



## **BHAGWAN PARSHURAM INSTITUTE OF TECHNOLOGY**

**INFORMATION TECHNOLOGY (IT)**

**BATCH: 2022-26**

**BUSINESS INTELLIGENCE (DS-431P)**

**SUBMITTED TO:**

**Dr. POOJA MUDGIL**

**SUBMITTED BY:**

**NAME: VISHAL KUMAR**

**ENROLLMENT: 13720803122**

**PROGRAMME: IT-C**

# INDEX

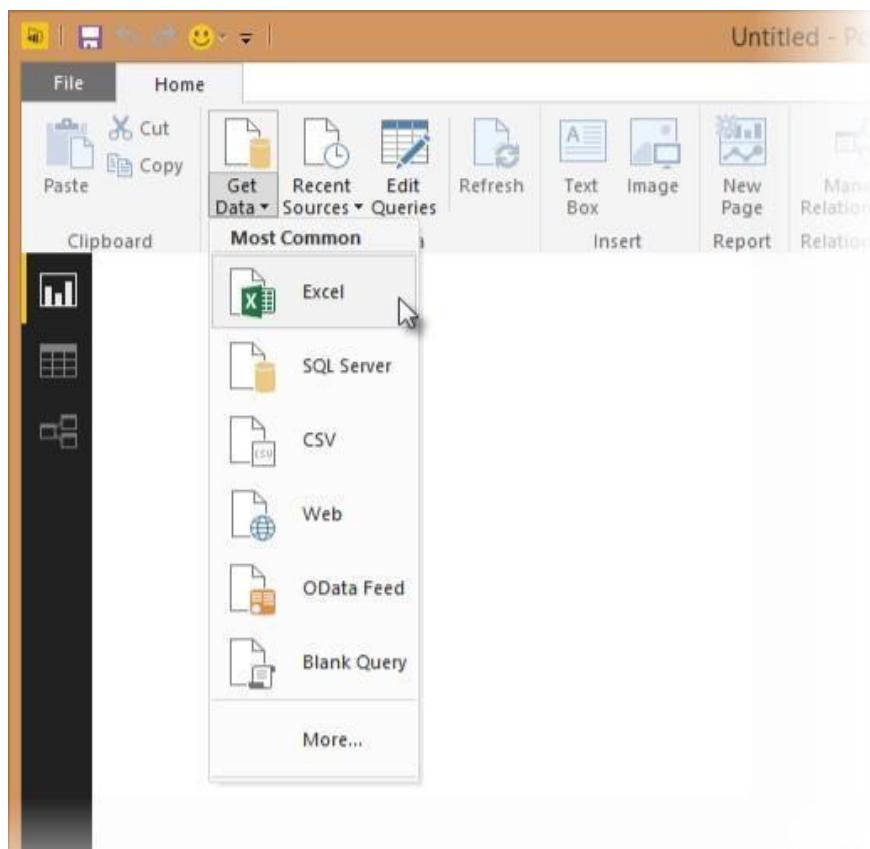
Sr. No.	Title of Lab Experiments	Date
1.	Import the legacy data from different sources such as (Excel, SqlServer, Oracle etc.) and load in the target system	
2.	Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sqlserver / Power B	
3.	Data Visualization from ETL Process	
4.	Creating a Data Cubes	
5.	Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data.	
6.	Implementation of Classification algorithm.	
7.	Practical Implementation of Decision Tree	
8.	Implementation of k-means clustering.	
9.	Prediction Using Linear Regression	
10.	Data Analysis using Time Series Analysis	
11.	Data Modelling and Analytics with Pivot Table in Excel	
12.	Data Analysis and Visualization using Advanced Excel	

# Practical 1

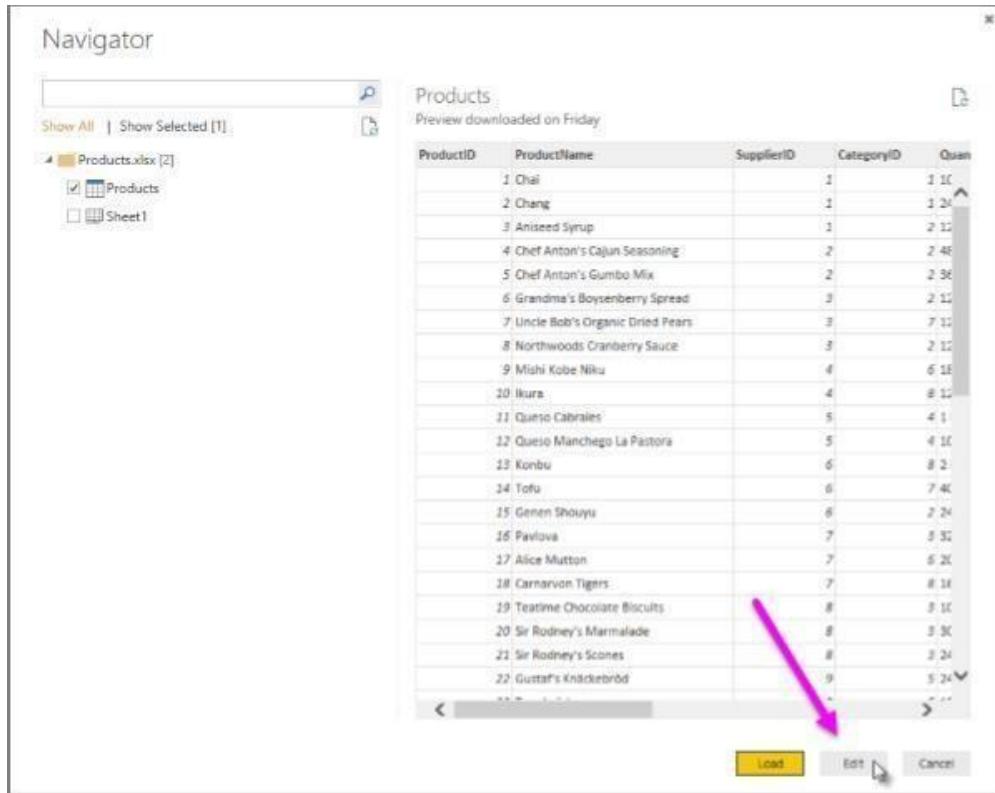
**AIM:** Import the legacy data from different sources such as (Excel, SqlServer, Oracle etc.) and load in the target system.

## Importing Excel Data

- 1) Launch Power BI Desktop.
- 2) From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu.



- 3) If you select the Get Data button directly, you can also select File > Excel and select Connect.
- 4) In the Open File dialog box, select the Products.xlsx file.
- 5) In the Navigator pane, select the Products table and then select Edit.



## Importing Data from OData Feed

In this task, you'll bring in order data. This step represents connecting to a sales system. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:

<http://services.odata.org/V3/Northwind/Northwind.svc/>

Connect to an OData feed:

- 1) From the Home ribbon tab in Query Editor, select Get Data.
- 2) Browse to the OData Feed data source.
- 3) In the OData Feed dialog box, paste the URL for the Northwind OData feed.
- 4) Select OK.
- 5) In the Navigator pane, select the Orders table, and then select Edit.

**Navigator**

Show All   Show Selected [1]	
http://services.odata.org/V3/Northwind/Nort...	<input type="button" value=""/>
<input type="checkbox"/> Alphabetical_list_of_products	
<input type="checkbox"/> Categories	
<input type="checkbox"/> Category_Sales_for_1997	
<input type="checkbox"/> Current_Product_Lists	
<input type="checkbox"/> Customer_and_Suppliers_by_Cities	
<input type="checkbox"/> CustomerDemographics	
<input type="checkbox"/> Customers	
<input type="checkbox"/> Employees	
<input type="checkbox"/> Invoices	
<input type="checkbox"/> Order_Details	
<input type="checkbox"/> Order_Details_Extendeds	
<input type="checkbox"/> Order_Subtotals	
<input checked="" type="checkbox"/> Orders	
<input type="checkbox"/> Orders_Qries	
<input type="checkbox"/> Product_Sales_for_1997	
<input type="checkbox"/> Products	
<input type="checkbox"/> Products_Above_Average_Prices	
<input type="checkbox"/> Products_by_Categories	
<input type="checkbox"/> Regions	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

**Orders**

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996
10249	TOMSP	6	7/5/1996 12:00:00 AM	8/16/1996
10250	HANAR	4	7/8/1996 12:00:00 AM	8/5/1996
10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996
10252	SUPRO	4	7/9/1996 12:00:00 AM	8/6/1996
10253	HANAR	3	7/10/1996 12:00:00 AM	7/24/1996
10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996
10255	RICSU	9	7/12/1996 12:00:00 AM	8/9/1996
10256	WELLI	3	7/15/1996 12:00:00 AM	8/12/1996
10257	HILAA	4	7/16/1996 12:00:00 AM	8/13/1996
10258	ERNSH	1	7/17/1996 12:00:00 AM	8/14/1996
10259	CENTC	4	7/18/1996 12:00:00 AM	8/15/1996
10260	OTTIK	4	7/19/1996 12:00:00 AM	8/16/1996
10261	QUEDE	4	7/19/1996 12:00:00 AM	8/16/1996
10262	RATTIC	8	7/22/1996 12:00:00 AM	8/19/1996
10263	ERNSH	9	7/23/1996 12:00:00 AM	8/20/1996
10264	POLKO	6	7/24/1996 12:00:00 AM	8/21/1996
10265	BLOTP	2	7/25/1996 12:00:00 AM	8/22/1996
10266	WARTH	3	7/26/1996 12:00:00 AM	8/23/1996
10267	FRANK	4	7/29/1996 12:00:00 AM	8/26/1996
10268	GROSR	8	7/30/1996 12:00:00 AM	8/27/1996
10269	WHITC	5	7/31/1996 12:00:00 AM	8/14/1996
10270	WARTH	1	8/1/1996 12:00:00 AM	8/29/1996

**Note -** You can click a table name, without selecting the checkbox, to see a preview.

## **Practical 2**

**AIM: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sqlserver / Power BI.**

**STEPS:**

### **Step 1 : Data Extraction :**

The data extraction is first step of ETL. There are 2 Types of Data Extraction

1. Full Extraction : All the data from source systems or operational systems gets extracted to staging area. (Initial Load).
2. Partial Extraction : Sometimes we get notification from the source system to update specific date. It is called as Delta load.
3. Source System Performance: The Extraction strategies should not affect source system performance.

### **Step 2 : Data Transformation :**

The data transformation is second step. After extracting the data there is big need to do the transformation as per the target system. I would like to give you some bullet points of Data Transformation.

- Data Extracted from source system is in to Raw format. We need to transform it before loading in to target server.
- Data has to be cleaned, mapped and transformed
- There are following important steps of Data Transformation :

**1.Selection :** Select data to load in target

**2.Matching :** Match the data with target system

**3.Data Transforming :** We need to change data as per target table structures

### **Real life examples of Data Transformation :**

- Standardizing data : Data is fetched from multiple sources so it needs to be standardized as per the target system.
- Character set conversion : Need to transform the character sets as per the target systems. (Firstname and last name example)
- Calculated and derived values: In source system there is first val and second val and in target we need the calculation of first val and second val.
- Data Conversion in different formats : If in source system date is in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.

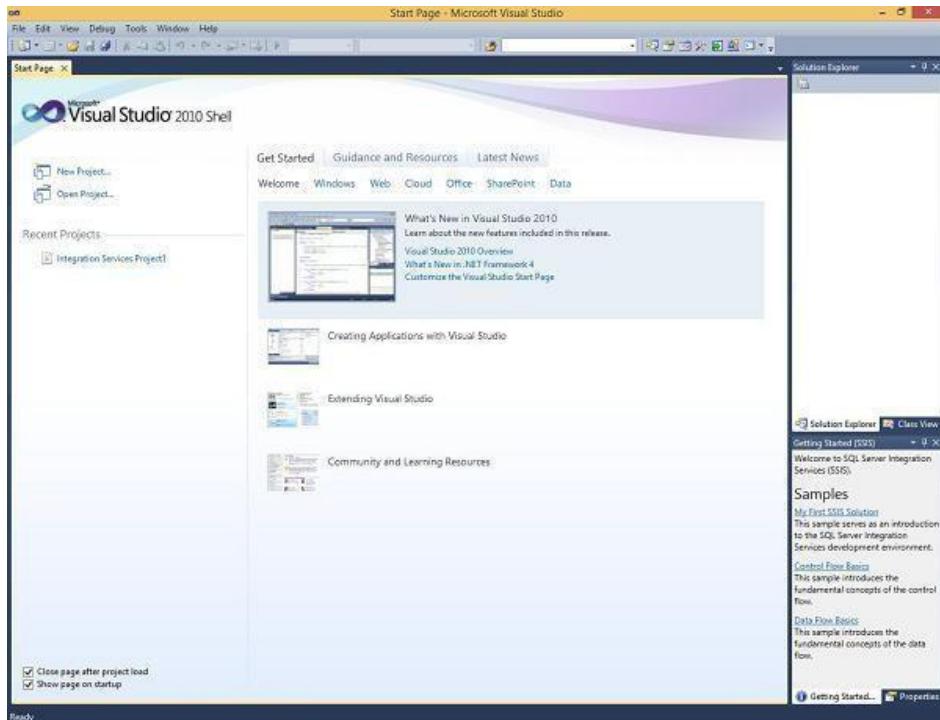
### **Step 3 : Data Loading**

- Data loading phase loads the prepared data from staging tables to main tables.

ETL process in SQL Server:

Following are the steps to open BIDS\SSDT.

**Step 1** – Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen appears.



**Step 2** – The above screen shows SSDT has opened. Go to file at the top left corner in the above image and click New. Select project and the following screen opens.

**Step 3** – Select Integration Services under Business Intelligence on the top left corner in the above screen to get the following screen.

**Step 4** – In the above screen, select either Integration Services Project or Integration Services Import Project Wizard based on your requirement to develop\create the package.

## Modes

There are two modes – Native Mode (SQL Server Mode) and Share Point Mode.

## Models

There are two models – Tabular Model (For Team and Personal Analysis) and Multi

Dimensions Model (For Corporate Analysis).

The BIDS (Business Intelligence Studio till 2008 R2) and SSDT (SQL Server Data Tools from 2012) are environments to work with SSAS.

**Step 1** – Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen will appear.

**Step 2** – The above screen shows SSDT has opened. Go to file on the top left corner in the above image and click New. Select project and the following screen opens.

**Step 3** – Select Analysis Services in the above screen under Business Intelligence as seen on the top left corner. The following screen pops up

**Step 4** – In the above screen, select any one option from the listed five options based on your requirement to work with Analysis services.

## ETL Process in Power BI

### 1) Remove other columns to only display columns of interest

In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **QuantityPerUnit**

Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

1. In **Query Editor**, select the **ProductID**, **ProductName**, **QuantityPerUnit**, and **UnitsInStock** columns (use **Ctrl+Click** to select more than one column, or **Shift+Click** to select columns that are beside each other).
2. Select **Remove Columns > Remove Other Columns** from the ribbon, or right-click on a column header and click Remove Other Columns.

The screenshot shows the Microsoft Power BI Query Editor interface. On the left, there's a navigation pane with options like 'Close & Load', 'New Source', 'Recent Sources', 'Refresh Preview', 'Properties', 'Advanced Editor', 'Choose Columns', 'Remove Columns', 'Reduce Rows', 'Split Column', 'Group By', 'Replace Values', and 'Combine'. The main area is titled 'Products - Query Editor' and contains a table named 'Table.TransformColumnTypes(Products\_Table)'. The table has columns: 'SupplierID', 'CategoryID', 'Quantity', 'UnitPrice', and 'UnitsInStock'. The 'UnitsInStock' column has a dropdown menu open, showing options like 'Remove Columns', 'Remove Other Columns', 'Remove Duplicates', 'Remove Errors', 'Replace Values...', 'Fill', 'Change Type', 'Transform', 'Group By...', 'Unpivot Columns', 'Unpivot Other Columns', and 'Move'. To the right of the table is the 'Query Settings' pane, which includes sections for 'PROPERTIES' (Name: Products) and 'APPLIED STEPS' (Source, Navigation, Changed Type). At the bottom of the editor, it says 'PREVIEW DOWNLOADED AT 9:44 AM'.

### 3. Change the data type of the UnitsInStock column

When Query Editor connects to data, it reviews each field and to determine the best data type. For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the **UnitsInStock** column's datatype is Whole Number.

1. Select the **UnitsInStock** column.
2. Select the **Data Type drop-down button** in the **Home ribbon**.
3. If not already a Whole Number, select **Whole Number** for data type from the drop down (the Data Type: button also displays the data type for the current selection).

### 2. Expand the Order\_Details table

The Orders table contains a reference to a Details table, which contains the individual products that were included in each Order. When you connect to data sources with multiples tables (such as a relational database) you can use these references to build up your query. In this step, you expand the **Order\_Details** table that is related to the Orders table, to combine the **ProductID**, **UnitPrice**, and **Quantity** columns from **Order\_Details** into the **Orders** table. This is a representation of the data in these tables:

The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (**Order\_Details**) are combined into rows from the subject table (**Orders**).

After you expand the Order\_Details table, three new columns and additional rows are added to the Orders table, one for each row in the nested or related table.

1. In the Query View, scroll to the Order\_Details column.
2. In the Order\_Details column, select the expand icon ( ).
3. In the Expand drop-down:
  - a. Select (Select All Columns) to clear all columns.

b. Select ProductID, UnitPrice, and Quantity.

c. Click OK.

#### 4. Calculate the line total for each Order\_Details row

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order\_Details row.

Calculate the line total for each Order\_Details row:

1. In the Add Column ribbon tab, click Add Custom Column.
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order\_Details.UnitPrice] \* [Order\_Details.Quantity].
3. In the New column name textbox, enter LineTotal.
4. Click OK.

#### 5. Rename and reorder columns in the query

In this step you finish making the model easy to work with when creating reports, by renaming the final columns and changing their order.

1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.

The screenshot shows the Power BI Query Editor interface. On the left, there's a '2 Queries' pane with 'Products' and 'Orders' selected. The main area displays a table with four columns: 'ShipCity', 'LineTotal', 'Order\_Details.ProductID', and 'Order\_Details.UnitPrice'. The 'LineTotal' column is currently selected and highlighted. To the right of the table is the 'Query Settings' pane, which includes sections for 'PROPERTIES' (Name: Orders) and 'APPLIED STEPS' (listing 'Added Custom'). At the bottom of the editor, it says '7 COLUMNS, 999+ ROWS' and 'PREVIEW DOWNLOADED AT 9:52 AM'.

2. Remove the Order\_Details. prefix from the Order\_Details.ProductID, Order\_Details.UnitPrice and Order\_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.

#### 6. Combine the Products and Total Sales queries

Power BI Desktop does not require you to combine queries to report on them. Instead, you can create Relationships between datasets. These relationships can be created on any column that is common to your datasets

We have Orders and Products data that share a common 'ProductID' field, so we need to ensure there's a relationship between them in the model we're using with Power BI Desktop. Simply specify in Power BI Desktop that the columns from each table are related (i.e. columns that have the same values). Power BI Desktop works out the direction and

cardinality of the relationship for you. In some cases, it will even detect the relationships automatically.

In this task, you confirm that a relationship is established in Power BI Desktop between the Products and Total Sales queries

#### Step 1: Confirm the relationship between Products and Total Sales

1. First, we need to load the model that we created in Query Editor into Power BI Desktop. From the Home ribbon of Query Editor, select Close & Load.
2. Power BI Desktop loads the data from the two queries.
3. Once the data is loaded, select the Manage Relationships button Home ribbon.
4. Select the New... button
5. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.
6. Select Cancel, and then select Relationship view in Power BI Desktop. We see the following, which visualizes the relationship between the queries.
7. When you double-click the arrow on the line that connects the two queries, an Edit Relationship dialog appears.
8. No need to make any changes, so we'll just select Cancel to close the Edit Relationship dialog.

# Practical 3

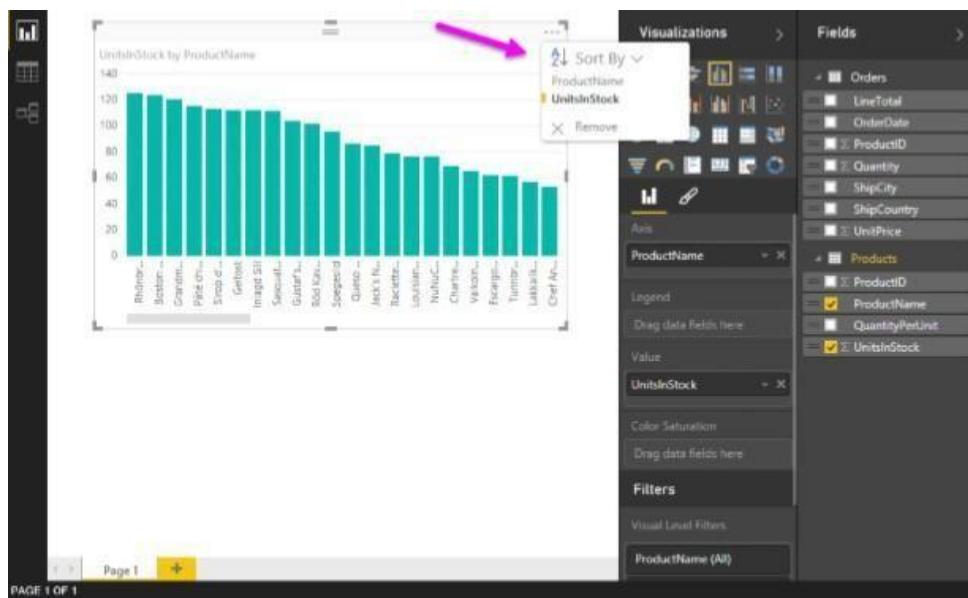
## AIM: Data Visualization from ETL Process

Power BI Desktop lets you create a variety of visualizations to gain insights from your data. You can build reports with multiple pages and each page can have multiple visuals. You can interact with your visualizations to help analyze and understand your data.

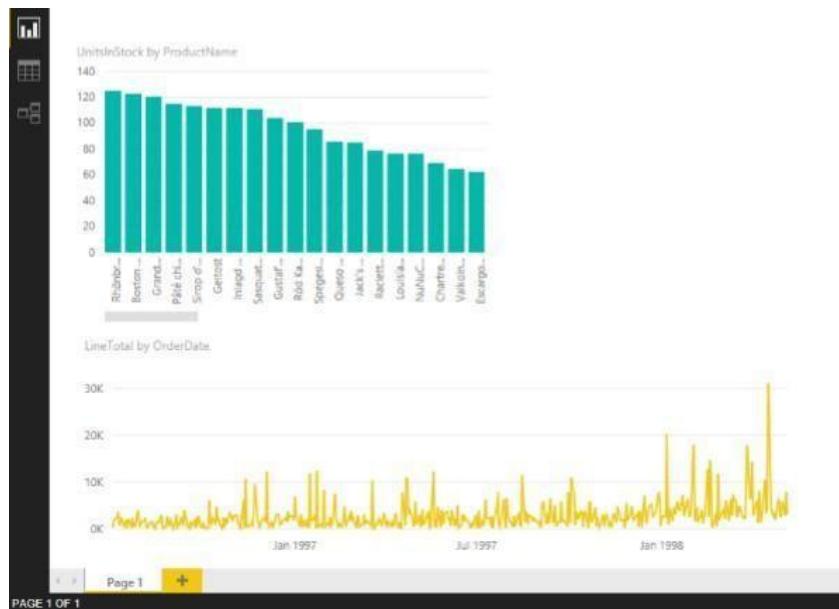
In this task, you create a report based on the data previously loaded. You use the Fields pane to select the columns from which you create the visualizations.

### Step 1: Create charts showing Units in Stock by Product and Total Sales by Year

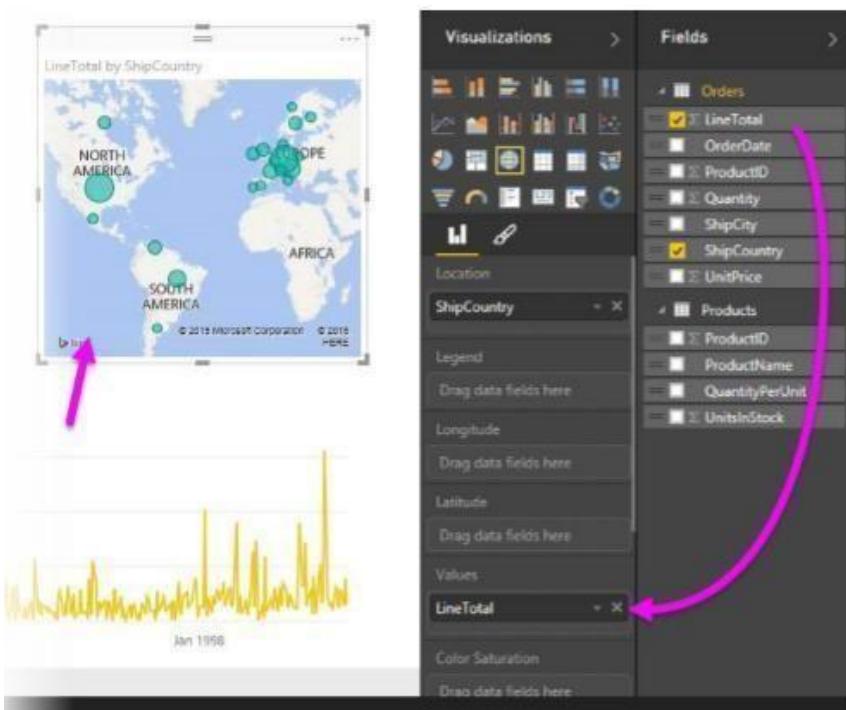
1. Drag UnitsInStock from the Field pane (the Fields pane is along the right of the screen) onto a blank space on the canvas. A Table visualization is created. Next, drag ProductName to the Axis box, found in the bottom half of the Visualizations pane. Then we then select Sort By > UnitsInStock using the skittles in the top right corner of the visualization.



2. Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, from the Fields pane) onto the visual, then select Line Chart. The following visualization is created.



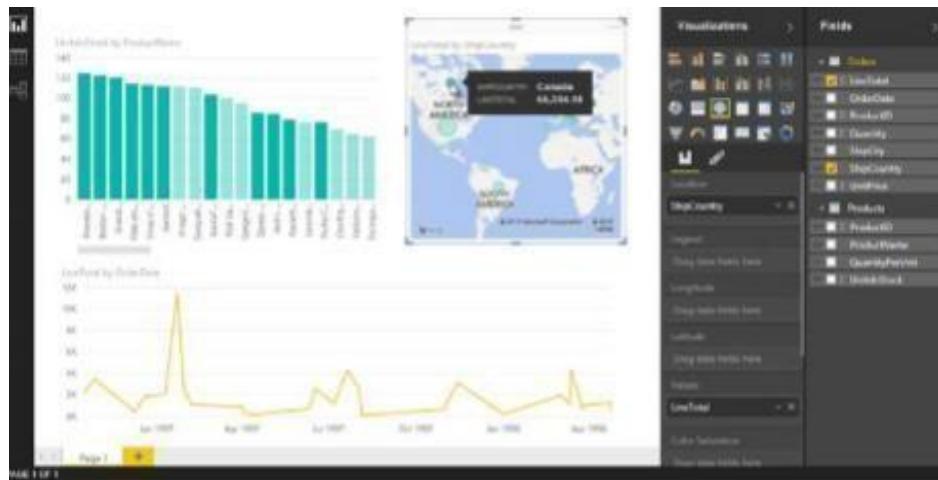
3. Next, drag ShipCountry to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag LineTotal to the Values field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.



## Step 2: Interact with your report visuals to analyze further

Power BI Desktop lets you interact with visuals that cross-highlight and filter each other to uncover further trends.

1. Click on the light blue circle centered in Canada. Note how the other visuals are filtered to show Stock (ShipCountry) and Total Orders (LineTotal) just for Canada.



# Practical 4

## AIM: Creating a Cube in SQL server 2012

### Creating Data Warehouse

Let us execute our T-SQL Script to create data warehouse with fact tables, dimensions and populate them with appropriate test values.

Download T-SQL script attached with this article for creation of Sales Data Warehouse or download from this article "[\*\*Create First Data Warehouse\*\*](#)" and run it in your SQL Server.

Follow the given steps to run the query in **SSMS** (SQL Server Management Studio).

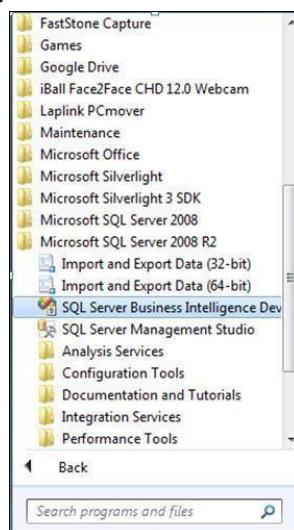
1. Open SQL Server Management Studio 2008
2. Connect Database Engine
3. Open **New Query** editor
4. Copy paste Scripts given below in various steps in new query editor window one by one
5. To run the given SQL Script, press **F5**
6. It will create and populate “Sales\_DW” database on your SQL Server

### Developing an OLAP Cube

For creation of OLAP Cube in Microsoft BIDS Environment, follow the 10 easy steps given below.

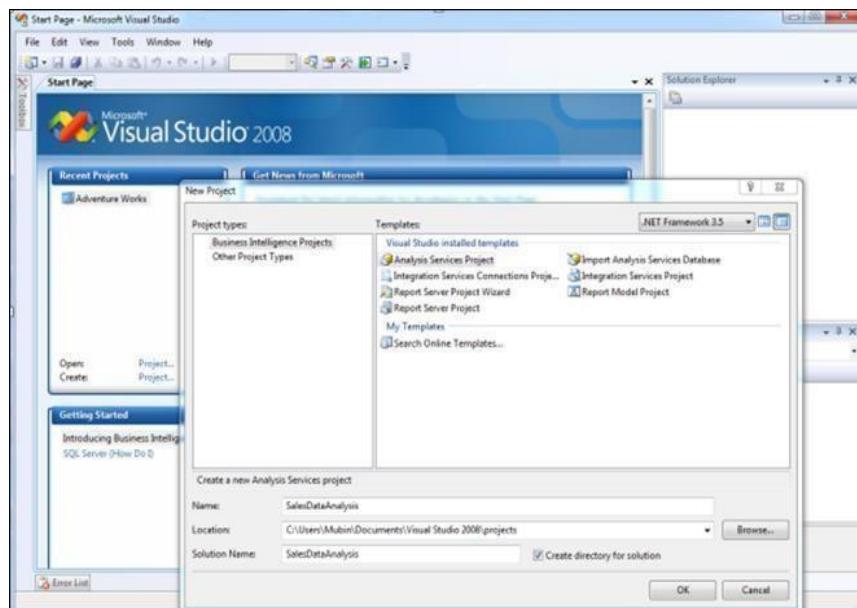
#### *Step 1: Start BIDS Environment*

Click on **Start Menu** -> Microsoft **SQL Server 2008 R2** -> Click **SQL Server Business Intelligence Development Studio**.



#### *Step 2: Start Analysis Services Project*

**Click File -> New -> Project ->Business Intelligence Projects ->select Analysis Services Project-> Assign Project Name -> Click OK**



### *Step 3: Creating New Data Source*

**3.1 In Solution Explorer, Right click on Data Source -> Click New Data Source**

### *Step 3: Creating New Data Source*

**3.1 In Solution Explorer, Right click on Data Source -> Click New Data Source**

**3.2 Click on Next**

**3.3 Click on New Button**

**3.4 Creating New connection**

1. Specify Your **SQL Server Name** where your Data Warehouse was created
2. Select Radio Button according to your **SQL Server Authentication** mode 3. Specify your **Credentials** using which you can connect to your SQL Server
4. Select database Sales\_DW.
5. Click on **Test Connection** and verify for its success
6. Click **OK**.

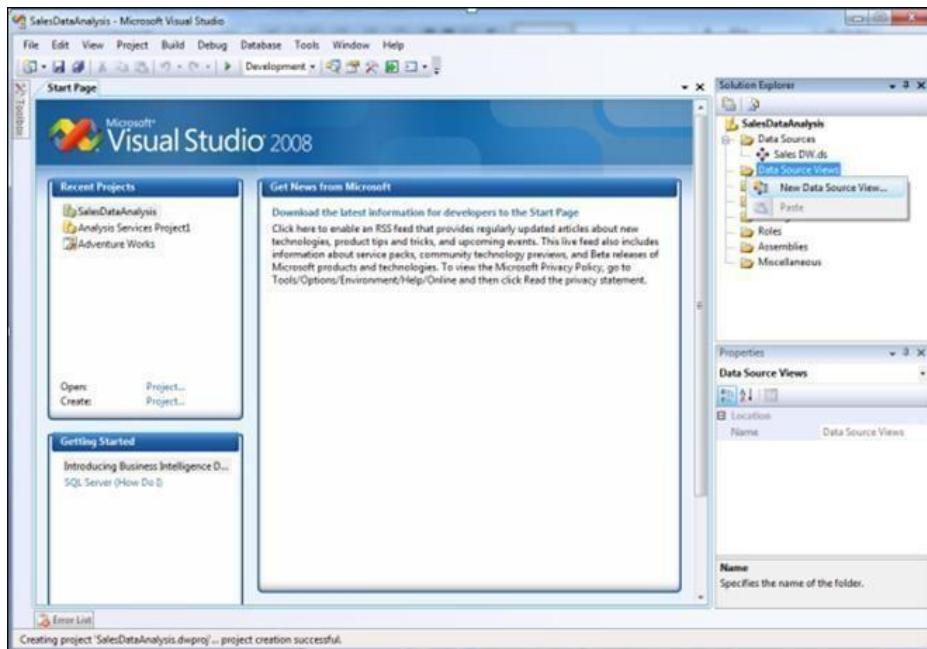
**3.5 Select Connection created in Data Connections-> Click Next**

**3.6 Select Option Inherit**

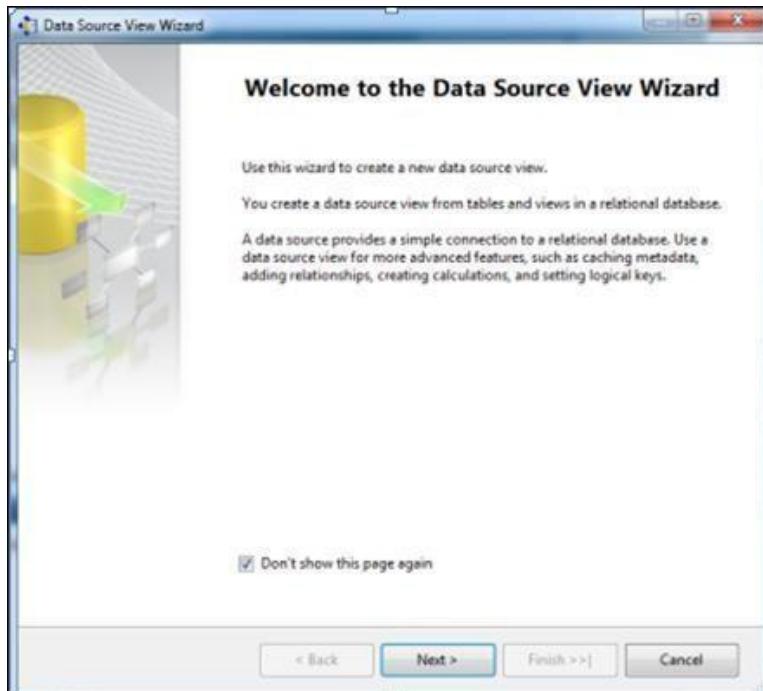
**3.7 Assign Data Source Name -> Click Finish**

### **Step 4: Creating New Data Source View**

**4.1 In the Solution Explorer, Right Click on Data Source View -> Click on New Data Source View**



#### 4.2 Click Next

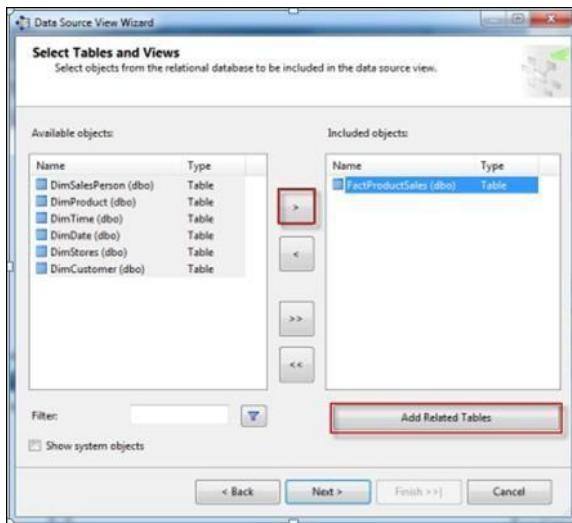


#### 4.3 Select Relational Data Source we have created previously (Sales\_DW)-> Click Next

4.4 First move your **Fact Table** to the right side to include in object list. Select FactProductSales Table -> Click on Arrow Button to move the selected object to Right Pane.

4.5 Now to **add dimensions** which are **related** to your **Fact Table**, follow the given steps:

Select **Fact Table** in Right Pane (Fact product Sales) -> Click On **Add Related Tables**



4.6 It will add all associated dimensions to your Fact table as per relationship specified in your SQL DW (Sales\_DW).

Click **Next**.

4.7 Assign Name (**SalesDW DSV**)-> Click **Finish**

4.8 Now Data Source View is ready to use.

#### *Step 5: Creating New Cube*

5.1 In Solution Explorer -> Right Click on **Cube**-> Click **New Cube**

5.2 Click **Next**

5.3 Select Option **Use existing Tables** -> Click **Next**

5.4 Select Fact Table Name from **Measure Group Tables (FactProductSales)** -> Click **Next**

5.5 Choose **Measures** from the List which you want to place in your Cube --> Click **Next**

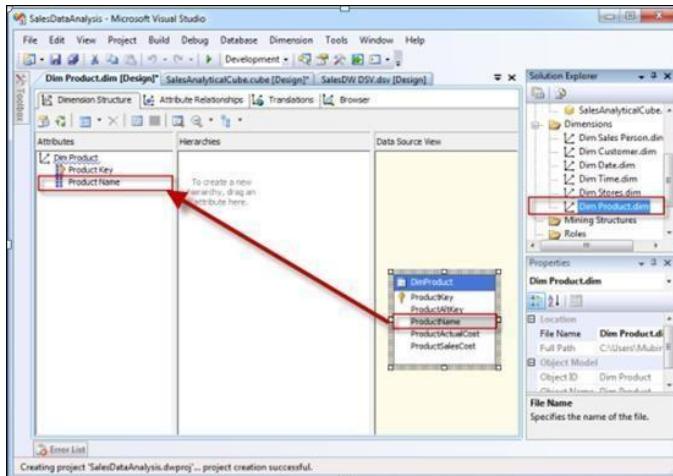
5.6 Select All **Dimensions** here which are associated with your Fact Table-> Click **Next**

5.7 Assign **Cube Name** (**SalesAnalyticalCube**) -> Click **Finish**

5.8 Now your Cube is ready, you can see the newly created cube and dimensions added in your solution explorer

#### **Step 6: Dimension Modification**

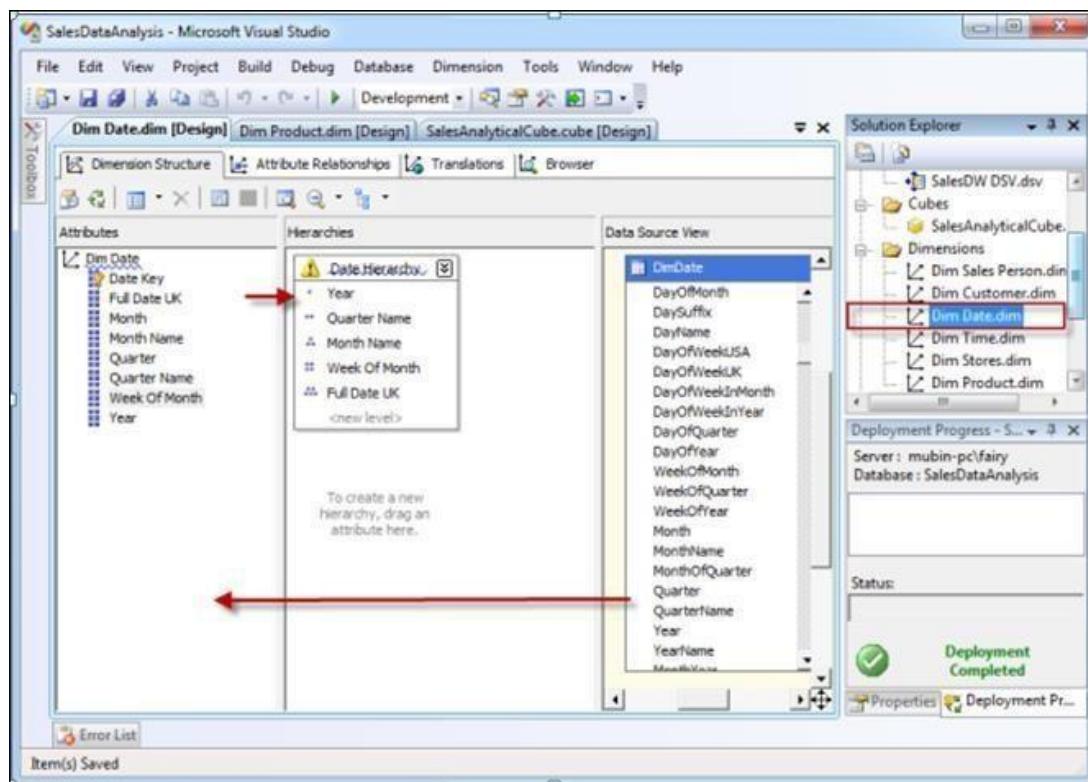
In Solution Explorer, double click on dimension **Dim Product** -> Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side.



### Step 7: Creating Attribute Hierarchy In Date Dimension

Double click On **Dim Date** dimension -> Drag and Drop Fields from Table shown in Data Source View to Attributes-> Drag and Drop attributes from leftmost pane of attributes to middle pane of Hierarchy.

Drag fields in sequence from Attributes to Hierarchy window (Year, Quarter Name, Month Name, Week of the Month, Full Date UK),



### Step 8: Deploy the Cube

8.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -- > Click **Properties**

#### 8.2 Set Deployment Properties First

In Configuration Properties, Select Deployment-> Assign Your SQL Server Instance Name

Where Analysis Services Is Installed (*mubin-pc\fairy*) (*Machine Name\Instance Name*) ->  
Choose Deployment Mode **Deploy All** as of now ->Select Processing Option **Do Not Process**  
-> Click **OK**

8.3 In Solution Explorer, right click on **Project Name** (SalesDataAnalysis) -- > Click **Deploy**

8.4 Once Deployment will finish, you can see the message **Deployment Completed** in deployment Properties.

#### *Step 9: Process the Cube*

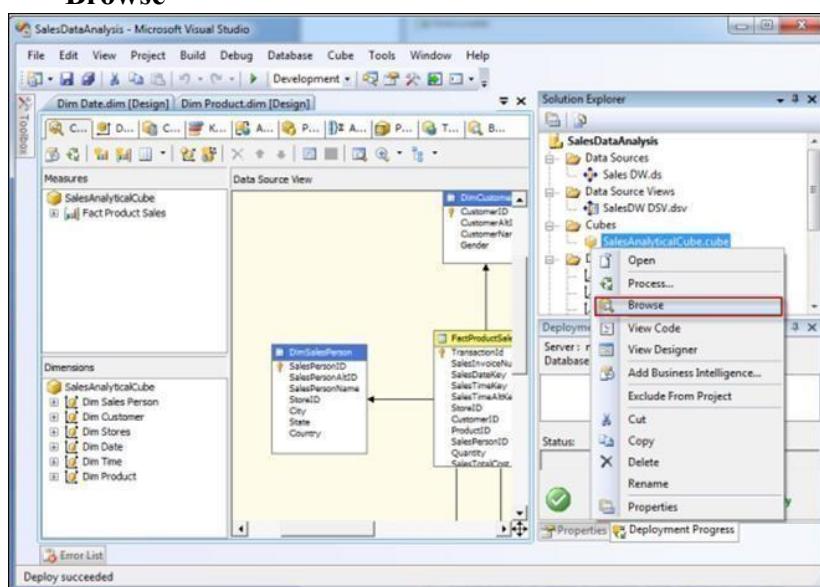
9.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -- > Click **Process**

9.2 Click on **Run** button to process the Cube

9.3 Once processing is complete, you can see **Status** as **Process Succeeded**-->Click **Close** to close both the open windows for processing one after the other.

#### *Step 10: Browse the Cube for Analysis*

10.1 In Solution Explorer, right click on Cube Name (SalesDataAnalysisCube) -- > Click **Browse**



10.2 Drag and drop measures in to Detail fields, & Drag and Drop Dimension Attributes in Row Field or Column fields.

## Now to Browse Our Cube

1. Product Name Drag & Drop into Column
2. Full Date UK Drag & Drop into Row Field
3. FactProductSalesCount Drop this measure in Detail area

The screenshot shows the Microsoft Visual Studio interface for cube design. The main window displays a cube named 'SalesAnalyticalCube'. On the left, there's a tree view of dimensions and measures. The 'Measures' node is expanded, showing 'FactProductSalesCount' as the selected measure. The 'Dimensions' node is also expanded, showing 'Dim Product', 'Dim Date', and 'Dim Customer'. The 'Dim Date' node has several children: 'Date Key', 'Full Date', 'Full Date UK', 'Month Name', 'Quarter Name', 'Weeks Of Month', 'Dim Hierarchy', 'Dim Product', 'Product Key', and 'Dim Sales Person'. The 'Dim Product' node is highlighted with a blue box and labeled 'Measure in Details Field'. The 'Dim Date' node is highlighted with a yellow box and labeled 'Dimension in Row Field'. The 'Dim Product' node is also highlighted with a yellow box and labeled 'Dimension in Column Field'. The main grid area shows sales data for various products like 'Ariel Washing Powder', 'Bigmama Soap', 'Rice Grains 5kg', 'SunFlower Oil 1Ltr', and 'Wheat Flour 5kg'. The data is grouped by date, showing sales counts and totals for each product per day.

Product Name	Ariel Washing Powder	Bigmama Soap	Rice Grains 5kg	SunFlower Oil 1Ltr	Wheat Flour 5kg	Grand Total
01/01/2013	4	4	6	2	18	
02/01/2013	4	4	2	2	16	
03/01/2013	4	4	4	4	20	
Grand Total	12	12	12	8	50	

## Practical 5

**AIM:** Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data.

A book store and have 100 books in storage. You sell a certain % for the highest price of \$50 and a certain % for the lower price of \$20.

The screenshot shows a Microsoft Excel spreadsheet with the following data:

C8			=B4*(1-C4)		
1	A	B	C	D	E
1	<b>Book Store</b>				
2					
3		total number of books	% sold for the highest price		
4		100	60%		
5					
6			number of books	unit profit	
7		highest price	60	\$50	
8		lower price	40	\$20	
9					
10			total profit	\$3,800	
11					

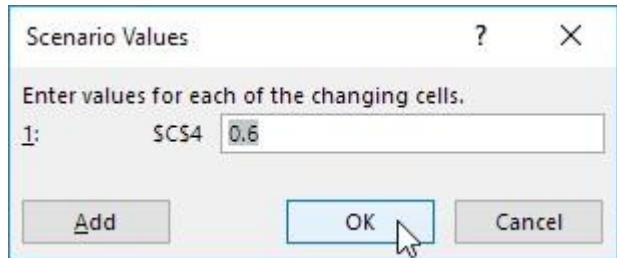
If you sell 60% for the highest price, cell D10 calculates a total profit of  $60 * \$50 + 40 * \$20 = \$3800$ .

### Create Different Scenarios

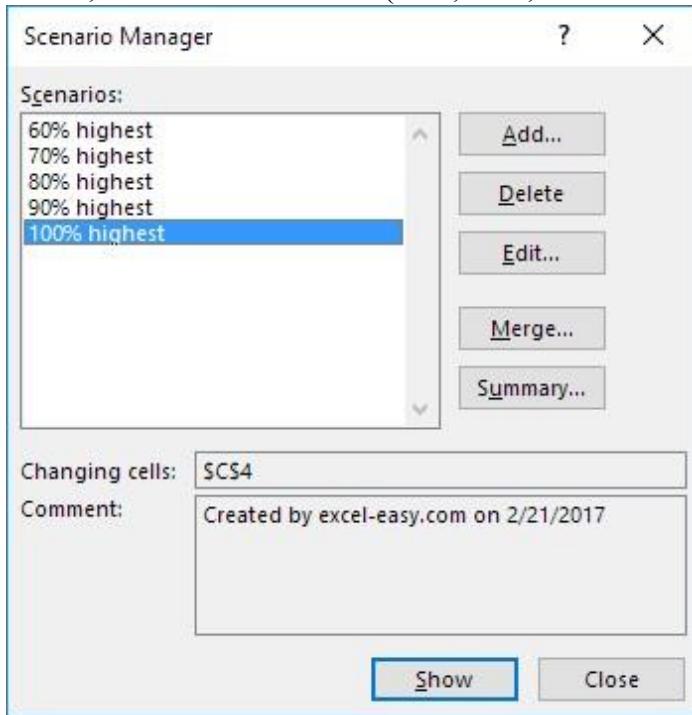
But what if you sell 70% for the highest price? And what if you sell 80% for the highest price? Or 90%, or even 100%? Each different percentage is a different **scenario**. You can use the Scenario Manager to create these scenarios.

Note: You can simply type in a different percentage into cell C4 to see the corresponding result of a scenario in cell D10. However, what-if analysis enables you to easily compare the results of different scenarios. Read on.

1. On the Data tab, in the Forecast group, click What-If Analysis.
2. Click Scenario Manager. The Scenario Manager dialog box appears.
3. Add a scenario by clicking on Add.
4. Type a name (60% highest), select cell C4 (% sold for the highest price) for the Changing cells and click on OK.
5. Enter the corresponding value 0.6 and click on OK again.



6. Next, add 4 other scenarios (70%, 80%, 90% and 100%).



Finally, your Scenario Manager should be consistent with the picture below:

## Practical 6

### AIM: Implementation of Classification algorithm in R Programming.

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R vector. rainfall <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object. rainfall.timeseries <-
ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data. print(rainfall.timeseries)

# Give the chart file a name. png(file =
#rainfall.png")

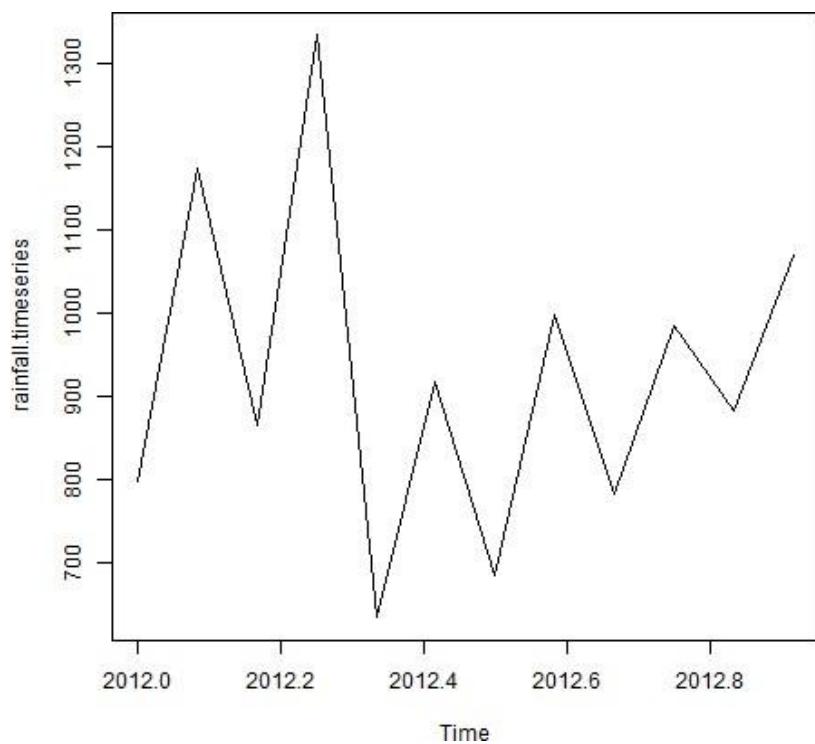
# Plot a graph of the time series. plot(rainfall.timeseries)

# Save the file. dev.off()
```

Output:

When we execute the above code, it produces the following result and chart –

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
2012	799.0	1174.8	865.1	1334.6	635.4	918.5	685.5	998.6	784.2
	Oct	Nov	Dec						
2012	985.0	882.8	1071.0						



# Practical 7

## AIM: Practical Implementation of Decision Tree using R Tool

Steps: `install.packages("party")`

The package "party" has the function `ctree()` which is used to create and analyze decision tree.

### Syntax

The basic syntax for creating a decision tree in R is –

`ctree(formula, data)`

### Input Data

We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age", "shoeSize", "score" and whether the person is a native speaker or not.

Here is the sample data.

```
# Load the party package. It will automatically load other # dependent packages.  
library(party)
```

```
# Print some records from data set readingSkills. print(head(readingSkills))
```

When we execute the above code, it produces the following result and chart –

	nativeSpeaker	age	shoeSize	score
1	yes	5	24.83189	32.29385
2	yes	6	25.95238	36.63105
3	no	11	30.42170	49.60593
4	yes	7	28.66450	40.28456
5	yes	11	31.88207	55.46085
6	yes	10	30.07843	52.83124

Loading required package: methods

Loading required package: grid

We will use the `ctree()` function to create the decision tree and see its graph.

```
# Load the party package. It will automatically load other # dependent  
packages. library(party)
```

```
# Create the input data frame. input.dat <- readingSkills[c(1:105),]
```

```
# Give the chart file a name. png(file = "decision_tree.png")
```

```
# Create the tree. output.tree
```

```

<- ctree(
nativeSpeaker ~ age + shoeSize + score, data
= input.dat)

# Plot the tree.
plot(output.tree)

# Save the file. dev.off()

```

### Output:-

```

null device
1
Loading required package: methods Loading
required package: grid Loading required
package: mvtnorm Loading required package:
modeltools Loading required package: stats4
Loading required package: strucchange Loading
required package: zoo

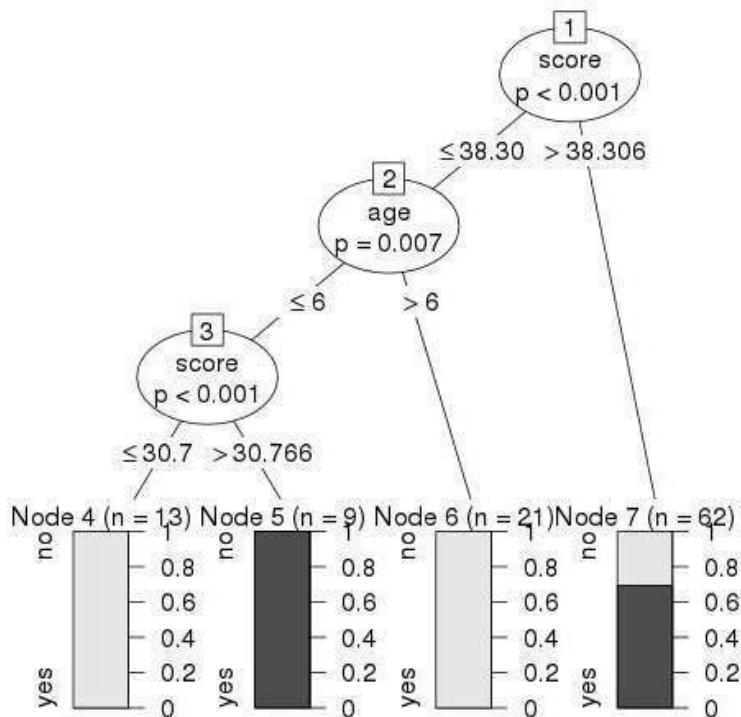
```

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

as.Date, as.Date.numeric

Loading required package: sandwich



# Practical 8

## AIM: Implementation of k-means clustering using R

```
R Console
> # Apply K mean to iris and store result
> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris,3))
K-means clustering with 3 clusters of sizes 62, 38, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1      5.901613    2.748387     4.393548    1.433871
2      6.850000    3.073684     5.742105    2.071053
3      5.006000    3.428000     1.462000    0.246000

Clustering vector:
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[47] 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[93] 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2 2 1 1 2 2 2 1 2 1 2 2 1 1 2 2 2
[139] 1 2 2 2 1 2 2 2 1 2 2 1

Within cluster sum of squares by cluster:
[1] 39.82097 23.87947 15.15100
(between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"       "totss"        "withinss"      "tot.withinss" "betweenss"
[7] "size"          "iter"          "ifault"
```

Compare the Species label with the clustering result

```
R Console
> #Compare the Species label with the clustering result
> table (iris$Species,kc$cluster)

      1   2   3
setosa  0   0 50
versicolor 48  2  0
virginica 14 36  0
> |
```

Plot the clusters and their centre

```
R Console
> # Plot the clusters and their centers
> plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)
> points(kc$centers[,c("Sepal.Length", "Sepal.Width")],col=1:3,pch=8,cex=2)
> |
```

# Practical 9

## AIM: Prediction Using Linear Regression

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

$y = ax + b$  is an equation for linear regression.

Where,  $y$  is the response variable,  $x$  is the predictor variable and  $a$  and  $b$  are constants which are called the coefficients.

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is –

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the **predict()** function in R.

### Input Data

Below is the sample data representing the observations –

```
# Values of height  
151, 174, 138, 186, 128, 136, 179, 163, 152, 131
```

```
# Values of weight.  
63, 81, 56, 91, 47, 57, 76, 72, 62, 48 lm()  
Function
```

This function creates the relationship model between the predictor and the response variable.

### Syntax

The basic syntax for **lm()** function in linear regression is –

```
lm(formula,data)
```

Following is the description of the parameters used –

- **formula** is a symbol presenting the relation between  $x$  and  $y$ .
- **data** is the vector on which the formula will be applied.

## Create Relationship Model & get the Coefficients

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131) y  
<- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
```

```
# Apply the lm() function. relation <- lm(y~x)
```

```
print(relation)
```

When we execute the above code, it produces the following result –

Call:

```
lm(formula = y ~ x)
```

Coefficients:

	x
(Intercept)	-38.4551
x	0.6746

## Get the Summary of the Relationship

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131) y  
<- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
```

```
# Apply the lm() function. relation <- lm(y~x)
```

```
print(summary(relation))
```

When we execute the above code, it produces the following result –

call:

```
lm(formula = y ~ x)
```

Residuals:

Min	1Q	Median	3Q	Max
-6.3002	-1.6629	0.0412	1.8944	3.9775

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	(Intercept)
38.45509	8.04901	8.04901	-4.778	0.00139	**
x	0.67461	0.05191	12.997	1.16e-06	***

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.253 on 8 degrees of freedom

Multiple R-squared: 0.9548,

Adjusted R-squared: 0.9491

F-statistic: 168.9 on 1 and 8 DF, p-value: 1.164e-06

predict()  
Function

## Syntax

The basic syntax for predict() in linear regression is –

```
predict(object, newdata)
```

Following is the description of the parameters used –

- **object** is the formula which is already created using the lm() function.
- **newdata** is the vector containing the new value for predictor variable.

Predict the weight of new persons

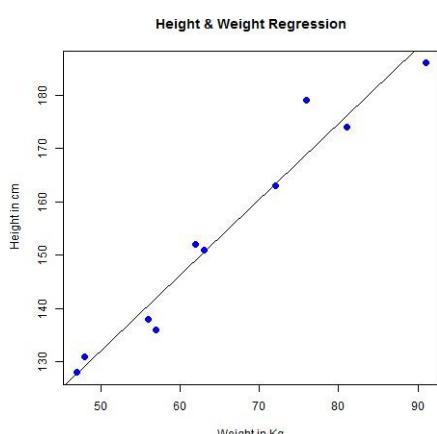
```
# The predictor vector.  
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)  
  
# The response vector. y <- c(63, 81, 56,  
91, 47, 57, 76, 72, 62, 48)  
  
# Apply the lm() function. relation <- lm(y~x)  
  
# Find weight of a person with height 170. a <-  
data.frame(x = 170) result <-  
predict(relation,a) print(result)
```

Result:

```
1  
76.22869
```

### Visualize the Regression Graphically

```
# Create the predictor and response variable. x <-  
c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131) y  
<- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48) relation <-  
lm(y~x)  
  
# Give the chart file a name. png(file = "linearregression.png")  
  
# Plot the chart.  
plot(y,x,col = "blue",main = "Height & Weight Regression", abline(lm(x~y)),cex = 1.3,pch =  
16,xlab = "Weight in Kg",ylab = "Height in cm")  
  
# Save the file. dev.off() output:
```



# Practical 10

## AIM: Data Analysis using Time Series Analysis

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called **time-series object**. It is also a R data object like a vector or data frame.

The time series object is created by using the **ts()** function.

### Syntax

The basic syntax for **ts()** function in time series analysis is –

```
timeseries.object.name <- ts(data, start, end, frequency)
```

Following is the description of the parameters used –

- **data** is a vector or matrix containing the values used in the time series.
- **start** specifies the start time for the first observation in time series.
- **end** specifies the end time for the last observation in time series.
- **frequency** specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional.

### Example

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R vector. rainfall <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object. rainfall.timeseries <-
ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data. print(rainfall.timeseries)

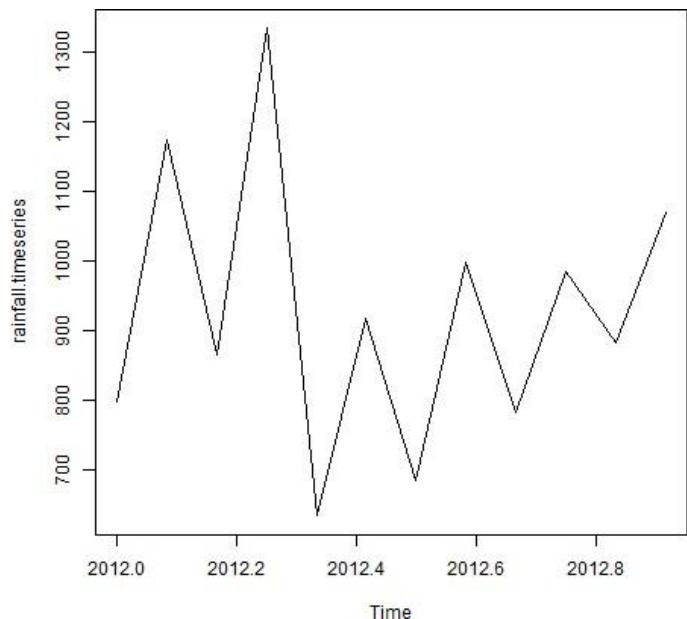
# Give the chart file a name. png(file
= "rainfall.png")

# Plot a graph of the time series. plot(rainfall.timeseries)

# Save the file. dev.off()
```

### Output:-

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
2012	799.0	1174.8	865.1	1334.6	635.4	918.5	685.5	998.6	784.2
	Oct	Nov	Dec						
2012	985.0	882.8	1071.0						



# Practical 11

## AIM: Data Modelling and Analytics with Pivot Table in Excel

**Data Model** is used for building a model where data from various sources can be combined by creating relationships among the data sources. A Data Model integrates the tables, enabling extensive analysis using PivotTables, Power Pivot, and Power View.

A **Data Model** is created automatically when you import two or more tables simultaneously from a database. The existing database relationships between those tables is used to create the Data Model in Excel.

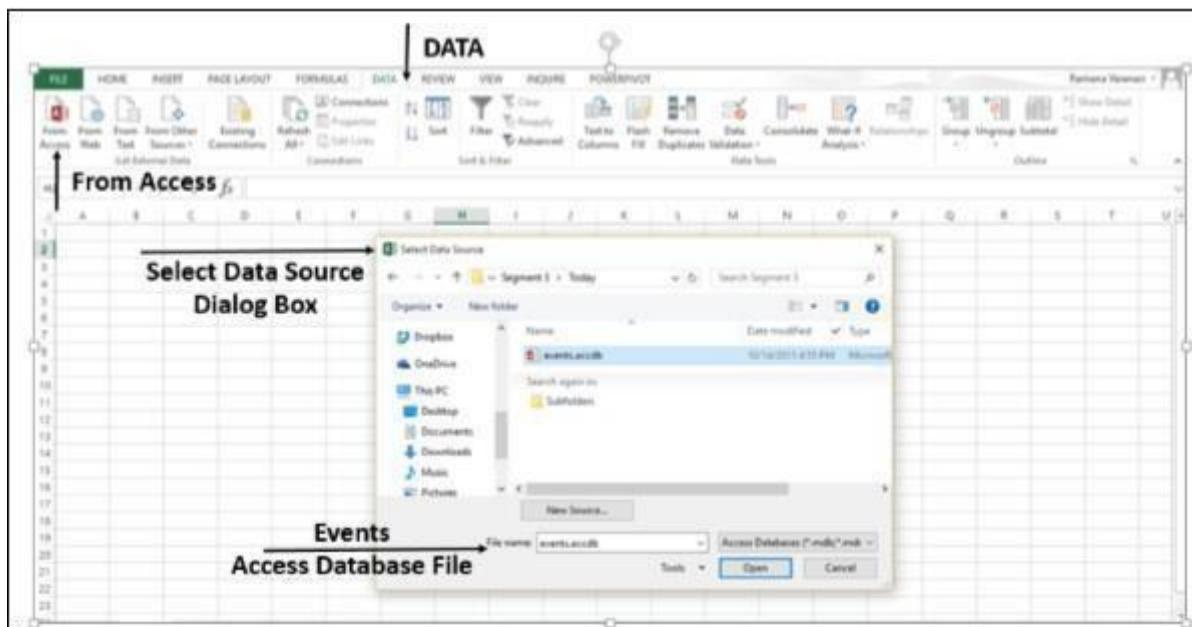
**Step 1** – Open a new blank Workbook in Excel.

**Step 2** – Click on the **DATA** tab.

**Step 3** – In the **Get External Data** group, click on the option **From Access**. The **Select Data Source** dialog box opens.

**Step 4** – Select **Events.accdb**, Events Access Database file.

(url:/ <https://fcschools.instructure.com/courses/373/files/10607>)



**Step 5** – The **Select Table** window, displaying all the **tables** found in the database, appears.

**Step 6** – Tables in a database are similar to the tables in Excel.

Check the ‘Enable selection of multiple tables’ box, and select all the tables. Then click **OK**.

**Step 7** – The **Import Data** window appears. Select the **PivotTable Report** option. This option imports the tables into Excel and prepares a PivotTable for analyzing the imported

tables. Notice that the checkbox at the bottom of the window - ‘Add this data to the Data Model’ is selected and disabled.

**Step 8** – The data is imported, and a **PivotTable** is created using the imported tables.

The screenshot shows a Microsoft Excel spreadsheet with a PivotTable Fields ribbon tab selected. In the center of the screen, there is a placeholder text 'To build a report, choose fields from the PivotTable Field List'. Below this text are two small icons representing tables. To the right of the placeholder is a 'Pivot Table Fields' pane. The pane has a title bar 'Pivot Table Fields' and a subtitle 'ACTIVE: ALL'. It contains a section titled 'Choose fields to add to report:' with two collapsed sections: 'Disciplines' and 'Medals'. Under 'Disciplines', there are three items: Discipline, Disciplined, and Speed. Under 'Medals', there are three items: Event, EventID, and Event. Below these sections is a 'Drag fields between areas below' section with four boxes: 'ROWS', 'COLUMNS', 'VALUES', and 'FILTERS'. The 'ROWS' box contains the 'Event' field from the 'Medals' table. The 'VALUES' box contains the 'EventID' field from the 'Medals' table. The 'FILTERS' box contains the 'Event' field from the 'Medals' table. A black arrow points to the 'Pivot Table' area in the center of the screen.

## Explore Data Using PivotTable

**Step 1** – You know how to add fields to PivotTable and drag fields across areas. Even if you are not sure of the final report that you want, you can play with the data and choose the best-suited report.

In **PivotTable Fields**, click on the arrow beside the table - **Medals** to expand it to show the fields in that table. Drag the **NOC\_CountryRegion** field in the **Medals** table to the **COLUMNS** area.

**Step 2** – Drag **Discipline** from the **Disciplines** table to the **ROWS** area.

**Step 3** – Filter **Discipline** to display only five sports: Archery, Diving, Fencing, Figure Skating, and Speed Skating. This can be done either in **PivotTable Fields** area, or from the **Row Labels** filter in the PivotTable itself.

**Step 4** – In **PivotTable Fields**, from the **Medals** table, drag **Medal** to the **VALUES** area.

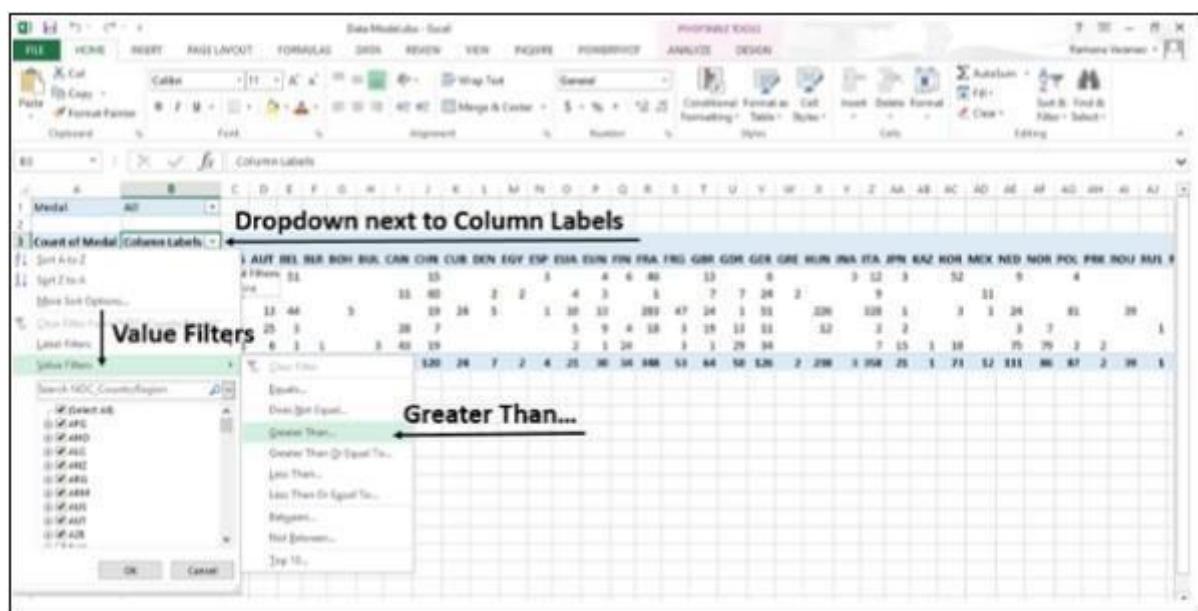
**Step 5** – From the **Medals** table, select **Medal** again and drag it into the **FILTERS** area.

A screenshot of Microsoft Excel showing a PivotTable titled "Count of Medal". The table includes columns for "Medal" (All), "Sport", and "Country". The data shows medal counts for Archery, Diving, Fencing, Figure Skating, and Speed Skating across countries like AUS, AUT, BEL, BGR, CAN, CHN, CUB, DEN, EGY, ESP, FRA, FIN, GBR, GER, GRE, HUN, INA, ITA, JPN, KAZ, KOR, MEX, NOR, POL, PRK, ROU, RUS, SWE, TUR, UZB, and VEN. A callout bubble labeled "PivotTable" points to the table.

**Step 6** – Click the dropdown list button to the right of the **Column labels**.

**Step 7** – Select **Value Filters** and then select **Greater Than...**

**Step 8** – Click **OK**.



The Value Filters dialog box for the count of Medals is greater than appears.

**Step 9** – Type **80** in the **Right Field**.

**Step 10** – Click **OK**.

The PivotTable displays only those regions, which has more than total 80 medals.

PivotTable displaying Regions with total no. of Medals Greater Than 80

Sport	Count of Medal - Column Labels													Grand Total
	CAN	CHN	FRA	GBR	HUN	IRL	NED	NOR	POL	RUS	SWE	URS	USA	
Archery	31	15	46	6	12	9	4	1	7	32	201		201	
Driving	11	60	1	24	9			24	14	131	239		239	
Fencing	44	19	283	31	226	228	24	81	41	140	48	1290	1290	
Figure skating	3	28	7	18	12	12	2	3	7	29	42	51	213	
Speed skating	1	43	19	14	7	75	76	2	8	60	73	401	401	
Grand Total	98	82	120	348	126	238	158	111	86	87	389	268	355	

## Create Relationship between Tables

Relationships let you analyze your collections of the data in Excel, and create interesting and aesthetic reports from the data you import.

**Step 1 – Insert** a new Worksheet.

**Step 2 – Create** a new table with new data. Name the new table as **Sports**.

**Step 3 – Now you can create relationship between this new table and the other tables that already exist in the Data Model in Excel. Rename the Sheet1 as **Medals** and Sheet2 as **Sports**.**

On the **Medals sheet**, in the **PivotTable Fields List**, click **All**. A complete list of available tables will be displayed. The newly added table - **Sports** will also be displayed.

PivotTable ALL

ACTIVE ALL

Choose fields to add to report

- Disciplines
- Medals
- Events
- Table - Sports

Table - Sports

Drag field between panes below:

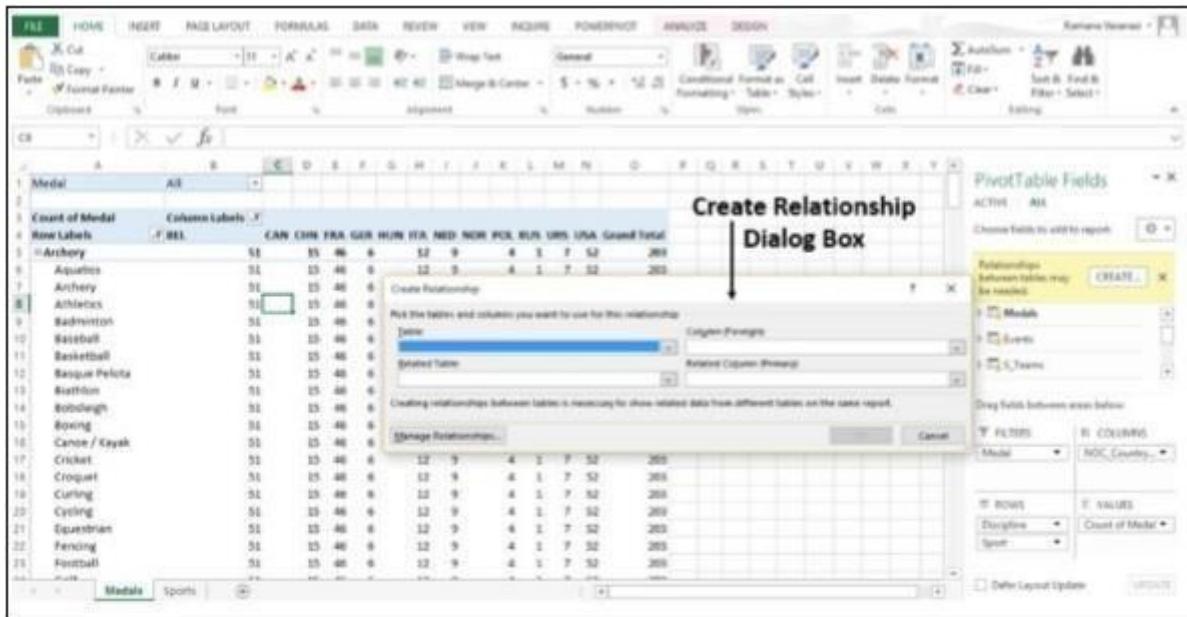
FILTERS      COLUMNS

ROWS      VALUES

Discipline      Count of Medal

**Step 4** – Click on **Sports**. In the expanded list of fields, select **Sports**. Excel messages you to create a relationship between tables

**Step 5** – Click on **CREATE**. The **Create Relationship** dialog box opens.



**Step 6** – To create the relationship, one of the tables must have a column of unique, non-repeated, values. In the **Disciplines** table,

**SportID** column has such values. The table **Sports** that we have created also has the **SportID** column. In **Table**, select **Disciplines**.

**Step 7** – In **Column (Foreign)**, select **SportID**.

**Step 8** – In **Related Table**, select **Sports**.

**Step 9** – In **Related Column (Primary)**, **SportID** gets selected automatically. Click **OK**.

**Step 10** – The **PivotTable** is modified to reflect the addition of the new **Data Field** **Sport**. Adjust the order of the fields in the Rows area to maintain the **Hierarchy**. In this case, **Sport** should be first and **Discipline** should be the next, as **Discipline** will be nested in **Sport** as a sub-category.

**PivotTable with the New Relationship**

**Rows ordered for proper Hierarchy**

The screenshot shows a Microsoft Excel spreadsheet with a PivotTable in the center. The PivotTable displays medal counts for various sports across different countries. The rows are ordered to reflect proper hierarchy, starting with 'Medals' and 'Sports', followed by specific disciplines like 'Aquatics', 'Diving', 'Archery', etc. The PivotTable Fields pane on the right shows the active fields: 'Medals' (Row Labels), 'Sports' (Column Labels), and 'Discipline' (Values). Relationships are established between 'Sports' and 'Discipline', and between 'Discipline' and 'NOC Country'. The PivotTable Fields pane also lists other available fields: 'Events' and 'L\_Team'. A note in the center of the screen says 'PivotTable with the New Relationship' and 'Rows ordered for proper Hierarchy'.

	CAN	CHN	FRA	GER	HUN	ITA	NED	NOR	POK	RUS	SWE	URS	USA	Grand Total
Row Labels	11	69	1	24	9	24	24	14	131	274				
Aquatics	11	15	46	8	12	9	4	1	3	52	261			
Diving	11	60	1	24	9	24	14	131	274					
Archery	51	15	46	8	12	9	4	1	3	52	261			
Archery	51	15	46	8	12	9	4	1	3	52	261			
Fencing	44	19	281	51	226	228	24	83	45	345	180	2790		
Fencing	44	19	281	51	226	228	24	83	45	345	180	2790		
Skating	4	75	26	18	45	52	9	28	80	2	37	352	524	
Figure skating	2	28	7	18	33	32	2	3	7	29	42	53	213	
Speed skating	1	43	19	34	7	25	79	2	8	60	75	402		
Grand total	90	82	120	149	120	238	258	213	80	87	181	268	2880	

# Practical 12

## AIM: Data Analysis and Visualization using Advanced Excel

Power View is a feature of Microsoft Excel 2013 that enables **interactive** data exploration, visualization, and presentation encouraging intuitive ad-hoc reporting.

### Create a Power View Sheet

Make sure **Power View** add-in is enabled in Excel 2013.

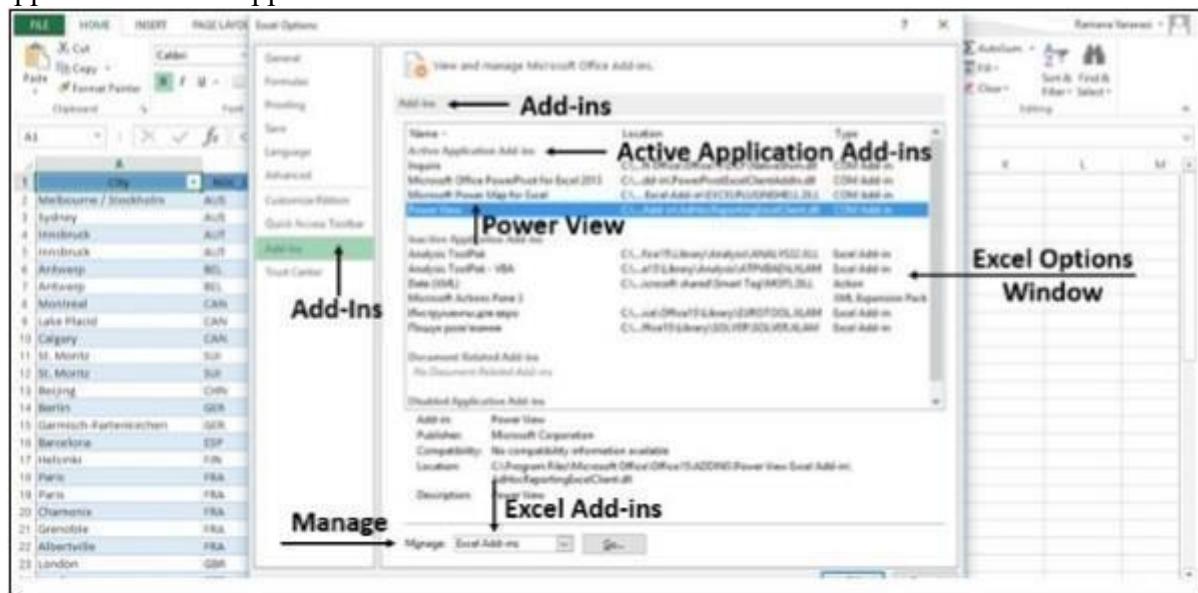
**Step 1** – Click on the **File** menu and then Click on **Options**.

The **Excel Options** window appears.

**Step 2** – Click on **Add-Ins**.

**Step 3** – In the **Manage** box, click the drop-down arrow and select **Excel Add-ins**.

**Step 4** – All the available **Add-ins** will be displayed. If **Power View** Add-in is enabled, it appears in Active Application Add-ins.



If it does not appear, follow these steps –

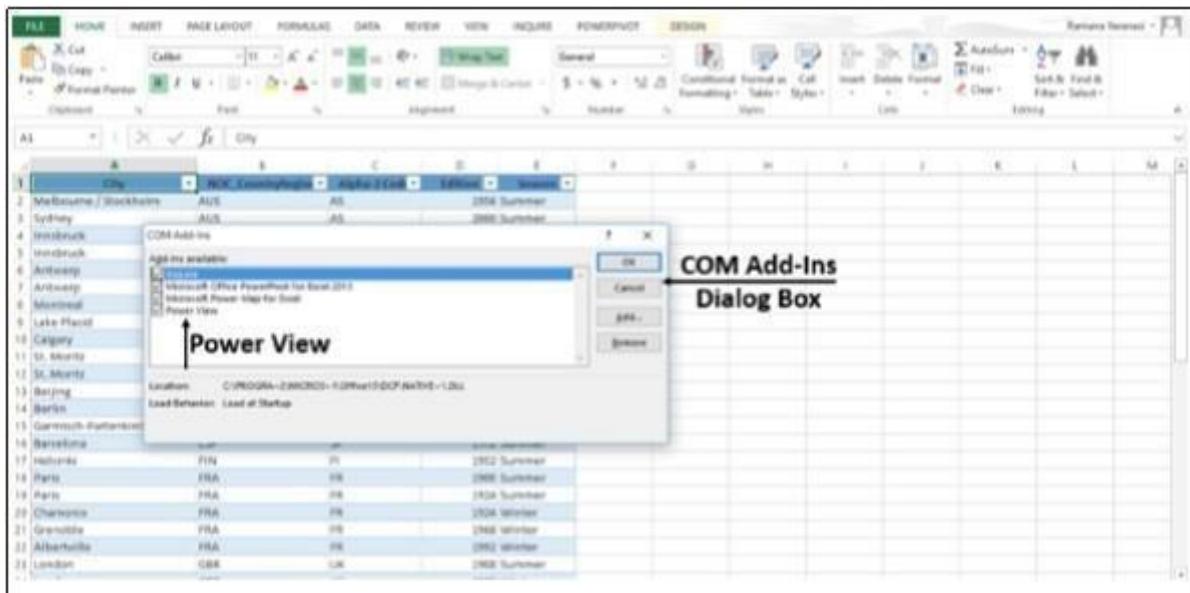
**Step 1** – In the **Excel Options** Window, Click on **Add-Ins**.

**Step 2** – In the **Manage** box, click the drop-down arrow and select **COM Add-ins Step 3**

– Click on the **Go** button. A **COM Add-Ins** Dialog Box appears.

**Step 4** – Check the **Power View** Check Box.

**Step 5** – Click **OK**.



Now, you are ready to create the **Power View sheet**.

**Step 1 – Click on the Data Table.**

**Step 2 – Click on Insert tab.**

**Step 3 – Click on Power View in Reports Group.**

An **Opening Power View** window opens, showing the progress of Working on opening Power View sheet.

The **Power View sheet** is created for you and added to your Workbook with the **Power View**. On the Right-side of the **Power View**, you find the **Power View Fields**. Under the **Power View Fields** you will find **Areas**.

In the Ribbon, if you click on **Design tab**, you will find various **Visualization** options.

## Create Charts and other Visualizations

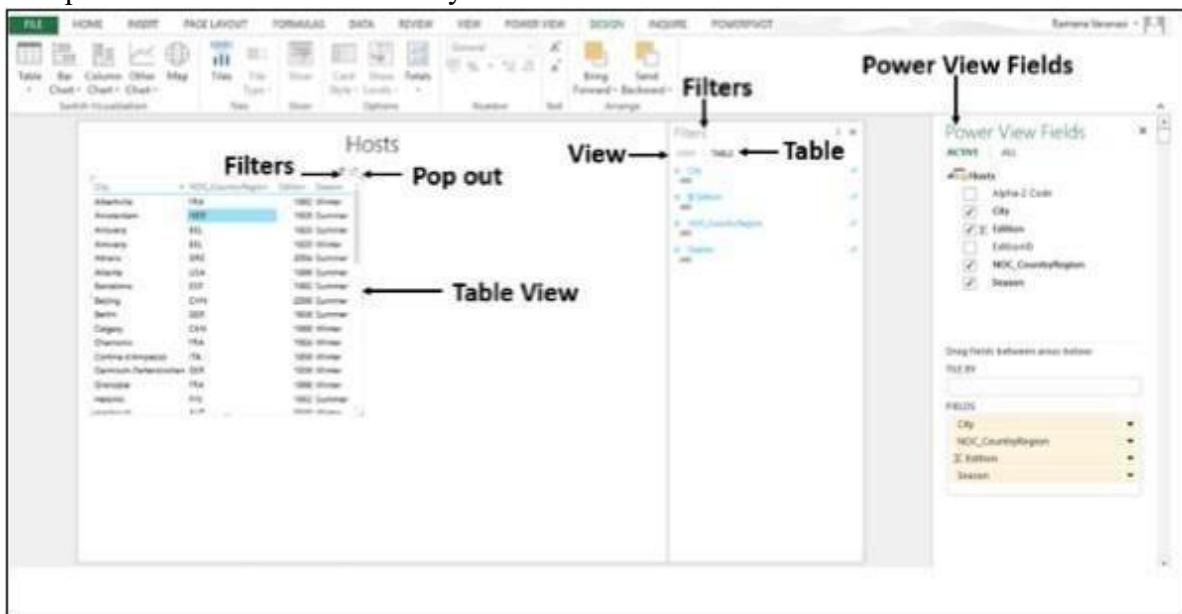
For every visualization you want to create, you start on a Power View sheet by creating a table, which you then easily convert to other visualizations, to find one that best illustrates your Data.

**Step 1 – Under the Power View Fields, select the fields you want to visualize.**

**Step 2** – By default, the Table View will be displayed. As you move across the Table, on the top-right corner, you find two symbols – Filters and Pop out.

**Step 3** – Click on the **Filters** symbol. The filters will be displayed on the right side.

**Filters** has two tabs. View tab to filter all visualizations in this **View** and **Table** tab to filter the specific values in this table only.



## Visualization – Matrix

A **Matrix** is made up of rows and columns like a **Table**. However, a Matrix has the following capabilities that a Table does not have –

- Display data without repeating values.
  - Display totals and subtotals by row and column.
  - With a hierarchy, you can drill up/drill down.

## **Collapse and Expand the Display**

### **Step 1 – Click on the DESIGN tab.**

**Step 2 – Click on Table in the Switch Visualization Group.**

### **Step 3 – Click on Matrix.**

The **Matrix Visualization** appears.

The screenshot shows a Microsoft Power View interface. On the left, there is a matrix visualization titled "Hosts" with data for various cities across different seasons. The columns represent "City" and "Edition", and the rows represent "Season". The data includes cities like Atlanta, Boston, Chicago, Detroit, and New York, with values for Summer, Fall, Winter, and Spring. On the right, there is a "Power View Fields" pane with sections for "ACTIVE", "ROWS", and "COLUMNS". The "ROWS" section contains fields for "City", "Edition", "Season", and "NDC\_CountryRegion". The "COLUMNS" section contains fields for "City", "NDC\_CountryRegion", "Edition", and "Season". A legend at the bottom indicates that blue cards represent "Edition" and orange cards represent "Season".

### Visualization – Card

You can convert a **Table** to a series of **Cards** that display the data from **each row** in the table laid out in a **Card format**, like an **index Card**.

**Step 1** – Click on the **DESIGN** tab.

**Step 2** – Click on **Table** in the **Switch Visualization Group**.

**Step 3** – Click on **Card**.

The **Card Visualization** appears.

The screenshot shows a Microsoft Power View interface with a card visualization titled "Hosts". The data is presented as a list of cards, where each card represents a row from the original table. Each card contains the city name, edition, and season. The cards are color-coded: blue for Edition and orange for Season. The "Power View Fields" pane on the right is identical to the one in the matrix visualization, showing fields for City, Edition, Season, and NDC\_CountryRegion under the ACTIVE, ROWS, and COLUMNS sections. A legend at the bottom indicates that blue cards represent "Edition" and orange cards represent "Season".

### Visualization – Charts

In **Power View**, you have a number of Chart options: Pie, Column, Bar, Line, Scatter, and Bubble. You can use several design options in a chart such as showing and hiding labels, legends, and titles.

Charts are interactive. If you click on a Value in one Chart –

- the **Value** in that chart is highlighted.
- All the Tables, Matrices, and Tiles in the report are filtered to that Value.
- That **Value** in all the other Charts in the report is highlighted.

The charts are interactive in a presentation setting also.

### **Step 1 – Create a Table Visualization from Medals data.**

You can use Line, Bar and Column Charts for comparing data points in one or more data series. In these Charts, the x-axis displays one field and the y-axis displays another, making it easy to see the relationship between the two values for all the items in the Chart.

Line Charts distribute category data evenly along a horizontal (category) axis, and all numerical value data along a vertical (value) axis.

### **Step 2 – Create a Table Visualization for two Columns, NOC\_CountryRegion and Count of Medal.**

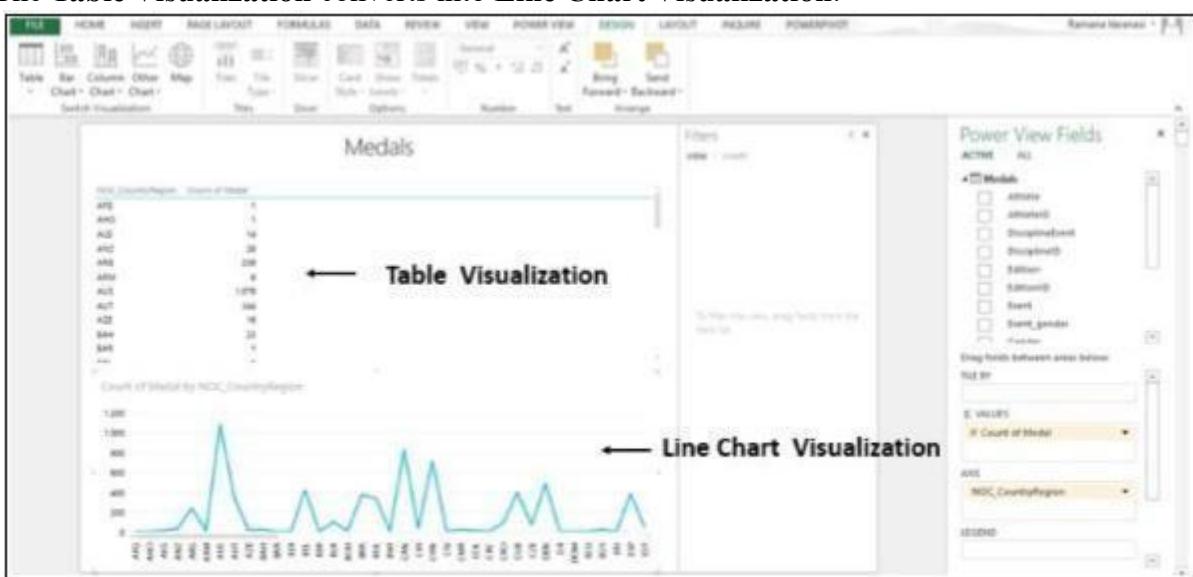
### **Step 3 – Create the same Table Visualization**

### **Step 4 – Click on the Table Visualization.**

### **Step 5 – Click on Other Chart in the Switch Visualization group.**

### **Step 6 – Click on Line.**

The **Table Visualization** converts into **Line Chart Visualization**.

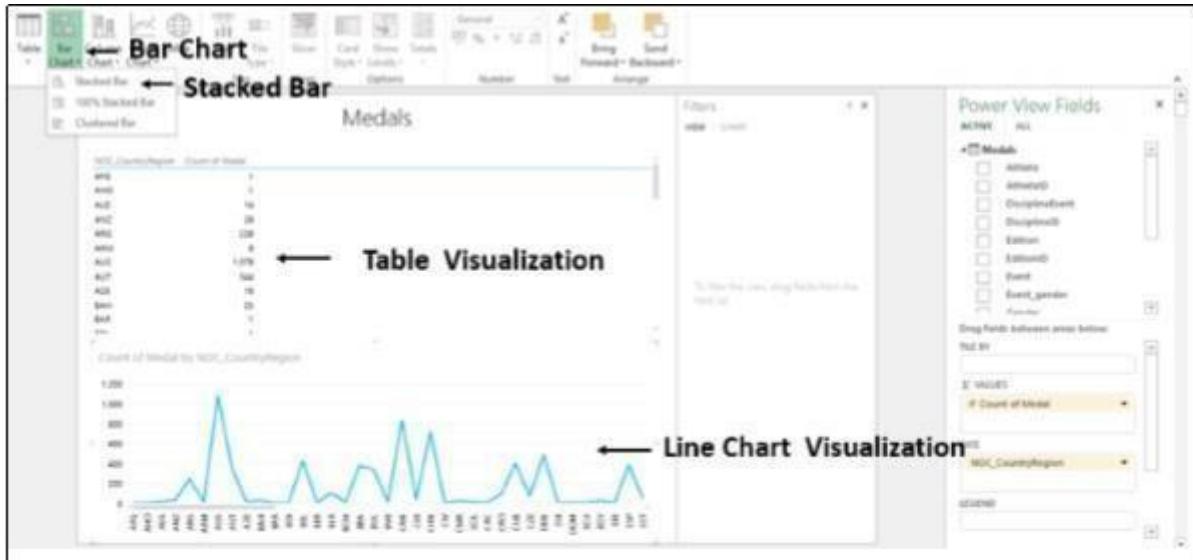


In a **Bar Chart**, categories are organized along the vertical axis and values along the horizontal axis. In **Power View**, there are three subtypes of the **Bar Chart**: **Stacked**, **100% stacked**, and **Clustered**.

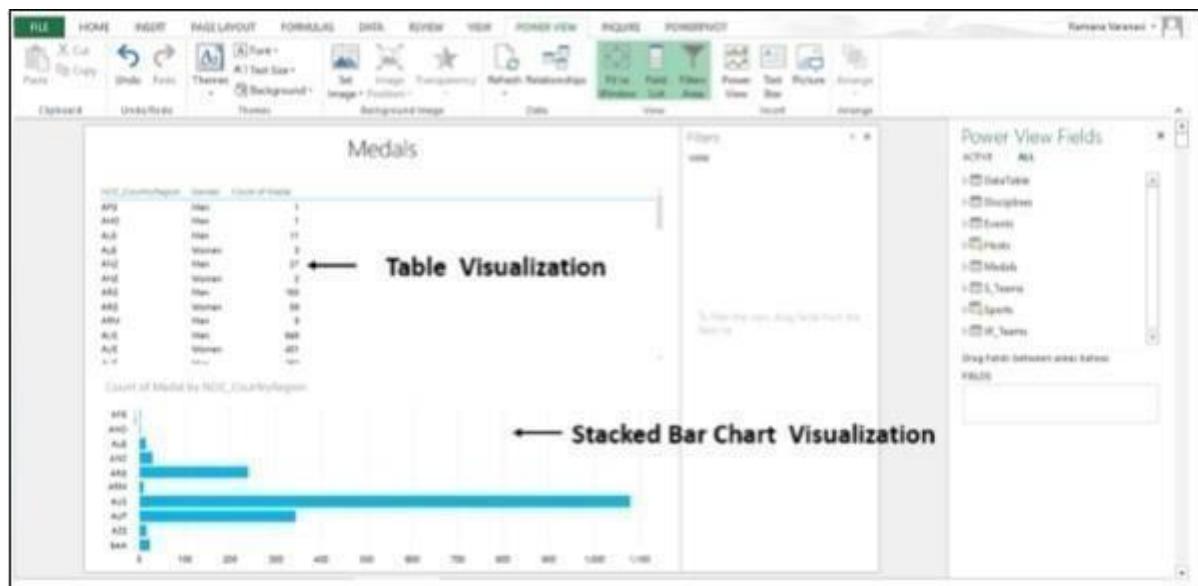
### **Step 7 – Click on the Line Chart Visualization.**

### **Step 8 – Click on Bar Chart in the Switch Visualization Group.**

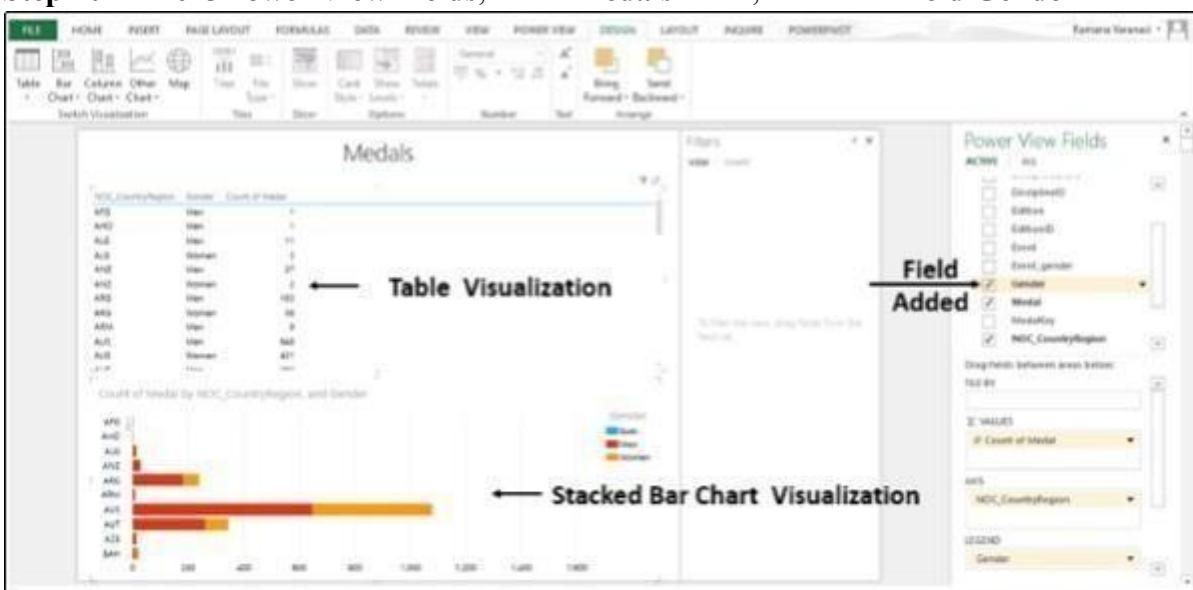
Step 9 – Click on the Stacked Bar option.



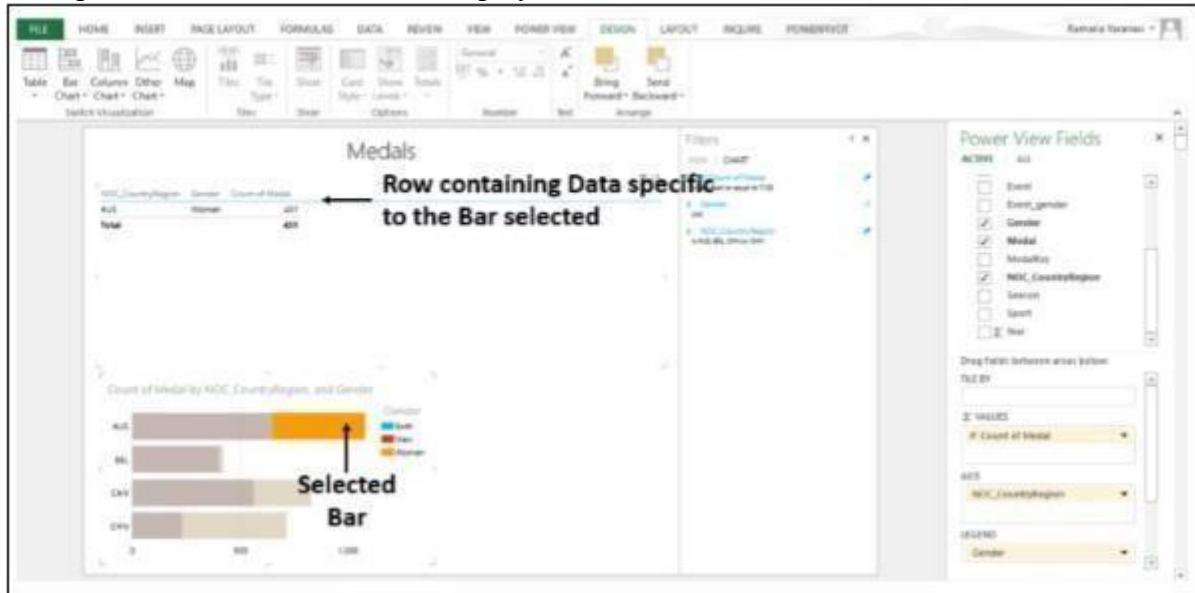
The Line Chart Visualization converts into Stacked Bar Chart Visualization.



Step 10 – In the Power View Fields, in the Medals Table, select the Field Gender also.



**Step 11** – Click on one of the bars. That portion of the bar is highlighted. Only the row containing the Data specific to the selected bar is displayed in the table above.



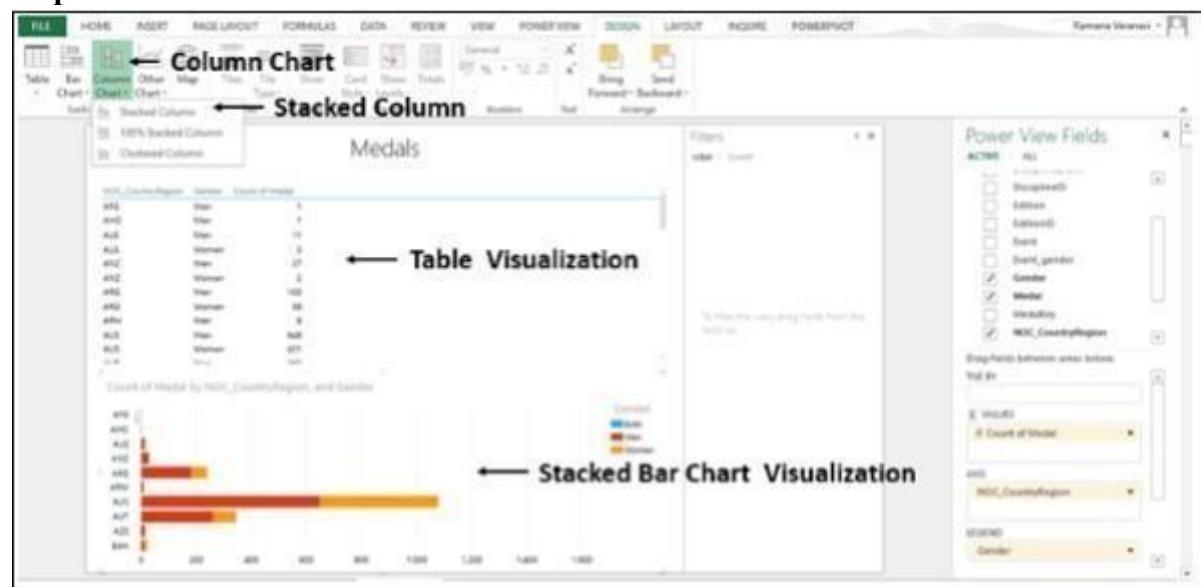
You can use the column charts for showing data changes over a period of time or for illustrating comparison among different items. In a Column Chart, the categories are along the horizontal axis and values are along the vertical axis.

In Power View, there are three **Column Chart** subtypes: **Stacked**, **100% stacked**, and **Clustered**.

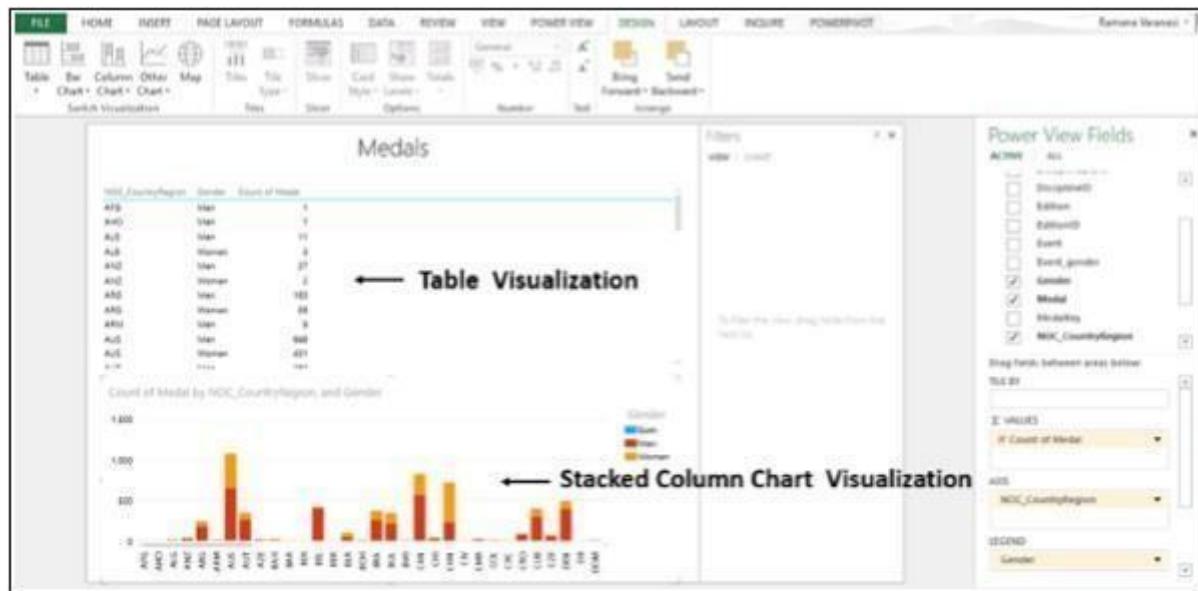
**Step 12** – Click on the **Stacked Bar Chart Visualization**.

**Step 13** – Click on Column Chart in the **Switch Visualization** group.

**Step 14** – Click on **Stacked Column**.



The Stacked Bar Chart Visualization converts into Stacked Column Chart Visualization.



You can have simple **Pie Chart Visualizations** in Power View.

**Step 1** – Click on the **Table Visualization** as shown below.

**Step 2** – Click on **Other Chart** in the **Switch Visualization** group.

**Step 3** – Click on **Pie** as shown in the image given below.

