PI World 2020 Lab

Build new or migrate existing PI SQL Queries targeting our next generation Real-Time Query Processing Engine

OSIsoft, LLC
1600 Alvarado Street
San Leandro, CA 94577 USA
Tel: (01) 510-297-5800
Web: http://www.osisoft.com

Published: April 17, 2020

Contents

1. Introduction

1.1 Lab Overview

This lab is designed for users familiar with SQL language who want to learn more about the new generation of PI SQL products – PI SQL Client and RTQP Engine.

The lab shows how to create optimized PI SQL Client queries as well as how to transform existing PI OLEDB Enterprise queries into PI SQL Client queries. You will compare performance and features of the aforementioned data providers on two sample AF databases, NuGreen and WindFarm.

1.2 PI SQL Product Family Evolution

The PI SQL product family has quite a long history. Let's have a look at the product evolution in time.

1.2.1 Products Released prior to PI SQL Client and RTQP Engine

1998 – **PI ODBC Driver**

- Exposes PI Data Archive data as an ODBC data source.
- Implements ODBC 2.x standard.
- PI API based.

2001 – **PI OLEDB Provider**

- Exposes PI Data Archive as an OLE DB data source.
- PI SDK based.

2010 – **PI JDBC Driver**

- Exposes PI Data Archive as a JDBC data source.
- Forwards query execution to PI OLEDB Provider instance hosted in a middleware component called PI SQL Data Access Server.
- Supported on Windows and selected Linux platforms.

2010 – **PI OLEDB Enterprise**

- Exposes PI AF as an OLE DB data source.
- AF SDK based.
- With introduction of this driver, PI JDBC was enhanced to support PI Data Archive as well as PI AF queries.

2014 – **PI ODBC Driver**

- Implements ODBC 3.x standard.
- Supports the same queries as PI JDBC because both drivers share PI SQL Data Access Server.

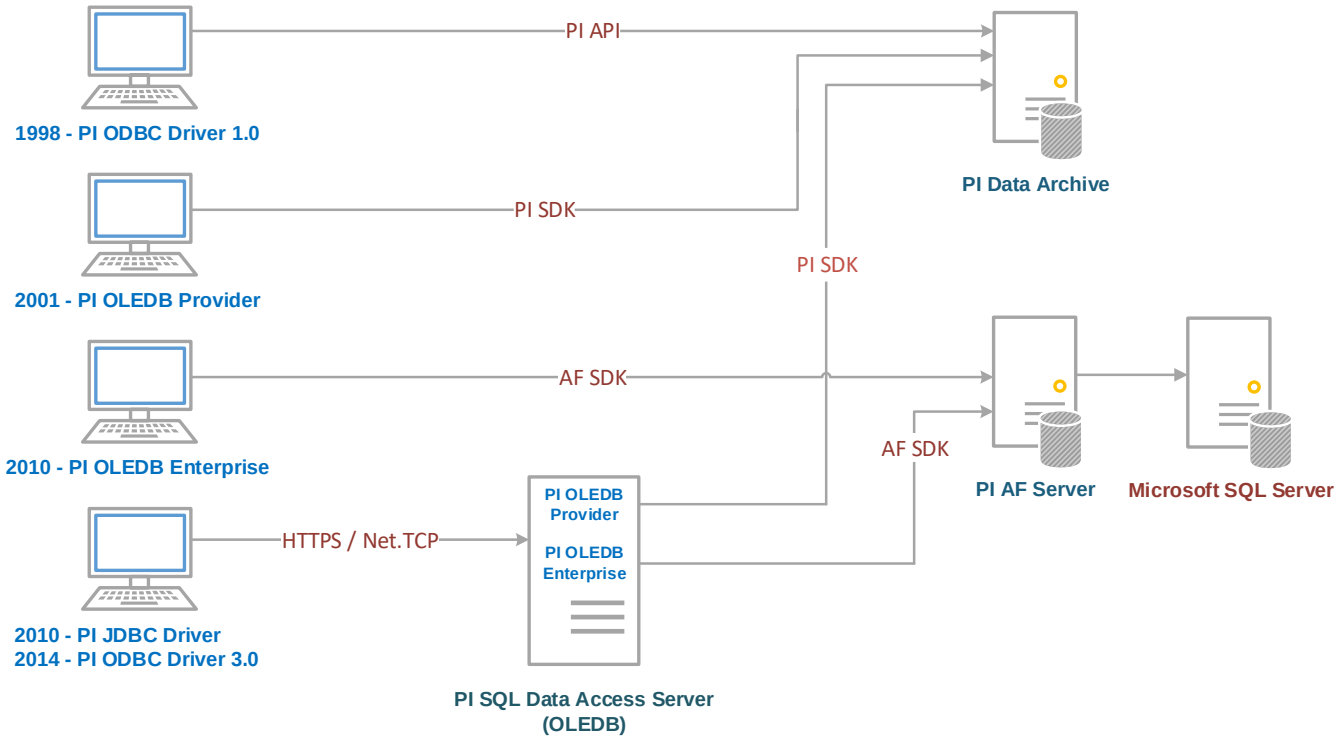


Figure 1 PI SQL Architecture prior to release of PI SQL Client and RTQP Engine

Tip	The PI SQL product family and its evolution prior to release of PI SQL Client is described in detail on our OSIsoft Learning YouTube channel in the **PI SQL Online Course** (on YouTube, search for PI SQL Online Course).

1.2.2 PI SQL Client and RTQP Engine

Finally, to improve performance and scalability issues, **PI SQL Client** and **RTQP Engine** (Real-Time Query Processing Engine) were introduced.

2018 – **PI SQL Client 2018**

- Bundle which is planned to contain all new generation PI SQL drivers (OLE DB, ODBC, JDBC).
- The new drivers forward SQL queries to the completely re-worked SQL query engine running on the AF Server – **RTQP Engine** (Real-Time Query Processing Engine).
- The new data model provides similar read-only access to PI AF data as PI OLEDB Enterprise, but it was de-normalized in order to simplify the SQL queries.
- The first release contained just **PI SQL Client OLEDB**.

2019 – **PI SQL Client 2018 R2**

- New PI SQL Client version which extends the bundle by adding **PI SQL Client ODBC** and multi-platform pure-Java **PI SQL Client JDBC**.

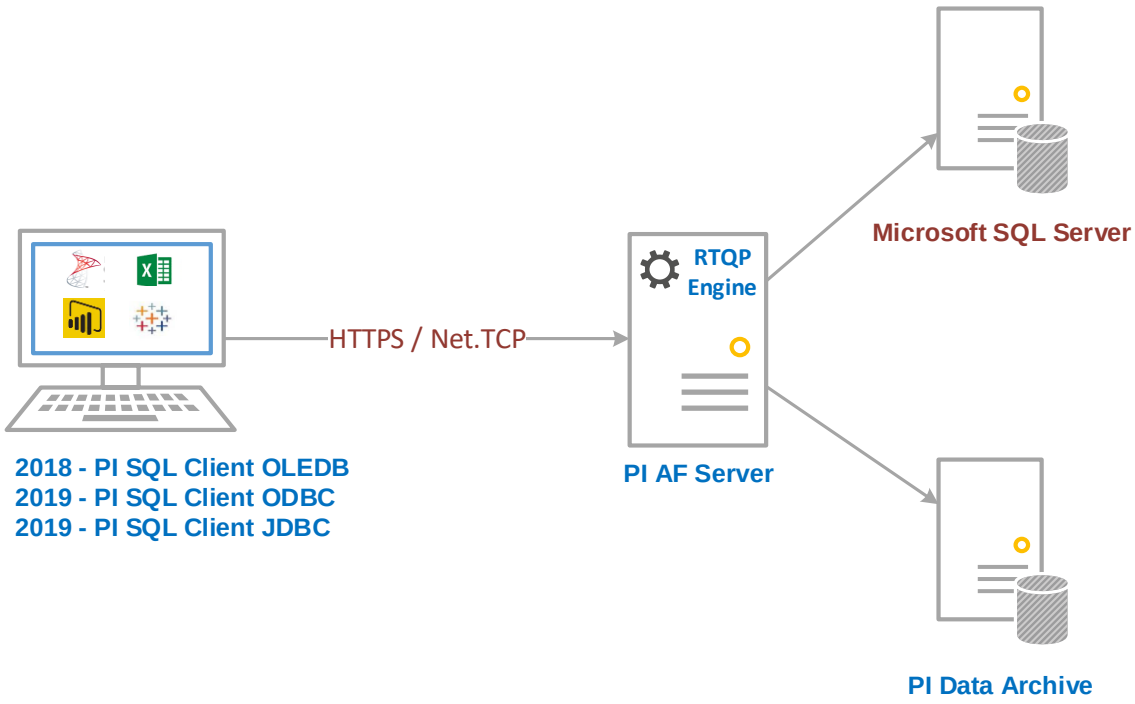


Figure 2 PI SQL Client and RTQP Engine architecture

1.3 Your PI System

This lab uses a simplified PI System demonstration environment. The server is deployed to the Microsoft Azure environment.

1. PISRV01: PI Server running the PI Data Archive, PI AF Server and the PI SQL products
    a. PI Server 2018 SP3 patch 1
    b. PI OLEDB Enterprise 2018
    c. PI SQL Client 2018 R2
    d. Microsoft Visual Studio Community Edition 2019
        - Microsoft Reporting Services Projects Extension
    e. Microsoft SQL Server 2017
2. PIDC: The PISCHOOL domain controller, not accessed for the lab

1.4 Lab content

The lab consists of the following parts:

1. **Get Familiar with PI SQL Client** – in this part, you will configure drivers included in the PI SQL Client bundle and will establish a connection to RTQP Engine using PI SQL Commander Lite and DBVisualizer.
2. **Understand the New Data Model** – in this part, you will learn the key concepts of the new data model. You will look at the model structure, the objects it contains, and how it differs from the PI OLEDB Enterprise data model.
3. **Build RTQP Engine Queries** – in this part, you will learn principles of how to form queries against the new data model.
4. **Migrate Custom Application** – in this part, you will migrate a C# console application from PI OLEDB Enterprise to PI SQL Client.
5. **Integrate PI SQL Client with Microsoft SQL Server Reporting Services** – in this part, you will use PI SQL Client as a data source for a simple Reporting Services report.

2. Get Familiar with PI SQL Client

2.1 Overview

In this chapter, you will look into what drivers are included in the latest PI SQL Client bundle. You will try to configure them and establish a connection to the lab RTQP Engine.

2.2 Goals

- Configure a PI SQL Client ODBC data source.
- Use PI SQL Commander Lite to establish a PI SQL Client ODBC and PI SQL Client OLEDB based connections to the RTQP Engine and to browse the data model.
- Use DBVisualizer to establish a PI SQL Client JDBC based connection and to browse the data model.

2.3 PI SQL Client ODBC

To establish a PI SQL Client ODBC connection using PI SQL Commander Lite, you can either:

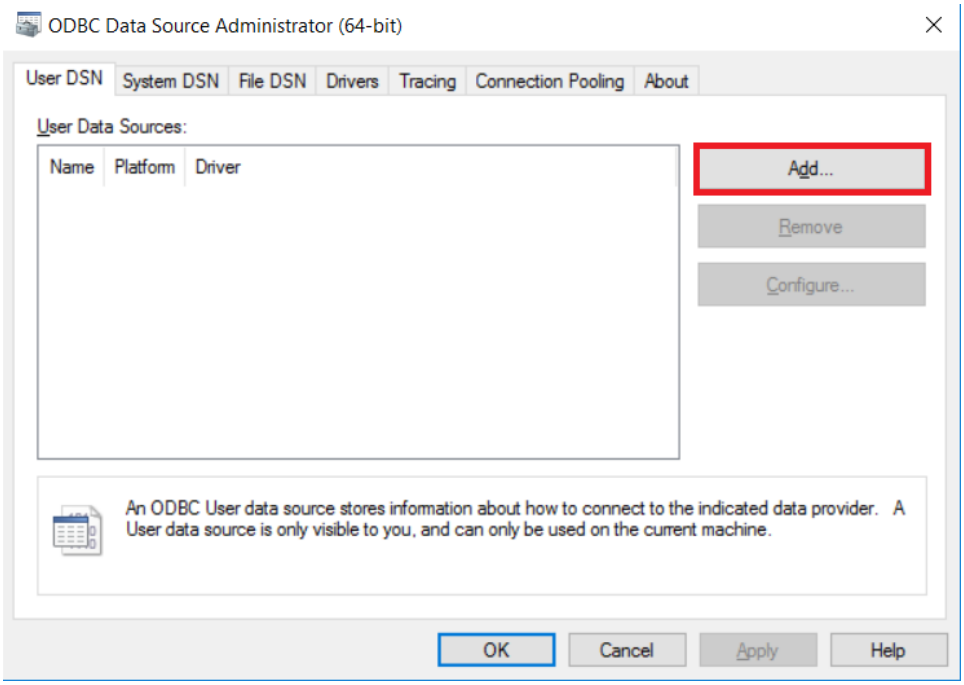
- Establish an ad-hoc PI SQL Client ODBC connection directly in PI SQL Commander Lite.
- Establish a connection using a pre-configured ODBC data source. The data source can then be re-used in all ODBC-aware client applications.

This section will guide you through the process of establishing a connection using a pre-configured ODBC data source.

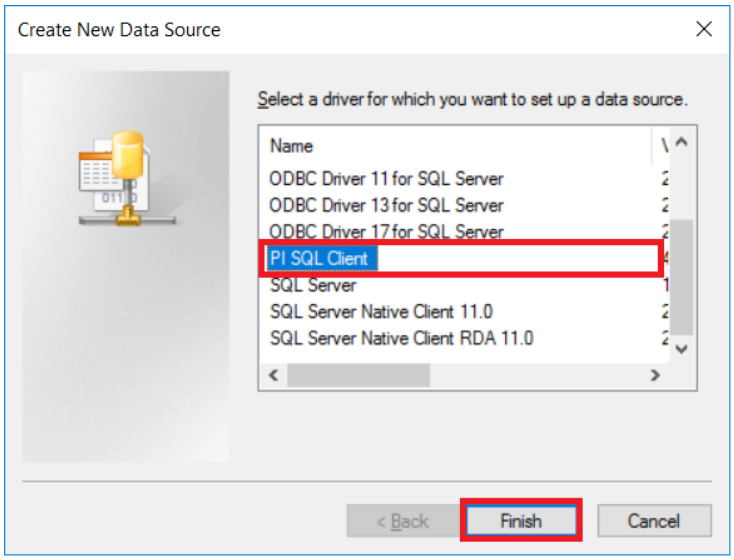
2.3.1 Create a new PI SQL Client ODBC Data Source

1. Launch ODBC Data Source Administrator.

2. Click the **Add** button to create a new user data source.

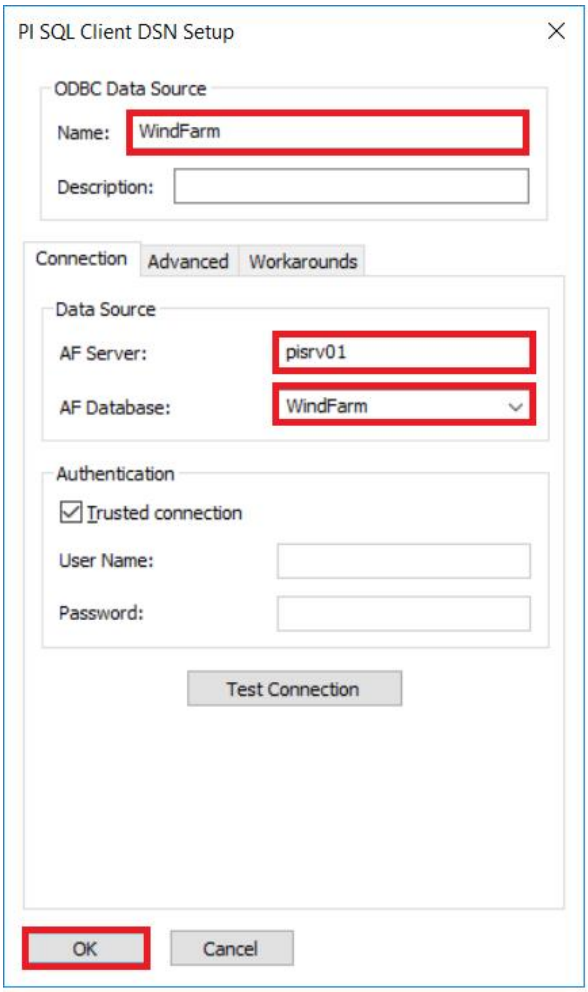


3. Select **PI SQL Client** and click the **Finish** button.



4. Configure the data source according to the screenshot below. Click the **OK** button to confirm the data source creation.

   Notice that for the configuration to be valid, you have to specify the AF database name. In other words, one data source allows you to retrieve data from just one AF database. The details will be discussed in the next chapter.

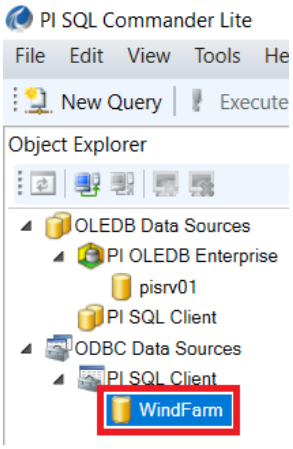


5. Close the ODBC Data Source Administrator.

2.3.2 Establish a PI SQL Client ODBC connection using PI SQL Commander Lite

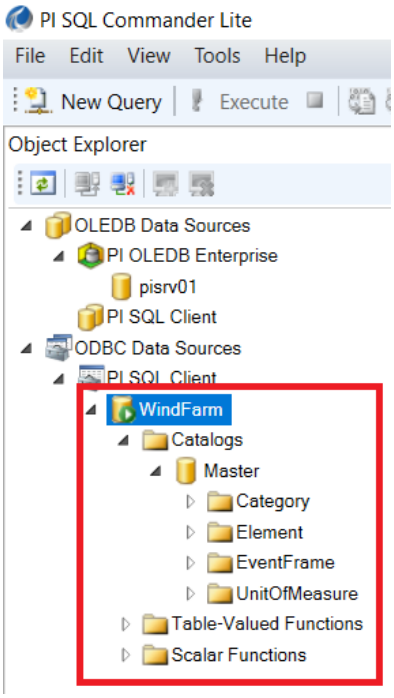
1. Launch PI SQL Commander Lite.

2. Double-click the **ODBC Data Sources-PI SQL Client-WindFarm** node.



3. The PI SQL Commander Lite establishes a connection to the RTQP Engine and displays its data model.

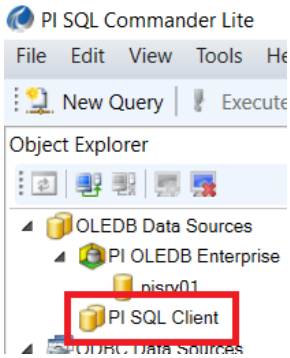
   The data model itself will be described in the next chapter.



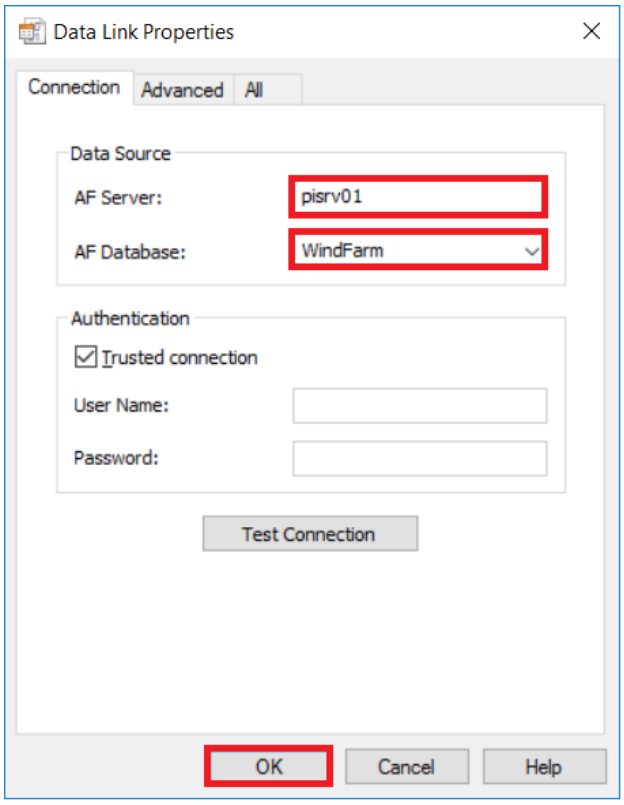
2.4 PI SQL Client OLEDB

In this section, you will use PI SQL Commander Lite to configure and establish a new PI SQL Client OLEDB connection.

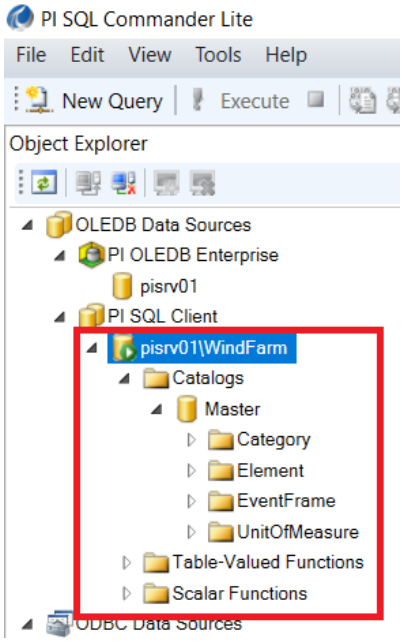
1. In PI SQL Commander Lite, double-click the **OLE DB Data Sources-PI SQL Client** node.



2. Configure the connection and confirm the settings by clicking the **OK** button.



3. The PI SQL Commander Lite establishes a connection to the RTQP Engine and displays its data model.

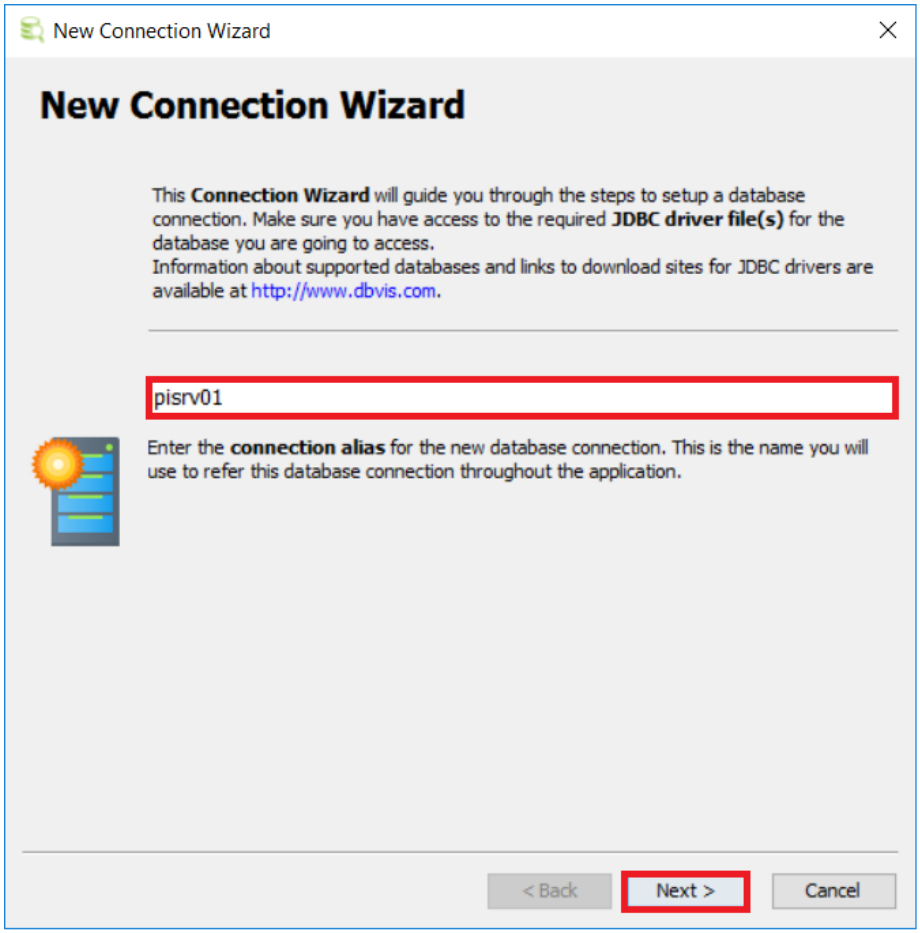


2.5 PI SQL Client JDBC

PI SQL Client JDBC needs a Java programming environment and thus cannot be tested from PI SQL Commander Lite. To see the driver in action, you will configure and establish a connection using a free JDBC testing tool – DBVisualizer.

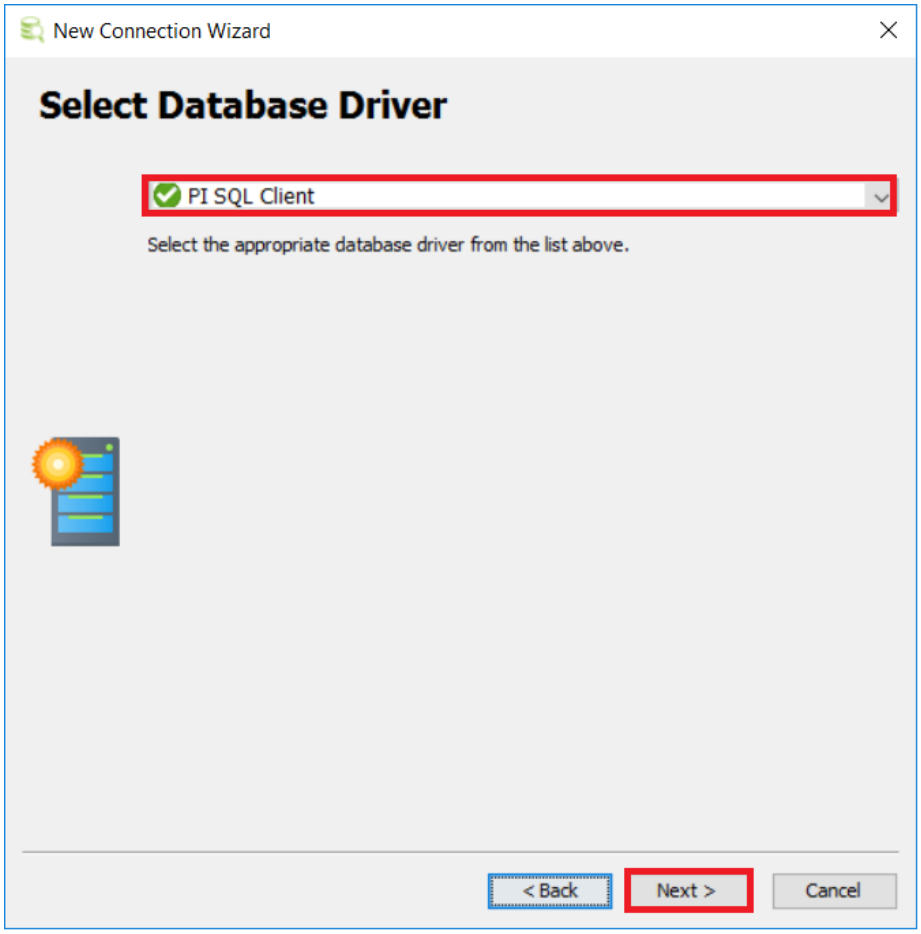
1. Launch DBVisualizer.

2. Set the connection alias to **pisrv01** and click the **Next** button.

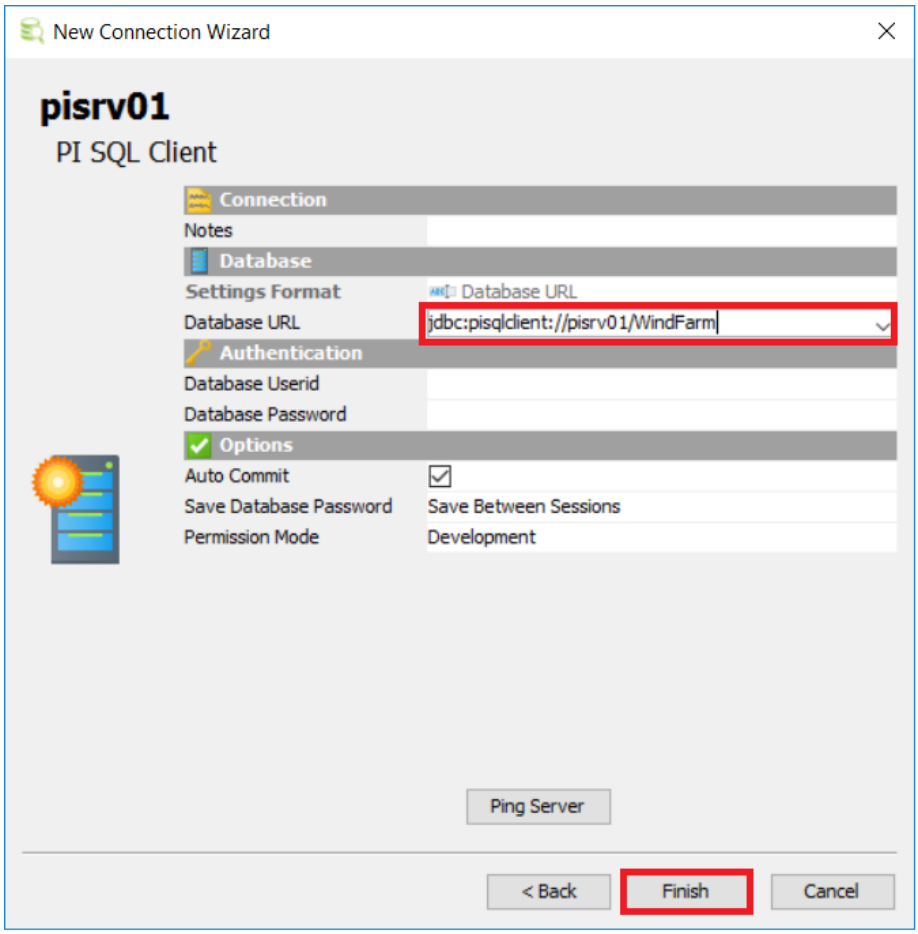


3. Select **PI SQL Client** database driver and click the **Next** button.

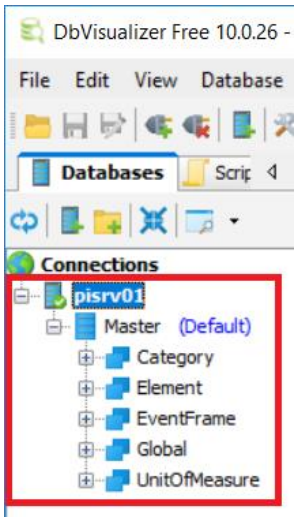
   By default, the list of drivers contains only drivers known to DBVisualizer.
   For PI SQL Client to appear in the list, we had to register it on the lab machine according to these instructions:
   http://confluence.dbvis.com/display/UG100/Installing+a+JDBC+Driver



4. Set the **Database URL** and click the **Finish** button.



5. The DBVisualizer establishes a PI SQL Client JDBC connection to the RTQP Engine and displays its data model. The tool is able to display only data model objects which are defined by the JDBC standard.



3. Understand the New Data Model

3.1 Overview

In this chapter, you will learn the key facts about the new data model.

If you are a PI OLEDB Enterprise user, you will also understand the rationale behind the move from PI OLEDB Enterprise data model to PI SQL Client / RTQP Engine data model.

3.2 Goals

- Use PI SQL Commander Lite to explain the key concepts of the new data model.
- Provide hints for migration of PI OLEDB Enterprise SQL queries.

3.3 Key Concepts of the New Data Model

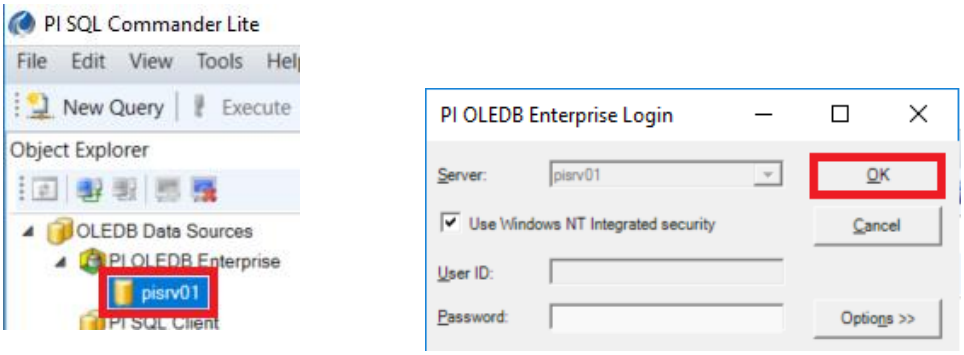
The design of the new data model was driven by lessons learned with the existing PI SQL products, especially PI OLEDB Enterprise. The rationale behind the changes was to:

- Simplify the data model while keeping all common use cases supported.
- Simplify queries.

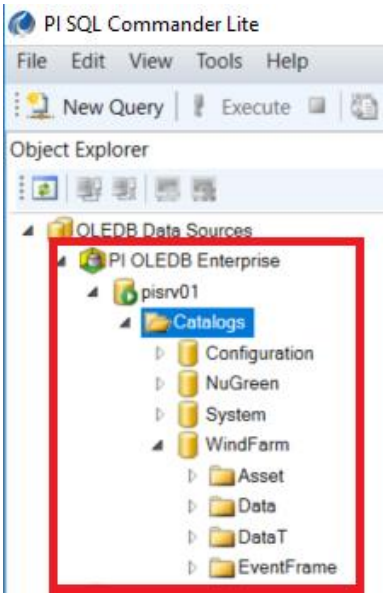
3.3.1 One Data Model Instance Represents Just One AF Database

In PI OLEDB Enterprise, one data model instance (i.e., one connection) allows you retrieve data from all AF databases from the connected AF server. Over time, we learned that the product is not used to execute queries across multiple databases and that having them all in one data model prevents us from better model structuring.

1. Switch back to PI SQL Commander Lite and double-click the **OLE DB Data Sources-PI OLEDB Enterprise-pisrv01** node. Confirm the login dialog default settings by clicking the **OK** button.



2. Browse the PI OLEDB Enterprise data model. Notice that the first hierarchy level of the object names is reserved for the AF database name which makes it confusing to use this level differently, e.g., for user-defined catalogs.



In PI SQL Client / RTQP Engine, one data model instance (i.e., one connection) allows you to retrieve data from just one AF database specified during connection configuration.

1. Use the PI SQL Client connection to the **WindFarm** database (see the previous section) to browse the PI SQL Client data model.

Notice that:

- There is just one top-level item of the name hierarchy – Master catalog.
- There are more types of data model objects – table-valued functions, scalar functions, and templates. PI SQL Client publishes much more metadata information than PI OLEDB Enterprise.

3.3.2 De-Normalization

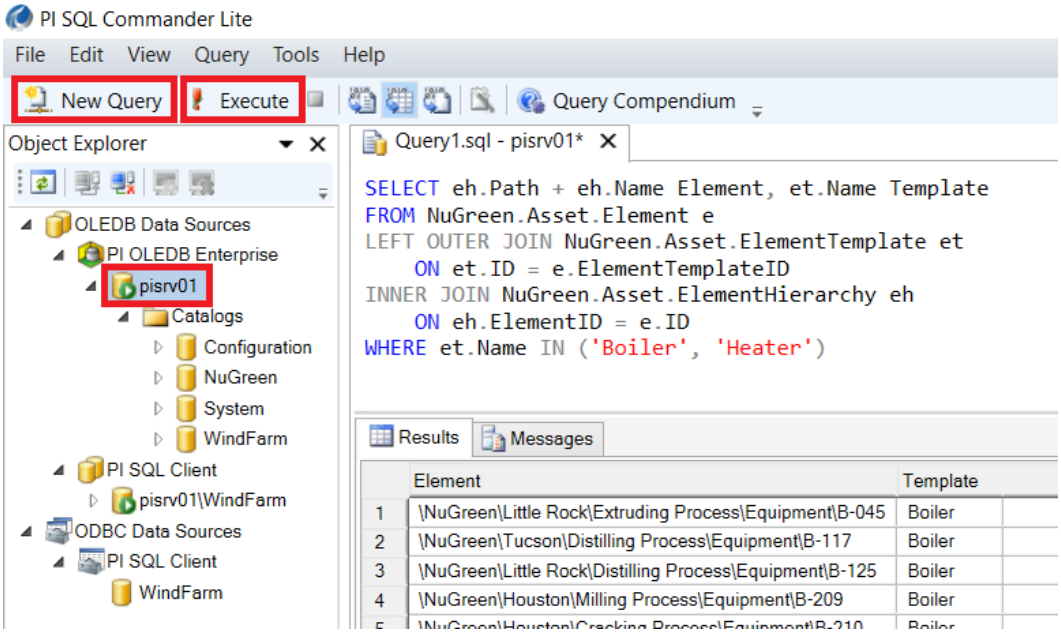
PI OLEDB Enterprise data model was designed to be normalized. This approach, which is recommended for real Relational Database Management Systems (RDBMS), was inappropriate for PI OLEDB Enterprise –
PI OLEDB Enterprise queries tend to be complex due to number of tables which need to be joined.

Thus, the new RTQP Engine data model was designed to be slightly de-normalized. Some tables contain additional columns which in turn decreases number of tables in the FROM clause.

Let's have a look at two equivalent queries – one PI OLEDB Enterprise query and one PI SQL Client query.

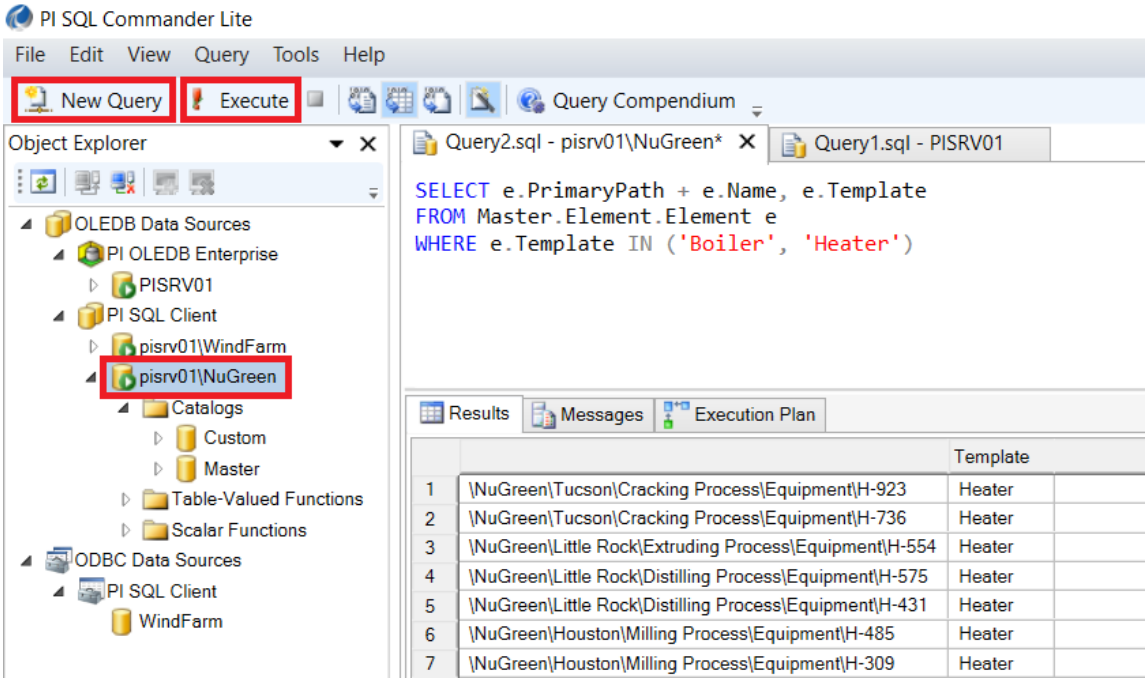
PI OLEDB Enterprise Query

1. Switch back to PI SQL Commander Lite.
2. Select **OLEDB Data Sources-PI OLEDB Enterprise-pisrv01**. If not still connected from the previous section, connect using the default settings.
3. Click the **New Query** button and type the query from the screenshot below. The query searches for all boilers and heaters and returns their hierarchy path.
4. Click the **Execute** button.



PI SQL Client Query

1. Double-click **OLEDB Data Sources-PI SQL Client** node and establish a connection to **pisrv01** AF server and **NuGreen** database.
2. Click the **New Query** button and type the query from the screenshot below.
3. Click the **Execute** button. The query returns the same result as the PI OLEDB Enterprise query above.



As you can see, the PI SQL Client query is much simpler. We could omit two tables from the FROM clause, because the Element table contains PrimaryPath and Template columns.

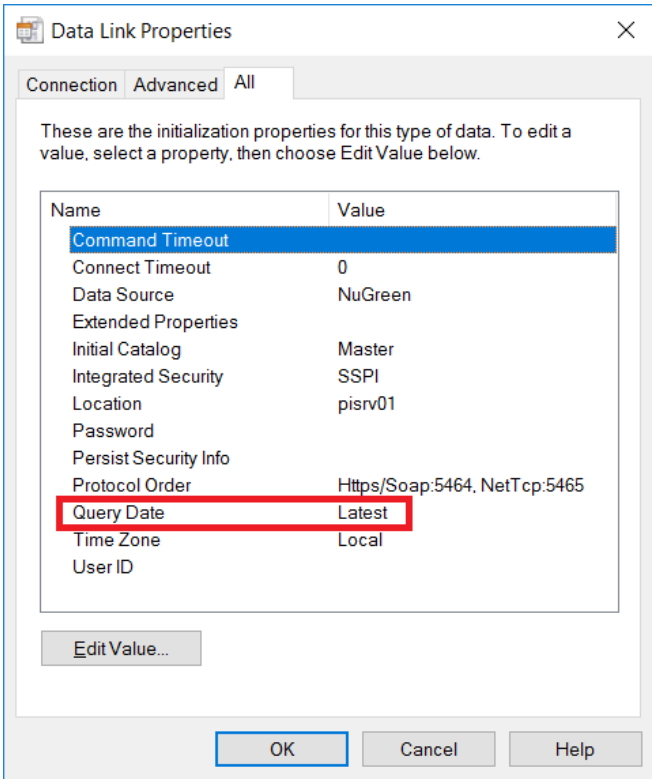
3.3.3 No Element Versioning

Based on the PI OLEDB Enterprise feedback, the element versioning support in the new data model was dropped. The asset representation without versioning is more concise and makes it simpler to pick the correct tables when forming queries.

Although the new data model does not contain versioning-aware element tables, the “non-versioned” element tables still respect the query date, which is set as the connection property.

As an optional exercise, you can try to use PI System Explorer to add a new version to any element in the NuGreen AF database and test PI SQL Client OLEDB connections with different “Query Date” property setting.

As the screenshot below shows, the default value for the “Query Date” property is “Latest”; other allowed values are “Now” or any valid timestamp literal.



3.3.4 Attribute Tables Contain Snapshot Values

To represent element attributes and their snapshot values, PI OLEDB Enterprise data model contains two tables: Asset.ElementAttribute and Data.Snapshot. Thus, to retrieve attributes with snapshot values, you need to form a SQL query which joins them together.

RTQP Engine data model, on the other hand, includes snapshot value columns directly in the Element.Attribute table. Thus, to retrieve attributes with snapshot values, the join with the Snapshot table is not needed (actually, there is no Snapshot table in RTQP Engine).

Event frame attributes and their snapshot values are represented similarly to their element counterparts.

To demonstrate the change, let's write two equivalent queries, one for PI OLEDB Enterprise and one for PI SQL Client. While testing them, you may also notice that the RTQP Engine execution is significantly faster!

Tip	The retrieval of snapshot values is expensive and thus you should include the columns in the query SELECT list only if needed!

PI OLEDB Enterprise Query

Query1.sql - PISRV01*

```
SELECT e.Name, ea.Name, s.Value
FROM WindFarm.Asset.Element e
INNER JOIN WindFarm.Asset.ElementAttribute ea ON ea.ElementID = e.ID
INNER JOIN WindFarm.Data.Snapshot s ON s.ElementAttributeID = ea.ID
WHERE e.Name LIKE 'Turbine1%'
```

Results | Messages

	Name	Name	Value
1	Turbine10000	Point Count	14
2	Turbine10000	Yield	5.399798E-05
3	Turbine10000	Wind Speed	8.97145175933838
4	Turbine10000	Wind Farm	San Gorgonio Pass Wind Farm
5	Turbine10000	Wind Direction	7.20936107635498
6	Turbine10000	Unit Status	Active
7	Turbine10000	Turbine State	Inactive
8	Turbine10000	Technology	Wind
9	Turbine10000	State 3	Schedule Maint
10	Turbine10000	State 2	Maint This Week
11	Turbine10000	State 1	Maint This Week
12	Turbine10000	Shift Hours	12
13	Turbine10000	Shift	15
14	Turbine10000	RPM	31.4937591552734
15	Turbine10000	Region	California

Query executed successfully | PISRV01 | PI OLEDB Enterprise | 00:01:06.719 | 133380 rows

PI SQL Client Query

Query2.sql - pisrv01\WindFarm* | Query1.sql - PISRV01*

```
SELECT Element, Name, Value
FROM Master.Element.Attribute
WHERE Element LIKE 'Turbine1%'
```

Results | Messages | Execution Plan

	Element	Name	Value
1	Turbine11477	MW Lost	31.625111294682
2	Turbine11477	Longitude	8343
3	Turbine11477	Latitude	45
4	Turbine11477	Gross Generation	88.11544
5	Turbine11477	Generating Efficiency	62.8067164042996
6	Turbine11477	Expected Power Output	56.4903290739704
7	Turbine11477	Distance from Control Room	263.819811201539
8	Turbine11477	Demand	54.82259
9	Turbine11477	Capacity	320000
10	Turbine11477	Point Count	14
11	Turbine11558	Yield	0.001275923
12	Turbine11558	Wind Speed	74.3295745849609
13	Turbine11558	Wind Farm	Lundgren Wind Farm
14	Turbine11558	Wind Direction	98.1897506713867
15	Turbine11558	Unit Status	Inactive

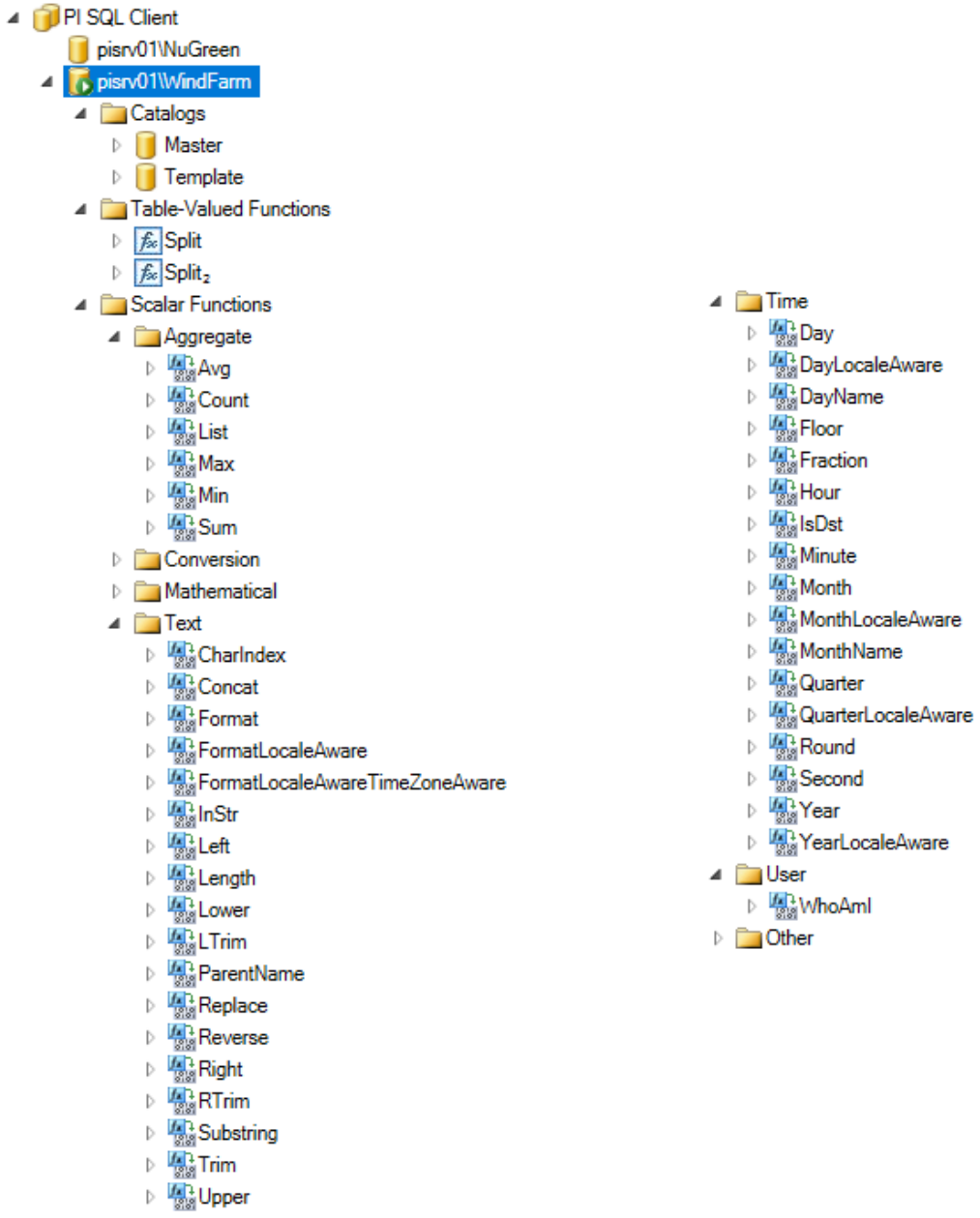
Double

Query executed succ | pisrv01\WindFarm | PI SQL Client OLEDB | 00:00:35.062 | 133380 rows

3.3.5 Helper Functions

The new RTQP Engine supports many functions, which you can use in your queries – scalar, aggregate, and even table-valued functions.

For PI SQL Client connections, you can now browse the supported function declarations in PI SQL Commander Lite.



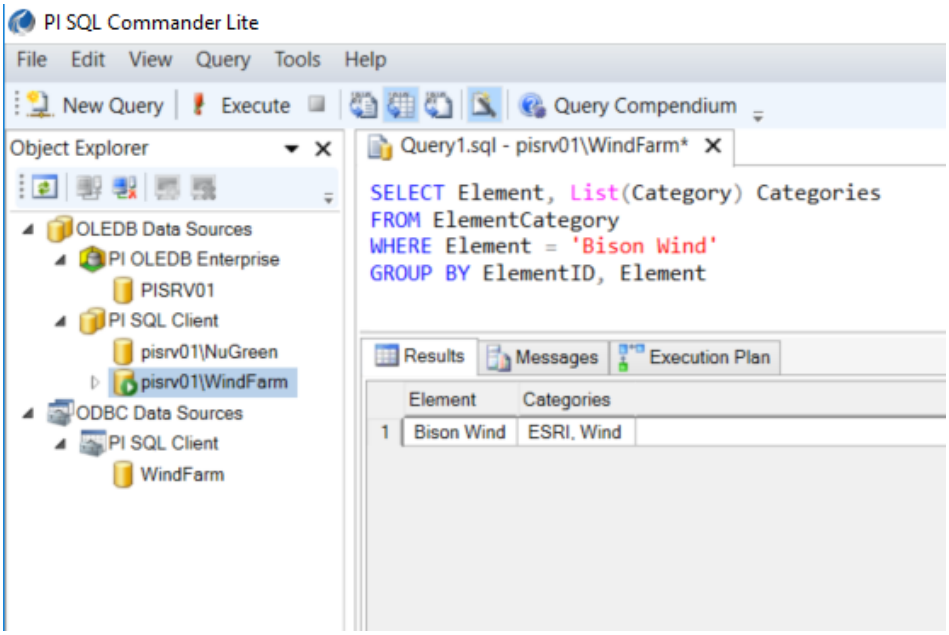
Tip	PI Server 2018 SP3 release contains a bug in RTQP Engine which makes internal functions visible (e.g., FormatLocaleAware or all functions of **Other** category). To avoid using internal functions in your queries, please consult **SQL for RTQP Engine Reference Guide** which lists all public functions (available at https://livelibrary.osisoft.com).

Let's have a look at some typical function use cases. Switch back to PI SQL Commander Lite and open a query window for either PI SQL Client OLEDB or PI SQL Client ODBC connection to the WindFarm database.

List

The **List** function is an aggregate function which allows you to transform multiple column values in the result into a single string value.

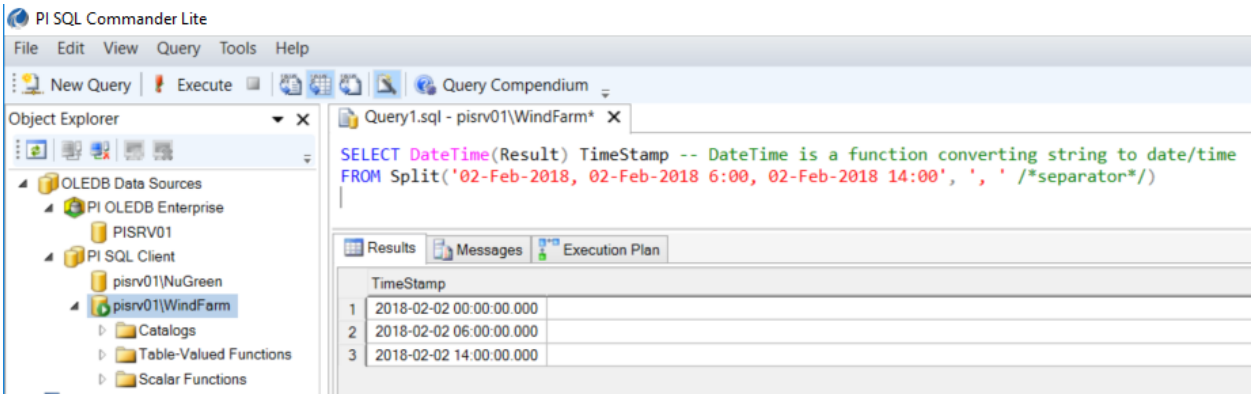
An example of how this function can be used is a query, which returns a list of categories an element is member of.



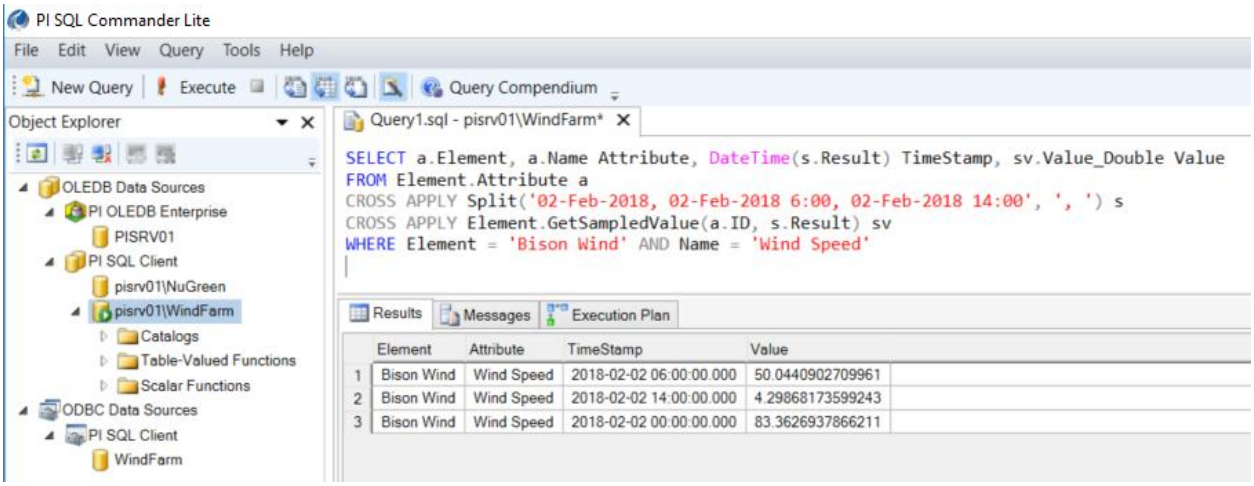
Split

The **Split** table-valued function is kind of an inverse function to the List aggregate function. It allows you to split a string value into a column with multiple values. The typical use case for this function is a query returning sampled values at several defined timestamps.

However, before we move on to the sample value query, try to play around with the Split function using the following simple example!

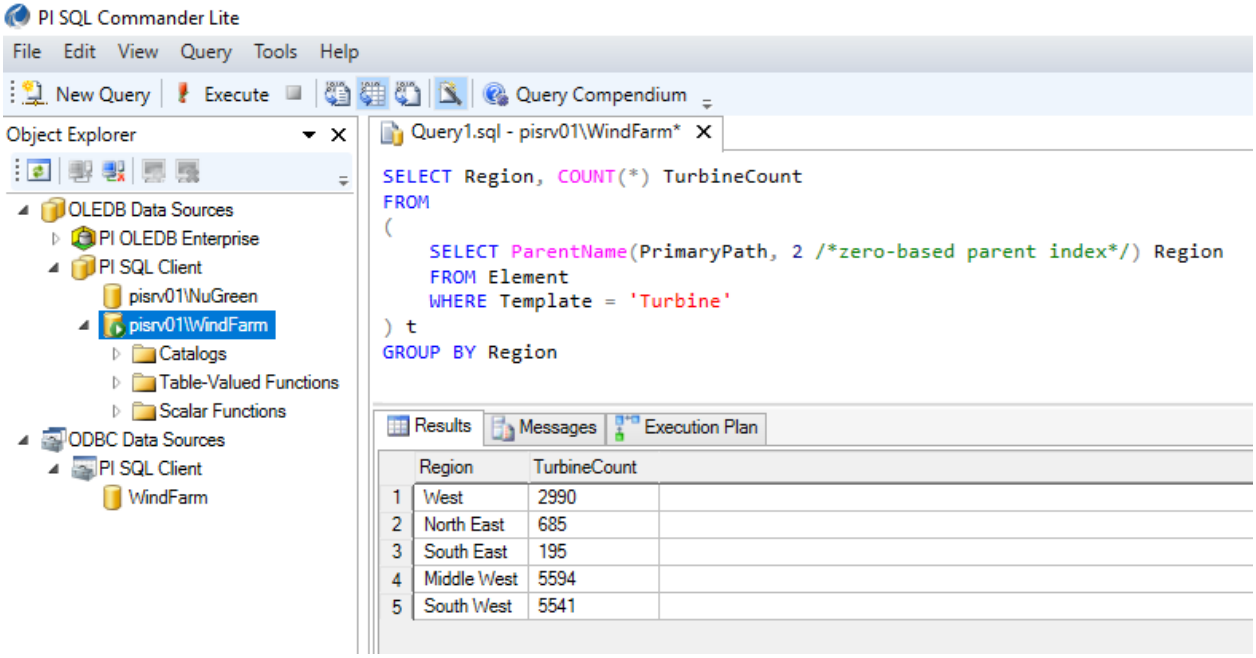


Next, use the Split function from the previous example to form the sampled value query. Note that the query shown below assumes that the AF database contains just one element named **Bison Wind**.



ParentName

The **ParentName** scalar function was added to simplify queries accessing hierarchies. It allows you to get the specified part of the path. The following screenshot captures the typical use case.

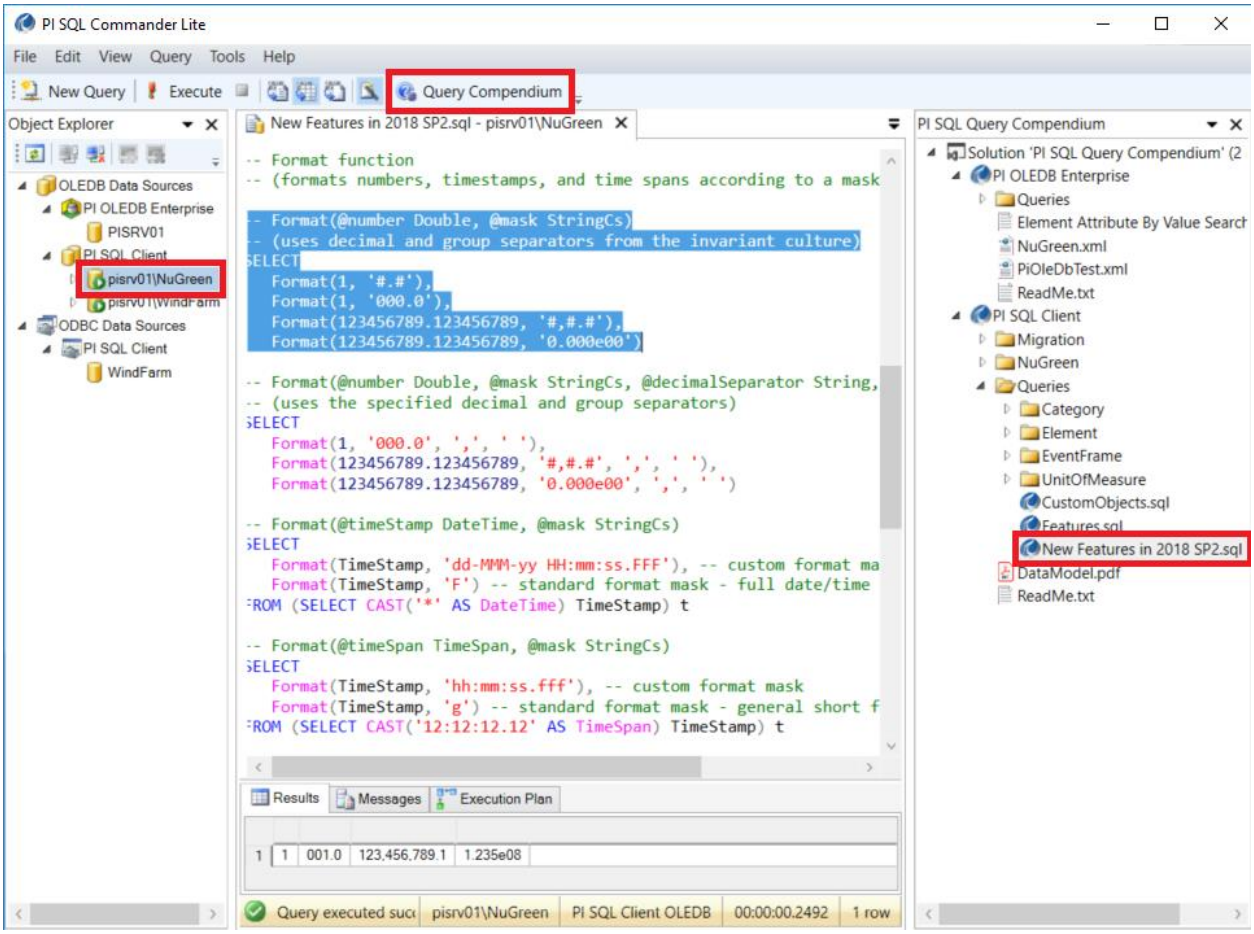


Format

The **Format** scalar function is very handy for report queries. It allows you to format numbers, timestamps, and time spans using various format masks.

To see this function in action, click the **Query Compendium** button in PI SQL Commander Lite, open the **New Features in 2018 SP2.sql** file, and find the part with **Format** function examples.

*Note that the **Format** function example queries do not reference any tables so they can be executed using any PI SQL Client connection. The rest of the compendium is bound to the **NuGreen** AF database. So if you want to also look at other queries in the compendium, use the NuGreen database PI SQL Client connection (i.e., select the **NuGreen** connection node in the **Object Explorer** before you double-click the compendium file).*



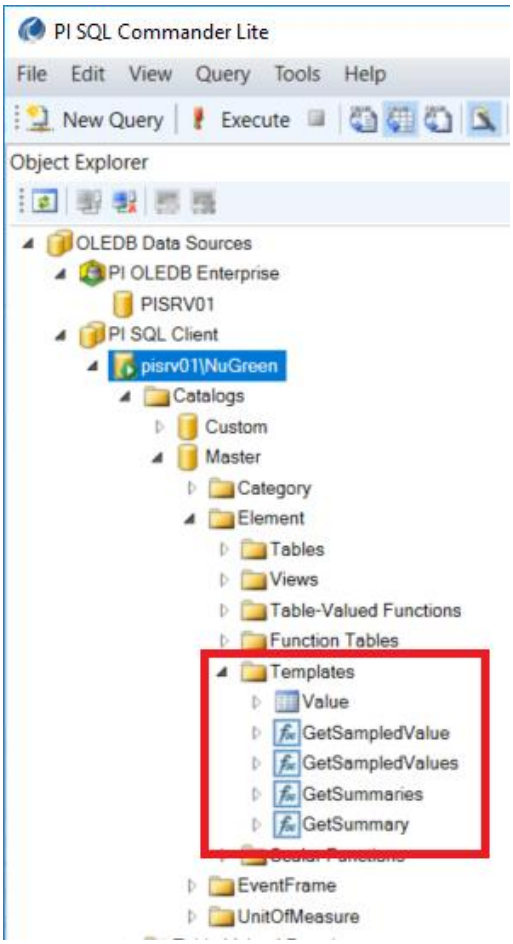
Tip	The Query Compendium contains many query examples you may find useful when putting together your own queries. It is split into two parts: - Migration part for customers with PI OLEDB Enterprise experience, - Queries part which does not assume any previous experience with PI SQL products. Before you form your own queries, we recommend you browse through the compendium to get familiar with the typical query patterns.

3.3.6 Table-Valued Function and Table Templates

Table-valued function (TVF) templates and table templates may seem complex at first. Although they were used internally in PI OLEDB Enterprise, they were not documented and displayed by PI SQL Commander Lite.

With PI SQL Client, we decided to make the templates visible and explain how they work, because the value they bring is significant.

Let's have a look at them in PI SQL Commander Lite.



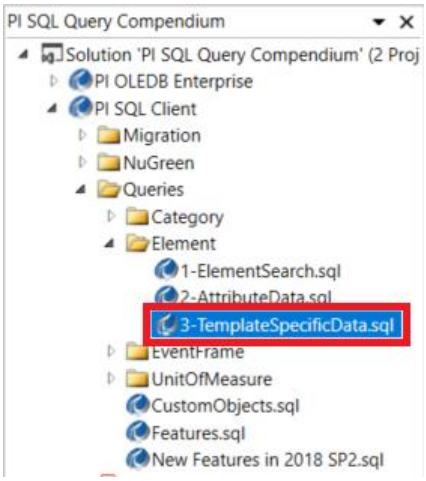
There are two similar groups of templates: one template group can be found under the **Element** schema, the other one under the **EventFrame** schema.

The first group allows you to retrieve time-series data for elements created from the same AF element template (e.g., sampled values of the **Fuel Gas Flow** attribute of all boilers). The second group does the same job for event frames.

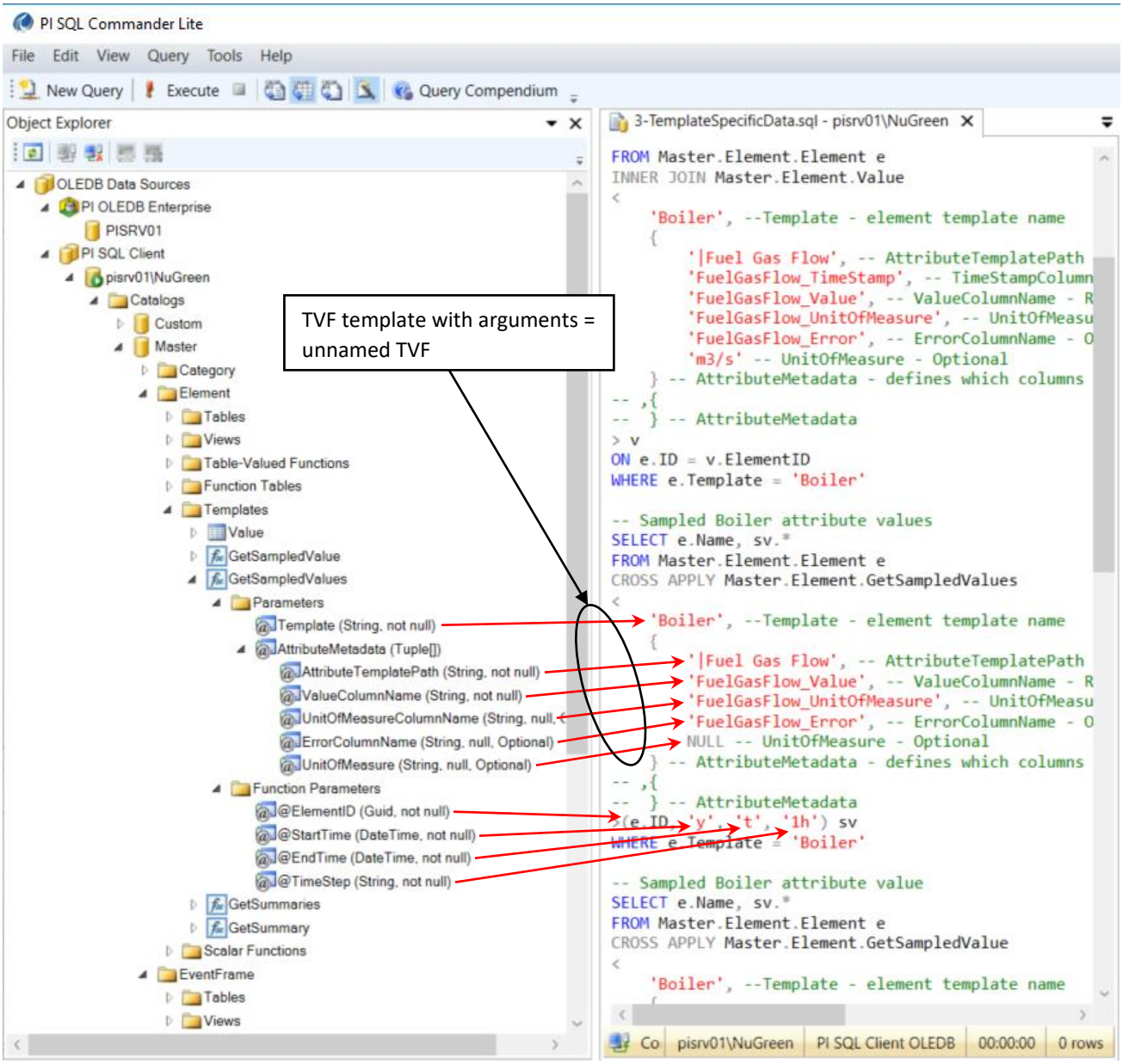
Tip	**Do not confuse TVF and table templates with AF element templates and AF event frame templates!**

The new data model contains templates to retrieve snapshot values, sampled values (i.e., interpolations), and summaries.

Let's go back to PI SQL Commander Lite and open the **Query Compendium 3-TemplateSpecificData.sql** file according to the following screenshots.



Inspect the query, which retrieves **Sampled Boiler attribute values,** and execute it.



First, let's understand what data the example query retrieves. This is the execution result.

Results | Messages | Execution Plan

	Name	TimeStamp	FuelGasFlow_Value	FuelGasFlow_UnitOfMeasure	FuelGasFlow_Error
1	B-914	2020-02-27 00:00:00.000	49.1125144958496	ft3/s	
2	B-914	2020-02-27 01:00:00.000	73.1765975952148	ft3/s	
3	B-914	2020-02-27 02:00:00.000	91.822151184082	ft3/s	
4	B-914	2020-02-27 03:00:00.000	98.3009185791016	ft3/s	
5	B-914	2020-02-27 04:00:00.000	91.4322509765625	ft3/s	
6	B-914	2020-02-27 05:00:00.000	74.2927856445313	ft3/s	
7	B-914	2020-02-27 06:00:00.000	50.5639038085938	ft3/s	
8	B-914	2020-02-27 07:00:00.000	26.8350238800049	ft3/s	
9	B-914	2020-02-27 08:00:00.000	8.42470264434814	ft3/s	
10	B-914	2020-02-27 09:00:00.000	1.74080300331116	ft3/s	
11	B-914	2020-02-27 10:00:00.000	8.08253955841064	ft3/s	
12	B-914	2020-02-27 11:00:00.000	25.0629348754883	ft3/s	
13	B-914	2020-02-27 12:00:00.000	49.1402587890625	ft3/s	
14	B-914	2020-02-27 13:00:00.000	73.2175827026367	ft3/s	
15	B-914	2020-02-27 14:00:00.000	91.3696365356445	ft3/s	
16	B-914	2020-02-27 15:00:00.000	98.3054351806641	ft3/s	
17	B-914	2020-02-27 16:00:00.000	91.9972534179688	ft3/s	
18	B-914	2020-02-27 17:00:00.000	74.9231719970703	ft3/s	
19	B-914	2020-02-27 18:00:00.000	49.9889945983887	ft3/s	
20	B-914	2020-02-27 19:00:00.000			Shutdown
21	B-914	2020-02-27 20:00:00.000	8.63867855072021	ft3/s	
22	B-914	2020-02-27 21:00:00.000	1.72884476184845	ft3/s	
23	B-914	2020-02-27 22:00:00.000	8.05380725860596	ft3/s	
24	B-914	2020-02-27 23:00:00.000	25.0696048736572	ft3/s	
25	B-914	2020-02-28 00:00:00.000	49.1531944274902	ft3/s	
26	B-352	2020-02-27 00:00:00.000	49.1125144958496	ft3/s	
27	B-352	2020-02-27 01:00:00.000	73.1765975952148	ft3/s	
28	B-352	2020-02-27 02:00:00.000	91.822151184082	ft3/s	
29	B-352	2020-02-27 03:00:00.000	98.3009185791016	ft3/s	
30	B-352	2020-02-27 04:00:00.000	91.4322509765625	ft3/s	
31	B-352	2020-02-27 05:00:00.000	74.2927856445313	ft3/s	

As you can see, the result contains sampled values (i.e., interpolations) of the **Fuel Gas Flow** attribute of elements created from the **Boiler** element template.

What syntax did we use to get this result?

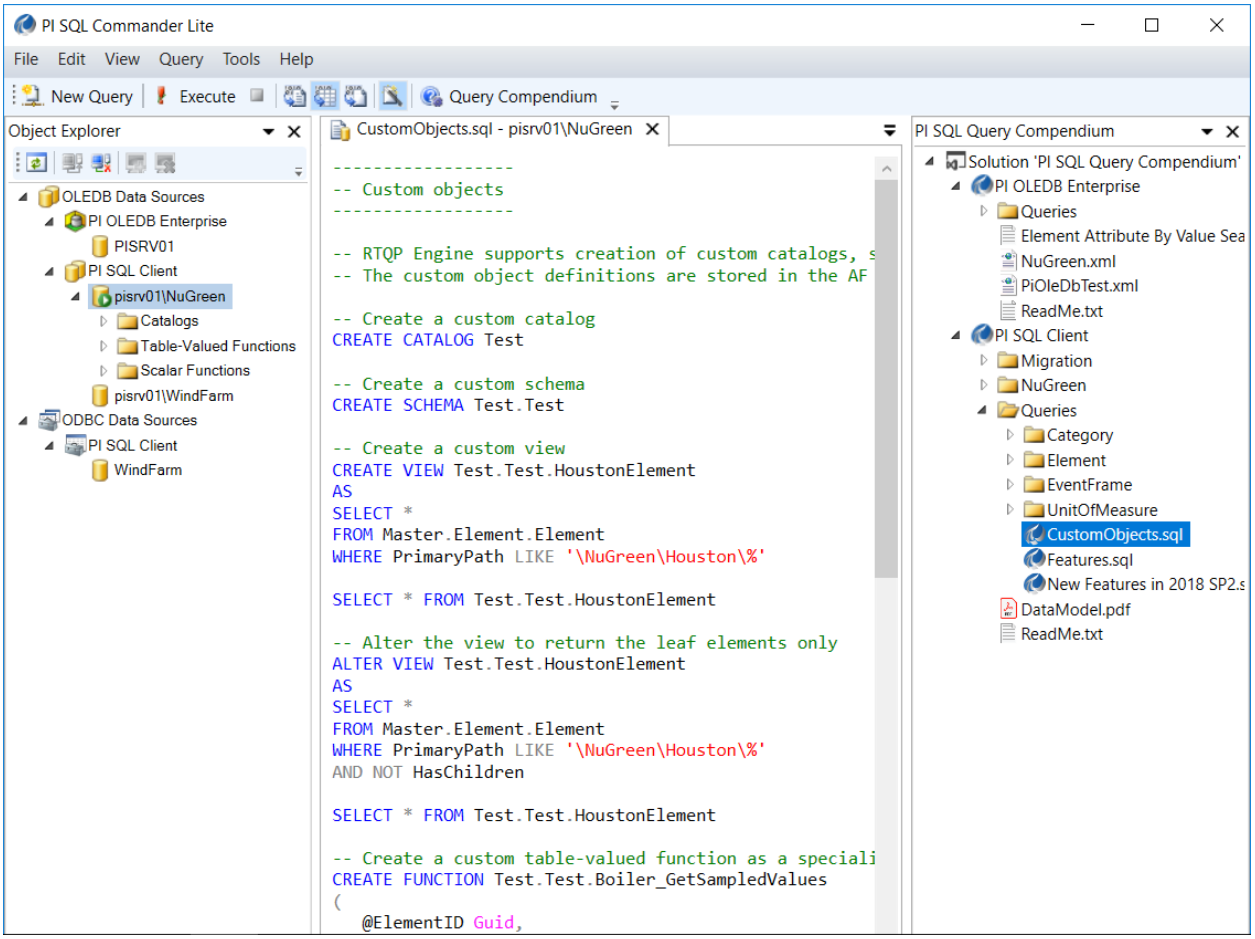
1. We created an unnamed TVF by instantiating the **GetSampledValues** TVF template. In other words, we provided template arguments to the GetSampledValues TVF template – we specified that we want to use the TVF template for the **Boiler** element template and that we want to retrieve data for the **Fuel Gas Flow** attribute and also return its unit of measure and errors (parameters inside angle brackets).
2. Then, we called this unnamed TVF with a set of arguments. We specified that we want to retrieve sampled values for yesterday, with one-hour step.

There are more example queries for TVF and table templates in the compendium. Try to play around with them and understand the syntax. Please ask questions!

3.3.7 User-Defined Objects

RTQP Engine supports various types of user-defined objects. You can create your own catalogs, schemas, views, and even table-valued functions.

The Query Compendium contains example queries demonstrating this functionality, so switch to PI SQL Commander Lite and test them! Since the queries are based on the **NuGreen** AF database, do not forget to open the **CustomObjects.sql** file under the appropriate PI SQL Client connection.

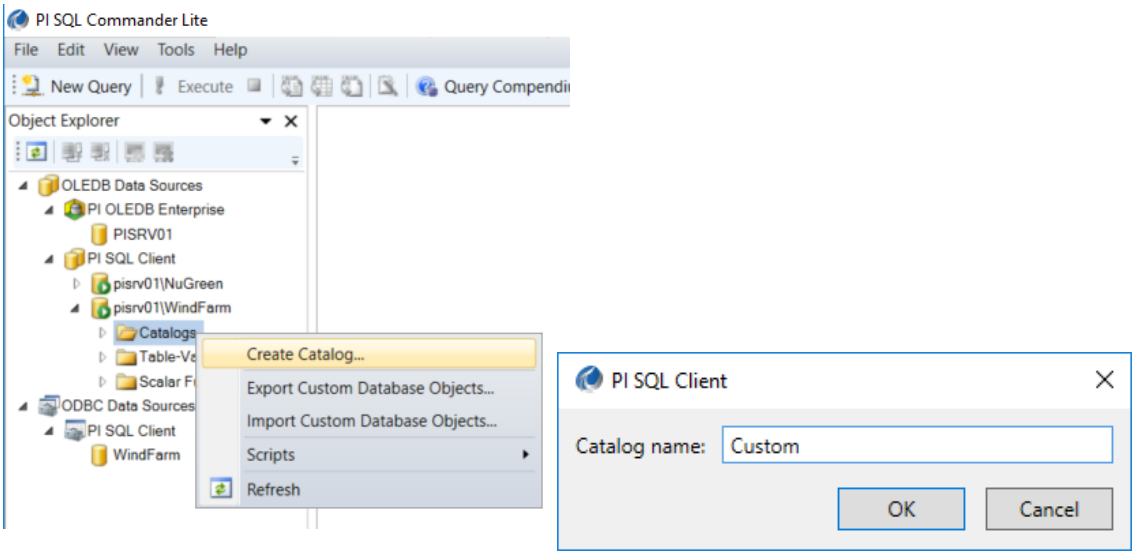


Tip	The RTQP Engine data model contains only one built-in catalog named Master which prevents name collisions with user-defined catalogs. Besides user-defined catalogs described in this section, our future plan is to support user-defined catalogs which would directly expose PI Data Archive data (without the need to define AF attributes). To prioritize this feature, we need to collect enough customer voices. So if you want to see this functionality in one of the next releases, vote for it at https://feedback.osisoft.com/forums/555145-pi-developer-technologies/suggestions/38984431-allow-direct-pi-data-archive-server-exposure

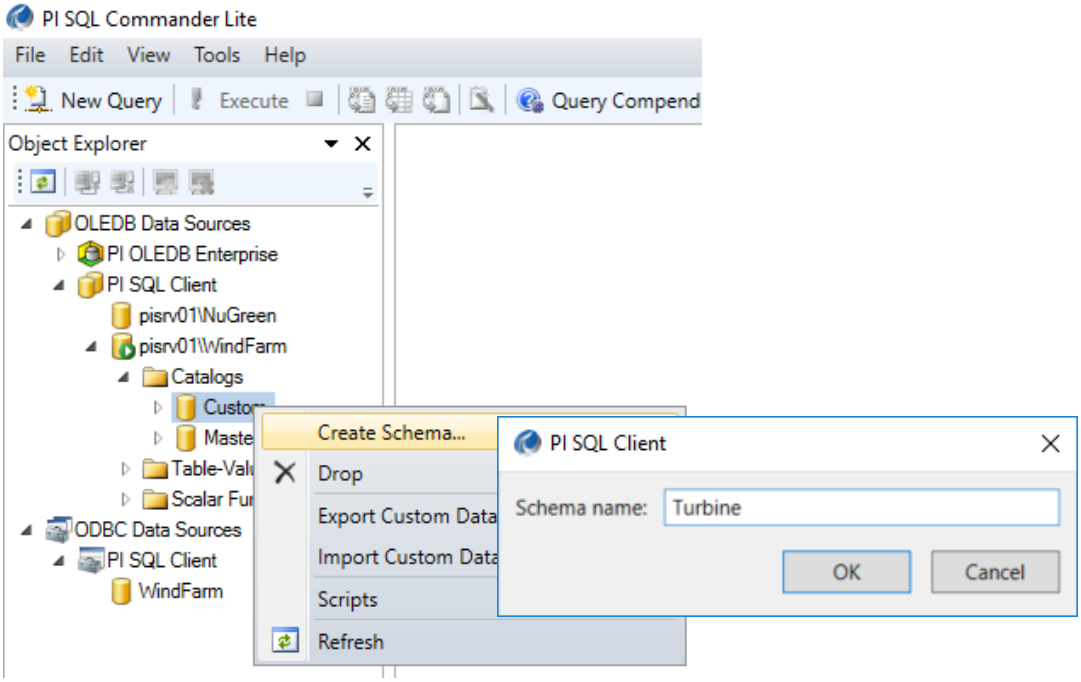
3.3.8 User-Defined Template-Specific Data Models

Have you had a hard time understanding how to use TVF templates and table templates? We have good news for you – PI SQL Commander Lite provides a wizard UI to create user-defined objects wrapping them. Let's have a look in detail.

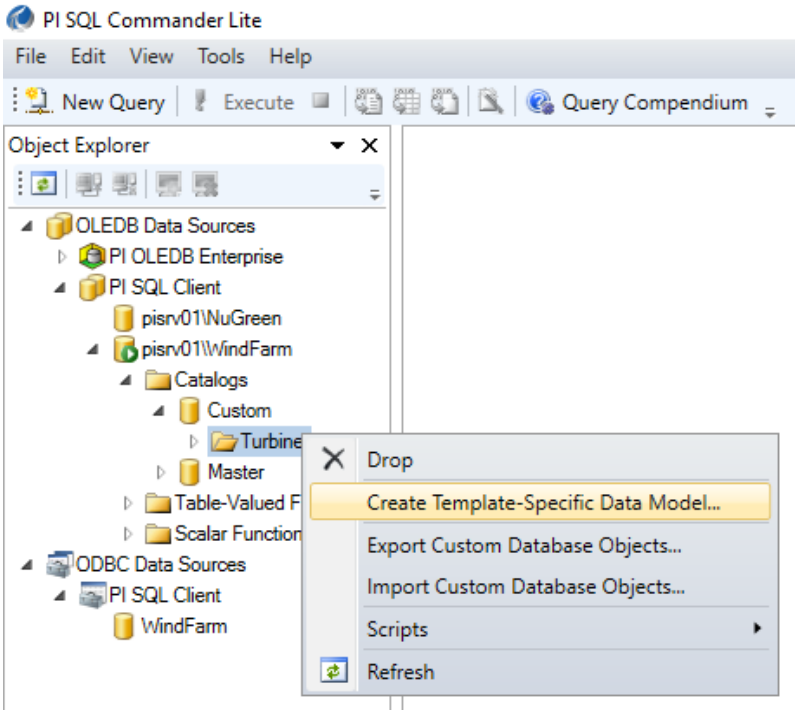
1. In PI SQL Commander Lite, under the PI SQL Client OLEDB p**isrv01\WindFarm** connection, right-click the **Catalogs** node in the **Object Explorer** and select the **Create Catalog…** menu item. Name our new catalog **Custom**.



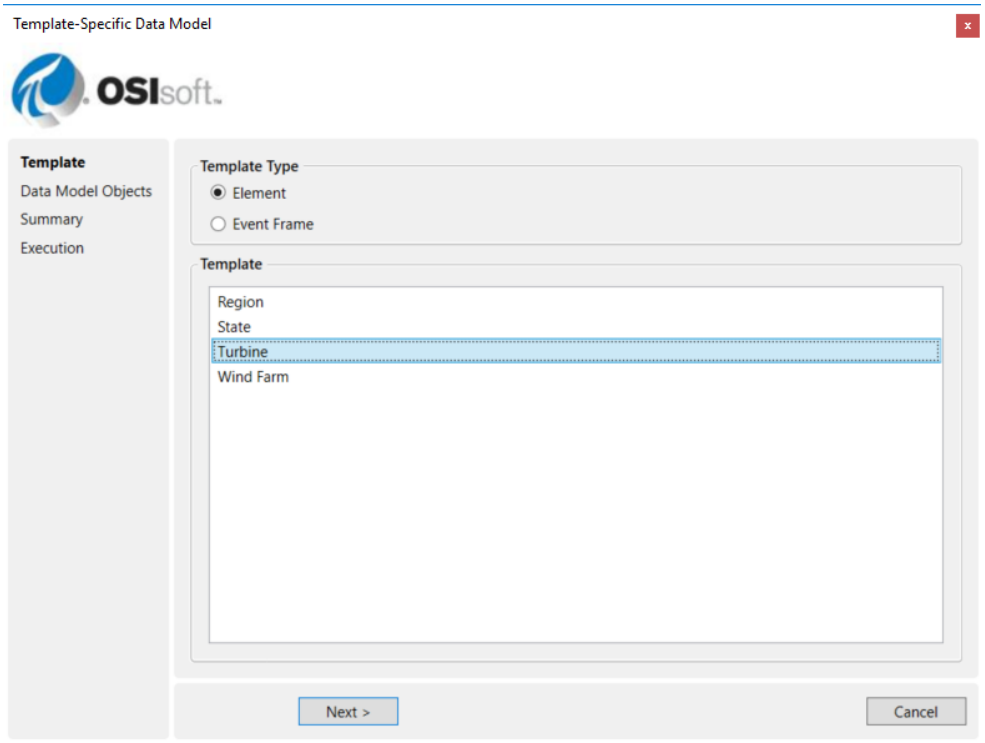
2. Right-click the **Custom** catalog and select the **Create Schema…** menu item. Name the new schema **Turbine** to indicate it will contain turbine-specific objects.



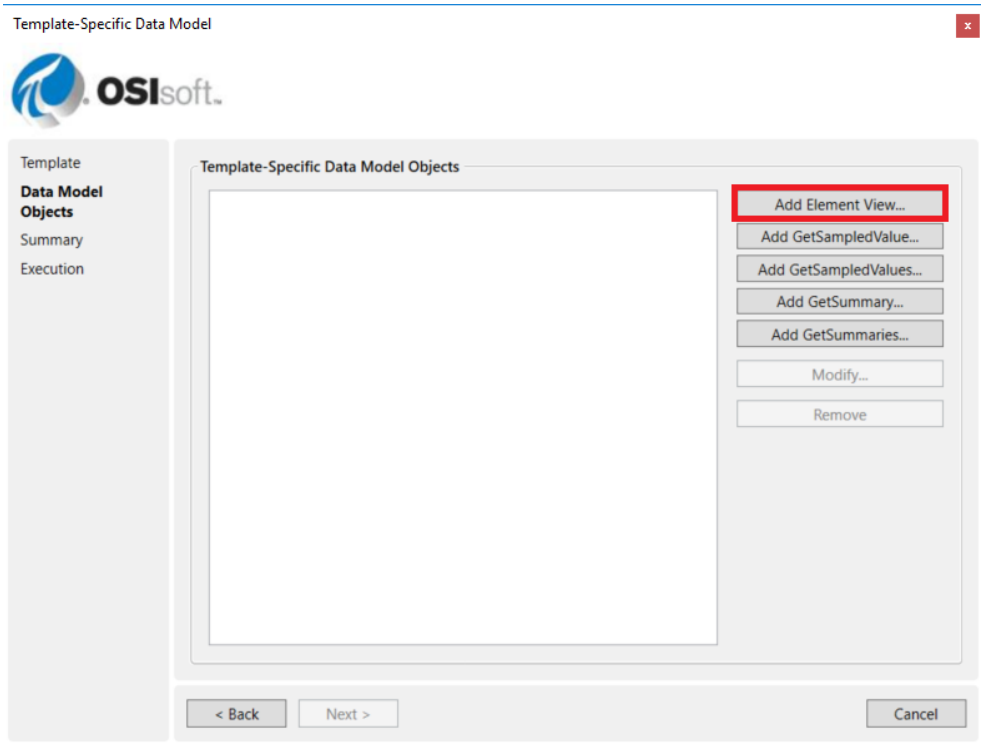
3. Right-click the **Turbine** schema and select the **Create Template-Specific Data Model…** menu item.



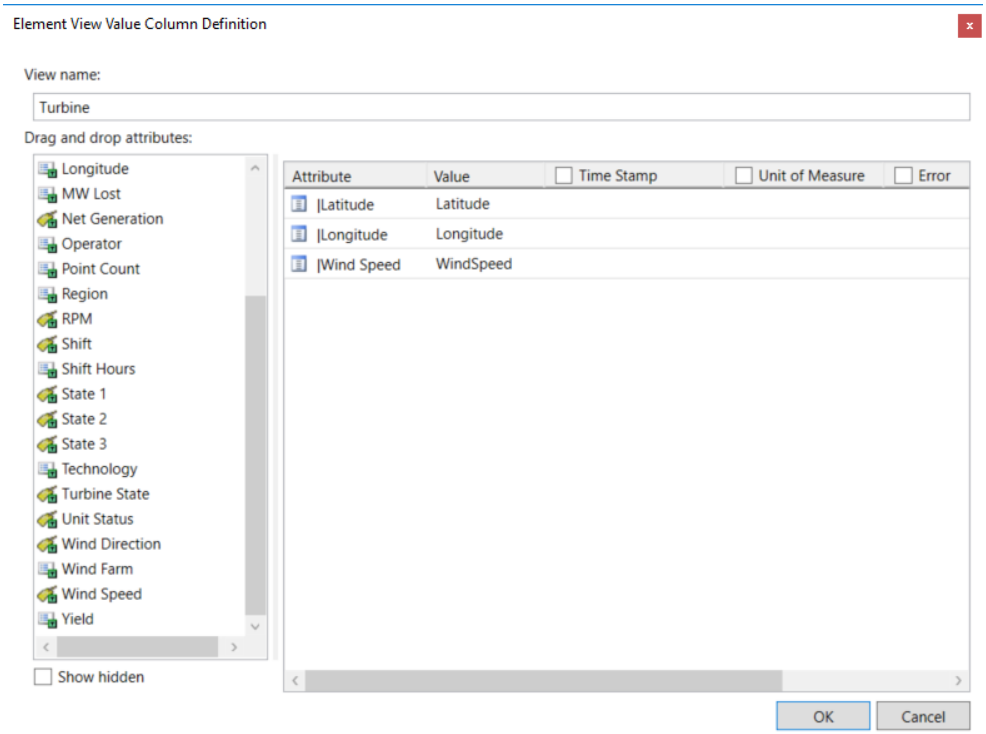
4. In the **Template-Specific Data Model** dialog, select the **Turbine** template and click the **Next** button.



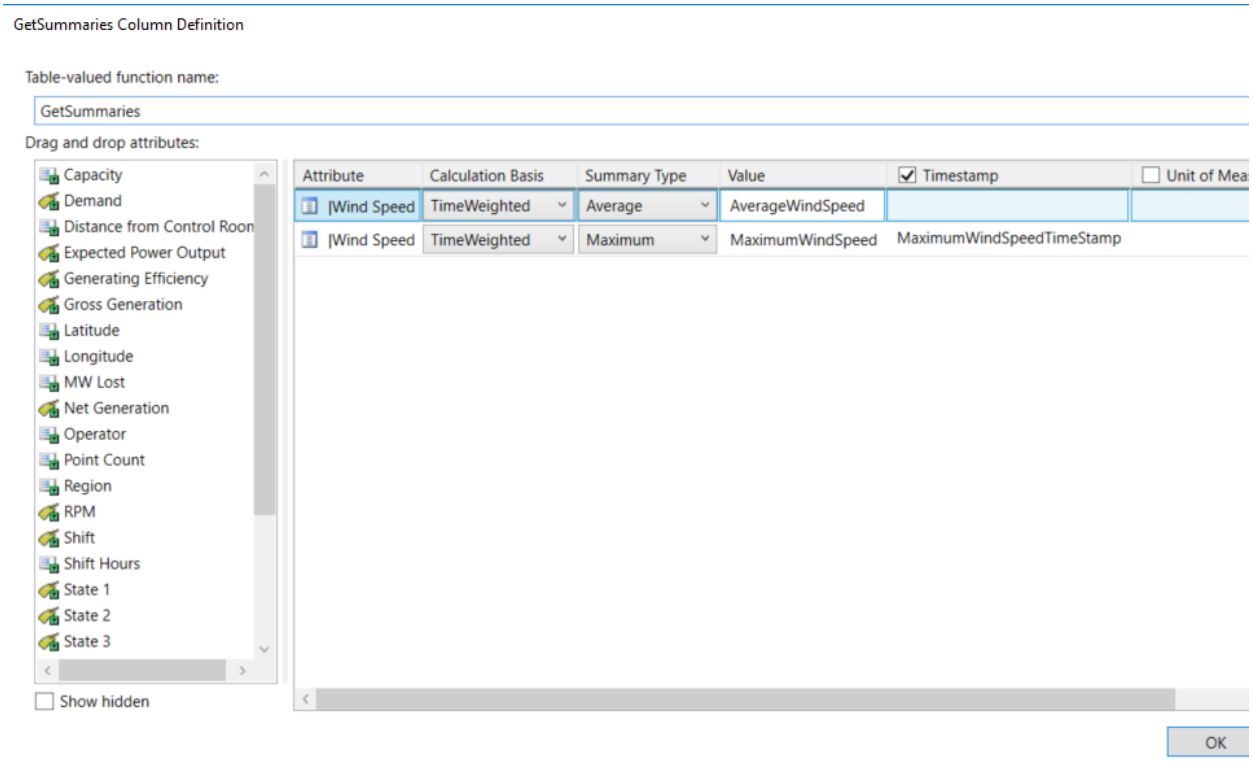
5. Now, you can define objects you want to create to be able to retrieve turbine-specific information from the AF database. Let's start with the **Element View**.



6. The element view for our turbines will represent all turbines (i.e., elements created from the **Turbine** AF element template) with snapshot values of the selected AF attributes.
    a. Drag **Latitude**, **Longitude**, and **Wind Speed** from the left list to the right-side.
    b. To get just the values, uncheck **Time Stamp**, **Unit of Measure**, and **Error** columns.
    c. Confirm by clicking the **OK** button.



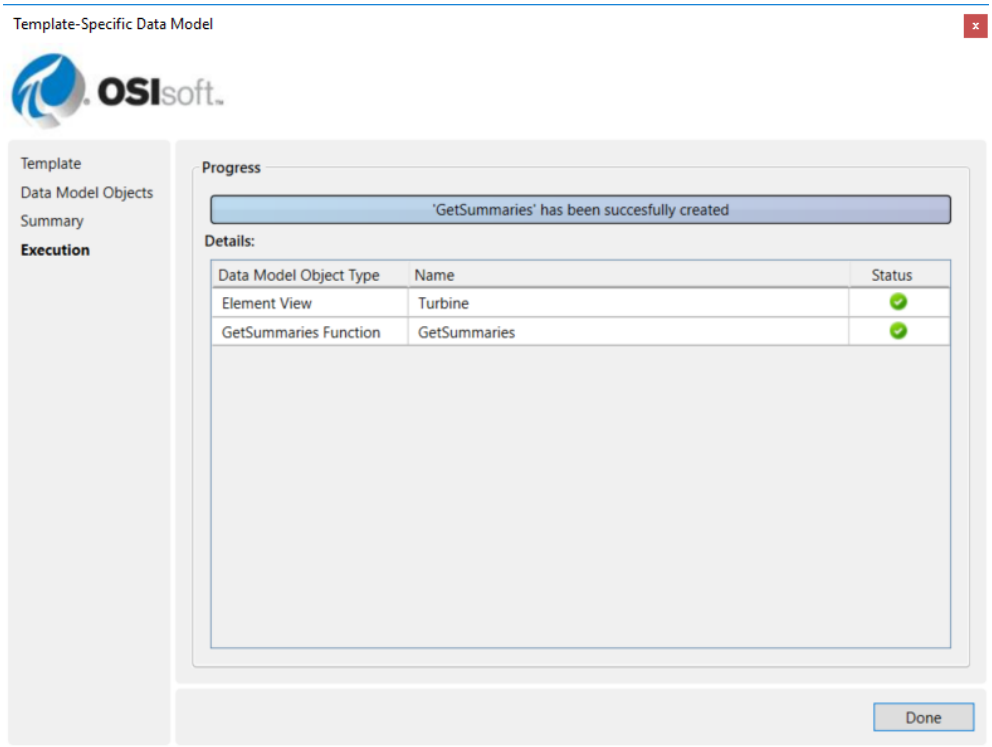
7. Click the **Add GetSummaries…** button to add a TVF to calculate turbine data summaries.
    a. Change the name to just **GetSummaries**.
    b. Drag the **Wind Speed** attribute from the left and select **Average**.
    c. Drag the **Wind Speed** attribute once again and select **Maximum**.
    d. Uncheck **Unit of Measure** and **Error** columns.
    e. If you like, adjust the generated column names.
    f. Confirm by clicking the **OK** button.



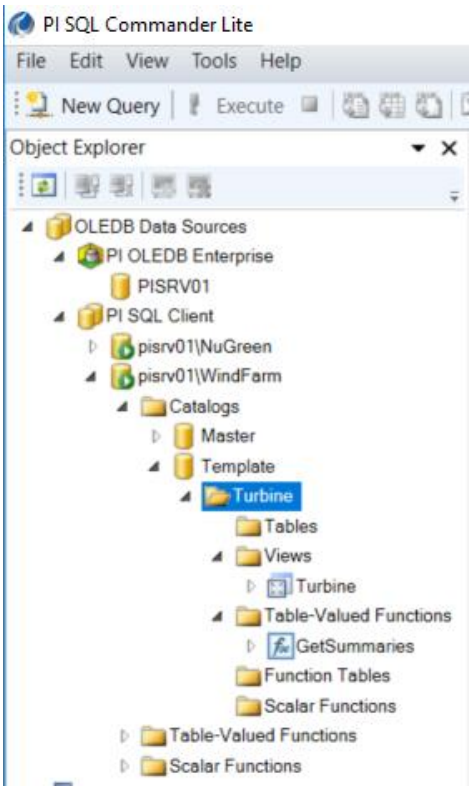
8. Click the **Next** button to continue. Before the objects get created, you can see a short summary with SQL statements PI SQL Commander Lite is about to execute.

   In other words, you can use the wizard not only to create objects, but also to generate SQL which parameterizes TVF and table templates. Then, just copy the TVF and table template instances and paste them into your queries!

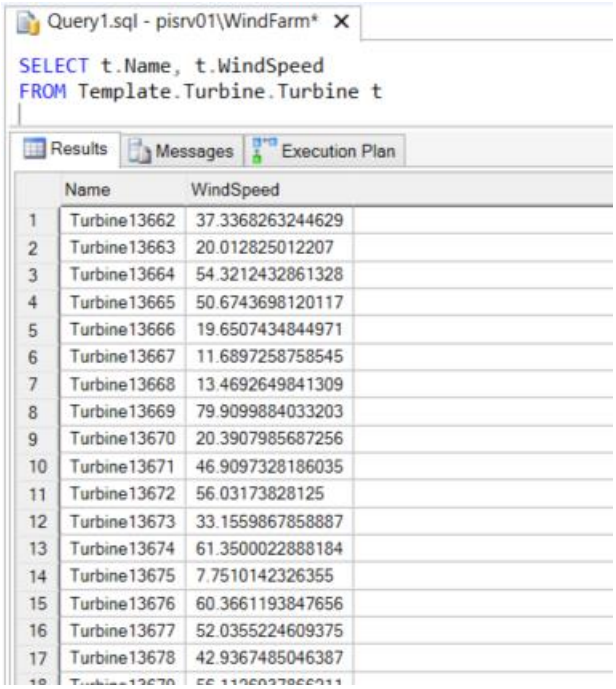
   Now, click the **Execute** button and let PI SQL Commander Lite do its job to create the objects for you.



9. Click the **Done** button and browse the **Turbine** schema. Here is what it now contains.



10. Open a new query window and test the new objects.



	Name	WindSpeed
1	Turbine13662	37.3368263244629
2	Turbine13663	20.012825012207
3	Turbine13664	54.3212432861328
4	Turbine13665	50.6743698120117
5	Turbine13666	19.6507434844971
6	Turbine13667	11.6897258758545
7	Turbine13668	13.4692649841309
8	Turbine13669	79.9099884033203
9	Turbine13670	20.3907985687256
10	Turbine13671	46.9097328186035
11	Turbine13672	56.03173828125
12	Turbine13673	33.1559867858887
13	Turbine13674	61.3500022888184
14	Turbine13675	7.7510142326355
15	Turbine13676	60.3661193847656
16	Turbine13677	52.0355224609375
17	Turbine13678	42.9367485046387

Query1.sql - pisrv01\WindFarm*

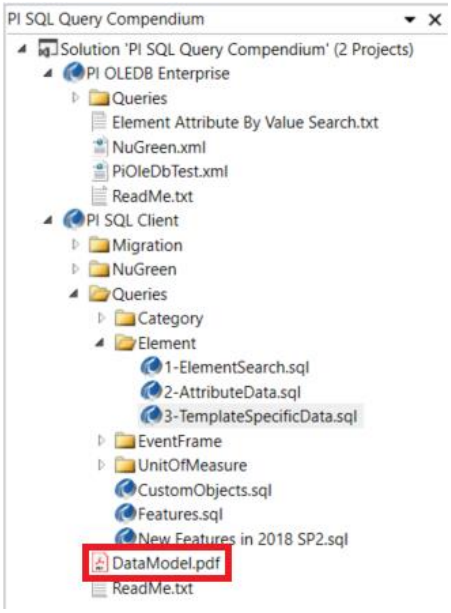
```
SELECT t.Name, s.AverageWindSpeed, s.MaximumWindSpeed
FROM Template.Turbine.Turbine t
CROSS APPLY Template.Turbine.GetSummaries(t.ID, '2018-02-02', '2018-02-02 06:00', '1h', 'MostRecentTime') s
```

Results | Messages | Execution Plan

	Name	AverageWindSpeed	MaximumWindSpeed
1	Turbine03040	24.2057666778564	34.5717544555664
2	Turbine03040	46.1681785583496	57.7646026611328
3	Turbine03040	52.8170204162598	57.7646026611328
4	Turbine03040	35.6763715744019	47.8694381713867
5	Turbine03040	43.8758535385132	64.2684020996094
6	Turbine03040	40.6118507385254	64.2684020996094
7	Turbine04021	65.0739631652832	76.6653137207031
8	Turbine04021	42.3520612716675	53.4826126098633
9	Turbine04021	58.5741147994995	85.9267196655273
10	Turbine04021	88.7024192810059	91.4781188964844
11	Turbine04021	79.8522491455078	91.4781188964844
12	Turbine04021	46.5914945602417	68.2263793945313
13	Turbine05002	40.5617923736572	44.8845443725586
14	Turbine05002	41.7668552398682	47.2946701049805
15	Turbine05002	71.4053230285645	95.5159759521484
16	Turbine05002	65.2245025634766	95.5159759521484
17	Turbine05002	34.4477462768555	34.9330291748047
18	Turbine05002	61.7863960266113	89.6103286743164
19	Turbine05083	75.1101531982422	91.5092926025391
20	Turbine05083	42.8902568817139	58.7110137939453

3.4 Data Model Diagram

To help you with the new data model understanding, the **Query Compendium** also contains an entity-relationship diagram. You will find it in the root compendium folder.



Due to the number of objects in the model, the diagram does not fit one screen and you have to zoom-in the part you are interested in.

4. Build RTQP Engine Queries

4.1 Overview

In this chapter, you will learn how to form queries against the new data model.

4.2 Goals

- Discuss the key query-building techniques.

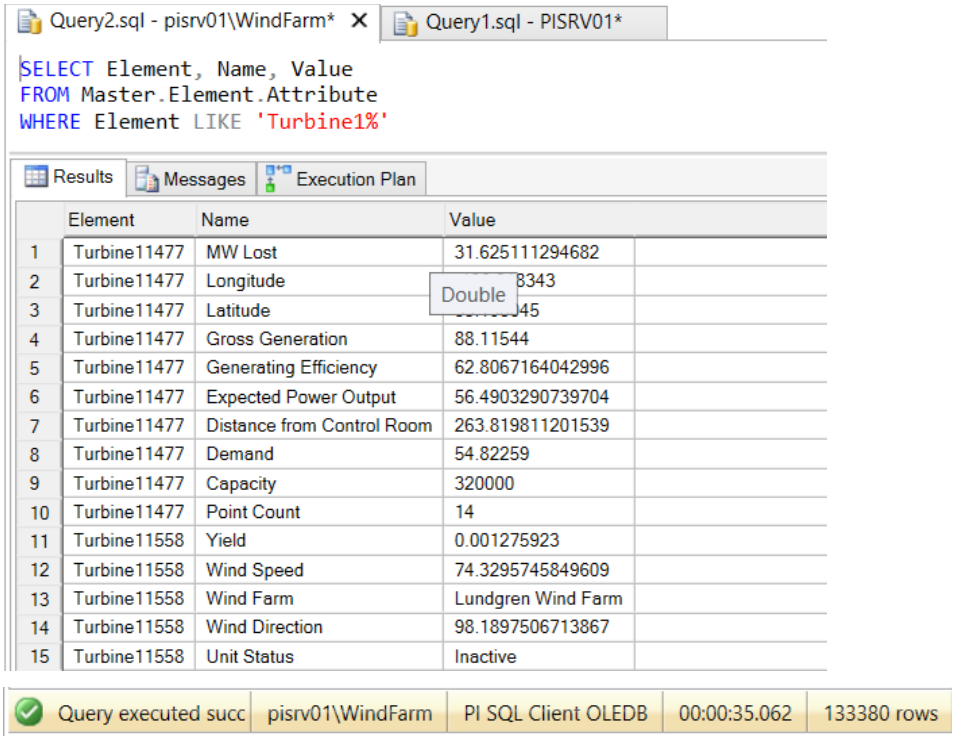
4.3 Request Just the Columns You Need

Requesting just the columns which you need to retrieve is a general rule of thumb in any Relational Database Management System (RDBMS; e.g., SQL Server or Oracle). However, with RTQP Engine, it is not just a rule, it is a must!

RTQP Engine is not a real RDBMS. Its tables are virtual and their data is not stored in a single place. One column in the SELECT list may change the execution plan completely.

Let's have a look at the typical scenario – element attribute table query with and without the snapshot value in the SELECT list.

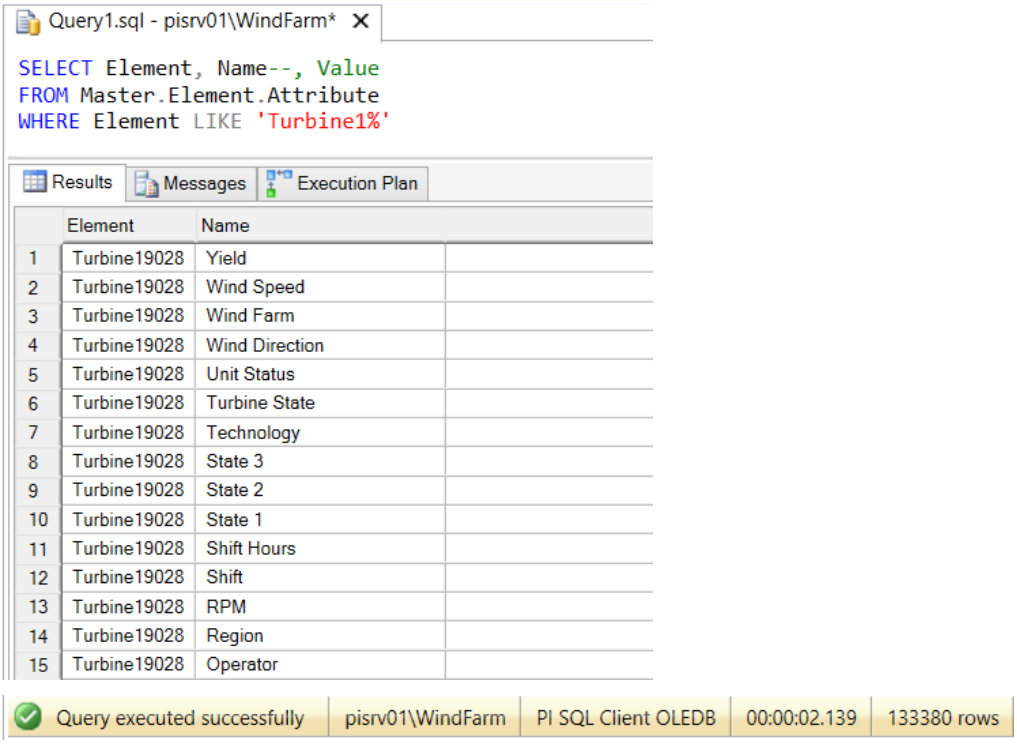
This is the query we executed in the previous workbook part:



	Element	Name	Value
1	Turbine11477	MW Lost	31.625111294682
2	Turbine11477	Longitude	[illegible]8343
3	Turbine11477	Latitude	[illegible]45
4	Turbine11477	Gross Generation	88.11544
5	Turbine11477	Generating Efficiency	62.8067164042996
6	Turbine11477	Expected Power Output	56.4903290739704
7	Turbine11477	Distance from Control Room	263.819811201539
8	Turbine11477	Demand	54.82259
9	Turbine11477	Capacity	320000
10	Turbine11477	Point Count	14
11	Turbine11558	Yield	0.001275923
12	Turbine11558	Wind Speed	74.3295745849609
13	Turbine11558	Wind Farm	Lundgren Wind Farm
14	Turbine11558	Wind Direction	98.1897506713867
15	Turbine11558	Unit Status	Inactive

The execution took **35 seconds.**

Now, let's execute the query once again without the Value column:



	Element	Name
1	Turbine19028	Yield
2	Turbine19028	Wind Speed
3	Turbine19028	Wind Farm
4	Turbine19028	Wind Direction
5	Turbine19028	Unit Status
6	Turbine19028	Turbine State
7	Turbine19028	Technology
8	Turbine19028	State 3
9	Turbine19028	State 2
10	Turbine19028	State 1
11	Turbine19028	Shift Hours
12	Turbine19028	Shift
13	Turbine19028	RPM
14	Turbine19028	Region
15	Turbine19028	Operator

The execution took just **2 seconds**, because RTQP Engine did not have to retrieve the snapshot values.

4.4 Supported SQL Syntax

4.4.1 Supported SELECT Statement Keywords

INNER | LEFT OUTER | RIGHT OUTER | FULL OUTER JOIN

CROSS | OUTER APPLY <table-valued function>(…)

TOP

DISTINCT

WHERE

GROUP BY

HAVING

ORDER BY

UNION [ALL]

Compared to PI OLEDB Enterprise, PI SQL Client supports sub-queries only in the FROM clause.

There are many example queries in the Query Compendium, Queries, and Migration sections. If you happen to finish the lab early, go ahead and test them!

4.4.2 Supported DDL Statements

CREATE | DROP CATALOG

CREATE | DROP SCHEMA

CREATE | ALTER | DROP VIEW

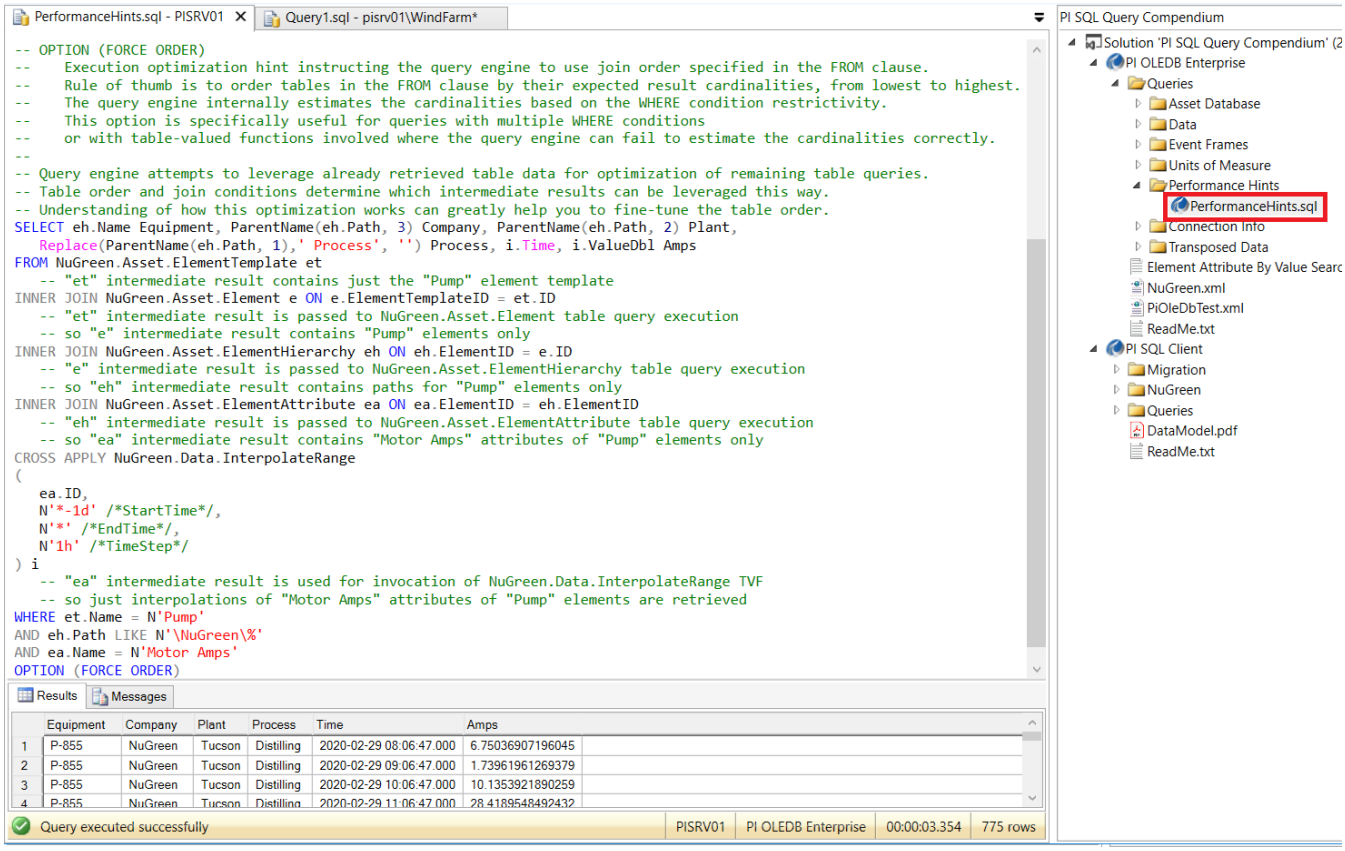
CREATE | ALTER | DROP FUNCTION

CREATE | DROP FUNCTION TABLE

4.5 Forget OPTION (FORCE ORDER)

The PI OLEDB Enterprise query engine in many cases fails to determine the best order of data retrieval. Thus, the driver supports OPTION (FORCE ORDER) clause which allows the query author to order the tables in the FROM clause explicitly. PI OLEDB Enterprise then retrieves the data in this order.

The behavior is described in detail in the PI OLEDB Enterprise **Query Compendium**, **PerformanceHints.sql**.



The RTQP Engine does not support **OPTION (FORCE ORDER)**. The new query engine makes use of a new cost-based optimizer, which minimizes the number of cases where the data retrieval order is suboptimal.

5. Migrate a Custom Application

5.1 Overview

In this chapter, you will learn how to migrate a custom application based on PI OLEDB Enterprise to PI SQL Client. For the purpose of this exercise, there is a simple console C# application that lists the wind farms according to percentage of active turbines. The same principles can be used in any more complex application or an application written in a different programming language.

5.2 Goals

- Understand the differences between PI OLEDB Enterprise and PI SQL Client OLEDB from a programming point of view.
- Learn how to create an OLE DB provider connection string.
- Migrate a simple console C# application from PI OLEDB Enterprise to PI SQL Client OLEDB.

5.3 Step-by-step Instructions

1. Launch the Microsoft Visual Studio 2019.

2. Open the **ActiveTurbines** solution from `C:\Users\student01.PISCHOOL\source\repos\ActiveTurbines\ActiveTurbines.sln`.
3. Open **Program.cs** by double-clicking it in the Solution Explorer on the right-hand side. Get familiar with the code.

   There are several points you need to understand:

   a. Query is executed by the driver specified in the connection string (**`Provider=PIOLEDBENT.1`** means that PI OLEDB Enterprise handles the execution).

   b. SQL query is created in the **`CreateCommandText`** method. **`{timestamp.ToString("yyyy-MM-dd HH:mm:ss.fffffff")}`** is a placeholder that gets replaced by the actual timestamp value formatted according to the specified format.

   c. **`reader.Read()`** advances the reading cursor to the next row in the result.

   d. **`reader.GetXYZ(index)`** reads the value of the column specified by the zero-based index in the current row. XYZ must correspond to the type of the value stored in the column. There must be an exact type

match, e.g., you cannot use **`GetInt64`** for 32-bit integer value even though 64-bit integer can store any 32-bit integer.

4. Execute the program by pressing **`Ctrl+F5`** and look at the results.

   Alternatively, you may specify a breakpoint anywhere in the code by pressing **`F9`** and start the program in debug mode by pressing **`F5`**.

   When the program hits the breakpoint, you may continue with the debugging by pressing:

   - **`F10`** – execute the statement at cursor and move to the next one.
   - **`F11`** – step inside the statement.
   - **`F5`** – continue with the execution until the next breakpoint is hit or programs ends.

5. Try to change the program in a way that it creates the same result, but uses PI SQL Client OLEDB underneath. You need to change the following things:

   a. Modify the connection string so that it connects to PI SQL Client OLEDB (see the **Create connection string** section below).

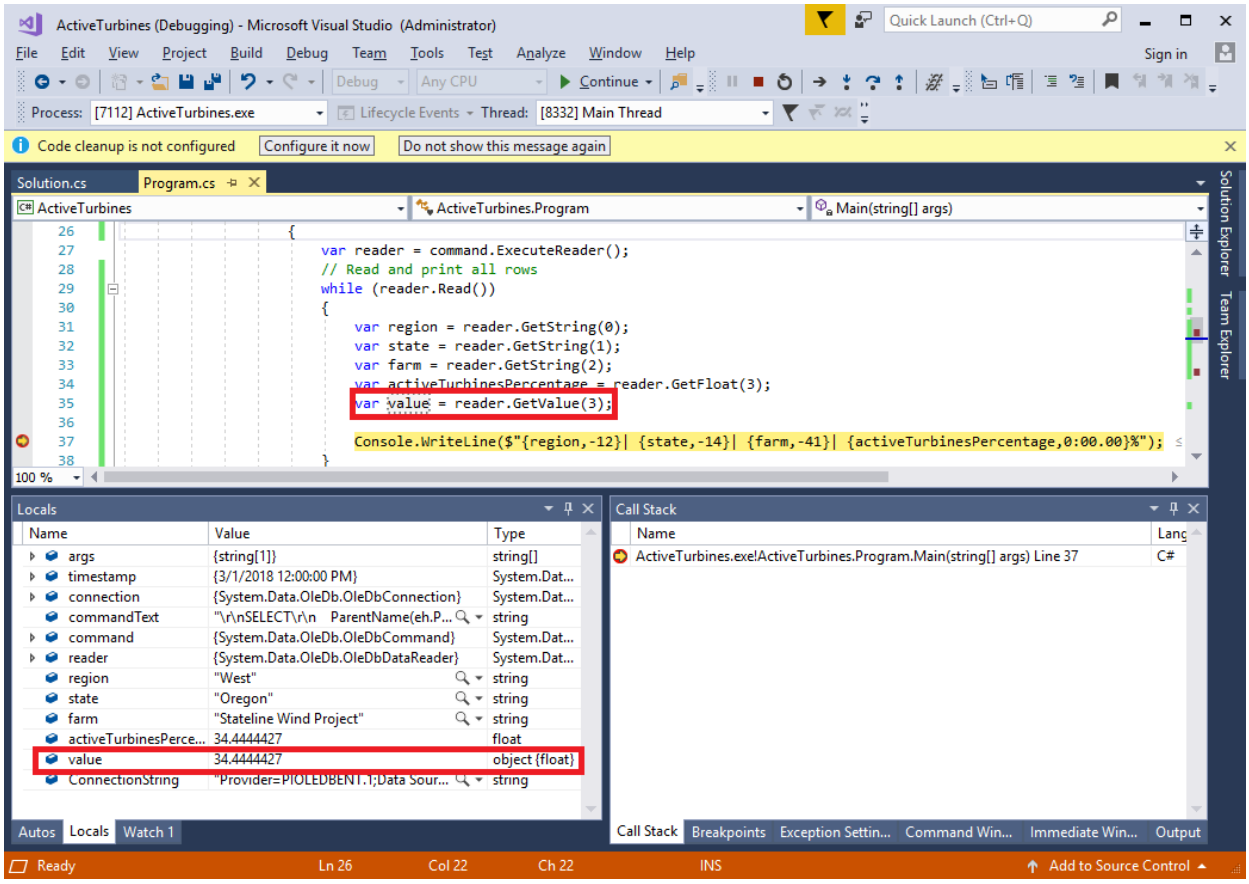
   b. Modify the SQL query – use the information learned in the previous parts of the lab to re-write the SQL query.

Tip	You can use PI SQL Commander Lite to tune the query before using it in the code.

Tip	Curly brackets and quotes must be escaped in the code by doubling them.

   c. Make sure, you are reading the value using the correct type.

Tip	You can use **`GetValue(index)`**, which can read the value of any type into an object, you can then inspect the local value for the actual type.

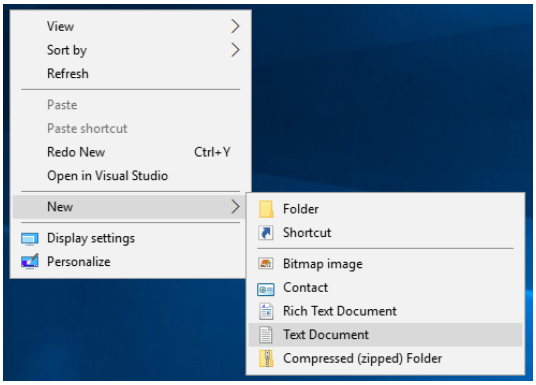


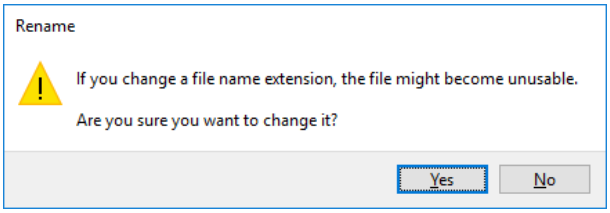
Tip	All constant real numbers in PI SQL Client OLEDB are of type double by default.

5.4 Create Connection String

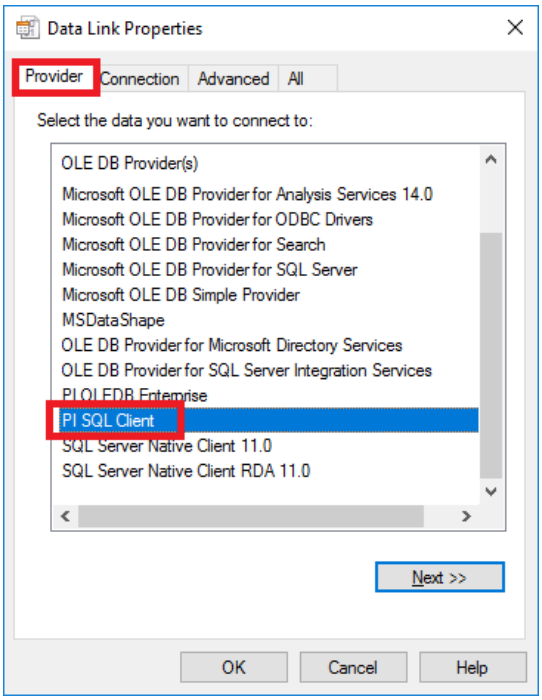
When creating connection strings, it is usually necessary to consult the provider documentation to understand the connection string keys. However, you can generate the OLEDB provider connection string using UI.

1. Create a new text file with an **.udl** extension.

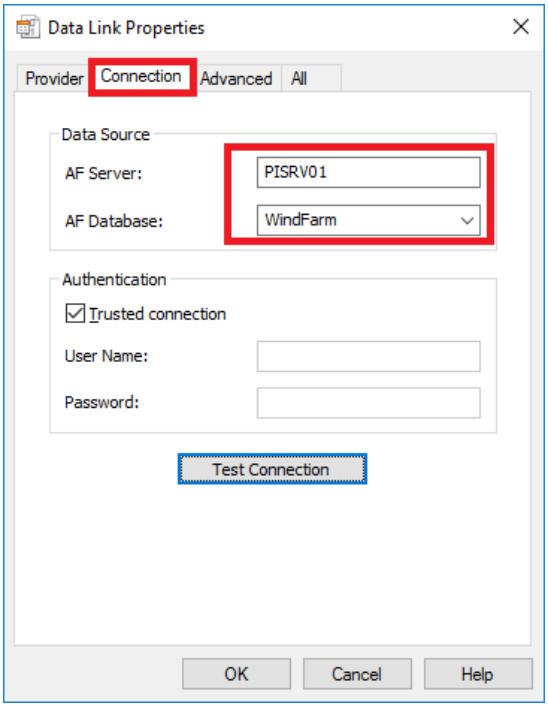




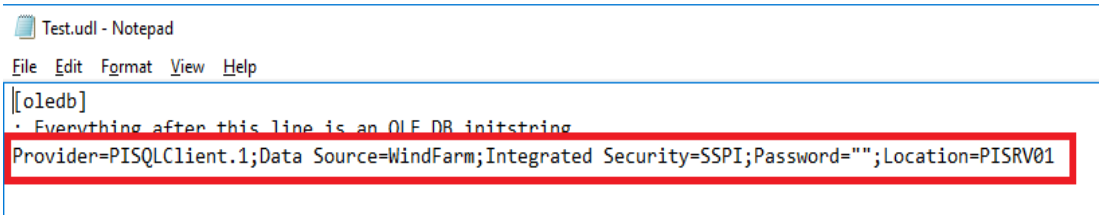
2. Open the **Data Link Properties** window by double-clicking the created file. Navigate to the **Provider** tab and select **PI SQL Client** provider.



3. Specify the data source as well as authentication information on the **Connection** tab.



4. You may configure additional options on the **Advanced** or **All** tabs. When done, click **OK**.
5. Right-click the **Test.udl** file and select the **Open with-Notepad** menu item.
6. You can now see the connection string, which you can use in your application.



5.5 Conclusion

You have learned how to migrate a simple application based on PI OLEDB Enterprise. The same principles can be applied to more complex projects. You have also learned how to create a PI SQL Client OLEDB connection string without using the documentation. Similarly, you can create a PI SQL Client ODBC connection string using File DSN in ODBC Data Source Administrator.

6. Integrate PI SQL Client with Microsoft SQL Server Reporting Services

6.1 Overview

In this chapter, you will learn how to integrate PI SQL Client with MS SQL Server Reporting Services (SSRS) and how to create a drill-down report.

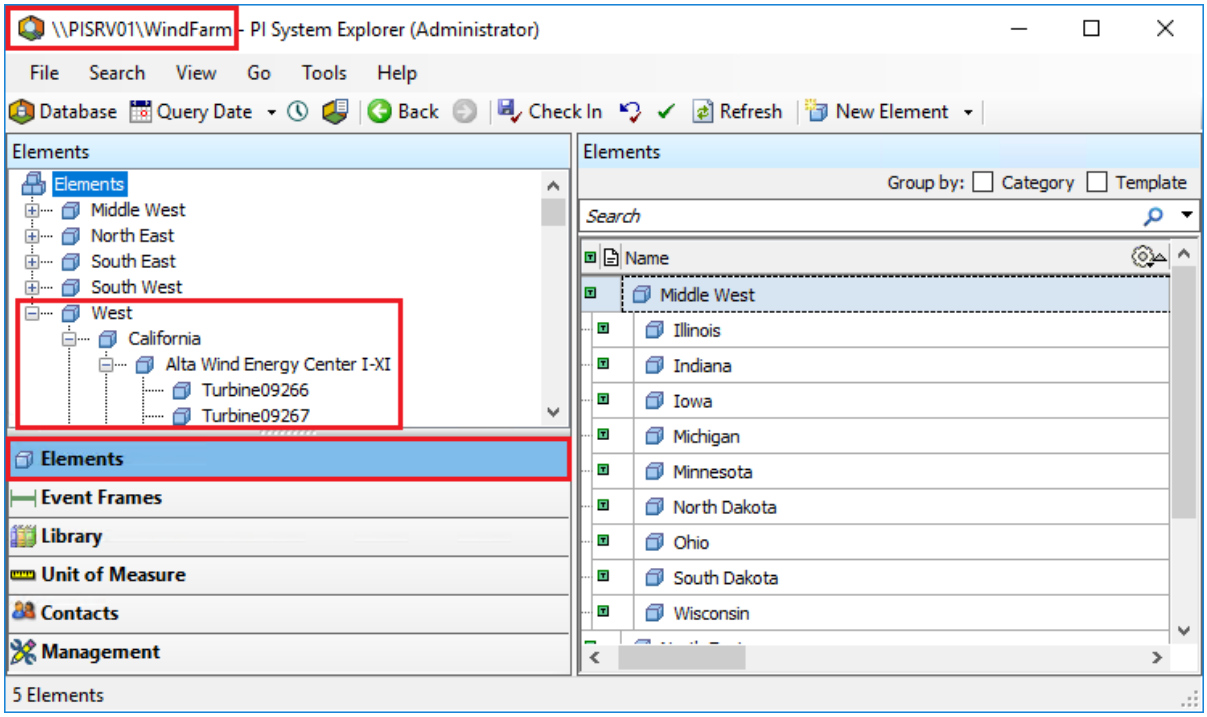
6.1.1 Goals

- Create a query for the report
    - Create a query which retrieves summary data for the West US region in the period between 2018-02-02 and 2018-02-14. The summaries should be organized by region, state and wind farm names.
      The summary data should contain time-weighted averages of wind speed and power generation for each wind farm in the region. The averages should only be calculated for very windy time periods.
- Create a query with a parameter for the drill-down report
    - Modify the query to accept a parameter used for filtering the wind farm by name
- Configure SQL Server Reporting Services to utilize the PI SQL Client query
- Create a report using a wizard
- Publish the report
- View the report

6.2 Step-by-Step Instructions

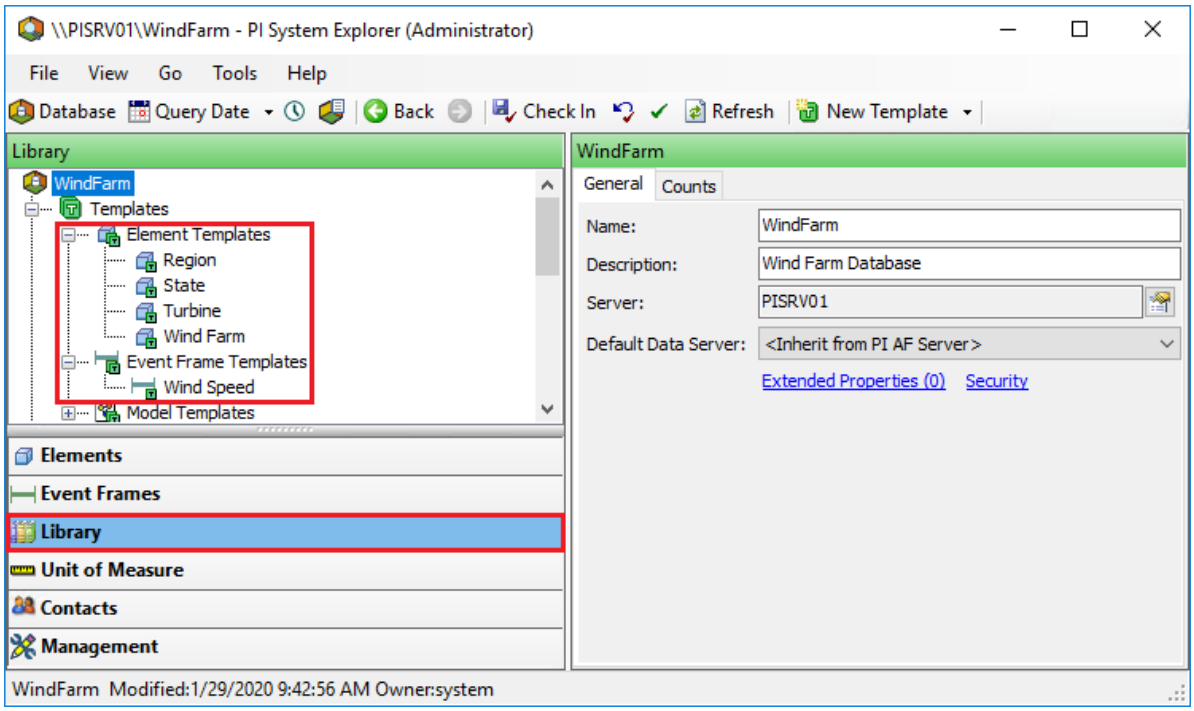
1. Open **PI System Explorer** to learn the structure of the **WindFarm** AF database.

2. Make sure you are connected to [\\PISRV01\WindFarm](\\PISRV01\WindFarm) AF database and explore the structure of AF elements.

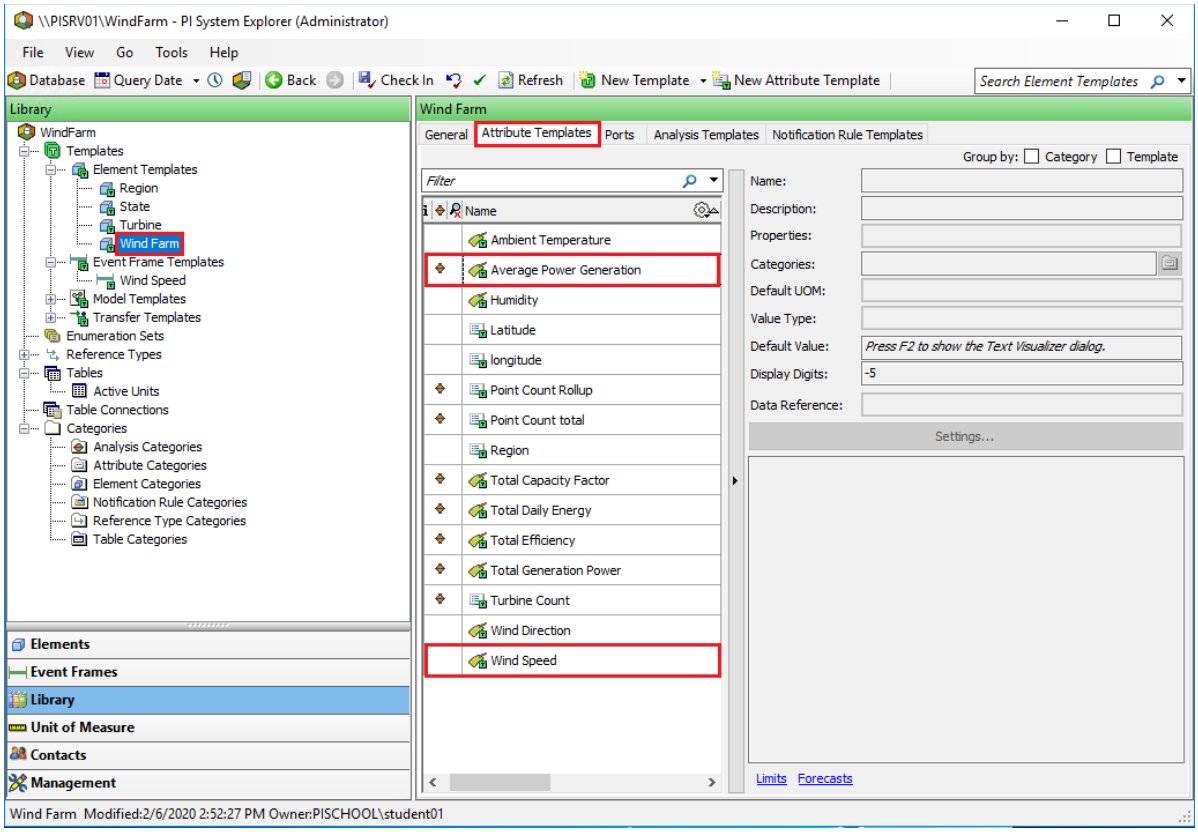


Tip	Notice how the element hierarchy is organized by region, state, and wind farm.

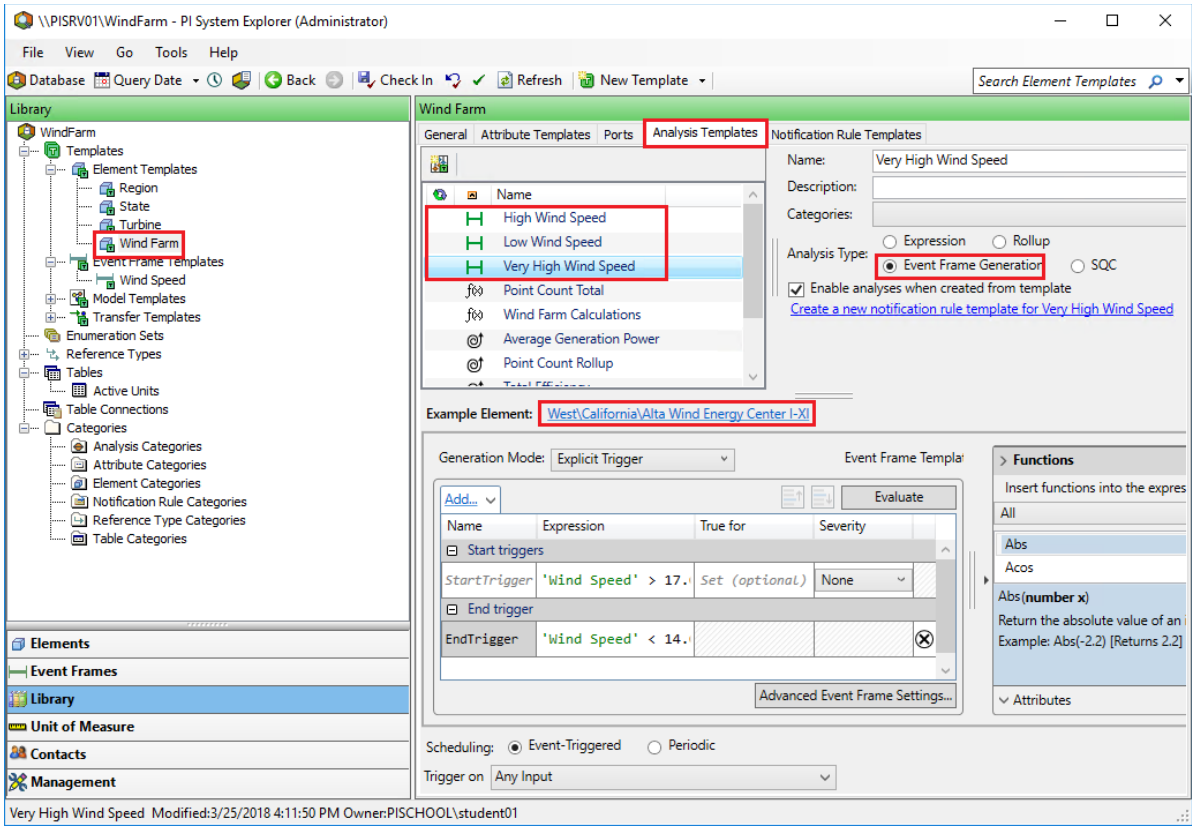
3. Switch to the library to learn what templates the elements are based on.



4. Select the **Wind Farm** template to explore the attribute templates. You may recognize that it contains two attributes, which are useful for your query: the **Average Power Generation** attribute which gathers the average power generation across all turbines in the farm and the **Wind Speed** attribute.



5. Next, switch to **Analysis Templates** to explore the analyses. In that screen, you may recognize some analyses, which trigger **Event Frame** generation. Your query should return averages only during very windy periods, which are captured in the **Very High Wind Speed** event frames.



6. Now, since you are familiar with the basic structure of the AF database, open **PI SQL Commander Lite**.

7. Use PI SQL Client OLEDB to connect to **PISRV01\WindFarm** and open a new query editor by clicking the **New Query** button.

8. First, you need to retrieve all wind farms located in the **West** region.

```
SELECT e.PrimaryPath Location, e.Name WindFarm
FROM Master.Element.Element e
WHERE e.PrimaryPath LIKE '\West%' AND e.Template = 'Wind Farm'
```

Note:
Unlike PI OLEDB Enterprise, you do not need to add a join with the ElementHierarchy table to get the path of the element and you do not need to add a join with the ElementTemplate table to create the template restriction.
An equivalent query in PI OLEDB Enterprise would look as follows:

```
SELECT eh.Path Location, eh.Name WindFarm
FROM WindFarm.Asset.Element e
INNER JOIN WindFarm.Asset.ElementHierarchy eh ON eh.ElementID = e.ID
INNER JOIN WindFarm.Asset.ElementTemplate et ON et.ID = e.ElementTemplateID
WHERE et.Name = 'Wind Farm' AND eh.Path LIKE '\West%'
```

9. Split the **Location** column into **Region** and **State** columns to make the data a bit more readable. You can use the **ParentName** function.

```
SELECT ParentName(e.PrimaryPath, 1) Region, ParentName(e.PrimaryPath, 0) State,
  e.Name WindFarm
FROM Master.Element.Element e
WHERE e.PrimaryPath LIKE '\West%' AND e.Template = 'Wind Farm'
```

10. Next, you are only interested in wind farms with **Very High Wind Speed** event occurrences in the period between 2nd of February 2018 and 14th of February 2018.

```
SELECT ParentName(e.PrimaryPath, 1) Region, ParentName(e.PrimaryPath, 0) State,
  e.Name WindFarm, ef.Duration, ef.EndTime Time
FROM Master.Element.Element e
INNER JOIN Master.EventFrame.EventFrame ef ON e.ID = ef.PrimaryReferencedElementID
WHERE e.PrimaryPath LIKE '\West%'
AND e.Template = 'Wind Farm'
AND ef.Name LIKE 'Very High Wind Speed%'
AND ef.StartTime >'2018-02-02'
AND ef.EndTime < '2018-02-14'
```

11. Using the template-based **GetSummary** table-valued function template, add the time-weighted averages for the **Wind Speed** and **Average Power Generation** attributes.

```
SELECT ParentName(e.PrimaryPath, 1) Region, ParentName(e.PrimaryPath, 0) State, e.Name
WindFarm, ef.Duration, ef.EndTime Time, ef.Name, s.*
FROM Master.Element.Element e
INNER JOIN Master.EventFrame.EventFrame ef ON e.ID = ef.PrimaryReferencedElementID
CROSS APPLY [Master].[Element].[GetSummary]
<
  'Wind Farm',
  {
    '|Wind Speed',
    'Average',
    'TimeWeighted',
    'Wind Speed_Average',
```

```
        'Wind Speed_Average_UOM'
    },
    {
        '|Average Power Generation',
        'Average',
        'TimeWeighted',
        'Power Generation_Average',
        'Power Generation_Average_UOM'
    }
)
(e.ID, ef.StartTime, ef.EndTime) s
WHERE
e.PrimaryPath LIKE '\West%'
AND e.Template = 'Wind Farm'
AND ef.Name LIKE 'Very High Wind Speed%'
AND ef.StartTime > '2018-02-02'
AND ef.EndTime < '2018-02-14'
```

12. Execute the query and check the results.

Results | Messages | Execution Plan

	Region	State	WindFarm	Duration	Time	Name	Wind Speed_Average	Wind	Power Generation_A	Pow
1	West	Oregon	Klondike Wind Farm	11:00:00	2018-02-02 14:00:00.000	Very High Wind Speed 2018-02-02 03:00:00.000	50.62920813127	mi/h	44.7966920886173	MW
2	West	Oregon	Klondike Wind Farm	05:00:00	2018-02-03 22:00:00.000	Very High Wind Speed 2018-02-03 17:00:00.000	45.5723289012909	mi/h	46.9007767163686	MW
3	West	Oregon	Klondike Wind Farm	12:00:00	2018-02-03 16:00:00.000	Very High Wind Speed 2018-02-03 04:00:00.000	71.2140558858712	mi/h	49.8810413234779	MW
4	West	Oregon	Klondike Wind Farm	09:00:00	2018-02-03 03:00:00.000	Very High Wind Speed 2018-02-02 18:00:00.000	40.9908265802595	mi/h	29.2498479759346	MW
5	West	Oregon	Klondike Wind Farm	01:00:00	2018-02-02 16:00:00.000	Very High Wind Speed 2018-02-02 15:00:00.000	15.3907051086426	mi/h	88.4991065545469	MW
6	West	Oregon	Klondike Wind Farm	09:00:00	2018-02-05 05:00:00.000	Very High Wind Speed 2018-02-04 20:00:00.000	46.7566084795528	mi/h	62.1325257264902	MW
7	West	Washington	Wild Horse Wind Farm	09:00:00	2018-02-02 15:00:00.000	Very High Wind Speed 2018-02-02 06:00:00.000	49.4674869643317	mi/h	58.102200892905	MW
8	West	Washington	Wild Horse Wind Farm	01:00:00	2018-02-02 05:00:00.000	Very High Wind Speed 2018-02-02 04:00:00.000	21.4417653083801	mi/h	74.4658442328059	MW
9	West	Washington	Wild Horse Wind Farm	01:00:00	2018-02-02 03:00:00.000	Very High Wind Speed 2018-02-02 02:00:00.000	12.0625504218042	mi/h	54.5799992767069	MW
10	West	Washington	Wild Horse Wind Farm	1.09:00:00	2018-02-04 05:00:00.000	Very High Wind Speed 2018-02-02 20:00:00.000	50.3530042966207	mi/h	49.4644577944076	MW
11	West	Washington	Wild Horse Wind Farm	03:00:00	2018-02-02 19:00:00.000	Very High Wind Speed 2018-02-02 16:00:00.000	49.8954265912374	mi/h	51.6263397806136	MW
12	West	Washington	Wild Horse Wind Farm	1.05:00:00	2018-02-05 15:00:00.000	Very High Wind Speed 2018-02-04 10:00:00.000	48.2785037632646	mi/h	40.3139940546361	MW
13	West	Washington	Wild Horse Wind Farm	03:00:00	2018-02-04 09:00:00.000	Very High Wind Speed 2018-02-04 06:00:00.000	41.5760163466136	mi/h	59.3405787333259	MW
14	West	Washington	Wild Horse Wind Farm	05:00:00	2018-02-06 15:00:00.000	Very High Wind Speed 2018-02-06 10:00:00.000	60.4393146514893	mi/h	42.8421984486478	MW
15	West	Washington	Wild Horse Wind Farm	02:00:00	2018-02-06 09:00:00.000	Very High Wind Speed 2018-02-06 07:00:00.000	34.3104911595583	mi/h	31.8237222413643	MW
16	West	Washington	Wild Horse Wind Farm	04:00:00	2018-02-06 06:00:00.000	Very High Wind Speed 2018-02-06 02:00:00.000	43.4661405086517	mi/h	70.2919700743128	MW
17	West	Washington	Wild Horse Wind Farm	02:00:00	2018-02-06 01:00:00.000	Very High Wind Speed 2018-02-05 23:00:00.000	34.3600410223007	mi/h	74.8695910325598	MW

Tip

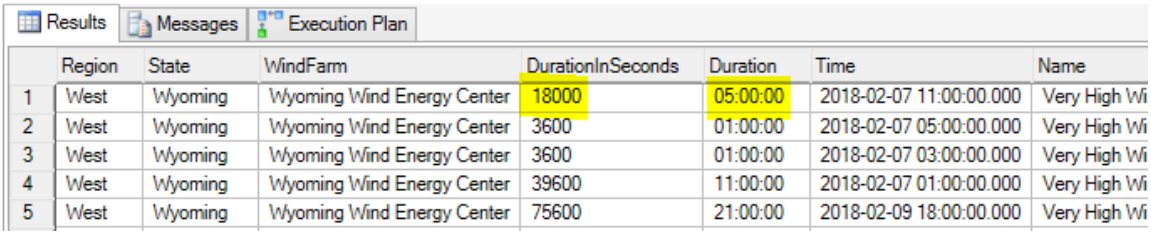
The **Duration** column is of type **TimeSpan**, which represents the time interval. Many tools run into issues if the time span is greater than or equal to 1.00:00:00 (one day). Some tools that do not support these values will run into an error (e.g., SQL Server Linked Server) or will display incorrect data (e.g., Power BI).

In case you only want to display the values, you may convert **TimeSpan** columns to **String**.

Alternatively, RTQP Engine supports functions to extract the individual time parts (**Day**, **Hour**, **Second**) and an explicit conversion to **Double** (floating-point number).

13. To allow SQL Server Reporting Services to aggregate the **Duration** column values, modify the SELECT list to retrieve **Duration** in seconds as **Double** type.

```
SELECT ParentName(e.PrimaryPath, 1) Region, ParentName(e.PrimaryPath, 0) State,
  e.Name WindFarm, Double(ef.Duration, second) DurationInSeconds, ef.Duration,
  ef.EndTime Time, ef.Name, s.*
FROM Master.Element.Element e…
```



Results | Messages | Execution Plan

	Region	State	WindFarm	DurationInSeconds	Duration	Time	Name
1	West	Wyoming	Wyoming Wind Energy Center	18000	05:00:00	2018-02-07 11:00:00.000	Very High Wi
2	West	Wyoming	Wyoming Wind Energy Center	3600	01:00:00	2018-02-07 05:00:00.000	Very High Wi
3	West	Wyoming	Wyoming Wind Energy Center	3600	01:00:00	2018-02-07 03:00:00.000	Very High Wi
4	West	Wyoming	Wyoming Wind Energy Center	39600	11:00:00	2018-02-07 01:00:00.000	Very High Wi
5	West	Wyoming	Wyoming Wind Energy Center	75600	21:00:00	2018-02-09 18:00:00.000	Very High Wi

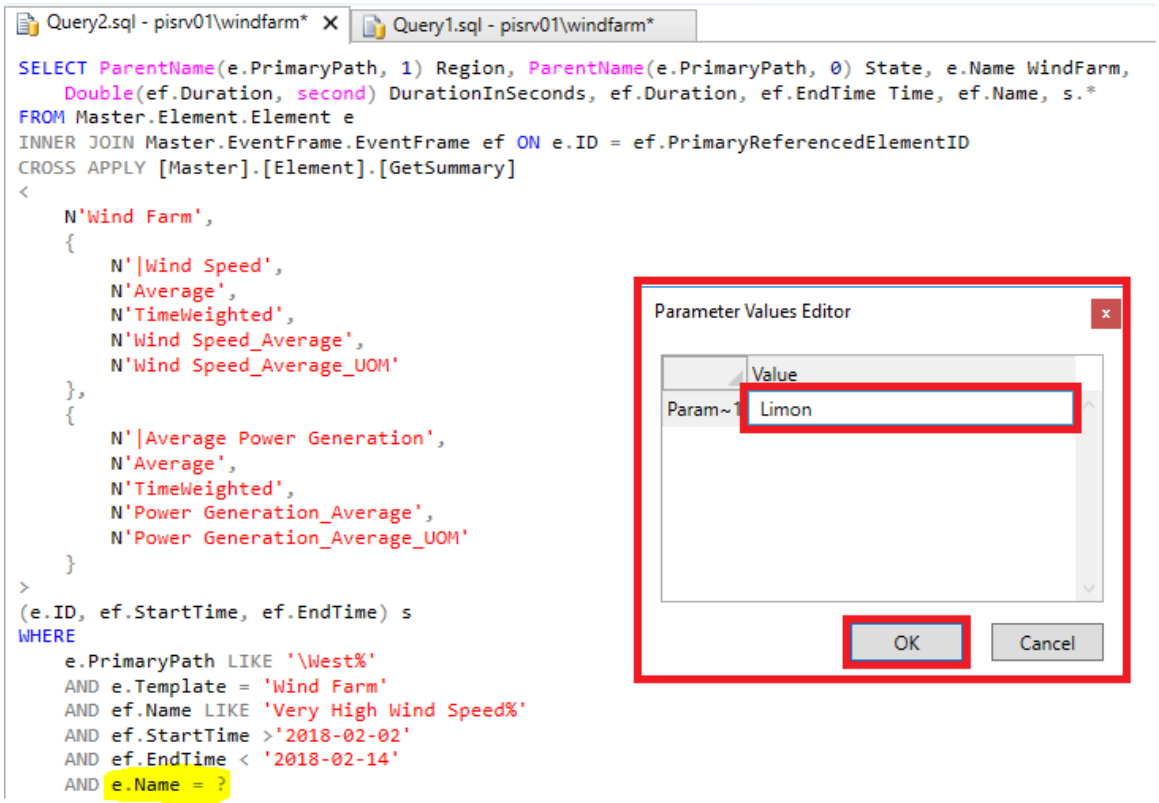
14. Next, add a parameter to filter the results by the wind farm name. To accomplish that, copy the previous query to a new query tab (**CTRL+N**) and add the parameter restriction to the WHERE clause.

```
SELECT ParentName(e.PrimaryPath, 1) Region, ParentName(e.PrimaryPath, 0) State,
  e.Name WindFarm, Double(ef.Duration, second) DurationInSeconds, ef.Duration,
  ef.EndTime Time, ef.Name, s.*
FROM Master.Element.Element e
INNER JOIN Master.EventFrame.EventFrame ef ON e.ID = ef.PrimaryReferencedElementID
CROSS APPLY [Master].[Element].[GetSummary]
<
  'Wind Farm',
  {
    '|Wind Speed',
    'Average',
    'TimeWeighted',
    'Wind Speed_Average',
    'Wind Speed_Average_UOM'
  },
  {
    '|Average Power Generation',
    'Average',
    'TimeWeighted',
    'Power Generation_Average',
    'Power Generation_Average_UOM'
  }
>
(e.ID, ef.StartTime, ef.EndTime) s
WHERE
e.PrimaryPath LIKE '\West%'
AND e.Template = 'Wind Farm'
AND ef.Name LIKE 'Very High Wind Speed%'
AND ef.StartTime >'2018-02-02'
```

```
AND ef.EndTime < '2018-02-14'
AND e.Name = ?
```

Tip	The RTQP Engine does not support binding parameters using name (named parameters), for example e.Name = @WindFarm. If you want to prioritize development of this feature, you can add a request to https://feedback.osisoft.com/forums/555145-pi-developer-technologies/

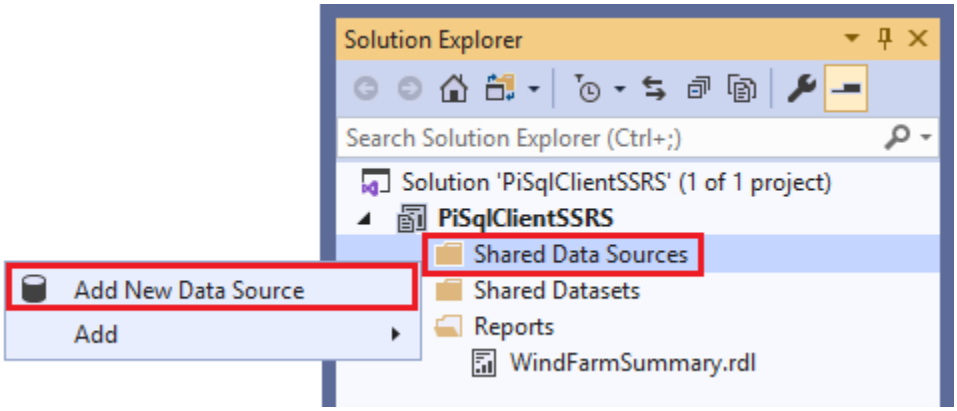
15. To test the query, press **F5** and provide a sample value for your parameter in the opened **Parameter Values Editor.**



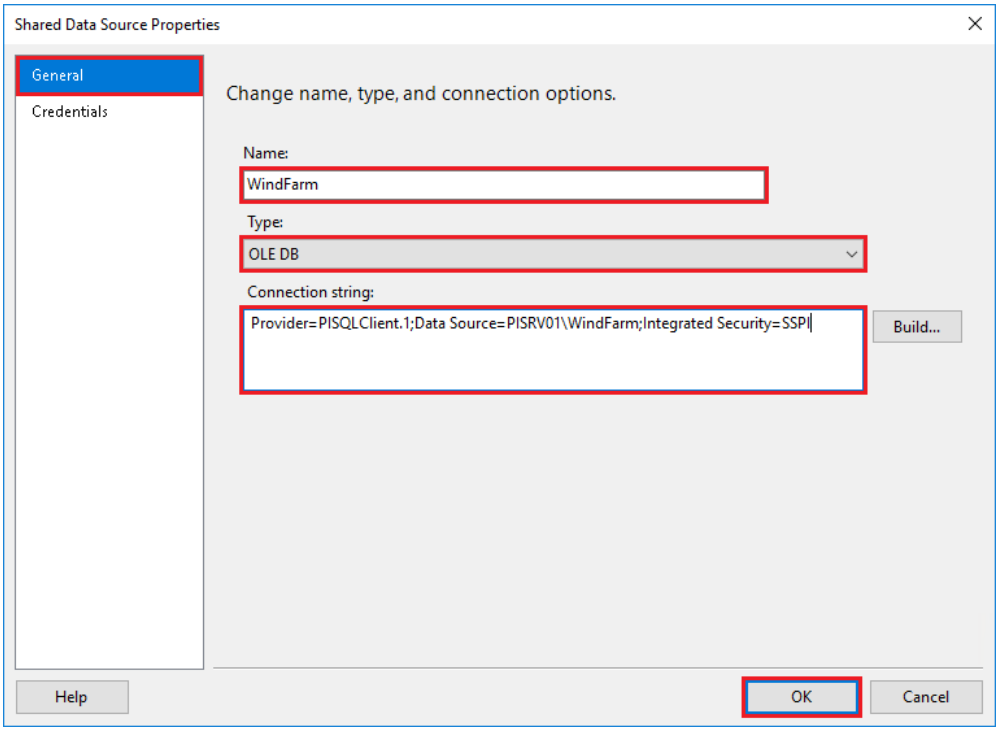
16. Start **Visual Studio** from the Windows taskbar.

17. Open the sample Reporting Services solution from `C:\Users\student01.PISCHOOL\source\repos\PiSqlClientSSRS\PISQLClientSSRS.sln`

18. In the **Solution Explorer** (**Ctrl + Alt + L**), you may notice that the solution contains just one item – WindFarmSummary.rdl. The WindFarmSummary.rdl is a sample report, which you will populate with data. In order to do that, we need to create a **Data Source.** Right-click the **Shared Data Sources** item and select the **Add New Data Source** option.

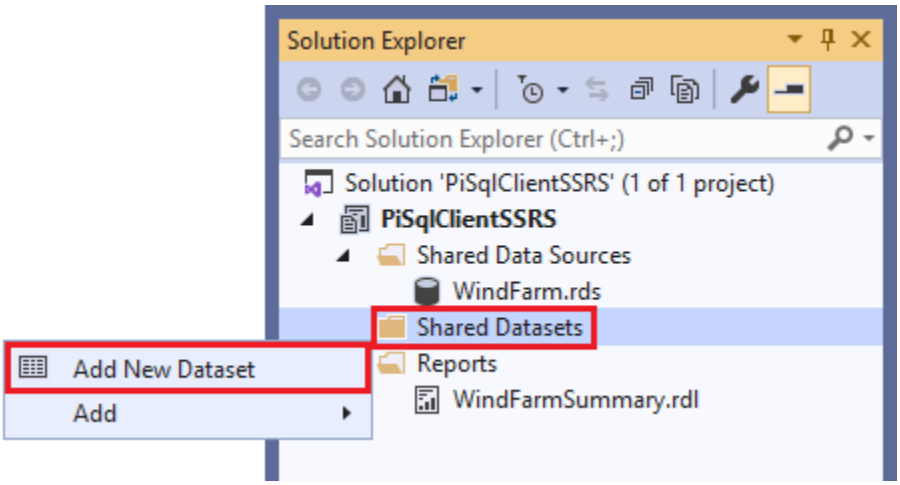


19. In the **Shared Data Source Properties** wizard, enter the connection name **WindFarm**. Change the **Type** to **OLE DB** and set the **Connection string** to **Provider=PISQLClient.1;Data Source=PISRV01\WindFarm;Integrated Security=SSPI**. Press the **OK** button.

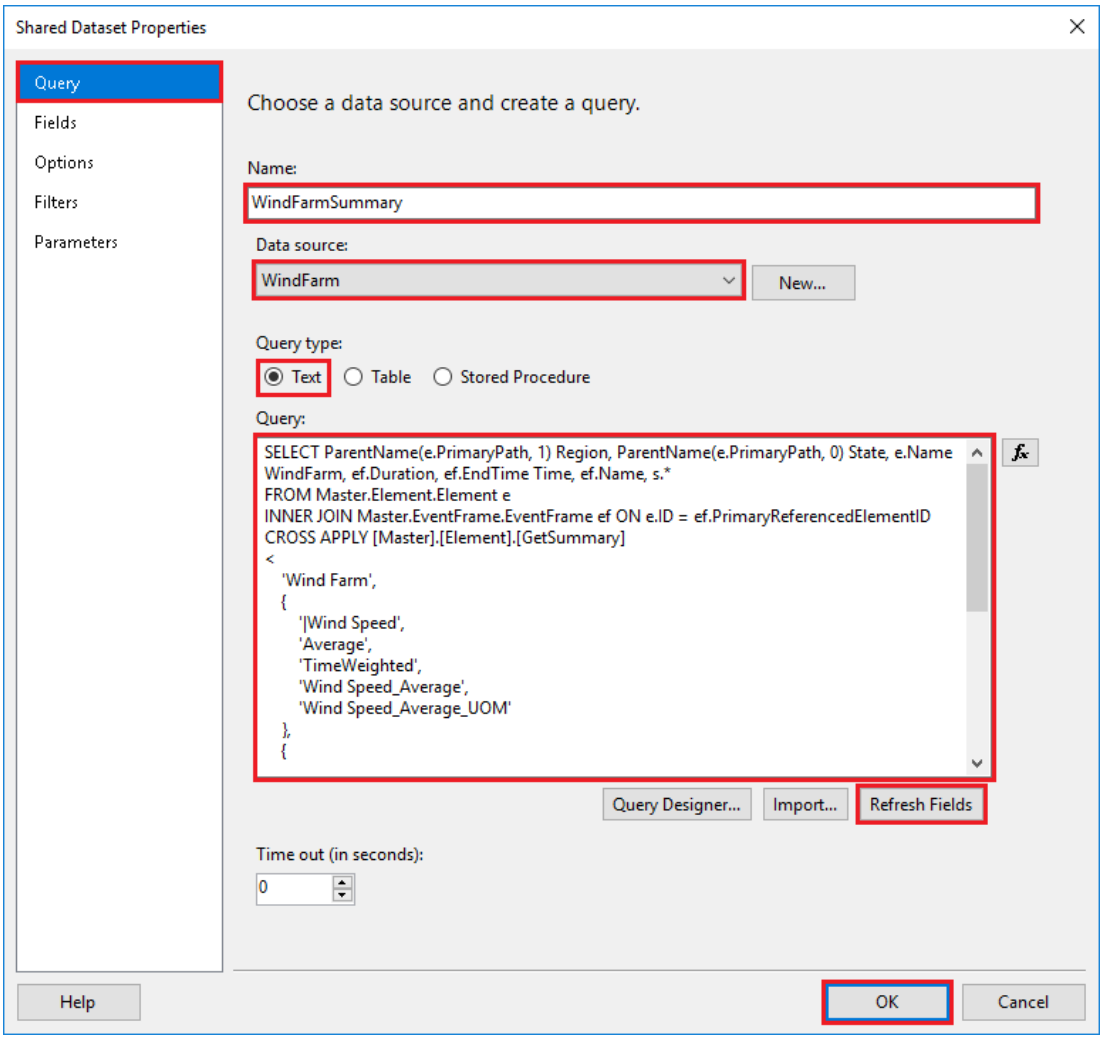


Tip	Alternatively, you could change the **Type** to **ODBC** and set the **Connection string** to **DRIVER=PI SQL Client;Integrated Security=SSPI;AF Server=PISRV01;AF Database=WindFarm**

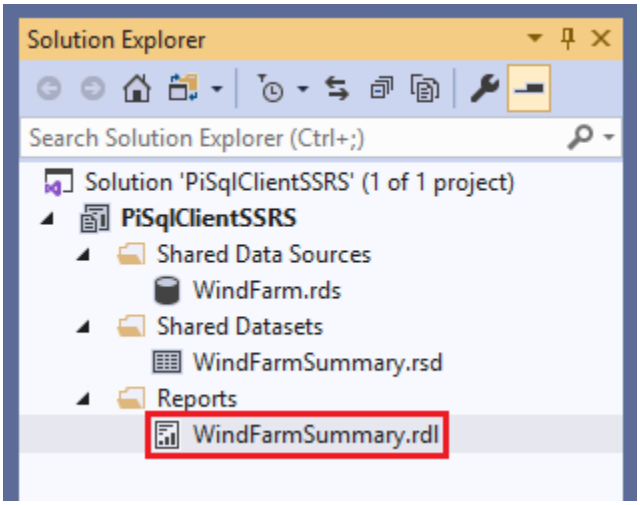
20. To create a dataset, move back to the **Solution Explorer,** right-click the **Shared Datasets** item and select the **Add New Dataset** option.



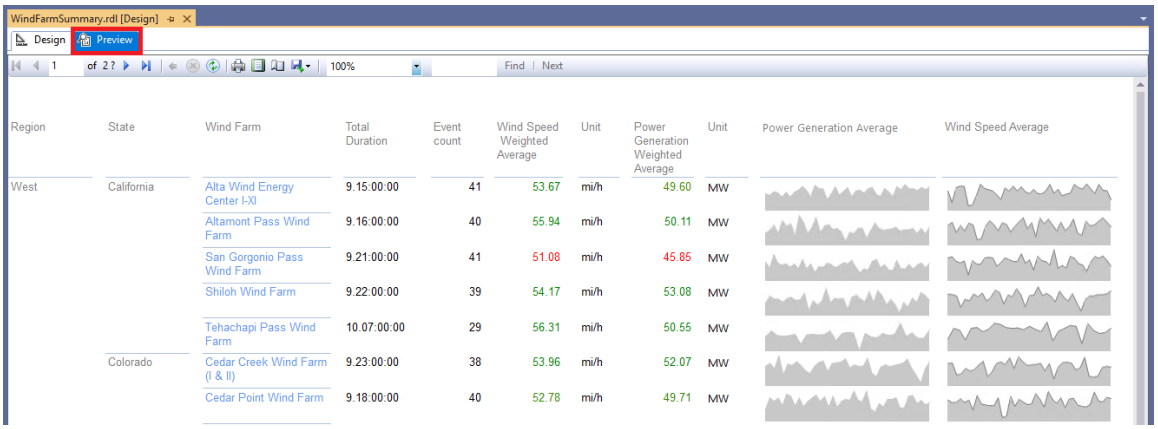
21. Move to the **Shared Dataset Properties** wizard.
    a. Enter the name **WindFarmSummary** (the report is set up to use a dataset with the WindFarmSummary dataset name) and select the previously created **WindFarm** data source.
    b. Make sure the **Query type** is set to **Text**.
    c. Copy and paste the query created in PI SQL Commander Lite (without the parameter restriction) to the **Query** field.
    d. To validate the query, press the **Refresh Fields** button.
    e. Press the **OK** button to proceed.



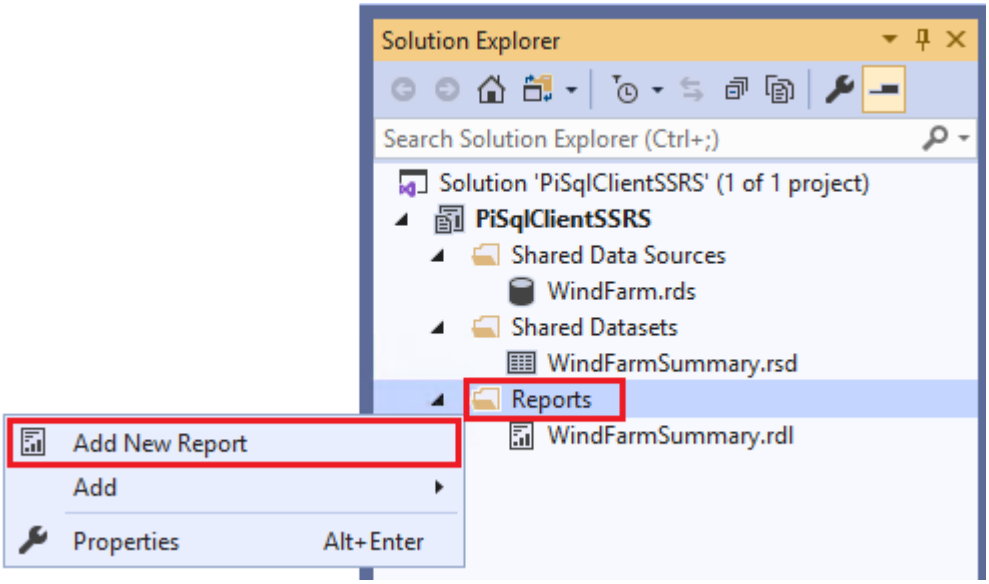
22. Next, open the report by double-clicking the **WindFarmSummary.rdl** item in the Solution Explorer.



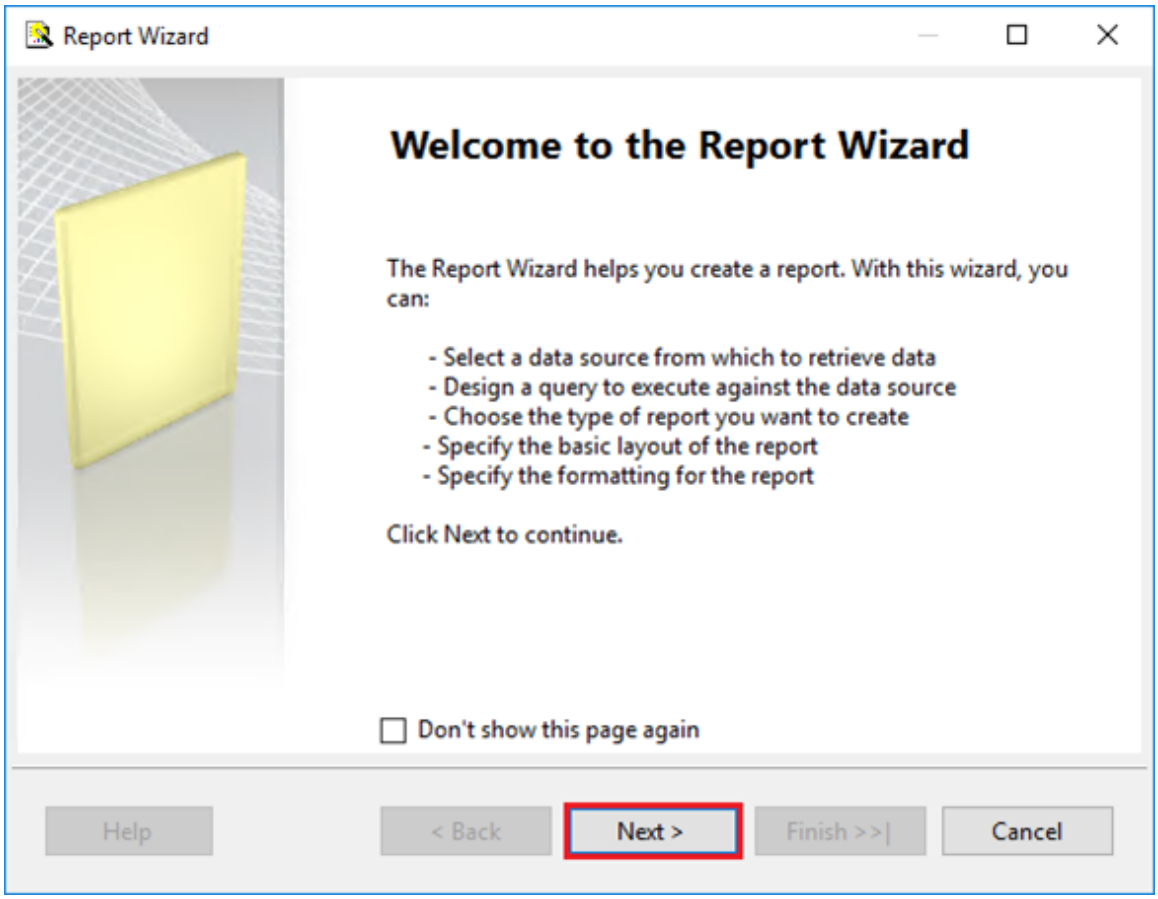
23. In the report tab, navigate to the **Preview** tab and make sure that the data was correctly retrieved.



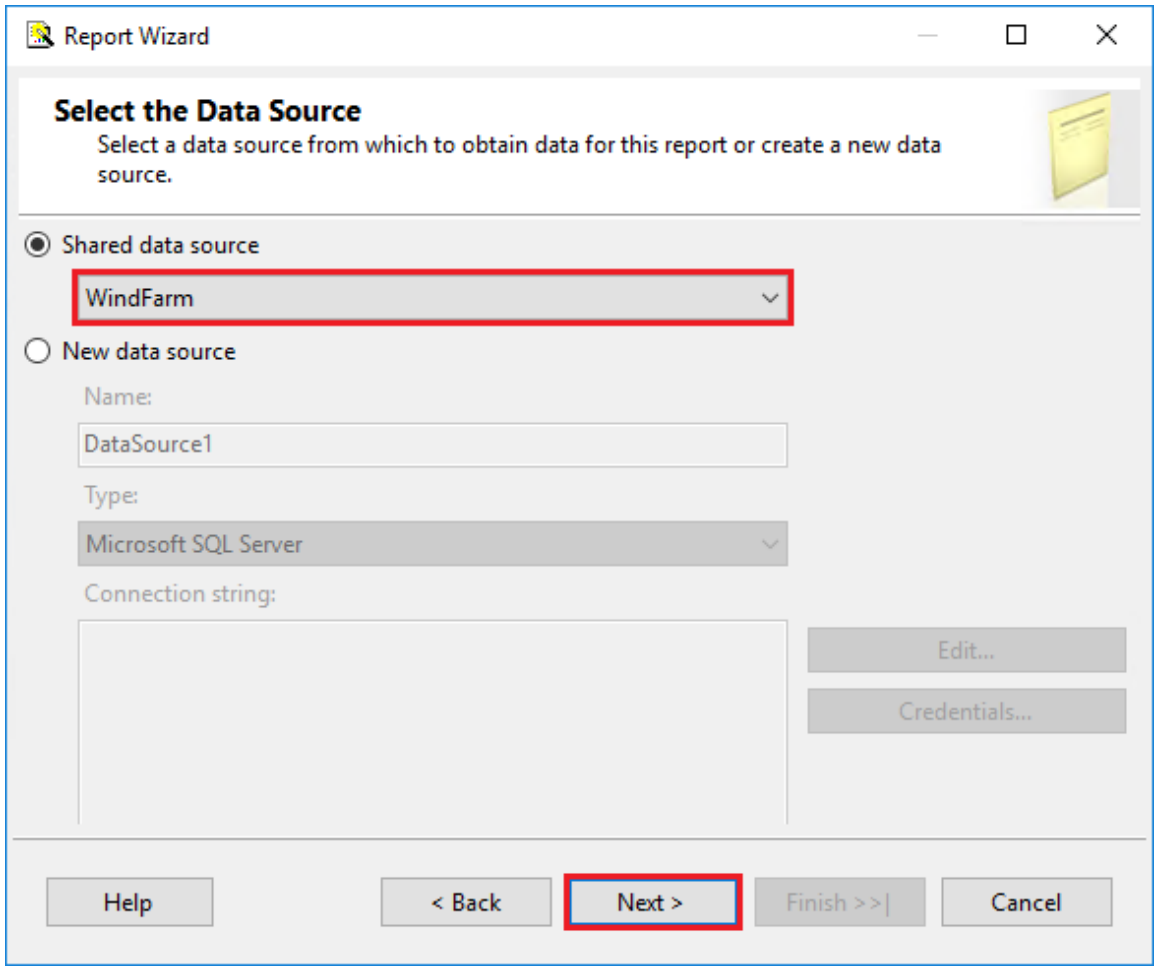
24. In the next few steps, you will create a drill-down report. To start, right-click the **Reports folder** and select **Add New Report**.



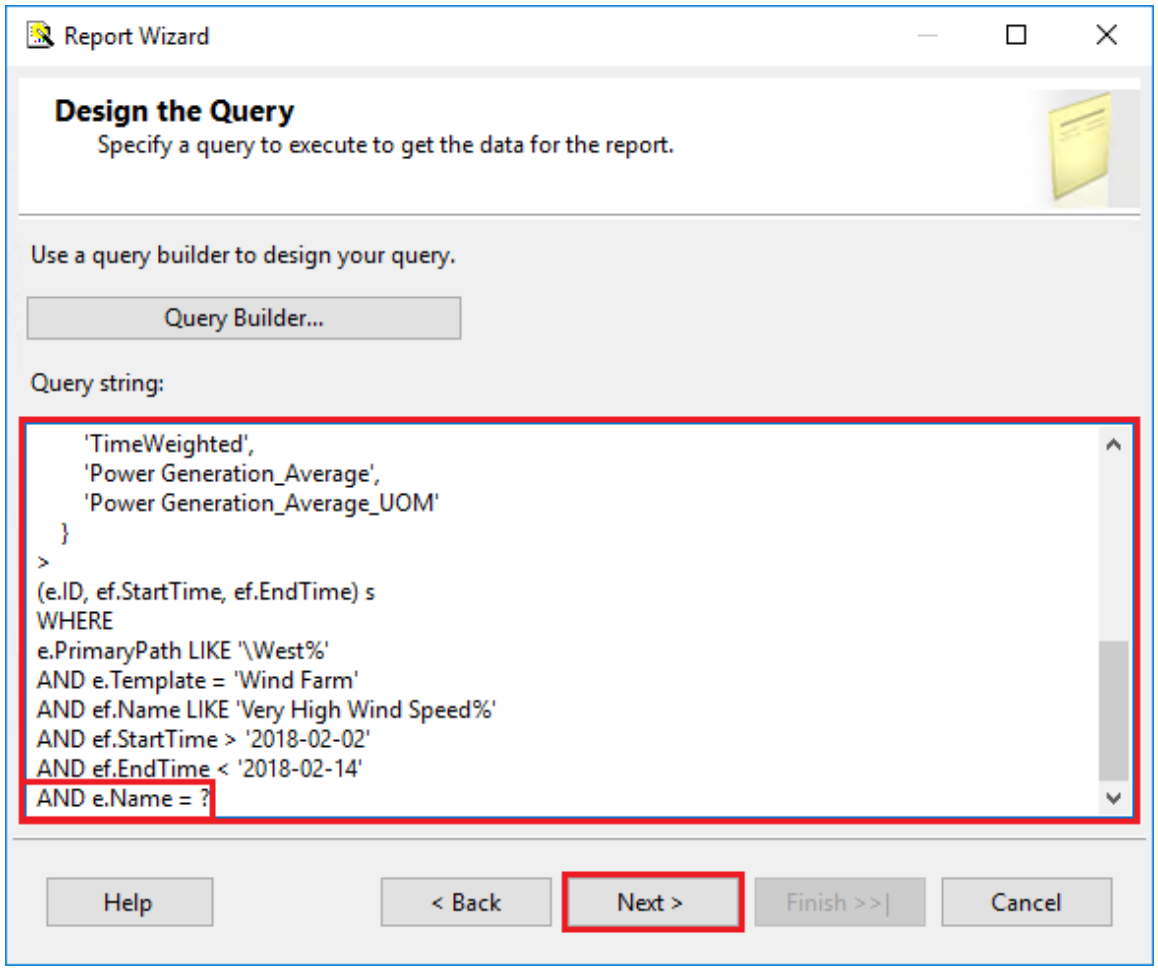
25. On the welcome page of the **Report Wizard**, press the **Next** button.



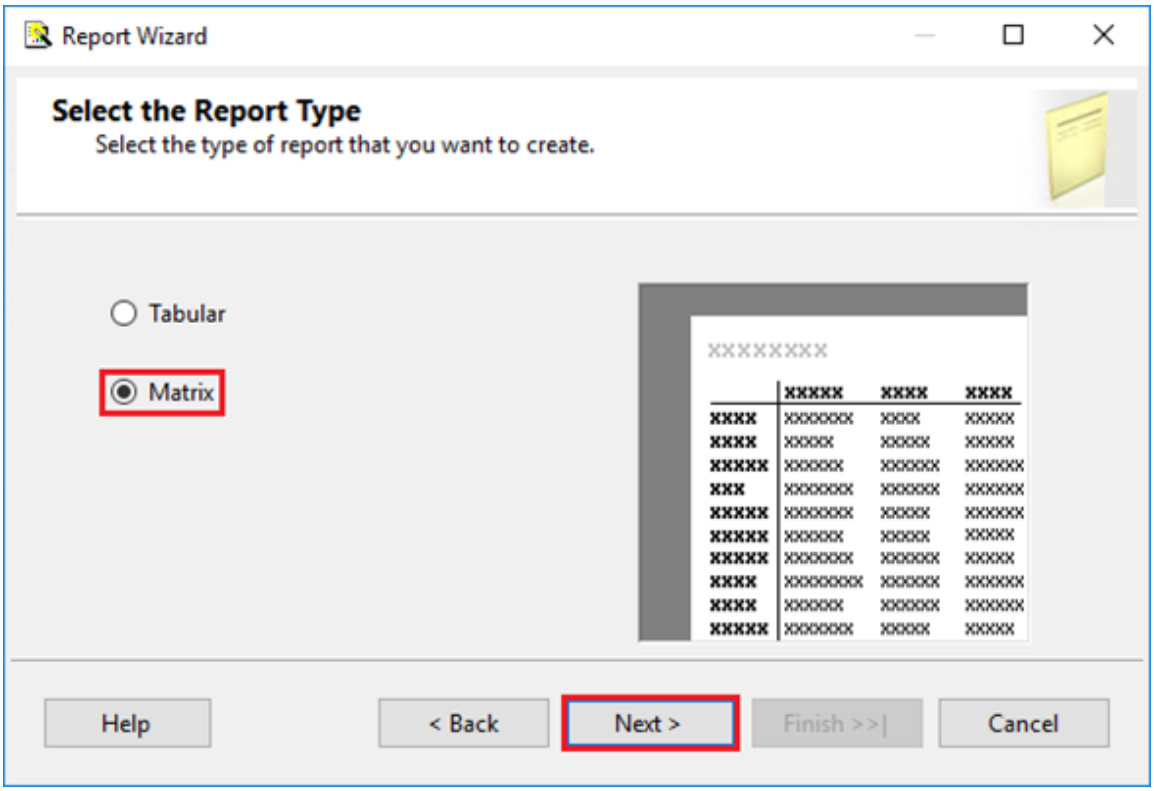
26. Set the **Shared data source** to the previously created **WindFarm** data source and press the **Next** button.



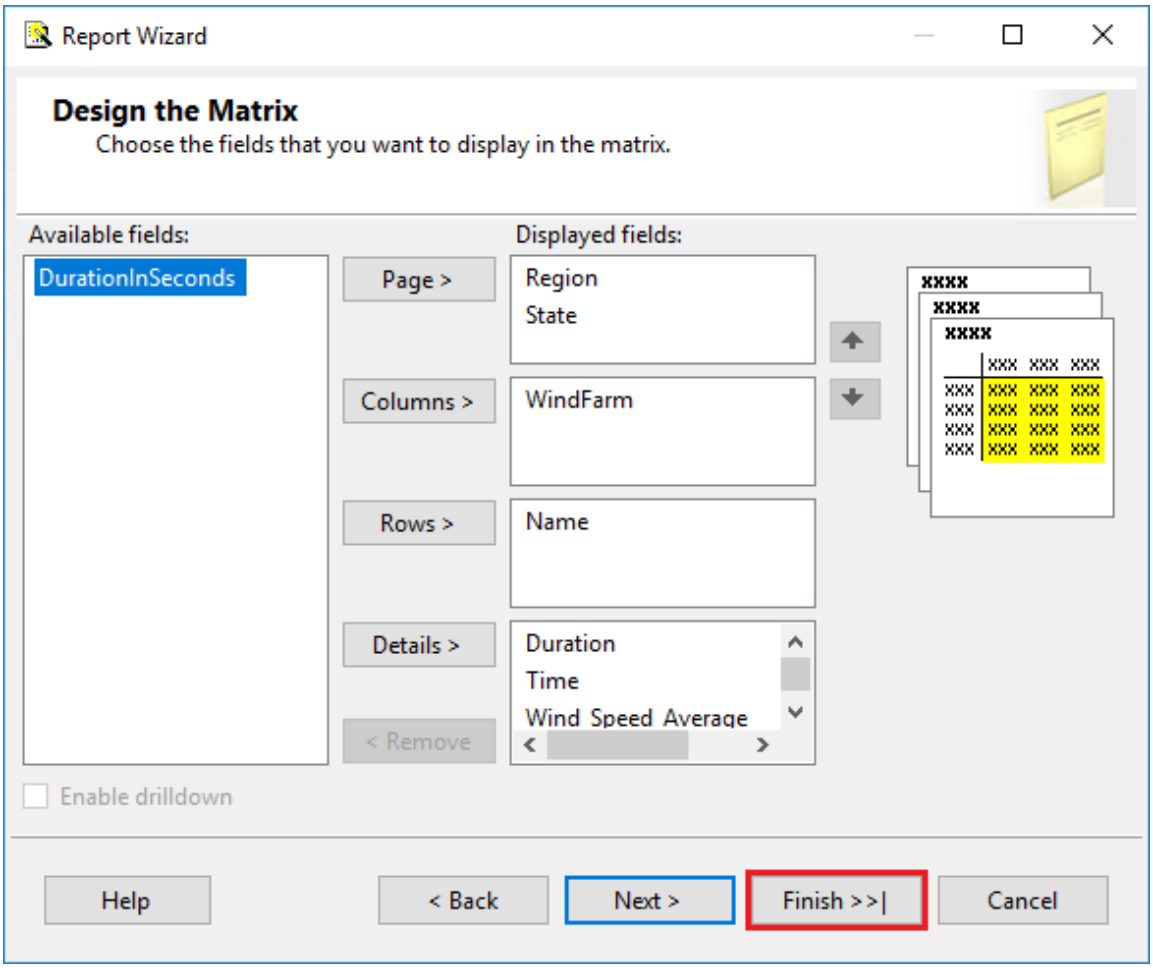
27. Copy and paste the query from **PI SQL Commander Lite** with the parameter restriction.



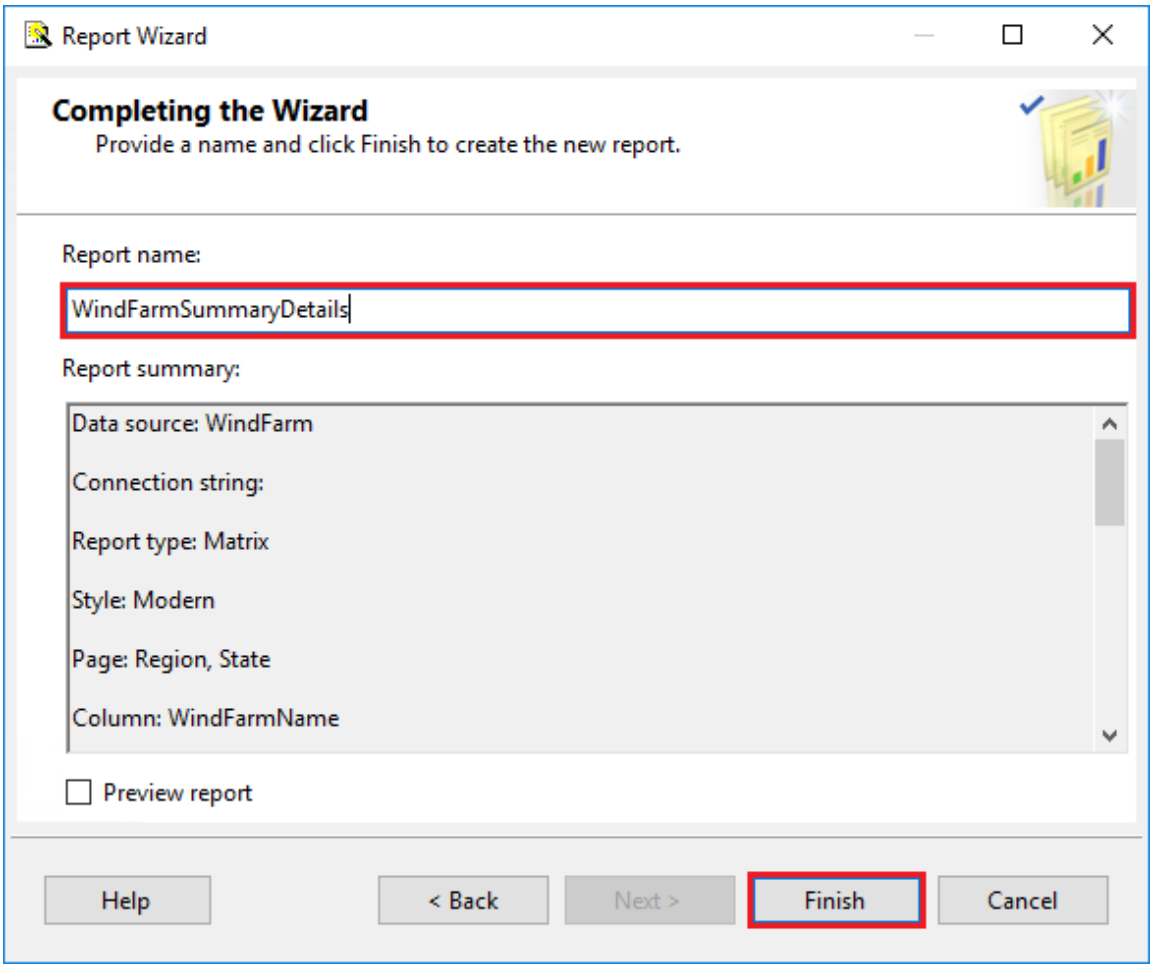
28. Change the Report Type to **Matrix.**



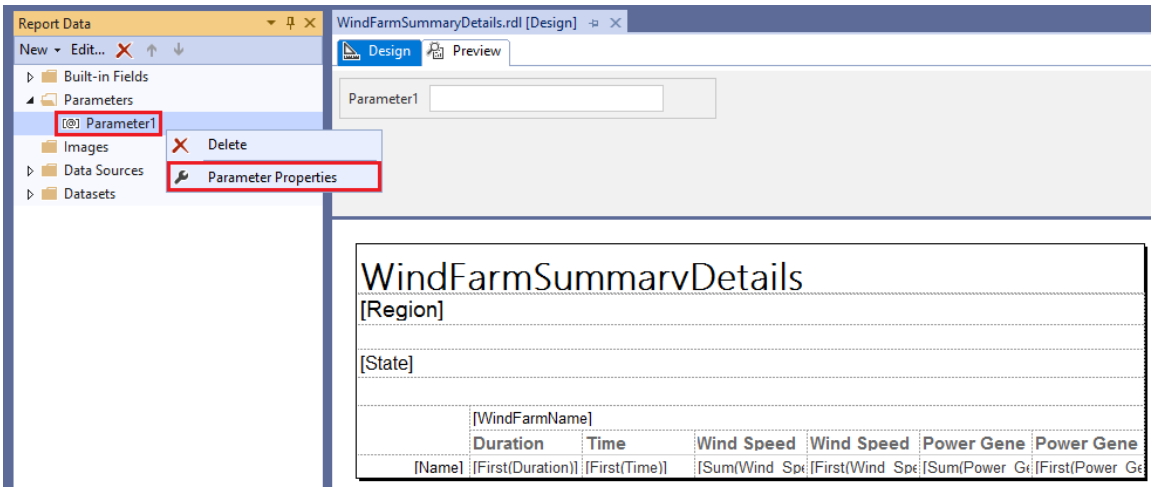
29. In the **Design the Matrix** section of the **Report Wizard,** move the **Available fields** to the appropriate **Display fields** sections.



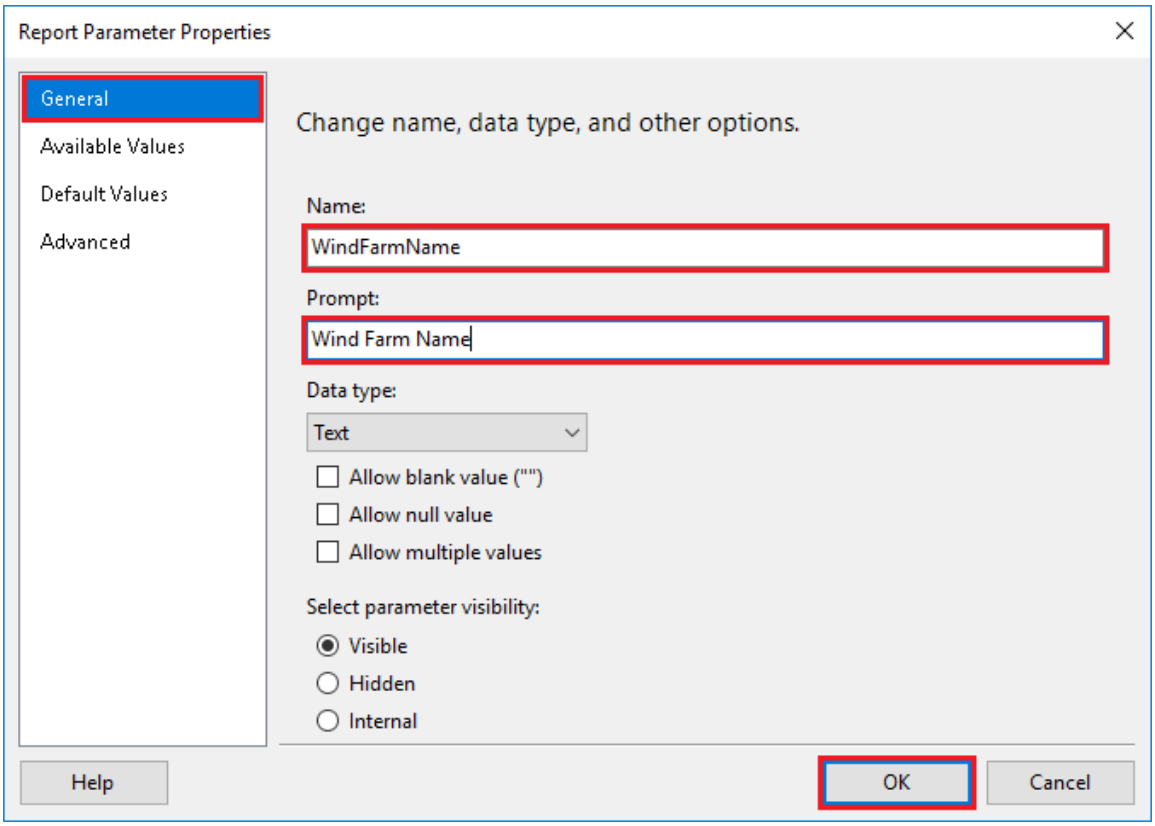
30. On the final page, set the **Report name** to **WindFarmSummaryDetails** and press the **Finish** button.



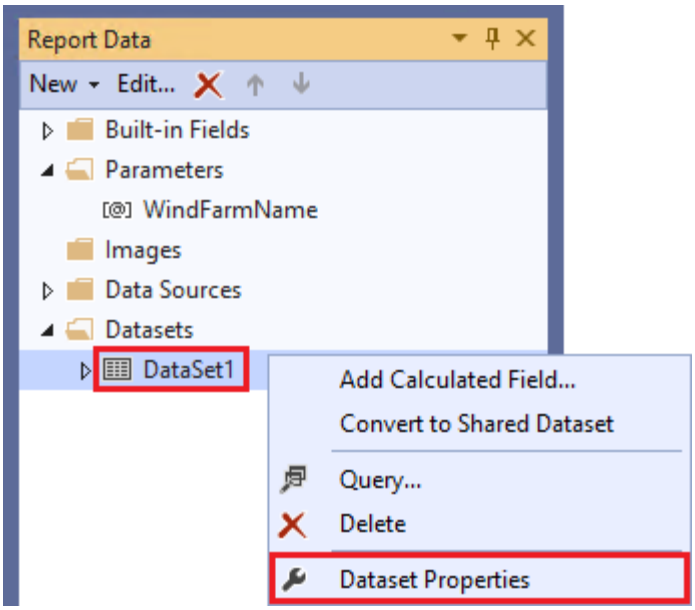
31. Next, right-click the **Parameter Properties**.



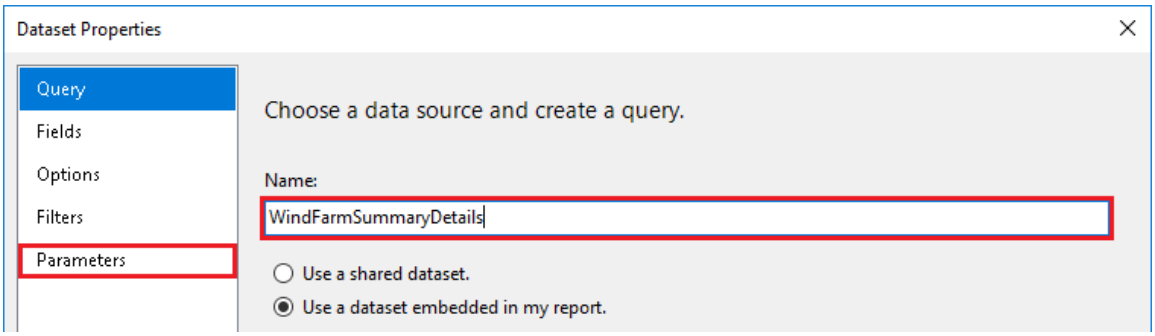
32. Change the name of the parameter to **WindFarmName** and the prompt to **Wind Farm Name.** Click **OK.**



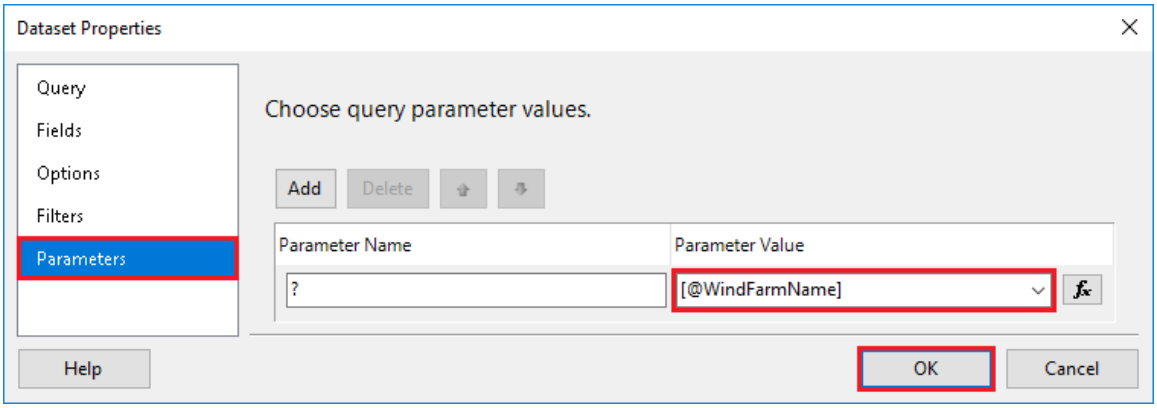
33. Right-click the **DataSets->DataSet1** object.



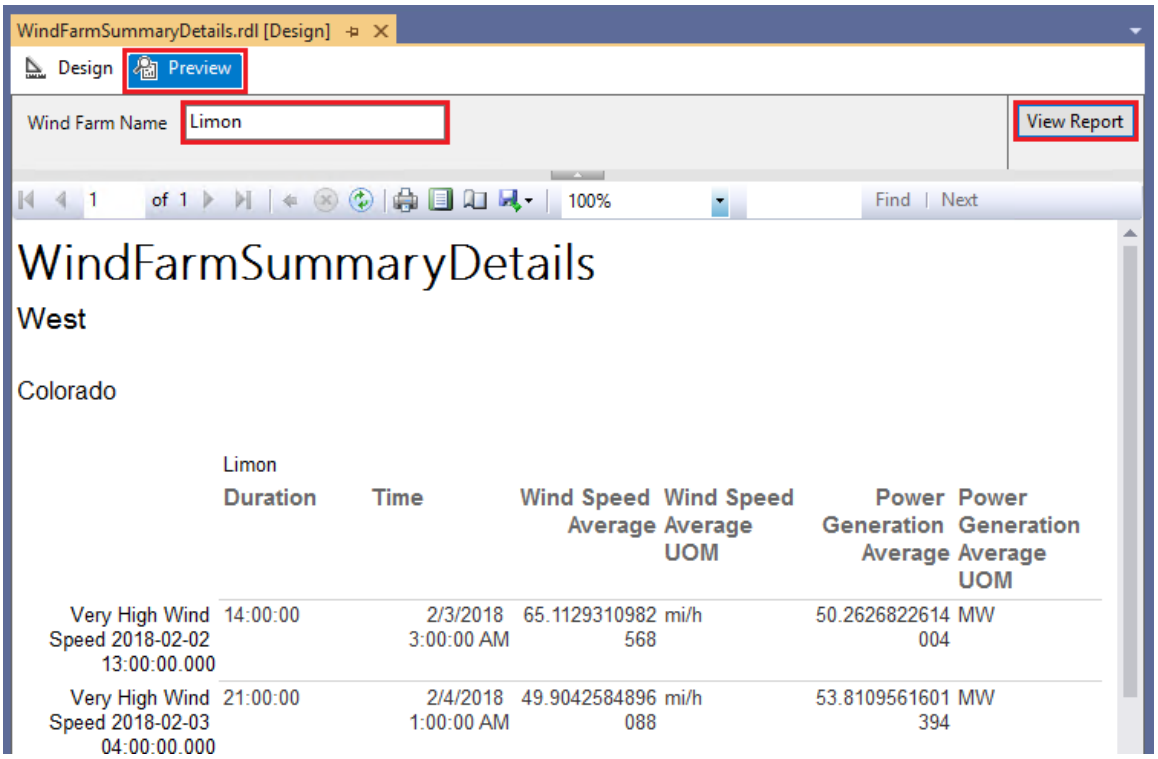
34. Rename the created data set to **WindFarmSummaryDetails** and navigate to the **Parameters** section.



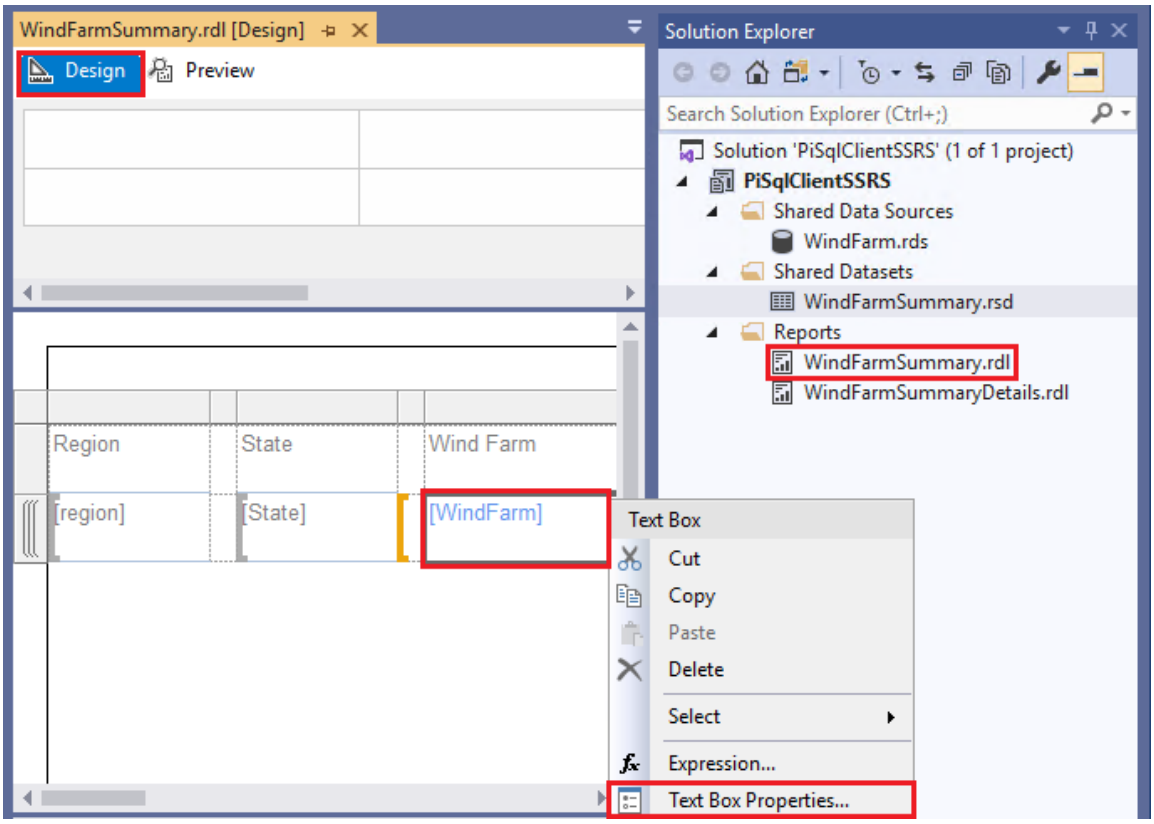
35. Set the **Parameter Value** to the **[@WindFarmName]** and press the **OK** button.



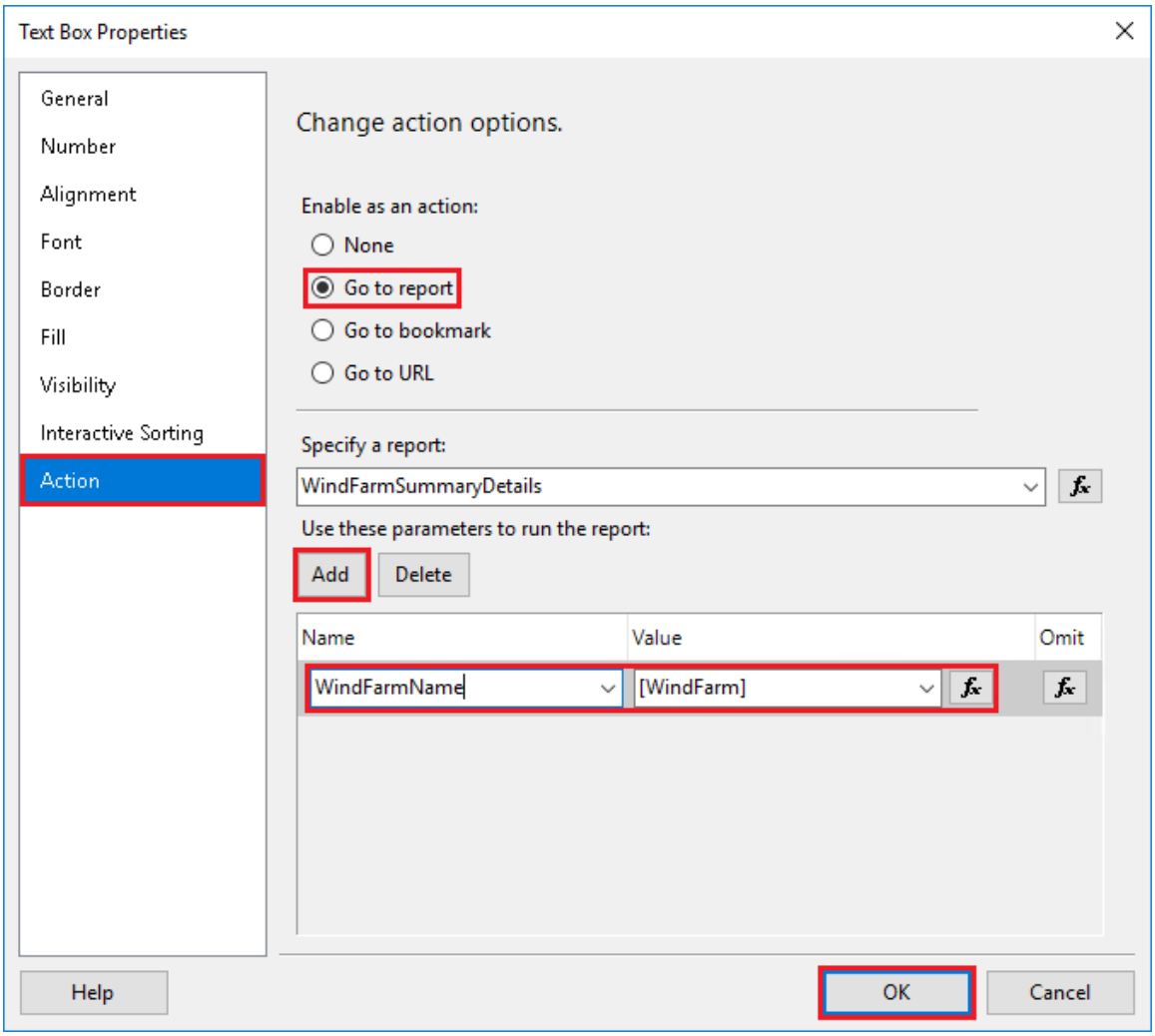
36. Navigate to the **Preview** tab, enter a wind farm name (e.g., Limon) and press the **View Report** button.



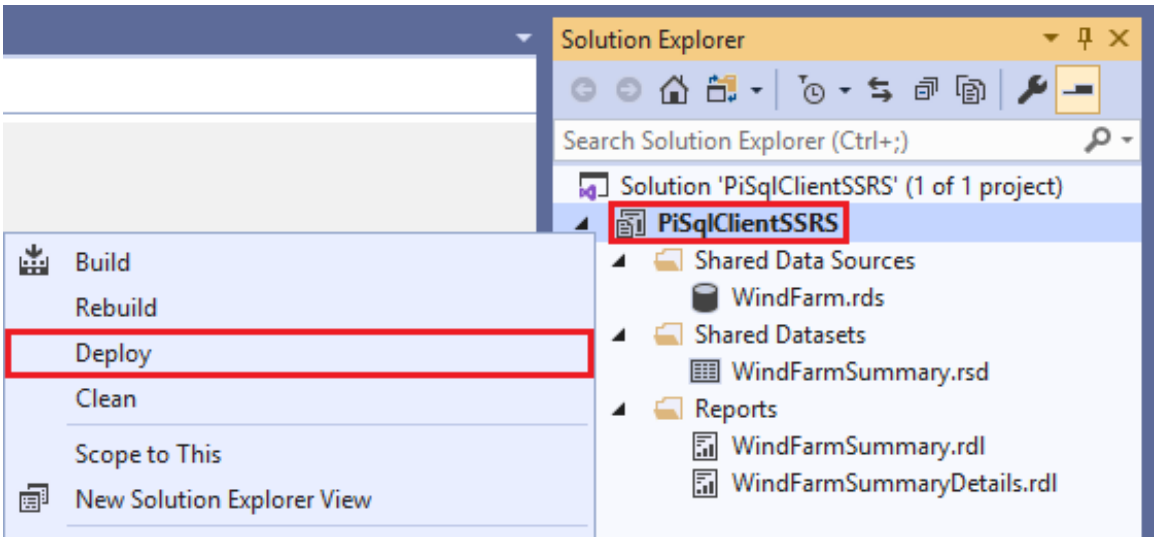
37. Navigate to the **WindFarmSummary.rdl** report to open a sub-report on click. Right-click the **[WindFarm]** cell and select the **Text Box Properties…** item.



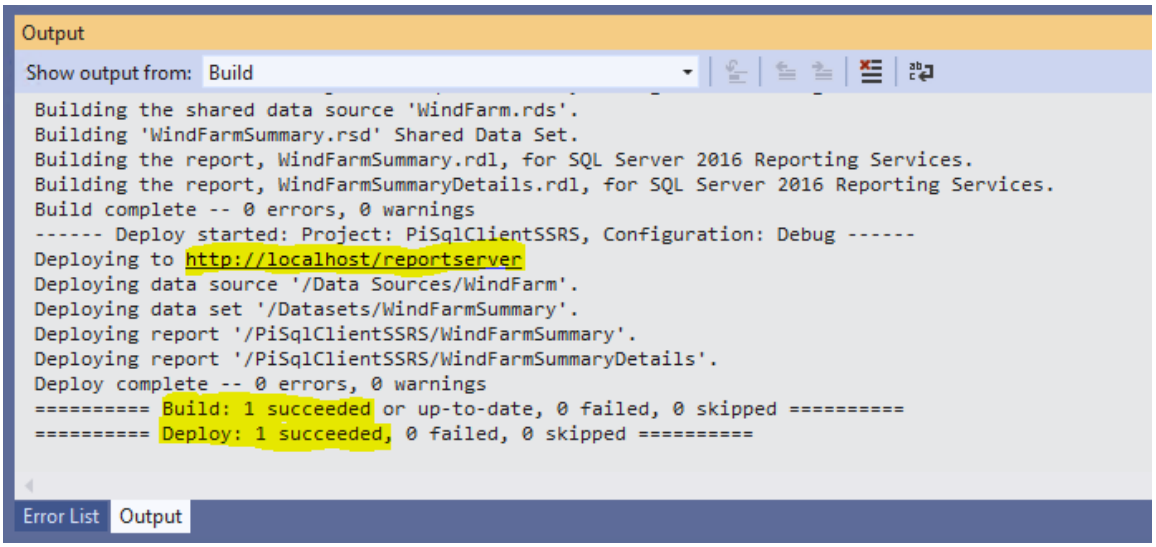
38. Navigate to the **Action** section, select the **Go to report** option, and change the **Specify a report** drop down item to the created **WindFarmSummaryDetails**. Next, specify which field should be passed as the report parameter by selecting the **[windFarmName]** value. Press the **OK** button.



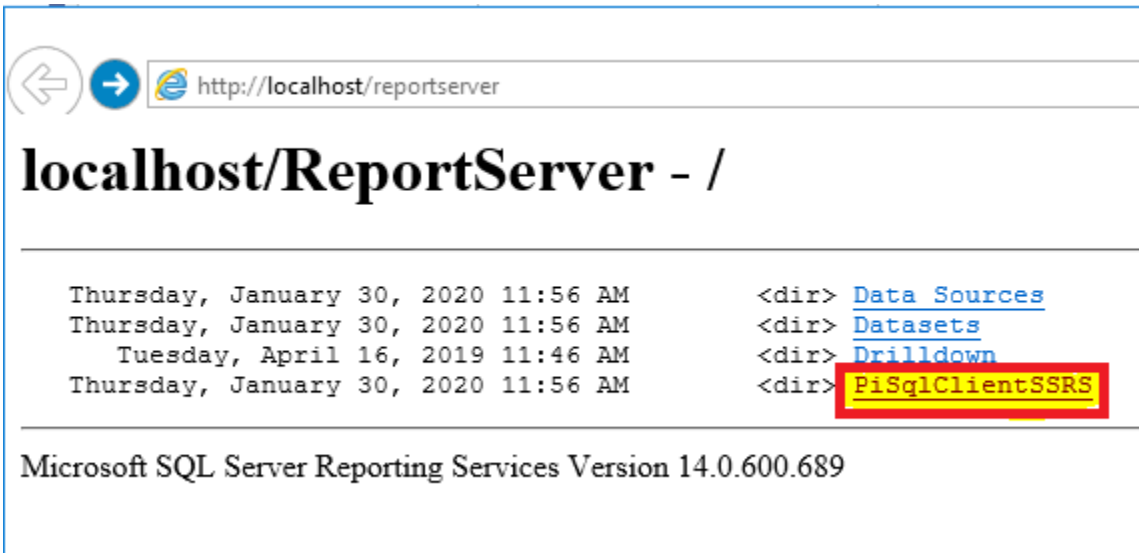
39. Next, publish the Wind Farm report by right clicking the **PiSqlClientSSRS** solution and selecting the **Deploy** context menu item.



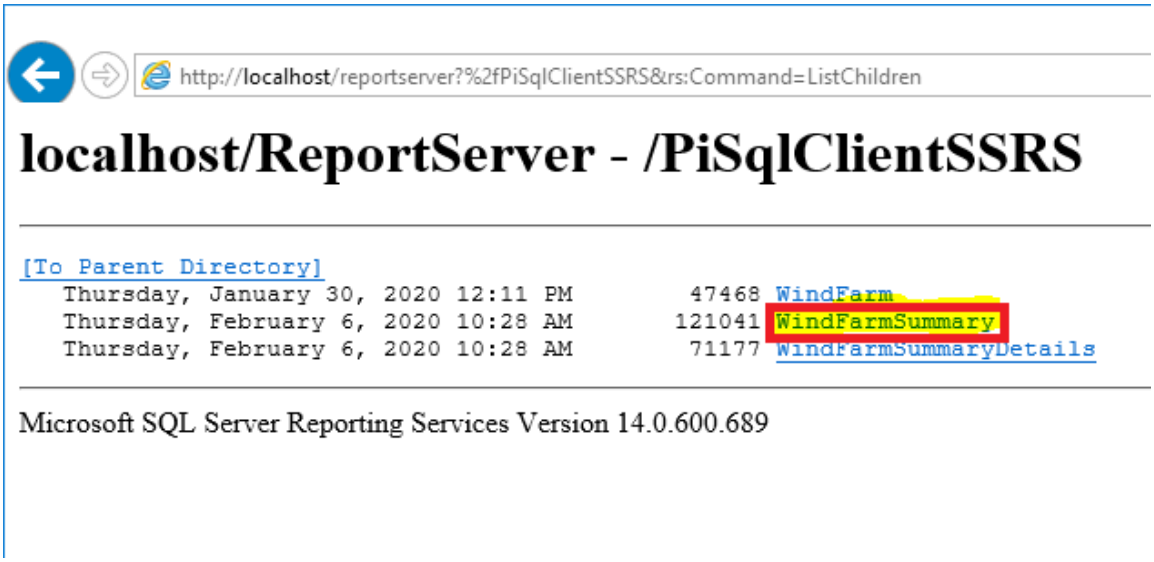
40. At the bottom of the **Output** window, make sure that the solution was built and published without any issues.



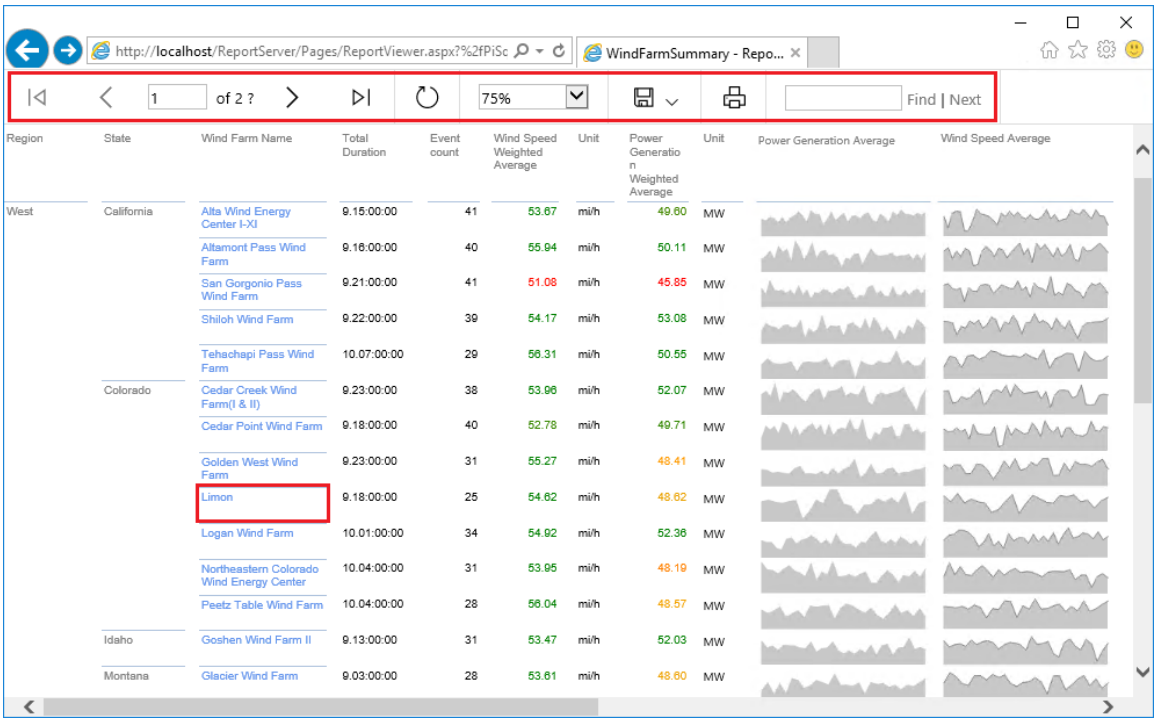
41. Navigate to the URL displayed in the output window and select **PiSqlClientSSRS**.



42. Select the **WindFarmSummary** report.

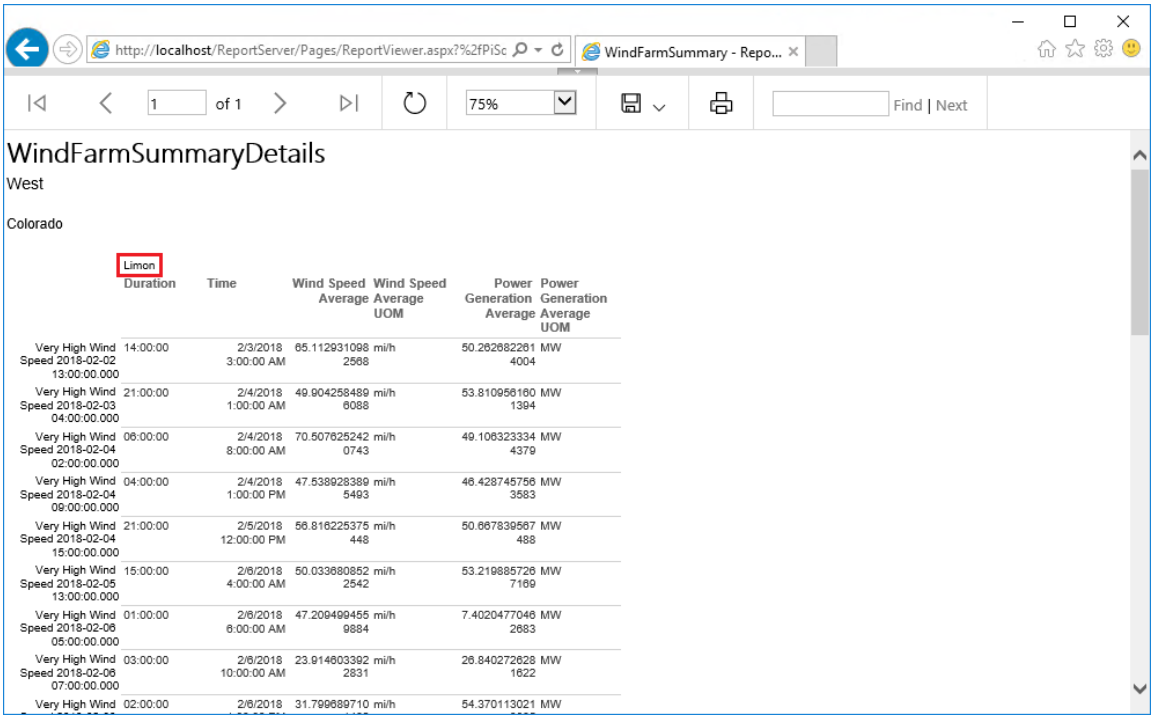


43. Navigate to a **WindFarm** (e.g., **Limon**) by clicking the wind farm name.



Region	State	Wind Farm Name	Total Duration	Event count	Wind Speed Weighted Average	Unit	Power Generation Weighted Average	Unit	Power Generation Average	Wind Speed Average
West	California	Alta Wind Energy Center I-XI	9.15:00:00	41	53.67	mi/h	49.60	MW		
		Altamont Pass Wind Farm	9.16:00:00	40	55.94	mi/h	50.11	MW		
		San Gorgonio Pass Wind Farm	9.21:00:00	41	51.08	mi/h	45.85	MW		
		Shiloh Wind Farm	9.22:00:00	39	54.17	mi/h	53.08	MW		
		Tehachapi Pass Wind Farm	10.07:00:00	29	56.31	mi/h	50.55	MW		
	Colorado	Cedar Creek Wind Farm(I & II)	9.23:00:00	38	53.96	mi/h	52.07	MW		
		Cedar Point Wind Farm	9.18:00:00	40	52.78	mi/h	49.71	MW		
		Golden West Wind Farm	9.23:00:00	31	55.27	mi/h	48.41	MW		
		Limon	9.18:00:00	25	54.62	mi/h	48.62	MW		
		Logan Wind Farm	10.01:00:00	34	54.92	mi/h	52.36	MW		
		Northeastern Colorado Wind Energy Center	10.04:00:00	31	53.95	mi/h	48.19	MW		
		Peetz Table Wind Farm	10.04:00:00	28	56.04	mi/h	48.57	MW		
	Idaho	Goshen Wind Farm II	9.13:00:00	31	53.47	mi/h	52.03	MW		
	Montana	Glacier Wind Farm	9.03:00:00	28	53.61	mi/h	48.60	MW		

44. Finally, you should be able to see the details for your selected wind farm.



WindFarmSummaryDetails

West

Colorado

Limon

Duration	Time	Wind Speed Average	Wind Speed Average UOM	Power Generation Average	Power Generation Average UOM
Very High Wind Speed 2018-02-02 13:00:00.000	14:00:00	2/3/2018 3:00:00 AM	65.112931098 2568	mi/h	50.262682261 4004	MW
Very High Wind Speed 2018-02-03 04:00:00.000	21:00:00	2/4/2018 1:00:00 AM	49.904258489 6088	mi/h	53.810956160 1394	MW
Very High Wind Speed 2018-02-04 02:00:00.000	06:00:00	2/4/2018 8:00:00 AM	70.507625242 0743	mi/h	49.108323334 4379	MW
Very High Wind Speed 2018-02-04 09:00:00.000	04:00:00	2/4/2018 1:00:00 PM	47.538928389 5493	mi/h	46.428745756 3583	MW
Very High Wind Speed 2018-02-04 15:00:00.000	21:00:00	2/5/2018 12:00:00 PM	56.816225375 448	mi/h	50.667839567 488	MW
Very High Wind Speed 2018-02-05 13:00:00.000	15:00:00	2/6/2018 4:00:00 AM	50.033680852 2542	mi/h	53.219885726 7169	MW
Very High Wind Speed 2018-02-06 05:00:00.000	01:00:00	2/6/2018 6:00:00 AM	47.209499455 9884	mi/h	7.4020477046 2683	MW
Very High Wind Speed 2018-02-06 07:00:00.000	03:00:00	2/6/2018 10:00:00 AM	23.914603392 2831	mi/h	26.840272628 1622	MW
Very High Wind	02:00:00	2/6/2018	31.799689710	mi/h	54.370113021	MW

6.3 Conclusion

You created two queries from scratch by leveraging the knowledge gathered in the previous exercises. Finally, you learned how to create a report using SQL Server Reporting Services by utilizing PI SQL Client.


Have an idea how to improve our products?
OSIsoft wants to hear from you!
https://feedback.osisoft.com/

Save the Date!

OSIsoft PI World Users Conference in Amsterdam; October 26-29, 2020.

Register your interest now to receive updates and notification early bird registration opening.

https://pages.osisoft.com/UC-EMEA-2020-Q4-10-26-PI-World-AMS-Register-Your-Interest_Register-Your-Interest.html

OSIsoft.
PI SYSTEM LEARNING

CONTINUE YOUR PI SYSTEM LEARNING

After the conference, the PI SYSTEM LEARNING does not have to stop. All registered attendees for PI World SFO 2020 will have access to all PI World Hands-on Lab cloud environments for 21 days using the discount cod below. You will receive detailed instructions via email after the conference.

Discount Code: 2020PIWSF-LAB-100

Offer expires July 3, 2020