

Reporting

Module Overview

In this module, you learn about how to create administrative reports using SQL Server Reporting Services and Power BI.

Required Software

- SQL Developer Edition (or equivalent)
- Visual Studio 2019 Community edition with the SSRS Extension
- SSMS
- Excel (Not Free)
- Report Builder
- PowerBI Desktop
- Tableau Public
- Custom Applications

Assignment

Each week you have to perform an assignment. Let us review this week's assignment.

Reporting Applications

There are **many companies** building reporting applications. In fact, I would not be surprised if there are **over a hundred different reporting applications** out there. Most of them are **small** companies trying to get market share, but there are many **large** companies that are also interested in getting people to use their software.

You can **specialize in a few reporting software applications and feel confident** that you will **learn and use other reporting software** as needed.

That is because, in general, they all need **to follow the same pattern**:

- Make a **connection** to the database
- Access send data via a SQL **select** statement
- Allow for **displaying** data in a table or graphic format.

An internet search will bring up **many different websites** directing you to their **proprietary** reporting Solutions.

However, some websites **will let you review a collection of reporting applications**. These often seem to be independent on the surface, some are, but **often** you'll find that they are **biased**.

"Find the best Reporting Software for your business. Compare product reviews and features to build your list."
(<https://www.capterra.com/reporting-software/>, 2017)

Microsoft

Microsoft has spent much money creating and improving reporting applications and offers **many to choose from**. Most applications' **features overlap**, and Microsoft will likely combine some of them in the future. Currently, all of them may be considered valid options for reporting applications.

Excel

Excel is a spreadsheet application that includes **many reporting features**. Microsoft considers this a vital part of its BI applications stack. **Easy to learn** and earlier versions have been **used for more than a decade**. Excel must be **purchased** and installed on PCs and Mac and is designed to be a single user application.

Power BI

Power BI is a **more complex** piece of software that comes in **both free and paid-for versions**. The **free** desktop version **provides many built-in functions**, more so than most other free versions of similar software in the industry.

Excel Reports

Excel is Microsoft's frontline reporting software. Starting initially as a commercial spreadsheet application, it has become much more. It has grown and changed over the years to include PowerBI engine.

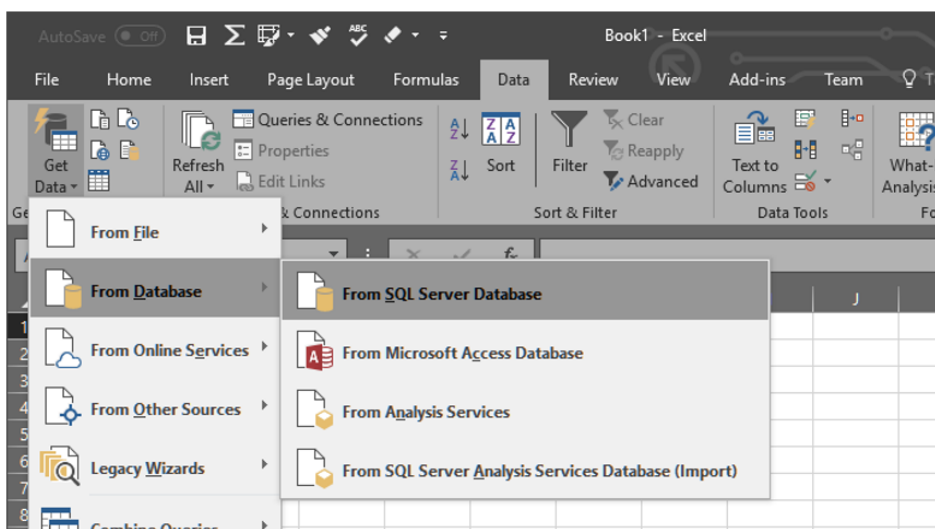
- Can display data in different forms such as in **charts, graphs**, or other types of reports
- Has programming features using "Visual Basic for Applications (**VBA**)" and Data Analysis Expressions (**DAX**)
- Reports are most commonly created by users with little to **no programming skills**
- Capable of creating reports from **many different data sources**

Creating a Connection

To make a connection to Data:

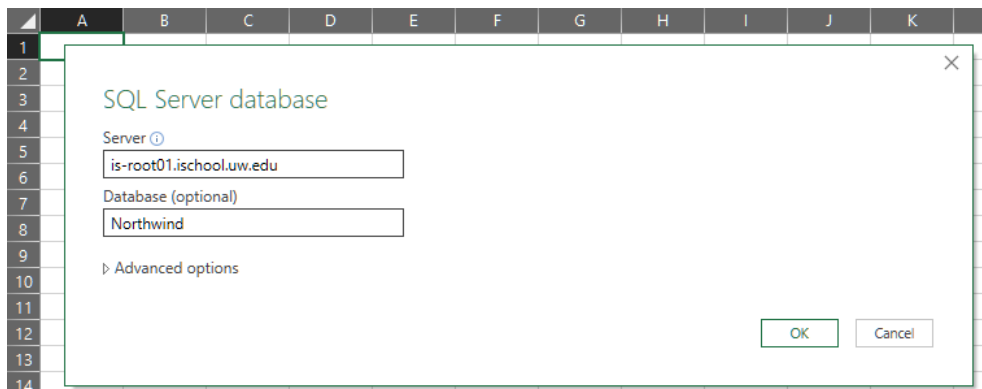
1. Create or open an Excel spreadsheet
2. Look for the **Data tab** on the ribbon interface
3. The **get data** button and select the data source you want to use from the context menu
4. Advance through the **wizard** answering its questions

For example, if I wish to connect to Microsoft's SQL server, choose the option, **From database -> SQL Server Database**.



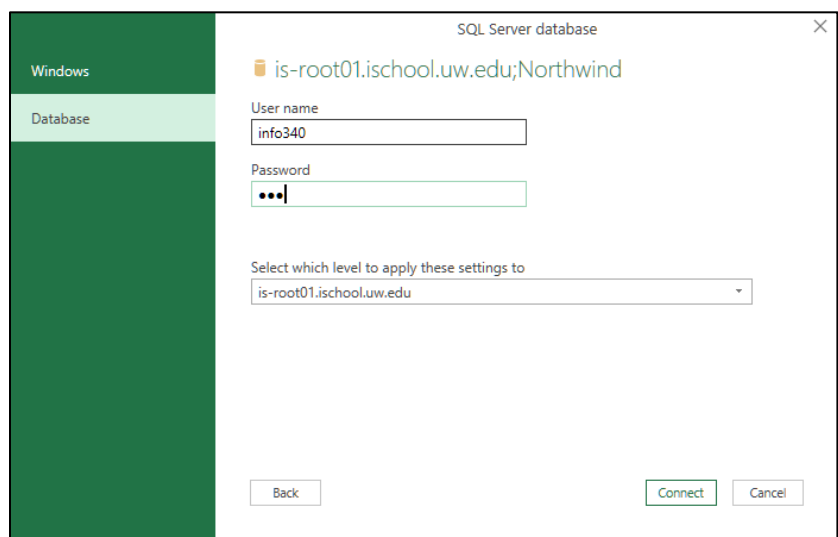
At first, I would have to enter the name of the **server** and optionally the name of the **database** I wish to use.

(SQL – Level 3 – Non-relational SQL Server Features and Tools)



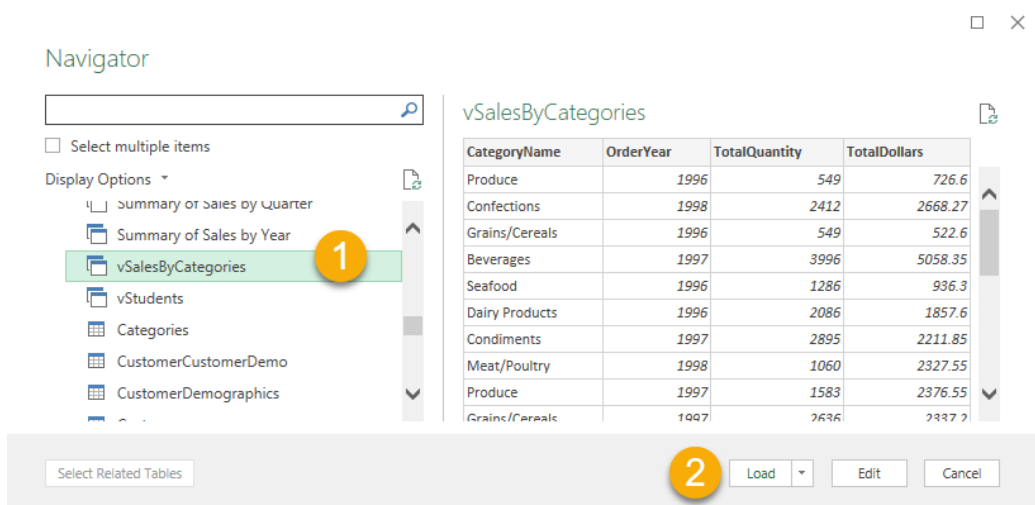
Next, choose the type of **authentication**, either Windows or database-based authentication.

With Windows-based, I only need to indicate a Windows account, but with database authentication, I will need to enter in a username and password.



Microsoft will **warn** you that the data will **not** be automatically **encrypted** as it passes from client to server unless you encrypt it yourself. **Just click, OK!**

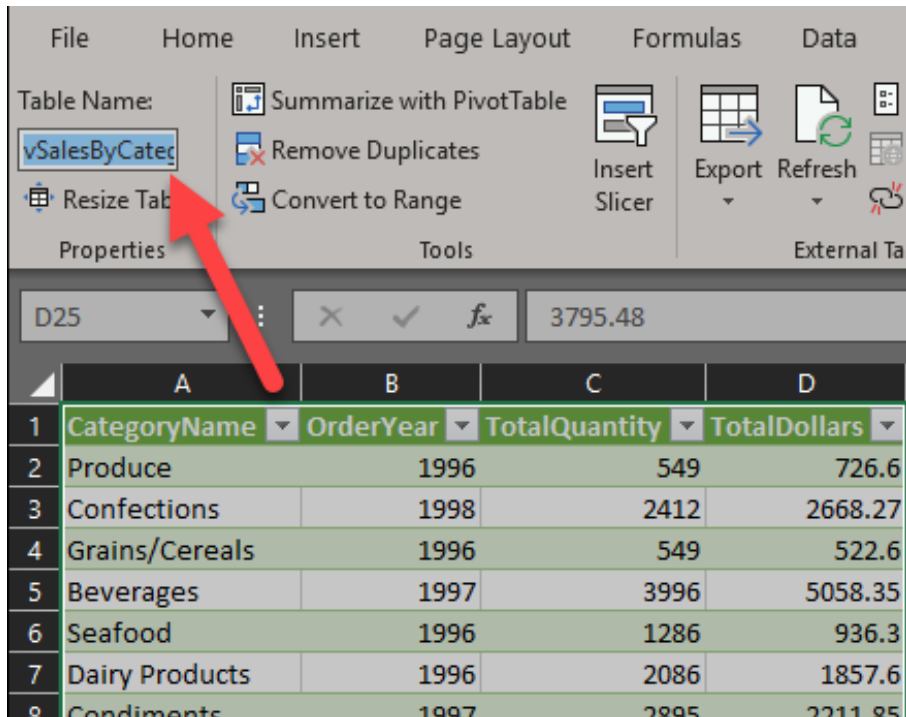
After that, you will be presented with a list of **objects to choose** from that contain reporting data.



After selecting an object and using the **load** button, you should see data in the spreadsheet.

Important: On the **Mac**, Microsoft chose **not to show views in the tree view**. Instead, you access them using the SQL Statement pane in the upper right of the dialog. Check out this video for an example of someone using that pane.
<https://youtu.be/9oUJEGoB4-8?t=124>

And this web page for how to make a connectin on Mac: <https://support.microsoft.com/en-ie/office/import-data-from-a-database-in-excel-for-mac-e127365a-02a5-47b5-8278-16dd65edbe61>



The screenshot shows the Microsoft Excel interface with the 'Table Name' field set to 'vSalesByCategories'. A red arrow points to this field. The spreadsheet displays the following data:

	A	B	C	D
1	CategoryName	OrderYear	TotalQuantity	TotalDollars
2	Produce	1996	549	726.6
3	Confections	1998	2412	2668.27
4	Grains/Cereals	1996	549	522.6
5	Beverages	1997	3996	5058.35
6	Seafood	1996	1286	936.3
7	Dairy Products	1996	2086	1857.6
8	Condiments	1997	2895	2211.85

Important: Note the name of the table for later use in the Tableau lab.

Demo 1: Creating Excel Reports

In this demo, we use Excel to create a report using data from the Northwind database.

1: Create a reporting view

Run the following SQL code, understand what it does, and then use it to create a reporting view called, "vSalesByCategories";

```
use Northwind;
go
Create View vSalesByCategories
as
Select
[CategoryName]
,[OrderYear] = Year(o.OrderDate)
,[TotalQuantity] = Sum(od.Quantity)
,[TotalDollars] = Sum(od.UnitPrice)
From Northwind.dbo.Categories as c
Join Northwind.dbo.Products as p
On c.CategoryID = p.CategoryID
Join Northwind.dbo.[Order Details] as od
On p.ProductID = od.ProductID
Join Northwind.dbo.Orders as o
```

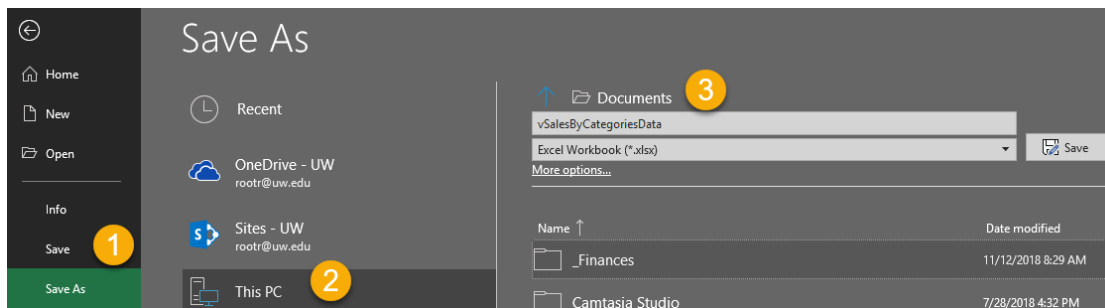
On od.OrderID = o.OrderID
Group By c.CategoryName, Year(o.OrderDate)

Step 3: Create an Excel report

Create an Excel report that connects to your database and displays data from your new view.

Step 4: Save Your File

Save this file in an easy place to locate, like the Documents folder. We will use this Excel data in the Tableau lab later.



Step 4: Review Your Work

Now, you will review your work with your instructor.

Reporting Server

Now we will look at Microsoft's reporting application, SQL Server Reporting Services (SSRS), also known as Reporting Services, Reporting Server, and Power BI Reporting Services.

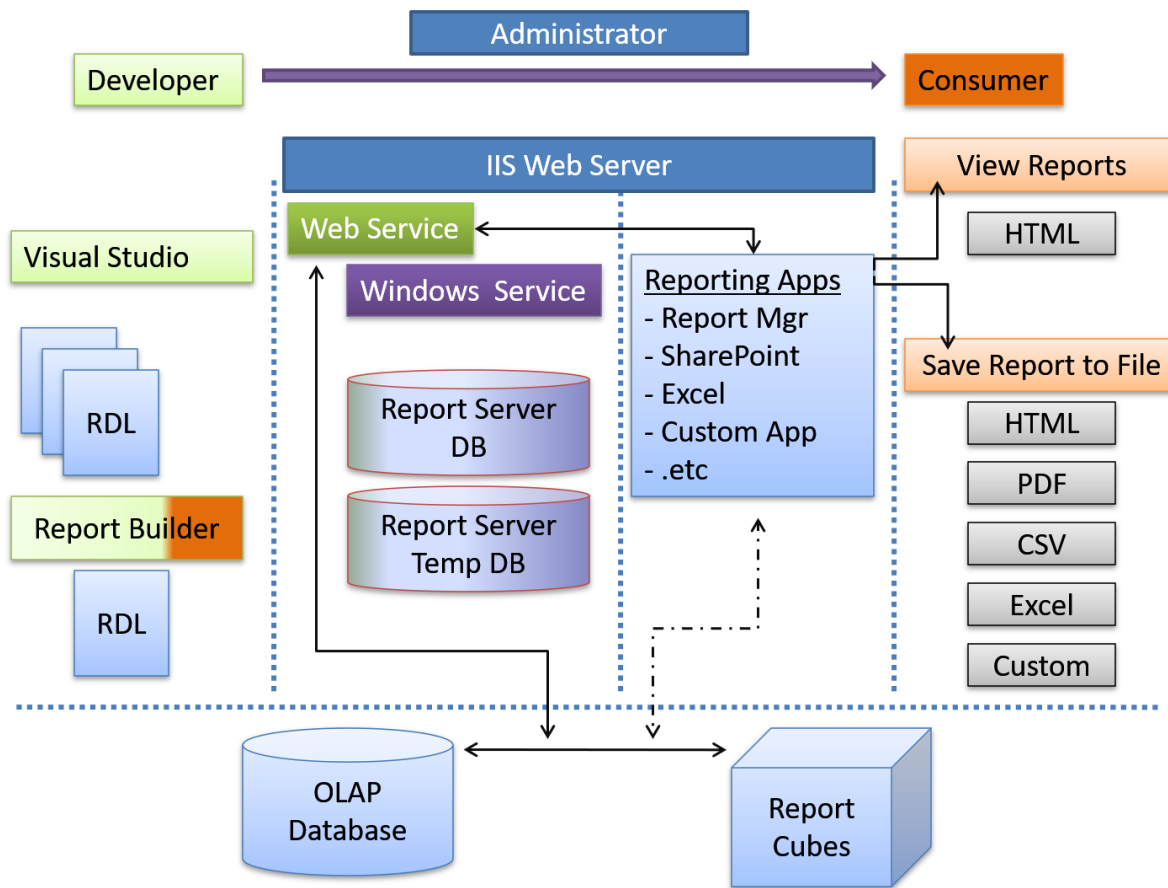
"SQL Server Reporting Services is a solution that customers **deploy on their own premises** for **creating, publishing, and managing reports**, then delivering them to the right users in different ways, whether that's viewing them in **web** browser, on their **mobile** device, or as an **email** in their in-box.

For SQL Server 2016, Reporting Services offers an updated suite of products:

- "Traditional" **paginated reports** brought up to date, so you can create modern-looking reports, with updated tools and new features for creating them.
- New **mobile reports** with a responsive layout that adapts to different devices and the different ways you hold them.
- A **modern web portal** you can view in any modern browser. In the new portal, you can organize and display mobile and paginated Reporting Services reports and KPIs plus Power BI Desktop reports. You can also store Excel workbooks on the portal." (<https://docs.microsoft.com/en-us/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports>, 2017)

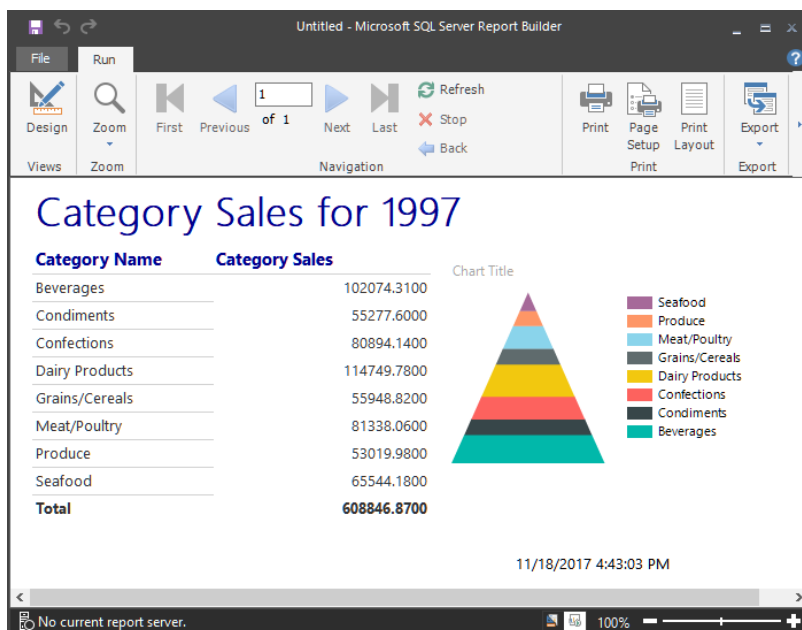
SSRS Architecture

SSRS is complex of SQL Server's because it is made up of **many components** that can be spread across **multiple computers** to provide a high degree of **scalability** and performance.



Report Builder

"Report Builder is a **stand-alone authoring environment** for creating Reporting Services paginated reports outside of Visual Studio. When you design a report, you specify where to get the data, which data to get, and how to display the data. When you **run** the report, the report processor takes all the information you have specified, **retrieves** the data, and **combines** it with the report layout to **generate** the **report**." (<https://docs.microsoft.com/en-us/sql/reporting-services/tools/report-builder-authoring-environment-ssrs, 2017>)



SSRS Projects

Visual Studio's Reporting Services project provides **full-featured development**. It can accomplish everything done in Report Builder, plus you have the ability to **manage multiple files** concurrently, work directly with source control, and include the Reporting Services project in the **same solution as your SSIS projects**.

Add a new project

Recent project templates

- Integration Services Project
- Analysis Services Tabular Project
- Analysis Services Multidimensional and Data Mining Project
- Console App (.NET Framework) C#
- Python Application Python
- ASP.NET Web Application (.NET Framework) C#

Report

C#

All platforms

Web

Clear all

No exact matches found

Other results based on your search



Report Server Project Wizard

Create a new Report Server project using Report Wizard.



Report Server Project

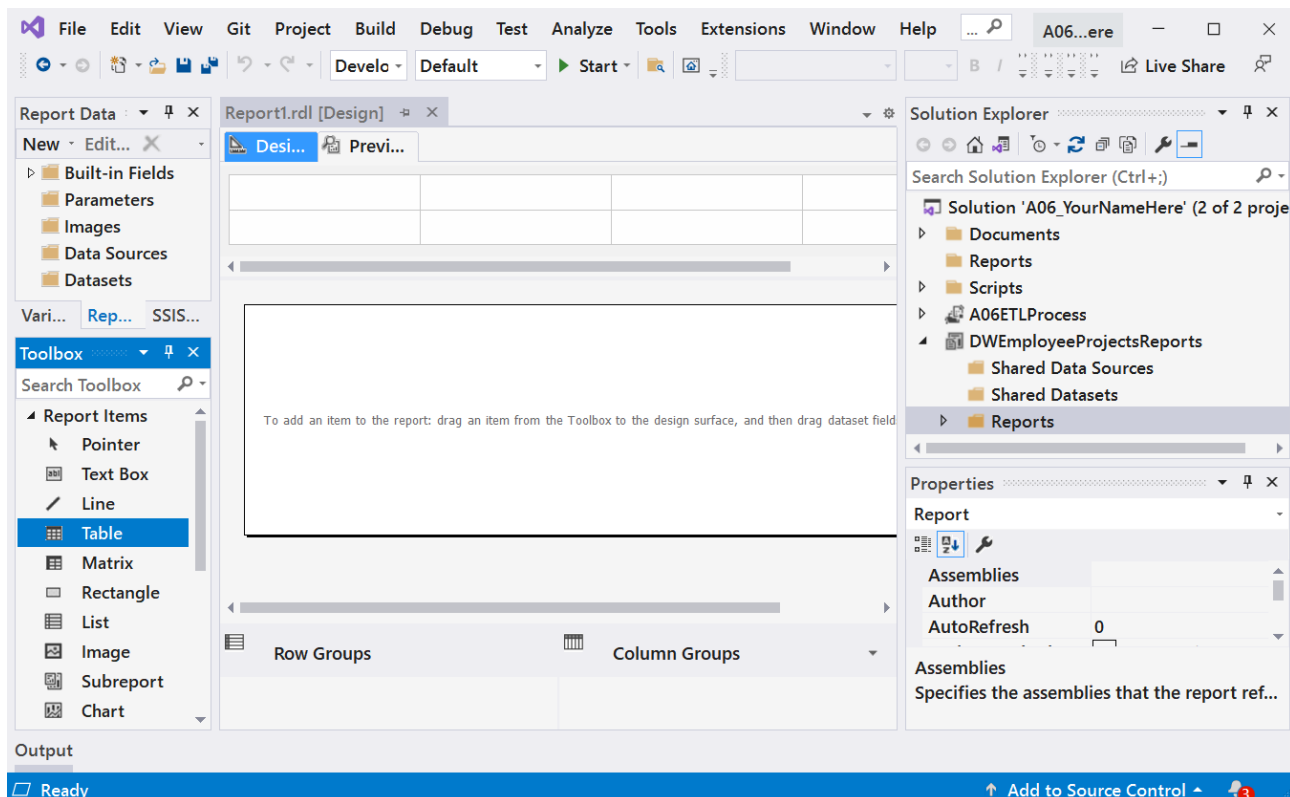
Create an empty Report Server project.



Import from Server (Multidimensional and Data Mining)

Creates a multidimensional and data mining project by extracting the metadata from an existing multidimensional and data mining database on an Analysis Services server.

Unlike Report Builder, Visual Studio is **more utilitarian** in design. This is partially because it provides additional functionality, but it is also because its intended audience is professional developers rather than casual developers.



The development tools are not difficult to use; they just take **more effort to learn**. It is similar to SSIS projects, in that you first drag and drop items from the Toolbox onto a design surface and then configure them.

SSRS Web Applications

Each installation of Reporting Services includes **two web applications**, both of which are built using ASP.NET. The first is the **web service** and the second is an **end-user application for viewing and managing reports**.

This **end-user application is known as Report Manager** and is designed to **interact with the web service**. Most companies find that the Report Manager web application works well for their needs.

The web service allows other developers to create custom windowed, console, and web applications that can interact with the web service as well. This means that if you do not want to use Microsoft's ready-made web application—Report Manager—you can create your own by programming it to interact with the web service. The SSRS **web service is not designed for humans to use directly**, but it can be accessed directly if you so desire.

SSRS Services

Once you are done creating and testing a report, you **can upload the file to the SSRS web service** using Visual Studio, Report Builder, the Report Manager website, or even a SharePoint website.

This variety of options is made possible because all of these applications can interact with the SSRS web service. This flexibility is one of the reasons **why Microsoft designed Reporting Services to use a Web service** for the report processing. In addition, the web service acts as a public **interface for the SSRS Windows service**, allowing Visual Studio, Report Builder, the SSRS Report Manager, SharePoint, and other software to interact with the SSRS windows service through an abstraction layer.

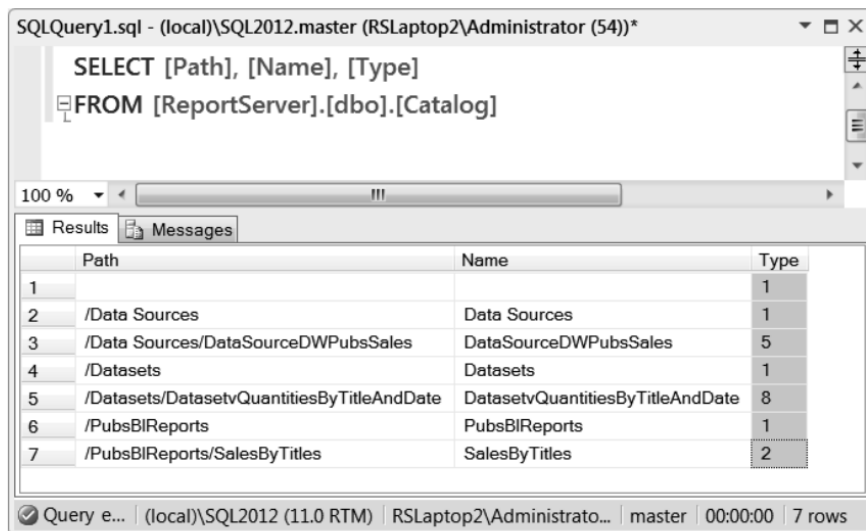
The web and Windows services' purpose is to process the programming instructions found in RDL files. The contents of the **RDL files will be stored in a pair of SQL Server databases**.

SSRS Databases

Microsoft installs two SQL Server databases with each installation of Reporting Services. The **first database, normally named ReportServer**, is designed to **hold the RDL file code, metadata, and configurations**. The **second database**, normally called ReportServerTempDB, is designed as a **workspace for processing report requests**. It also acts as a repository for cached reports.

***Tip:** If you are using a named instance, the databases may have the instance name added to their standard names: ReportServer and ReportServerTempDB. Not to worry, though, the name variation does not change the way they work.*

When a report is uploaded to the ReportServer database, the **report's code is stored in a binary format, within a table named Catalog**. If you select the data from the Catalog table, you will see a listing for every SSRS folder, data source, dataset, and report that has been created on the Reporting Server websites.



Typeld	Type Description
1	A logical folder
2	A report
5	A shared data source
8	A shared dataset

Demo 2: Creating Reporting Server Reports

In this demo, we use Visual Studio SSRS Project to create a report using view in Demo1.

Power BI Reports

"Power BI is a **suite** of business **analytics tools** that deliver insights throughout your organization. Connect to hundreds of data sources, simplify data prep, and drive **ad hoc analysis**. Produce beautiful reports, then publish them for your organization to consume on the **web and across mobile devices**. Everyone can create personalized dashboards with a unique, 360-degree view of their business. And scale across the **enterprise**, with governance and security built-in." (<https://powerbi.microsoft.com/en-us/>, 2017)

It is hard to find good information about the different components that make up the Power BI family of applications. That seems to be due to marketing and rapid changes. However, here is a list of the general components:

Power BI Service

Power BI web service is a cloud-based application that allows subscribers to **store, manage, and view Power BI reports and custom components**.

Power BI Web Site

The Power BI website acts as a **front-end to the Power BI cloud** web service.

Power BI Desktop

The Power BI Desktop application allows users to **create reports that can be viewed with the desktop application or uploaded to the Power BI web service**. Once uploaded to the Power BI web service, these reports can be accessed through the Power BI website, several Microsoft applications, and custom applications accessing the web service.

Power BI Mobile

The Power BI mobile application allows users to **view and interact with Power BI reports** from tablets and cellphones.

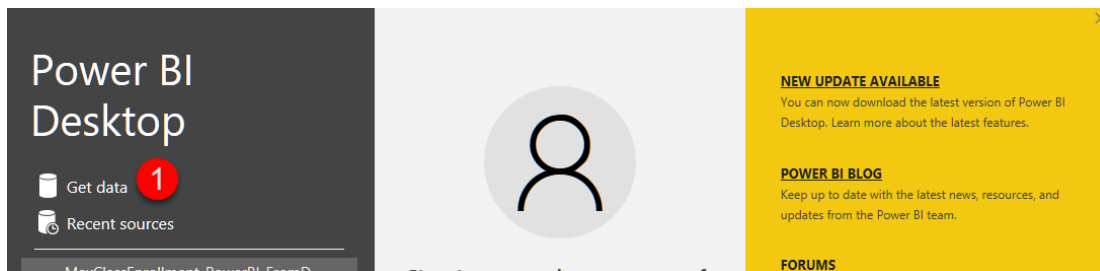
Power BI Reporting Server

The Power BI Reporting Server is an on-premise server to **store and manage reports without a cloud**.

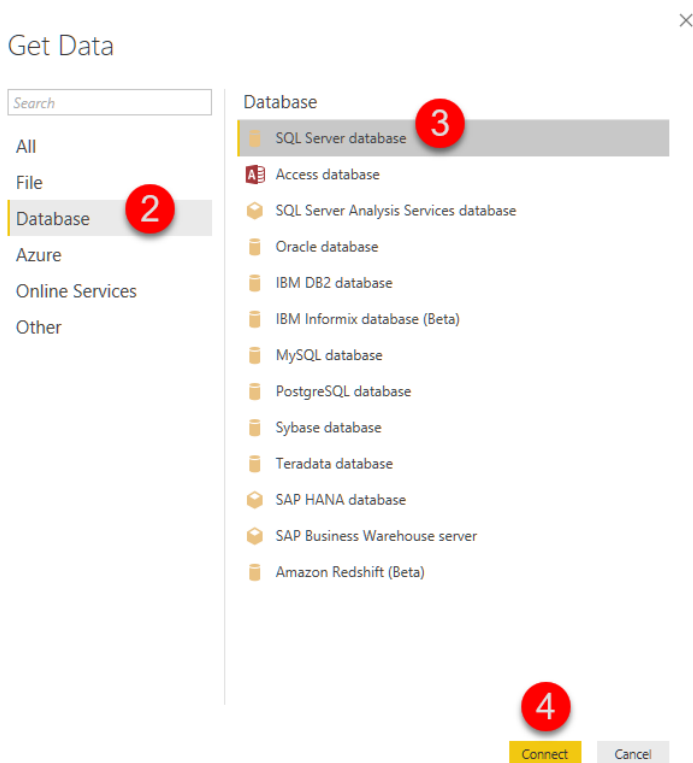
Creating a Power BI Report

"The **3 major building blocks** of Power BI are: **dashboards, reports, and datasets**. You can't have dashboards or reports without data (well, you can have empty dashboards and empty reports, but they're not very useful until they have data), so let's start with datasets." (<https://docs.microsoft.com/en-us/power-bi/service-basic-concepts#power-bi-concepts>, 2017)

When you first **open the Power BI desktop** application, you will see a **splash screen** with a **"Get data"** icon. Clicking this icon will launch several dialogue windows that will help you create a new report.



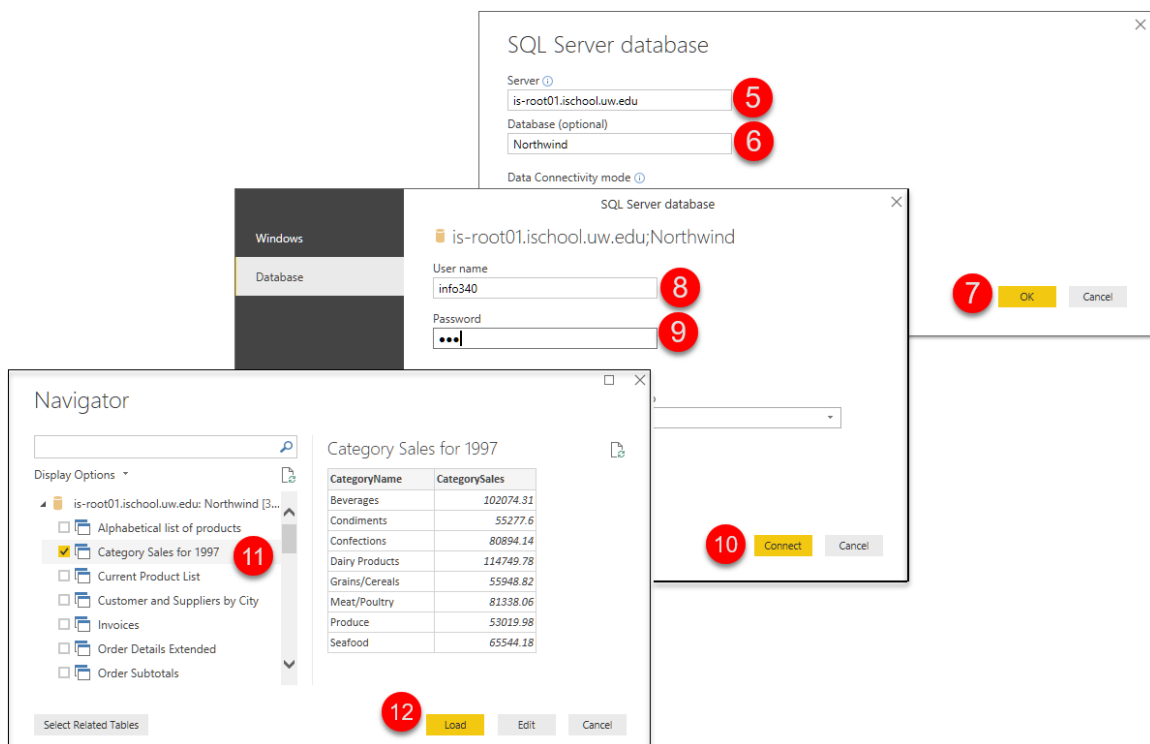
The get data dialogue window allows you to **choose between many different data sources**. More data sources are being added all the time by Microsoft, and custom ones can be created as well. For our examples, we will be using a SQL Server database connection. To make this selection choose from the **database** category on the **left** side of the dialogue and the **SQL Server** database option on the **right**.



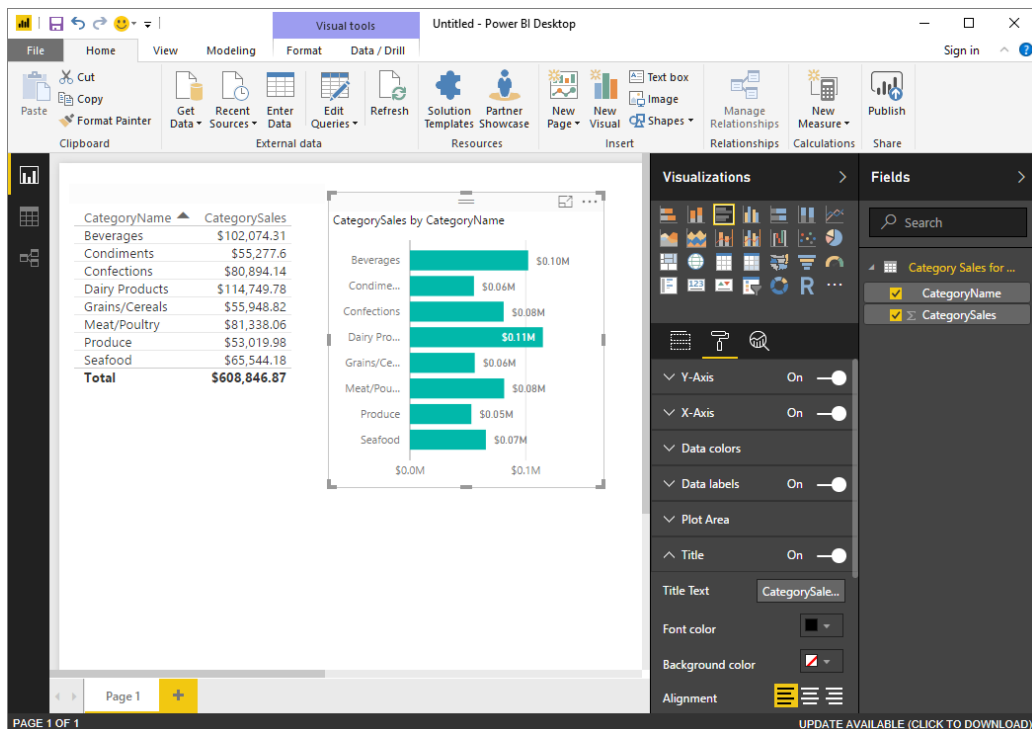
(SQL – Level 3 – Non-relational SQL Server Features and Tools)

The next several **screens look like** the database **connector in Excel**. This because Power BI functionality was integrated into Excel's most recent versions.

You would configure these screens by **filling in** your server's name, the name of your **database**, the **username** and **password**, and the **database object** or query you wish to use for your reports data.



Upon using the "**Load**" button on the last dialogue window, you will be presented with the Power BI editing tools. Here you will be able to **change the fonts**, and **other visual** aspects of your report.



Demo 3: Creating PowerBI Reports

In this demo, we use Visual Studio SSRS Project to create a report using view in Demo1.

Non-Microsoft

There are plenty of other companies out there providing reporting software. Some examples include:

- IBM Cognos
- Tableau Business Intelligence
- SAS Business Intelligence
- SAP Business Objects Web Intelligence
- SAP Crystal Reports

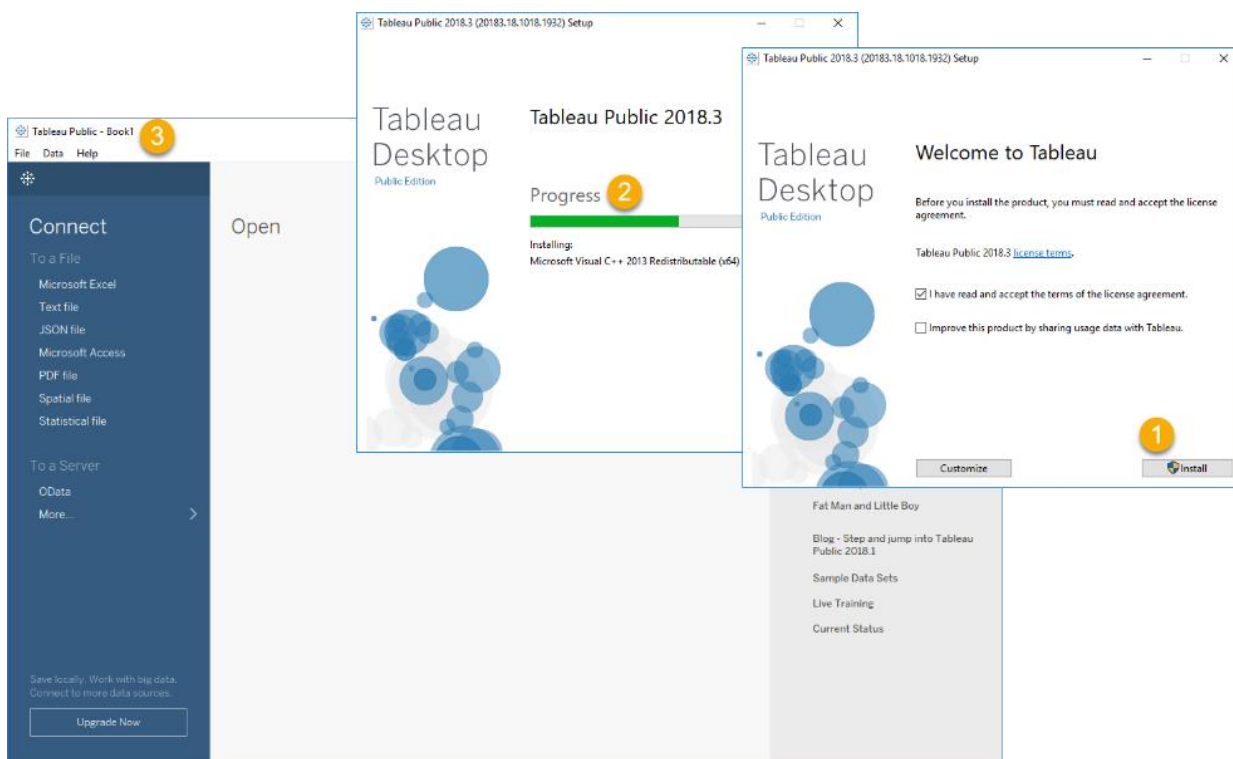
Tableau Reports

Tableau has become the most popular reporting tool out there, followed by Microsoft's Power BI for its early use of cloud-based reporting. **Still**, the **most common** reporting application you are likely to see is Microsoft's **Excel due to it being readily available on most users' computers!**

"Create and share interactive charts and graphs, stunning maps, live dashboards and fun applications in minutes, then publish anywhere on the web. Anyone can do it, it's that easy—and it's free." (<https://public.tableau.com/en-us/s/download>, 2018)

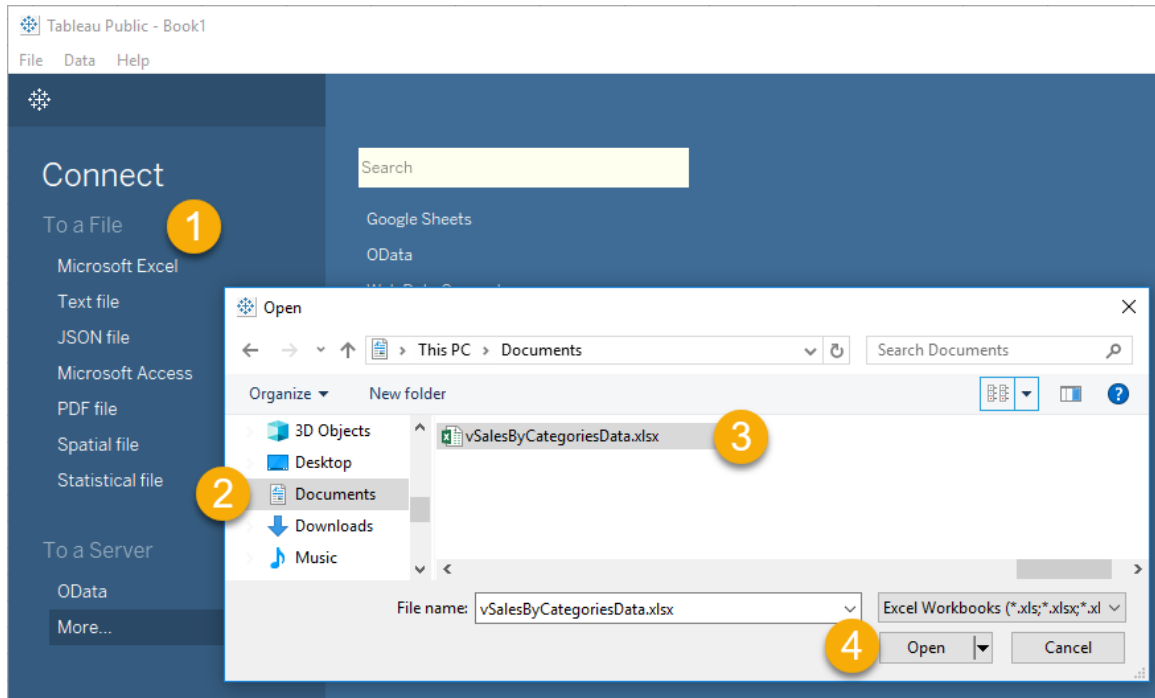
Important: Unlike Microsoft's PowerBI, the **Tableau public edition does not let you save your work as a file or connect directly to a database**, so you have to pull data from a file and save your report using a screenshot!

You can download Tableau Public edition from their website and start the Installer. When it finishes, it should launch the application, so you can start building reports right away!

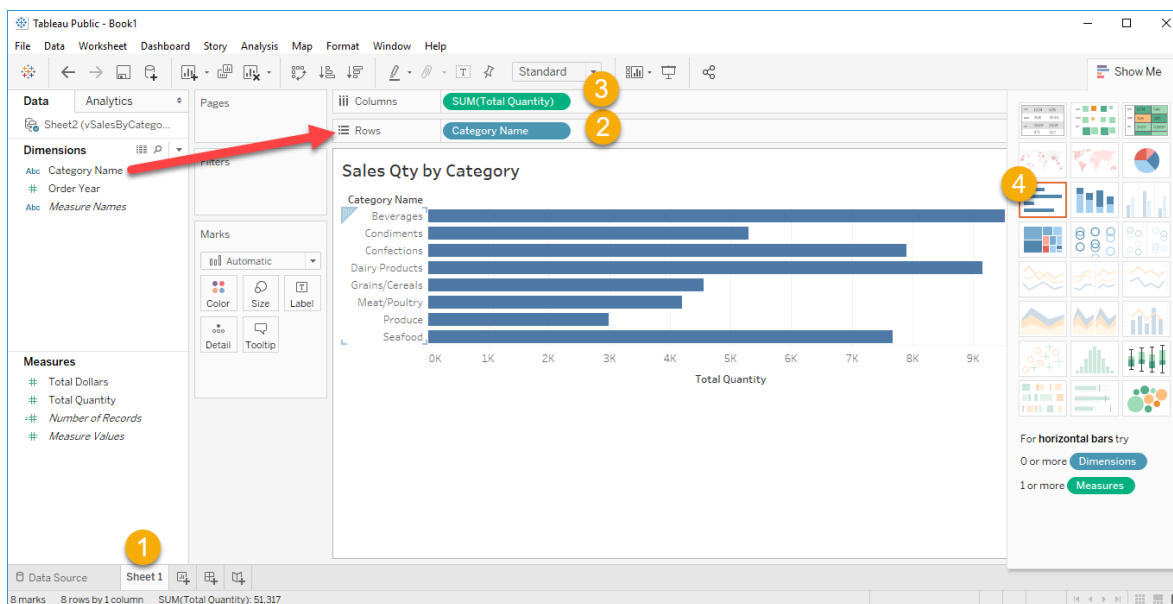


(SQL – Level 3 – Non-relational SQL Server Features and Tools)

The first step in building a report is to connect to a data source. Tableau public does not let you connect to a SQL Server database (You need to pay more to do that!), but it can connect to both Excel and Access.



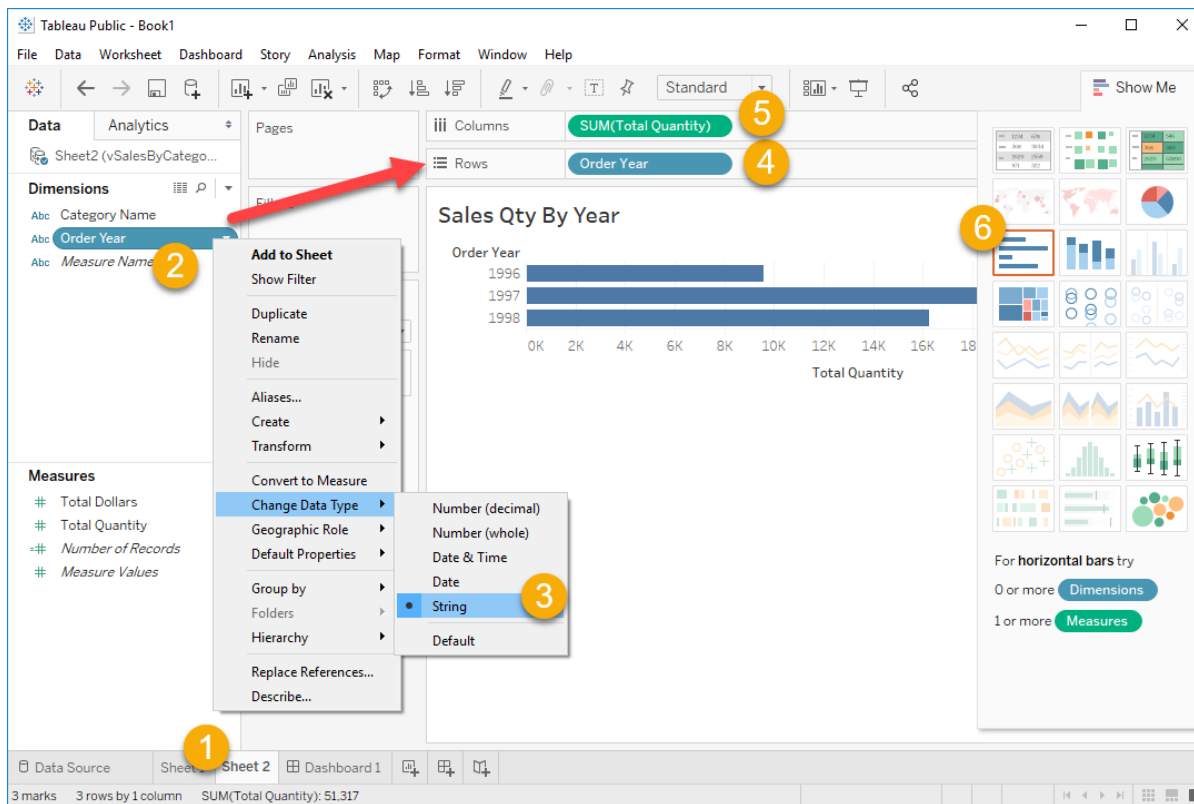
Once you have connected to a data source, you configure a Tableau's Book's Sheet to create reporting objects. You can create many different sheets in a single Book. Each can show a different aspect of the data.



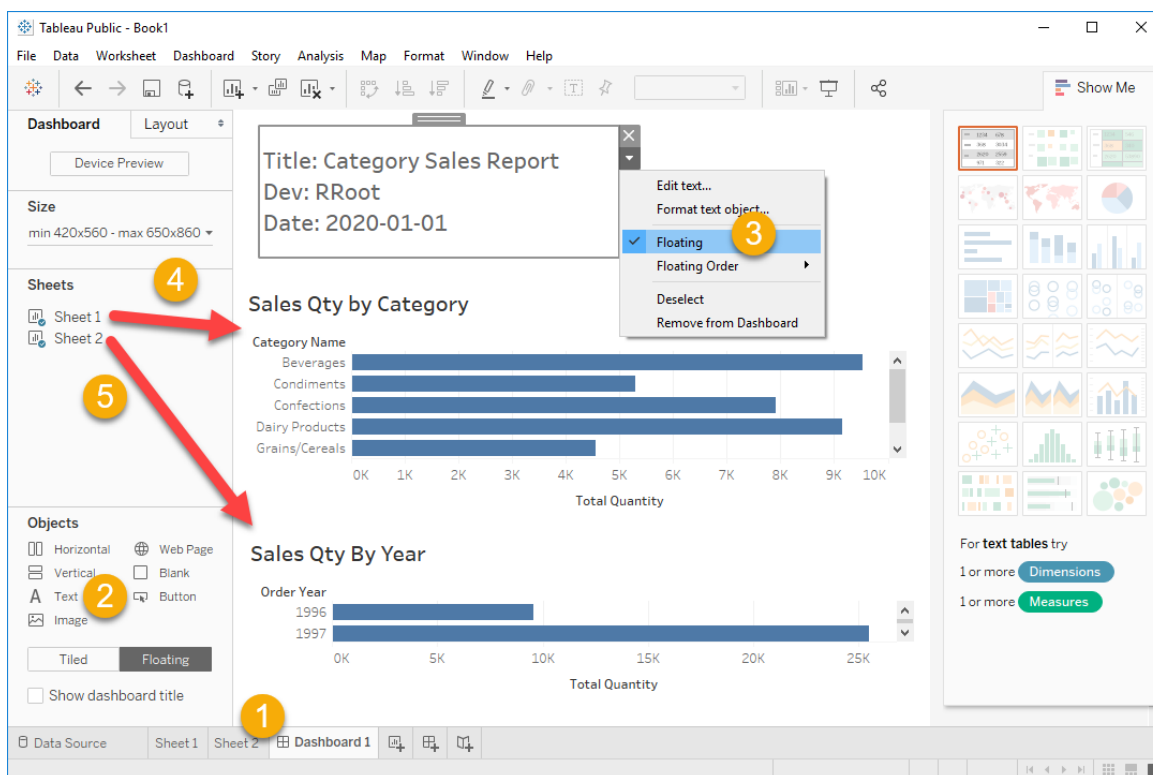
Tip: The use of Dimension and Measures are terms we saw when we looked at the Data Warehouse design.

Tableau automatically chooses data types for your dimensions and measures. Sometimes you will need to change these data types to create the reports you want.

(SQL – Level 3 – Non-relational SQL Server Features and Tools)



Individual Sheets are collected into Dashboards. Dashboards can also contain other objects, like a Text area where a report header can be added.



Demo 4: Creating Tableau Reports

In this lab, you will use Tableau to create a report using data from the Excel Spreadsheet in Demo 1.

Custom Applications

In addition to using Pre-made **database applications**, you can **create your own using languages**, like Microsoft's **C#** or Open Source **Python**. **Both** languages are **simple to learn**, but **Python** is more forgiving, while **C#** has more advanced programming options. **Both** languages now **run on Windows, Linux, and Mac OS**. **Both** languages are **Free** to use for production development.

There are **many** different **languages** and technologies used to create data-driven applications. You can **expect** them to follow the **same pattern** as seen in both the Python and C# examples:

- **Open a connection**
- **Issue a command**
- **Process any results**
- **Close a connection**

While the code may be a bit different in every language, if you **look for this pattern**, you should be able to figure out how the code works!

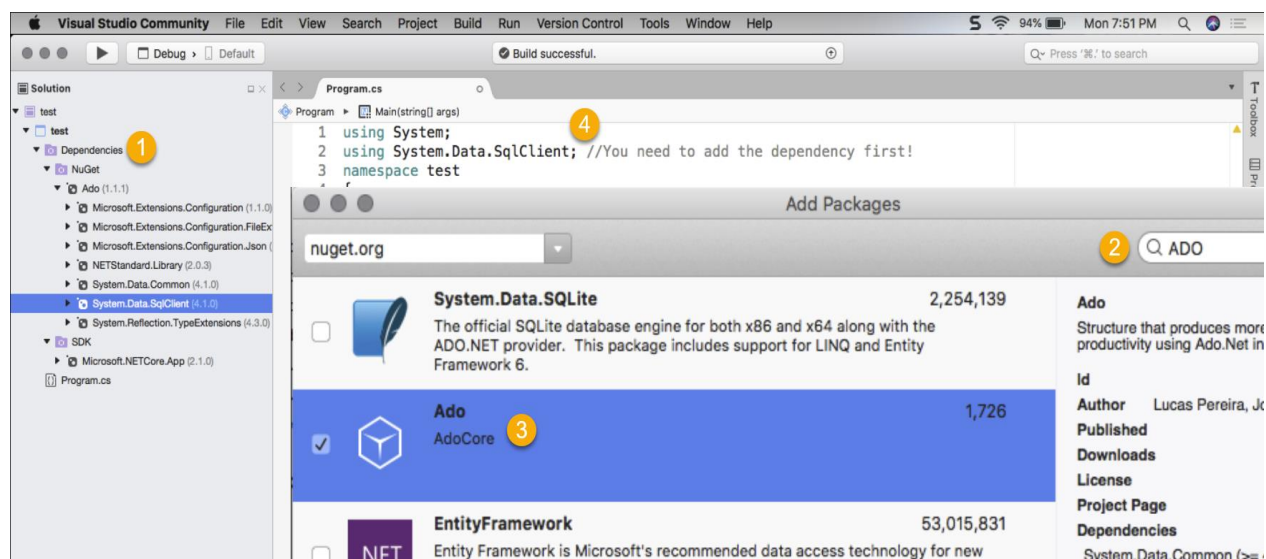
Microsoft's C#

"You can use C# to create **Windows client applications, XML Web services, distributed components, client-server applications, database applications**, and much, much more." (<https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx>)

You can now run C# on both Windows and Mac OS using Visual Studio (VS). The free version of VS is called the Community Edition, and it is, "A fully-featured, extensible, free IDE for creating modern applications for Android, iOS, Windows, as well as web applications and cloud services." (<https://visualstudio.microsoft.com/vs/community/>, 2018)



Once installed you will also need add an additional component called Active Data Objects to access SQL Server databases.



Active Data Objects .Net

"ADO.NET is a **set of classes that expose data access** services to the .NET programmer." (<https://msdn.microsoft.com/en-us/library/aa286484.aspx>)

Here is some example code to try out:

```
using System;
using System.Data.SqlClient;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            SqlConnectionStringBuilder objSB;
            objSB = new SqlConnectionStringBuilder();
            objSB.DataSource = "continuumsql.westus2.cloudapp.azure.com";
            objSB.InitialCatalog = "Northwind";
            objSB.UserID = "BICert";
            objSB.Password = "BICert";

            System.Data.SqlClient.SqlConnection objCon;
            objCon = new System.Data.SqlClient.SqlConnection();
            objCon.ConnectionString = objSB.ConnectionString;
            objCon.Open();

            System.Data.SqlClient.SqlCommand objCmd;
            objCmd = new System.Data.SqlClient.SqlCommand();
            objCmd.Connection = objCon;
            string strCategory = "seafood";
            objCmd.CommandText = "Exec pSelSalesByCategory @CategoryName = " + strCategory + ";";

            System.Data.SqlClient.SqlDataReader objDR;
            objDR = objCmd.ExecuteReader(); //Cursor

            while (objDR.Read() == true)
            {
                Console.WriteLine(objDR[0] + "," + objDR[1]);
            }
            objDR.Close();
            objCon.Close();
        }
    }
}
```

Note: We will talk more about **how applications work** with databases **over the next two modules**.

Demo 5: Creating Report Data Files with C#

In this demo, we use Visual Studio SSRS Project to create a report using view in Demo1.

Advanced Reporting Functions

No matter which technology you use, **all report strategies require valid and well formed data** in order to extract useful information. To get that data from a relational database you need SQL programming skills and a knowledge of various reporting functions. Let's look at some **advanced functions in SQL Server that are very useful for reports**.

Partitioned or Windowed Functions

Partitioned or **Windowed Functions** allow you to group data differently than the standard Group By clause.

Here is an example of a **SELECT** statement with a simple Group By clause.

-- Get the Min and Max Qty per store

Select

[StoreID] = stor_id

,[store min was] = min(qty)

,[store max was] = max(qty)

From Pubs.dbo.sales

Group By stor_id

Order By stor_id;

	stor_id	store min was	store max was
1	6380	3	5
2	7066	50	75
3	7067	10	40
4	7131	15	25
5	7896	10	35
6	8042	10	30

Note how just **adding an individual order quantities** doesn't work correctly, because **each line becomes part of its own group**

-- Get the Min and Max Qty per store and order

Select

[StoreID] = stor_id

,[TitleId] = title_id

,[OrderQty] = qty-- does NOT work correctly

,[store min was] = min(qty)

,[store max was] = max(qty)

From Pubs.dbo.sales

Group By stor_id, title_id , qty -- qty is used for grouping

Order By stor_id, qty;

	StoreID	TitleId	OrderQty	store min was	store max was
1	6380	PS2091	3	3	3
2	6380	BU1032	5	5	5
3	7066	PC8888	50	50	50
4	7066	PS2091	75	75	75
5	7067	PS2091	10	10	10
6	7067	TC4203	20	20	20
7	7067	TC7777	20	20	20
8	7067	TC3218	40	40	40

Over function

Using the **Over-Partition** function lets you **group by only some of the columns**.

<https://docs.microsoft.com/en-us/sql/t-sql/queries/select-over-clause-transact-sql>

-- Get the Min and Max Qty per store and order

Select

[StoreID] = stor_id

,[TitleID] = title_id

,[OrderQty] = qty -- Works!

,[store min was] = **min**(qty) **over**(partition by stor_id)

,[store max was] = **max**(qty) **over**(partition by stor_id)

From Pubs.dbo.sales

Order By stor_id;

	StoreID	TitleID	OrderQty	store min was	store max was
1	6380	BU1032	5	3	5
2	6380	PS2091	3	3	5
3	7066	PC8888	50	50	75
4	7066	PS2091	75	50	75
5	7067	PS2091	10	10	40
6	7067	TC3218	40	10	40
7	7067	TC4203	20	10	40
8	7067	TC7777	20	10	40

Ranking Functions

"Ranking functions return a ranking value for each row in a partition. Depending on the function that is used, some rows might receive the same value as other rows." -- <https://docs.microsoft.com/en-us/sql/t-sql/functions/ranking-functions-transact-sql>

SELECT o.OrderID, o.CustomerID, od.Quantity,

-- Ignore duplicate Values in the (Quantity) column

ROW_NUMBER() **OVER**(**ORDER BY** od.Quantity) AS rownum,

-- Group duplicate Values in the (Quantity) column and tells how many rows have come before it

RANK() **OVER**(**ORDER BY** od.Quantity) AS rank,

-- Groups duplicate Values in the (Quantity) column and tells what the last row NUMBER came before it

DENSE_RANK() **OVER**(**ORDER BY** od.Quantity) AS dense_rank,

-- Divide rows into groups based in the number of groups you ask for. Ntile(2) would be 2 groups.

NTILE(2) **OVER**(**ORDER BY** od.Quantity) AS ntile

FROM Northwind.dbo.Orders as o Join Northwind.dbo.[Order Details] as od

On o.OrderID = od.OrderID

Where CustomerID = 'ALFKI'

ORDER BY od.Quantity;

	OrderID	CustomerID	Quantity	rownum	rank	dense_rank	ntile
1	10643	ALFKI	2	1	1	1	1
2	10835	ALFKI	2	2	1	1	1
3	10952	ALFKI	2	3	1	1	1
4	10702	ALFKI	6	4	4	2	1
5	10702	ALFKI	15	5	5	3	1
6	10835	ALFKI	15	6	5	3	1
7	10643	ALFKI	15	7	5	3	2
8	10952	ALFKI	16	8	8	4	2
9	10692	ALFKI	20	9	9	5	2
10	11011	ALFKI	20	10	9	5	2
11	10643	ALFKI	21	11	11	6	2
12	11011	ALFKI	40	12	12	7	2

Extracting a Percentage

A common task is to calculate the percentage of one value compared to another. Here is how it can be done with SQL code.

Tip: Creating an artificial column just for sorting my data is useful in reporting!

```
Select [sortcolumn] = 'a', [result] = (3 / 9) Union
      Select 'b', (3 * 100 / 9) Union
      Select 'c', Cast((3 * 100 / 9) as float) Union
      Select 'd', (1. * 3 * 100 / 9) Union -- No need to cast to float now!
      Select 'e', (1. * 12 * 100 / 706) -- Ex. 12 products of 706 total products
```

Order By [sortcolumn]

Go

	sortcolumn	result
1	a	0
2	b	33
3	c	33
4	d	33.333333
5	e	1.699716

Here is an example of how that code could be used.

```
Select
  [stor_id]
,[title_id]
,[QtyByTitle] = Sum(Qty) Over (Partition By [stor_id])
,[QtyByStoreAndTitle] = Sum(Qty) Over (Partition By [title_id], [stor_id]) -- The Order Matters here!
,[Title Percent Per Store] = Cast((1. * (Sum(Qty) Over (Partition By [title_id], [stor_id])) * 100)
                               /
                               Sum(Qty) Over (Partition By [stor_id]) as decimal(10,2))
```

From pubs.dbo.sales

Order By 1;

	stor_id	title_id	QtyByTitle	QtyByStoreAndTitle	Title Percent Per Store
1	6380	BU1032	8	5	62.50
2	6380	PS2091	8	3	37.50
3	7066	PC8888	125	50	40.00
4	7066	PS2091	125	75	60.00
5	7067	PS2091	90	10	11.11
6	7067	TC3218	90	40	44.44
7	7067	TC4203	90	20	22.22
8	7067	TC7777	90	20	22.22

Lag and Lead

These functions are very useful in reporting! It shows the previous or following values based on a given group of values.

Here is an example that shows the following year's values.

Select

```
[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
,[PreviousYearlyTotalQty] = lead(Sum(Quantity)) Over(Order By Year(OrderDate))
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
Group By Year(OrderDate);
Go
```

	OrderYear	YearlyTotalQty	PreviousYearlyTotalQty
1	1996	7800184	20937336
2	1997	20937336	13855590
3	1998	13855590	NULL

Here is another example that shows the previous year's values.

Select

```
[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
,[PreviousYearlyTotalQty] = Lag(Sum(Quantity)) Over(Order By Year(OrderDate))
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
Group By Year(OrderDate);
go
```

	OrderYear	YearlyTotalQty	PreviousYearlyTotalQty
1	1996	7800184	NULL
2	1997	20937336	7800184
3	1998	13855590	20937336

Using Functions for Reporting

To create reporting queries, you start off with a simple Select statement and then build on it by adding more and more detailed code to finally get what you want as a result.

To create reporting queries, you start off with a simple Select statement like this one.

Select Distinct

```
OrderDate
From Northwind.dbo.Orders;
```

	OrderDate
1	1996-07-04 00:00:00.000
2	1996-07-05 00:00:00.000
3	1996-07-08 00:00:00.000

Next, add simple functions and test the results!

Select Distinct

```
[OrderYear] = Year(OrderDate)
From Northwind.dbo.Orders;
```

	OrderYear
1	1998
2	1996
3	1997

Then, add more columns, functions, or tables as needed.

Select Distinct

```
[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
Group By Year(OrderDate);
```

	OrderYear	YearlyTotalQty
1	1998	13855590
2	1996	7800184
3	1997	20937336

Keep adding more complex function as needed (like using the Lag Function).

Select

```
[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
,[PreviousYearlyTotalQty] = Lag(Sum(Quantity)) Over(Order By Year(OrderDate))
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
```

```
Group By Year(OrderDate);
go
```

	OrderYear	YearlyTotalQty	PreviousYearlyTotalQty
1	1996	7800184	NULL
2	1997	20937336	7800184
3	1998	13855590	20937336

Continue adding more features until you get what you are looking for.

Select

```
ProductName
,[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
,[PreviousYearlyTotalQty] =
    IIF(Year(OrderDate) = 1996, 0, Lag(Sum(Quantity)) Over (Order By ProductName, Year(OrderDate)))
,[Bad-PreviousYearlyTotalQty] = IsNull(Lag(Sum(Quantity)) Over (Order By ProductName, Year(OrderDate)), 0) -- Don't use Is Null!
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
Join Northwind.dbo.Products as P
On OD.ProductID = P.ProductID
Group By ProductName, Year(OrderDate);
go
```

	ProductName	OrderYear	YearlyTotalQty	PreviousYearlyTotalQty	Bad-PreviousYearlyTotalQty
1	Alice Mutton	1996	148656	0	0
2	Alice Mutton	1997	399024	148656	148656
3	Alice Mutton	1998	264060	399024	399024
4	Aniseed Syrup	1996	49856	0	264060
5	Aniseed Syrup	1997	133824	49856	49856
6	Aniseed Syrup	1998	88560	133824	133824
7	Boston Crab Meat	1996	167656	0	88560

Once you feel getting too complex or if you think you might reuse the results, create a reporting View like this one.

Create -- Drop

View vProductOrderQtyByYear

AS

Select

```
ProductName
,[OrderYear] = Year(OrderDate)
,[YearlyTotalQty] = Sum(Quantity)
,[PreviousYearlyTotalQty] = IIF(Year(OrderDate) = 1996, 0, Lag(Sum(Quantity)) Over (Order By ProductName, Year(OrderDate)))
From Northwind.dbo.Orders as O
Join Northwind.dbo.[Order Details] as OD
On o.OrderID = o.OrderID
Join Northwind.dbo.Products as P
On OD.ProductID = P.ProductID
Group By ProductName, Year(OrderDate);
Go
```

When using the View, you can always add on more functions -- as needed. For example, our current view makes it easy to create a Key Performance Indicators (KPIs) report

Select

```
ProductName
,[OrderYear]
,YearlyTotalQty
,PreviousYearlyTotalQty
,[QtyChangeKPI] = Case
When YearlyTotalQty > PreviousYearlyTotalQty Then 1
When YearlyTotalQty = PreviousYearlyTotalQty Then 0
When YearlyTotalQty < PreviousYearlyTotalQty Then -1
End
From vProductOrderQtyByYear
Go
```

	ProductName	OrderYear	YearlyTotalQty	PreviousYearlyTotalQty	QtyChangeKPI
1	Alice Mutton	1996	148656	0	1
2	Alice Mutton	1997	399024	148656	1
3	Alice Mutton	1998	264060	399024	-1
4	Aniseed Syrup	1996	49856	0	1
5	Aniseed Syrup	1997	133824	49856	1
6	Aniseed Syrup	1998	88560	133824	-1
7	Boston Crab Meat	1996	167656	0	1