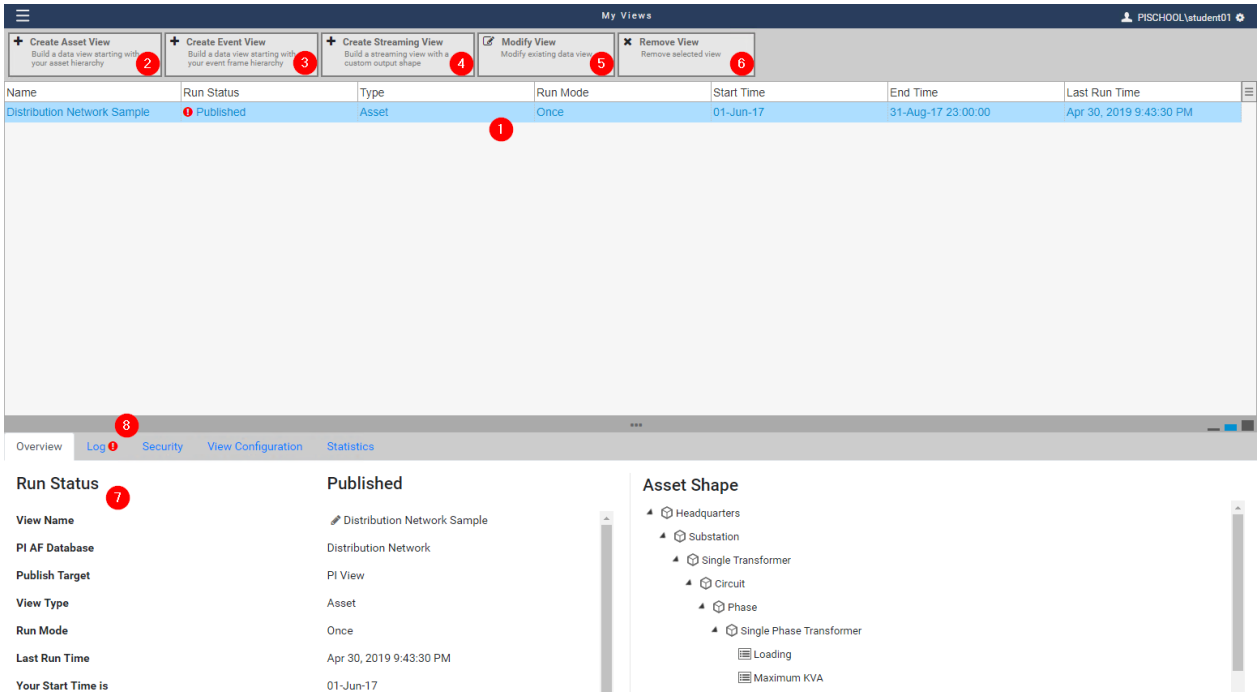
4.2 PI Integrator Web UI

The PI Integrator for Business Analytics site can be accessed via https://pisrv01.pischool.int:444 or from the desktop. If prompted for credentials, enter your student account, as this has been given access rights.

Views can be created within the PI Integrator portal that is hosted on PISRV01.A list of previously generated views is present within the portal on the **My Views** page, allowing for previewing and maintenance. These existing views can also be cloned and modified, allowing different views to be created and utilized within BI client tools.

The following is a breakdown of the **My Views** page layout, and the different operations available.

Note: The information regarding the My Views page layout is available within the PI Integrator for Business Analytics User Guide.



The My Views page shows details about your views.

1. All the views to which you have access are listed in the table
2. Click to create an **Asset View** that is based on Elements and Element Templates
3. Click to create an **Event View** that is based on Event Frames and Event Frame Templates
4. Click to create a **Streaming View** for publish targets that support streaming such as Apache Kafka, Azure Event Hub, and Azure IoT Hub.
5. To modify a view, select the view in the table and click **Modify View**.
6. To delete it, click **Remove View**. Deleting a view removes data from the buffer, therefore freeing up space. However, this does not free up the available output streams allowed with your license.
7. For the selected view, the Overview, Log and Security tabs provide additional details about the view.
8. The red message counter icon at top right show that there are warning and error messages recorded by PI Integrator for Business Analytics. Click the icon to open the message list.
9. Click the gear icon at top right to see the version of PI Integrator for Business Analytics and AF you are using.

4.3 Directed Activity – Create the Fleet Generation View

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

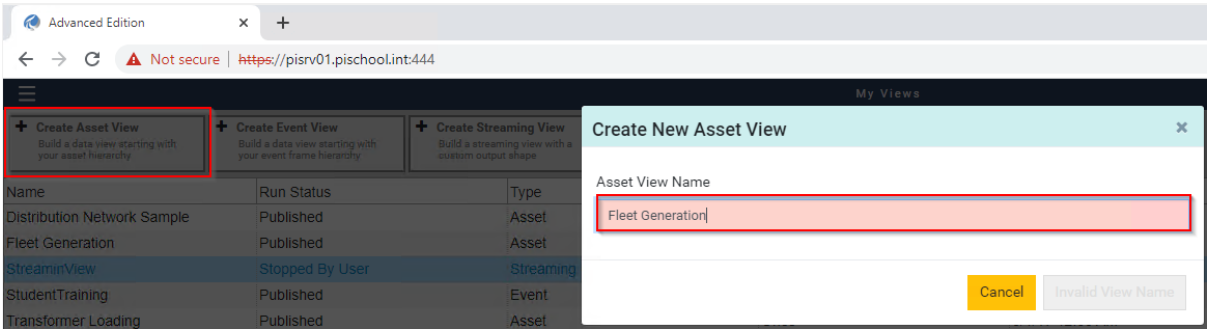
Objective:

- Use the PI Integrator for Business Analytics to create an Asset View, which will be used in later exercises.

Approach:

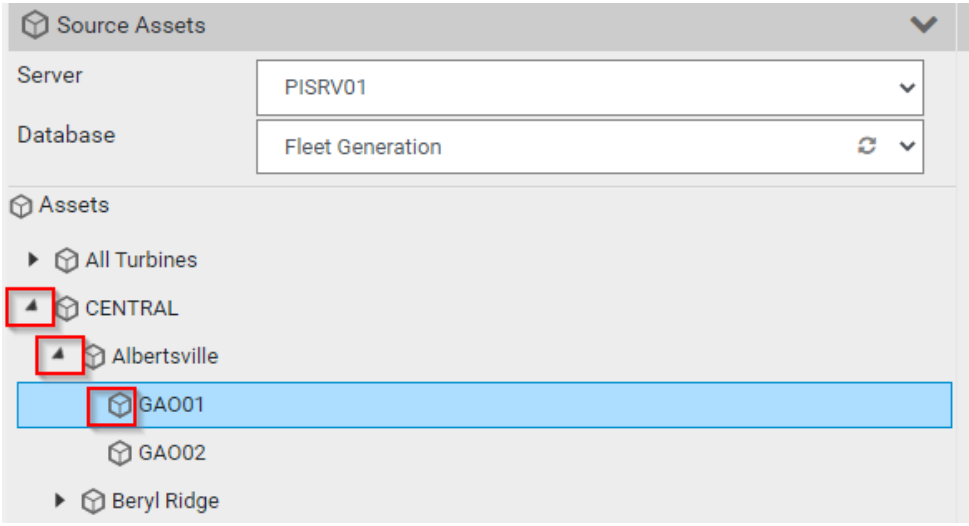
Open Google Chrome and Navigate to the PI Integrator for BA Web UI at https://pisrv01.pischool.int:444

Click **Create Asset View** and name it Fleet Generation, click **Create View**:

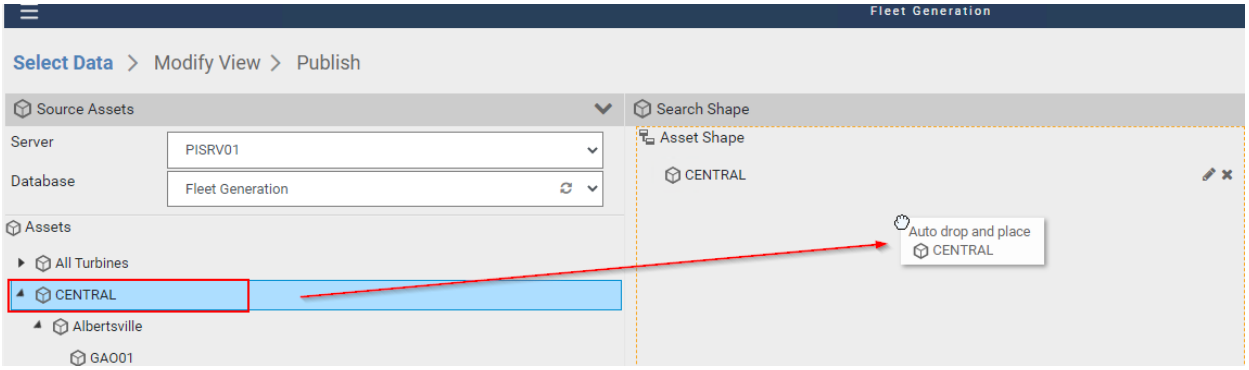


Click **Create a New Shape**

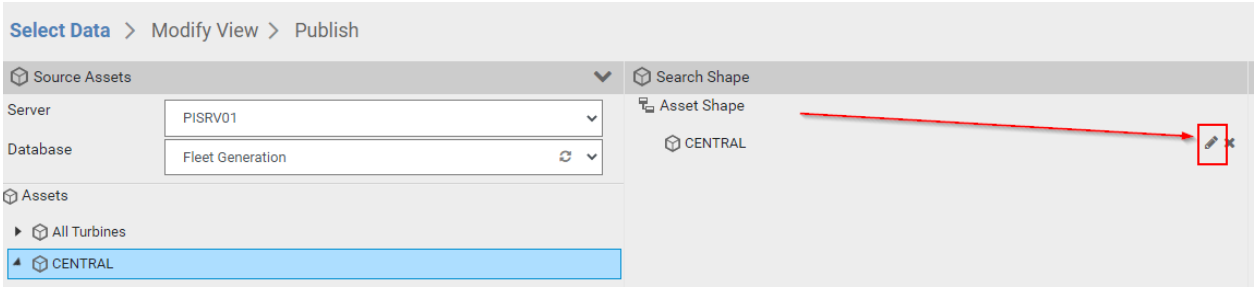
Select Fleet Generation as the AF Database, then drill down to GAO01:



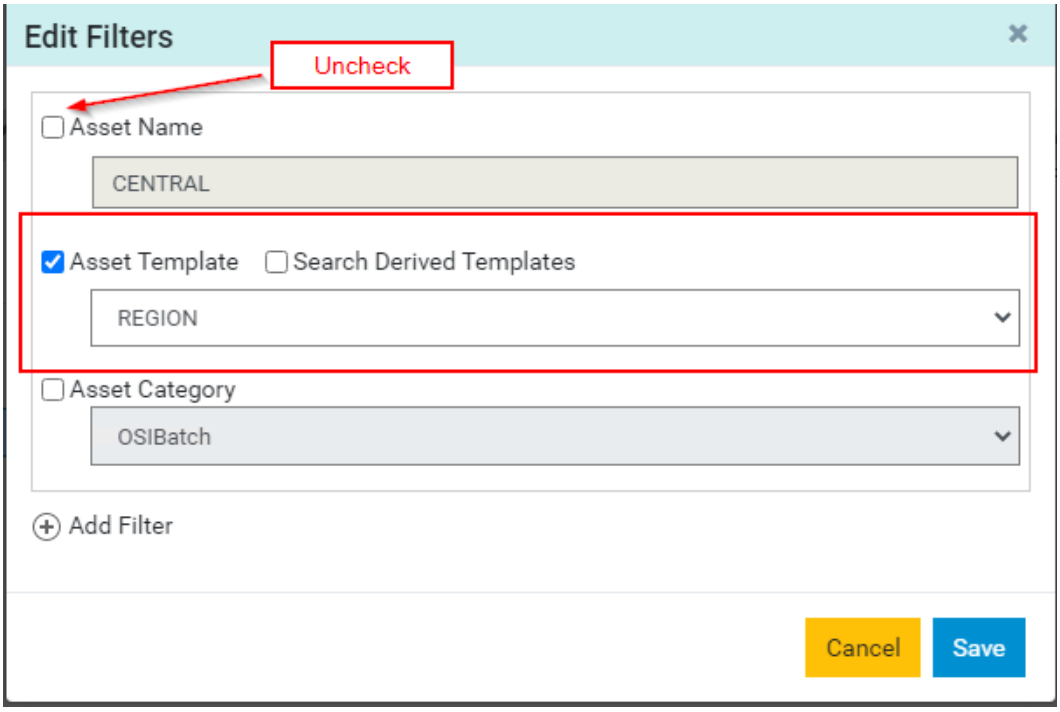
Drag and drop Central to the Shape Builder



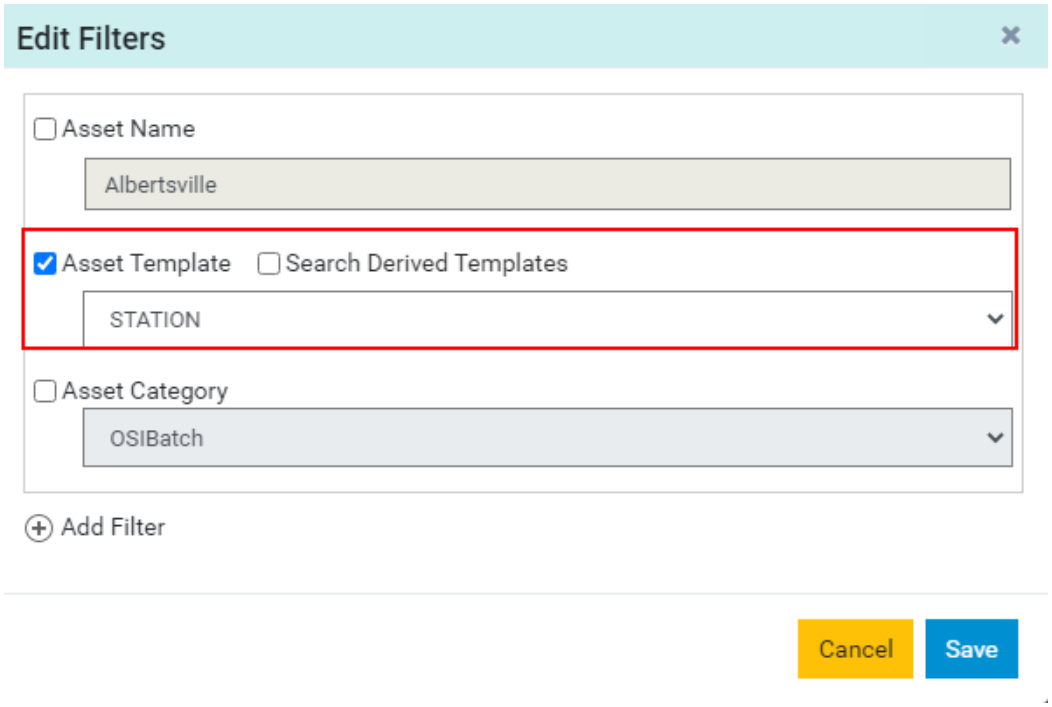
Edit the Filter on Central:



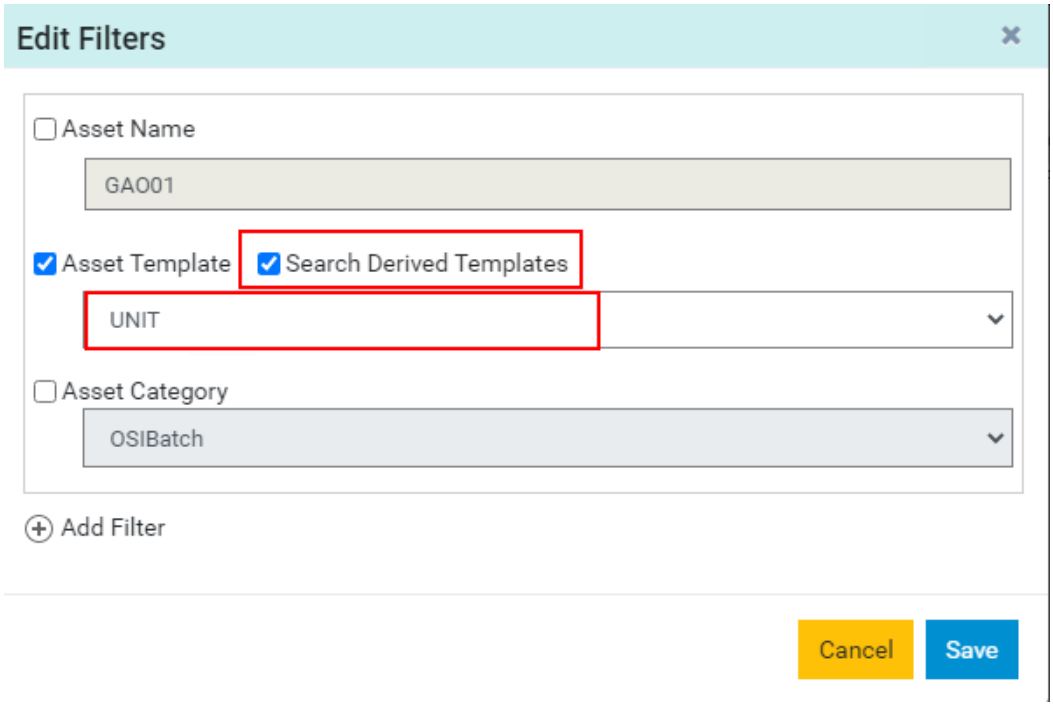
Clear the Asset Name Checkbox, Change it to filter on the **Region** template, click **Save**:



Drag and drop **Albertsville** to the Shape configuration, and change it to filter on the **Station** Template:

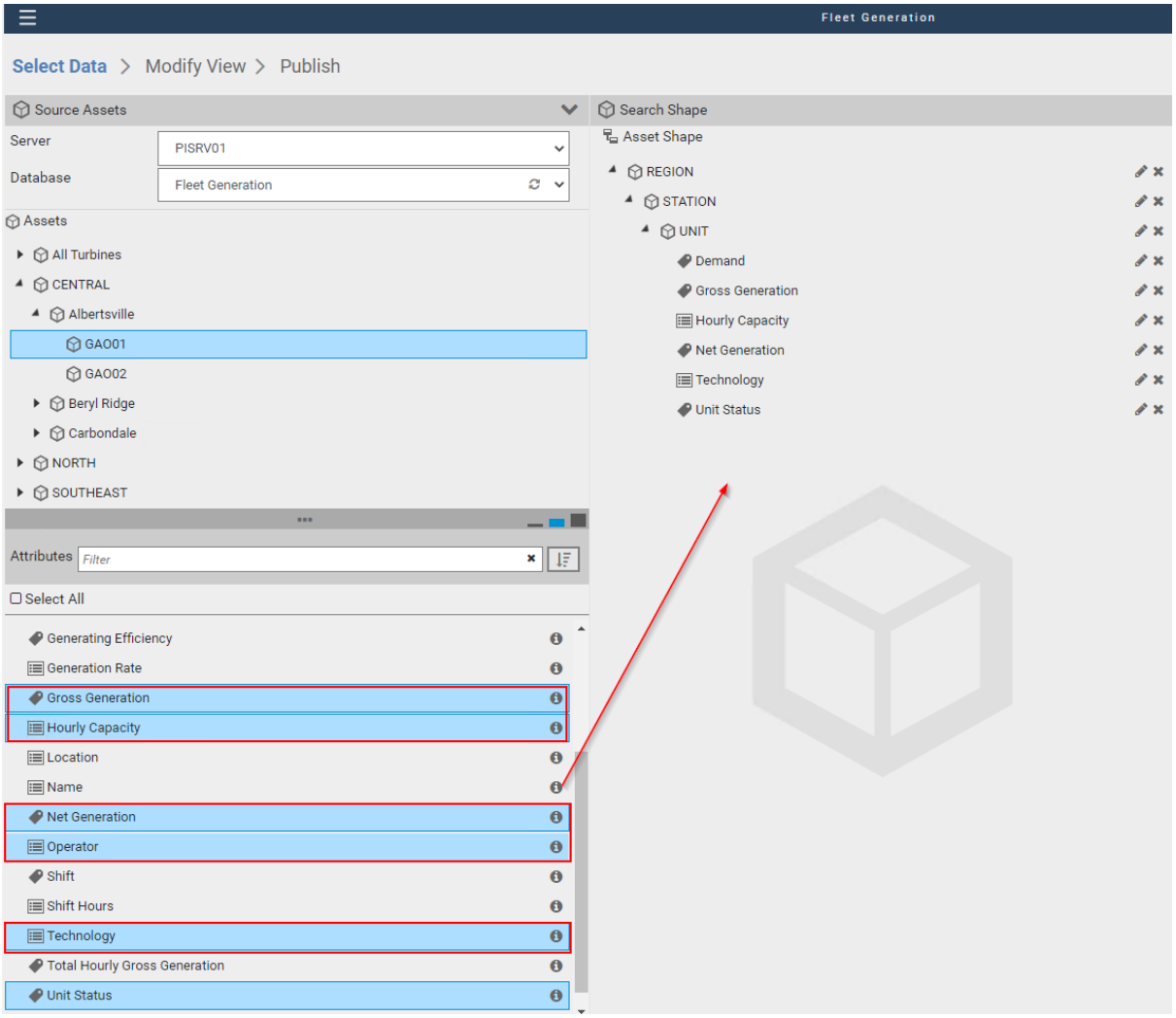


Drag and drop **GAO01** and select **Unit** as the Template, this time check the box to search derived templates.

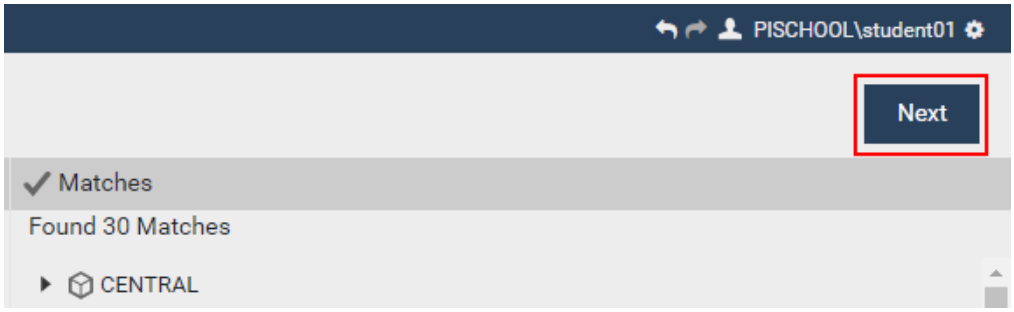


The shape configuration should look like this:

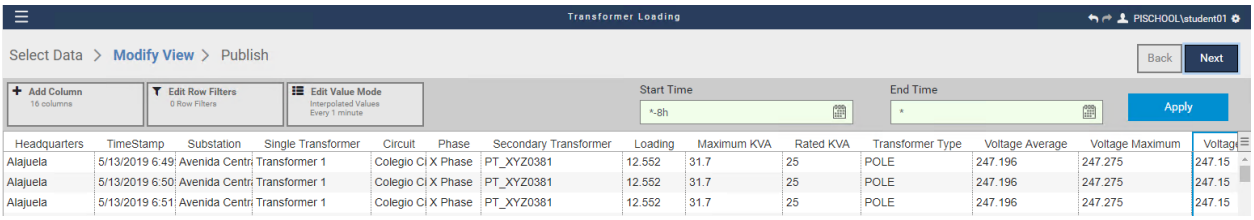
Click **GAO01** then **hold control** and multi-select **Demand, Gross Generation, Hourly Capacity, Net Generation, Technology and Unit Status**. Drag and drop these selections to the Shape configuration.



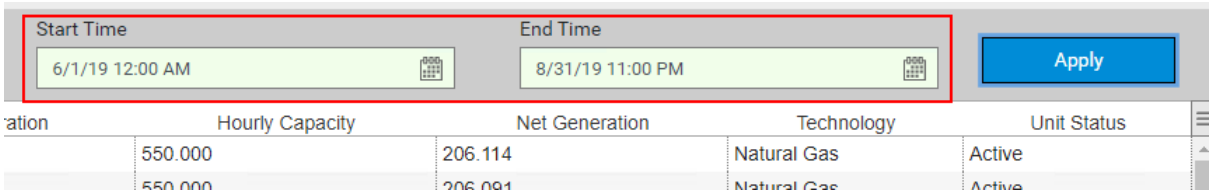
There should be 30 matches in the preview, click Next in the top right corner.



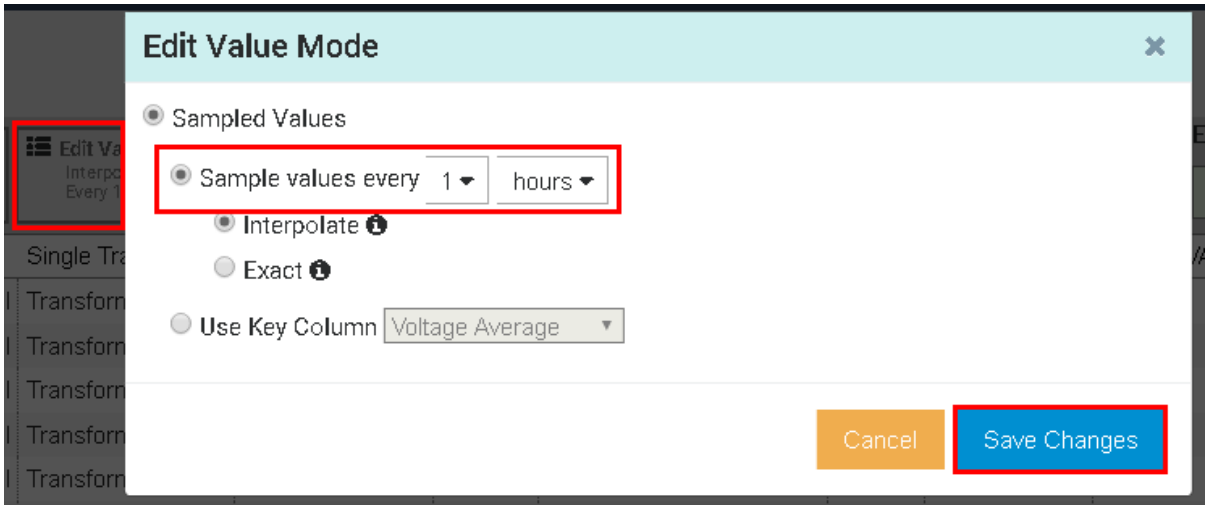
We now see a preview of the data using the default Time Range and interpolation mode.



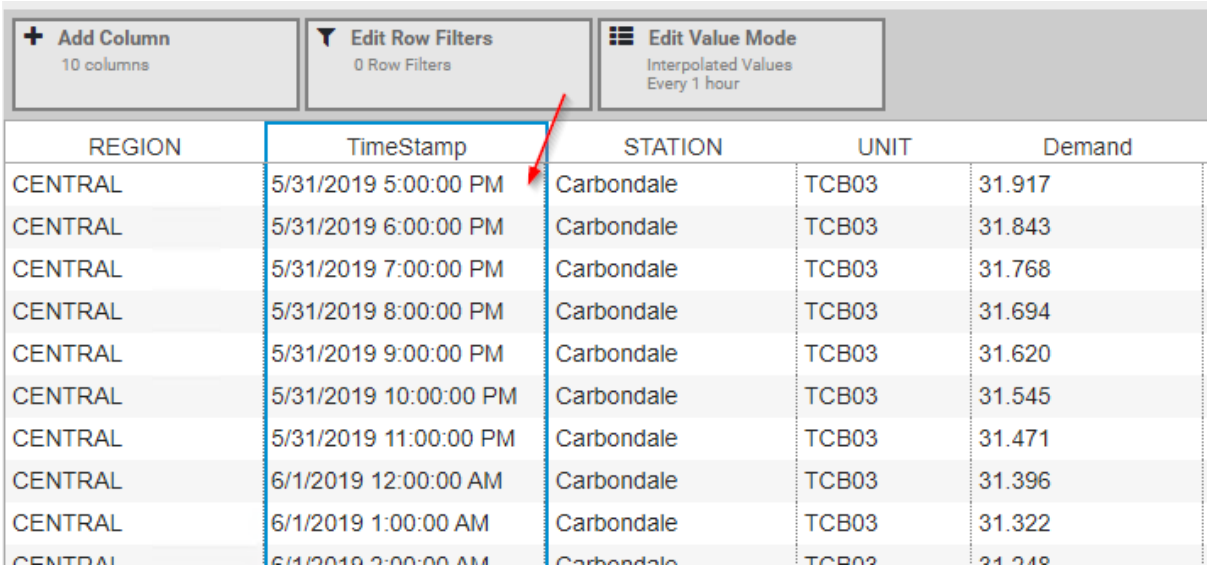
We want to publish Hourly data for the time period 01-Jun-19 00:00:00 to 31-Aug-19 23:00:00. Modify the Start Time and End Time and click Apply:



Click **Edit Value Mode** and change the time step to 1 hour, then **Save Changes**:

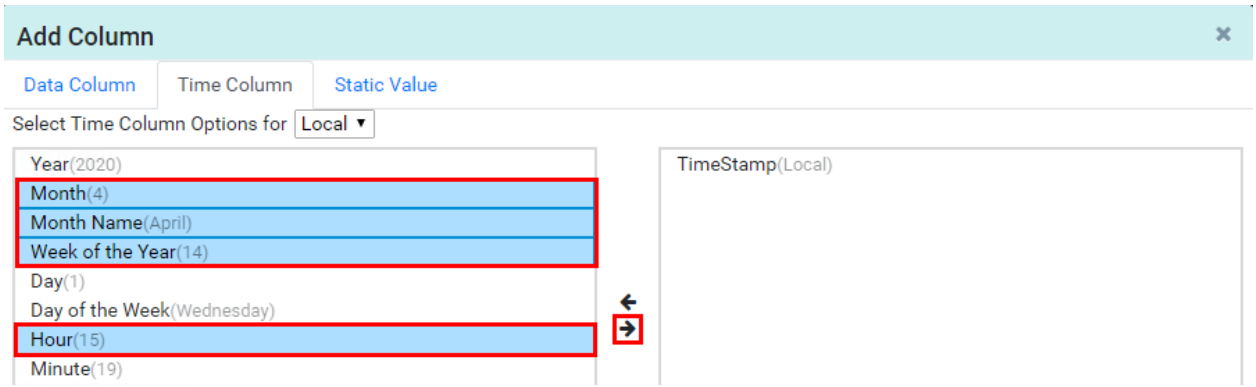


The TimeStamp column should now reflect changes to the Start, End, and Value Mode:

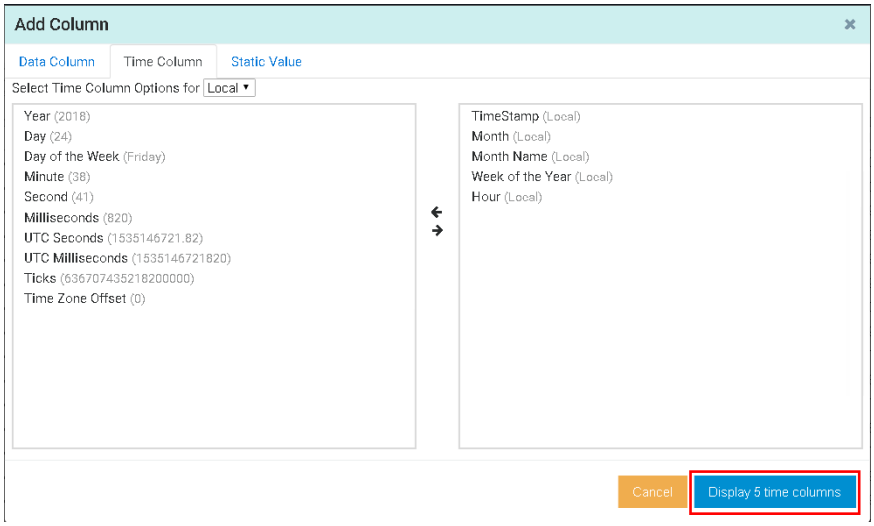


REGION	TimeStamp	STATION	UNIT	Demand
CENTRAL	5/31/2019 5:00:00 PM	Carbondale	TCB03	31.917
CENTRAL	5/31/2019 6:00:00 PM	Carbondale	TCB03	31.843
CENTRAL	5/31/2019 7:00:00 PM	Carbondale	TCB03	31.768
CENTRAL	5/31/2019 8:00:00 PM	Carbondale	TCB03	31.694
CENTRAL	5/31/2019 9:00:00 PM	Carbondale	TCB03	31.620
CENTRAL	5/31/2019 10:00:00 PM	Carbondale	TCB03	31.545
CENTRAL	5/31/2019 11:00:00 PM	Carbondale	TCB03	31.471
CENTRAL	6/1/2019 12:00:00 AM	Carbondale	TCB03	31.396
CENTRAL	6/1/2019 1:00:00 AM	Carbondale	TCB03	31.322

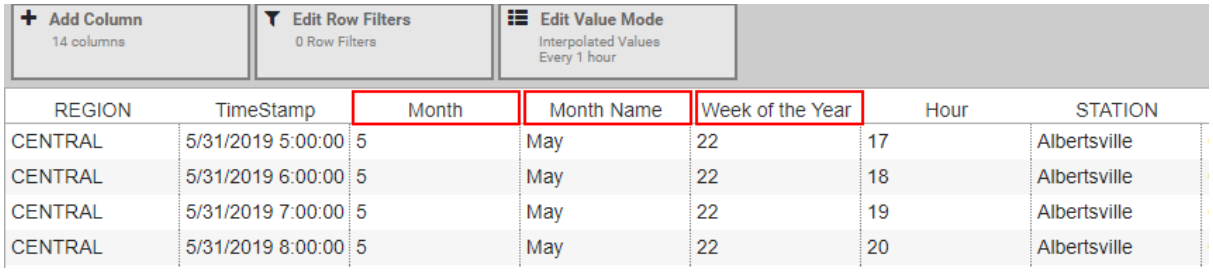
Now we'll add some additional time columns that will come in handy later when building the reports. **Click Add Colum**. Select the **Time Column** tab. Select Month, Month Name, Week of the Year, and Hour, then click the arrow to bump them over to the right:



Click Display 5 Time Columns:

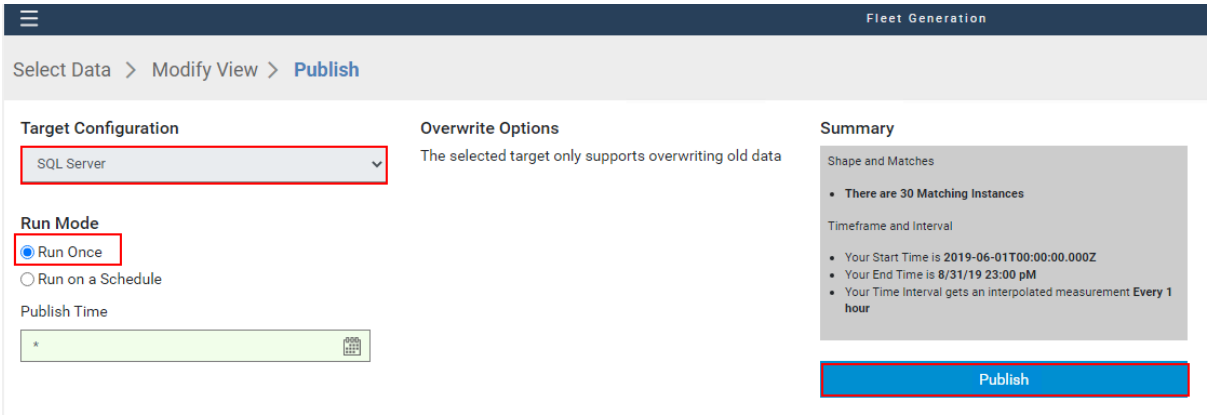


Now that the time ranges and columns have been specified, click **Next**.



REGION	TimeStamp	Month	Month Name	Week of the Year	Hour	STATION
CENTRAL	5/31/2019 5:00:00	5	May	22	17	Albertsville
CENTRAL	5/31/2019 6:00:00	5	May	22	18	Albertsville
CENTRAL	5/31/2019 7:00:00	5	May	22	19	Albertsville
CENTRAL	5/31/2019 8:00:00	5	May	22	20	Albertsville

Now we can choose what target to publish to. This depends on the platform used to support front-end application, but for our purposes we'll publish to a SQL Server. Select **SQL Server** for the Target Configuration, Leave Run Once checked, and click **Publish**:



It will take a few minutes to publish the data.

5 Building the Fleet Generation Reports

We will now spend some time configuring a Microsoft Power BI report. The first step is importing the data.

5.1 Preparing and Importing the Tables

Now that the Fleet Generation table has been published, we will import the SQL table into Power BI.

5.1.1 Directed Activity – Import Data from Microsoft SQL Server.

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

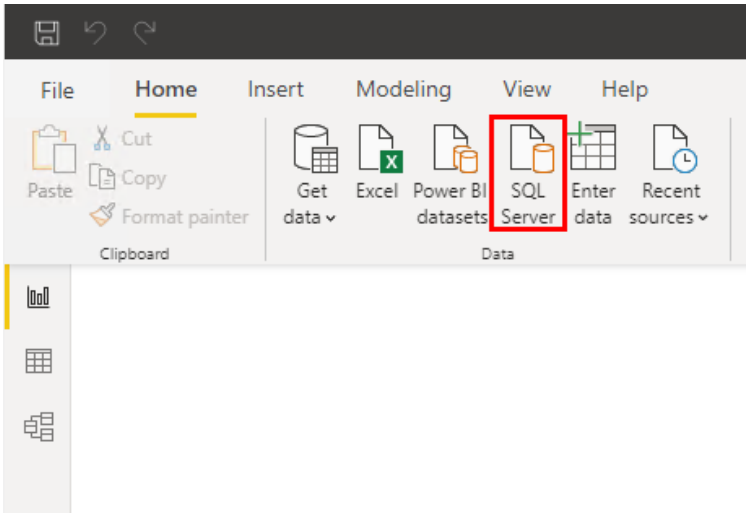
Objective:

Import the Fleet Generation table.

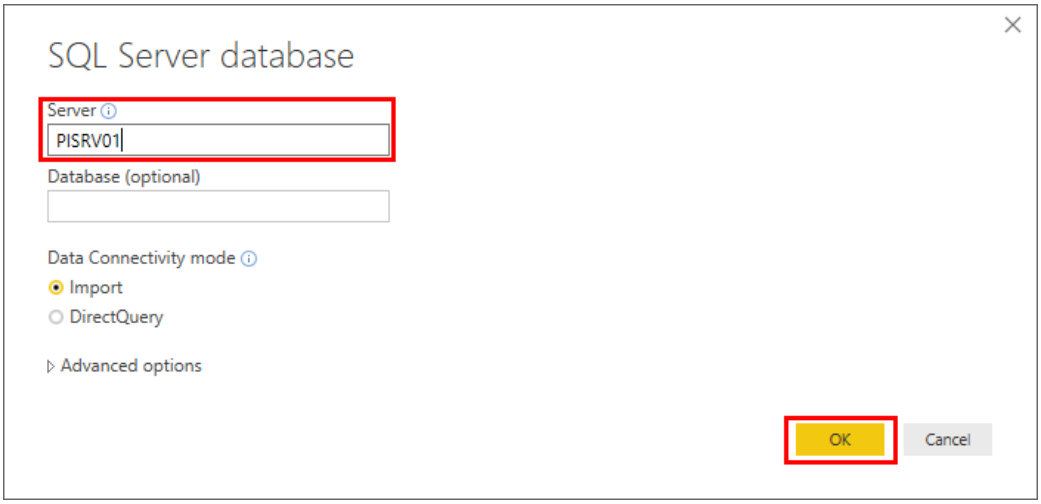
Approach:

Open Microsoft Power BI and start a new report.

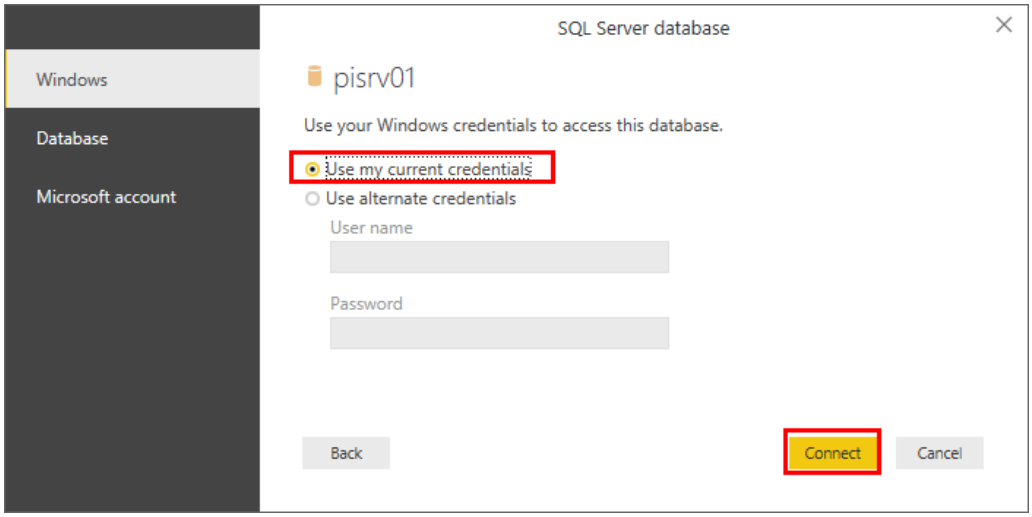
Select **SQL Server** in the Data Group.



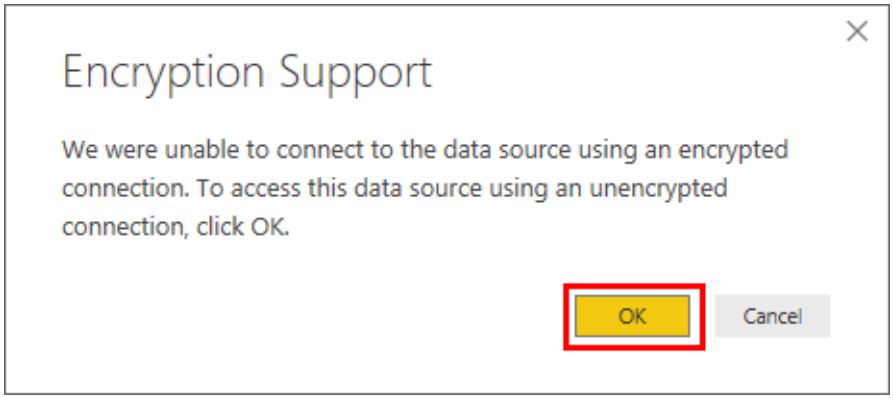
Enter **PISRV01** as the server name and click **OK**.



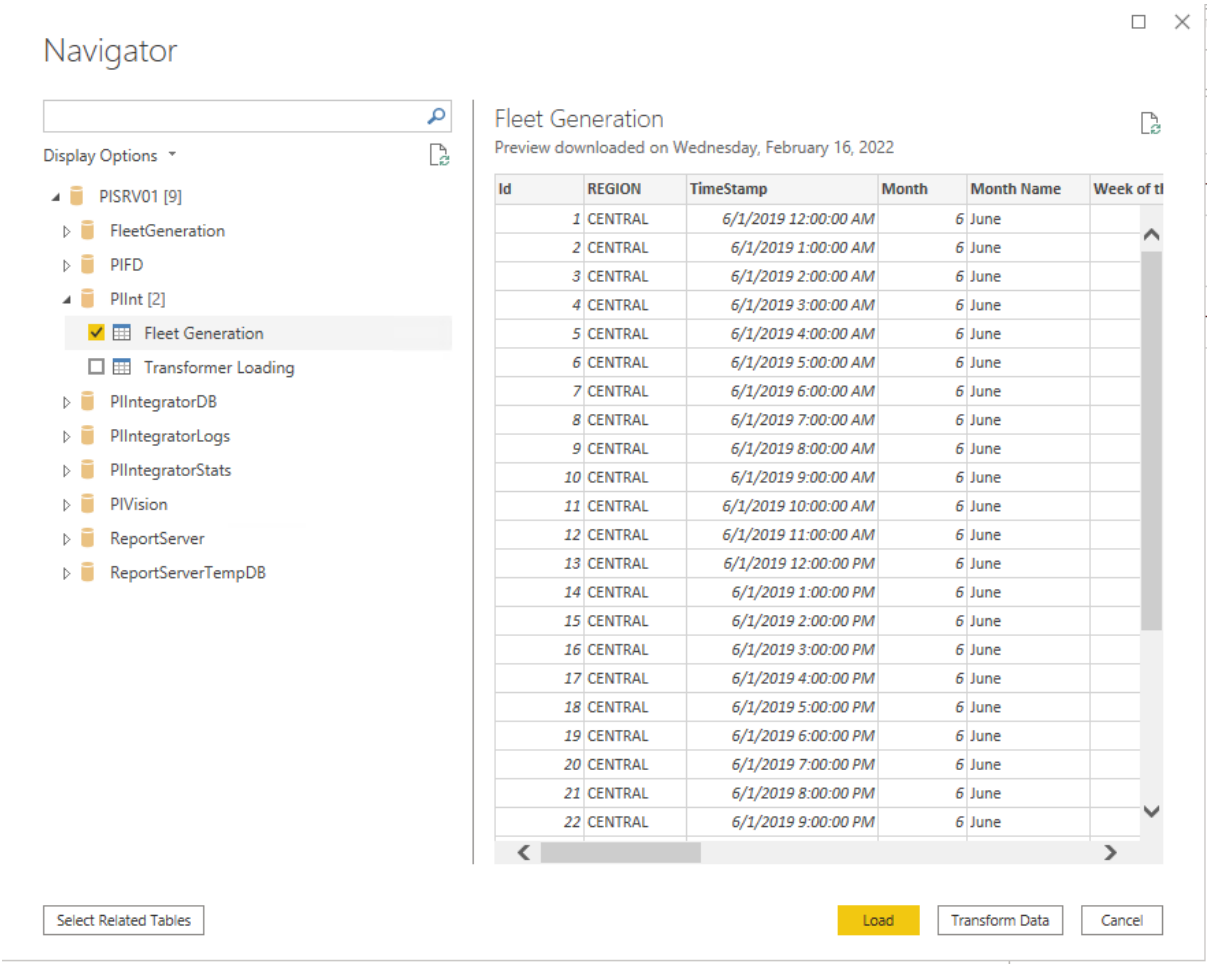
If Prompted, Leave “use my current credentials” selected and click **Connect**:



There may be a warning that the connection is not encrypted, this can be safely ignored, **click OK:**



Expand the PIInt database and Select the **Fleet Generation** table, click **Load**



Note that 66 240 rows have been imported. It can load much more data, while Microsoft Excel has a limitation of **1 million rows.**

5.2 Building the Report Visuals

Now that the Fleet Generation table has been imported, the rest of the chapter will be a walkthrough of configuring various report visuals.

In case there were mistakes or problems with the previous steps, a starter .pbix file has been created with the raw data set already imported with columns that will match the exercises exactly.

Open **C:\Class\Part 1 - PI Integrator for BA\Starter File - Part 1 Fleet Generation.pbix** and use this as a starting point for the remaining exercises.

5.2.1 Directed Activity – Fleet Generation Analysis

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objectives:

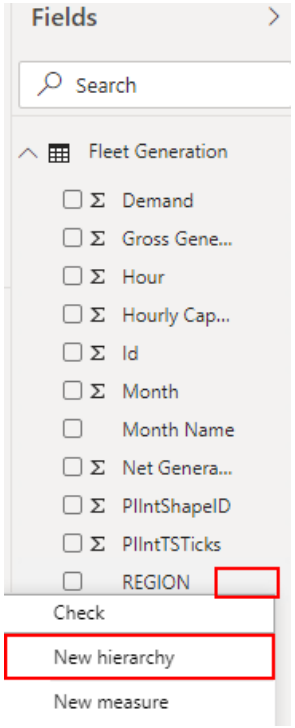
- Configure a **Hierarchy**
- Configure a **Hierarchy Slicer**
- Configure a **Measure** to calculate service hours
- Configure a **Group** to create bins for different load ranges which can then be used for highlighting and filtering
- Configure a **Stacked Bar Chart** to display the service hours spent in each Load Range by unit
- Configure a **Table** to show the top 20 units by average Loading
- Configure a **Slicer** to filter by Month

In this exercise, we will analyze characteristics of different assets, which generate energy. The goal is to assess the number of service hours spent in various high load conditions to better understand which assets need more attention and inspect the reasons of high energy generation.

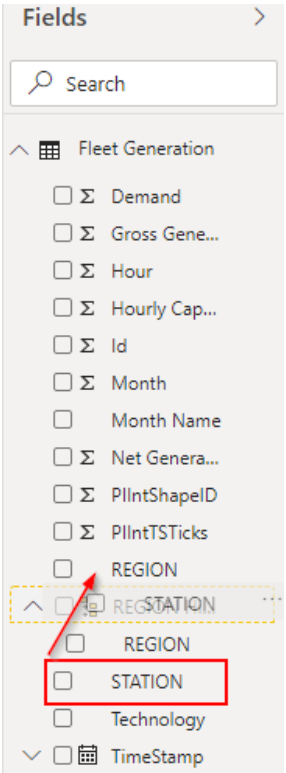
Approach:

Configuring the Hierarchy

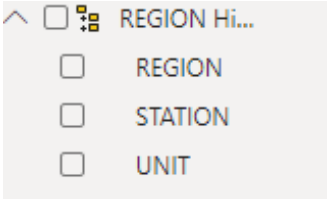
We will now create a hierarchy. In the **Fields List**, click the ellipses next to Region and select **New hierarchy**:



Within the fields list, drag and drop the **Station** field on top of the new Region hierarchy:



Repeat for **Unit**.



Downloading the Hierarchy Slicer

For this part, there is no need to visit the web site, sign up, or download the file. We have downloaded the file for use in class so that students do not need to sign up!

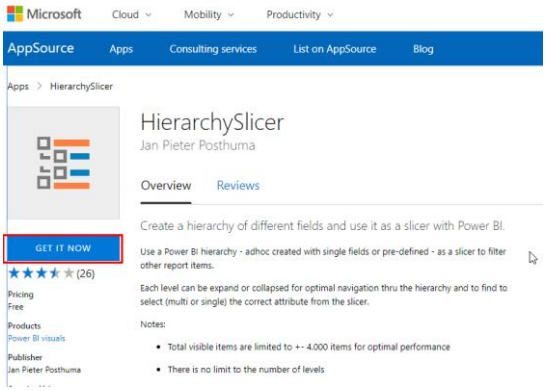
The Hierarchy Slicer is a custom visual that can be used to filter reports and mimic the PI AF hierarchy. This is similar to the PI TreeView from PI WebParts.

Most custom visuals can be found on Microsoft AppSource. We will briefly go through the procedure of how one would normally obtain a custom visual.

Search for a custom visual on Google or within AppSource and you'll arrive at a page like this:

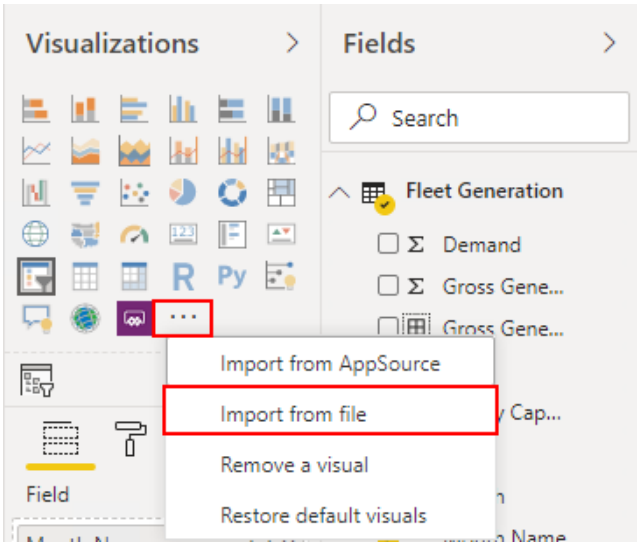
https://appsource.microsoft.com/en-us/product/power-bi-visuals/WA104380820?tab=Overview

At which point you would click Get It Now, sign in using your work or school account, and download the .pbiviz file.

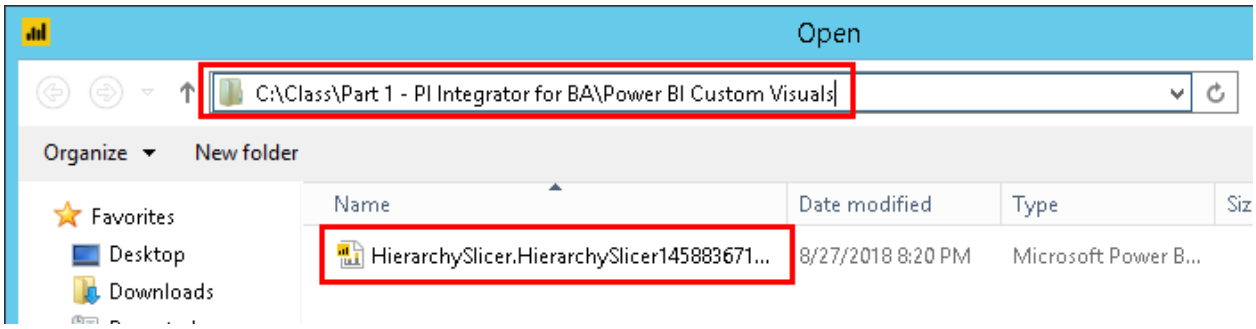


Importing and Configuring the Hierarchy Slicer

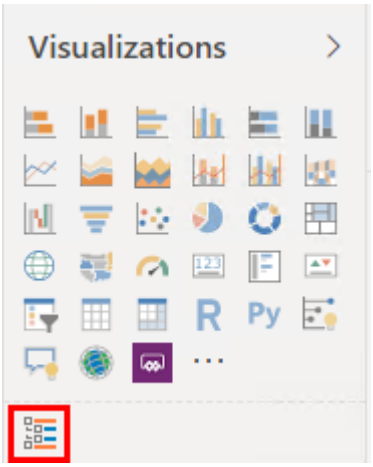
Now it's time to import the custom visual. Open Power BI, click the ellipses within the Visualization Pane, and select Import from file:



Navigate to **C:\Class\Part 1 - PI Integrator for BA\Power BI Custom Visuals** and select the **HierarchySlicer** file.



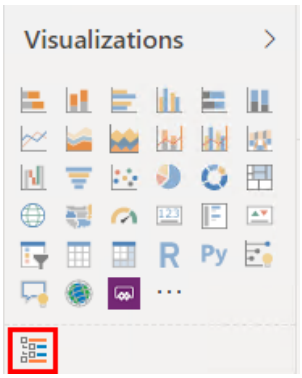
We should now see the Hierarchy Slicer in the list of available visuals:



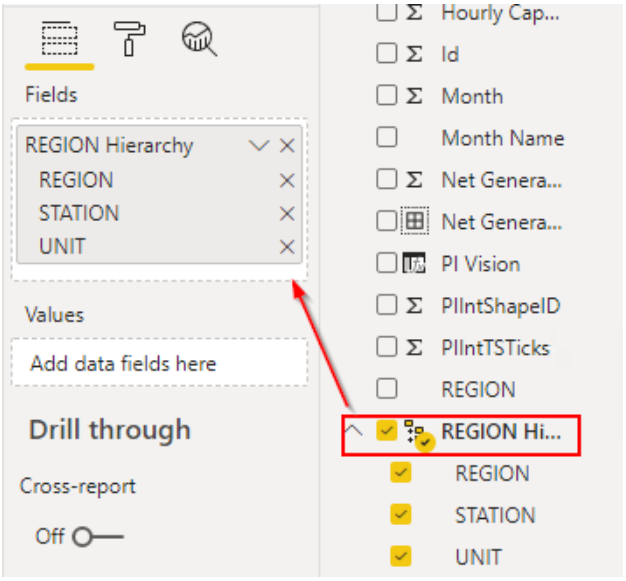
Mimic PI AF Hierarchy – Hierarchy Slicer

This exercise requires the Hierarchy Slicer custom visual be imported and assumes the Hierarchy has been configured.

We will use a Hierarchy Slicer to leverage the existing PI AF hierarchy for filtering. Add a Hierarchy Slicer by clicking the icon:

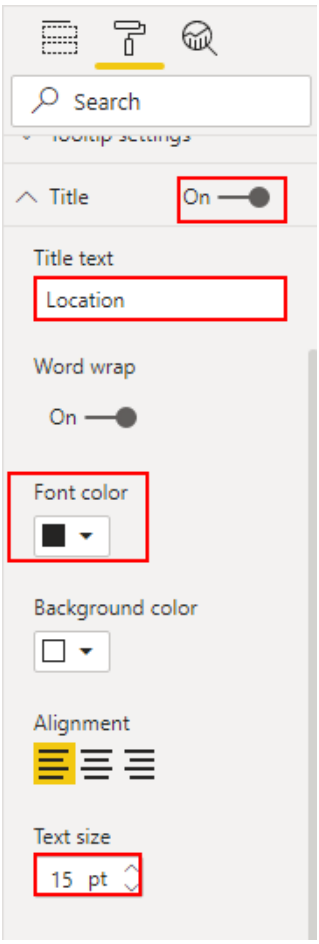


Drag and drop the **Hierarchy** to the visual fields:



Experiment with the Hierarchy Slicer for a bit by drilling down through the levels. Note that checking a box for a parent will also include the children. This is a great way to visualize how filtering works in Power BI.

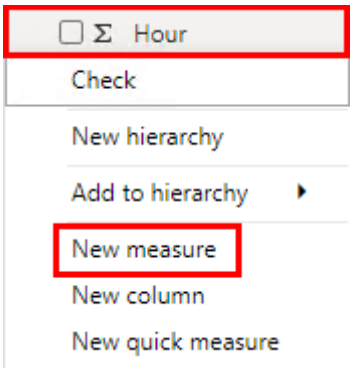
Change the Title of the Hierarchy Slicer to Location in the formatting options. Change the color and increase the text size.



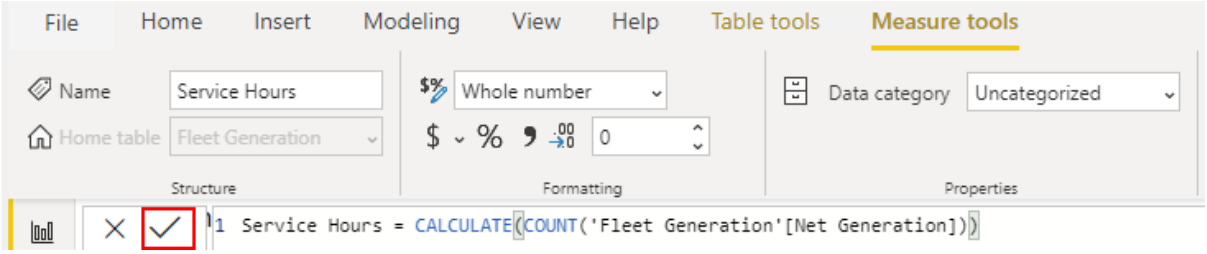
Service Hours

Now we'll configure a Measure to calculate service hours. Each row in the data set represents 1 hour, so we can simply count the number of rows that have been filtered through user selection. This should make a bit more sense when it all comes together.

Right click **ANY** of the fields from the Fields list and select **New measure**:



Enter the below formula into the configuration box and hit **Enter or click the Checkmark**:



The raw text is given below for convenience.

```
Service Hours = CALCULATE(COUNT('Fleet Generation'[Net Generation]))
```

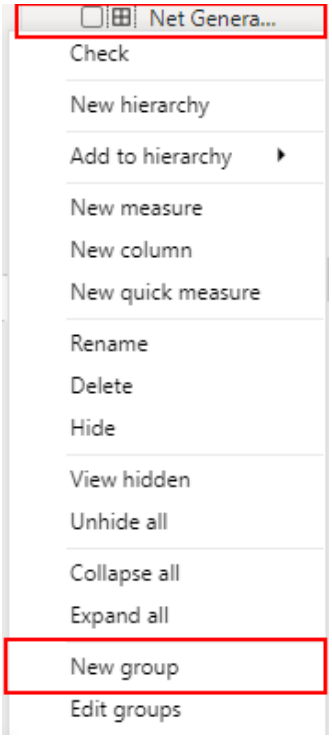
From a configuration perspective, Measures and Calculated Columns are configured similarly so the distinction may not be obvious.

Measures and calculated columns both use DAX expressions. The difference is the context of evaluation. A measure is evaluated on the fly using a subset of data, whereas a calculated column is pre-calculated at the row level within the table to which it belongs. A simple way to put it is that Measures take into account the filtering that has been set by the end user of the report (the stuff they've clicked on), while calculated columns are computed row by row and are not influenced by the report filtering.

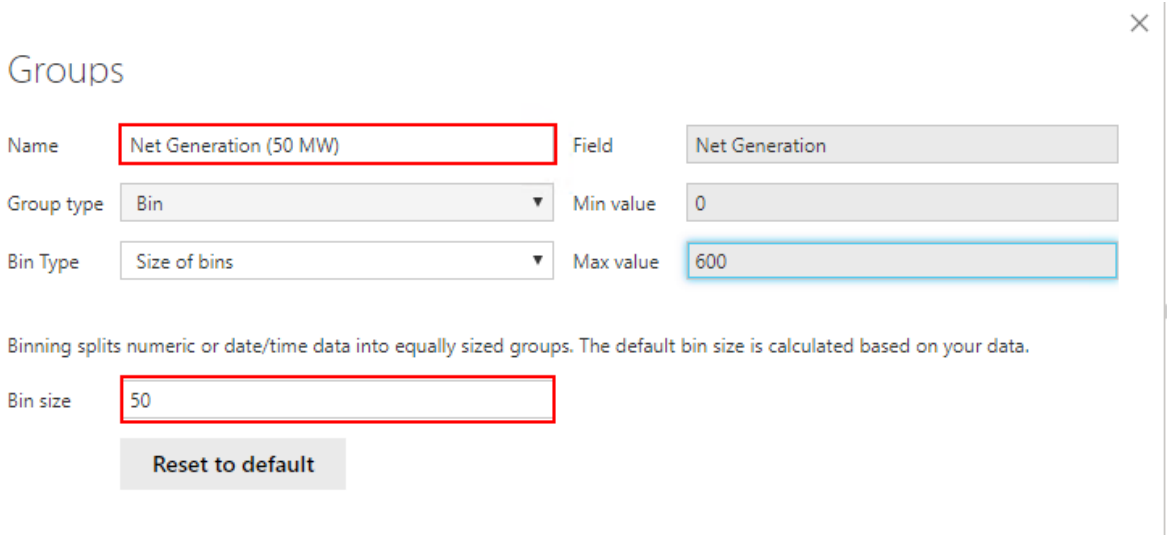
Net Generation Groups

Different ranges for Net Generation will be grouped into bins representing different Load Ranges. In order to calculate service hours in the different Load Ranges, a group must be configured in the data set for filtering and counting by the Service Hours Measure.

Right click on **Net Generation** and select **New group**.

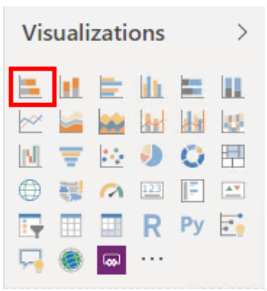


Change the name to **Net Generation (50 MW)** and set the bin size to 50, then click **OK**.

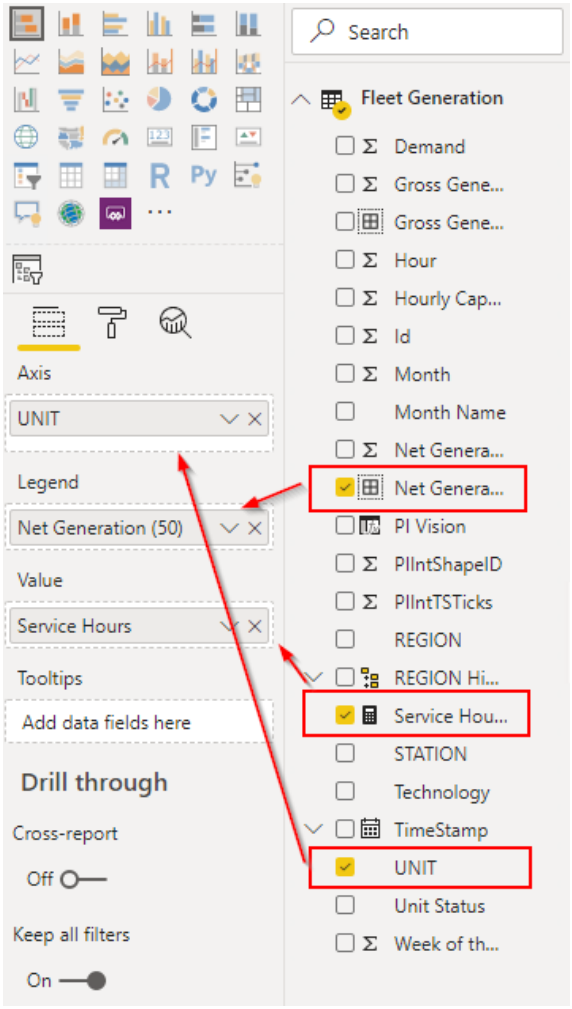


Net Generation by Asset – Stacked Bar Chart

Now we can begin to configure the report. **Click some empty space** and then **click the Stacked Bar Chart icon**:

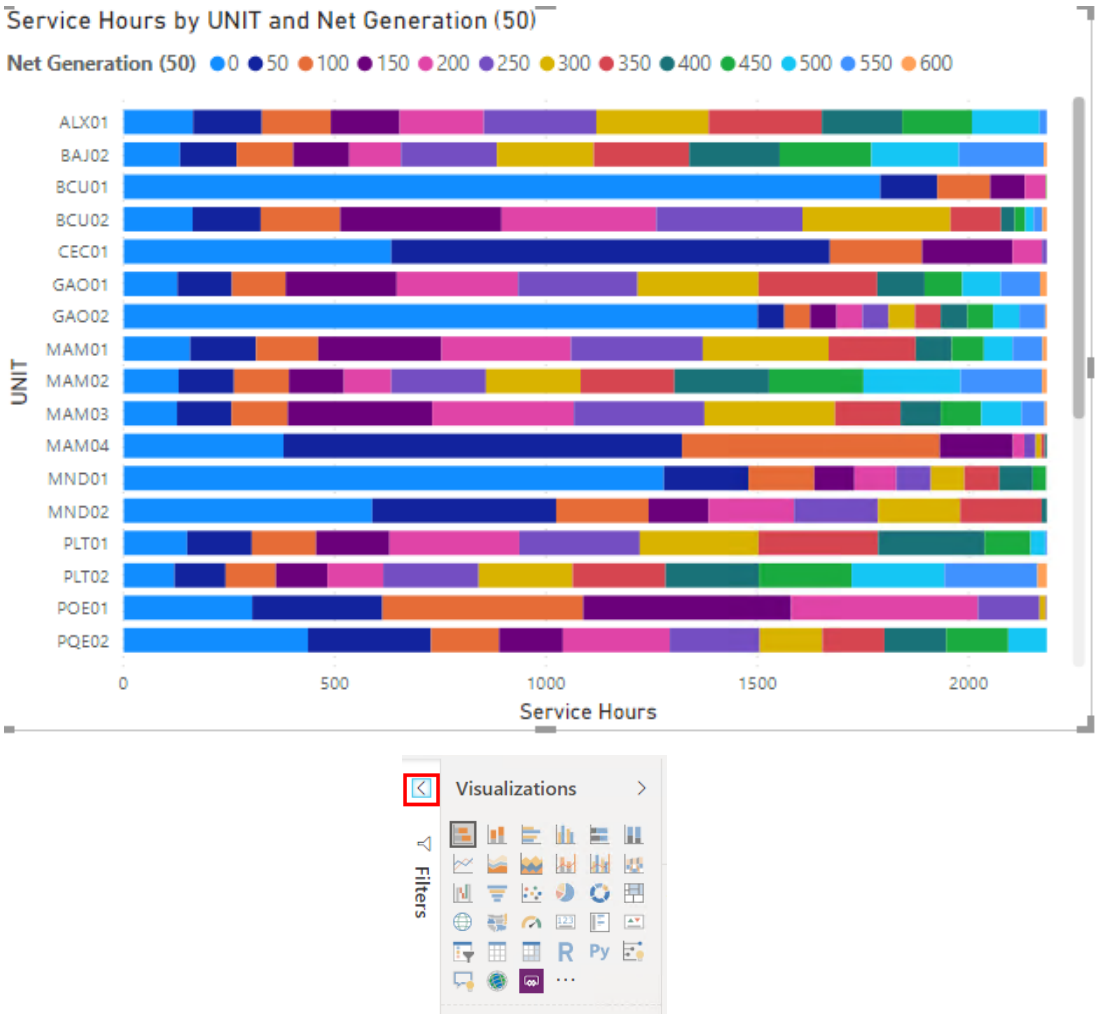


With the Stacked Bar Chart selected, drag and drop Fields from the data set into the field configuration boxes. Use **Unit** for the Axis, **Net Generation (50%)** for the Legend, and **Service Hours** for the Value:

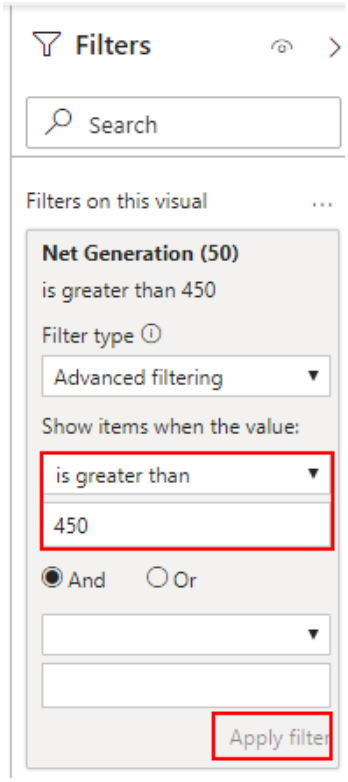


Next we will apply some formatting and filters to make the data set more manageable. We'll change the color scheme and only show Net Generation greater than 450MW, since Net Generation in the normal range is not of interest to us.

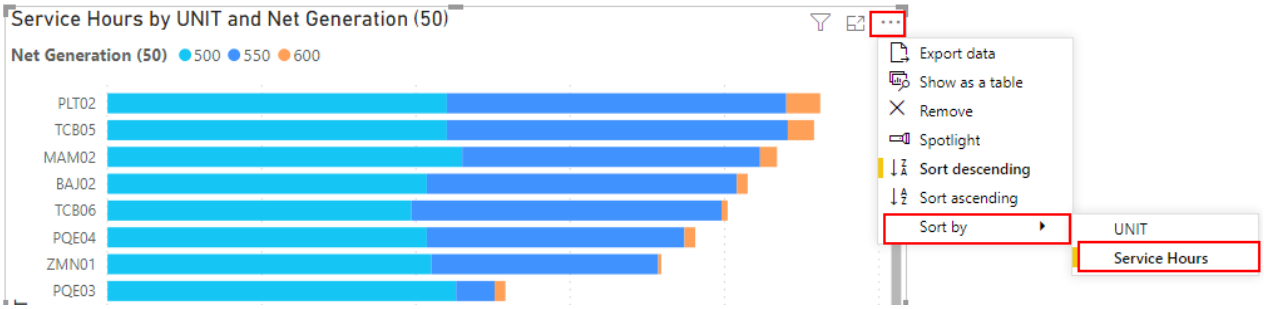
Expand the Filters Pane:



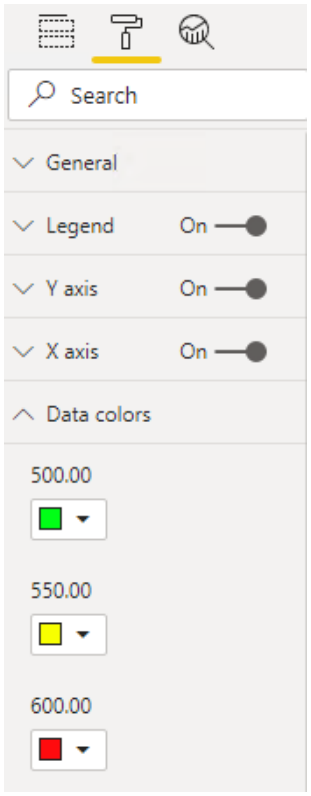
Filter for Net Generation **greater than** 450MW. Be sure to click **Apply Filter**:



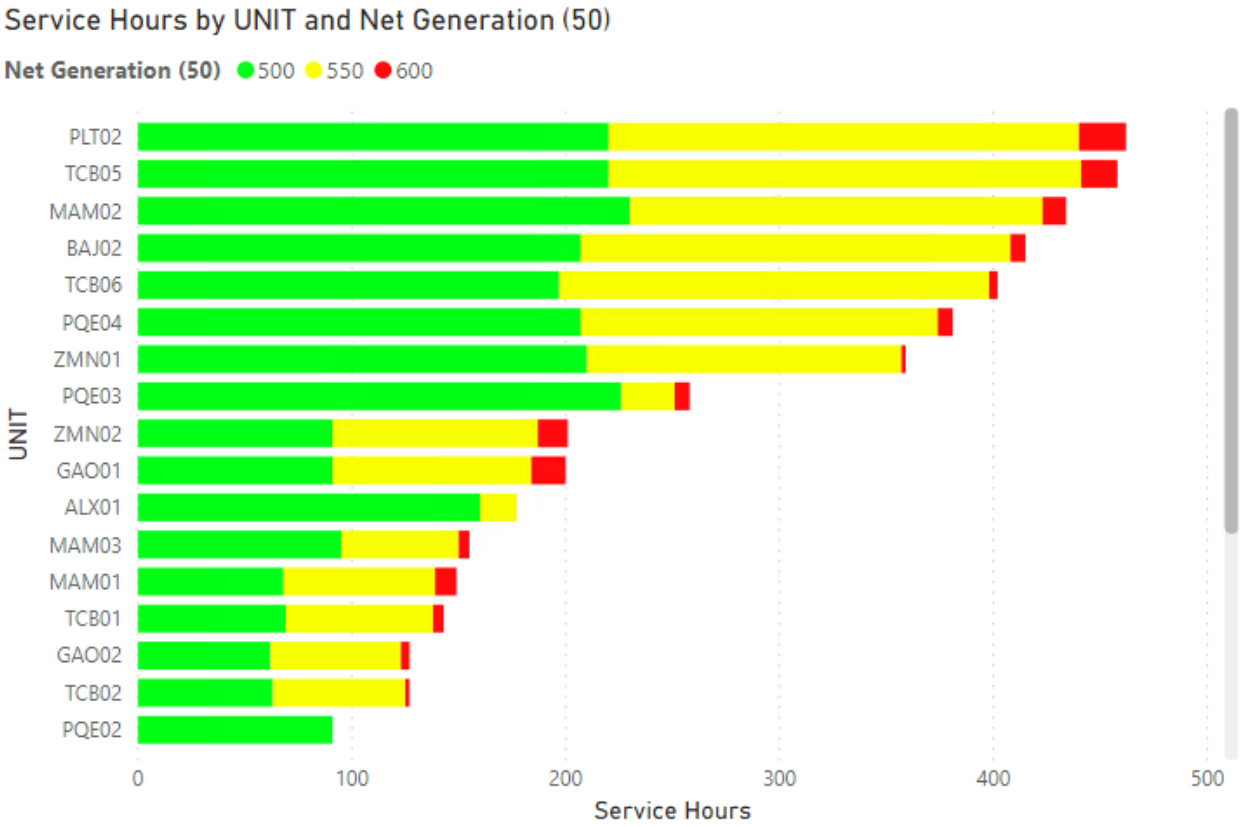
Next go to the Visualization Options and **sort by Service Hours** (done by default in this version of Power BI):



Next change the color scheme. With the Visualization selected, click the Format Icon in the Visualization Pane and adjust the colors to better convey the severity of the Net Generation levels.



The stacked bar chart should now look something like this:

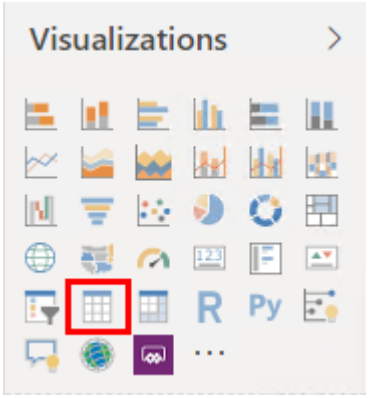


Service Hours and Average Net Generation by Unit – Table

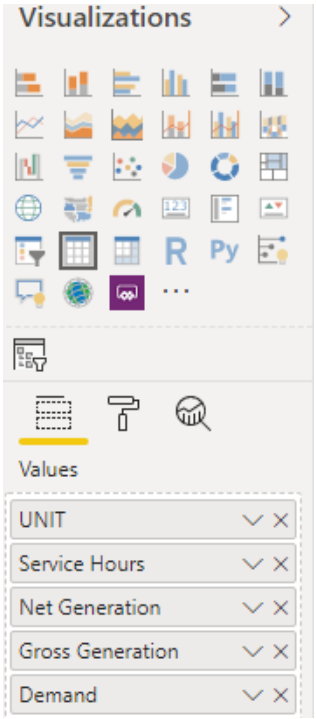
The next visual we will add is a basic table showing the Unit Name, Service Hours, Average of Net Generation, Average of Gross Generation and Average of Demand. We will then filter the table to show only the top 10 transformers by average load. This will give us a quick indicator of which Units are consistently overloaded.

Click some blank space on the canvas to deselect any visuals, otherwise you will accidentally convert the Stacked Bar Chart to a Table.

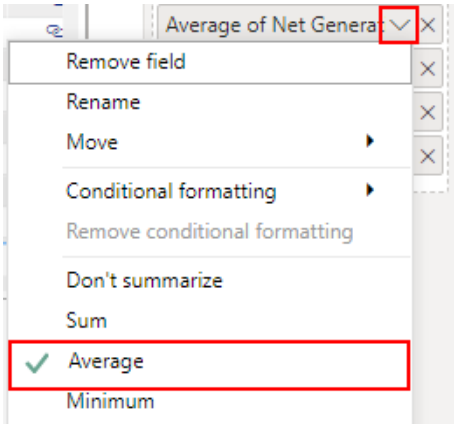
Create a **Table**:



Drag and drop the **Unit, Service Hours**, **Net Generation**, **Gross Generation** and **Demand** Fields into the Values section:

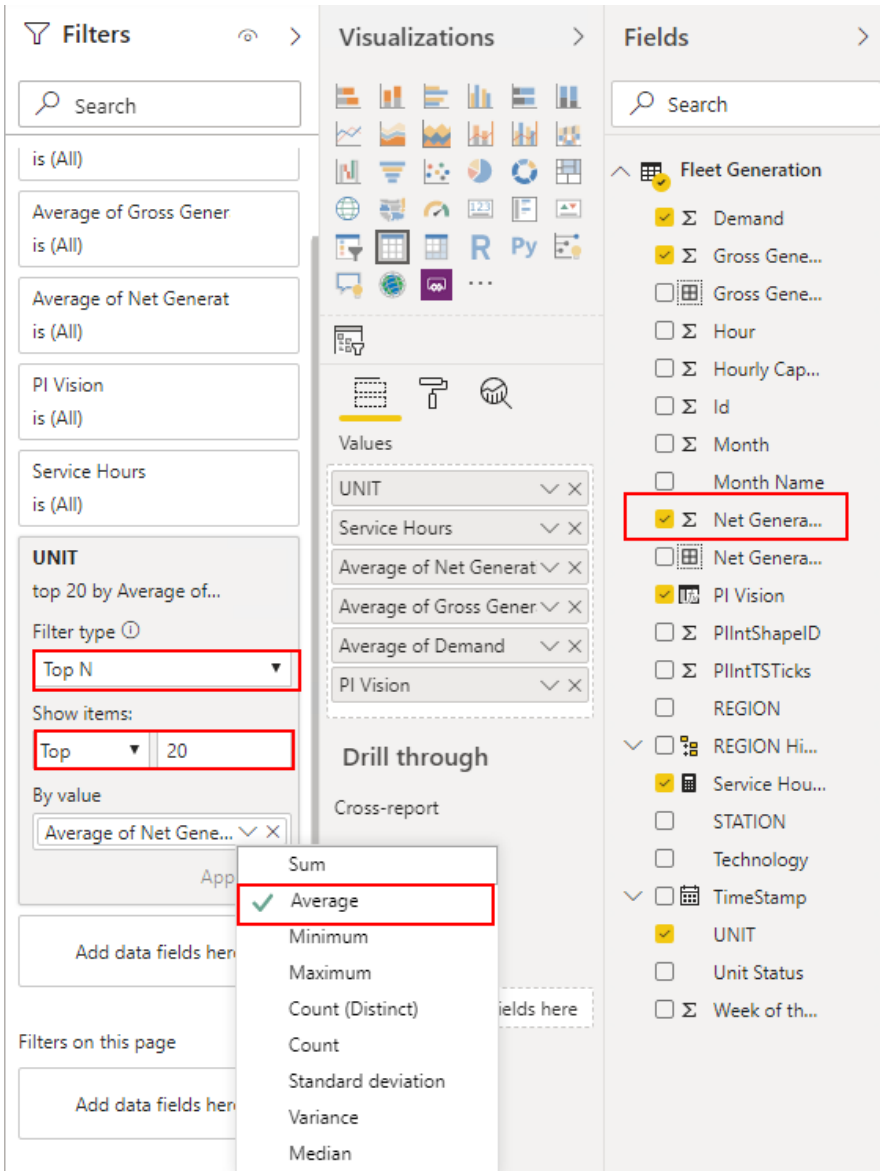


Change the **Net Generation** Value to summarize by **Average**:



Repeat the same for **Gross Generation** and **Demand**.

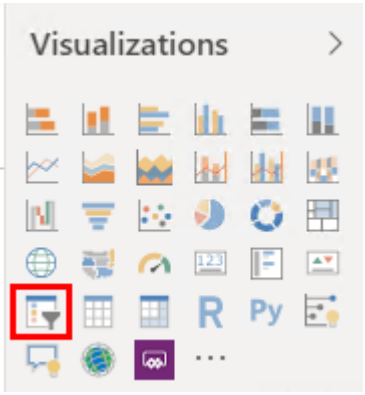
Change the Visual Level Filters to Show the **Top 20** Units by **Net Generation**.



Change to summarize **Net Generation as Average**, then be sure to click **Apply filter**.

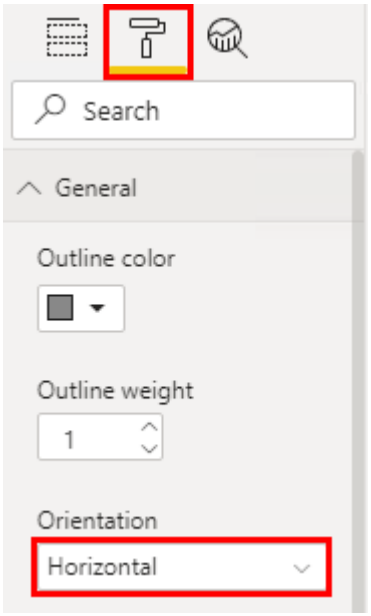
Filtering by Month – Slicer

We'll now add a basic Slicer to filter by Month. **Click some blank space** and then **add a Slicer**:

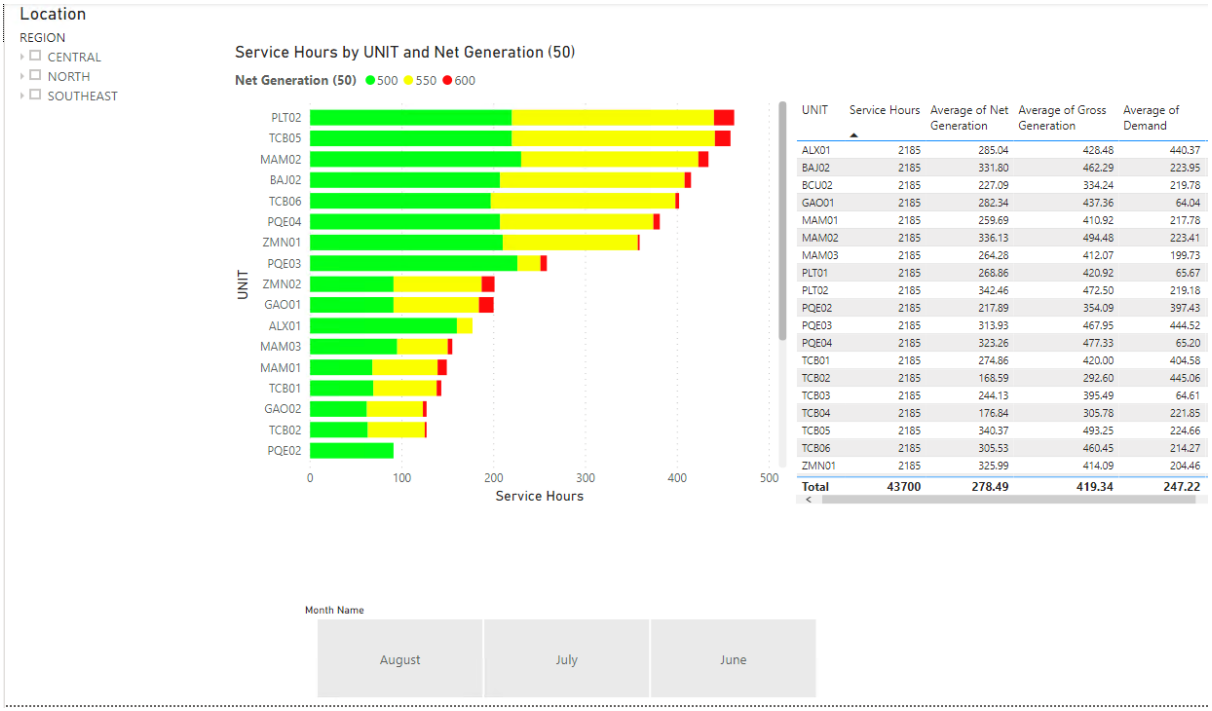


Drag **Month Name** to the field list.

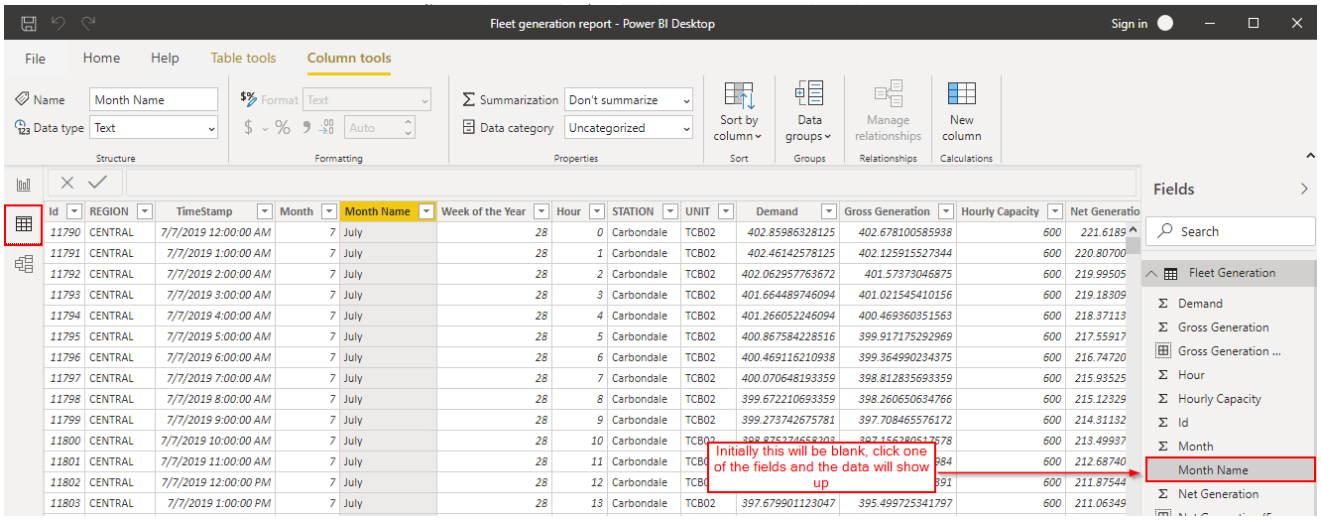
Go into the formatting options and change the orientation to **horizontal** to change the look of the Slicer.



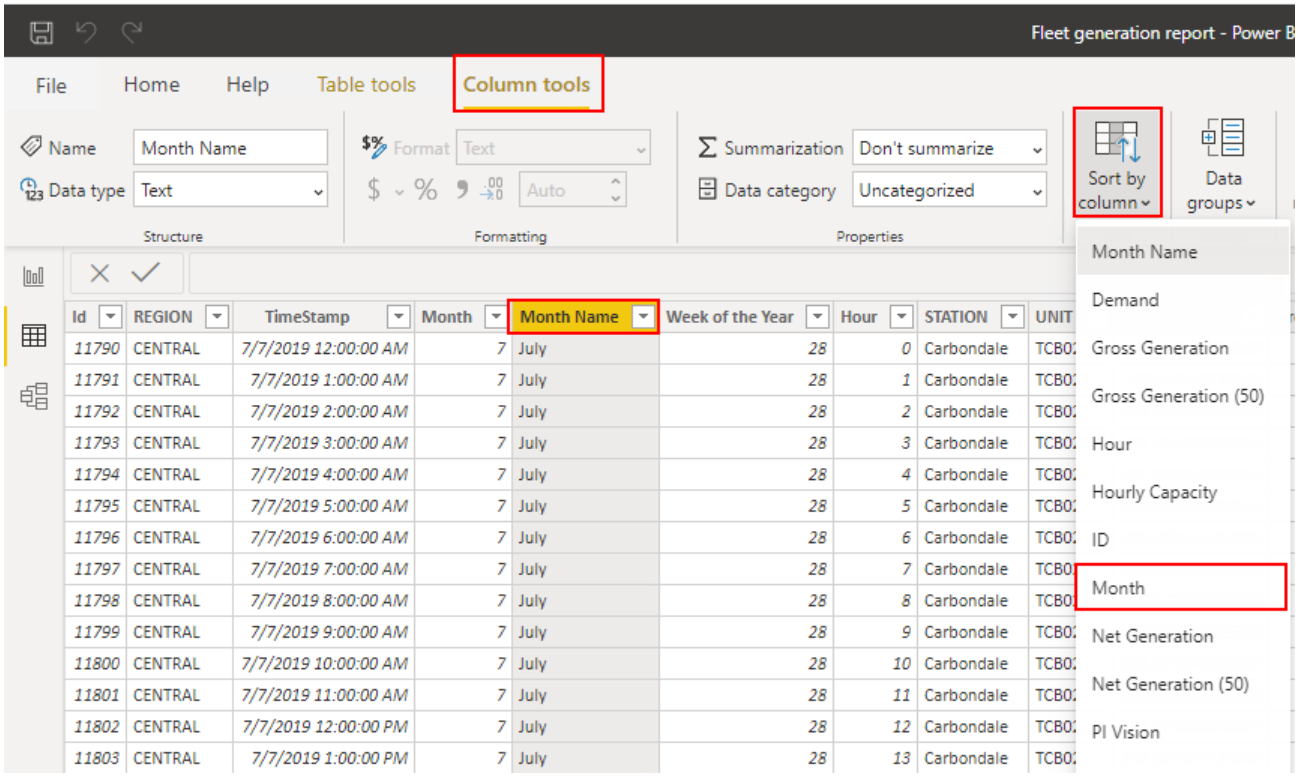
Reposition & Resize the slicer so all months are in a single row. **Reposition & Resize the table** and stacked bar chart:



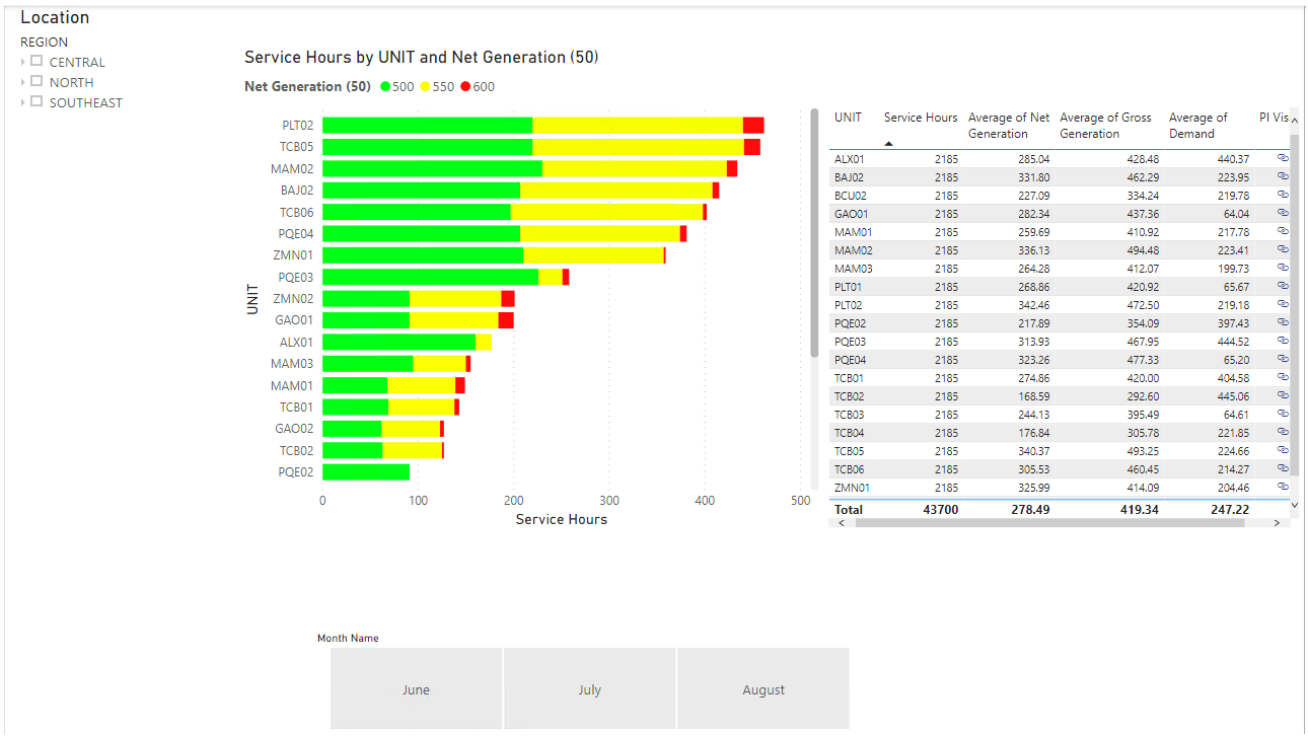
To put the Months in chronological order, we will sort the Month Name column in the data set by the Month column where the months are numbered. Go to the **Data View** and click one of the fields to make the data show up:



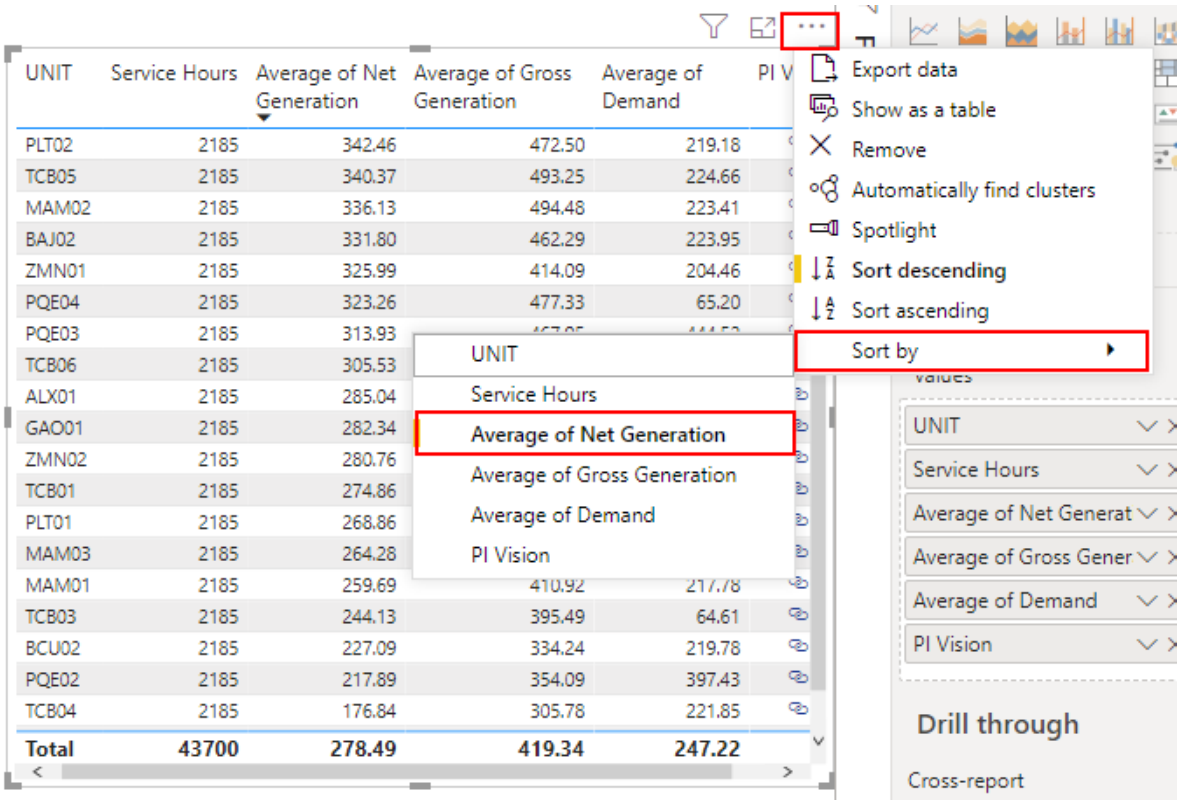
Select the **Month Name** column, open the **Column Tools** Ribbon, and **Sort by Column -> Month**:



The report should now look something like this:



Sort the table by Average of Net Generation:



Click the bars on the Net Generation by Unit chart and the Month slicer buttons and note how the service hours and units for that load range update on the table.

We will save formatting until the end in case we need to save time, but feel free to adjust the formatting and add a title.

Linking to PI Vision

We have a PI Vision display for Units that we can link to from this report. We will utilize PI Vision URL Parameters to set the same Unit in the PI Vision display that the user clicks on in the Power BI report. The URL parameters reference guide can be found in the docs.osisoft.com.

From within the client virtual machine, Navigate to:
https://pisrv01.pischool.int/PIVision/#/Displays/5/FleetGeneration

Take the above URL and append the following string to it in a text editor, then paste the URL into Chrome:

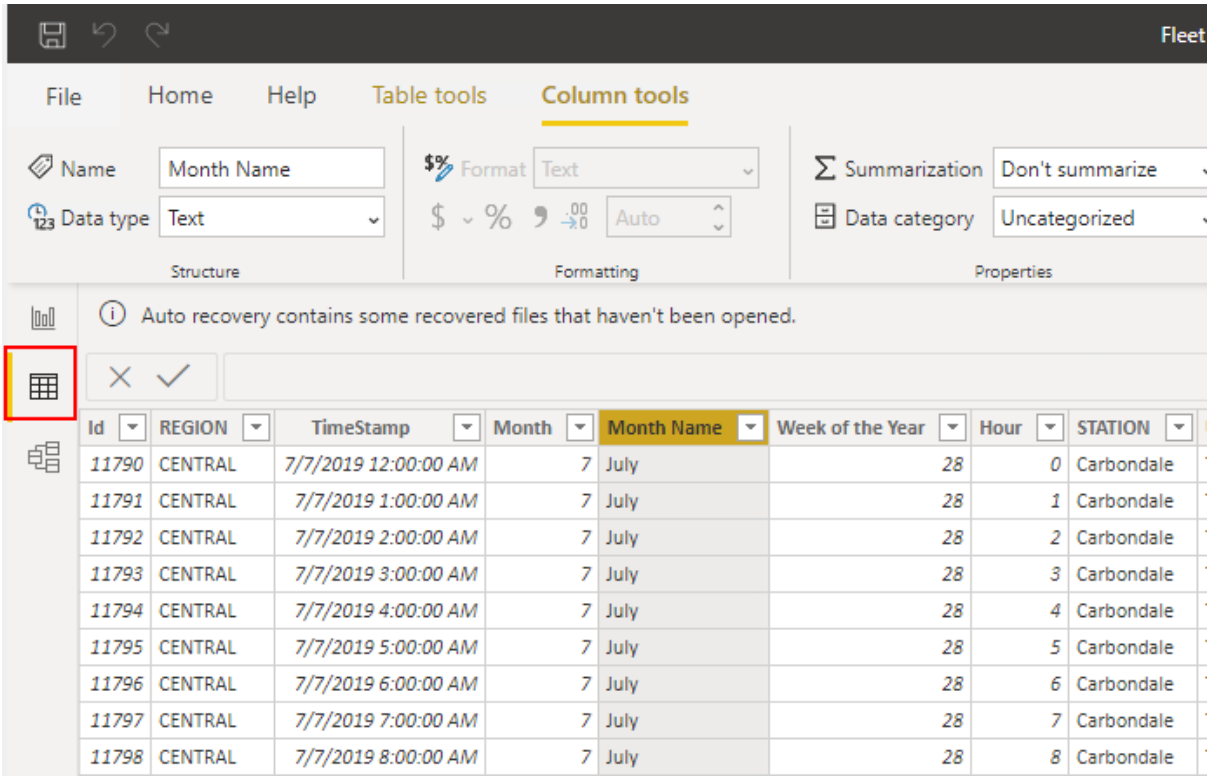
?Asset=\\PISRV01\Fleet Generation\All Turbines\GAO01

Unit GAO01 should be the selected Asset in the FleetGeneration display.

Note that the **?Asset** parameter denotes the path to the Asset in the PI AF hierarchy.

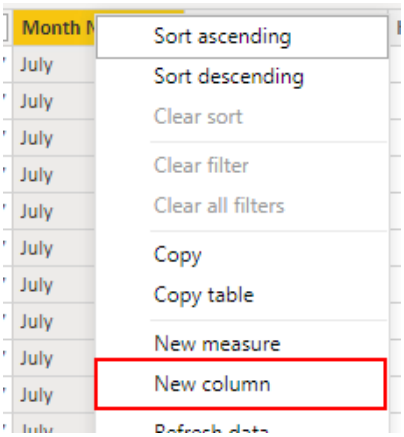
Once that is working, configure a Calculated Column to concatenate the URL with the Unit asset path.

Go to the **Data** Tab:



Id	REGION	TimeStamp	Month	Month Name	Week of the Year	Hour	STATION
11790	CENTRAL	7/7/2019 12:00:00 AM	7	July	28	0	Carbondale
11791	CENTRAL	7/7/2019 1:00:00 AM	7	July	28	1	Carbondale
11792	CENTRAL	7/7/2019 2:00:00 AM	7	July	28	2	Carbondale
11793	CENTRAL	7/7/2019 3:00:00 AM	7	July	28	3	Carbondale
11794	CENTRAL	7/7/2019 4:00:00 AM	7	July	28	4	Carbondale
11795	CENTRAL	7/7/2019 5:00:00 AM	7	July	28	5	Carbondale
11796	CENTRAL	7/7/2019 6:00:00 AM	7	July	28	6	Carbondale
11797	CENTRAL	7/7/2019 7:00:00 AM	7	July	28	7	Carbondale
11798	CENTRAL	7/7/2019 8:00:00 AM	7	July	28	8	Carbondale

Right click on the header of **ANY** column and select **New column**:



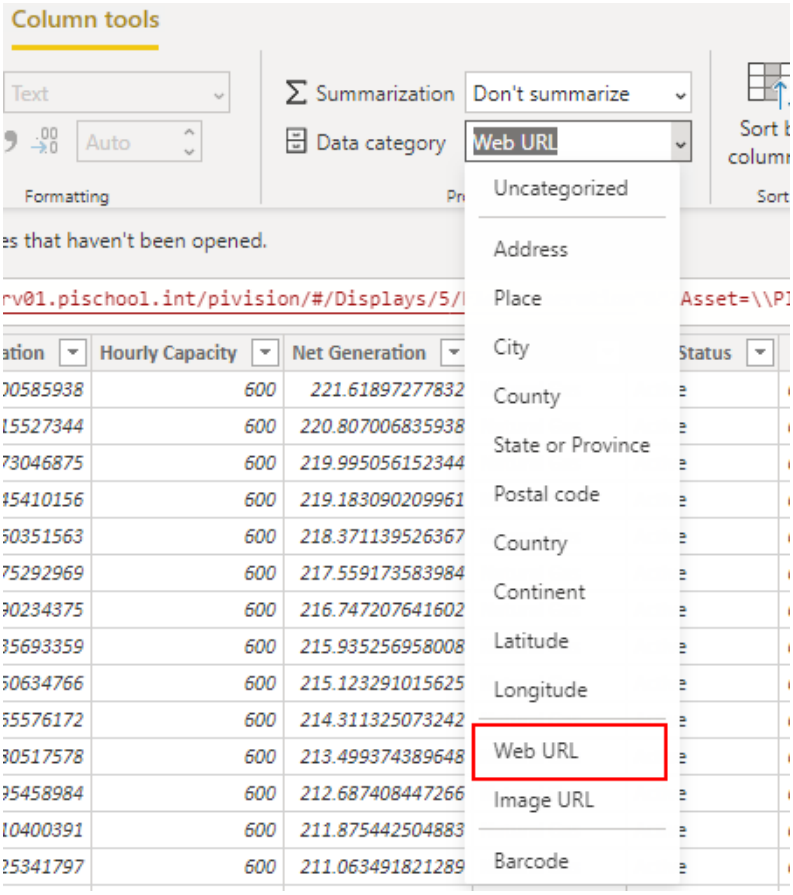
For the DAX formula, enter the following and **hit enter or click the checkmark**:

```
PI Vision =
"https://pisrv01.pischool.int/pivision/#/Displays/5/FleetGeneration"&"?Asset=\\PISRV01\Fleet Generation\All Turbines\" & 'Fleet Generation'[UNIT]
```

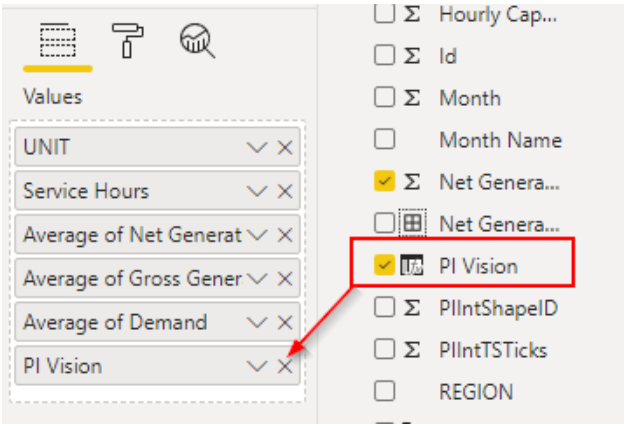
```
PI Vision = "https://pisrv01.pischool.int/pivision/#/Displays/5/FleetGeneration"&"?Asset=\\PISRV01\Fleet Generation\All Turbines\" & 'Fleet Generation'[UNIT]
```

Next scroll all the way to the right and find the **PI Vision column**, then select it.

Go to the **Column Tools** ribbon, and change the **Data Category** to Web URL.

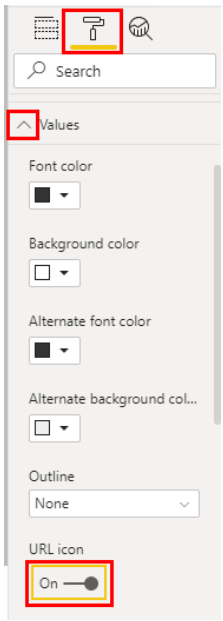


Now go back to the **Report Tab** and select the Table, then drag and drop the **PI Vision** field as one of the table values



The links are now displayed, and they work, but they are not pretty to look at. Luckily, Power BI has a feature that addresses this.

Go into the **Formatting Options,** scroll down to the Values section, and turn on the URL icon:



Now the links look much cleaner:

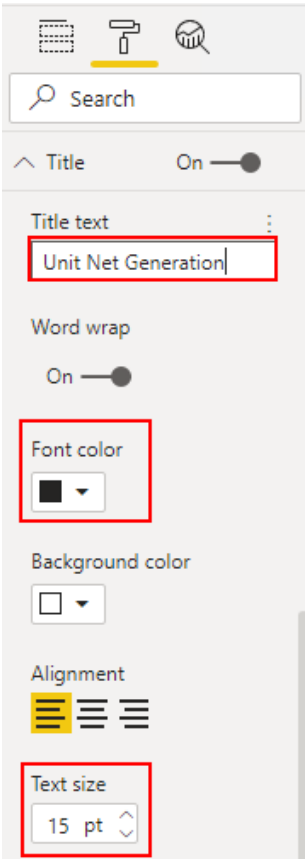
UNIT	Service Hours	Average of Net Generation	Average of Gross Generation	Average of Demand	PI Vision
ALX01	2185	285.04	428.48	440.37	
BAJ02	2185	331.80	462.29	223.95	
BCU02	2185	227.09	334.24	219.78	
GAO01	2185	282.34	437.36	64.04	
MAM01	2185	259.69	410.92	217.78	
MAM02	2185	336.13	494.48	223.41	
MAM03	2185	264.28	412.07	199.73	
PLT01	2185	268.86	420.92	65.67	
PLT02	2185	342.46	472.50	219.18	
PQE02	2185	217.89	354.09	397.43	
PQE03	2185	313.93	467.95	444.52	
PQE04	2185	323.26	477.33	65.20	
TCB01	2185	274.86	420.00	404.58	
TCB02	2185	168.59	292.60	445.06	
TCB03	2185	244.13	395.49	64.61	
TCB04	2185	176.84	305.78	221.85	
TCB05	2185	340.37	493.25	224.66	
TCB06	2185	305.53	460.45	214.27	
ZMN01	2185	325.99	414.09	204.46	
Total	**43700**	**278.49**	**419.34**	**247.22**	

Test the links to confirm that the PI Vision display is launched and the correct unit is set.

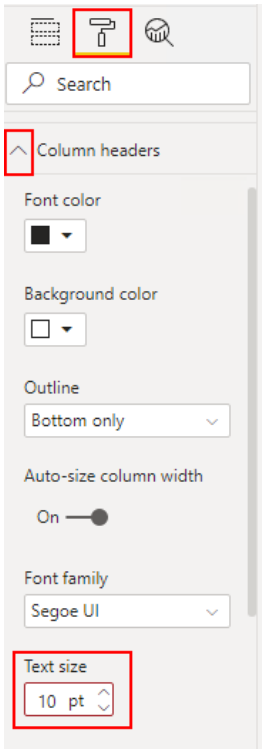
(Optional) Formatting

Take some time to apply formatting to make the report more visually appealing and easier to read.

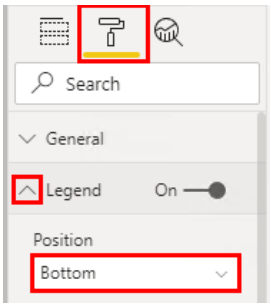
1. Add a Title text box for the report (Insert Ribbon -> Insert Text Box)
2. Add titles for the Stacked Bar Chart and Table, change the font color to black and bump up the font size



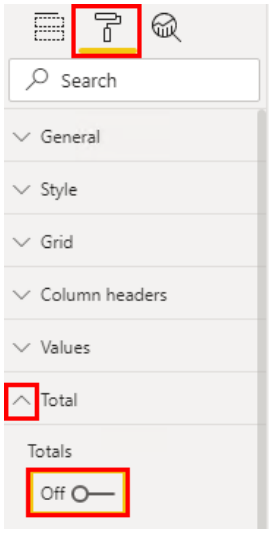
3. Adjust the sizes of the header text on the Table



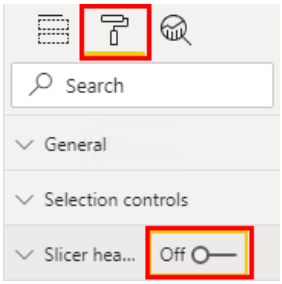
4. Resize the columns
5. Move the Legend on the Stacked Bar Chart to the bottom



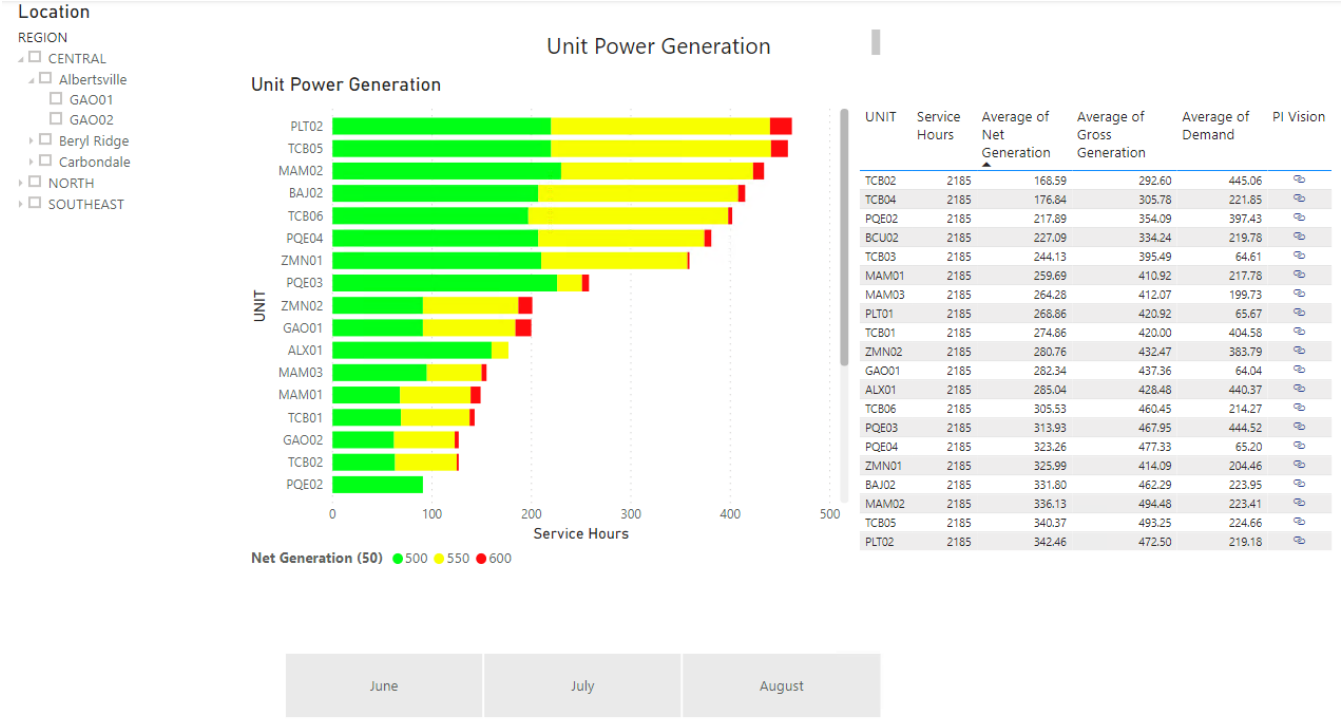
6. Remove the totals from the Table



7. Remove the header from the Slicer



The result should look something like this:



Finally test the links and experiment with filtering the report. We will get back to Fleet Generation database in one of the next chapters and add more analytics and event frames in it, which allows us to find more valuable information like units' utilization, generating efficiency and even more.

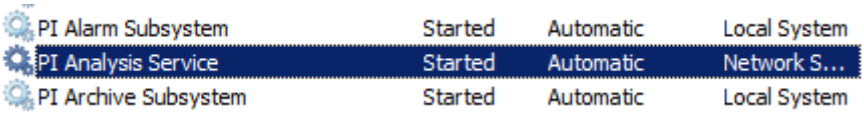
PI Integrator for BA (asset view) saved a lot of time and helped to display PI Data in a 3rd party application like Microsoft Power BI. Let's explore other functionalities of PI Integrator for Power BA, such as Event View and Streaming View. Next chapter explains it very good and is based on the real project that was done in San Leandro. Let's change a bit our production area and imagine ourselves as the data scientists for a moment. All that approaches might be applied in any industry and help to solve a lot of real cases.

6 PI Analysis Service

PI Asset Framework is a powerful tool to help model the infrastructure of a company, region, or division. Through PI Asset Framework Formula Data References, you can create simple, on-the-fly calculations. PI Asset Framework also comes packaged with the PI Analysis Service, for more advanced analyses. The analytic capabilities include three analyses types, Expressions, Rollups, and Event Frame Generation, which allow for calculations to be applied at the template level as well as the ability to persist the results back to the PI Data Archive.

6.1 Capabilities of the PI Analysis Service

The PI Analysis Service, runs as a service that monitors all analyses and attributes associated with these analyses.



Expressions:

Expressions allow for multi-lined calculations that utilize mathematical operators and functions, if-conditions, and PI time-based functions to perform advanced analyses. Expressions, created for a given asset type (element template), are automatically applied to all elements of that type.

Rollups:

Rollups allow for the calculation of summary statistics (averages, maximums, minimums) of values from a set of AF attributes. Current statistical values can be written directly to the PI Data Archive.

Event Frame Generation:

PI Analysis Service allows for the automatic detection of events that occur. These events are bookmarked and information for any event type can be retrieved for further analysis.

Scheduling:

Expressions and Rollups can be scheduled to run whenever a new event arrives into the PI Data Archive or calculated on a periodic basis.

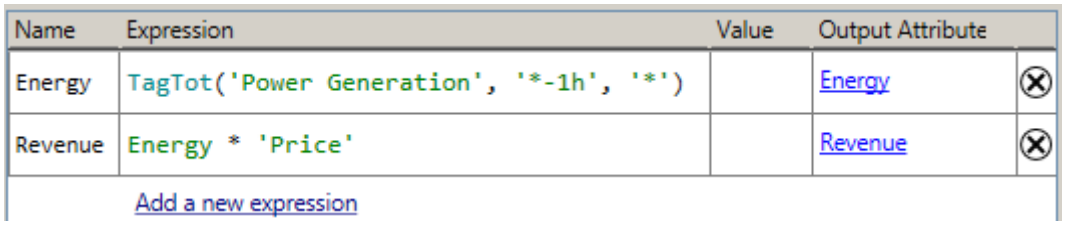
Backfilling:

Results from all three types of analyses can be backfilled into the PI System.

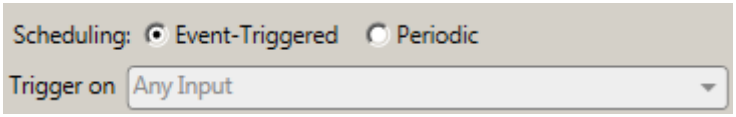
6.2 Expressions

With Expressions, you can implement calculations through a set of built-in functions that take values of attributes in PI Asset Framework as inputs, and outputs results to other PI AF

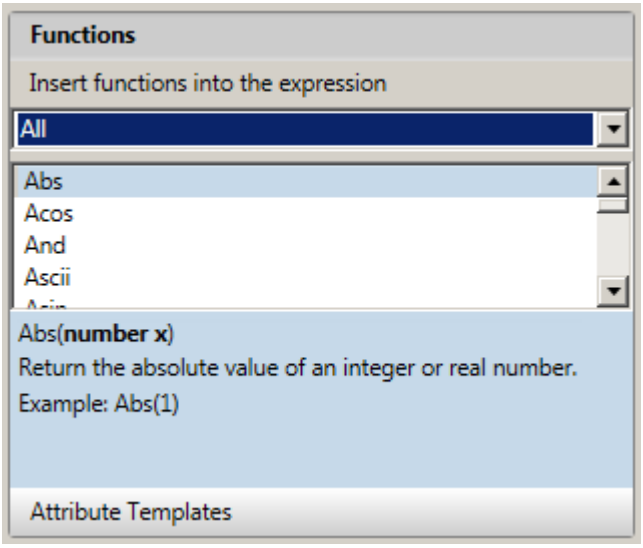
attributes. Expressions can be scheduled to run periodically or scheduled to run whenever the input parameters of the expressions receive a new value.



Multi-line calculation dependency allows for each expression to be written to different output attributes as well as re-using calculated results in subsequent expressions.



Each set of expressions allows for periodic or event-triggered scheduling.



Function Category	Example
Archive Value Statistics	TagAvg, PctGood
Date and Time	Bod, Hour
Logical	And, If
Math	Abs, Sqr
Operators	>, <>, *
PI Data Archive Digital States	DigState, DigText
Point Attributes	TagSpan, TagType
Search and Retrieval	TimeEq, NextEvent
Statistical	Rand, Total
Status	NoOutput, TagBad
String	Len, Text

A set of built-in performance equation-like syntax allows for access to a range of functions. The available options include mathematical and logical operators and functions, date and time functions, PI-specific performance equation functions, and string manipulation functions.

It is recommended to configure analyses at the template level.

The following procedure can be used to configure an Expression analysis using a template:

1) In the AF Database Library, create a new analysis template of type Expression.
2) Define expressions for the calculations in the analysis template.
3) Define the scheduling for the analysis template.
4) Define output attribute templates to store results.
5) Create the PI tags used to store the results.
6) Evaluate and preview the data to validate calculations.
7) Backfill the calculation if required.
8) Confirm the backfilled data
9) Backfill the data for other elements sharing the same template.

6.2.1 Directed Activity – Calculate Utilization for Assets

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

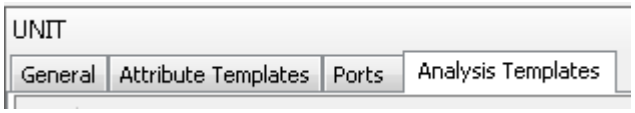
The Utilization is a percentage that represents the amount of electrical power that a unit produced against its theoretical capacity. Configure, test, run, and validate analyses to calculate the percent utilization of all generating units.

Approach:

- In PI System Explorer, navigate to the Library in the Fleet Generation database.
- Under Element Templates, select the UNIT element template.
- Select the Analysis Templates tab to configure the multi-lined expression for Utilization:
  Utilization = Total Hourly Gross Generation / Hourly Capacity
- Specify and configure an attribute template to store the results.
- Schedule the calculation to run periodically every hour.
- Backfill unit GAO01 for the past seven days.

Approach

From the **Unit Template**, found in the Library plug-in of the Fleet Generation database, select the **Analysis Templates tab**.



Configure a **new analysis**. Name the analysis Utilization and set the analysis type to Expression.

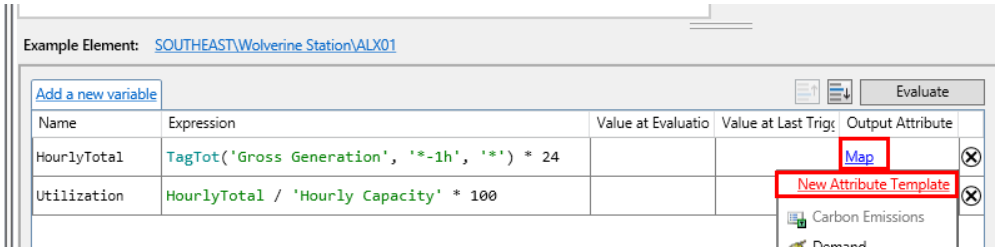
Configure the expressions for the hourly total of Gross Generation and Utilization.

HourlyTotal = TagTot('Gross Generation','*-1h','*') * 24
Utilization = HourlyTotal / 'Hourly Capacity' * 100

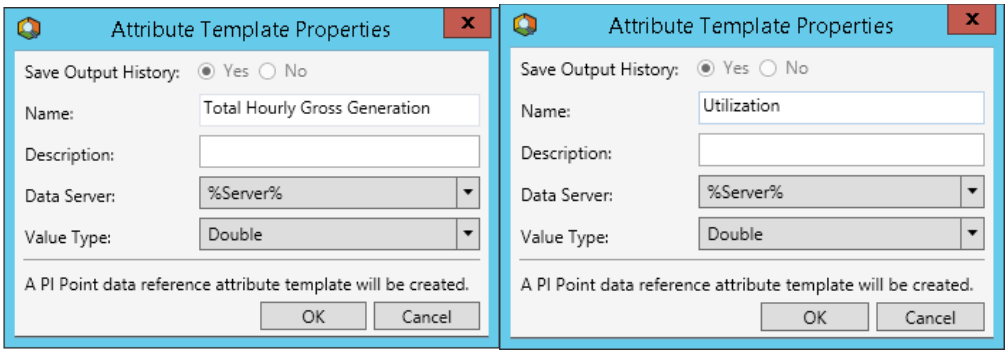
Name	Expression	Value
HourlyTotal	TagTot('Gross Generation','*-1h','*')*24	
Utilization	HourlyTotal / 'Hourly Capacity' * 100	

Note: The HourlyTotal must be multiplied by 24, as the Performance Equation function TagTot assumes the units of the input attributes are per day. Conversion factors should not be used elsewhere with PI Asset Framework, as UOM conversions occur automatically.

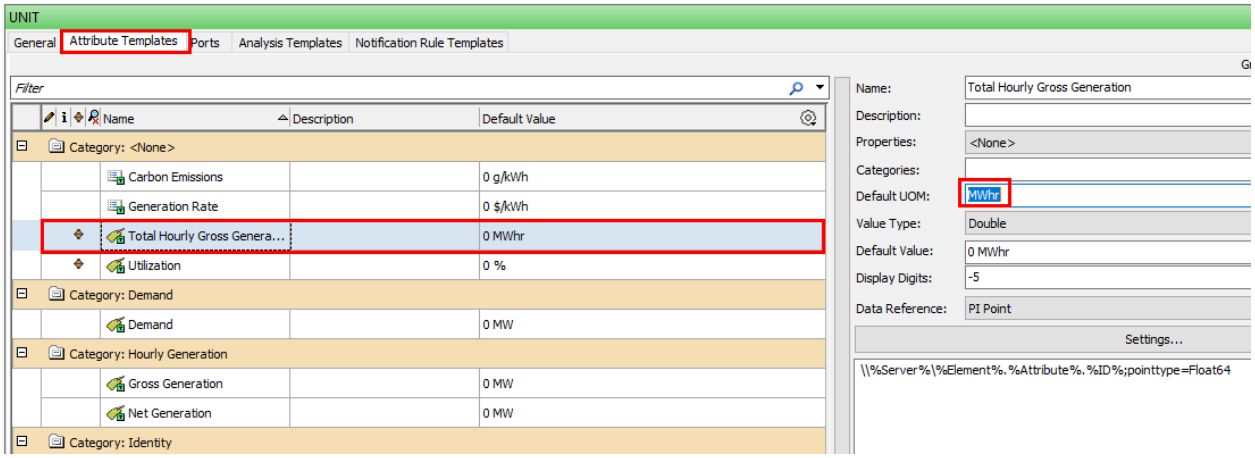
Define two new output attribute templates by clicking **Map** -> New Attribute Template.



Name them **Total Hourly Gross Generation** and **Utilization**, respectively.

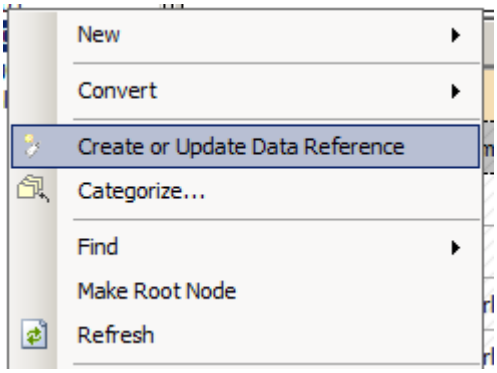


The UOMs can be set to **MWh** and **%** in the Attribute Templates tab:



Create the PI Tags

After the new attribute template has been configured, switch over to the Element Hierarchy. The attribute values for the new tags should be “Pt Created.” If not, right-click on the root Elements object. Select Create or Update Data Reference to automatically create the PI tags to store the calculated results.



Switch back to the Unit Template Analysis Templates tab to **schedule** the Analysis Template to run periodically at the top of each hour.

Set a Periodic Schedule

- (•) Hours, minutes, and seconds
- ( ) Sub-seconds
- ( ) Daily

Period

Specify the amount of time between evaluations.

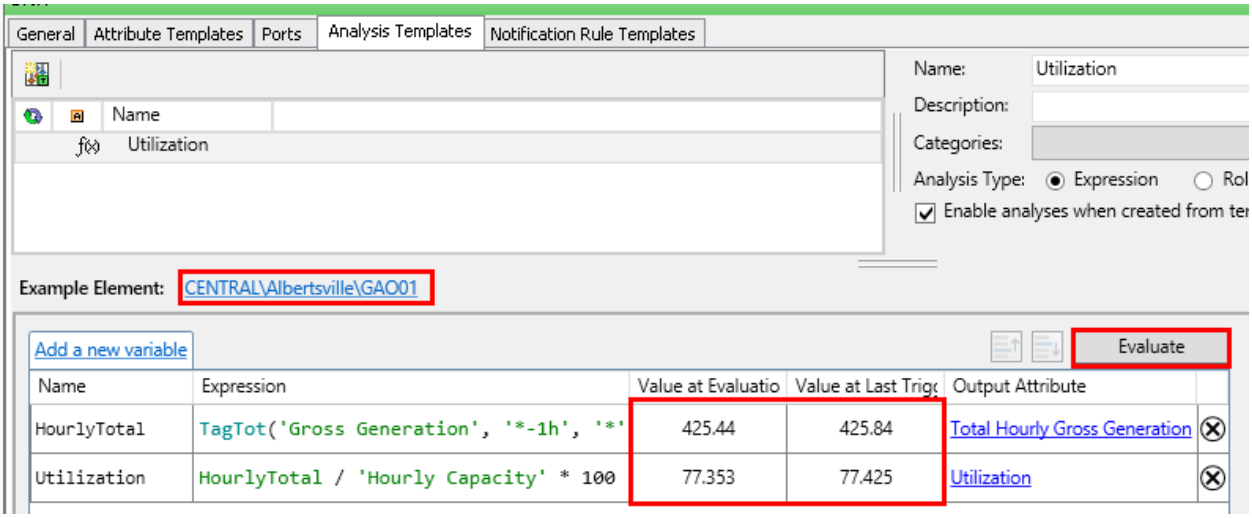
01 h 00 m 00 s

☐ Specify Offset

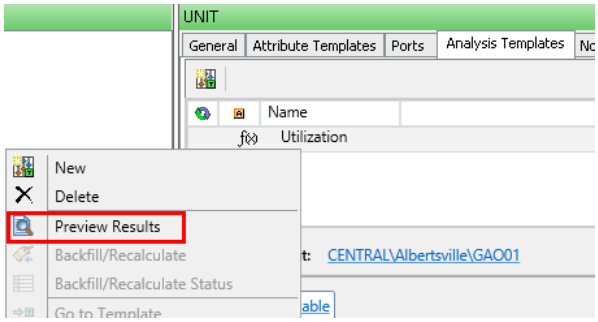
Example evaluation times

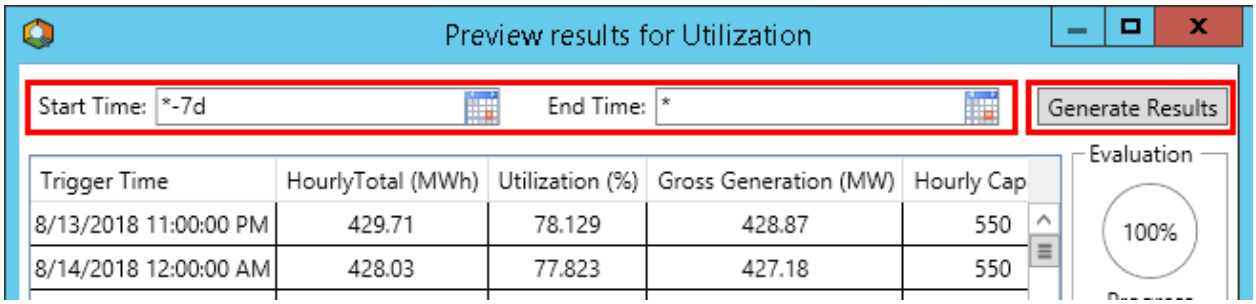
5/20/2014 1:00:00 AM
5/20/2014 2:00:00 AM
5/20/2014 3:00:00 AM

Set GAO01 as the Example Element and click on the Evaluate button to validate the expressions.



Prior to backfilling data into the PI Data Archive, it is usually a good idea to preview the results. Right-click **Utilization** and select **Preview Results**. Look at the results for the past 7 days:





6.2.2 Directed Activity – Backfill Utilization

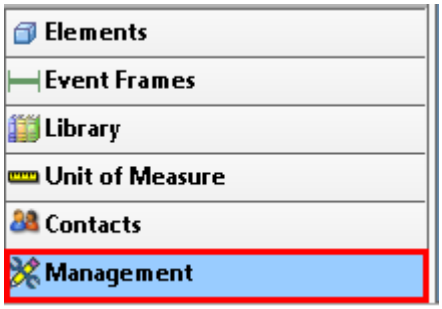
In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

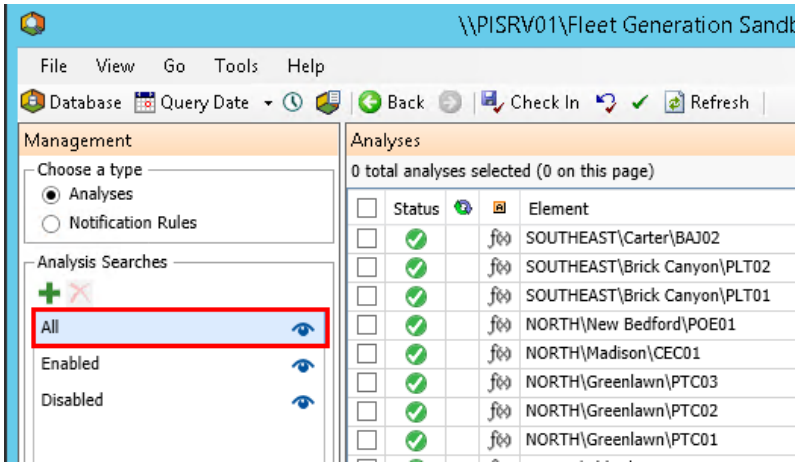
At this point, all the analyses for event frame generation have been set up for all the units of Fleet Generation. In order to calculate past Utilization values and generate history for analysis, the calculations must be backfilled.

Approach:

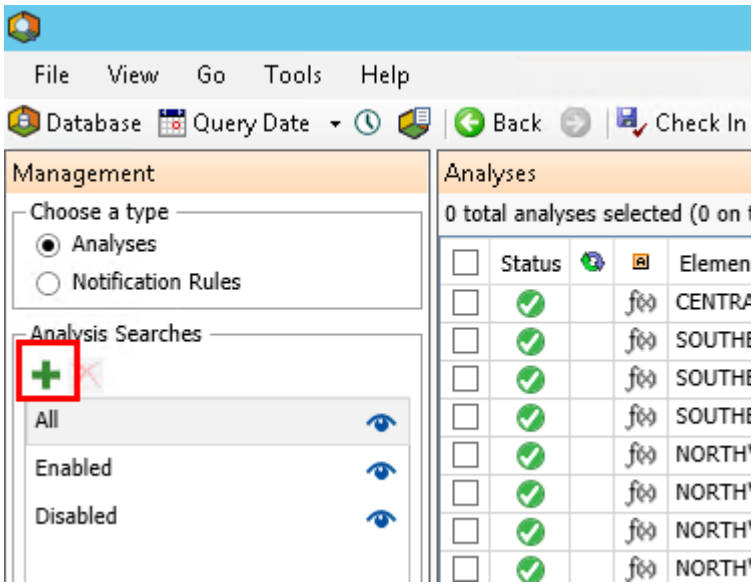
From PI System Explorer, select the Management plugin



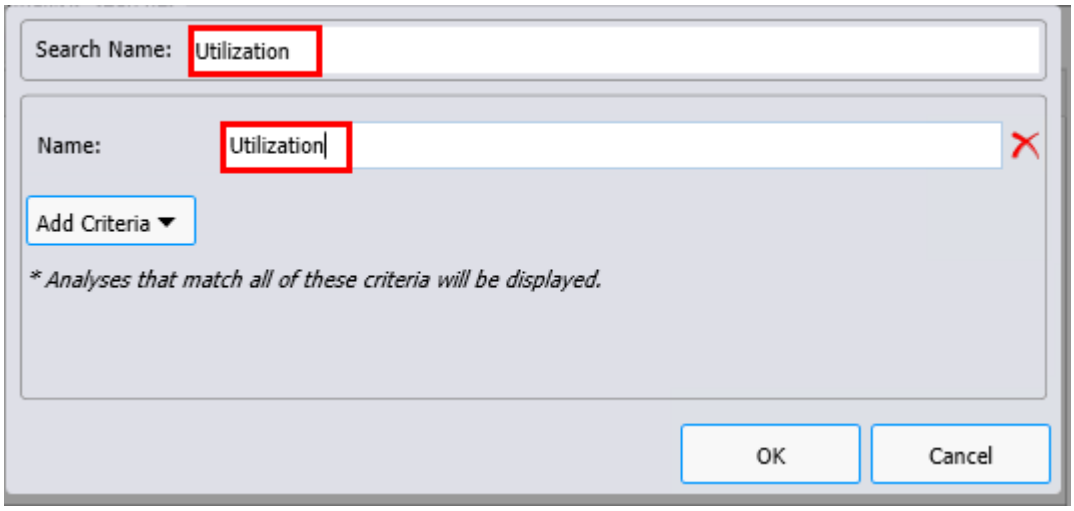
Right now, the only Analyses that exist are those we just created, so one can simply select **All** or **Enabled** to view the Utilization Analyses that we want to backfill.



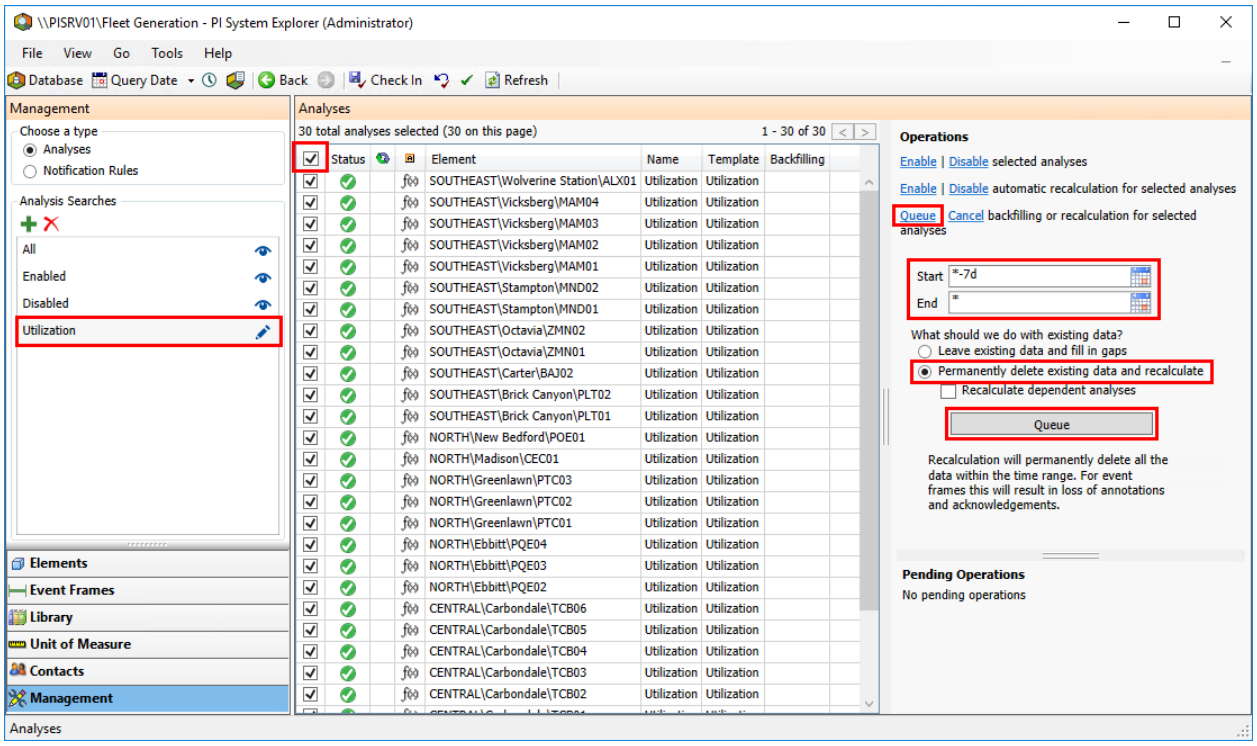
Normally there would be several types of calculations, so we'd want to filter them by setting up a search. Create a new search:



Name the search **Utilization**, then do **Add Criteria -> Name** and enter the name of the Analyses and click **OK**.



Click the checkbox to select **all** Utilization Analyses. Then Select the Queue operation and set the start time to "***-7d**" and the end time to "*****", select "**Permanently delete existing data and recalculate**", then click **Queue**:



6.2.3 Exercise – Calculate Generating Efficiency

This solo or group activity is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the activity.

Objective:

Not all of the electricity produced by our generators will make it out to the grid. Some will be consumed by the internal circuity in the generator itself. The net generation is defined as the amount of gross generation, or the amount of electricity that a generator produces, less the electricity required to operate the unit. Calculate the generating efficiency, or the *ratio between the net generation to the gross generation*, expressed as a percentage.

Which unit is performing with the greatest efficiency?

Approach:

- In the PI System Explorer, navigate to the Library in the Fleet Generation database.
- Under Element Templates, select the UNIT element template.
- Select the Analysis Templates tab to configure the expression for generating efficiency, named **Generating Efficiency**.
- Specify and configure an attribute named Generating Efficiency to store the results with units of %.
- Schedule the calculation to run periodically every hour.
- Evaluate the calculation using example element GAO01 and preview the results.
- Backfill all Efficiency analyses for the past seven days.

6.3 Rollups

The second analysis capability of the PI Analysis Service Analytics is known as rollups. Rollups allow for the calculation of summary statistics for a set of attribute values.

The types of summary statistics that are allowed are:

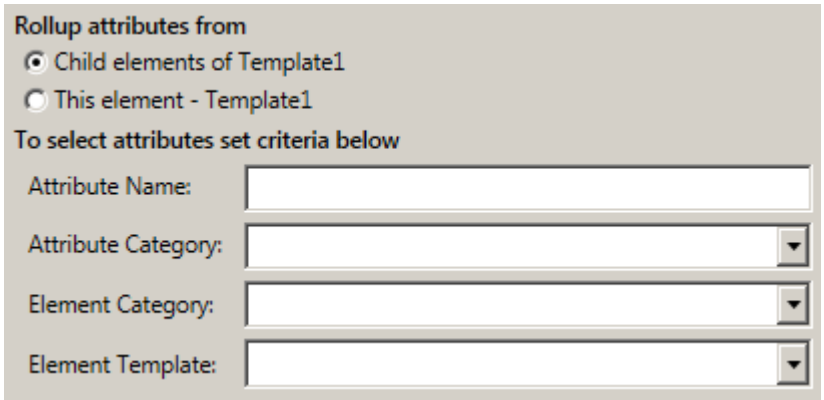
- Sum
- Average
- Minimum
- Maximum
- Count
- Median

Examples of rollup calculations include:

- Total mass of all contents in a tank farm
- Total production from all generating units for a particular site
- Maximum temperature of boilers within a building
- Average engine temperature of mining trucks
- Average temperatures for each asset with varying temperature sensors.

Selecting attributes to rollup

Attributes used in rollup calculations can come from 1) attributes from child elements relative to the element of interest or 2) the element of interest. One can set search criteria to specify the specific attributes to rollup. Depending on the source of the attributes (child elements or current element), the search criteria includes a masking pattern for the 1) Attribute Name, 2) Attribute Category, 3) Element Category, and 4) Element Template.



What is an element Example?

During the configuration of a rollup template analysis, when the source of the attributes to roll up are from the child elements, PI System Explorer is not aware of which parent element to retrieve child elements from. As such, when configuring a roll-up analysis template, you will need to specify an example element. Note that when configuring a roll-up at the element level, one will not need to select an example element as the child elements are from the specific, selected element.

Scheduling and backfilling

Similar to Expressions, the rollup analyses can be scheduled to run as new events come into the PI Data Archive or scheduled to run periodically. The PI Analysis Service also allow the results from Rollup calculations to be written back to the PI Data Archive.

The general process to properly configure and backfill an analysis template is:

1) Create a new analysis of type Rollup.
2) Define the source of the attributes to rollup (child element or current element).
3) Select the type(s) of summary statistics to calculate.
4) Define output attributes to store results.
5) Define the scheduling for the analysis.
6) Create the PI tags used to store the results.
7) Evaluate and preview the data to validate calculations.
8) Backfill the calculation.

6.3.1 Directed Activity – Calculate Average Utilization for Substations

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

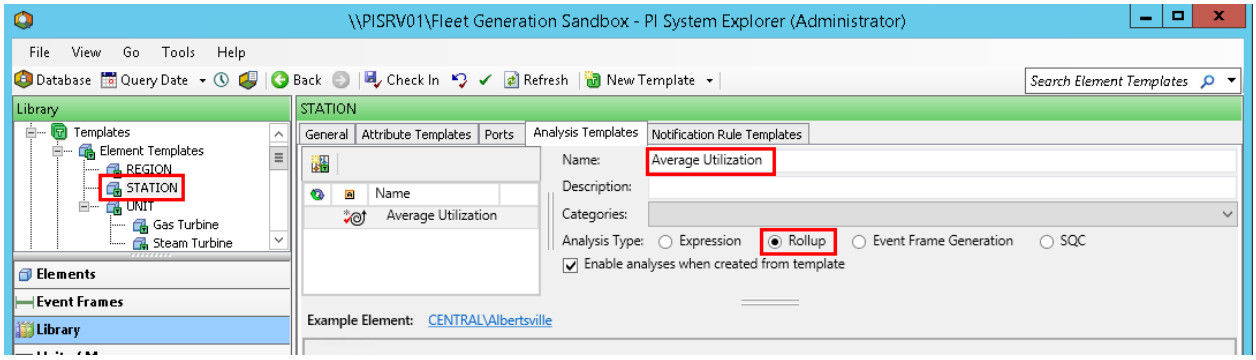
Objective: Management would like to have visibility over the average percent utilization of all generating units for each substation. Roll up the average utilization to the substation level.

Approach:

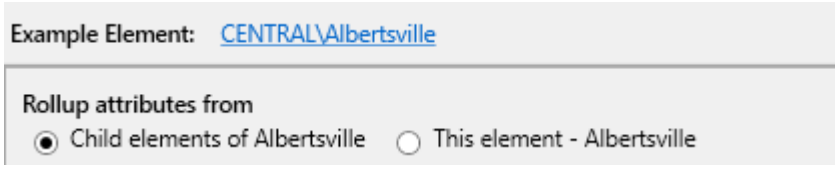
- Open up the Station Element Template from the Fleet Generation Database Library.
- Add a new analysis called Average Utilization with analysis type of Rollup.
- Select Central\Albertsville as the example element.
- Specify the criteria to select the attributes used for the rollup calculation.
- Select the summary statistic function for the average.
- Specify the output attributes (be sure to create the tags).
- Schedule the calculation to be event-triggered.
- Verify data.
- Backfill for the past 7 days.

Step-by-Step Approach

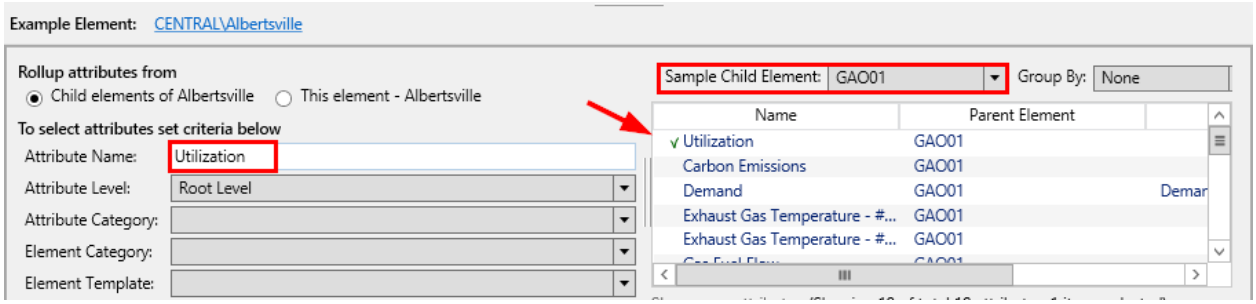
From PI System Explorer, go to the Library. Then select the Station Element Template. From the Analysis Templates tab, create a new Analysis called **Average Utilization** with Analysis Type **Rollup**.



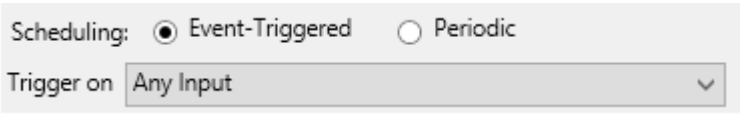
Specify the rollup attributes from child elements and set the example element to be **Central\Albertsville**.



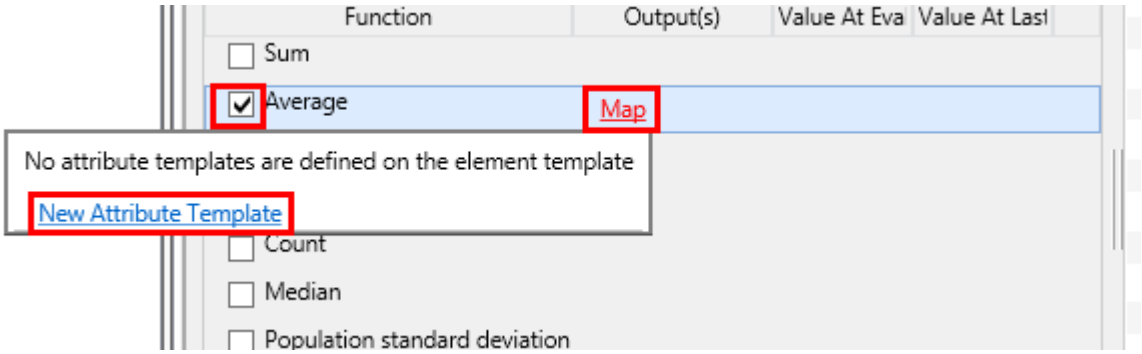
Set the attribute name field to **Utilization**. This mask will automatically select all Utilization attributes from the child elements of the Albertsville station. However in the preview only the Utilization from the Sample Child Element will be shown:

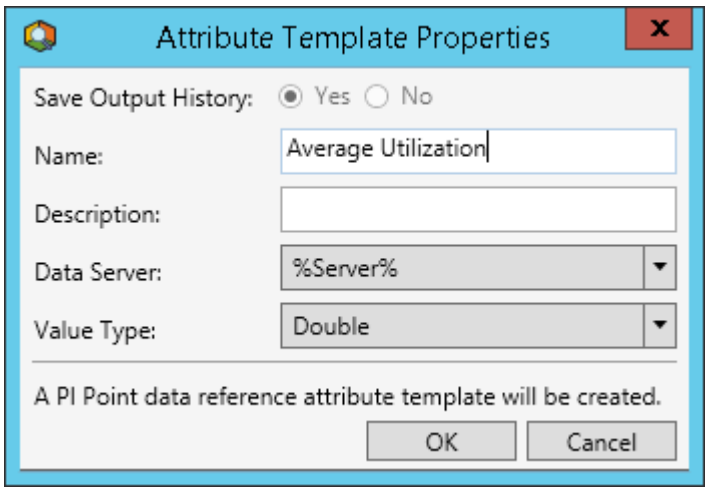


Set the scheduling to be event-triggered. Each time the Utilization analysis finishes calculating each hour, the rollup analysis will run.

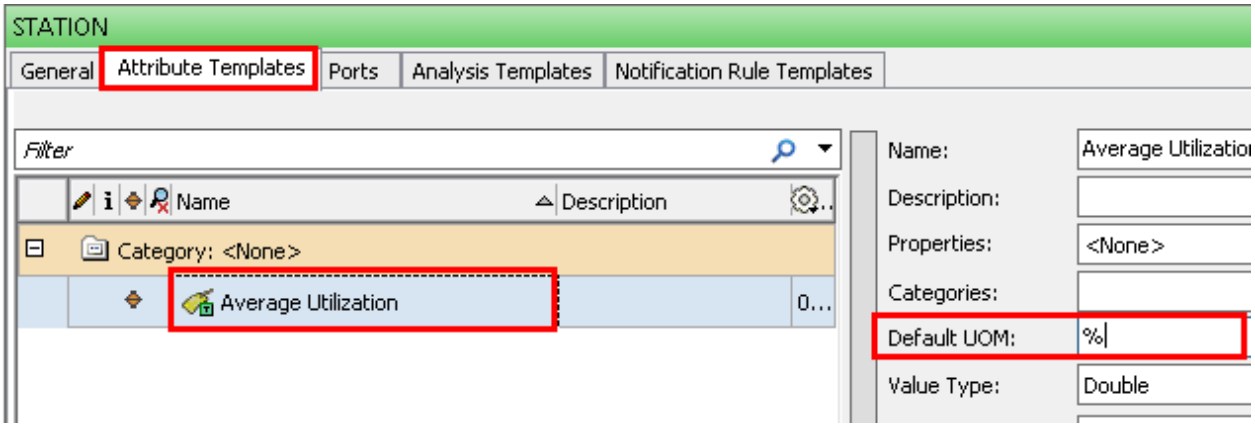


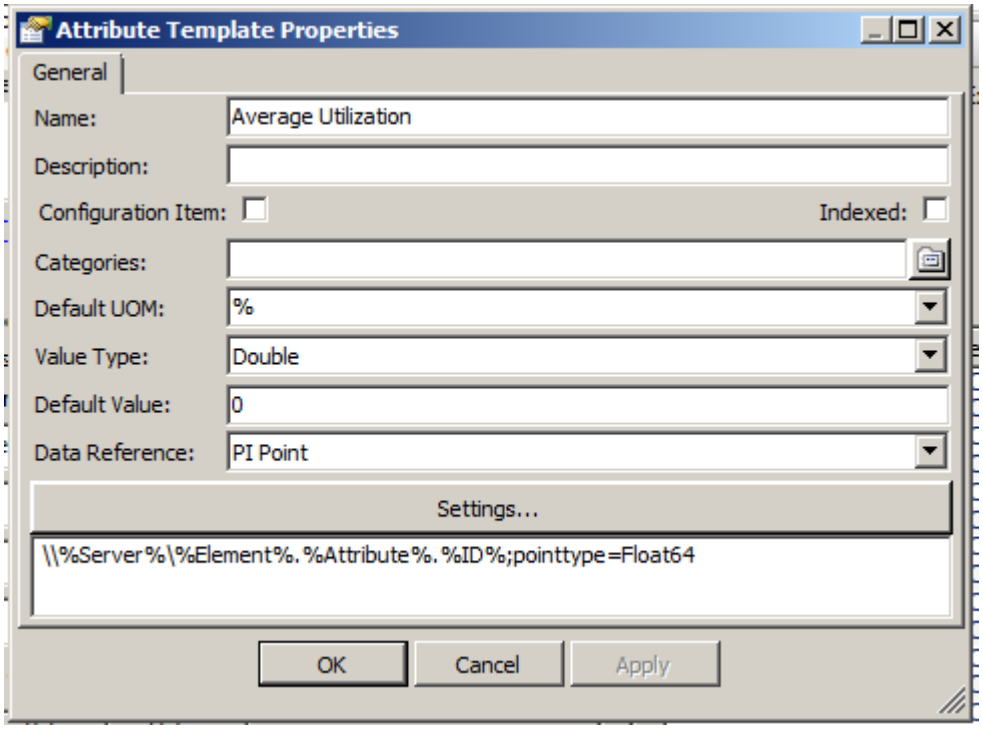
Select **Average** as the rollup function and create a new Output Attribute called **Average Utilization**.



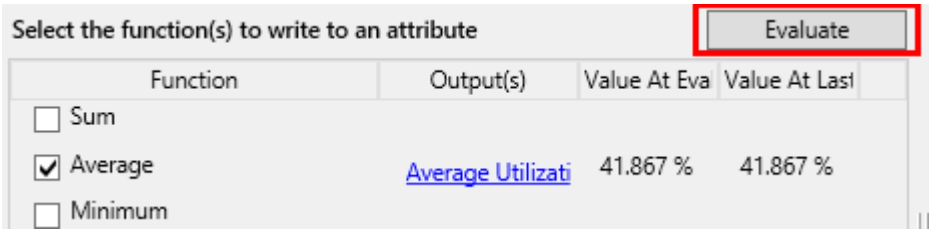


Set the default **UOM** of this new attribute to **%** in the Attribute Templates tab:





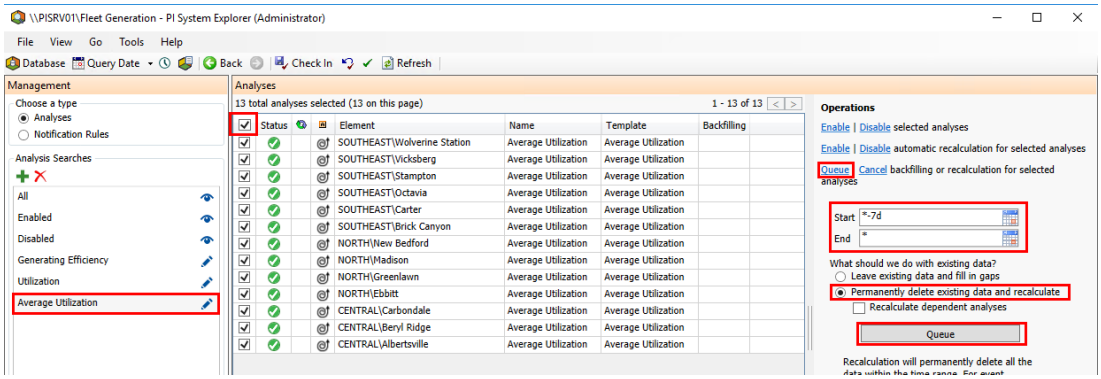
In the **Analysis Templates tab**, Click on the **Evaluate** button to verify the result of the rollup function.



Check-in your changes.

From the element hierarchy, verify that the PI tag exists for the attribute.

From the **Management** pane, backfill your Average Utilization rollup analyses for the past **7 days** and verify the data has been backfilled by trending the Average Utilization attributes.



6.3.2 Exercise – Calculate Total Hourly Gross Generation for Each Station

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective:

Management would like to gain more insight into the Total Hourly Gross Generation at each station. Create a **rollup analysis** to totalize the Total Hourly Gross Generation at the station level.

Which station produces the most power?

Approach:

- Open up the Station Element Template from the Fleet Generation Database Library.
- Add a new analysis called Total Hourly Gross Generation with analysis type of Rollup.
- Select Central\Albertsville as the example element.
- Specify the criteria to select the attributes used for the rollup calculation.
- Use the Sum function and output Attribute **Total Hourly Gross Generation**.
- Specify the output attributes (ensure tags are created).
- Set the **UOM** to **MWhr**.
- Schedule the calculation to be event-triggered.
- Verify data using Evaluate and Preview Results.
- Backfill for the past 7 days and verify.

7 Event Frame Generation

Events are important process or business time periods that represent something happening that affects your operations. In the PI System, events are known as event frames. Thanks to PI Event Frames, you can analyze your PI data in the context of these events rather than by continuous time periods. Instead of searching by time, PI Event Frames enables users to easily search the PI System for the events they are trying to analyze or report on.

With PI Event Frames, the PI System helps you capture, store, find, compare and analyze the important events and their related data.

PI Event frames represent occurrences in your process that you want to know about, for example:

- Downtime tracking
- Environmental monitoring excursions
- Process excursions
- Product tracking batches
- Equipment startups and shut downs
- Operator shifts

The following table presents some of the features and advantages of PI Event Frames:

Flexibility	✓ Reference multiple elements within the same event. ✓ Support multiple overlapping events on a PI AF element. ✓ Capture any event; a "batch" is just one type of capturable event.
Powerful search	✓ Search by time range, type of event or event frame attribute. ✓ Most common search attributes can be configures as indexed attributes to speed up end-user searches
Scalability	✓ PI Event Frames are extremely scalable.

A PI Event Frame is defined by three characteristics:

1. Name.
2. Start time and end time: defines the event's time range.
3. Context: event attributes and related assets.

7.1 What are Event Frames?

7.1.1 Creating Event Frames

The Fleet Generation database contains a series of Elements representing the regions and units associated with each generation plant. In order to keep up with the power demands, it is important that the plant is up and running. We need to keep track of the downtime associated with the generation plant and control the level of temperature for gas turbines.

A 'Unit Status' attribute is associated with each generating plant in our hierarchy. This attribute will be used to monitor the downtime associated with each plant. 'Exhaust Gas Temperature - #1 Probe' and 'Exhaust Gas Temperature - #2 Probe' are associated only with gas turbines and they will be used to monitor the temperature anomalies.

7.1.2 Time Range Retrieval Methods

There are three time range retrieval methods, the use of which depends on what data is to be captured, and how it is to be displayed.

Time Range

This method allows a time range to be supplied by the end user. When any single value query is made, this period of time is used for calculations. If, however a period of time is supplied from an application, such as a generated Event Frame or Vision display, then the user specified time range is discarded and the application time period is used.

Time Range Override

The Time Range Override behaves in the same way as the Time Range method during all single value queries, as uses the user specified time period. When a period of time is supplied from an application, the application time range is discarded and the user specified period is used.

Not Supported

Not Supported does not allow for a time range to be supplied by the end user. As such, an error is returned by any request for a single value. If a period of time is supplied however, then this range is adopted by the method for the calculation. The result is then the same as the Time Range method.

There are different use cases for the methods, so care must be taken to ensure the correct method is used.

METHOD	SINGLE VALUE	APPLICATION SUPPLIED
TIME RANGE	User Specified range result	Application Specified range result
TIME RANGE OVERRIDE	User Specified range result	User Specified range result
NOT SUPPORTED	Error: This attribute requires a Time Range to calculate a value in…	Application Specified range result

Single timestamp query results (sample element with 1h specifications)

ⓘ	Not Supported	This attribute requires a Time Range to calculate a value in '…
	Time Range	110.93823012085859
	Time Range Override	110.93823012085859

Application supplied time range query results (sample 3h event frame)

	Not Supported	259.00273501602908
	Time Range	110.93823012085859
	Time Range Override	259.00273501602908

7.1.3 Directed Activity – Create a Temperature Anomaly Event Frame Template

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

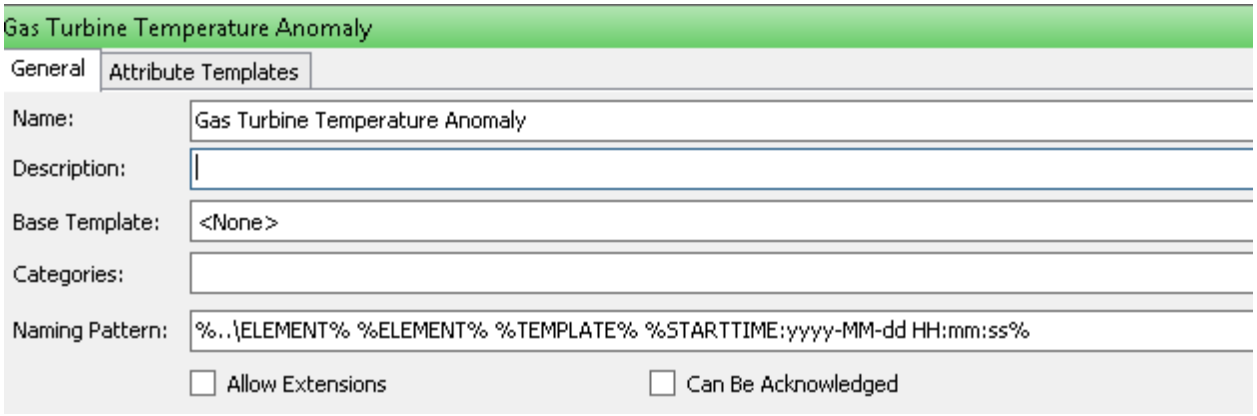
Objective:

The gas turbines in the Fleet Generation database each have two temperature sensors. Create an Event Frame template with appropriate attributes to help monitor and analyze potential issues with gas turbines. The event frame should capture the real-time data specific to gas turbines and the current status and duration of the gas turbine.

Approach:

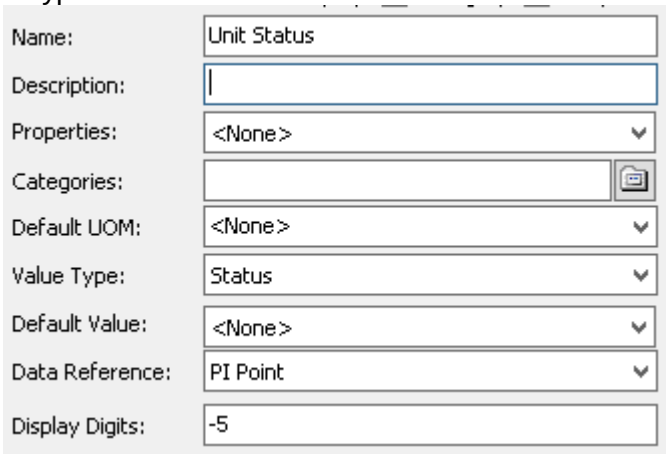
- Create an **Event Frame template**.

In the Library, create an Event Frame template called "**Gas Turbine Temperature Anomaly**". Set the Naming Pattern to %..\ELEMENT% %ELEMENT% %TEMPLATE% %STARTTIME:yyyy-MM-dd HH:mm:ss%



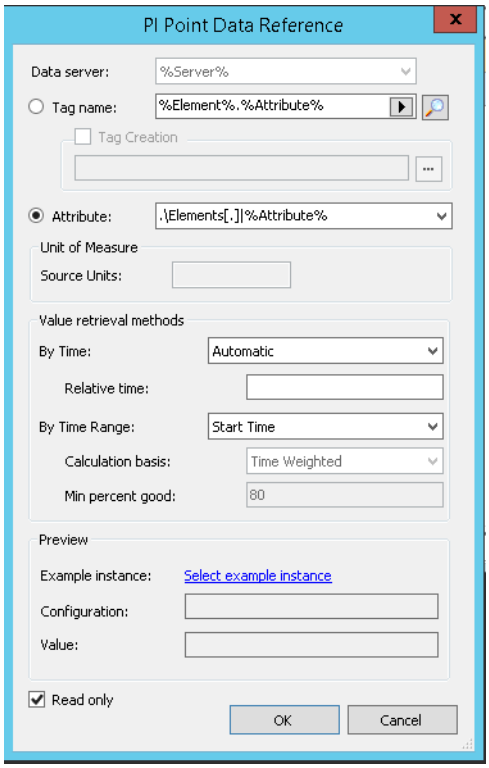
Select the Attribute Templates tab. Right click in the white space to create an attribute.

Name the Attribute **Unit Status**. Select **Enumeration Sets => Status** as the value type.



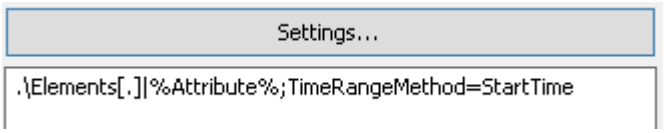
Select the PI Point Data Reference, then select **Settings…**

Click the radio button next to Attribute, and enter **.\Elements[.]|%Attribute%**. The Event Frame references a PI AF Element. The [.] syntax points to this PI EF Template's primary referenced PI AF element within the Elements collection. Set the By Time Range dropdown option to "Start Time."

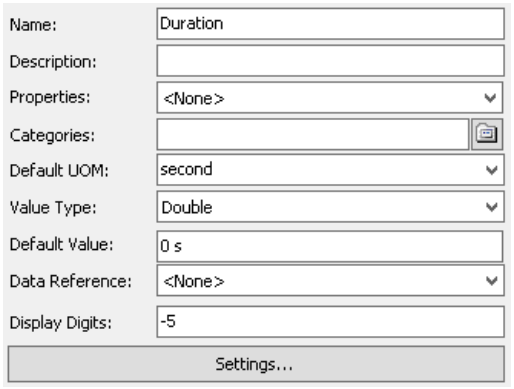


Note: Substitution parameters cannot be used to make a reference to an attribute from the Element Template that is not a PI Tag.

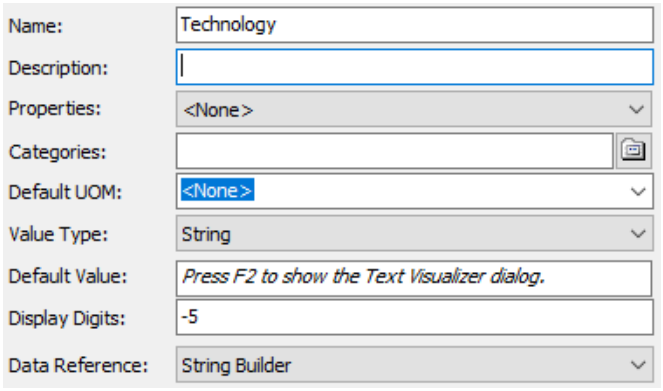
Upon completing the definition, click **OK**. The Settings will be completed as seen below:



Create a second attribute to store the Duration (UOM = second) of the event frame. **The Duration attribute will be populated by the new EventFrame() function in a later exercise**. It's just a placeholder for now.

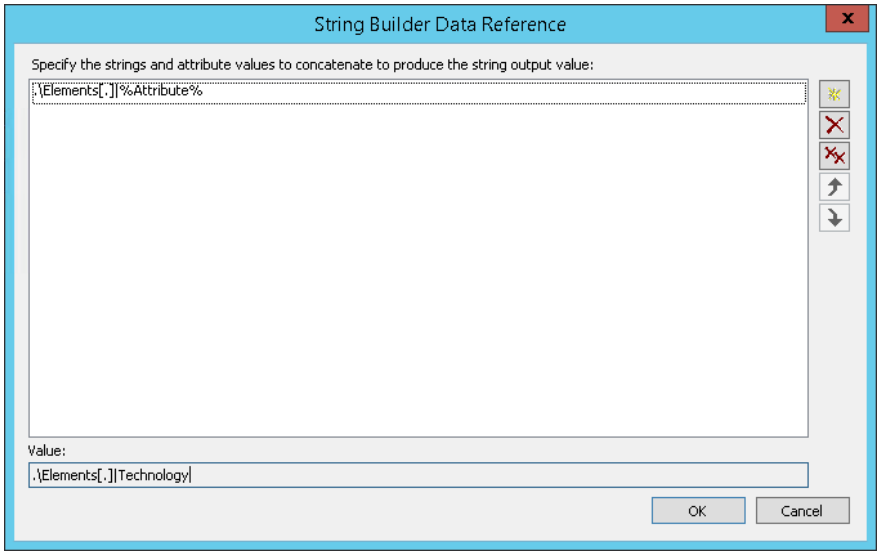


Create a third attribute to store the Technology. For the Value Type, select String and for the Data Reference, select String Builder.



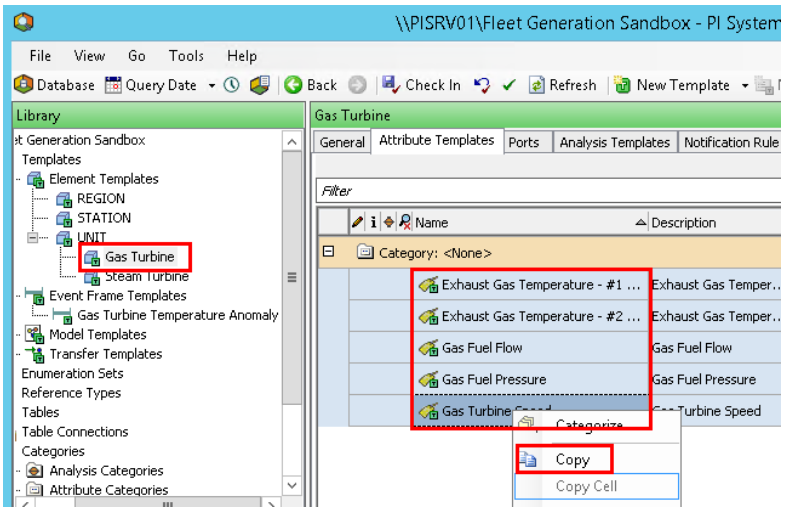
Note: When the event frame attribute's data reference is set to PI Point, the syntax .\Elements[.]|Attribute only allows for the reference to PI Point Data Reference attributes. Element attributes configured as formulas and table lookups cannot be passed to event frames using a PIPoint Data Reference. Instead, for attributes configured as formulas or table lookups, select String Builder as the data reference.

Set the settings for the attribute as .\Elements[.]|%Attribute%:

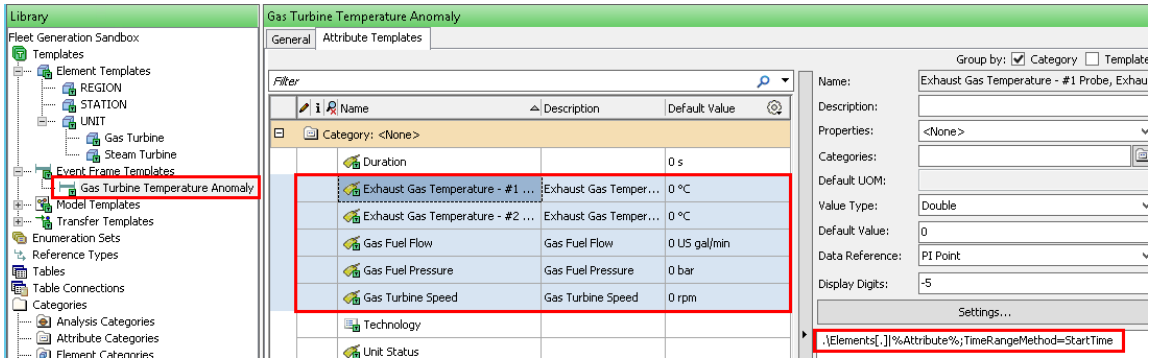


Continue to create the following additional attributes. Make sure units are properly set. **The fastest way to accomplish this is to copy and paste these attributes templates from the Gas Turbine element template**.

Exhaust Gas Temperature - #1 Probe
Exhaust Gas Temperature - #2 Probe
Gas Fuel Flow
Gas Fuel Pressure
Gas Turbine Speed



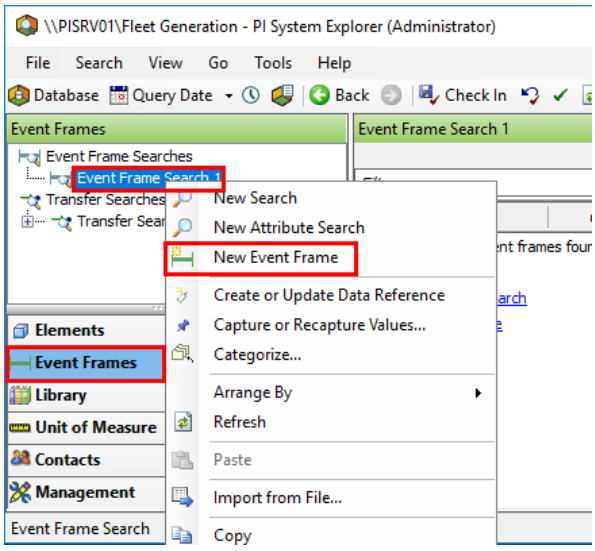
Once these 5 attribute have been pasted into the Gas Turbine Temperature Anomaly Event Frame Template, select them **all** and enter **.\Elements[.]|%Attribute%;TimeRangeMethod=StartTime** as the configuration string (copy/paste from Unit Status) to set the data references and retrieval method in bulk:



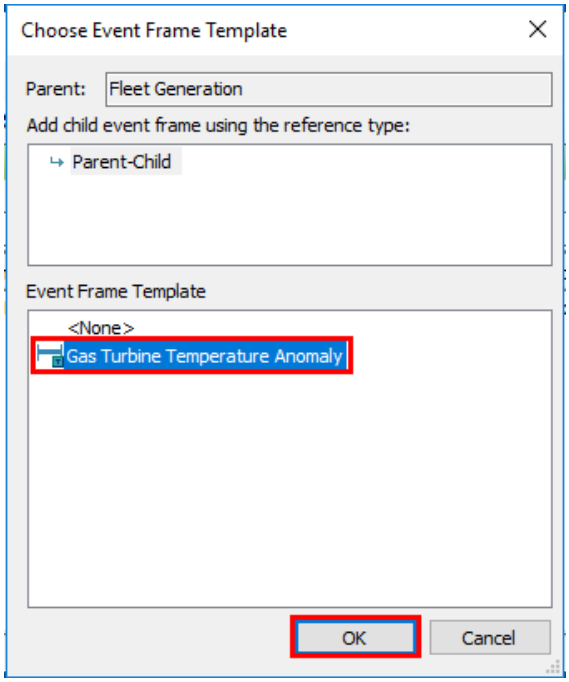
Check in your changes.

Note: %attribute% will substitute in the name of the event frame attribute template. This will then point to the corresponding attribute in the referenced element. You can also select multiple attributes when making modifications to the attribute configuration.

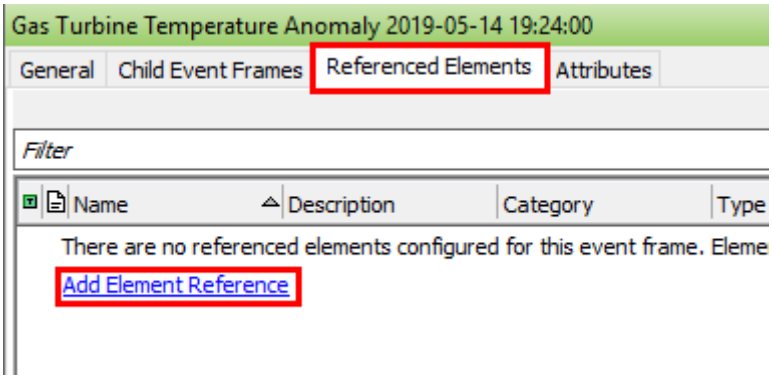
Create a manual Event Frame to test the PI Point Data Reference configurations and naming pattern. From the Event Frames section, right click on Event Frame Search 1 and create a New Event Frame.



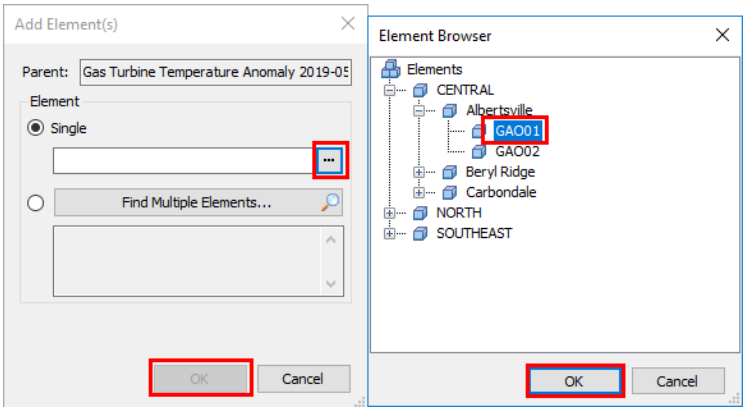
Use the Gas Turbine Temperature Anomaly EF Template:



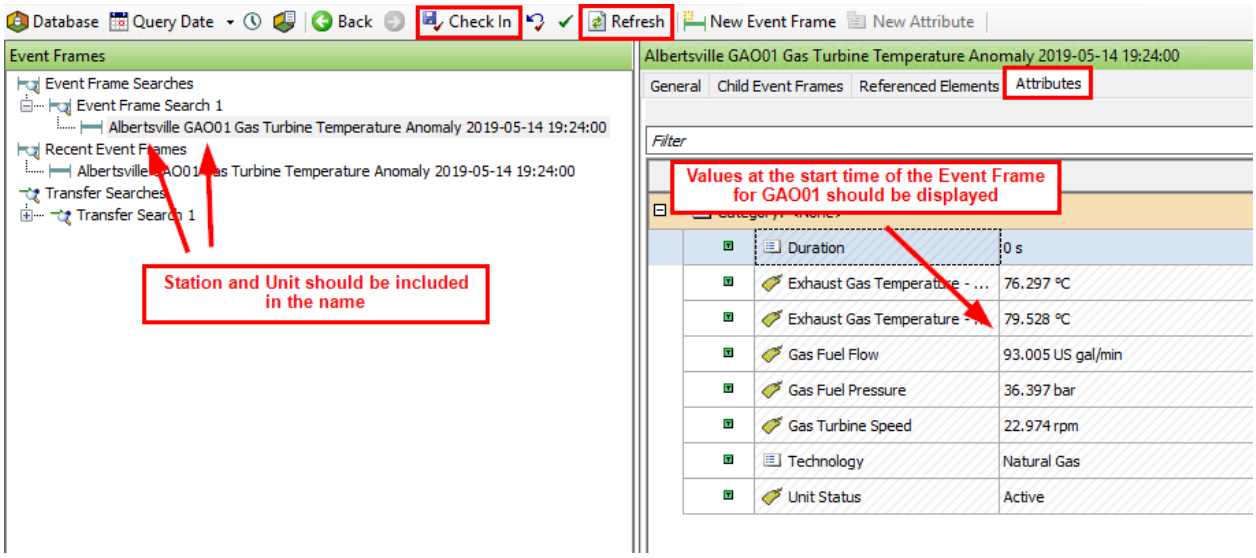
Add an Element Reference



Set to GAO01, OK, OK



Check the Attributes tab, check in, refresh, and confirm that there are no errors in the PI Point DRs (Attributes). Also confirm that the naming pattern resolved correctly. You will get different values than the screenshot since values are randomly generated and of course get a different timestamp (Event Frame start time) than when the screenshot was taken.


Database
Query Date
Back
Check In
Refresh
New Event Frame
New Attribute
Event Frames
Event Frame Searches
Event Frame Search 1
Albertsville GAO01 Gas Turbine Temperature Anomaly 2019-05-14 19:24:00
Recent Event Frames
Transfer Searches
Transfer Search 1
Station and Unit should be included in the name
Albertsville GAO01 Gas Turbine Temperature Anomaly 2019-05-14 19:24:00
General
Child Event Frames
Referenced Elements
Attributes
Filter
Values at the start time of the Event Frame for GAO01 should be displayed
Duration
0 s
Exhaust Gas Temperature - ...
76.297 °C
Exhaust Gas Temperature - ...
79.528 °C
Gas Fuel Flow
93.005 US gal/min
Gas Fuel Pressure
36.397 bar
Gas Turbine Speed
22.974 rpm
Technology
Natural Gas
Unit Status
Active

7.1.4 Exercise - Create Inactivity Event Frame Template

This solo or group activity is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the activity.

Objective:

Generating units sometimes trip or go down. Management would like to understand these downtimes, and determine how much demand was not serviced. Event frames can help capture and bookmark these events for future analysis. Develop an Event Frame template, called **Inactivity** using the same Naming Pattern as the previous exercise, with fields required to track the desired plant information to create reports for management. Specifically, management would like to know the following:

1. Unit Status – Real-time (copy/paste from previous exercise)
2. Duration – in seconds (copy/paste from previous exercise)
3. Technology – Metadata (copy/paste from previous exercise)
4. Hours Down – in hours (simple formula to convert seconds to hours)
5. Demand – Real-time (PI Point data reference)
6. Operator – Metadata (string builder)
7. Carbon Emissions in g/kWh – Metadata (string builder)
8. Total Demand in MWhr – Real-time, Aggregation of Demand

Hints:

- For **metadata**, use **String Builder** as the Data Reference.
- For **Total Demand**, configure the attribute's source units as MJ / s By Time as "**Time Range**", Relative time as "**-1s**" and By Time Range as "**Total**"
- Verify correct event frame template configuration through the creation of a test event frame.

7.2 Event Frame Generation

The Event Frames Generation analysis allows for the automated detection and generation of event frames in the PI AF database based on values from trigger attributes. The type of events and the types of data captured inside each event are defined with event frame templates in PI AF.

Some notable features of Event Frame Generation in the PI Analysis Service include the following:

Generate events: Easily configure event generation and automatically generate your events from the trigger tags that are already collecting data in the PI Data Archive.

Handle multiple event types: Generate all your different event types, such as downtime, excursions, batches, and other events, on the same asset with no restrictions on overlapping events.

Standardize using event frame templates and populate event attributes: Different event types have different attributes and information that are important for analysis. Standardize your events using event frame templates, and use the PI Analysis Service to automatically populate event's attributes with data from the PI Data Archive and PI Asset Framework.

Backfill events: PI Analysis Service enables you to define your history backfill time window, then it backfills the events from previous time periods automatically.

Using PI AF element attributes as event triggers or event attribute values: Trigger conditions for event frames can be linked to element attributes.

Configure using PI AF element templates: Apply the configuration of event frame detection and generation to PI AF element templates. The same event detection automatically applies to newly created assets of the same asset type. There is no need to configure the event frame generation again.

Root Cause: Event frames are great for capturing events that have occurred. However, often times, the time period prior to the event provides more information on the cause of the event. PI Analysis Service allows for root cause analysis and will capture a fixed time period (default five minutes) before the event start time for further analysis. This will be recorded as a Child Event Frame.

Time True: The trigger condition for event frames could potentially be noisy. PI Analysis Service allows for the specification of a minimum time true period before an event frame will generate.

7.2.1 Directed Activity – Gas Temperature Anomalies

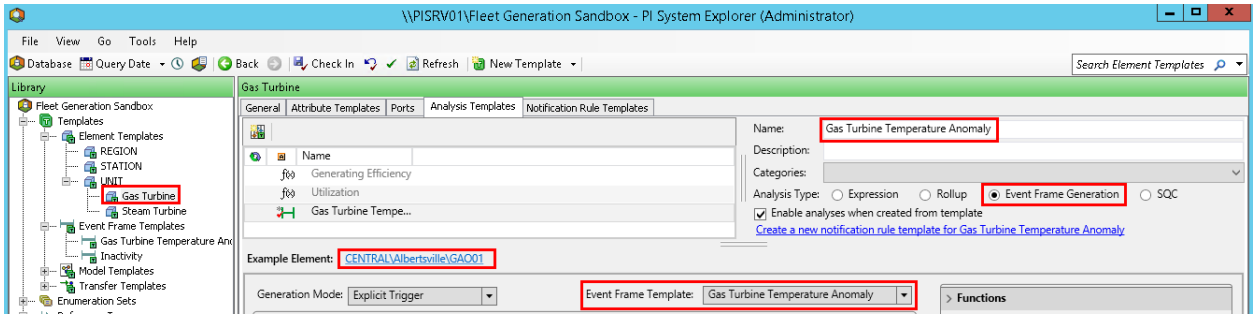
In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

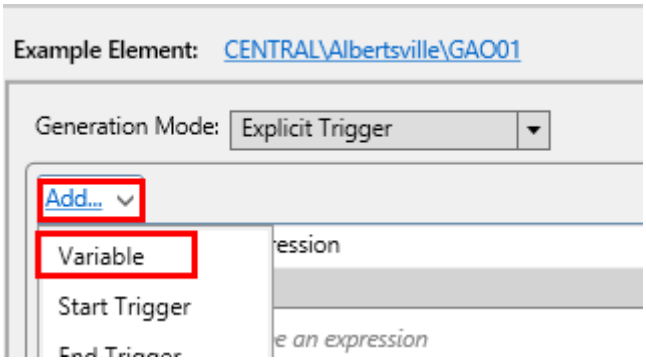
Each gas turbine has multiple temperature sensors. If any temperature reading deviates more than 20% from the average, then servicing is required. Use the Gas Turbine Temperature Anomaly Event Frame Temperature to help define these types of events.

Approach:

From the Fleet Generation Library, select the **Gas Turbine Element Template** and select the **Analysis Templates tab**. Create a new analysis template called **Gas Turbine Temperature Anomaly**, Set the example element to GAO01, and set the Event Frame Template to Gas Turbine Temperature Anomaly.



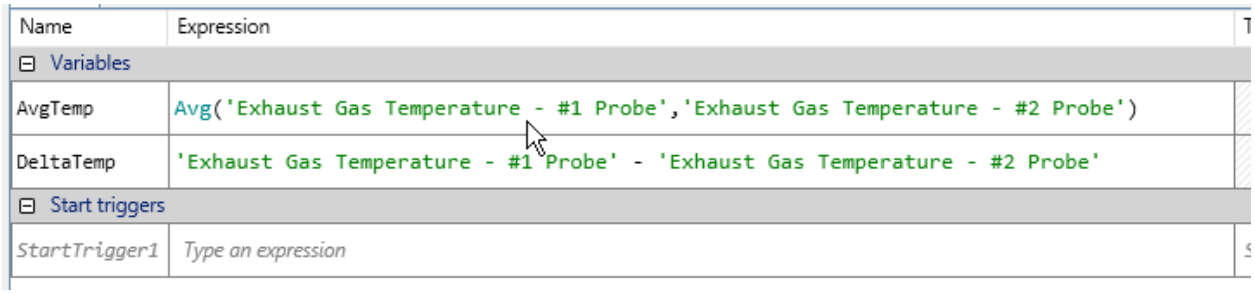
Add two new variables called **AvgTemp** and **DeltaTemp**.



Set the expressions to:

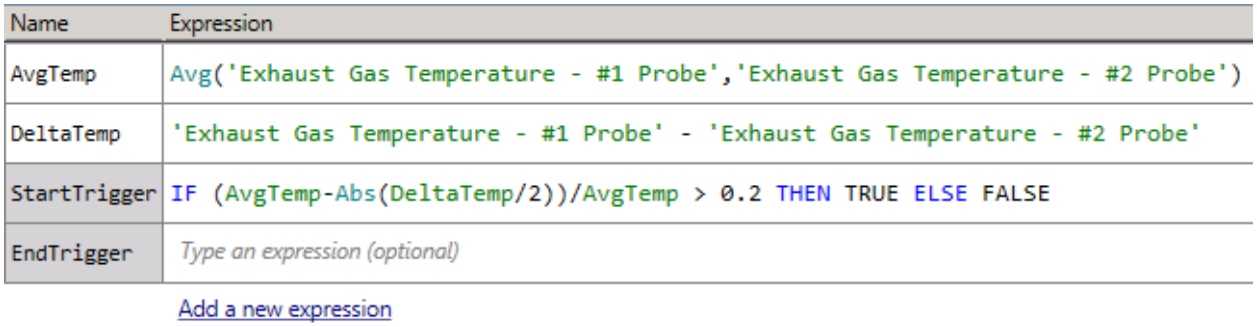
Avg('Exhaust Gas Temperature - #1 Probe','Exhaust Gas Temperature - #2 Probe')

'Exhaust Gas Temperature - #1 Probe' - 'Exhaust Gas Temperature - #2 Probe'

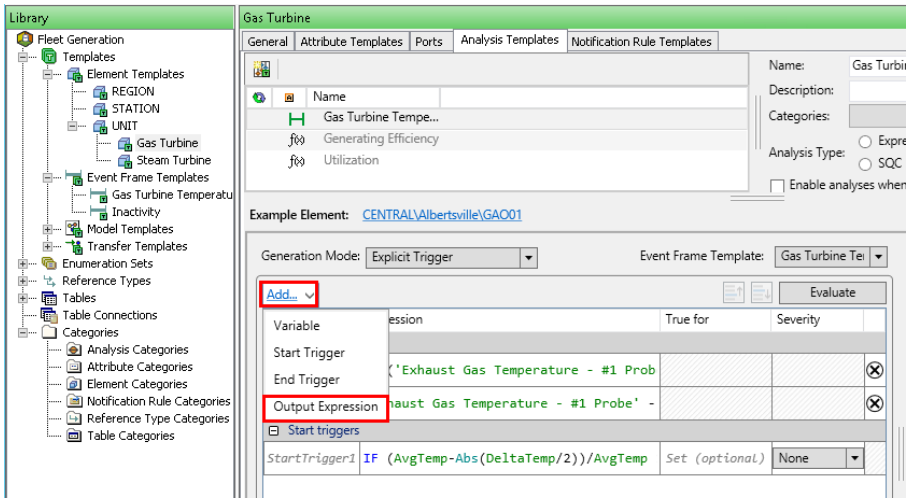


Define the **StartTrigger** as:

IF (AvgTemp-Abs(DeltaTemp/2))/AvgTemp > 0.2 THEN TRUE ELSE FALSE



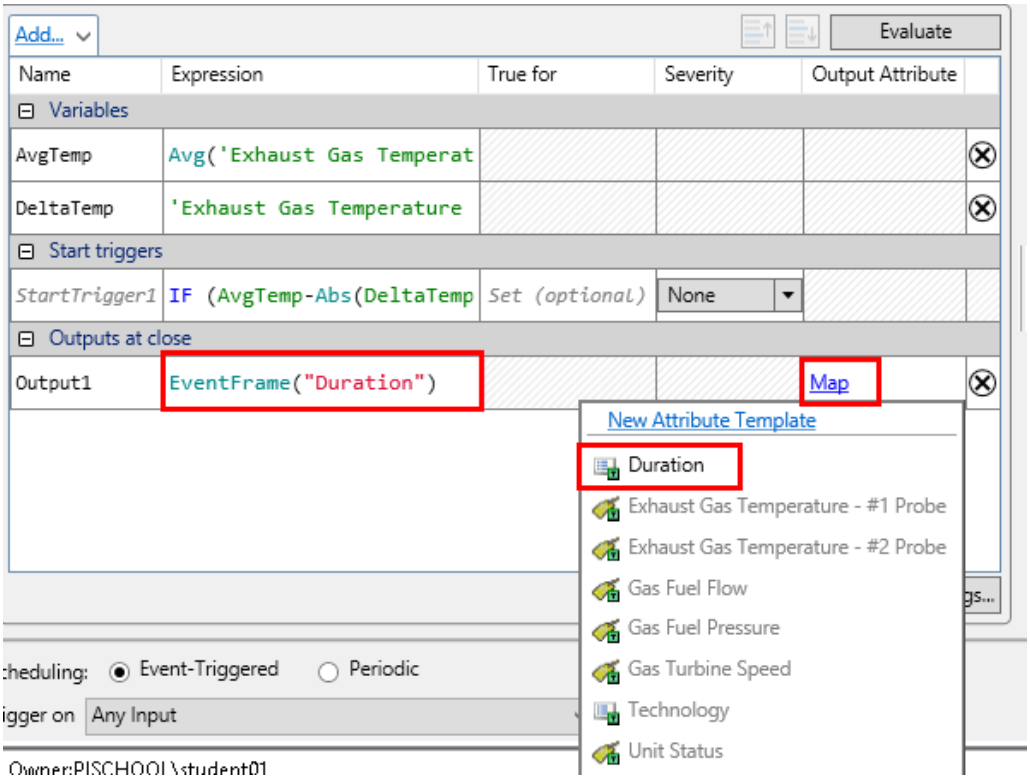
Add an **Output Expression**



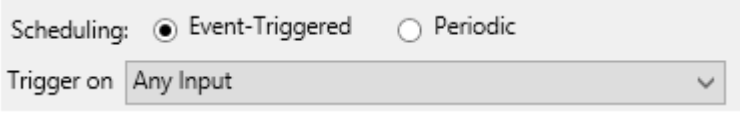
Enter the expression

EventFrame("Duration")

Map the output to the **Duration** attribute

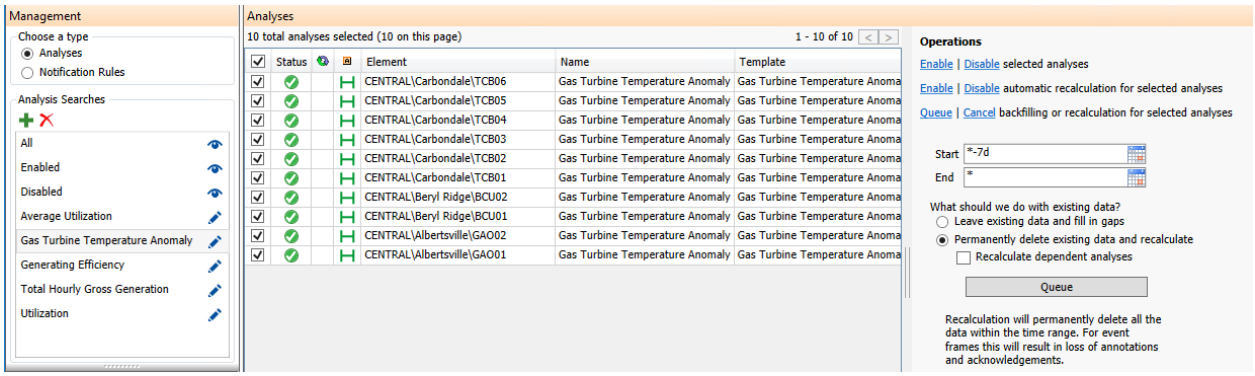


Set the **scheduling** to Event-Triggered and triggering to **Any Input**.



Evaluate and preview the results to confirm there are no syntax errors.

From the Analyses plug-in, backfill event frames for the **past seven days** for **all** Gas Turbine Temperature Anomaly analysis templates.



7.2.2 Exercise - Detect Inactive Units

This solo or group activity is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the activity.

Objective:

Engineering would like to perform a deeper analysis into events over the past week in which the generating units are inactive. Configure the event frame generation to automatically capture new events and detect historical events.

How many inactivity events have been occurring?

Approach:

- Open up the **UNIT** Element Template from the Fleet Generation Database Library.
- Add a new analysis called **Inactive Units** with analysis type of Event Frame Generation.
- Specify the event frame template: **Inactivity**.
- Define the trigger condition to automatically detect inactive events.
- Add an Output Expression using the EventFrame(“Duration”) function.
- Verify data.
- Backfill for the past seven days.

7.3 Discussion

This is a discussion designed to maximize learning in a specific topic area. Your instructor will have questions, and will prompt for communication within the class. This is an open-ended section and the result depends on your needs.

Objective: Brainstorm some real world uses for event frames at your own company. Event frames can be used to capture duration and summary information for events such as process excursions or downtime, but how would this be implemented in your workplace?

Approach

- What kinds of events are of interest in your own process?
- Can you think of reliable trigger conditions?
- Do you have all the required data to identify these events?

Estimated Completion time 10 minutes.

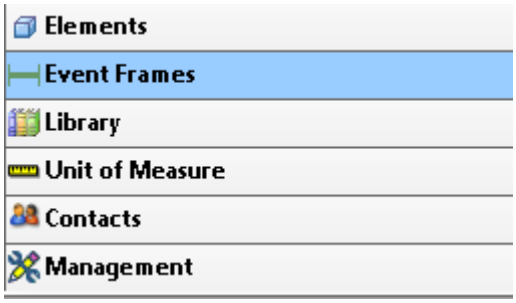
8 Analyzing Events

8.1 Objectives

PI Event Frames are stored in PI AF databases. These event frames can be viewed, filtered, analyzed using PI tools such as PI System Explorer, PI Vision, and PI DataLink.

8.2 PI Event Frames in PI System Explorer

The easiest way to view PI Event Frames is through PI System Explorer. From the Event Frames Pane, you can perform searches against all the event frames within an AF database. You can filter based on specific referenced elements, specific time ranges, and much more.



From the properties of an Event Frame Search, you can specify the following search parameters for the time of the event frame, and the properties of the event frame:

Search type: Specify how to perform an event frame search. Find all event frames that are entirely between a start and end time? Starting or ending between a start and end time?

Search start: Specify the start time for event frame search.

Search end: Specify the end time for event frame search.

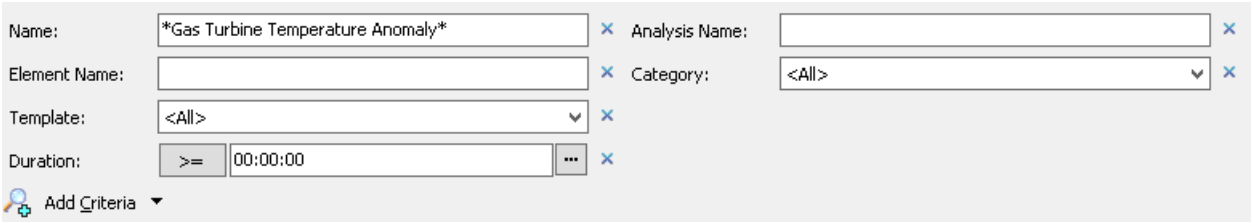
Include descendants: Search for all child event frames in addition to parent event frames.

Event Frame Name: Filter based on the name of an event frame. Can use wildcards.

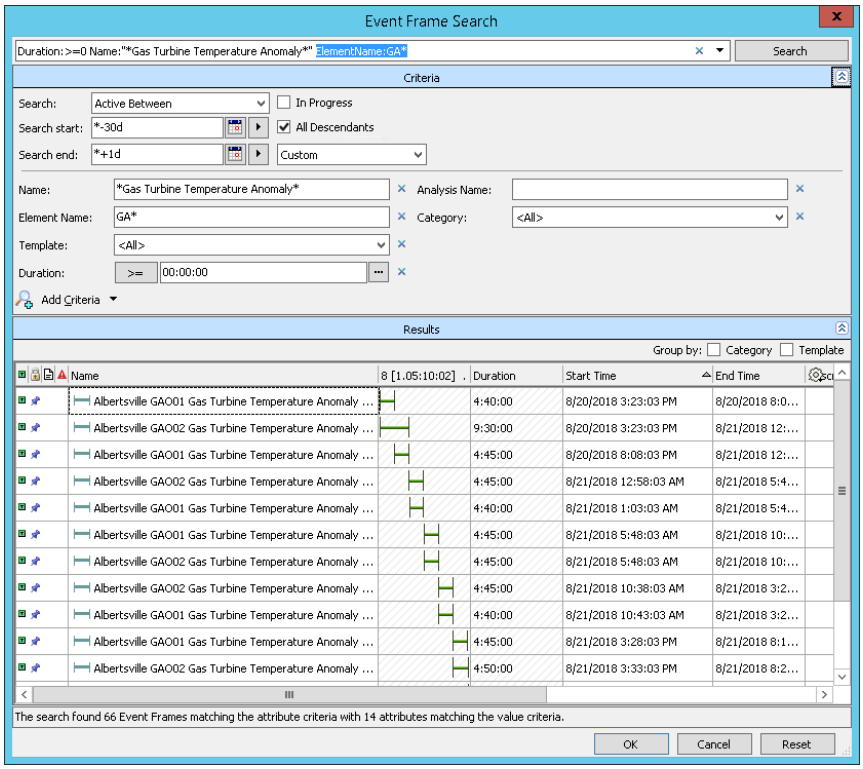
Element Name: Filter based on the name of the referenced element. Can use wildcards.

Template: Filter based on the event frame type.

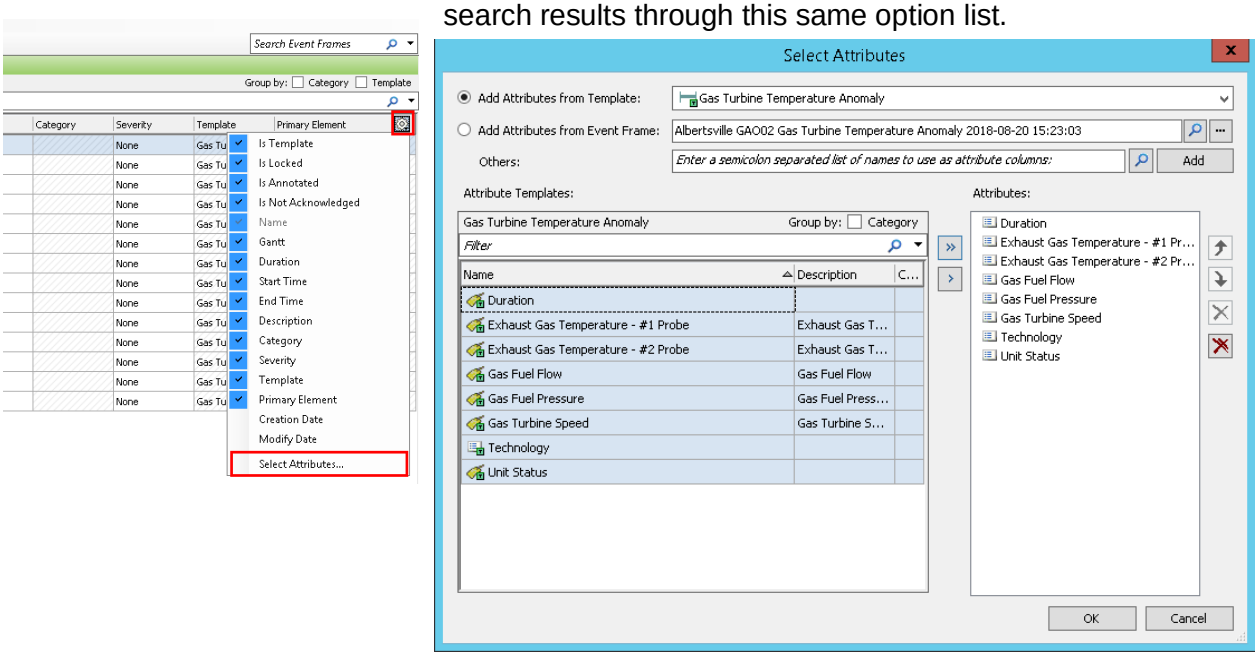
Additional Criteria: Ability to filter based on duration, attribute value, event frame search root, and specify how many results to return.



The resulting search query is combined into a string within the search field. This allows for direct manipulation of the data fields without using the menu options.



The default search results bring back fields detailing the duration, start time, end time, description, category, template, and a Gantt chart. Any of these fields can be hidden by using the settings cog on the top right corner of the search results. Additionally, values from the event frame attributes can be pulled back into the search results through this same option list.



8.2.1 Directed Activity – Search for Inactive Events for GAO01

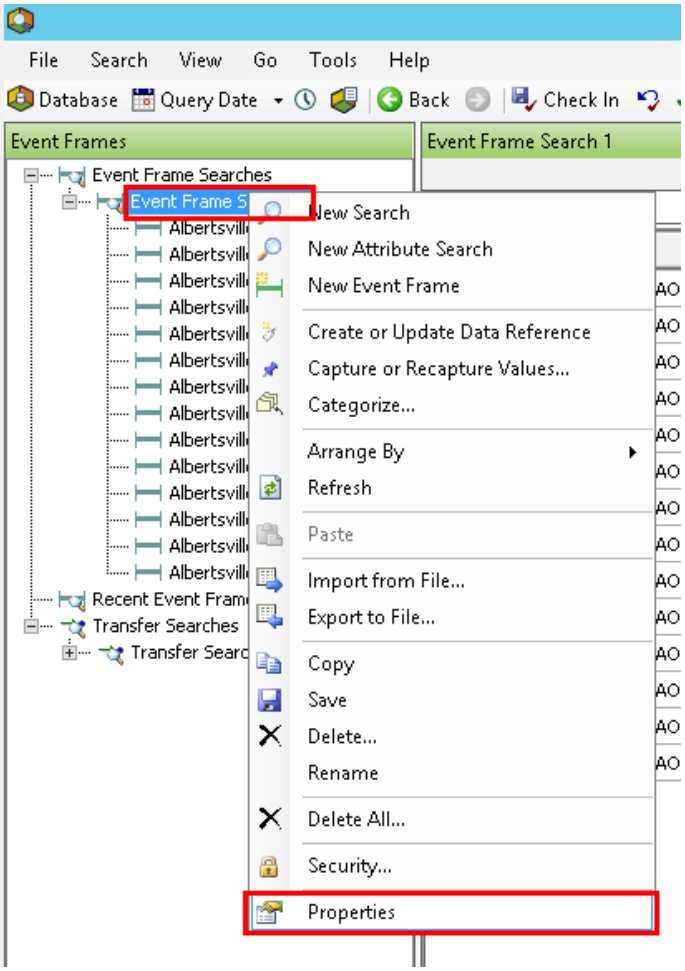
In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

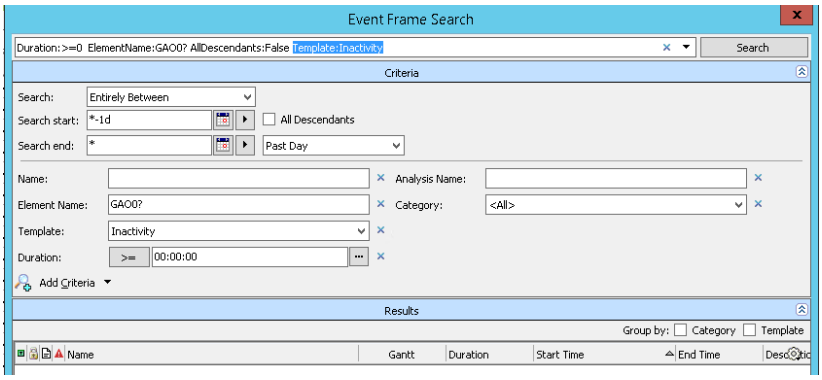
Find all Inactive events for the unit GAO01 and GAO02 over the past 24 hours. Examine the technologies that are involved in these inactive events.

Approach:

Click on the event frame plug-in. Right-click on **Event Frame Search 1** and select **Properties**.



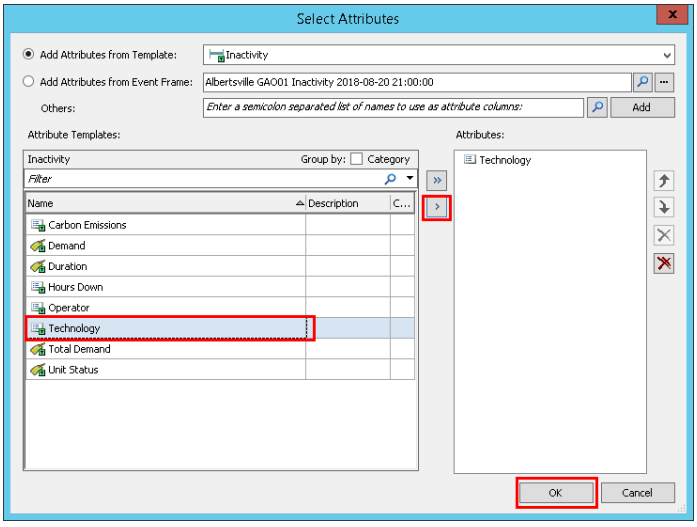
From the Event Frame Search screen, specify the search start to "***-1d**", end to "*****", and uncheck the "All Descendants" checkbox. For the Element Name textbox, specify **GAO0?** and set the Template to **Inactivity**.



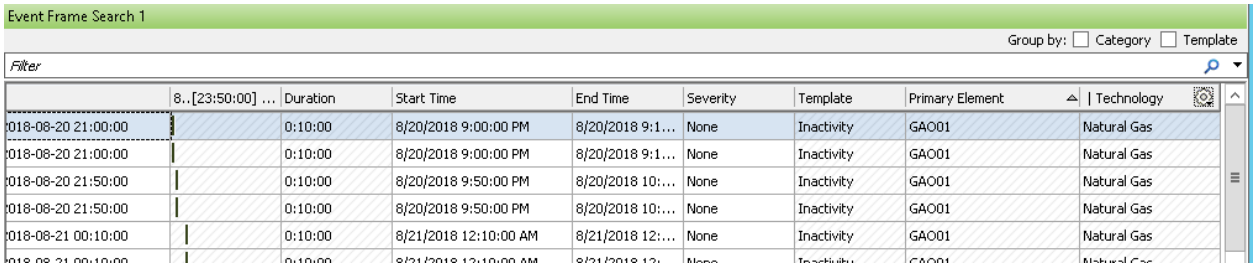
The search will return several inactive event frames. Select **all** of them and click on **OK**.

Click on the gear icon to the right of the fields, and **remove the description and category fields**. Then click on "**Select Attributes.**"

Select the **Technology** attribute from the Select Attributes wizard.



Examine the Technology that is leading to the downtime for these Inactive Units.



8.2.2 Exercise – Search for Recent Temperature Anomalies

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Find all temperature anomaly events for the gas turbines over the past 48 hours that last for more than one hour. Add columns for Fuel Gas Pressure and for each of the two gas temperature sensors.

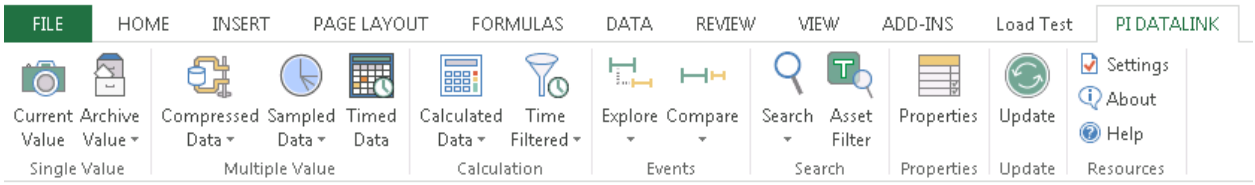
Which unit has the highest starting Gas Fuel Pressure during a temperature anomaly, and when was it?

Approach:

Perform an event frame search and format results for the desired attributes.

8.3 PI Event Frames in PI DataLink

PI DataLink allows you to retrieve current, historical, and calculated data back into Microsoft Excel. In addition to these capabilities, PI DataLink also allows for the retrieval of event frames back into Excel for further analysis.

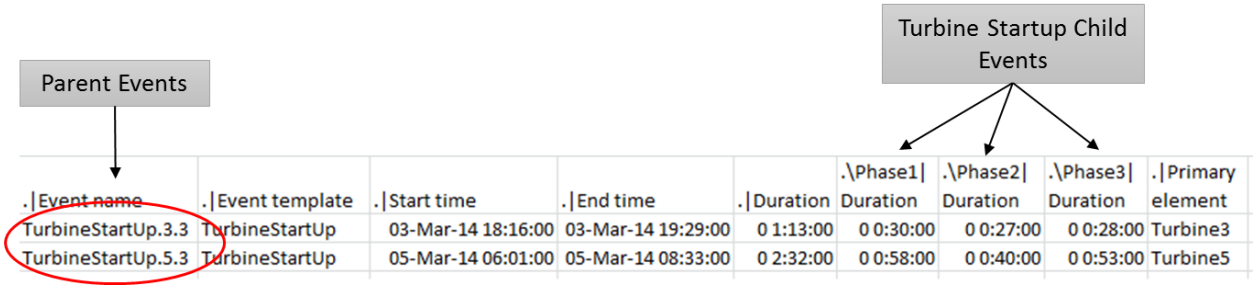


There are two retrieval methods for Event Frames inside of PI DataLink:

Explore: Find Event Frames that meet the specified criteria and display them in a hierarchical format, which is useful to analyze events sharing the same EF template.

Event name	Start time	End time	Primary element	ReasonCode	ShutDownType
BoilerShutDown.5.20130403.1	03-Apr-13 18:00:00	03-Apr-13 19:00:00	Boiler5	P	Planned
BoilerShutDown.5.20130404.1	04-Apr-13 18:00:00	04-Apr-13 19:00:00	Boiler5	P	Planned
BoilerShutDown.5.20130404.2	04-Apr-13 22:04:00	04-Apr-13 23:31:00	Boiler5	E	Emergency
BoilerShutDown.5.20130405.1	05-Apr-13 18:00:00	05-Apr-13 19:00:00	Boiler5	P	Planned

Compare: Find Event Frames that meet the specified criteria and compare their attributes in a flat format. This allows a flat list of events with attributes relating to child events all within a single row.



For either the Compare or Explore Events, you can specify parameters to search for specific event frames. You can specify the following:

Database: AF Database to search against.

Event Name: Search pattern to search for specifically named event frames.

Search Start: Search for all event frames that occurred after this time.

Search End: Search for all event frames that occurred before this time.

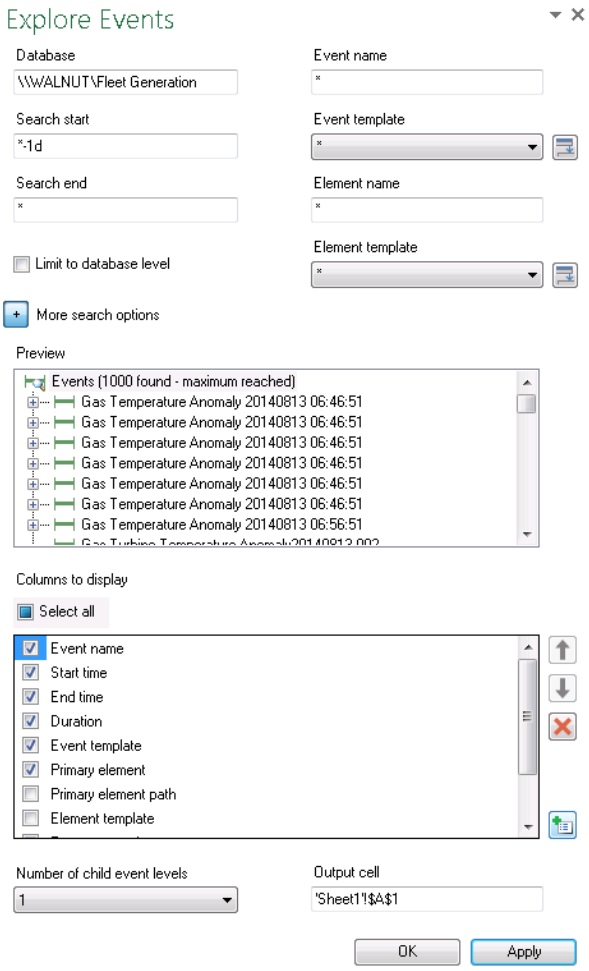
Event Template: Search for specific types of events.

Element Template: Search based off of the type of referenced element.

Element Name: Search pattern for the name of the event frame.

More search options: Search based on attribute values, duration, and category.

Number of child event levels: Only for “Explore Events” and allows for the hierarchical display of events.



Searching for event frames can be based off multiple attributes.

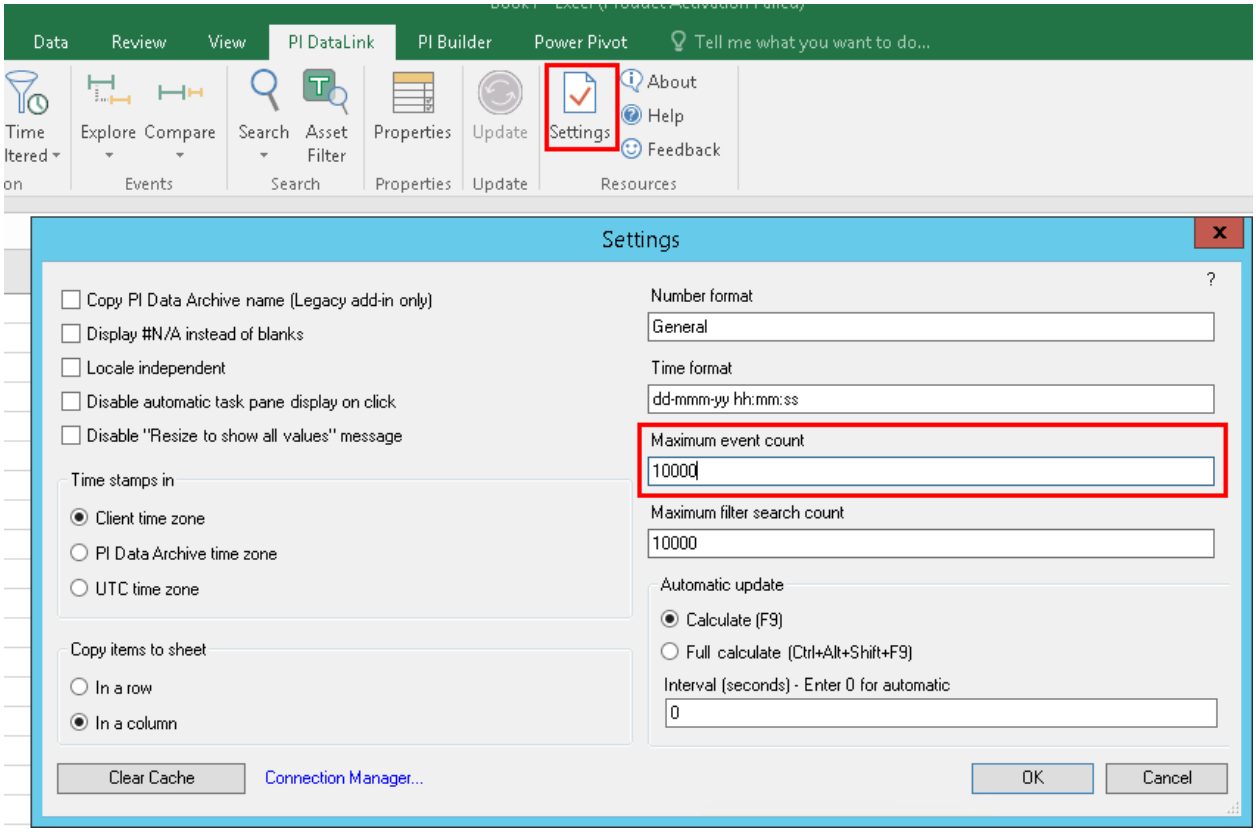
Attribute value filters

Attribute	Operator	Value
Technology	=	Natural Gas
Gas Fuel Presssure	>=	50

When searching with Explore Events, the results can be displayed hierarchically based on the relationships between child and parent event frames.

Event name	Child 1	Start time	End time	Duration
Gas Temperature Anomaly 20140813 11:16:51		8/13/14 11:16 AM	8/13/14 11:51 AM	0.024306
Gas Temperature Anomaly 20140813 11:16:51	Root Cause	8/13/14 10:46 AM	8/13/14 11:16 AM	0.020833

To return more than 1000 event frames in the search preview, go to **Settings** in the ribbon. **Change the setting to 10,000 Event Frames.**



8.3.1 Directed Activity – How many temperature deviations occurred?

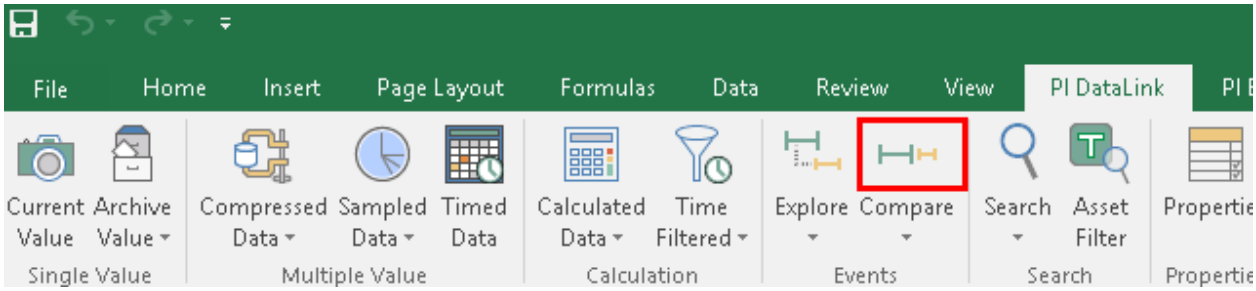
In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

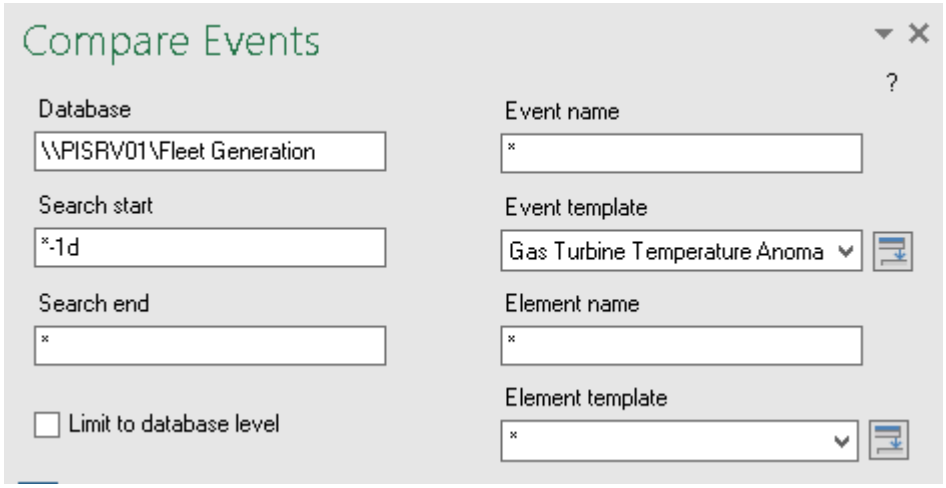
Temperature deviations could potentially mean damaged machinery. Engineering is interested in analyzing the Natural Gas units. Find out how many instances of temperature deviations occurred for gas turbines that lasted for more than 30 minutes.

Approach:

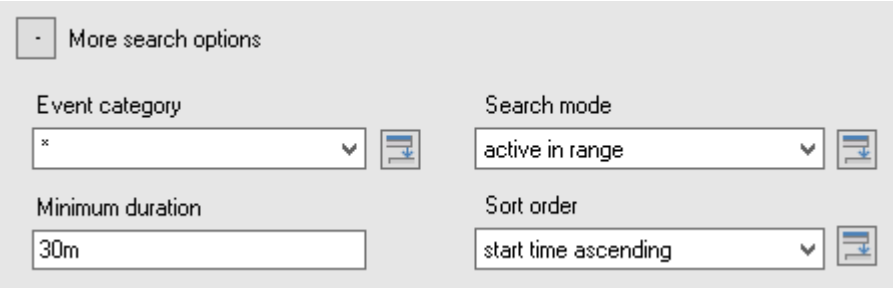
From the PI DataLink tab in of Excel, select cell **A1** and click **Compare** in the ribbon.



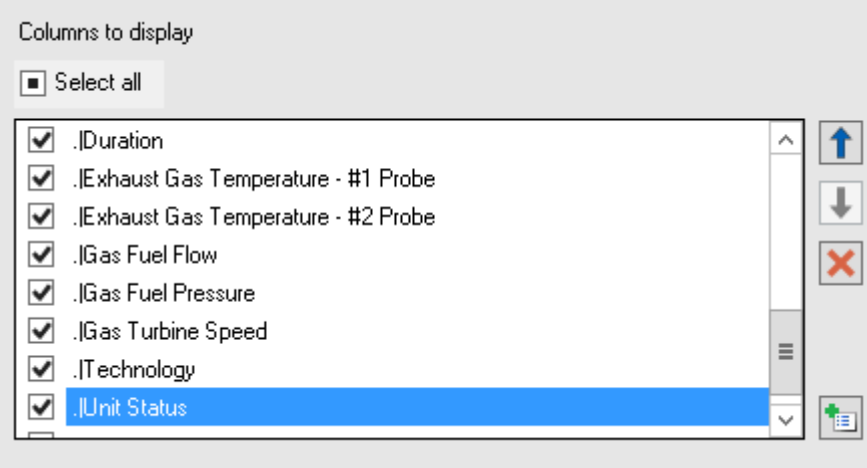
Specify the Database as **\\PISRV01\Fleet Generation**, Event name as "*****", Search start as "***-1d**", and Event template as "**Gas Turbine Temperature Anomaly**."



From **More Search Options**, set the minimum duration to **30 minutes**.



Select the columns that you would like to display:



8.3.2 Exercise – Analyzing Inactivity

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Inactivity events can be costly as the generating units are not generating any power. Analyze with PI DataLink the total number of Inactivity events as well as the total amount of time the units were in an Inactive state for the 24 hours.

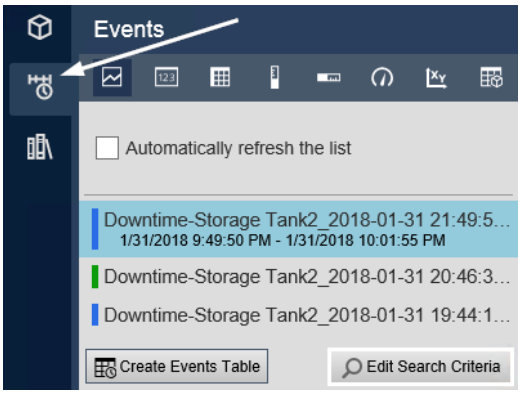
Which generating unit had the most downtime events? Which generating unit had the largest total downtime?

Approach:

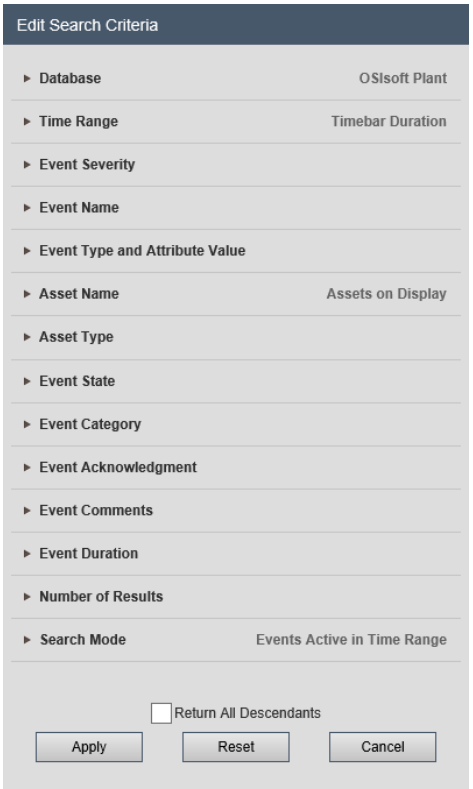
Use PI DataLink to search for PI Event Frames and specify which attributes to return. Use Excel to aggregate the events. It's probably easiest to use a Pivot Table.

8.4 PI Event Frames in PI Vision

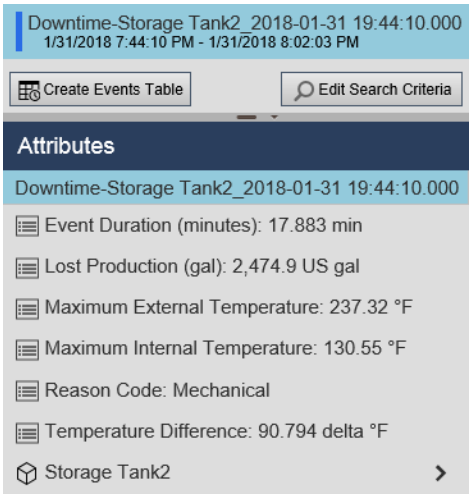
PI Vision enables you to view and analyze your PI data during the time range of a particular event. For example, you may want to examine the performance of an asset during an operator shift or compare the data for several assets during a downtime period.



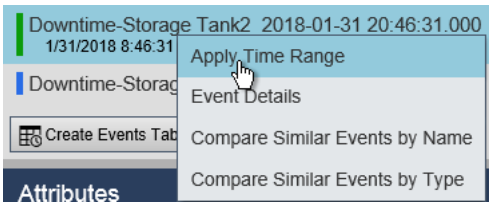
To view events, open the Events tab on the left side. Here you will find events related to your process, the color to the left of each event indicates its severity. By default, the time range of the display and the context of the symbols in the display determine what events are shown in the Events list in PI Vision. To discover additional events, modify the time range or choose *Edit Search Criteria.* When you edit the search criteria, there are a number of filtering options to find the Event Frames you are looking for.



You can select an event to find its Data Items (event attributes) and its start and end time.



By right clicking on an event, you can choose *Apply Time Range* apply the event's time range to the display.



8.4.1 Directed Activity – Inactivity Events in PI Vision

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

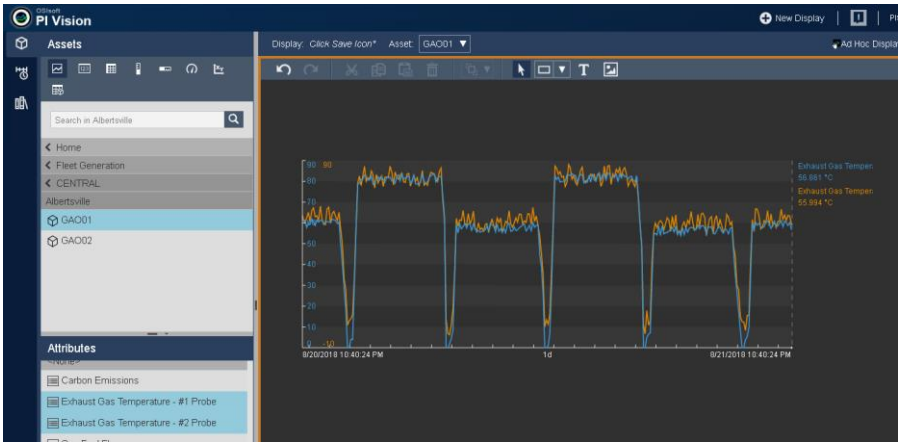
Objective:

Visualize Inactivity Events using PI Vision.

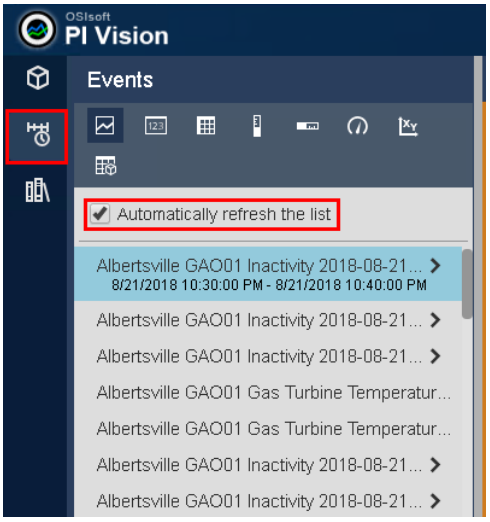
Approach:

Create a new PI Vision display. Drill down to asset GAO01 in the Fleet Generation database

Trend the **Exhaust Gas Temperature Probes** for the past **24 hours**.

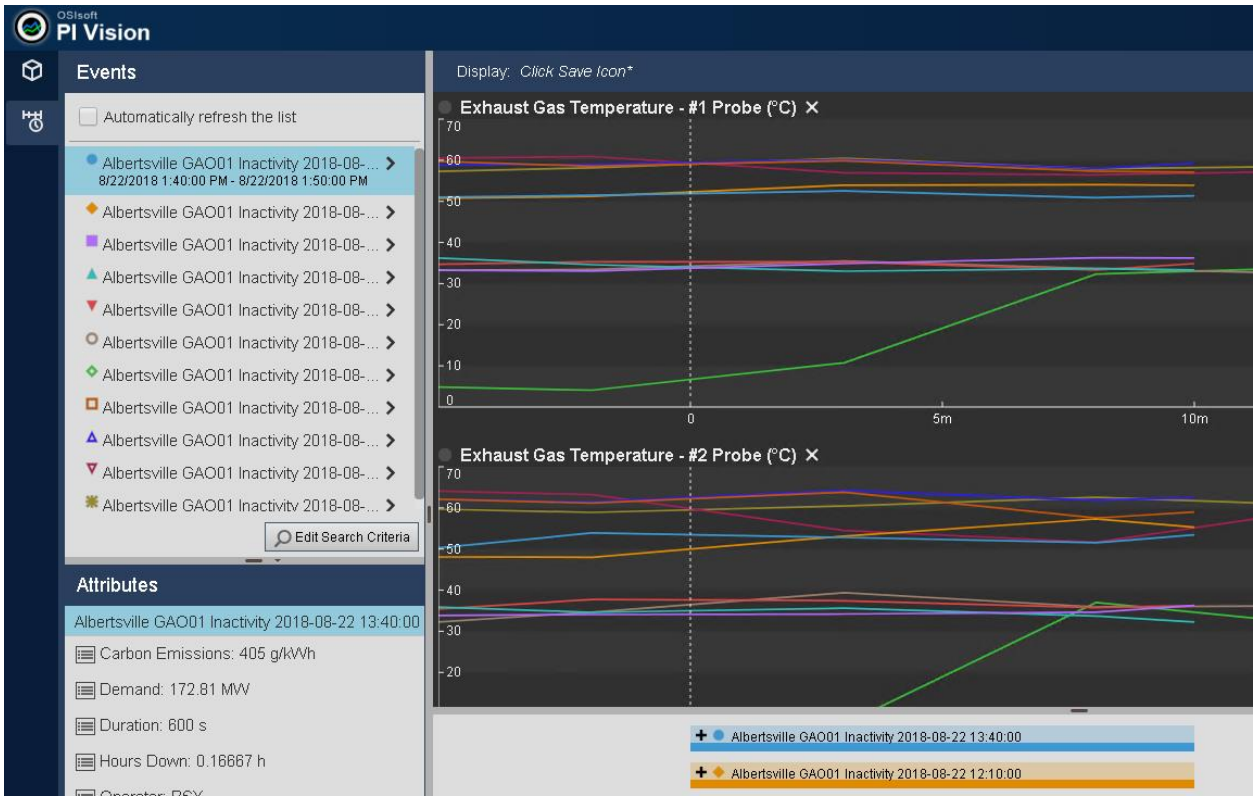


Click on Events in the top left and check “Automatically refresh the list”. By default, this will load Event Frames for Assets on the display (in this case Turbine GAO01).

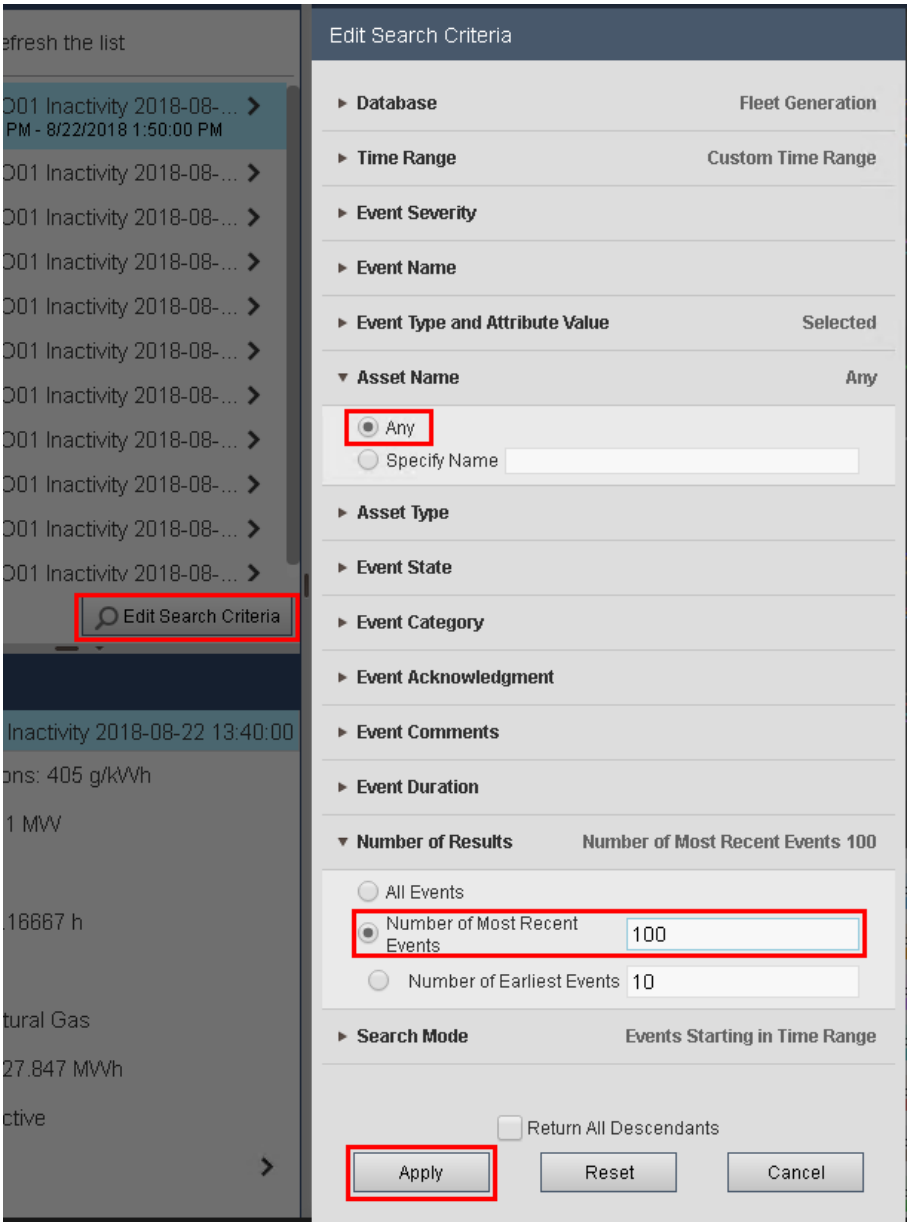


Right-click on one of the **Inactivity** Events and select **Apply Time Range.** The time range will be applied to the temperature trends.

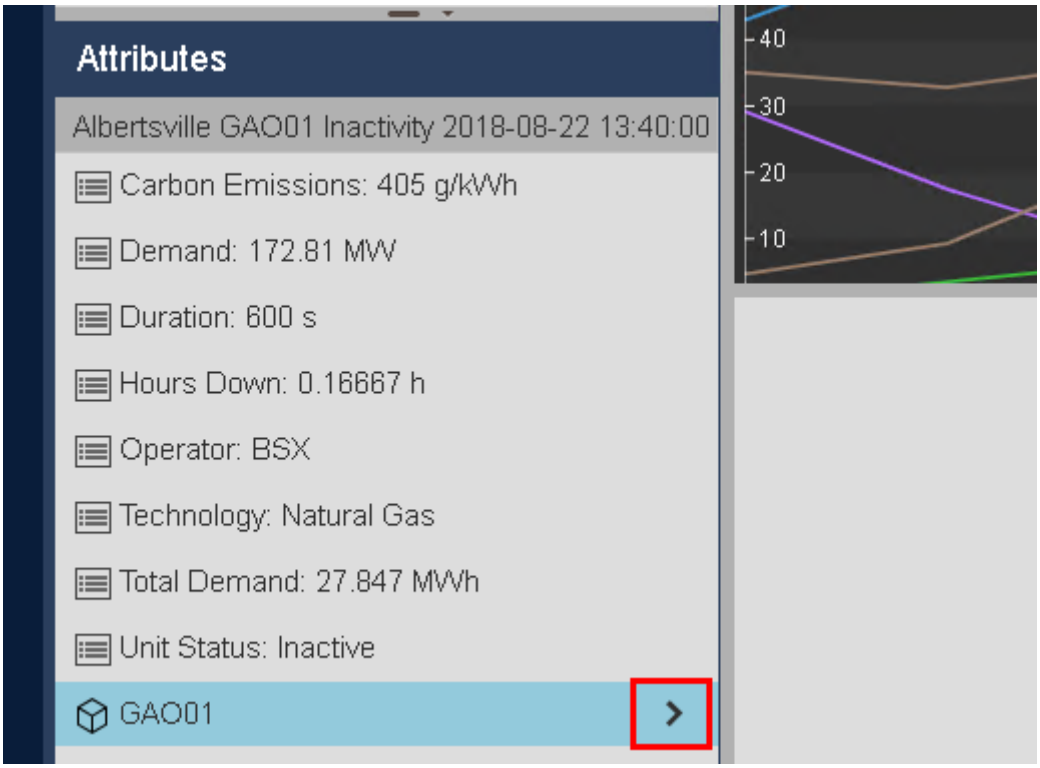
Right-click on one of the Events and select **Compare Similar Events by Type.** Trends of the Event Frame trigger attributes for the selected Event Frame and 10 recent event frames will be shown.



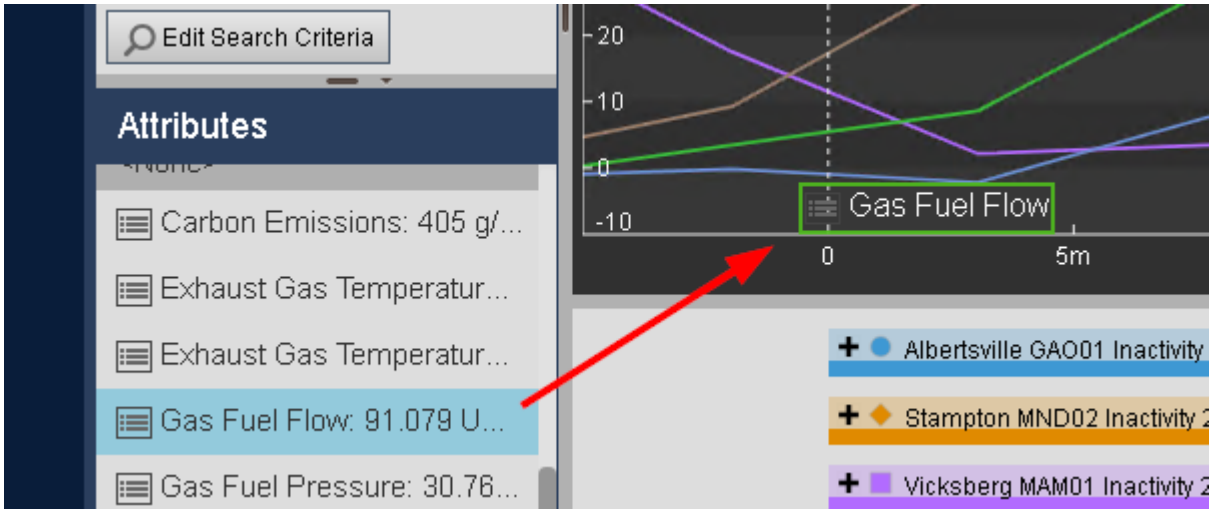
Edit Search Criteria to compare 100 Inactivity Events for All Turbines:



Other attributes from the Event Frames can be trended, but instead we will trend attributes that are not included in the Event Frame but are included in the Asset. In the Attributes Pane, drill into the turbine:



Then drag/drop the Fuel Gas Flow onto the trend area to add new trends



Use the scroll wheel on the right to scroll down and see the new trends

8.5 Discussion

This is a discussion designed to maximize learning in a specific topic area. Your instructor will have questions, and will prompt for communication within the class. This is an open ended section and the result depends on your needs.

Objective: Event frames can be difficult to grasp at first. Let's repeat the discussion from the previous chapter now that you've seen some examples. Brainstorm some real world uses for event frames at your own company. Event frames can be used to capture duration and summary information for events such as process excursions or downtime, but how would this be implemented in your workplace?

Approach

- What kinds of events are of interest in your own process?
- Can you think of reliable trigger conditions?
- Do you have all the required data to identify these events?

Estimated Completion time 10 minutes.

9 PI Integrator for BA (Event and Streaming View). Accessing PI Data programmatically.

Another important functionality that PI Integrator for BA has is event view and streaming view. Streaming view is available in advanced edition of the integrator. PI System Access license also includes PI SQL Client and the Real Time Query Provider (RTQP) Engine. PI SQL Client is generally more flexible than PI Integrator for BA. The main drawback is the difficulty of writing SQL queries and reduced throughput. For example, PI SQL Client would take longer to directly import several million rows during a report refresh. These two options have their GUI and are comfortable to work with. PI System Access also includes programmatic access to PI System Data through developer technologies, such as PI AF SDK, PI WEB API, that don't have any user interface and requires good programming knowledge. This allows better integration with different types of 3rd party applications and makes PI System data available for building advanced analytics and applying different data science methods.

In this part we will explore different tools for interacting with PI data as a data scientist. We will learn how to explore PI data within the system before exporting it to develop a model. At the end of this part, you will have gone through a small data science problem from data exploration to deployment using data stored in the PI System.

More specifically, we are going to do the following:

- Part 1 – Understanding asset hierarchy and raw data
  - Explore asset hierarchy
  - Understand time behavior
  - Aggregate time features using event frames

- Part 2 – Data exploration using PI DataLink and Python
  - Explore available data in PI
  - Understand the project scope through PI Vision displays
  - Explore generated Event Frames

- Part 3 – Develop a predictive model
  - Export data using the PI Integrator for Business Analytics
  - Train and evaluate model

- Part 4 – Operationalize the predictive model
  - Stream data to Kafka using the PI Integrator for Business Analytics
  - Ingest data stream using model

For this purpose, we are going to work with another AF database "Lab Building Data", which is not related to any specific production area. It is about cooling process with air conditioning, which is known to all of us. Based on this simple data you will be able to understand much easier all the approaches and data science methods, that are going to be applied further.

Business Problem

We need to reduce the energy that is spent for cooling by optimizing daily startup of individual cooling units in a commercial building.

Every day, the VAVCO cooling units adapt to changing temperature, relative humidity, thermostat setpoints, building occupancy level, and other factors. The control system adjusts several factors to provide the necessary cooling to the rooms to ensure tenant comfort.

A pre-established occupancy schedule requires that rooms should be at a comfortable temperature between **7 AM and 7 PM** when employees are in the building. During those hours, the individual units keep the temperature at the desired setpoint. The units follow this schedule to meet that requirement:

- Turn-on at some point in the morning, before 7 AM, and bring the room temperature to setpoint by 7 AM
- Keep the room temperature at setpoint during occupied hours
- Shut-down at 7 PM when the building becomes unoccupied

The initial startup should finish as close to 7 AM as possible. Reaching the setpoint too early results in wasted energy cooling an unoccupied room. Conversely, if the setpoint is not reached by 7 AM, employees will not be comfortable in rooms that have not been cooled. The business unit believes that the current startup schedule could be improved by examining the historical data.

Our objective is to predict how long a unit will take to reach the setpoint depending on current conditions to ensure the unit reaches the setpoint as close to 7 AM as possible.

9.1 Understanding asset hierarchy and raw data

9.1.1 Directed Activity – explore asset hierarchy

In this part of the exercise you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

- Better understand the data set used in the following chapters

Approach:

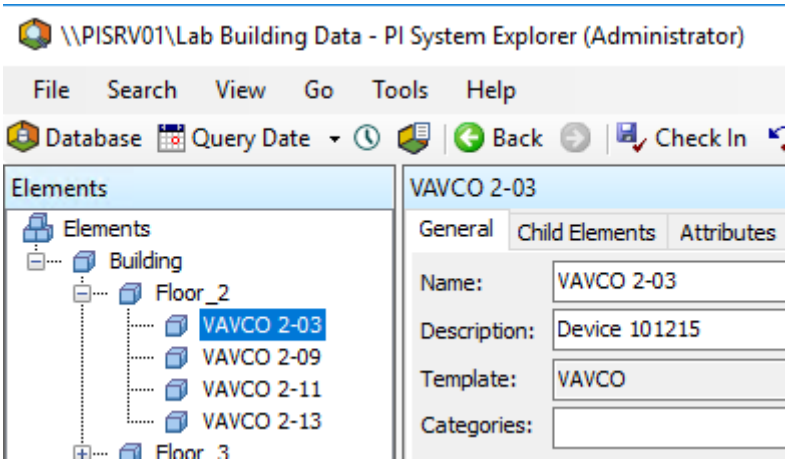
We need to know where to find the data before thinking about developing a model. Unlike the PI administrators and subject matter experts (SMEs) who know their system intimately, we don't know what data is available, where to look to find that data, and how that data is structured.

Questions:

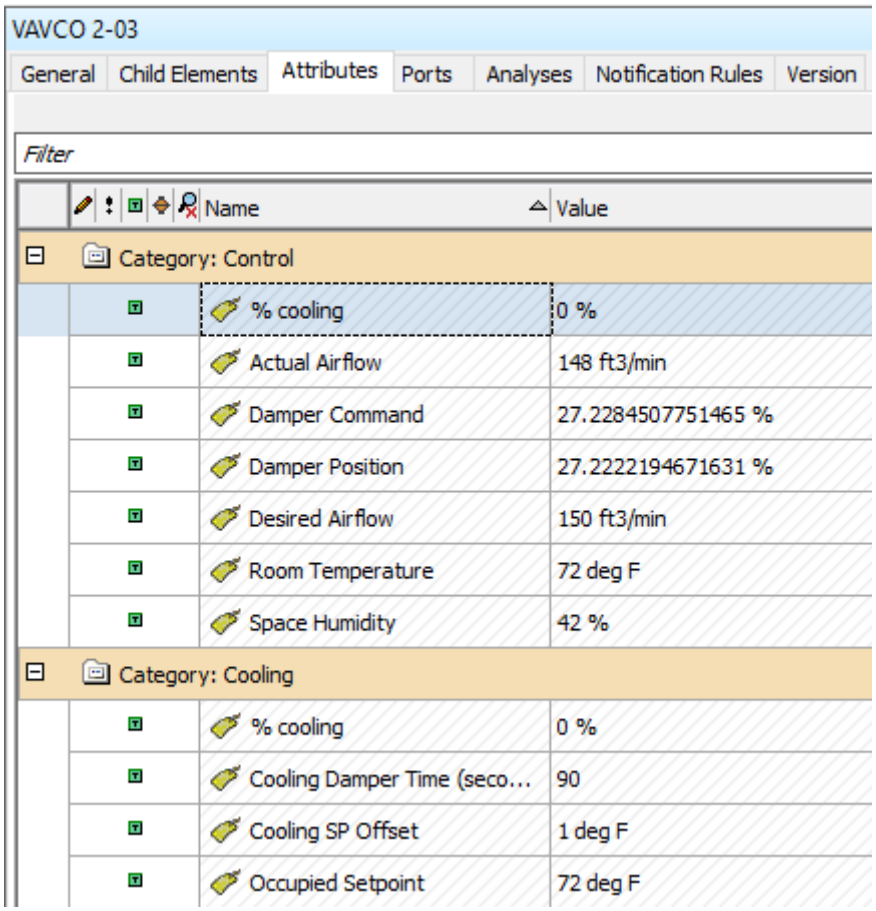
- How many floors are in the building? How many units in each floor?
- What data streams may influence the cooling time?
- What other data might we need to include?
- Does every unit always have the same setpoint?

To answer these questions please follow the steps:

1. Open PI System Explorer on PISRV01 and make sure that the **Lab Building Data** database is selected.
2. Expand the **Building** and **Floor_2** elements, then select the **VAVCO 2-03** element
3. Under the **General** tab, note the template (**VAVCO**)
    a. Templates make data shaping process much easier
    b. Templatized assets typically have the same attribute structure



4. Under the **Attributes** tab, observe these attributes
    a. % cooling
    b. Cooling SP Offset
    c. Occupied Setpoint
    d. Unoccupied Cooling Setpoint
    e. Room Temperature
    f. Space Humidity



We now have a good sense for the data related to the asset in question. However, this may not tell the whole story. Some information included in other assets could be useful in the final model. Clear communication with subject matter experts is critical here – some data may be important even though it's not directly associated with the asset in the AF structure. What other data might be valuable when considering air conditioning?

5. Select the **Weather** element under **Building**

9.1.2 Directed Activity – understand time behavior

In this part of the exercise you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

- Plot dynamics during times of interest (PI Vision)

Approach:

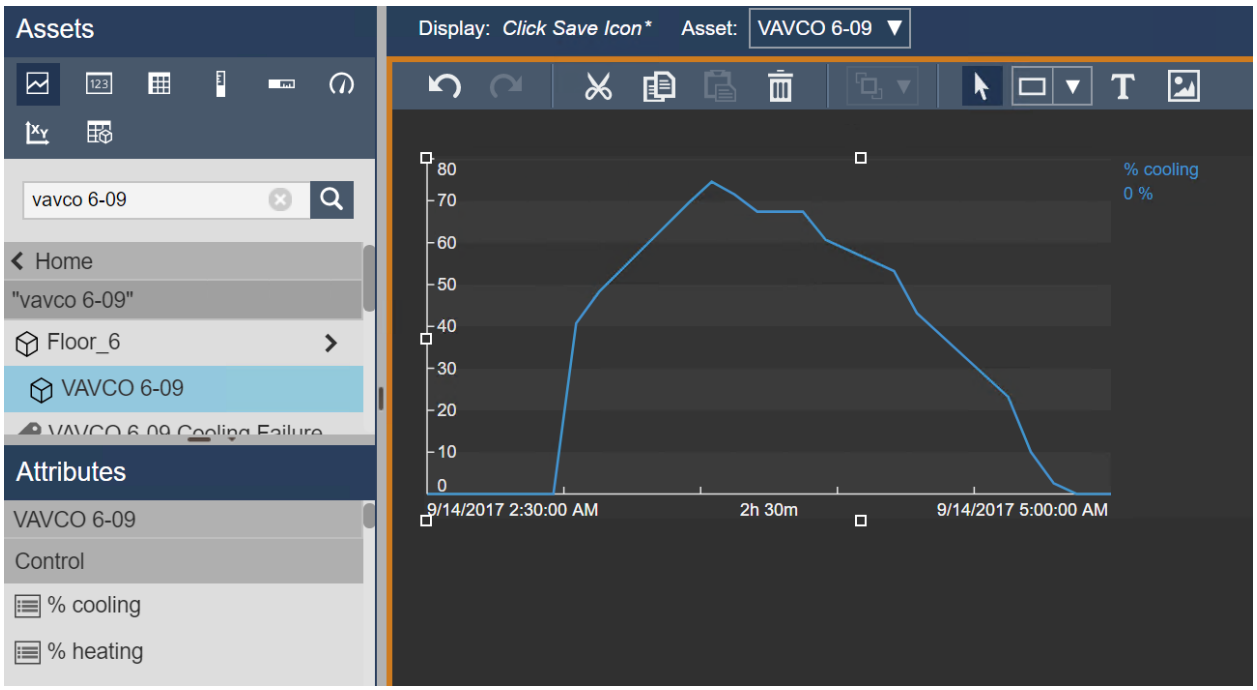
Since we're not experts in facilities management, it's important to start building intuition about how these data behave. This added context can help inform decisions down the line as well as communication with SMEs. Visualizing evolution in time is necessary to understand the time series data stored in the PI System.

Questions:

- How often does the weather data update? Operational data?
- Does the cooling operate continuously or in cycles?
- Does the interior temperature always reach the setpoint after cooling begins?

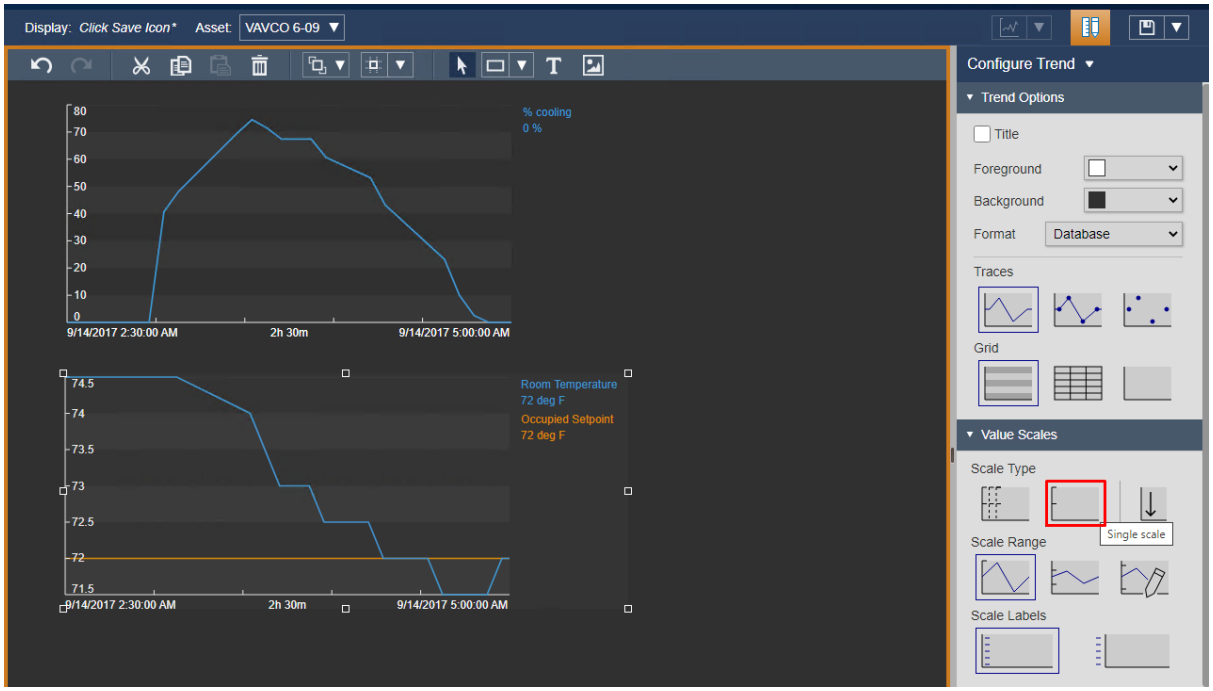
To answer these questions please do the following:

- Open **Google Chrome** and navigate to the bookmark labeled **PI Vision** (https://pisrv01.pischool.int/pivision)
- Click **New Display**
- Type **VAVCO 6-09** in the search bar and select the asset
- Drag **% cooling** from the attribute pane onto the display to create a trend
- Change the start/end times in the bottom-left and bottom-right of the display to the same range as before (**14-Sep-17 02:30:00** to **14-Sep-17 05:00:00**)



- Drag the **Room Temperature** and **Occupied Setpoint** attributes to a new trend just below the **% cooling**

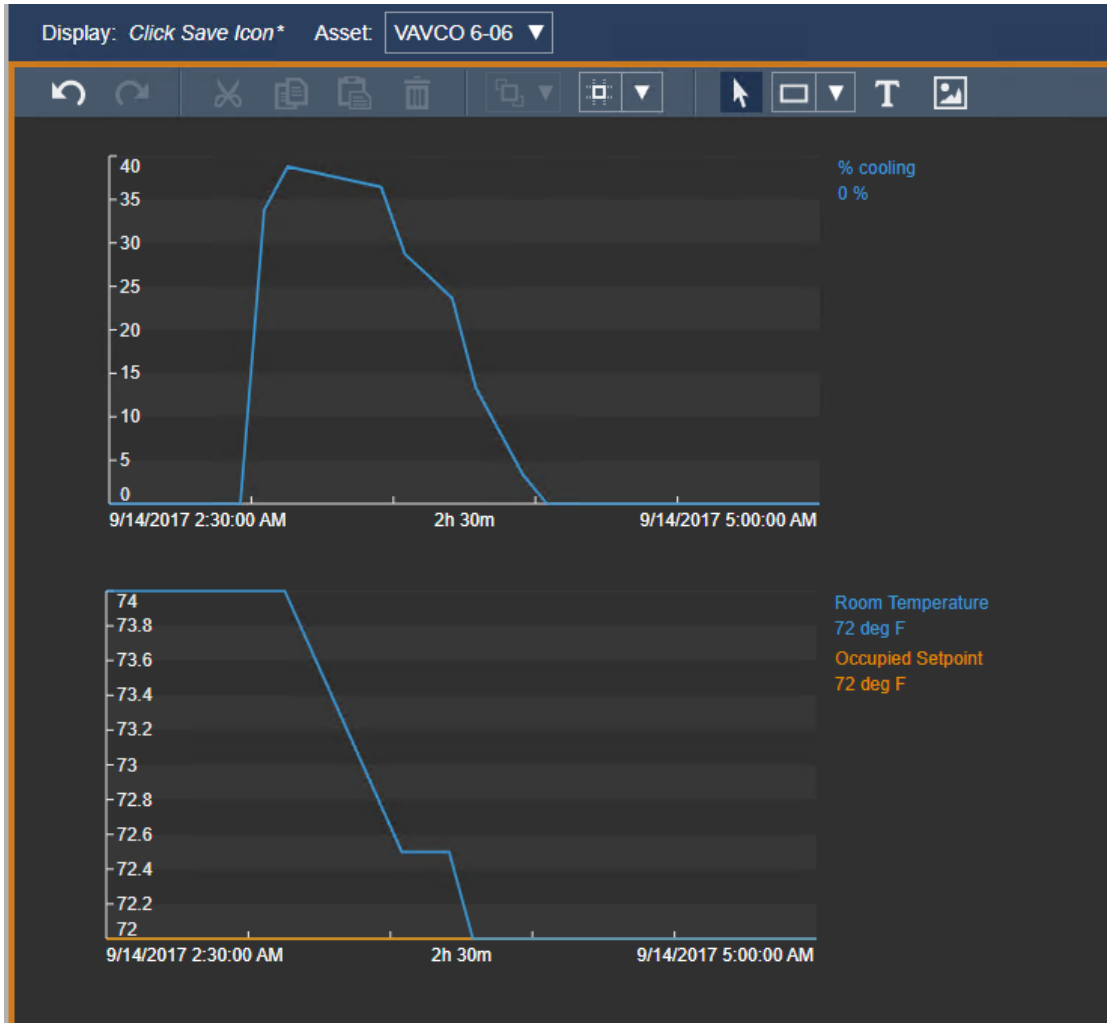
- Right-click on the new trend and select **Configure Trend**
- Change **Multiple Scales** to **Show single scale** under **Value Scales**



We can now see that the unit continues to cool even after the room temperature reaches the setpoint. We will come back to this later. Now, we're looking at a trend for a specific VAVCO unit. Did other units run that day? We can leverage AF templates to quickly change contexts to other units.

- In the **Asset** dropdown, select **VAVCO 6-06**

Notice that the trends update to reflect the new asset. This unit reaches the setpoint more quickly than **VAVCO 6-09.**



- Select **VAVCO 3-16** in the **Asset** dropdown

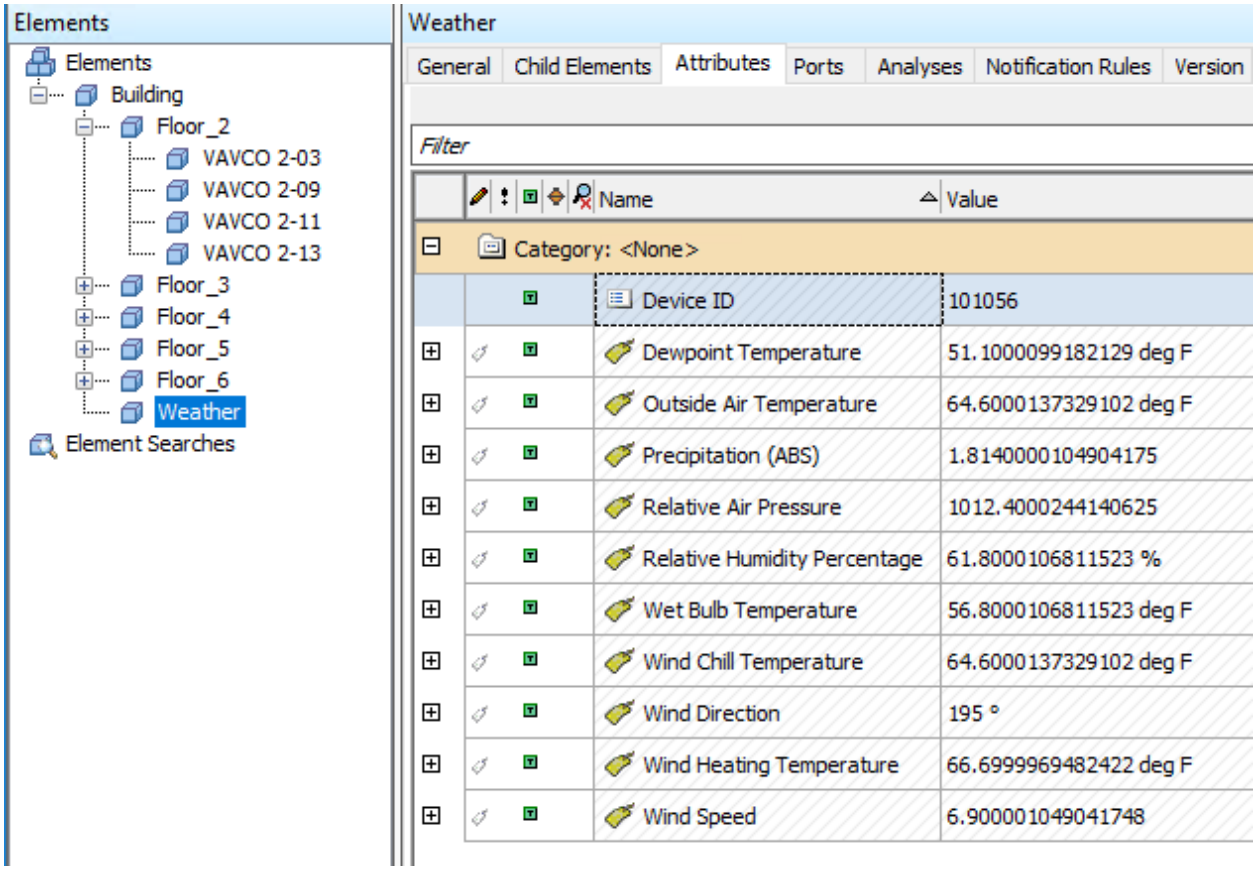
This unit never reaches the setpoint. Maybe someone left the window open…

- Select **VAVCO 6-09** again

Now we have some sense about what happens to some data when the VAVCO units turn on:

1. **% cooling** increases rapidly, peaks, then gradually decreases
2. **% cooling** can remain greater than zero even after the temperature reaches the setpoint
3. **Room Temperature** may reach the setpoint, pass the setpoint, or remain above the setpoint for many hours
4. Different VAVCO units reach the setpoint at different times

Now that we're more comfortable with what happens during the initial cooling, we can move toward preparing the data.



Now we know where to find important data for this system. For the purposes of this exercise, we will be working with attributes on the **Weather** element and elements based on the **VAVCO** template.

9.1.3 Directed Activity – Aggregate time features using event frames

We will now shape the data into something that can be used in a machine learning algorithm. Since we're trying to predict the time it takes to cool a room to setpoint at the beginning of the day, we'll need to format the data into row-column format where each row is a different cooling period and columns represent the outcome as well as possible predictors. Event Frames enable this sort of time-based aggregation.

Objective: Create event frames and visualize in PI Vision

Questions:

- What information should we include in the event frames?

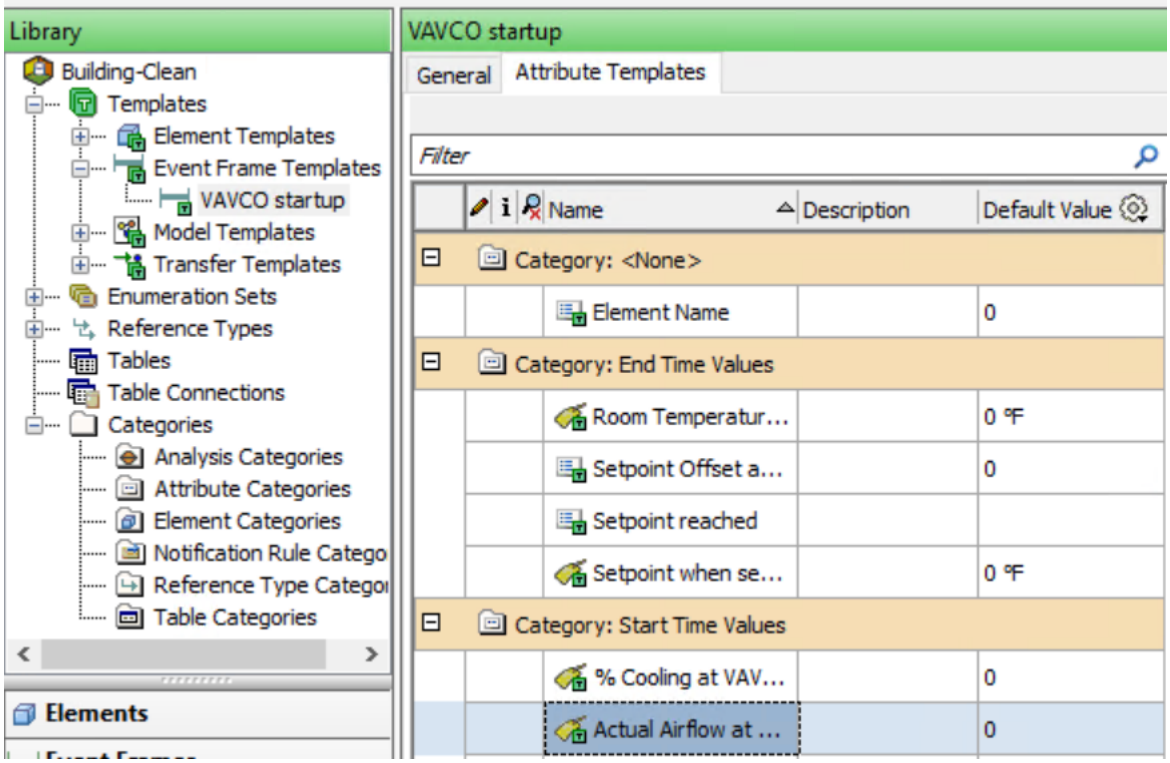
- How should they be triggered?

- Compare the temperature profiles for a few event frames. On a given day, how much do they vary between floors? Units? For a given unit, how much does it vary over few consecutive days?

Solution:

- Open PI System Explorer, click the **Library** tab, expand **Event Frame Templates** and select **VAVCO startup**

The **Attribute Templates** tab shows the data that the event frame will aggregate. Note that it includes both start and end time values as well as the name of the reference element. See below for a description of each of the attributes.

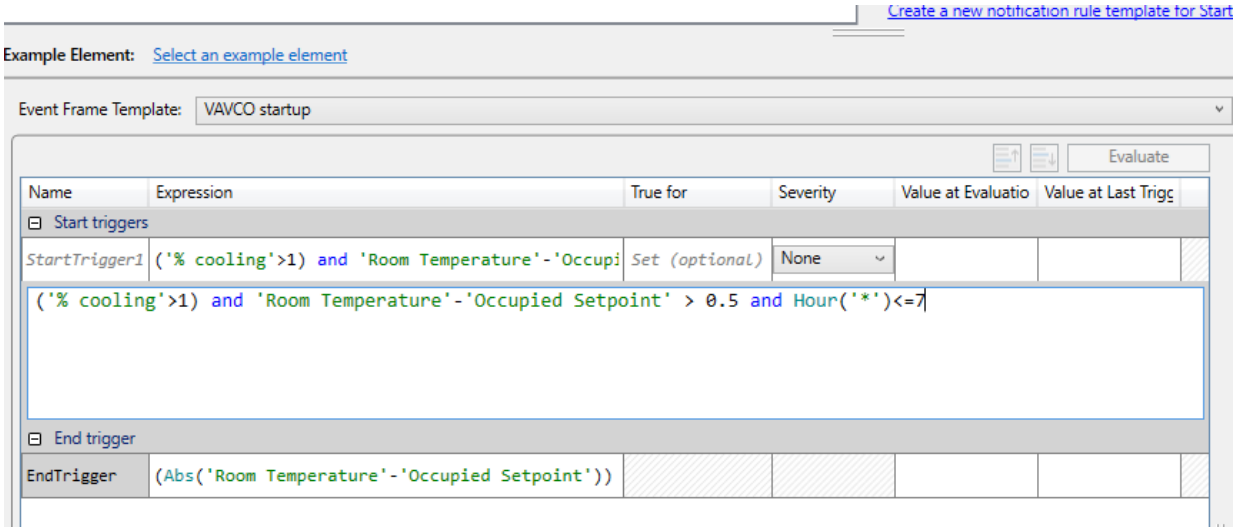


- **% Cooling at VAV start** -> The “cooling rate” at the start of the event
- **Actual Airflow at VAV Start** -> The airflow at the start of the event
- **Damper Position at VAV Start** -> The position of the damper at the start of the event
- **Outside Air Temperature at VAV Start** -> The temperature outside the building at the start of the event which is measured by a weather station placed at the top of the building
- **Outside Relative Humidity at VAV Start** -> The outside Relative Humidity at the start of the event
- **Room Temperature at VAV Start** -> The room temperature at the start of the event

- **Setpoint at VAV Start** -> The occupied setpoint at the start of the event. We have captured this attribute because the setpoint can be changed manually and it's not always constant.
- **Setpoint Offset at start time** -> Calculated attribute to calculate how many degrees off the setpoint we are at the start of the event
- **Space Humidity at VAV Start** -> The humidity of the room at the start of the event
- **Setpoint reached** -> Indicates if the setpoint was reached within the period of the event frame or not. During some very hot days, the setpoint is never reached that day. Since we are only interested in the daily startup events, we close the events at 8 AM even if the setpoint hasn't been reached.

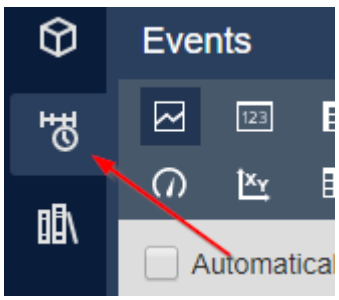
The start and end triggers of the event frames must be defined for the system to know which time periods are of interest. Explore the triggers for the event frames that have already been created:

- Open **PSE** and select **Library**
- Expand **Element Templates** and select **VAVCO**
- Click the **Analysis Templates** tab and select **Startup Event**
- Look at the logic for the **StartTrigger** and **EndTrigger**

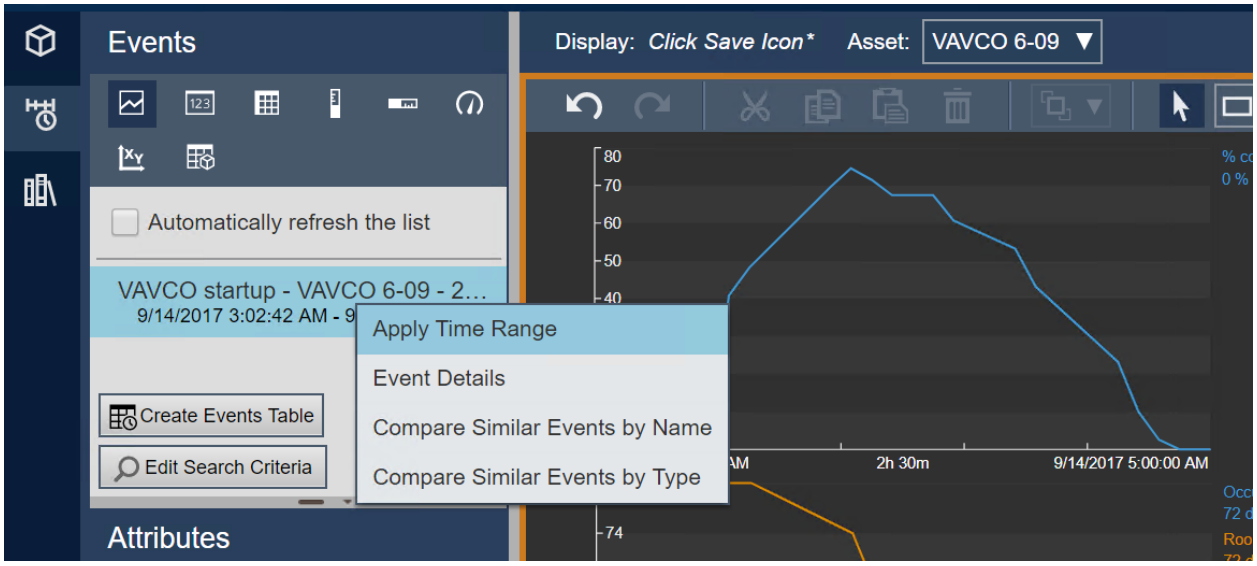


This guarantees that the event frames will only captures the first event of the day and will end when either the room temperature is within 0.5 degrees of the setpoint or it does not reach the setpoint by 8 am. For the purposes of this exercise, we will only consider cases where the setpoint is reached before 8 am. Let's compare event frames directly.

- Click the **Events** tab on the PI Vision display that you just created



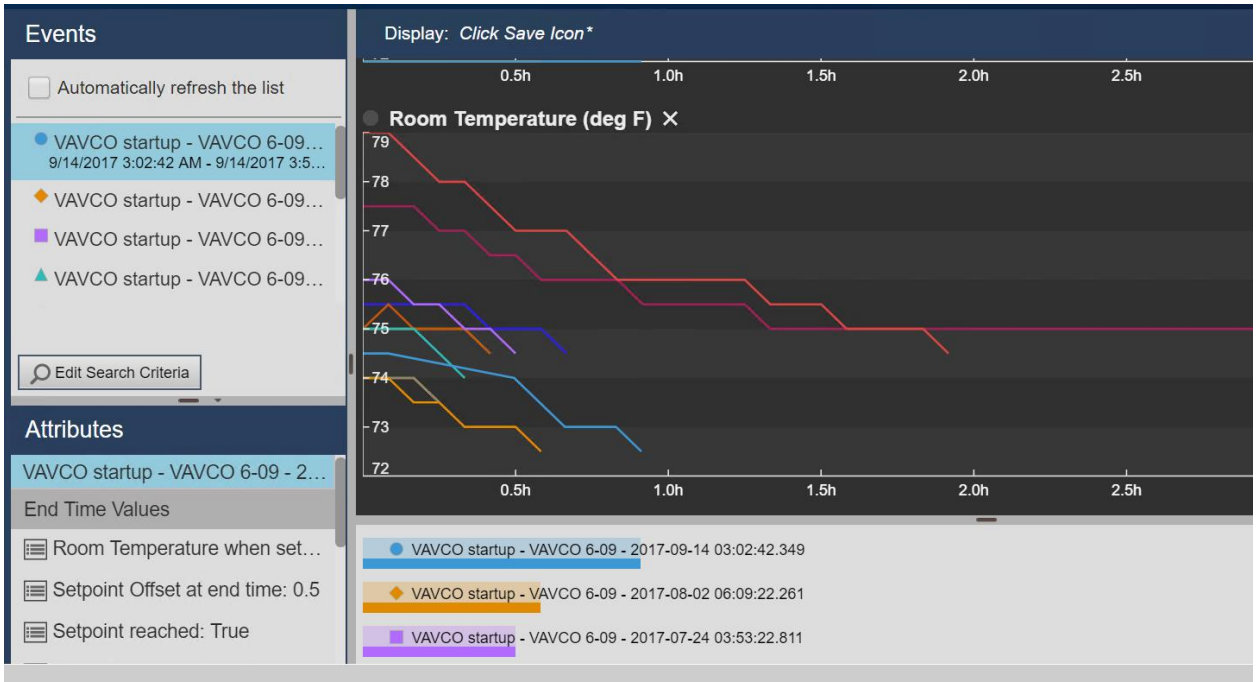
- Right click on the event frame and select **Apply Time Range**



The display now only shows data within the time range of the event frame.

- Right click on the event frame and select **Compare Similar Events by Type**
- Save the display if you have not done so already

A trend of the attribute value over the course of each event frame that matches the search criteria appears. This gives some sense of how the values change with time and how those changes vary from event frame to event frame.



9.2 Data exploration using PI DataLink, Python and R libraries

9.2.1 Directed Activity – explore data using PI DataLink

PI DataLink is an Excel add-in that enables users to easily retrieve data from the PI system into Excel spreadsheets via several different functions. Given Excel's ubiquity and low learning curve, PI DataLink and Excel can be a good tool for quick ad-hoc analysis if you're less comfortable with Python.

Objective: Import event frame information in Excel.

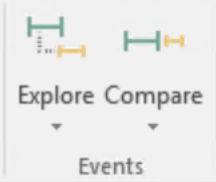
Questions:

- How many event frames are there?
- When is the earliest event frame?
- Are there any NULL values?

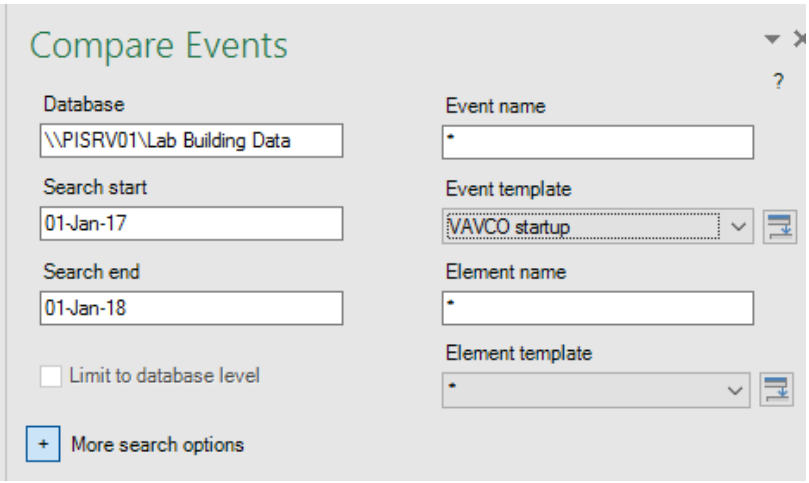
Solution:

Now that relevant pieces of data have been structured by Event Frames, let's put that data together where we can do some analysis.

- Open **Excel on PICLIENT01** and click the **PI DataLink** tab
- Click **Compare** in the **Events** pane



- Set **Search Start = 01-Jan-17** and **Search End = 01-Jan-18**
- Select **Event template = VAVCO Startup**
- Leave other values blank
- Click **Ok**



The event frames will be loaded into **Excel**. You should notice that there are nearly 2000 event frames starting on March 3, 2017 and ending on December 12, 2017. It makes sense that we wouldn't have any more recent event frames as air conditioning isn't needed during the winter months. The Space Humidity has a value of "No Data" for all event frames before April 29. In practice, this can mean that the humidity sensor wasn't running before then, or at least the data was not stored before that date.

9.2.2 Directed Activity – Load PI Data into data frames with Python and R

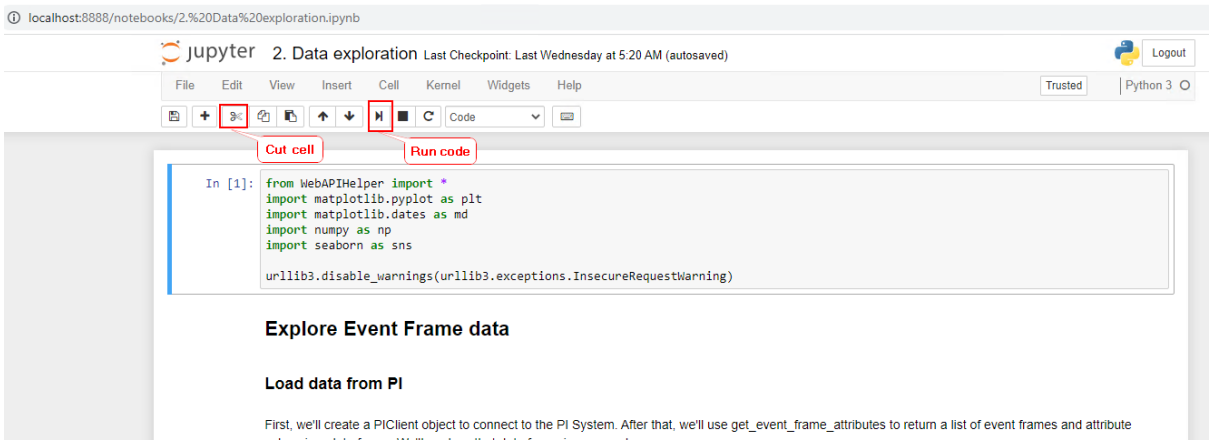
Python and R have become very popular for data science. The AVEVA (PI) Technology Enablement Team developed some libraries for data scientists to interact with PI data using these familiar tools. They leverage the PI Web API, a REST endpoint for the PI System, to access and shape data.

For the purposes of this exercise, some helper functions have been added on top of these libraries to simplify the syntax.

If you're not comfortable with Python, feel free to analyze the data in Excel. The worked examples use Jupyter notebooks, but RStudio is also available if you're more familiar with R.

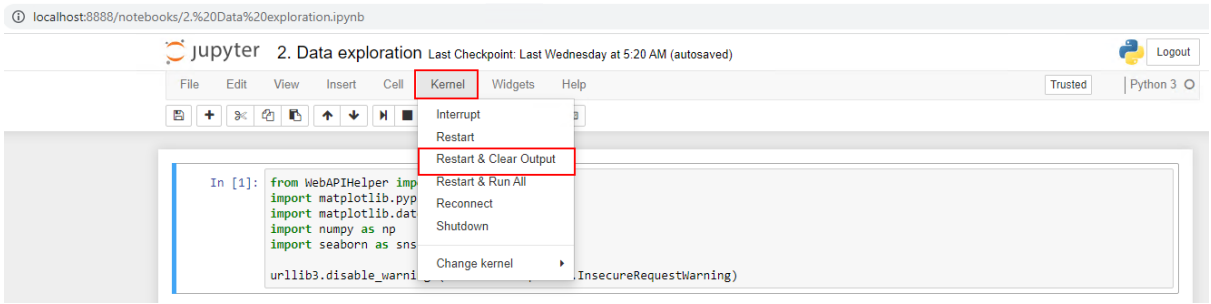
We are going to use Jupyter notebook in further exercises. Let's take a look on how to work with Jupyter.

Jupyter notebook is an open-source, interactive web application that allows users to create and share documents that contain interactive calculations, code, images, etc. Users can combine data, code, and visualizations into a single notebook, and create interactive "stories" that they can edit and share. Notebooks are documents which contain both computer code (such as Python) and other text elements such as paragraph, markdown, figures, links, etc.



We are not going to write any code by ourselves, required code is already available for us. To run the code cell we are going to use hot keys Shift + Enter or button, which is marked in red on the image above.

To clear the variable outputs and restart the code please select Kernel tab:



Objective: Import the PI data into a Jupyter notebook

Questions:

- Do any of the columns need processing before analyzing the data?
- Which features should we include in our model?

Solution:

We've installed the Anaconda Python distribution for the purposes of this exercise.

- Launch **Jupyter Notebook** from the desktop on PISRV01
- Open the notebook called **2. Data exploration**
- Create a PIClient object to connect to the PI Server
- Here we just import all the Python libraries to work with

```
In [*]: from WebAPIHelper import *
        import matplotlib.pyplot as plt
        import matplotlib.dates as md
        import numpy as np
        import seaborn as sns

        urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

- Update the **password** variable in the 2nd code block
- Here we set PI Data Archive name, AF database and specify PI Web API path to be able to read the data from the PI Server

```
In [2]: webapiurl = 'https://pisrv01.pischool.int/piwebapi'
        dataarchive = 'PISRV01'
        afserver = 'PISRV01'
        afdatabase = 'Lab Building Data'
        password = 'securepassword'

        client = PIClient(webapiurl,dataarchive,afserver,afdatabase,password)
```

- Load event frame data, similarly, as what we did in PI DataLink

Load Event Frame data

```
In [9]: df = client.get_event_frame_attributes("Name:=*")
```

- Examine the first few rows with df.head()

```
In [6]: df.head()
```

Out[6]:

	EventFrame	EFStartTime	EFEndTime	Space Humidity at VAV Start	Setpoint when setpoint reached	Setpoint reached	Setpoint Offset at start time	Setpoint Offset at end time	Setpoint at VAV Start	Room Temperature when setpoint reached	Room Temperature at VAV Start	Outside Relative Humidity at VAV Start
0	VAVCO startup - VAVCO 6-11 - 2017-03-03 07:05:...	2017-03-03T15:05:12.0022735Z	2017-03-03T15:25:14.0071563Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	77.976303
1	VAVCO startup - VAVCO 4-03 - 2017-03-22 07:04:...	2017-03-22T14:04:07.6927185Z	2017-03-22T18:49:11.8333892Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'False', 'Value': 0, 'IsSystem': False}	1.5	1.0	72.0	73.0	73.5	98.962120
2	VAVCO startup - VAVCO 3-10 - 2017-03-22 07:04:...	2017-03-22T14:04:07.7084503Z	2017-03-22T14:19:07.7552185Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120
3	VAVCO startup - VAVCO 6-11 - 2017-03-22 07:04:...	2017-03-22T14:04:07.7084503Z	2017-03-22T14:29:08.7083435Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120
4	VAVCO startup - VAVCO 3-09 - 2017-03-22 07:04:...	2017-03-22T14:04:07.8020935Z	2017-03-22T14:34:08.7395935Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.5	0.5	72.0	72.5	73.5	98.962135

We'll need to convert the data to appropriate data types. Notice that the **Setpoint reached** and **Space Humidity at VAV Start** have DICT types as values (curly braces {} show dictionary data type). Those attributes correspond to formula data references in the event frames. We'll need to convert those to numerical values before we can proceed.

- Run these commands, which create a new column "Setpoint reached_values" in our data frame
- This column includes extracted Setpoint value

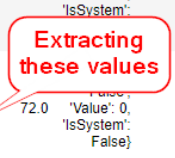
```
: df['Setpoint reached_values'] = None

for row in df.loc[:,'Setpoint reached'].items():
    df.loc[row[0], 'Setpoint reached_values'] = row[1]['Name']

df.head()
```

Setpoint when setpoint reached	Setpoint reached	Setpoint Offset at start time	Setpoint Offset at end time	Setpoint at VAV Start	Room Temperature when setpoint reached	Room Temperature at VAV Start	Outside Relative Humidity at VAV Start	Outside Air Temperature at VAV Start	Element Name	Damper Position at VAV Start	Actual Airflow at VAV Start	% Cooling at VAV Start	Setpoint reached_values
72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	77.976303	46.024254	VAVCO 6-11	50.000000	0.0	19.916670	True
72.0	{'Name': 'False', 'Value': 0, 'IsSystem': False}	1.5	1.0	72.0	73.0	73.5	98.962120	52.112152	VAVCO 4-03	51.333328	341.0	27.824970	False
72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120	52.112156	VAVCO 3-10	50.000000	0.0	19.883341	True
72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120	52.112156	VAVCO 6-11	50.000000	0.0	19.850000	True
72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.5	0.5	72.0	72.5	73.5	98.962135	52.112171	VAVCO 3-09	50.000000	0.0	27.608311	True

- Run these commands, which create a new column “Space Humidity at VAV Start_values” in our data frame
- This column includes extracted “Space Humidity at VAV Start” values

```
df['Space Humidity at VAV Start_values'] = None

for row in df.loc[:,'Space Humidity at VAV Start'].items():
    if type(row[1])==float:
        df.loc[row[0], 'Space Humidity at VAV Start_values'] = row[1]
    else:
        df.loc[row[0], 'Space Humidity at VAV Start_values'] = row[1]['Name']

df.head()
```

	EventFrame	EFStartTime	EFEndTime	Space Humidity at VAV Start	Setpoint when setpoint reached	Setpoint reached	Setpoint Offset at start time	Setpoint Offset at end time	Setpoint at VAV Start	Room Temperature when setpoint reached	Room Temperature at VAV Start	Outside Relative Humidity at VAV Start
0	VAVCO startup - VAVCO 6-11 - 2017-03-03 07:05:...	2017-03-03T15:05:12.0022735Z	2017-03-03T15:25:14.0071563Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem':	1.0	0.5	72.0	72.5	73.0	77.976303
1	VAVCO startup - VAVCO 4-03 - 2017-03-22 07:04:...	2017-03-22T14:04:07.6927185Z	2017-03-22T18:49:11.8333892Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	'Value': 0, 'IsSystem': False}	1.5	1.0	72.0	73.0	73.5	98.962120
2	VAVCO startup - VAVCO 3-10 - 2017-03-22 07:04:...	2017-03-22T14:04:07.7084503Z	2017-03-22T14:19:07.7552185Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120
3	VAVCO startup - VAVCO 6-11 - 2017-03-22 07:04:...	2017-03-22T14:04:07.7084503Z	2017-03-22T14:29:08.7083435Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.0	0.5	72.0	72.5	73.0	98.962120
4	VAVCO startup - VAVCO 3-09 - 2017-03-22 07:04:...	2017-03-22T14:04:07.8020935Z	2017-03-22T14:34:08.7395935Z	{'Name': 'No Data', 'Value': 248, 'IsSystem': ...	72.0	{'Name': 'True', 'Value': 1, 'IsSystem': False}	1.5	0.5	72.0	72.5	73.5	98.962135



Note that the time stamps don't align with what we expect. This is because they have been translated to UTC time. Make sure to adjust them to Pacific time:

```
# convert exported start times to appropriate datetime objects
df['EFStartTime'] = pd.to_datetime(df['EFStartTime'])
df['EFEndTime'] = pd.to_datetime(df['EFEndTime'])
df['EFStartTime'].head()
```

```
df['EFStartTime']=df['EFStartTime'].dt.tz_localize('UTC')
df['EFEndTime']=df['EFEndTime'].dt.tz_localize('UTC')
df['EFStartTime'].head()
```

```
df['EFStartTime']=df['EFStartTime'].dt.tz_convert('US/Pacific')
df['EFStartTime'].head()
```

```
df['EFEndTime']=df['EFEndTime'].dt.tz_convert('US/Pacific')
df['EFEndTime'].head()
```

```
# duration in minutes
df['Event Frame Duration'] = (df.EFEndTime-df.EFStartTime).values / np.timedelta64(1, "m")
```

For the purposes of this exercise, we will focus on the event frames that eventually reach their setpoint. Filter the event frames based on the "Setpoint reached" attribute.

```
# create filter for events where the setpoint was reached
reachSP = df['Setpoint reached_values']=="True"
```

Then convert the end time to a float value. E.g. if setpoint was reached at 7:30am, then the endtime is 7.5h.

```
# convert the end time to a float value
endtime = df.loc[reachSP, 'EFEndTime'].dt.hour + df.loc[reachSP, 'EFEndTime'].dt.minute/60
```

If the setpoint was reached earlier than 7am, then we call it waste and find the value of it.

```
# hours running before 7am = wasted energy
waste = 7-endtime[endtime<7]
```

If the setpoint was reached later than 7am, then we name it as employee discomfort.

```
# hours running after 7am = employee discomfort
discomfort = endtime[endtime>7]-7
```

Total hours of wasted energy:

```
# hours of wasted energy
waste.sum()
```

```
5503.100000000005
```

Total hours of employee discomfort:

```
# hours of employee discomfort
discomfort.sum()
```

```
281.0999999999997
```

Let's exclude the features (columns) that don't affect event frame duration (reaching setpoint / cooling process):

```
exclude = [
    'EFStartTime',
    'EFEndTime',
    'Event Frame Duration',
    'Setpoint reached',
    'Setpoint Offset at end time',
    'Room Temperature when setpoint reached',
    'Setpoint reached_values',
    'Element Name',
    'Setpoint when setpoint reached',
    'Space Humidity at VAV Start',
    'EventFrame'
]
candidates = df.drop(exclude, axis=1)
x = df.loc[reachSP, candidates.columns].copy()
```

Based on the correlation plots of each of the possible attributes, a few attributes appear related to the event frame duration:

- Outside Air Temperature at VAV Start
- Outside Relative Humidity at VAV Start
- Setpoint at VAV Start
- Setpoint Offset at start time
- Space Humidity at VAV Start

Note that we do not include **% cooling at VAV start** because SMEs have informed us that this is not a property of the room, but just reflects how the unit is operating. It's very important to use your SMEs. We also don't include room temperature because we already have the setpoint and the setpoint offset (difference between room temperature and setpoint). We could include the room temperature instead of the setpoint but not both since they are not independent.

9.3 Develop a predictive model

9.3.1 Export data using PI Integrator for BA

We'll use Python to build a model to describe the data. Rather than accessing the PI data each time we run the model, the PI Integrator for Business Analytics will automatically shape and export the PI data to a flat file that can be read for analysis.

Objective: Export data to a flat file using the PI Integrator for Business Analytics

Questions:

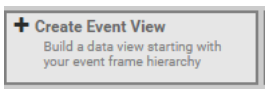
- Follow the instructions below to export the PI data to a text file. Does it have the format that you expect?

- How is this different from loading data with PI DataLink or the PI Web API libraries?

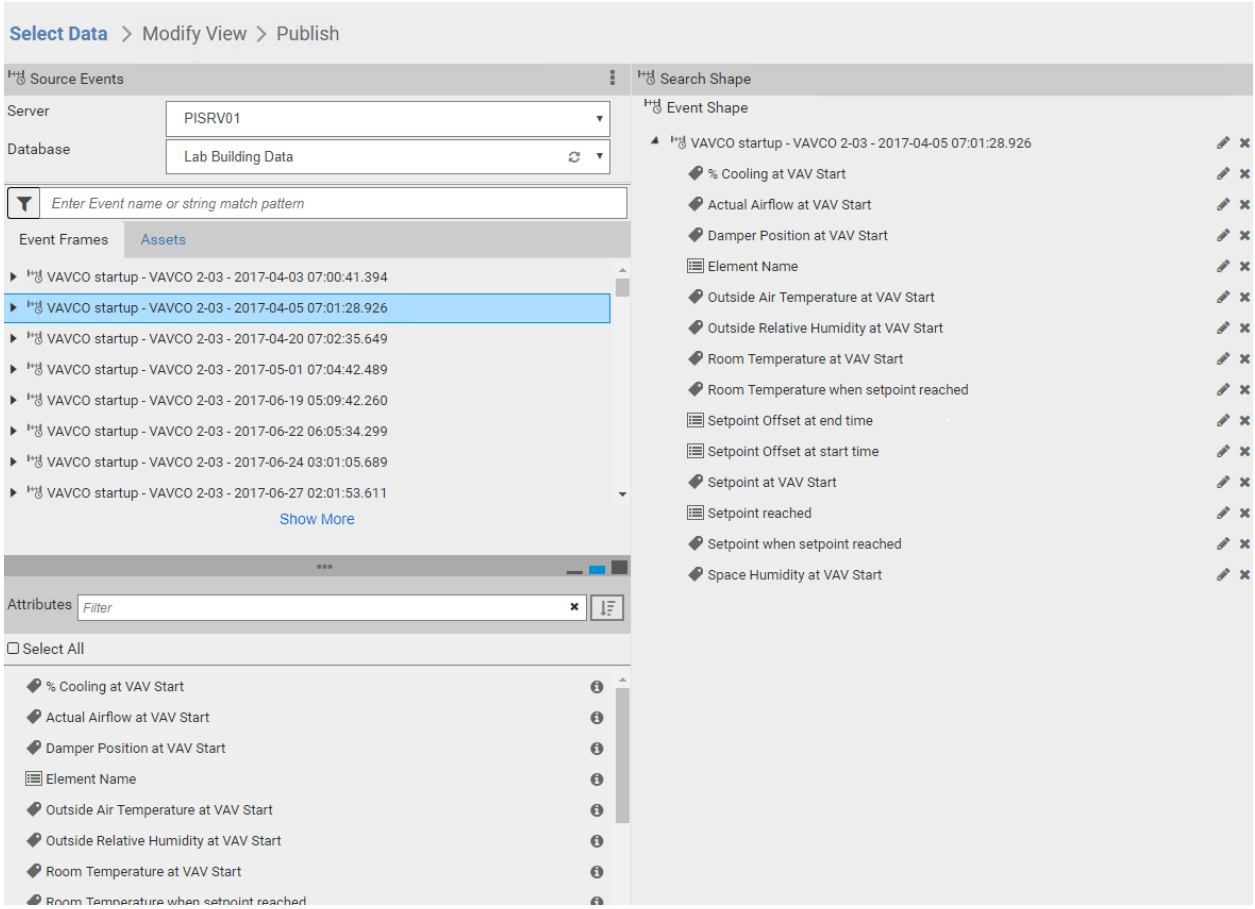
Solution:

We'll use Python or R to train a model to describe the data. Rather than accessing the PI data each time the model is run, we'll use the PI Integrator for Business Analytics to automatically shape and export the PI data to a SQL table. The SQL table will then be read by our Python/R code for analysis.

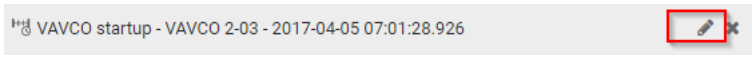
- Open **Google Chrome on PISRV01** and navigate to **https://pisrv01.pischool.int:444**
- Click **Create Event View** and name the view **StudentTraining**



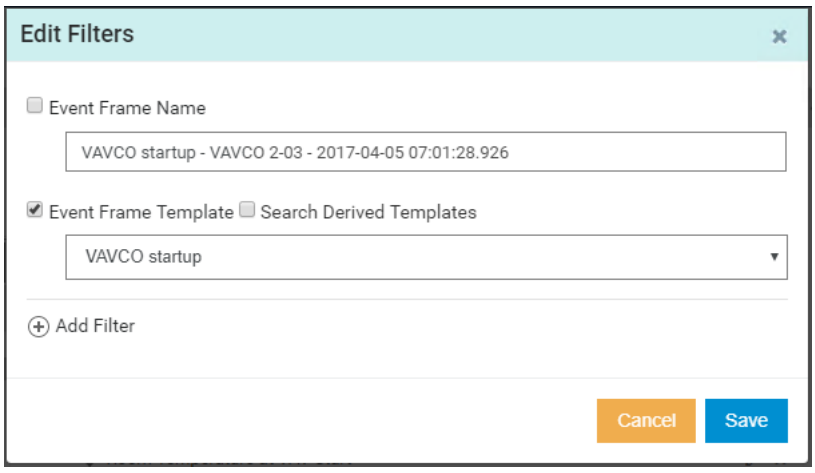
- Make sure the **Lab Building Data** database is selected
- Click one of the event frames in the navigation pane, click **Select All** in the attribute pane, and drag an attribute to the **Search Shape** pane



- Click the edit button next to the event frame name



- Uncheck **Event Frame Name** and check **Event Frame Template**

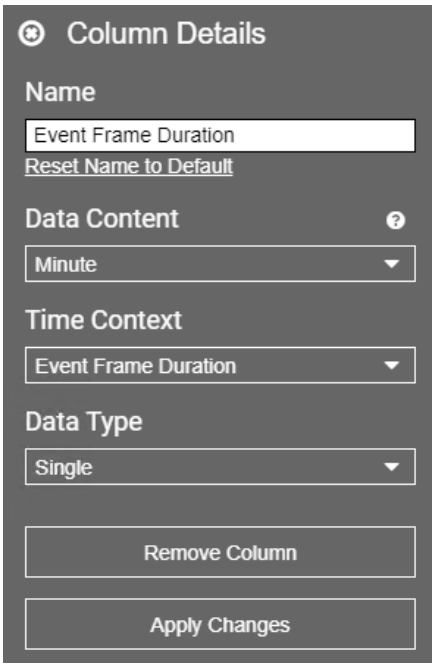


- Click Save
- The Matches list should populate with all the event frames.

- Click Next

Now is a good opportunity to review the data that the integrator has retrieved. You may notice that the **Duration** column has integer values of 0-3, reflecting an integer number of hours that the event frame was active. This does not give us much information. We'll want to use a floating data type with minutes as the unit of measure.

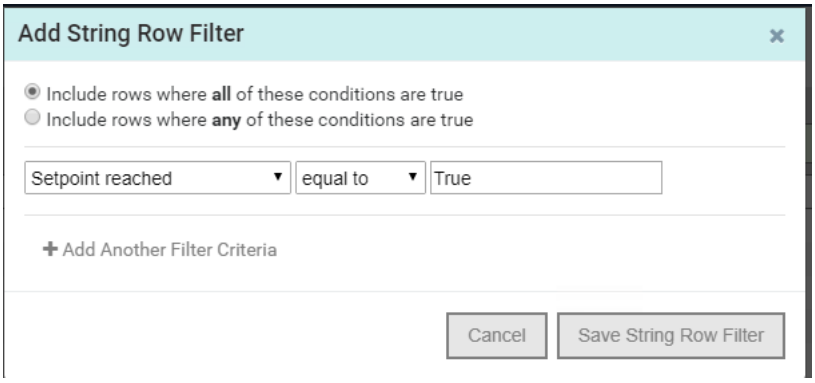
- Select the **Event Frame Duration** column
- Change **Data Content** to **Minute** and **Data Type** to **Single**



- Click **Apply Changes**

For the purposes of this exercise, we'll only consider startup periods where the setpoint was reached in a reasonable time. Let's filter out startups where **Setpoint reached = False**

- Click **Edit Row Filters**
- Select **String**
- Add the condition so **Setpoint reached equal to True**

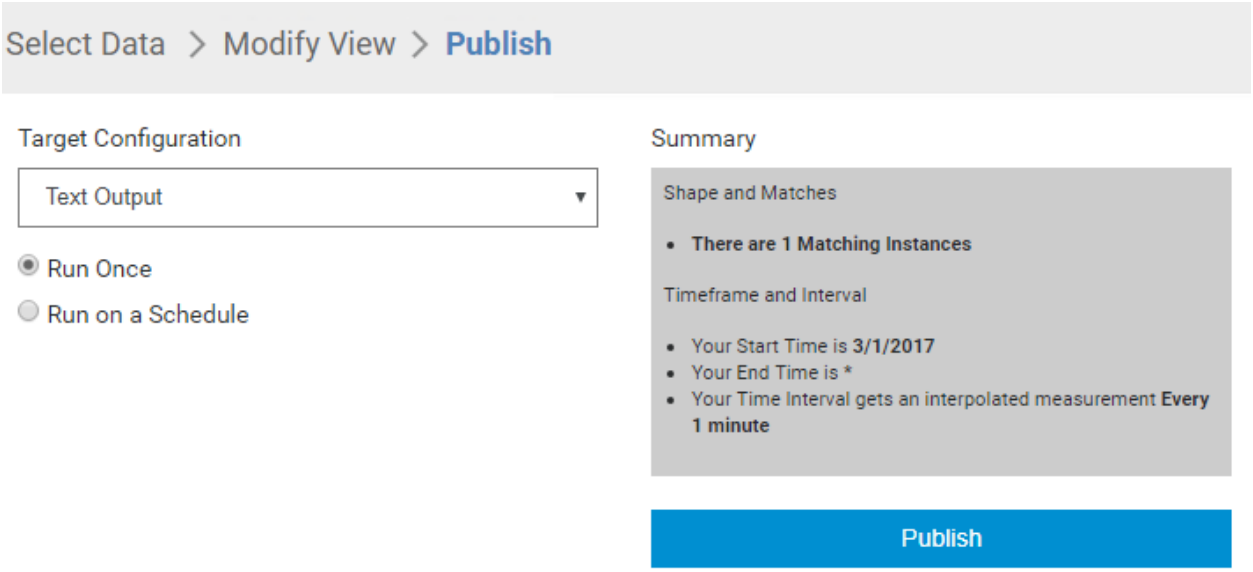


- Click **Save String Row Filter** and **Close**
- Set the **Start Time to 01-Mar-2017** and leave all other values as defaults and click **Apply**

- Click **Next**

Now we must point to the destination for the formatted data.

- Set **Target Configuration** to **Text**
- Select the **Run Once** radio button
- Click **Publish**



Now the data is stored in a format that can easily be loaded into whatever software you like.

9.3.2 Directed Activity – Train and evaluate model

Objective: Make a model to predict the cooling duration

Questions:

- What model can we use to predict startup time?
- How well does this model perform?

Solution:

We can load the data from the output file into our program of choice.

- Open the Jupyter notebook titled **3. Model Training and Evaluation** and follow the instructions
- There may be a compatibility warning – this can be safely ignored
- Load the modules

```
import matplotlib.pyplot as plt
import matplotlib.dates as md
import numpy as np
import pandas as pd

import statsmodels.api as sm
from statsmodels.formula.api import OLS
from sklearn.model_selection import train_test_split
from sklearn.externals import joblib
import seaborn as sns
```

- Load the data file, that PI Integrator for BA generated in the previous exercise

```
df = pd.read_csv("C:\\Lab\\Data\\" + view_name + ".txt")
df.head()
```

- Change endtime to Pacific timezone and convert to a float value:

```
endtime = pd.to_datetime(df['Event Frame End Time']).dt.tz_localize('US/Pacific')
df['EndTimeValue'] = endtime.dt.hour + endtime.dt.minute/60
```

Assign “Event Frame duration” to y (what we try to predict - target), and assign features to x, what helps us to a prediction.

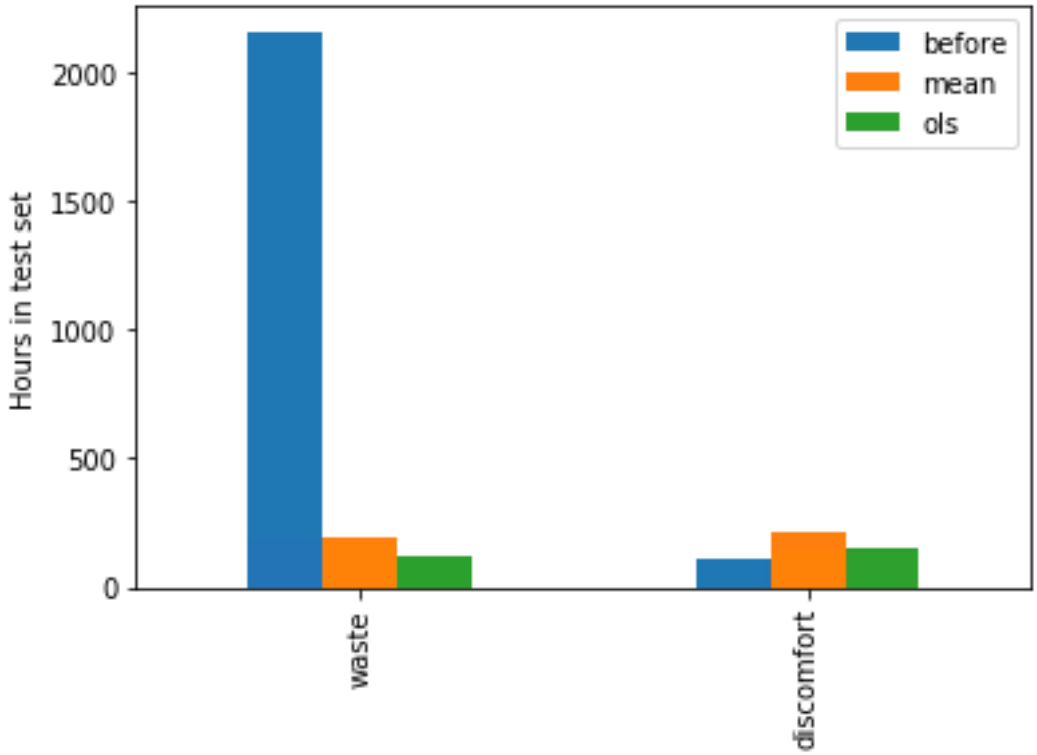
```
features = [
    'Outside Air Temperature at VAV Start',
    'Outside Relative Humidity at VAV Start',
    'Setpoint at VAV Start',
    'Setpoint Offset at start time',
    'Space Humidity at VAV Start'
]

x = df.loc[:, features]
y = df.loc[:, 'Event Frame Duration']
```

The rest of the notebook goes through the process of assigning appropriate data types, splitting the data into a training and a test set, then fitting the training data using an ordinary least squares (OLS) model:

$$y = \sum \beta_i x_i$$

Where x_i is the predictor value and β_i is a fitted coefficient. This model has the advantage of being easily interpreted, as variable importance can be easily compared using the coefficients.

A naïve solution to this problem – using the mean startup time in all cases – would result in significant energy savings but also increase the amount of time that rooms fail to reach the set point on time. The OLS model would save more energy than the naïve approach without the added discomfort (note that this only includes data from the test set – actual savings would be much higher):



	before	mean	ols
waste	2152.200000	195.051352	123.911036
discomfort	112.816667	211.594256	147.715519

Now that we're satisfied with the model, we'll move forward with operationalizing it. For sure, there is a room for improvement, you can try other prediction methods on your own using this data. OLS model is saved to **C:\Lab\Python\ols.pkl**, but feel free to export whatever model you've created.

9.4 Operationalize the predictive model to time cooling correctly

9.4.1 Stream data to Kafka using PI Integrator for BA

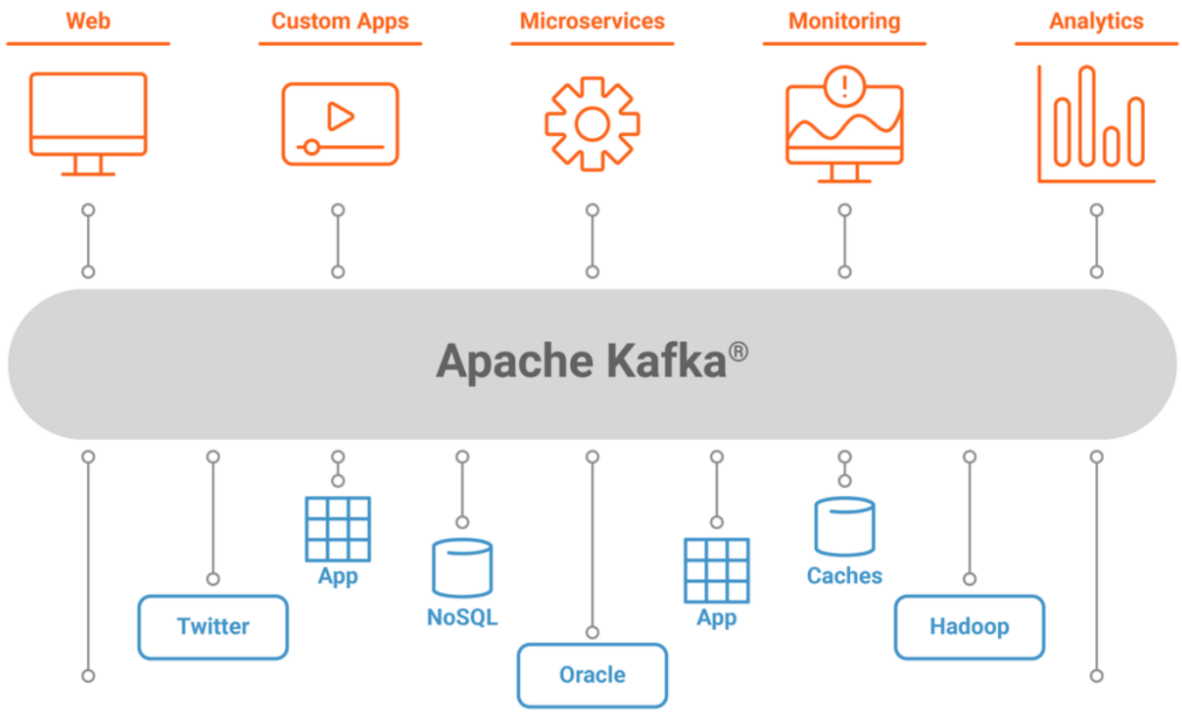
Apache Kafka is a distributed streaming platform that:

- Publishes and subscribes to streams of records, similar to a message queue or enterprise messaging system.
- Stores streams of records in a fault-tolerant durable way.
- Processes streams of records as they occur.

Kafka is used for these broad classes of applications:

- Building real-time streaming data pipelines that reliably get data between systems or applications.
- Building real-time streaming applications that transform or react to the streams of data.

Kafka is run as a cluster on one or more servers that can span multiple datacenters. The Kafka cluster stores streams of records in categories called topics. Each record consists of a key, a value, and a timestamp.



Kafka has these core APIs:

Producer API

Applications can publish a stream of records to one or more Kafka topics.

Consumer API

Applications can subscribe to topics and process the stream of records produced to them.

Streams API

Applications can act as a *stream processor*, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.

Connector API

Build and run reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

In Kafka the communication between the clients and the servers is done with a simple, high-performance, language agnostic TCP protocol. This protocol is versioned and maintains backwards compatibility with older version. The Java client is provided for Kafka, but clients are available in many languages.

More information you can find here:
https://docs.confluent.io/5.5.1/kafka/introduction.html

ZooKeeper is used in distributed systems for service synchronization and as a naming registry. When working with Apache Kafka, ZooKeeper is primarily used to track the status of nodes in the Kafka cluster and maintain a list of Kafka topics and messages.

For now, Kafka services cannot be used in production without first installing ZooKeeper. This is true even if your use case requires just a single broker, single topic, and single partition.

Starting with v2.8, Kafka can be run without ZooKeeper. However, this update isn't ready for use in production.
For any distributed system, there needs to be a way to coordinate tasks. Kafka is a distributed system that was built to use ZooKeeper.

Objective: Configure a local instance of Apache Kafka as a target for the integrator

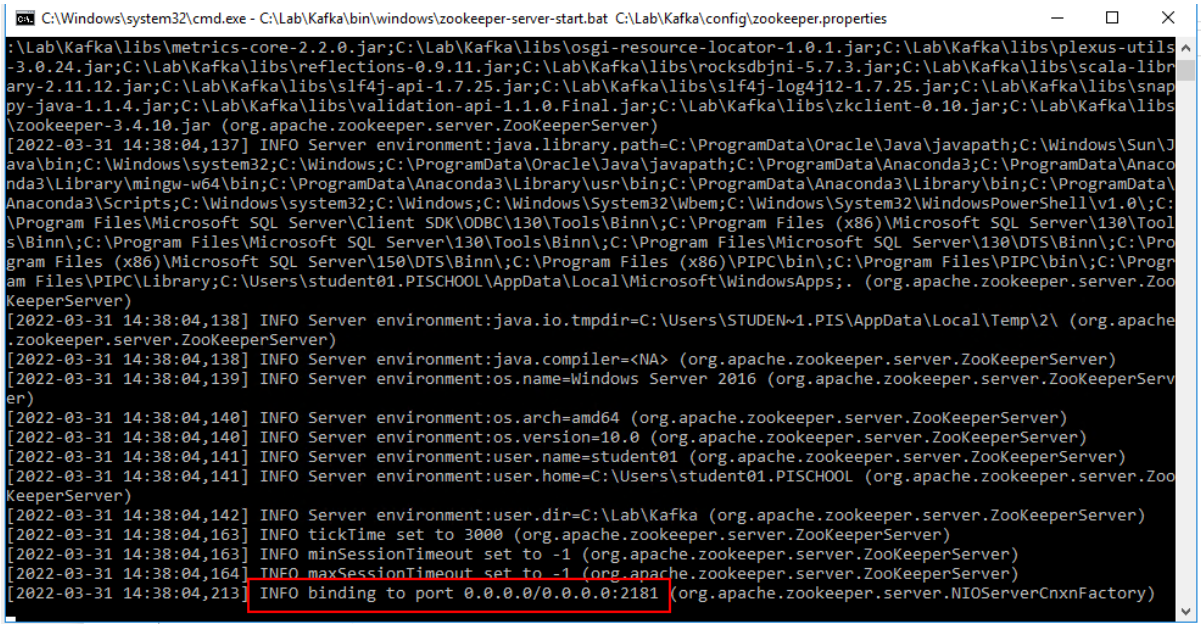
Questions:

- Start Zookeeper and the Kafka server. Are you able to stream values via the console?
- Configure the integrator to stream to the server. Do you see the expected objects?
- When would we want to poll vs. send live updates?

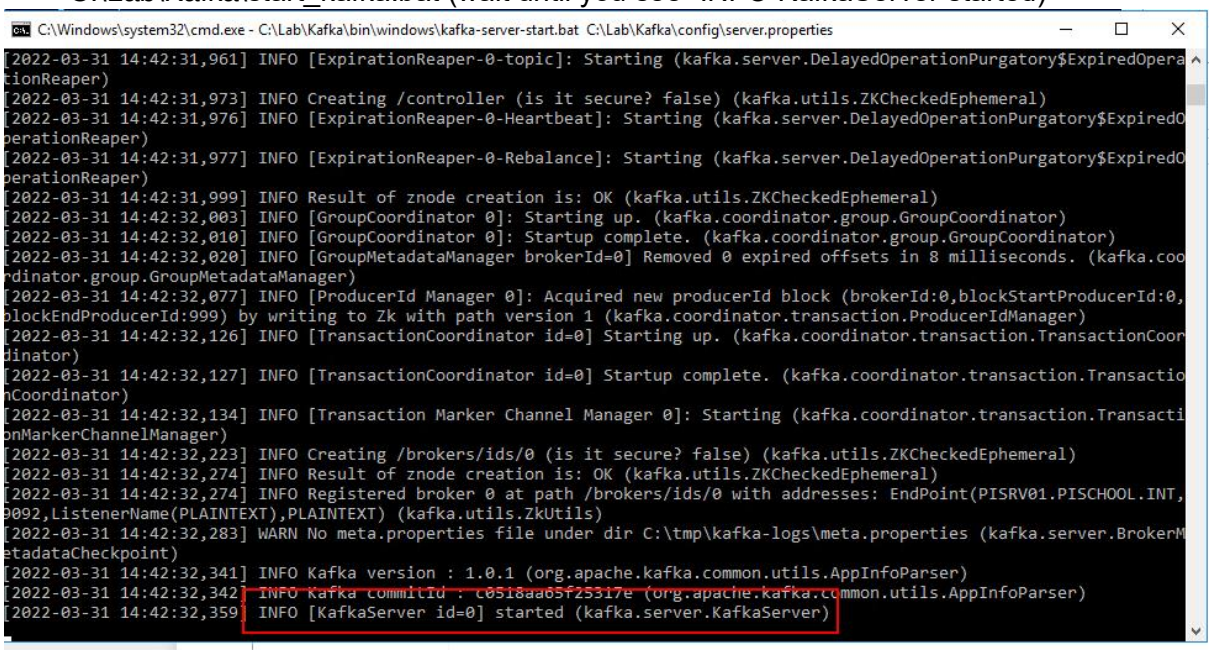
Solution:

Before we implement the model, let's start Zookeeper and Kafka on PISRV01. Run the following batch commands:

- C:\Lab\Kafka\start_zookeeper.bat (wait until you see an "INFO binding to port" message)

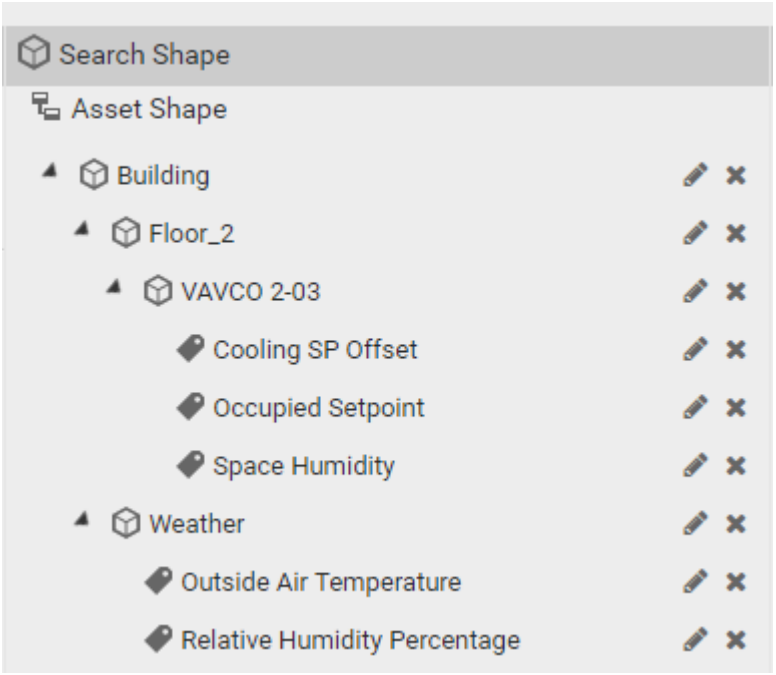


- C:\Lab\Kafka\start_kafka.bat (wait until you see “INFO KafkaServer started)

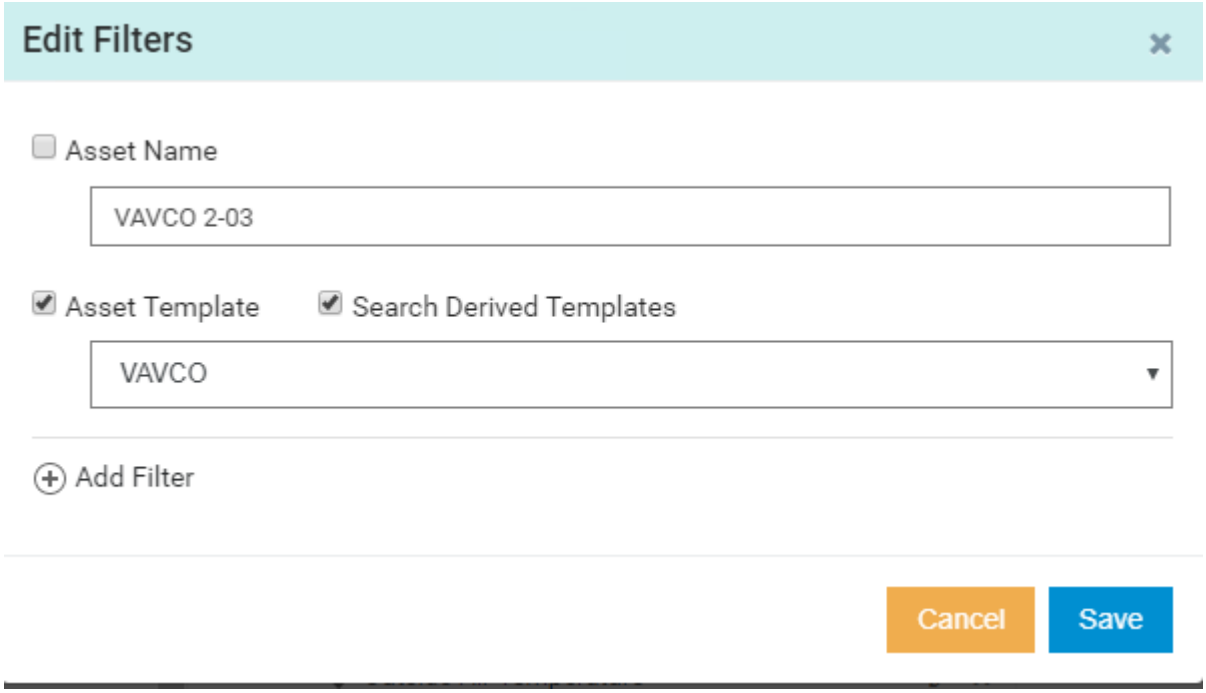


Once both services have started up, we’ll publish a streaming view to the integrator.

- Open **Google Chrome** and navigate to the integrator website
- Click **Create Streaming View** and give the view a name **StreamingView**
- Build out the asset hierarchy to include all attributes to be included in the model
    - NOTE: Since the attributes on **Weather** and the **VAVCO** elements are on different levels, you will need to include the common parent in the shape (see screenshot)



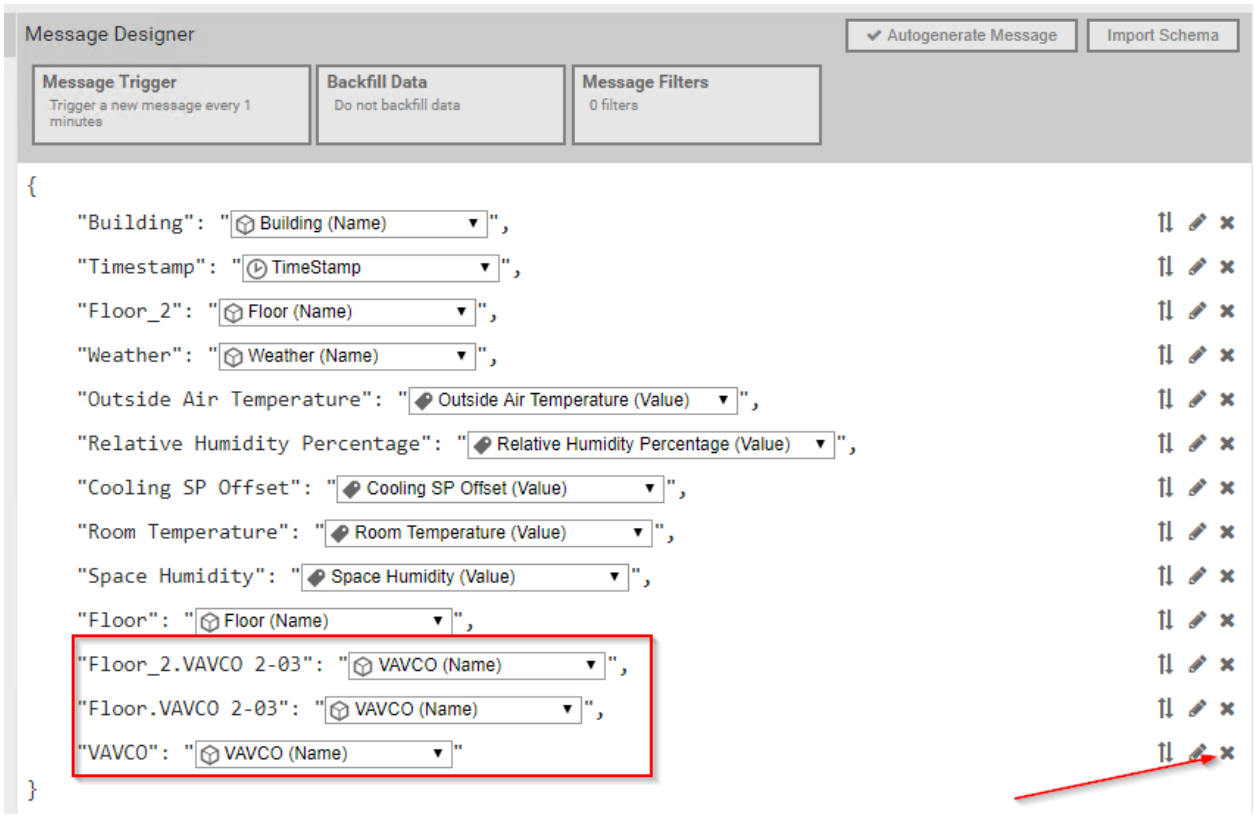
- Change the **VAVCO** and **Floor** elements in the shape to their templates so all values will get streamed
- Select **Search Derived Templates** when editing the **VAVCO** shape element



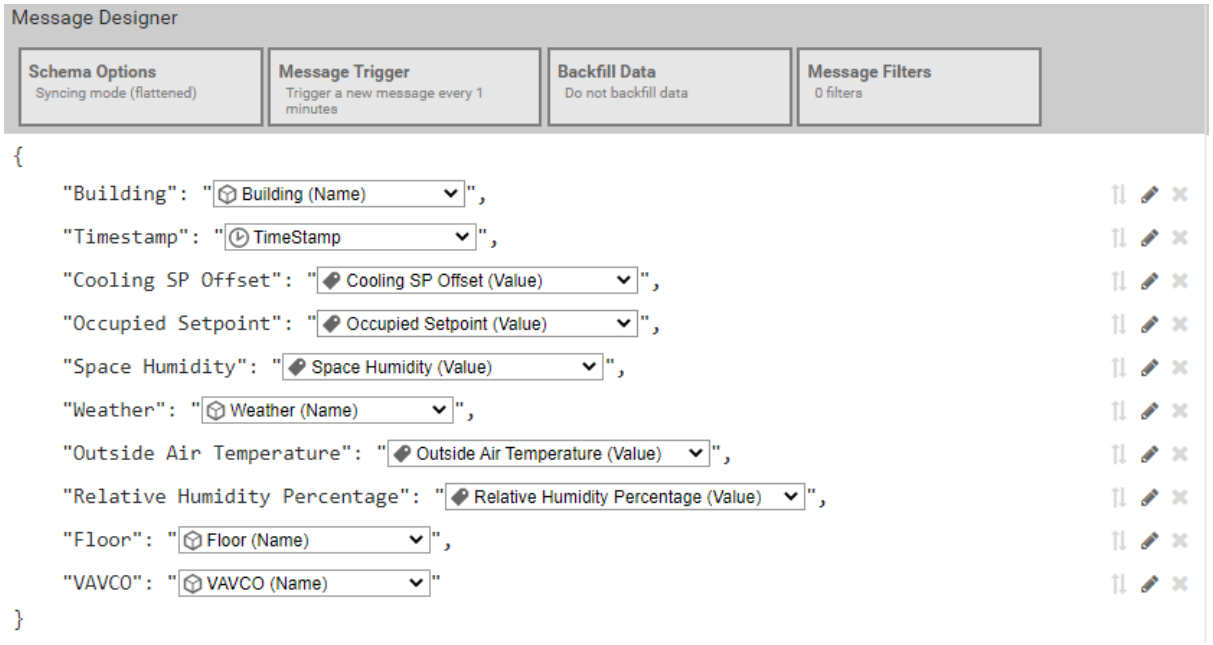
- Click **Next**
- Make sure the **Message Trigger** is set to every 1 minute
- Make sure that the message does not have multiple copies of the same attribute – the pre-release build that we're using does not always handle this elegantly

```
{
  "Timestamp": "2018-03-30 03:28:59",
  "Outside Air Temperature": 50.70000076293945,
  "Relative Humidity Percentage": 72.60001373291016,
  "Cooling SP Offset": 0,
  "Room Temperature": 74.5,
  "Floor_2": "Floor_3",
  "Space Humidity": 44,
  "Floor_2.VAVCO 2-03": "VAVCO 3-11",
  "Floor.VAVCO 2-03": "VAVCO 3-11",
  "Building": "Building",
  "Floor": "Floor_3",
  "VAVCO": "VAVCO 3-11",
  "Weather": "Weather"
},
```


- If you do find duplicates, remove all of them except **"VAVCO"** using the **Message Designer** until there's only one left



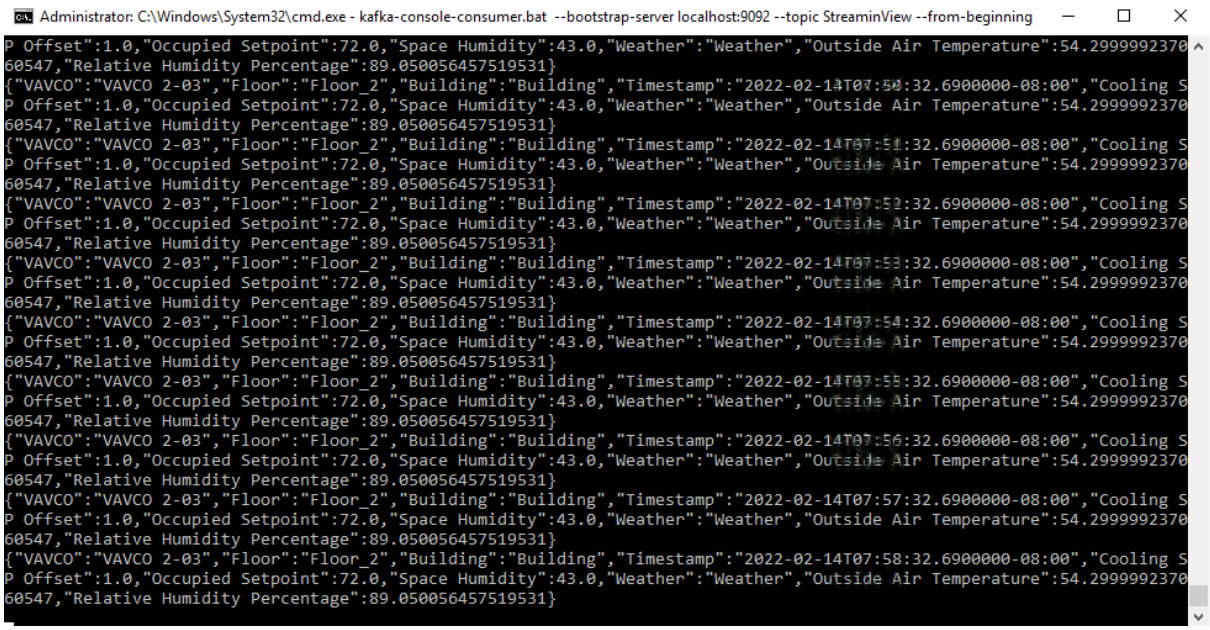
Please verify the format of streaming message:



- Click **Next**
- Set **Target Configuration** to **Kafka** and check the **Topic Name** (matches view name by default). **Click "Get Topics" button.** If the topic name has changed to some GUID, please select the right topic name from the drop-down list.
- Click **Publish**

Check the topic publication from the command prompt

- Run the command prompt from C:\Lab\Kafka\bin\windows
- Run the command
- `kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic StreamingView --from-beginning`
- This should update with JSON objects every minute as the integrator publishes data



9.4.2 Ingest Kafka stream using model

Objective: Operationalize the model

Questions:

- Write a script that consumes the topic created by the integrator. Are you able to see the predicted values written to the console?
- Is there another way that we could achieve this same goal?

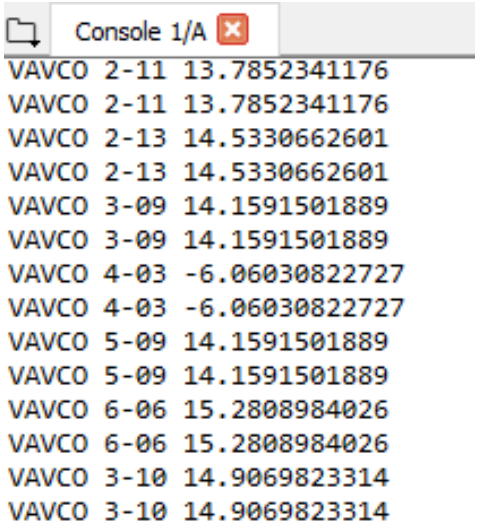
Solution:

Review the Python script called **4-StreamingKafka.py** in **C:\Lab\Python**.

- Open the **Spyder** editor on the desktop – it should open **4-StreamKafka.py** in the editor pane
- Review the different elements of the script
  - **Section 1** contains configuration information
  - **Section 2** translates the data stream to a prediction
  - **Section 3** connects to the PI Server using the PI Web API
  - **Section 4** maps the column names from the streaming view to the column names of the event view (these will differ if the event frame attribute has a different name than the element attribute)
  - **Section 5** connects to the Kafka topic where the integrator is publishing, then makes the prediction and writes back to PI
- Update the parameters in the **Configuration** section of the script
  - **view_name**: the name of the streaming view created by the integrator
  - **password**: Windows password
  - **vavco_name**: name of the key that contains the VAVCO value from the streaming view – should not need to be changed in most cases

```
15 #-----------------------------------------------------------------------------
16 # Configuration parameters (update)
17 #-----------------------------------------------------------------------------
18
19 view_name = 'testview'
20 password = 'password'
21
22 # VAVCO column name from integrator stream - may not need to change
23 vavco_name = 'VAVCO'
24
25 #-----------------------------------------------------------------------------
```

- Click **Run > Run** in **Spyder** and watch the new values written to the console



9.4.3 Concluding remarks

We have now taken a data science project from initial understanding to the proof-of-concept stage built around PI System data. The model would result in measurable cost savings for the customer. We used several off-the-shelf PI System products to enable this process, while also leveraging Python and Apache Kafka for model development and implementation.

This exercise is not meant to cover every way a data scientist might interact with PI data. For example, we used the PI Integrator for Business Analytics to shape and export the raw data. You could also use the PI OLEDB Enterprise for this purpose, though that would require a bit more scripting. The Optional section, below, walks through that process.

10 Build the Final Report

With all this data we are now ready to build a report from scratch. We are going to repeat some concepts that we learned before in the course and also take a look at one of the other approaches, how PI Data can be integrated with 3rd party application. This approach needs SQL scripting knowledge. No need to worry about it for now. Just follow the instructions and analyze the results. Appendix A of the book has all the required information about accessing PI Data with SQL queries. So, you can check it later and do self-paced exercises with your own pace (Chapter 11. Appendix A).

10.1 Preparing and Importing the Tables

For the report, we are going to separate the time-series data from the static data and configure table relationships to join the data sets together. Technically, we could design the queries such that the result set is a single table. However, in real life not all of the data is always in PI and several data sources must be joined together. This can of course be done at the query level, but also in Power BI.

It's not always clear at what level you should solve a given problem:

- Do I import static data into AF and then retrieve the data from AF, or do I bring the data into the report and join it with PI data there?
- Do I implement the calculation in AF and retrieve the result from AF, or do I implement the calculation in the report?

There are ways to avoid doing many of the following steps, but this will help prepare you for real world reports where constant modifications, revisions, and fine-tuning must be performed.

10.1.1 Directed Activity – Prepare Static Data Table

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Modify the existing query to only include static data and exclude unnecessary columns.

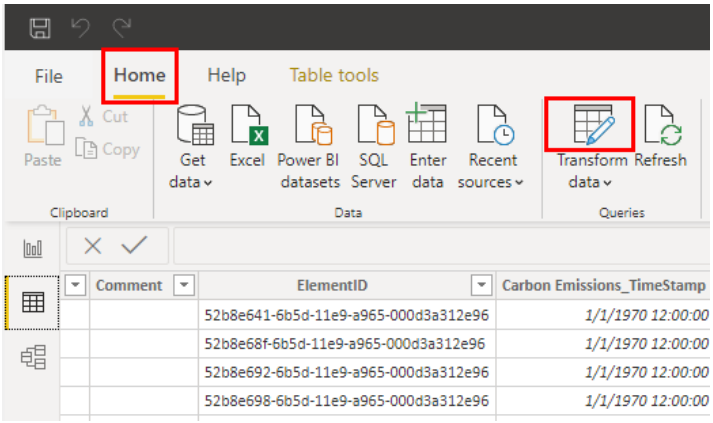
Approach:

- Open a Power BI Report and inspect a pre-existing query
- Modify the SELECT statement to only include static data columns
- Replace the query in the query editor with a new query

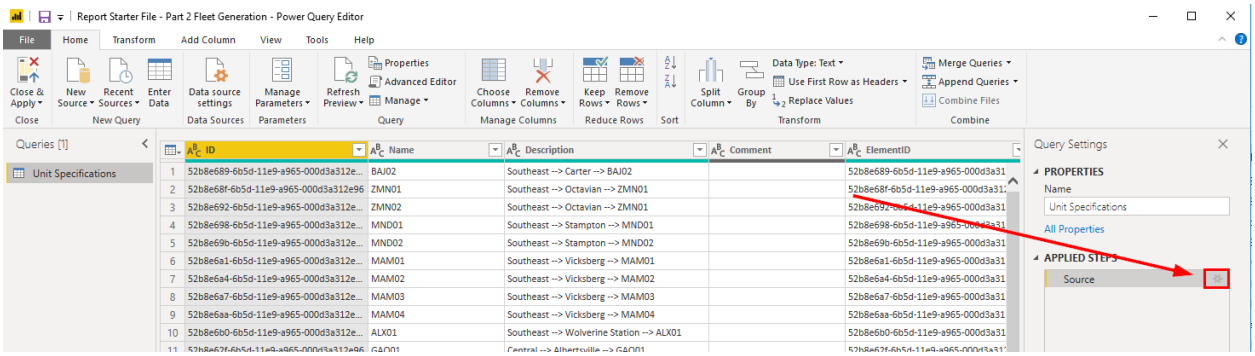
Open **C:\Class\Final Exercise\Starter File - Final Exercise.pbix**

Go to the Data Tab or the Fields list and look at all the useless fields. We can make this data set much easier to work with by either transforming the data, or making changes to the query. This is the result of the PI SQL query and retrieve only static values for all our fleet generation units. Let's see, how this query looks like.

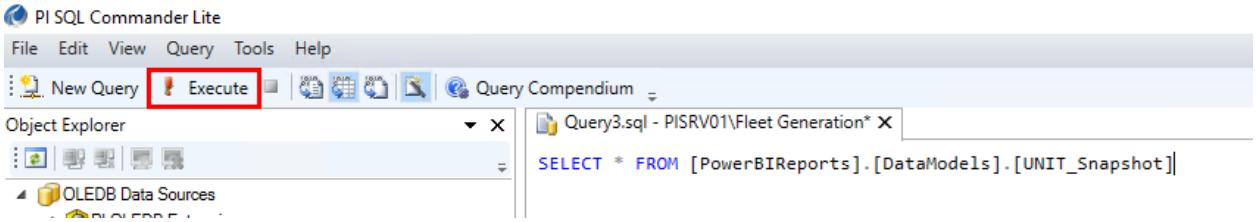
On the Home Tab, Select Transform Data to launch the Power Query Editor:



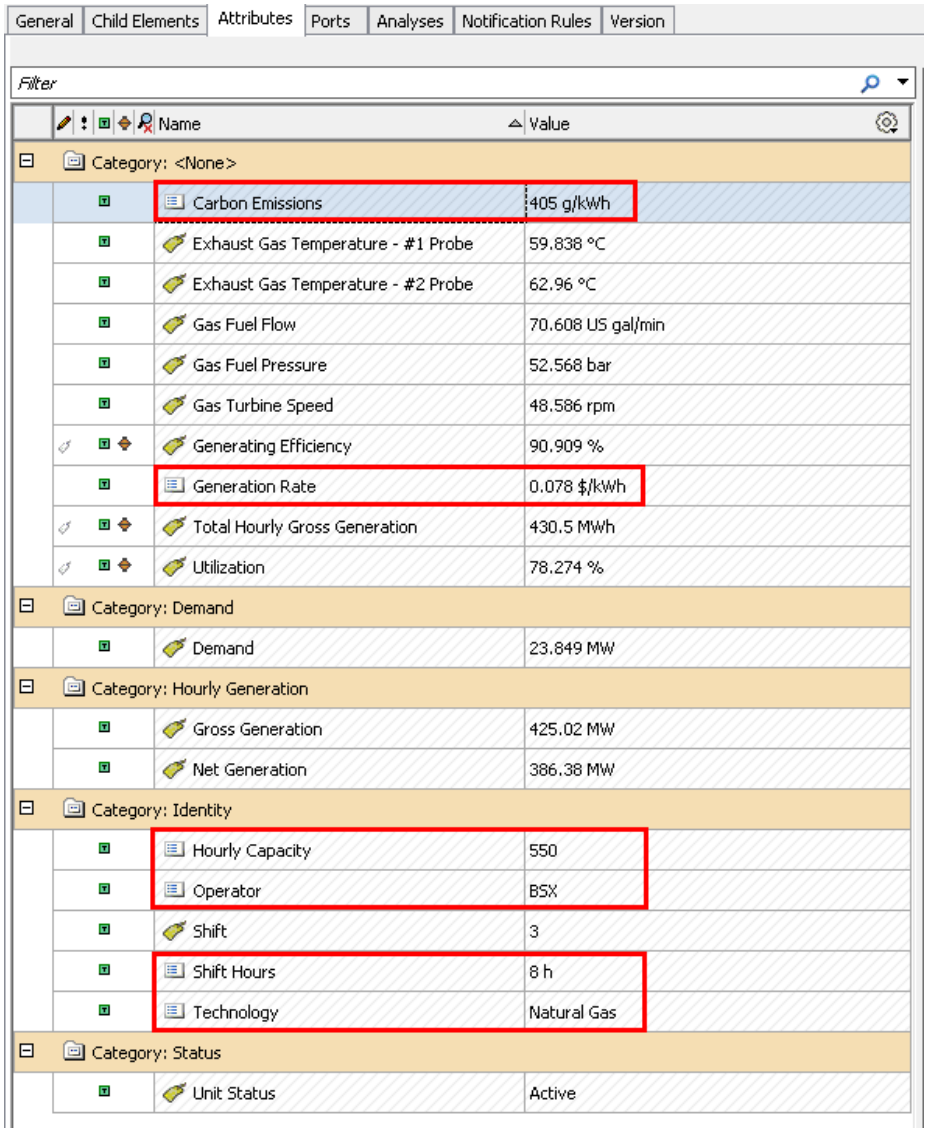
Under Applied Steps, click the gear next to Source:



Copy the query to a new query in PI SQL Commander and **Execute** it as a sanity check:



Now the actual modifications. Edit the select statement to only include the Name field and static attributes: Carbon Emissions, Generation Rate, Hourly Capacity, Operator, Shift Hours, and Technology.



The resulting query is then:

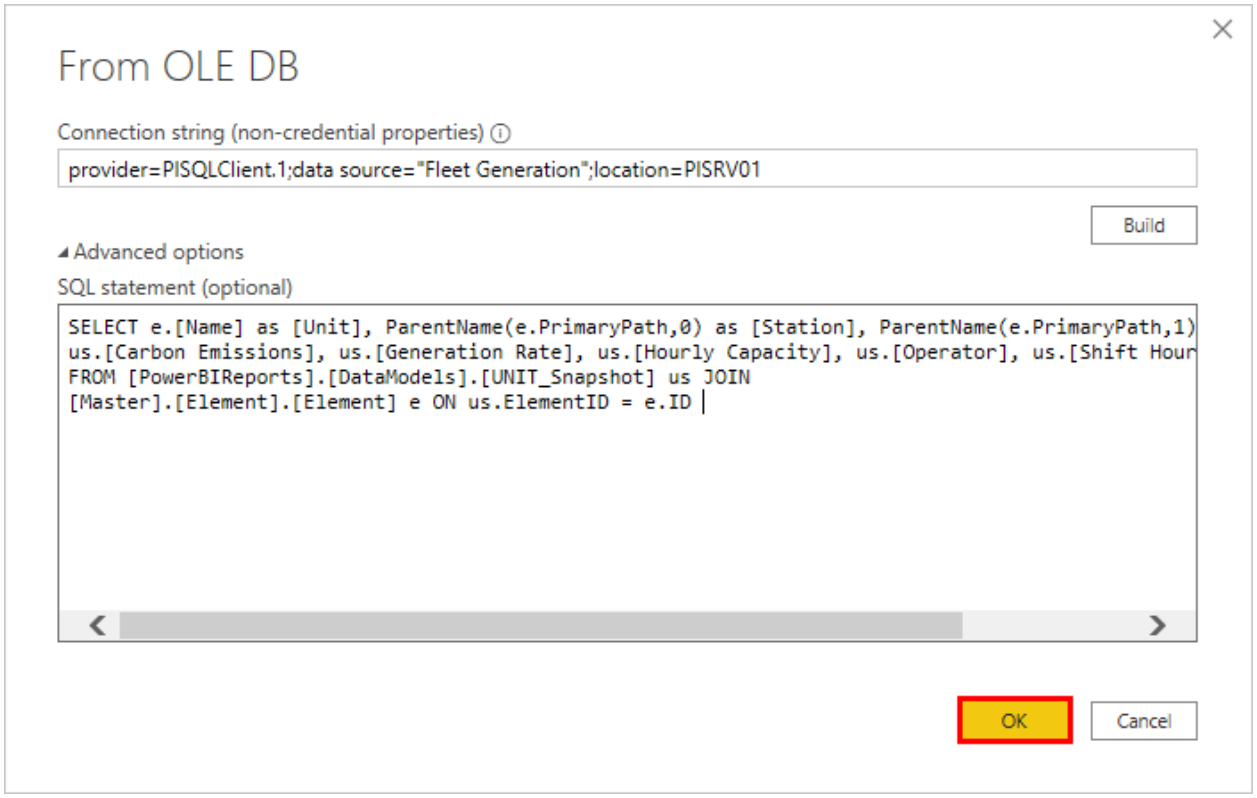
```
SELECT [Name] as [Unit], [Carbon Emissions], [Generation Rate], [Hourly
Capacity], [Operator], [Shift Hours], [Technology]
FROM [PowerBIReports].[DataModels].[UNIT_Snapshot]
```

We're also going to want Station and Region information which we could get from the ParentName() function, however PrimaryPath is not available in the UNIT_Snapshot view. We'll have to:

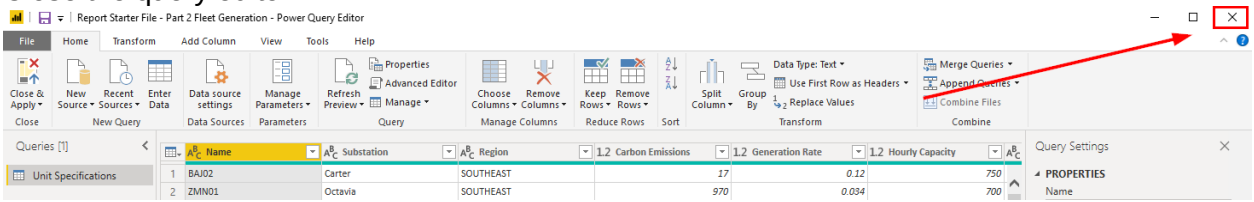
- JOIN to the Element Table using the element ID
- Alias the UNIT_Snapshot view as us and the Element table as e
- Explicitly specify whether each column is from e or us

```
SELECT e.[Name] as [Unit], ParentName(e.PrimaryPath,0) as [Station],
ParentName(e.PrimaryPath,1) as [Region],
us.[Carbon Emissions], us.[Generation Rate], us.[Hourly Capacity],
us.[Operator], us.[Shift Hours], us.[Technology]
FROM [PowerBIReports].[DataModels].[UNIT_Snapshot] us JOIN
[Master].[Element].[Element] e ON us.ElementID = e.ID
```

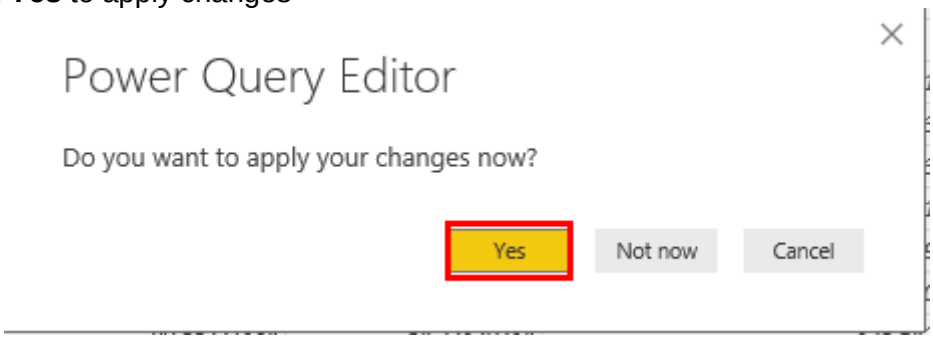
Paste the above query back into the Power BI query editor and click **OK**



Close the query editor



Click **Yes** to apply changes



It should reload the data (30 rows) successfully.

Carbon Emissions	Generation Rate	Hourly Capacity	Operator	Shift Hours	Technology	Region	Station	Unit
17	0.12	750	PRT	12	Wind	SOUTHEAST	Carter	BAJ02
970	0.034	700	BSX	12	Coal	SOUTHEAST	Octavia	ZMN01
970	0.034	550	BSX	12	Coal	SOUTHEAST	Octavia	ZMN02
405	0.078	650	BSX	12	Natural Gas	SOUTHEAST	Stampton	MND01
405	0.078	550	BSX	12	Natural Gas	SOUTHEAST	Stampton	MND02
970	0.034	600	BSX	12	Coal	SOUTHEAST	Vicksberg	MAM01
970	0.034	700	BSX	12	Coal	SOUTHEAST	Vicksberg	MAM02
970	0.034	700	BSX	12	Coal	SOUTHEAST	Vicksberg	MAM03
970	0.034	700	BSX	12	Coal	SOUTHEAST	Vicksberg	MAM04
17	0.12	500	COG	12	Wind	SOUTHEAST	Wolverine Station	ALX01
405	0.078	550	BSX	8	Natural Gas	CENTRAL	Albertsville	GAO01
405	0.078	650	BSX	8	Natural Gas	CENTRAL	Albertsville	GAO02
405	0.078	600	BSX	8	Natural Gas	CENTRAL	Beryl Ridge	BCU01
405	0.078	550	BSX	8	Natural Gas	CENTRAL	Beryl Ridge	BCU02
405	0.078	650	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB01
405	0.078	600	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB02
405	0.078	550	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB03
405	0.078	600	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB04
405	0.078	650	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB05
405	0.078	600	BSX	8	Natural Gas	CENTRAL	Carbondale	TCB06
405	0.078	650	BSX	8	Natural Gas	NORTH	Ebbitt	PQE02
405	0.078	500	BSX	8	Natural Gas	NORTH	Ebbitt	PQE03
405	0.078	550	BSX	8	Natural Gas	NORTH	Ebbitt	PQE04
970	0.034	600	NOP	8	Coal	NORTH	Greenlawn	PTC01
970	0.034	500	NOP	8	Coal	NORTH	Greenlawn	PTC02
970	0.034	750	PEE	8	Coal	NORTH	Greenlawn	PTC03
17	0.12	600	COG	8	Wind	NORTH	Madison	CEC01
17	0.12	600	COG	8	Wind	NORTH	New Bedford	POE01
17	0.12	500	BSX	12	Wind	SOUTHEAST	Brick Canyon	PLT01
17	0.12	550	BSX	12	Wind	SOUTHEAST	Brick Canyon	PLT02

10.1.2 Directed Activity – Import the GetSampledValues data

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

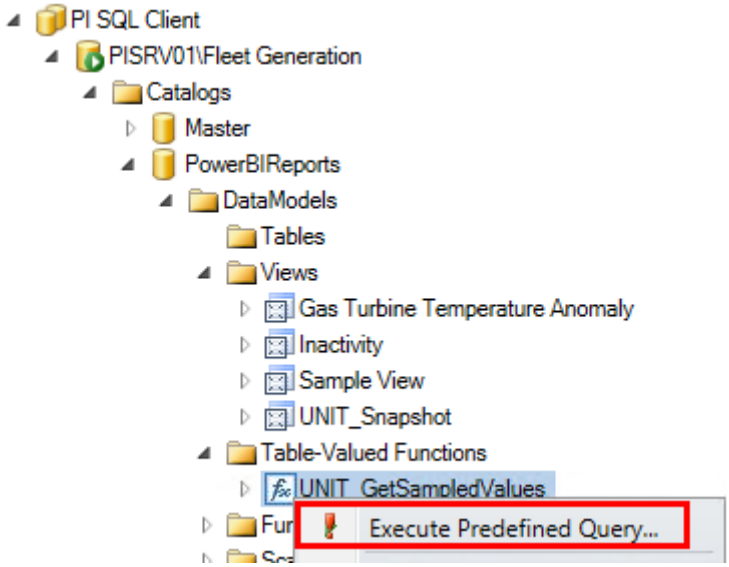
Objective:

Start with the predefined query for UNIT_GetSampledValues and restrict the result set to include relevant time-series data only. UNIT_GetSampledValues contains 1 interpolated value per interval per element based on start time, end time and interval for the Unit template. You can find more information on how UNIT_GetSampledValues was created in Appendix A.

Approach:

- Execute predefined query for UNIT_GetSampledValues
- Modify the SELECT statement to only include Name and time-series data columns
- Import the data set to Power BI

Go to PI SQL Commander and execute the predefined query for **UNIT_GetSampledValues**.



Modify the select statement to **remove the TOP 100 part**, change the start time to ***-7d**, and change the end time to ***. **Alias e.Name as Unit**. The query is then:

```
SELECT e.Name as [Unit], s.*
FROM
(
   SELECT ID, Name, Template
   FROM [Master].[Element].[Element]
) e
CROSS APPLY
[PowerBIReports].[DataModels].[UNIT_GetSampledValues]
(
   e.ID,  --Element ID
   '*-7d',  --Start Time
   '*',  --End Time
   '1h'  --Time Step
) s
WHERE e.Template = N'UNIT'
```

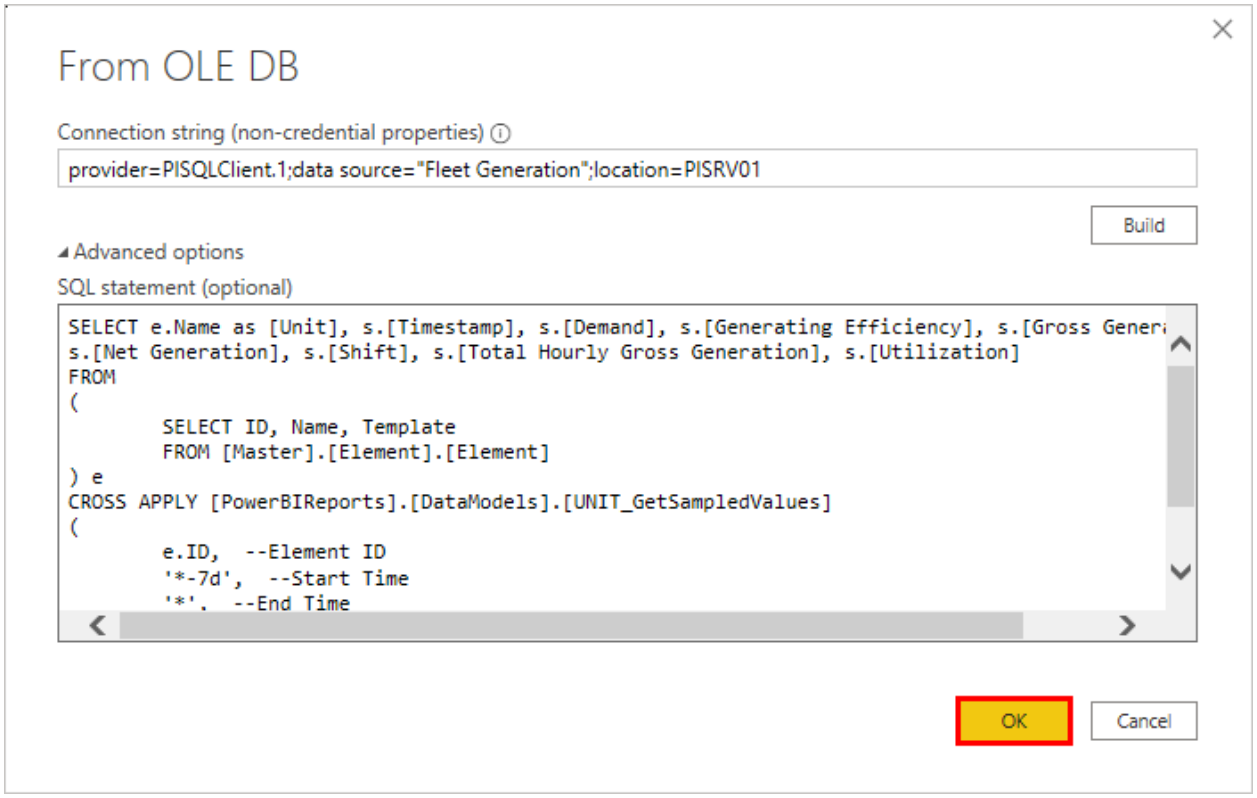
Then change the s.* part to only include the following columns: **Timestamp, Demand, Generating Efficiency, Gross Generation, Net Generation, Shift, Total Hourly Gross Generation, and Utilization**. Include UNITs and Gas Turbines. The query becomes:

```
SELECT e.Name as [Unit], s.[Timestamp], s.[Demand], s.[Generating
Efficiency], s.[Gross Generation],
s.[Net Generation], s.[Shift], s.[Total Hourly Gross Generation],
s.[Utilization]
FROM
(
   SELECT ID, Name, Template
   FROM [Master].[Element].[Element]
) e
CROSS APPLY
[PowerBIReports].[DataModels].[UNIT_GetSampledValues]
(
   e.ID,  --Element ID
   '*-7d',  --Start Time
   '*',  --End Time
   '1h'  --Time Step
) s
WHERE e.Template = N'UNIT' or e.Template = 'Gas Turbine'
```

Execute the query to make sure it still works. Then head back to Power BI.

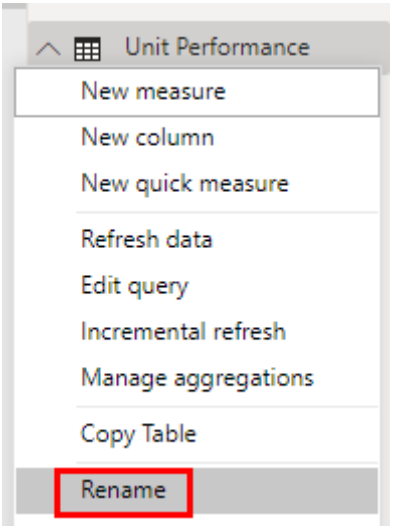
In Power BI, do **Get Data -> More -> Other -> OLE DB**. Build the connection string, enter the query where it says **Advanced options**, and click **OK**. Inspect the preview and Load the data.

We've done this before so there isn't a screenshot for every click this time.



It should import 5070 rows, 30 units x 24 hours x 7 days = 5040, plus 30 rows (1 per unit) for the start time.

Rename **Query1** to **Unit Performance**



10.1.3 Directed Activity – Inspect Table Relationship

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

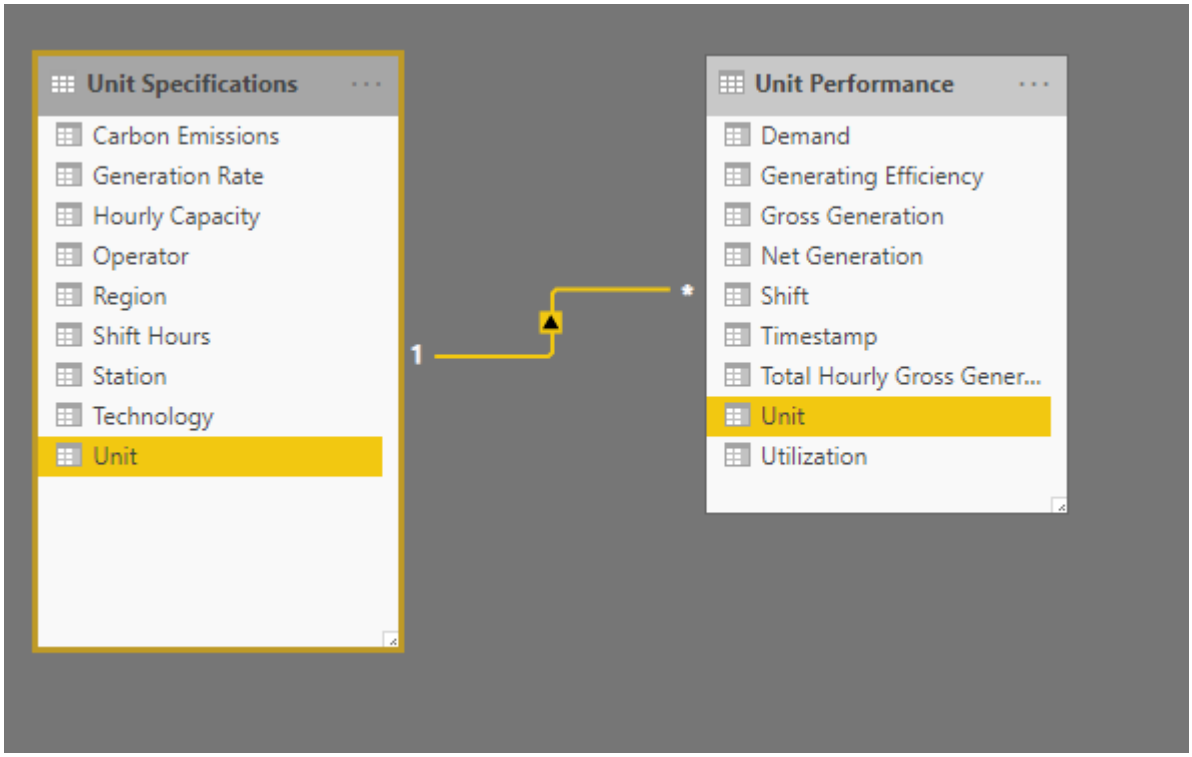
Objective:

Inspect the automatically created table relationship. Power BI should have detected two identically named columns exhibiting a one-to-many relationship.

Approach:

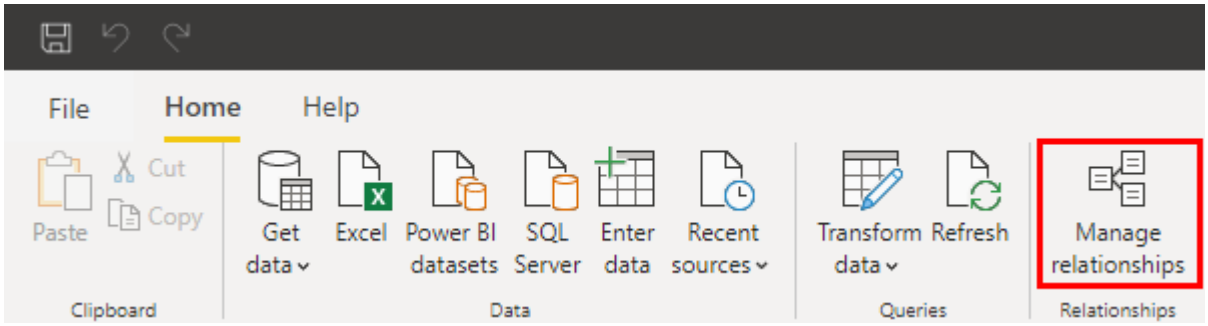
- Open the Power BI relationships tab and inspect the existing relationship

In Power BI, Go to the **Relationships** tab, then move the Unit Performance table to the right so that the relationship line is clearly visible and click on the line:



We can see that Power BI has already detected the relationship between the two tables. This can be thought of as a graphical representation of an INNER JOIN statement. These tables are now joined on the Unit column. For this to work, one of the tables must only contain unique values in the Unit column (ie. the column can serve as a key), as is the case here. This is referred to as a **one-to-many relationship** in some documentation. Each Unit only appears **once** in the Unit Specifications table, whereas each Unit appears **many** times in the Unit Performance table.

Relationships can be manually defined using a drag and drop interface, or through Manage Relationships.



However at this point there is no need.

10.2 Augmenting the Data using DAX

Next we will add a few calculations to the Unit Performance table that will help assess the total Emissions produced and the total cost of generation. We will also add columns for the day of the week and sort the Weekday in Sunday -> Saturday order.

10.2.1 Directed Activity – Calculate the amount of CO2 produced every hour

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Add a DAX formula Calculate the amount of CO2 produced every hour

Approach:

- Add an additional column to the Unit Performance table with the amount of carbon emissions produced.

In Power BI, navigate to the **Data Tab** and select the **Unit Performance** table.

Right-click any column and add a **new column**. Enter the following formula:

CO2 = 'Unit Performance'[Total Hourly Gross Generation]*RELATED('Unit Specifications'[Carbon Emissions])

Note that Total Hourly Gross Generation has units of MWh, and Carbon Emissions has units of g/kWh. Grams/kWh is the same as Kilograms/MWh, and therefore the result will be in KG.

10.2.2 Exercise – Calculate the Generation Cost

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective: Add the cost calculation column to your Unit Performance table

Do you prefer having the calculations within AF as a formula data reference, or within the BI client tools? What are some advantages and disadvantages of each?

Approach:

- Add and additional column named **Cost** to the Unit Performance table with the dollar cost per hour.
- **Hint:** Use this formula
  Cost = 'Unit Performance'[Total Hourly Gross Generation]*RELATED('Unit Specifications'[Generation Rate])*1000
- Take note of the input units. Cost should be in dollars.

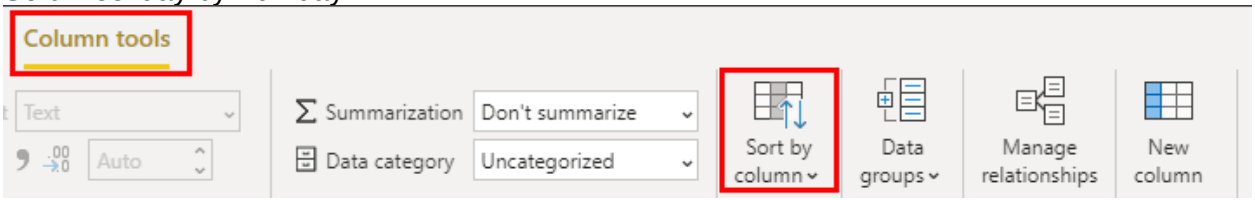
10.2.3 Exercise – Add Column for Day of the Week and sort

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective: Add the day of the week to your Unit Performance table, also add a column with the numerical day of the week and sort by this value

Approach:

- Add and additional column named **Weekday** which shows the day of the week as a string using the FORMAT() function
  Weekday = FORMAT('Unit Performance'[Timestamp],"DDD")
- Add another column named **Numday** which gives the numerical day of the week using the WEEKDAY() function
  Numday = WEEKDAY('Unit Performance'[Timestamp],1)
- Sort Weekday by Numday



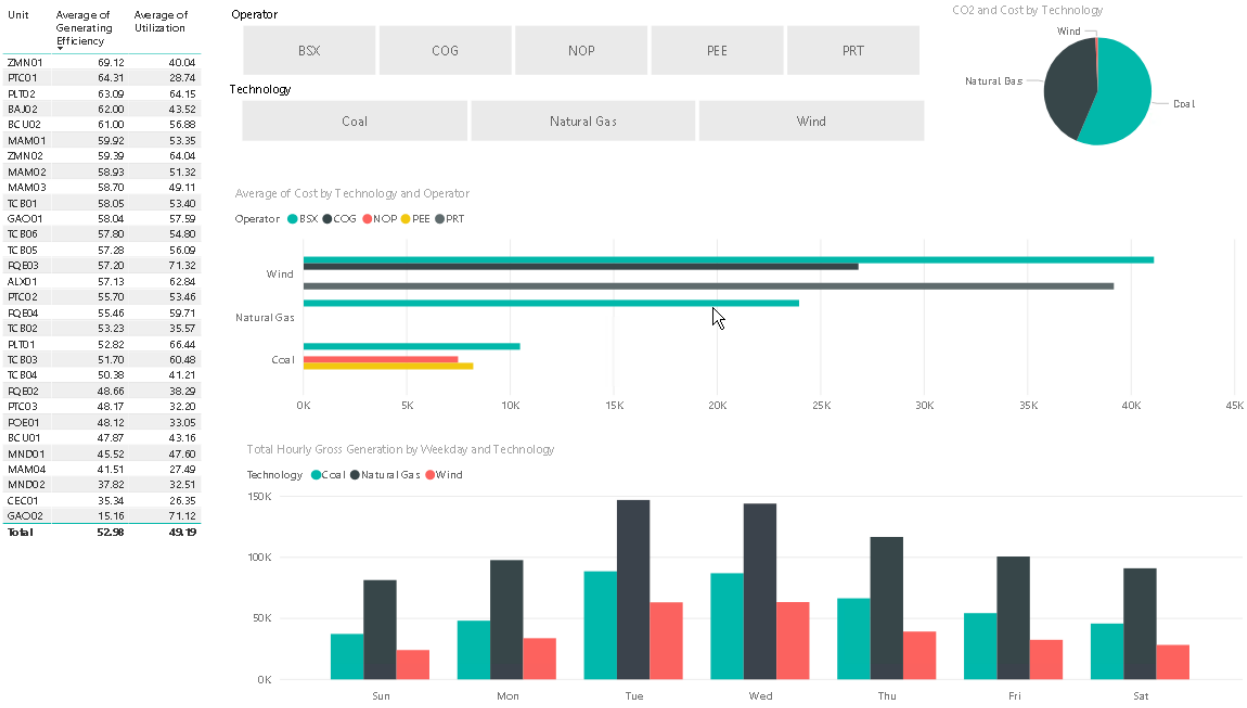
10.3 Configuring the Visualizations

Now we will add visuals to the report to convey useful information about the generating units.

10.3.1 Exercise – Build the Report

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective: Build an interactive Report comparing KPIs for different generation technologies and operators.

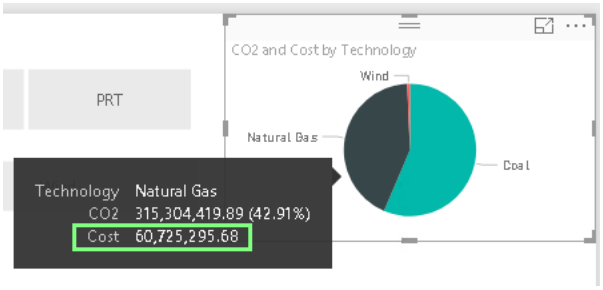


Approach:

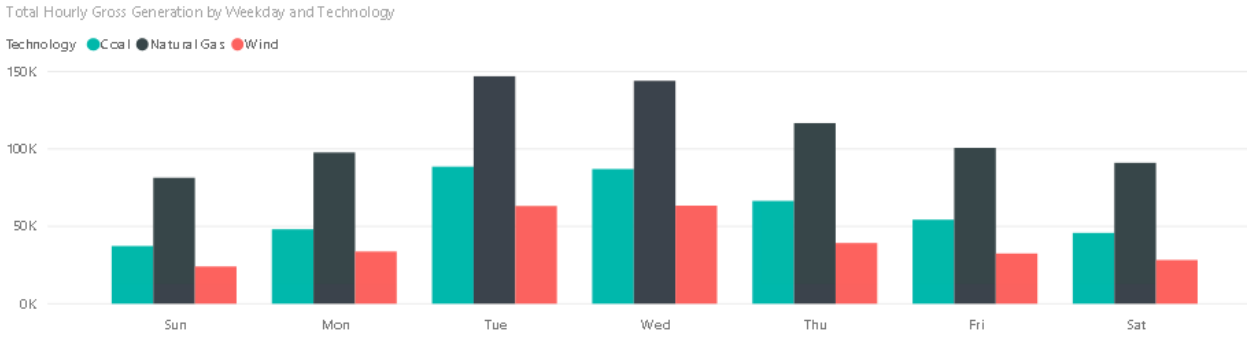
- Add a **Table** showing Average Generating Efficiency and Average Utilization by Unit

Unit	Average of Generating Efficiency	Average of Utilization
ZMN01	69.12	40.04
PTC01	64.31	28.74
PLT02	63.09	64.15
BAJ02	62.00	43.52
BCU02	61.00	56.88
MAM01	59.92	53.35
ZMN02	59.39	64.04
MAM02	58.93	51.32
MAM03	58.70	49.11
TCB01	58.05	53.40
GAO01	58.04	57.59
TCB06	57.80	54.80
TCB05	57.28	56.09
PQB03	57.20	71.32
ALX01	57.13	62.84
PTC02	55.70	53.46
PQB04	55.46	59.71
TCB02	53.23	35.57
PLT01	52.82	66.44
TCB03	51.70	60.48
TCB04	50.38	41.21
PQB02	48.66	38.29
PTC03	48.17	32.20
POE01	48.12	33.05
BCU01	47.87	43.16
MND01	45.52	47.60
MAM04	41.51	27.49
MND02	37.82	32.51
CEC01	35.34	26.35
GAO02	15.16	71.12
Total	**52.98**	**49.19**

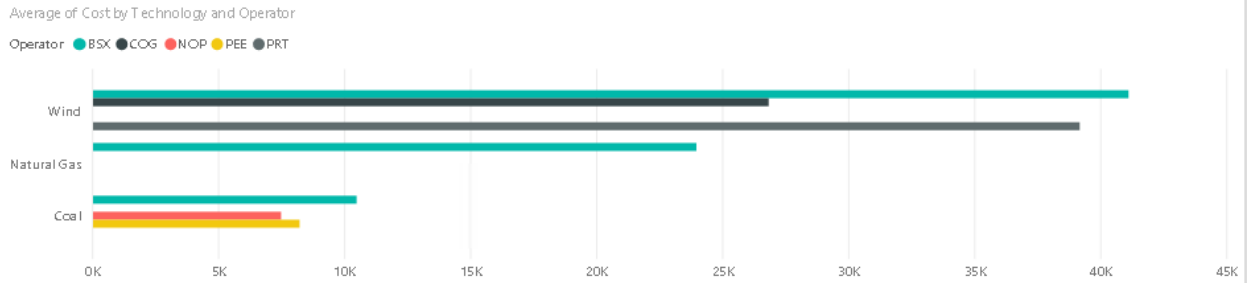
- Add a **Pie Chart** showing how the **C02 emissions** from each generation technology contribute to the whole. Add a **Tooltip** that shows the **Cost** when the user hovers over the Pie Chart



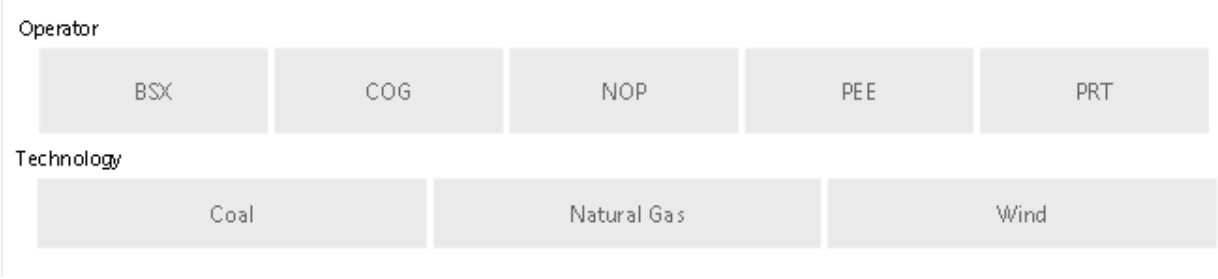
- Add a **Clustered Column Chart** showing the Sum of Total Hourly Gross Generation with Technology as the Legend and Weekday as the Axis



- Add a **Clustered Bar Chart** showing the Average Hourly Cost with Operator as the Legend and Technology as the Axis.



- Add **Slicers** for the Operator and Technology



- Optionally improve the look and feel of the report through the use of formatting. Bump up the font sizes, adjust column names and titles, etc.

10.4 Adding Event Frame context to the report

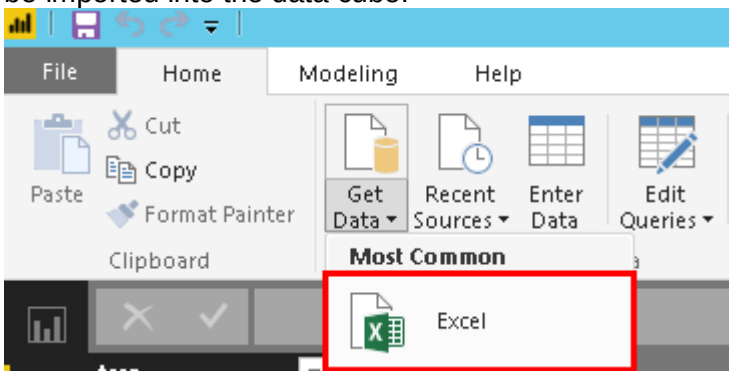
Now you are going to create another list within Fleet Generation report, which includes interactive view of the carbon footprint on a US map. This part is considered to be more solo exercise with less hints and step by step approaches.

Objective:

- Determine the carbon footprint of each unit and display on a US map. Also create a report to analyze downtime (Inactivity) events.

Approach:

- Create a new Sheet in the Fleet Generation Report (the imported tables will be re-used)
- Geospatial information for all units in Fleet Generation is located in **C:\Class\Final Exercise\Unit Coordinates.xlsx**. This data will need to be imported into the data cube.



- To get the Inactivity Events, you can either use PI SQL Client or PI Integrator for BA (both examples and hints you can find in hints section below in **"Hints: Event Frames with PI Integrator for BA and PI SQL Client",** It is explained based on a different event frame template specifically).
    - You need a column to form the relationship between the Unit Specifications table and the Inactivity Event Frames, it's probably easiest to join on Unit Name (GAO01, etc).
    - Extract Event frames for the last 7 days
    - **If using PI Integrator for BA** to publish the event frames, it's probably easiest to add the Unit Name to the Event Frame template.
    - **If using PI SQL Client**, Unit Name is the primary referenced element. Start with the ft_TransposeEventFrameSnapshot_Inactivity predefined query and modify it as necessary.

- Import the Inactivity events for the last 7 days using whichever method you prefer.

- Create the table relationships (should happen automatically if all columns are named Units).
    - Between the Unit Specifications table and the longitude/latitude table
    - Between the Unit Specifications table and the Inactivity query results

- Insert a map within the client to display the location of each of the units and the associated total hourly carbon emissions.

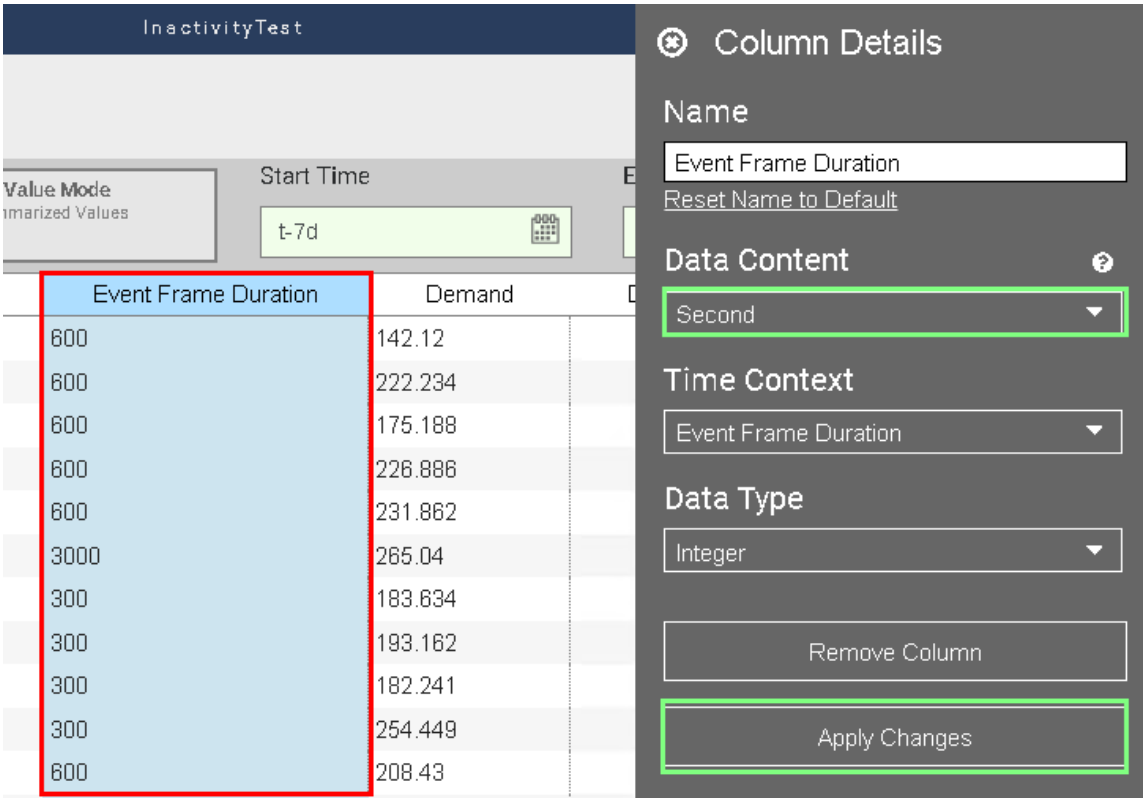
- Insert a table showing the number of downtime events (Inactivity Event Frames) and average duration of event frames for each unit. Add the Average Utilization to the same table.

- Configure the report in such a way that the Table relationships are tested. Use data from multiple tables in the same Visual.

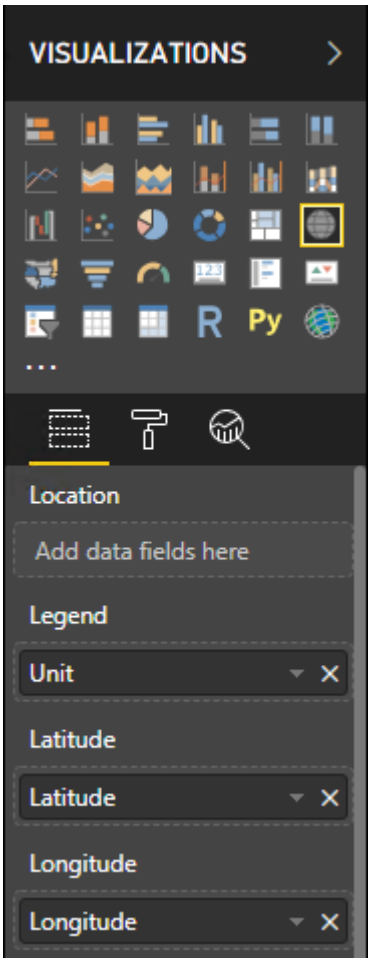
- Customize the display to make it more user friendly for later use and report generation. Improve the formatting and add slicers.

Hints:

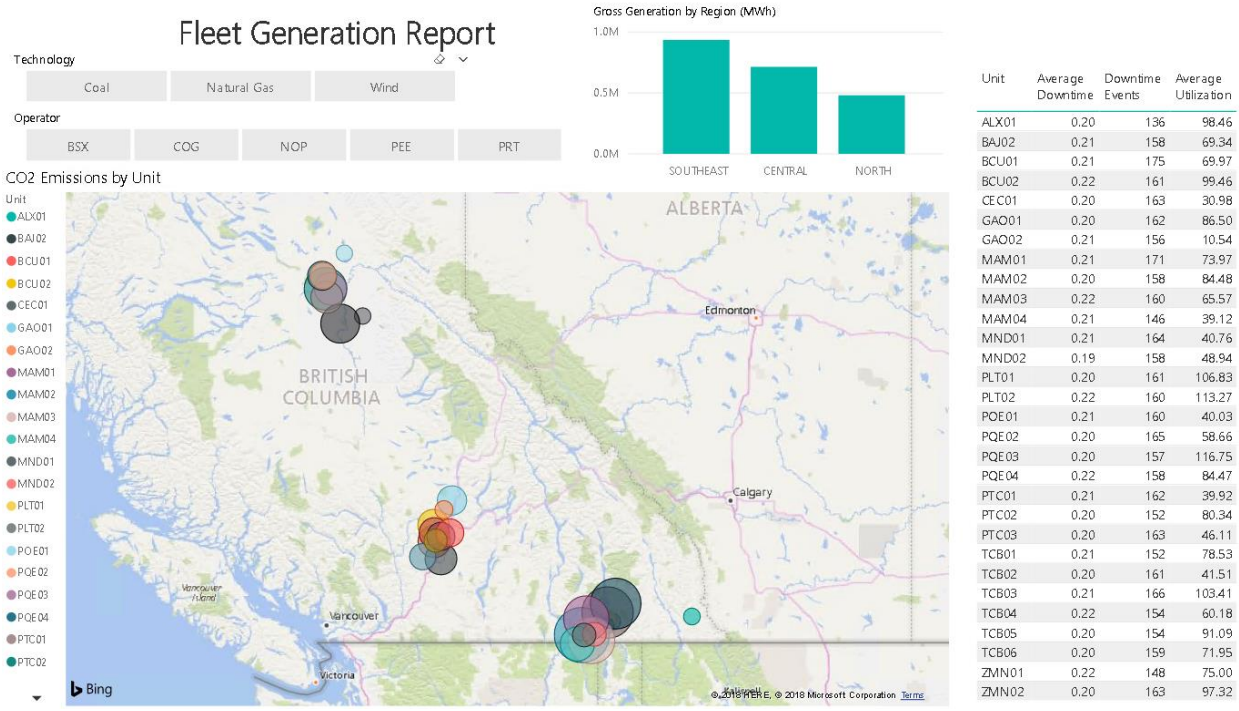
- If using PI Integrator for BA to publish the Inactivity Event frames, the Data Context must be set to Second or else it will round to the nearest whole hour (which will always be zero).



- Use the ordinary map, not the ESRI one. Drag and drop latitude and longitude from the table that was imported from the Unit Coordinates.xlsx spreadsheet.



A sample of what the report could look like:



Unit	Average Downtime	Downtime Events	Average Utilization
ALX01	0.20	136	98.46
BAJ02	0.21	158	69.34
BCU01	0.21	175	69.97
BCU02	0.22	161	99.46
CEC01	0.20	163	30.98
GAO01	0.20	162	86.50
GAO02	0.21	156	10.54
MAM01	0.21	171	73.97
MAM02	0.20	158	84.48
MAM03	0.22	160	65.57
MAM04	0.21	146	39.12
MND01	0.21	164	40.76
MND02	0.19	158	48.94
PLT01	0.20	161	106.83
PLT02	0.22	160	113.27
POE01	0.21	160	40.03
PQE02	0.20	165	58.66
PQE03	0.20	157	116.75
PQE04	0.22	158	84.47
PTC01	0.21	162	39.92
PTC02	0.20	152	80.34
PTC03	0.20	163	46.11
TCB01	0.21	152	78.53
TCB02	0.20	161	41.51
TCB03	0.21	166	103.41
TCB04	0.22	154	60.18
TCB05	0.20	154	91.09
TCB06	0.20	159	71.95
ZMN01	0.22	148	75.00
ZMN02	0.20	163	97.32

10.5 Hints: Event Frames with PI Integrator for BA and PI SQL Client

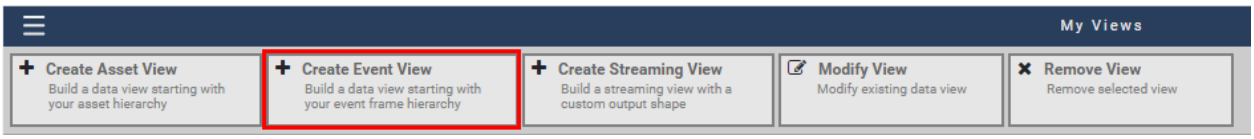
Objective:

Create an Event View with PI Integrator for BA (first approach)

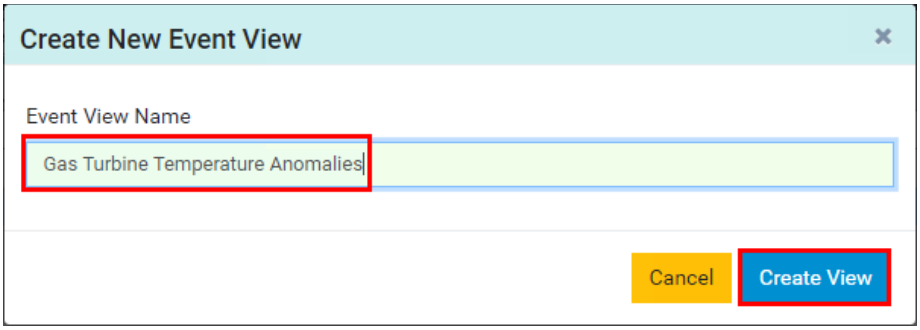
Approach:

We'll create an event view for Gas Turbine Temperature Anomaly events. Open Chrome and navigate to https://pisrv01.pischool.int:444/.

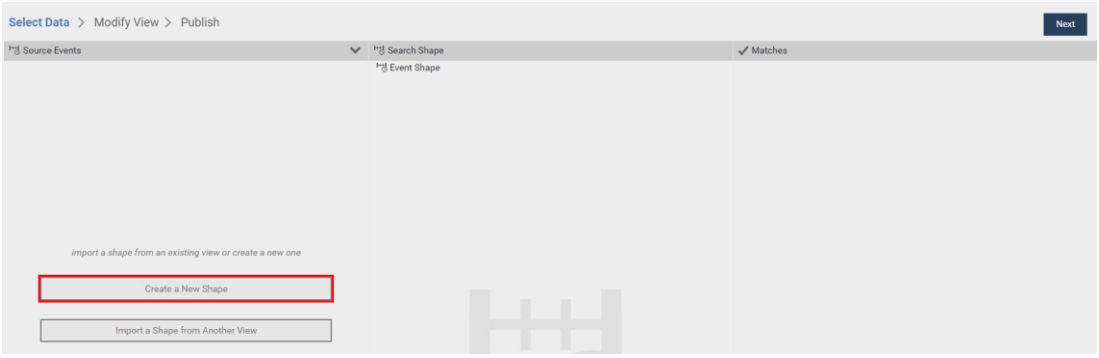
Create an Event View:



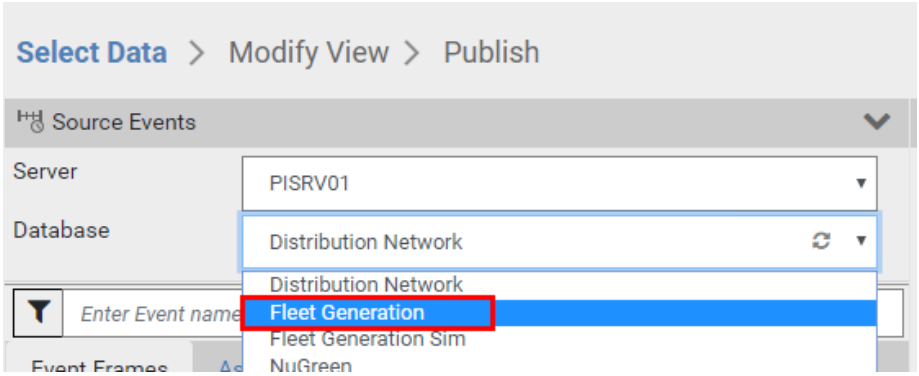
Name it Gas Turbine Temperature Anomalies and click Create View:



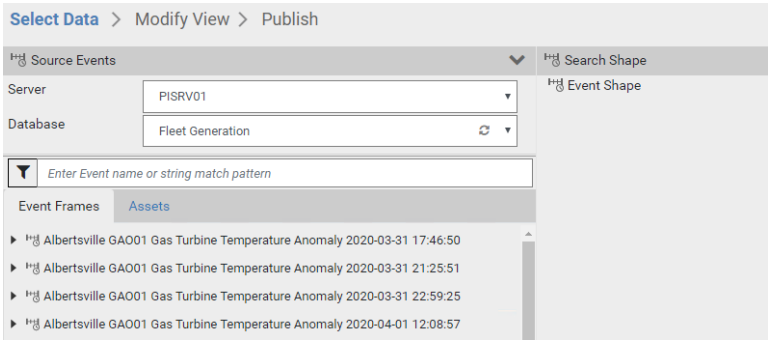
Create a new Shape:



Point at the Fleet Generation database, since that's where the Event Frames are:

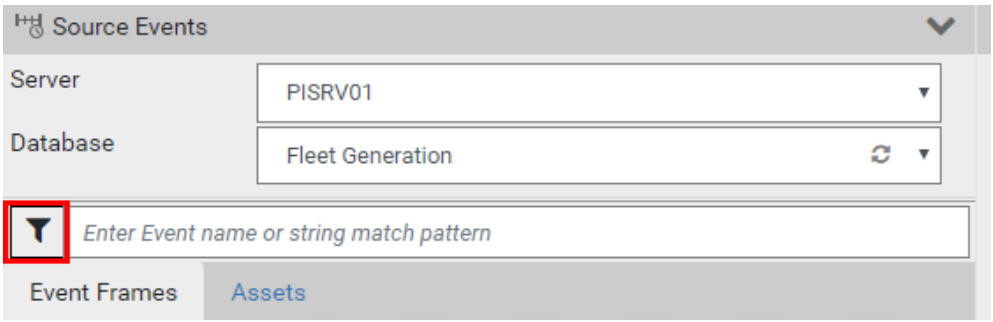


Recent Event Frames should show up in the preview. **Everyone's data is random, so everyone's preview will look different**:

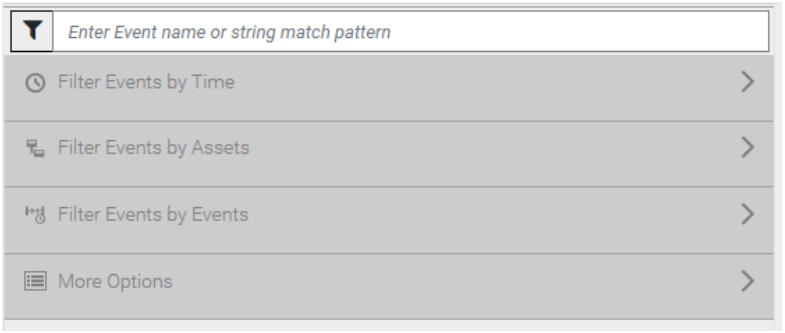


You just need to find an event frame of the proper type in order to start building the shape, but let's look at the filtering options which will allow you to narrow down the search.

Click the filter icon:

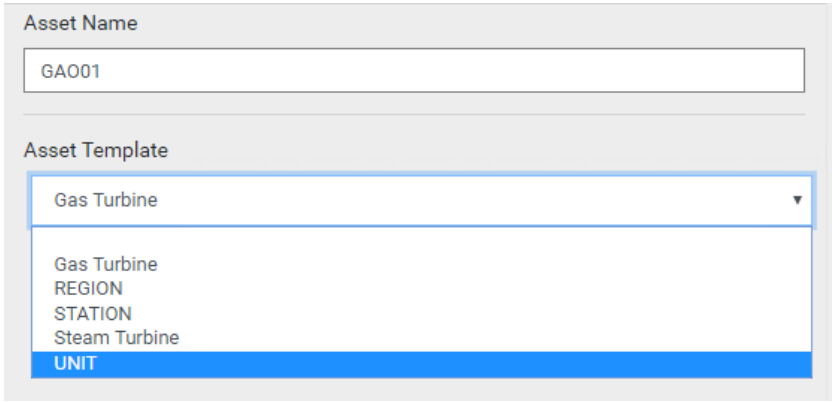


These settings allow filtering the preview and will help you find the event you're looking for. On a production system there could be over a million Event Frames spanning many different types.

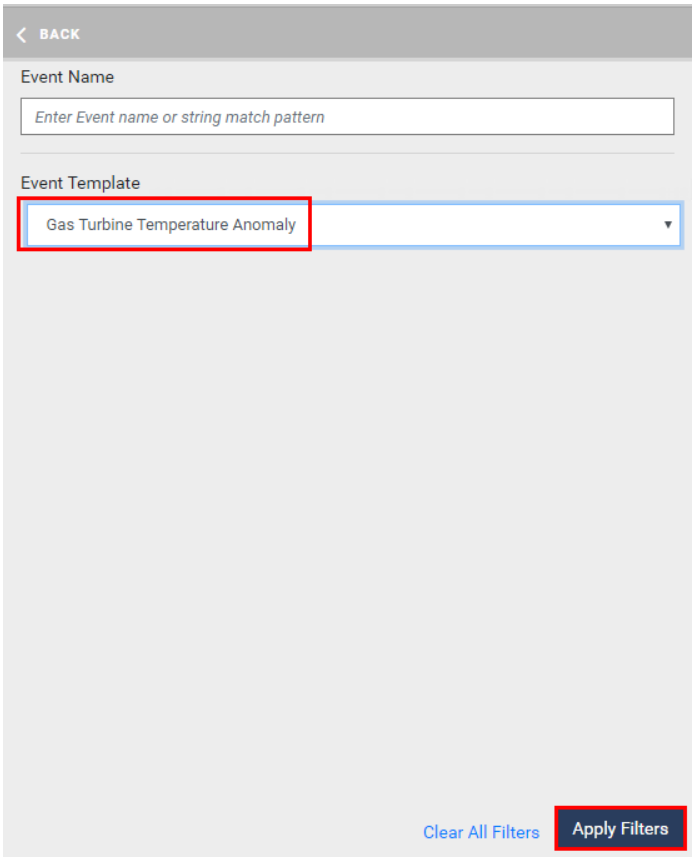


Filter Events by Time is pretty straightforward.

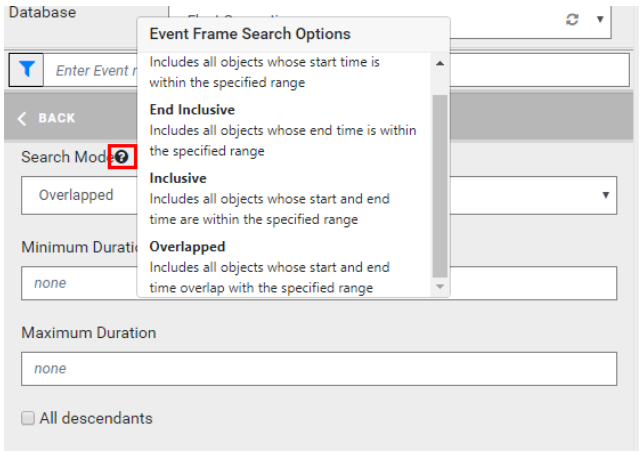
Filter Events by Asset allows you to filter by primary referenced Element (the Element in AF whose Analysis generated the event). **No need to set anything here, this is just an example.**



Filter Events by Events allows you to filter by Event Frame name or Event Frame Template. **Select Gas Turbine Temperature Anomaly** as the Event Template and click **Apply Filters** to filter out the Inactivity Events:

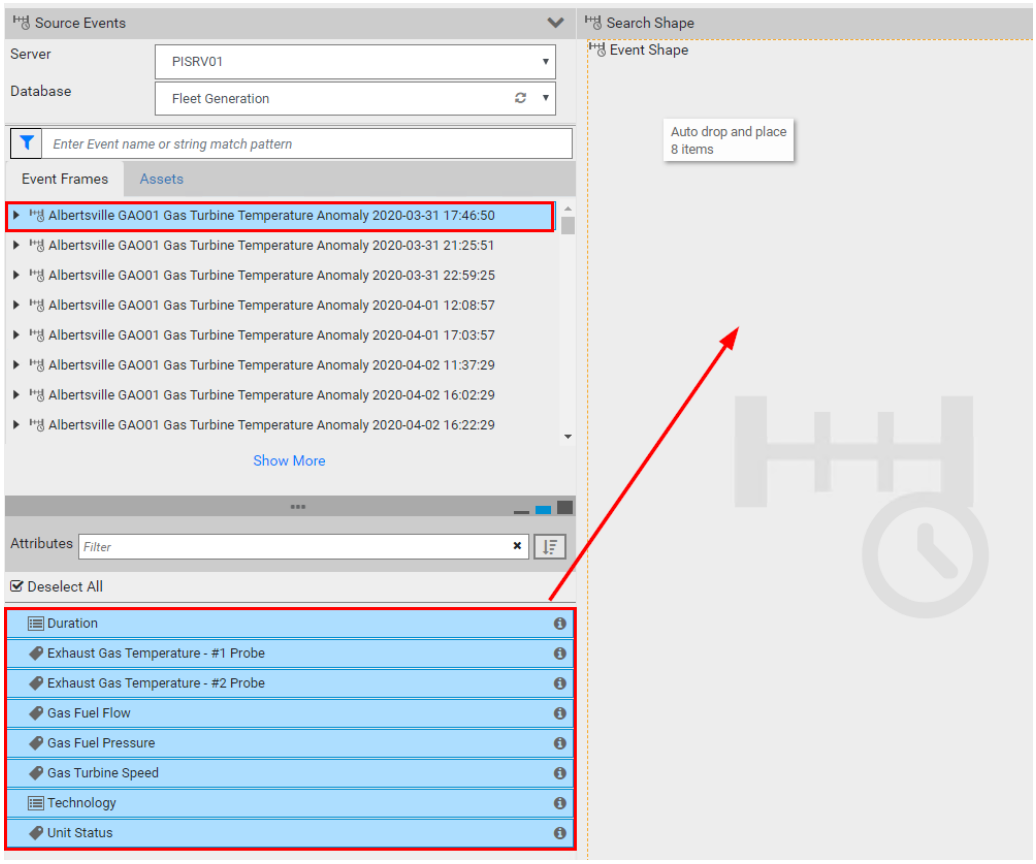


And while we won't set anything, let's take a look at **More Options**. Click the question mark to see explanations for the **Search Mode**. Minimum Duration and Maximum Duration are self-explanatory. **All descendants** applies to hierarchical event frames, which we don't have.

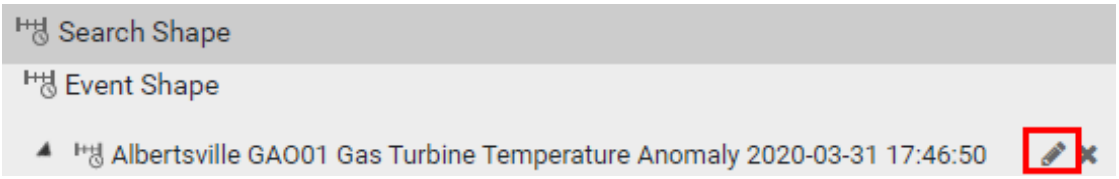


If you can't find Gas Turbine Temperature Anomaly events, it's possible that they weren't generated in one of the previous chapters. You may have to go back to the Event Frame Generation chapter and complete the exercises.

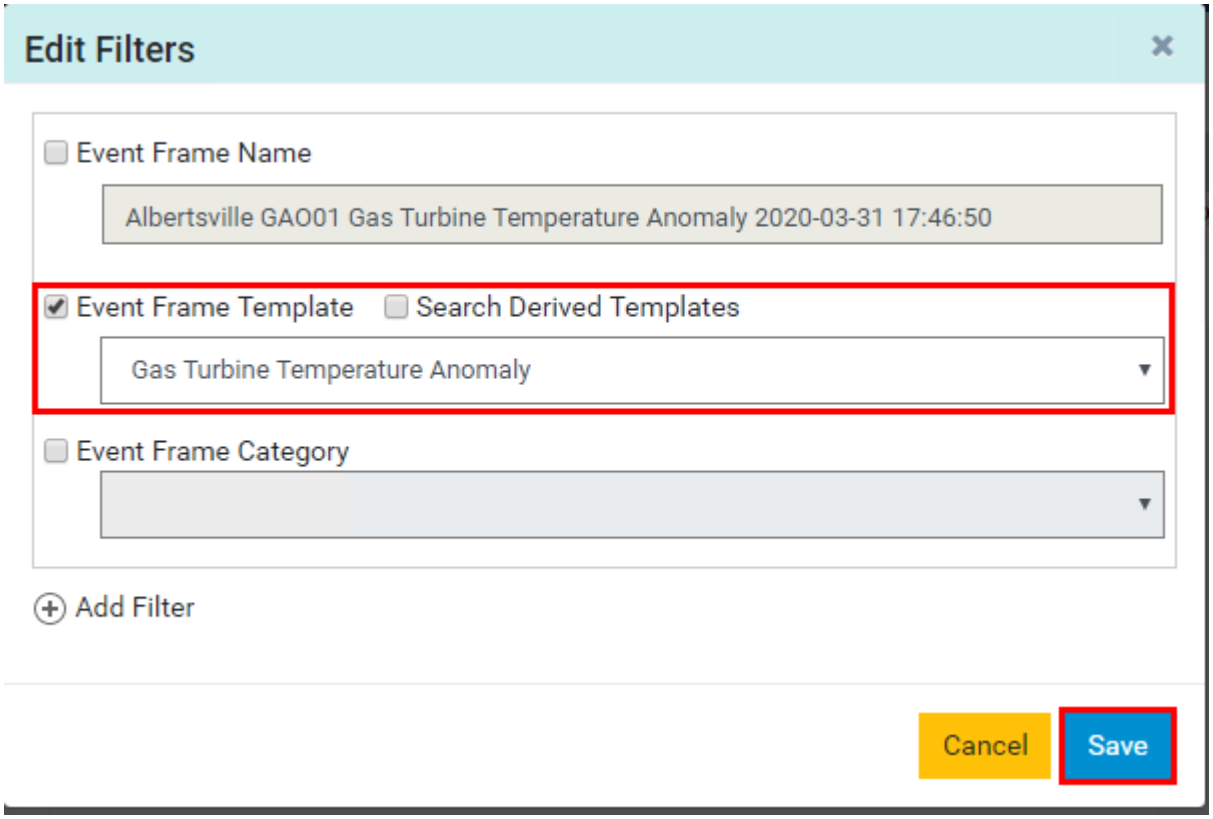
Once you see some events, you can start to configure the Shape. Click one of the Events, then drag and drop all Attributes:



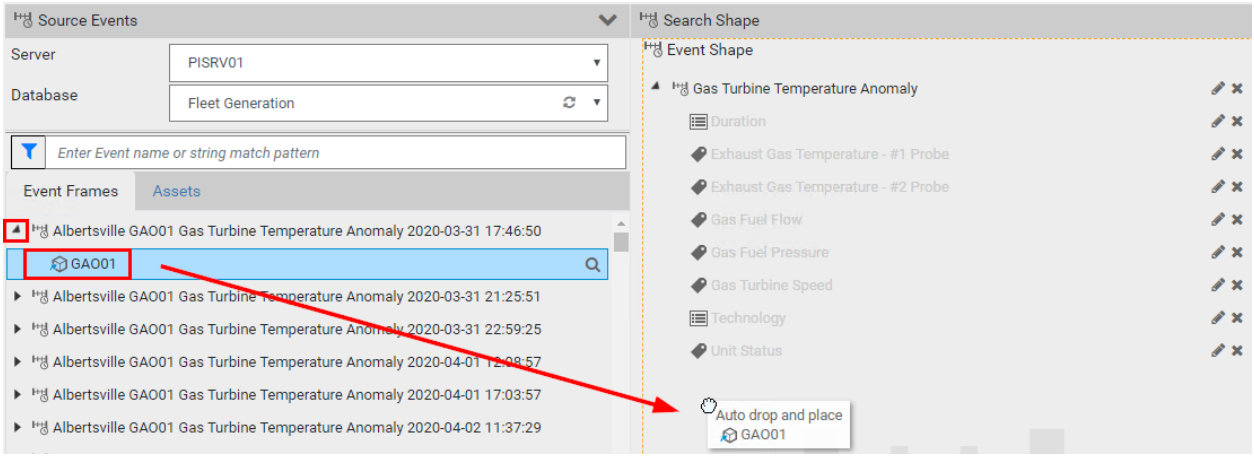
Edit the Shape:



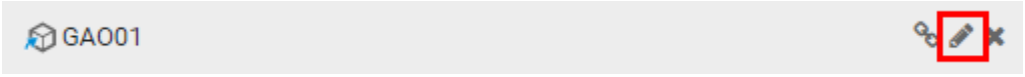
Uncheck the box next to Event Frame Name and match Event Frames by Template then Save.



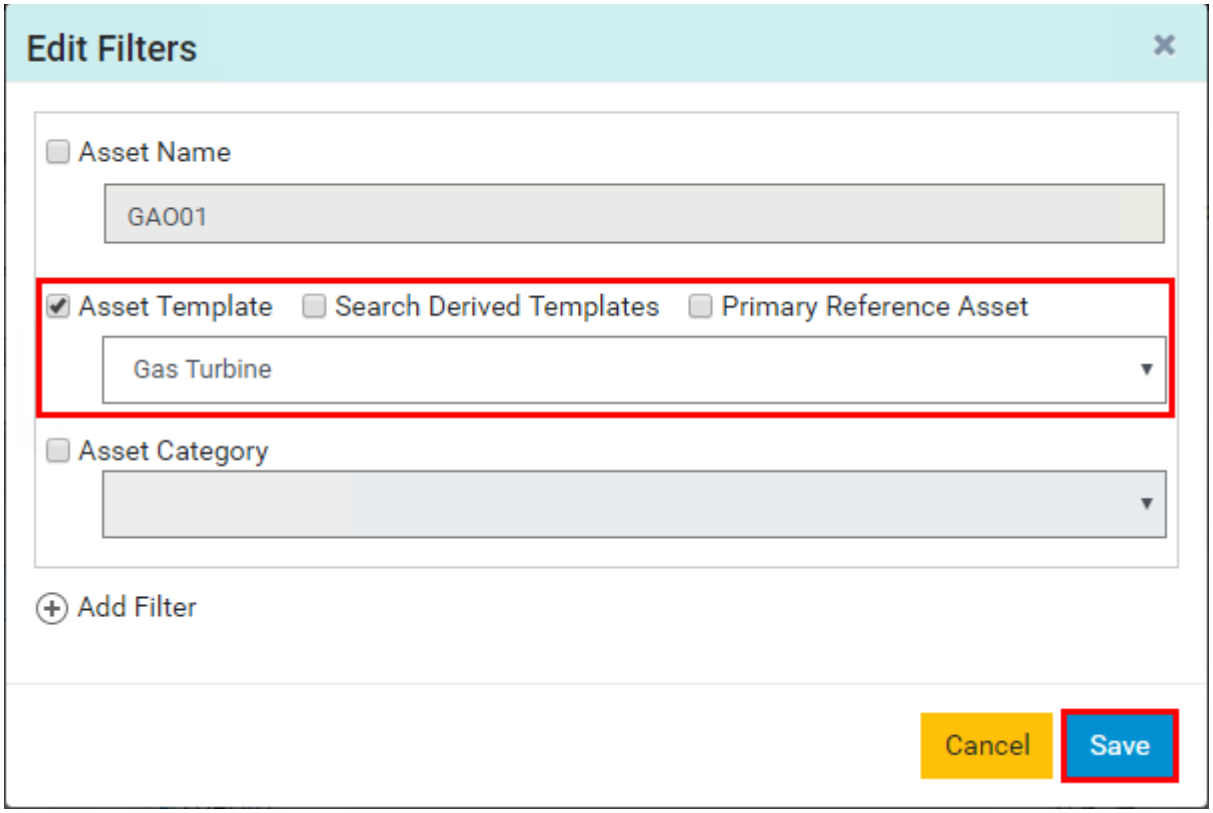
Expand the Event Frame and drag and drop the Gas Turbine to the shape configuration:



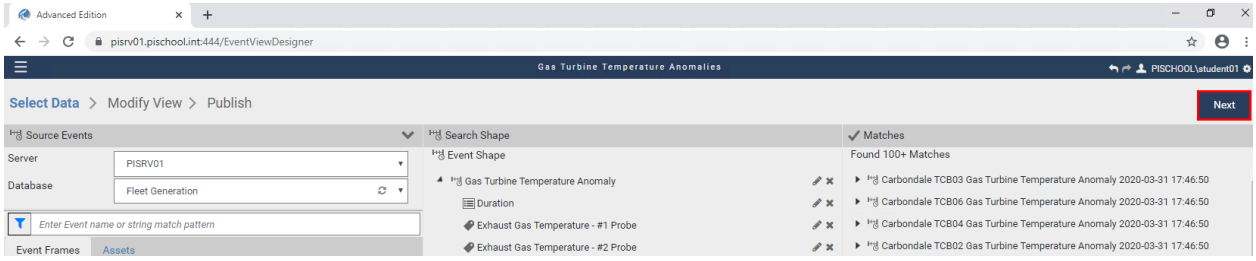
Edit the Gas Turbine object:



Filter by Asset Template, click Save:

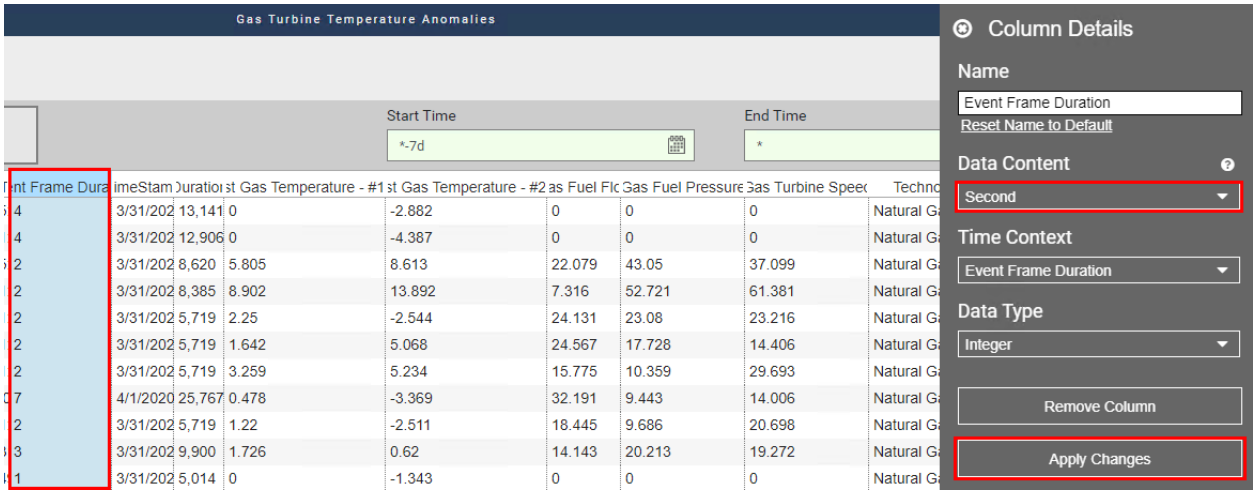


You should see a bunch of matches. Go to the Next screen:

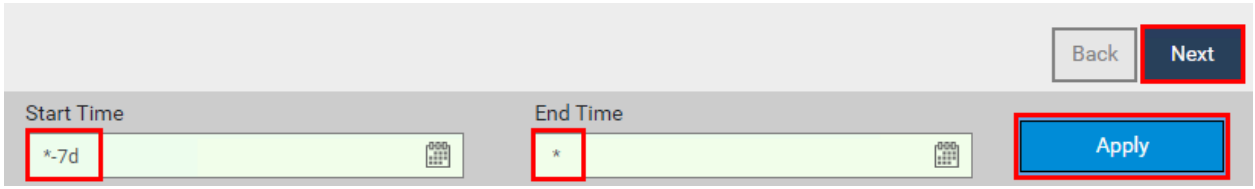


Change the Event Frame Duration Data Content to Second otherwise it will be displayed as a round number of hours.

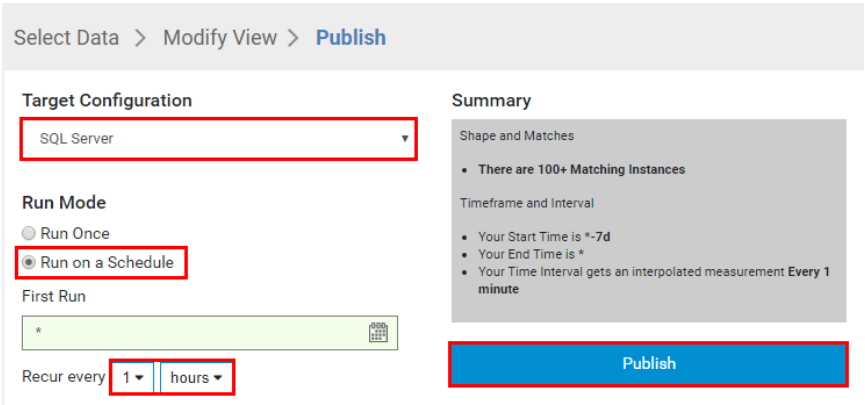
Be sure to click Apply.



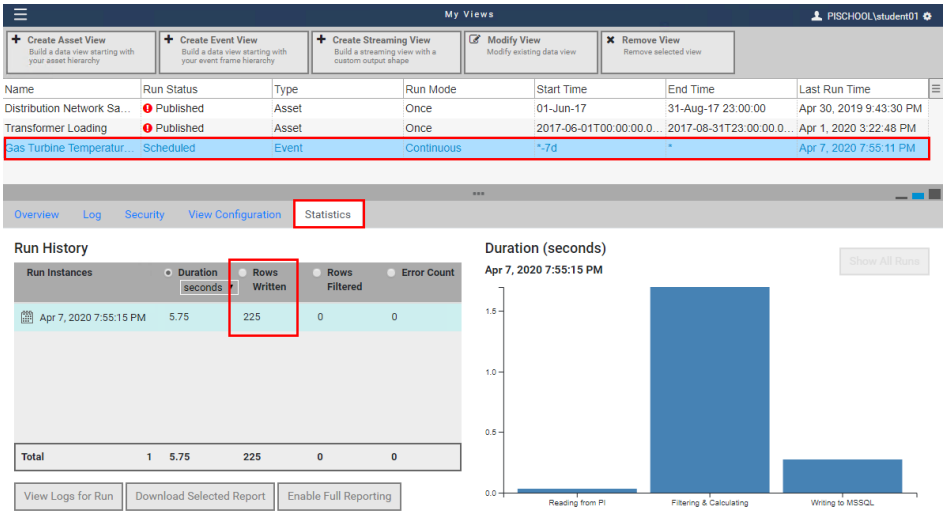
Change the Start Time to *-7d and end time to *, click Apply, then move to the Next Screen:



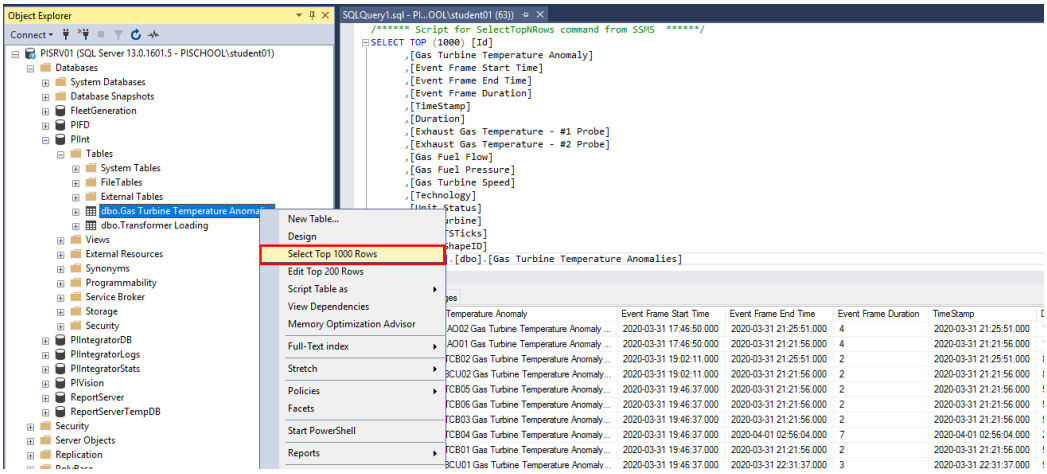
Select SQL Server as the target and have it run on an hourly schedule to keep the Event Frames current, then click Publish.



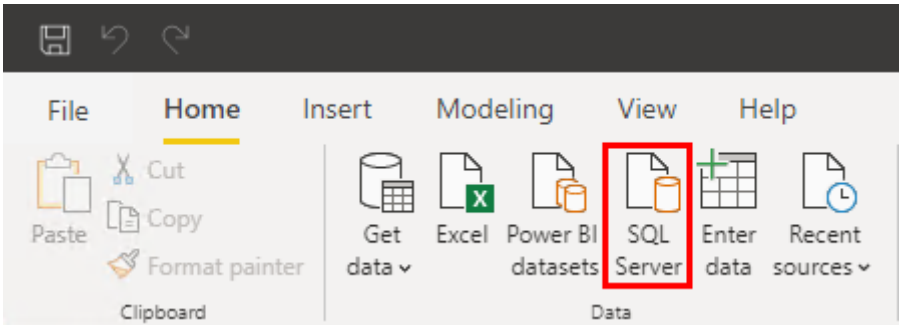
Check the statistics to confirm that Rows were written:



Use **SQL Server Management Studio** to confirm that Event Frames were written to the SQL Server table:



Remember that to import to Power BI from SQL Server, you use the SQL Server provider:



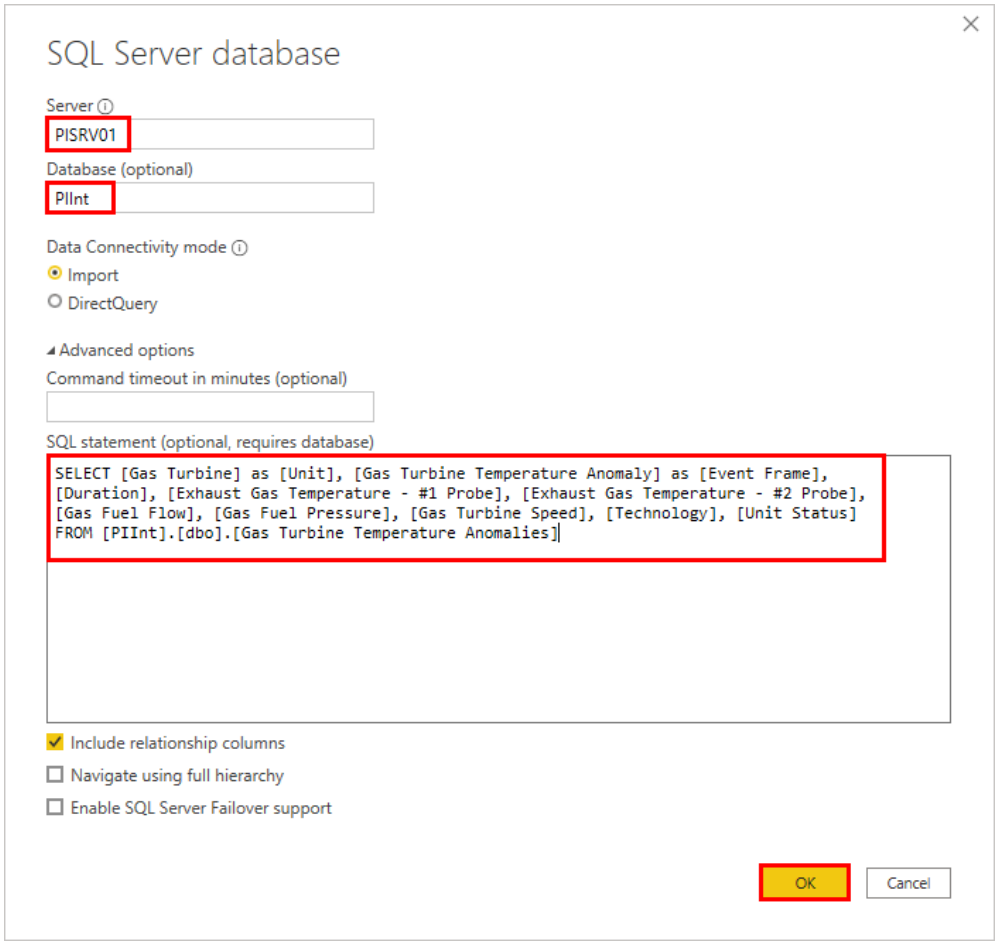
And then you have 2 options:

1. Just enter the Server and browse the available databases and tables
2. Enter the Server, Database, and a SQL Query

Using option 2 for example, you could clean up the query in SQL Server Management studio to the following in SQL Server Management Studio by starting with the Select Top 1000 Rows query:

```
SELECT [Gas Turbine] as [Unit], [Gas Turbine Temperature Anomaly] as [Event Frame],
[Duration], [Exhaust Gas Temperature - #1 Probe], [Exhaust Gas Temperature - #2 Probe],
[Gas Fuel Flow], [Gas Fuel Pressure], [Gas Turbine Speed], [Technology], [Unit Status]

FROM [PIInt].[dbo].[Gas Turbine Temperature Anomalies]
```

The configuration would look like:



And then Load.

10.5.1 Directed Activity (optional) – Create an Event Frame Query with PI SQL Client (second approach)

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

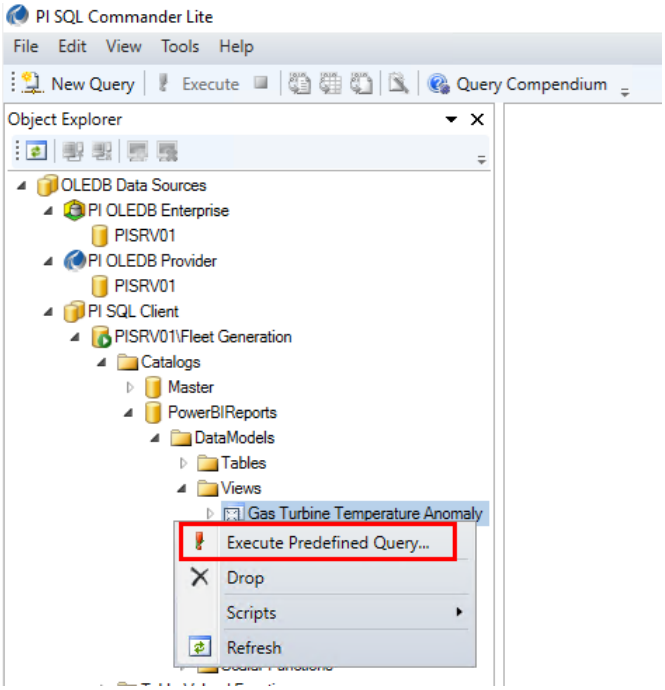
Create an Event Frame Query with PI SQL Client

Approach:

We'll get the same Gas Turbine Temperature Anomaly events as in the previous exercise, but using PI SQL Client.

Open PI SQL Commander and Connect to the Fleet Generation database.

Recall that Template-Specific Data Models for both types of Event Frames were created in a previous exercise. Execute the predefined query for the Gas Turbine Temperature Anomaly View:



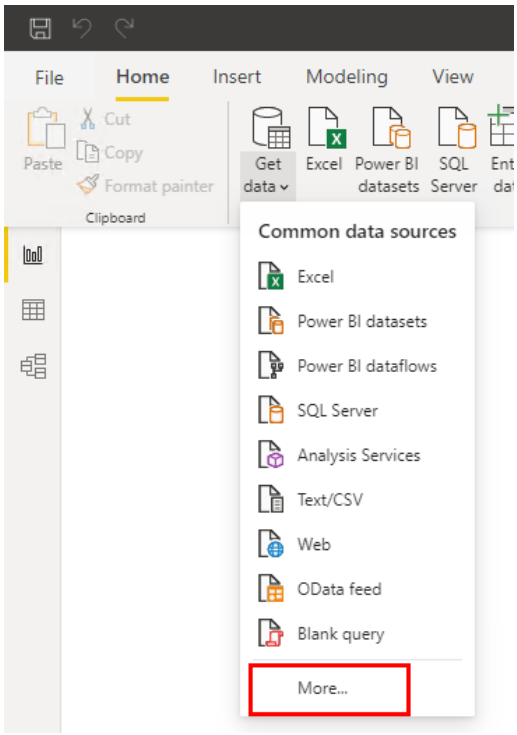
We need to JOIN to the EventFrame table in order to get to the PrimaryReferencedElement (Unit Name). In the final Power BI report, the Table relationships are based on Unit. The query becomes:

```
SELECT ef.[PrimaryReferencedElement] as [Unit], v.*
FROM [PowerBIReports].[DataModels].[Gas Turbine Temperature Anomaly] v
JOIN
[Master].[EventFrame].[EventFrame] ef ON ef.ID = v.EventFrameID
```

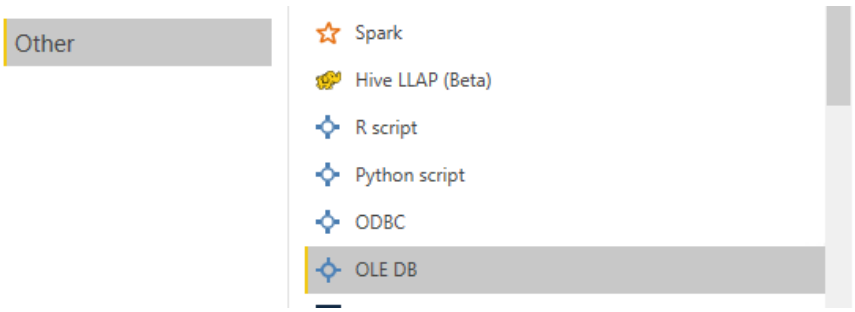
Modify the SELECT statement to remove all unnecessary columns, which means explicitly selecting all the necessary columns. The result is the below query:

```
SELECT ef.[PrimaryReferencedElement] as [Unit], v.[Name] as [Event Frame],
v.[Duration], v.[Exhaust Gas Temperature - #1 Probe],
v.[Exhaust Gas Temperature - #2 Probe], v.[Gas Fuel Flow],
v.[Gas Fuel Pressure], v.[Gas Turbine Speed],
v.[Technology], v.[Unit Status]
FROM [PowerBIReports].[DataModels].[Gas Turbine Temperature Anomaly] v
JOIN
[Master].[EventFrame].[EventFrame] ef ON ef.ID = v.EventFrameID
```

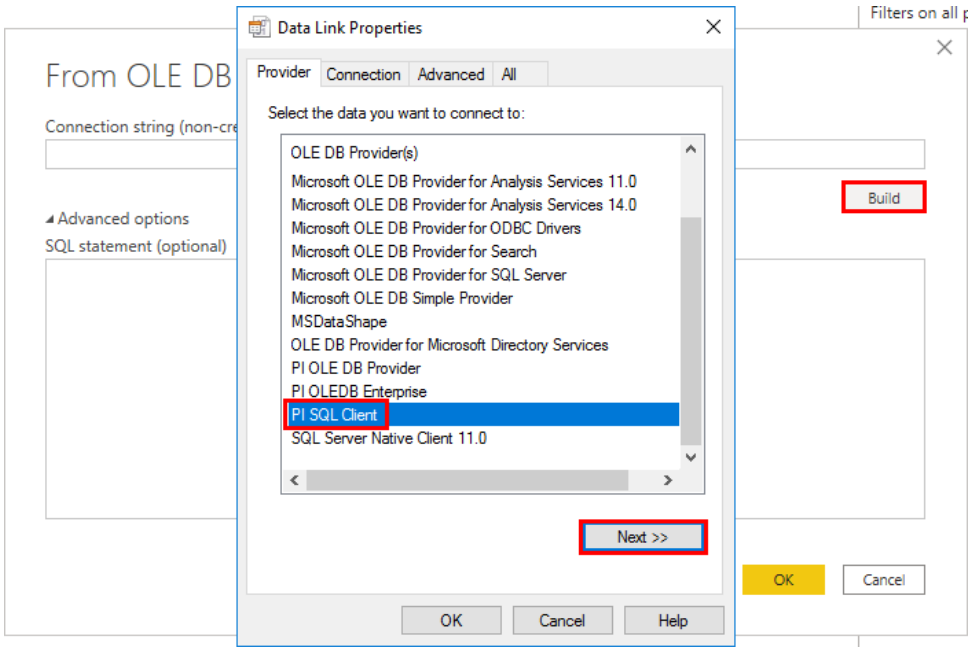
And then Importing to Power BI is accomplished using **Get data -> More**:



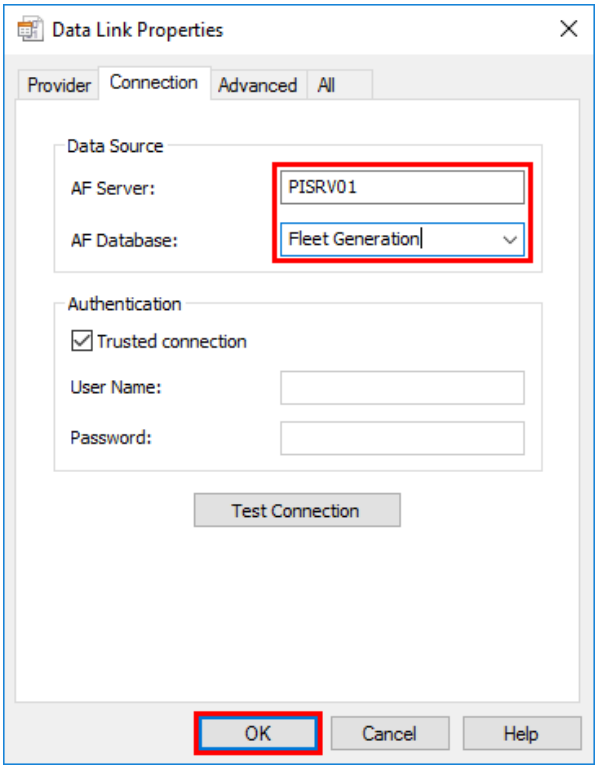
Other -> OLE DB -> Connect:



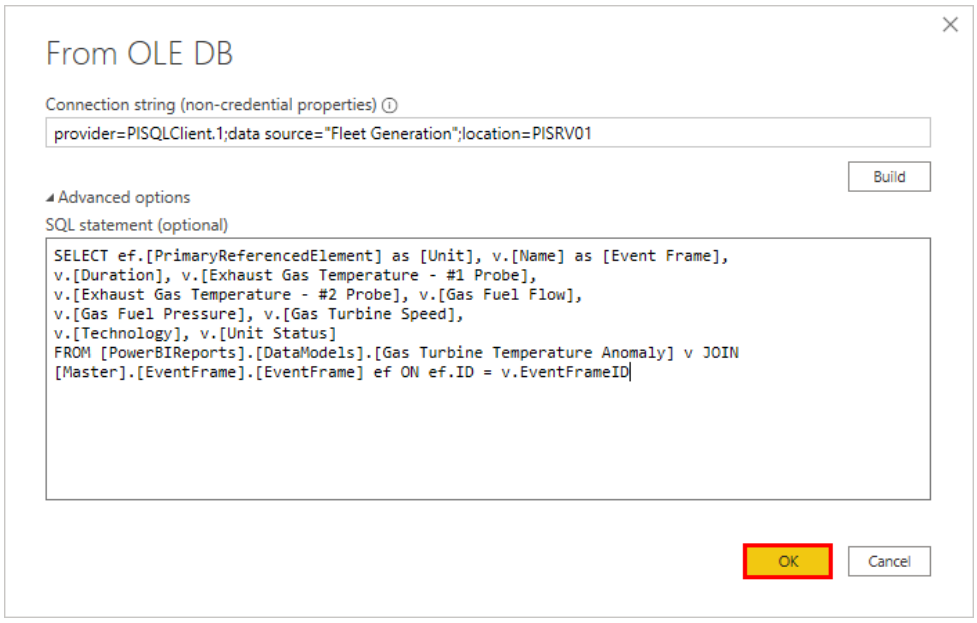
Build -> PI SQL Client -> Next:



PISRV01 -> Fleet Generation -> OK:



Paste in the query under Advanced options, OK:



And then Load.

Appendix A Self – Paced topic. PI SQL Client Queries

SQL stands for Structured Query Language. SQL is an American National Standards Institute (ANSI) definition for the language used to communicate with relational database systems. It is used by virtually all relational databases in the world today. (Even the PI Data Archive has a SQL Subsystem that can act as a translator to make it “look” like a relational database). SQL Commands are often called “**SQL Statements**.” They can be executed interactively or as stored procedures.

The good part is that it is a standard and that every relational database you encounter will understand it. There is no need to learn many languages. However, there is a down side. Most databases have unique extensions and/or syntaxes that are unique to those systems.

To give a simple example, when passing dates into Access you use pound signs (#) for surrounding dates. On the other hand, in SQL Server you need to use apostrophes (').

Access: [...] WHERE dtColumn >= #2001-11-05#

SQL Server: [...] WHERE dtColumn >= '20011105'

A SQL result set is a set of rows from a database, as well as meta-information about the query such as the column names, the data types and sizes of each column. Depending on the database system, the number of rows in the result set may or may not be known. Usually, this number is not known up front because the result set is built on-the-fly.

This flexibility allows for complex queries to be constructed and saved to return a very specific subset of information from the AF Database that would be either too cumbersome or impossible through the likes of PI System Explorer or PI Datalink.

10.6 Dissecting the Syntax

A common SQL syntax starting command is **SELECT** which is used to query the database. The data retrieved from the statement is based on the criteria specified in the SELECT statement.

Following the **SELECT** command identifies the columns to be selected from the tables(s).

SELECT * - retrieves all the columns from the table being referenced.

SELECT column1, column2, column3 – retrieves 3 columns of the table being referenced.

The **FROM** command identifies the first (or perhaps only) table being queried.

SELECT * FROM tablename – retrieves all the columns from tablename.

SELECT column1, column2, column3 FROM tablename – retrieves all data for the 3 columns of tablename.

The **WHERE** command contains criteria to filter the data being retrieved.

The conditional operators include:

equal (=)
greater than (>)
less than (<)
greater than or equal (>=)
less than or equal (<=)
not equal to (<>)
LIKE (which is a pattern matching operator)

Note: If the conditional clause is set to compare to text, the text value is encased in single quotes ('text').

SELECT * from tablename WHERE column1 = 5

Retrieves only rows where column1 has a value equal to the number 5.

AND and OR statements

- **AND** indicates both statements must be TRUE for the row to be returned when the query is executed.
    - SELECT column1, column2, column3 from tablename WHERE column1 = 5 **and** column2 = 'junk'
    - Retrieve only rows where column1 has a value equal to the number 5 **and** column2 value equals junk.
- **OR** returns data rows if either condition is met
    - SELECT column1, column2, column3 from tablename WHERE column1 = 5 **or** column2 = 'junk'
    - Retrieve only rows where column1 has a value equal to the number 5 **or** column2 value equals junk.

The **LIKE** operator is used to search for a specific pattern in a column. In conjunction with the LIKE operator a **wildcard of %** is used for comparison. The % can represent a single character or multiple characters. Another wildcard is the underscore (_) which can be used to represent a single character.

SELECT * from tablename WHERE column2 LIKE '%unk'

Retrieves rows from tablename where column2 values end with the letters 'unk'

SELECT * from tablename WHERE column2 LIKE '%un%'

Retrieves rows from tablename where column2 values contain the letters 'un'

SELECT * from tablename where column2 like '_un_'

Retrieves rows from the tablename where column 2 values only contains 4 characters and the middle two characters are un.

SELECT * from tablename WHERE column2 LIKE 'j%'

Retrieves rows from tablename where column2 values start with the letter 'j'

To work with column/table names which have special characters, such as a space, use square brackets:

If you wish to SELECT a column called *Product Orders*, enclose it in square brackets: [Product Orders]

If you're referring to a table whose full path is *Fleet Generation, Region, Station, Unit*, that must be written as [Fleet Generation].[SouthEast].[Brick Canyon].[PLT02]

Any name may be wrapped in square brackets, so when in doubt as to what constitutes a special character, wrap the name in square brackets.

PI SQL Client

The RTQP Engine via PI SQL Client became available in PI Server 2018 SP2 and is the successor to PI OLEDB Enterprise. The functionality is very similar to PI OLEDB Enterprise in that PI AF and the underlying tags can be queried using SQL statements. The main benefits over PI OLEDB Enterprise are better performance, simpler queries, and simplified software architecture.

The following video, which is part of the new PI SQL Online course, explains the history and landscape of the various PI SQL family of products. Your instructor will play the video in class.

https://www.youtube.com/watch?v=W2nj29eNseQ

PI SQL Commander

The PI SQL Client installation includes a test environment, which handles the OLE connection process and allows the user to execute queries and perform other tasks. This test environment is *PI SQL Commander Lite.*

PI SQL Commander is the user interface to assist with creating queries, transpose functions, and views against PI AF using PI SQL Client. This user interface also provides access to the legacy PI OLEDB Provider and PI OLEDB Enterprise providers.

PI SQL Commander Lite is an application to navigate a relational view of the PI System using SQL Queries that are exposed by PI ODBC. This can be used to create, edit, test and save SQL queries of PI System data.

Directed Activity – Review Predefined Queries

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Review predefined queries associated with the tables defined in PI SQL Commander.

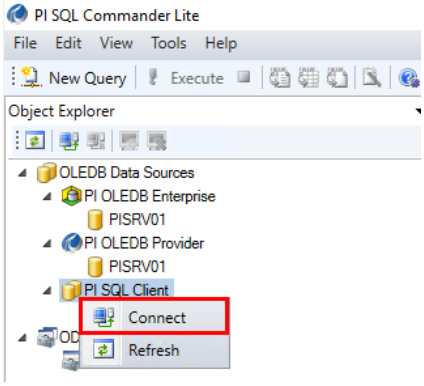
Approach:

- Open PI SQL Commander
- Navigate to the Fleet Generation Database/Catalog
- Execute a Predefined Query associated with the Element Hierarchy table.

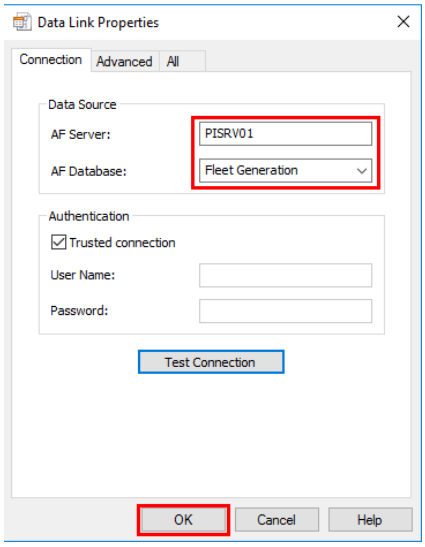
Launch PI SQL Commander -
Click Start > All Programs > PI System > PI SQL Commander Lite (64-bit)

Through PI SQL Commander, either a PI AF Server or a PI Data Archive can be accessed through SQL statements based on the item selected for connection.

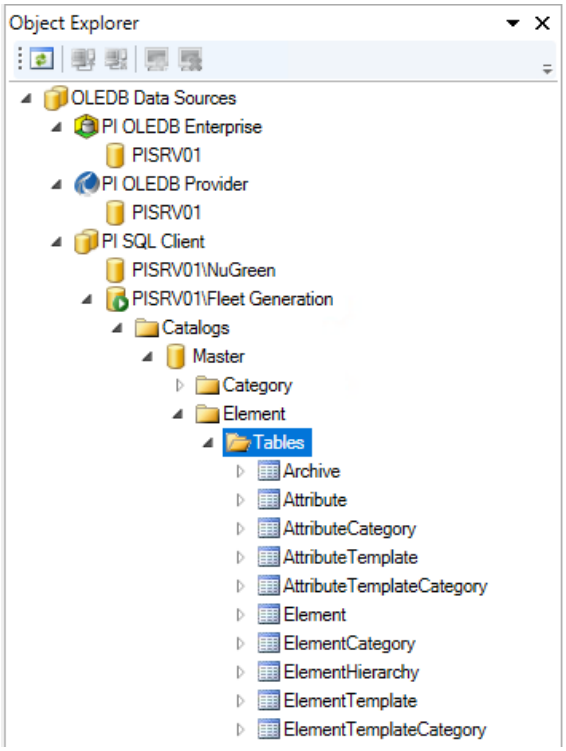
From within the Object Explorer, right click on PI SQL Client and click Connect:



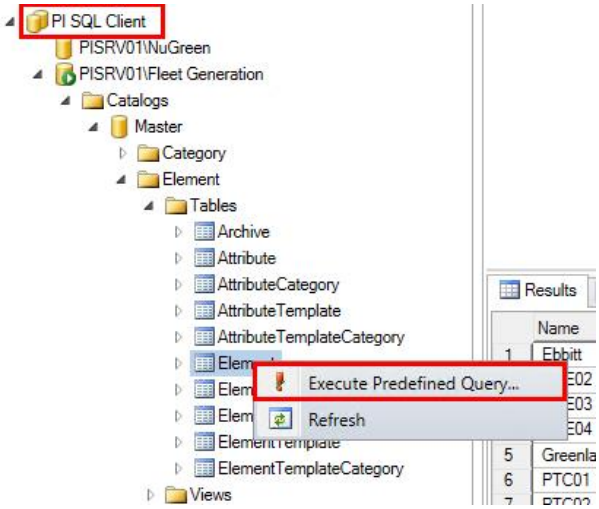
Enter PISRV01 as the AF Server and Fleet Generation as the AF Database, then click OK:



After making the connection, drill down to and expand the Master Catalog, then drill down further to the Element Tables.



Right-click on the Element Table and select **Execute Predefined Query**:



This runs a sample query, which returns the top 100 elements from the Fleet Generation AF Database:

Query1.sql - PISRV01\Fleet Generation

```
-- Element table.
-- Returns elements.
SELECT TOP 100 Name, Description, Comment, Revision, HasChildren, Template
FROM [Master].[Element].[Element]
```

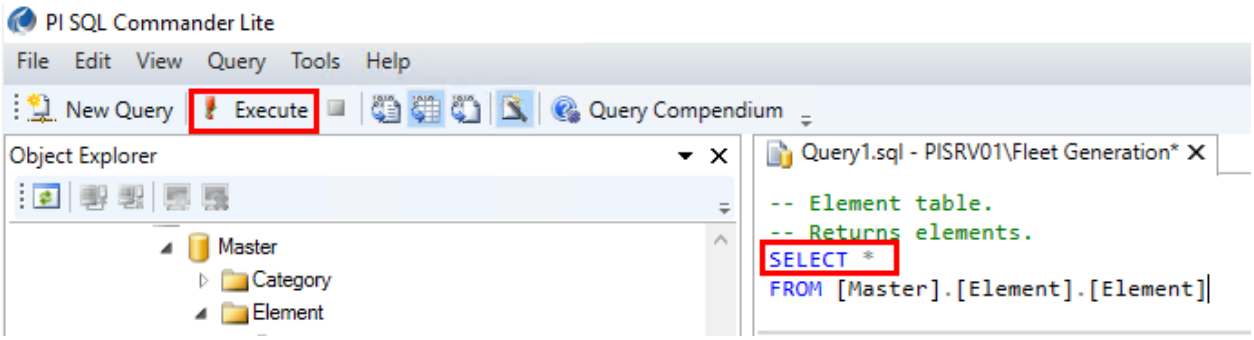
Results | Messages

	Name	Description	Comment	Revision	HasChildren	Template
1	Ebbitt	Ebbitt Station		5	True	STATION
2	PQE02	North --> Ebbitt --> PQE02		5	False	UNIT
3	PQE03	North --> Ebbitt --> PQE03		5	False	UNIT
4	PQE04	North --> Ebbitt --> PQE04		4	False	UNIT
5	Greenlawn	Greenlawn Station		5	True	STATION
6	PTC01	North --> Greenlawn --> PTC01		4	False	UNIT
7	PTC02	North --> Greenlawn --> PTC02		4	False	UNIT
8	PTC03	North --> Greenlawn --> PTC03		4	False	UNIT

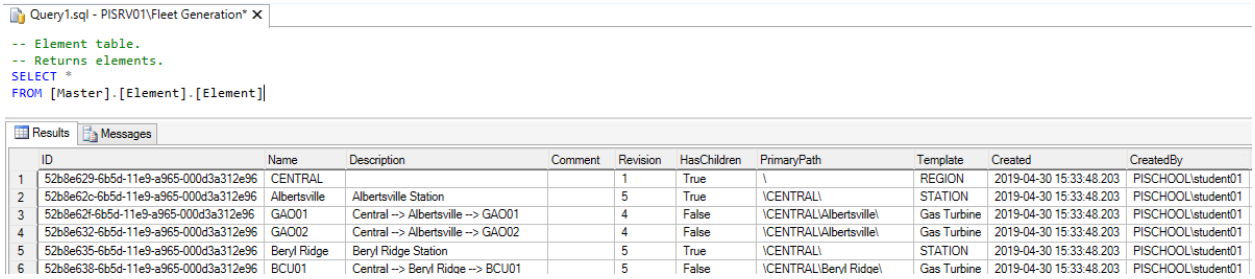
PI SQL Commander includes one sample SQL query for each Table.

Expand the Element Table and examine the available columns. It looks like the Sample Query doesn't return all available columns. Modify the query to select everything and then Execute the query:

```
SELECT *
FROM [Master].[Element].[Element]
```



Now all columns are returned, the PrimaryPath would probably come in handy.



	ID	Name	Description	Comment	Revision	HasChildren	PrimaryPath	Template	Created	CreatedBy
1	52b8e629-6b5d-11e9-a965-000d3a312e96	CENTRAL			1	True	\	REGION	2019-04-30 15:33:48.203	PISCHOOL\student01
2	52b8e62c-6b5d-11e9-a965-000d3a312e96	Albertsville	Albertsville Station		5	True	\CENTRAL\	STATION	2019-04-30 15:33:48.203	PISCHOOL\student01
3	52b8e62f-6b5d-11e9-a965-000d3a312e96	GAO01	Central --> Albertsville --> GAO01		4	False	\CENTRAL\Albertsville\	Gas Turbine	2019-04-30 15:33:48.203	PISCHOOL\student01
4	52b8e632-6b5d-11e9-a965-000d3a312e96	GAO02	Central --> Albertsville --> GAO02		4	False	\CENTRAL\Albertsville\	Gas Turbine	2019-04-30 15:33:48.203	PISCHOOL\student01
5	52b8e635-6b5d-11e9-a965-000d3a312e96	Beryl Ridge	Beryl Ridge Station		5	True	\CENTRAL\	STATION	2019-04-30 15:33:48.203	PISCHOOL\student01
6	52b8e638-6b5d-11e9-a965-000d3a312e96	BCU01	Central --> Beryl Ridge --> BCU01		5	False	\CENTRAL\Beryl Ridge\	Gas Turbine	2019-04-30 15:33:48.203	PISCHOOL\student01

Looks like we probably don't need most of these columns after all, pare it down to just the Name, HasChildren, PrimaryPath, and Template columns and **Execute** the following:

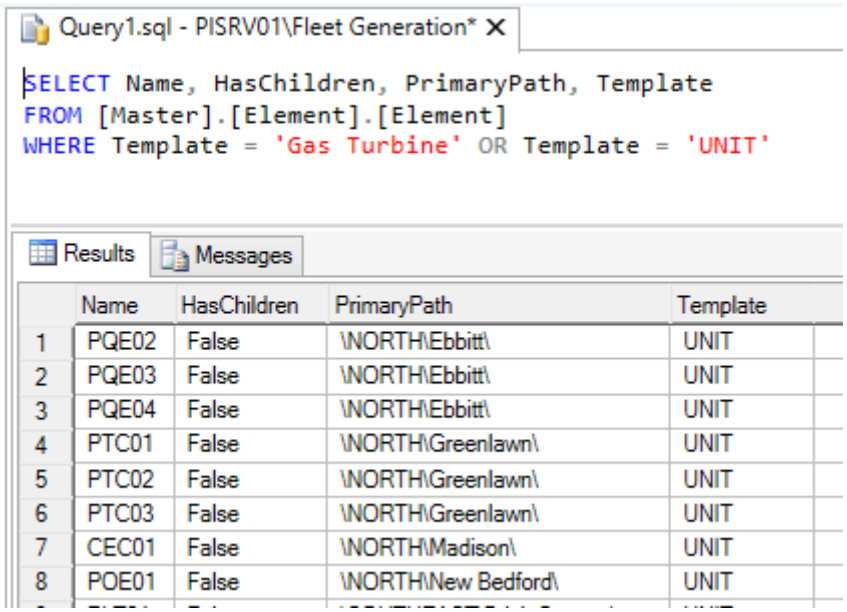
```
SELECT Name, HasChildren, PrimaryPath, Template
FROM [Master].[Element].[Element]
```

Now exclude the Regions and Stations. There are two easy ways to do this:

```
SELECT Name, HasChildren, PrimaryPath, Template
FROM [Master].[Element].[Element]
WHERE HasChildren = False
```

OR

```
SELECT Name, HasChildren, PrimaryPath, Template
FROM [Master].[Element].[Element]
WHERE Template = 'Gas Turbine' OR Template = 'UNIT'
```



	Name	HasChildren	PrimaryPath	Template
1	PQE02	False	\NORTH\Ebbitt\	UNIT
2	PQE03	False	\NORTH\Ebbitt\	UNIT
3	PQE04	False	\NORTH\Ebbitt\	UNIT
4	PTC01	False	\NORTH\Greenlawn\	UNIT
5	PTC02	False	\NORTH\Greenlawn\	UNIT
6	PTC03	False	\NORTH\Greenlawn\	UNIT
7	CEC01	False	\NORTH\Madison\	UNIT
8	POE01	False	\NORTH\New Bedford\	UNIT

10.6.1 Directed Activity – The Query Compendium

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

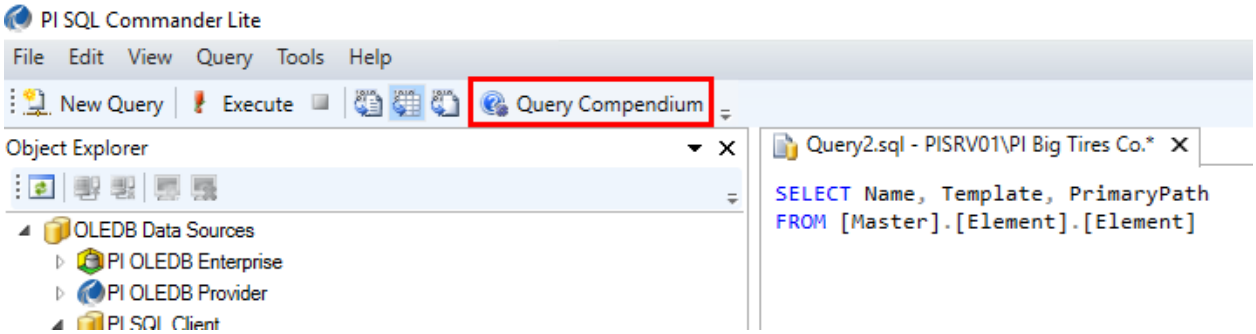
Objective:

Explore the Query Compendium

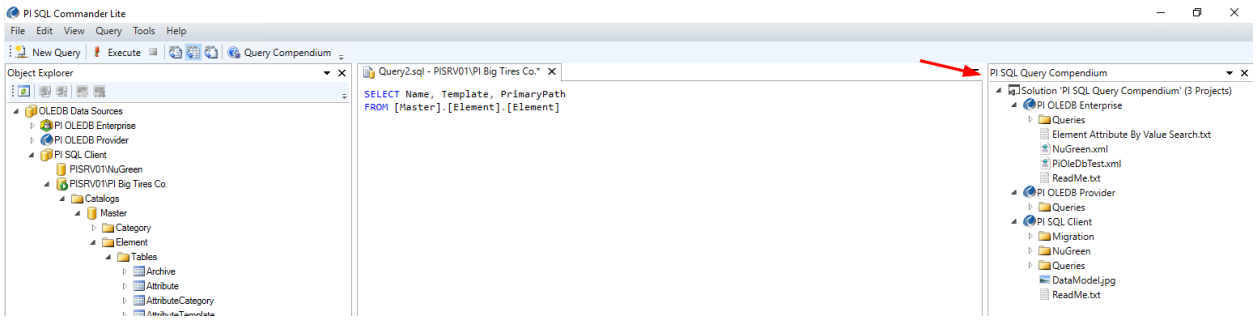
Approach:

We will now explore the Query Compendium, which is simply a library of example queries.

Click Query Compendium along the top bar

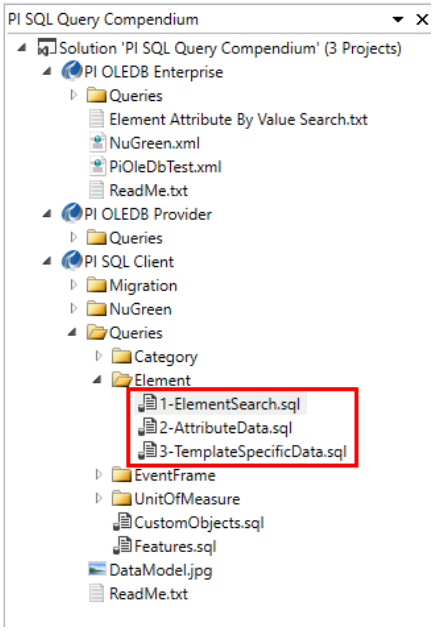


The PI SQL Query Compendium will open on the right hand side

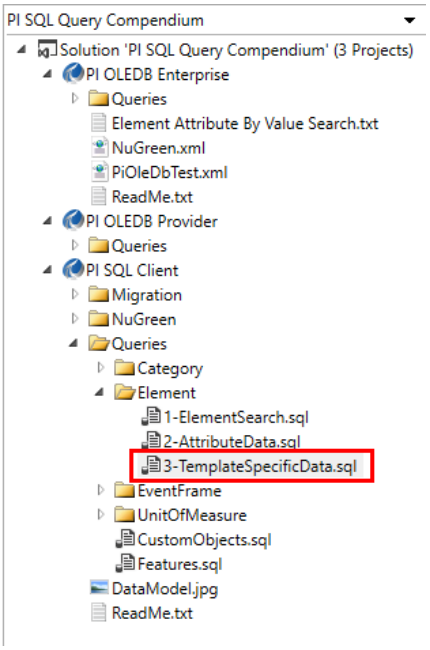


The main samples are under Queries\Element:

- 1-ElementSearch.sql contains sample queries for metadata (Paths, Element Names, Descriptions, Categories, etc)
- 2-AttributeData.sql contains sample queries for attribute data where a template is not specified
- 3-TemplateSpecificData.sql contains sample queries for attribute data by Template – similar to the functionality of the PI OLEDB Enterprise transpose functions



Double-click 3-TemplateSpecificData.sql to examine the contents



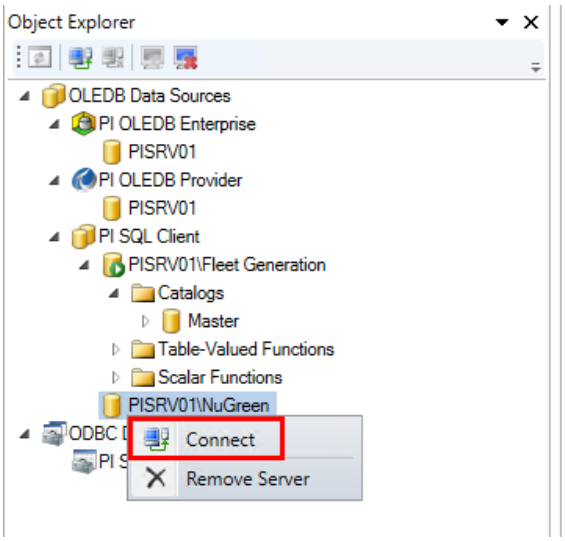
The first query sample has the structure to retrieve Fuel Gas Flow readings for Boilers. This won't run against the Fleet Generation database since there is no Boiler template. There are also samples for:

- Sampled Data (Interpolated data with a start time, end time, and interval)
- Data sampled at a specific timestamp
- Summaries (Averages, Maximums, Minimums, etc)

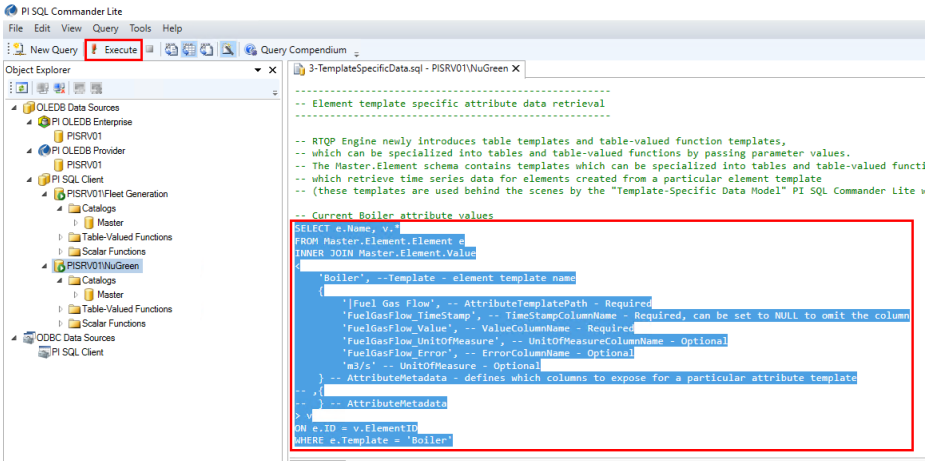
All of the samples in the Query Compendium run against the NuGreen database.

XML files to deploy the sample NuGreen database are bundled with every installation of PI SQL Client and can be found in C:\Program Files\PIPC\SQL\SQL Commander\PI SQL Query Compendium\PI SQL Client\NuGreen**.**

For the Query Compendium queries to run as-is, we'll have to connect to the NuGreen database in PI SQL Client:



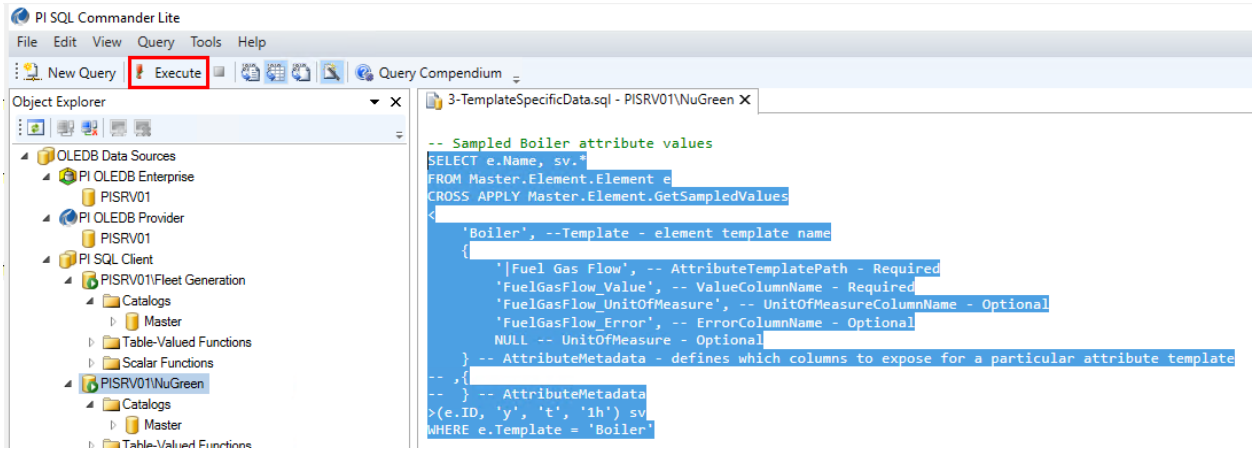
Highlight the first sample query and click execute:



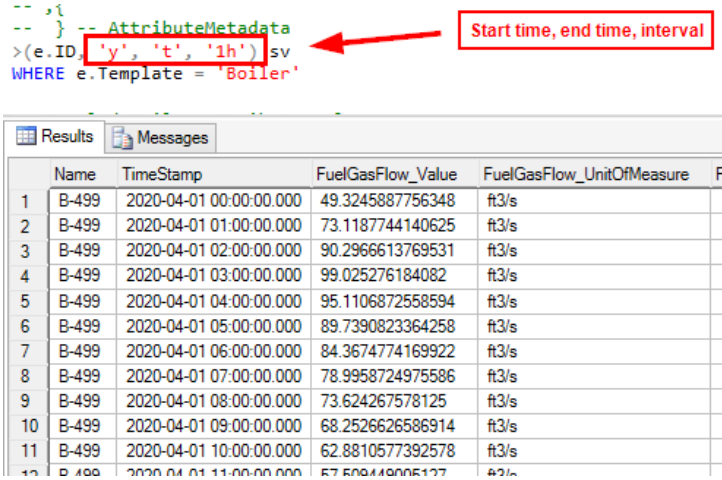
Note that it returns the current value of Fuel Gas Flow for each Boiler:

	Name	ElementID	FuelGasFlow_TimeStamp	FuelGasFlow_Value	FuelGasFlow_UnitOfMeasure	FuelGasFlow_Error
1	B-499	a1884547-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
2	B-209	a1884544-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
3	B-765	a1884514-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
4	B-210	a18844ed-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
5	B-235	a18844e1-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
6	B-481	a1884454-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
7	B-352	9b781f79-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
8	B-459	9b781f61-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
9	B-334	9b781f31-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
10	B-125	9b781f0d-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
11	B-555	a18844c9-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
12	B-309	a18844ba-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
13	B-737	a18844ab-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
14	B-914	a1884499-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
15	B-117	a188447b-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	
16	B-045	9b781edd-6b55-11e9-a964-000d3a312e96	2020-04-02 13:45:19.000	2.54135272587813	m3/s	

Highlight the second query (Sampled Boiler attribute values) and **execute it**.



This time hourly data between ‘y’ and ‘t’ is returned:



	Name	TimeStamp	FuelGasFlow_Value	FuelGasFlow_UnitOfMeasure	FuelGasFlow_Error
1	B-499	2020-04-01 00:00:00.000	49.3245887756348	ft3/s	
2	B-499	2020-04-01 01:00:00.000	73.1187744140625	ft3/s	
3	B-499	2020-04-01 02:00:00.000	90.2966613769531	ft3/s	
4	B-499	2020-04-01 03:00:00.000	99.025276184082	ft3/s	
5	B-499	2020-04-01 04:00:00.000	95.1106872558594	ft3/s	
6	B-499	2020-04-01 05:00:00.000	89.7390823364258	ft3/s	
7	B-499	2020-04-01 06:00:00.000	84.3674774169922	ft3/s	
8	B-499	2020-04-01 07:00:00.000	78.9958724975586	ft3/s	
9	B-499	2020-04-01 08:00:00.000	73.624267578125	ft3/s	
10	B-499	2020-04-01 09:00:00.000	68.2526626586914	ft3/s	
11	B-499	2020-04-01 10:00:00.000	62.8810577392578	ft3/s	

The text for these queries can be copied to a new query and modified.

10.6.2 Directed Activity - Modify a Sample Query for Fleet Generation

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

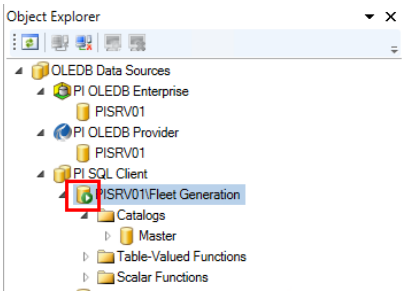
Objective:

Use the Query Compendium as a starting point to develop a query.

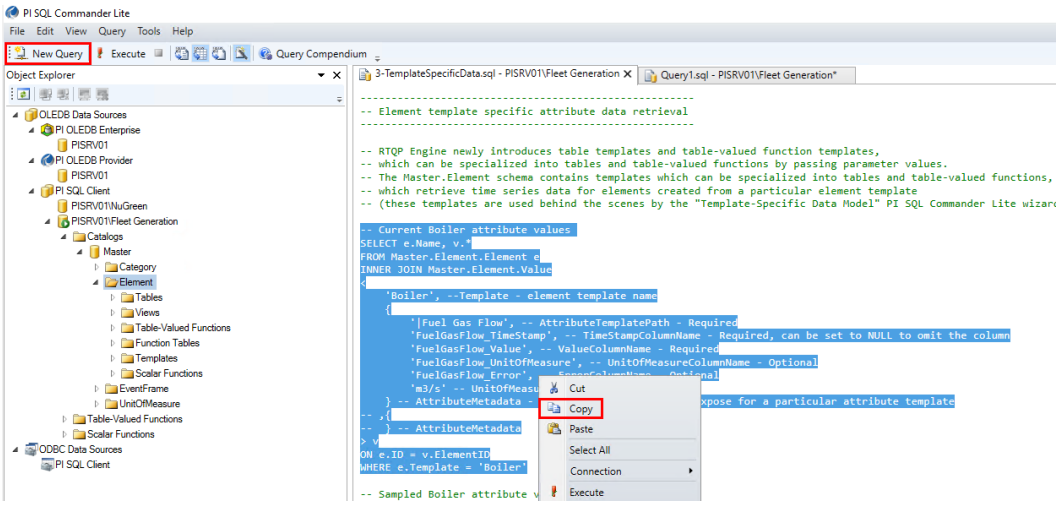
Approach:

We will now modify one of the queries from the Query Compendium to run against the Fleet Generation database. Let's get a list of Generating Units and their current Demand and Net Generation values.

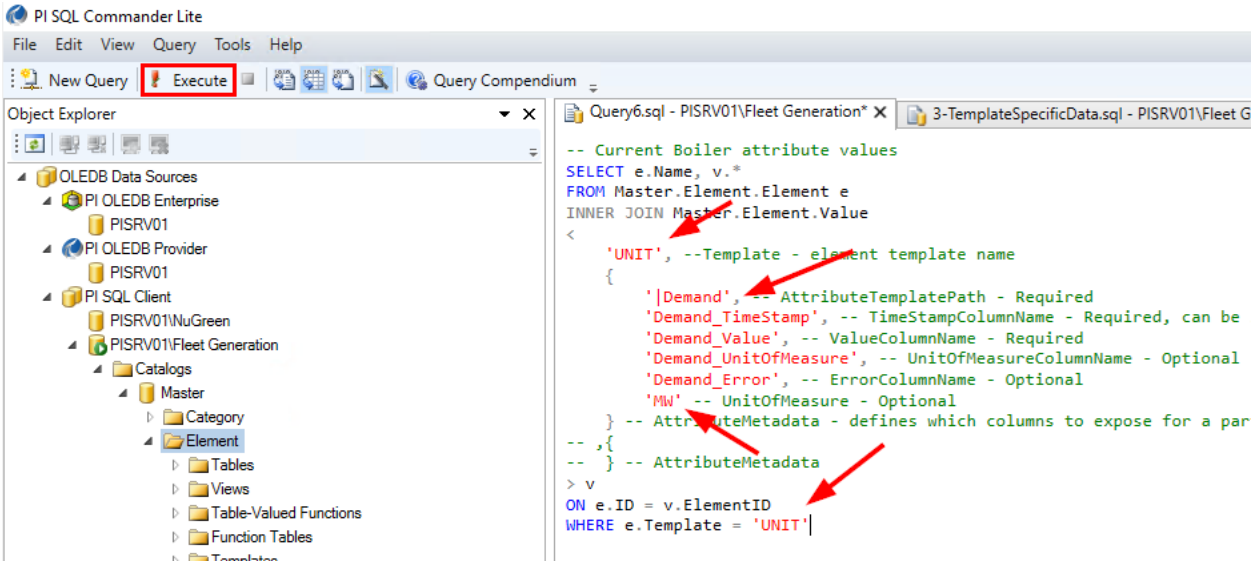
Ensure that you're connected to Fleet Generation:



Copy the first query from 3-TemplateSpecificData.sql into a new query.

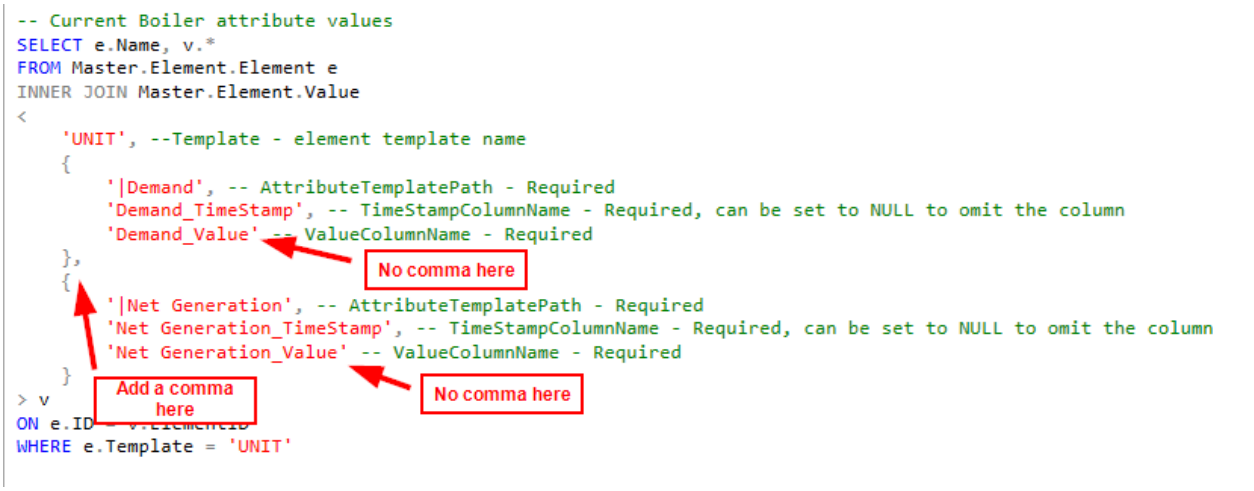


Change Boiler to **UNIT**, change Fuel Gas Flow to **Demand**, and change the **units to MW** then execute the query.



Now we can **add Net Generation** to the mix and remove some of the optional columns.

Be mindful of the commas, you'll need to add a comma between each attribute grouping and also remove the final comma within each attribute grouping.



Execute the query to make sure it functions.

Now modify the SELECT statement to remove unnecessary columns (explicitly select Demand_Value and Net Generation_Value). Also alias the columns. We'll discuss aliases in the following sections.

You can copy and paste the below query if you're having trouble getting everything perfect:

```
SELECT e.Name, v.Demand_Value as Demand, v.[Net
Generation_Value] as [Net Generation]
FROM Master.Element.Element e
INNER JOIN Master.Element.Value
<
   'UNIT',
   {
      '|Demand',
      'Demand_TimeStamp',
      'Demand_Value'
   },
   {
      '|Net Generation',
      'Net Generation_TimeStamp',
      'Net Generation_Value'
   }
> v
ON e.ID = v.ElementID
WHERE e.Template = 'UNIT'
```

Execute the query to make sure it functions. The output should be as follows however keep in mind that **everyone will have different data**.

	Name	Demand	Net Generation
1	PTC02	66.8131	365.2076
2	PTC03	340.1134	240.8696
3	CEC01	475.5364	121.8466
4	POE01	4.356421	273.4194
5	PLT01	93.66851	398.3531
6	PLT02	599.0018	600
7	BAJ02	71.83372	498.7563
8	MAM03	316.6634	516.4491
9	MAM04	498.5371	148.0519
10	ALX01	0.9355849	445.116
11	PQE02	467.4824	269.6476
12	PQE03	8.321041	600
13	PQE04	96.05633	413.7107
14	PTC01	599.2962	176.5635
15	ZMN01	327.1069	565.3942
16	ZMN02	485.9812	600
17	MND01	2.584461	108.1458
18	MND02	92.24759	255.3658
19	MAM01	599.2962	600
20	MAM02	78.32897	478.2043

Field Aliases

There's a lot going on in the previous query that may have been glossed over. Let's take a closer look at the individual parts starting with the first line.

```
SELECT e.Name, v.Demand_Value as Demand, v.[Net Generation_Value] as [Net Generation]
```

Aliases are being used to provide cleaner column names, essentially renaming the columns. We made a decision not to include the timestamp columns and the _Value part is therefore unnecessary.

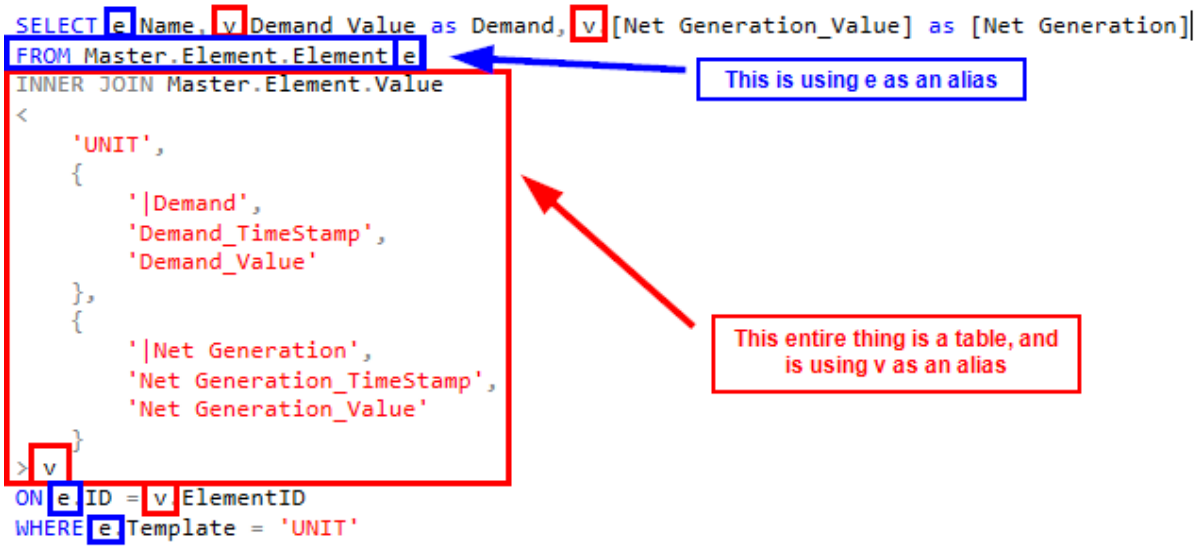
The keyword **AS** is used anytime an **ALIAS** is defined, however **the as keyword can be omitted.**

In addition, **square brackets** (eg. v.[Net Generation_Value] as [Net Generation]) are necessary for any fields that contain spaces. This is sort of like using double quotes when specifying the path to a file. When PI SQL Commander sees a space, it expects that the field or command has ended and wants to process the next thing.

Table Aliases

Tables are also being aliased in the previous query though the **AS** keyword is omitted:

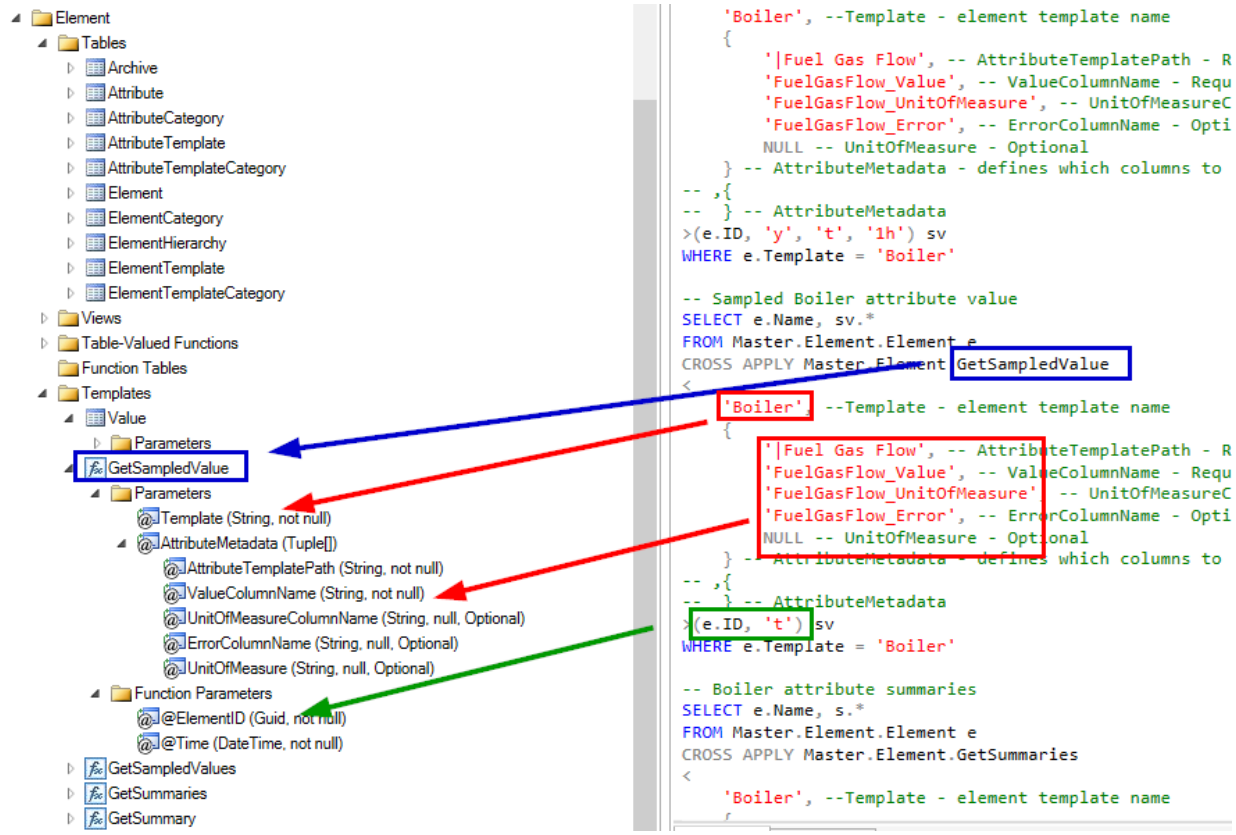
Sometimes table name or columns are lengthy or lack clarity. Using an **ALIAS** can simplify typing and clarify table field names that are otherwise unclear.



In the above query, e is being used to identify the Element Table and v is being used to identify the Value table.

Parameters

Parameters are being passed into Master.Element.Value between the < >. If you expand Element\Templates you'll see the core functions that are used in the 3-TemplateSpecificData.sql examples.

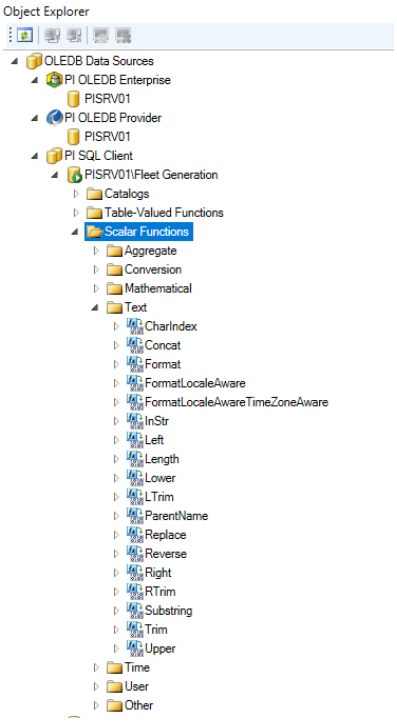


Builtin Functions

PI SQL Client has some built-in functions specific to the PI System. If you are familiar with SQL, you may already be familiar with functions. For example, aggregation functions such as Max() or Avg() return the maximum or average of a group of rows.

An entire list of built-in functions is available in the SQL for RTQP Engine Reference Guide. The documentation is also available as a PDF download on the OSIsoft Customer portal.

They are also listed under Scalar Functions in PI SQL Commander:



A commonly used function is the ParentName function, which gives the name of an element's parent. For example, we can change the query as follows to get the Station name for each UNIT:

```
SELECT e.Name, ParentName(e.PrimaryPath,0) as Station,
v.Demand_Value as Demand, v.[Net Generation_Value] as [Net
Generation]
FROM Master.Element.Element as e
INNER JOIN Master.Element.Value
<
   'UNIT',
   {
      '|Demand',
      'Demand_TimeStamp',
      'Demand_Value'
   },
   {
      '|Net Generation',
      'Net Generation_TimeStamp',
      'Net Generation_Value'
   }
> as v
ON e.ID = v.ElementID
WHERE e.Template = 'UNIT'
```

The first argument in ParentName is an AF element path, and the second argument tells it how many levels up to look:

- ParentName(e.PrimaryPath,0) gives the direct parent (eg. Octavia Station)
- ParentName(e.PrimaryPath,1) gives the parent of the parent (eg. NORTH Region)

UNION Statements

You may have noticed that the query we've been running only return Assets that use the UNIT template and not those that use the Gas Turbine template, despite the Gas Turbine template being derived from the UNIT template via template inheritance. One way to address this is with UNIONs.

In simple terms, UNIONs take the results of two queries and stack the result sets on top of each other to form a single result set. One limitation of UNIONs is that the input result sets must have identical columns, which may require removing and aliasing columns to match the data sets. This will be demonstrated in the following exercise.

The syntax is quite simple, place the keyword UNION in between two queries to union them together. The OPTION statement must be at the very end:

```
SELECT * FROM Table1 WHERE Condition='TRUE'
UNION
SELECT * FROM Table2 WHERE Condition='TRUE'
```

Directed Activity – Using UNION Statements

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Create a query to display Demand and Net Generation for all UNITs, including Gas Turbines.

Approach:

You may have noticed that the query we've been working with has been missing the Gas Turbines. One way to address this with a UNION. It's not necessarily the best approach, but you'll use a UNION in a later exercise.

Start with the below query:

```
SELECT e.Name, ParentName(e.PrimaryPath,0) as Station,
v.Demand_Value as Demand, v.[Net Generation_Value] as [Net
Generation]
FROM Master.Element.Element as e
INNER JOIN Master.Element.Value
<
   'UNIT',
   {
      '|Demand',
      'Demand_TimeStamp',
      'Demand_Value'
   },
   {
      '|Net Generation',
      'Net Generation_TimeStamp',
      'Net Generation_Value'
   }
> as v
ON e.ID = v.ElementID
WHERE e.Template = 'UNIT'
```

Copy it, paste the copy directly below it, add a UNION in between, and change the template in the bottom half to Gas Turbine, then execute.

Exercise 10.8.1.sql - PISRV01\Fleet Generation | Query2.sql - PISRV01\Fleet Generation*

```
SELECT e.Name, ParentName(e.PrimaryPath,0) as Substation, v.Demand_Value as
FROM Master.Element.Element as e
INNER JOIN Master.Element.Value
<
    'UNIT',
    {
        '|Demand',
        'Demand_TimeStamp',
        'Demand_Value'
    },
    {
        '|Net Generation',
        'Net Generation_TimeStamp',
        'Net Generation_Value'
    }
> as v
ON e.ID = v.ElementID
WHERE e.Template = 'UNIT'
UNION
SELECT e.Name, ParentName(e.PrimaryPath,0) as Substation, v.Demand_Value as
FROM Master.Element.Element as e
INNER JOIN Master.Element.Value
<
    'Gas Turbine',
    {
        '|Demand',
        'Demand_TimeStamp',
        'Demand_Value'
    },
    {
        '|Net Generation',
        'Net Generation_TimeStamp',
        'Net Generation_Value'
    }
> as v
ON e.ID = v.ElementID
WHERE e.Template = 'Gas Turbine'
```

There should now be 30 rows in the result set. The bottom 10 rows are the Gas Turbines.

JOIN Statements

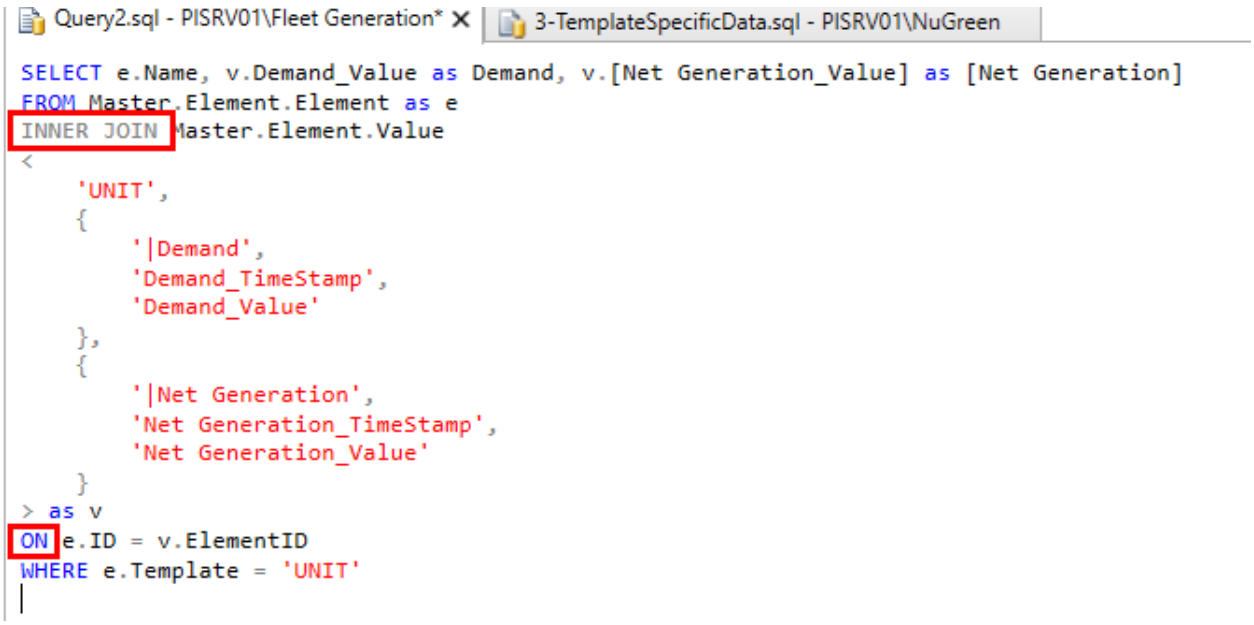
Rarely does data exist in one place or in one table. Sometimes the results of a query have to come from a correlation of two or more distinct tables. To JOIN tables, a relationship is required between the tables and must be identified in the SQL statement.

Within the joining operations, we want a result set than contains assets with useful information from both tables, like performing a logical AND operation. There should be no gaps where a match could not be found. This is called an INNER JOIN, and is the default joining operation used by PI SQL Commander. Therefore, INNER JOIN and JOIN may be used interchangeably

Two key words are used when creating joins between tables. **The words JOIN and ON can be used in the statement to identify the relationship between the tables being used.** The key word ON sets up the relationship of columns in the selected tables so the desired rows are returned.

In our query, the time series data from the Master.Element.Value table (aliased as v) is being JOINed to the Master.Element.Element table (aliased as e) where the ID column from e matches the ElementID column from v.

This is necessary in order to display the element names rather than the element ids. Master.Element.Value doesn't contain actual element names or path information.



Directed Activity – Using JOIN Statements

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

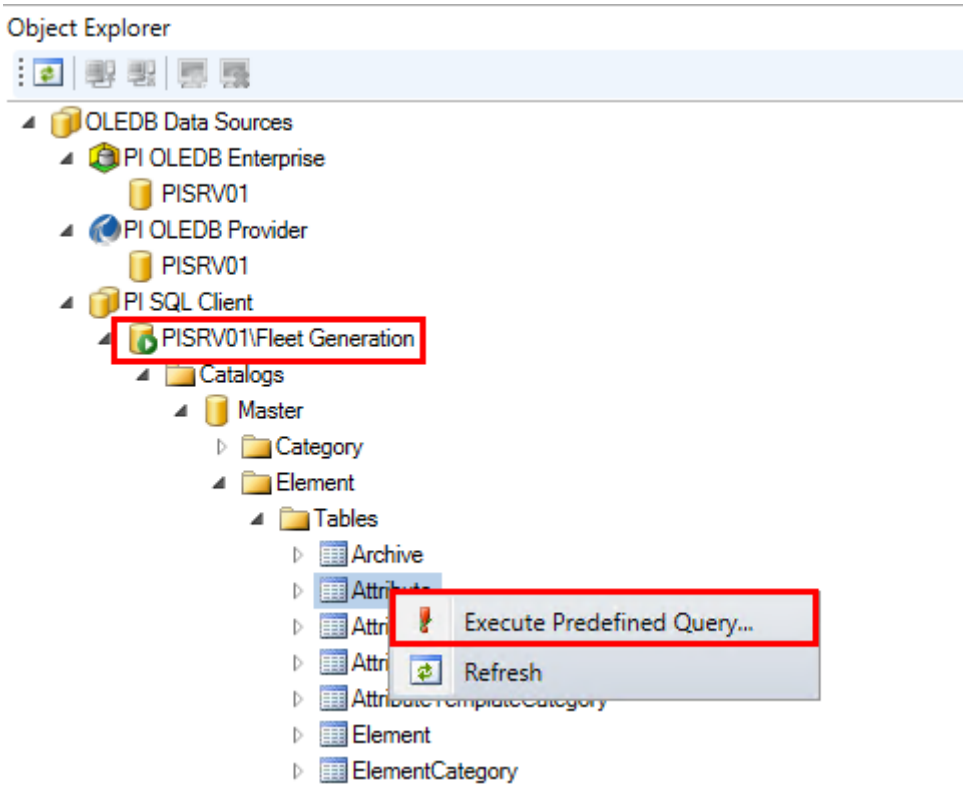
Get some experience with JOINs by combining data from multiple tables.

Approach:

In this example, we'll create a query with the following columns:

- Net Generation Attribute Values
- Attribute Categories (the Category of Net Generation from PI AF)
- Unit Name and Station information

Start by executing the predefined query for Attribute under PISRV01\Fleet Generation:



Note that the Path is not the element path, it is the attribute path. They are all | (root) because there are no child attributes.

Query6.sql - PISRV01\Fleet Generation | Query2.sql - PISRV01\Fleet Generation* | 3-TemplateSpecificData.sql - PISRV01\NuGreen

```
-- Attribute table.
-- Returns element attributes.
SELECT TOP 100 Element, Path, Name, Description, ValueType, TraitType, IsConfigurationItem, IsHidden, IsManualDataEntry,
    DataReference, UnitOfMeasure, TimeStamp, Value, Value_Int, Value_Double, Value_String, Value_DateTime,
    IsValueAnnotated, IsValueGood, IsValueQuestionable, IsValueSubstituted, Error
FROM [Master].[Element].[Attribute]
```

Results | Messages

	Element	Path	Name	Description	ValueType	TraitType	IsConfigurationItem	IsHidden	IsManualDa
1	Wolverine Station	|	Total Hourly Gross Generation		Double		False	False	False
2	Wolverine Station	|	Average Utilization		Double		False	False	False
3	Vicksberg	|	Total Hourly Gross Generation		Double		False	False	False
4	Vicksberg	|	Average Utilization		Double		False	False	False

Alias [Master].[Element].[Attribute] as a, and then SELECT a.* and execute to confirm that there is no Category column.

```
SELECT a.*
FROM [Master].[Element].[Attribute] a
```

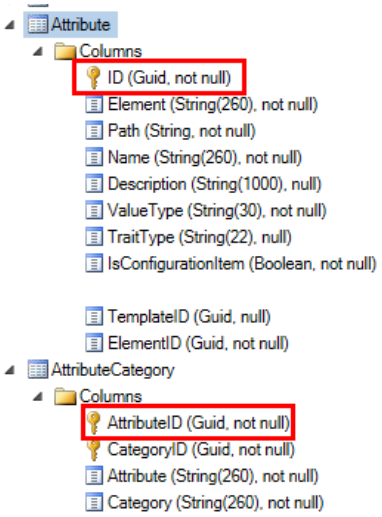
This could also be proven by inspecting the Attribute Table column list:

- Attribute
  - Columns
    - ID (Guid, not null)
    - Element (String(260), not null)
    - Path (String, not null)
    - Name (String(260), not null)
    - Description (String(1000), null)
    - ValueType (String(30), not null)
    - TraitType (String(22), null)
    - IsConfigurationItem (Boolean, not null)
    - IsHidden (Boolean, not null)
    - IsManualDataEntry (Boolean, not null)
    - DataReference (String(260), null)
    - UnitOfMeasure (String(260), null)
    - TimeStamp (DateTime, not null)
    - Value (Variant, null)
    - Value_Int (Int64, null)
    - Value_Double (Double, null)
    - Value_String (String, null)
    - Value_DateTime (DateTime, null)
    - IsValueAnnotated (Boolean, not null)
    - IsValueGood (Boolean, not null)
    - IsValueQuestionable (Boolean, not null)
    - IsValueSubstituted (Boolean, not null)
    - Error (String, null)
    - UnitOfMeasureID (Guid, null)
    - TemplateID (Guid, null)
    - ElementID (Guid, not null)

So we can get Net Generation values from this table, but…

- We'll need to JOIN to the AttributeCategory table in order to get the Category
- We'll need to JOIN to the Element table in order to get the PrimaryPath (element path) which can be fed into the ParentName() function.

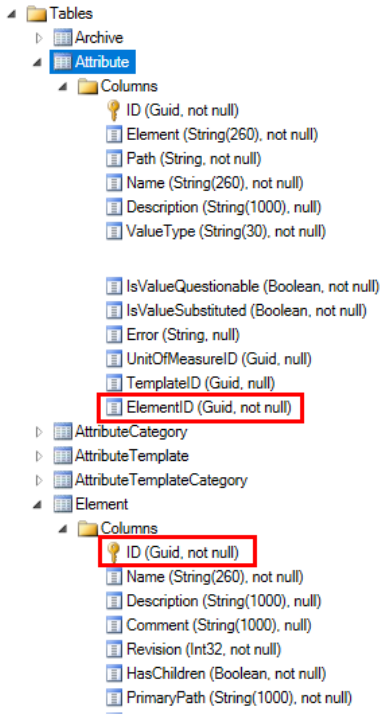
Upon inspection of the Attribute and AttributeCategory columns, it looks like we can look up the Category by joining on the attribute ID.



Join the tables, set aliases, select only the relevant columns, and only include attributes named Net Generation to arrive at the following query:

```
SELECT a.Name as [Attribute Name], a.Value, ac.Category
FROM [Master].[Element].[Attribute] as a JOIN
[Master].[Element].[AttributeCategory] as ac ON a.ID = ac.AttributeID
WHERE a.Name = 'Net Generation'
```

Upon inspection of the Attribute and Element columns, it looks like we can look up the PrimaryPath by joining on the element ID.



Join the tables, set aliases, and apply the ParentName function to arrive at the following query:

```
SELECT a.Name as [Attribute Name], a.Value, ac.Category,
ParentName(e.PrimaryPath,0) as Station
FROM [Master].[Element].[Attribute] as a JOIN
[Master].[Element].[AttributeCategory] as ac ON a.ID = ac.AttributeID JOIN
[Master].[Element].[Element] as e ON e.ID = a.ElementID
WHERE a.Name = 'Net Generation'
```

Template-Specific Data Models

For templates with many attributes, it can be quite cumbersome to use the Query Compendium examples. Imagine if you had to add 20 attributes. Ensuring each field was spelled perfectly, and troubleshooting missing commas would be a tedious process.

Template-Specific Data models are an easier way to develop queries against templatized AF Elements and Attributes.

For those familiar with PI OLEDB Enterprise, Template-Specific Data Models are the successor to the Transpose Functions.

Directed Activity – Create Template-Specific Data Models for Elements

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

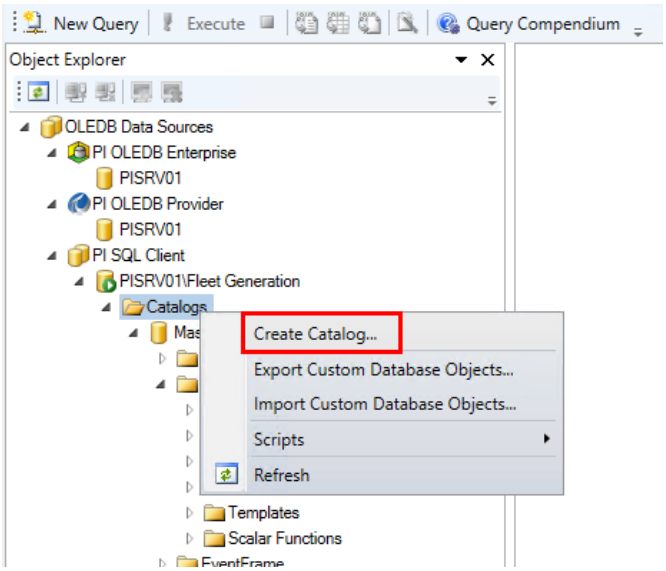
Objective:

Create Template-Specific Data Models for the Fleet Generation database to be used in analyzing unit generation data.

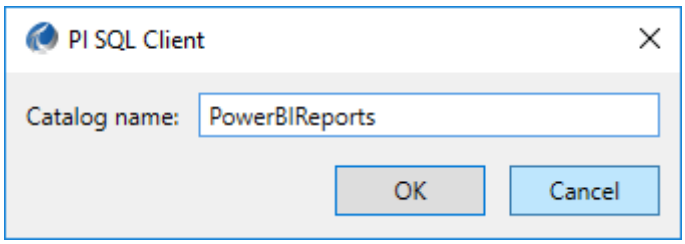
Approach:

First we'll create a new Catalog and Schema to keep our custom database objects separated from the Master Catalog. This step isn't mandatory, it's just to avoid cluttering the Master Catalog. Catalogs can be used to organize custom database objects. For example, each person could have their own sandbox, or objects could be separated by report or application.

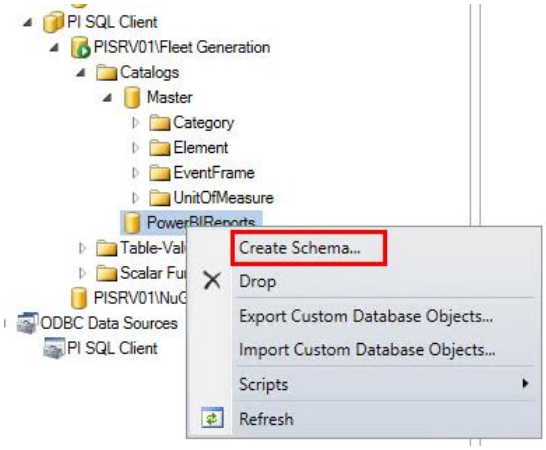
Create a new Catalog:



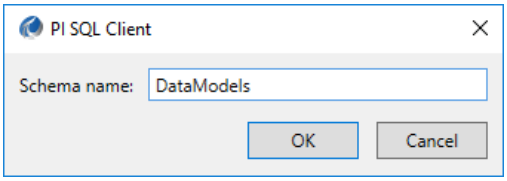
Name it PowerBIReports:



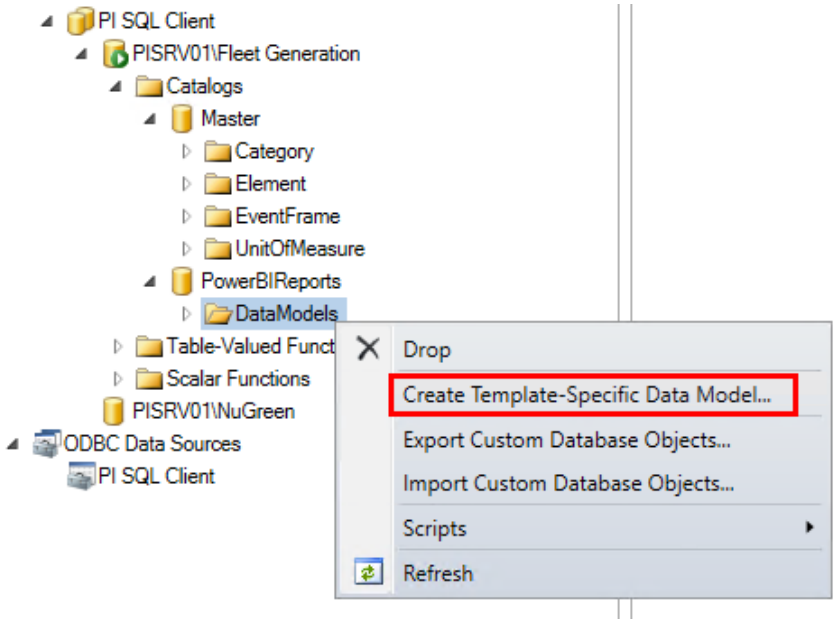
Then create a new Schema. A schema is a collection of tables and generally describes how a database is constructed. We could avoid creating a Schema by working in the Master Catalog, but since we created a new Catalog we need to add a Schema to it in order to create Template-Specific Data Models:



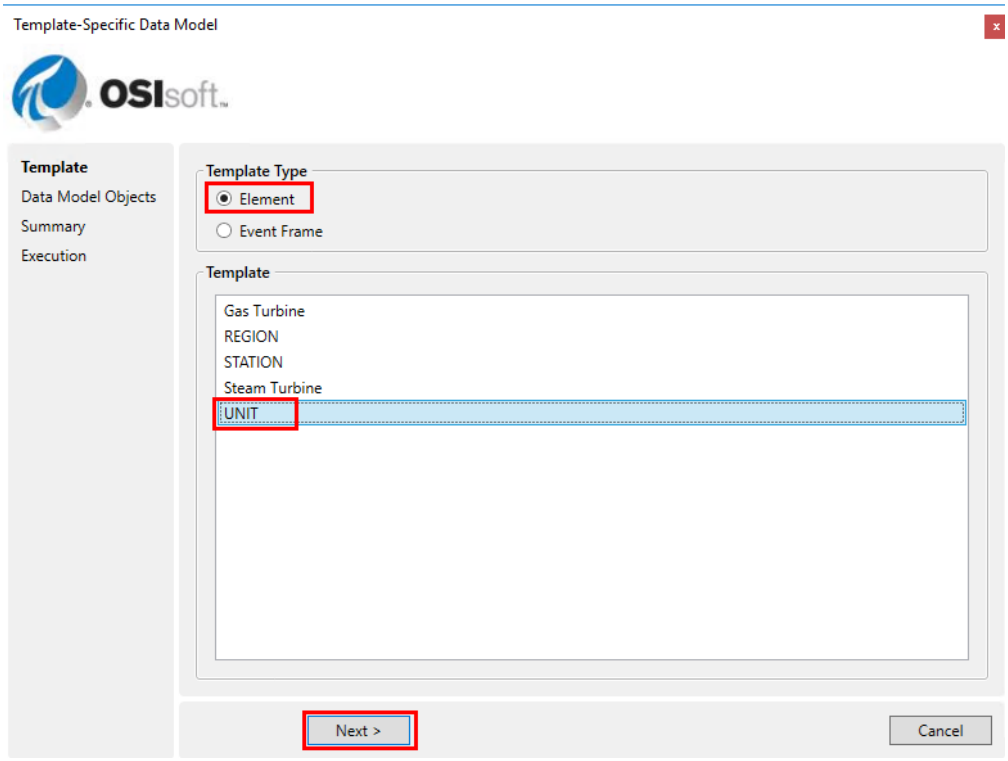
Name it DataModels:



Now you can right-click the new Schema and select Create Template-Specific Data Model…:



The process is fairly straightforward. Select whether you want to create them for Event Frames or Elements. In our case we want Elements of type UNIT. Click Next:

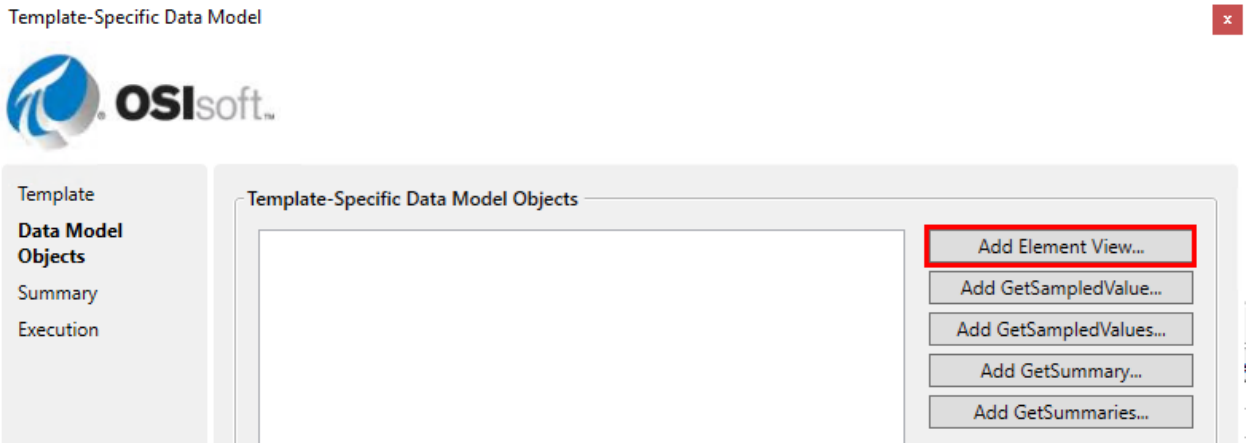


On the next screen you have the following options:

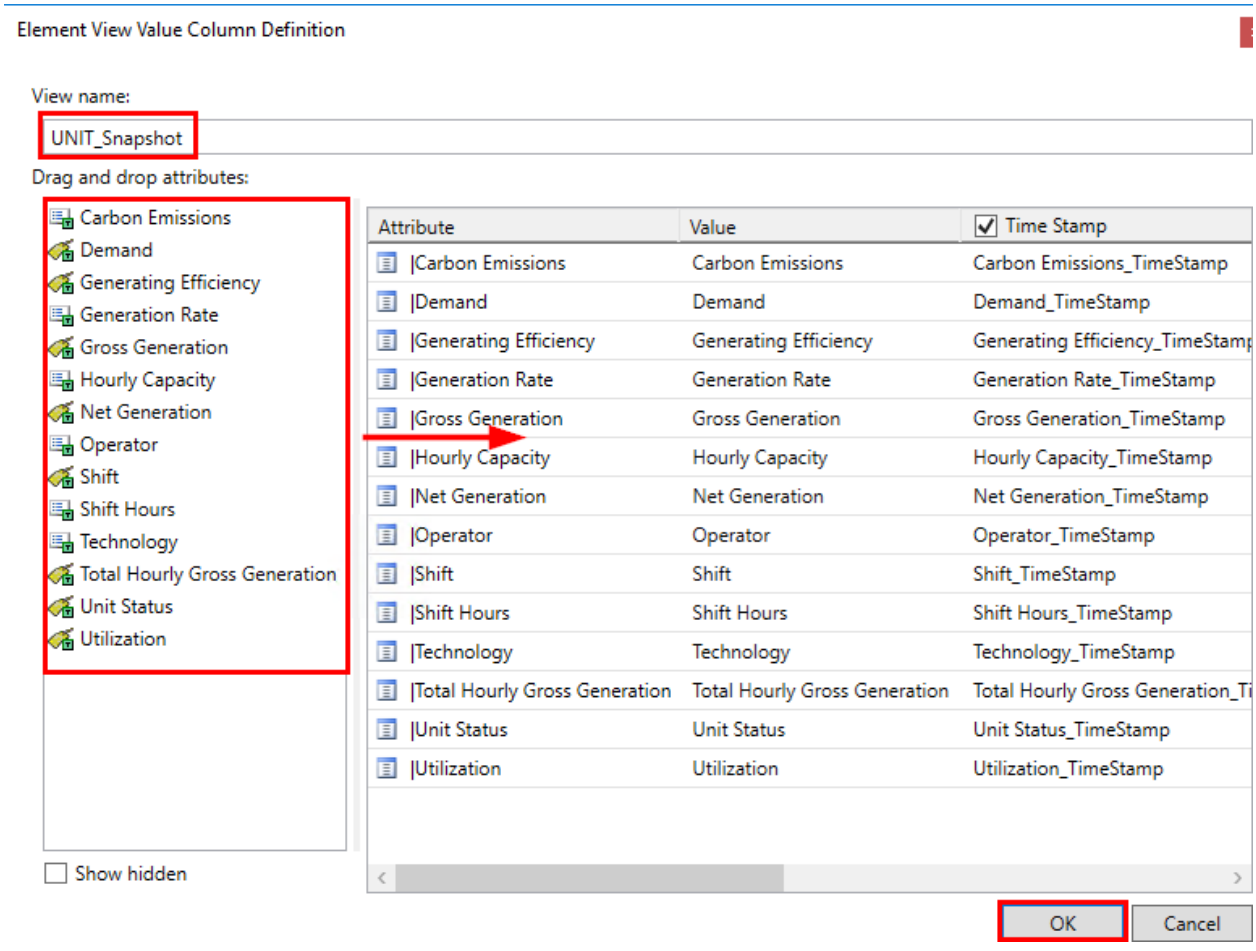
Add Element View	Snapshot or Static	Single value per element at the current time
Add GetSampledValue	Interpolated	Single value per element at the supplied timestamp
Add GetSampledValues	Interpolated	1 interpolated value per interval per element based on start time, end time, and interval
Add GetSummary	Calculated	Single value per element for the specified summary calculation (Average, Min, Max, etc)
Add GetSummaries	Calculated	1 summary calculation per interval per element based on start time, end time, and interval

For the report in the next chapter, we'll want an Element View model and a GetSampledValues model.

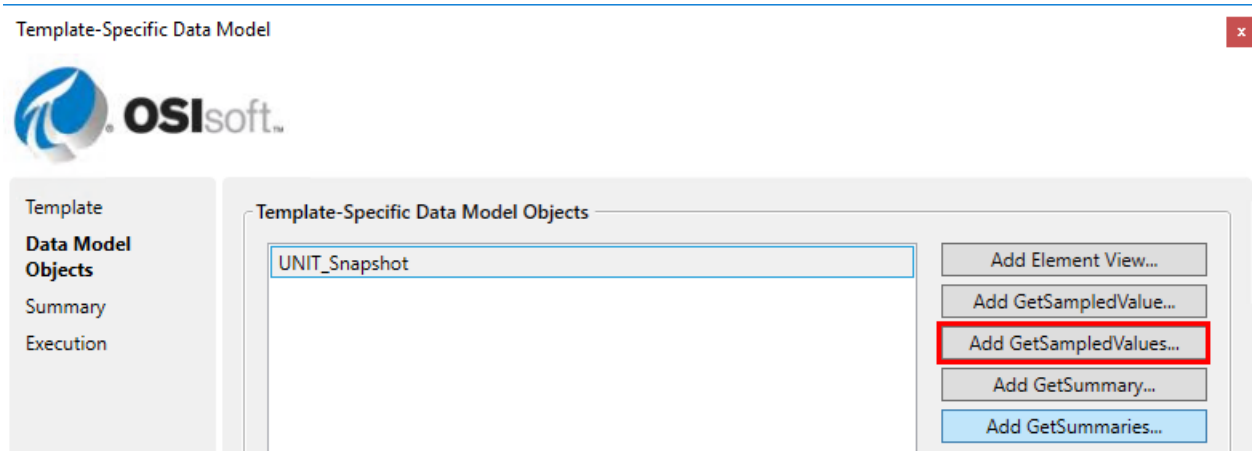
Click Add Element View…



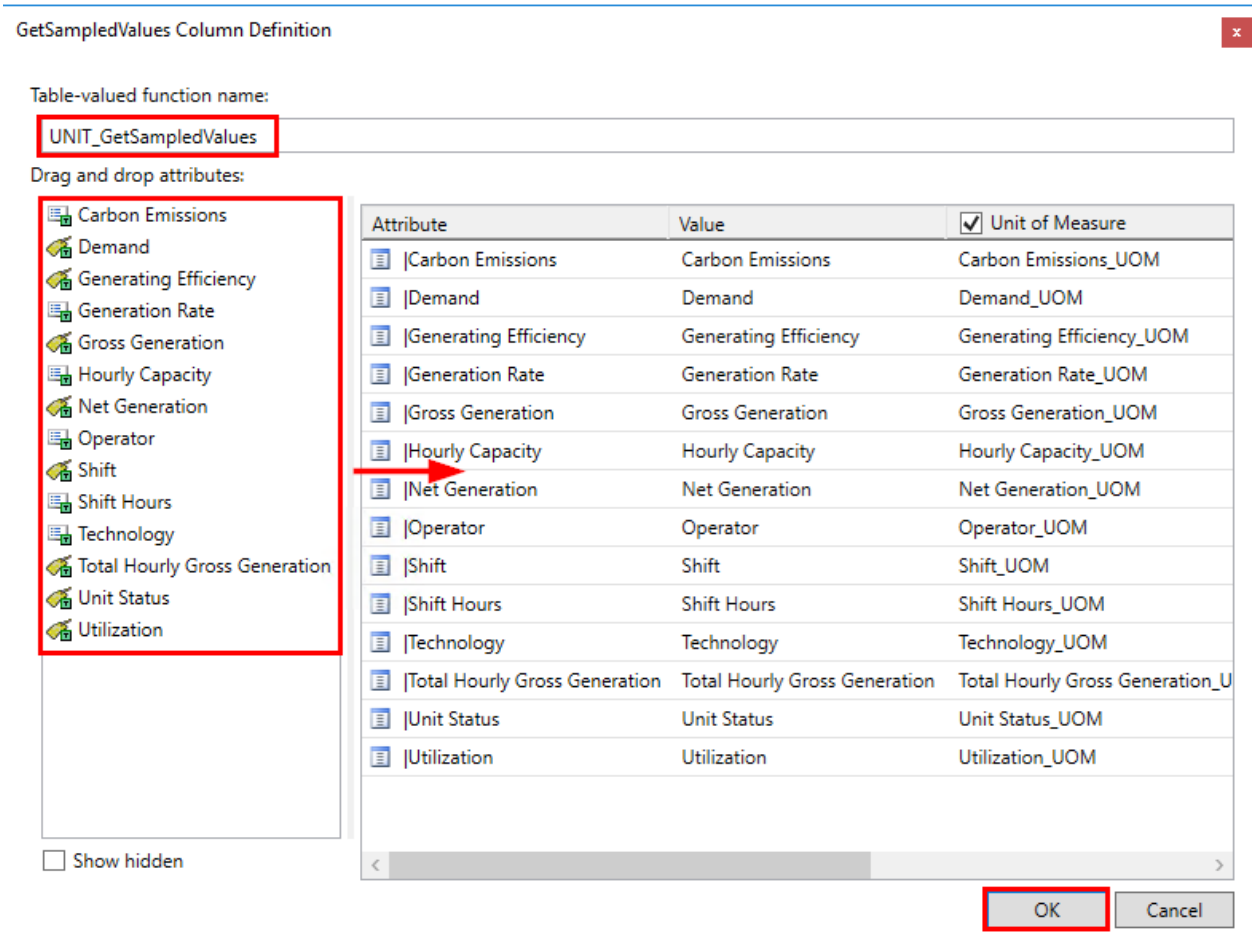
Name it UNIT_Snapshot, then drag and drop all attributes **except Location and Name** and click OK:



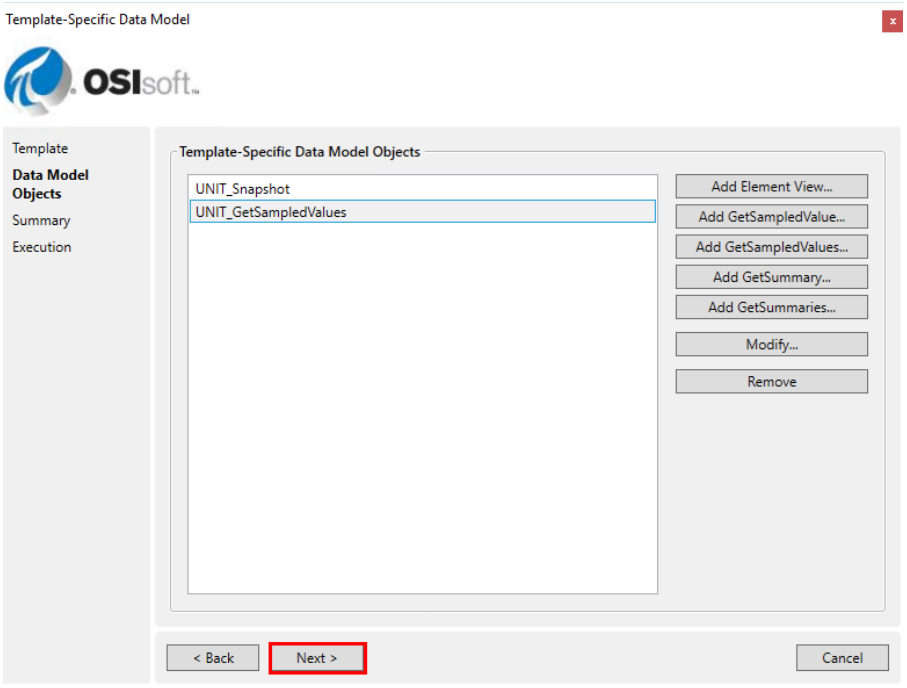
Click Add GetSampledValues:



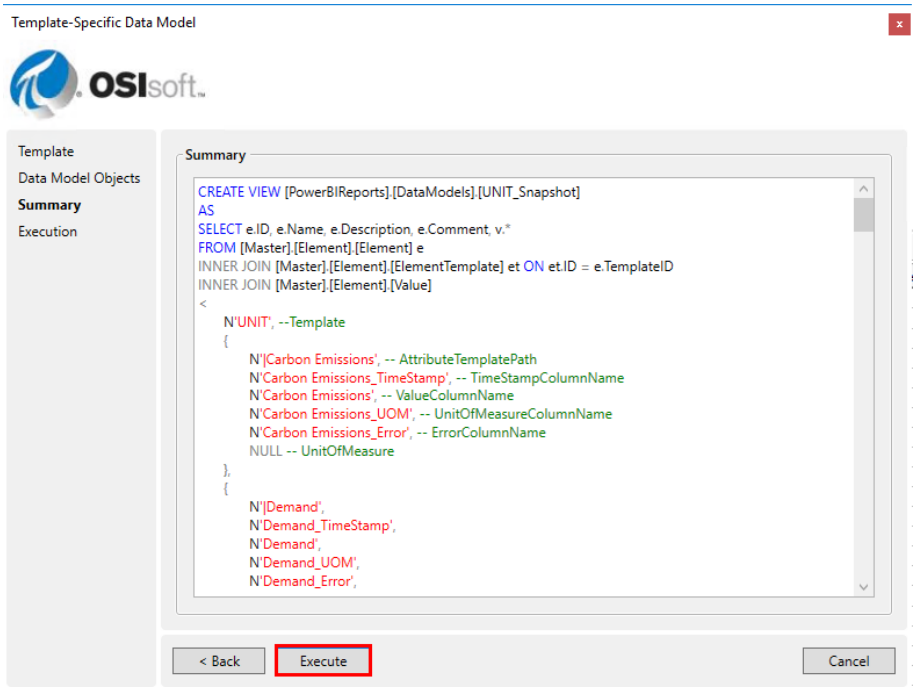
Leave the default name, drag and drop all attributes, then click OK:



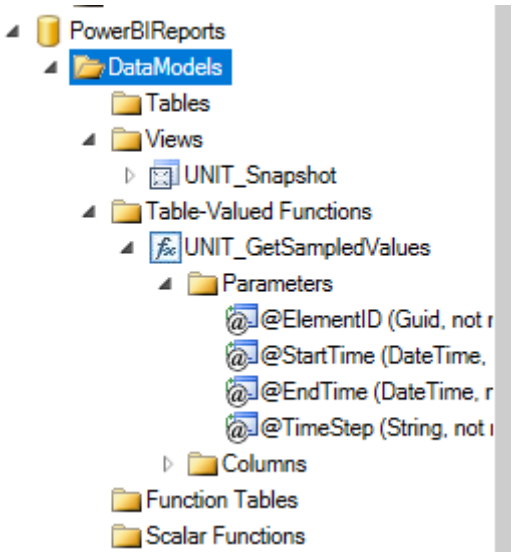
Now that both data models have been specified, click Next:



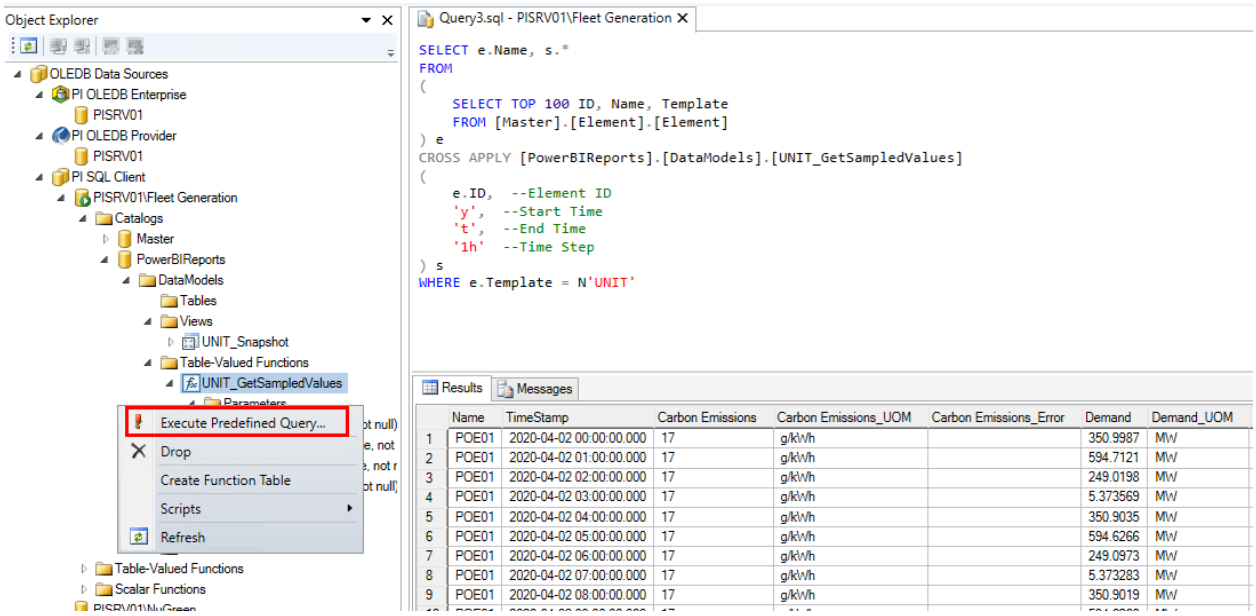
And then Execute:



UNIT_Snapshot will show up under Views whereas UNIT_GetSampledValues will show up under Table-Valued Functions because it has Parameters (element ID, start time, end time, and interval):



Once the data models have been created, we can execute the predefined queries and modify the query to suit the application or report:



Exercise – Template-Specific Data Models for Event Frames

This solo or group activity is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the activity.

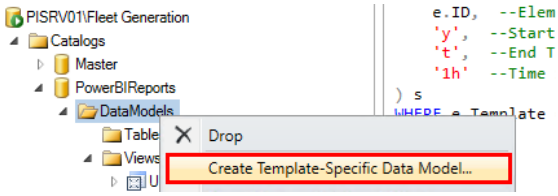
Objective:

- Create Template-Specific Data Models for Inactivity and Gas Turbine Temperature Anomaly Event Frames

Approach:

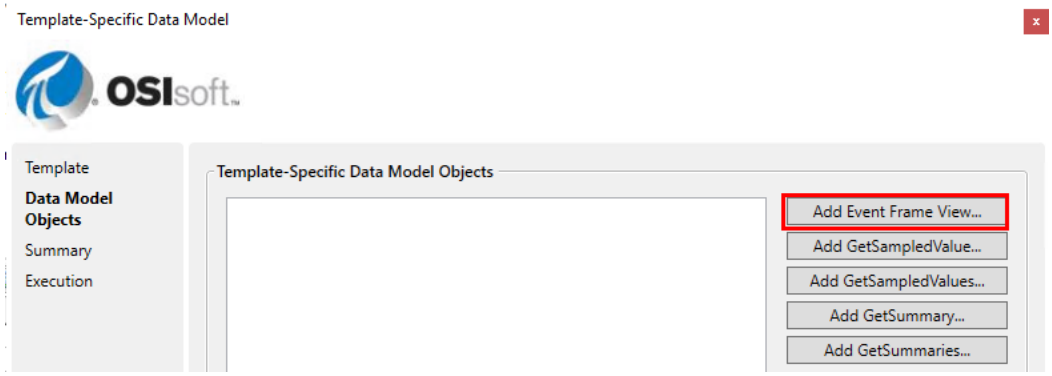
- Create Template-Specific Data Models for Inactivity and Gas Turbine Temperature Anomaly Event Frames.
- Use the default names and include all attributes.
- Verify the results of the transpose function through the execution of the pre-defined query.

Hints: The steps are almost identical to the previous exercise except this time for Event Frames.



You'll have to go through the wizard for each type of Event Frame.

Only the Event Frame View is necessary for each Event Frame type.



Saved Views

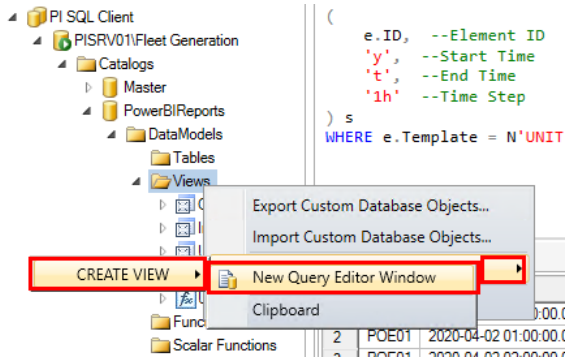
Often Administrators would prefer to create Views for end-users who are not familiar with SQL queries. Often Views are queried using a basic SELECT * query to return all data without any WHERE clause and without selecting individual columns. This masks the complexity and size of the query (eg. table JOINS and UNIONs of several tables) but places the burden of maintaining the

query on the administrator. In future exercises we will be using the queries directly, but it is still useful to know how to create and query views in PI SQL Commander.

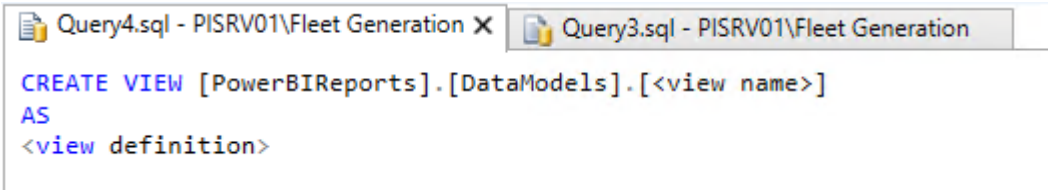
Creating dataset views

PI SQL Commander supports the creation of views. Views allow you to name a stored query and it is this name that appears in the table list when importing data into BI clients. Views are the easiest way to allow users to select which datasets they want from PI AF when creating a report, as they do not need to understand the complexity of the underlying SQL query.

Views are created using SQL syntax, but PI SQL Client can give you a template to start with.



Selecting Scripts -> Create View -> New Query Editor Window produces the beginning of a query:



At this point, it is a matter of naming the View by replacing <view name> and copy pasting the query by into <view definition> placeholder.

Directed Activity – View Creation

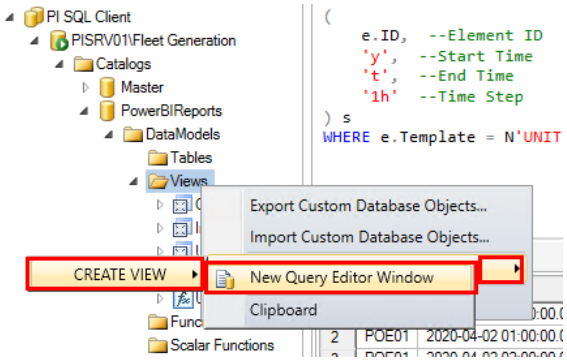
In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

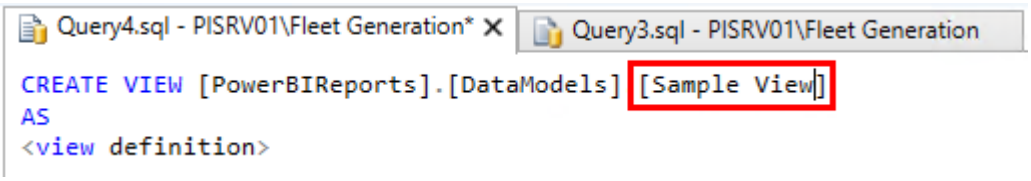
Create a View using a pre-existing query.

Approach:

In the PowerBIReports catalog, run the Create View script:



Replace <view name> with Sample View:



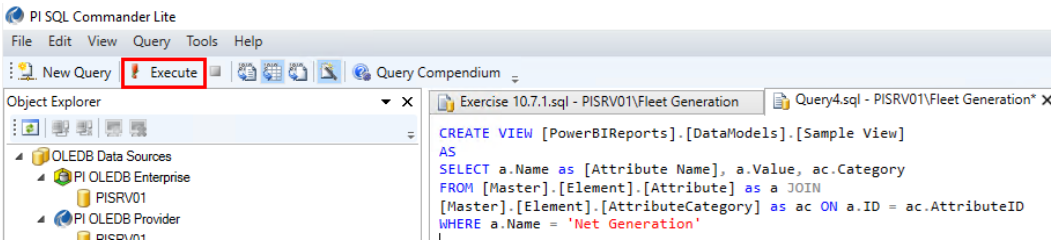
Paste the following query in place of <view definition>:

```
SELECT a.Name as [Attribute Name], a.Value, ac.Category,
ParentName(e.PrimaryPath,0) as Station
FROM [Master].[Element].[Attribute] as a JOIN
[Master].[Element].[AttributeCategory] as ac ON a.ID = ac.AttributeID JOIN
[Master].[Element].[Element] as e ON e.ID = a.ElementID
WHERE a.Name = 'Net Generation'
```

The resulting query is then:

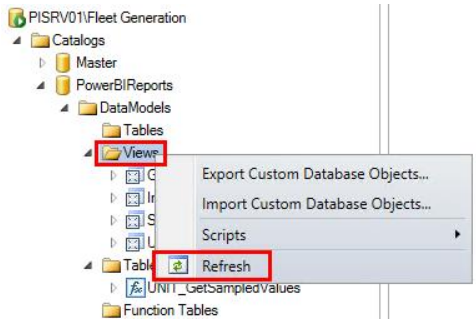
```
CREATE VIEW [PowerBIReports].[DataModels].[Sample View]
AS
SELECT a.Name as [Attribute Name], a.Value, ac.Category,
ParentName(e.PrimaryPath,0) as Station
FROM [Master].[Element].[Attribute] as a JOIN
[Master].[Element].[AttributeCategory] as ac ON a.ID = ac.AttributeID JOIN
[Master].[Element].[Element] as e ON e.ID = a.ElementID
WHERE a.Name = 'Net Generation'
```

Execute to create the View:

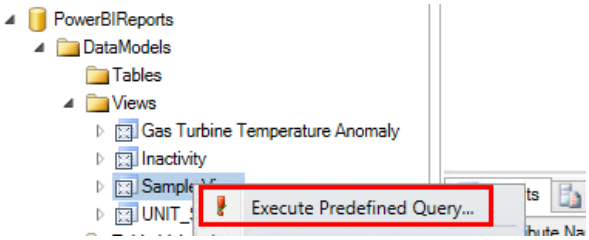


It should say the query executed successfully along the bottom:

Refresh the View list:



You should see Sample View. **Execute** the predefined query to confirm that it's functional:



Change the query to the below and **Execute**:

```
SELECT * FROM [PowerBIReports].[DataModels].[Sample View]
```

This is much simpler for end-users to work with and also locks down the view so that users are not tempted to modify it.

Importing PI SQL Client data to Power BI

The first thing to do when using a client is to import the data you want to analyze. Importing data requires connecting to the data source holding the data, specifying the data you need from the data source (by selecting a database table, view, or writing a query), and then importing the data into the client tool. The following steps will describe how to import the complete datasets from the PI SQL Client data models defined in the previous sections.

Directed Activity – Importing Data Using PI SQL Client

In this part of the class, you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

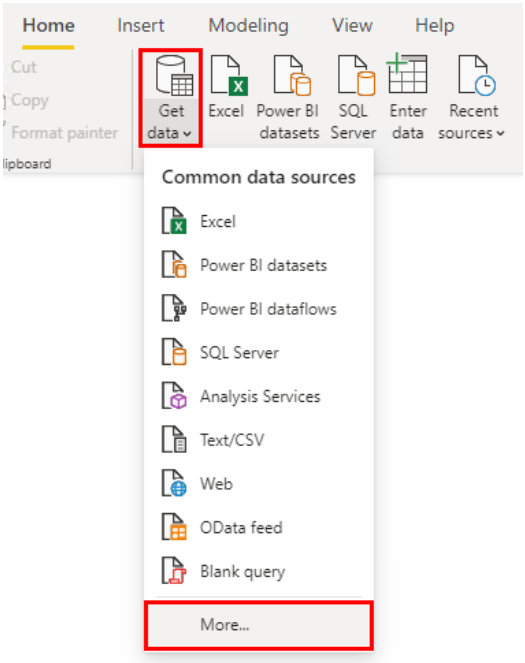
Objective:

Import PI SQL Client query results into Power BI.

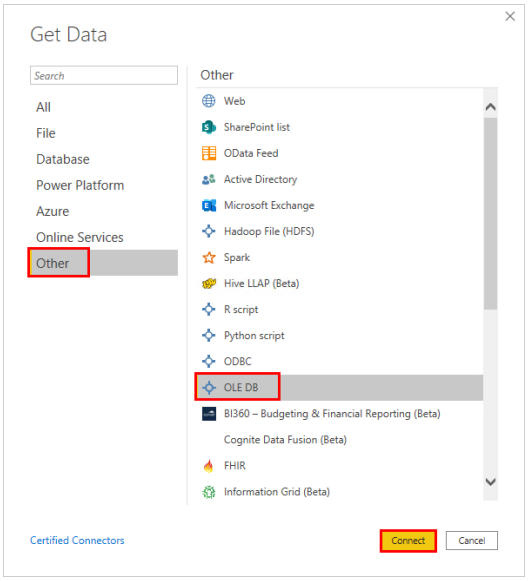
Approach:

Open Power BI and start a new report.

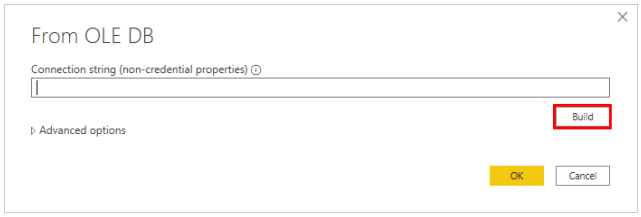
Rather than using the SQL Server provider we'll use the PI SQL Client provider which is found under Get data -> More:



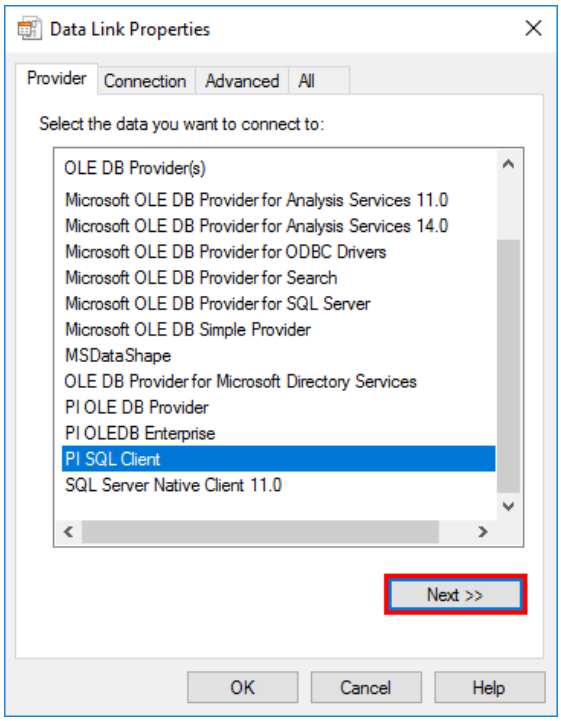
Select Other -> OLE DB, Connect:



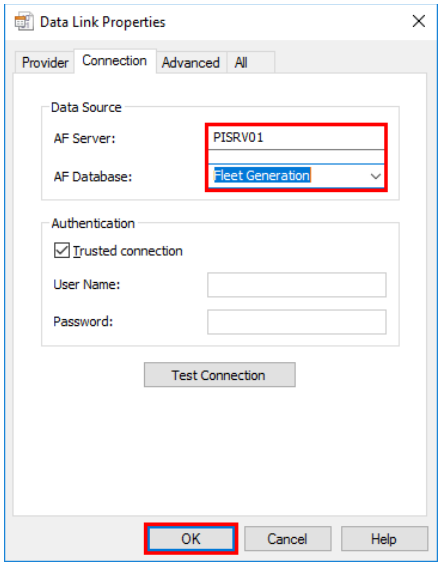
Select Build to build a connection string:



Select PI SQL Client, Next:

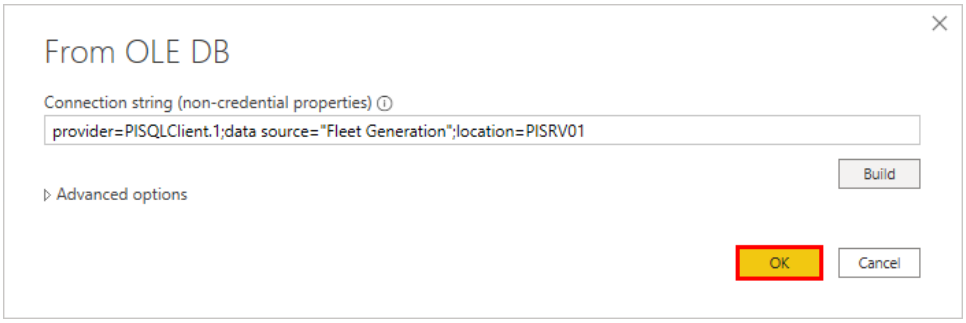


Enter PISRV01 and Fleet Generation, OK:

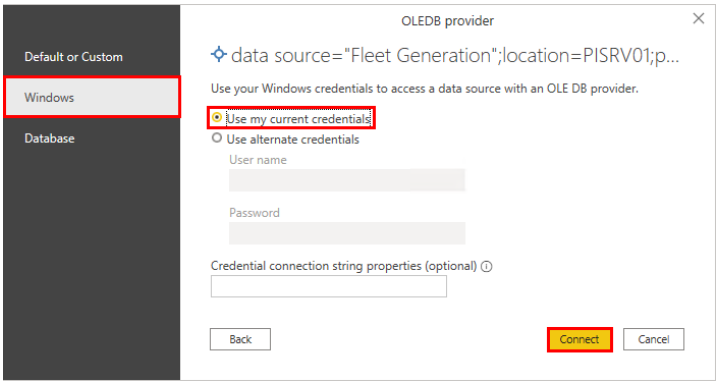


At this point you have a couple options, you can expand Advanced and paste in a query or click through and use a Wizard to pick from the available Views. We'll show the wizard first.

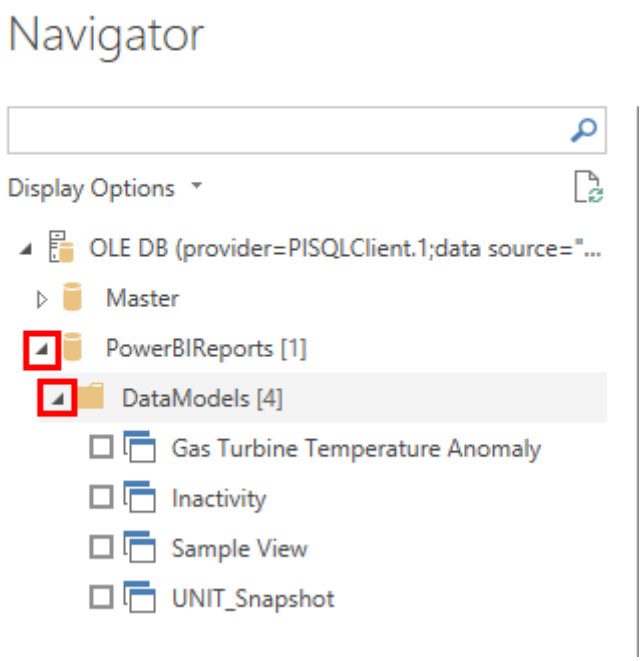
Click OK:



Select **Windows Security** and **Use my current credentials**, Connect:

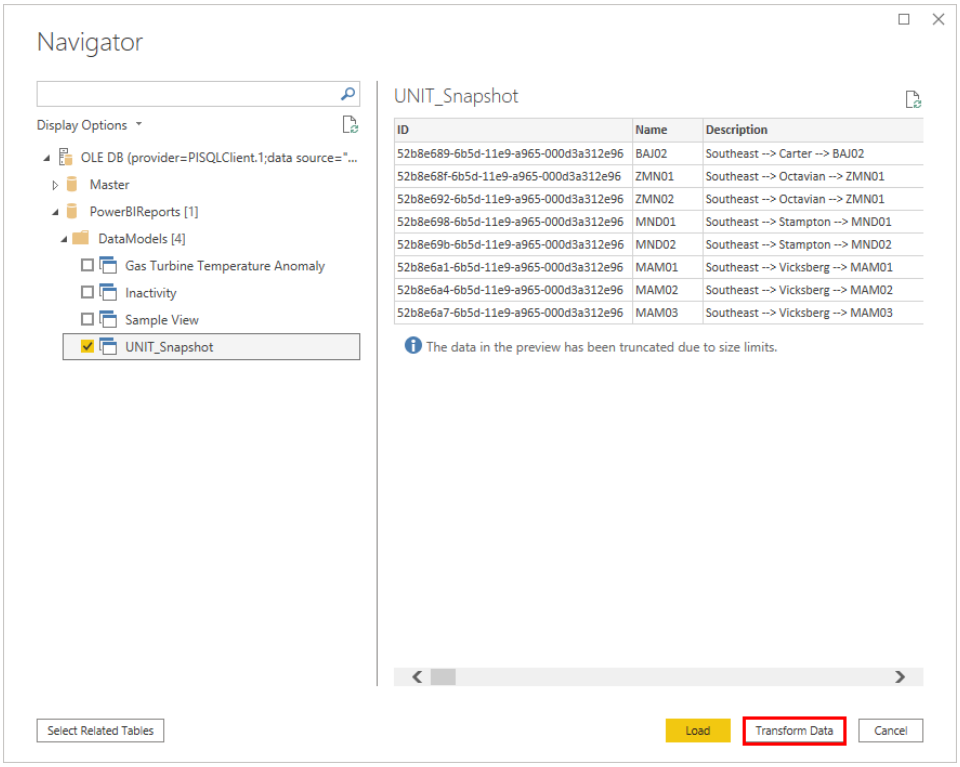


Expand PowerBIReports -> DataModels and the Views will be available for selection:

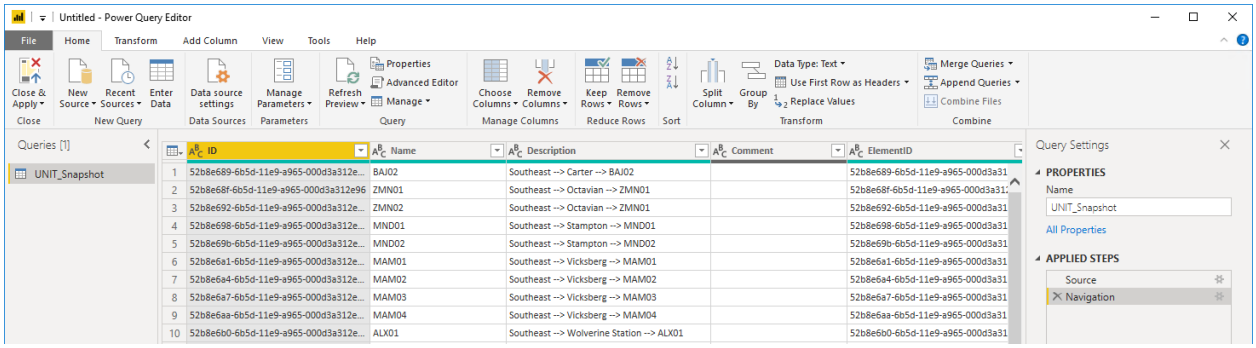


Check the box next to UNIT_Snapshot. From here you can click Load to import all columns, which is fine but will import a lot of columns we won't ever use.

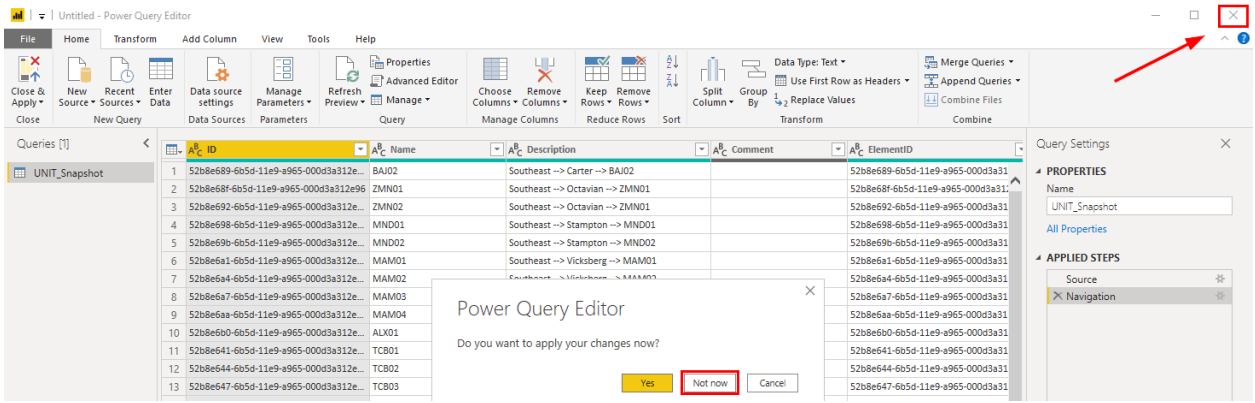
If you choose Transform Data, you'll have the opportunity to perform a variety of operations, notably removing columns.



Transform Data opens the Power Query Editor.



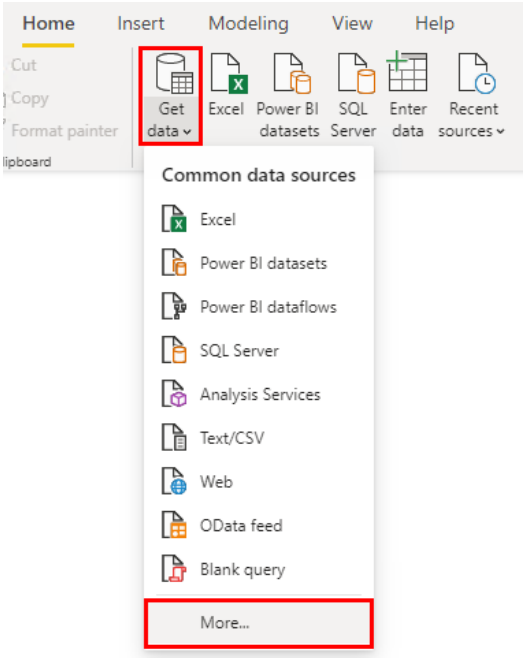
But we're not going to use this just now. **Close the Power Query Editor**, click **Not now**.



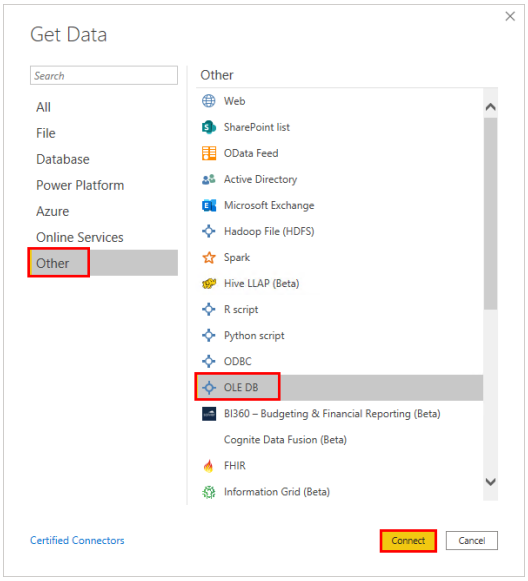
Ignore the banner. You can kill it but it will just come back. We'll eventually be discarding this Power BI file and starting fresh in the next Chapter.

Now we go through the entire Get data process again to get to the point where we paste in a query instead of selecting a View.

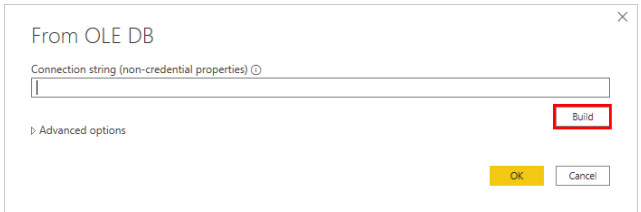
Get data -> More:



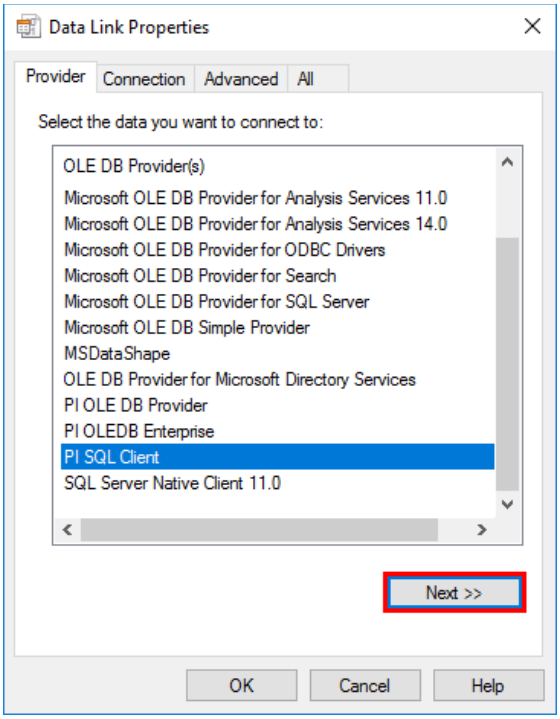
Select Other -> OLE DB, Connect:



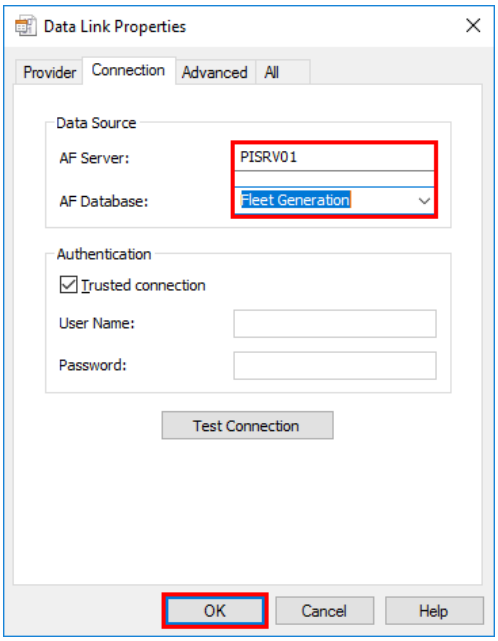
Select Build to build a connection string:



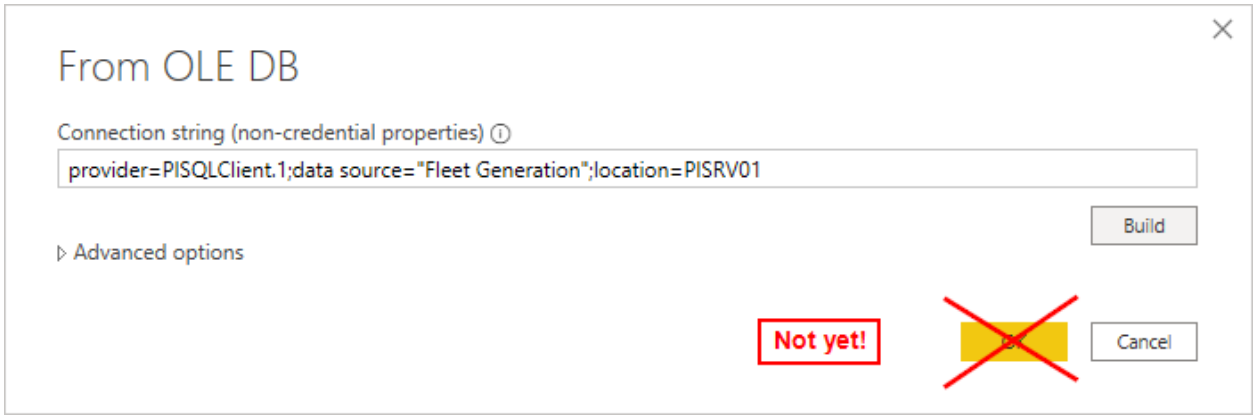
Select PI SQL Client, Next:



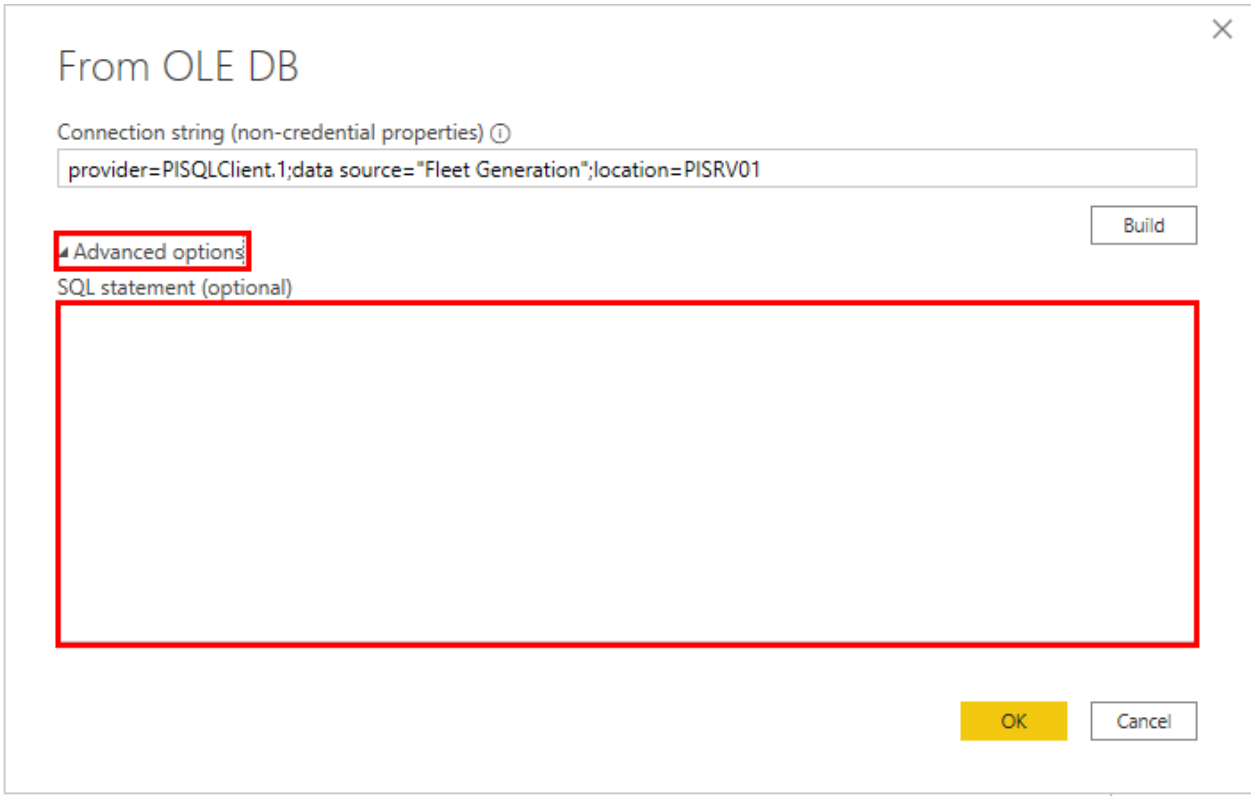
Enter PISRV01 and Fleet Generation, OK:



Don't click OK at the next screen or you'll have to start the get data process over again!



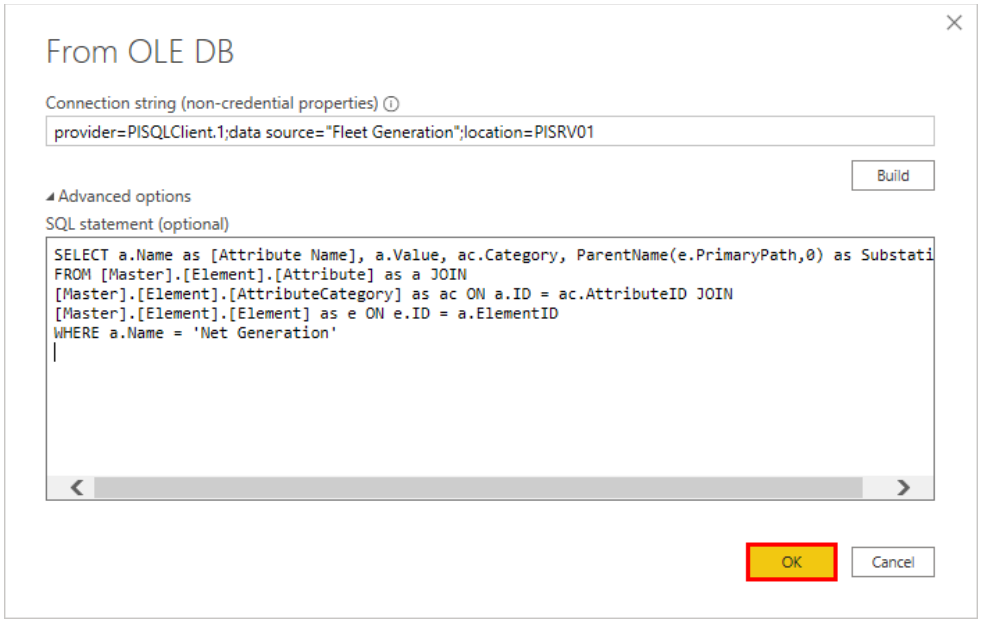
Expand Advanced options, you can paste in any query from PI SQL Commander in the SQL Statement field:



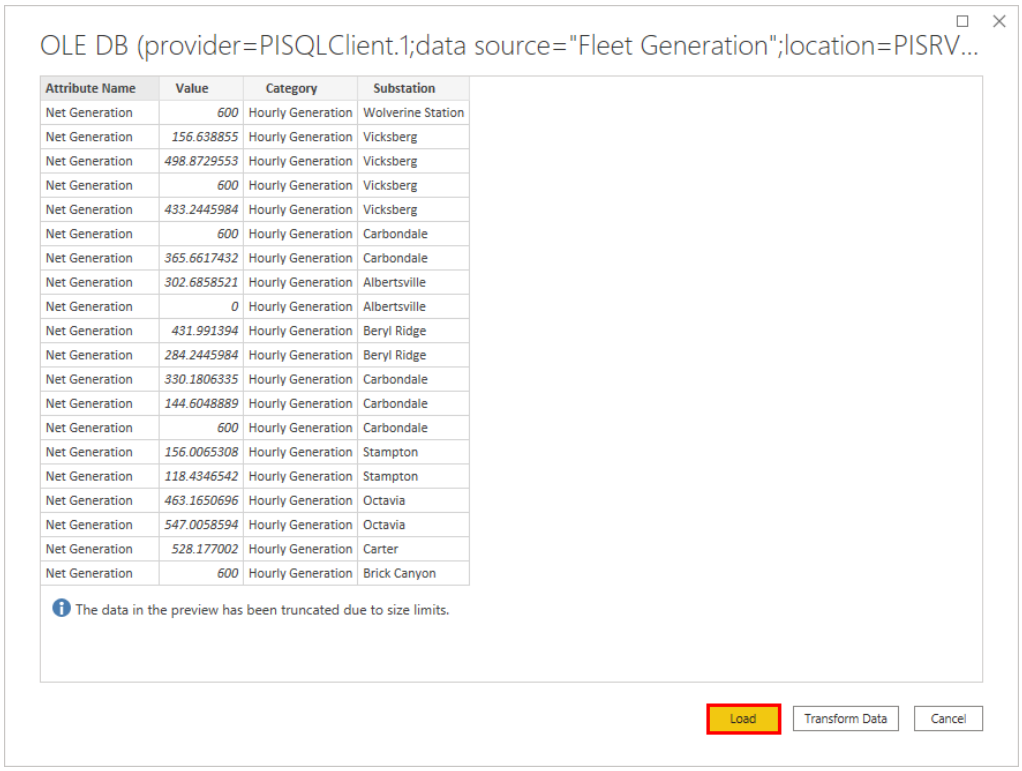
Paste in the below query:

```
SELECT a.Name as [Attribute Name], a.Value, ac.Category,
ParentName(e.PrimaryPath,0) as Station
FROM [Master].[Element].[Attribute] as a JOIN
[Master].[Element].[AttributeCategory] as ac ON a.ID = ac.AttributeID JOIN
[Master].[Element].[Element] as e ON e.ID = a.ElementID
WHERE a.Name = 'Net Generation'
```

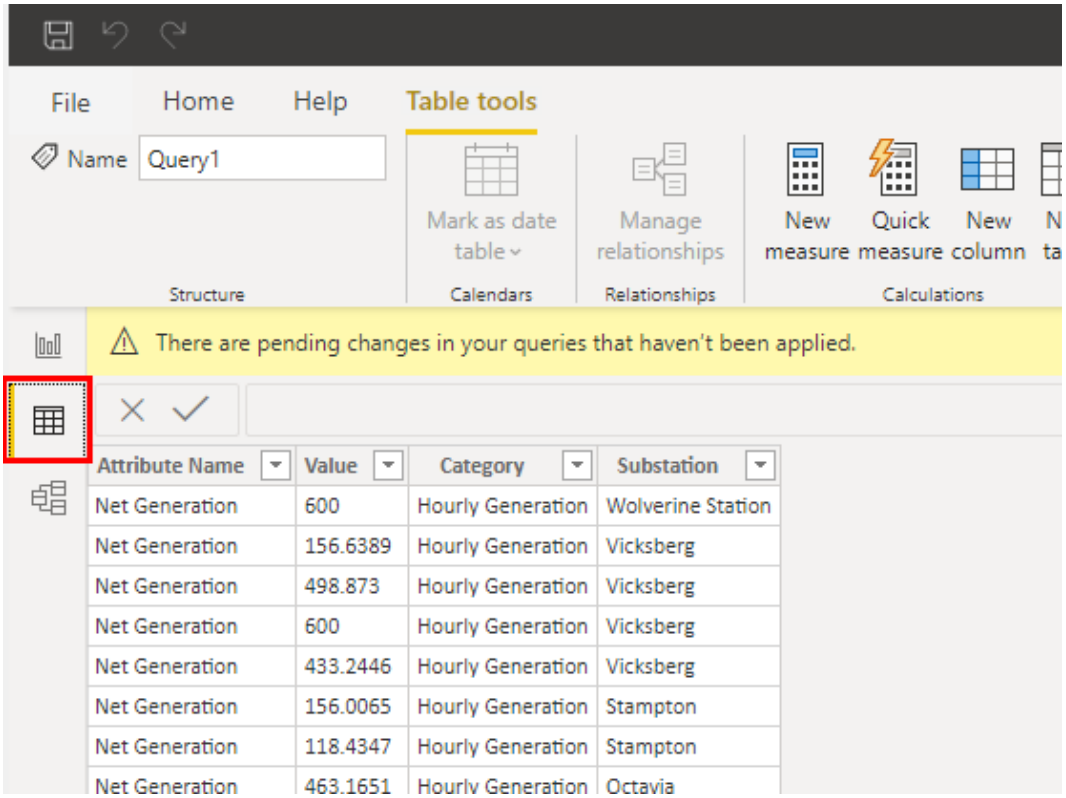
Now you can click OK:



Click Load.



Select the Data view and confirm the data has been imported:



Attribute Name	Value	Category	Substation
Net Generation	600	Hourly Generation	Wolverine Station
Net Generation	156.6389	Hourly Generation	Vicksberg
Net Generation	498.873	Hourly Generation	Vicksberg
Net Generation	600	Hourly Generation	Vicksberg
Net Generation	433.2446	Hourly Generation	Vicksberg
Net Generation	156.0065	Hourly Generation	Stampton
Net Generation	118.4347	Hourly Generation	Stampton
Net Generation	463.1651	Hourly Generation	Octavia

Close the report without saving. We'll start fresh in the next chapter.

Discussion

This is a discussion designed to maximize learning in a specific topic area. Your instructor will have questions, and will prompt for communication within the class. This is an open ended section and the result depends on your needs.

Objective:

Discuss differences between PI Integrator for BA and PI SQL Client

Approach

- Which method do you prefer to create views? PI Integrator for Business Analytics or PI SQL Client?
- Pros and Cons of both systems?
- What format would we like the data to be in for processing by BI clients?
- What should be added to the SQL queries to improve the format?
- Do these queries match what we want in our reports?
- If not, what is lacking?

Estimated Completion time 10 minutes.

Appendix B. SSRS Reports using PI SQL Client/RTQP

SQL Server Reporting Services is a builtin feature of Microsoft SQL Server that provides a web-based portal for hosting reports that leverage relational data sources. It's not flashy, but it meets a lot of reporting requirements. As we will see, SSRS reports are more difficult to configure than Power BI reports. A big benefit of SSRS is that it is likely already installed and available at your organization. If you're lucky enough to have admin rights on your own SQL Server, then you could get started without going to IT for additional licenses.

We will be using Microsoft SQL Server Report Builder to build a simple report and publish this report to the SSRS Reports portal.

Directed Activity – PI AF Hierarchy and Data Set

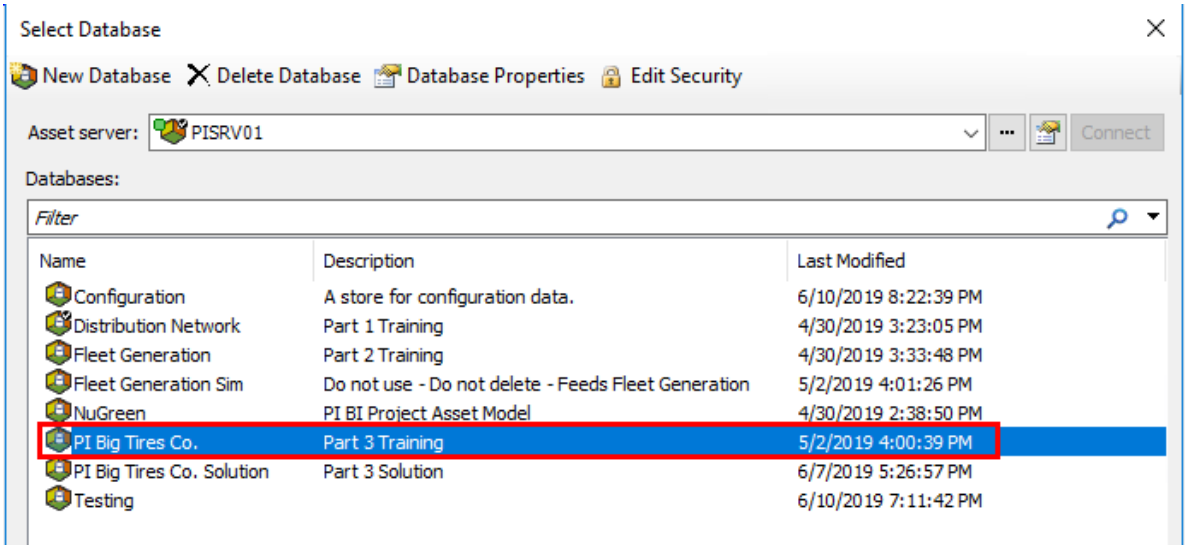
In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

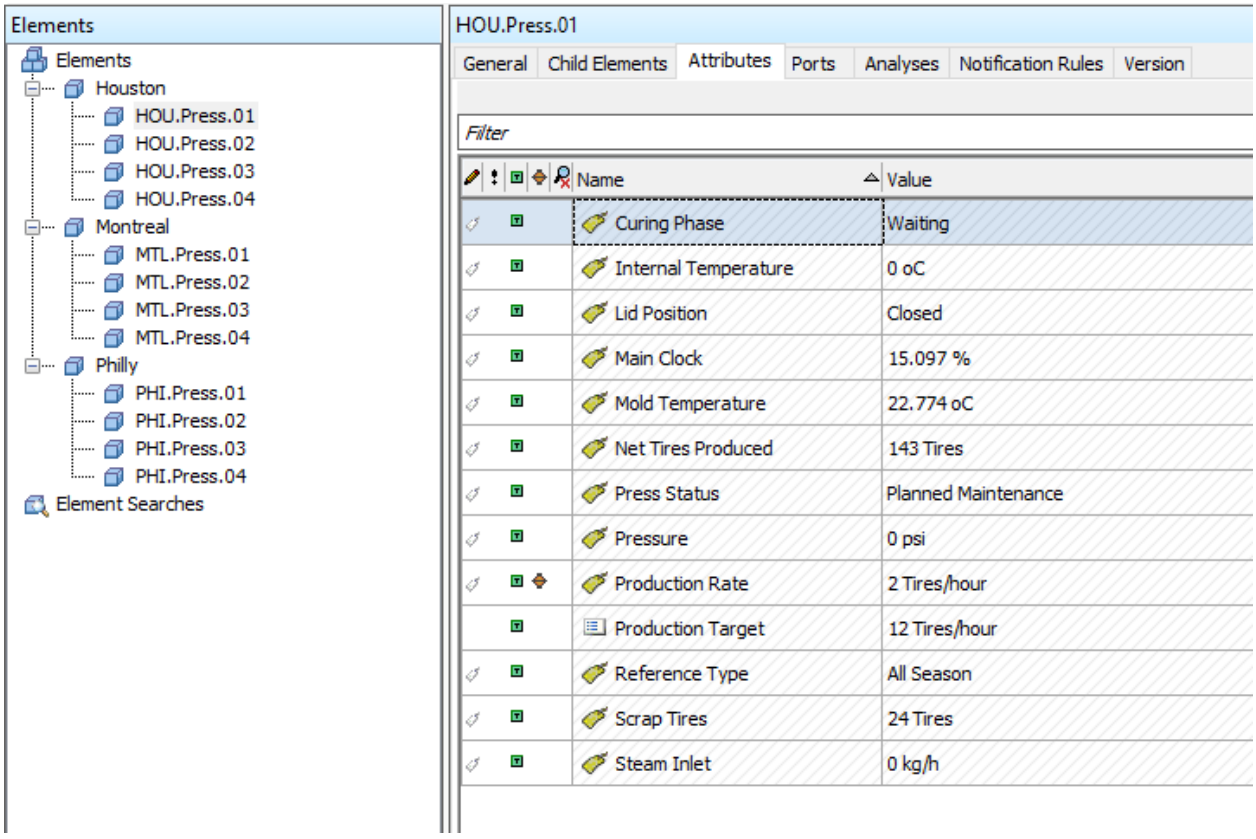
- Better understand the data set used in the following chapters

Approach:

We will take a few minutes to review the PI Big Tires Co. PI AF Database. We are going to build upon the Tire Press example, which you may recognize from the Building PI System Assets and Analytics with PI AF Class. Open **PI System Explorer** and navigate to the **PI Big Tires Co. database**.



The database is loosely based on a workshop one of our Systems Engineers did with a tire company. It's a basic hierarchy with 12 Tire Presses.



Here is how a Tire Curing Press works: raw tires are loaded individually into a Tire Curing Press. Once the tire is loaded, the press closes up and temperature and pressure is applied to cook and mold the tire. After the cooking time has elapsed, the press opens up and the tire is unloaded into a cooling unit where fans blow air until the tire reaches a specific temperature.

Preparing the Queries Needed for the Report

We'll prepare the queries necessary to build a simple SSRS report for tire production data.

Exercise – Get a List of Sites

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective:

Write a PI SQL Client query that returns all elements that use the Site Template. This will be used later during the report configuration. The desired data set is:

	Site
1	Houston
2	Montreal
3	Philly

Approach:

- Connect to the PI Big Tires Co. database in PI SQL Commander
- In the Query Compendium, find a sample query that returns all Elements that are based on a certain template.
- Include only the Name column and change the template to 'Site'.
- Optionally use the ORDER BY statement to sort the output alphabetically

Exercise – Get a List of Presses by Site

This solo or group exercise is designed to maximize learning in a specific topic area. Your instructor will have instructions, and will coach you if you need assistance during the exercise.

Objective:

Write a PI SQL Client query that returns all elements that use the PressTemplate template and the site they belong to. This will be used later during the report configuration. The desired data set is:

	Press	Site
1	HOU.Press.01	Houston
2	HOU.Press.02	Houston
3	HOU.Press.03	Houston
4	HOU.Press.04	Houston
5	MTL.Press.01	Montreal
6	MTL.Press.02	Montreal
7	MTL.Press.03	Montreal
8	MTL.Press.04	Montreal
9	PHI.Press.01	Philly
10	PHI.Press.02	Philly
11	PHI.Press.03	Philly
12	PHI.Press.04	Philly

Approach:

- Start with the sample query from the previous exercise.
- Remove the Description column and change the template to 'PressTemplate'.
- Use the ParentName() function to get the Site associated with each Press.
- Optionally use the ORDER BY statement to sort the output alphabetically by Press

Directed Activity – Daily Production Data

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective:

Get the daily Net Tires Produced and Scrap Tires values, which will be used later in an SSRS Report.

Approach:

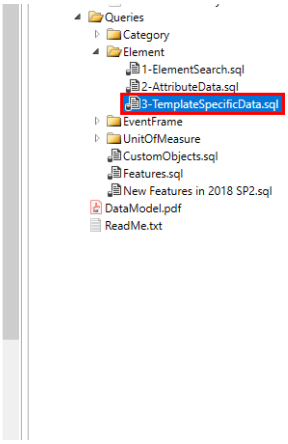
The desired result set is one row per day for the past several days. The columns will be Press, the Date, Net Tires Produced, and Scrap Tires.

We will start by modifying a query from the query compendium that returns sampled data.

Double click on 3-TemplateSpecificData.sql and find the query for Sampled Boiler attribute values

```
        'FuelGasFlow_Value', -- ValueColumnName - Required
        'FuelGasFlow_UnitOfMeasure', -- UnitOfMeasureColumnName - Optional
        'FuelGasFlow_Error', -- ErrorColumnName - Optional
        'm3/s' -- UnitOfMeasure - Optional
    } -- AttributeMetadata - defines which columns to expose for a particular attribute template
-- ,{
--  } -- AttributeMetadata
> v
ON e.ID = v.ElementID
WHERE e.Template = 'Boiler'
```

```
-- Sampled Boiler attribute values
SELECT e.Name, sv.*
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
    'Boiler', --Template - element template name
    {
        '|Fuel Gas Flow', -- AttributeTemplatePath - Required
        'FuelGasFlow_Value', -- ValueColumnName - Required
        'FuelGasFlow_UnitOfMeasure', -- UnitOfMeasureColumnName - Optional
        'FuelGasFlow_Error', -- ErrorColumnName - Optional
        NULL -- UnitOfMeasure - Optional
    } -- AttributeMetadata - defines which columns to expose for a particular attribute template
-- ,{
--  } -- AttributeMetadata
>(e.ID, 'y', 't', '1h') sv
WHERE e.Template = 'Boiler'
```



Start a new query, and copy/paste the above sample.

Modify the query to use PressTemplate and the Net Tires Produced and Scrap Tires attributes. Remove unnecessary attribute properties. Your query should then be:

```
SELECT e.Name, sv.*
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_Value'
   }
>(e.ID, 'y', 't', '1h') sv
WHERE e.Template = 'PressTemplate'
```

Change the start time to ‘T-6d-1s’, end time to ‘T-1s’, and sample interval to ‘1d’. This will sample at the close of the day (11:59:59 PM) before the count resets to zero, while still giving the date the production occurred in, rather than the day after.

Execute the below query which includes the aforementioned modifications.

```
SELECT e.Name, sv.*
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_Value'
   }
>(e.ID, 'T-6d-1s', 'T-1s', '1d') sv
WHERE e.Template = 'PressTemplate'
```

Format the TimeStamp column to remove the hours, minutes, and seconds using the Format() function and add some aliases to improve the column names.

```
SELECT e.Name as Press,
sv.[Net Tires Produced],
sv.[Scrap Tires],
FORMAT(TimeStamp, 'dd-MMM-yy') as Date
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced'
   },
   {
      '|Scrap Tires',
      'Scrap Tires'
   }
>(e.ID, 'T-6d-1s', 'T-1s', '1d') sv
WHERE e.Template = 'PressTemplate'
```

There is a requirement for the report that the current tire production be displayed on the report for the bottom row using today's date. GetSampledValues uses interpolation, and hence can't provide this value. We will have to UNION the above query with another query that provides the current values.

A similar process to the above can be used to get the current values. You can just copy and paste this from the workbook .docx in the class folder.

```
SELECT e.Name as Press,
v.[Net Tires Produced_Value] as [Net Tires Produced],
v.[Scrap Tires_Value] as [Scrap Tires],
FORMAT(DATETIME('T'),'dd-MMM-yy') as Date
FROM Master.Element.Element e
INNER JOIN Master.Element.Value
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_TimeStamp',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_TimeStamp',
      'Scrap Tires_Value'
   }
> v
ON e.ID = v.ElementID
WHERE e.Template = 'PressTemplate'
```

Finally, UNION the above two queries together and order the results by Press and Date. The final query is then:

```
SELECT e.Name as Press,
sv.[Net Tires Produced],
sv.[Scrap Tires],
FORMAT(TimeStamp, 'dd-MMM-yy') as Date
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced'
   },
   {
      '|Scrap Tires',
      'Scrap Tires'
   }
>(e.ID, 'T-6d-1s', 'T-1s', '1d') sv
WHERE e.Template = 'PressTemplate'
UNION
SELECT e.Name as Press,
v.[Net Tires Produced_Value] as [Net Tires Produced],
v.[Scrap Tires_Value] as [Scrap Tires],
FORMAT(DATETIME('T'),'dd-MMM-yy') as Date
FROM Master.Element.Element e
INNER JOIN Master.Element.Value
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_TimeStamp',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_TimeStamp',
      'Scrap Tires_Value'
   }
> v
ON e.ID = v.ElementID
WHERE e.Template = 'PressTemplate'
ORDER BY Press, Date
```

Building the Tire Production Report

We will use Report Builder as the report-authoring tool for the tire production report. The client itself is a free download from Microsoft and there are numerous tutorials available online. Let's dive into the report configuration and learn by example.

Report Builder and SSRS Basics

We'll create a blank report and review basic functionality.

Directed Activity – New Report

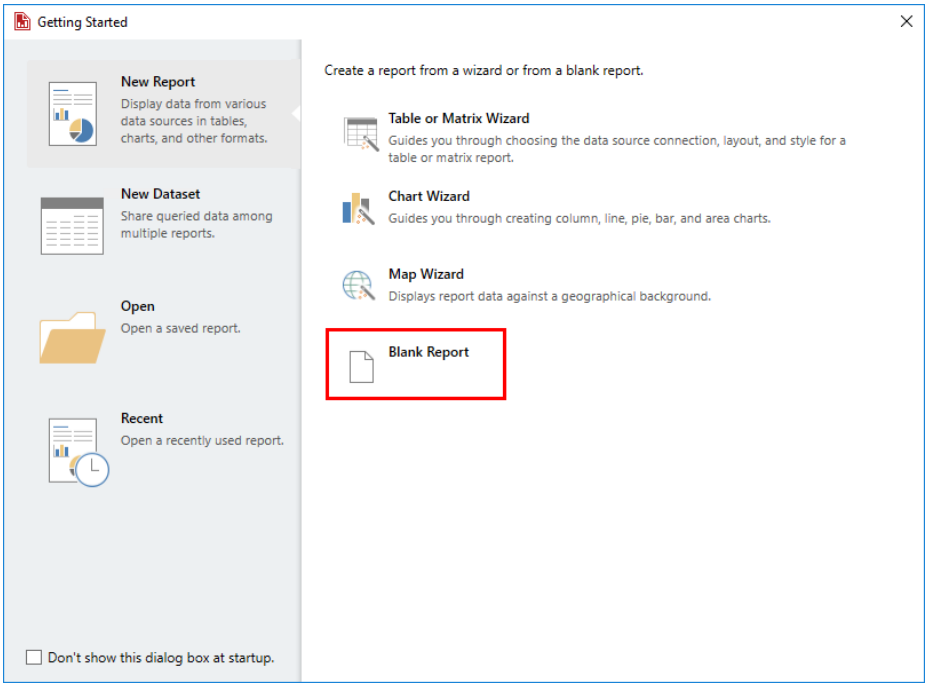
In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective: Publish a blank report with a title to the reports portal

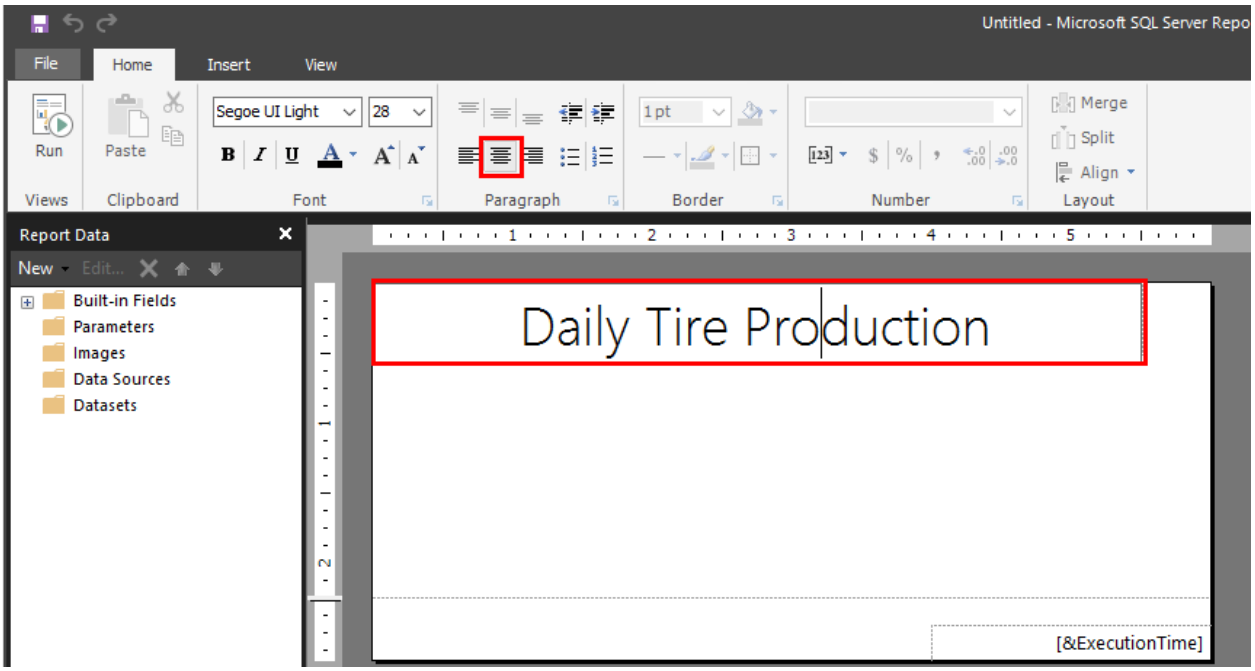
Approach:

Launch Report Builder from the Start Menu, Taskbar or the desktop shortcut

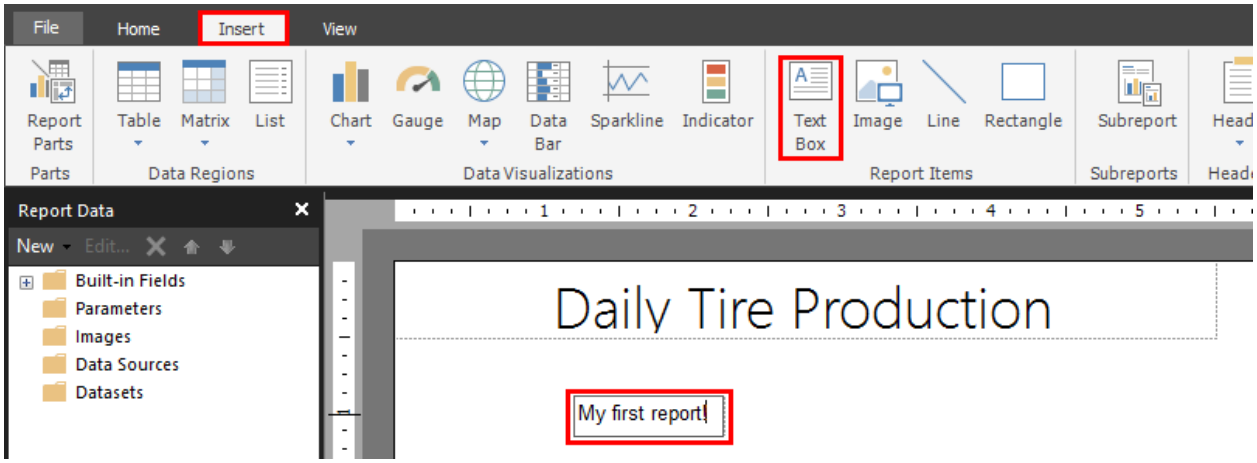
It will take a little while to connect to the report server on first launch. Eventually you'll be prompted to open or create a report. Select Blank Report.



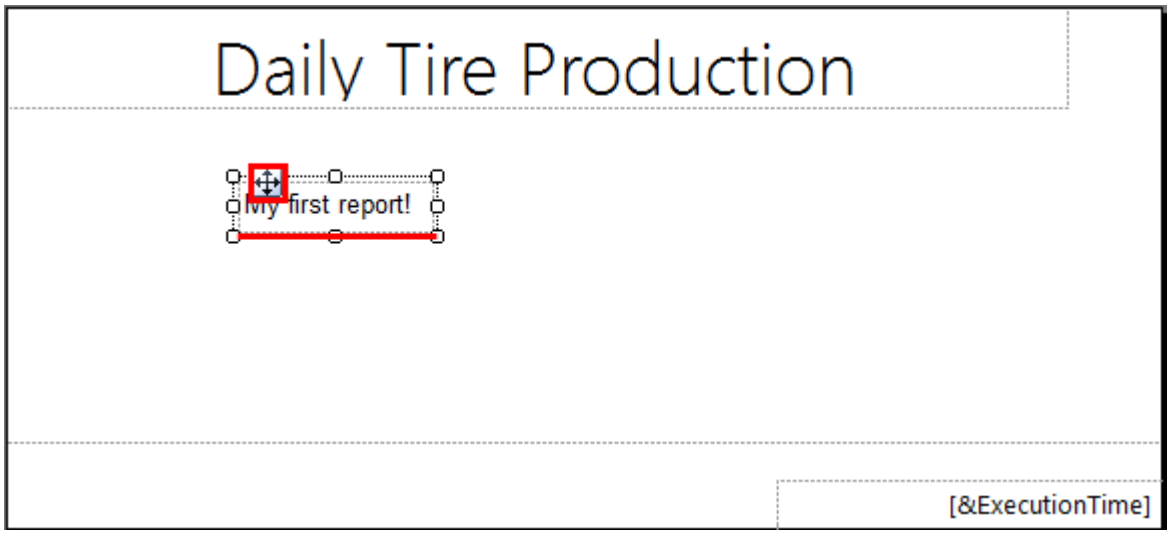
Click the Title box and enter the title "Daily Tire Production" and center the text. Typical text editing capabilities are available on the Home tab.



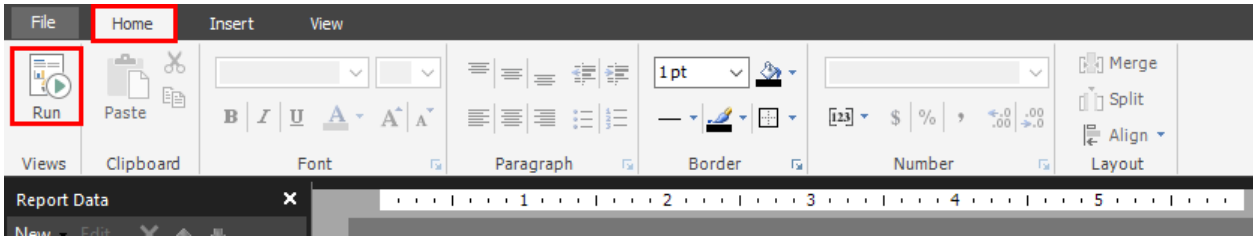
The Insert Tab includes all the objects that can be included in the report. Most of the visuals require data which we have not yet imported. For now add a text box and enter whatever text you'd like:



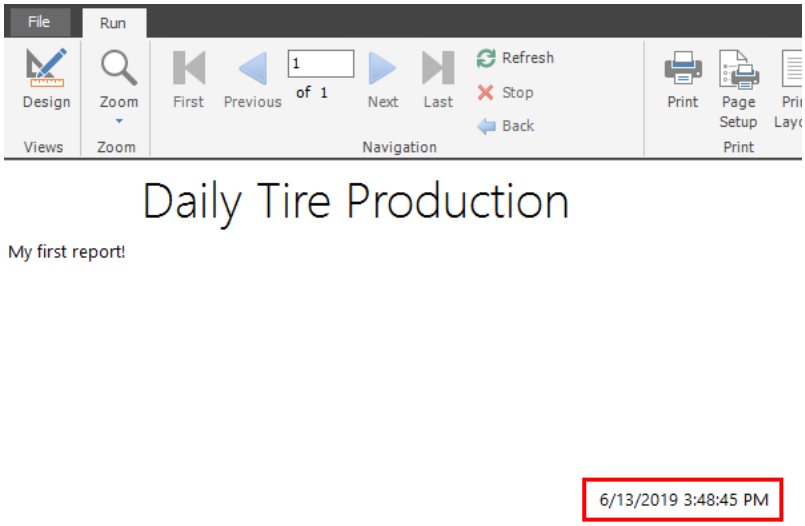
Reposition objects by clicking an edge, then grapping the positioning icon



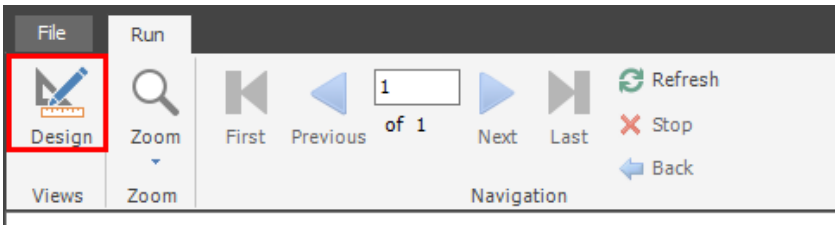
There's not much on the report, but we can do some sanity test. In the Home tab, **click Run** to render the report on the client. This is used during report development before the report is published to the report server.



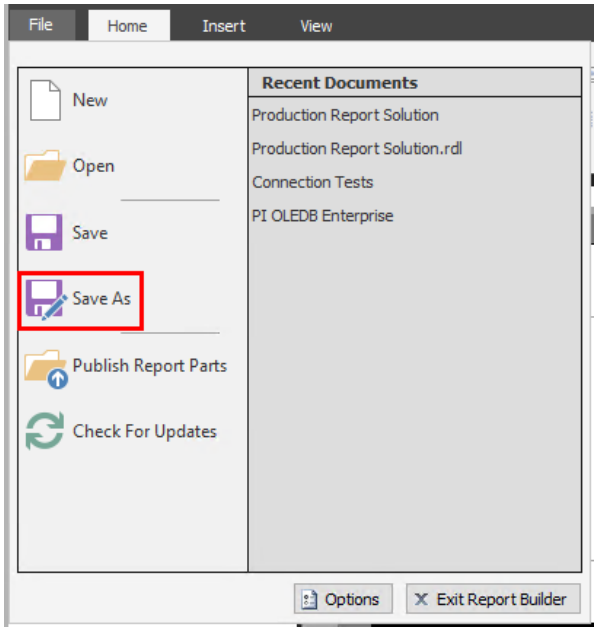
Note that the ExecutionTime placeholder shows the time the report was run



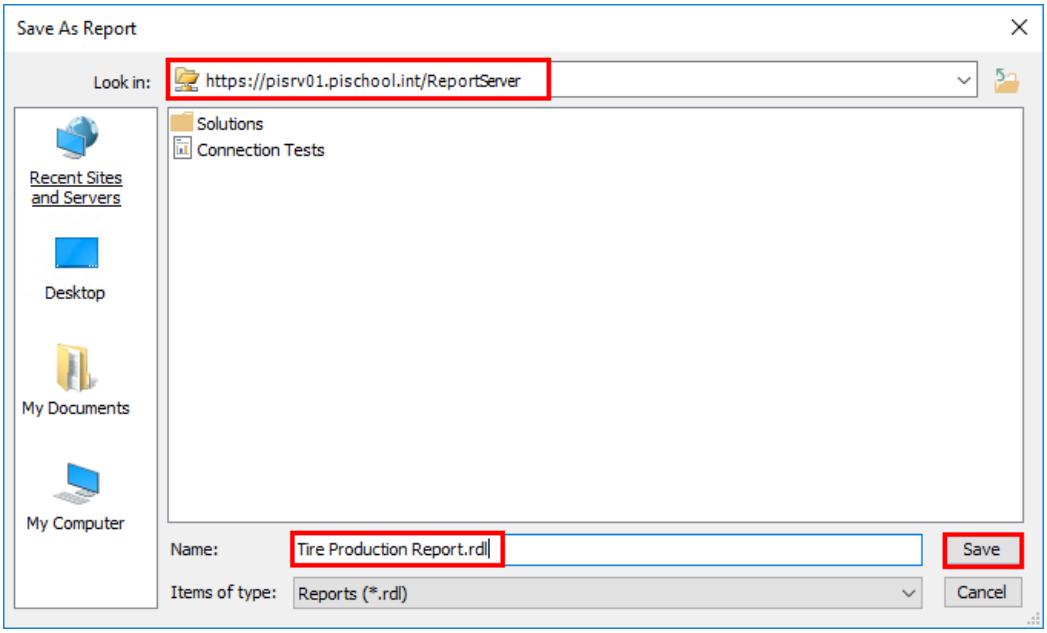
Click Design to go back to editing the report:



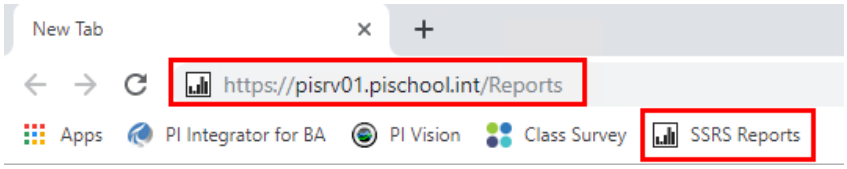
Publish the report to the Report Server by doing a **File -> Save As:**



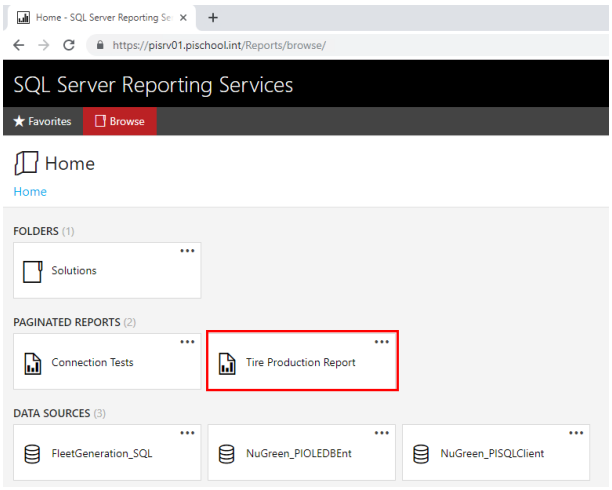
Name it Tire Production Report.rdl and save it to
https://pisrv01.pischool.int/ReportServer



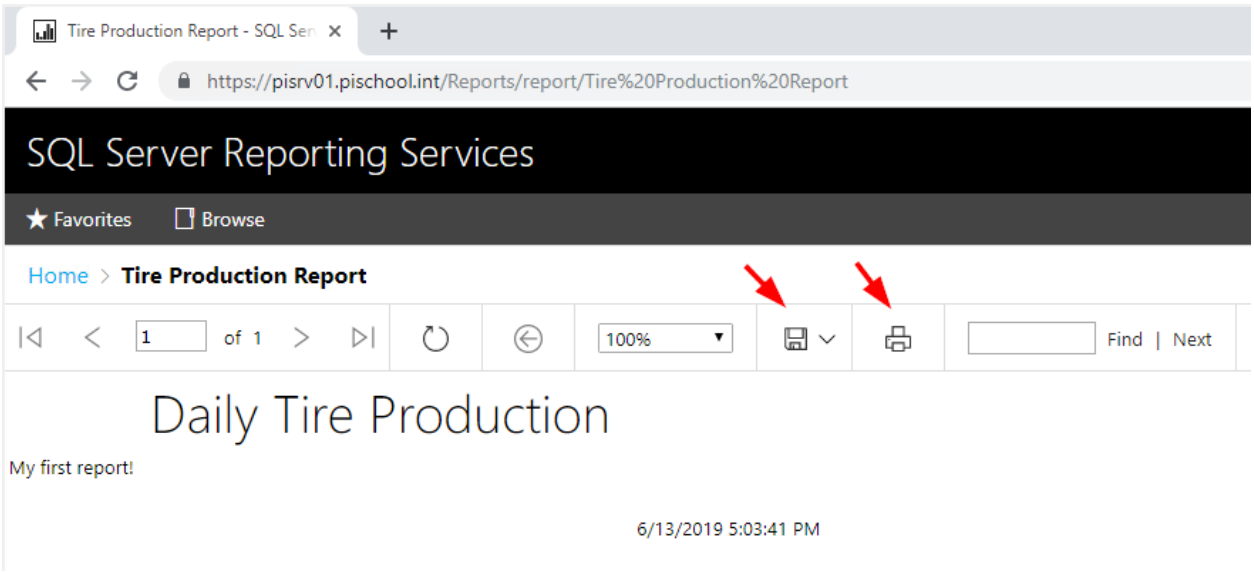
Now that the report has been saved, launch Chrome and navigate to https://pisrv01.pischool.int/Reports. There's a shortcut on the Desktop and the bookmark bar.



Once the Portal loads, open Tire Production Report



The report should then load. At this point you can download or print the report in a variety of formats, but most of the time you would just view the report online in the portal.



Directed Activity – Shared Data Source

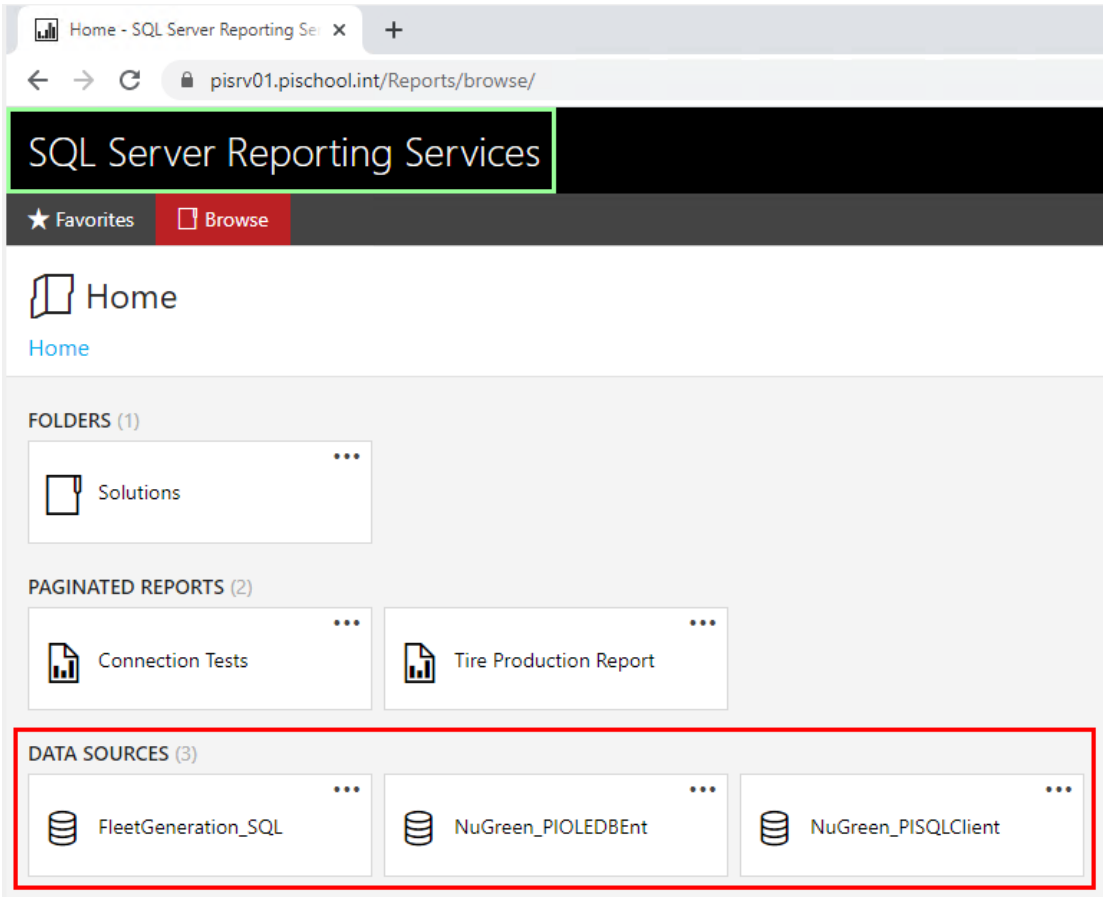
In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

Objective: Create a shared data source hosted in the report portal

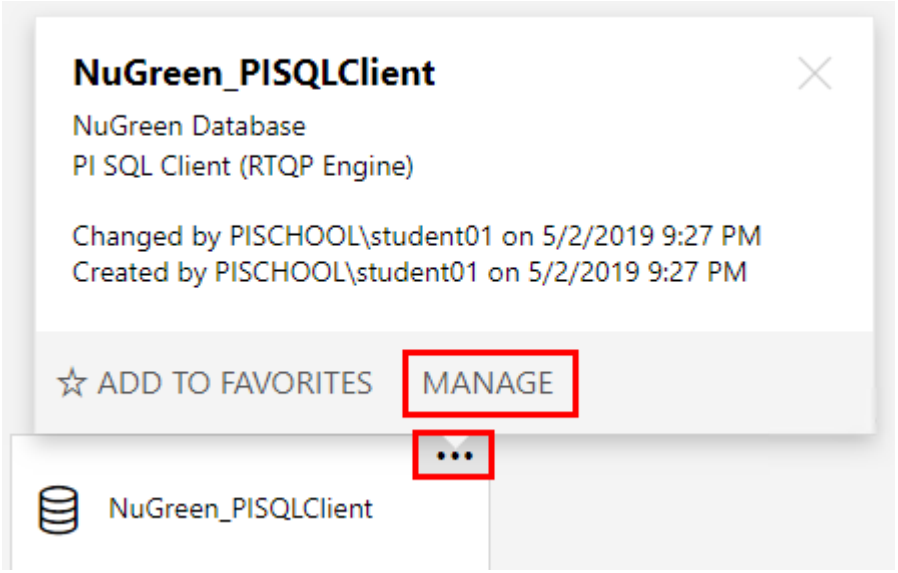
Approach:

In order to use the queries developed in the previous exercises, we will need to define a data source. It's a judgement call, but much of the time it's better to expose the data source through the portal so that it can be shared amongst several users and reports.

Go back to the SSRS Reports home page by clicking SQL Server Reporting Services**.** Three data sources are defined already: One for SQL Server, one for the NuGreen AF database via PI OLEDB Enterprise and another for the NuGreen AF database via PI SQL Client.

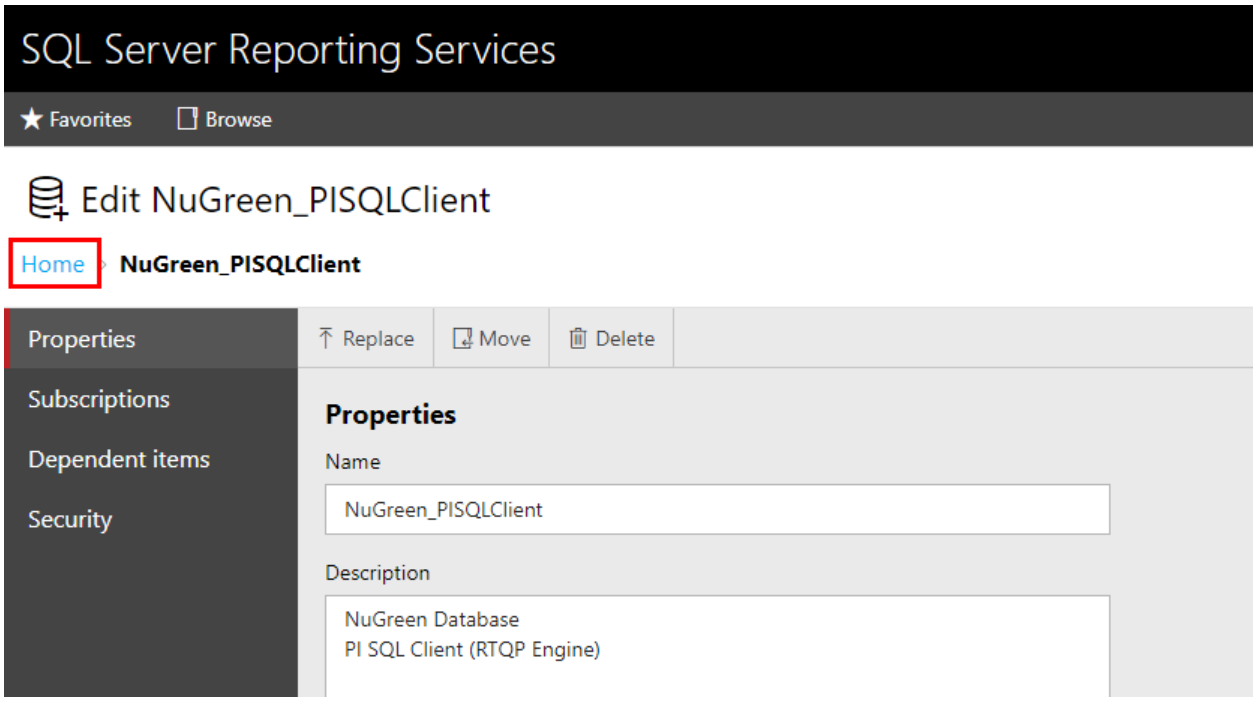


Before creating out own, let's inspect **NuGreen_PISQLClient**. Click the ellipses and select Manage.

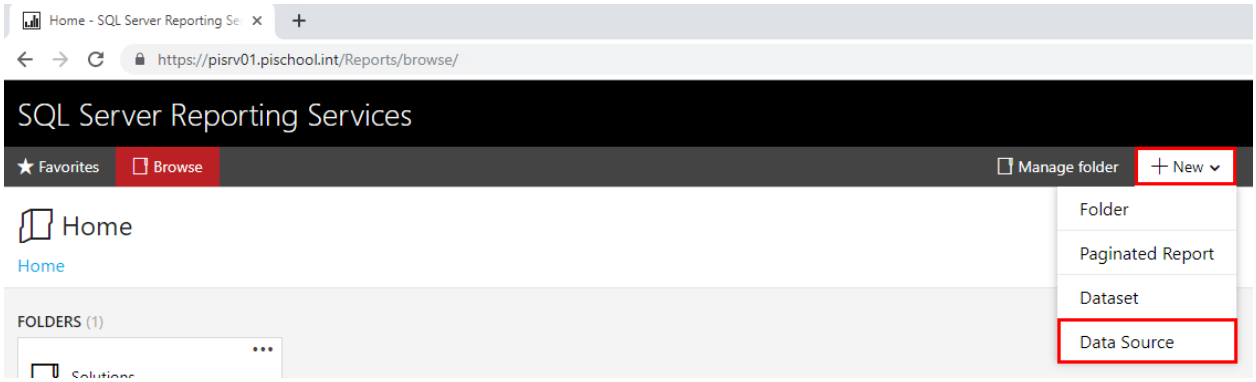


Note that the connection type is OLE DB and review the Connection string. **There's no GUI for configuring the Connection string here, so it has to be manually configured.** You might decide to copy an existing connection string and modify the relevant fields to make this easier.

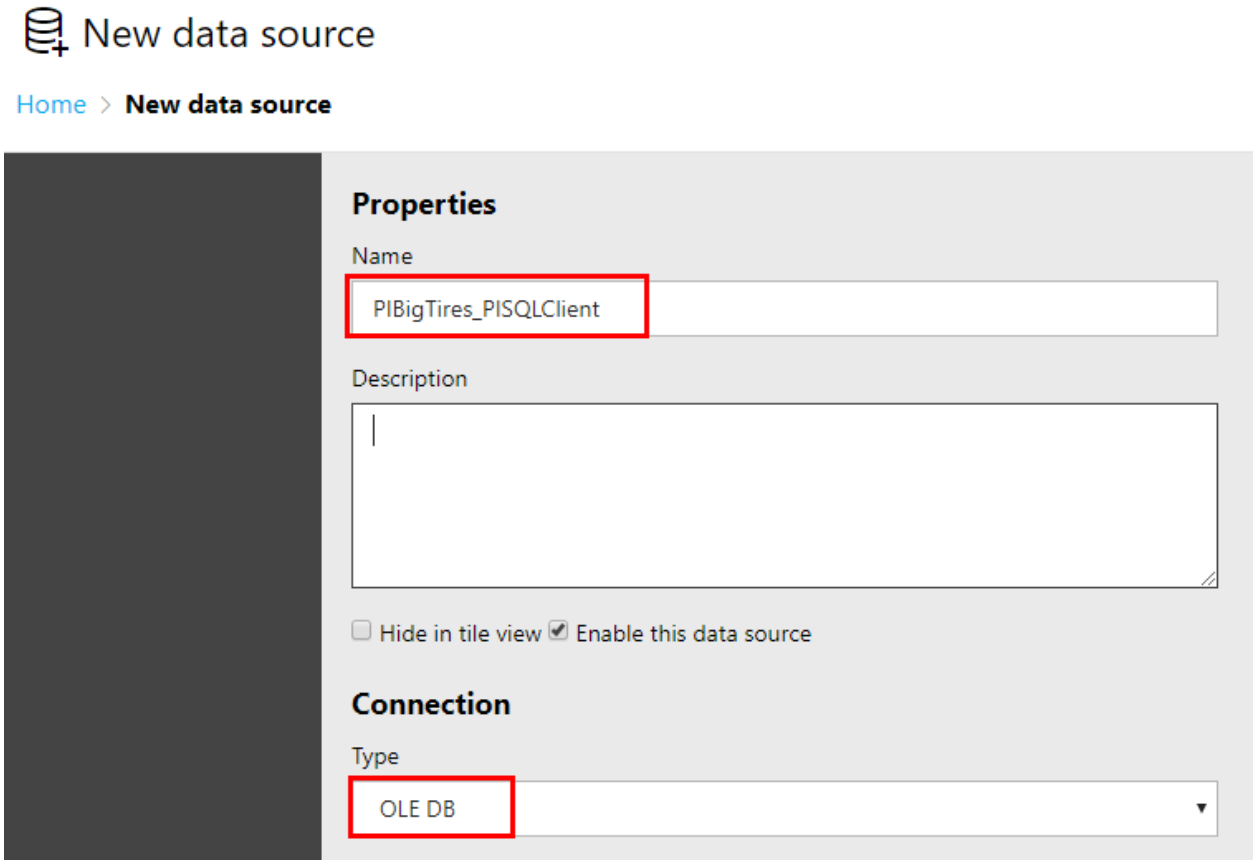
Click Home to go back



Go New->Data Source to create a new data source.



Enter **PIBigTires_PISQLClient** as the Name. Optionally enter a description.
Enter **OLE DB** as the Type.

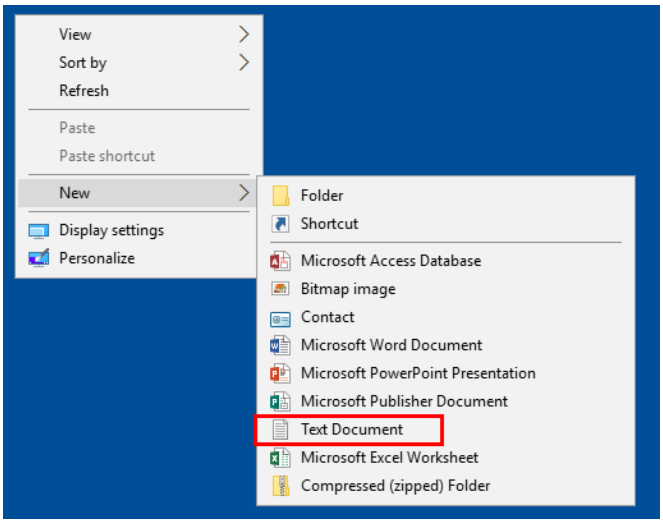


Since there is no GUI in the report portal for generating the connection string, we'll have to enter it manually or use an existing example.

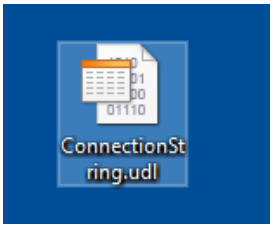
We could modify the connection string from the NuGreen_PISQLClient data source or Power BI, but let's say we don't have an existing connection string to work with.

One trick is to use a .udl file to build the connection string.

Right click on the desktop and create a new text file.

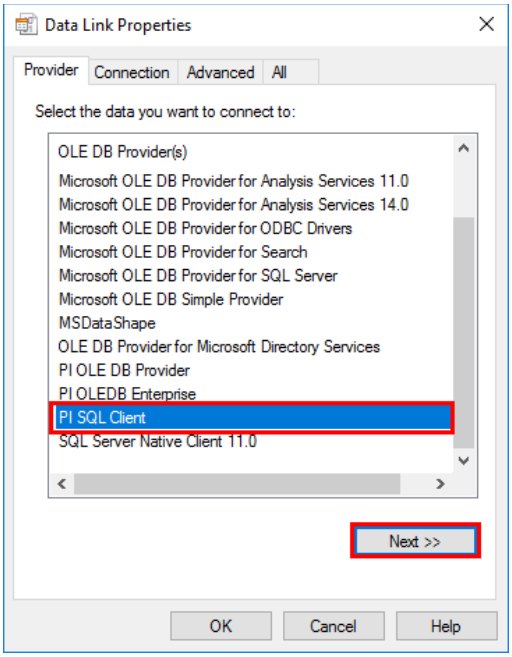


Name it ConnectionString.udl. **The .udl file extension here is key.** Windows should recognize the file type and change the icon.

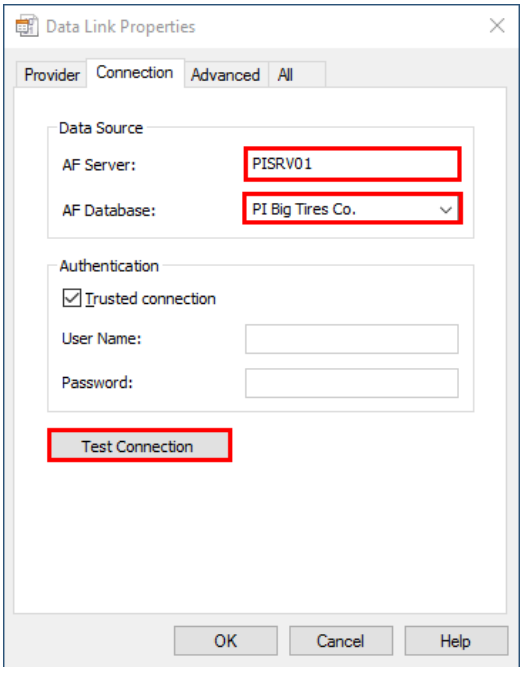


Open the file and the Data Link properties window should launch.

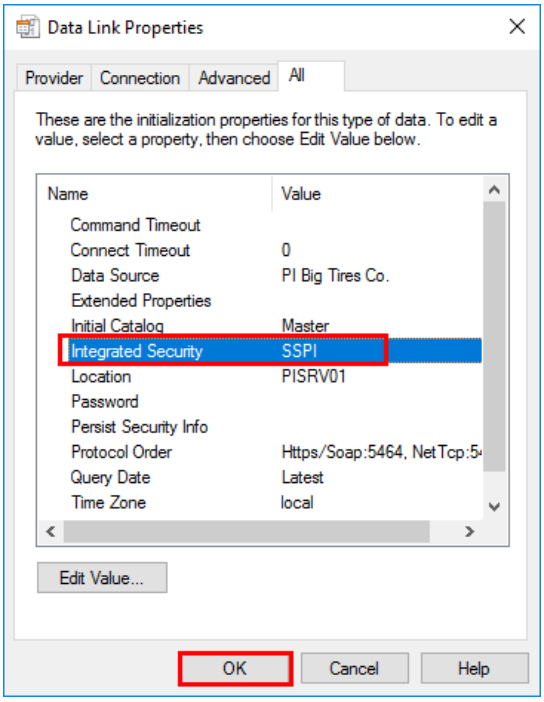
In the Provider tab, select PI SQL Client, Next



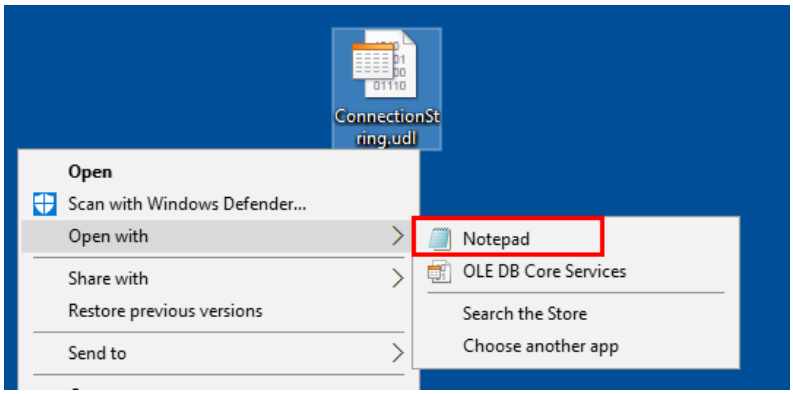
Enter PISRV01 as the AF Server and PI Big Tires Co. as the AF Database, then test the connection.



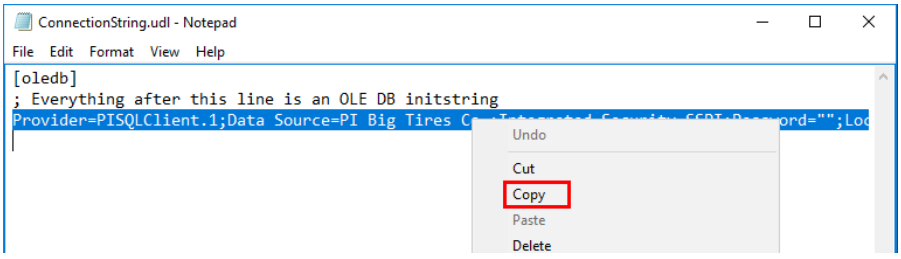
On the All tab, confirm that Integrated Security is set to SSPI. Click OK to finish editing the file.



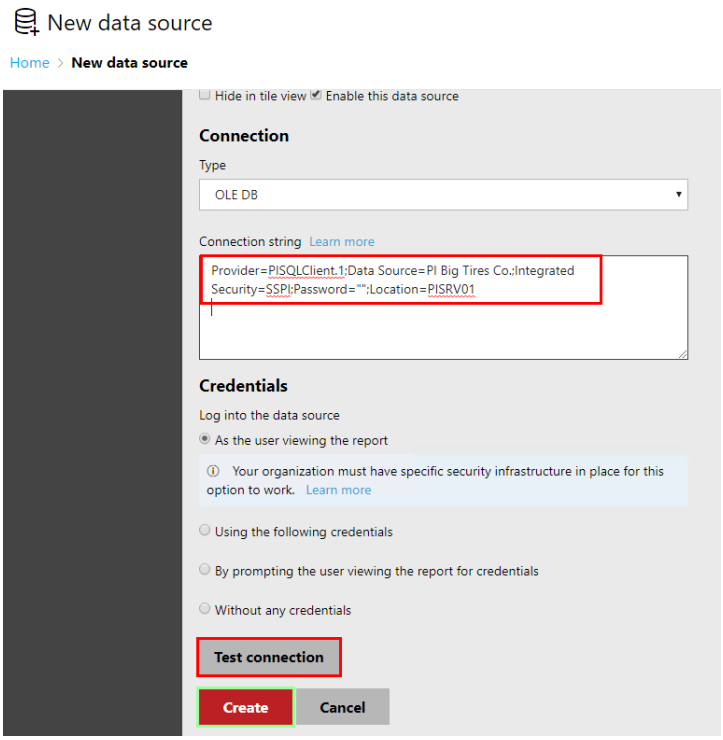
Now open ConnectionString.udl with a text editor to view the text, which is a connection string we can paste into the data source configuration.



Copy the 3rd line of text.



Paste it into the Data Source configuration, Test the connection, and click Create



The new data source will now be available to those with access to the report server and sufficient permissions.

Displaying Data on the Report

Now it's time to utilize the queries from the previous sections and build a basic daily production report.

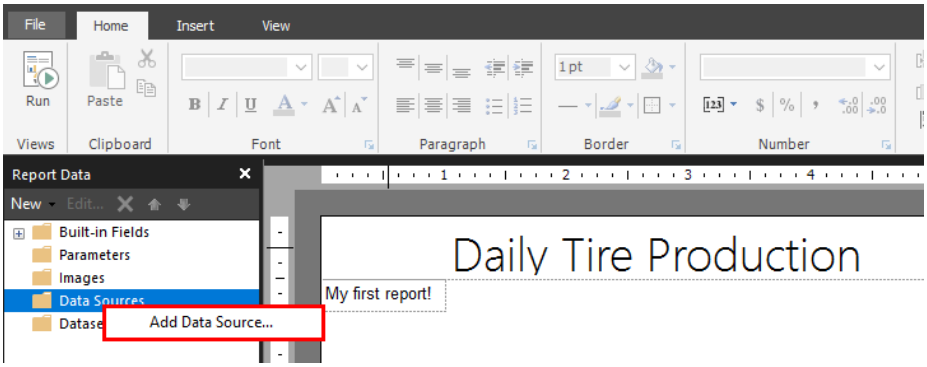
Directed Activity – Import Data and Display in a Table

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

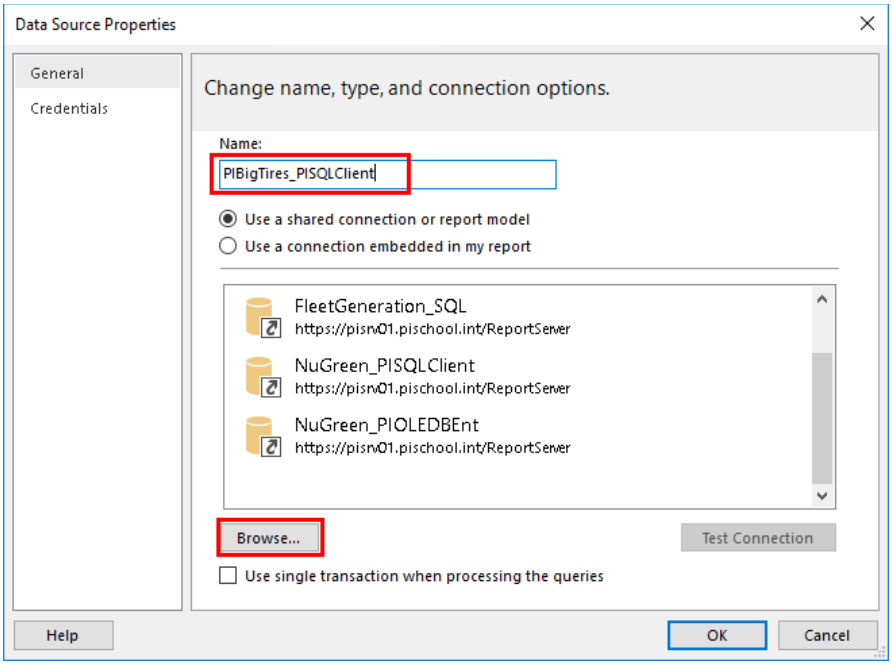
Objective: Import the daily production data and display the results in a simple table

Approach:

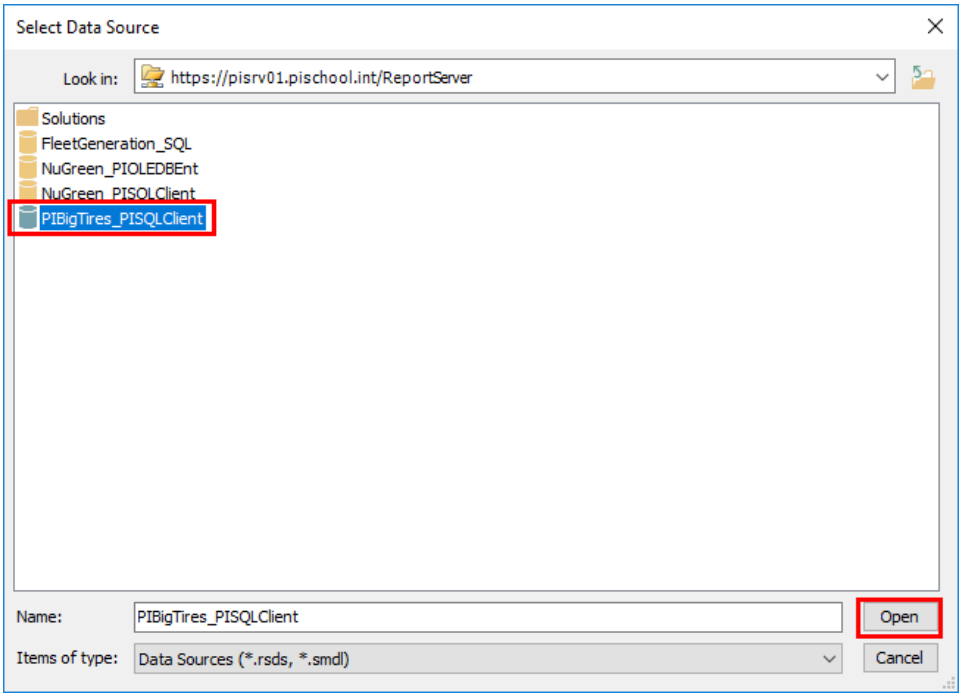
Using the same Tire Production Report from the previous exercises, add the Data Source we just created. Right-click on Data Sources -> Add Data Source…



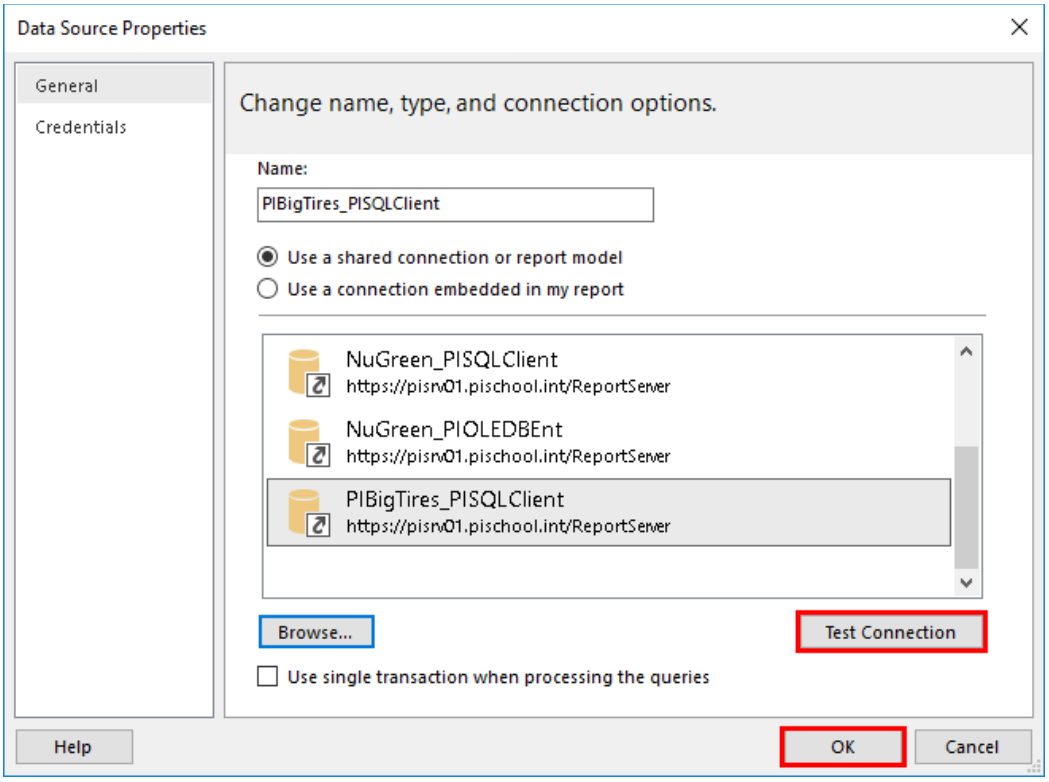
Name it PIBigTires_PISQLClient and click Browse…



Select PIBigTires_PISQLClient and click Open.

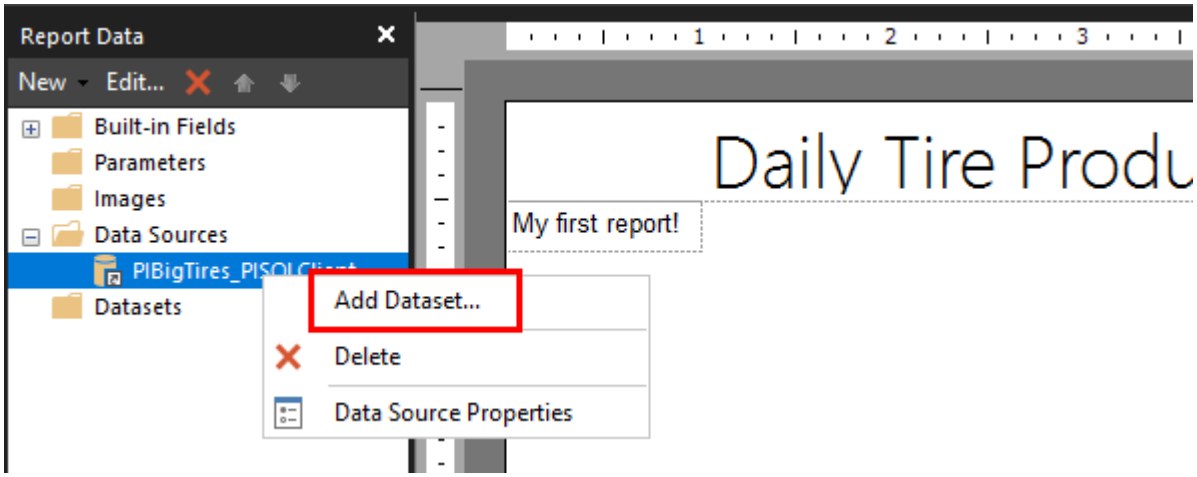


Test the connection, click OK.



Now that a Data Source is defined, we can add a dataset which contains the actual imported data.

Right-click the PIBigTires_PISQLClient Data Source and select Add Dataset…



Name it TireProduction. Leave “Use a dataset embedded in my report” checked. PIBigTires_PISQLClient should already be selected.

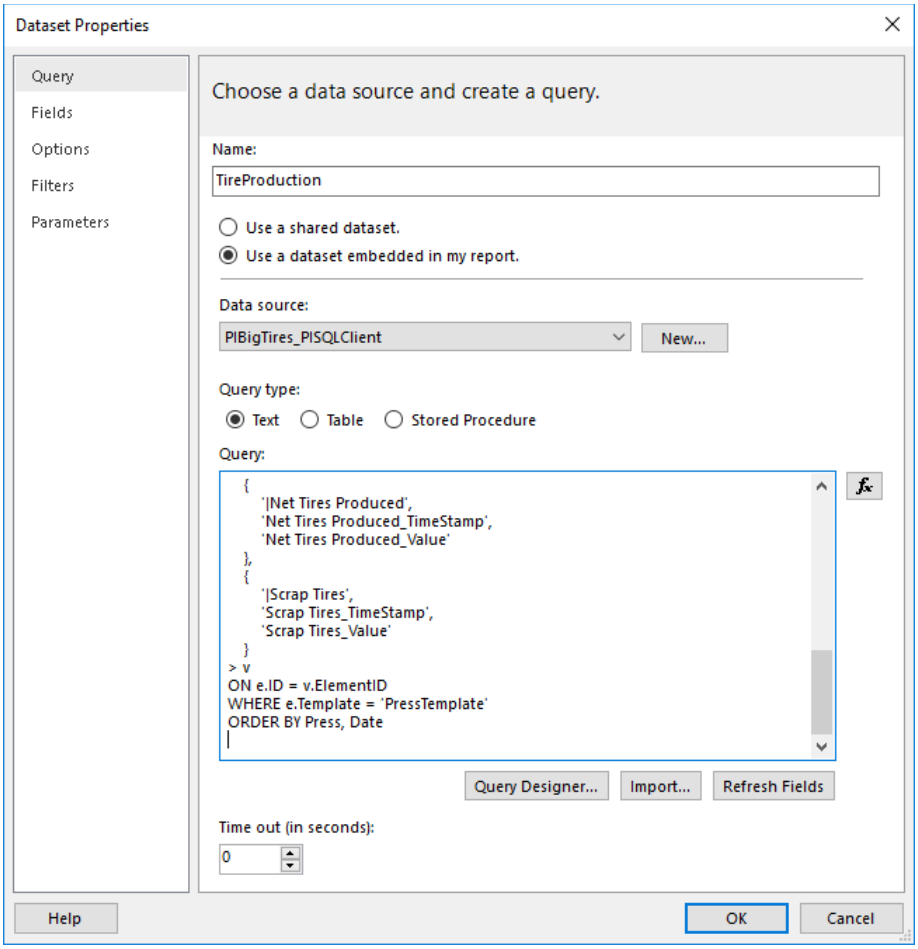
Note that there is an option to use a shared dataset, which we could configure on the report server. In our case, one could argue that the dataset is specific to the report and doesn’t have a lot of value outside the report, so sharing it may do more harm than good.

In the query box, past the Daily Production Data query, given below:

```
SELECT e.Name as Press,
sv.[Net Tires Produced_Value] as [Net Tires Produced],
sv.[Scrap Tires_Value] as [Scrap Tires],
```

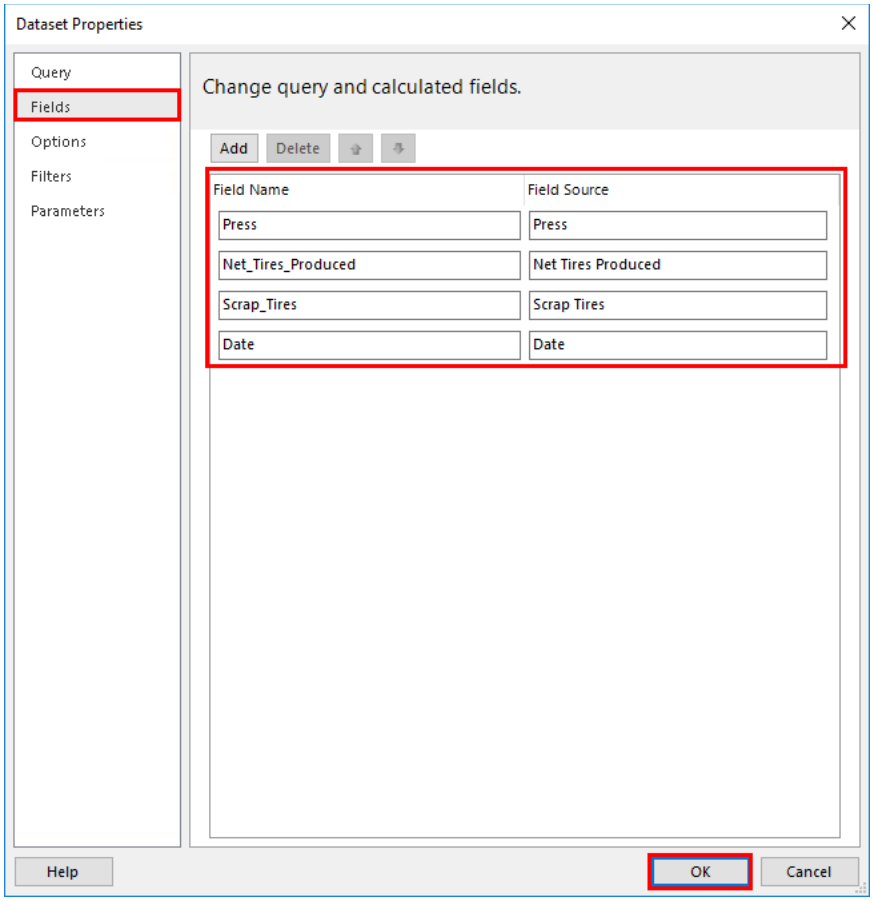
```
FORMAT(TimeStamp, 'dd-MMM-yy') as Date
FROM Master.Element.Element e
CROSS APPLY Master.Element.GetSampledValues
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_Value'
   }
>(e.ID, 'T-6d-1s', 'T-1s', '1d') sv
WHERE e.Template = 'PressTemplate'
UNION
SELECT e.Name as Press,
v.[Net Tires Produced_Value] as [Net Tires Produced],
v.[Scrap Tires_Value] as [Scrap Tires],
FORMAT(DATETIME('T'),'dd-MMM-yy') as Date
FROM Master.Element.Element e
INNER JOIN Master.Element.Value
<
   'PressTemplate',
   {
      '|Net Tires Produced',
      'Net Tires Produced_TimeStamp',
      'Net Tires Produced_Value'
   },
   {
      '|Scrap Tires',
      'Scrap Tires_TimeStamp',
      'Scrap Tires_Value'
   }
> v
ON e.ID = v.ElementID
WHERE e.Template = 'PressTemplate'
ORDER BY Press, Date
```

The Dataset properties should now look like this:

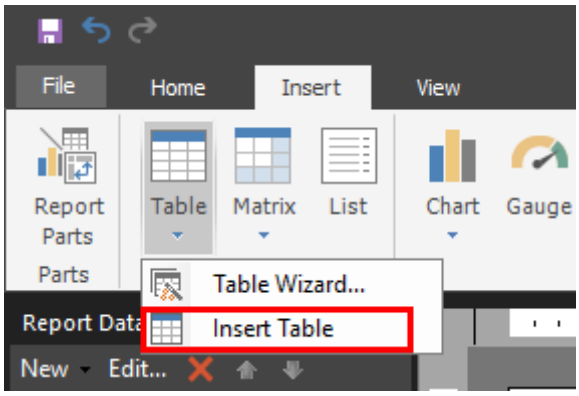


You can quickly check that the query is valid by examining the fields list. If the query returned results then the fields list will show the column headers from the result set.

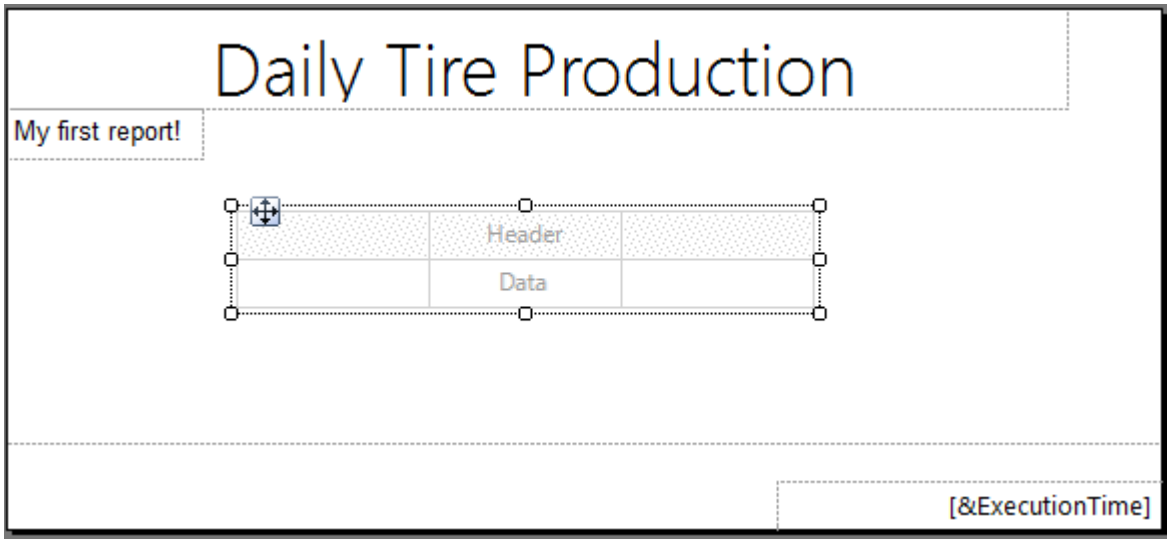
Click Fields, then OK once you've confirmed that the fields show up.



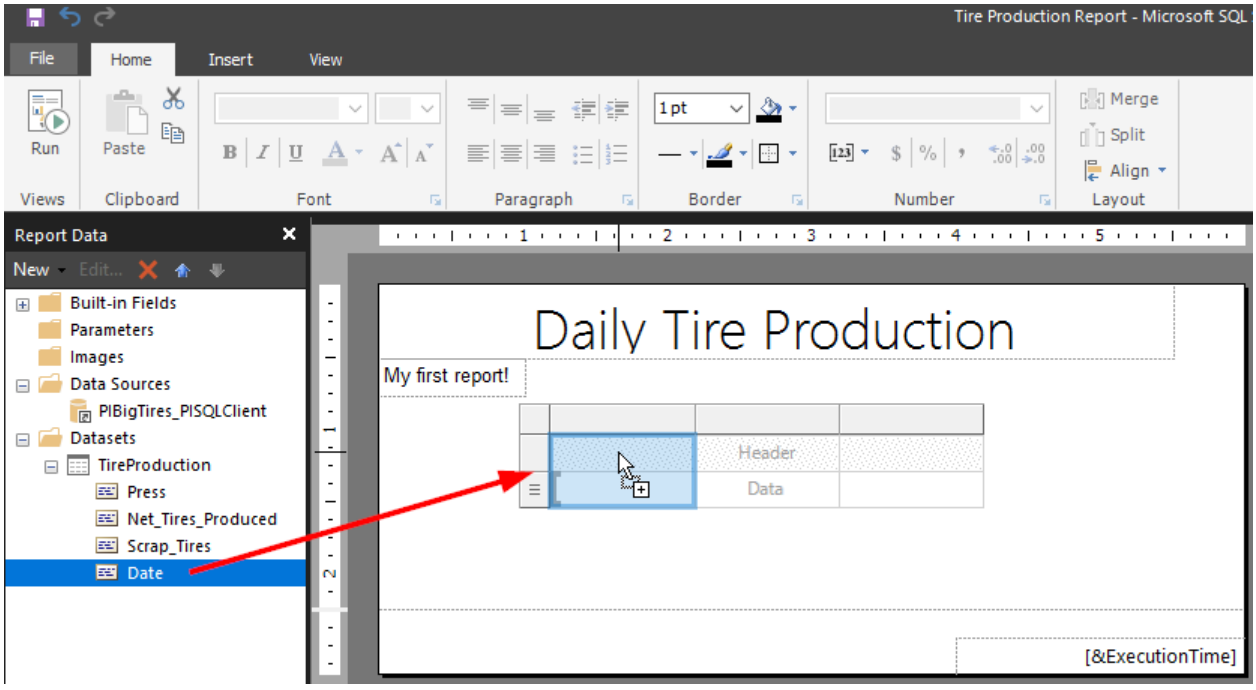
Next let's add a table. From the Insert tab, select Table -> Insert Table



Click some white space where you want the table to show and the table object will appear:



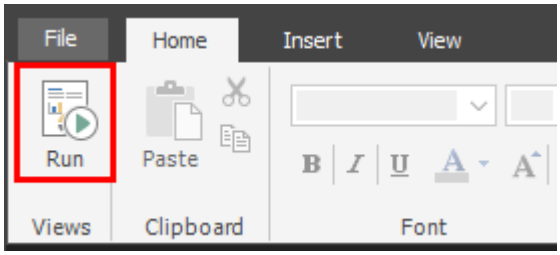
Now simply drag and drop fields from the TireProduction dataset to add them to the table. **Add the Date, Press, Net_Tires_Produced, and Scrap_Tires fields to the table.**



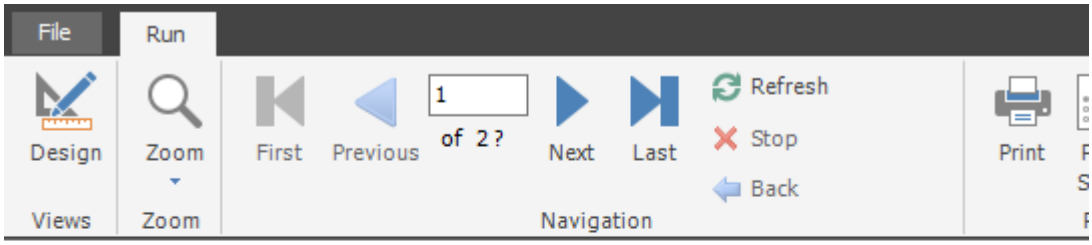
When you're done the table should look like this:

Date	Press	Net Tires	Scrap Tires
[Date]	[Press]	[Net_Tires_Proc	[Scrap_Tires]

Run the report as a sanity check.



The table should display query results.



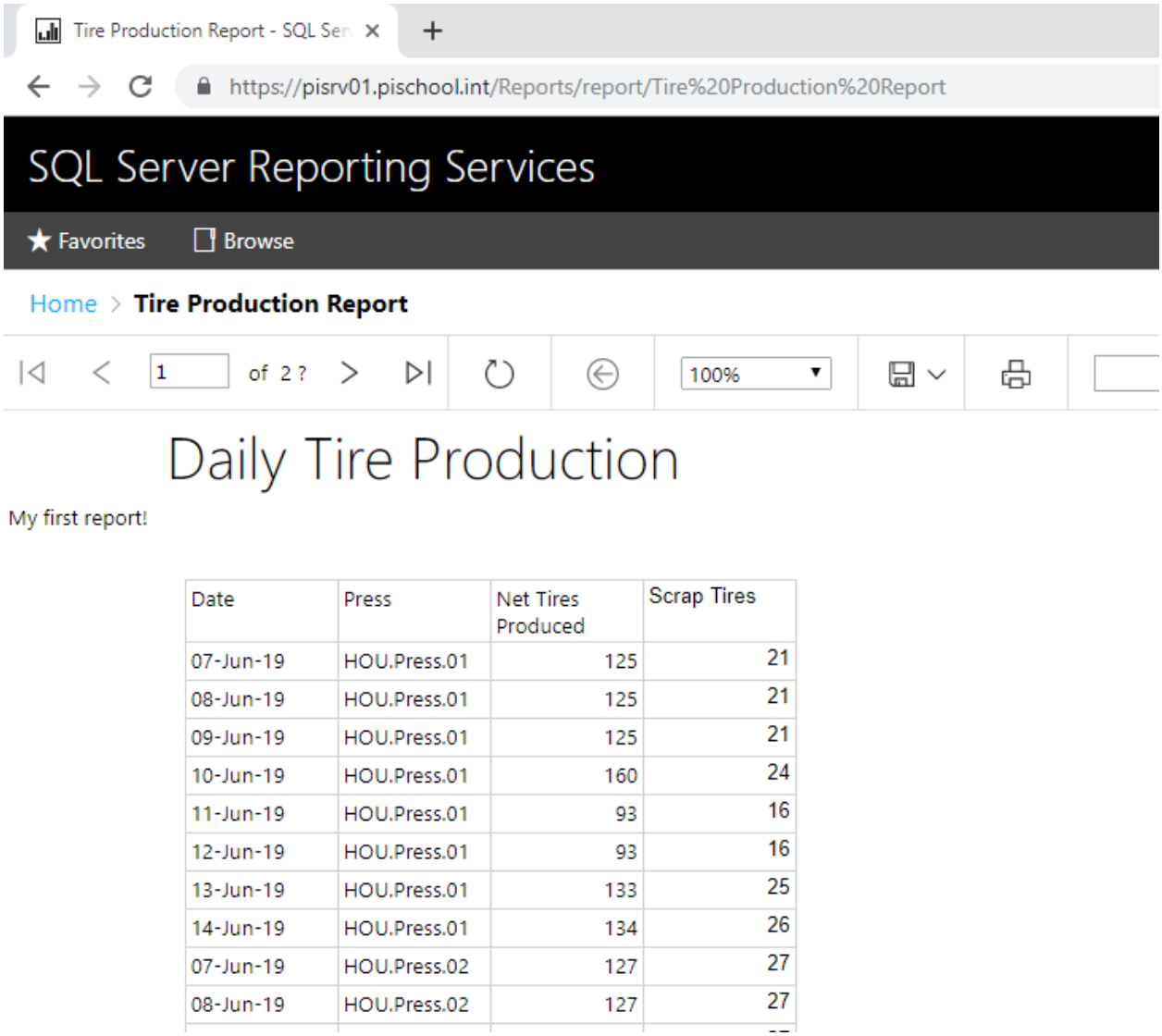
Daily Tire Production

My first report!

Date	Press	Net Tires Produced	Scrap Tires
06-Jun-19	HOU.Press.01	121	19
07-Jun-19	HOU.Press.01	125	21
08-Jun-19	HOU.Press.01	125	21
09-Jun-19	HOU.Press.01	125	21
10-Jun-19	HOU.Press.01	160	24
11-Jun-19	HOU.Press.01	93	16
12-Jun-19	HOU.Press.01	93	16
13-Jun-19	HOU.Press.01	132	24
06-Jun-19	HOU.Press.02	122	25
07-Jun-19	HOU.Press.02	127	27
08-Jun-19	HOU.Press.02	127	27

Another sanity check, **Save the report** and go back to the report server https://pisrv01.pischool.int/Reports.

Open your freshly saved report and confirm that the data displays correctly.

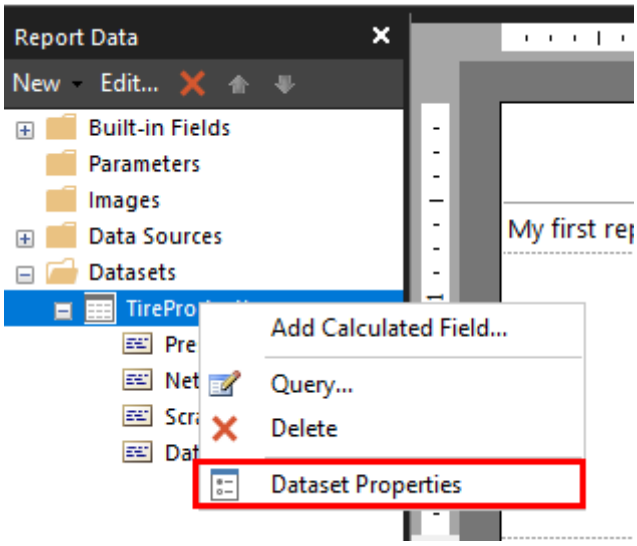


Date	Press	Net Tires Produced	Scrap Tires
07-Jun-19	HOU.Press.01	125	21
08-Jun-19	HOU.Press.01	125	21
09-Jun-19	HOU.Press.01	125	21
10-Jun-19	HOU.Press.01	160	24
11-Jun-19	HOU.Press.01	93	16
12-Jun-19	HOU.Press.01	93	16
13-Jun-19	HOU.Press.01	133	25
14-Jun-19	HOU.Press.01	134	26
07-Jun-19	HOU.Press.02	127	27
08-Jun-19	HOU.Press.02	127	27

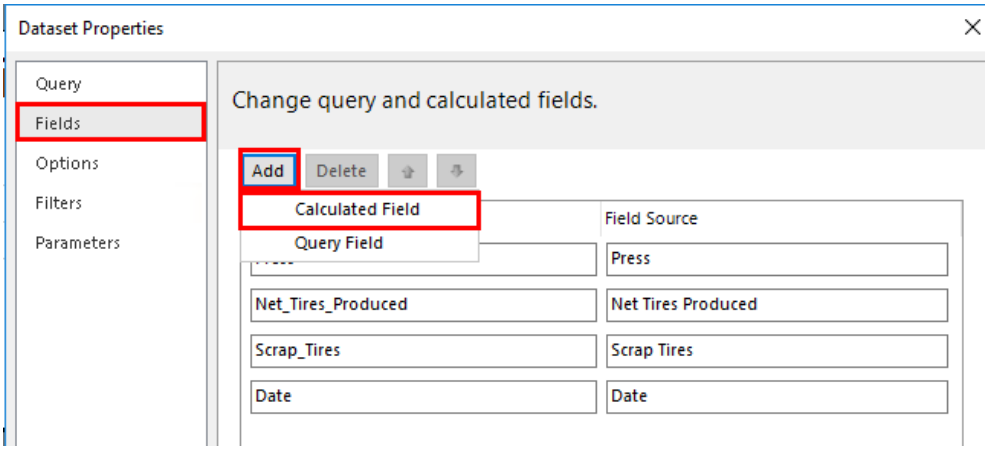
Depending on when the report is viewed, you may see a problem with the report. **The dates are actually casted as strings and listed in alphabetical order**. If the result set straddles a boundary between months, then they won't display in chronological order. Let's fix this.

We're going to add another column to the dataset to convert the Date strings to the date data type. There are of course other ways to do this.

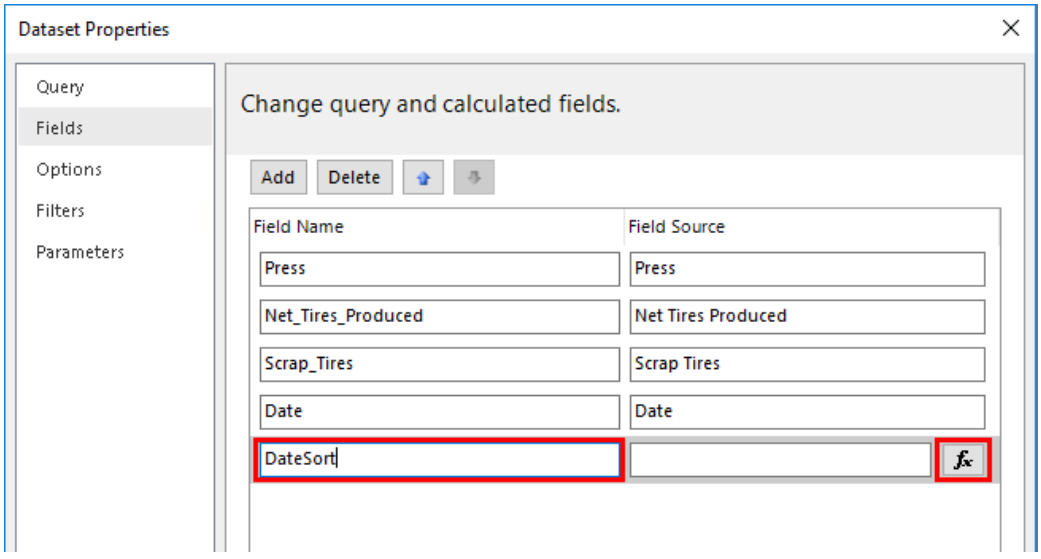
Right-click on the TireProduction dataset and select Dataset Properties.



In the fields tab, add a **Calculated Field**.

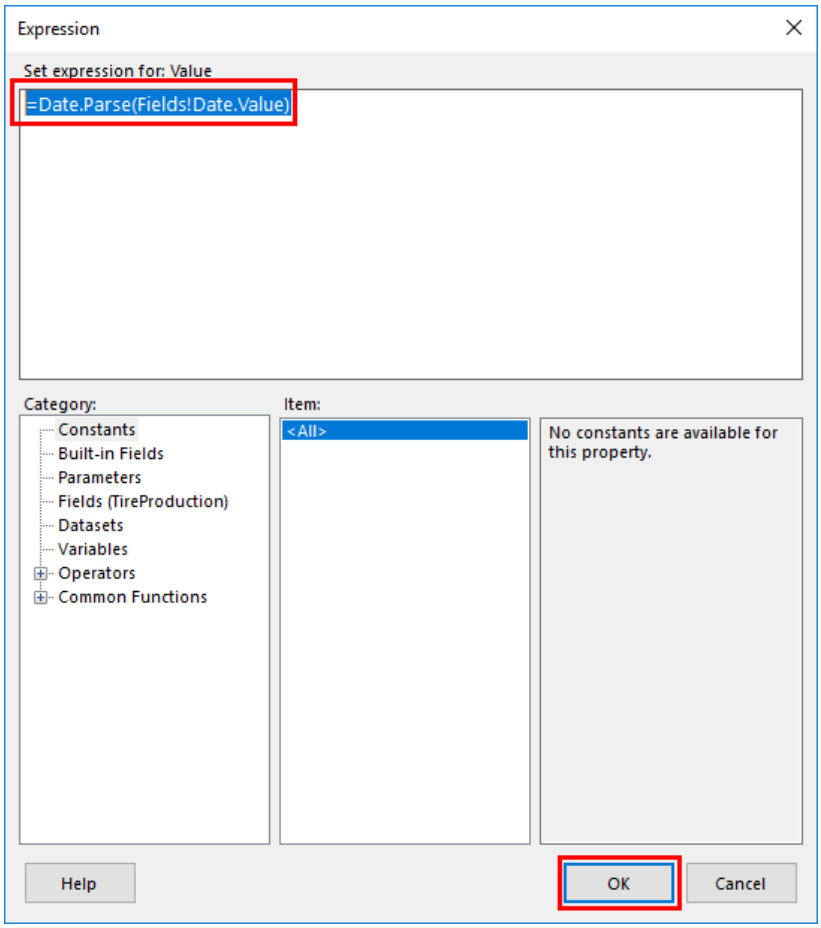


Name it DateSort and configure it as an expression.



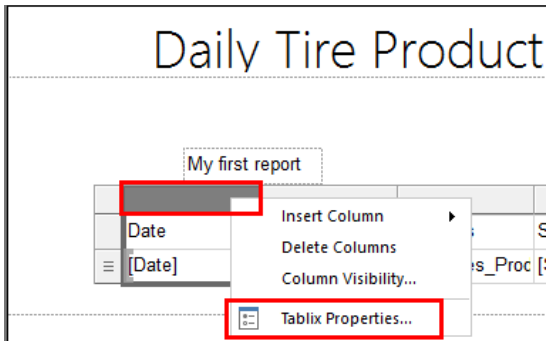
Paste in the below expression and click OK. You'd find the expression by Googling "SSRS Report Builder convert string to date" or something like that.

=Date.Parse(Fields!Date.Value)

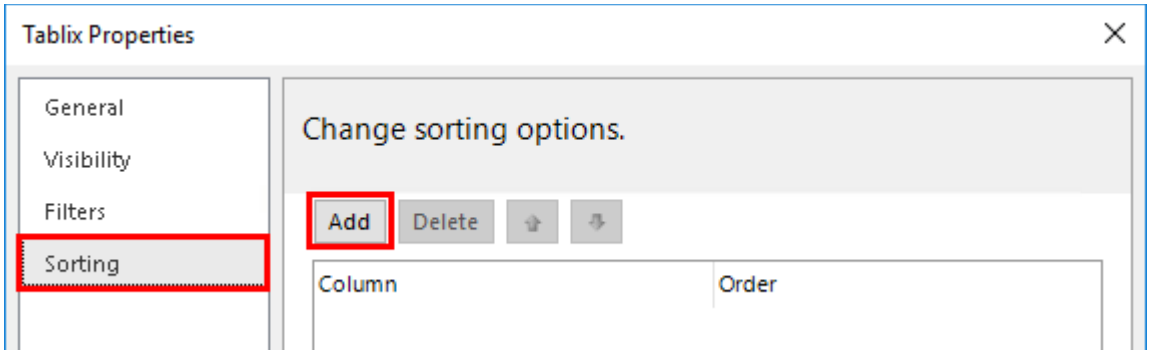


Click OK again to close the window.

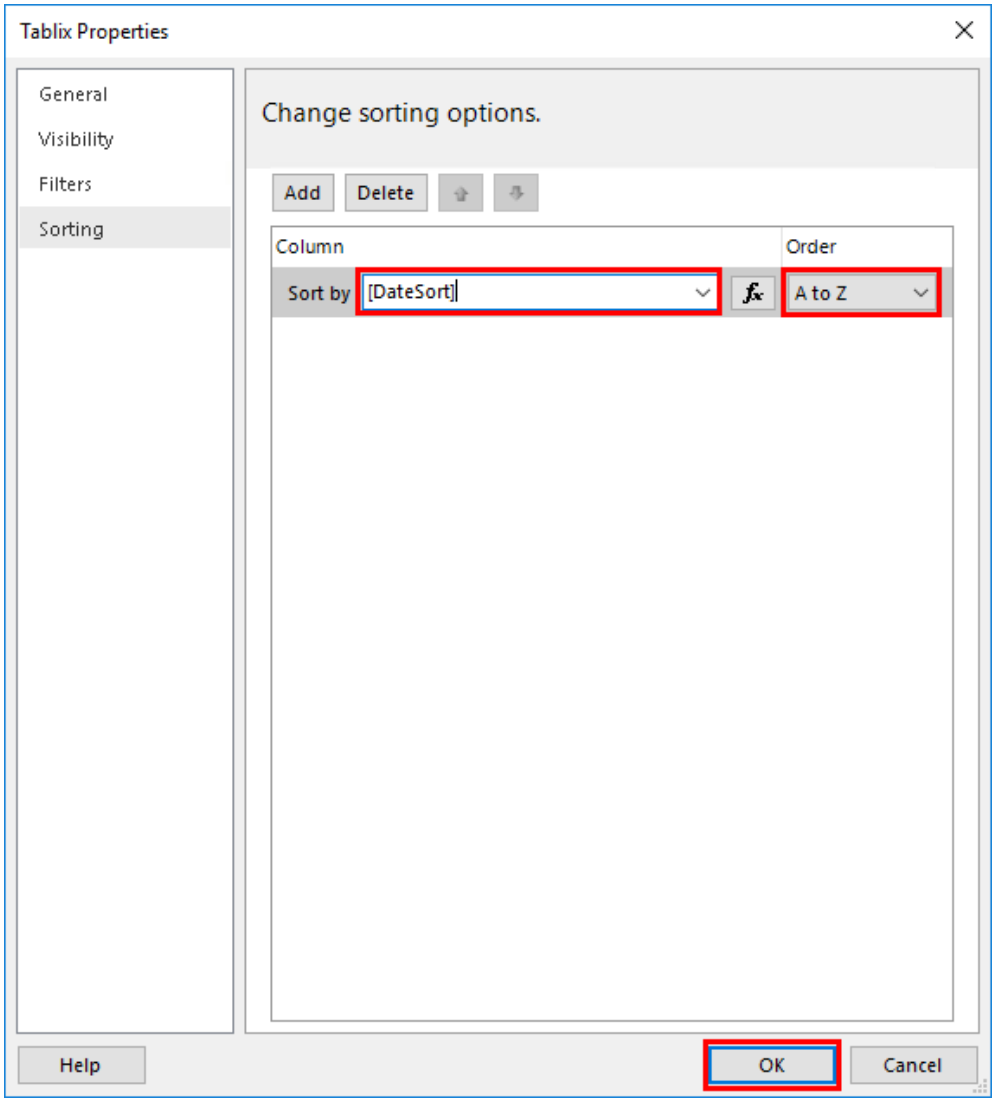
In the table, select the Date Column, right-click and select Tablix Properties.



In the sorting tab, add a sorting rule.



Sort by DateSort in A to Z order, click OK.



Optionally Run the report and confirm the sort order. You won't see any difference if all the dates are in the same month.

Directed Activity – Add Interactive Parameters for Filtering

In this part of the class you will perform a learning activity to explore the different concepts presented in this chapter or section. You may be invited to watch what the instructor is doing or perform the same steps at the same time. You may play a game or hold a quiz. Your instructor will have directions.

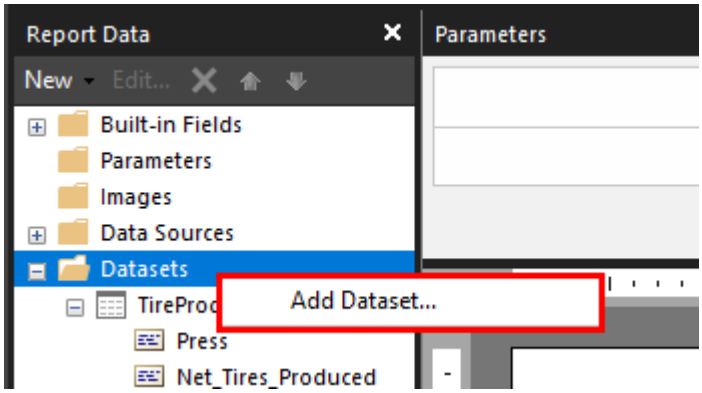
Objective: Add user selections for Site and Press in order to filter the table

Approach:

We'll add some parameters to filter the report. We want to be able to filter by Site and by Press.

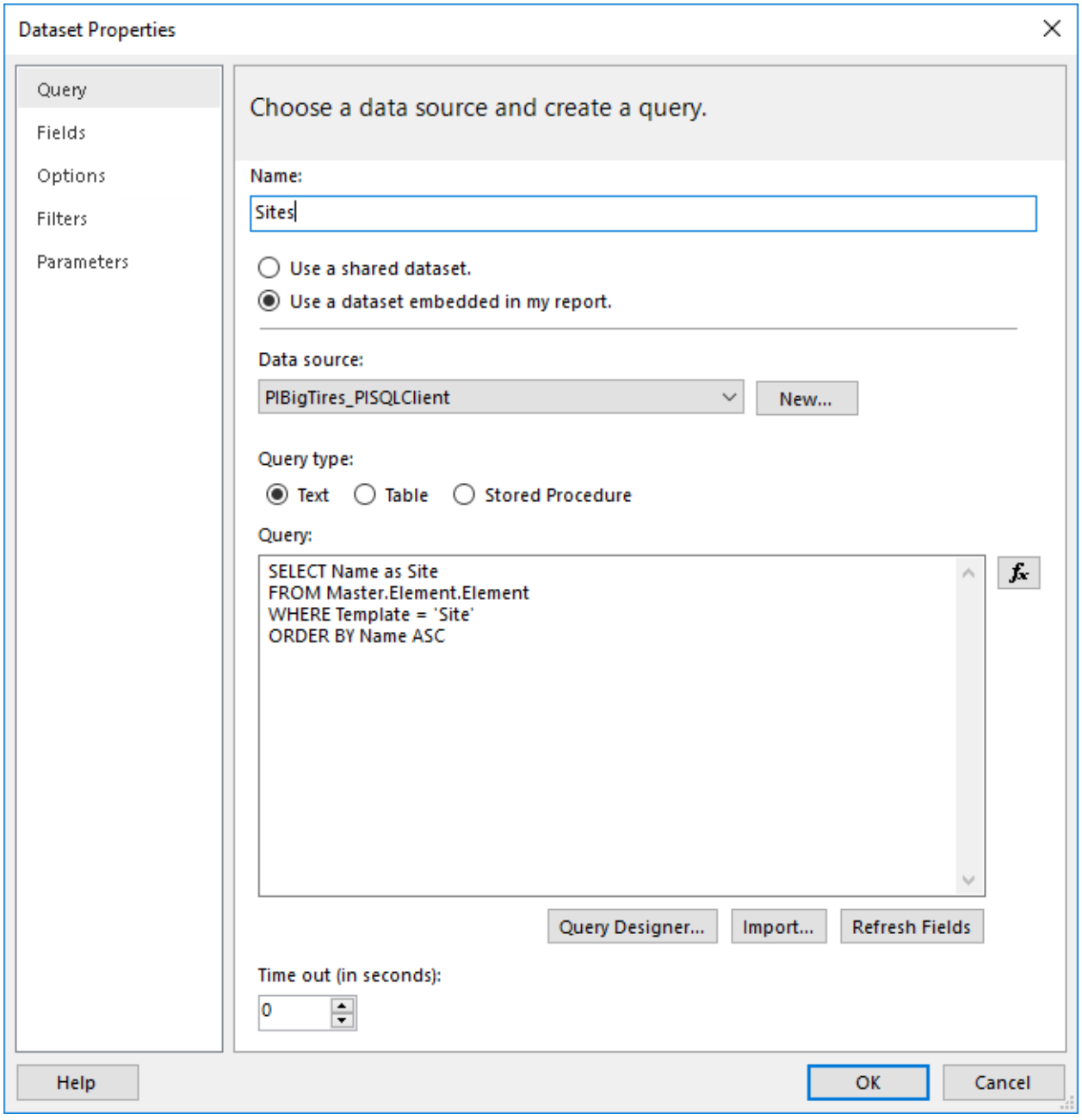
The first step is to add datasets for the parameter selections.

Create a new dataset.

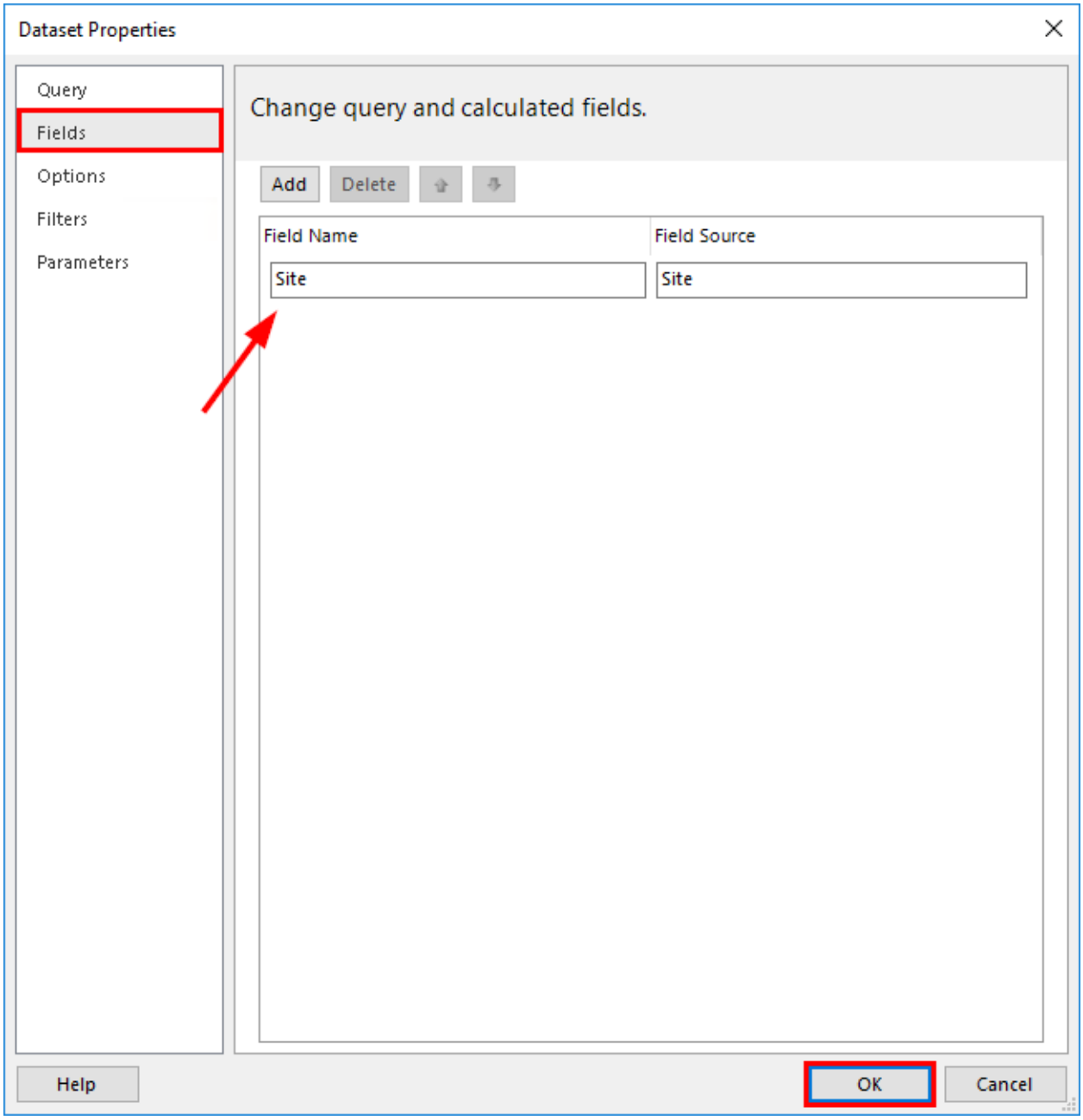


Name it Sites, select “Use a dataset embedded in my report”. Use PIBigTires_PISQLClient as the Data Source. Use the below query.

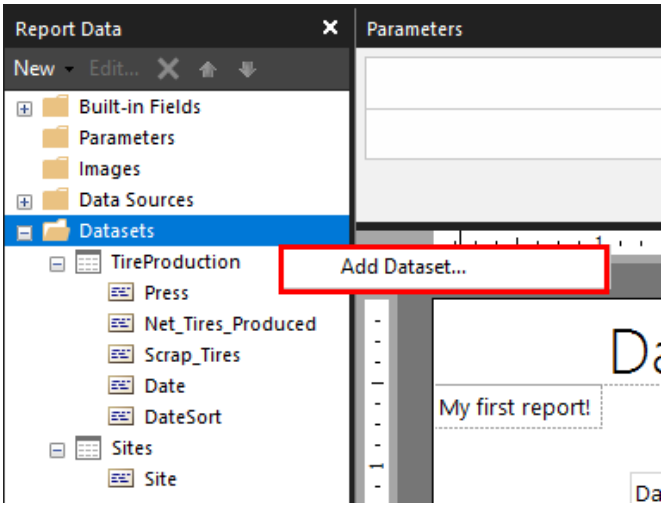
```
SELECT Name as Site
FROM Master.Element.Element
WHERE Template = 'Site'
ORDER BY Name ASC
```



Check the Fields tab to confirm the query is valid. There should be one field called Site. Click OK.

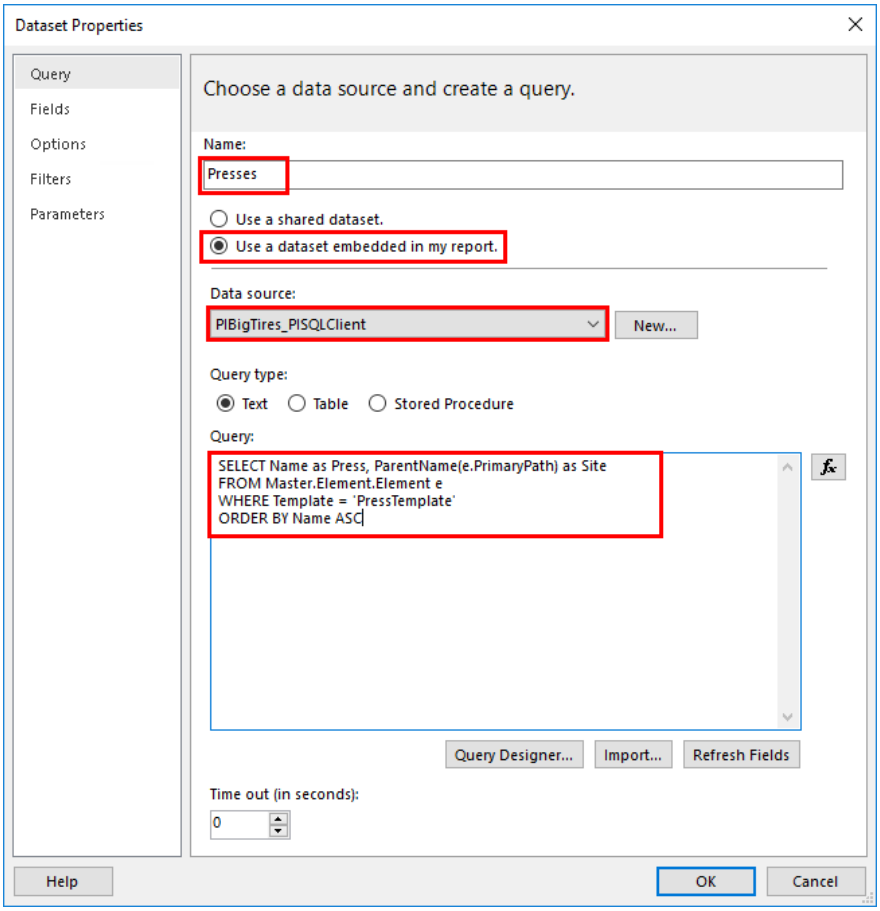


Create another dataset for the list of Presses.



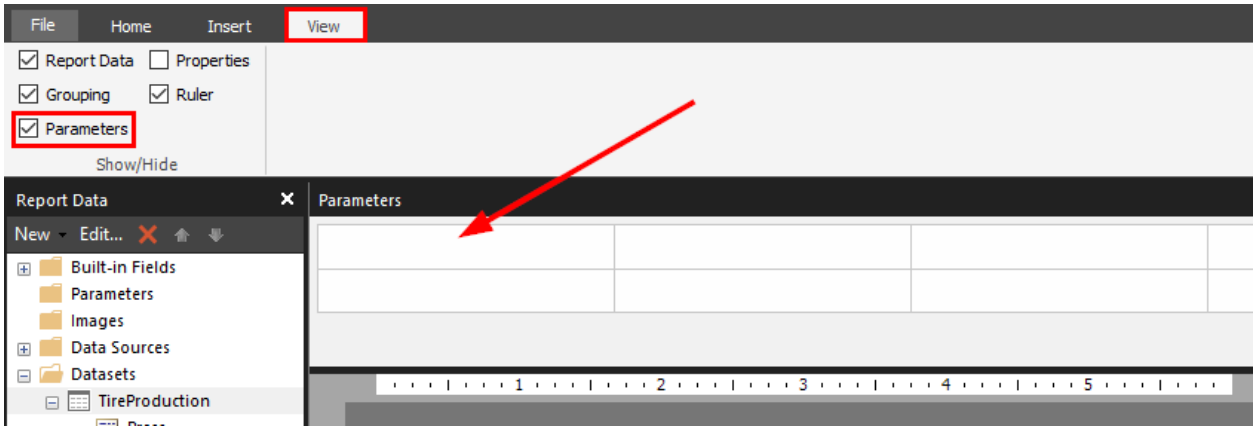
Configure it as follows using the following query.

```
SELECT Name as Press, ParentName(e.PrimaryPath) as Site
FROM Master.Element.Element e
WHERE Template = 'PressTemplate'
ORDER BY Name ASC
```

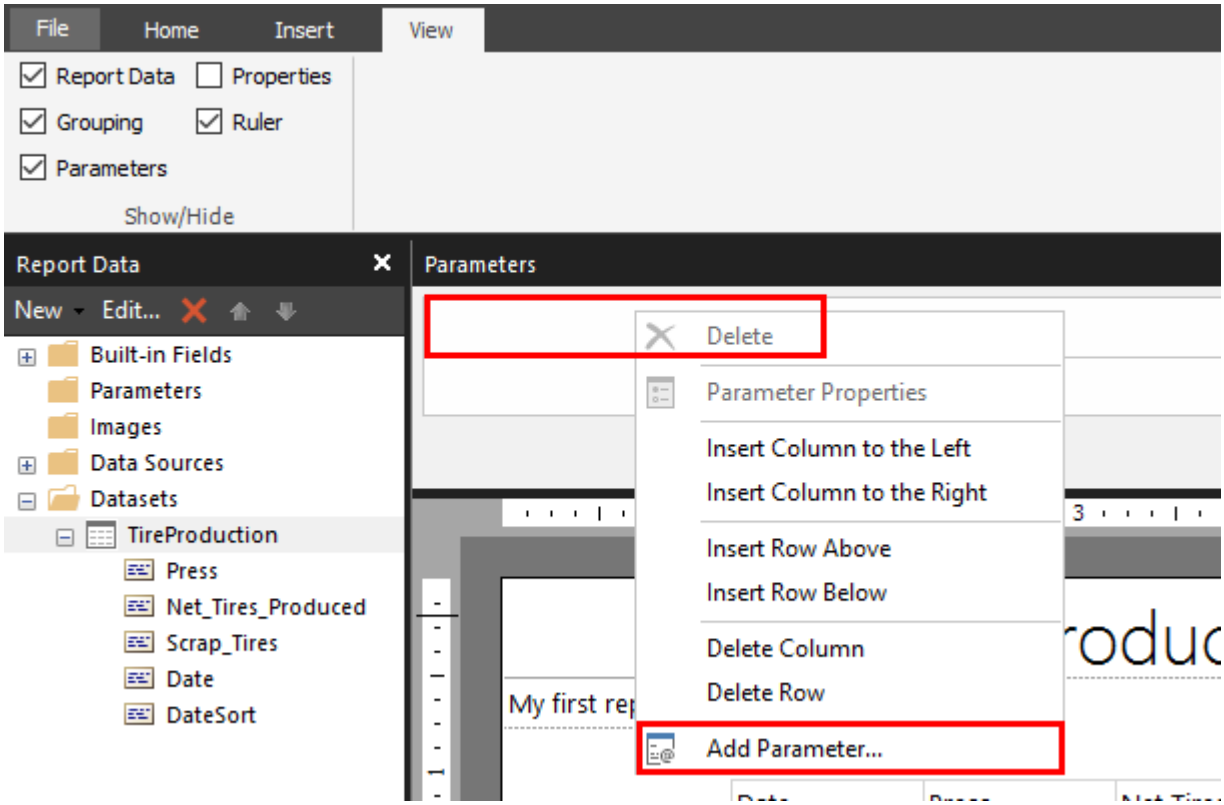


Check the Fields tab to confirm the query is valid. Click OK.

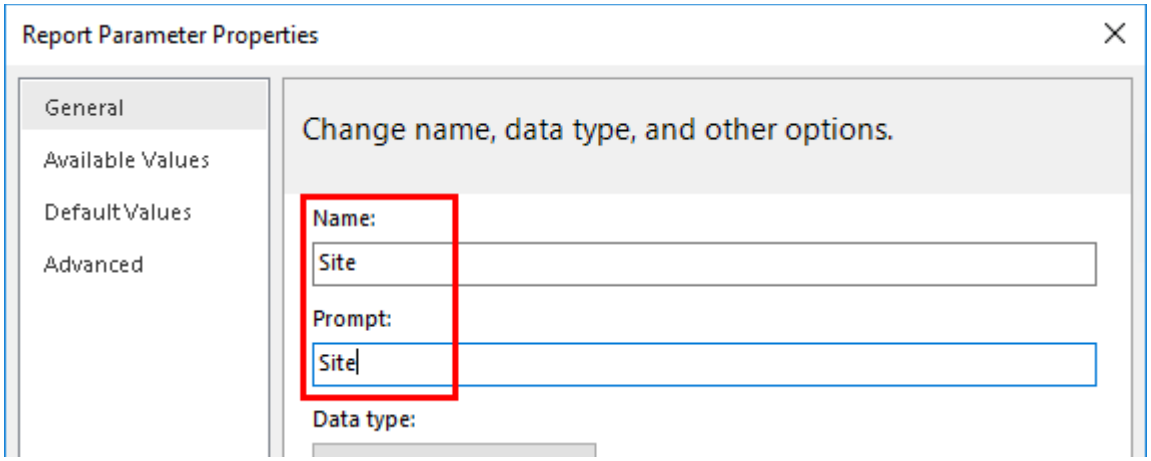
On the View tab, check the box for Parameters and the parameters section will be added to the report.



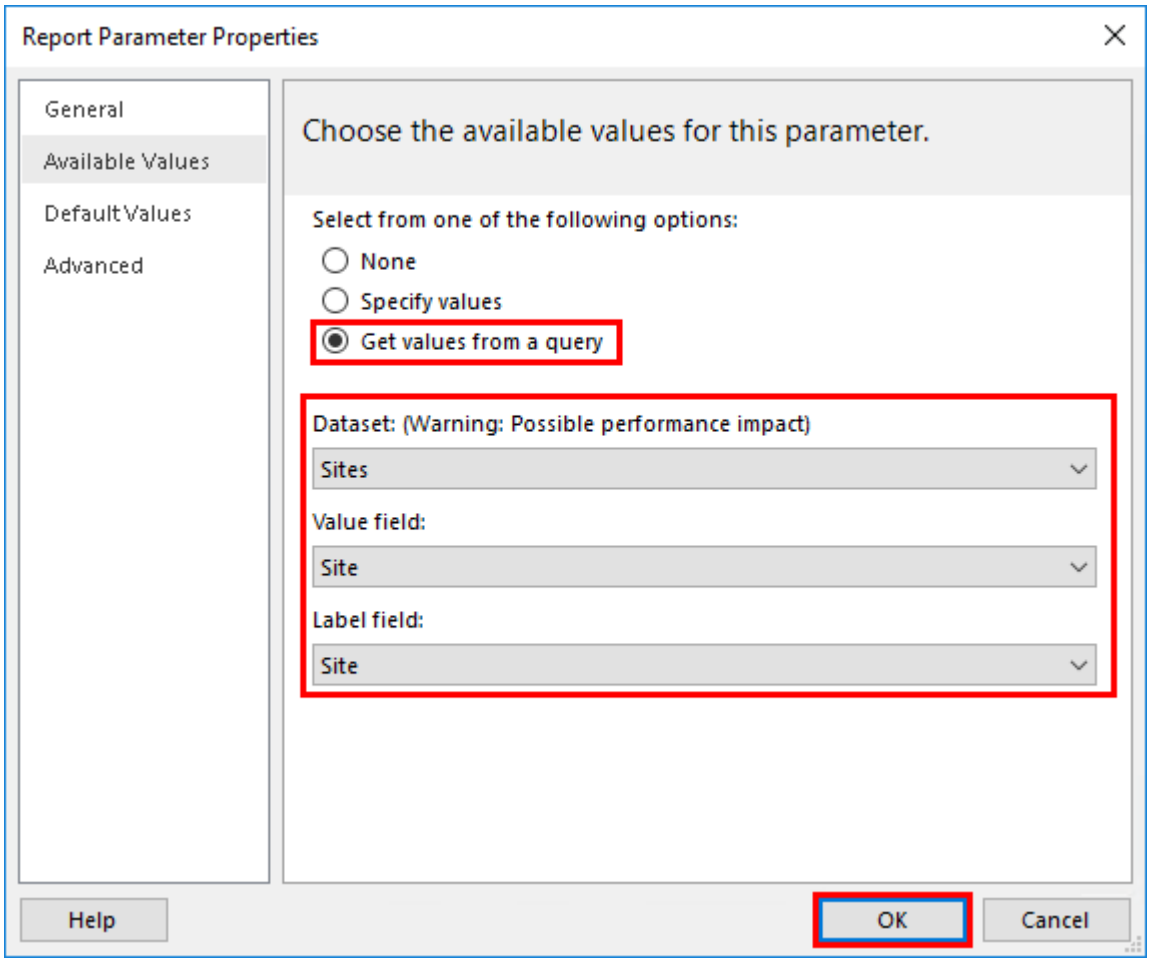
Right-click on the parameters table and select Add Parameter.



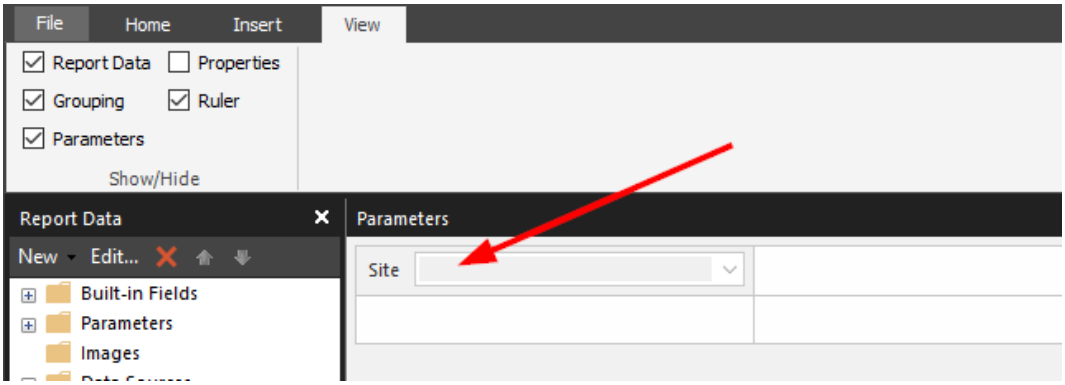
Enter Site for the Name and for the Prompt.



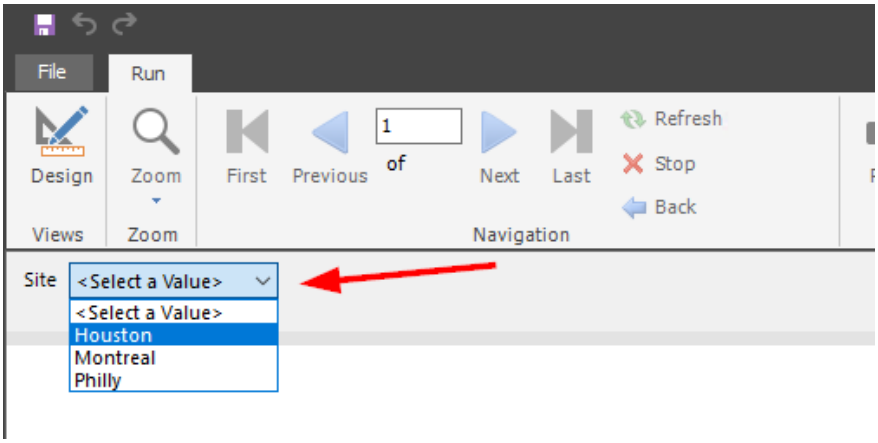
On the Available Values tab, Get values from a query using Dataset Sites. Use the Site column for both the Value field and the Label field. Click OK.



There should now be a site parameter.



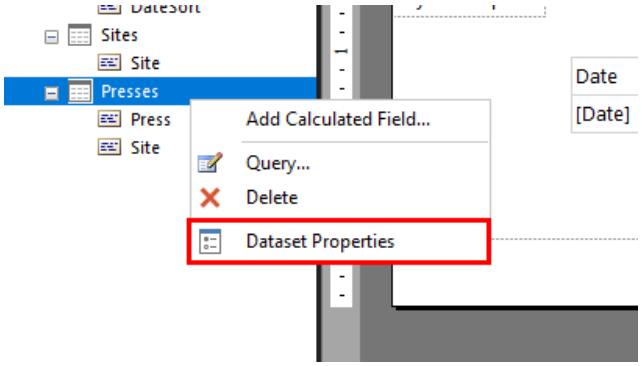
Run the report from the Home tab. When the report is run the user will be able to select a Site. **The selection doesn't filter the report yet. This is just a sanity check.**



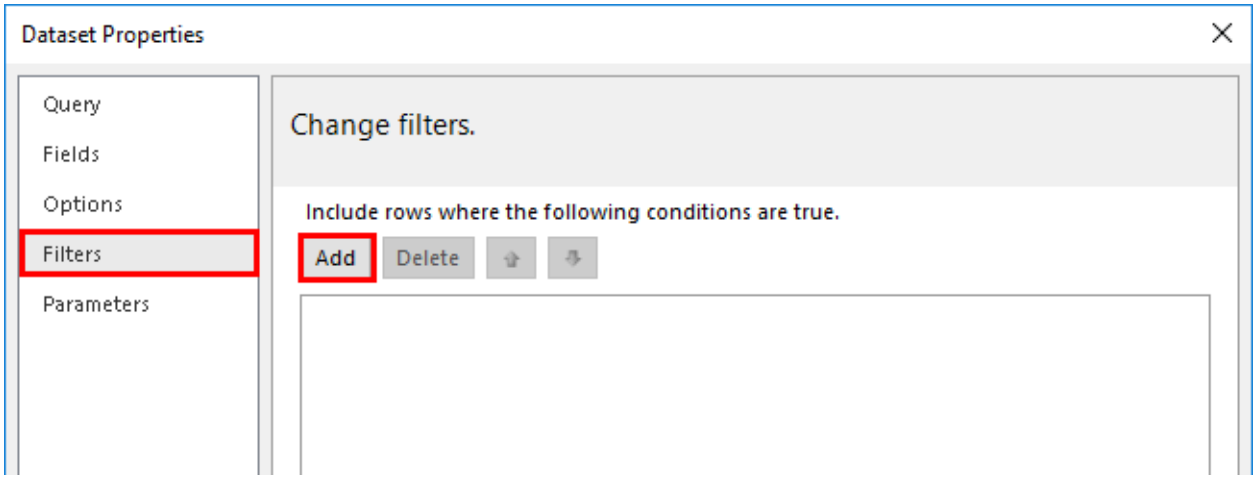
Go back to Design Mode.

Next we'll filter the list of Presses based on the Site selection. Once that's done we'll add Press as a parameter.

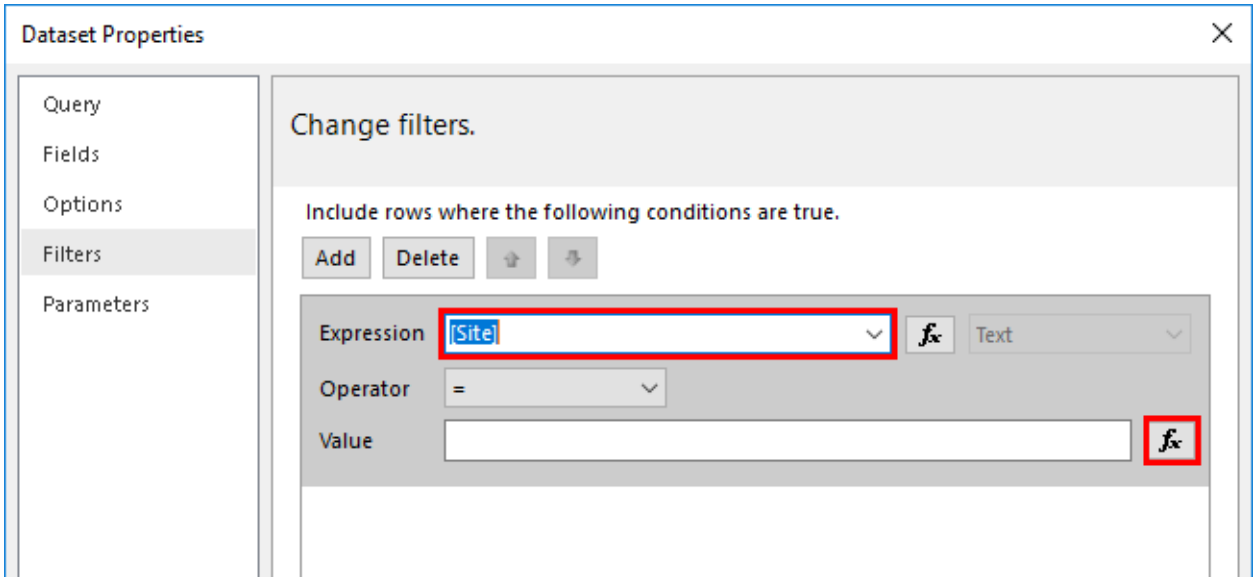
Right-click on the Presses Dataset and enter the properties.



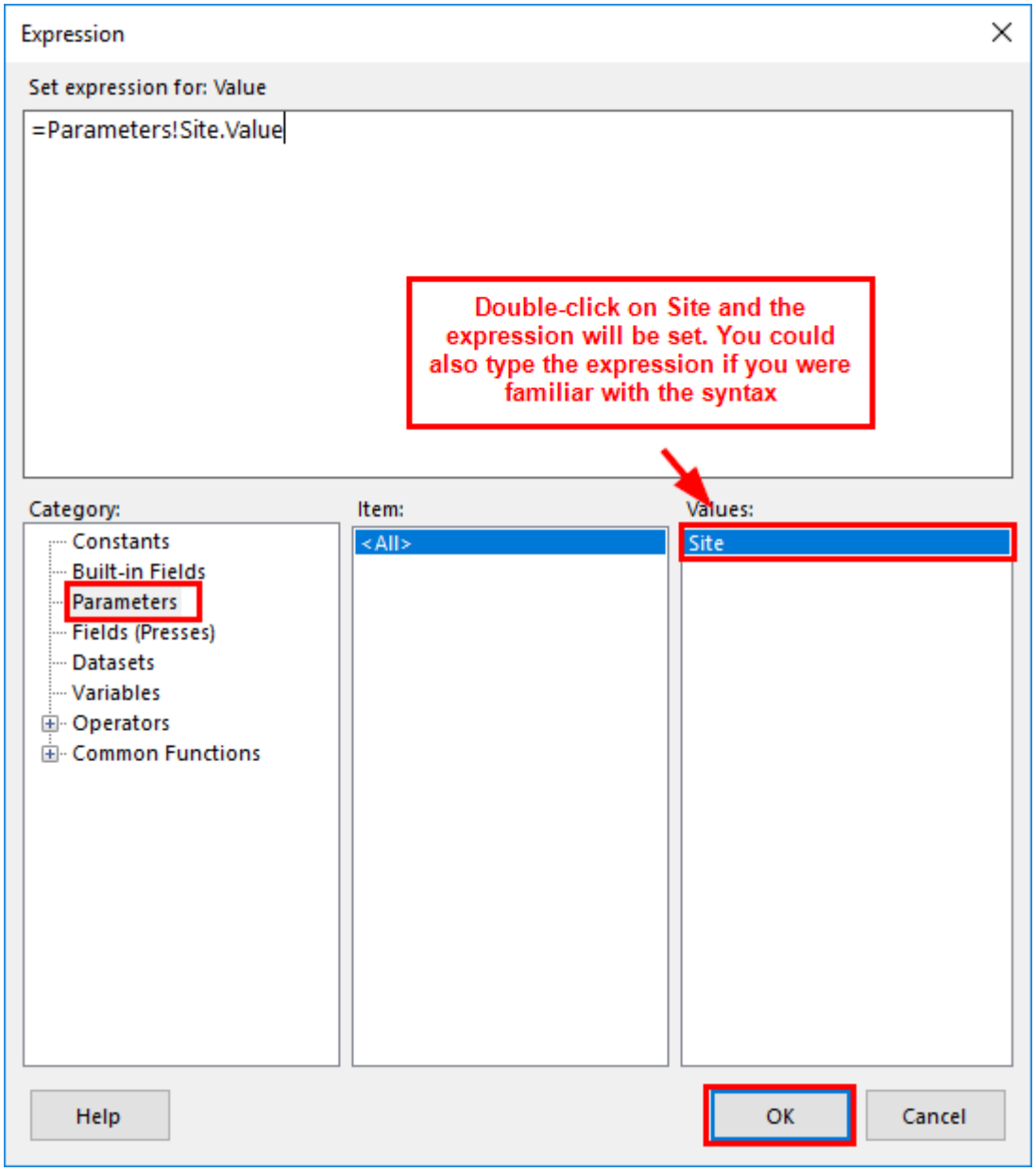
Go to the Filters tab and Add a filter.



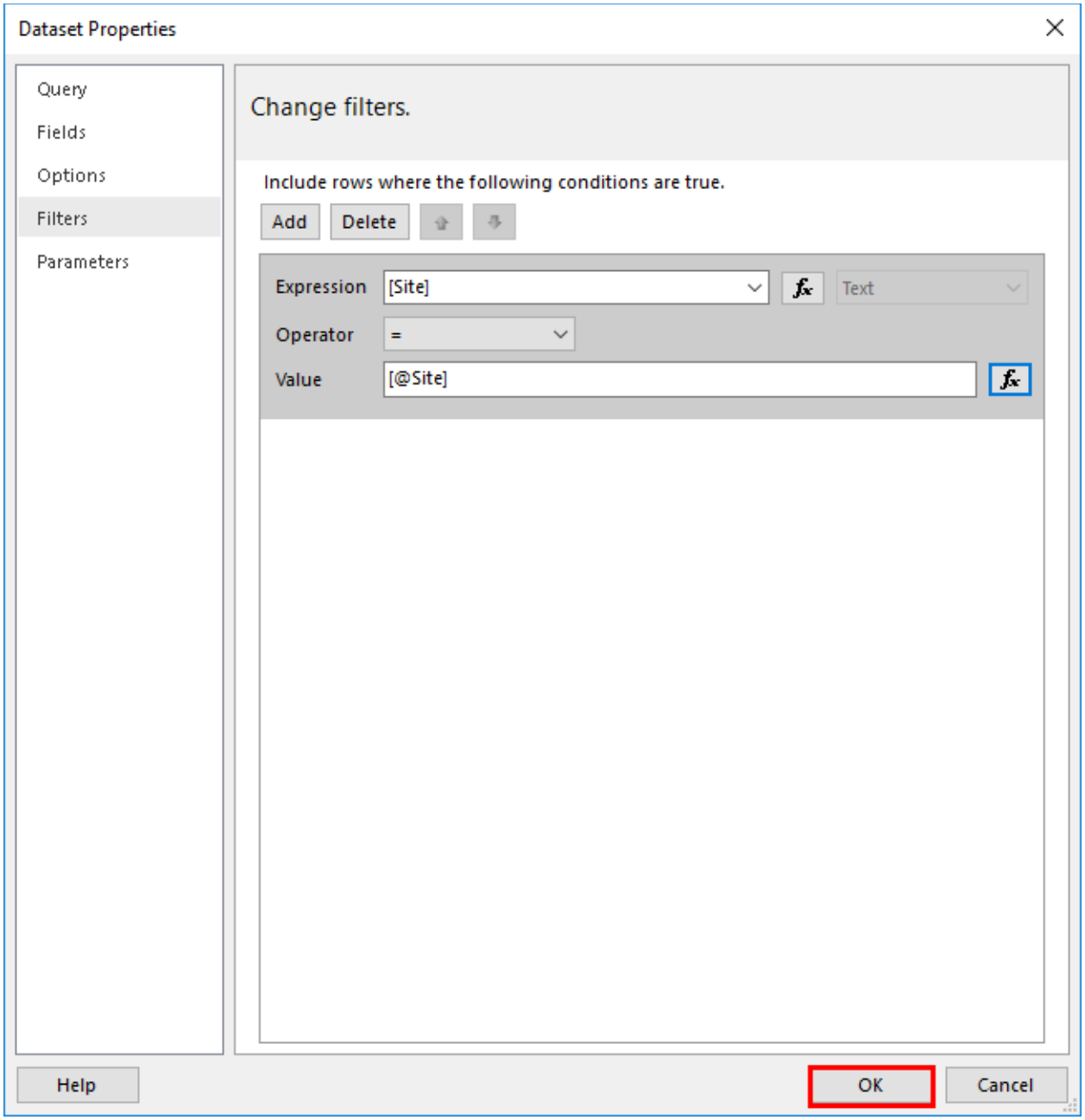
Select Site from the drop down, then enter an expression.



Select the Parameters category, then double-click on Site to set the expression. Click OK.

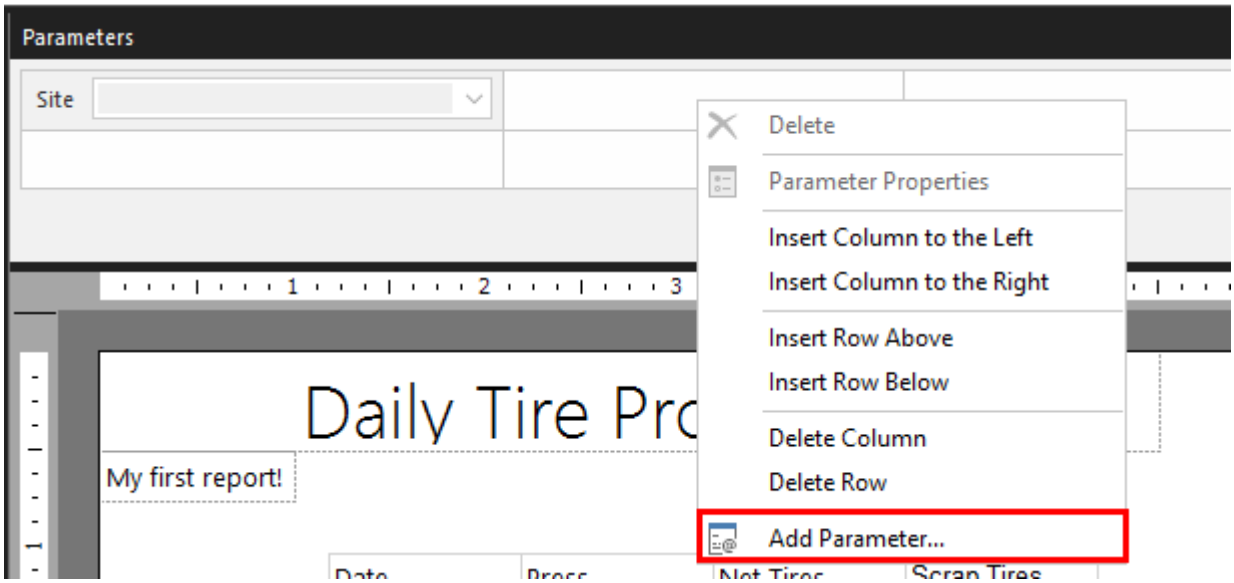


The filter configuration should look like the following screenshot. Click OK.

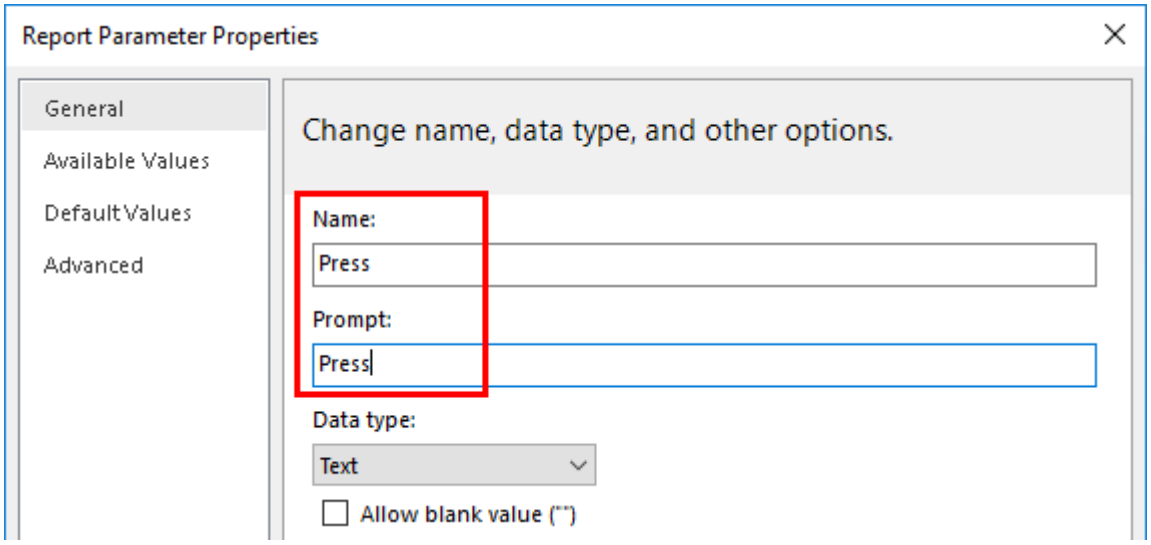


Keep in mind that the Site selection does nothing at this point except filter the Presses Dataset. This does not yet interact with the Tire Production table. We still have to add a Press parameter and filter the TireProduction Dataset based on the selection.

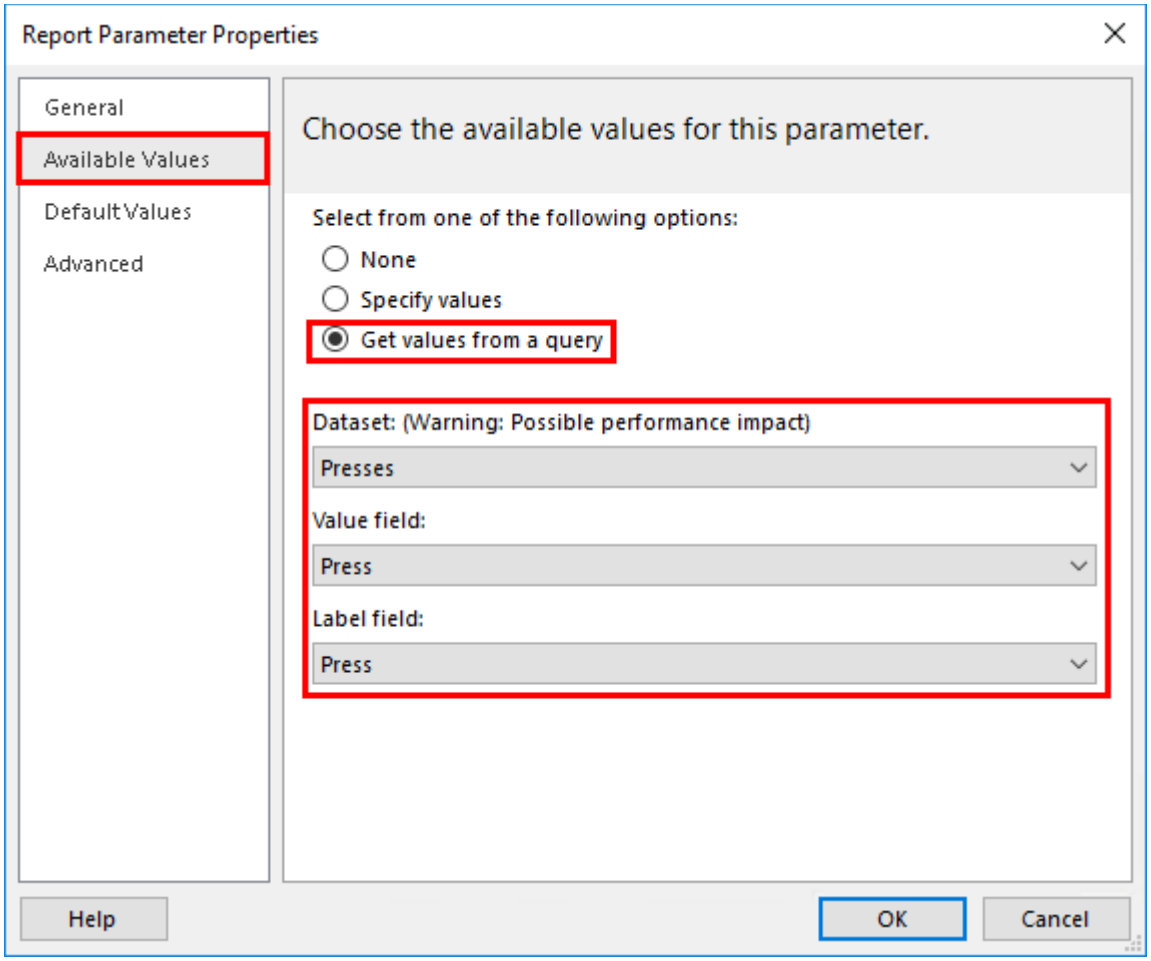
Right-click on the parameters table and add another parameter.



Use Press for the Name and Prompt.



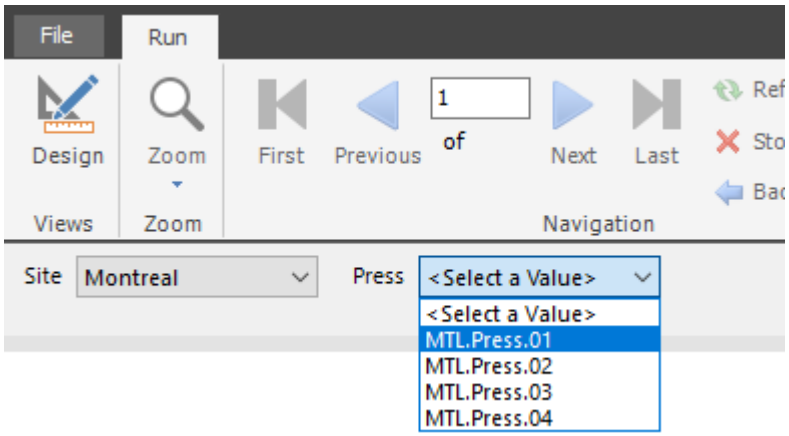
In the Available Values tab, Get the press list from the filtered query. Click OK.



Now there are parameters for Site and Press.

Run the Report as a sanity check.

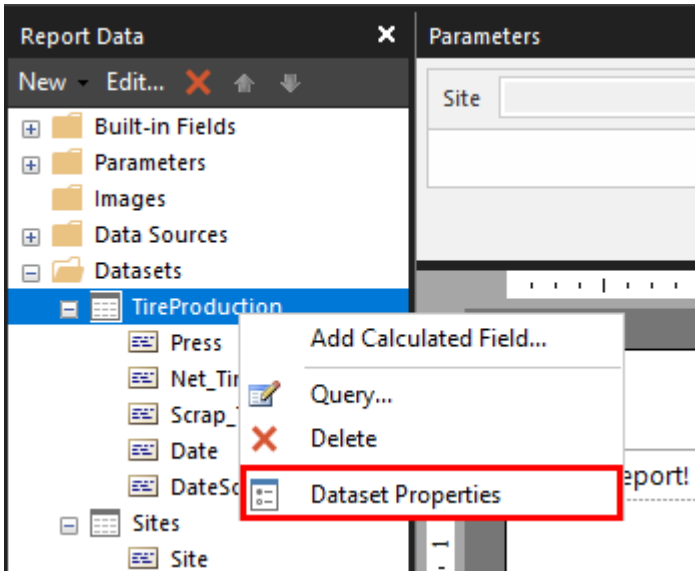
Now the Site selection actually does something. It filters the Press selection.



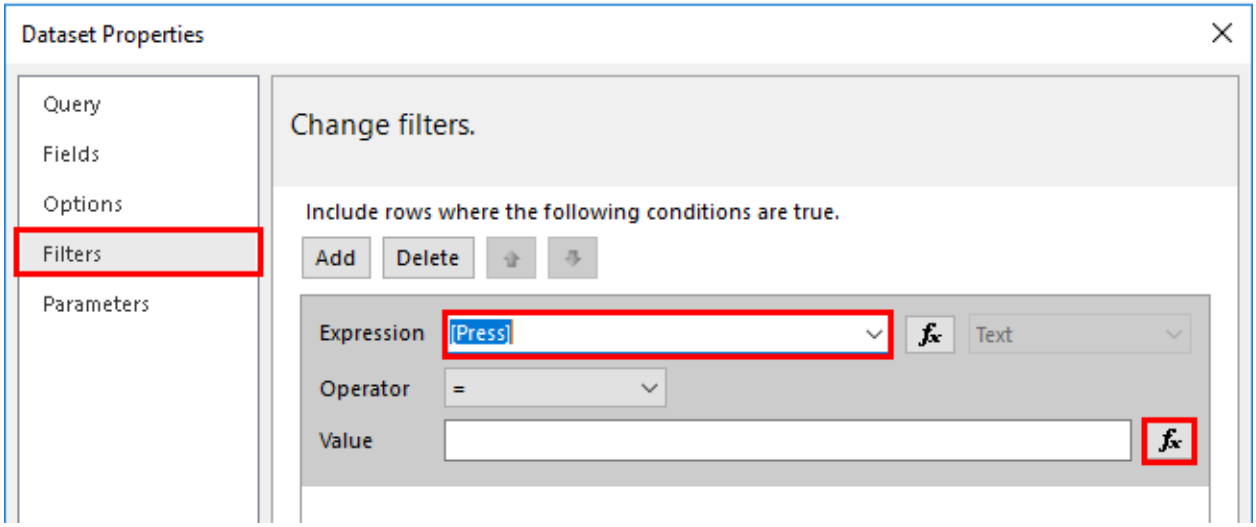
The Press selection still has no impact on the Tire Production table. We still need to filter the TireProduction dataset based on the Press parameter.

Go back to Design Mode.

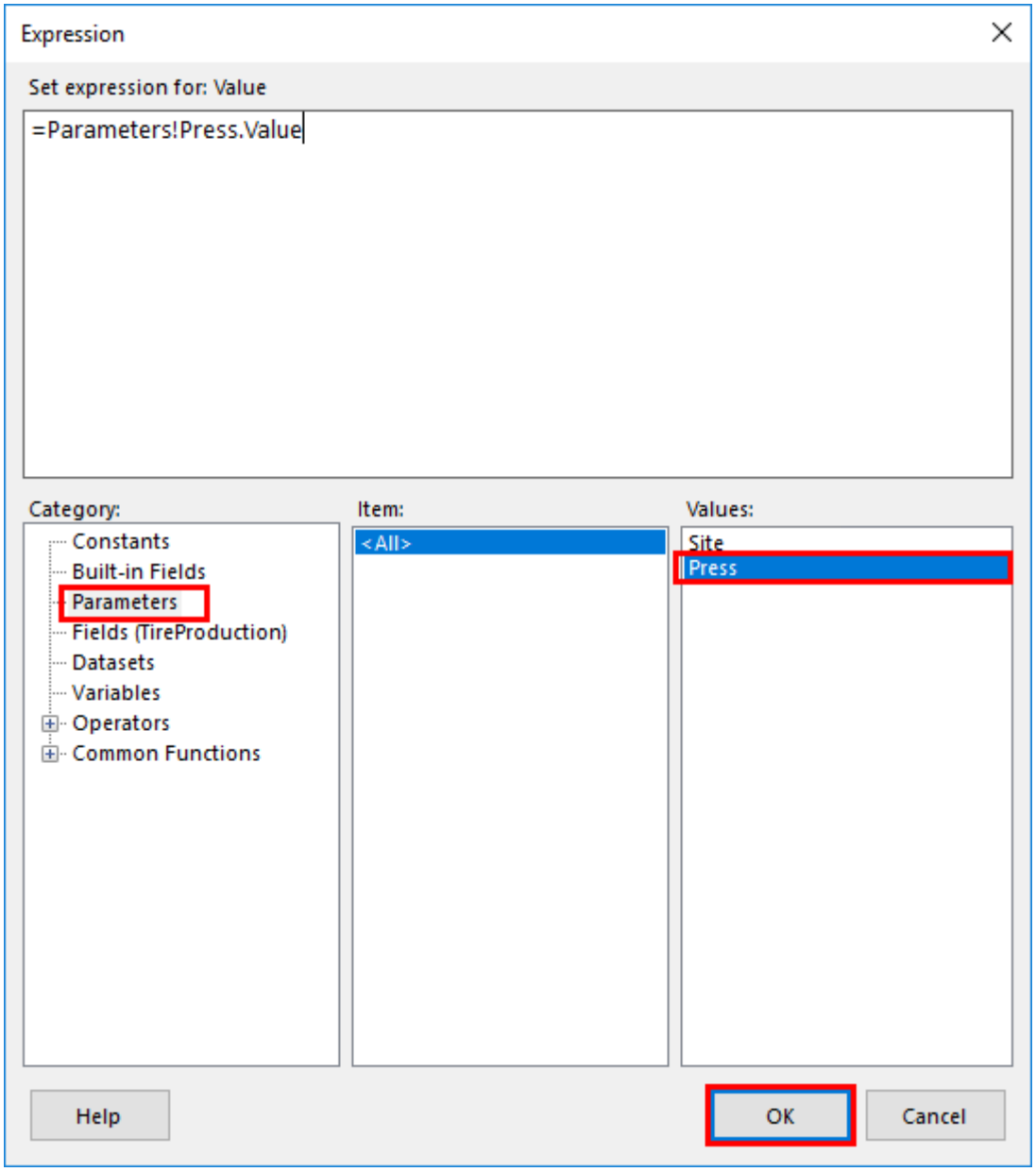
Modify the TireProduction properties.



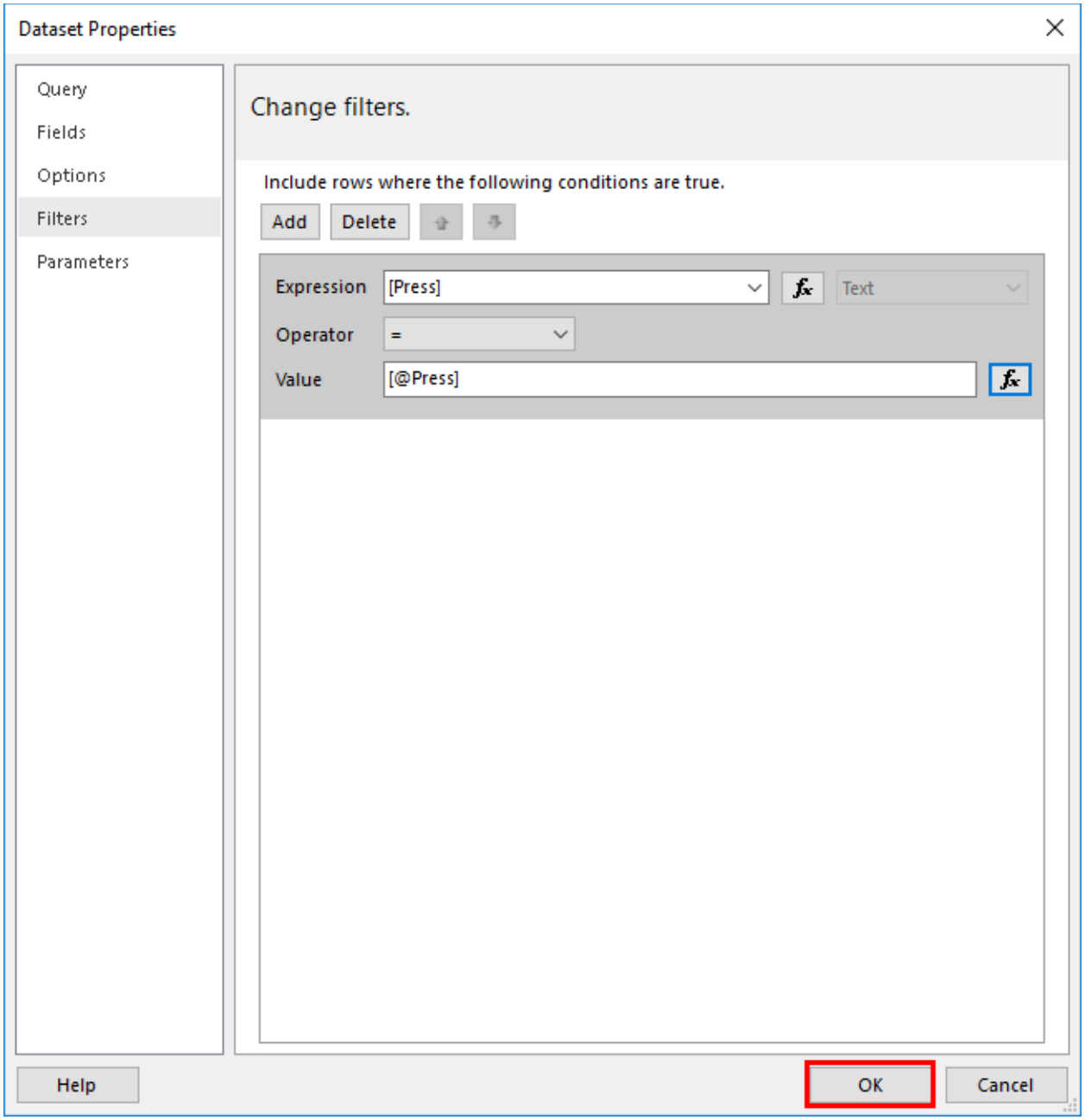
Add a filter. Use Press as the first selection. Click the expression icon.



We want to filter the Dataset to include all rows that match the Press parameter. Double-click Press to complete the expression. Click OK.



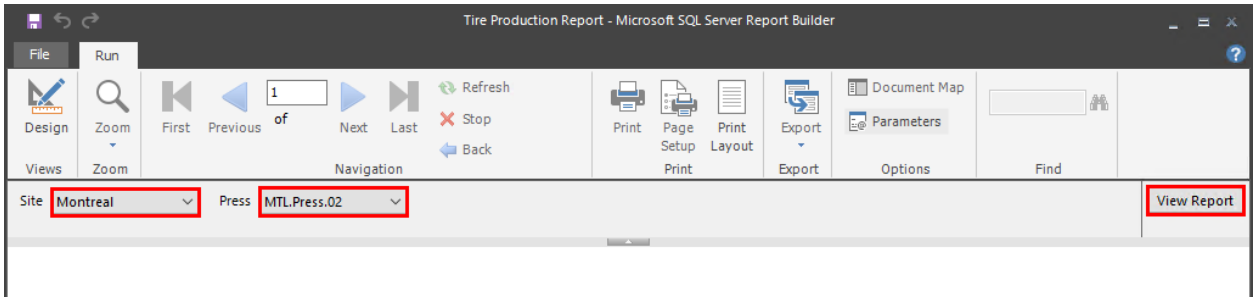
The filter should be configured as follows. Click OK.



To Recap: The Site parameter selection filters the available Press parameter dropdown list. The Press selection filters the TireProduction Dataset.

Let's test it out (**Run the Report**).

Choose a site, choose a press, then click View Report.



The results should be restricted to the parameter selections.

Finally, Save the Report and test it on the Report Server.

Discussion

This is a discussion designed to maximize learning in a specific topic area. Your instructor will have questions, and will prompt for communication within the class. This is an open ended section and the result depends on your needs.

Objective: Let's circle back and compare SSRS to Power BI as alternative reporting solutions.

Approach

- Which tool is easier to configure?
- How much of a game changer is the slicer capability?
- Why would you use one tool over the other?

Appendix C. Substitution Parameters

Defining the Substitution Parameters
The substitution parameters are listed in the following table. The ones in bold are the commonly used "Name" substitution parameters for Elements, Attributes, or Event Frames.

Parameter	Will be replaced by this object's name:
%..\Element%	The name of the owning element of the element in which the attribute resides. To retrieve further ancestors, use the '..\' notations, such as %..\..\Element%.
%..|Attribute%	The name of the owning attribute in which the attribute resides. To retrieve further ancestors, use the '..|' notations, such as %..|..|Attribute%.
%@Attribute%	The value of the attribute referenced. To retrieve further ancestors, use the '..|' notations, such as %@..|..|Attribute%.
%\Element%	The name of the root AF Element in which the attribute resides.
%<Environment Variable>%	The matching System Environment Variable's value. For example %COMPUTERNAME% is replaced with the name of the computer on which the Data Reference is executing.
%Analysis%	The name of the analysis if it can be obtained from the context.
%Attribute%	The name of the attribute that holds this data reference.
%AttributeId%	The attribute ID that holds this data reference.
%Database%	The name of the AF Database in which the attribute resides.
%Description%	The description of the attribute that holds this data reference.
%Element%	The name of the AF Element in which the attribute resides.
%ElementDescription%	The description of the element in which the attribute resides.
%ElementId%	The element ID that holds this data reference.
%EndTime%	The local end time if it can be obtained from the time context.
%Model%	The name of the model if it can be obtained from the context.
%Server%	The name of the default PI Data Archive of the AF Database in which the attribute resides.
%StartTime%	The local start time if it can be obtained from the time context.
%System%	The name of the PI System in which the attribute resides.

%Time%	The local time if it can be obtained from the time context.
%UtcEndTime%	The coordinated universal (UTC) end time if it can be obtained from the time context.
%UtcStartTime%	The coordinated universal (UTC) start time if it can be obtained from the time context.
%UtcTime%	The coordinated universal (UTC) time if it can be obtained from the time context.
.\	The current reference
[.]	The default object of the parent collection. For example .\Elements[.] |Temperature returns the temperature attribute from the primary element of the current reference's Elements collection.
[@filter=text]	The search string in text (e.g. Tank*) matches the given filter. Supported filters are: @Name, @Index, @Template, @Category, @ReferenceType, @Description, @Type, @UOM.
[@Index=#]	Returns the result at location # from the collection result.