Course Code: COMP 309

Course Name: Data Warehousing and Mining

Section Number: 005

Assignment 1: Building an OLAP cube

Written by: Kevin Ma

Student Number: 300867968

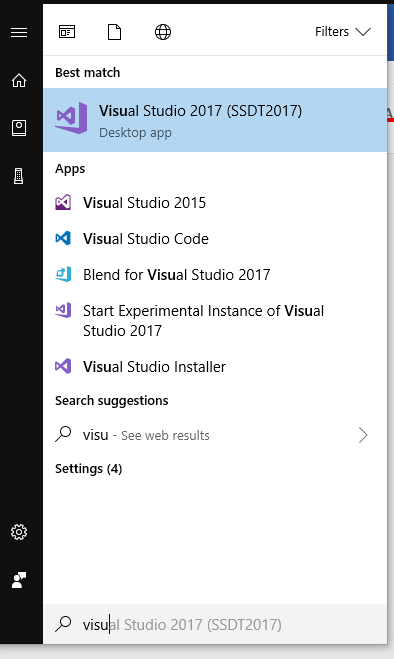
Date: Thursday, September 27, 2018

# Objectives

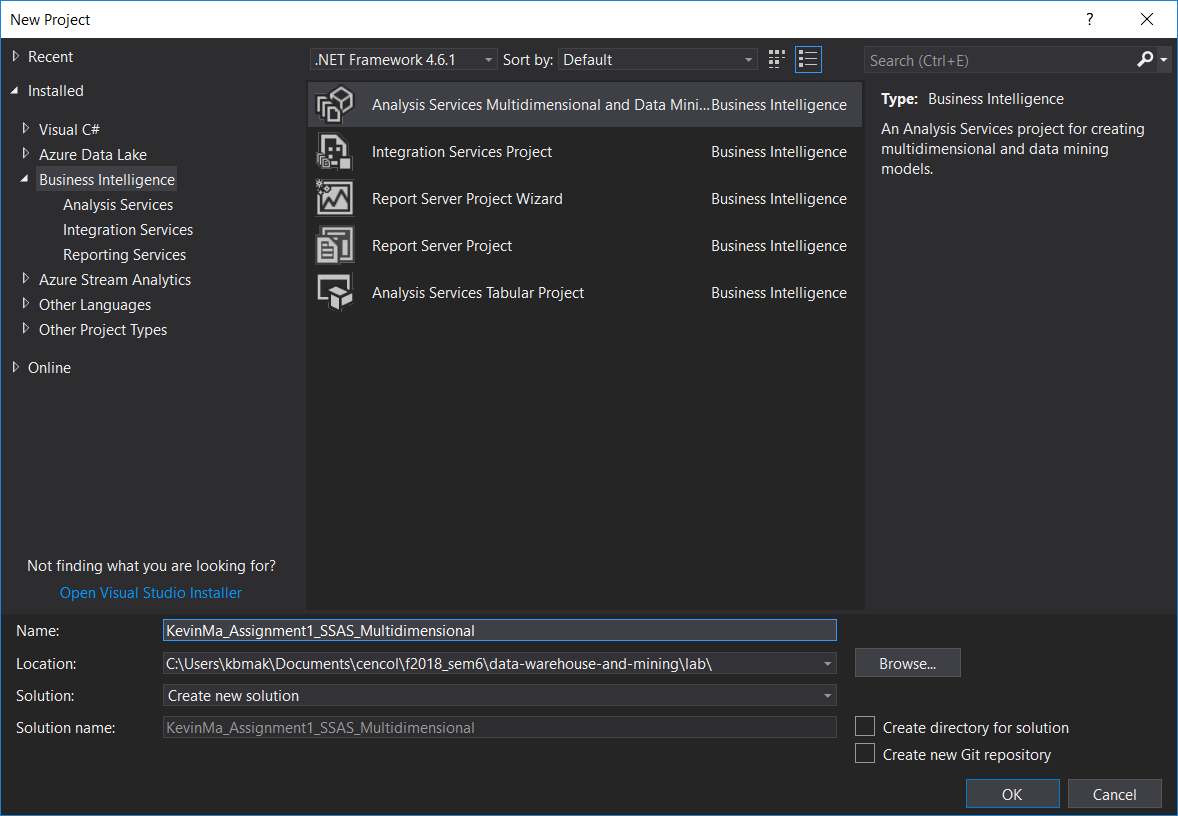
In this project we will be utilizing a Data Warehouse to make information more accessible and easier to use. There are many advantages a data warehouse can bring to the business. People will be able to spend less time retrieving information from data and can thus can spend more time on analyzing it and making better and smarter business decisions to bring in more revenue. Congregating data from multiple data sources allows users to have consolidated views of information in a consistent format. This allows people to create reports much more efficiently and easily. Thus, people will have time to dedicate to more mission critical tasks. Users from the management will especially find data warehouses useful for making business decisions which assist them in deciding how to run their business. Without data warehouses, data would be scattered all over the place in multiple data sources. The data would not be in consistent formats, hard to understand, hard to find, hard to access and would not help management make business decisions. However, through the usage of data warehouses, users will simply be able to enter a simple query or push a button and they will be able to view consolidated reports for easy access. Other users can also use the presented information for analysis to identify trends, answer business questions or to investigate potential risks to the business.

# Time for action – creating an Analysis Services Project

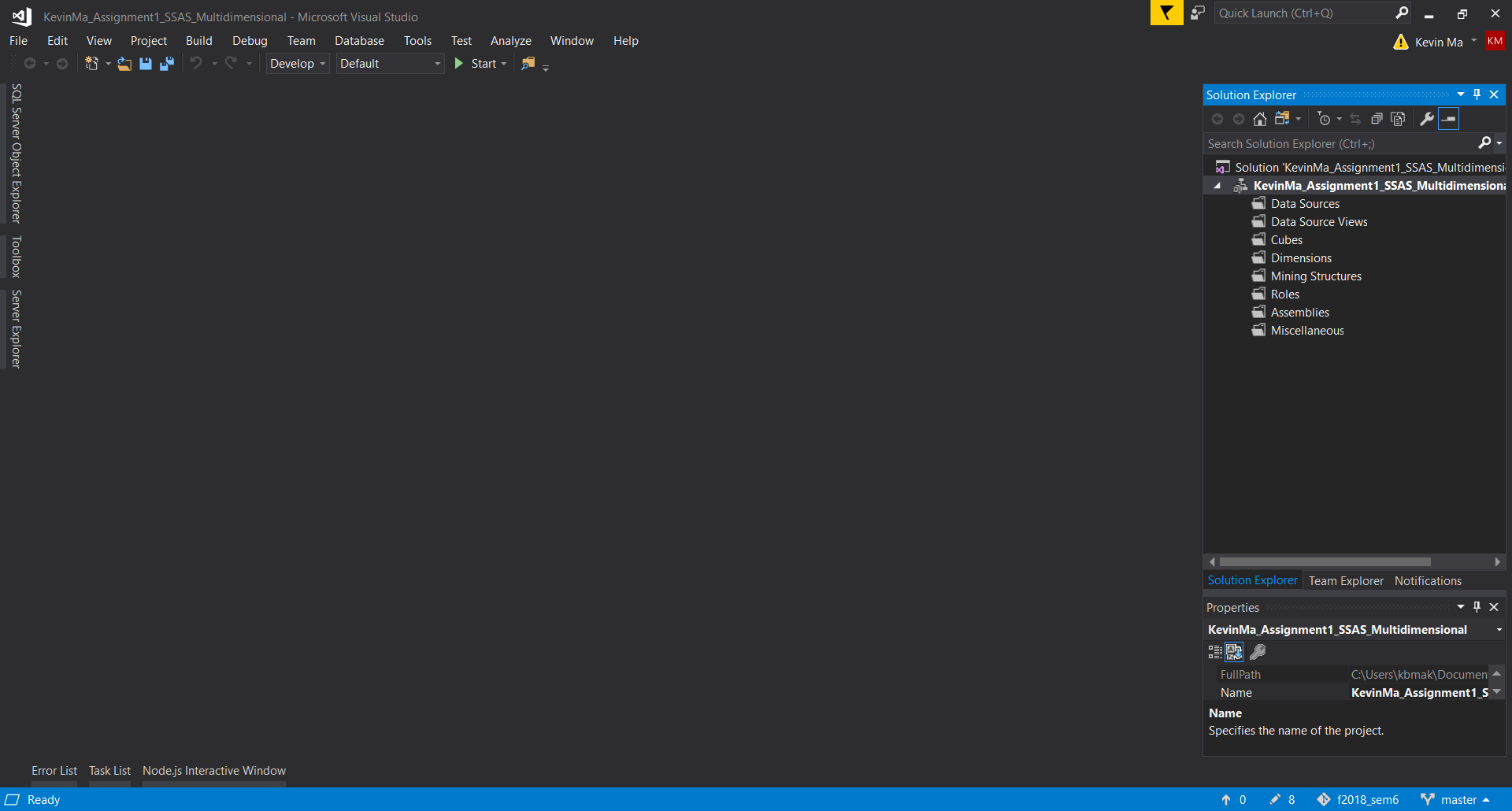
1. Open SQL Server Data Tools for Visual Studio:



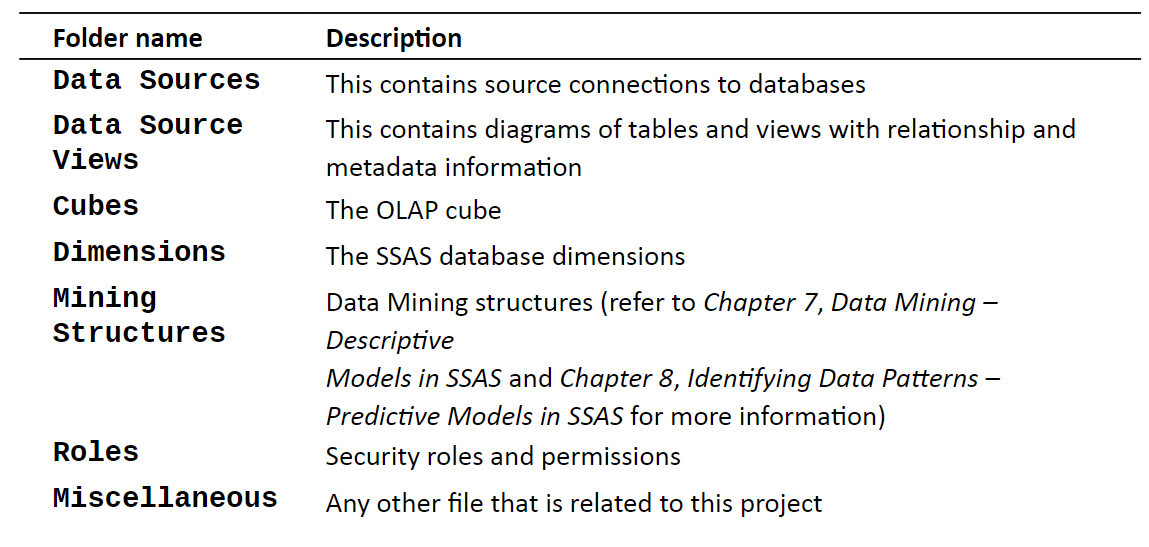
1. Create a new Analysis Services Multidimensional and Data Mining Project from the templates:



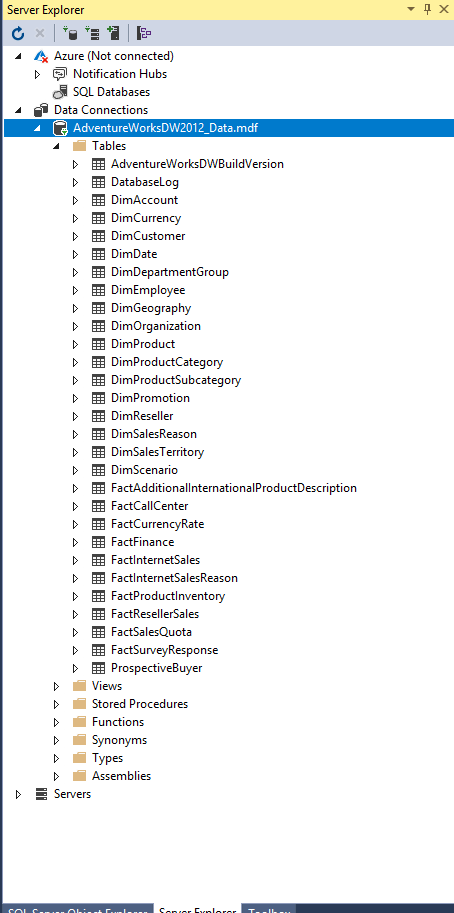
1. Results: A new project was created ☺!



The folders created in the new project represent the following:



1. Add a Data Connection with the downloaded .mdf database file

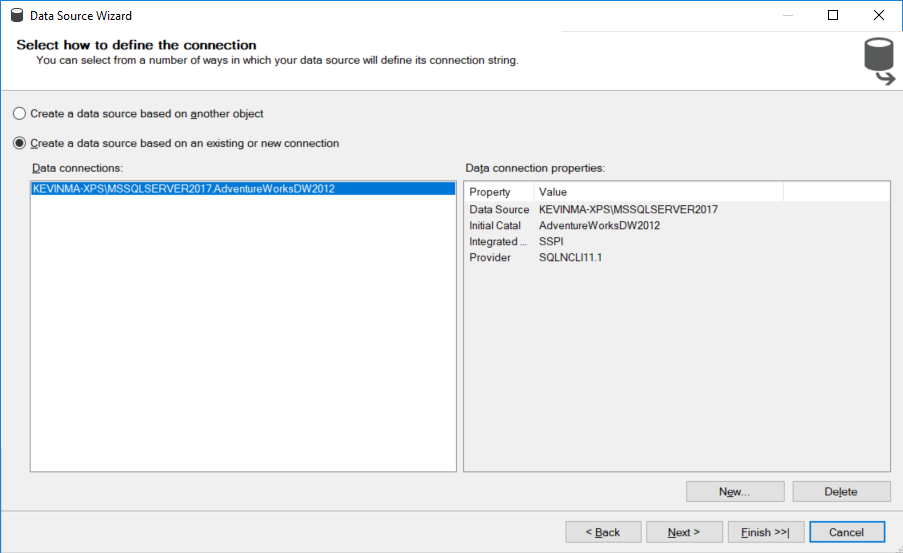


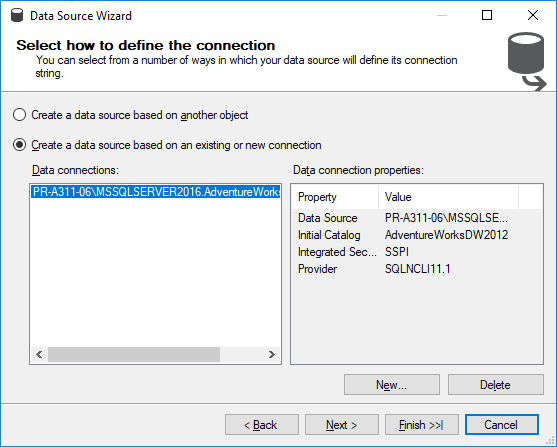
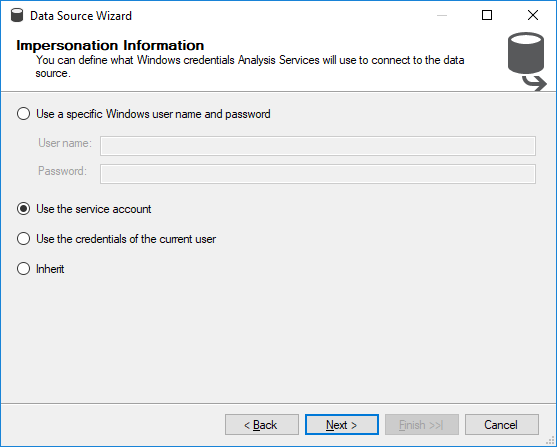
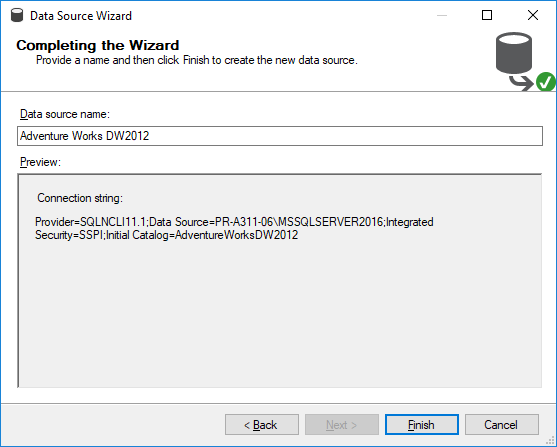
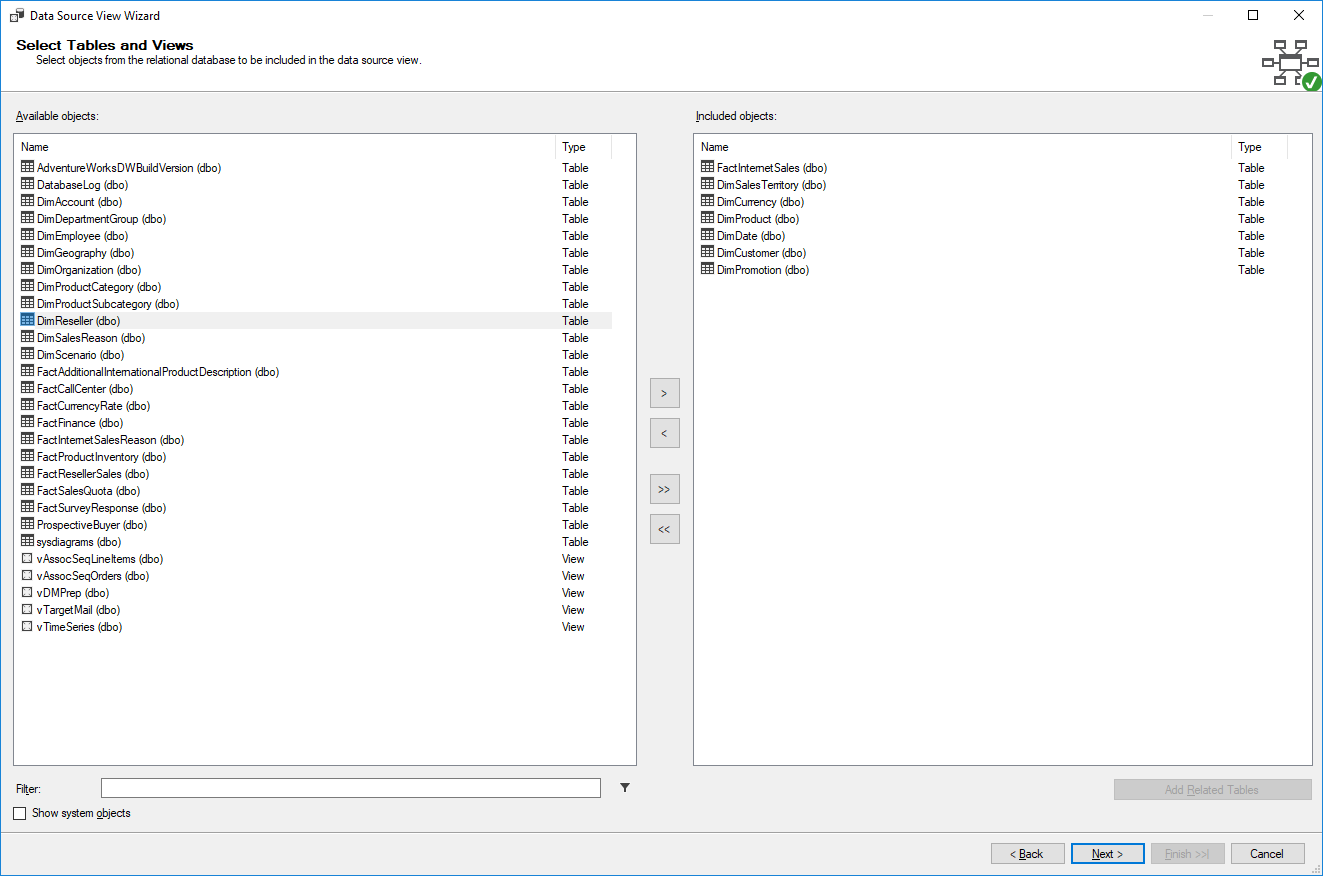
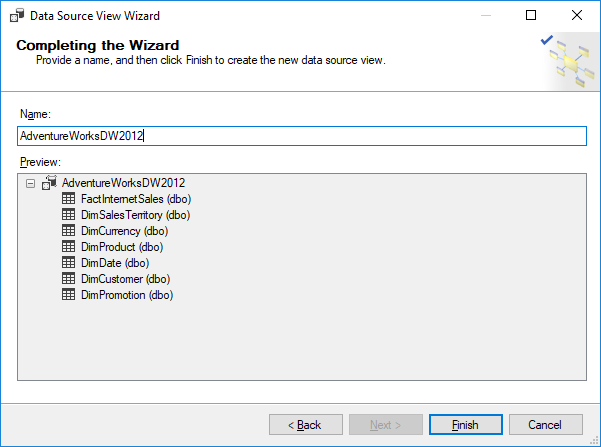
## What did we learn

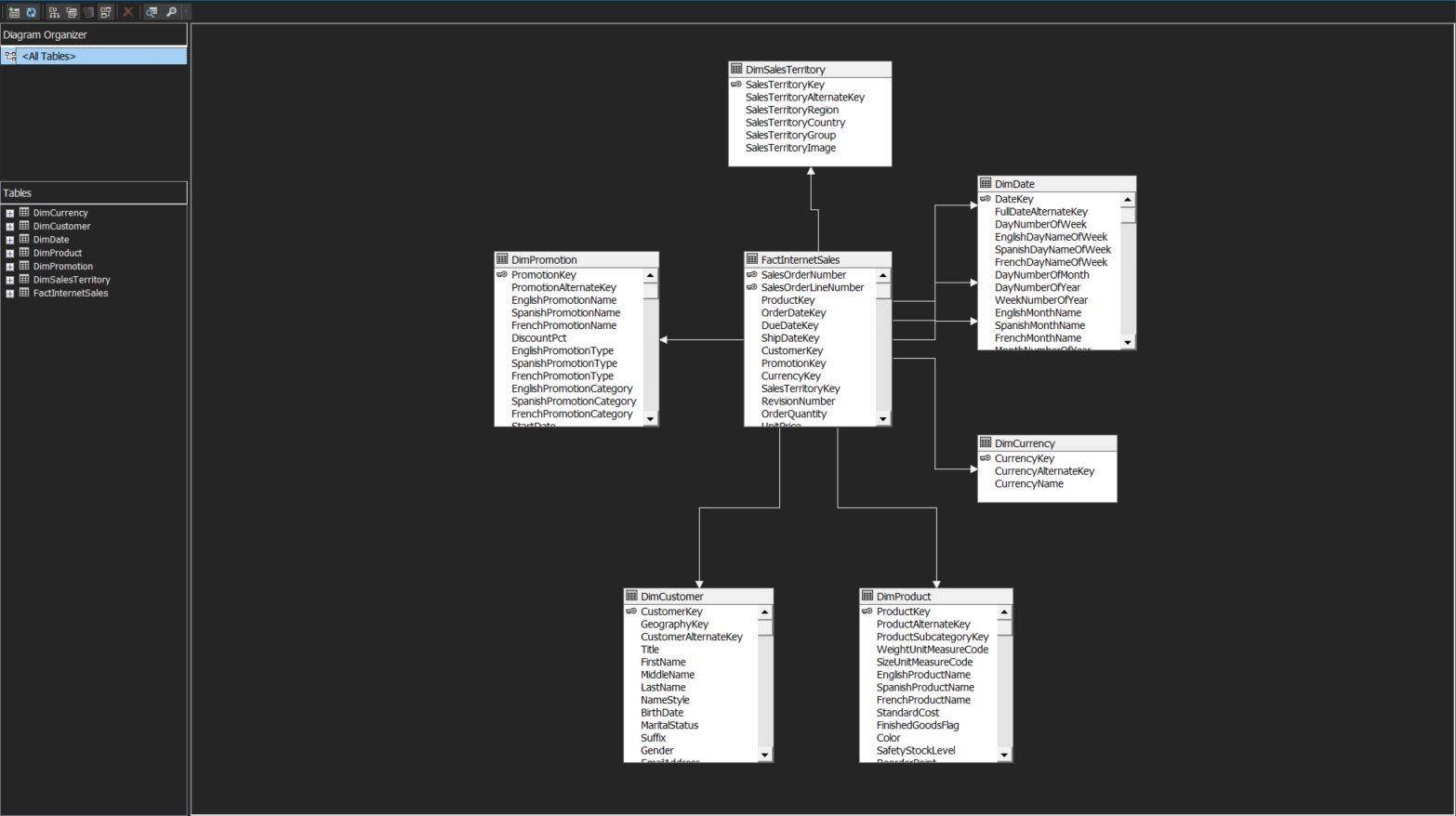
Here we learned how to create an Analysis Services Project. We also saw that by adding a data connection we can see all the database information within visual studio server explorer.

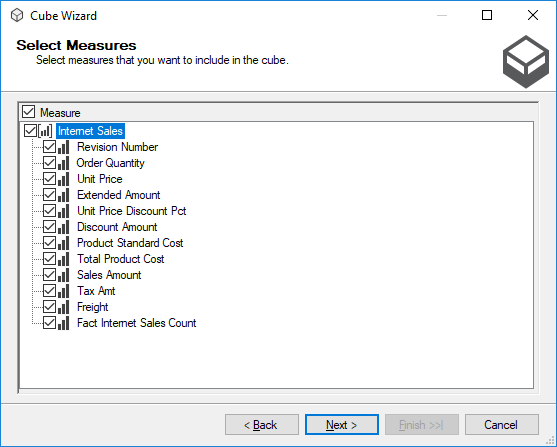
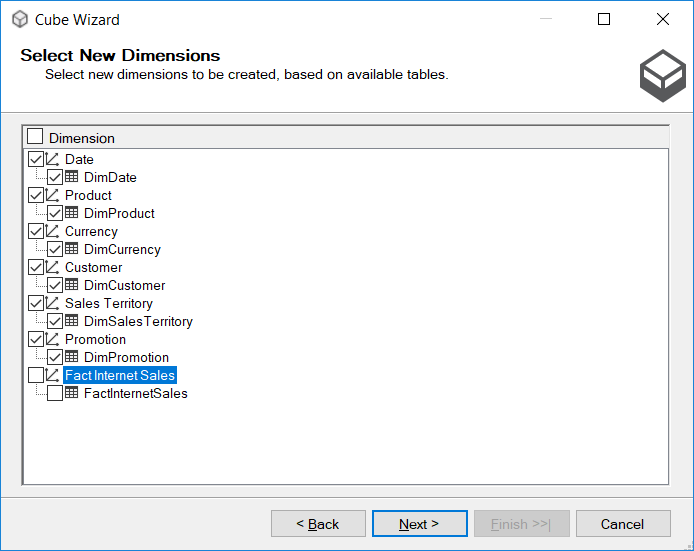
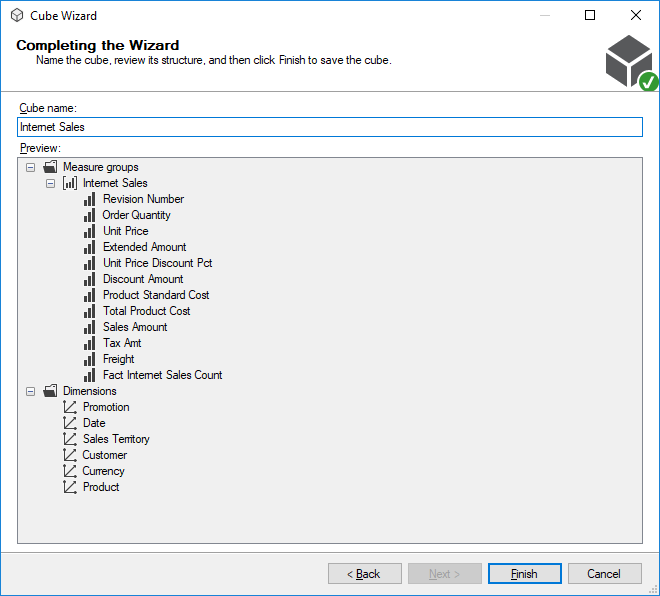
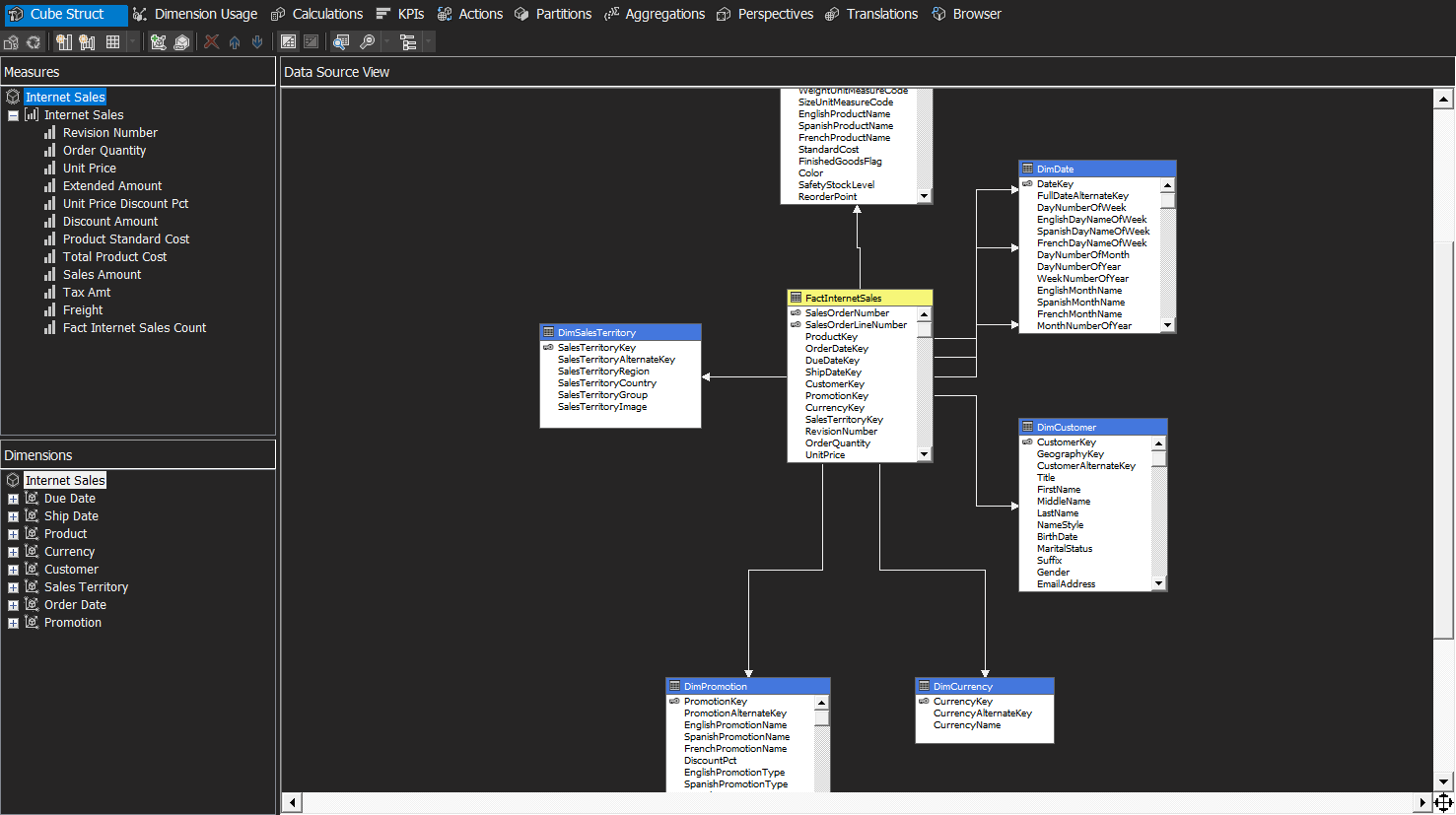
# Time for action – creating the first cube

1. Create a data source connection



1. 
2. 
3. 
4. A data source connection was created using the Data Source Wizard.
5. Create a New Data Source View
6. 
7. 
8. A data source view was created using the Data Source View Wizard



1. Create a Cube using the Cube Wizard
2. 
3. 
4. 
5. 
6. A cube has been created using the Cube Wizard ☺!

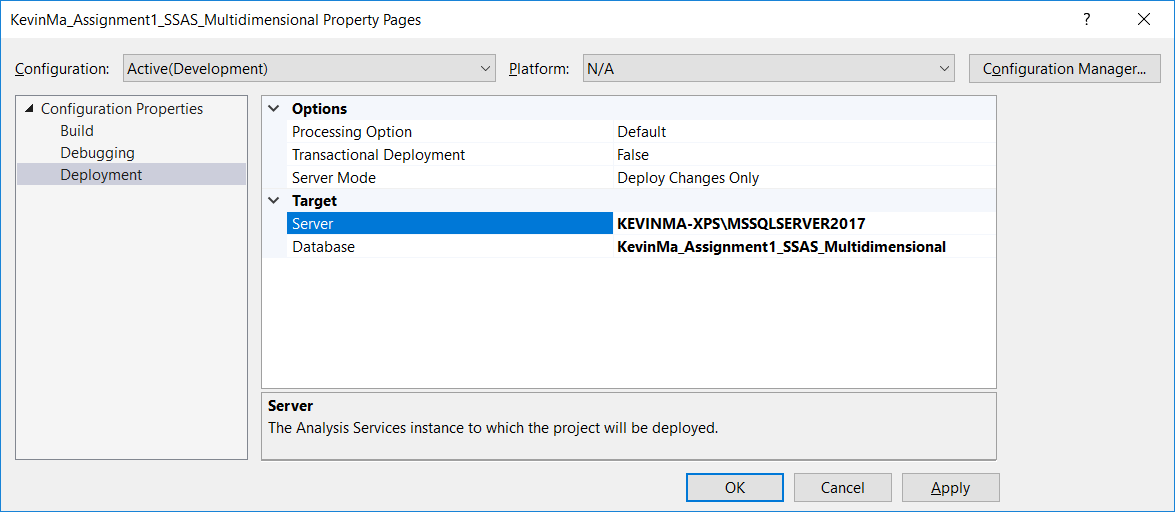
## What did we learn

In this “Time for Action”, we created a connection to a data warehouse using the Data Source Wizard and provided credentials for SSAS (SQL Server Analysis Services) to connect to the underlying database. Following that, we created a data source view using the Data Source View Wizard and the data source we created previously. The data source view will be used to create the base structure for our cube. We then created a cube using the Cube Wizard. Measure groups and dimensions were defined to create the cube. Measure groups are similar to fact tables and each measure group can contain one or more facts. We selected FactInternetSales table as the measure group and it contains measures such as Sales Amount and Order Quantity. Finally, we defined the dimensions to create the cube. There are two different sets of Dimensions we see here. On the right side panel, the dimensions shown are the database dimensions. The dimensions on the left side panel are the cube dimensions. The difference between these two are that we may have some database dimensions that are not used within the cube. We can have multiple cubes within the same project! Or we can have the same database dimension re-used within multiple cubes. We even have dimensions which may be used multiple times within the same cube. We call these role-playing dimensions. For example, there is only one database dimension, Date, but there are three role-playing dimensions in the cube, which are named OrderDate, ShipDate, and DueDate.

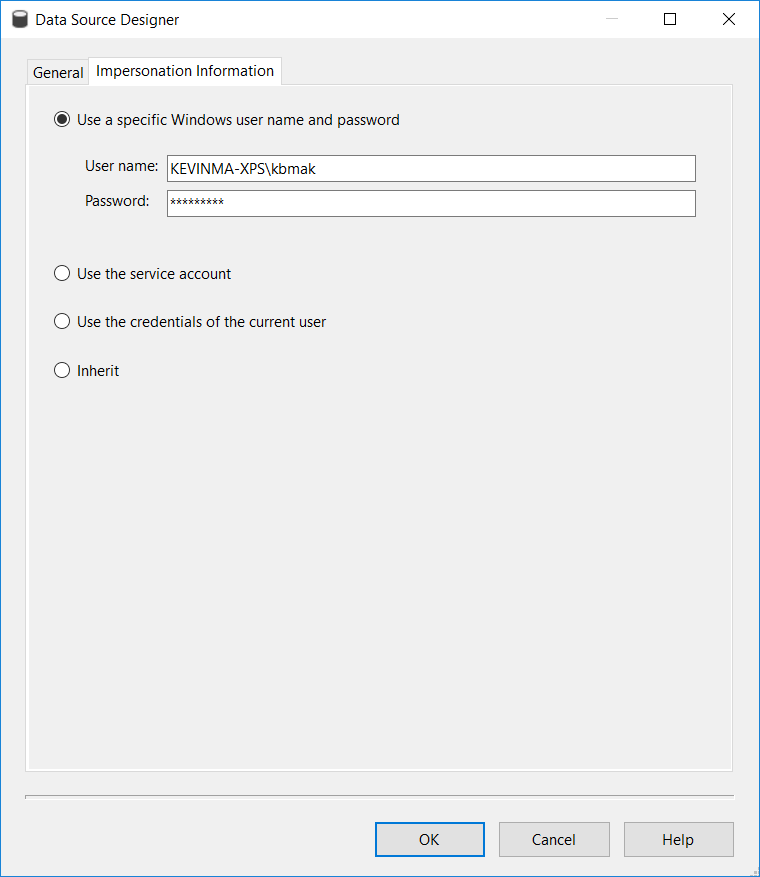
# Time for action – viewing the cube in the browser

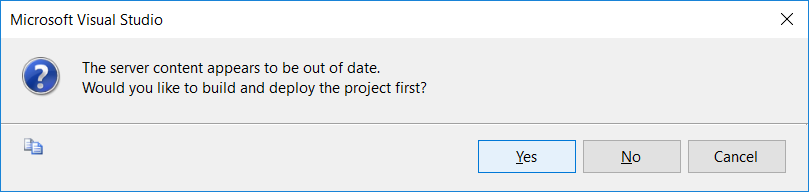
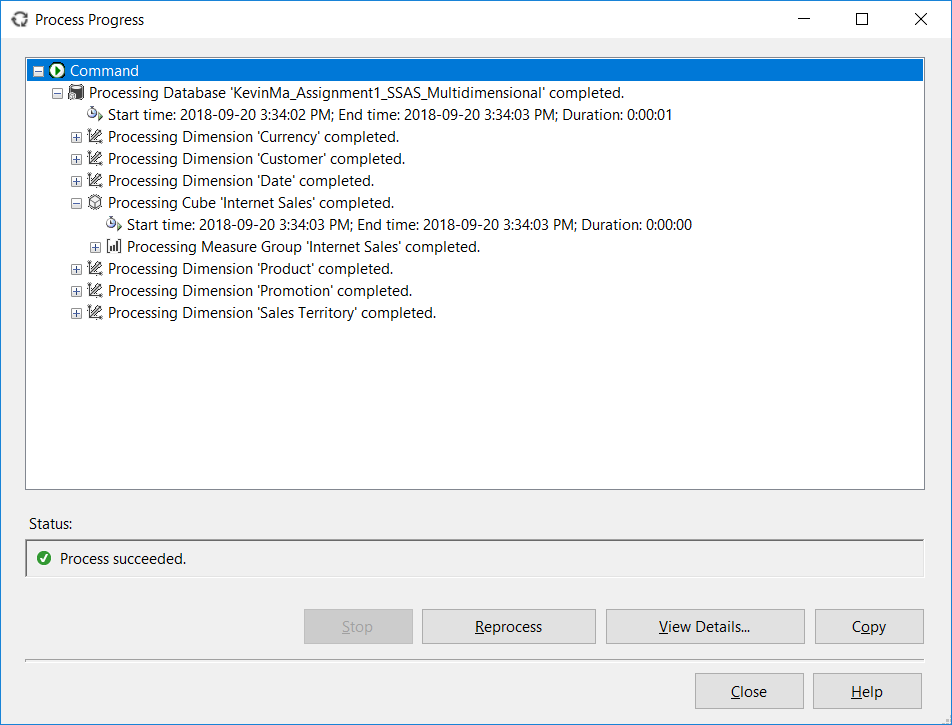
1. Change from localhost to ComputerName\MsSqlServerInstanceName

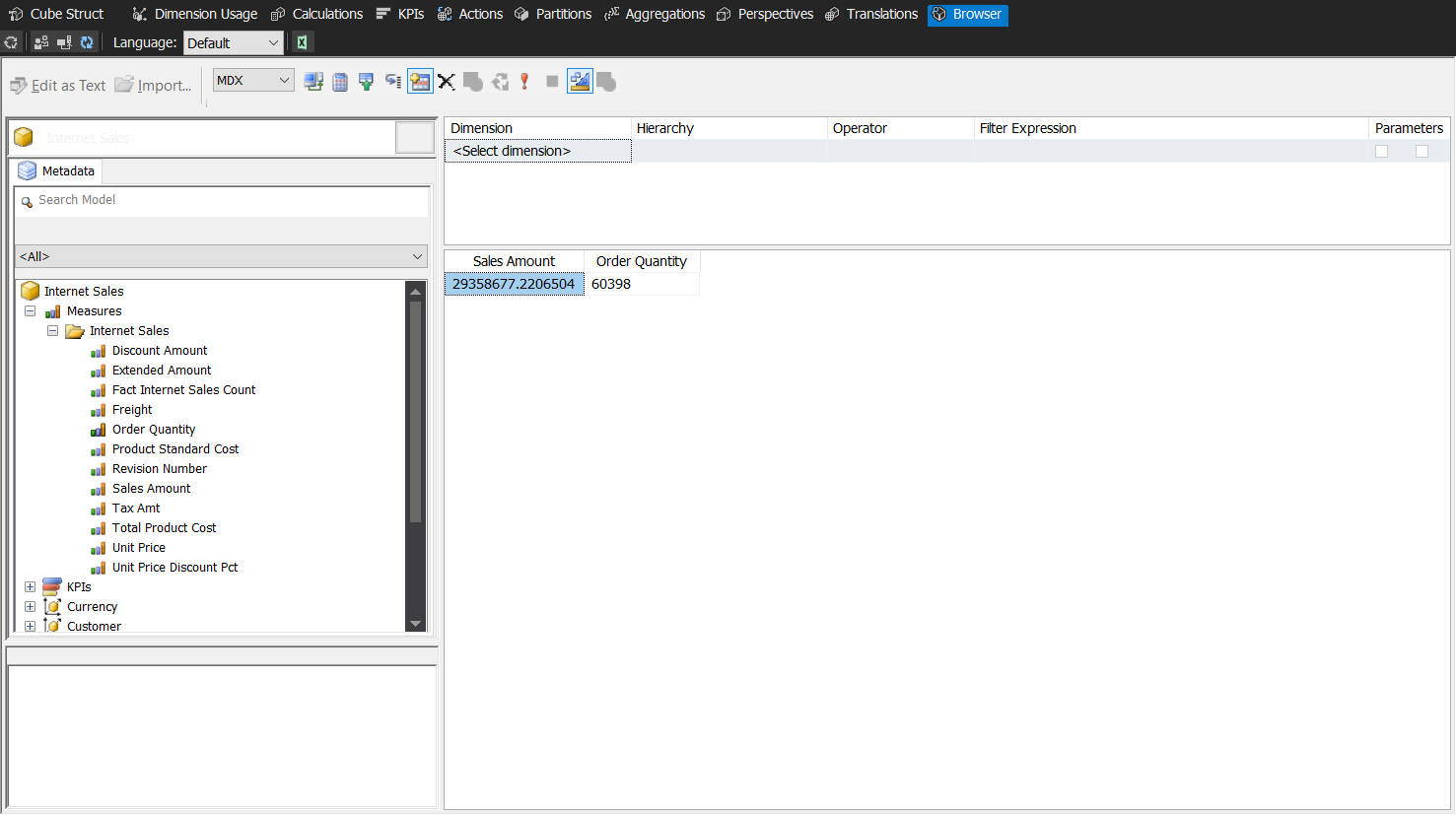
to



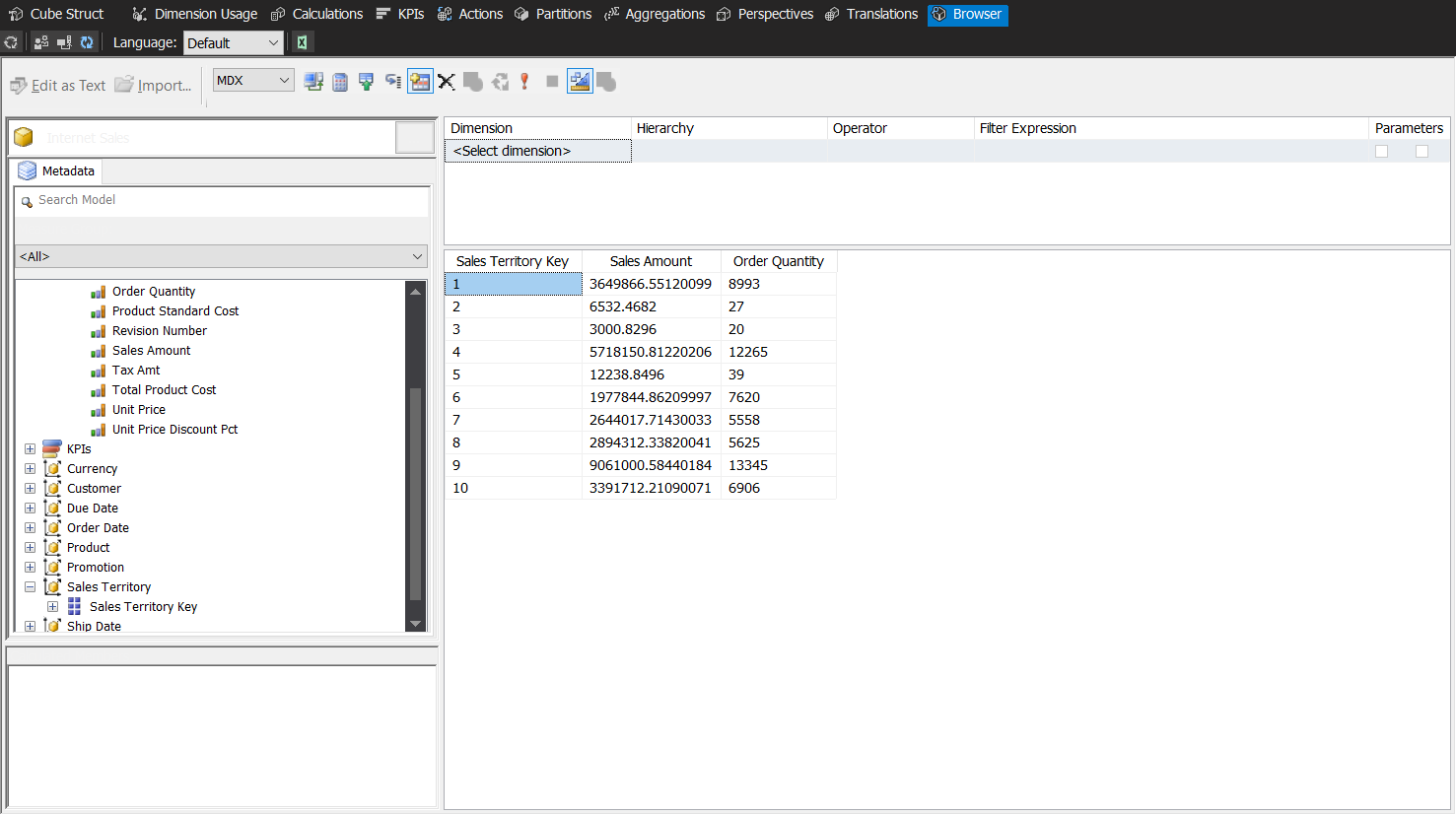
1. Change from service account in data source to



1. 
2. 
3. Go to Browser tab, which is the last tab in the cube designer
4. To see grand total of Sales Amount and Order Quantity calculated from the cube, drop-and-drag Sales Amount and Order Quantity measures from the Metadata pane on the left side to the pane on the right side and press execute query:



1. Slice and dice the Sales Amount by Sales Territory by dragging the Sales Territory Key dimension from the Metadata pane and re-executing the query on the right:



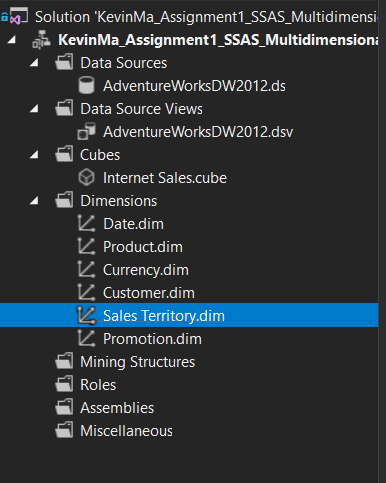
## What did we learn

Here we learned that we can view the cube in the browser tab. Viewing the cube in the browser of SSDT helps SSAS developers check the frontend of the cube from the user’s perspective. We also learned how to view grand totals of measures, and how to slice and dice dimensions.

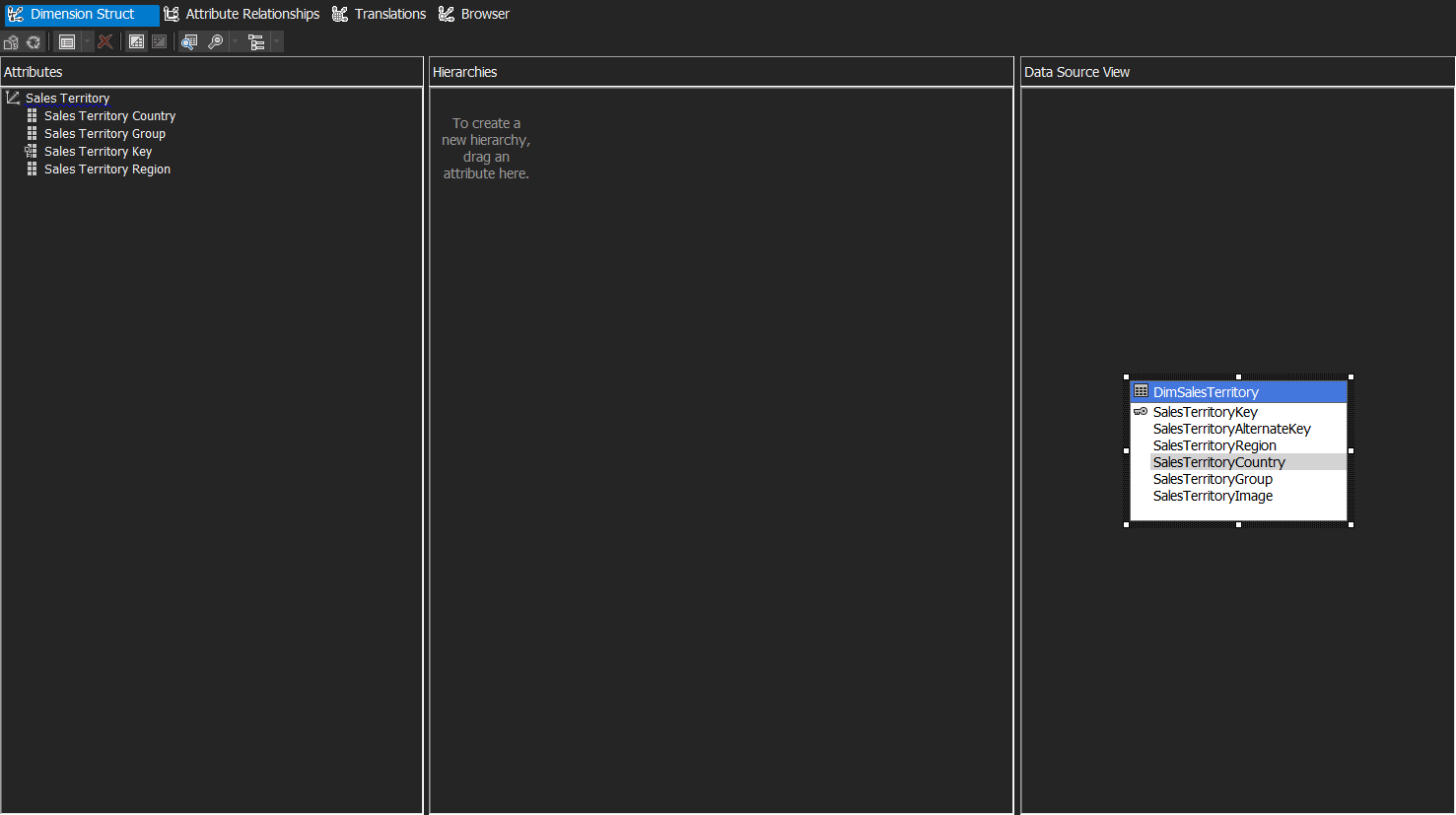
While this may be useful for the SSAS developers, this does not make much sense to the Business Users. So in the next section we will be bringing more descriptive information as additional attributes and better designing the output to make it more useful.

# Time for action – using the Dimension Designer

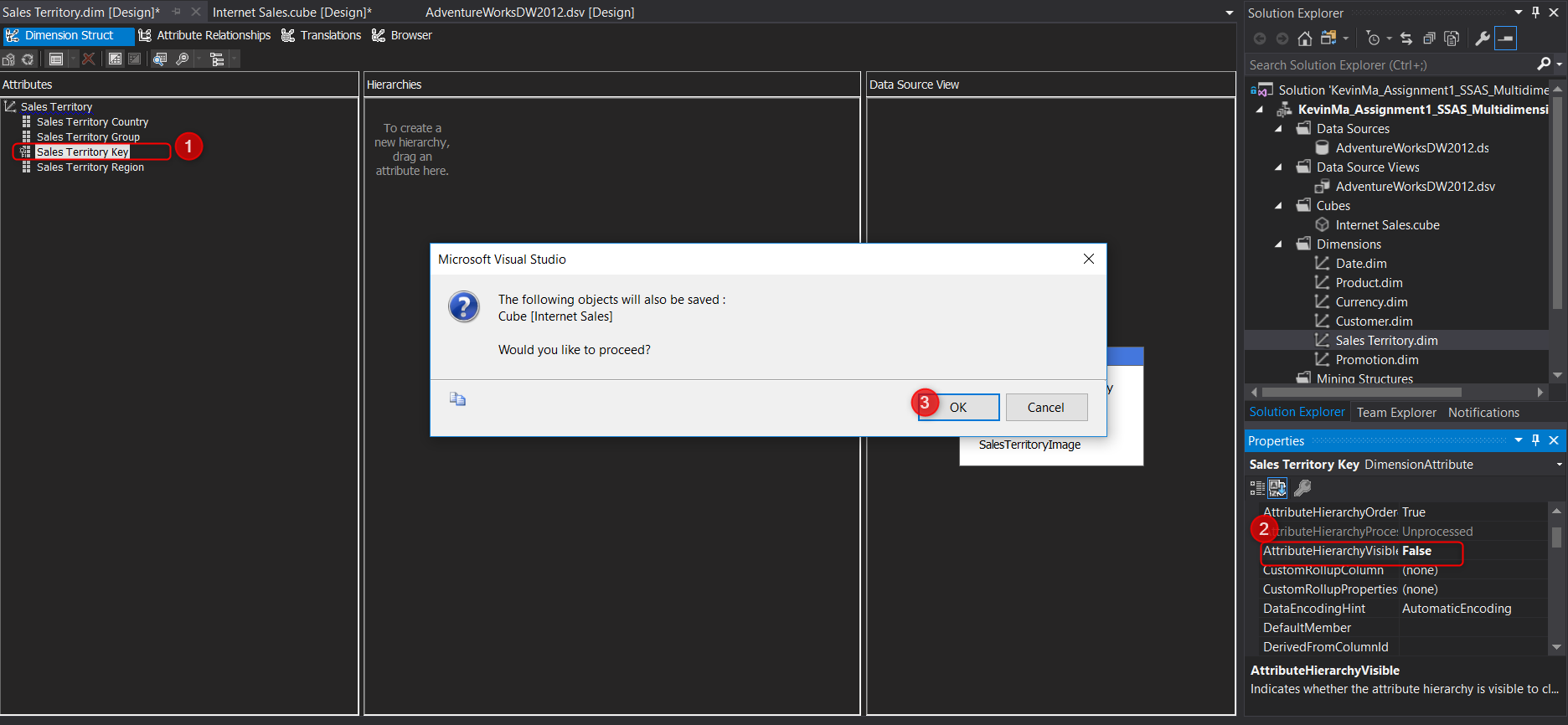
1. Open the Dimension Designer by double clicking on the Sales Territory dimension in the Solution Explorer



1. Drag-and-drop Sales Territory Region, Sales Territory Group and Sales Territory Country from the Data Source View to the Attributes panel



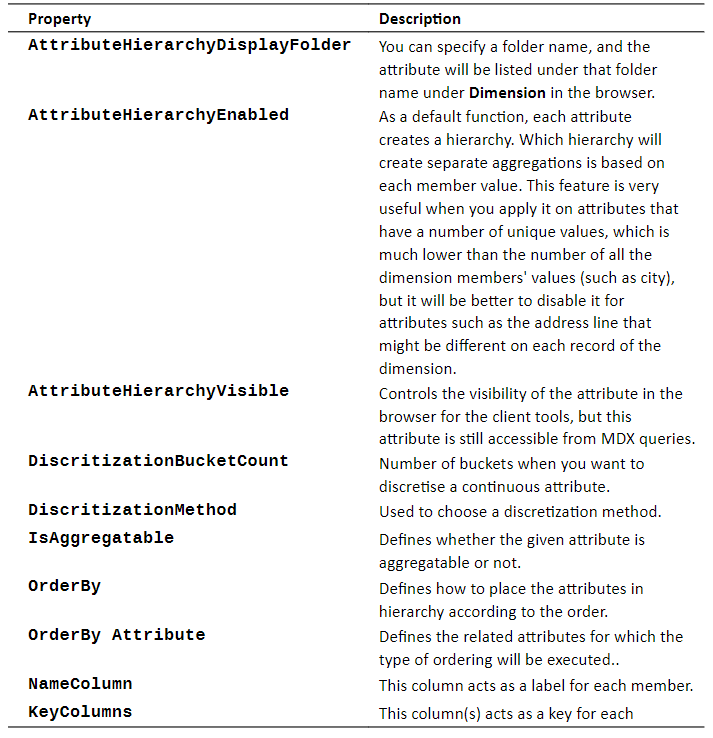
1. Select Sales Territory Key in the Attributes pane, and then in the Properties window, change the AttributeHierarchyVisible property of this attribute to false.



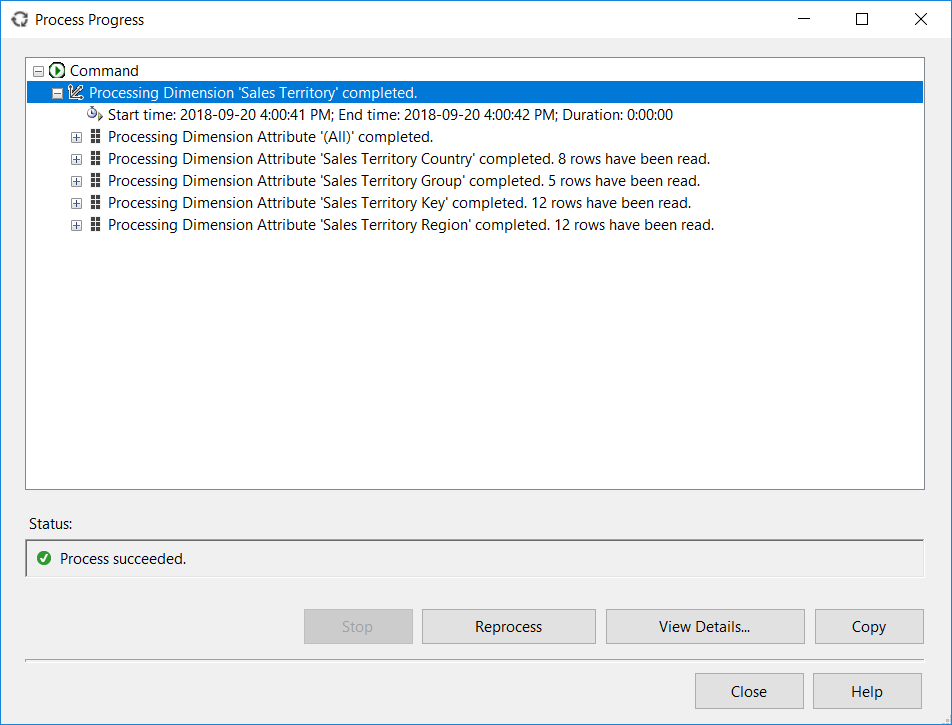
This property:



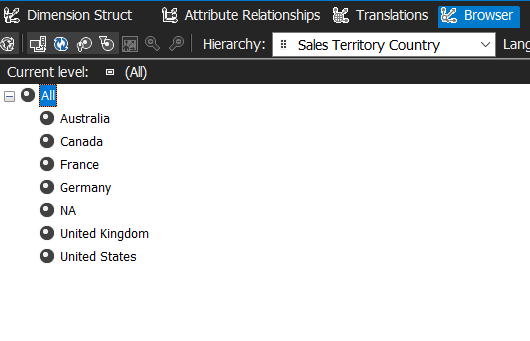
Some other useful properties of attributes are:



1. Re-process the dimension



1. Go back to the Browser tab and you should now be able to see the values for the new attribute selected (e.g. Sales Territory Country)



1. To see the changes in the cube, we need to process and deploy again (NOTE: Every time we make a change to the metadata of the SSAS cube, dimension or measures, we need to deploy the changes and process again to see the changes).

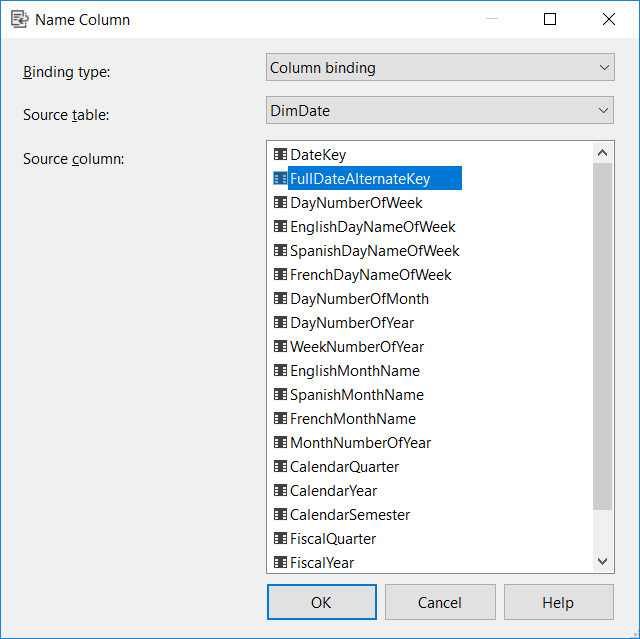
## What did we learn

We learned that we can modify dimensions using the Dimension Designer. This allows us a the developers to bring more descriptive information as additional attributes and better designing the output to make it more useful for the business users.

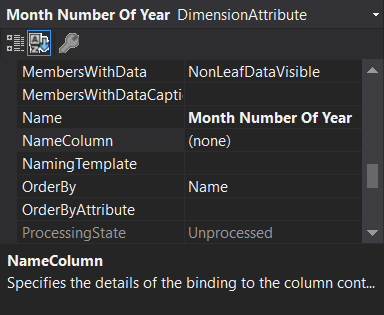
# Time for action – change the order of the Month attribute

The date dimension is currently an integer in the format of YYYYMMDD. This is not well formed to a business user, so we want to change this format and add two other attributes for the years and month.

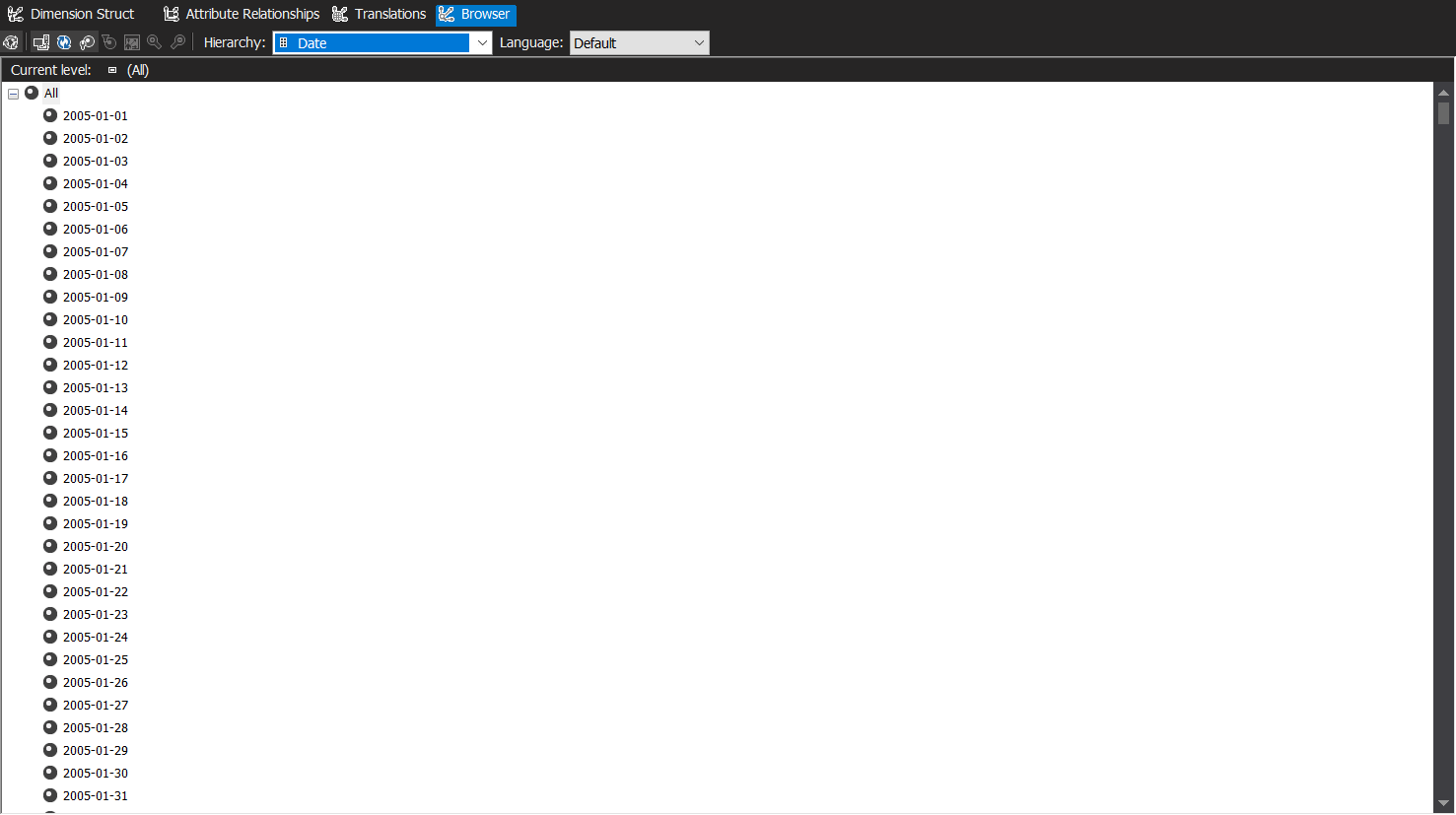
1. Open the Dimension Designer for the Date dimension by double clicking on the Date dimension in the Solution Explorer.
2. Go to the Dimension Structure tab, select DateKey in the attributes pane, and then click on the ellipsis button of the NameColumn property in the Properties window.
3. Change the source column from DateKey to FullDateAlternateKey

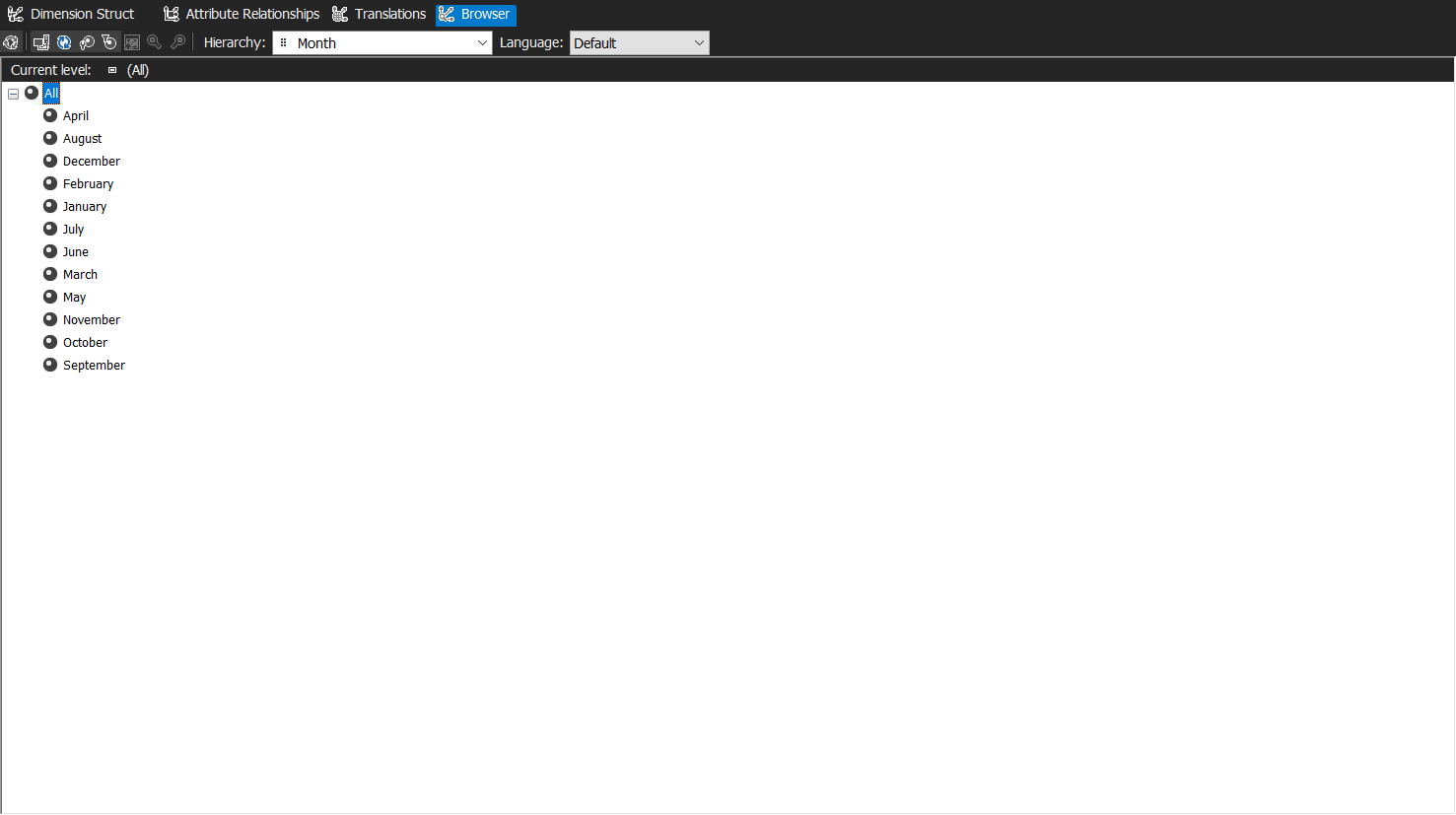


1. Rename the Date Key attribute to Date
2. Drag-and-drop Calendar Year and Month Number of Year from the Data Source View into the Attributes pane
3. Change the NameColumn for the Month Number of Year to show the English month names for the label value and rename the attribute to simply be called Month

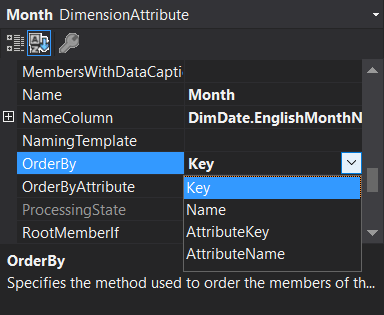
1. After we Process the Project, we can see the changes we made in the Browser Tab



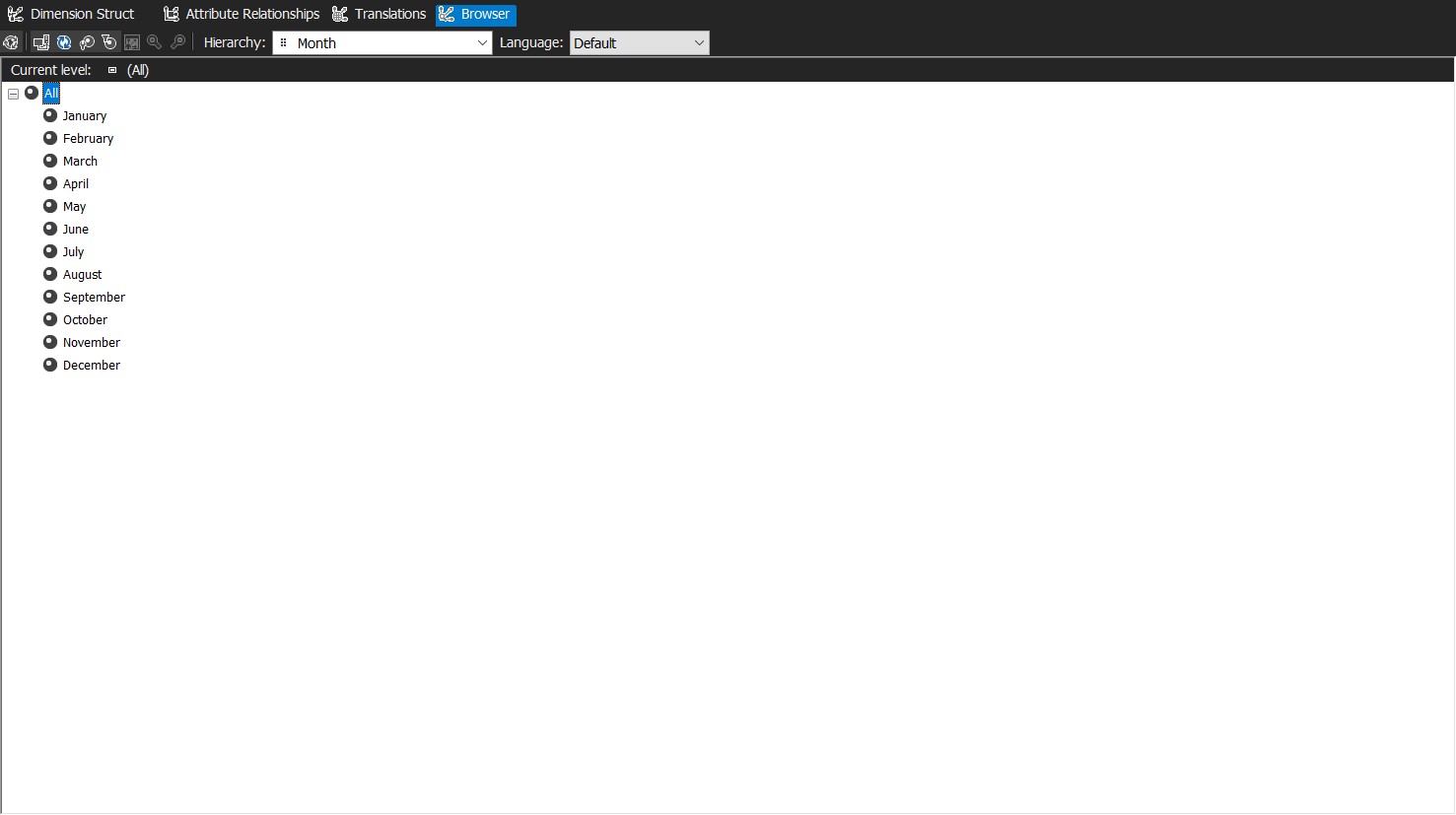


However, the Month names are not in their correct order

1. Go back to the Dimension Structure tab and change the Month attribute’s OrderBy Property to ordering by key instead of name:



1. After processing the changes, we can see that the month names are in their correct order in the Browser tab

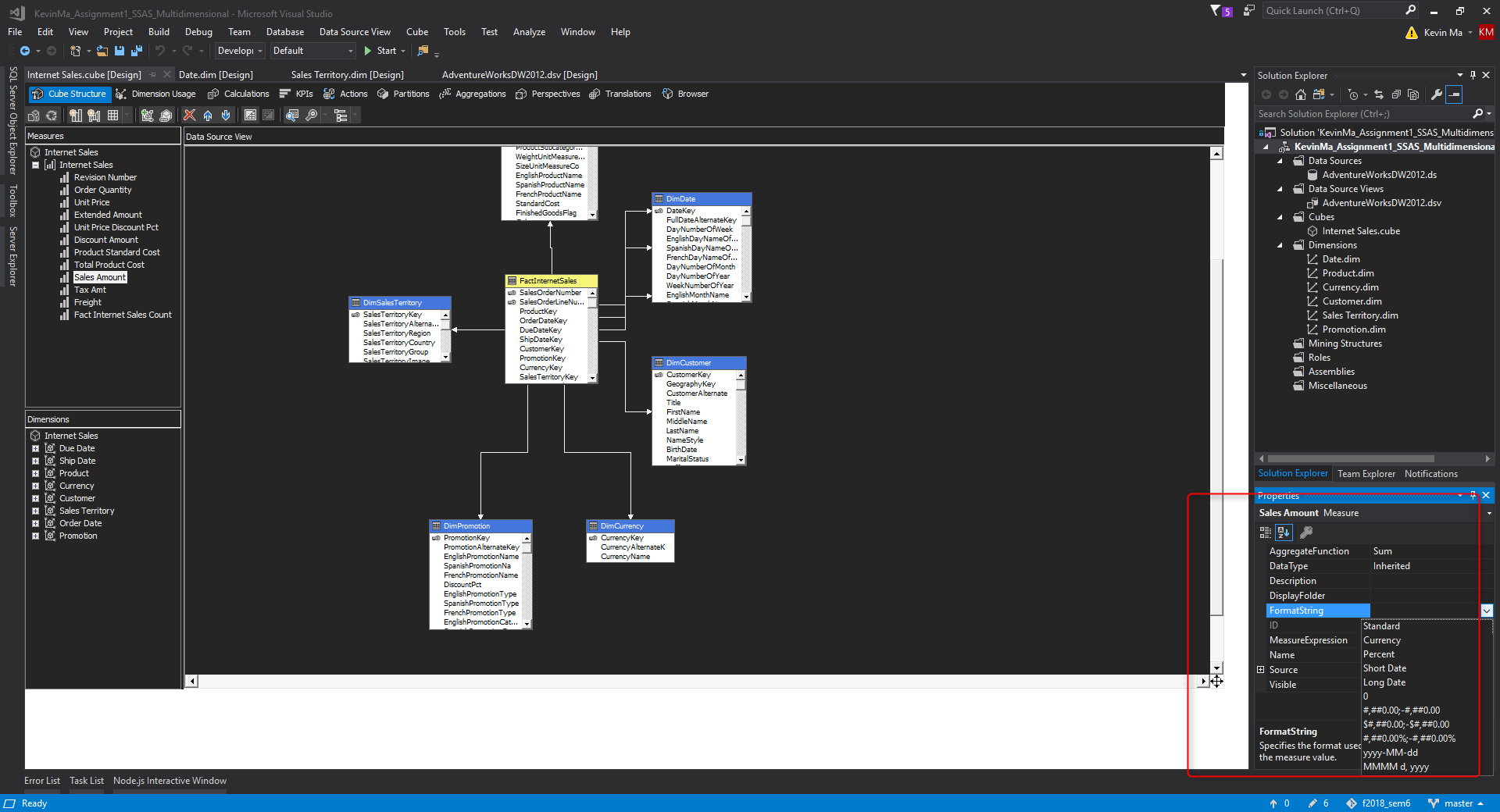


## What did we learn

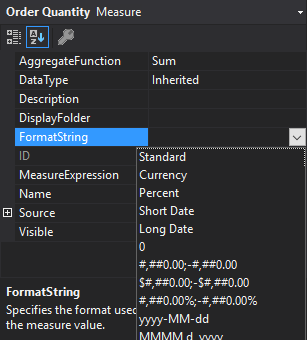
Here we learned how to add attributes to existing dimensions. This allows us to add more descriptive fields to a dimension which would allow for more polished displays. We can also change the ordering of the attributes when a user views the dimension.

# Time for action – modifying the measure properties

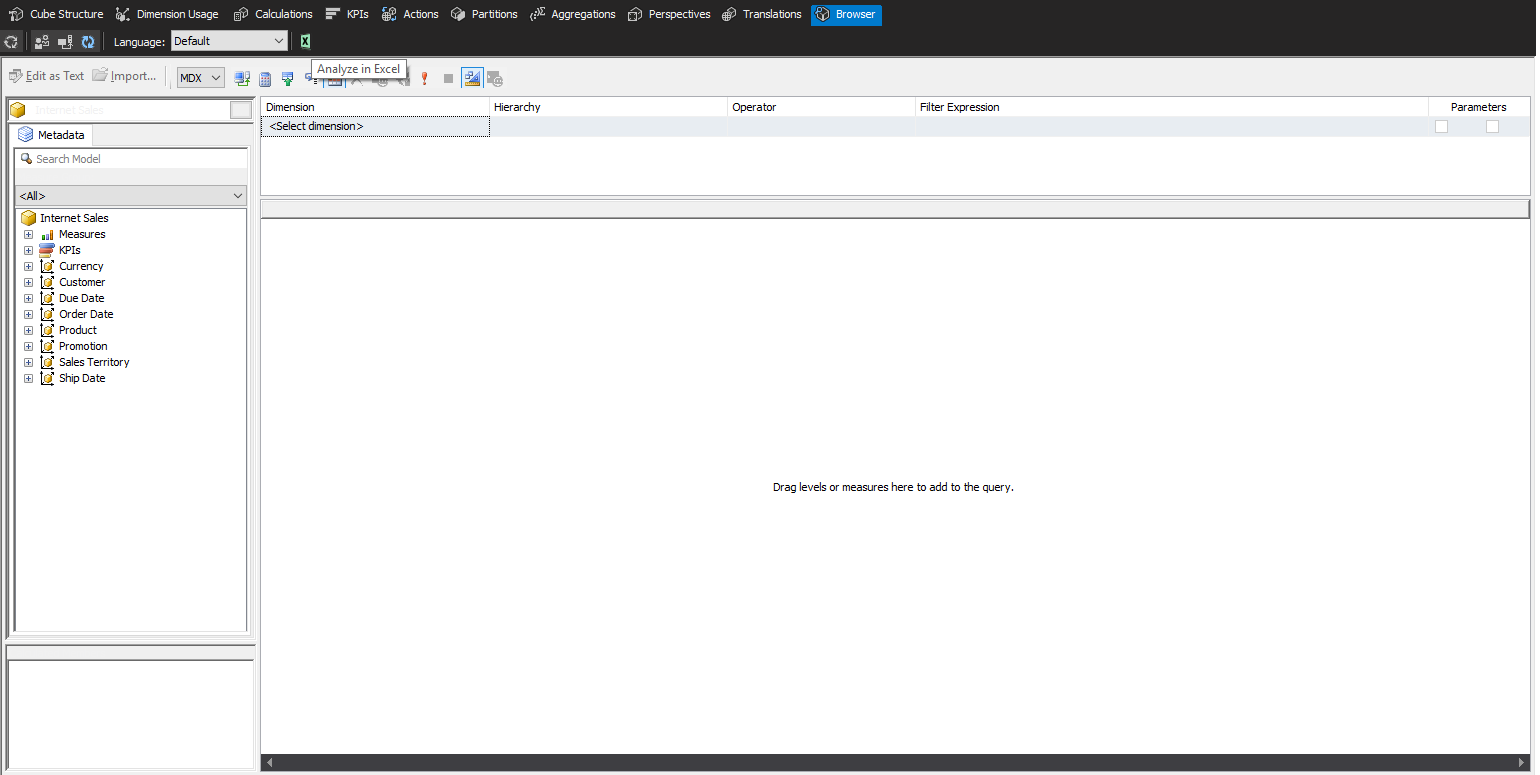
1. In SSDT > Cube designer > Cube structure tab > Sales Amount measure > Properties, change FormatString property of this measure to Currency

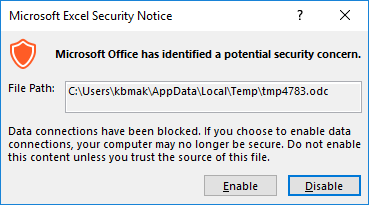


1. Similarly, change the FormatString of the Order Quantity measure to #,##:



1. Go to the Excel browser by clicking on the option in the browser tab. Click Enable when prompted for enabling content regarding data connections:





1. Choose Sales Amount and Order Quantity, and from Order Date, choose Calendar Year:



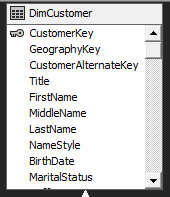
## What did we learn

Here we learned to change the display format of measures by configuring the **FormatString** property of measures. Another property we can change is the **AggregateFunction** property to things such as **Sum, Average, Count**, and many others. We also learned that we can use Microsoft Excel as a browser for the Analysis Service Cube instead of simply viewing it through SSDT/Visual Studio. This is a great addition to our toolset as business users are usually much more accustomed and familiar with using Microsoft Excel and will be easily accessible on every system that has Microsoft Office installed. Furthermore, Excel has many native features for filtering and charting which makes it easier for users to work with the data.

# Time for action – creating a Named Calculation

Sometimes there may be requirements to change the Data Source View (DSV). For example, we may want to add a calculated column in the DSV with SQL commands and functions, create relationships between views or write queries to create DSV tables based on those queries. We can do all these modifications within the DSV.

In our DSV, we want to create a calculated column for full name because the customer dimension only contains the first and last name separately:

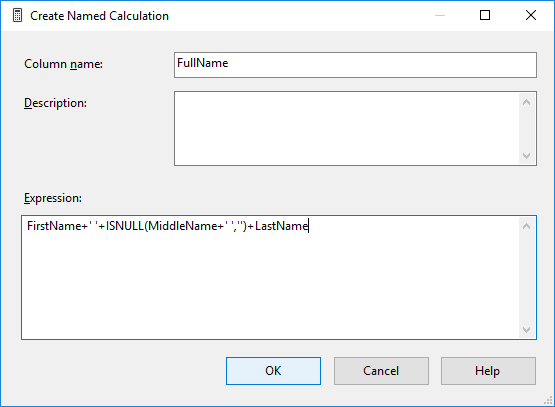


1. In the **Adventure Works DW2012 DSV** designer, right click on the Customer dimension and select **New Named Calculation**:

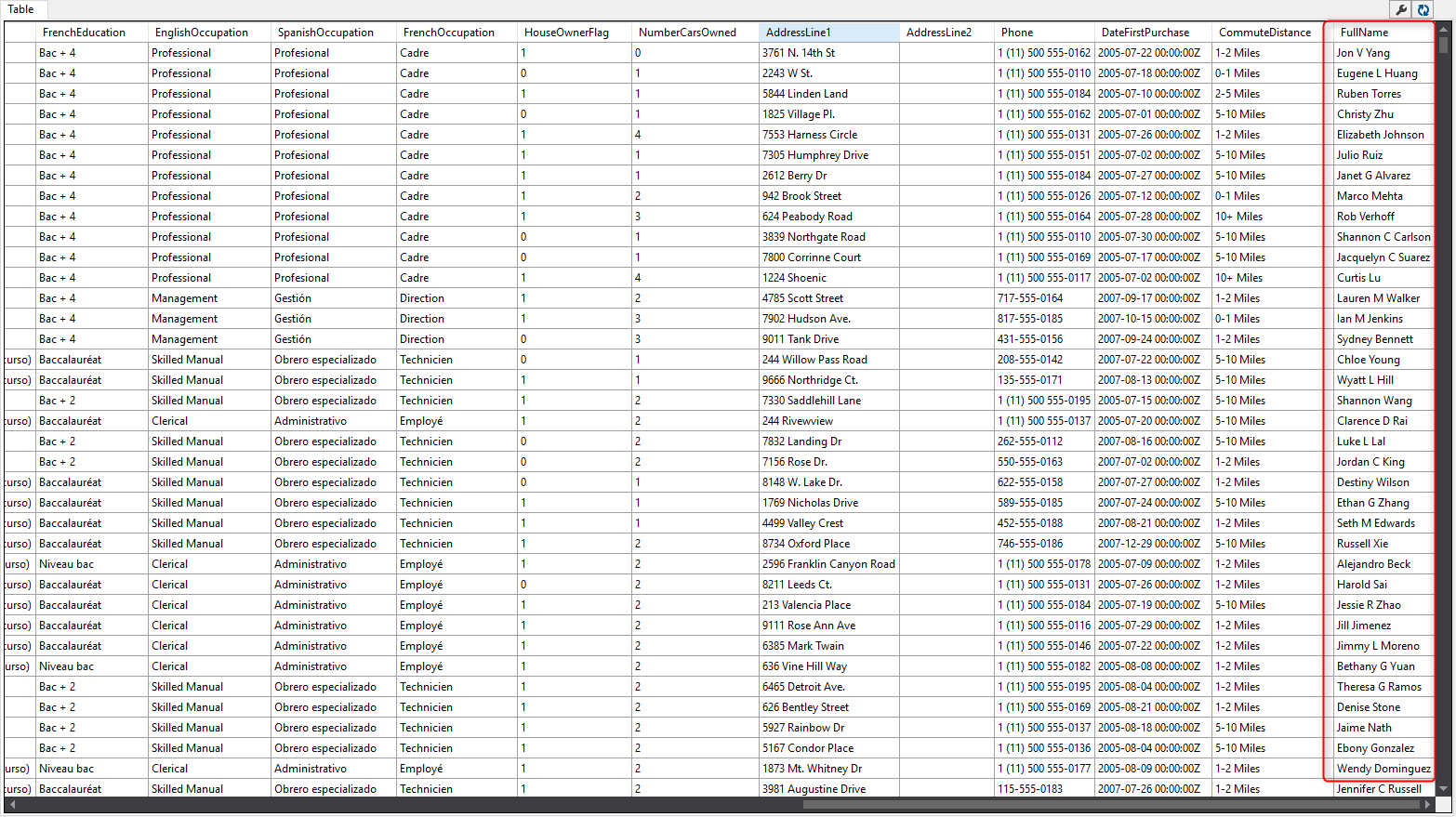


1. Create the **FullName** column by providing the following expression:

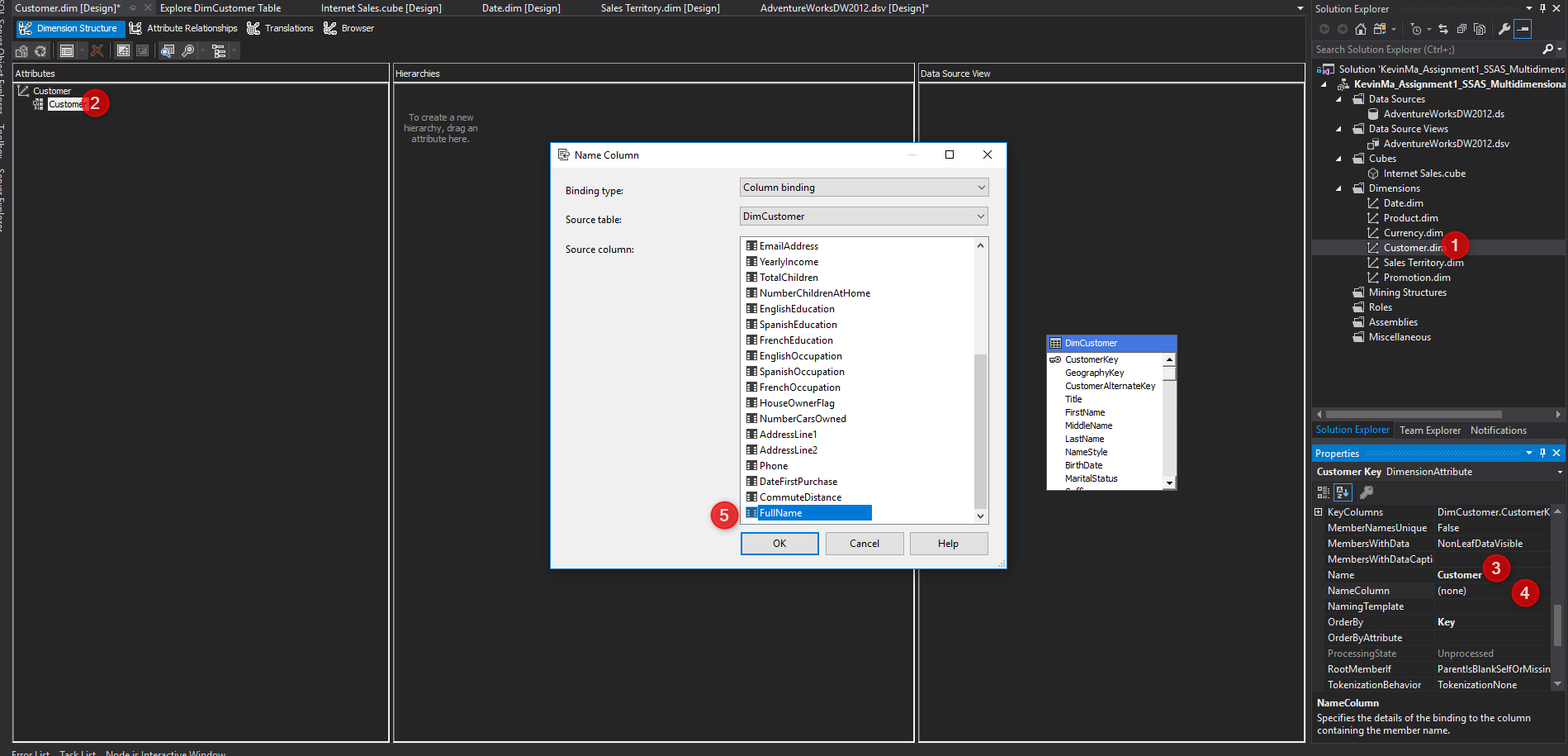
**FirstName+’ ‘+ISNULL(MiddleName+’ ‘,’’)+LastName**



1. You can see the calculated column with the provided sample data by right clicking on **DimCustomer** and selecting **Explore Data**:



1. Go to the **Cuustomer** Dimension Designer, select the **Customer Key** attribute, and change the **NameColumn** to **FullName** from **DimCustomer**. Also, rename the **Customer Key** to **Customer** in the **Name** property.



1. Re-deploy and process before going to the Browser tab. Now you will see that the full name of the customer appears as **Customer Values** in the browser:

|  |  |
| --- | --- |
| Past State | Current State |
|  |  |

## What did we learn

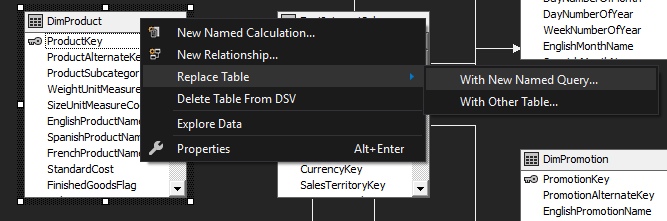
Named Calculations are very useful if we want to add calculated columns to the underlying database but lack the required permissions (i.e. read-only permissions on the database). We can create Named Calculations by defining them in the **expression** area. The language of the expression used is based on the underlying database (e.g. Oracle environments, SQL Server, etc.). We also learned that we can check the result of the execution of a Named Calculation with the **Explore Data** option.

# Time for action – using a Named Query

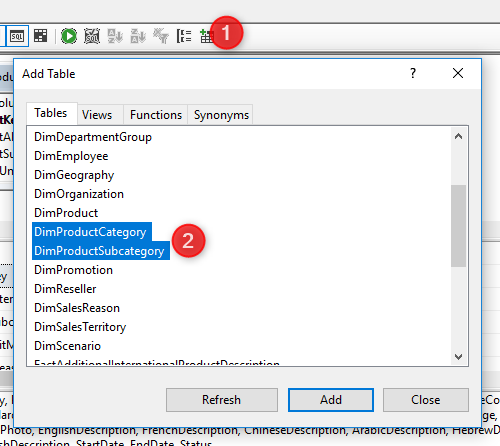
Sometimes we need to write queries instead of simply reading the native tables and views from a database. If we have read-only access to a database, we do this by using Named Queries within SSDT. A Named Query is a query that runs on the database and its result will be shown as a table in DSV.

Let’s do this to bring the product category and subcategory along with the product table columns:

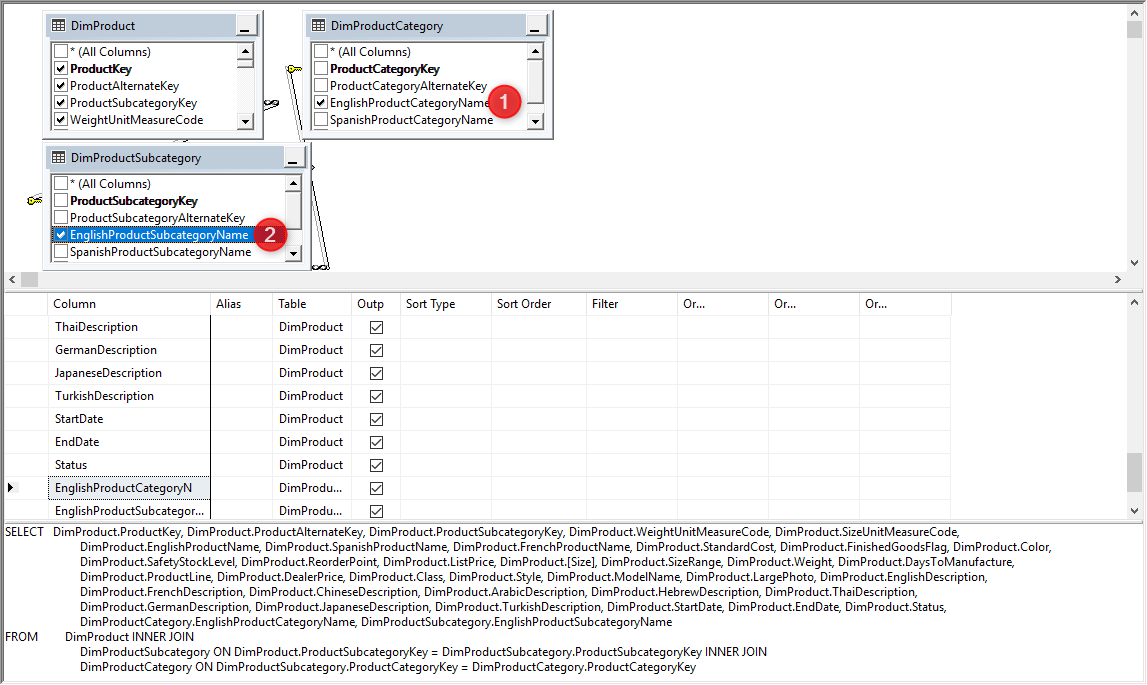
1. In the **Data Source View** designer, right-click on the **DimProduct** table and under **Replace Table**, choose **With New Named Query**



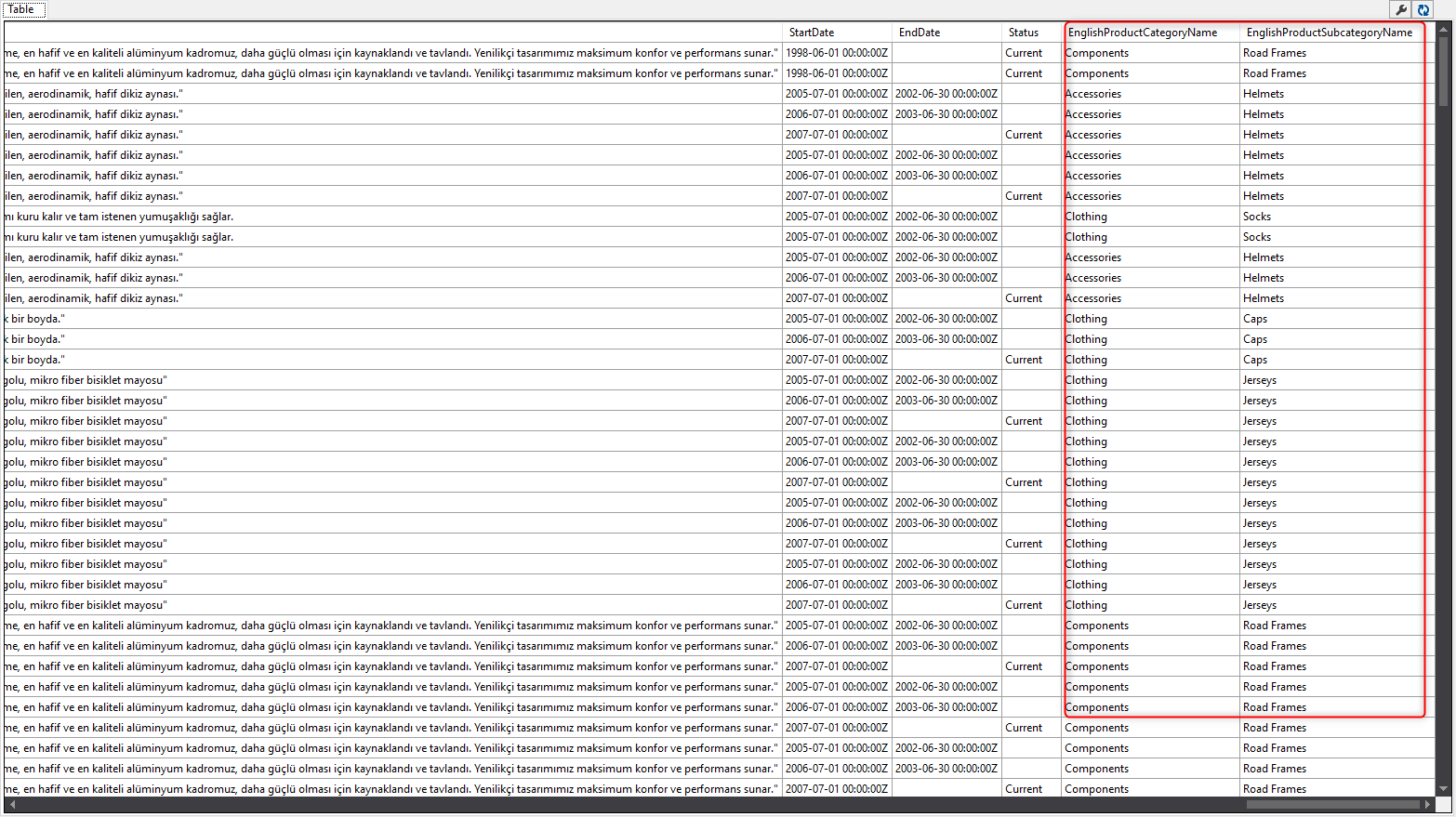
1. In the **Create Named Query** window, add the **ProductCategory** and **ProductSubCategory** tables, and the designer will automatically write joined statements:



1. Select **EnglishProductCategoryName** and **EnglishProductSubcategoryName** to add the columns to the Named Query:



1. Explore the data of the **DimProduct** Named Query to see the two new columns from the ProductCategory and ProductSubcategory tables:



## What did we learn

We were able to create customized queries on the database using the Named Query and query designers of SSDT. The query language used was T-SQL because the underlying database is SQL Server. This language would differ depending on the underlying database used. In the DSV, we can also create logical keys for the Named Queries/views, and create relationships between tables, views and Named Queries. These features grant us the ability to develop a powerful base for the cube.

# Time for action – adding a Fact relationship

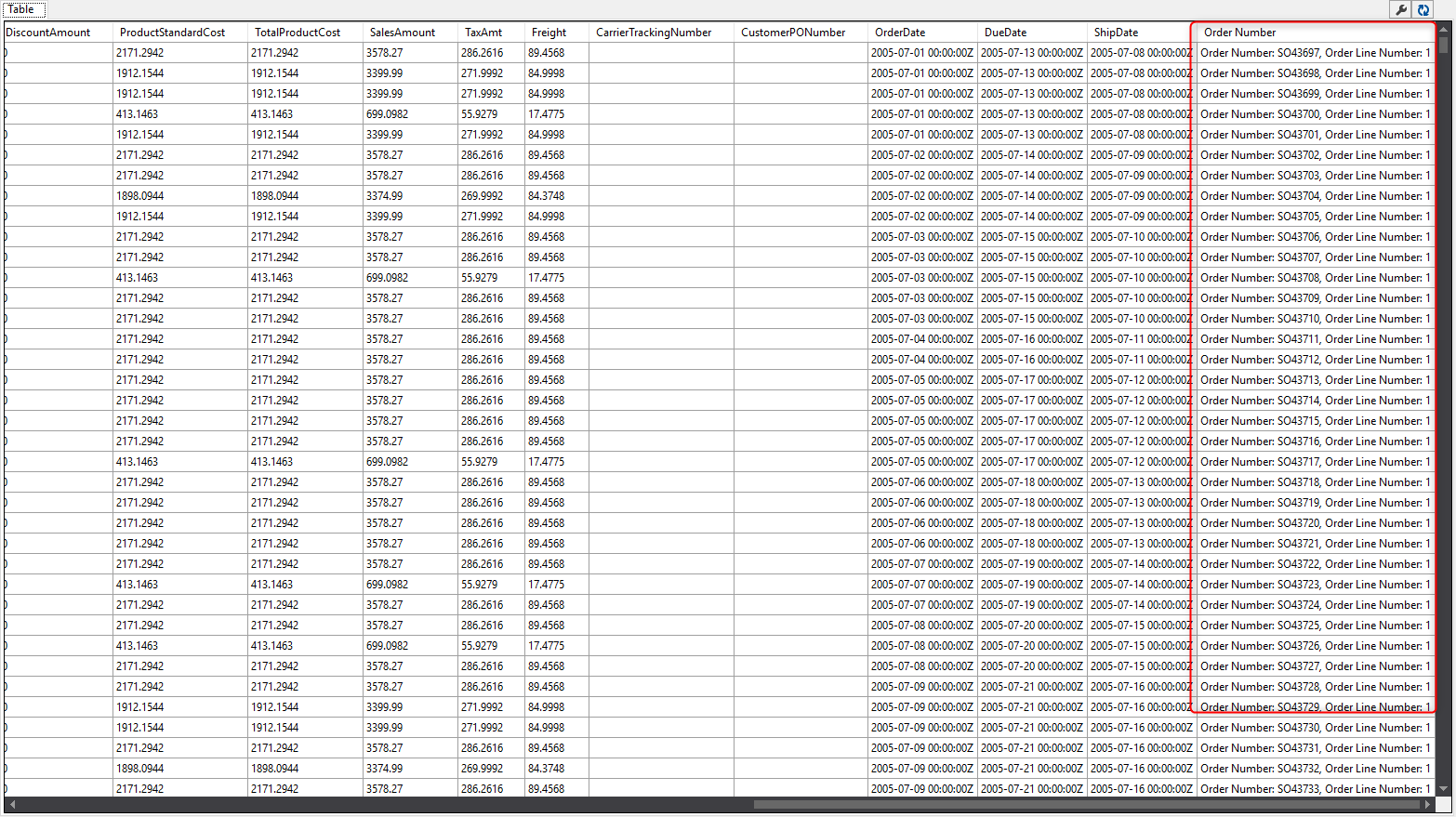
As a business requirement, we need to add the order line number as a dimension attribute and display it in the browser. In the current cube, we have an **Internet Sales** measure group but we need to fetch the order line number from **FactInternetSales** as a dimension attribute (this is a **degenerate dimension**: *According to Ralph Kimball, in a data warehouse, a degenerate dimension is a dimension key in the fact table that does not have its own dimension table, because all the interesting attributes have been placed in analytic dimensions*).

In data warehousing, a **fact table** consists of the measurements, metrics or **facts** of a business process. It is located at the center of a star schema or a snowflake schema surrounded by dimension **tables**.

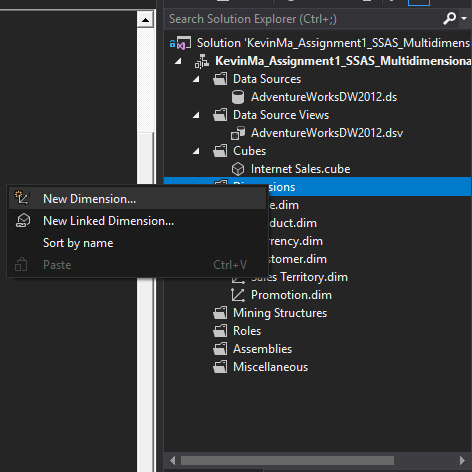
1. In the DSV designer, add a Named Calculation in the **FactInternetSales** table and name it **Order Number**. Write the following expression to calculate the concatenation of the order line and order number:

**'Order Number: '+SalesOrderNumber+', Order Line Number: '+convert(varchar(max),SalesOrderLineNumber)**

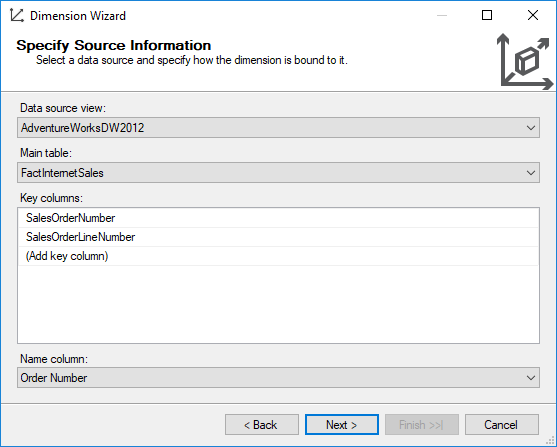
1. Explore data in **FactInternetSales** table to see the Named Calculation:



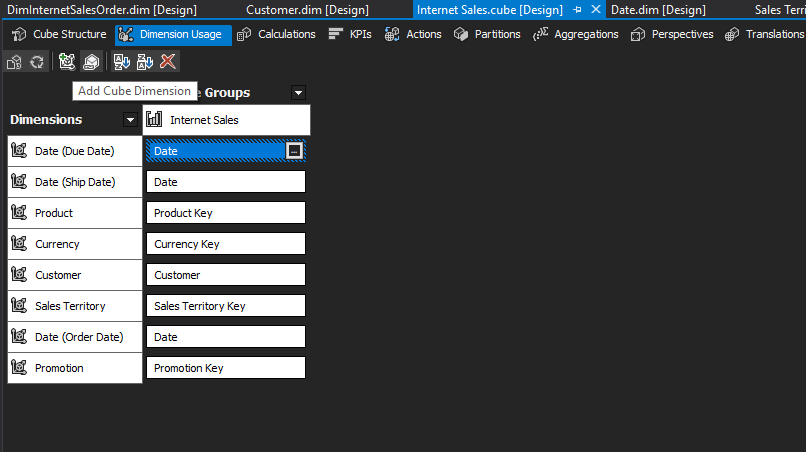
1. In the **Solution Explorer**, right-click on the **dimensions** folder to create a **New Dimension**



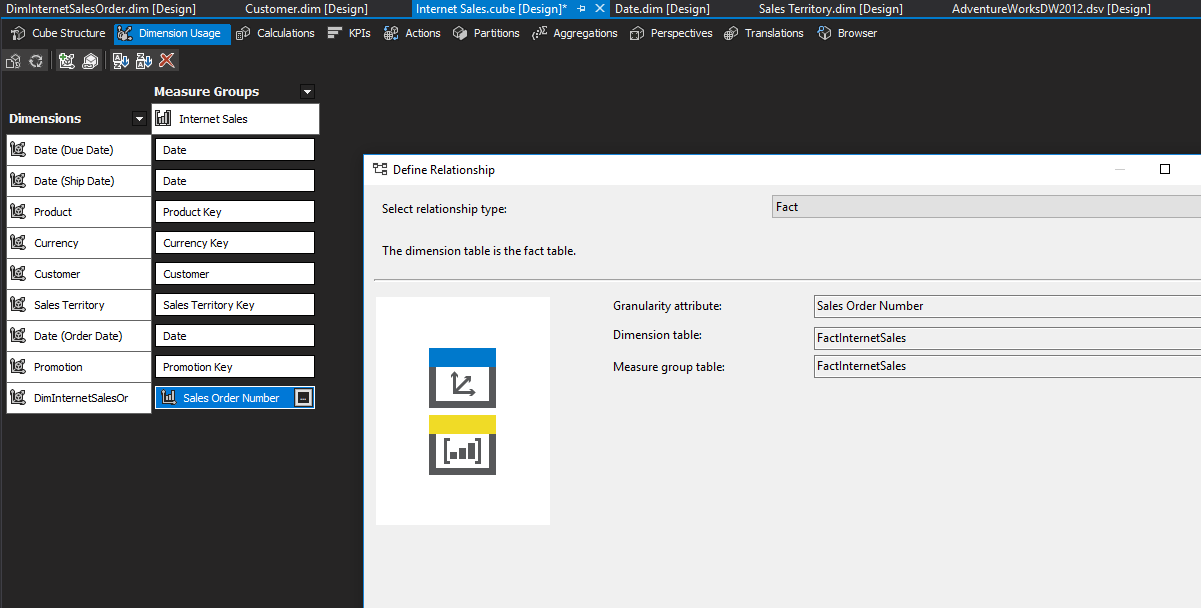
1. In the **Select Creation Method** step, choose **use an existing table**.
2. Choose **FactInternetSales** as the table, and choose **Order Number** as **Name column**:



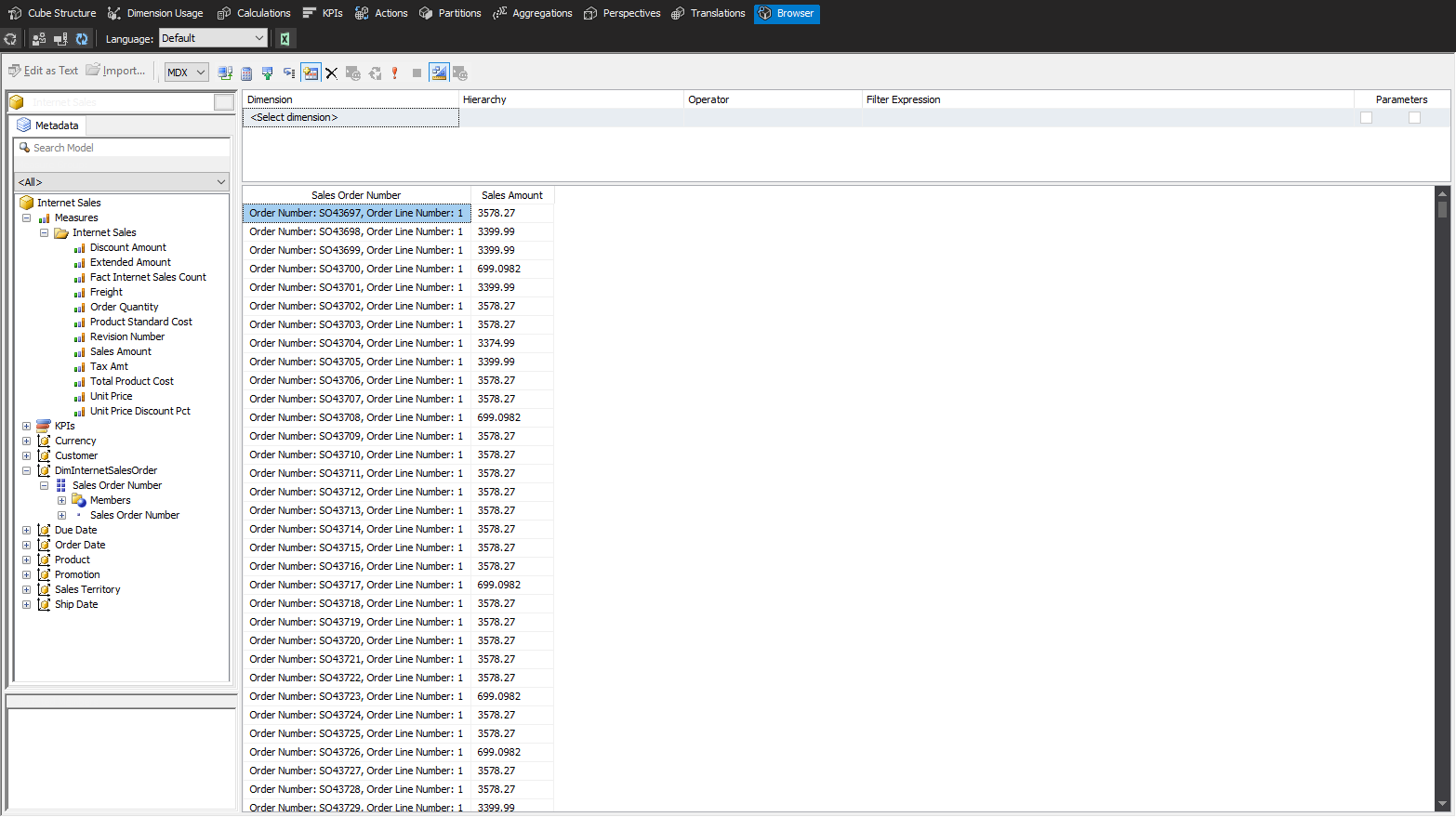
1. Uncheck all tables in the **Select Related Tables** step
2. Uncheck all the attributes and only check the **Sales Order Number** option in the **Select Dimension Attributes** step
3. Rename the dimension to **DimInternetSalesOrder** in the last step of the Dimension Wizard
4. When a new dimension is created, it will not be added by default to the cube. Go to the **Dimension Usage** tab in the cube designer and click on **Add Cube Dimension**



1. This will automatically create the relationship type of **Fact** for the dimension. The dimension will be related to the **Internet Sales** measures group based on the **Sales Order Number**:

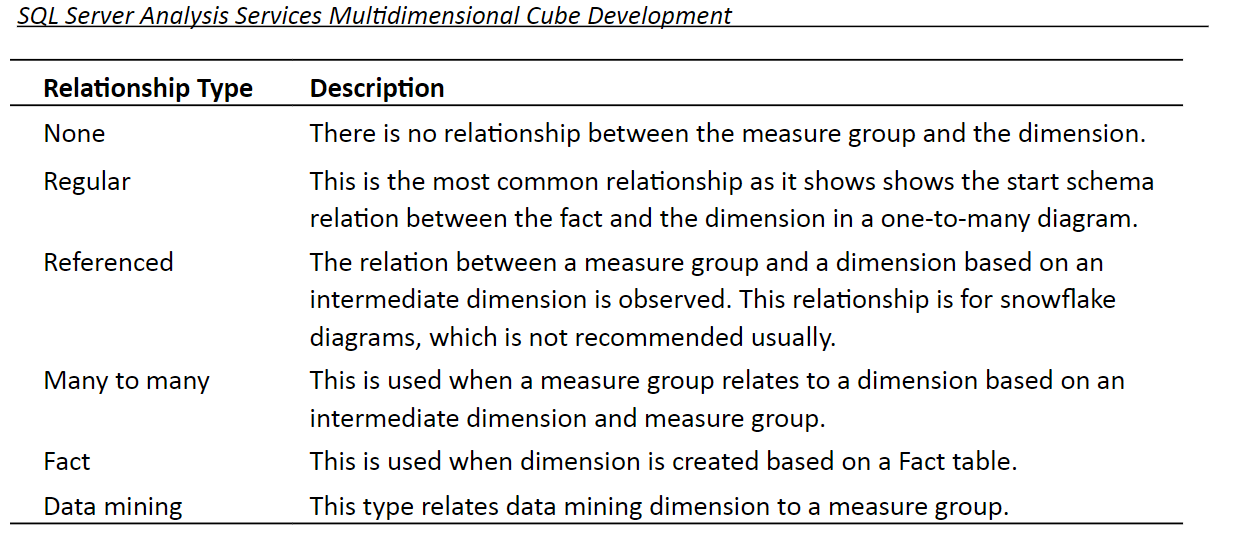


1. After deploying and processing the dimension, we can see the result in the cube Browser. We can view **Sales Order Number** as a dimension attribute and see other measures sliced and diced by this attribute:



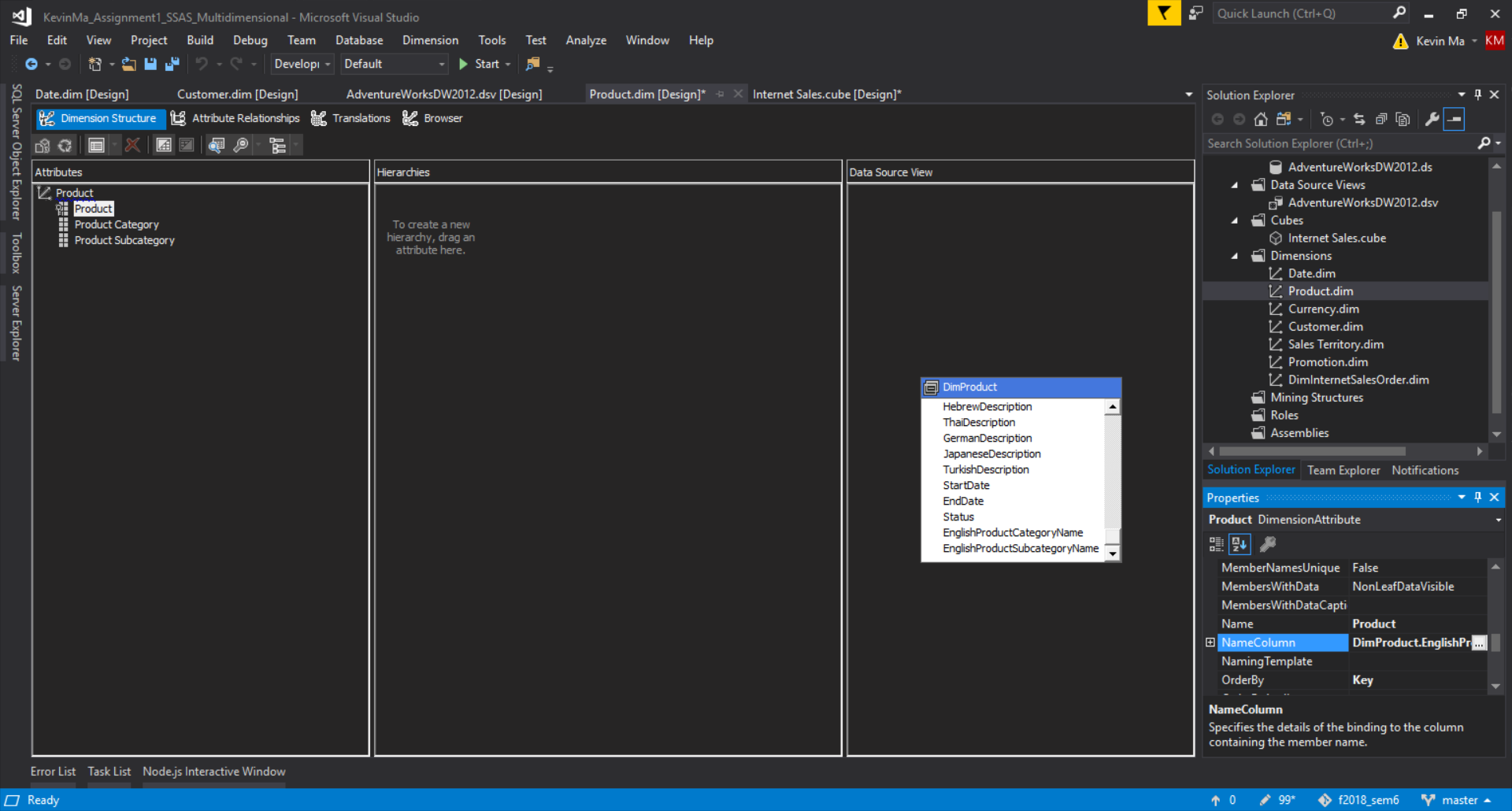
## What did we learn

One of the most important parts of modeling in SSAS Multidimensional Projects is setting up the correct relationships between measure groups and dimensions. **Dimension Usage** tab is where we can create, modify, or remove these kinds of relationships. There are different types of relationships between Dimensions and Measure Groups. In our example, we saw the **Fact** relationship, which is useful when the dimension is made up of a Fact Table. Shown below are the other types of relationships:

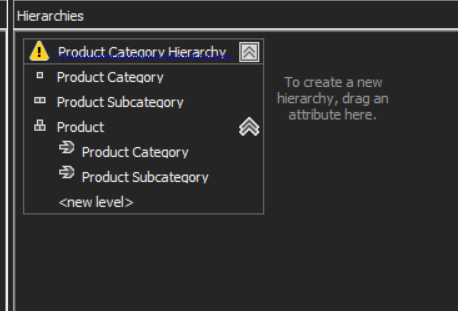


# Time for action – creating a hierarchy

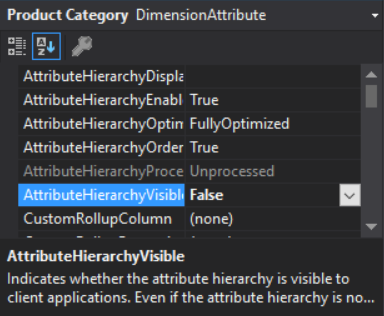
1. Go to the product’s Dimension Designer, drag-and-drop the **EnglishProductCategory** name into the **Attributes** pane, and rename it **Product Category**. Do the same for **EnglishProductSubcategoryName** and rename it **Product Subcategory**. Rename **Product Key** attribute **Product** and change its name column to **EnglishProductName**.



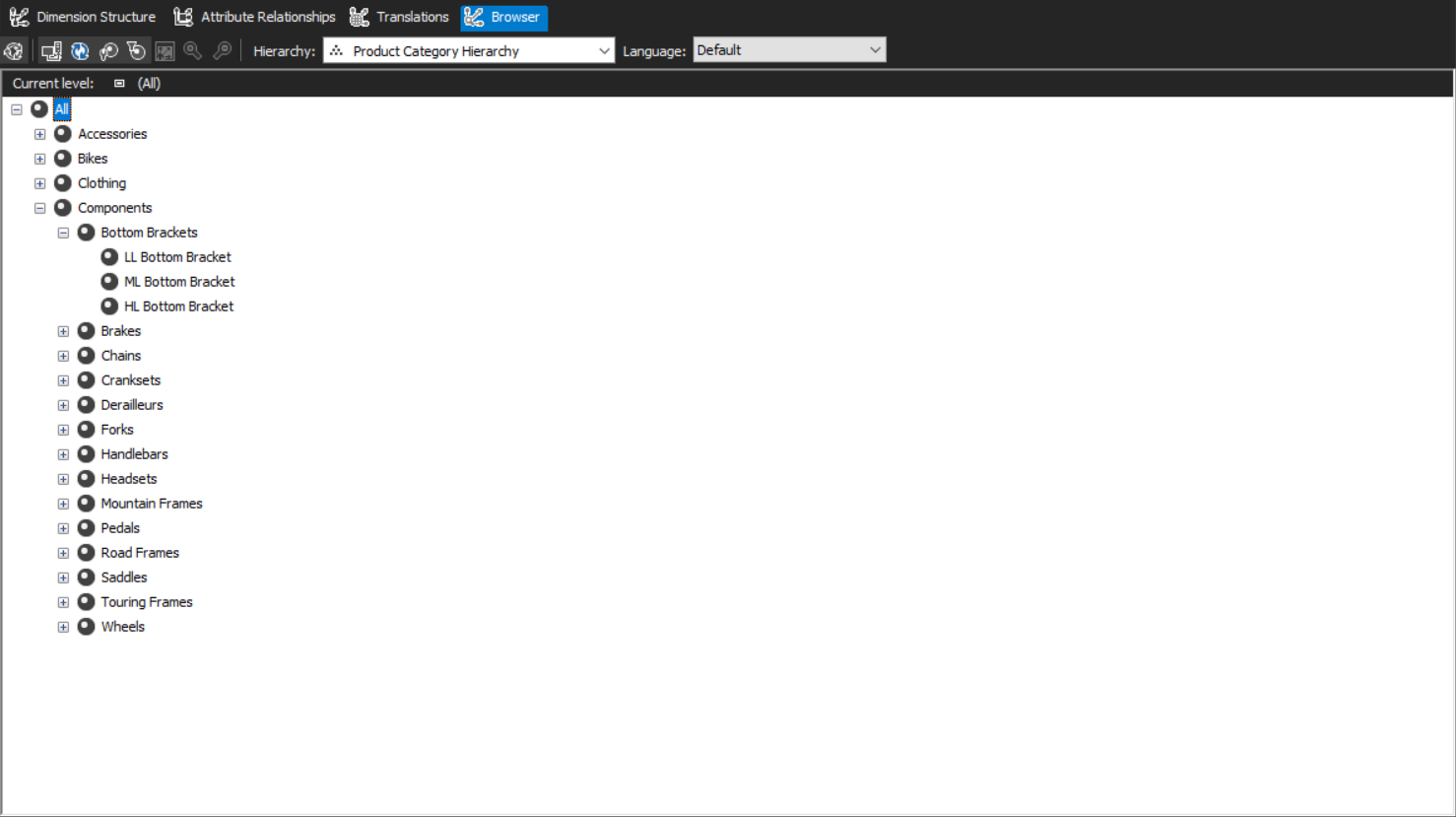
1. Drag-and-drop **Product Category** from the Attributes pane into the **Hierarchies** pane.
2. A new hierarchy will be created; rename it **Product Category Hierarchy**
3. Drag-and-drop the **Product** **Subcategory** under **Product Category** and **Product** under Product Category.



1. Change the **AttributeHierarchyVisible** option of these 3 attributes to false.



1. We can redeploy and see the updated hierarchy within the Browser tab



## What did we learn

We can custom build hierarchies to display data that makes more sense and presentability for business users through the Dimension Structure tab. We set the visibility of the attributes to false because if we already see all the data inside of our hierarchy we don’t need to be presented with unstructured redundant data. This will help reduce the user’s confusion with regards to too many attributes.

# Time for action – writing an MDX query

We often face situations where we get new business requirements after we have designed the data warehouse. Sometimes the requirements cannot be fulfilled in the data warehouse and should be fulfilled in the cube. SSAS provides a way to create calculated members based on an expression language named **Multidimensional eXpression** (MDX); this is the query language for SSAS.

1. Open SQL Server Management Studio connect to the Analysis Services instance, and click on **KevinMa\_Assignment1\_SSAS\_Multidimensional**, and then click on **New Query**.
2. In the **New Query** window, write the following expression:

*select [Order Date].[Calendar Year].members on 0 from [Internet Sales]*

You will see a list of the years, the sales amount for each year, and the grand total.

1. Now, run the following statement:

*select [Sales Territory].[Sales Territory Country].members on 0,[Order Date].[Calendar].[Calendar Year].members on 1 from [Internet Sales]*

1. Check the result and you will see the years listed in each row, the countries listed in each column, and **the Sales Amount** measure shown in each cell, which represents the sales amount of that country in that year.

## What did we learn

MDX is a query language that returns the results in a multidimensional format. When we specify members on 0/1, we are defining the axes. Members on 0 was specifying the column axis. 1 was the row axis. However, we are not only limited to two axes; we can write up to 128 axes.

The MDX select query structure consists of the **SELECT** clause on multiple axes, **FROM** clause, and the **WHERE** clause.

SELECT [<axis\_specification>

[, <axis\_specification>...]]

FROM [<cube\_specification>]

[WHERE [<slicer\_specification>]]

There are many MDX functions to work with time, hierarchies, and other common applications that are helpful when writing MDX queries.

# Time for action – calculated members

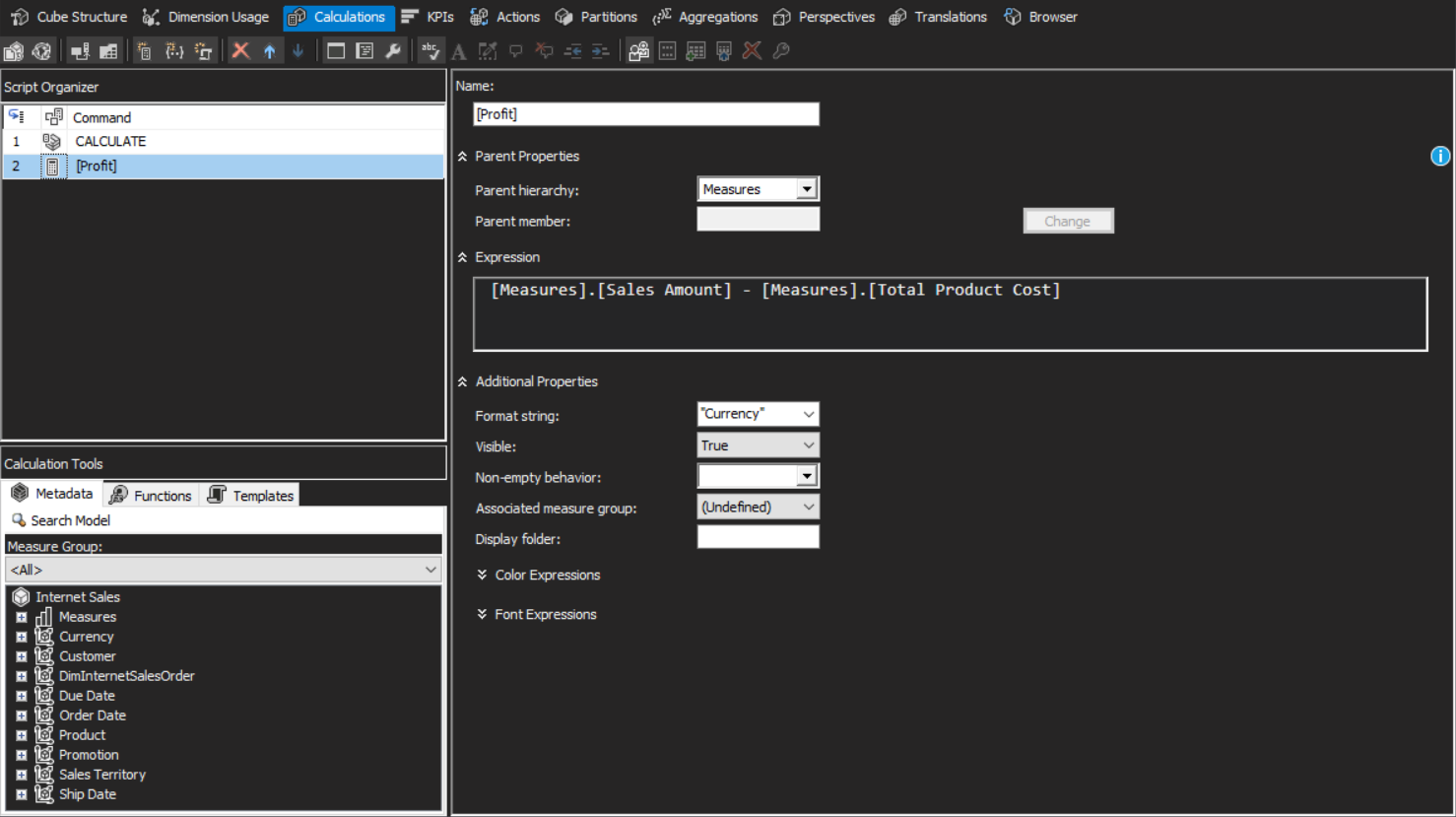
Calculated members help to perform calculation based on measures, dimension hierarchies, and MDX functions. They are very helpful for covering business requirements with the help of a cube.

Let’s try to create a new calculated member for **Profit**.

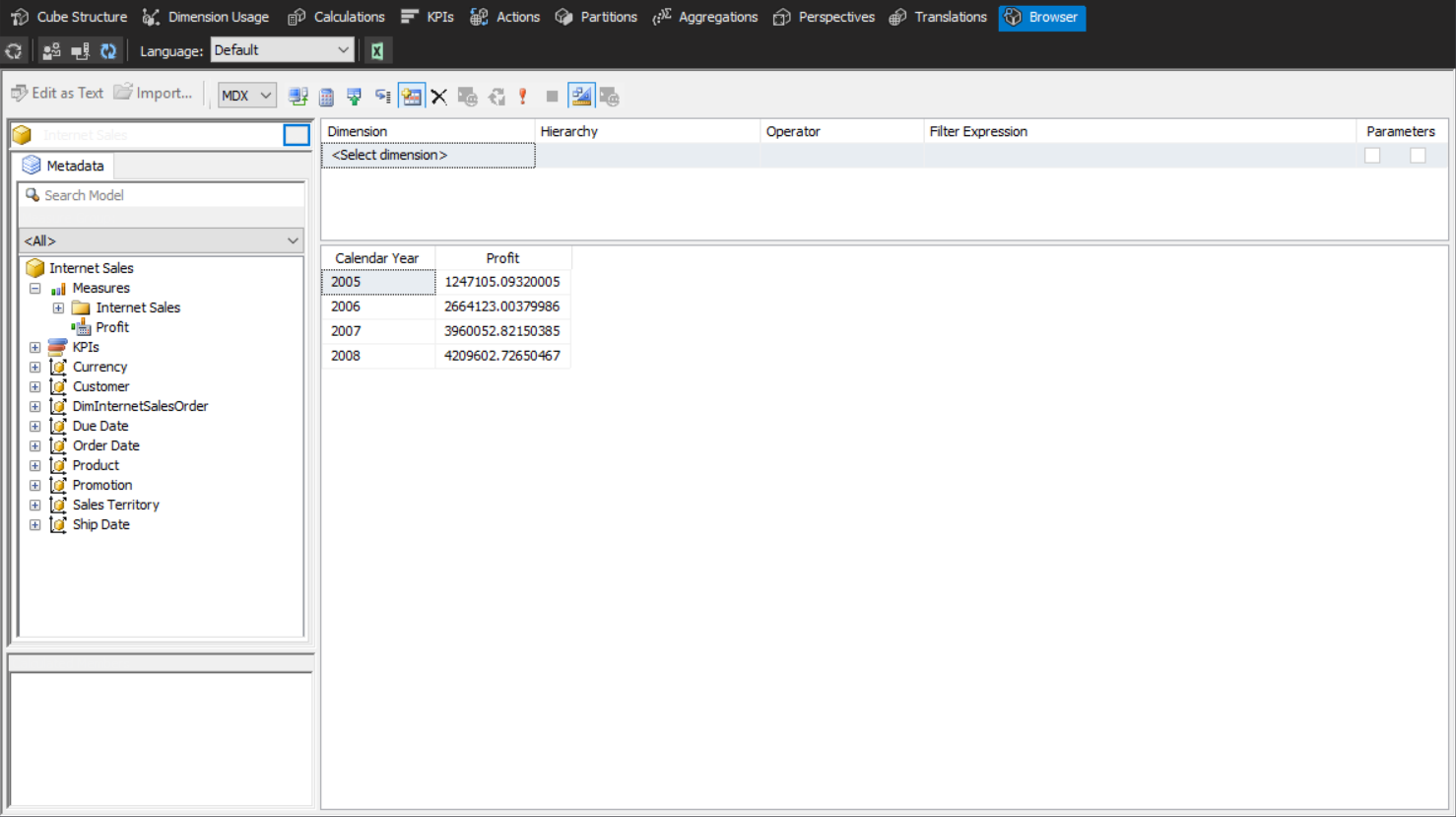
1. In the cube designer, go to the **Calculations** tab.
2. Create a new calculated member and name it as **[Profit]**.
3. Write the following expression in the **Expression** field:

*[Measures].[Sales Amount] - [Measures].[Total Product Cost]*

1. Set the format string as **“Currency”**



1. Deploy and process to be able to see the **Profit** measure in the cube browser



## What did we learn

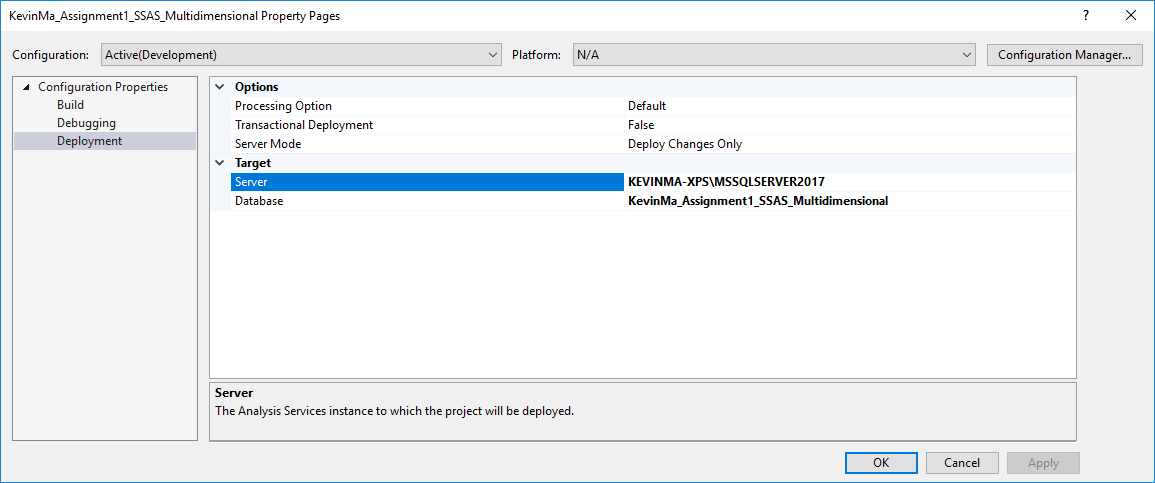
Different from what we did in SQL Server Management Studio, we didn’t need to write the whole **select** statement here. We just need to write the expression that calculates or applies conditions.

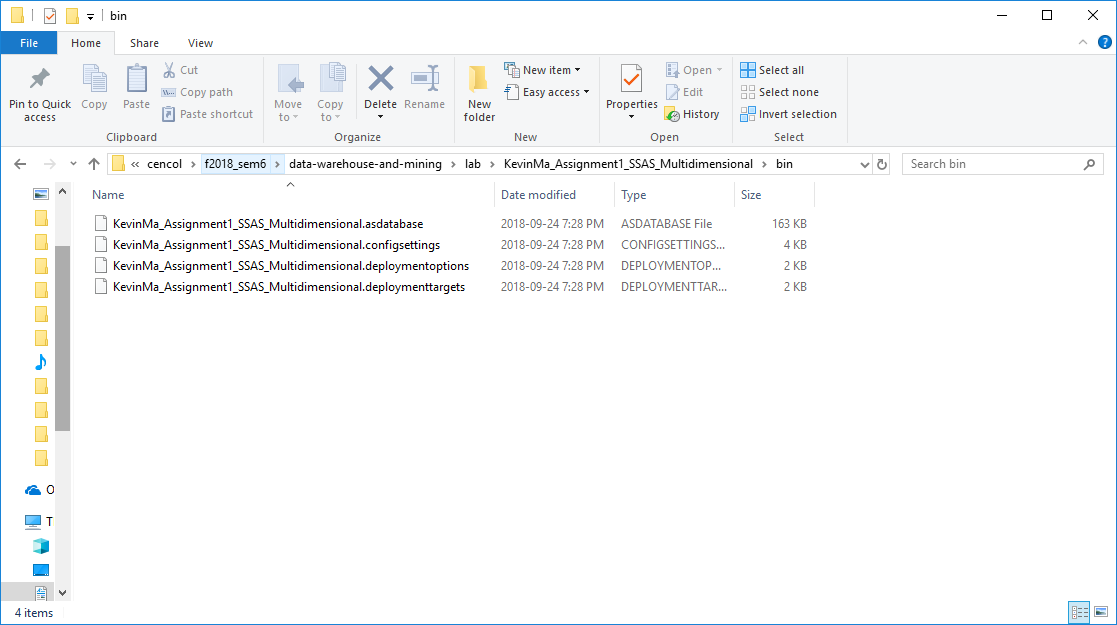
The MDX expression can be made more complex with the help of functions and metadata information listed in the bottom-left side of the preceding screenshot. You can drag-and -drop functions or measures or dimension hierarchies in the expression box and create the expression easily. There are also some properties for calculated columns, such as font, color, and format string, which can be set in this window.

# Time for action – deploying an SSAS project

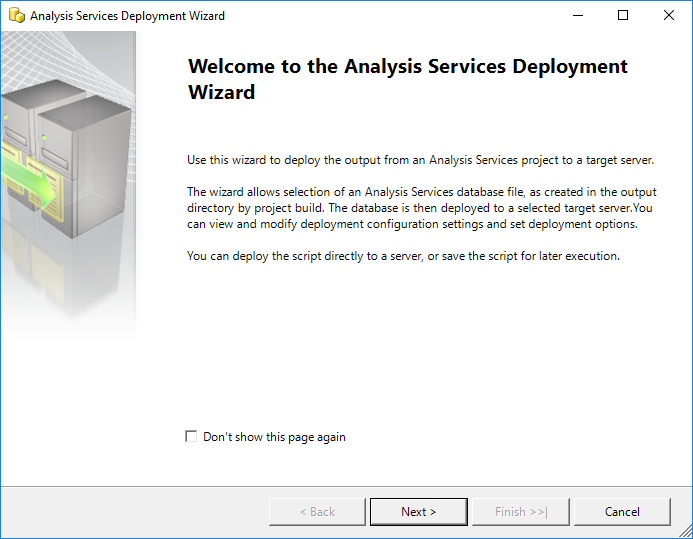
The Analysis Service project contains metadata and data. The metadata of the project can be stored to a server with deployment. Loading data into the SSAS project is called processing. There are multiple ways to deploy a project to the SSAS server, and the processing options also differs based on the method for loading data into the cube.

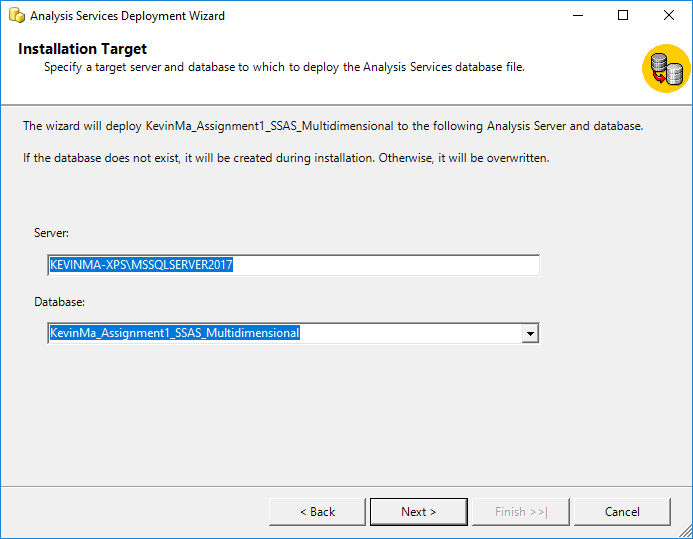
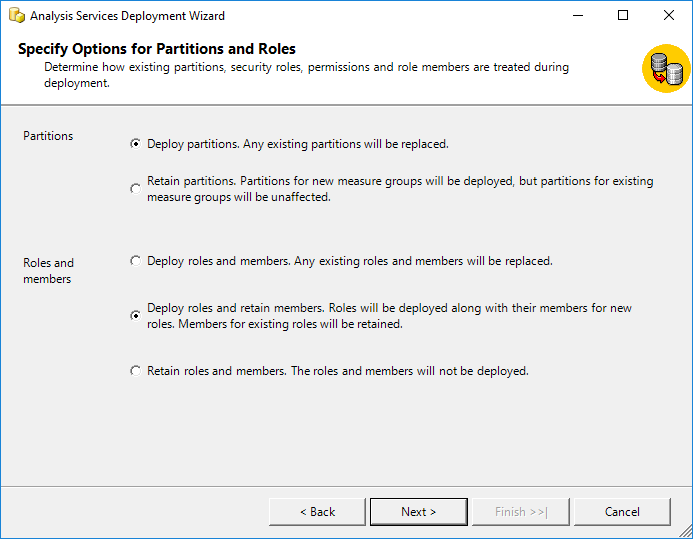
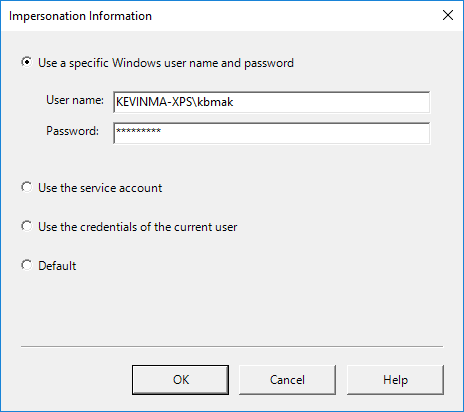
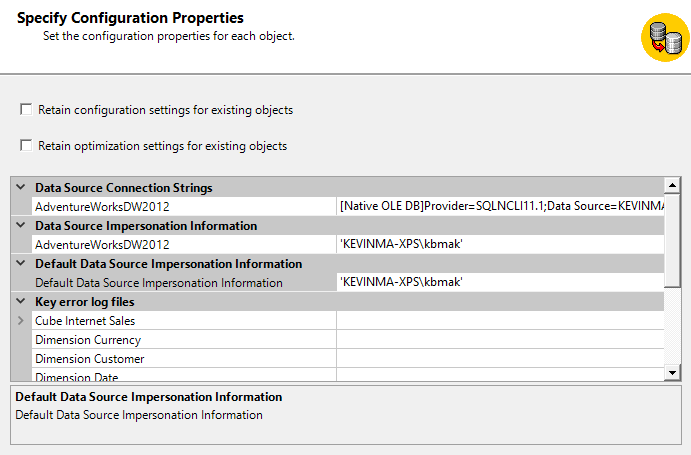
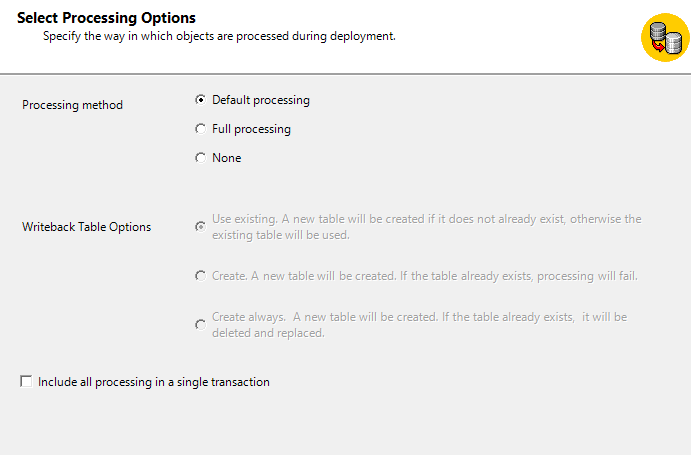
So far we have deployed from SSDT. There are other ways for deployment: deployment from **Deployment Wizard** and deployment with XMLA.

1. In the **Solution Explorer** pane, right-click on a project and select **Properties**.
2. We can see deployment properties such as the server and database name here:
3. Build the project by right clicking on the project in the Solution Explorer.
4. If we go to the **project** folder in Windows Explorer; under the **bin** folder, we will find four files. These files are deployment files that are created after a successful build action:



1. In the **Start** menu under Microsoft SQL Server xxxx, under **Analysis Services**, click on **Deployment Wizard.**



1. 
2. 
3. 
4. 
5. 
6. 
7. 

## What did we learn

SSDT is a tool for developers and it is good for developers to deploy their projects to the development environment servers. However, in many scenarios, the production environment is not accessible for developers. So, you will have to use other methods of deployment.

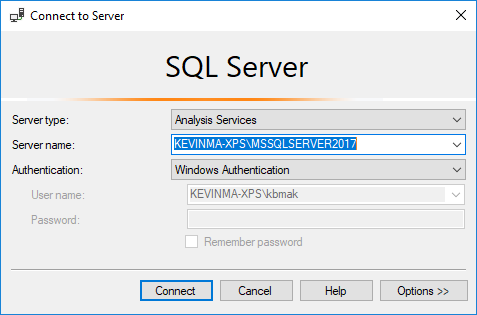
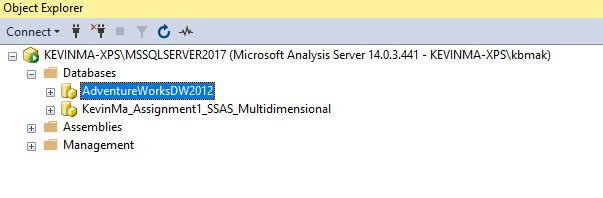
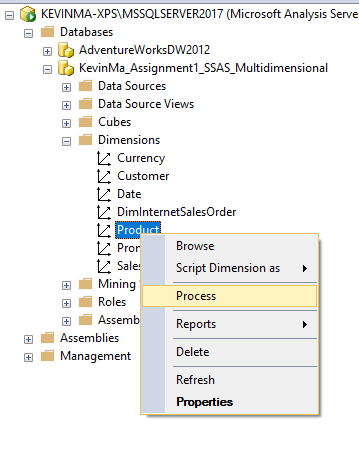
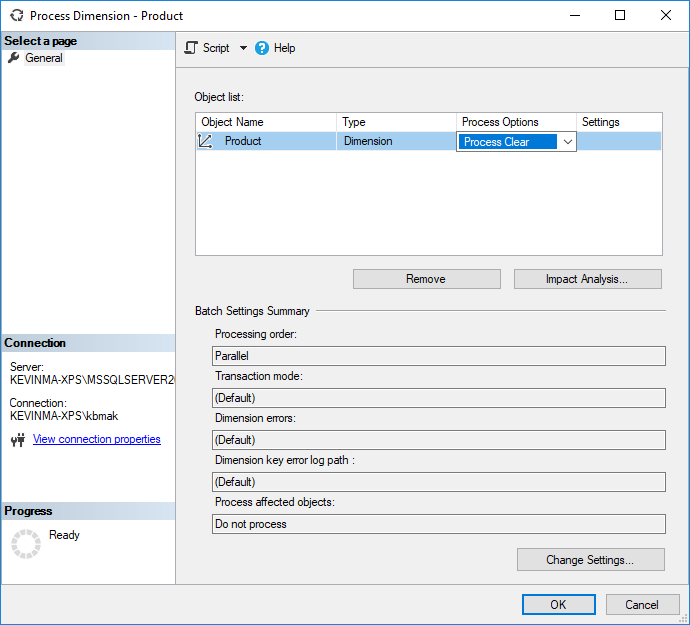
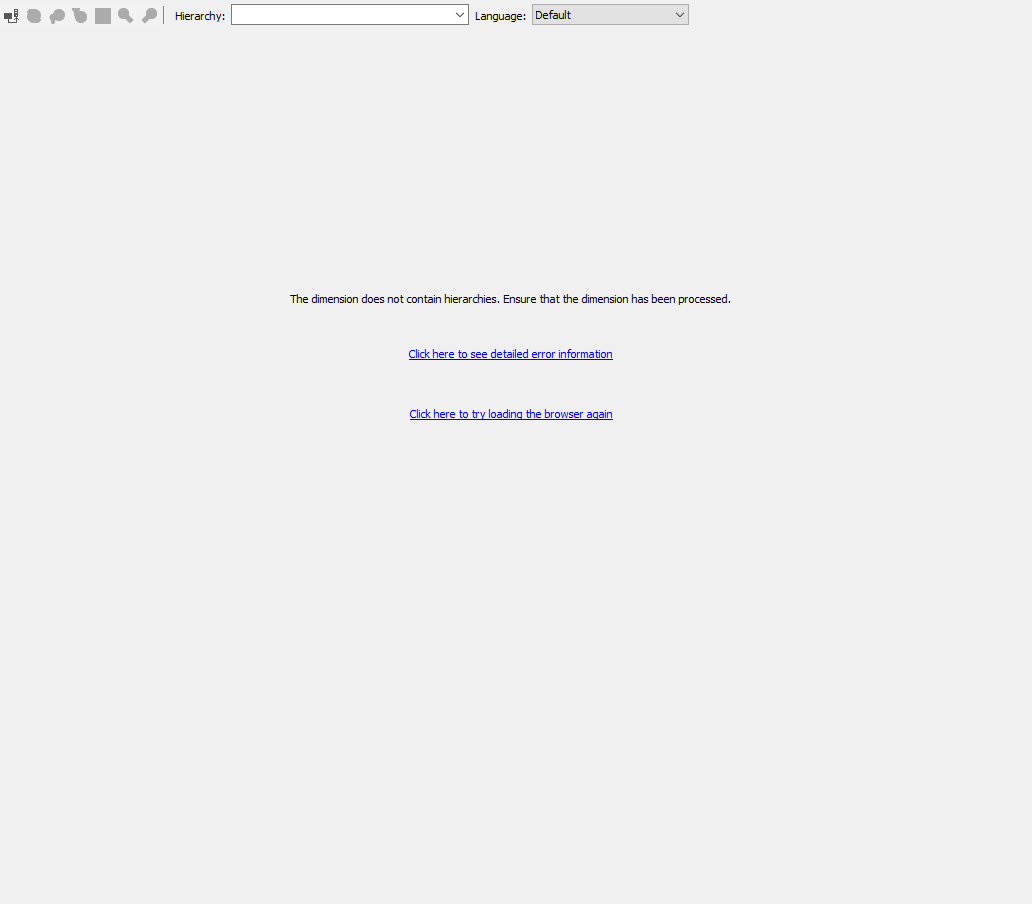
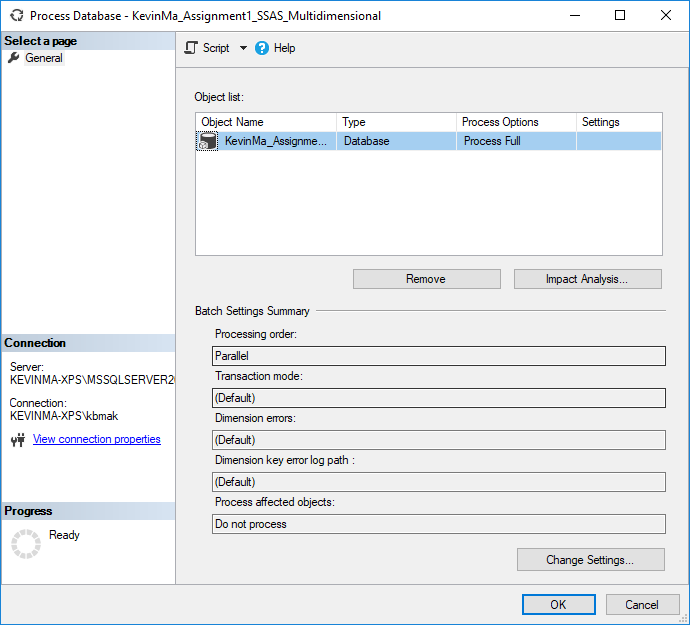
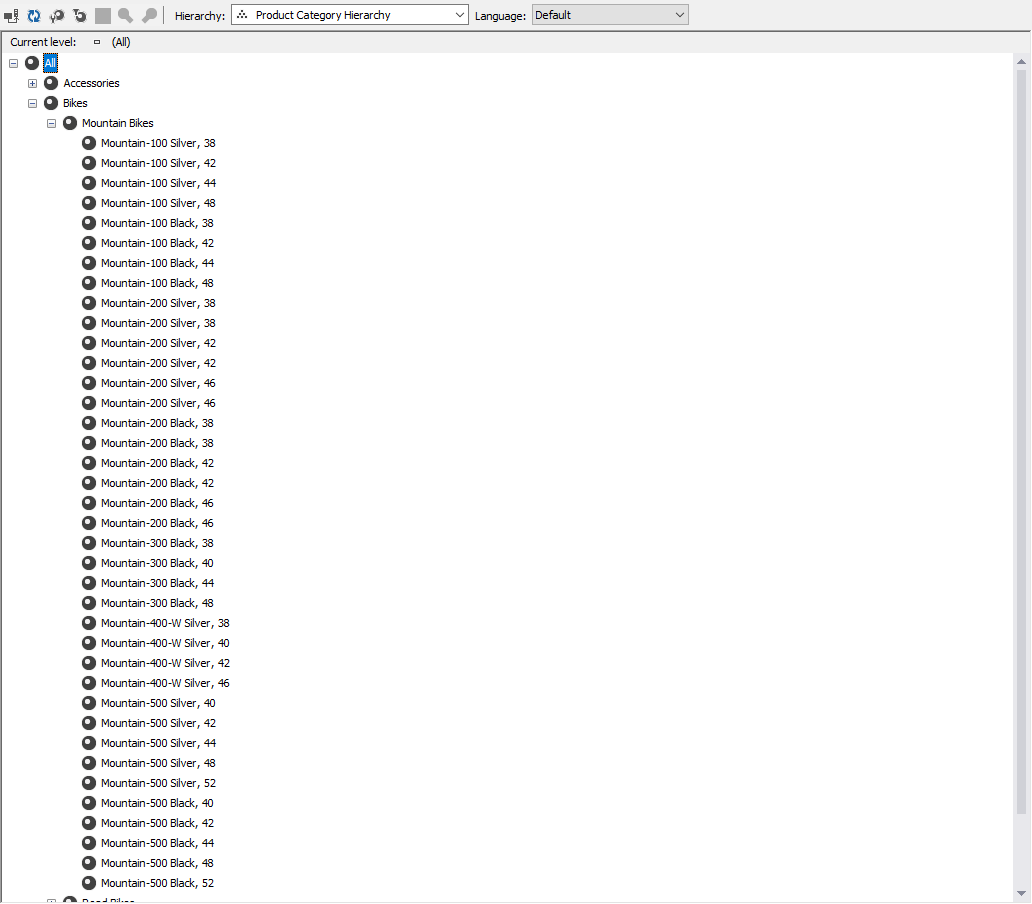
**Deployment Wizard** is a deployment tool that will be installed with the SSAS service. There are four files required to deploy an SSAS project. These files can be created while building the project. Each file consists of specific information about deployment. The DBA or the gatekeeper of the production environment can easily run the Deployment Wizard and give the **\*.asdatabase** file path; follow the steps in the wizard and deploy the project easily.

One of the main benefits of deploying the project with **Deployment Wizard** is that you can choose to retain roles and membership or retain partitioning on the destination database. When you conduct deployment through SSDT, the partitioning and roles will be overwritten. This is an important option because in the production environment, partitioning might be set up in the database, and security and role settings might be different from the security configuration in the SSAS project in SSDT.

XMLA is the language for metadata scripts in SSAS. Running the deployment wizard will produce and run an XMLA command on the SSAS server. If you want to overlook all the wizard steps, you can create an XMLA deployment script, modify it as you wish to make it run on the production environment, and then just pass this script to the DBA and ask them to run

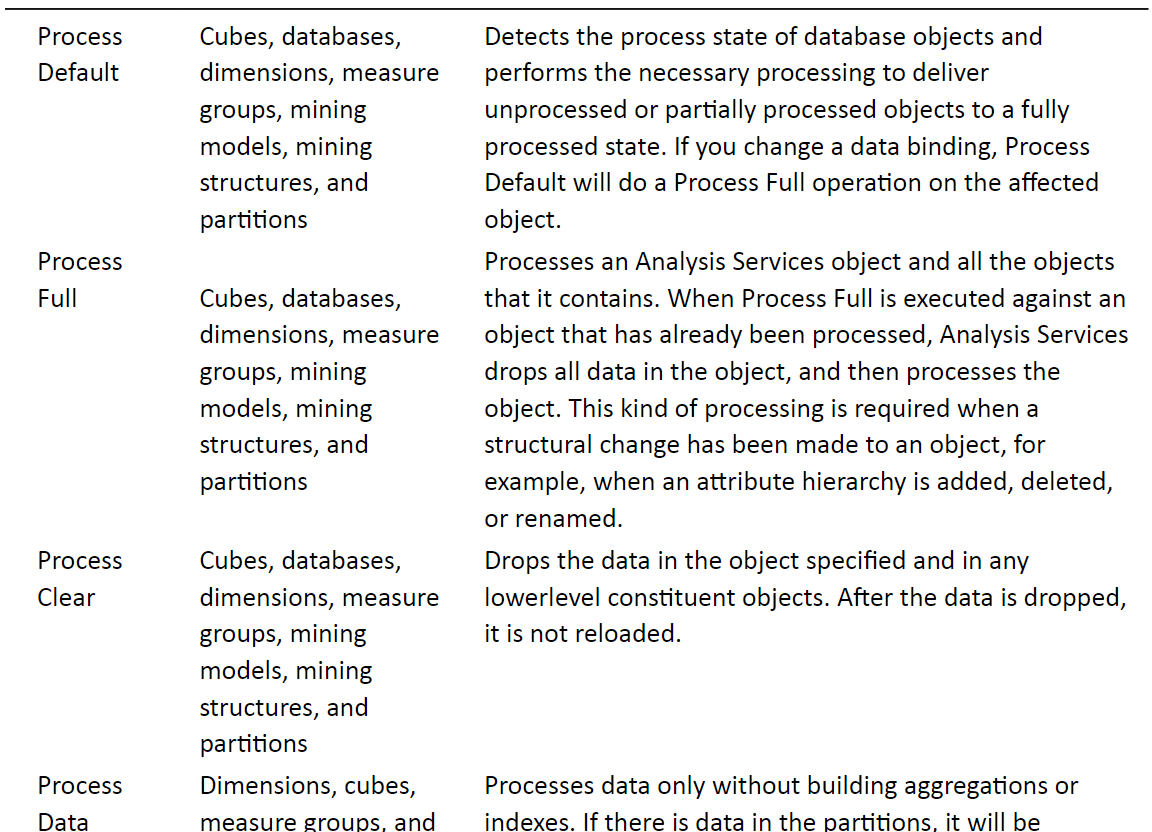
# Time for action – processing the data

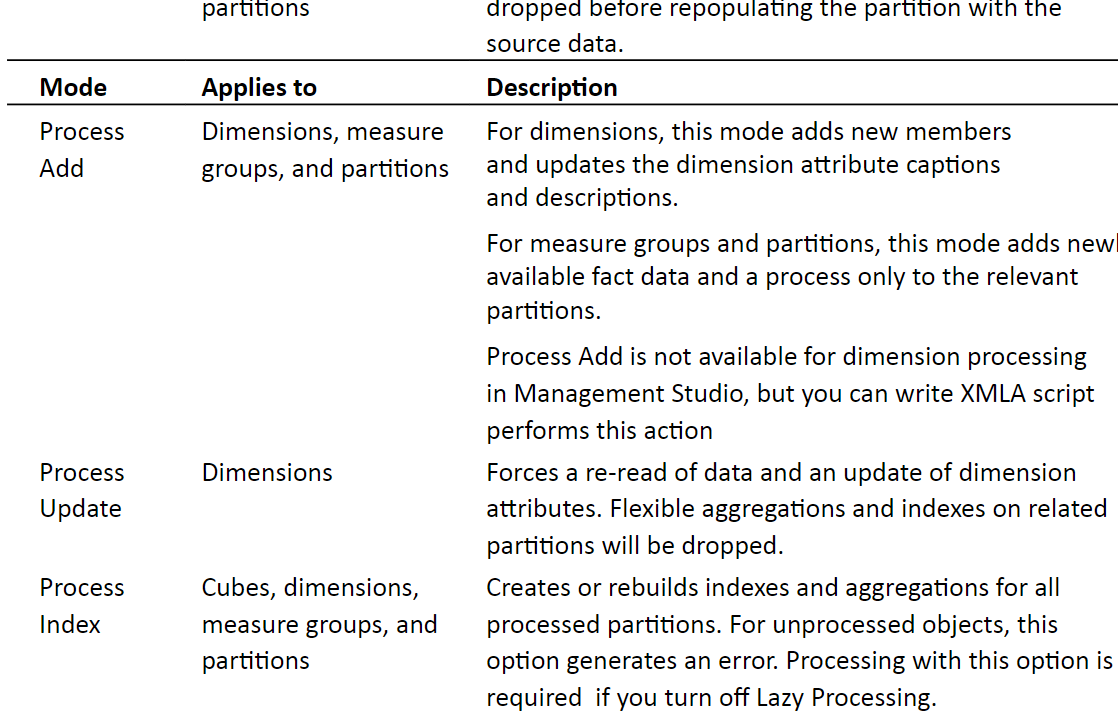
Deployment will load metadata from an existing Analysis Services project into the SSAS Server, but it won’t load data. Processing is another step we must do to load the data.

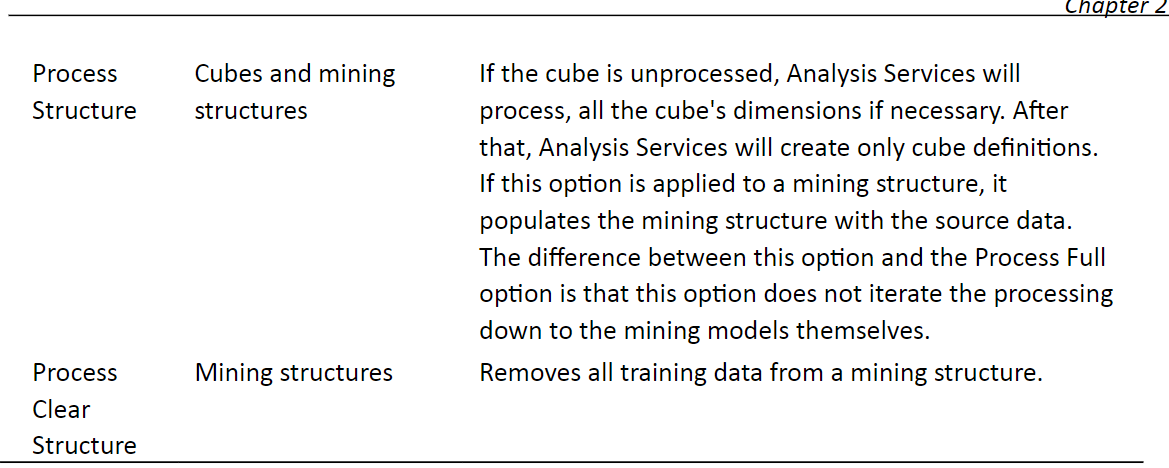
1. 
2. 
3. 
4. 
5. 
6. 
7. 

## What did we learn

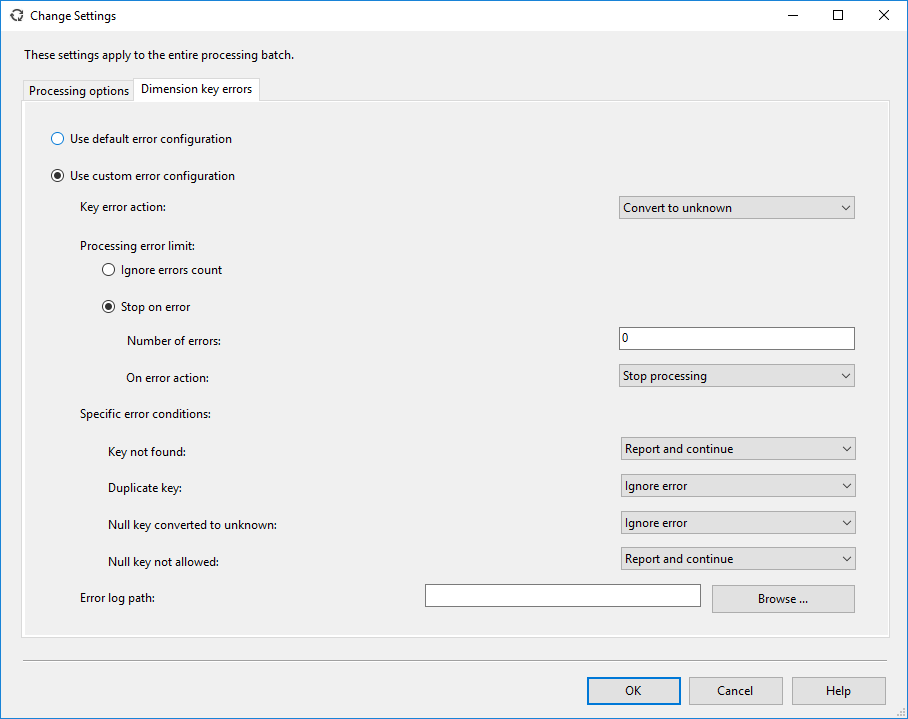
There are different kinds of processing; can be at dimension level or cube level.





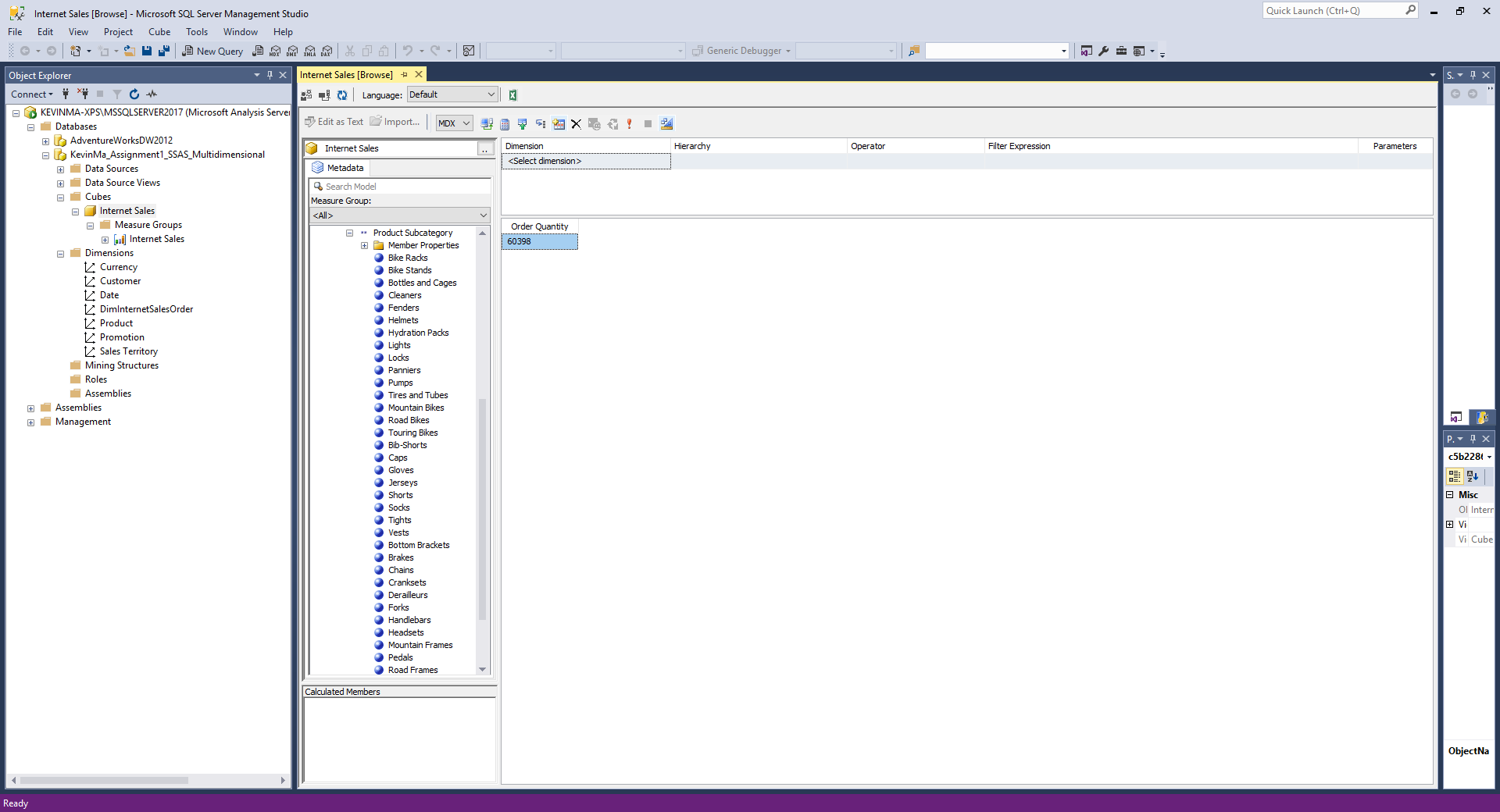
When processing, errors may arise. These errors my be caught during processing if error configuration has been configured. This configuration can be found in the Dimension Designer, cube designer and also in the Process window:

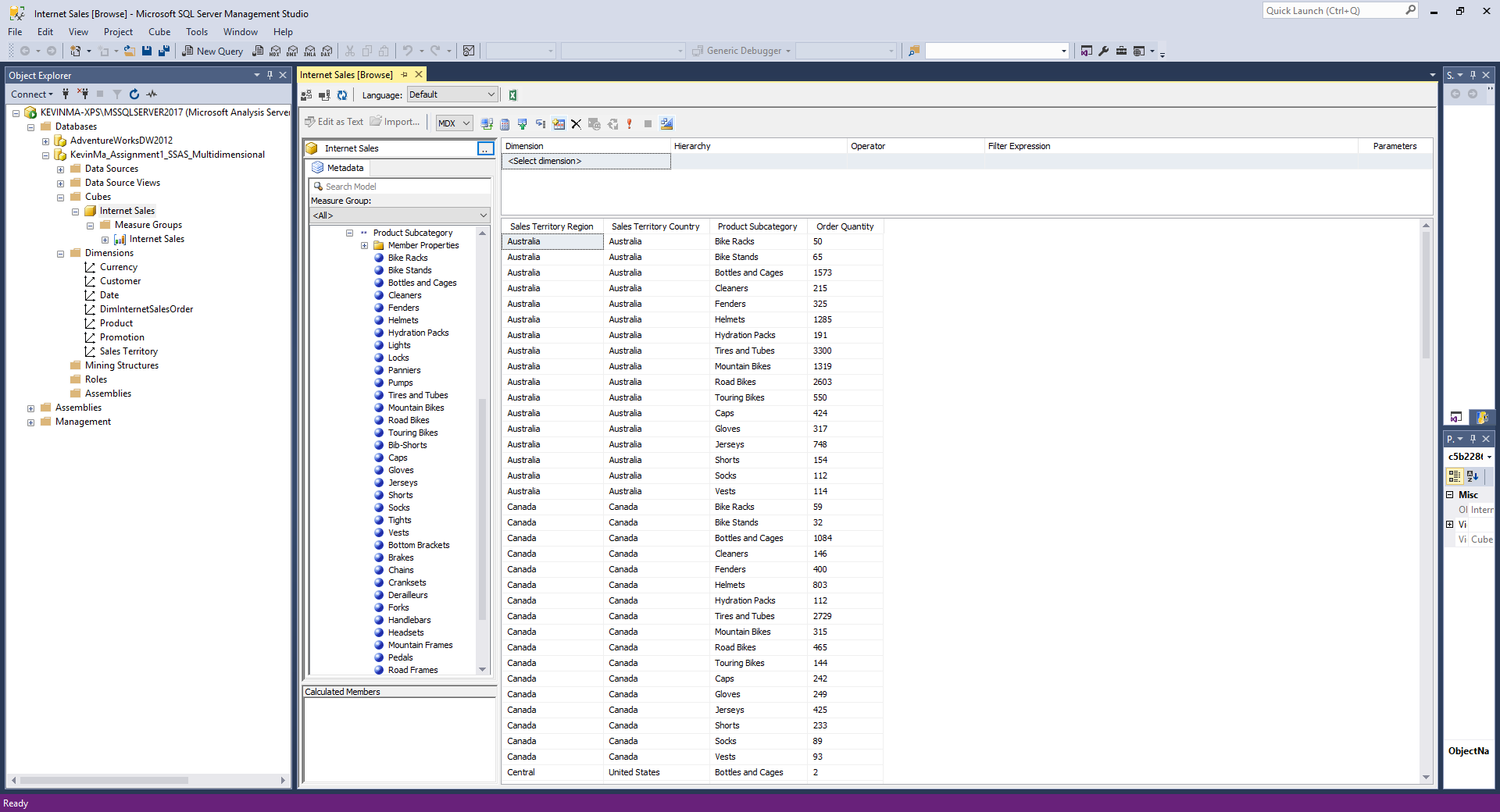


# Conclusions

A data warehouse has many functions such as providing the ability to congregate data from multiple data sources, standardizing their formats and presenting concise information which can be used for data analysis. Because of this notion, I believed that this project would be very challenging and that it would be very confusing in building an OLAP cube. However, contrary to my beliefs, building an OLAP cube using the SSDT (SQL Server Data Tools) along with Visual Studio Enterprise Edition and SQL Server Management Studio was straightforward. There are many toolboxes, views, windows and panels within the applications. However, the tools are well developed as there are helpful descriptions when you hover over them which made the OLAP building process smooth. After doing this project, I have come to better understand and appreciate the benefits an OLAP cube can bring to a business. Shown below are some of the screenshots I have captured showing my final product.

Within the cube, we can easily view the total values of measures without needing to execute any queries against a database:



Within the cube we can slice and dice different dimensions with measures in order to get concise views which can illustrate business trends for us:

Using the views within the cube we can more easily identify important pieces of information. For example, we may be able to identify where there were more products sold in a month by looking at the below view. This may then allow us to narrow down our focus into the data for that month. This would help us identify any specific events which occurred around that time to trigger this increased amount of sales. Finally, the business could possibly reproduce this positive behavior when the business applies the knowledge it has gained from the information which was garnered from the analysis of the data.

