
VIKAS VIDYA EDUCATION TRUST'S

# Lords Universal College

## Department of Information Technology

## CERTIFICATE

**Class:** TYBSc.IT **Year:** 2024-2025

This is to certify that the work of Business Intelligence practical entered in this Journal is the work of Shree / Kumari _____

Of TYIT Roll no: _____

University Exam No: _____ has satisfactorily completed the required number of practical and worked for the 6th term the term of the year 2024 2025 in the college laboratory as laid down by the university.

| | | |
|---|---|---|
| **Internal Examiner** | **External Examiner** | **Head of the Department** |

**Date: / / 2025** **Department of Bsc.IT**

# INDEX

| Practical No | Details | Date | Signature |
|---|---|---|---|
| 1 | a. Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart. b. Import the cube in Microsoft Excel and create the Pivot table and Pivot Chart to perform data analysis.. | | |
| 2 | Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data. Use Excel. | | |
| 3 | Perform the data classification using classification algorithm using R/Python. | | |
| 4 | Perform the data clustering using clustering algorithm using R/Python. | | |
| 5 | Perform the Linear regression on the given data warehouse data using R/Python. | | |
| 6 | Perform the logistic regression on the given data warehouse data using R/Python. | | |
| 7 | Write a Python program to read data from a CSV file, perform simple data analysis, and generate basic insights. (Use Pandas is a Python library). | | |
| 8 | a. Perform data visualization using Python on any sales data. b. Perform data visualization using PowerBI on any sales data. | | |
| 9 | Create the Data staging area for the selected database using SQL. | | |
| 10 | Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model. | | |

## Practical – 1A: Perform the analysis for the following:
## Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

Input :

Create table into SQL database as follows :

Database Name : pivot_demo

Table Name : Employee

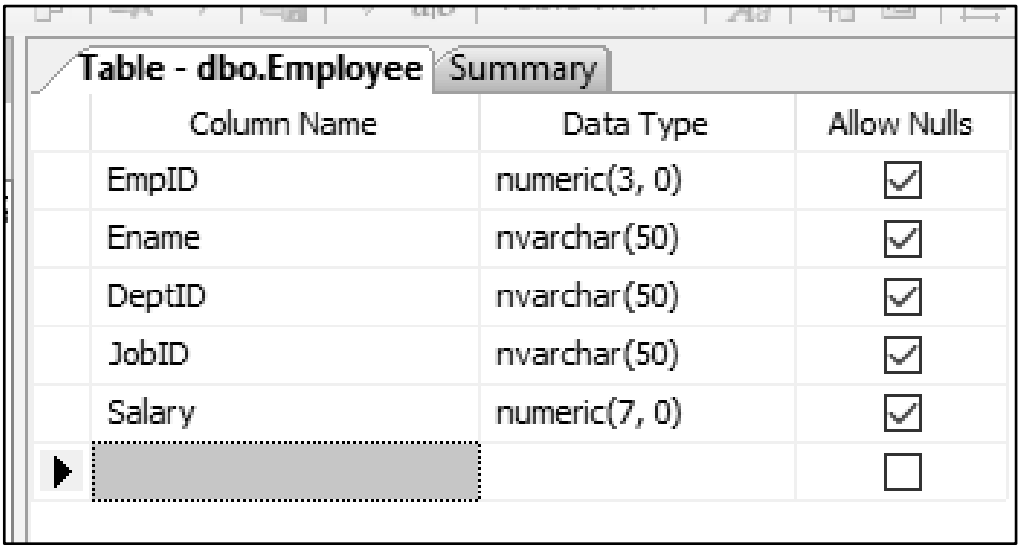


Table - dbo.Employee | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| EmpID | numeric(3, 0) | ☑ |
| Ename | nvarchar(50) | ☑ |
| DeptID | nvarchar(50) | ☑ |
| JobID | nvarchar(50) | ☑ |
| Salary | numeric(7, 0) | ☑ |
| | | ☐ |

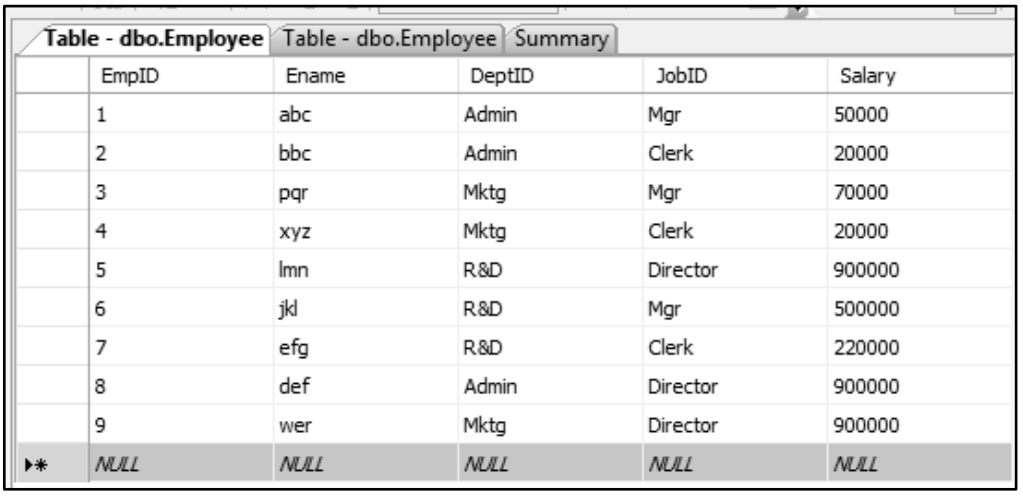
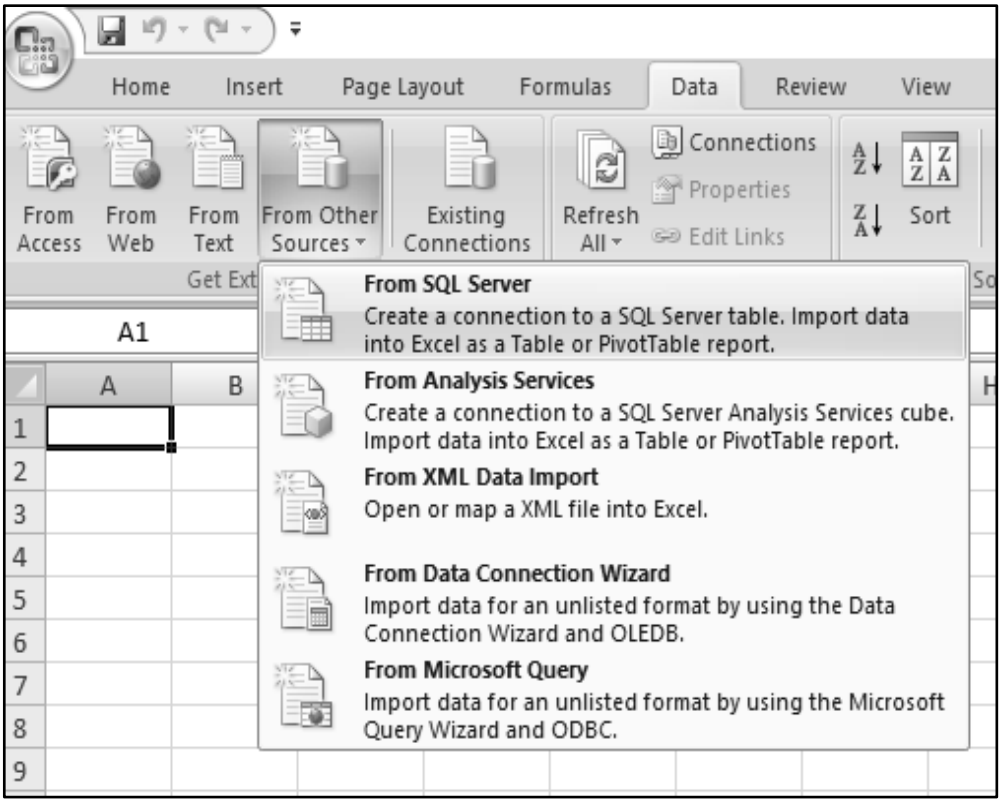
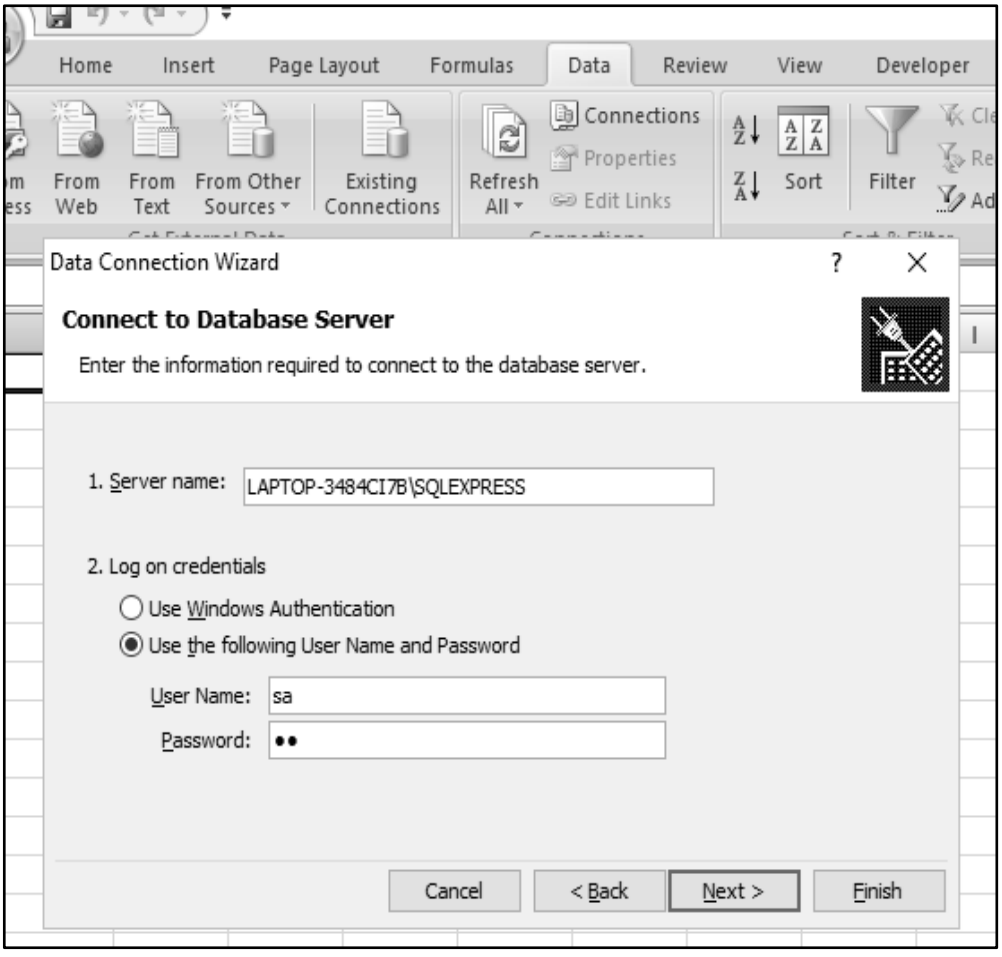


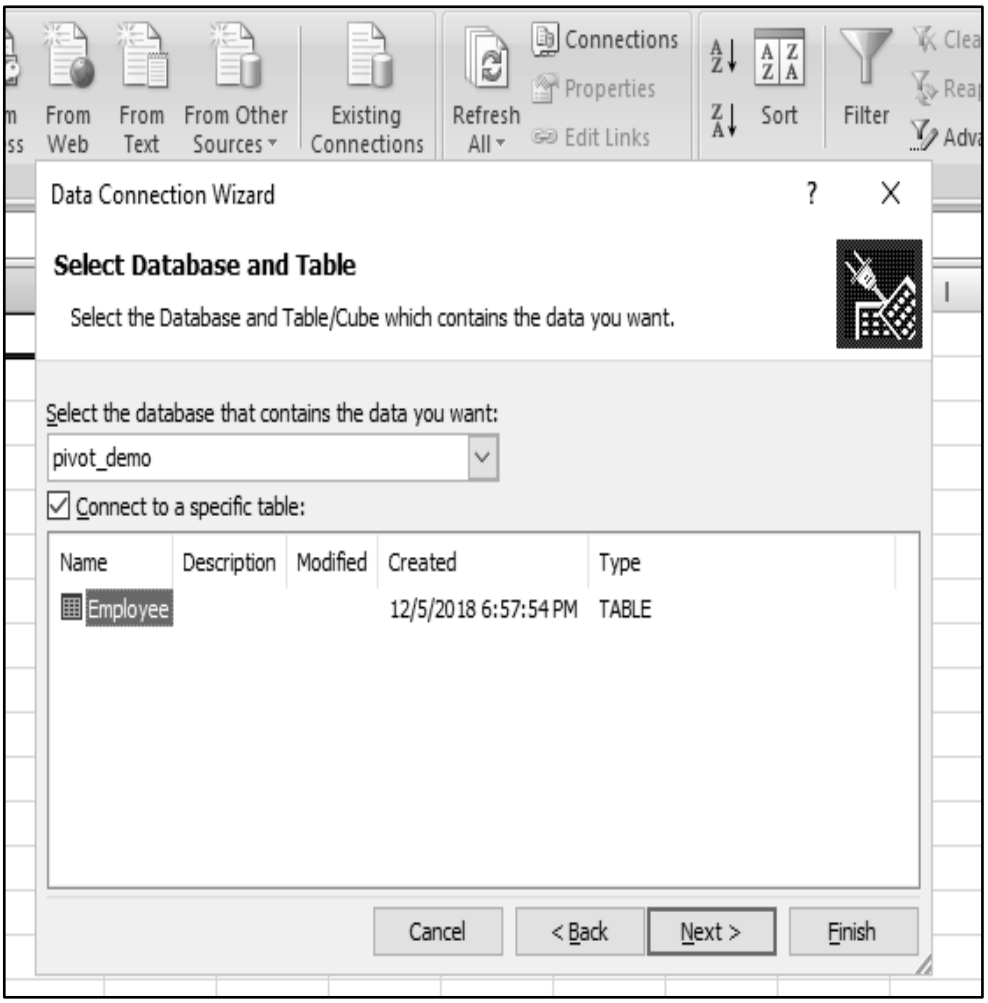
Table - dbo.Employee | Table - dbo.Employee | Summary

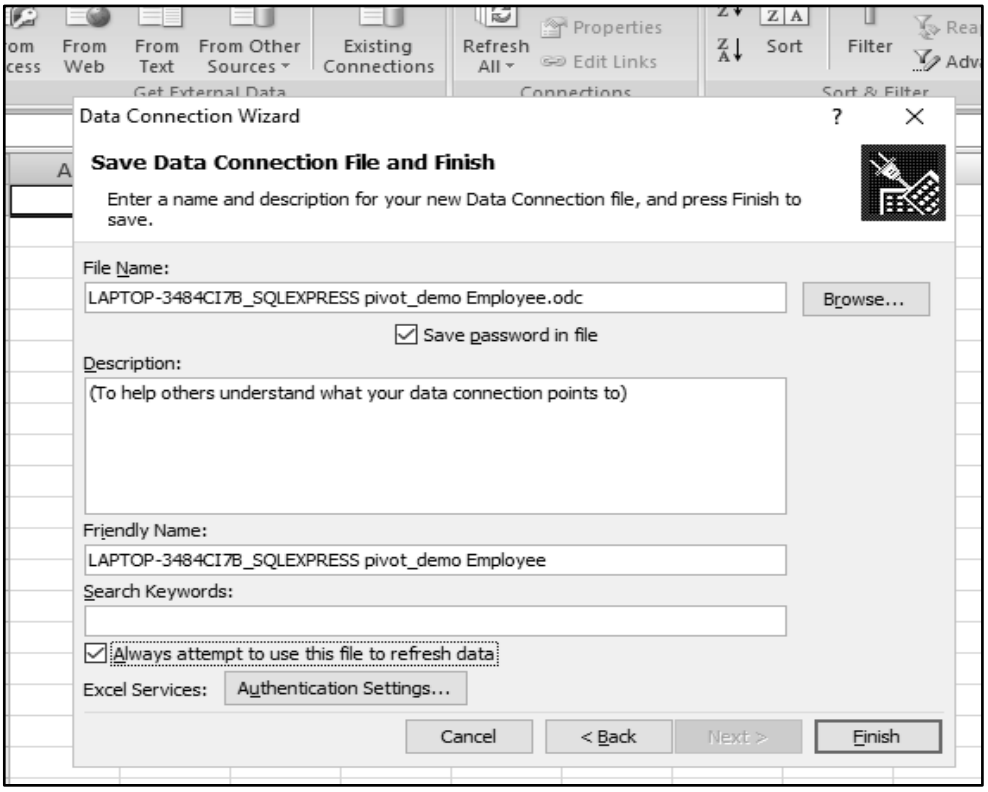
| EmpID | Ename | DeptID | JobID | Salary |
|---|---|---|---|---|
| 1 | abc | Admin | Mgr | 50000 |
| 2 | bbc | Admin | Clerk | 20000 |
| 3 | pqr | Mktg | Mgr | 70000 |
| 4 | xyz | Mktg | Clerk | 20000 |
| 5 | lmn | R&D | Director | 900000 |
| 6 | jkl | R&D | Mgr | 500000 |
| 7 | efg | R&D | Clerk | 220000 |
| 8 | def | Admin | Director | 900000 |
| 9 | wer | Mktg | Director | 900000 |
| *NULL* | *NULL* | *NULL* | *NULL* | *NULL* |

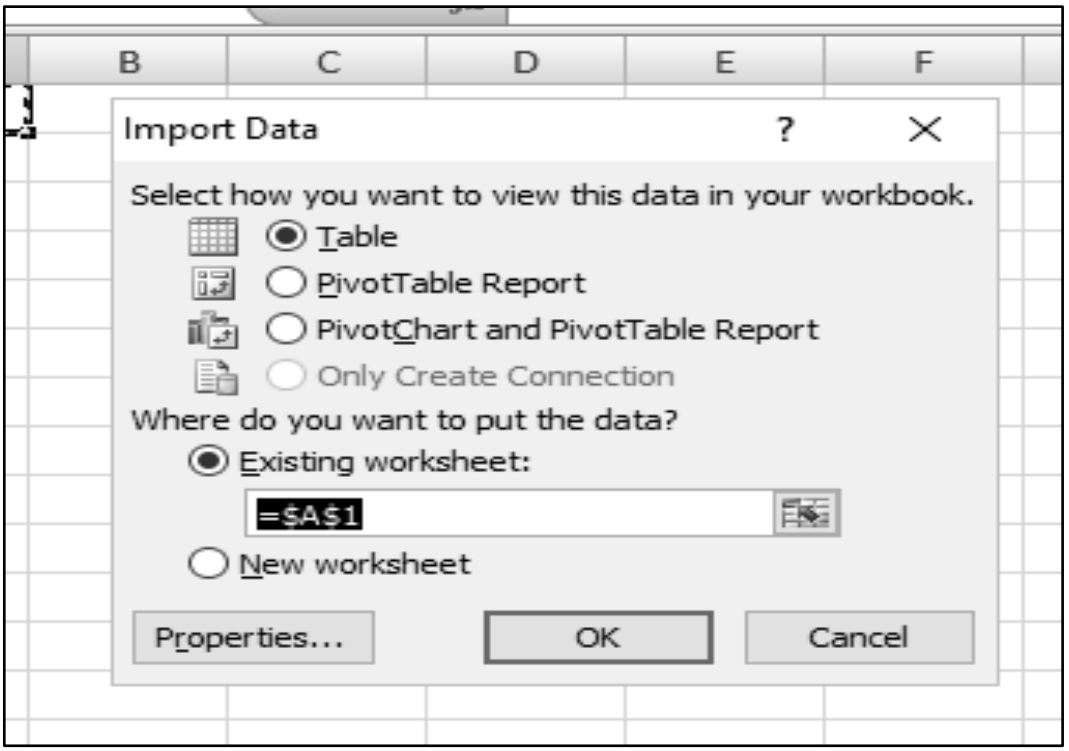
Open Ms. Excel and Import above table as follows :



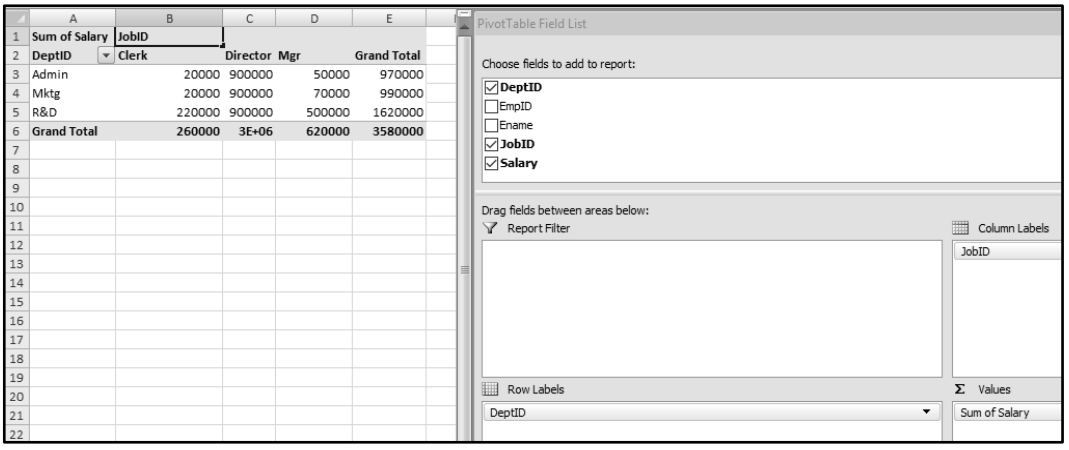




From Web
From Text
From Other Sources
Existing Connections
Refresh All
Connections
Properties
Edit Links
Sort
Filter
Data Connection Wizard
Select Database and Table
Select the Database and Table/Cube which contains the data you want.
Select the database that contains the data you want:
pivot_demo
Connect to a specific table:
Name
Description
Modified
Created
Type
Employee
12/5/2018 6:57:54 PM
TABLE
Cancel
< Back
Next >
Finish


From Web
From Text
From Other Sources
Existing Connections
Refresh All
Properties
Edit Links
Sort
Filter
Get External Data
Connections
Sort & Filter
Data Connection Wizard
Save Data Connection File and Finish
Enter a name and description for your new Data Connection file, and press Finish to save.
File Name:
LAPTOP-3484CI7B_SQLEXPRESS pivot_demo Employee.odc
Browse...
Save password in file
Description:
(To help others understand what your data connection points to)
Friendly Name:
LAPTOP-3484CI7B_SQLEXPRESS pivot_demo Employee
Search Keywords:
Always attempt to use this file to refresh data
Excel Services:
Authentication Settings...
Cancel
< Back
Next >
Finish




In next window, Select RowLabels as DeptID, ColumnLabels as JobID, and  Values as Sum of Salary as follows :




We will get **Pivot table** as below :

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Sum of Salary** | **JobID** | | | |
| 2 | **DeptID** | **Clerk** | **Director** | **Mgr** | **Grand Total** |
| 3 | Admin | 20000 | 900000 | 50000 | 970000 |
| 4 | Mktg | 20000 | 900000 | 70000 | 990000 |
| 5 | R&D | 220000 | 900000 | 500000 | 1620000 |
| 6 | **Grand Total** | **260000** | **2700000** | **620000** | **3580000** |

# Practical – 1B Perform the analysis for the following:

## Import the cube in microsoft excel and create the pivot table and pivot chart to perform data analysis.

SOLUTION :

For this practical, we will have to use SQL Server and Ms. Excel.

PART–1 : WORKING IN SQL SERVER : ( REF. muresults.net).

1. ***Create Cube in SQL :***

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).

i) Open SQL Server Management Studio 2008.

ii) Connect Database Engine.

iii) Open **New Query** editor.

iv) Copy paste Scripts given below in various steps in new query editor window one by one.

v) To run the given SQL Script, press **F5.**

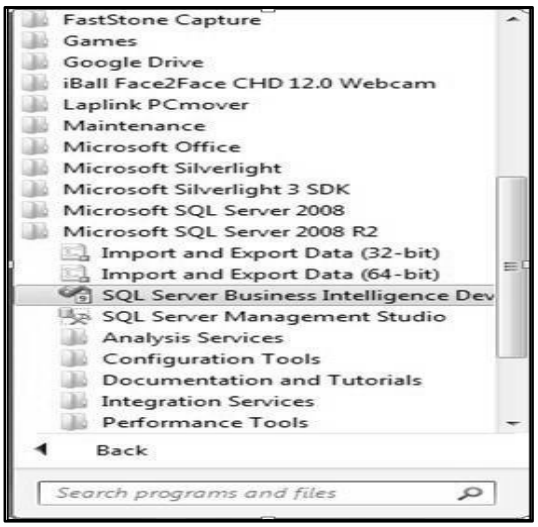
vi) It will create and populate "Sales_DW" database on your SQL Server.

2. Developing an OLAP Cube :

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

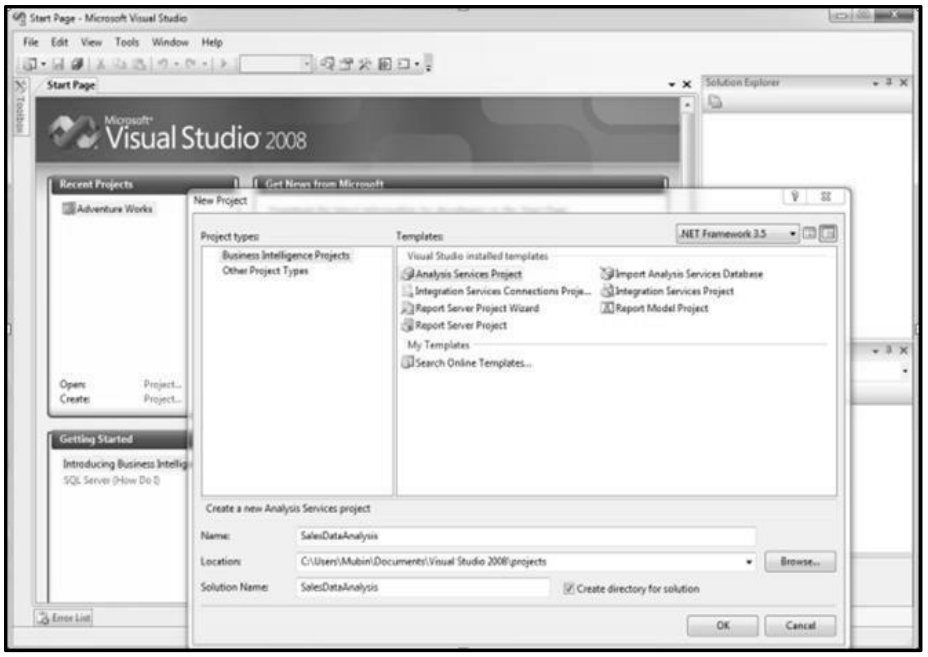
Step 1 : Start BIDS Environment.

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.

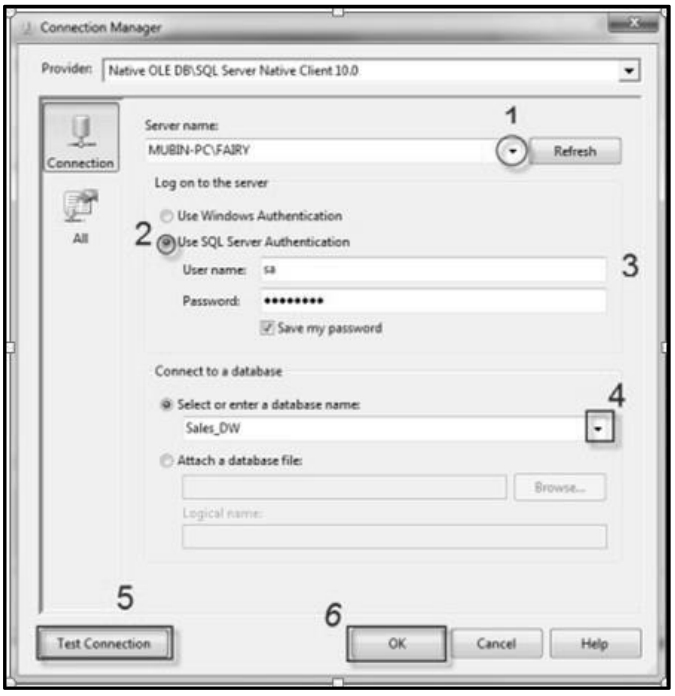
3.1 In Solution Explorer, Right click on **Data Source →** Click **New Data Source.**

3.2 Click on **Next.**

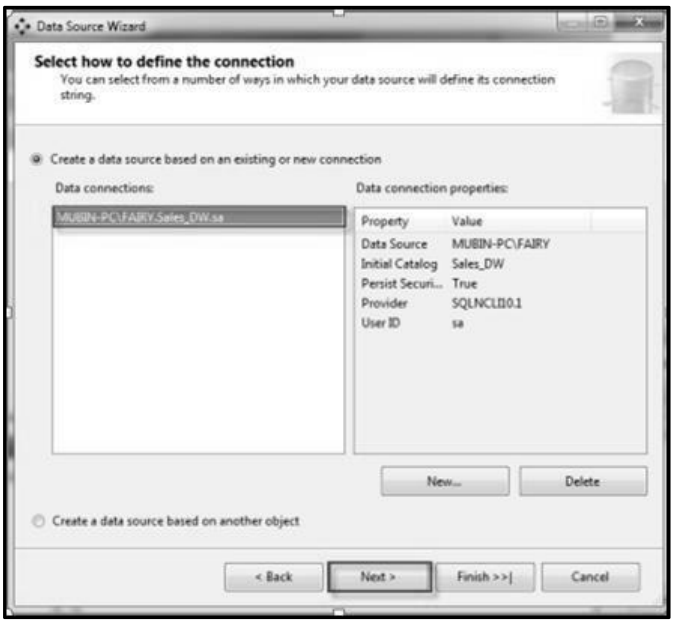

3.3 Click on New **Button.**

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.**

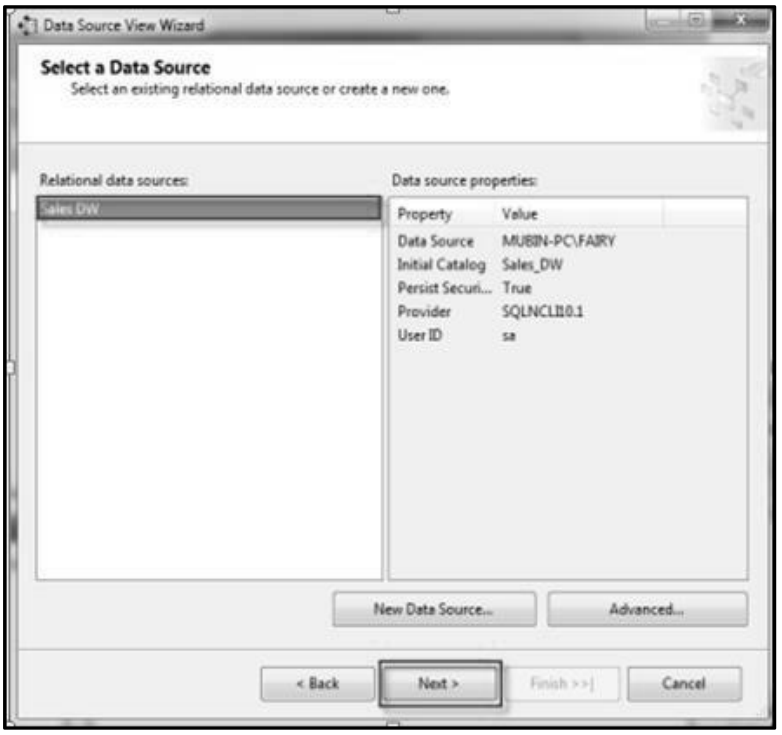

3.7 Assign Data Source **Name** → Click **Finish.**


Step 4 : Creating New Data Source View.

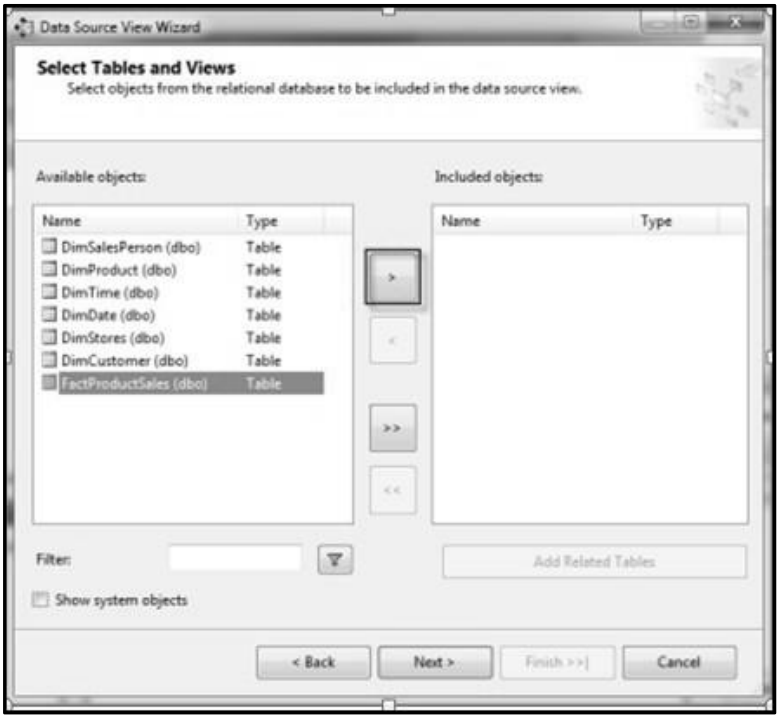
4.1 In the Solution Explorer, Right Click on **Data Source View** → Click on
New Data Source View.


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.



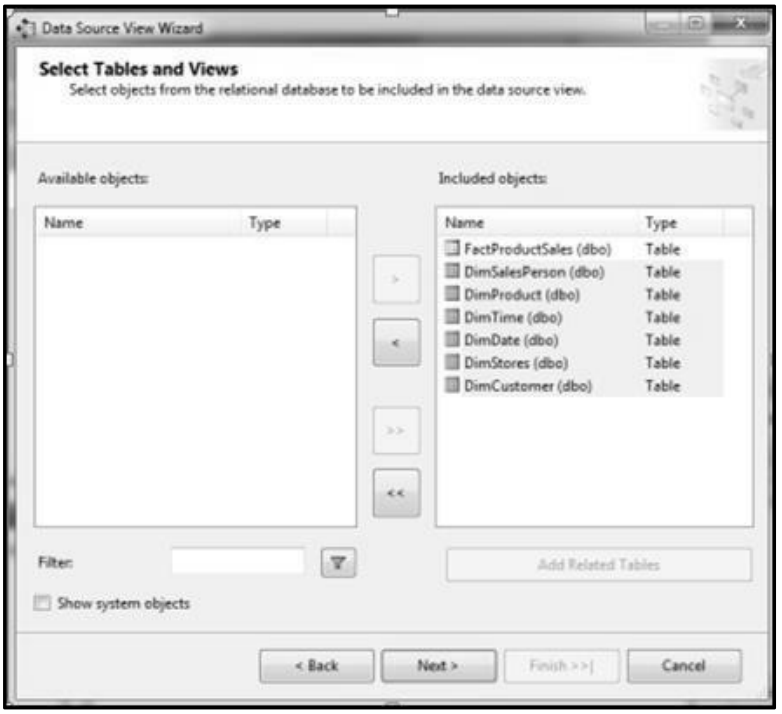

Select FactProductSales Table → Click on Arrow Button to move the selected object to Right Pane.

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.**

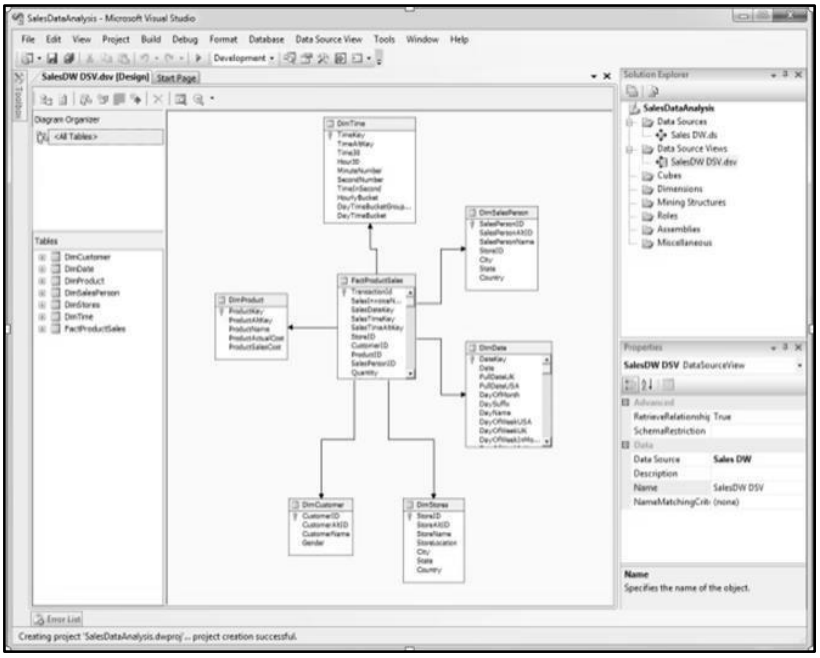

4.6 It will add all associated dimensions to your Fact table as per relationship specified in your SQL DW (Sales_DW).

Click **Next**.



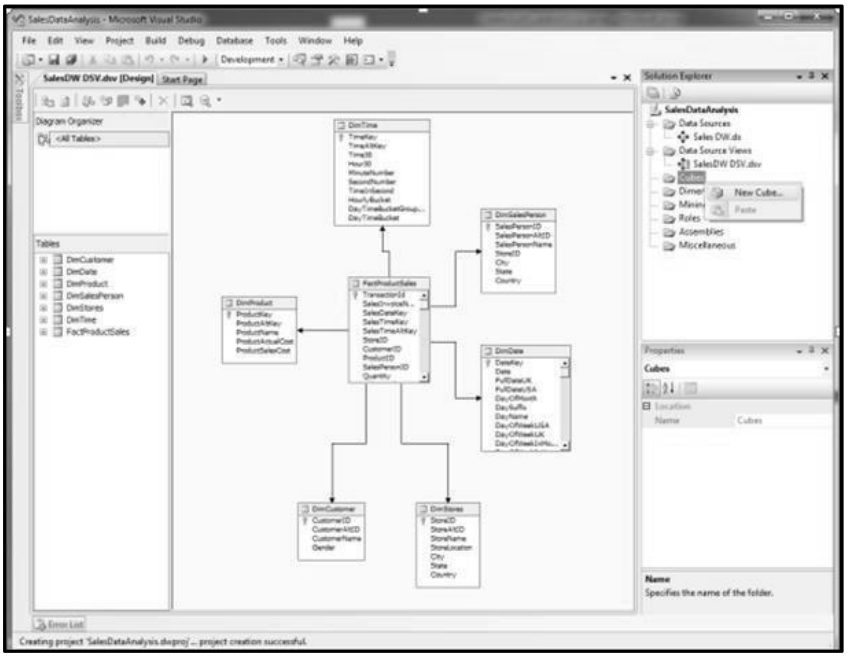
4.7 Assign **Name (SalesDW DSV)** → Click **Finish.**


4.8 Now Data Source View is ready to use.



Step 5 : Creating New Cube.

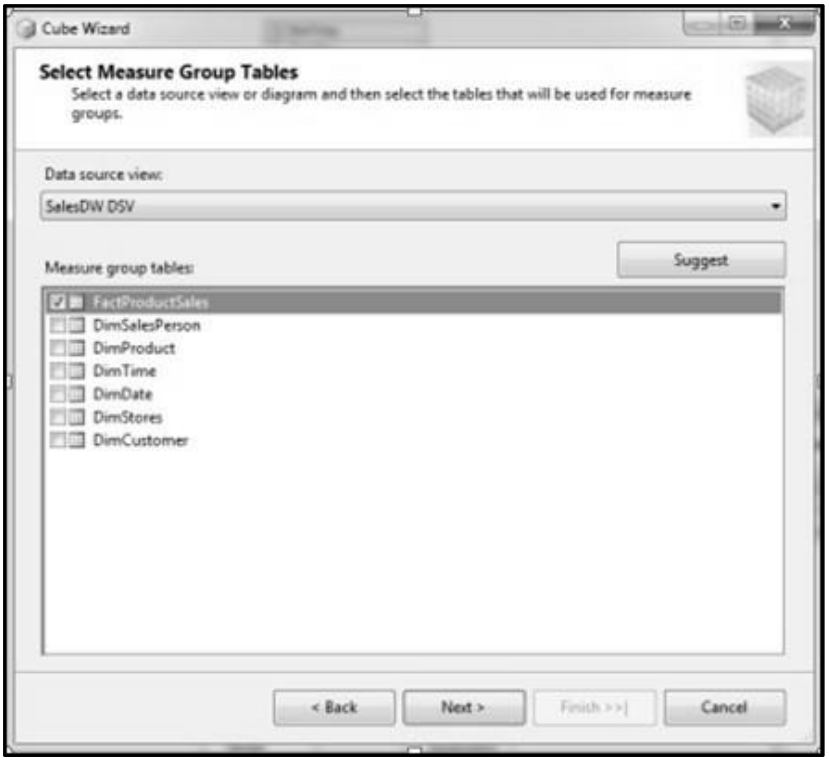
5.1 In Solution Explorer → Right Click on **Cube** → Click **New Cube.**



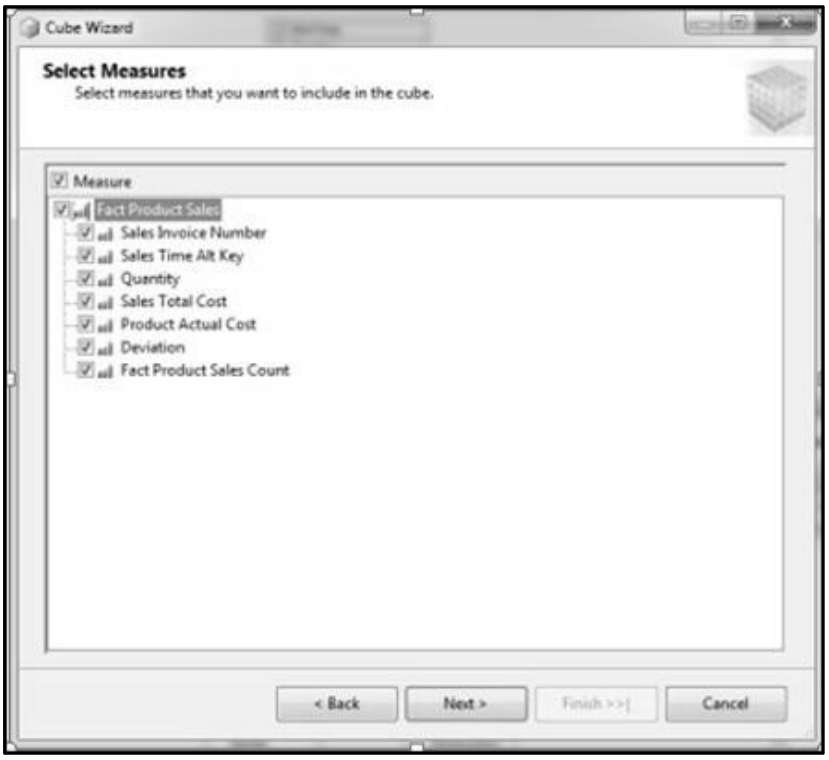

5.2 Click **Next.**

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.**



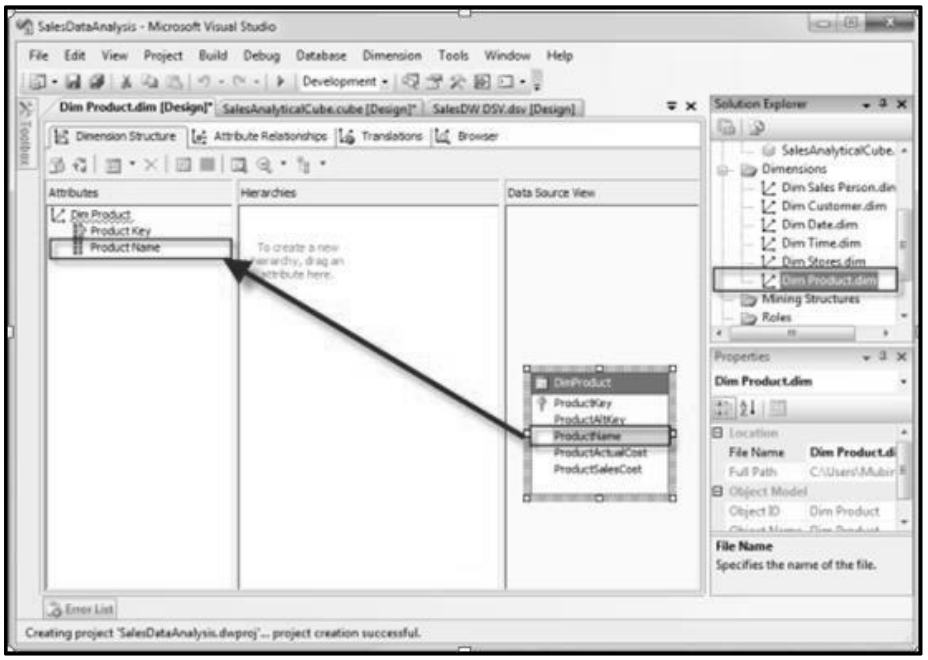

5.6 Select All **Dimensions** here which are associated with your Fact Table
→ Click **Next.**

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.

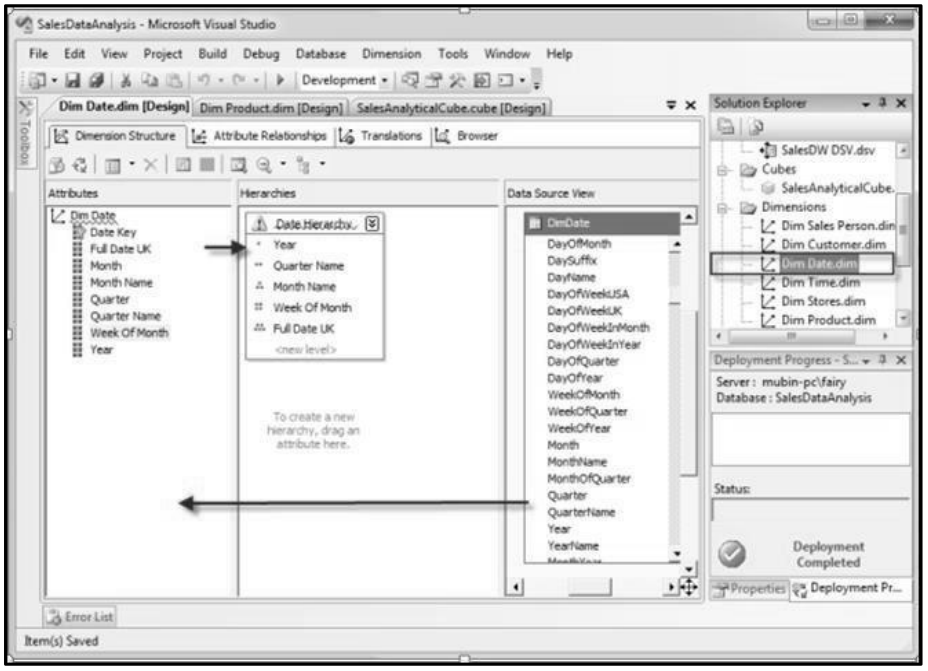
In Solution Explorer, double click on dimension **Dim Product** → Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side.



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) →

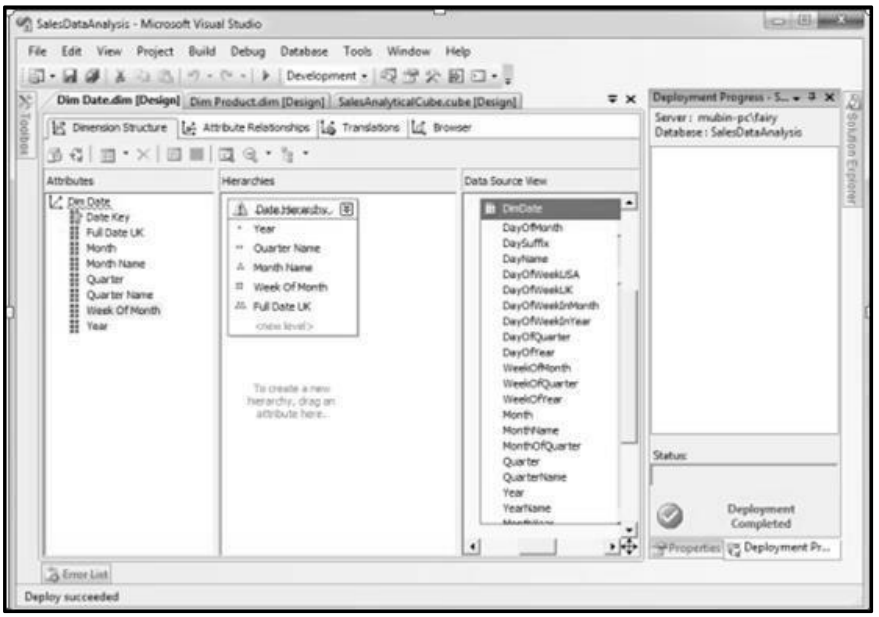
Click **Properties.**

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.**

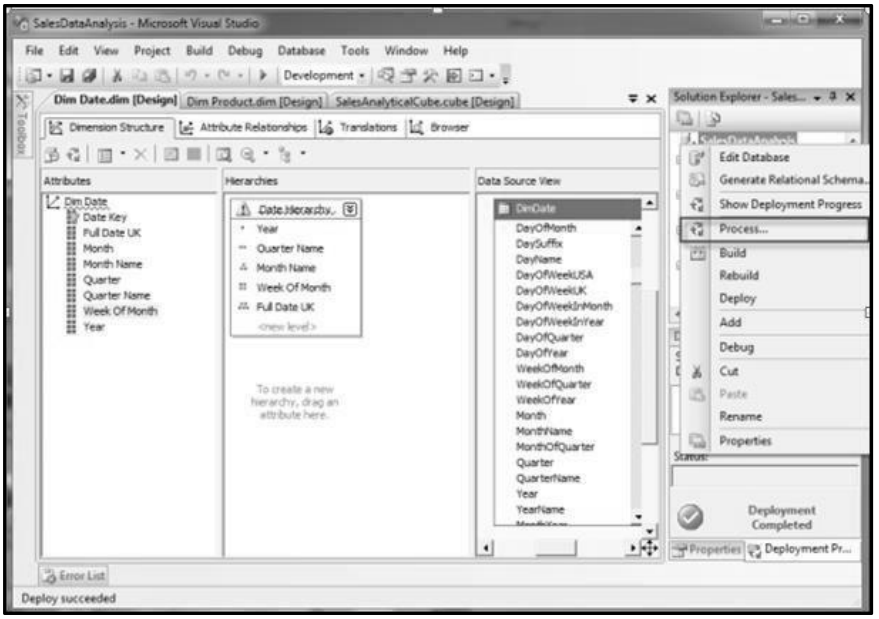

8.3 In Solution Explorer, right click on **Project Name** (SalesDataAnalysis)
→ Click **Deploy.**

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

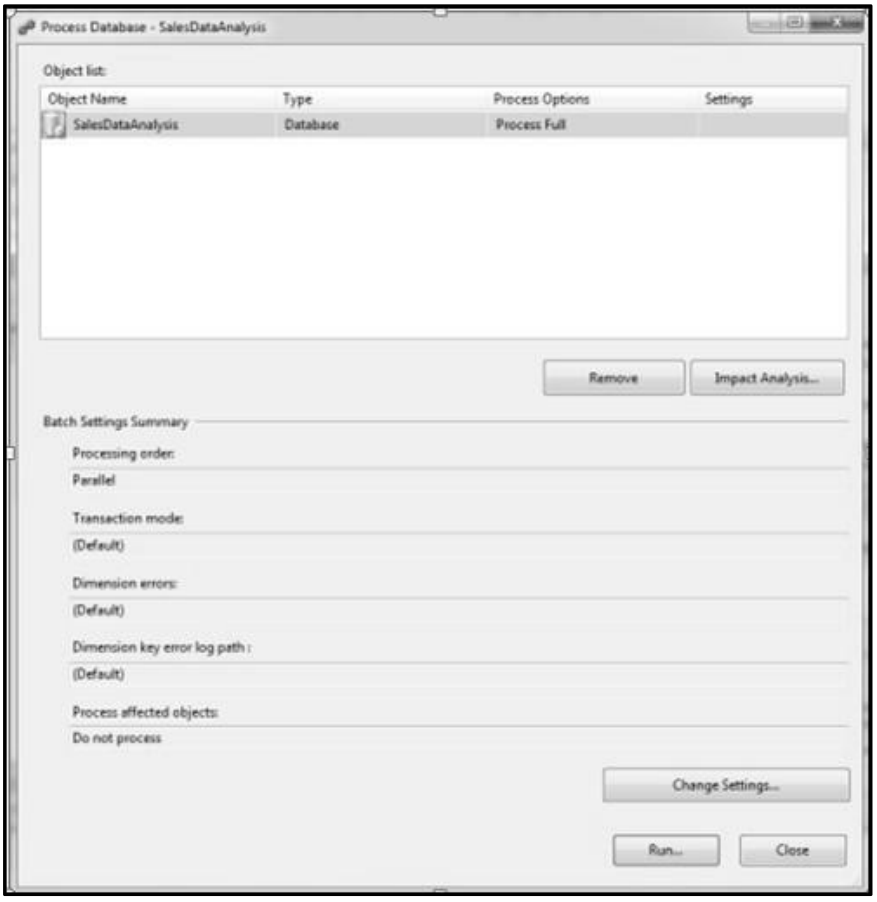



Step 9 : Process the Cube.

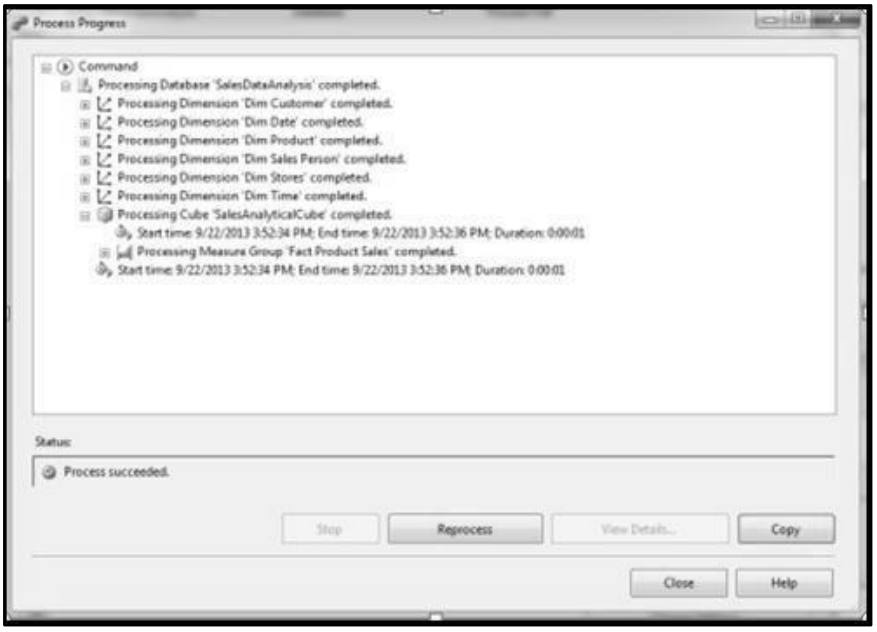
9.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) →
Click **Process.**



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

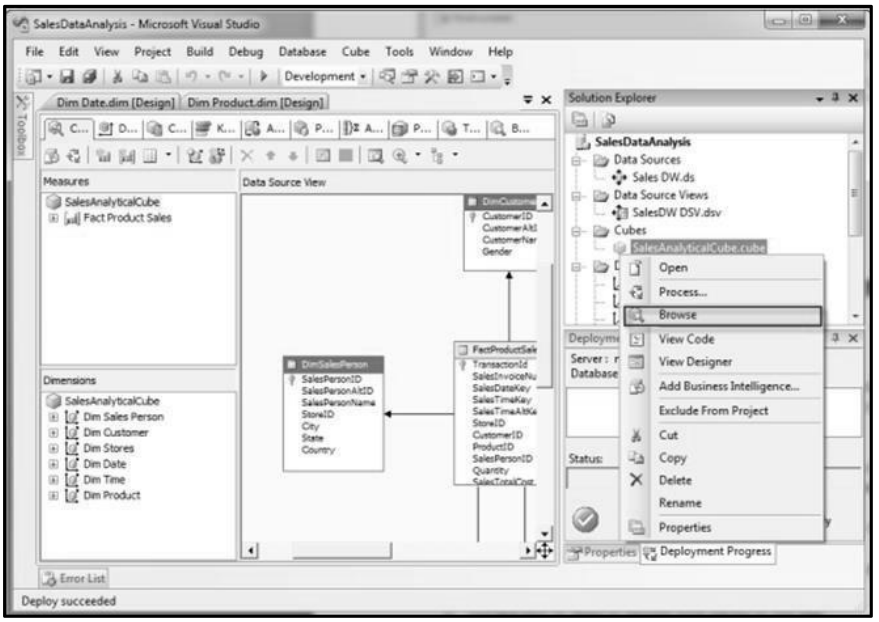



9.3 Once processing is complete, you can see **Status** as **Process Succeeded** → Click **Close** to close both the open windows for processing one after the other.



Step 10 : Browse the Cube for Analysis.

10.1 In Solution Explorer, right click on Cube Name (SalesDataAnalysisCube)

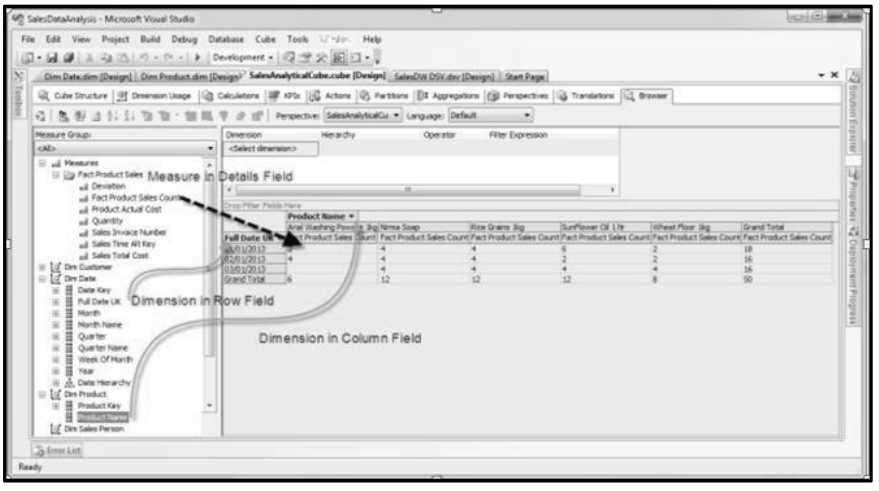



→ Click **Browse.**

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

Now to **Browse Our Cube :**

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




PART–2 : IMPORT CUBE ALREADY PREPARED IN MS. EXCEL.

Step 1 : Open Excel, Click on Data → From Other Sources → From Analysis Services.

Step 2 : Select SQL Server Name and Windows Authentication (or as per security set in your system for SQL).

Step 3 : Select Database Name and Name of the Cube created.

Step 4 : Browse and select the path for selection of data connection (*.odc ) file and click on Excel Services : Authentication Setting.

Step 5 : Select Pivot Table Report and select existing worksheet, then click on OK.

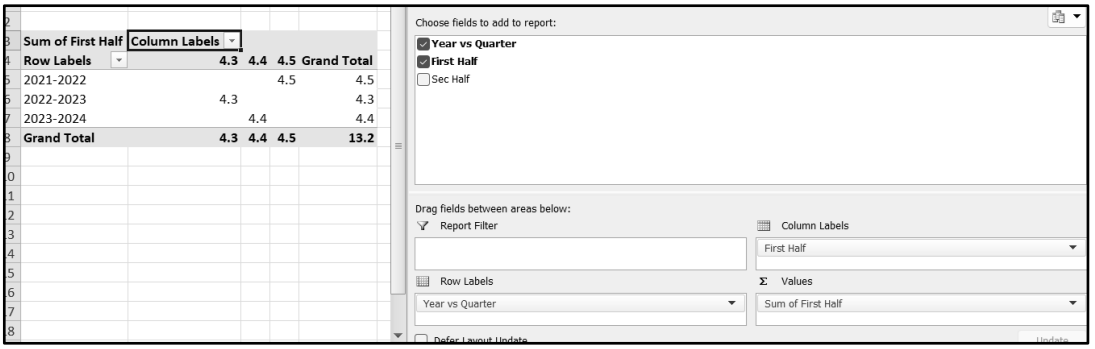
For Example :

Following data are imported for 3 years data collection for total sales in2 halves of each year.

| | A | B | C |
|---|---|---|---|
| 1 | **Three years Quaterwise Sales Data in Lakh Rs./-** | | |
| 2 | | | |
| 3 | **Year vs Quarter** | **First Half** | **Sec Half** |
| 4 | **2021-2022** | 4.5 | 5.5 |
| 5 | **2022-2023** | 4.3 | 5.3 |
| 6 | **2023-2024** | 4.4 | 5.4 |
| 7 | | | |

Then Create a pivot table/report as follows :

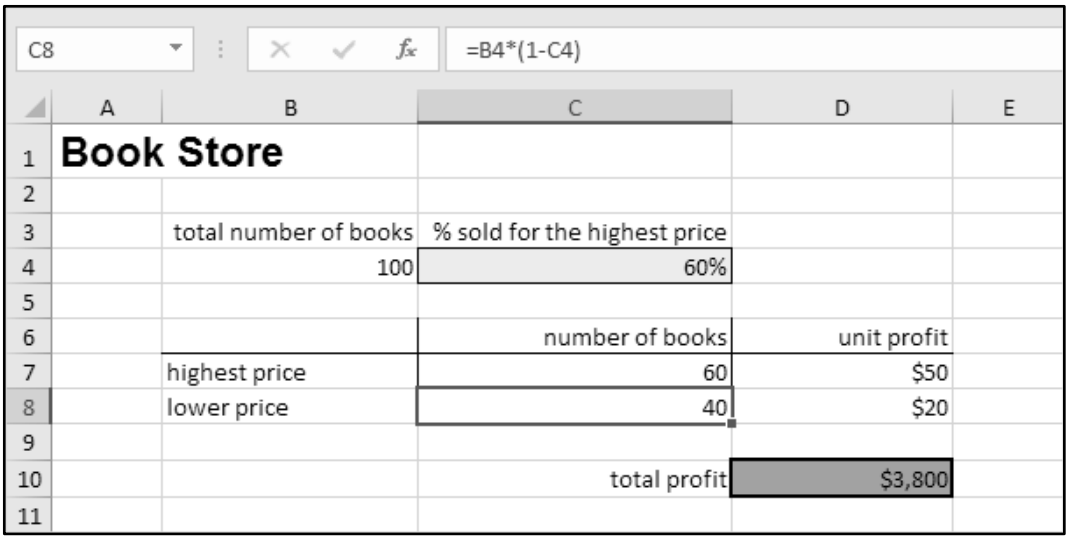
Click on menu Insert → Pivot Table, You will get a window as below, select Year as row data and First Half and Sec Half as column data , as well as in  Values as shown in below:



## Practical 2:

## Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data. Use Excel.

EXAMPLE 1 :



C8 | =B4*(1-C4)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Book Store** | | | | |
| 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 | | | | | |

A bookstore 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. If you sell 60% for the highest price, cell D10 calculates a total profit of

60 * $50 + 40 * $20= $3800.

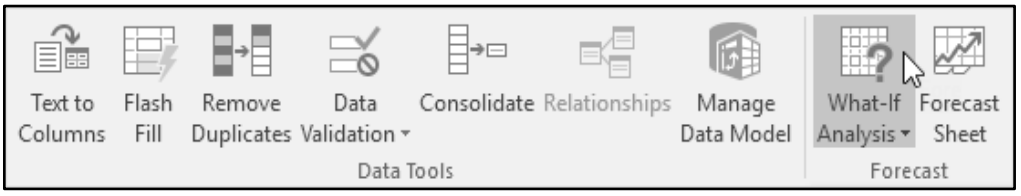
Create Different Scenarios :

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.

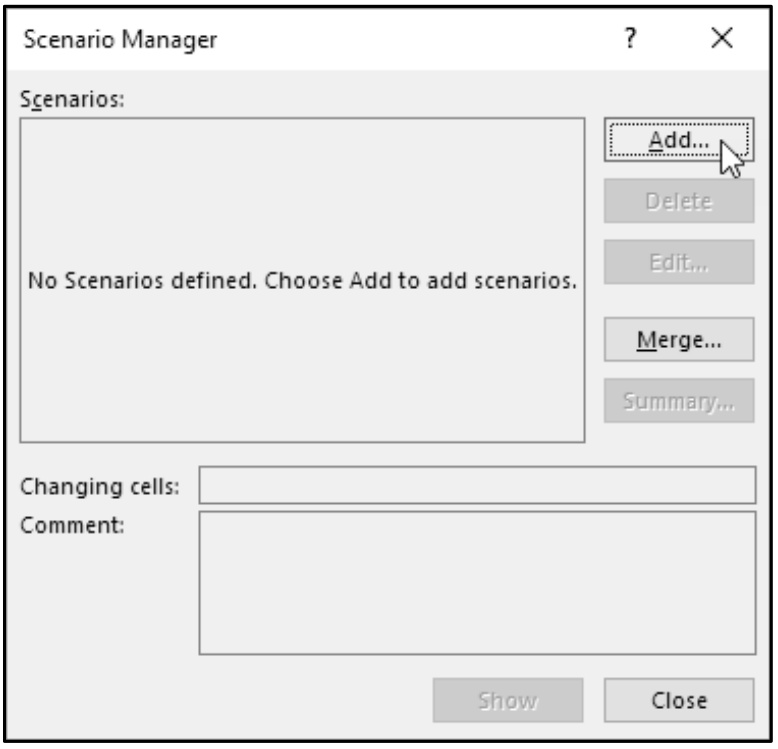
1. On the Data tab, in the Forecast group, click What□If Analysis.



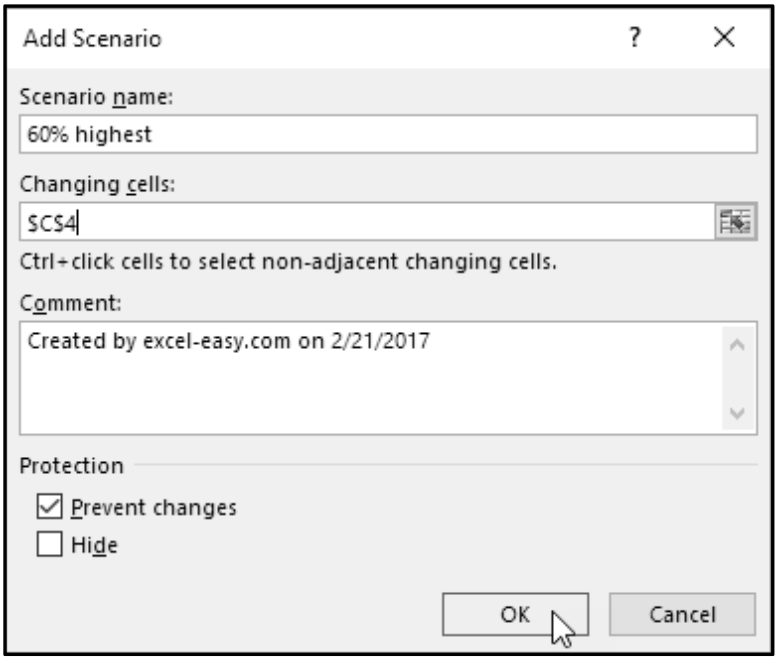
2. Click Scenario Manager.


The Scenario Manager dialog box appears.

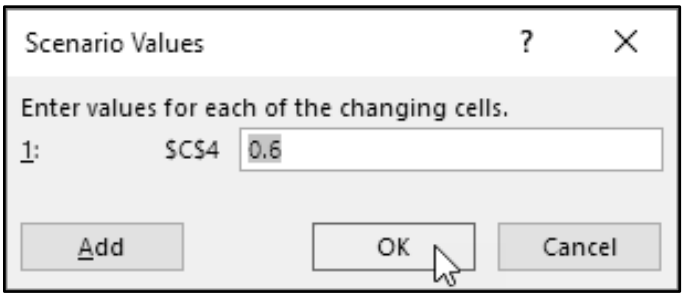
3. Add a scenario by clicking on Add.



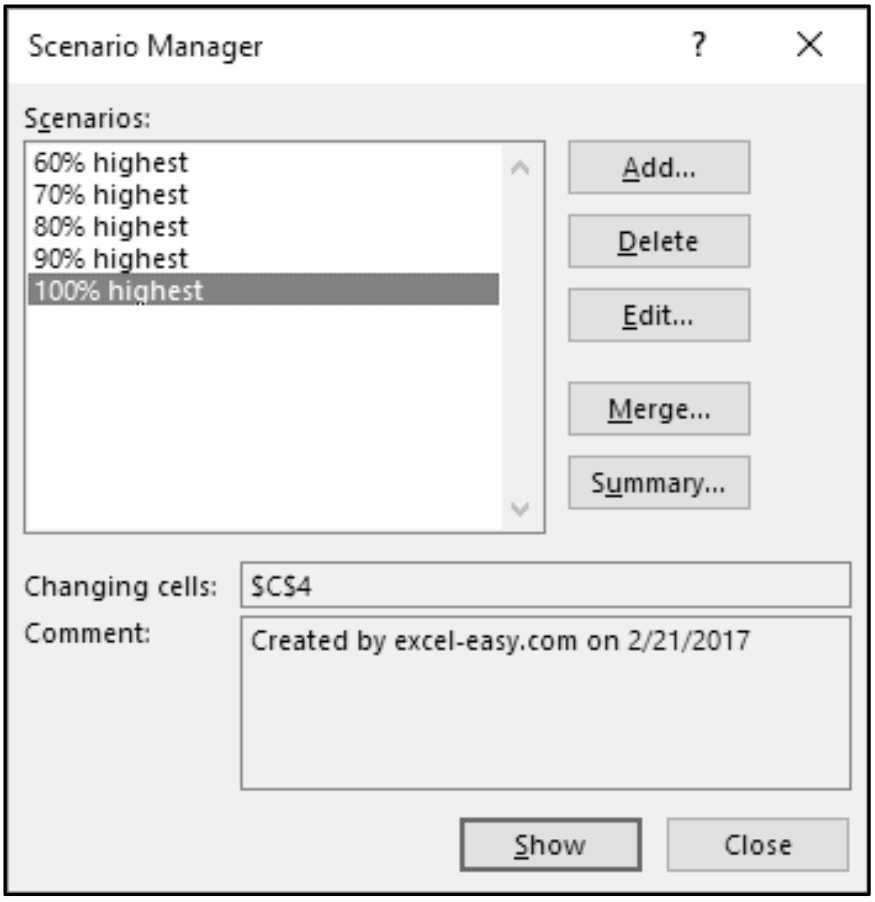

4. Type a name (60% highest), select cell C4 (% sold for the highest price) for the Changing cells and click on OK.



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 :




*(Ref. for Above example is "https://muresults.net")*

EXAMPLE 2 :

Note :

To solve above problem, We are using data warehousing of inventory system stored in SQL.

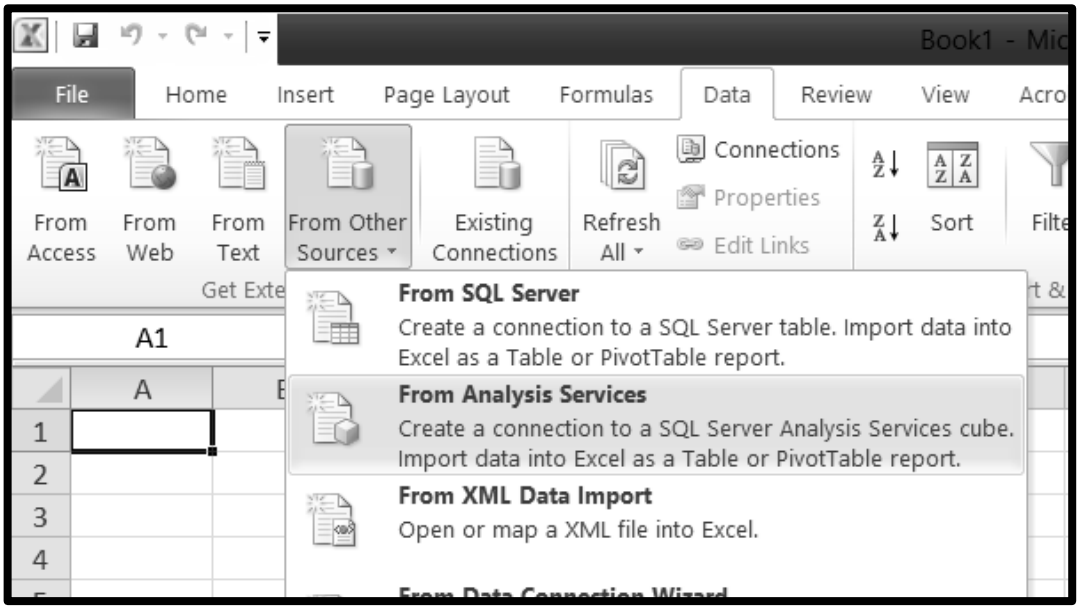
Here, Gross Income = Total Sales + Commission gain,

Total Deduction = Total Cost Price + Exchange Rate + Tax Payable + Other

Cost Net Profit = Gross Income – Total Deduction

Import table stored in SQL are imported in Excel as shown below :

(Click on Menu Data → From Other Sources → From Analysis Services, then fill server name, User ID and password in SQL)




Assume, following Data will be imported from SQL :

| | Category | Rs. In Lakhs |
|---|---|---|
| | Commission gain | 5000 |
| | Total Sales | 500000 |
| | **Gross Income** | **505000** |
| | Total Cost price | 300000 |
| | Exchange Rate | 3000 |
| | Tax Payable | 50000 |
| | Other Cost | 4000 |
| | **Total Deduction** | **357000** |
| | **Net Profit** | **148000** |

Inventory | Sheet3

What□If Analysis :

What would be Net Profit on variation of amount of Total Sales and Other cost? We will solve it by using What□If Analysis's Data Table techniques as follows : Modify above sheets as follows :

Apply value = D11 at cell "G2" and "J2".

Fill various value for total sales amount from F3: F11 and Other Cost I3:I11 respectively. Select Range F2:G11.

The status of worksheet is as follows :

| C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|
| | | | Variation in total sales amount | Respective Net Profit | | Variation in total sales amount | Respective Net Profit |
| **Category** | **Rs. In Lakhs** | | | 148000 | | | 148000 |
| Commission gain | 5000 | | 505000 | | | 5000 | |
| Total Sales | 500000 | | 510000 | | | 6000 | |
| **Gross Income** | **505000** | | 515000 | | | 7000 | |
| Total Cost price | 300000 | | 520000 | | | 8000 | |
| Exchange Rate | 3000 | | 525000 | | | 9000 | |
| Tax Payable | 50000 | | 530000 | | | 10000 | |
| Other Cost | 4000 | | 535000 | | | 11000 | |
| **Total Deduction** | **357000** | | 540000 | | | 12000 | |
| **Net Profit** | **148000** | | 545000 | | | 13000 | |

Click on Menu → Data → What□If Analysis → Data Table. You will get input popup as follows :

| | | | Variation in total sales amount | Respective Net Profit | | | |
|---|---|---|---|---|---|---|---|
| **Category** | **Rs. In Lakhs** | | | 148000 | | | |
| Commission gain | 5000 | | 505000 | | | | |
| Total Sales | 500000 | | 510000 | | | 6000 | |
| **Gross Income** | **505000** | | 515000 | | | 7000 | |
| Total Cost price | 300000 | | 520000 | | | 8000 | |
| Exchange Rate | 3000 | | 525000 | | | 9000 | |
| Tax Payable | 50000 | | 530000 | | | 10000 | |
| Other Cost | 4000 | | 535000 | | | 11000 | |
| **Total Deduction** | **357000** | | 540000 | | | 12000 | |
| **Net Profit** | **148000** | | 545000 | | | 13000 | |

Data Table ? ×
Row input cell:
Column input cell:
OK Cancel

At Column input cell, enter "$D$4", and click on OK.

Respective Net Profit will be populated automatically in range "G3:G11".

Similarly, by applying Column input cell value for Other Cost as "$D$9", Respective Net
Profit will be populated automatically in range "J3:J11".

Output will be :

| C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|
| | | | Variation in Total Sales amount | Respective Net Profit | | Variation in Other Cost amount | Respective Net Profit |
| **Category** | **Rs. In Lakhs** | | | 148000 | | | 148000 |
| Commission gain | 5000 | | 505000 | 153000 | | 5000 | 147000 |
| Total Sales | 500000 | | 510000 | 158000 | | 6000 | 146000 |
| **Gross Income** | **505000** | | 515000 | 163000 | | 7000 | 145000 |
| Total Cost price | 300000 | | 520000 | 168000 | | 8000 | 144000 |
| Exchange Rate | 3000 | | 525000 | 173000 | | 9000 | 143000 |
| Tax Payable | 50000 | | 530000 | 178000 | | 10000 | 142000 |
| Other Cost | 4000 | | 535000 | 183000 | | 11000 | 141000 |
| **Total Deduction** | **357000** | | 540000 | 188000 | | 12000 | 140000 |
| **Net Profit** | **148000** | | 545000 | 193000 | | 13000 | **139000** |

## Practical 3:
## Perform the data classification using classification algorithm using R/Python.

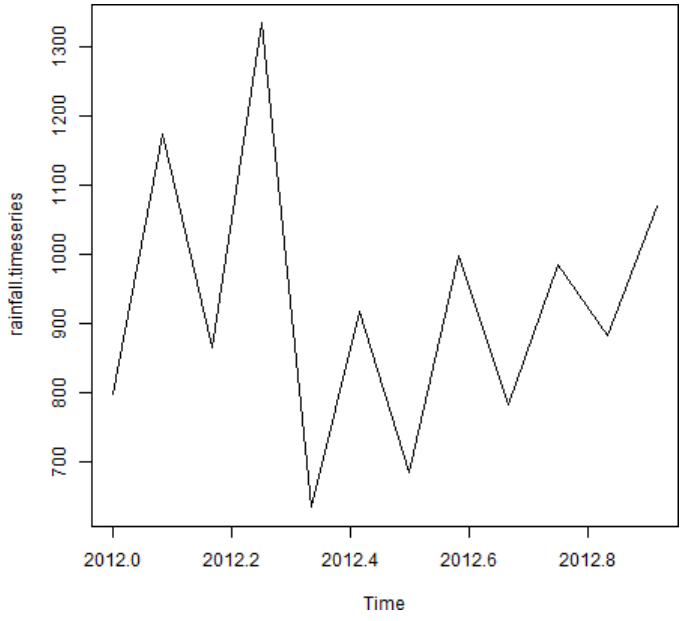
Consider the annual rainfall details at a place starting from January 2012.
We create an R time series object for a period of 12 months and plot it.

```
# 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 :

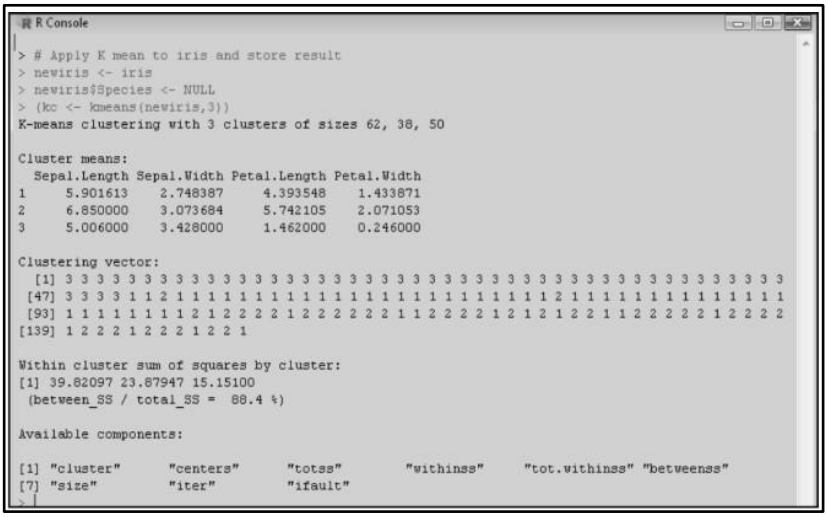
When we execute the above code, it produces the following result and chart :

| | 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 | | | | | | |

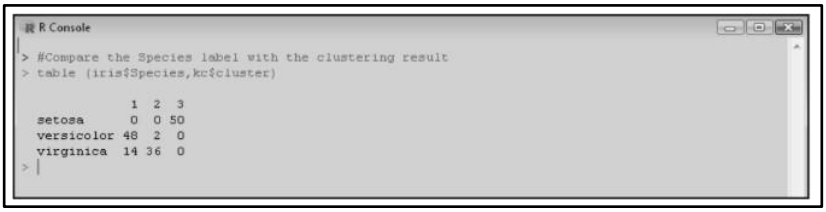


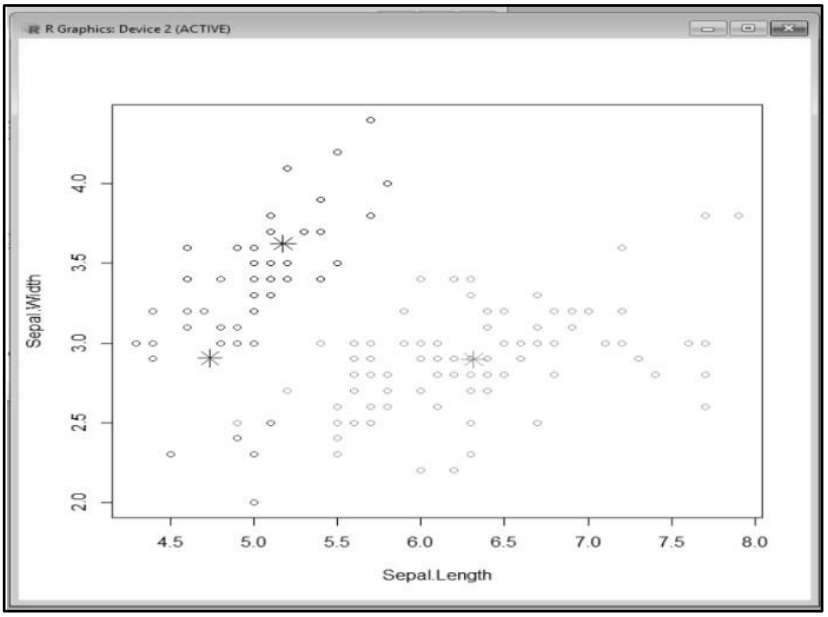
# Practical 4:
# Perform the data clustering using clustering algorithm using R/Python.

SOLUTION :




Compare the species label with the clustering result :



## Practical 5:

## Perform the Linear regression on the given data warehouse data using R/Python.

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 :

1. Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
2. Create a relationship model using the ***lm()*** functions in *R*.
3. Find the coefficients from the model created and create the mathematical equation using these.
4. Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
5. 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 :

i) ***formula*** is a symbol presenting the relation between *x* and *y*.

ii) ***data*** is the vector on which the formula will be applied.

Create Relationship Model and 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 :

(Intercept) *x*

---38.4551 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 | ---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 :

i) ***object*** is the formula which is already created using the *lm() function.*

ii) ***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 :

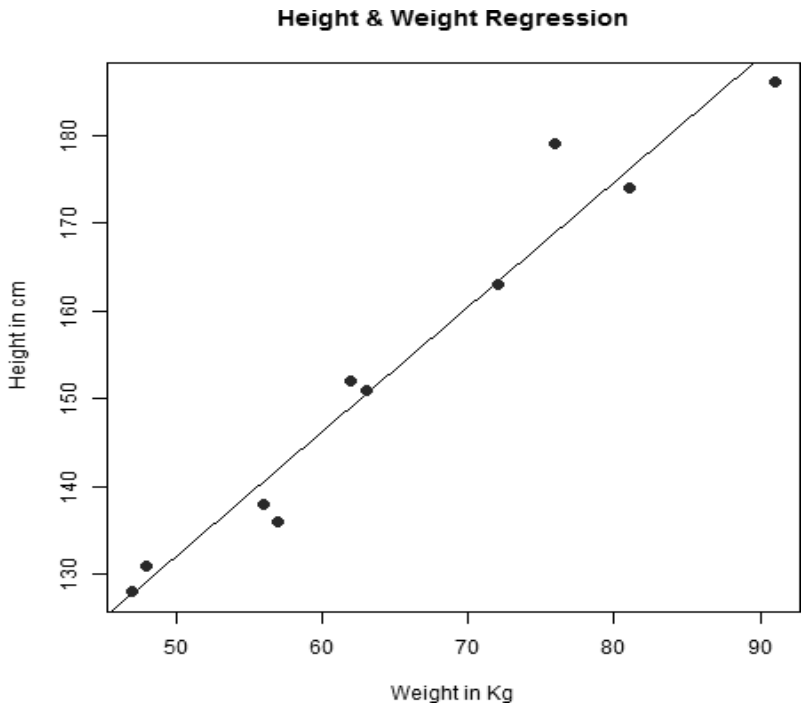
*# Create the predictor and response variable.*

*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:*

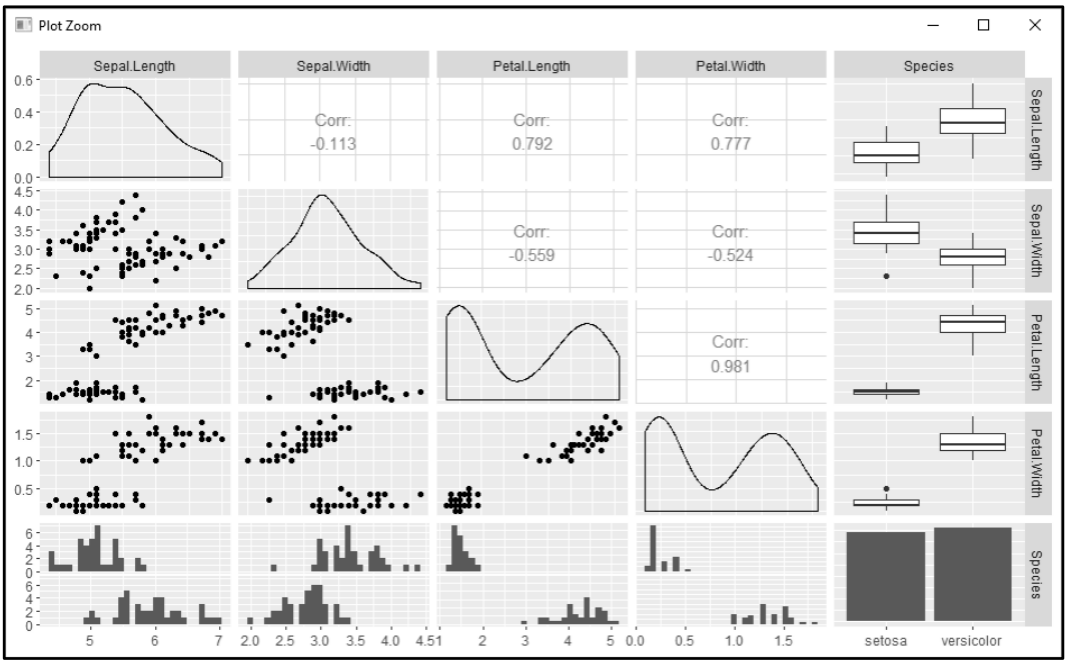


## Practical 6:
## Perform the logistic regression on the given data warehouse data using R/Python.

SOLUTION :

```
library(datasets)
ir_data<-iris
head(ir_data)
str(ir_data)
levels(ir_data$species)
sum(is.na(ir_data))
ir_data<-
ir_data[1:100,]
set.seed(100)
samp<-sample(1:100,80)
ir_test<-ir_data[samp,]
ir_ctrl<-ir_data[-samp,]
install.packages("ggplot2
") library(ggplot2)
install.packages("GGally"
) library(GGally)
ggpairs(ir_test)
y<-ir_test$Species;x<-
ir_test$Sepal.Length glfit<-
glm(y~x,family='binomial')cls
summary(glfit)
newdata<-data.frame(x=ir_ctrl$Sepal.Length)
predicted_val<-
predict(glfit,newdata,type="response")
prediction<-data.frame(ir_ctrl$Sepal.Length,ir_ctrl$Species,predicted_val)
prediction<-
    qplot(prediction[,1],round(prediction[,3]),col=prediction[,2],xlab='Sepal
    Length',ylab='prediction using Logistic Reg.')
```

OUTPUT :


Plot Zoom
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
Corr:
-0.113
Corr:
0.792
Corr:
0.777
Corr:
-0.559
Corr:
-0.524
Corr:
0.981
setosa
versicolor

## Practical 7:

## Write a Python program to read data from a CSV file, perform simple data analysis, and generate basic insights. (Use Pandas is a Python library).

**SOLUTION :**

To perform this practical create following "customers.csv" file having following

```
'CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)'
```

fields and insert 25 records :

**Apply following command :**

**1.** read() :

```
import pandas as pd

#Loading data into a DataFrame
data_frame=pd.read_csv('Mall_Customers.csv')
```

**2.** ***head(), tail()***

**#** ***data_frame.head()*** displays the first five rows and ***data_frame.tail()*** displays last five rows.

```
#displaying first five rows
display(data_frame.head())
#displaying last five rows
display(data_frame.tail())
```

# Program to print all the column name of the dataframe

```
print(list(data_frame.columns))
```

The functions ***info()*** prints the summary of a DataFrame that includes the data type of each column, RangeIndex (number of rows), columns, non-null values, and memory usage.

```
data_frame.info()
```

The ***describe()*** function outputs descriptive statistics which include those that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values.

```
data_frame.describe()
```

The ***data_frame.dropna( )*** function removes columns or rows which contains atleast one missing values.

```
data_frame = data_frame.dropna()
```

**Removing 4th indexed value from the dataframe :**

```
data_frame.drop(4).head()
```

Drop(i).head is used to remove $i^{th}$ index value from dataframe

**Renaming rows :**

The rename function can be used to rename the rows or columns of the data frame.

```
data_frame.rename({0:"First",1:"Second"})
```

```
data_frame['NewColumn'] = 1
data_frame.head()
```

To Create a new column with all the values equal to 1 :

The ***sort_values( )*** are the values of the column whose name is passed in the **by** attribute in the ascending order by default we can set this attribute to false to sort the array in the descending order.

```
data_frame.sort_values(by='Age',  ascending=False).head()
```

**Sort by multiple columns :**

```
data_frame.sort_values(by=['Age','Annual Income (k$)']).head(10)
```

The ***merge()*** function in pandas is used for all standard database join operations. Merge operation on data frames will join two data frames based on their common column values.

Let us create a data frame.

```
#Creating dataframe1

df1 = pd.DataFrame({
        'Name':['Jeevan', 'Raavan', 'Geeta', 'Bheem'],
        'Age':[25, 24, 52, 40],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']})
df1

#Creating dataframe2

df2 = pd.DataFrame({'Name':['Jeevan', 'Raavan', 'Geeta', 'Bheem'],
                    'Salary':[100000, 50000, 20000, 40000]})
df2

Merge two distinct data frames created earlier:
#Merging two dataframes
df = pd.merge(df1, df2)
df
```

Apply function on dataframe:

```
def fun(value):
    if value > 70:
        return "Yes"
    else:
        return "No"

data_frame['Customer Satisfaction'] = data_frame['Spending Score (1-100)'].apply(fun)
data_frame.head(10)
```

**Uses of the lambda operator :**

This syntax is generally used to apply log transformations and normalize the data to bring it in the range of 0 to 1 for particular columns of the data.

```
const = data_frame['Age'].max()
data_frame['Age'] = data_frame['Age'].apply(lambda x: x/const)
data_frame.head()
```

The **plot( )** function is used to make plots of the data frames. # Visualization

```
data_frame.plot(x ='CustomerID', y='Spending Score (1-100)',kind = 'scatter')
```

## Practical 8A: Perform data visualization
## Perform data visualization using Python on any sales data.

Create a CSV file, having field name as Item_Name and Qty_sold as listed below :

| Item_name | Qty_sold |
|---|---|
| Mouse | 98 |
| Keyboard | 75 |
| Pendrive | 50 |
| Speaker | 92 |
| Bluetooth | 90 |

To plot CSV data using Matplotlib and Pandas in Python, we can take the following steps:

*[**Introduction to Matplotlib :** Matplotlib is a Python library that lets you create plots, graphs and charts. It is a popular data visualization tool that can be used to create static, animated, and interactive visualizations.]*

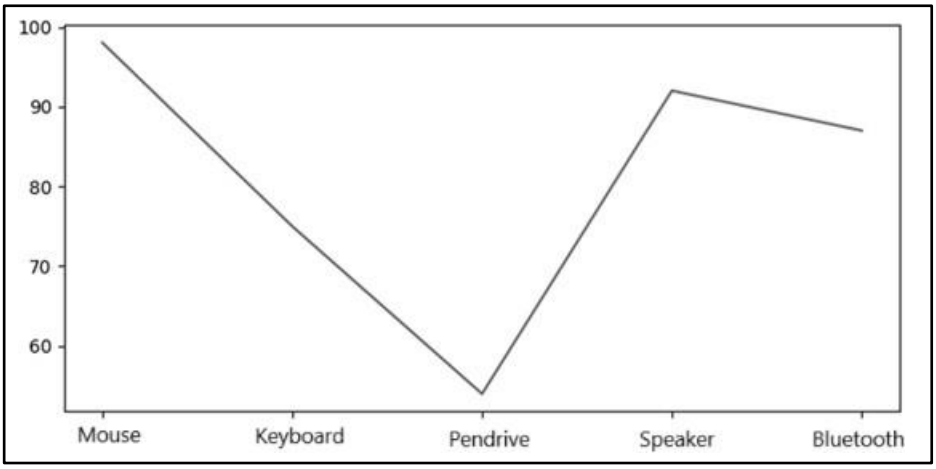
1. Set the figure size and adjust the padding between and around the subplots.
2. Make a list of headers of the .*CSV* file.
3. Read the CSV file with headers.
4. Set the index and plot the dataframe.
5. To display the figure, use ***show()***

method. Execute following code on python

IDLE :

```
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams["figure.figsize"] = [8.00, 4.50]
plt.rcParams["figure.autolayout"] = True
columns = ["Item_Name", "Qty_sold"]
df = pd.read_csv("Sales.csv",
usecols=columns) print("Contents in csv
file:", df) plt.plot(df.Item_Name, df.Qty_sold)
plt.show()
```

OUTPUT :



## Practical 8B: Perform data visualization

## Perform data visualization using PowerBI on any sales data.

**SOLUTION:**

Following "Northwind" database will have to create in database application. (It is a open source database, can be downloaded from internet.)

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings =
on; SET check_function_bodies = false;
SET client_min_messages = warning;



SET default_tablespace =

''; SET default_with_oids =

false;



---
--- drop tables
---


DROP TABLE IF EXISTS
customer_customer_demo; DROP TABLE IF
EXISTS customer_demographics; DROP
TABLE IF EXISTS employee_territories;
DROP TABLE IF EXISTS order_details;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS
customers; DROP TABLE IF
EXISTS products; DROP TABLE
IF EXISTS shippers; DROP TABLE
IF EXISTS suppliers;
```

```
DROP TABLE IF EXISTS
territories; DROP TABLE IF
EXISTS us_states; DROP TABLE
IF EXISTS categories; DROP
TABLE IF EXISTS region;
DROP TABLE IF EXISTS employees;

--
-- Name: categories; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE categories (
        category_id smallint NOT NULL PRIMARY KEY,
        category_name character varying(15) NOT NULL,
        description text,
        picture bytea
);
--
-- Name: customer_demographics; Type: TABLE; Schema: public;
Owner: -; Tablespace:
--

CREATE TABLE customer_demographics (
        customer_type_id bpchar NOT NULL PRIMARY
        KEY, customer_desc text
);
--
-- Name: customers; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE customers (
        customer_id bpchar NOT NULL PRIMARY
        KEY, company_name character varying(40)
        NOT NULL, contact_name character
        varying(30),  contact_title character
        varying(30),
        address character
        varying(60), city character
        varying(15), region character
        varying(15),
        postal_code character
        varying(10), country character
        varying(15),
```

```
        phone character varying(24),
        fax character varying(24)
);
--
-- Name: customer_customer_demo; Type: TABLE; Schema: public;
Owner: -; Tablespace:
--

CREATE TABLE customer_customer_demo (
        customer_id bpchar NOT NULL,
        customer_type_id bpchar NOT NULL,
        PRIMARY KEY (customer_id, customer_type_id),
        FOREIGN KEY (customer_type_id) REFERENCES
        customer_demographics, FOREIGN KEY (customer_id) REFERENCES
        customers
);
--
-- Name: employees; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE employees (
        employee_id smallint NOT NULL PRIMARY
        KEY, last_name character varying(20) NOT
        NULL, first_name character varying(10)
        NOT NULL, title character varying(30),
        title_of_courtesy character varying(25),
        birth_date date,
        hire_date date,
        address character
        varying(60), city character
        varying(15), region character
        varying(15),
        postal_code character
        varying(10), country character
        varying(15), home_phone
        character varying(24), extension
        character varying(4), photo bytea,
        notes text,
        reports_to
        smallint,
        photo_path character varying(255),
FOREIGN KEY (reports_to) REFERENCES employees
        );
```

```
--
-- Name: suppliers; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE suppliers (
        supplier_id smallint NOT NULL PRIMARY
        KEY, company_name character varying(40)
        NOT NULL, contact_name character
        varying(30),  contact_title character
        varying(30),
        address character
        varying(60), city character
        varying(15), region character
        varying(15),
        postal_code character
        varying(10), country character
        varying(15), phone character
        varying(24),
        fax character varying(24),
        homepage text
);
--
-- Name: products; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE products (
        product_id smallint NOT NULL PRIMARY KEY,
        product_name character varying(40) NOT NULL,
        supplier_id smallint,
        category_id smallint,
        quantity_per_unit character varying(20),
        unit_price real,
        units_in_stock smallint,
        units_on_order smallint,
        reorder_level smallint,
        discontinued integer NOT NULL,
            FOREIGN KEY (category_id) REFERENCES categories,
            FOREIGN KEY (supplier_id) REFERENCES suppliers
);
--
-- Name: region; Type: TABLE; Schema: public; Owner: -; Tablespace:
```

```
--

CREATE TABLE region (
        region_id smallint NOT NULL PRIMARY
        KEY, region_description bpchar NOT NULL
);


--
-- Name: shippers; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE shippers (
        shipper_id smallint NOT NULL PRIMARY KEY,
        company_name character varying(40) NOT
        NULL, phone character varying(24)
);


--
-- Name: orders; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE orders (
        order_id smallint NOT NULL PRIMARY
        KEY, customer_id bpchar,
        employee_id smallint,
        order_date date,
        required_date date,
        shipped_date date,
        ship_via smallint,
        freight real,
        ship_name character varying(40),
        ship_address character varying(60),
        ship_city character varying(15),
        ship_region character varying(15),
        ship_postal_code character varying(10),
        ship_country character varying(15),
        FOREIGN KEY (customer_id) REFERENCES
        customers, FOREIGN KEY (employee_id)
        REFERENCES employees, FOREIGN KEY
        (ship_via) REFERENCES shippers
);
```

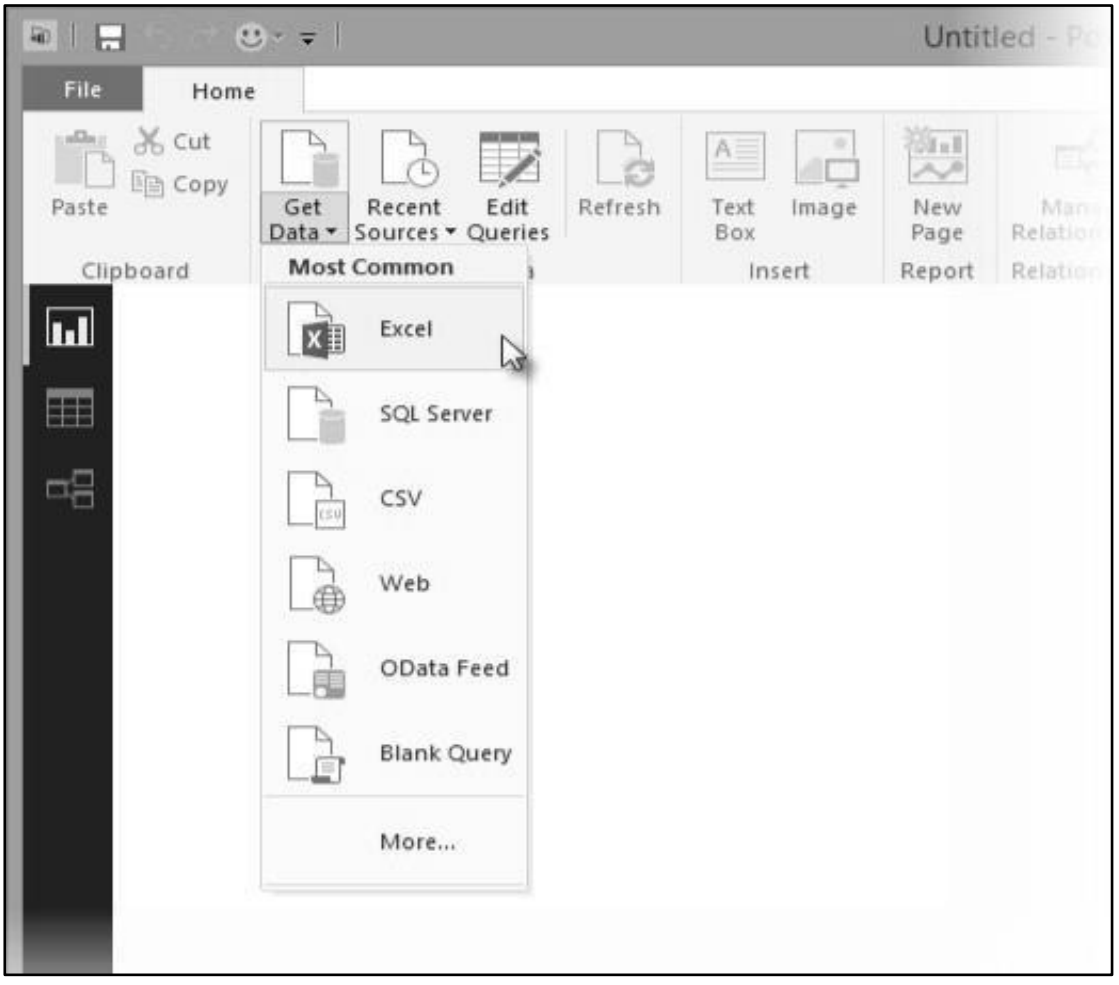
```
--
-- Name: territories; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE territories (
    territory_id character varying(20) NOT NULL PRIMARY KEY,
    territory_description bpchar NOT NULL,
    region_id smallint NOT NULL,
        FOREIGN KEY (region_id) REFERENCES region
);



--
-- Name: employee_territories; Type: TABLE; Schema: public; Owner: -;
Tablespace:
--

CREATE TABLE employee_territories (
    employee_id smallint NOT NULL,
    territory_id character varying(20) NOT
    NULL, PRIMARY KEY (employee_id,
    territory_id),
    FOREIGN KEY (territory_id) REFERENCES
    territories, FOREIGN KEY (employee_id)
    REFERENCES employees
);
--
-- Name: order_details; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE order_details (
    order_id smallint NOT NULL,
    product_id smallint NOT
    NULL, unit_price real NOT
    NULL, quantity smallint NOT
    NULL, discount real NOT
    NULL,
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (product_id) REFERENCES products,
    FOREIGN KEY (order_id) REFERENCES orders
);
```

```
-- Name: us_states; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE us_states (
    state_id smallint NOT NULL PRIMARY
    KEY, state_name character varying(100),
    state_abbr character varying(2),
    state_region character varying(50)
);
```

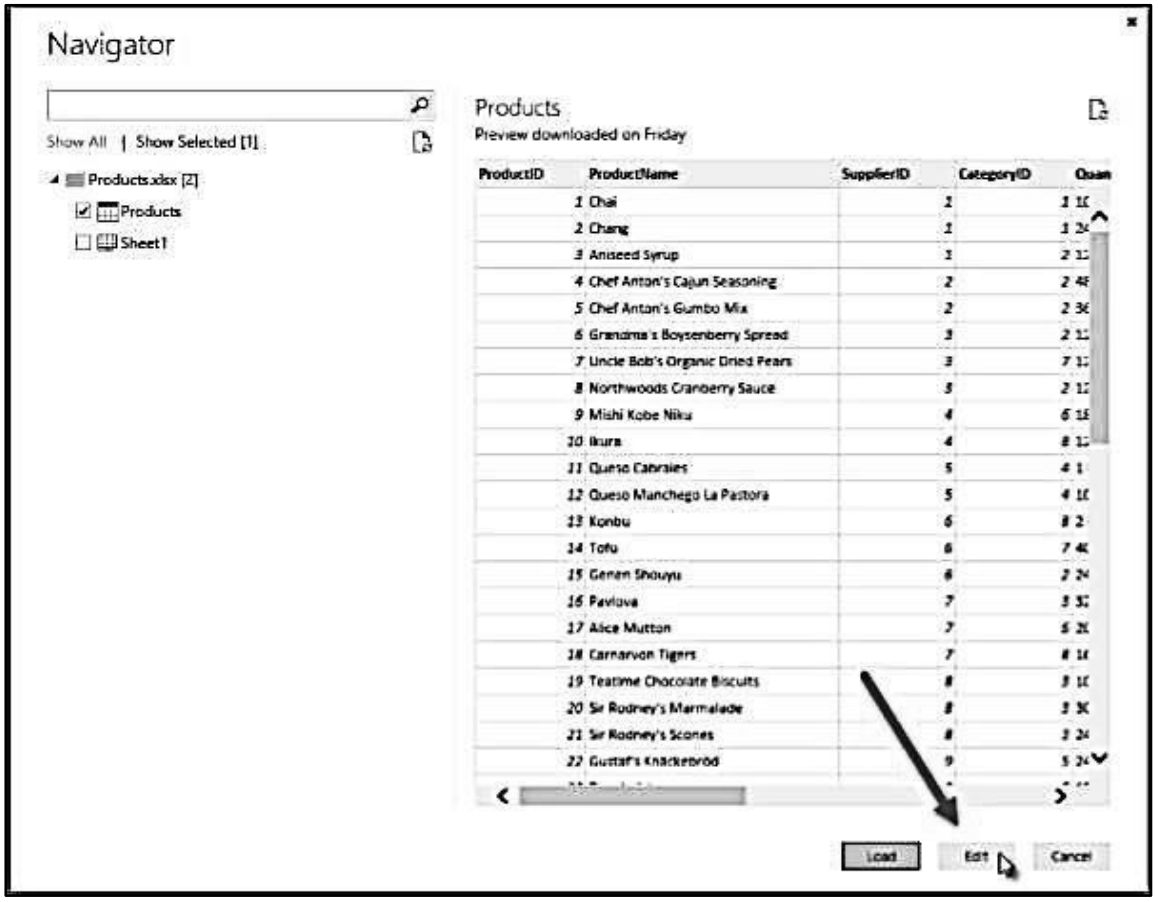
SOLUTION FOR "PERFORM DATA VISUALIZATION USING POWERBI ON ANY SALES DATA":

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.



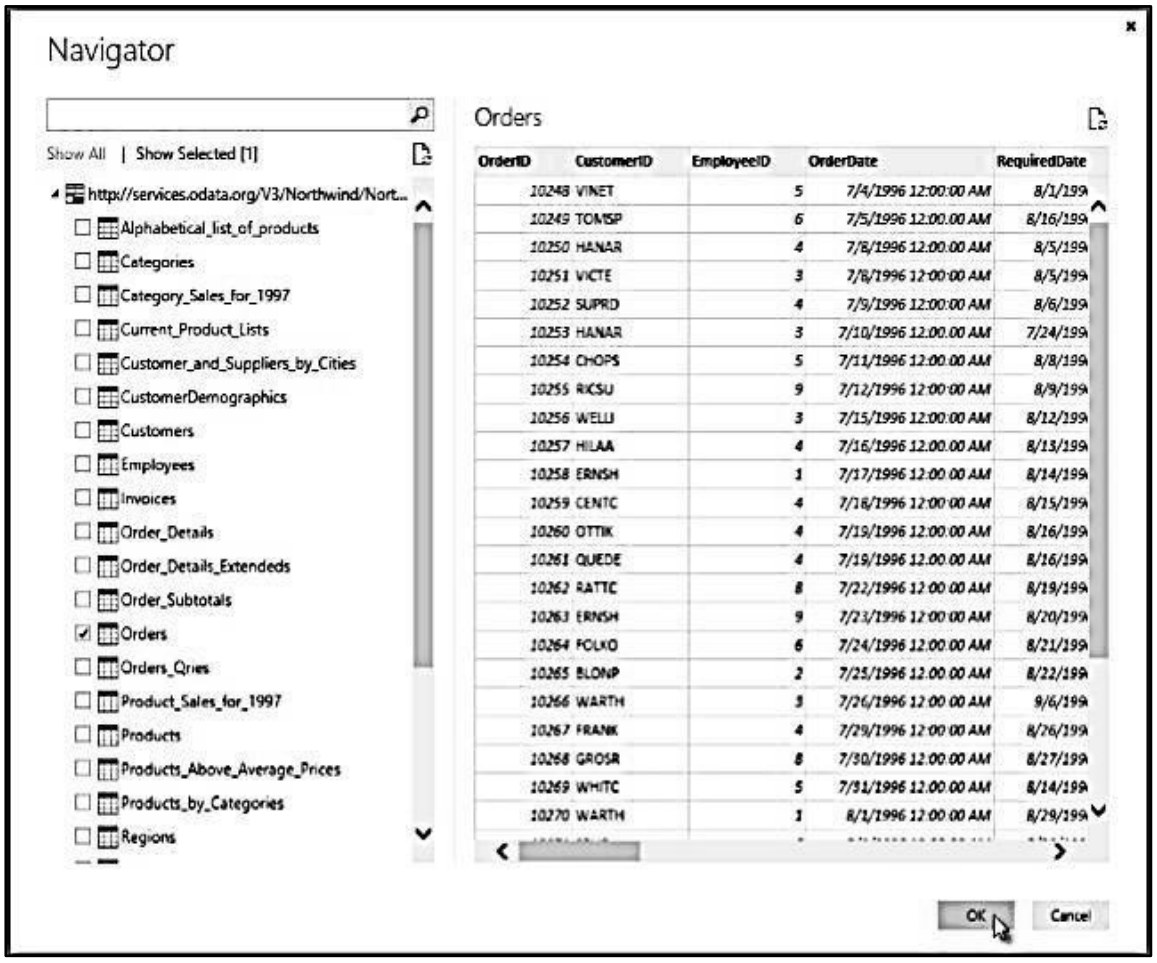

Importing Data from OData Feed :

In this task, you will 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.




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

# Practical 9:
# Create the Data staging area for the selected database using SQL.

SOLUTION :

Introduction :

Data staging refers to the process of temporarily storing raw data extracted from various sources in a designated "staging area" before further processing, while ETL (Extract, Transform, Load) is a complete data integration process that includes extracting data from sources, transforming it into a usable format, and loading it into a target system, with the staging area being a key part of the ETL process itself; essentially, data staging is a single step within the broader ETL workflow where raw data is held before transformation and loading.

A data staging area is defined as a storage area within a data warehouse where data is stored in its original format before being loaded into the warehouse. It allows for loose coupling of timing between data sources and the warehouse and provides an audit trail for analyzing data-related issues.

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.

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 into 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. (First name 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 in 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 :

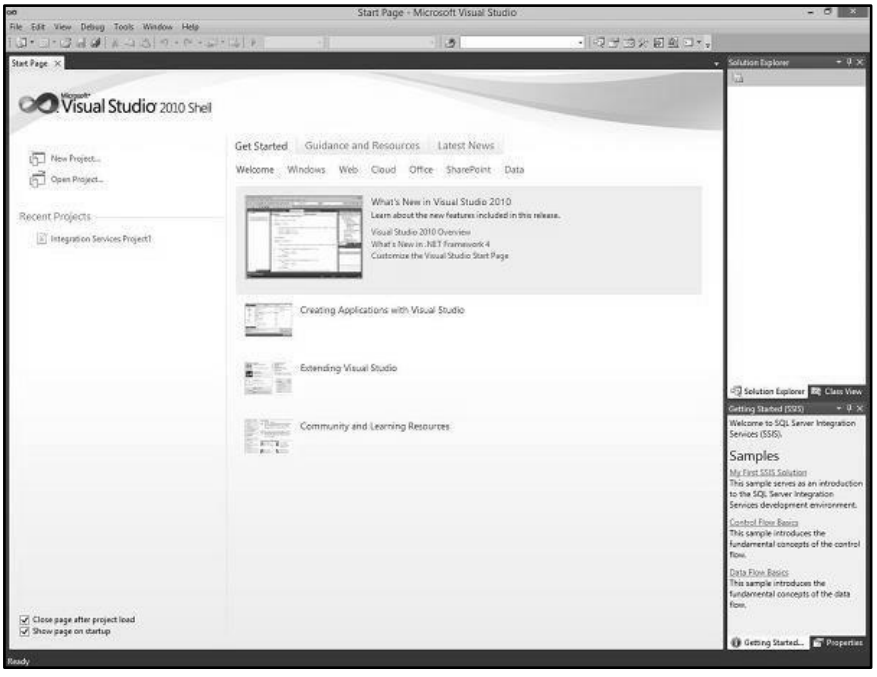
- Data loading phase loads the prepared data from staging tables to main tables.

ETL PROCESS IN SQL SERVER :

Following are the steps to open BIDS\SSDT.

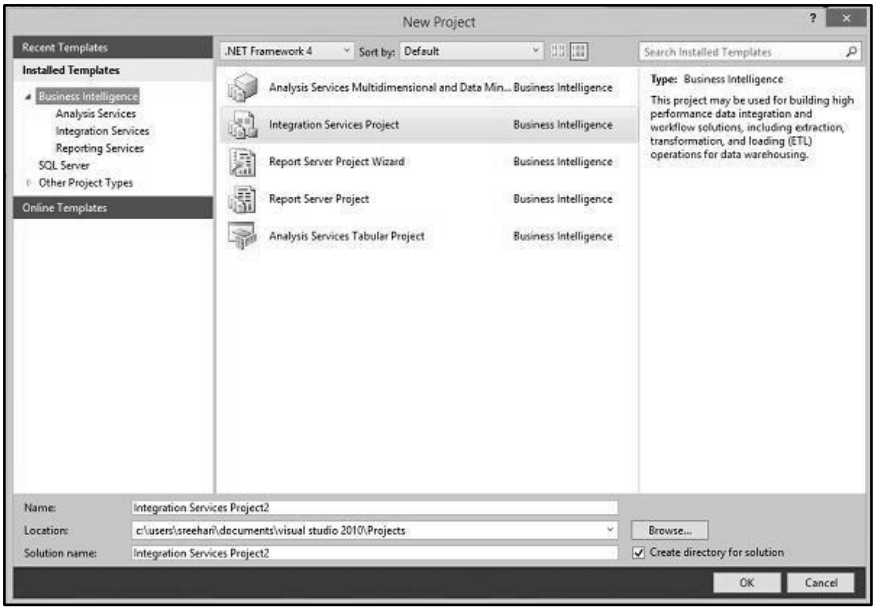
**Step (i) :** Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group.

The following screen appears.

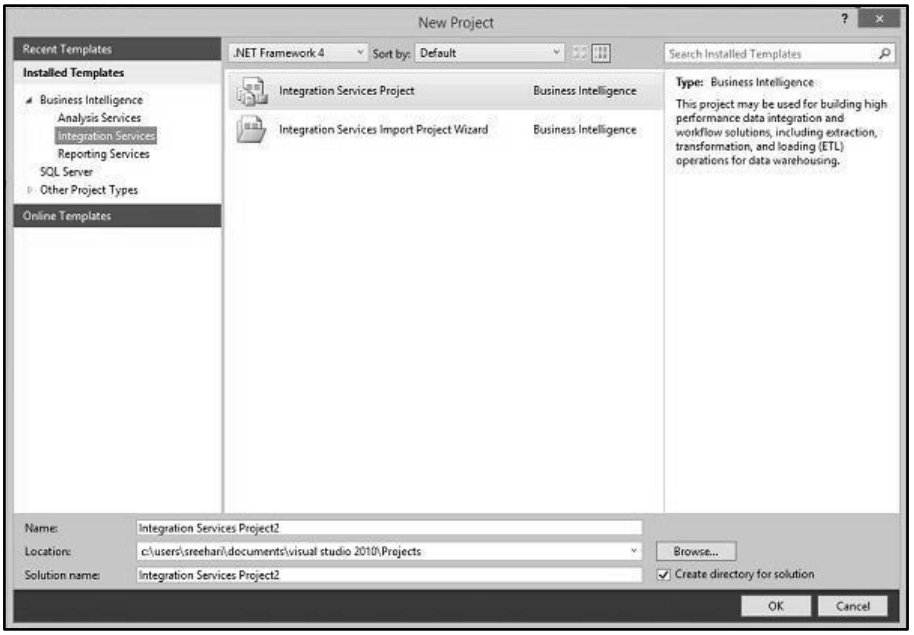


**Step (ii) :** The above screen shows SSDT has opened. Go to file at the top left corner in the above image and click New.

Select project and the following screen opens.




**Step (iii) :** Select Integration Services under Business Intelligence on the top left corner in the above screen to get the following screen.




**Step (iv) :** 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.

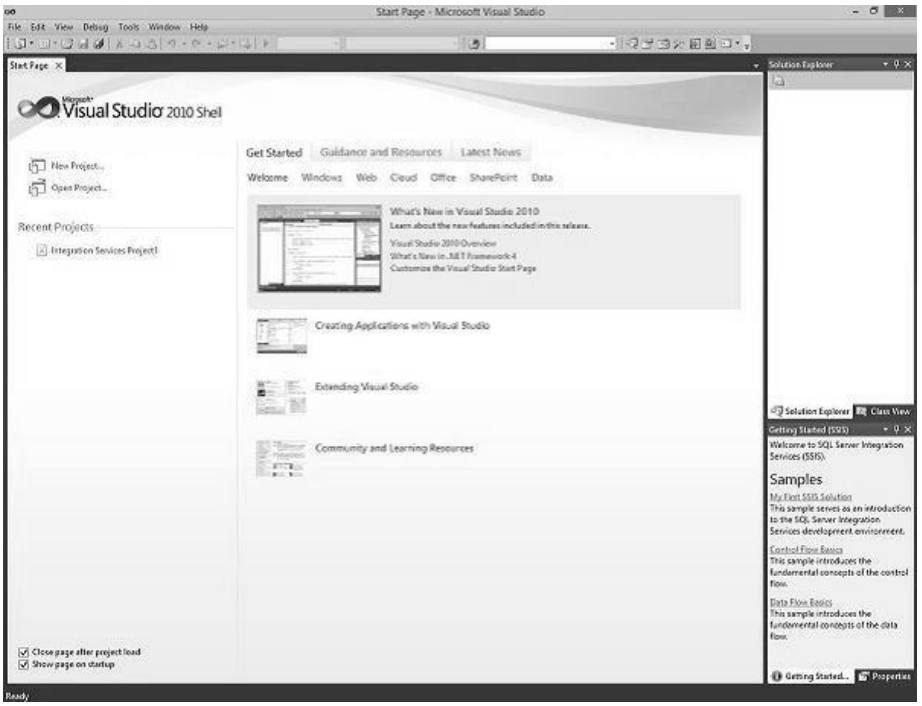
Models :

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.

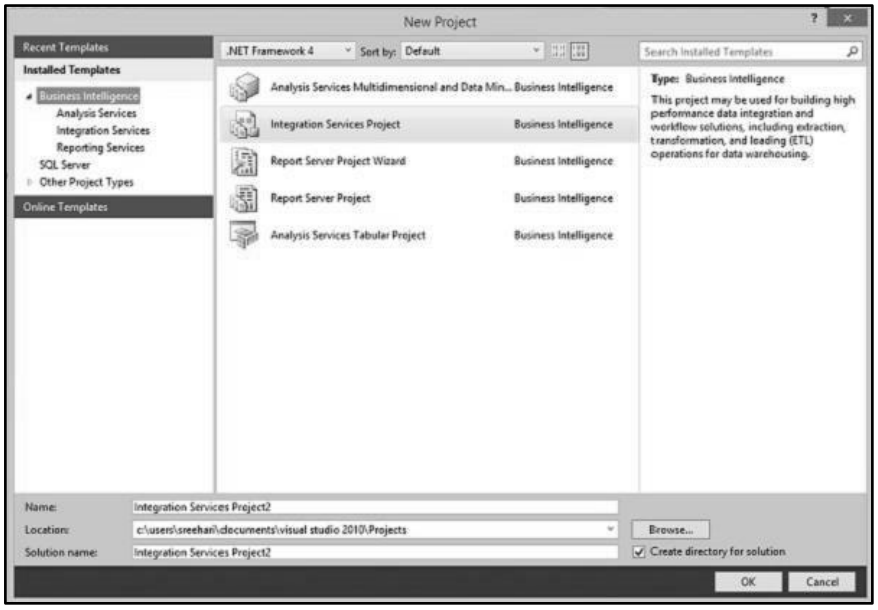
**Step (i) :** Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group.

The following screen will appear.



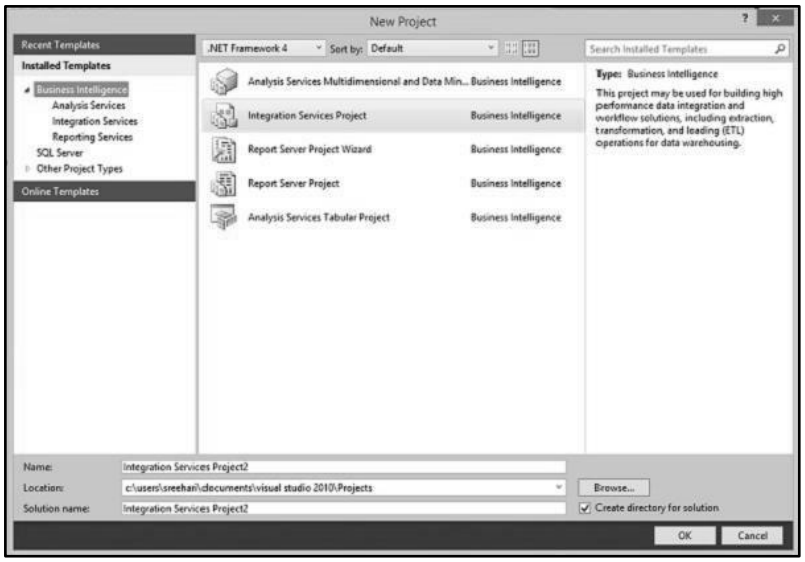

**Step (ii) :** The above screen shows SSDT has opened. Go to file on the top left corner in the above image and click New.

Select project and the following screen opens.



**Step (iii):** Select Analysis Services in the above screen under Business Intelligence as seen on the top left corner.

The following screen pops up.




**Step (iv) :** 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 POWERBI :

1. **Remove other columns to only display columns of interest.**

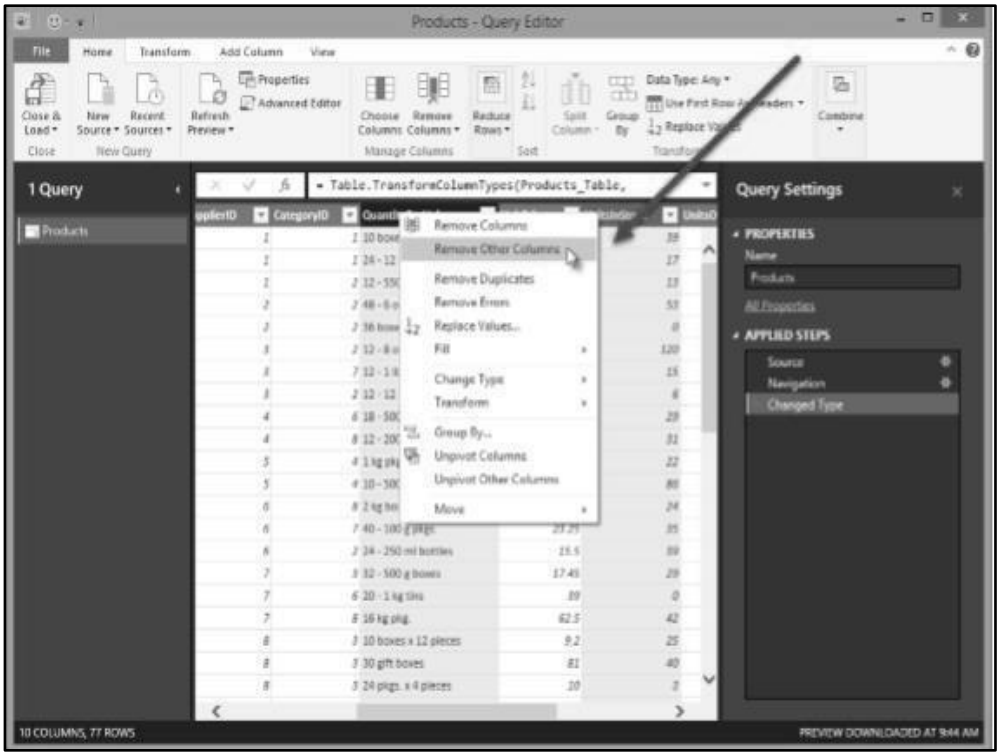
   In this step, you remove all columns except **ProductID**, **ProductName**, **UnitsInStock** and **QuantityPerUnit.**

   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.

   i) 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).

   ii) Select **Remove Columns > Remove** Other Columns from the ribbon, or right- click on a column header and click Remove Other Columns.

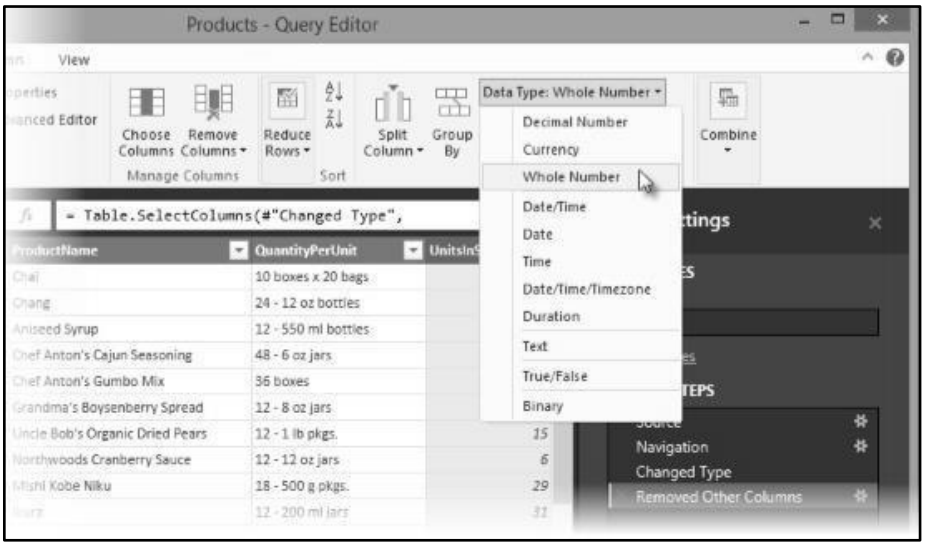

Products - Query Editor
File
Home
Transform
Add Column
View
Close & Load
New Source
Recent Sources
Refresh Preview
Properties
Advanced Editor
Choose Columns
Remove Columns
Reduce Rows
Split Column
Group By
Data Type: Any
Use First Row as Headers
Replace Values
Combine
Close
New Query
Manage Columns
Sort
Transform
1 Query
Products
= Table.TransformColumnTypes(Products_Table,
Remove Columns
Remove Other Columns
Remove Duplicates
Remove Errors
Replace Values...
Fill
Change Type
Transform
Group By...
Unpivot Columns
Unpivot Other Columns
Move
Query Settings
PROPERTIES
Name
Products
All Properties
APPLIED STEPS
Source
Navigation
Changed Type
10 COLUMNS, 77 ROWS
PREVIEW DOWNLOADED AT 9:44 AM

**2.** 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.

i) Select the **UnitsInStock** column.

i) Select the **Data Type drop-down button** in the **Home ribbon**.




iii) 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).

**3.** 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.

i) In the Query View, scroll to the Order_Details column.

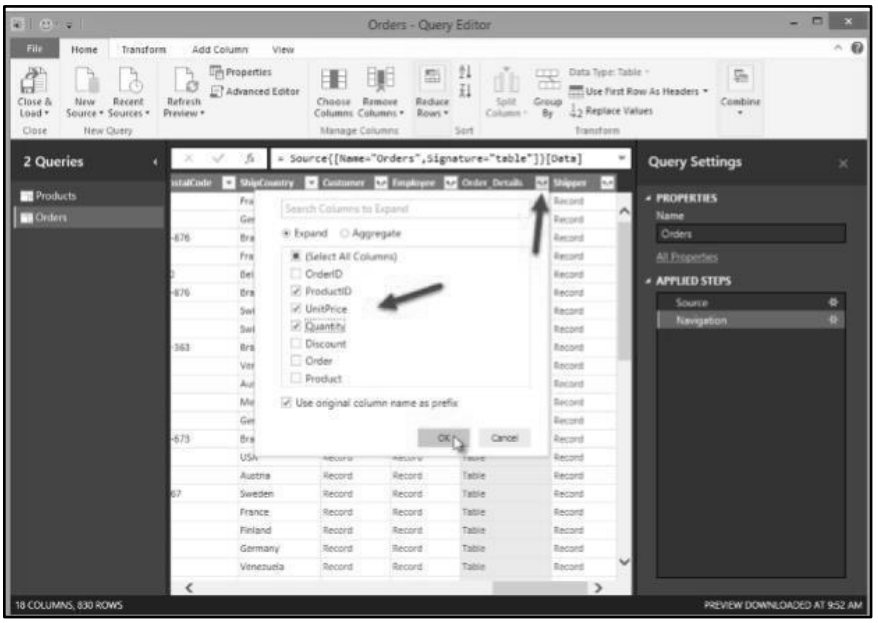
ii) In the Order_Details column, select the expand icon ( ).

iii) In the Expand drop-down :

a) Select (Select All Columns) to clear all columns.

b) Select ProductID, UnitPrice, and Quantity.

c) Click OK.



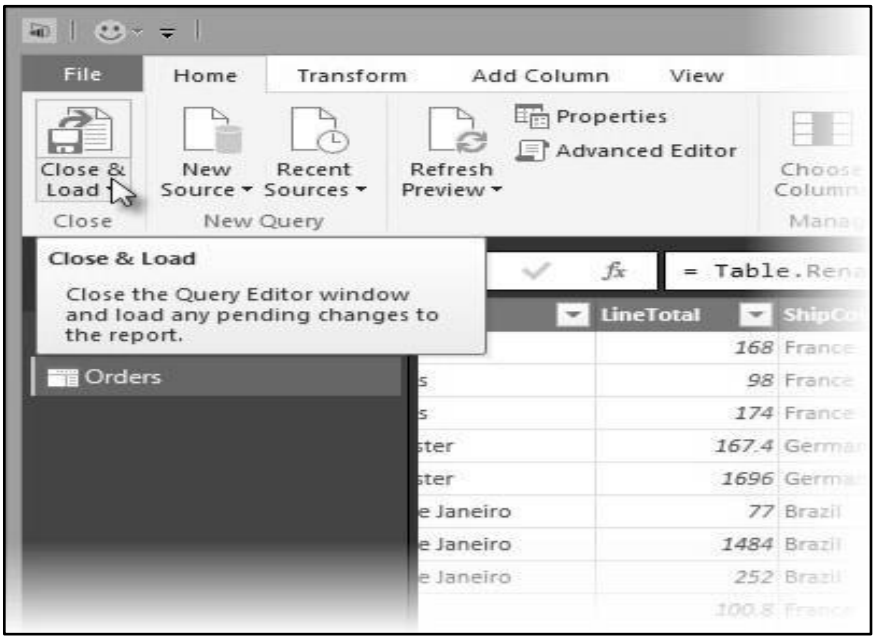

**4.** Calculate the line total for each Order_Details row.

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 :

i) In the Add Column ribbon tab, click Add Custom Column.




ii) In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity].

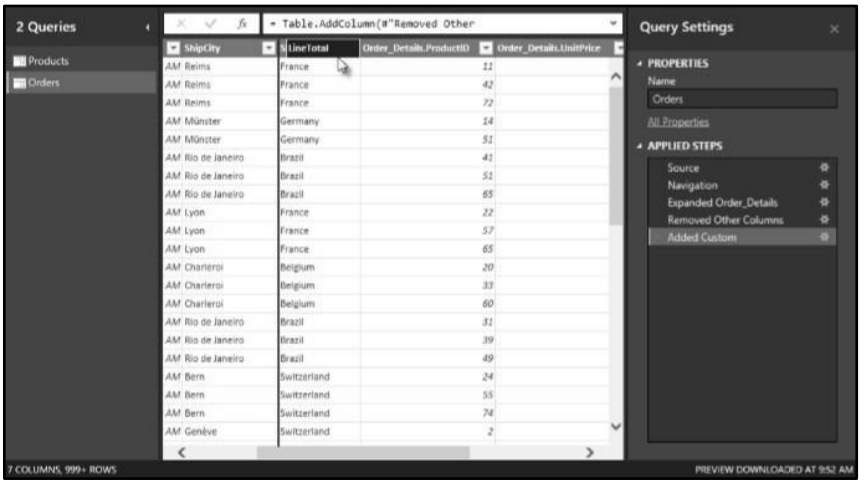
iii) In the New column name textbox, enter LineTotal.

iv) 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.

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




ii) Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice & 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.

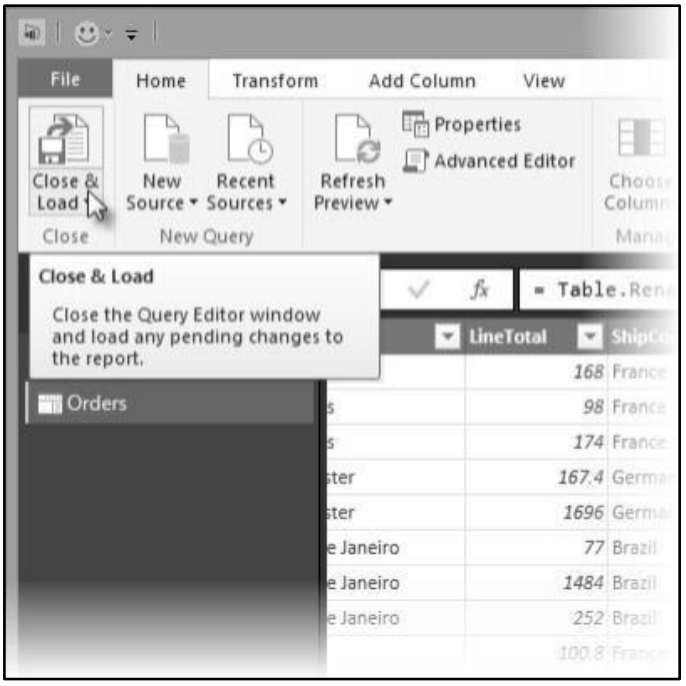
Power BI Desktop does not require you to combine queries to report on them. Instead, you can create Relationships between datasets. These relationships can be created on any column that is common to your datasets.

We have Orders and Products data that share a common 'ProductID' field, so we need to ensure there is a relationship between them in the model we are 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.

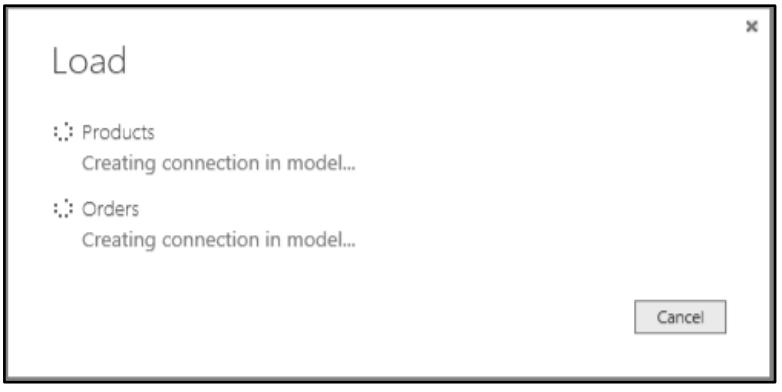
In this task, you confirm that a relationship is established in Power BI Desktop between the Products and Total Sales queries.

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

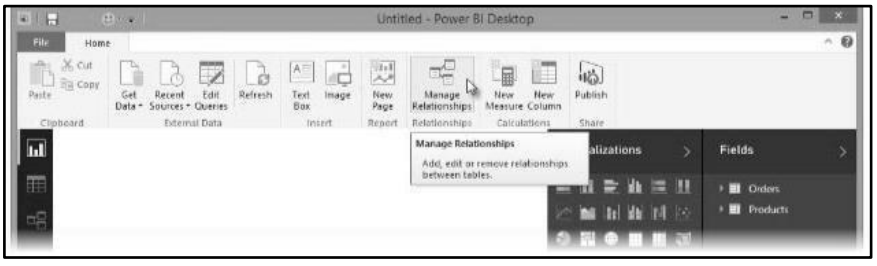
i) First, we need to load the model that we created in Query Editor into Power BI Desktop. From the Home ribbon of Query Editor, select Close and Load.



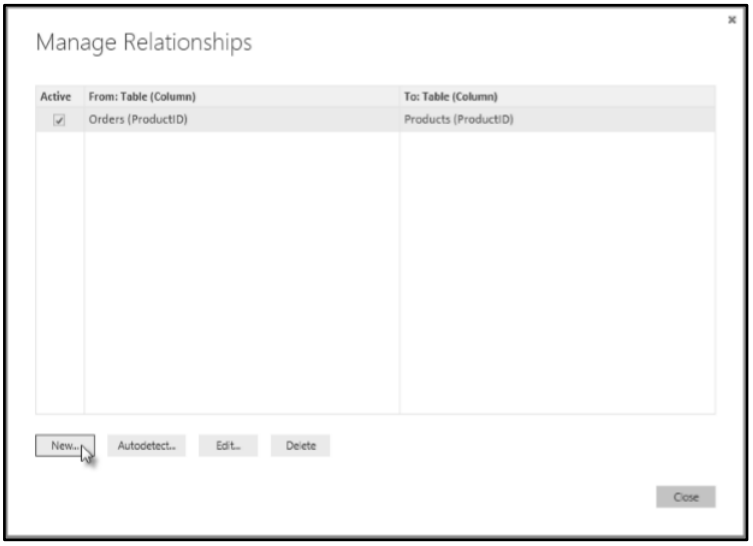

ii) Power BI Desktop loads the data from the two queries.



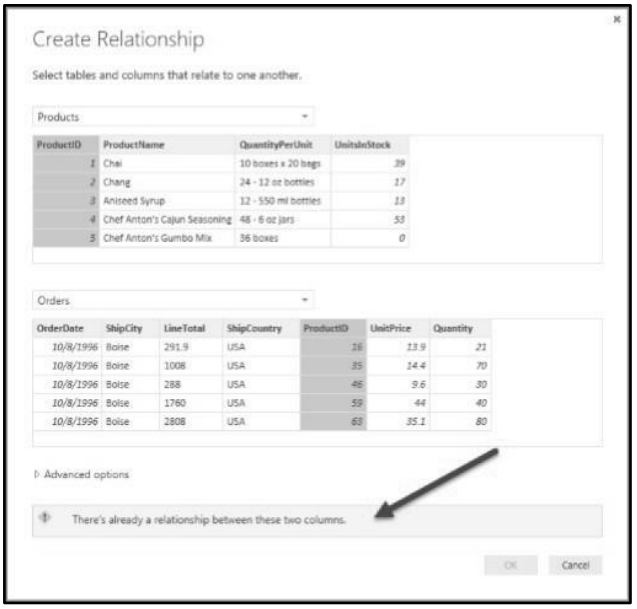

iii) Once the data is loaded, select the Manage Relationships button Home ribbon.



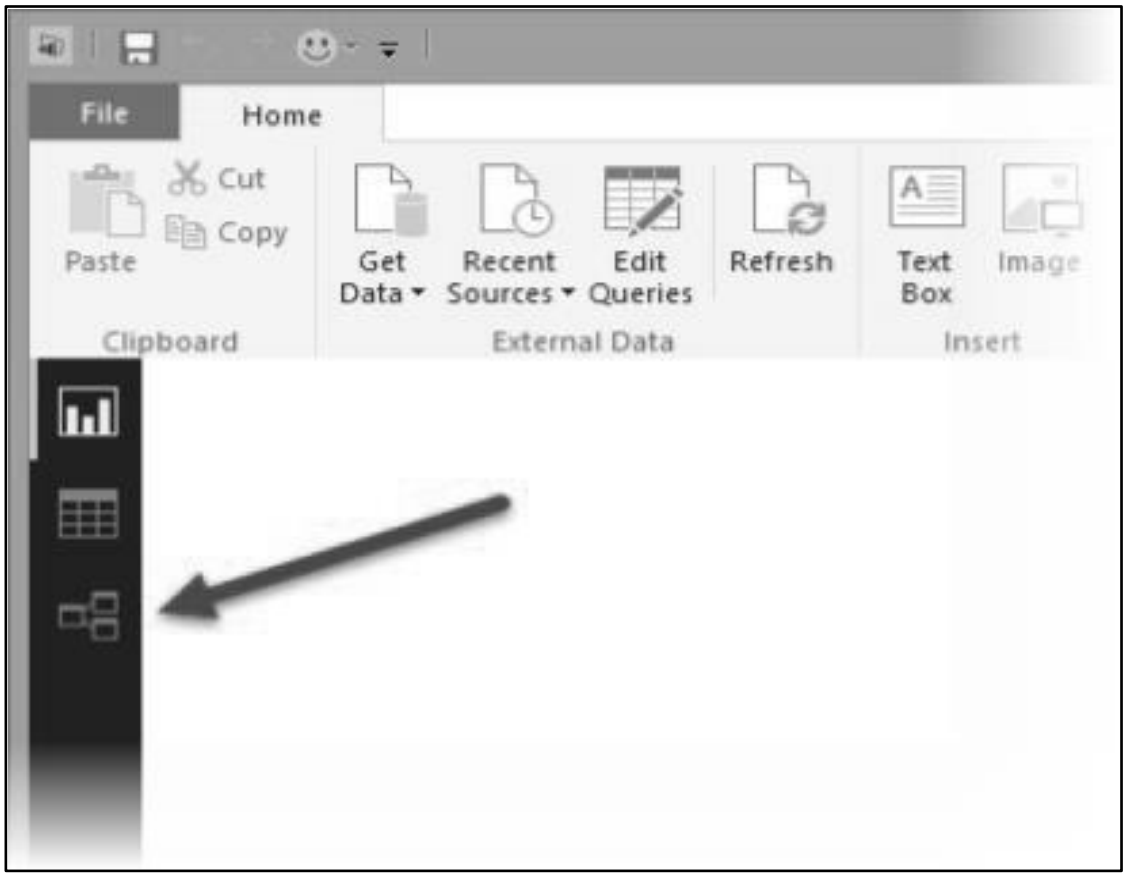
iv) Select the New… button.




v) 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.




vi) Select Cancel & then select Relationship view in Power BI Desktop.


File
Home
Cut
Copy
Paste
Get Data
Recent Sources
Edit Queries
Refresh
Text Box
Image
Clipboard
External Data
Insert

**Step 2 :** i) We see the following, which visualizes the relationship between the queries.


ii) When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.


iii) No need to make any changes, so we will just select Cancel to close the Edit Relationship dialog.

DATA VISUALIZATION FROM ETL PROCESS WITH POWERBI :

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.

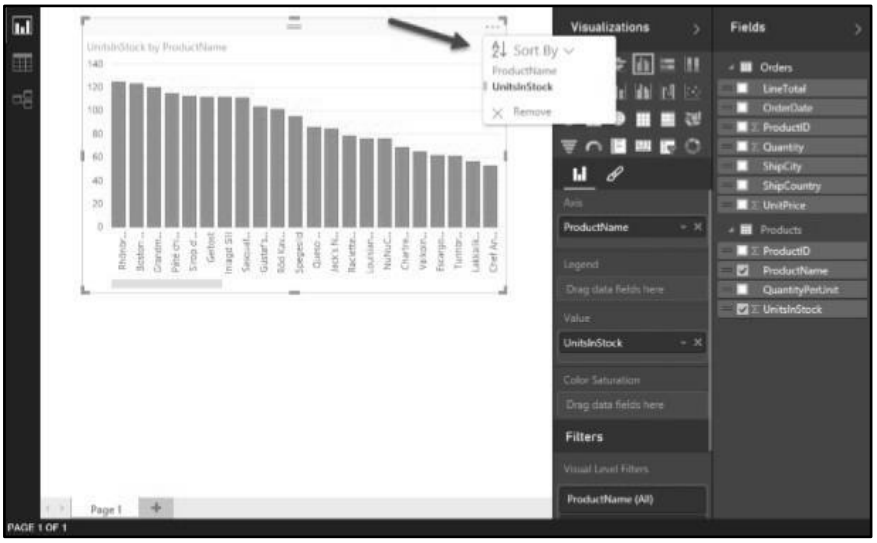
Step 1 : Create charts showing Units in Stock by Product and Total Sales by Year.

i) 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.

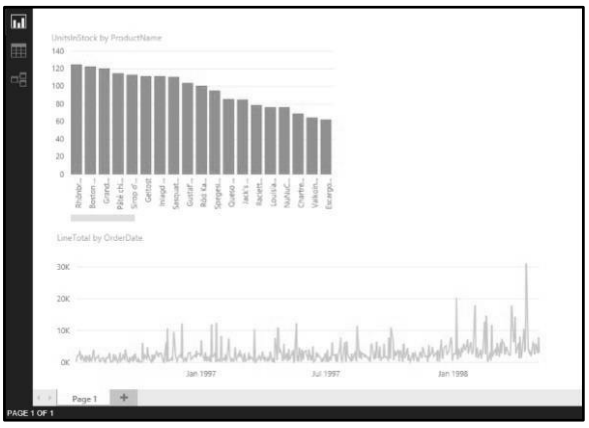
Next, drag ProductName to the Axis box, found in the bottom half of the Visualizations pane.

Then, select Sort By > UnitsInStock using the skittles in the top right corner of the visualization.




ii) 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.



iii) 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.

i) 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 10:

## Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model.

SOLUTION :

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).

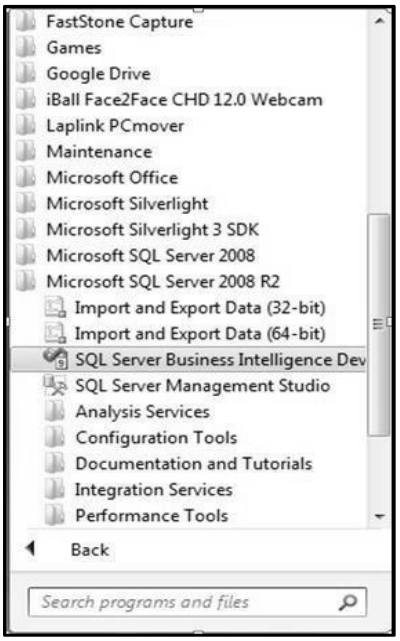
1. Open SQL Server Management Studio 2008.
2. Connect Database Engine.
3. Open **New Query** editor.
4. Copy paste Scripts given below in various steps in new query editor window one by one.
5. To run the given SQL Script, press **F5.**
6. It will create and populate "Sales_DW" database on your SQL Server.

DEVELOPING AN OLAP CUBE :

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

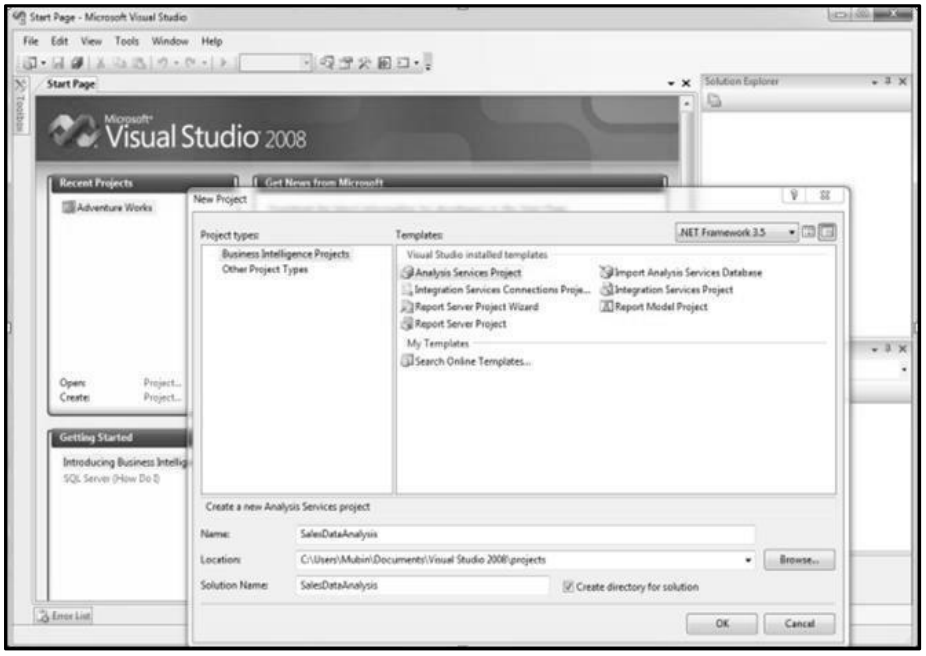
Step 1 : Start BIDS Environment.

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 : 3.1 Creating New Data Source.

i) In Solution Explorer, Right click on **Data Source** → Click **New Data Source Step 3 : Creating New Data Source.**

ii) In Solution Explorer, Right click on **Data Source** → Click **New Data Source.**

iii) Click on **Next.**


iv) Click on New **Button.**


Step 3 : 3.2 Creating New Connection.

i) Specify Your SQL **Server Name** where your Data Warehouse was created.

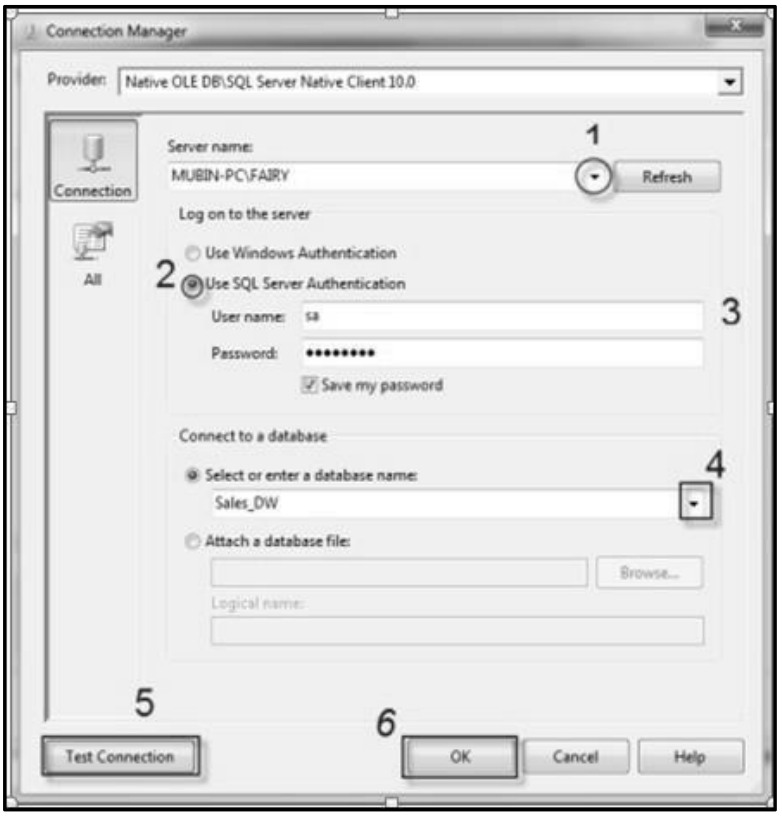
ii) Select Radio Button according to your **SQL Server Authenti- cation** mode.

iii) Specify your **Credentials** using which you can connect to your SQL Server.

iv) Select database Sales_DW.

v) Click on **Test Connection** and verify for its success.

vi) Click **OK**.




vii) Select Connection created in **Data Connections** → Click **Next.**

viii) Select Option **Inherit.**


ix) Assign Data Source **Name** → Click **Finish.**


Step 4 : Creating New Data Source View.

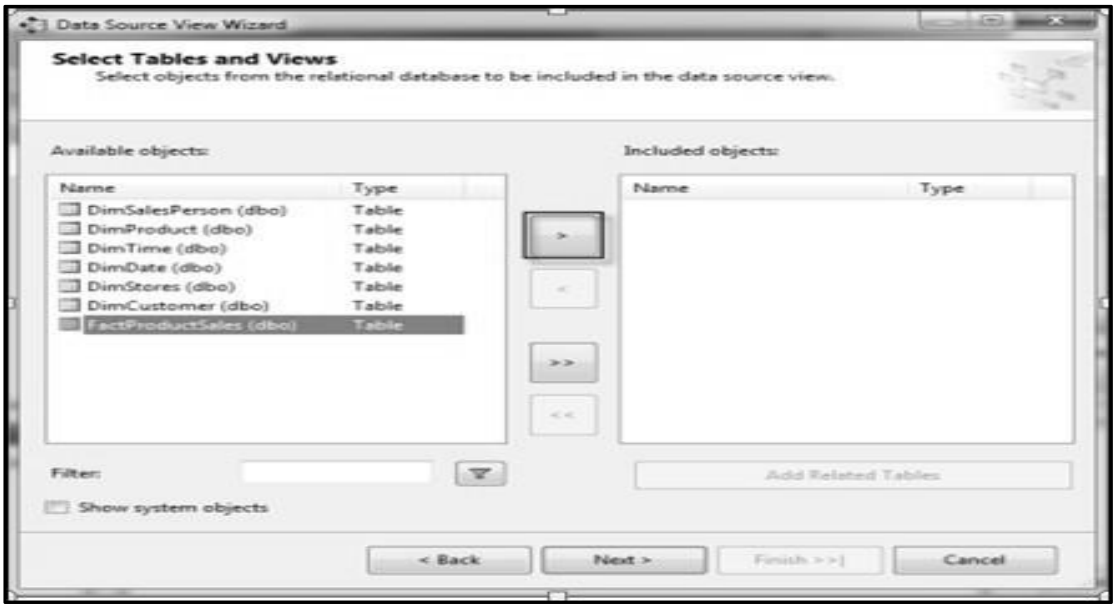
i) In the Solution Explorer, Right Click on **Data Source View** → Click on
New Data Source View.


ii) Click **Next.**


iii) Select **Relational Data Source** we have created previously (Sales_DW)
→ Click **Next.**


Data Source View Wizard
Select a Data Source
Select an existing relational data source or create a new one.
Relational data sources:
Sales DW
Data source properties:
Property
Value
Data Source
MUBIN-PC\FAIRY
Initial Catalog
Sales_DW
Persist Securi...
True
Provider
SQLNCLI10.1
User ID
sa
New Data Source...
Advanced...
< Back
Next >
Finish >>|
Cancel

iv) First move your **Fact Table** to the right side to include in object list.



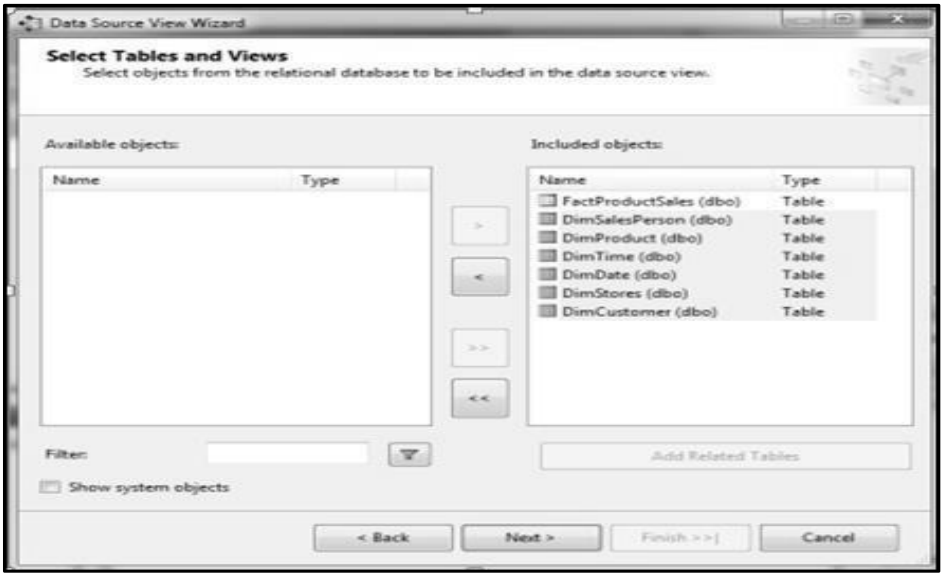

v) Select FactProductSales Table → Click on Arrow Button to move the selected object to Right Pane.

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

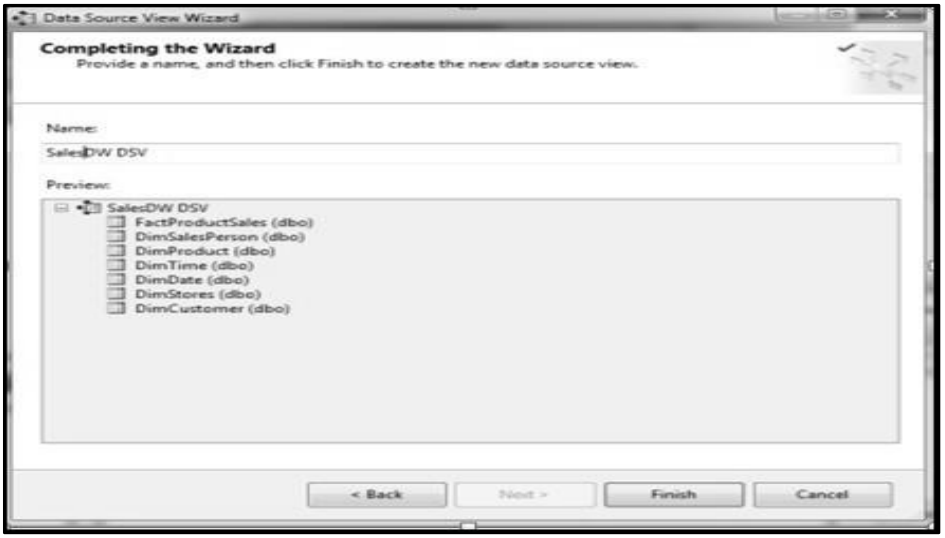
Select **Fact Table** in Right Pane (Fact product Sales) → Click On **Add Related Tables**


vii) It will add all associated dimensions to your Fact table as per relationship specified in your SQL DW (Sales_DW).

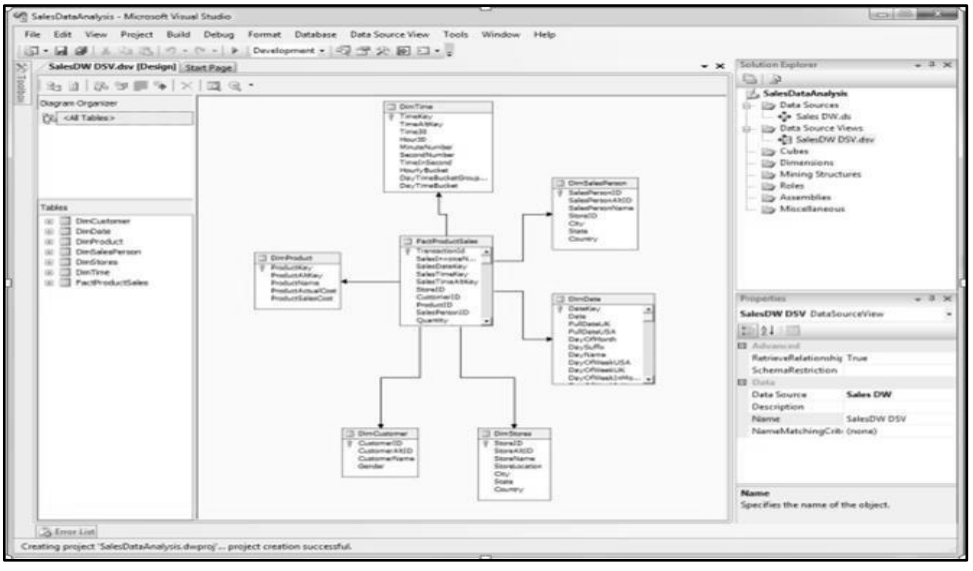
viii) Click **Next**.




ix) Assign **Name (SalesDW DSV)** → Click **Finish.**




x) Now Data Source View is ready to use.



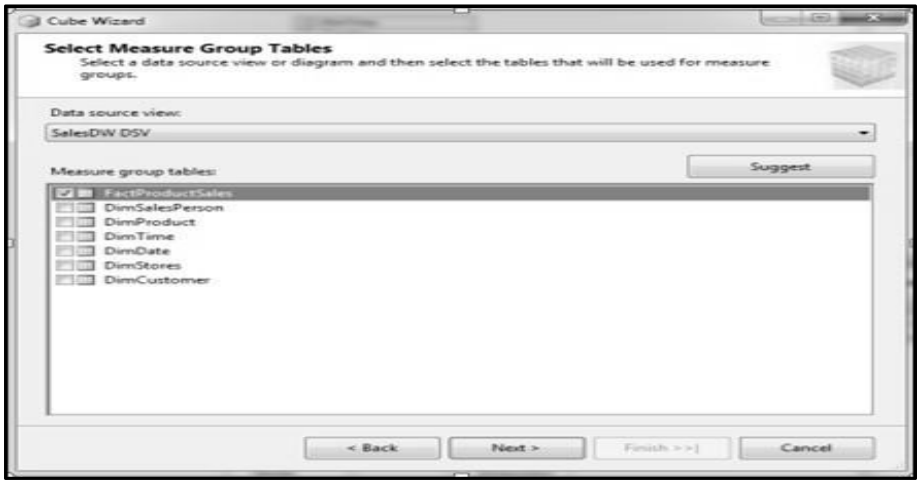
Step 5 : Creating New Cube.

i) In Solution Explorer → Right Click on **Cube** → Click **New Cube.**

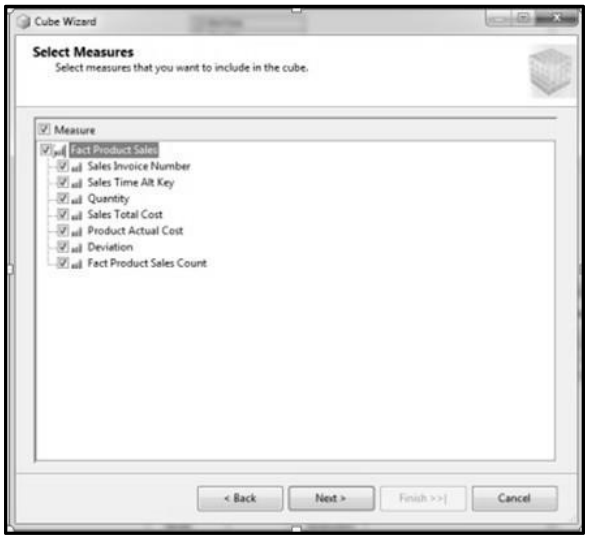
ii) Click **Next.**


iii) Select Option **Use existing Tables** → Click **Next.**


iv) Select Fact Table Name from **Measure Group Tables (FactProductSales)** → Click Next.



v) Choose **Measures** from the List which you want to place in your Cube
→ Click **Next.**

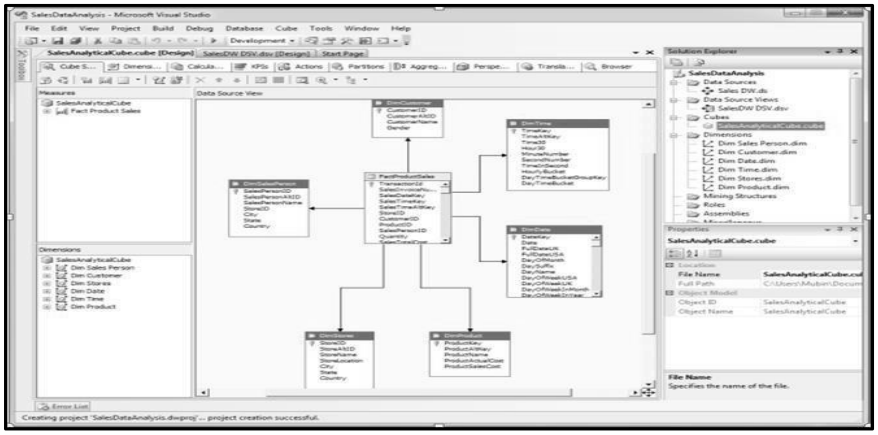



vi) Select All **Dimensions** here which are associated with your Fact Table
→ Click **Next.**


vii) Assign **Cube Name** (SalesAnalyticalCube) → Click **Finish.**

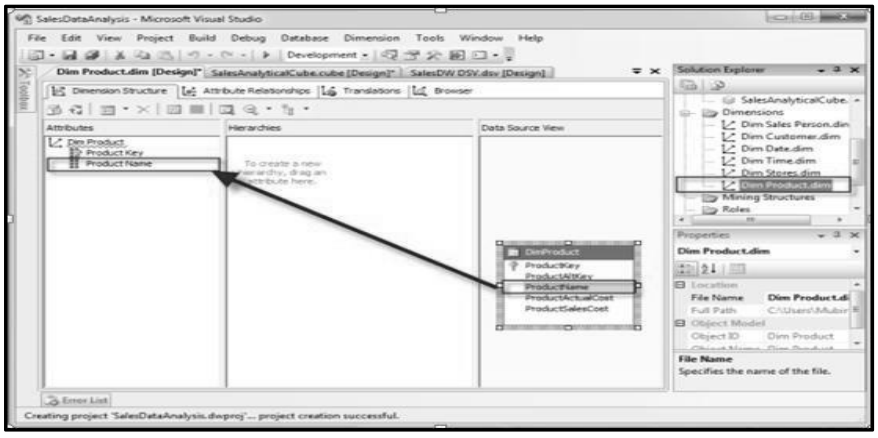

Cube Wizard
Completing the Wizard
Name the cube, review its structure, and then click Finish to save the cube.
Cube name:
SalesAnalyticalCube
Preview:
Measure groups
Fact Product Sales
Sales Invoice Number
Sales Time Alt Key
Quantity
Sales Total Cost
Product Actual Cost
Deviation
Fact Product Sales Count
Dimensions
Dim Sales Person
Dim Customer
Dim Date
Dim Time
Dim Stores
Dim Product
< Back
Next >
Finish
Cancel

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



Step 6 : Dimension Modification.

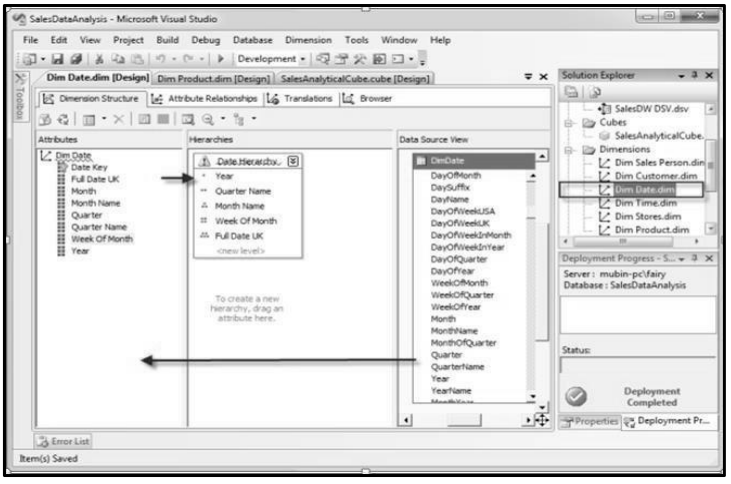
In Solution Explorer, double click on dimension **Dim Product** → Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side.



Step 7 : Creating Attribute Hierarchy In Date Dimension.

i) 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.

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

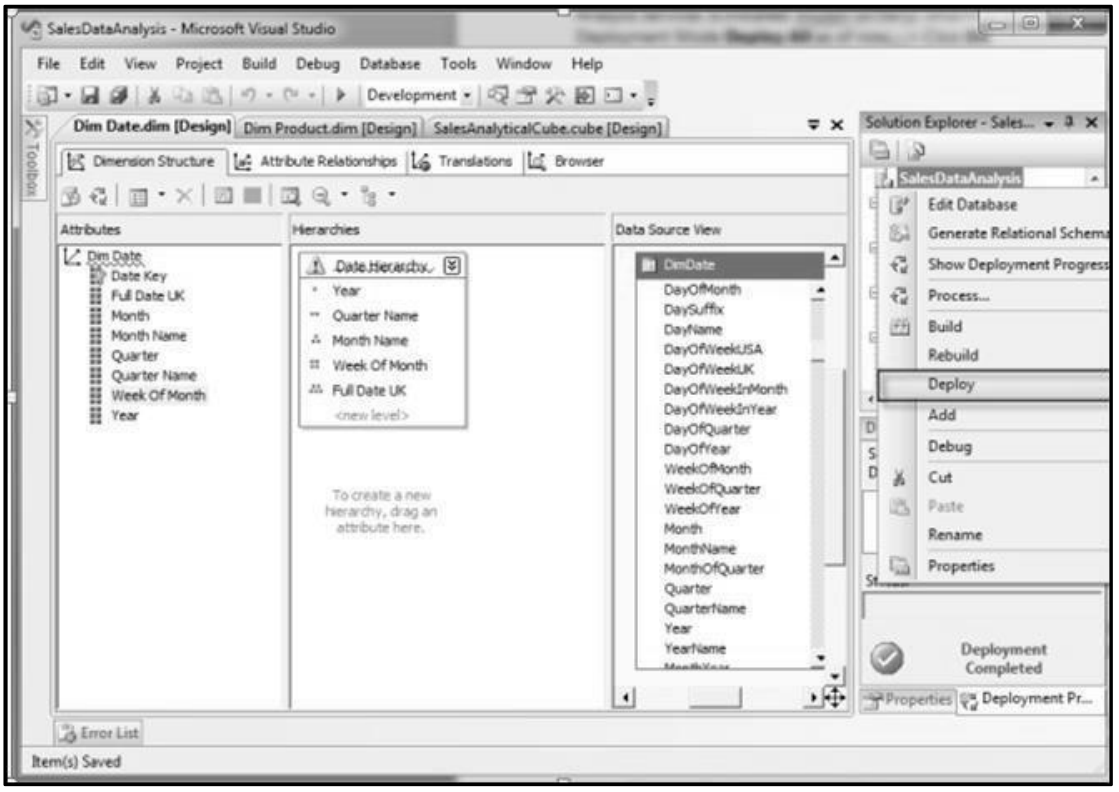


Step 8 : Deploy the Cube.

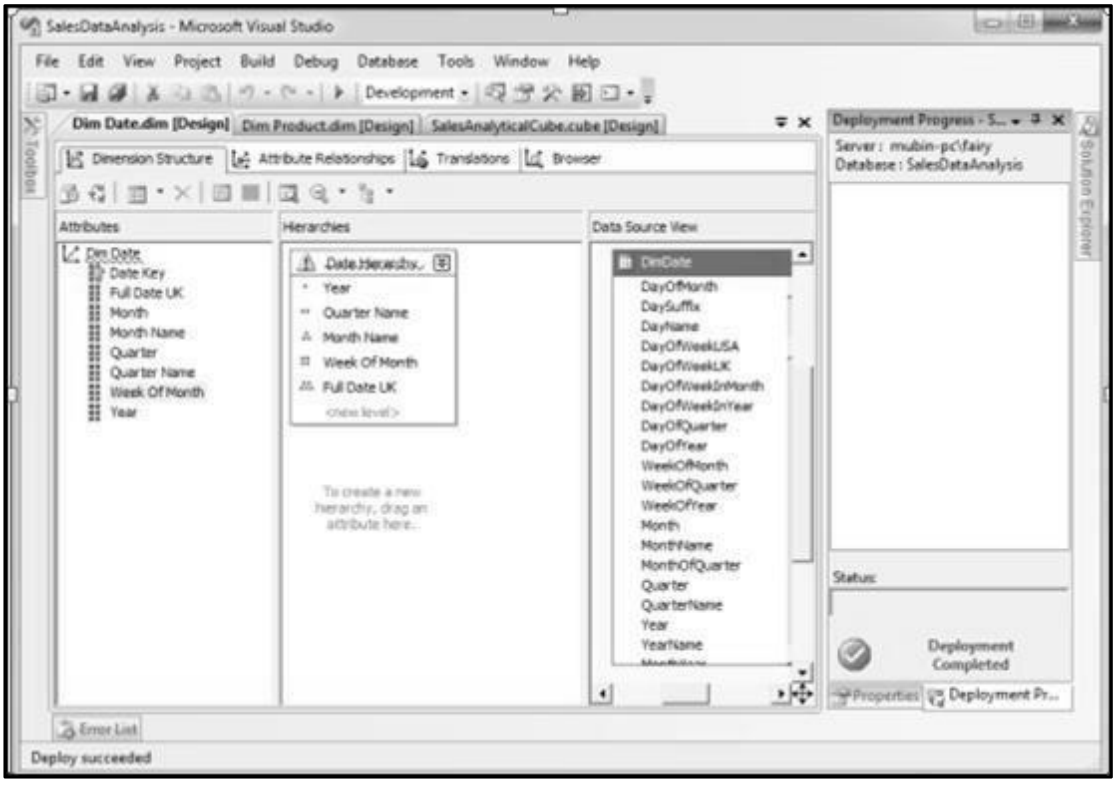
i) In Solution Explorer, right click on Project Name (SalesDataAnalysis) →
Click Properties**.**


ii) 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.**

iii) In Solution Explorer, right click on **Project Name** (SalesDataAnalysis)
→ Click **Deploy.**

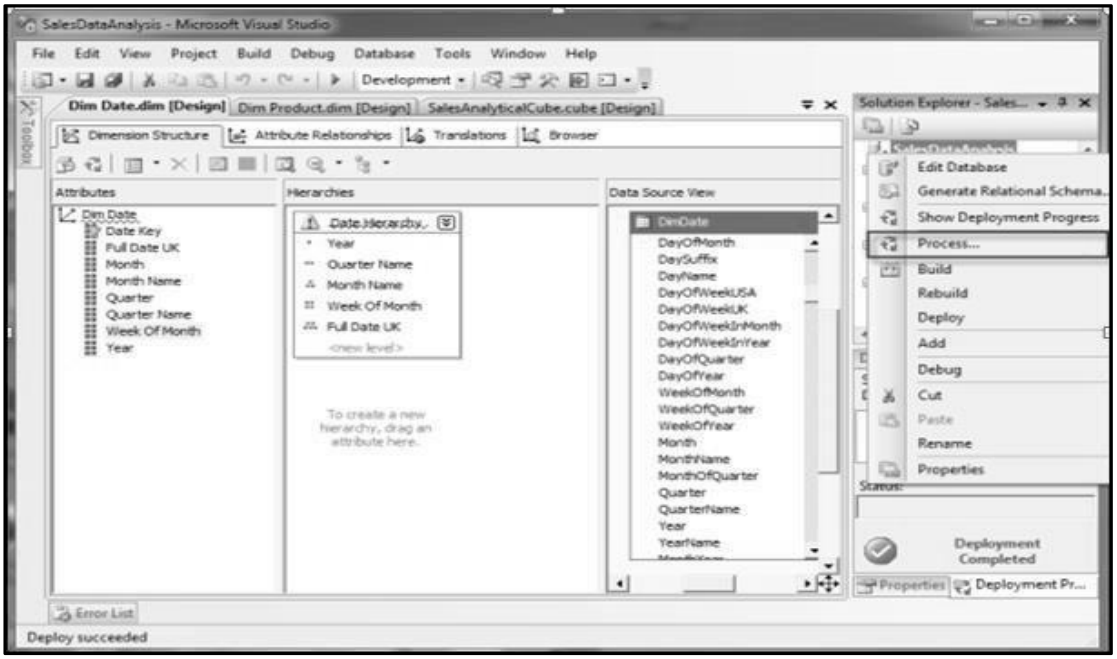



iv) Once Deployment will finish, you can see the message **Deployment Completed** in deployment Properties.

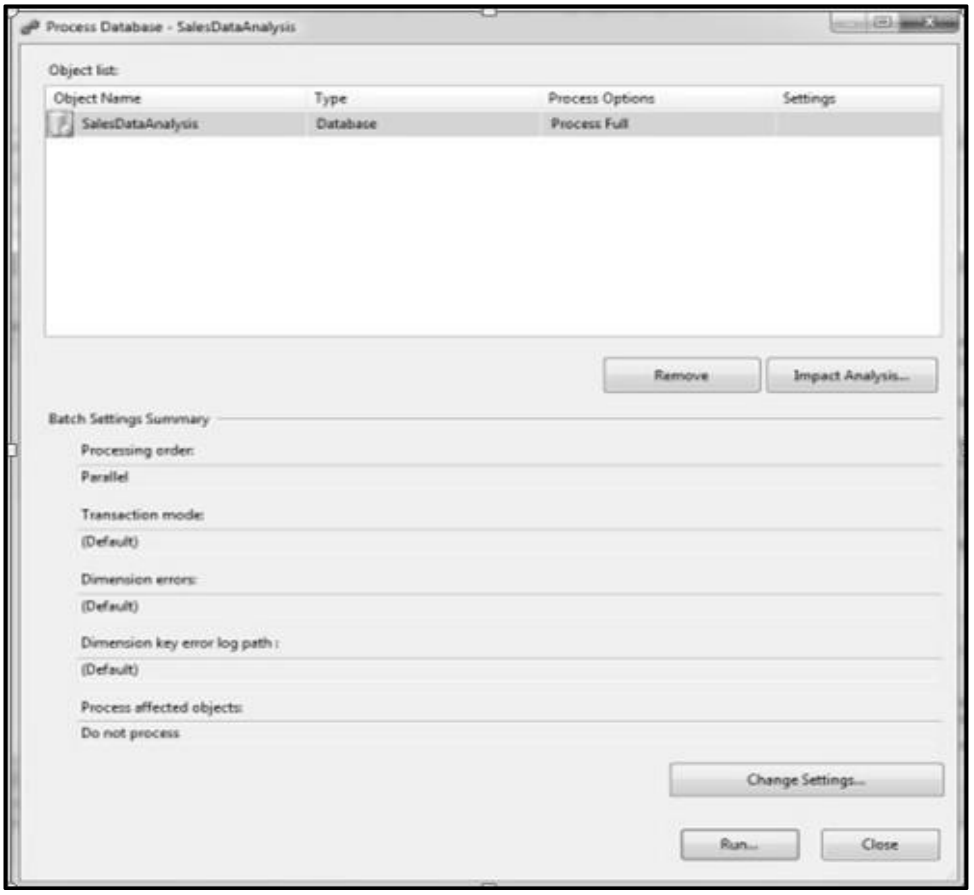


Step 9 : Process the Cube.

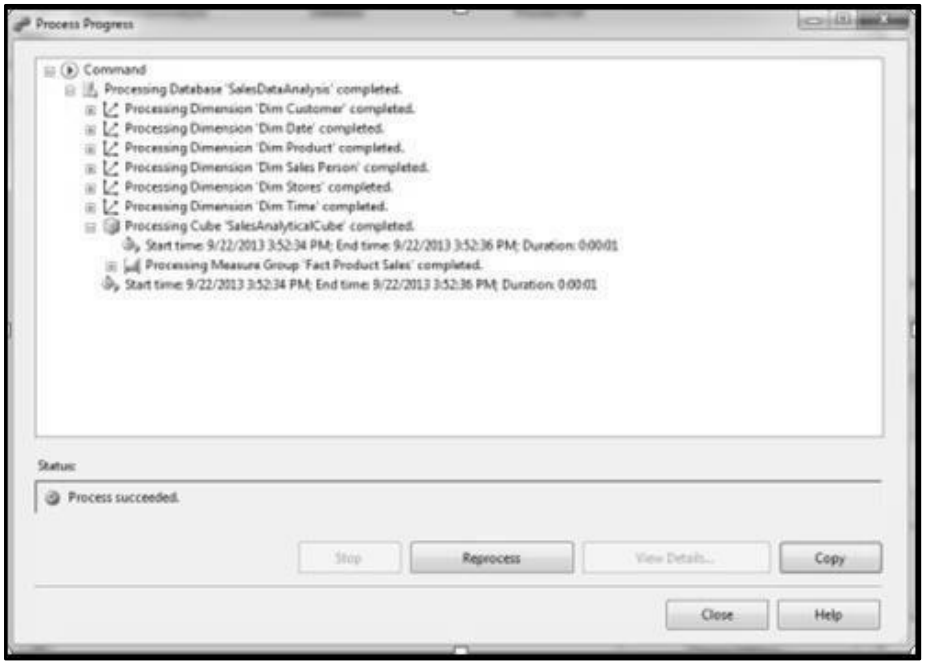
i) In Solution Explorer, right click on Project Name (SalesDataAnalysis) → Click **Process**




ii) Click on **Run** button to process the Cube.

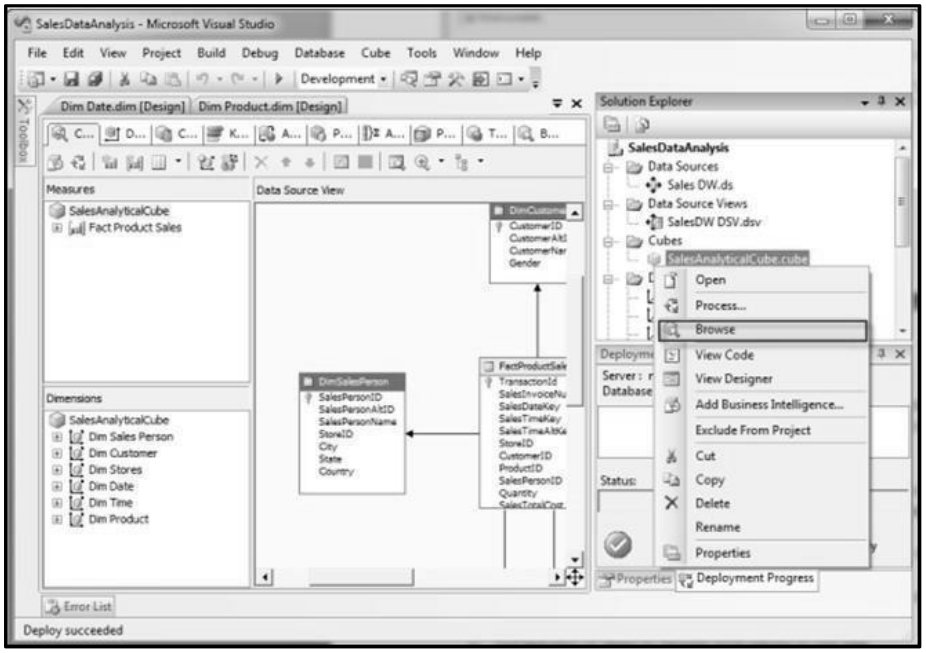


iii) Once processing is complete, you can see **Status** as **Process Succeeded** → Click **Close** to close both the open windows for processing one after the other.




Step 10 : Browse the Cube for Analysis.

i) In Solution Explorer, right click on Cube Name (SalesDataAnalysisCube) → Click **Browse.**



ii) Drag and drop measures into Detail fields and drag and Drop Dimension Attributes in Row Field or Column fields.

iii) Now to **Browse Our Cube :**

a) Product Name Drag and Drop into Column.

b) Full Date UK Drag and Drop into Row Field.

c) FactProductSalesCount Drop this measure in Detail area.