Pratical -1(A)

Aim :- Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

STEPS:-

Step 1: Open Microsoft Excel

Step 2: Connect to Data Warehouse

Go to the Data tab on the Ribbon.

Click Get Data > From Database (Choose the appropriate option based on your database type).

From SQL Server Database (for Microsoft SQL Server)

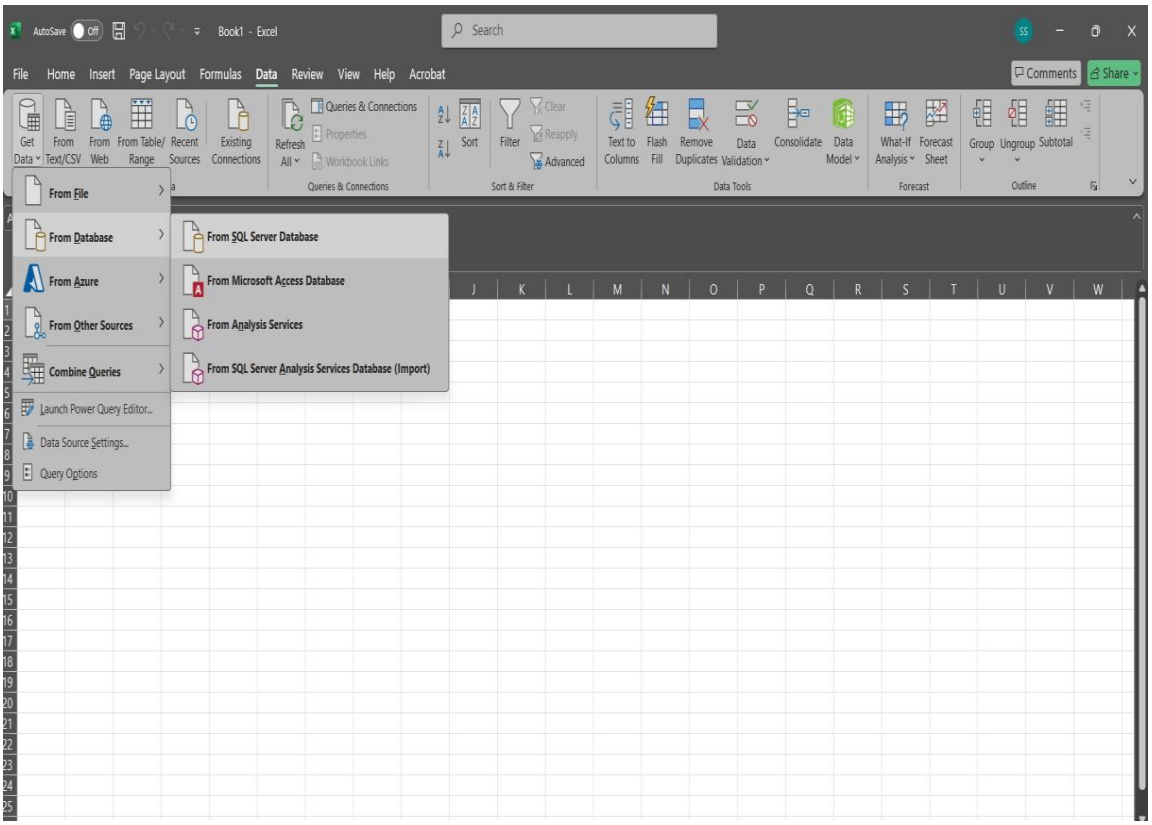
From Oracle Database (for Oracle)

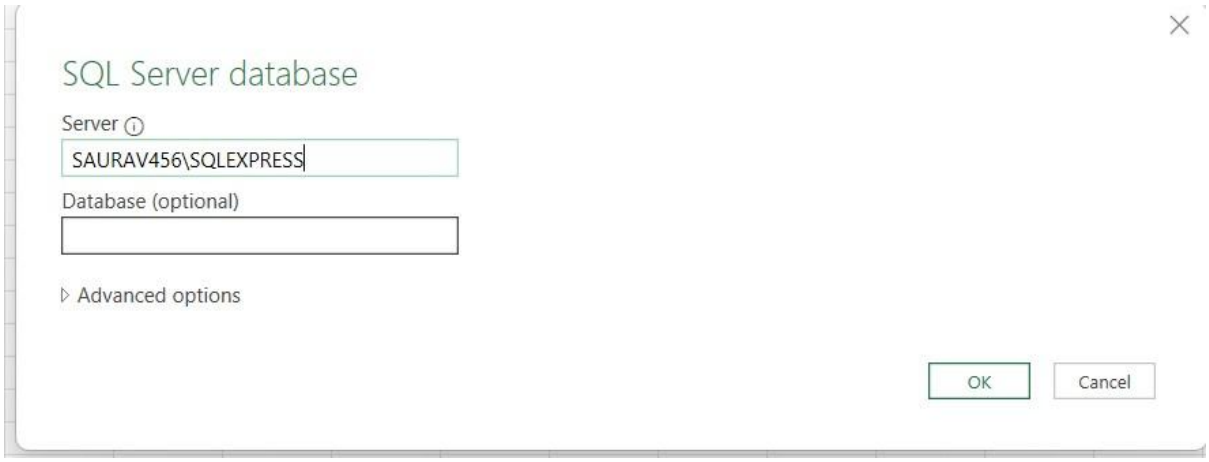
From MySQL Database (for MySQL)

From Other Sources (for other databases)

Enter the Server Name and Database Name when prompted.

Click OK and then choose the required table or query.





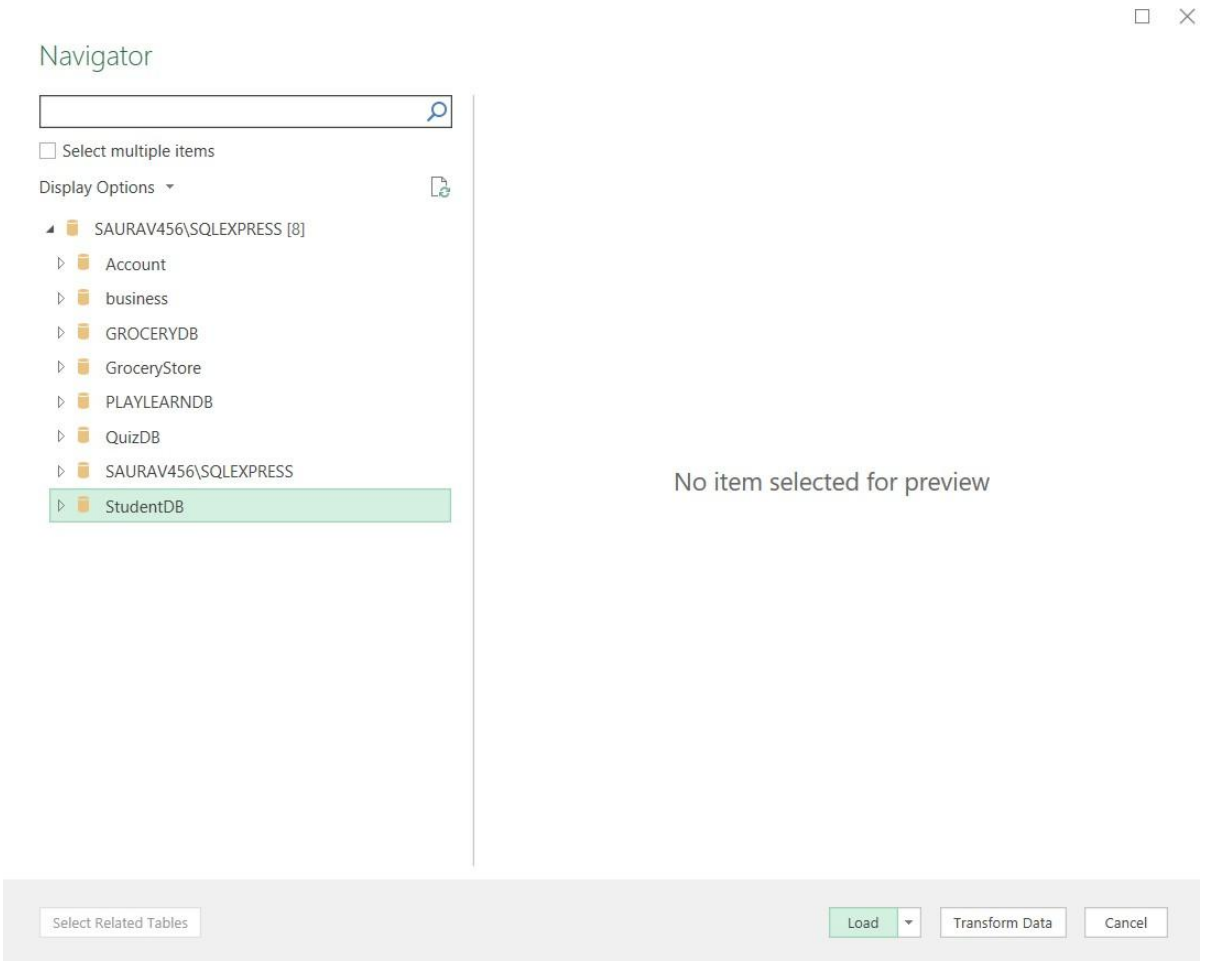
Step 3: Load Data into Excel

After selecting the table, you can either:

Click **Load** to import the data directly into an Excel sheet.

Click **Transform Data** to clean or filter data before importing (optional).

Wait for the data to load into an Excel worksheet.



Step 4: Create a Pivot Table

Click anywhere in the imported data.

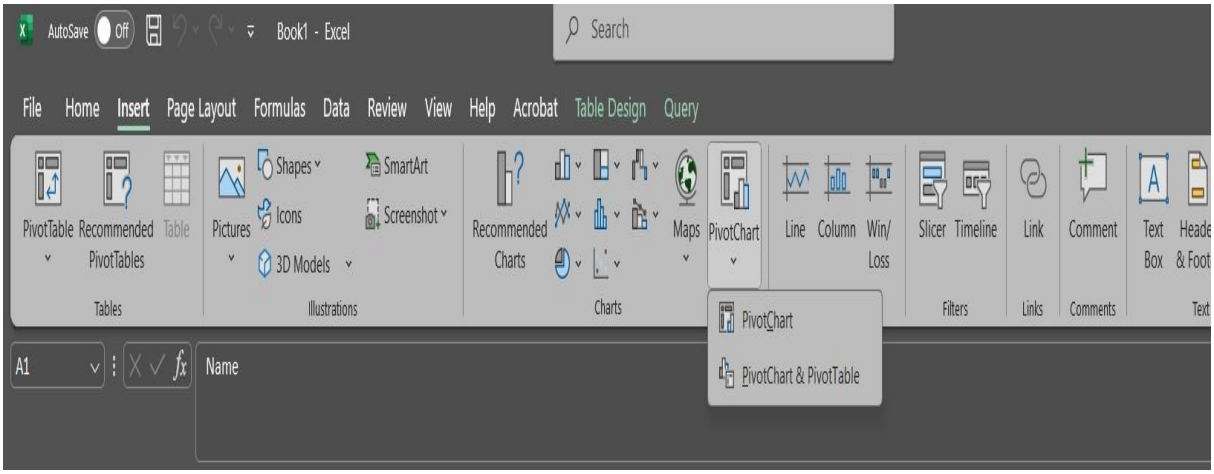
Go to the **Insert** tab and select **PivotTable**.

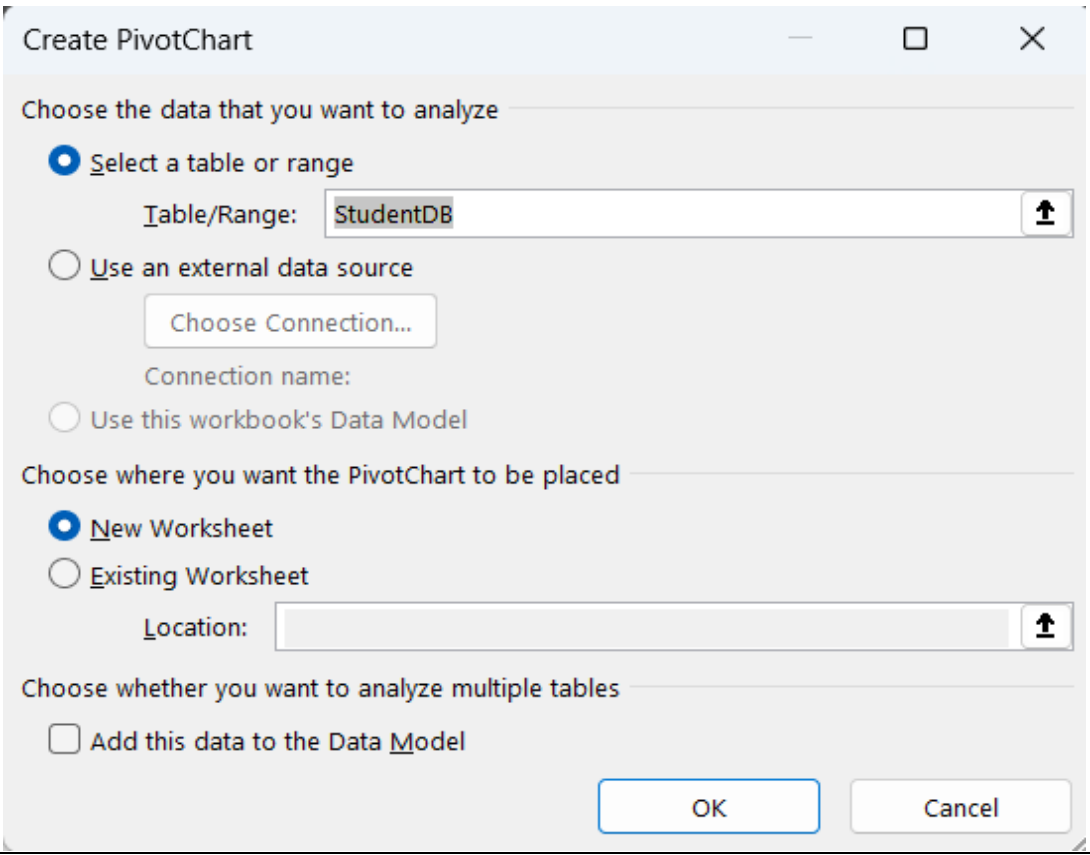
In the dialog box:

Choose **Select a table or range** (if data is already in the worksheet).

Choose **Use an external data source** (if connecting directly to the database).

Click **OK** to create the Pivot Table in a new or existing worksheet.





Step 5: Customize the Pivot Table

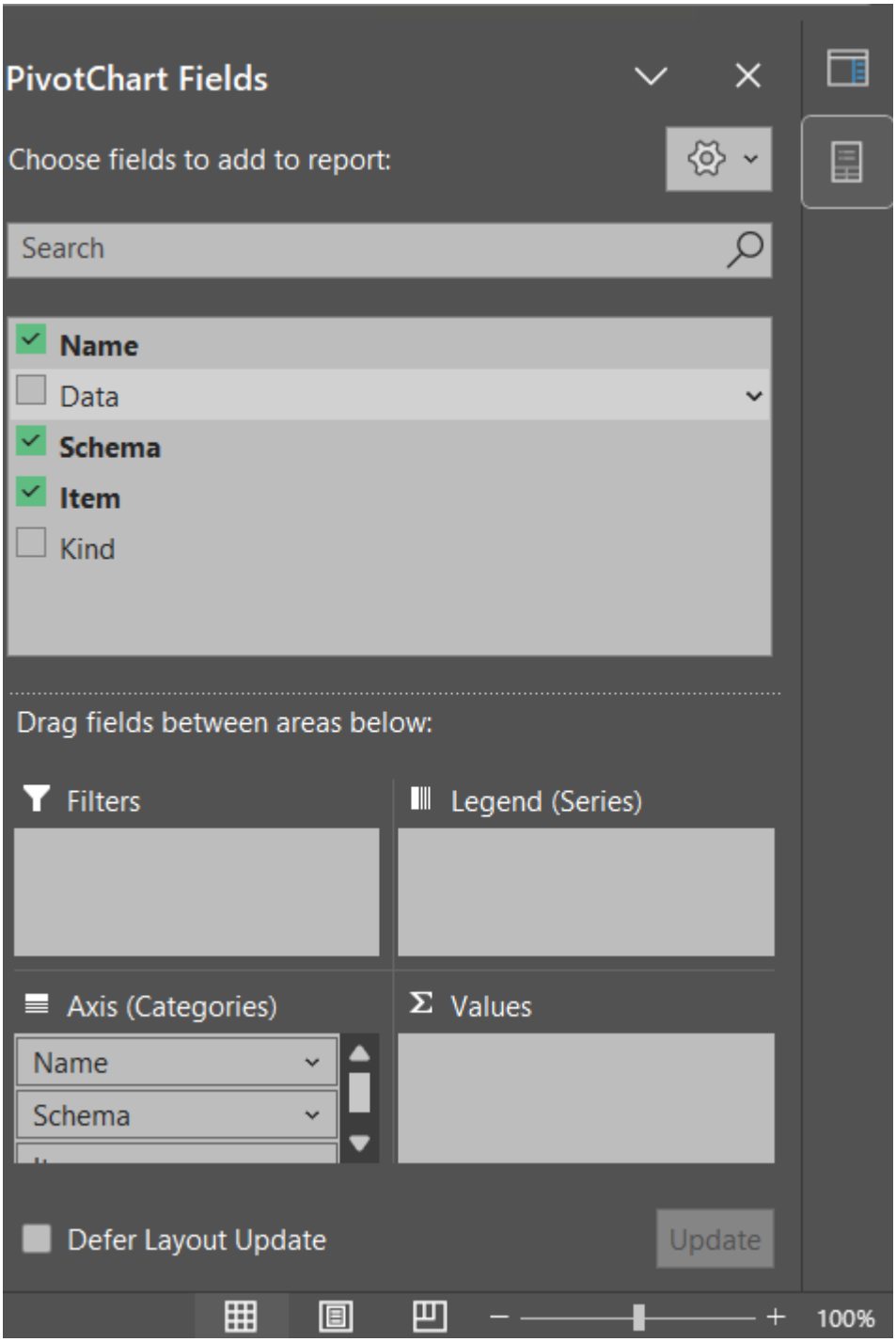
Use the **PivotTable Fields** pane to drag and drop fields:

Rows: Drag fields for grouping (e.g., Region, Product Category).

Columns: Drag fields for column categories.

Values: Drag numerical fields (e.g., Sales, Revenue) to perform calculations like Sum, Average.

Filters: Add fields to filter data dynamically.



Step 6: Create a Pivot Chart

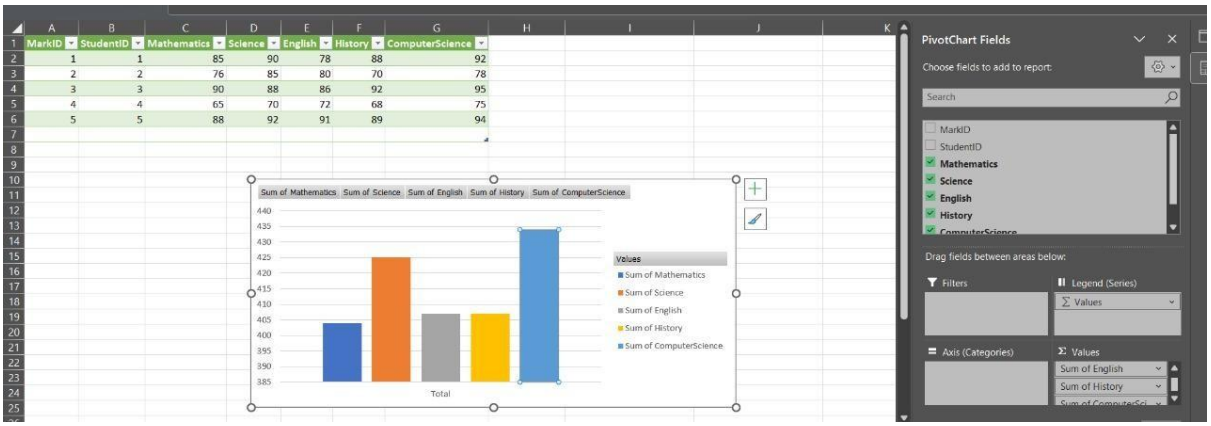
Click anywhere in the **Pivot Table**.

Go to the **Insert** tab and select **PivotChart**.

Choose the chart type (e.g., Column, Bar, Line, Pie).

Click **OK**, and the chart will appear in the worksheet.

Customize the chart using the **Chart Tools** tab.



Step 7: Save and Refresh Data

Save the Excel file for future use.

To refresh the Pivot Table and Chart when new data is added, go to **Data** > **Refresh All**.

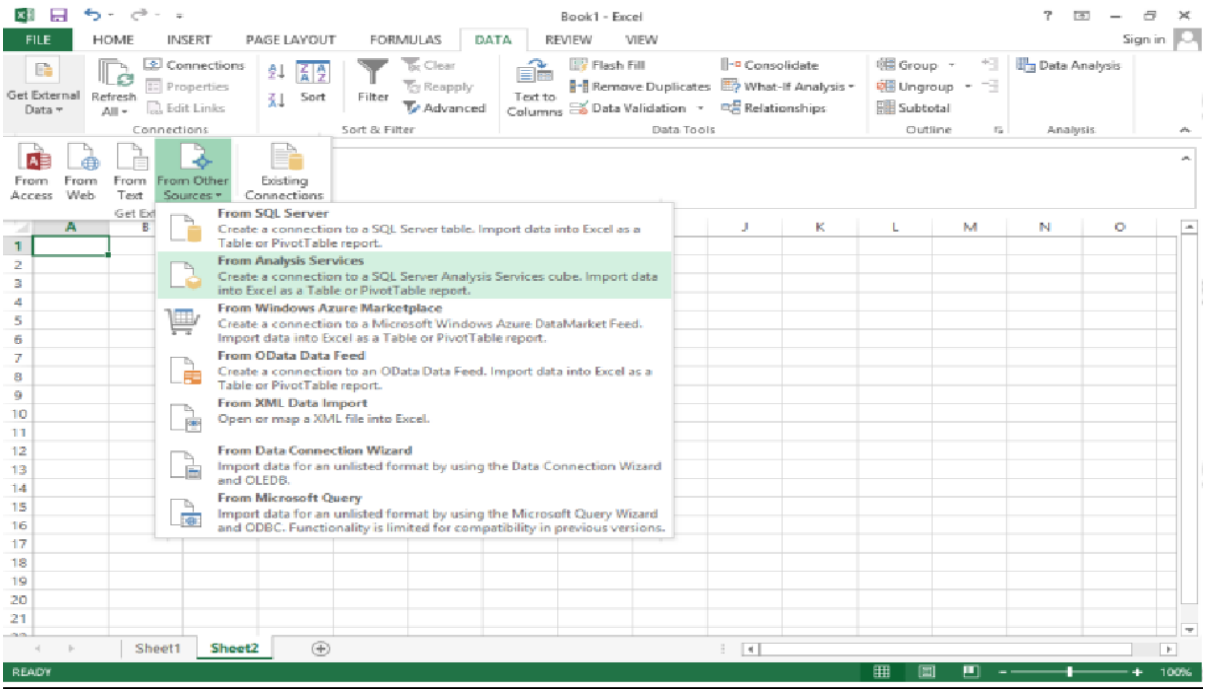
PRATICAL-1(B)

AIM:- Import the cube in Microsoft Excel and create the Pivot table and Pivot Chart to perform data analysis

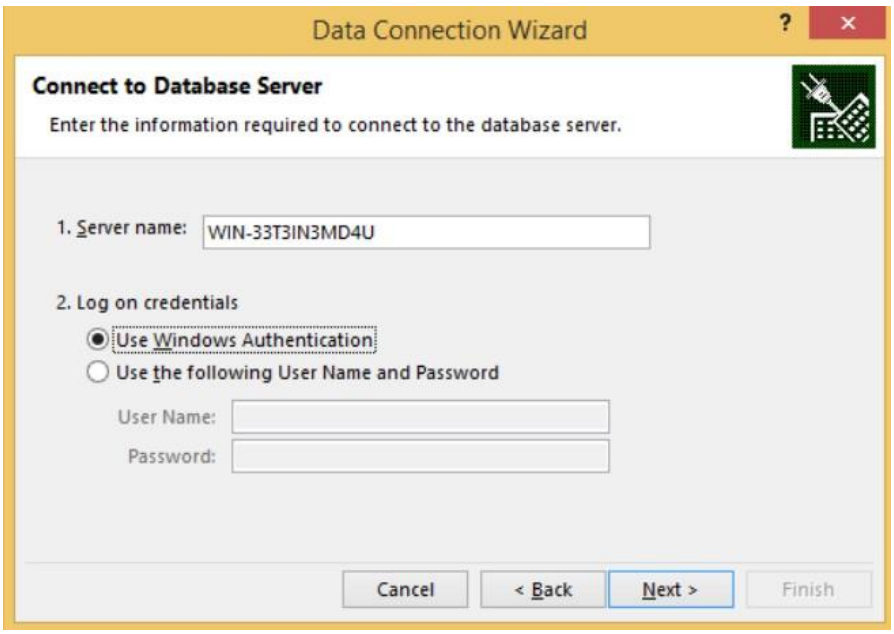
STEPS:-

Step 1:-Open Microsoft Excel

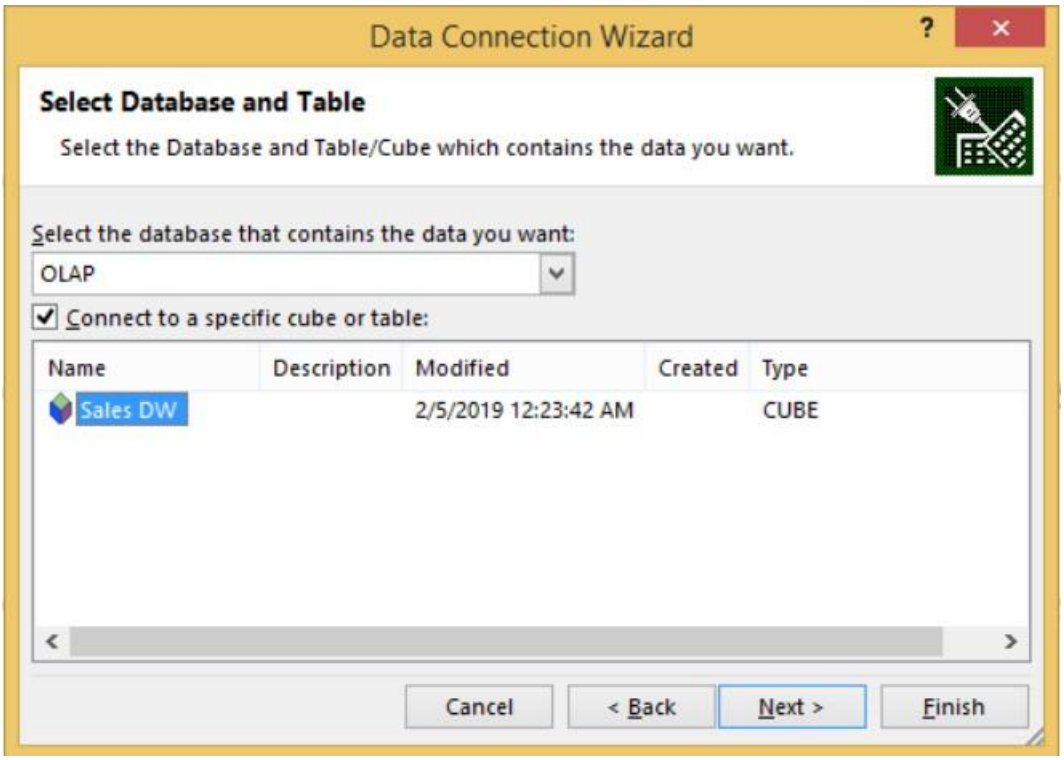
Step2:- Step Go to Data tab → Get External Data → From Other Sources → From Analysis Services



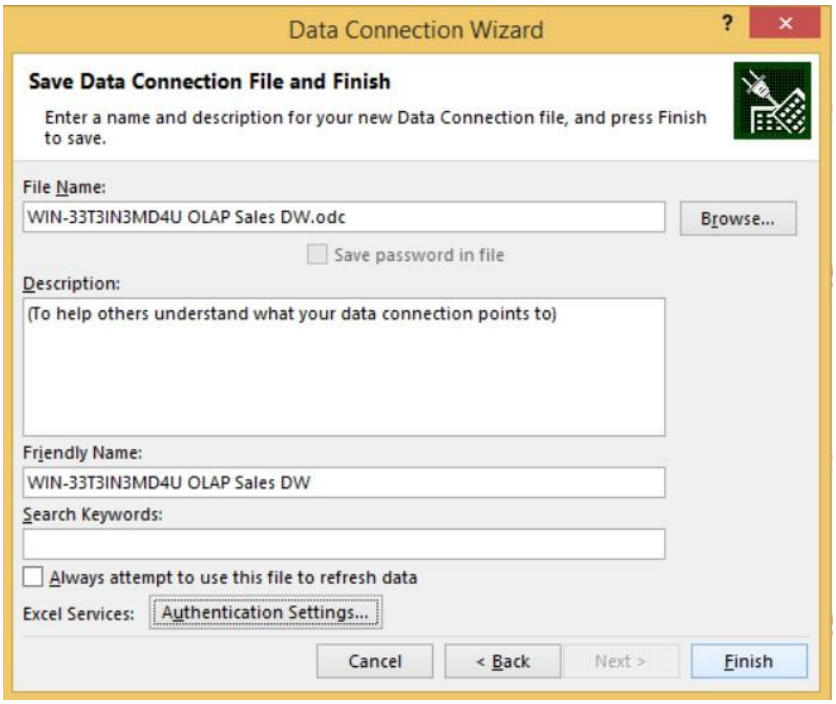
Step 3:- Select Server name and Windows Authentication and click on Next



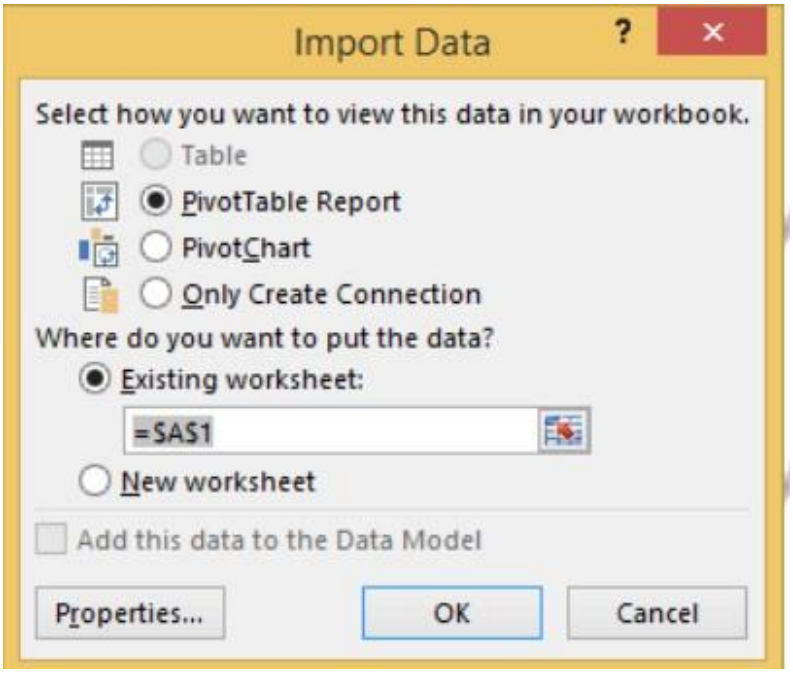
Step 4:- Select OLAP(as per created before) click on Next



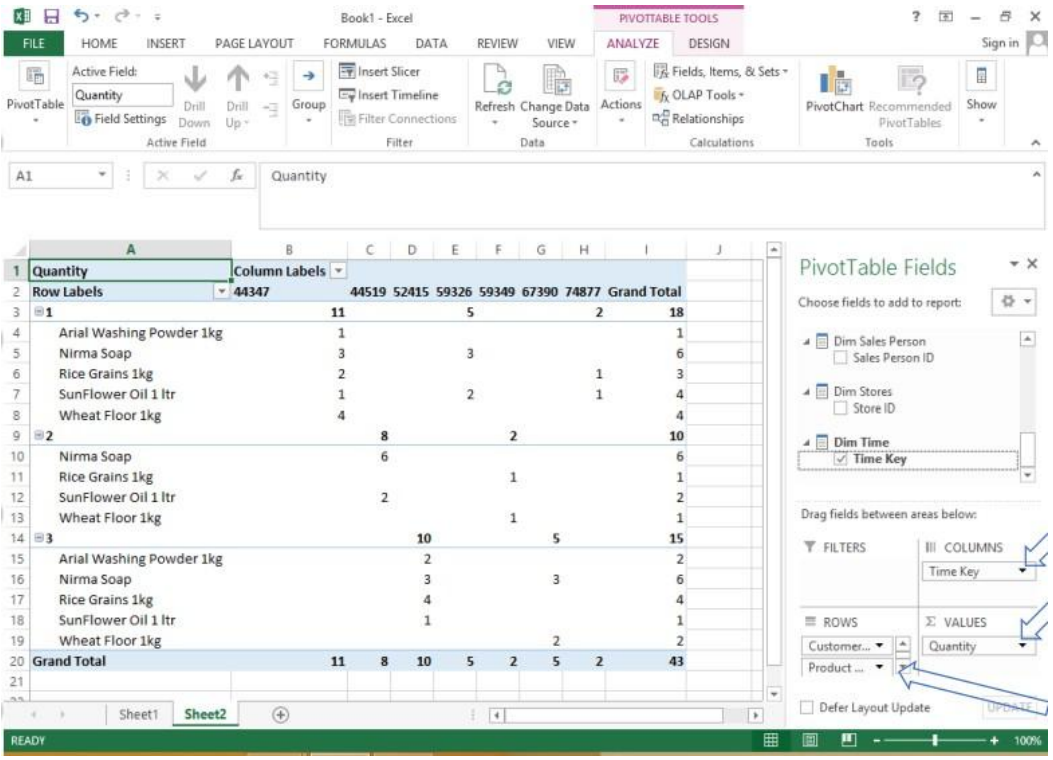
Step5:- Browse and select path name and click on Finish



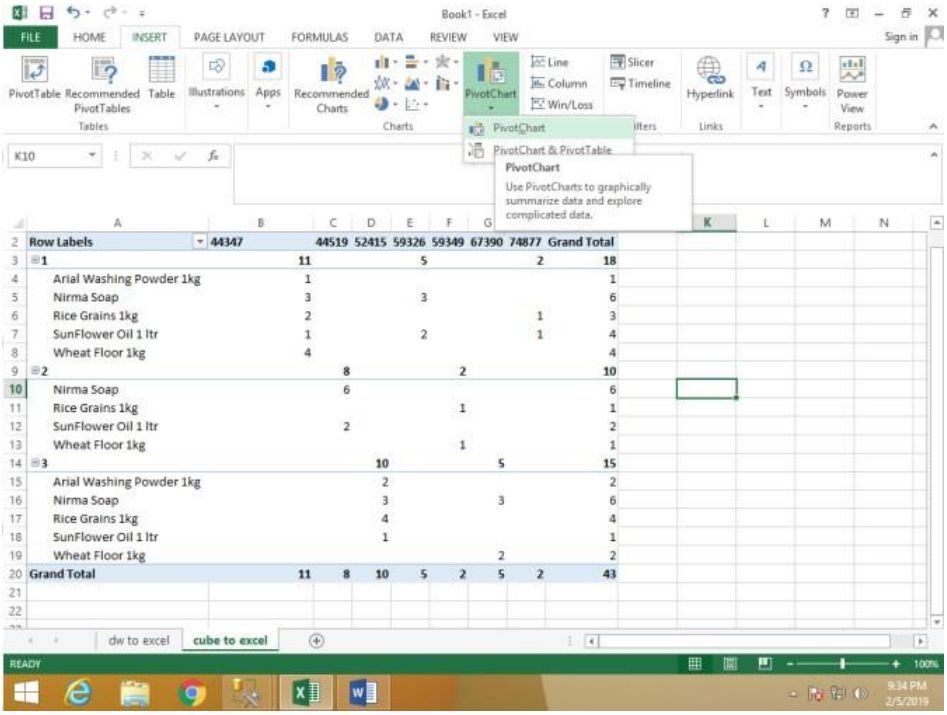
Step 6:- Select Pivot Table Report → OK



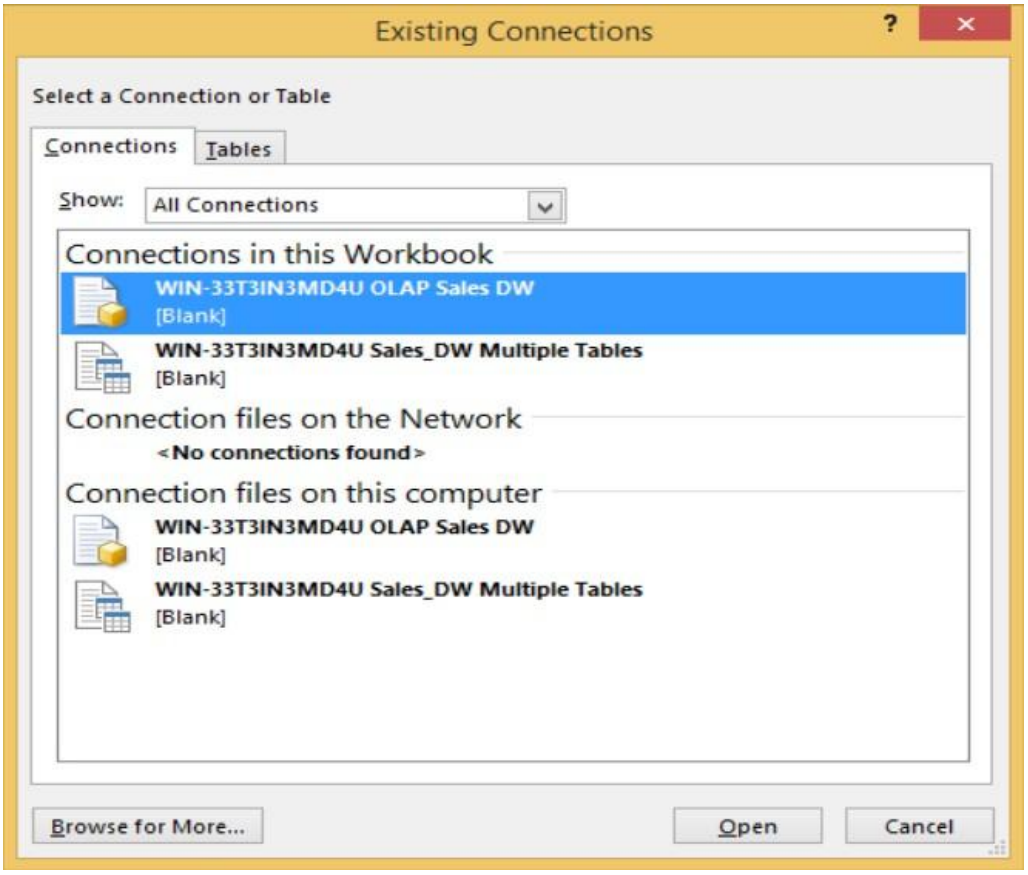
Step 7:- Drag and Drop Fields in rows column and values



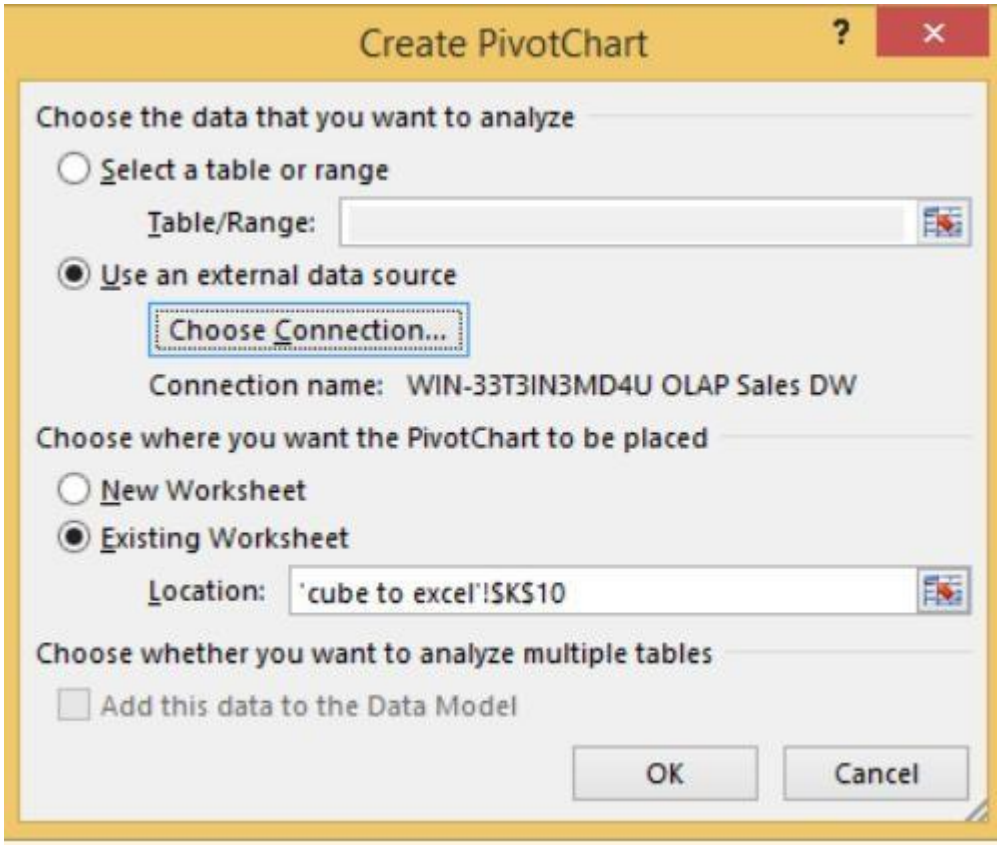
Step 8:-Go to Insert tab → pivot chart and select Pivot Chart from drop down



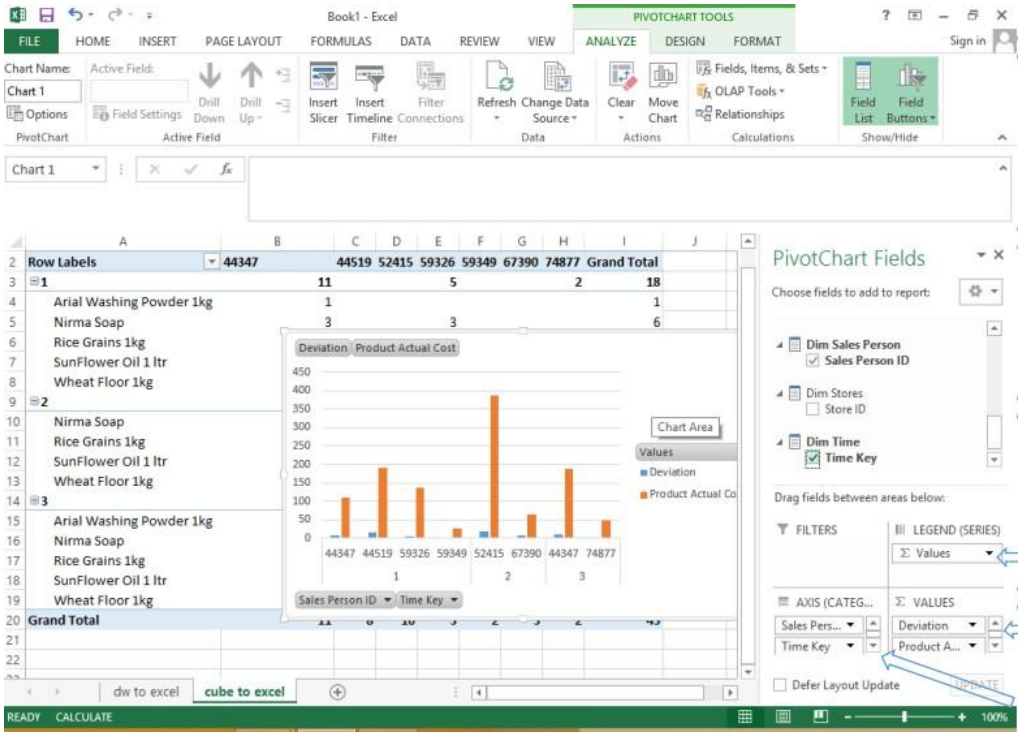
Step 9:- Select existing connection OLAP Sales DW and click on Open



Step 10:- Click on Choose connection to select path



Step 11:- Click on OK

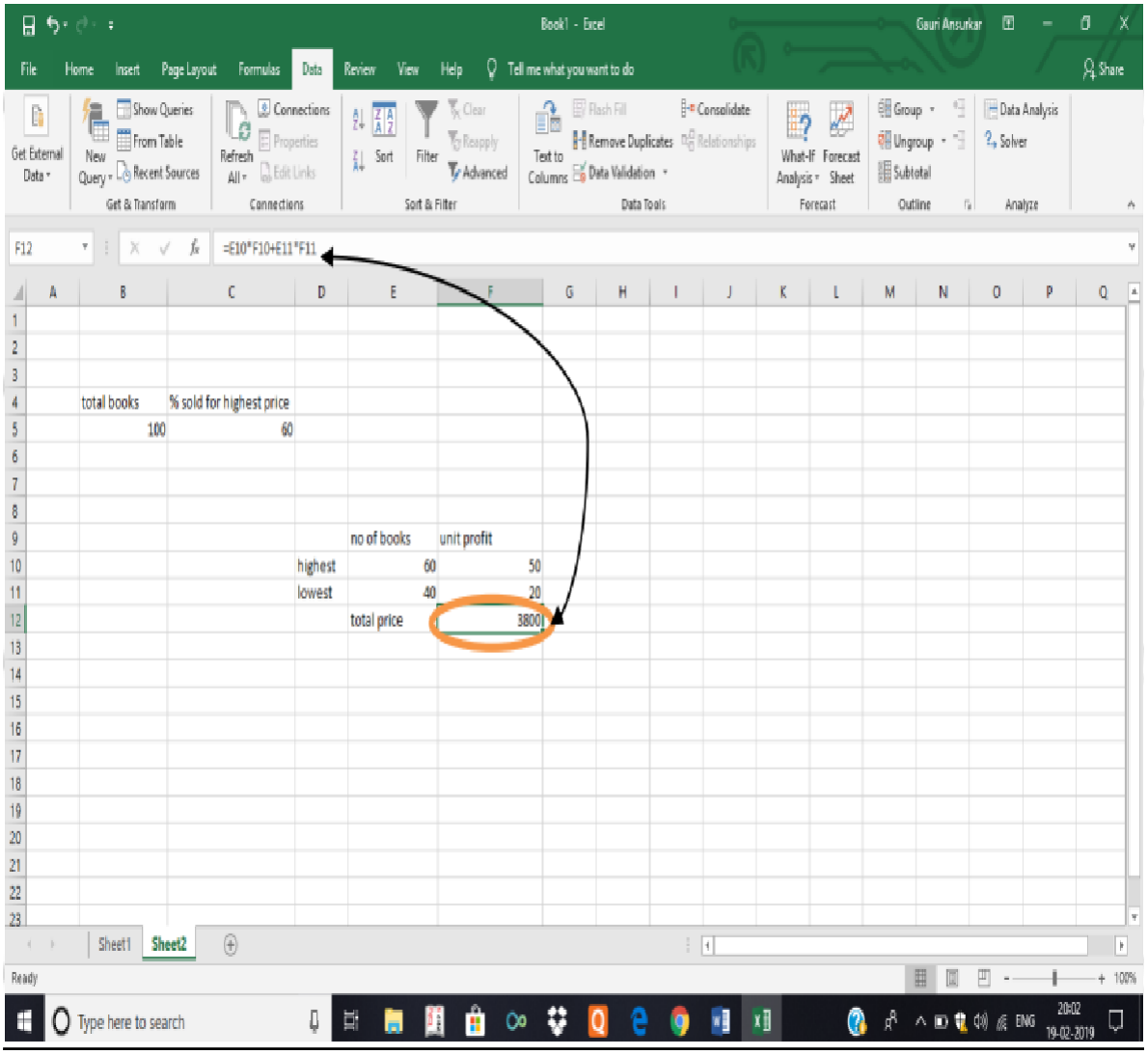


PRATICAL-2

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

STEPS:-

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.



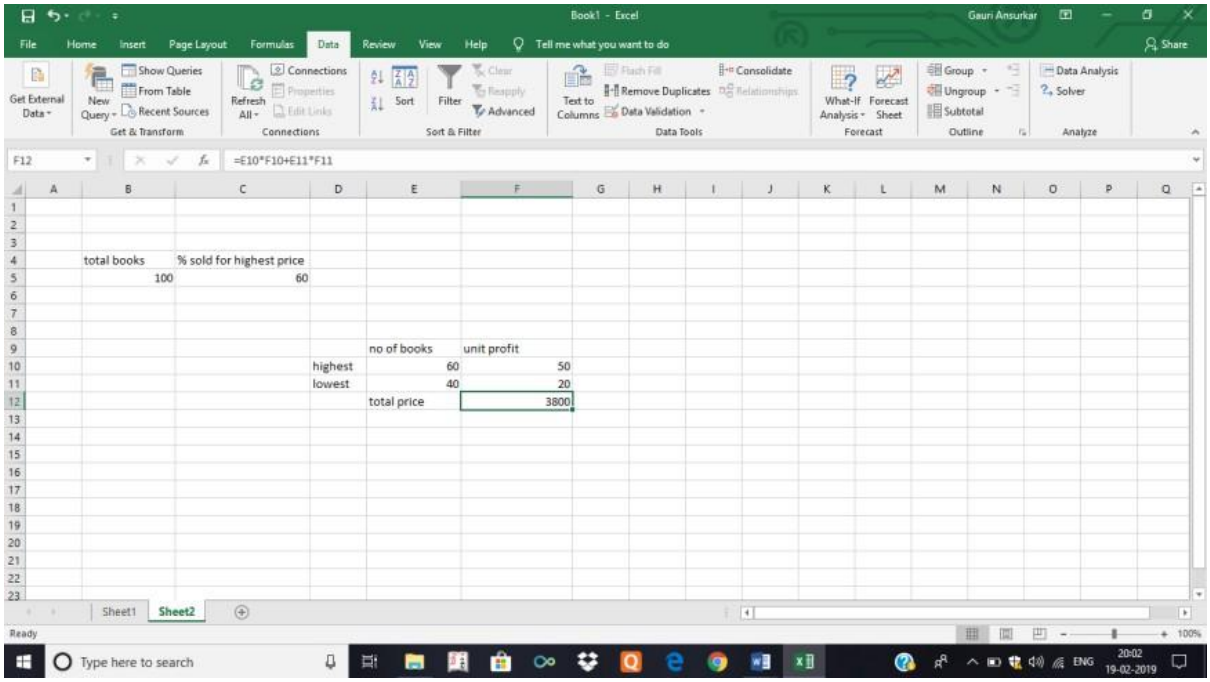
If you sell 60% for the highest price, cell D10 calculates a total profit of 60 * 50 + 40 * 20 = 3800.

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.

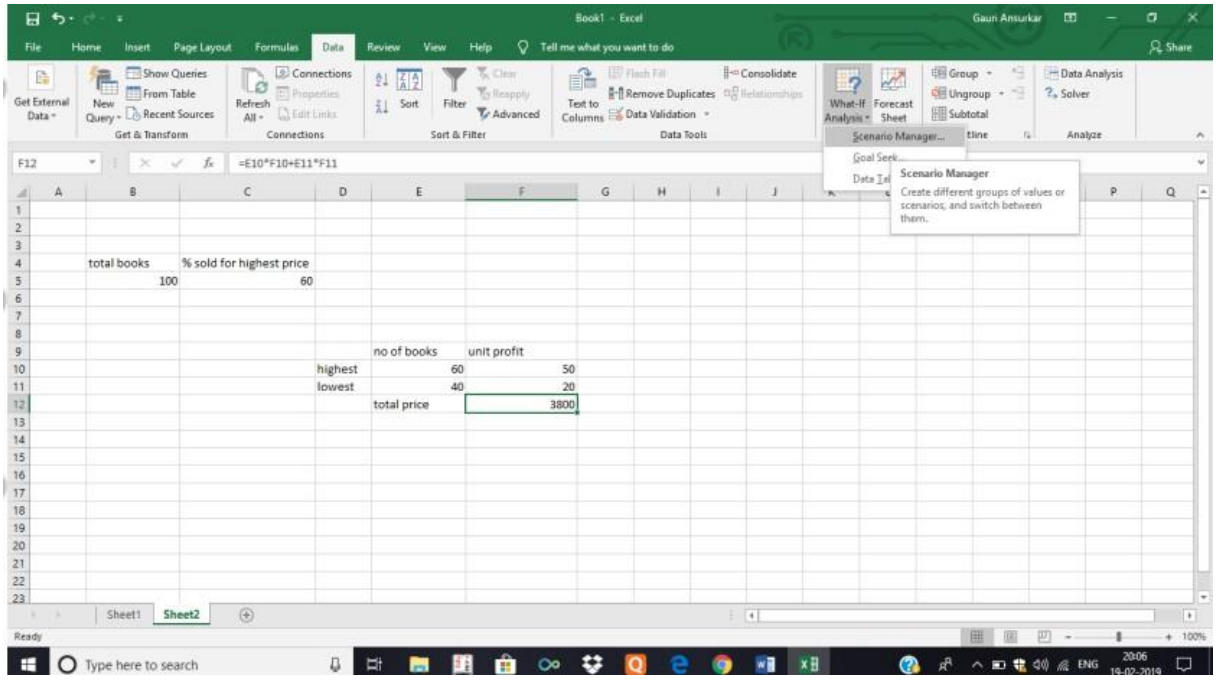
Note: To type different percentage into cell C4 to see the corresponding result of a scenario in cell D10 we use what if analysis.

What-if analysis enables you to easily compare the results of different scenarios.

Step 1: In Excel, On the Data tab, in the Data tools group, click What-If Analysis

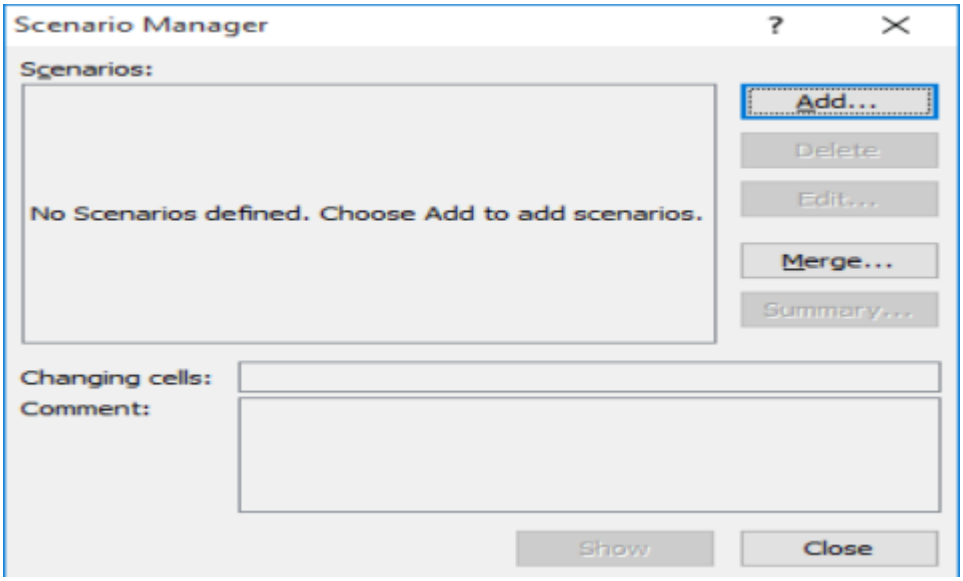


Step 2: Click on What –if-Analysis and select scenario manager.



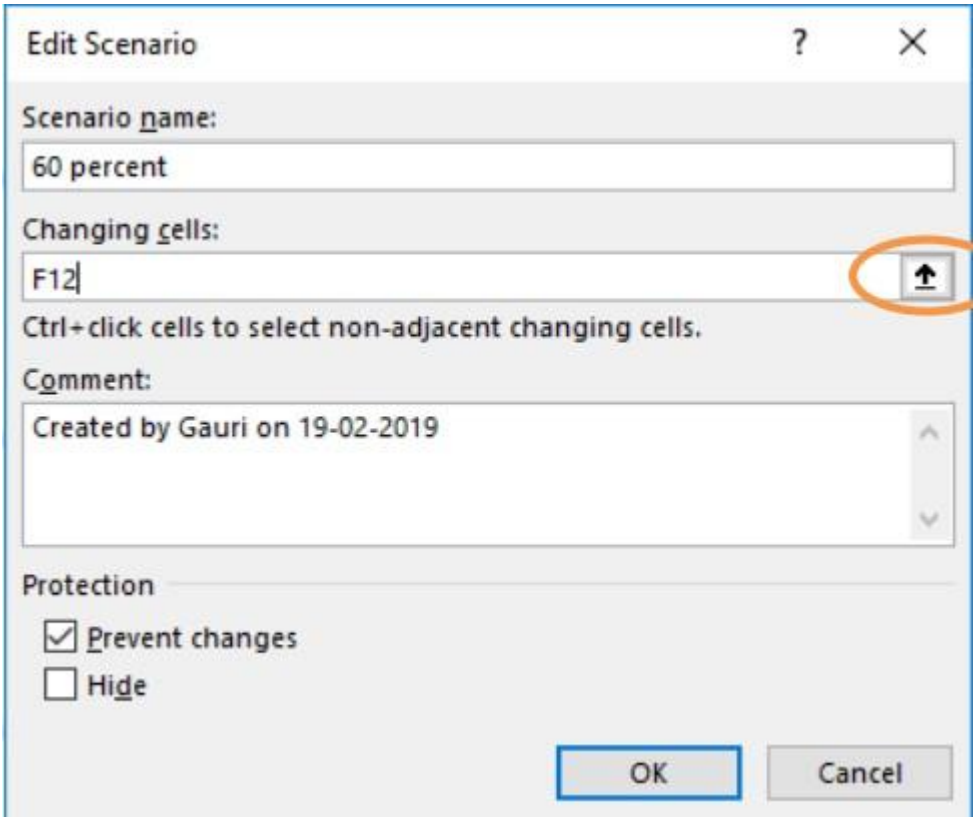
The Scenario Manager Dialog box appears.

Step 3: Add a scenario by clicking on Add.

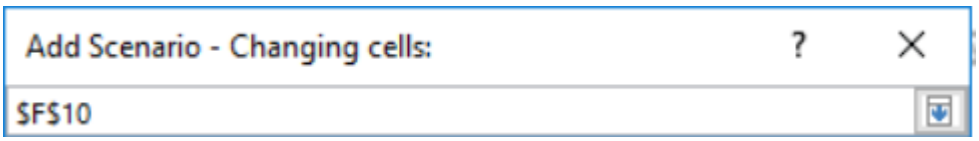


Step 4: Type a name (60percent), select cell F10 (% sold for the highest price) for the Changing cells and click on OK.

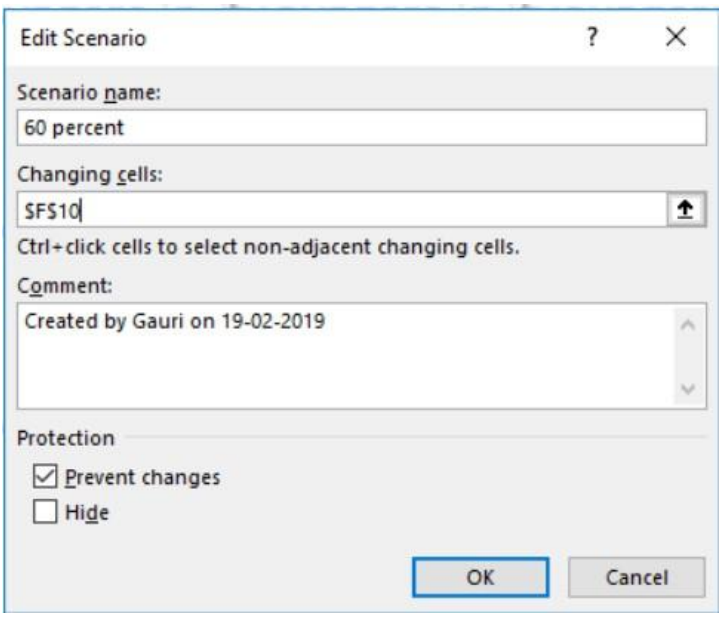
Click on icon which is circled.



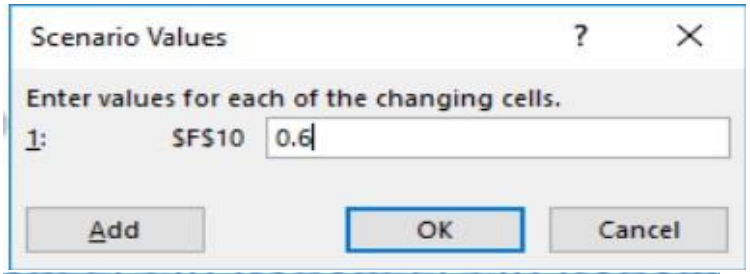
Select F10 cell.

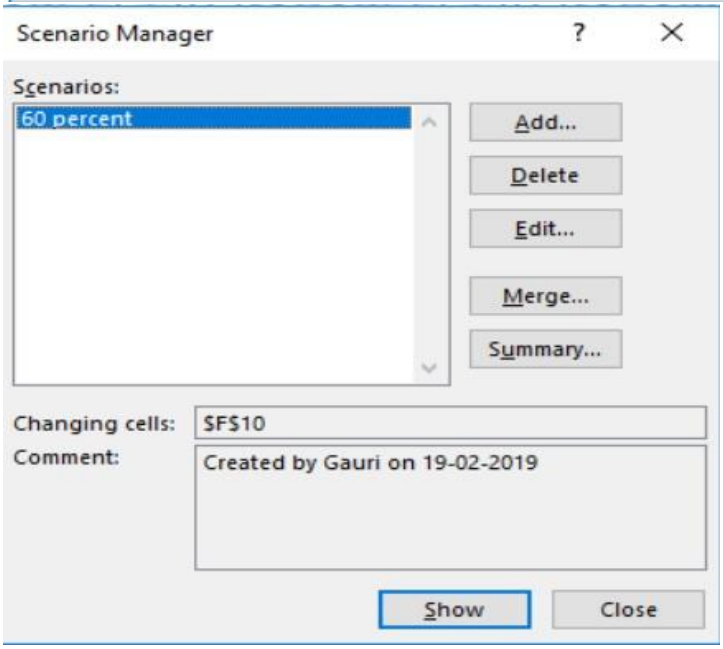


Click back on the icon again and then click OK

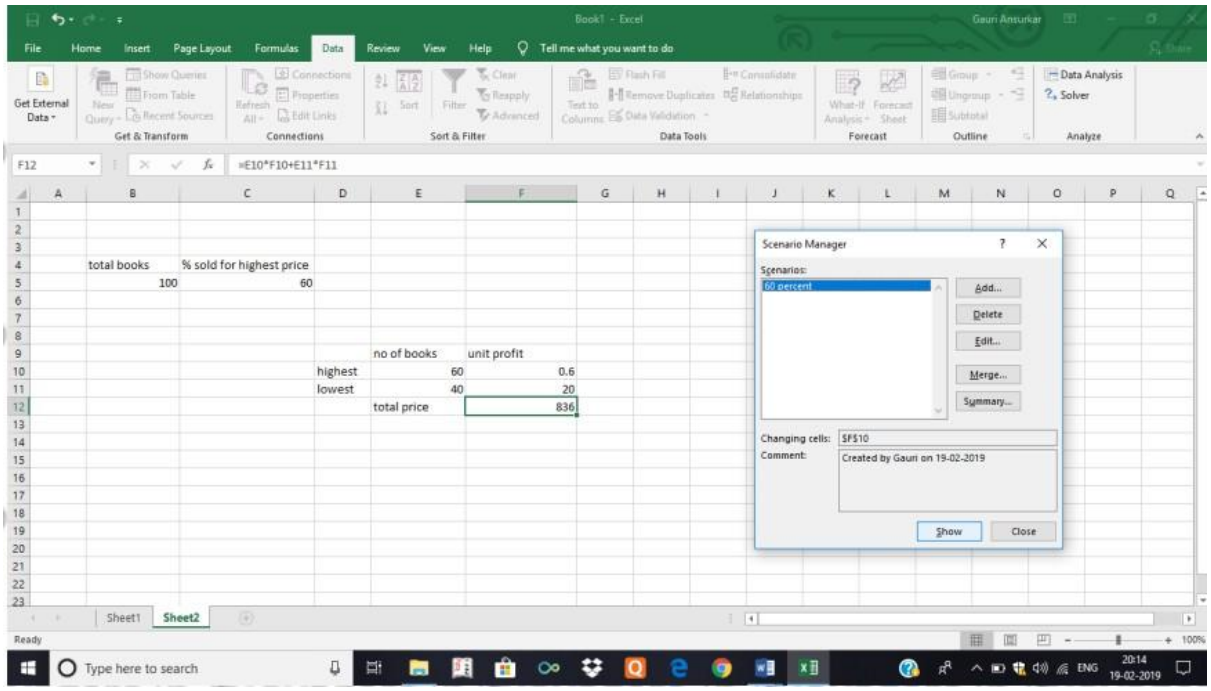


Step 5: Enter the corresponding value 0.6 and click on OK again.

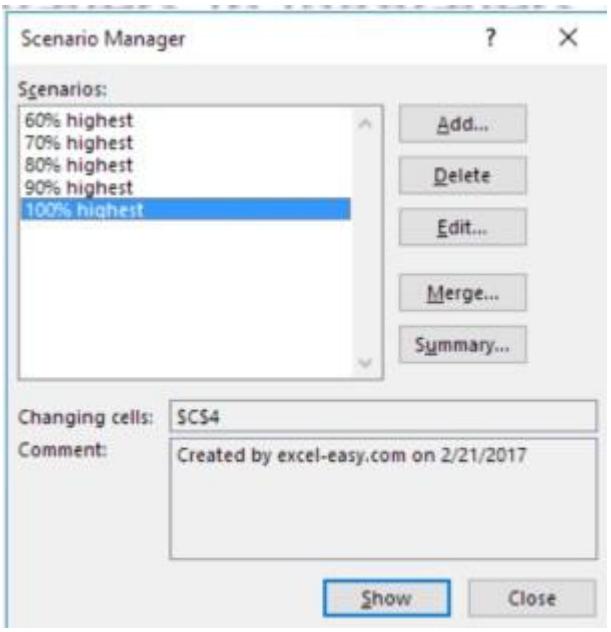




Step 6: To apply scenarios click on Show



Step 7: Next, add 4 other scenarios (70%, 80%, 90% and 100%) Finally, your Scenario Manager should be consistent with the picture below:



Practical No 3

Aim : Perform the data classification using classification algorithm using R/Python.

```
#Get the data in the form of R vector
rainfall<-c(799,1174.8,885.1,1334.6,635.4,981.5,685.5,998.6,784.2,985,882.8,1071)
#converting it to a timeseries object
rainfall.timeseries<-ts(rainfall,start=c(2024),frequency = 12)
#print the timeseries data
print(rainfall.timeseries)
print(rainfall.timeseries)
```

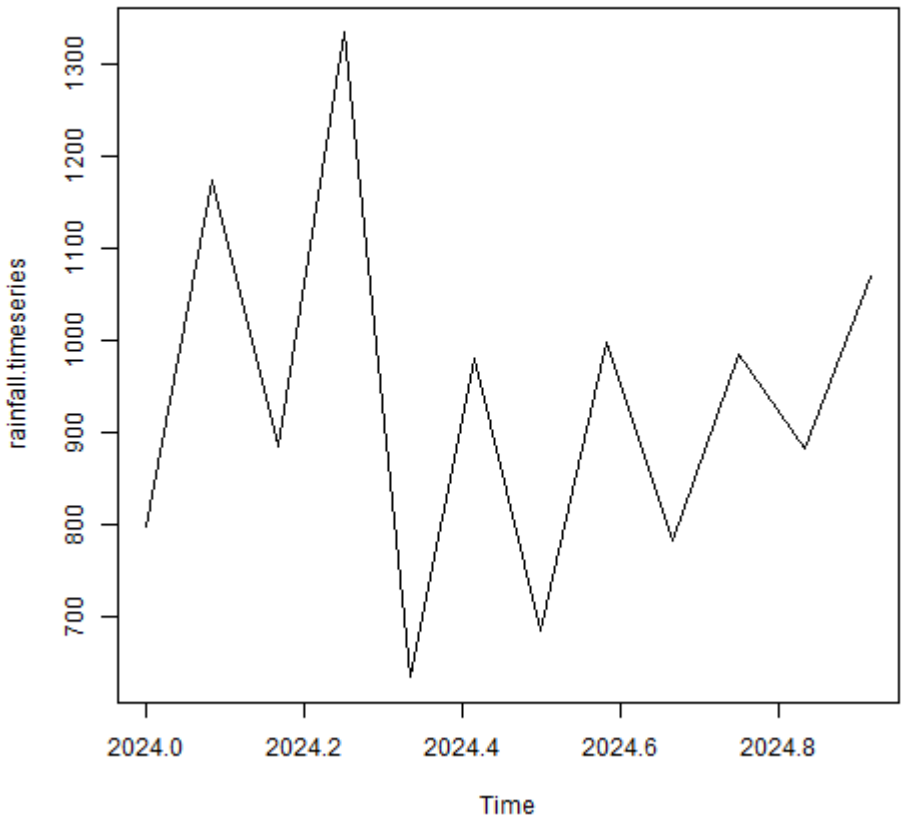
Output:

```
     Jan    Feb    Mar    Apr    May    Jun    Jul    Aug
2024  799.0 1174.8  885.1 1334.6  635.4  981.5  685.5  998.6
     Sep    Oct    Nov    Dec
2024  784.2  985.0  882.8 1071.0
```

```
# give the file name
png(file="rainfall.png")
#plot the graph of timeseries
plot(rainfall.timeseries)
#save the file
dev.off()
```

Output:

```
null device
          1
```


rainfall.timeseries
700
800
900
1000
1100
1200
1300
2024.0
2024.2
2024.4
2024.6
2024.8
Time

Practical No 4

Aim : Perform the data clustering using clustering algorithm using R/Python.

Syntax: kmeans(x, centers, nstart)

where,

x represents numeric matrix or data frame object

centers represents the K value or distinct cluster centers

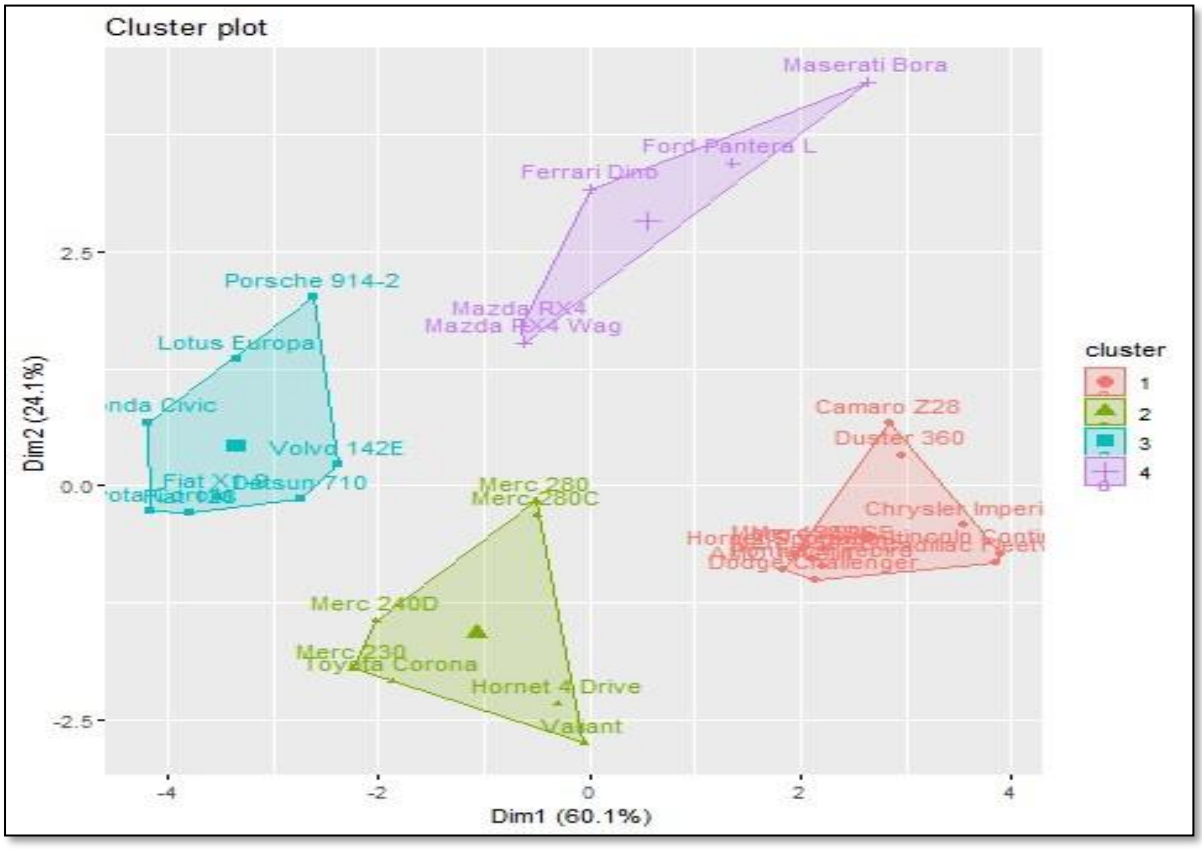
nstart represents number of random sets to be chosen

```
# Library required for fviz_cluster function
install.packages("factoextra")
library(factoextra)
# Loading dataset
df <- mtcars
# Omitting any NA values
df <- na.omit(df)
# Scaling dataset
df <- scale(df)
# output to be present as PNG file
png(file = "KMeansExample11.png")
km <- kmeans(df, centers = 4, nstart = 25)

# Visualize the clusters
fviz_cluster(km, data = df)

# saving the file
dev.off()

# output to be present as PNG file
```

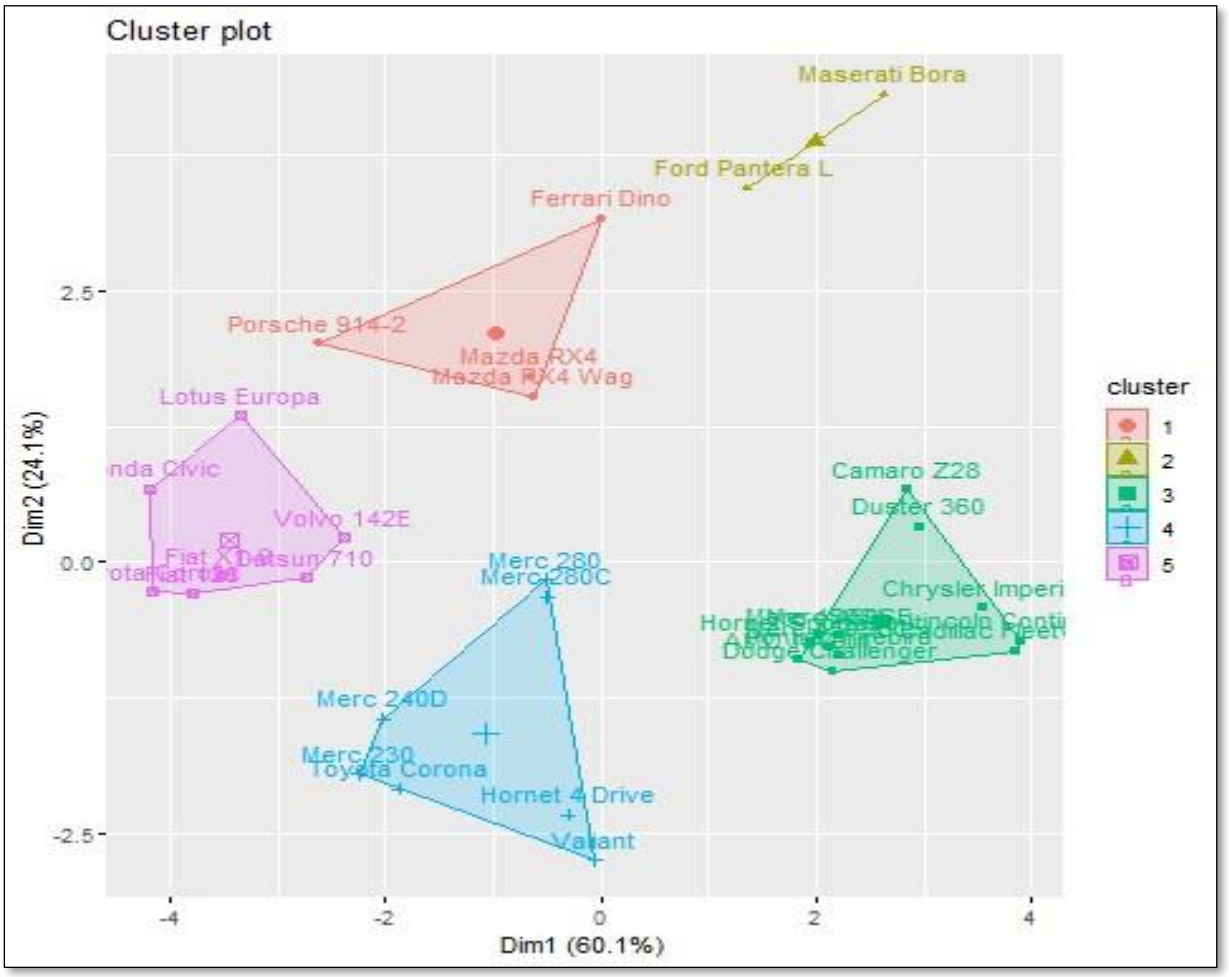
```
png(file = "KMeansExample12.png")

km <- kmeans(df, centers = 5, nstart = 25)

# Visualize the clusters
fviz_cluster(km, data = df)

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

Output:

When k = 4



When k = 5


Cluster plot
Maserati Bora
Ford Pantera L
Ferrari Dino
Porsche 914-2
Mazda RX4
Mazda RX4 Wag
Lotus Europa
Volvo 142E
Datsun 710
Merc 280
Merc 280C
Merc 240D
Merc 230
Toyota Corona
Hornet 4 Drive
Valiant
Camaro Z28
Duster 360
Chrysler Imperi
Dodge Challenger
cluster
1
2
3
4
5
Dim2 (24.1%)
Dim1 (60.1%)
2.5
0.0
-2.5
-4
-2
0
2
4

Practical No 5

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

What is Linear Regression?

Linear Regression is a commonly used type of predictive analysis. Linear Regression is a statistical approach for modelling the relationship between a dependent variable and a given set of independent variables. It is predicted that a straight line can be used to approximate the relationship. The goal of linear regression is to identify the line that minimizes the discrepancies between the observed data points and the line's anticipated values.

There are two types of linear regression.

Simple Linear Regression

Multiple Linear Regression

Simple Linear Regression:

In Machine Learning Linear regression is one of the easiest and most popular Machine Learning algorithms.

It is a statistical method that is used for predictive analysis.

Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable changes according to the value of the independent variable.

Implementation in R

In R programming, lm() function is used to create linear regression model.

Syntax: lm(formula)

Parameter:

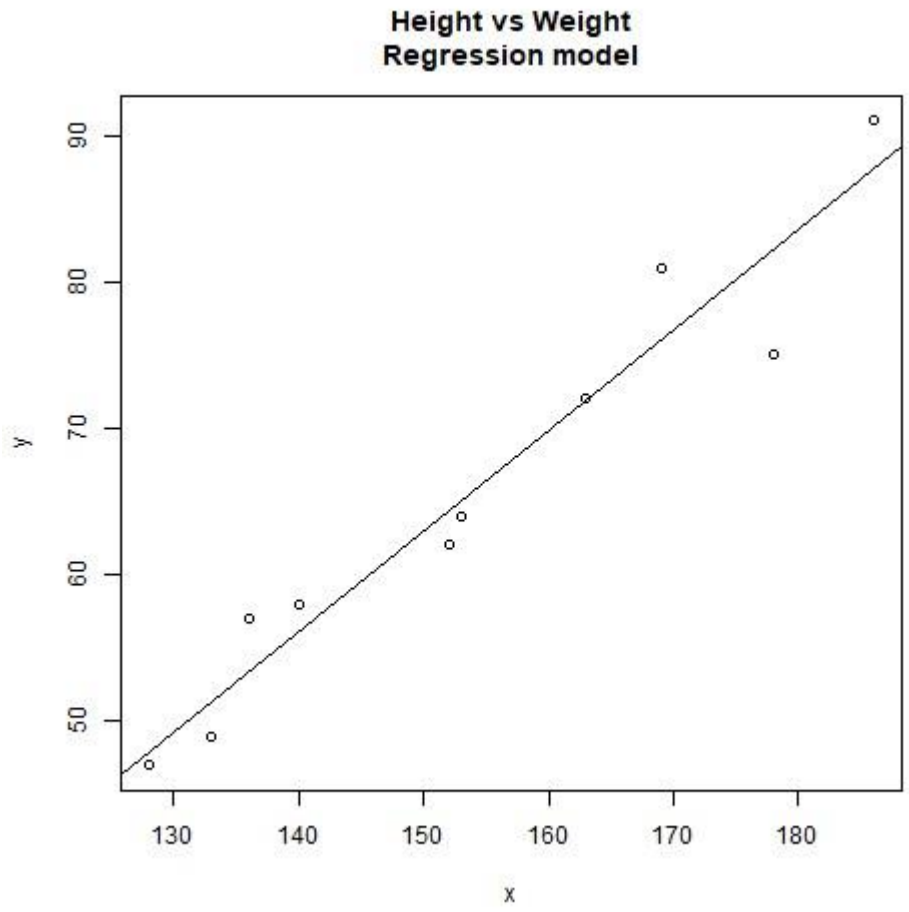
formula: represents the formula on which data has to be fitted To know about more optional parameters, use below command in console: help("lm")

Example: In this example, let us plot the linear regression line on the graph and predict the weight-based using height.

Code:

```
# R program to illustrate
# Linear Regression
# Height vector
x <- c(153, 169, 140, 186, 128,
       136, 178, 163, 152, 133)
# Weight vector
y <- c(64, 81, 58, 91, 47, 57,
       75, 72, 62, 49)
# Create a linear regression model
model <- lm(y~x)
# Print regression model
print(model)
# Find the weight of a person
# With height 182
df <- data.frame(x = 182)
res <- predict(model, df)
cat("\nPredicted value of a person
                          with height = 182")
print(res)
# Output to be present as PNG file
png(file = "linearReg.png")
# Plot
plot(x, y, main = "Height vs Weight
                                  Regression model")
abline(lm(y~x))
# Save the file.
dev.off()
```

Output:



Extend the analysis to multiple linear regression and assess the impact of additional predictors.

Multiple Linear Regression :

It is the most common form of Linear Regression. Multiple Linear Regression basically describes how a single response variable Y depends linearly on a number of predictor variables.

The basic examples where Multiple Regression can be used are as follows:

The selling price of a house can depend on the desirability of the location, the number of bedrooms, the number of bathrooms, the year the house was built, the square footage of the lot, and a number of other factors.

The height of a child can depend on the height of the mother, the height of the father, nutrition, and environmental factors.

Implementation in R

Multiple regression in R programming uses the same lm() function to create the model.

Syntax: lm(formula, data)

Parameters:

formula: represents the formula on which data has to be fitted

data: represents dataframe on which formula has to be applied

Example: Let us create a multiple regression model of air quality dataset present in R base package and plot the model on the graph.

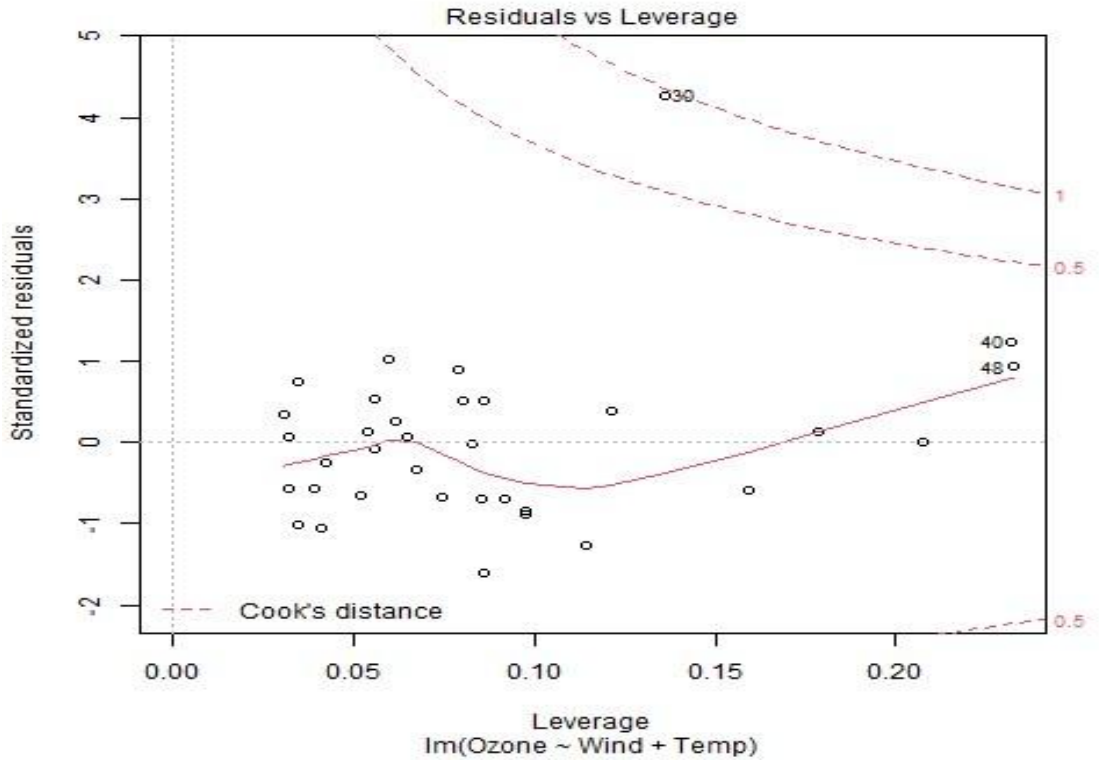
Code:

```
# R program to illustrate
# Multiple Linear Regression

# Using airquality dataset
input <- airquality[1:50,
                    c("Ozone", "Wind", "Temp")]

# Create regression model
model <- lm(Ozone~Wind + Temp,
            data = input)

# Print the regression model
cat("Regression model:\n")
print(model)

# Output to be present as PNG file
png(file = "multipleRegGFG.png")

# Plot
```

plot(model)

Save the file.

dev.off()



Practical No.6

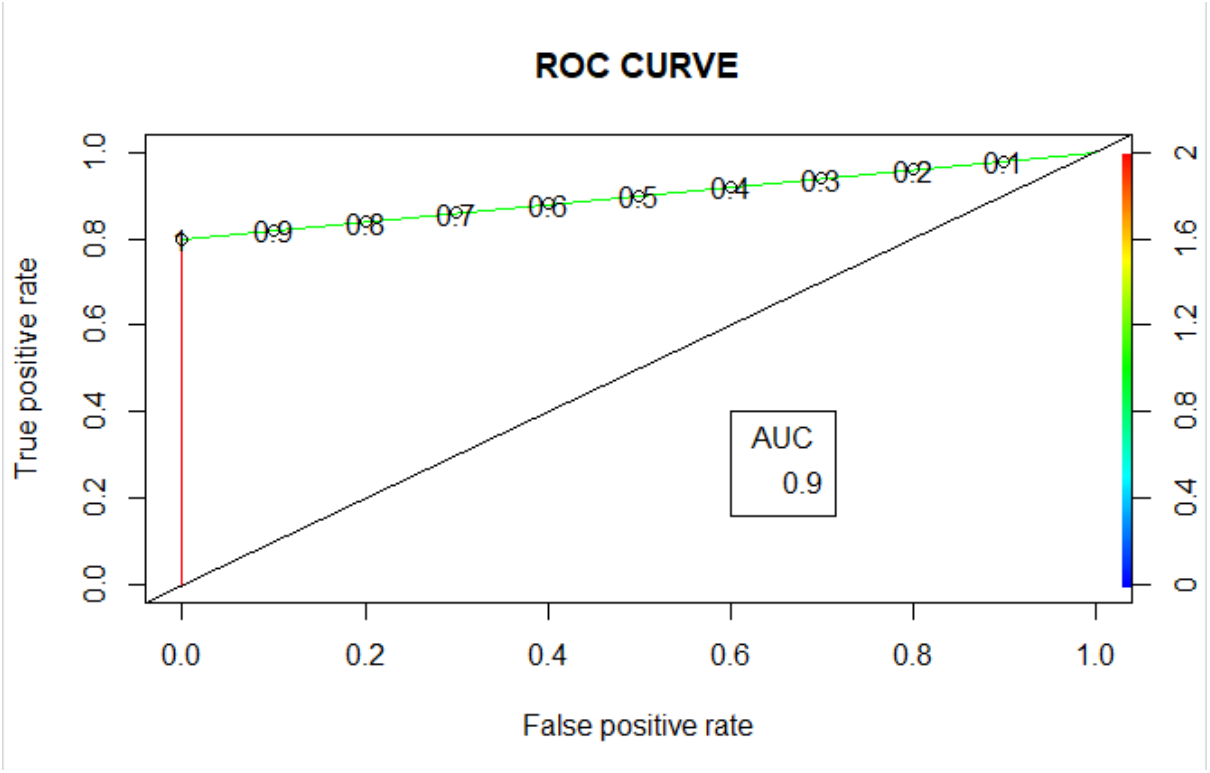
Aim : Perform the logistic regression on the given data warehouse data using R/Python.

Code:

```
# Installing the package
install.packages("dplyr")


# Loading package
library(dplyr)


# Summary of dataset in package
summary(mtcars)


# Installing the package


# For Logistic regression
install.packages("caTools")


# For ROC curve to evaluate model
install.packages("ROCR")


# Loading package
library(caTools)
library(ROCR)


# Splitting dataset
split <- sample.split(mtcars, SplitRatio = 0.8)
split
```

```
train_reg <- subset(mtcars, split == "TRUE")
test_reg <- subset(mtcars, split == "FALSE")

# Training model
logistic_model <- glm(vs ~ wt + disp,
                      data = train_reg,
                      family = "binomial")
logistic_model

# Summary
summary(logistic_model)

predict_reg <- predict(logistic_model,
                       test_reg, type = "response")
predict_reg



# Changing probabilities
predict_reg <- ifelse(predict_reg >0.5, 1, 0)

# Evaluating model accuracy
# using confusion matrix
table(test_reg$vs, predict_reg)

missing_classerr <- mean(predict_reg != test_reg$vs)
print(paste('Accuracy =', 1 - missing_classerr))

# ROC-AUC Curve
```

```
ROCPred <- prediction(predict_reg, test_reg$vs)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr")

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
auc

# Plotting curve
plot(ROCPer)
plot(ROCPer, colorize = TRUE,
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROC CURVE")
abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```



Practical No 7

AIM:- 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).

STEPS:-

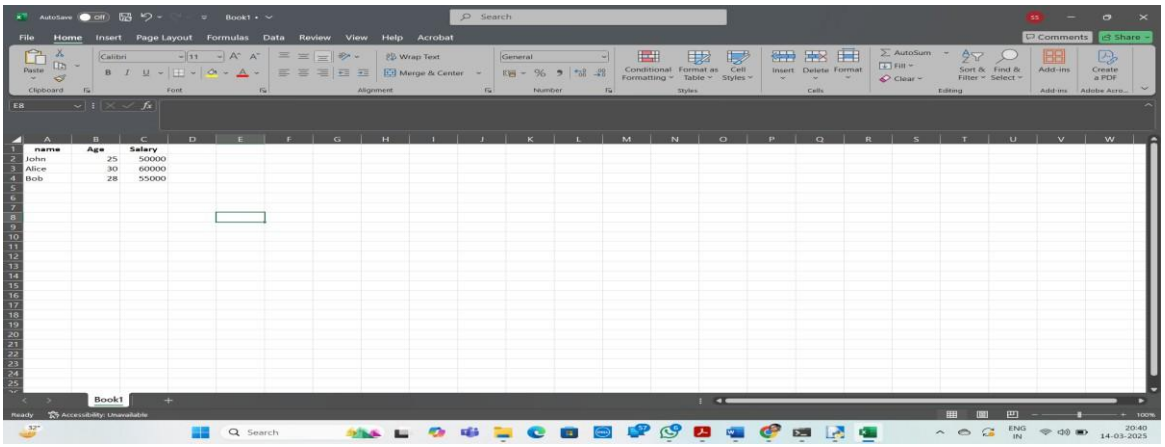
Step 1:- Install Required Library

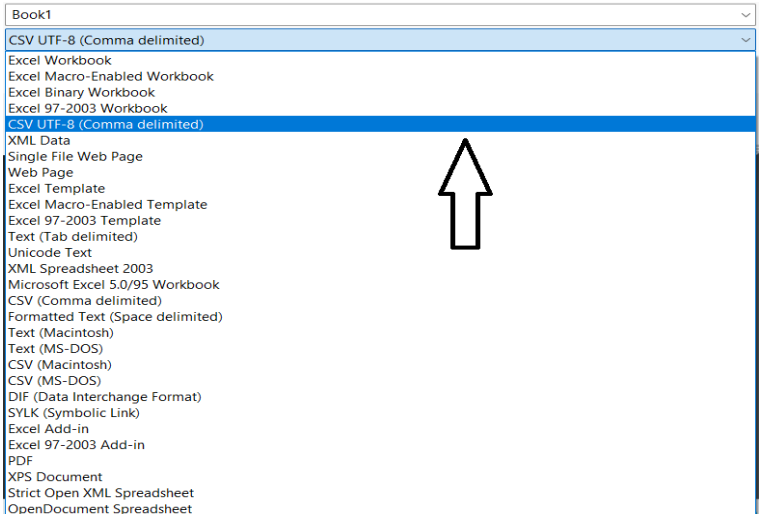
Before running the program, you need to install Pandas. Open a command prompt or terminal and type:

pip install pandas

Step 2:- Open MS EXCEL

Prepare a CSV File



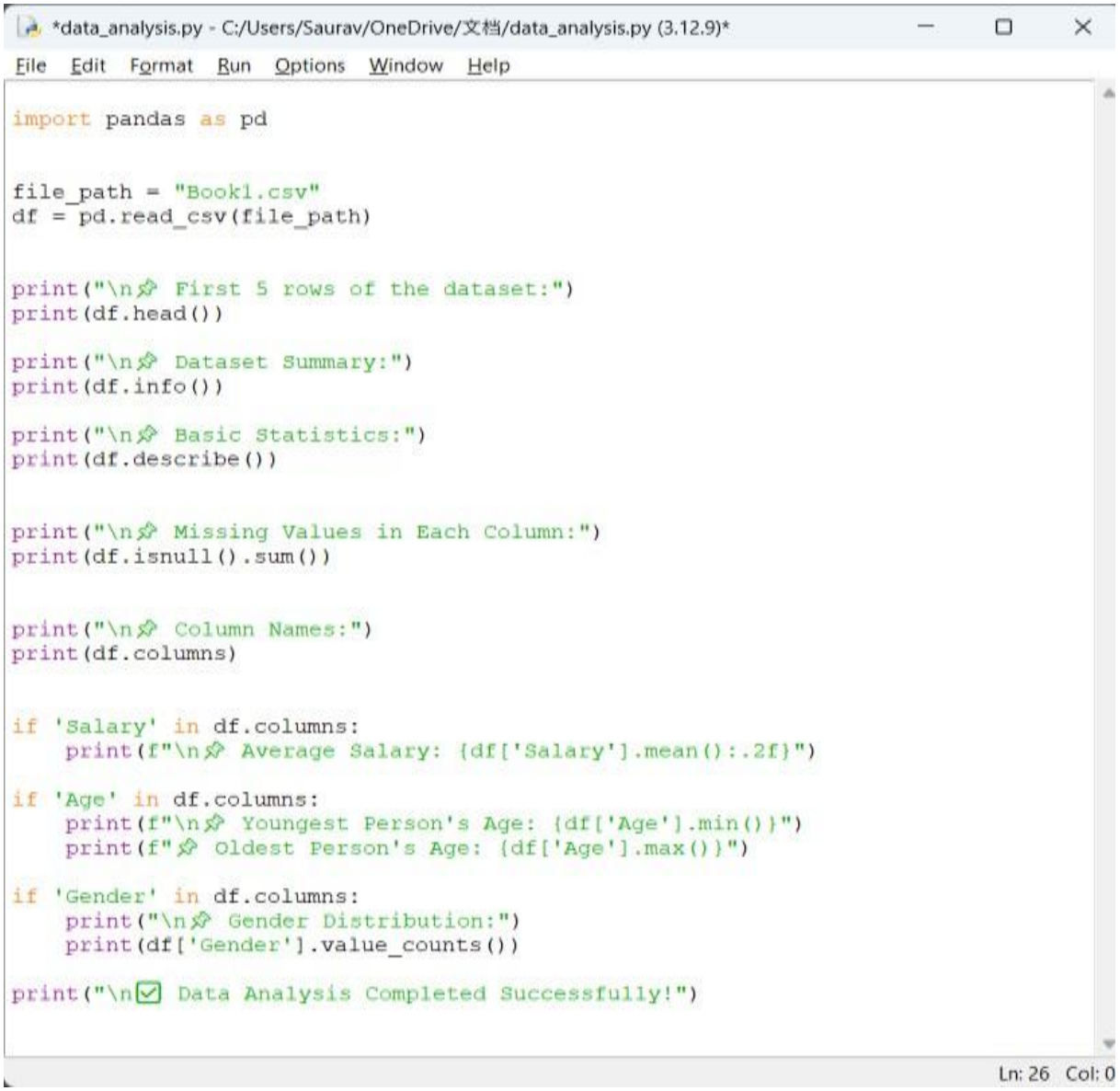


Step 3:- Open PYTHON IDLE

Write the Python Code

-

```
import pandas as pd
file_path = "Book1.csv”
df = pd.read_csv(file_path)
print(“\n First 5 row of the data set:”)
print(df.head()) #show the first 5 rows
print(“\n Data Summary:”)
print(df.info())
print(“\n Basic Statistics: ”)
print(df.describe())
print(“\n Missing Value in each Column:”)
print(df.isnull().sum())
print(“\n Column Names: ”)
print(df.columns)
if 'Salary' in df.columns:
printf(“\n Average Salary: {df['Salary'].mean():.2f}”)
if 'Age' in df.columns:
printf(“\n Youngest Person's Age: {df['Age'].min()}")
printf(“\n Youngest Person's Age: {df['Age'].min()}")
if 'Gender' in df.columns:
print(“\n Gender Distribution:”)
print(df['Gender'].value_counts())
print(“\n Data Analysis Completed Successfully!")
```


data_analysis.py - C:/Users/Saurav/OneDrive/文档/data_analysis.py (3.12.9)
File Edit Format Run Options Window Help
import pandas as pd
file_path = "Book1.csv"
df = pd.read_csv(file_path)
print("\n📌 First 5 rows of the dataset:")
print(df.head())
print("\n📌 Dataset Summary:")
print(df.info())
print("\n📌 Basic Statistics:")
print(df.describe())
print("\n📌 Missing Values in Each Column:")
print(df.isnull().sum())
print("\n📌 Column Names:")
print(df.columns)
if 'Salary' in df.columns:
print(f"\n📌 Average Salary: {df['Salary'].mean():.2f}")
if 'Age' in df.columns:
print(f"\n📌 Youngest Person's Age: {df['Age'].min()}")
print(f"📌 Oldest Person's Age: {df['Age'].max()}")
if 'Gender' in df.columns:
print("\n📌 Gender Distribution:")
print(df['Gender'].value_counts())
print("\n☑ Data Analysis Completed Successfully!")
Ln: 26 Col: 0

IDLE Shell 3.12.9

File Edit Shell Debug Options Window Help

```
📌 First 5 rows of the dataset:
    name  Age  Salary
0   John   25   50000
1  Alice   30   60000
2    Bob   28   55000

📌 Dataset Summary:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    3 non-null      object
 1   Age     3 non-null      int64
 2   Salary  3 non-null      int64
dtypes: int64(2), object(1)
memory usage: 204.0+ bytes
None

📌 Basic Statistics:
             Age   Salary
count   3.000000      3.0
mean   27.666667  55000.0
std     2.516611   5000.0
min    25.000000  50000.0
25%    26.500000  52500.0
50%    28.000000  55000.0
75%    29.000000  57500.0
max    30.000000  60000.0

📌 Missing Values in Each Column:
name      0
Age       0
Salary    0
dtype: int64

📌 Column Names:
Index(['name', 'Age', 'Salary'], dtype='object')

📌 Average Salary: 55000.00

📌 Youngest Person's Age: 25
📌 Oldest Person's Age: 30

☑ Data Analysis Completed Successfully!
>>>
```

Practical No. 8(A)

AIM:- Perform data visualization using Python on any sales data.

STEPS:-

Step 1:- Install Required Libraries

Before starting, install the necessary libraries.

Run the following command in your terminal or command prompt:

pip install pandas numpy matplotlib seaborn

Step 2:- Import Libraries

Now, open your Python script and import the required libraries: import pandas as pd

import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

Step 3:- Create Sample Sales Data

```
import pandas as pd import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
# Generate Sample Sales Data
dates = pd.date_range(start='2023-01-01', periods=100, freq='D') categories =
['Electronics', 'Clothing', 'Groceries', 'Furniture'] data = {
'Date': np.random.choice(dates, 200),
'Category': np.random.choice(categories, 200),
'Sales_Amount': np.random.randint(100, 1000, 200),
'Units_Sold': np.random.randint(1, 20, 200)
}
df = pd.DataFrame(data)

# Convert Date column to datetime format df['Date'] = pd.to_datetime(df['Date'])
# Summary Statistics print(df.describe())
# Sales Trend Over Time plt.figure(figsize=(12, 6))
```

sns.lineplot(data=df.groupby('Date')['Sales_Amount'].sum().reset_index(), x='Date', y='Sales_Amount')

plt.title('Sales Trend Over Time') plt.xlabel('Date') plt.ylabel('Total Sales') plt.xticks(rotation=45)

plt.show()

Sales by Category plt.figure(figsize=(10, 5))

sns.barplot(data=df.groupby('Category', as_index=False)['Sales_Amount'].sum(), x='Category', y='Sales_Amount', palette='viridis')

plt.title('Total Sales by Category') plt.xlabel('Category') plt.ylabel('Sales Amount') plt.show()

Distribution of Sales Amounts plt.figure(figsize=(10, 5))

sns.histplot(df['Sales_Amount'], bins=20, kde=True, color='blue') plt.title('Distribution of Sales Amounts')

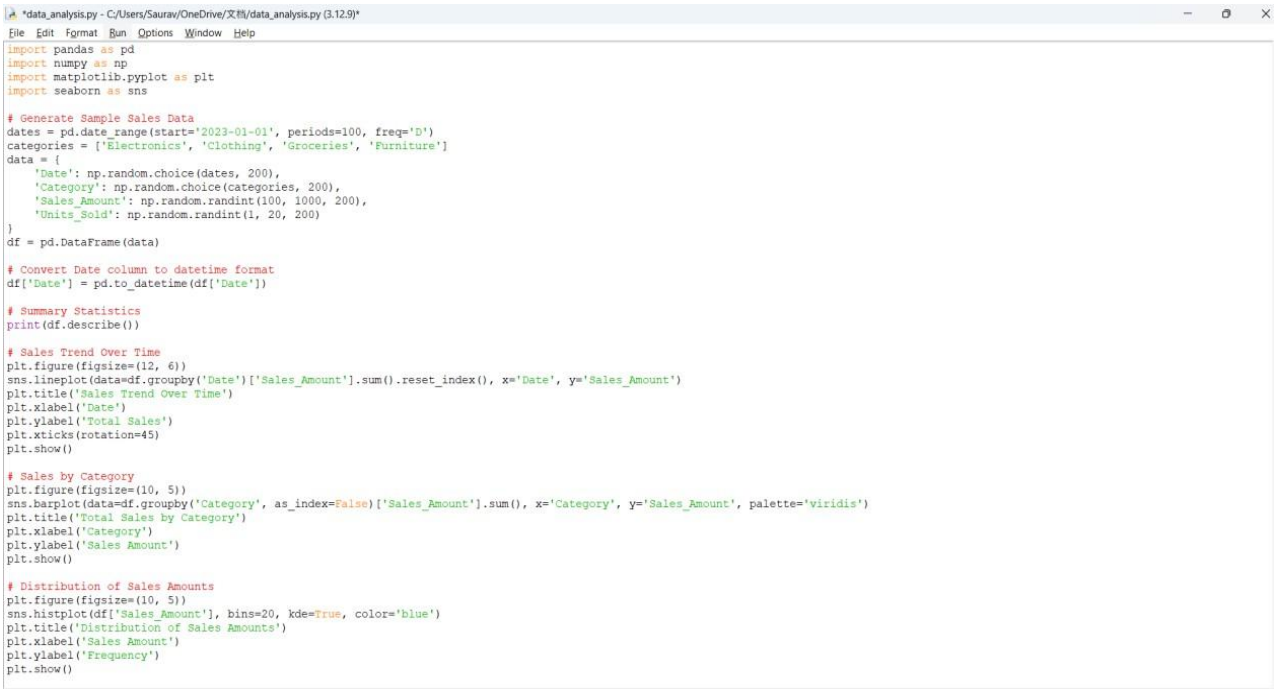
plt.xlabel('Sales Amount') plt.ylabel('Frequency') plt.show()

Scatter Plot: Sales Amount vs. Units Sold plt.figure(figsize=(10, 5))

sns.scatterplot(data=df, x='Units_Sold', y='Sales_Amount', hue='Category', palette='coolwarm')

plt.title('Sales Amount vs. Units Sold') plt.xlabel('Units Sold') plt.ylabel('Sales Amount')

plt.show()



Output:-

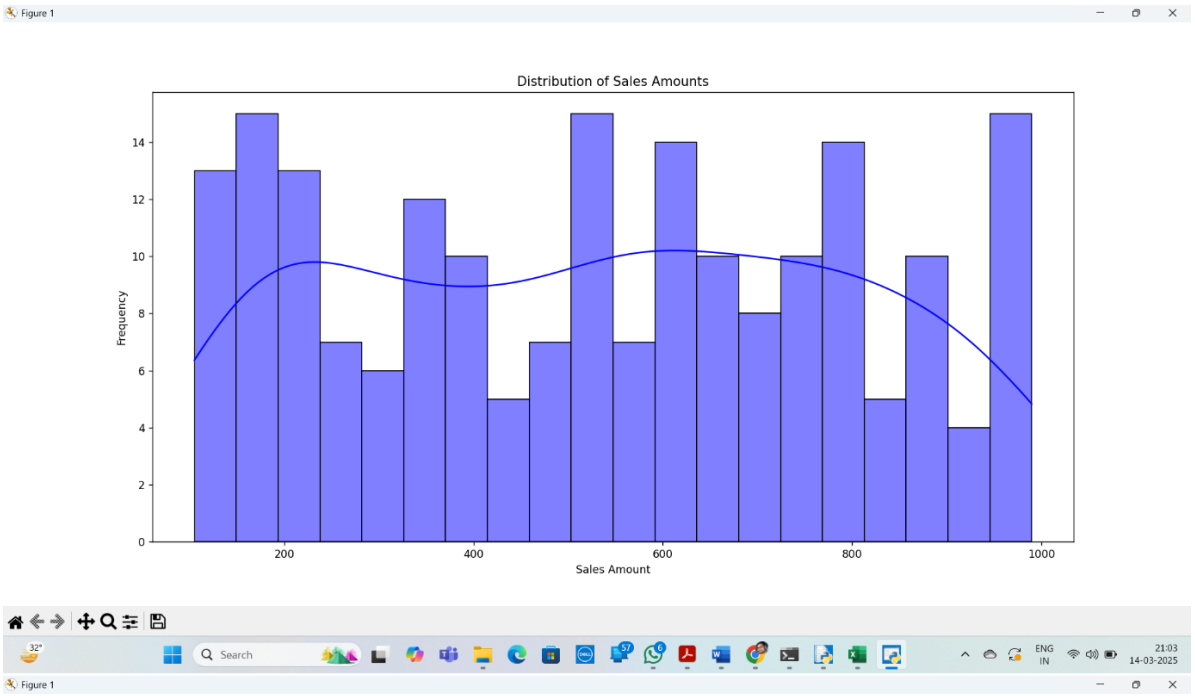


Figure 1
Distribution of Sales Amounts
Frequency
Sales Amount
0
2
4
6
8
10
12
14
200
400
600
800
1000
Search
ENG
IN
21:03
14-03-2025

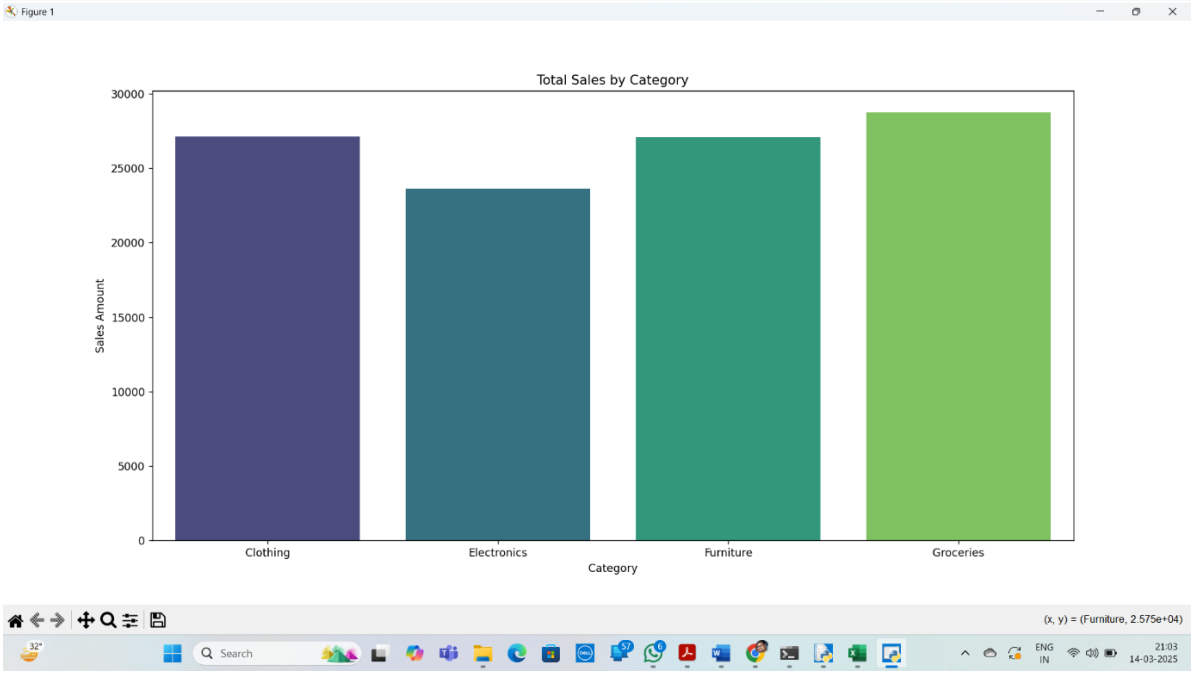


Figure 1
Total Sales by Category
Sales Amount
Category
0
5000
10000
15000
20000
25000
30000
Clothing
Electronics
Furniture
Groceries
(x, y) = (Furniture, 2.575e+04)
Search
ENG
IN
21:03
14-03-2025

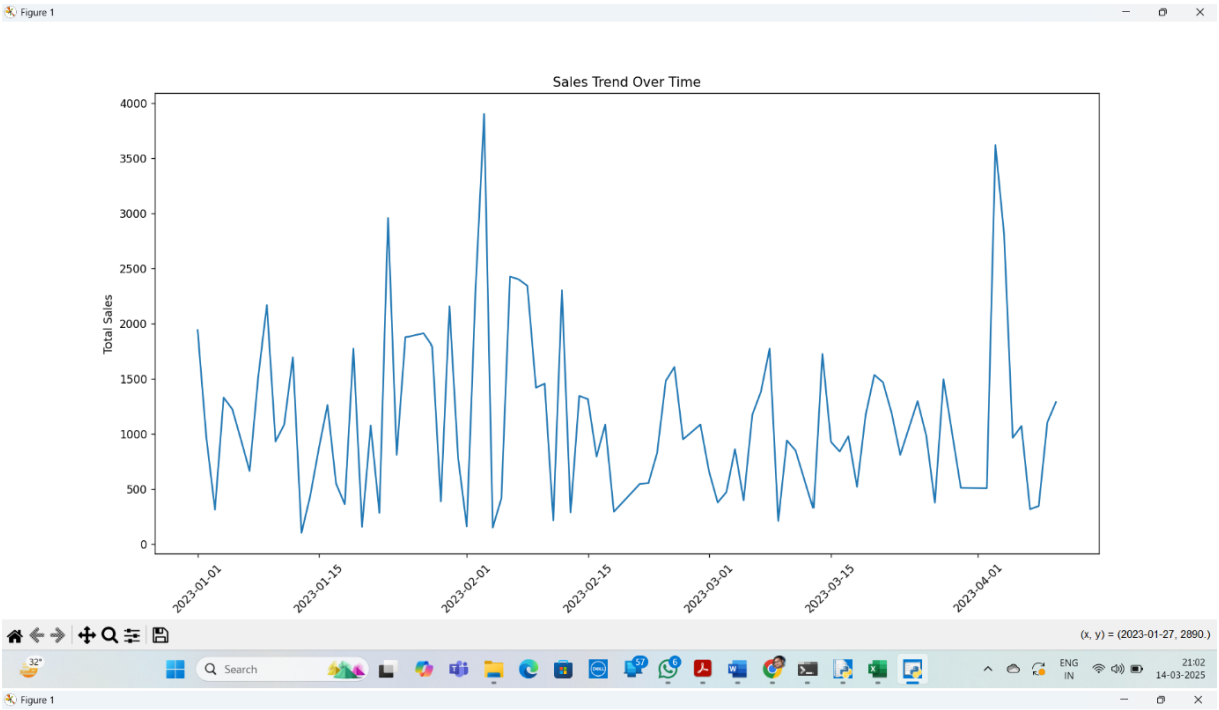


Figure 1
Sales Trend Over Time
Total Sales
4000
3500
3000
2500
2000
1500
1000
500
0
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01
(x, y) = (2023-01-27, 2890.)
Search
ENG
IN
21:02
14-03-2025

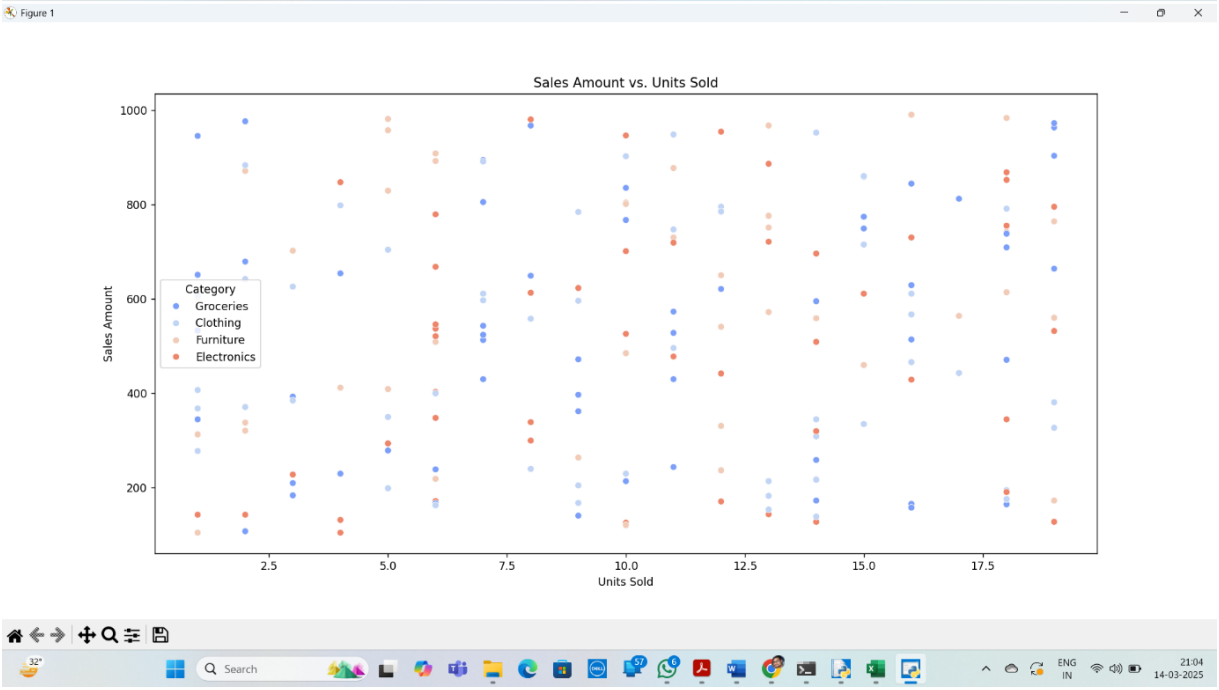


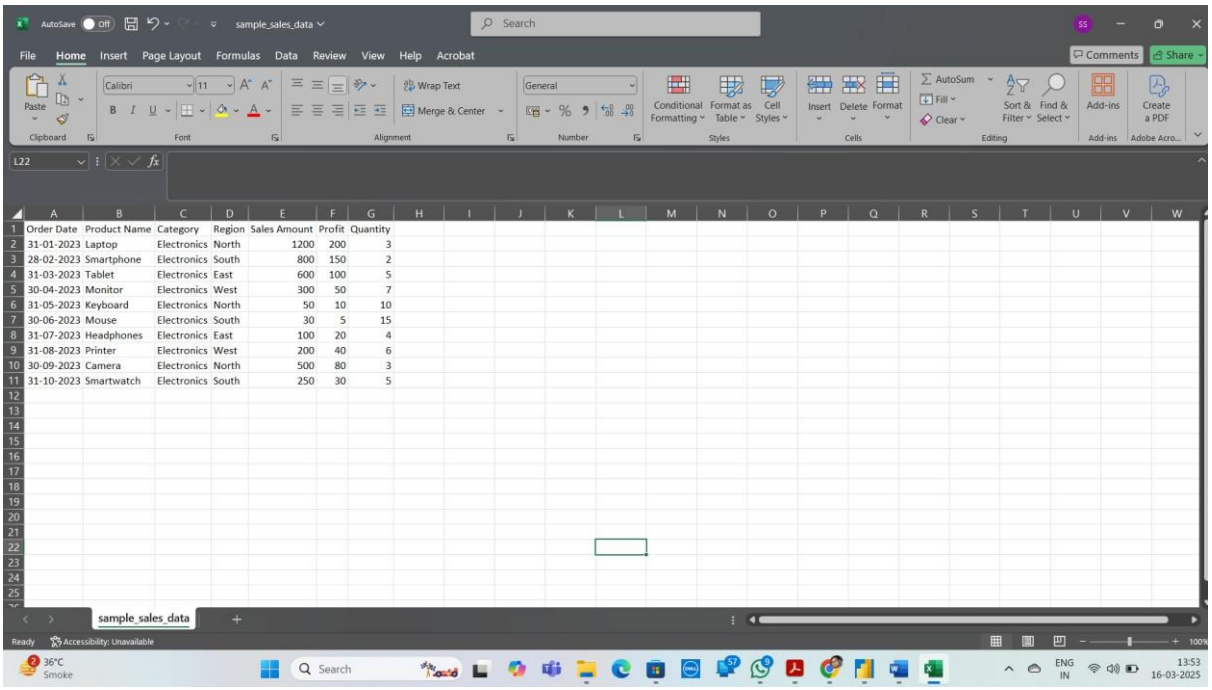
Figure 1
Sales Amount vs. Units Sold
Sales Amount
1000
800
600
400
200
Category
Groceries
Clothing
Furniture
Electronics
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Units Sold
Search
ENG
IN
21:04
14-03-2025

PRATICAL-8(b)

AIM:- Perform data visualization using PowerBI on any sales data

Steps:-

Step 1:- Download & Install Power BI Step 2:- create a sample data



Order Date	Product Name	Category	Region	Sales Amount	Profit	Quantity
31-01-2023	Laptop	Electronics	North	1200	200	3
28-02-2023	Smartphone	Electronics	South	800	150	2
31-03-2023	Tablet	Electronics	East	600	100	5
30-04-2023	Monitor	Electronics	West	300	50	7
31-05-2023	Keyboard	Electronics	North	50	10	10
30-06-2023	Mouse	Electronics	South	30	5	15
31-07-2023	Headphones	Electronics	East	100	20	4
31-08-2023	Printer	Electronics	West	200	40	6
30-09-2023	Camera	Electronics	North	500	80	3
31-10-2023	Smartwatch	Electronics	South	250	30	5

Step 3 :- Import the Sales Dataset

Click "Home" > "Get Data".

Choose your data source:

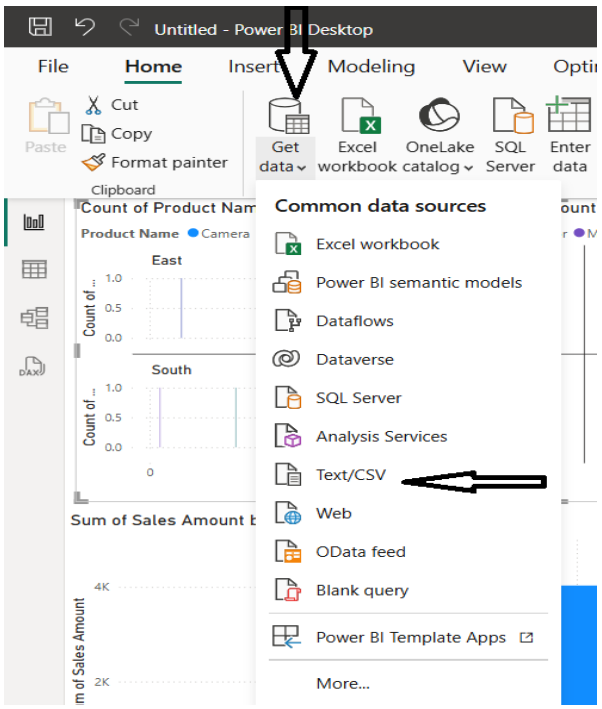
Excel (XLSX/CSV)

SQL Server

Online Services (Google Sheets, SharePoint, etc.)

Browse and select the dataset.

Click "Load" to import.



Step 4 :- Clean & Transform Data (Power Query)

Step 5 :- Create Data Visualizations

Click on **"Line Chart"** in the "Visualizations" pane.

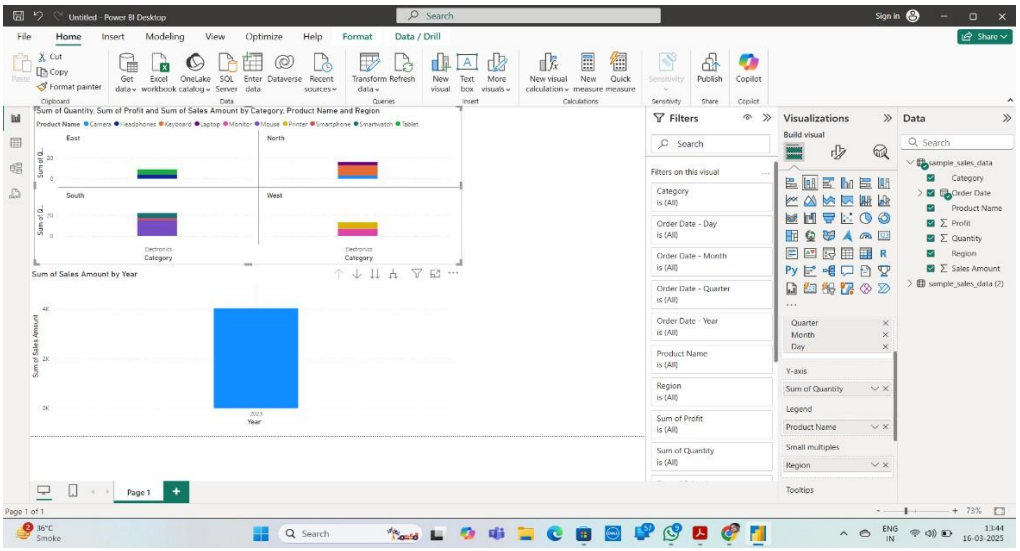
Drag **Order Date** to the **X-axis**.

Drag **Sales Amount** to the **Y-axis**.

Customize the chart (format labels, add title, etc.).

Step 6:- Add Filters & Interactivity

Output:-



Practical No 9

AIM: Create the Data staging area for the selected database using SQL

STEPS:-

Step 1:- Create a Staging Database

First, create a staging database to store raw sales data.

CREATE DATABASE Sales_Staging; USE Sales_Staging**;**

Step 2:- Create Staging Tables

Create tables that match the structure of raw sales data but include additional fields like **load date**

and **batch ID**.

CREATE TABLE Staging_Sales (SalesID INT PRIMARY KEY,

OrderDate DATE,

ProductName VARCHAR(100), Category VARCHAR(50),

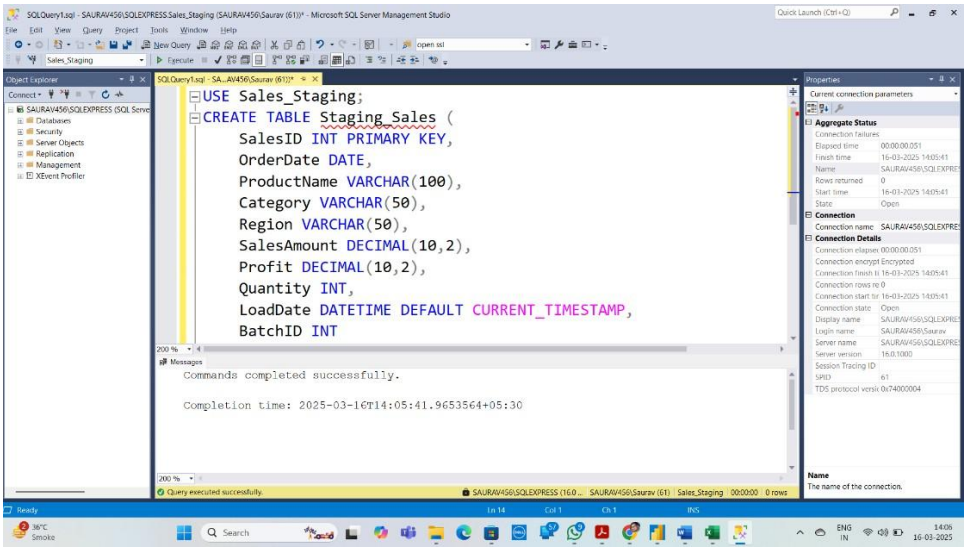
Region VARCHAR(50),

SalesAmount DECIMAL(10,2), Profit DECIMAL(10,2),

Quantity INT,

LoadDate DATETIME DEFAULT CURRENT_TIMESTAMP,

BatchID INT);



Step 3:- Load Raw Data into the Staging Table

Simulating data load from a **CSV file, API, or external source**:

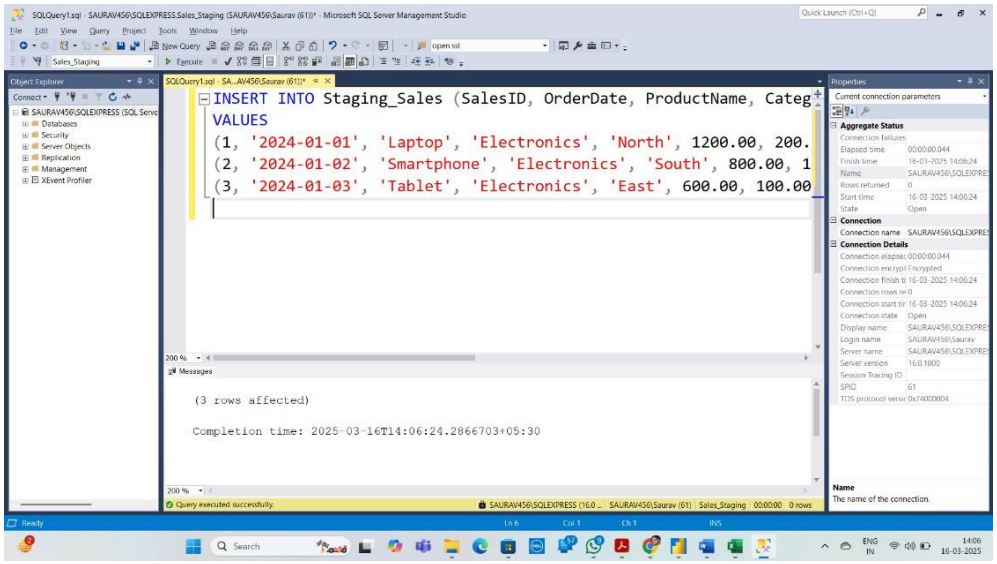
INSERT INTO Staging_Sales (SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity, BatchID)

VALUES

(1, '2024-01-01', 'Laptop', 'Electronics', 'North', 1200.00, 200.00, 3, 101),

(2, '2024-01-02', 'Smartphone', 'Electronics', 'South', 800.00, 150.00, 2, 101),

(3, '2024-01-03', 'Tablet', 'Electronics', 'East', 600.00, 100.00, 5, 101);



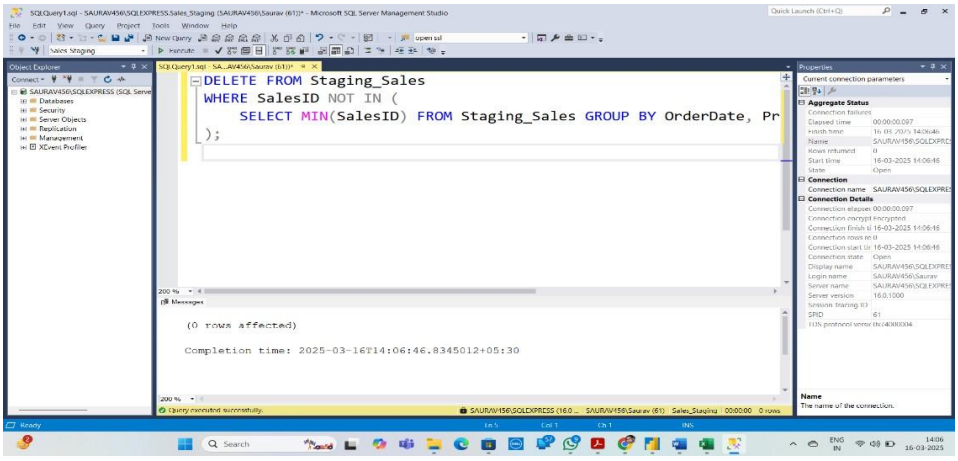
Step 4:- Perform Data Cleansing & Transformation

Remove Duplicates

DELETE FROM Staging_Sales WHERE SalesID NOT IN (

SELECT MIN(SalesID) FROM Staging_Sales GROUP BY OrderDate, ProductName, Region

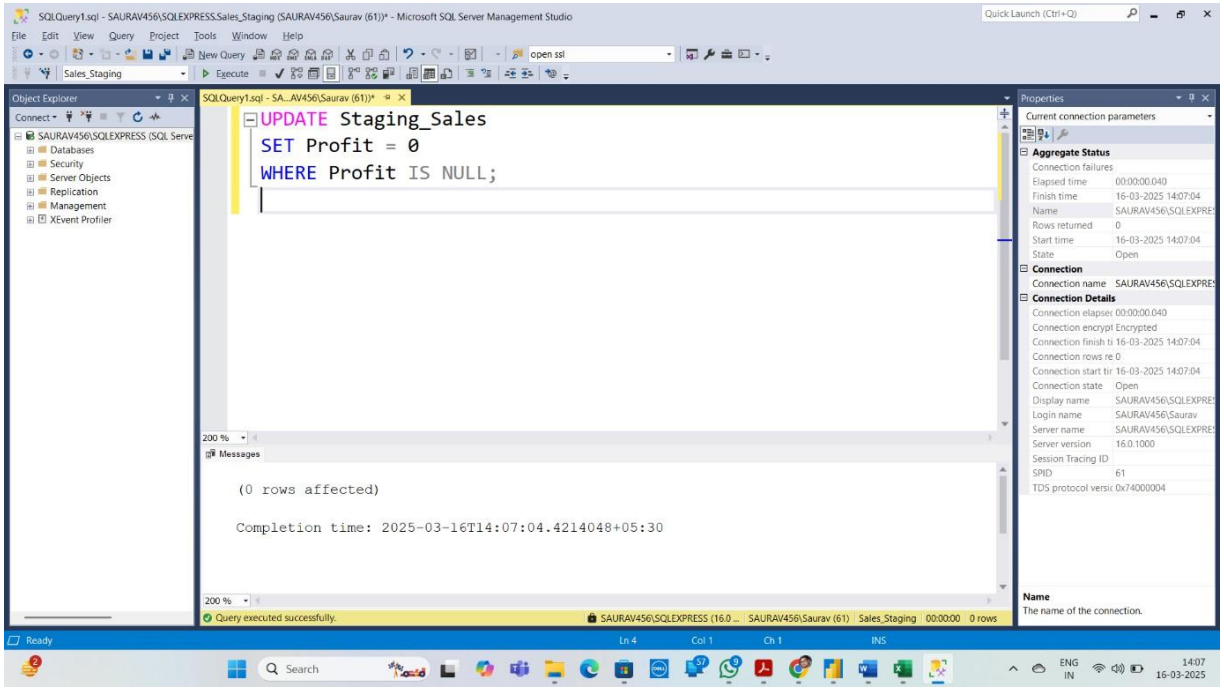
);



Handle Null Values

UPDATE Staging_Sales SET Profit = 0

WHERE Profit IS NULL;



Step 5:- Transfer Clean Data to the Final Sales Table

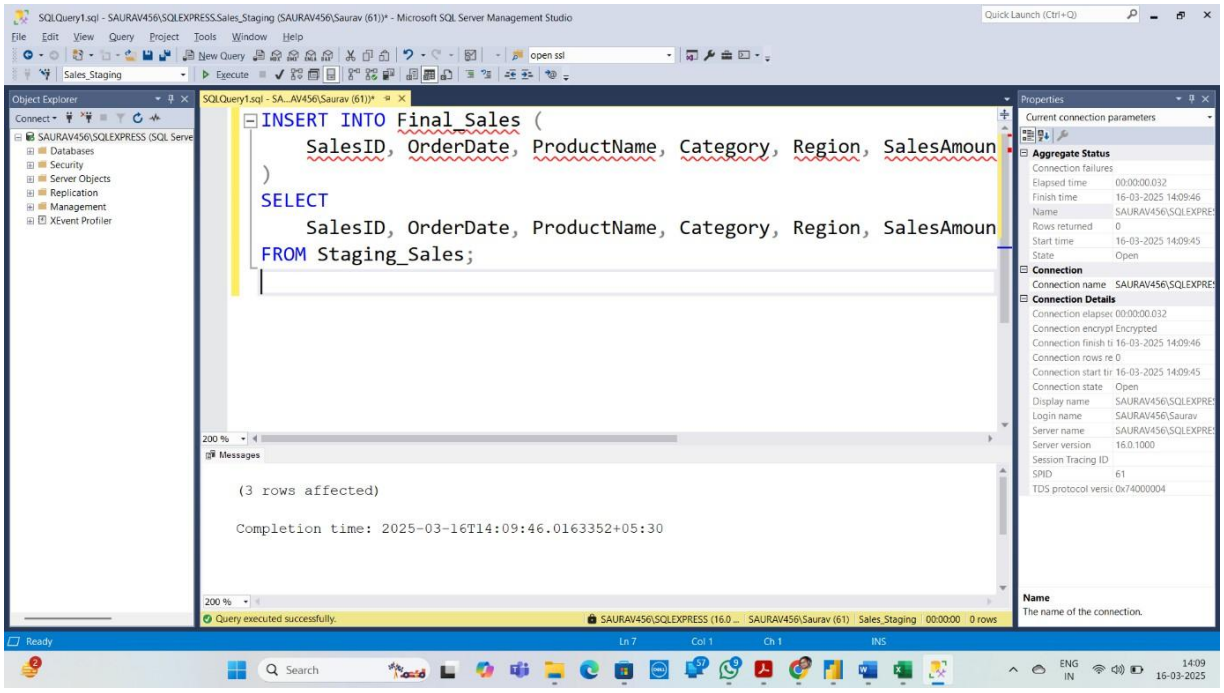
Move the cleaned data into the Data Warehouse (DWH).

INSERT INTO Final_Sales (

SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity

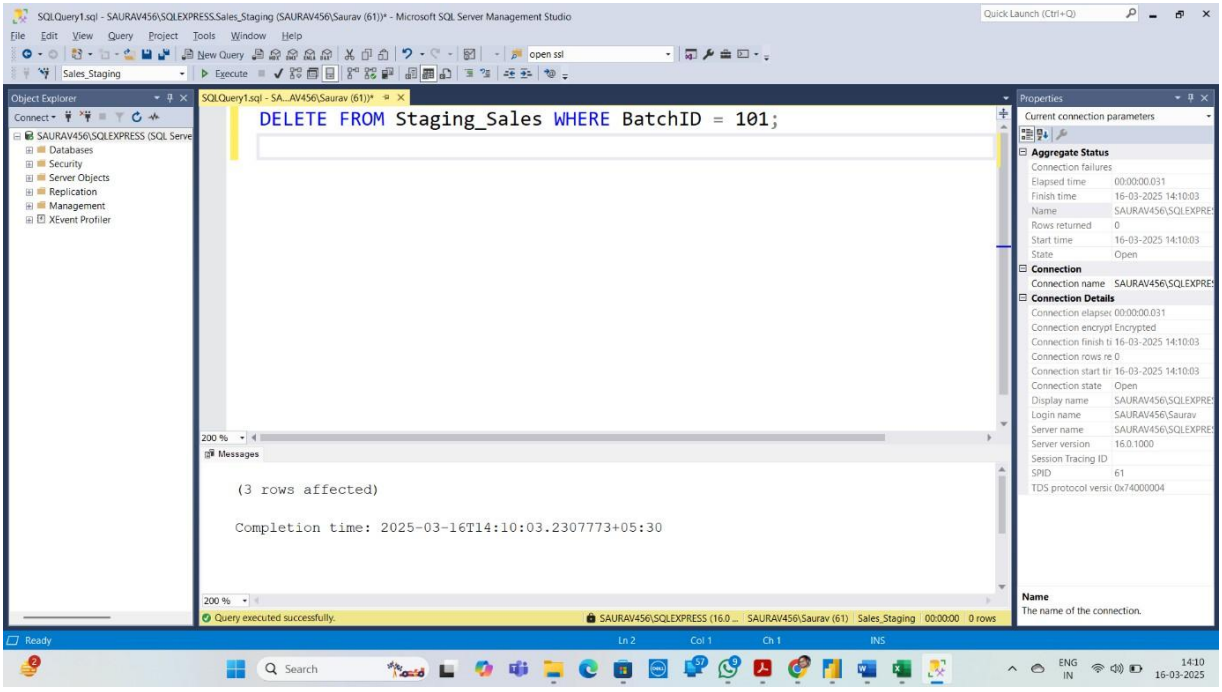
) SELECT

SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity FROM Staging_Sales;



Step 6:- Archive or Delete Processed Data

Once data is loaded, either archive it or delete it from the staging area.



DELETE FROM Staging_Sales WHERE BatchID = 10

Practical No.10

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

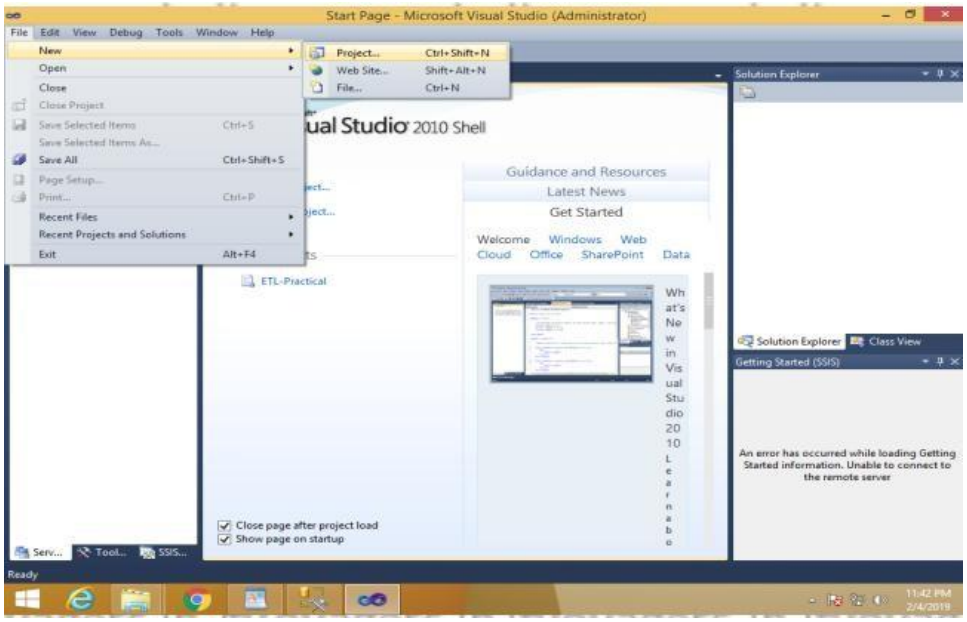
Steps:-

Step 1: Creating Data Warehouse

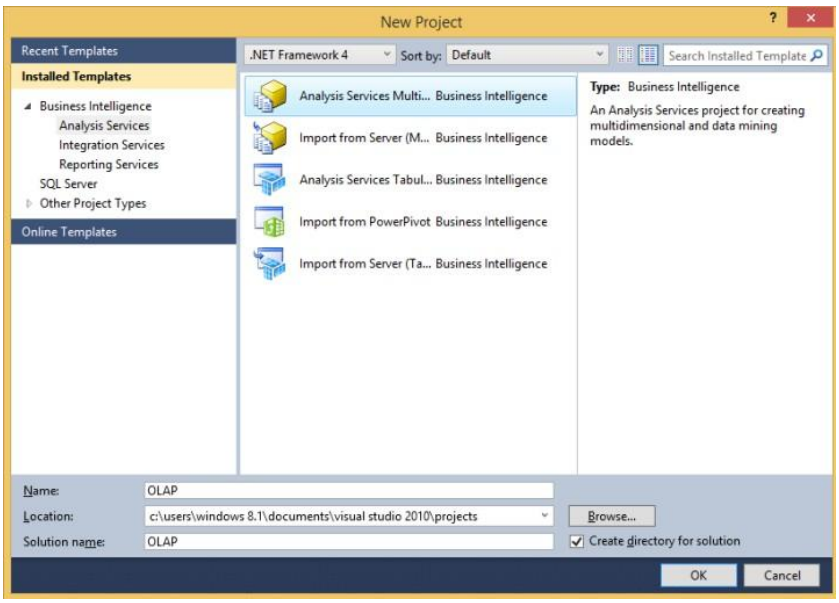
CREATED IN PRACTICAL 9

Step 2: Start SSDT environment and create New Data Source

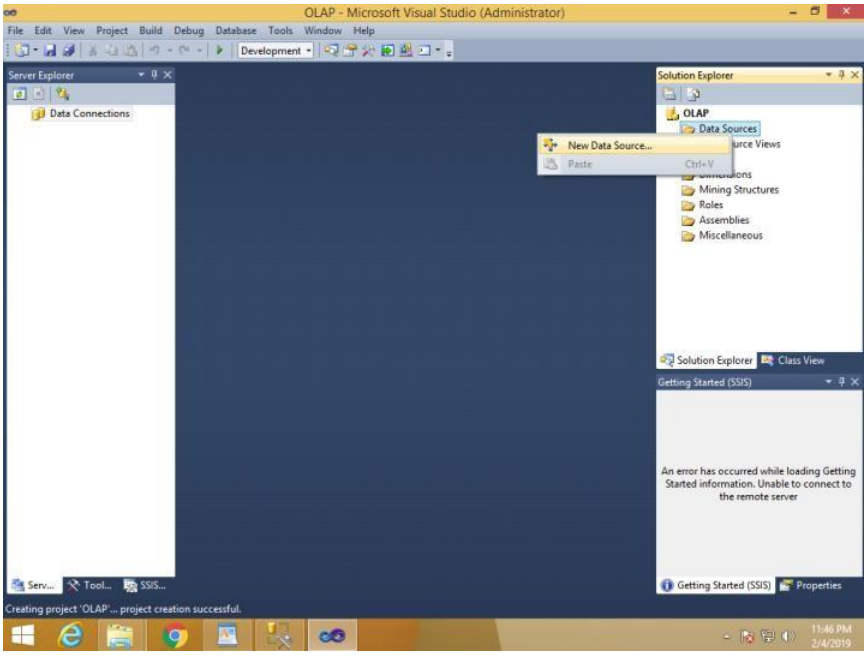
Go to Sql Server Data Tools --> Right click and run as administrator Click on File → New → Project



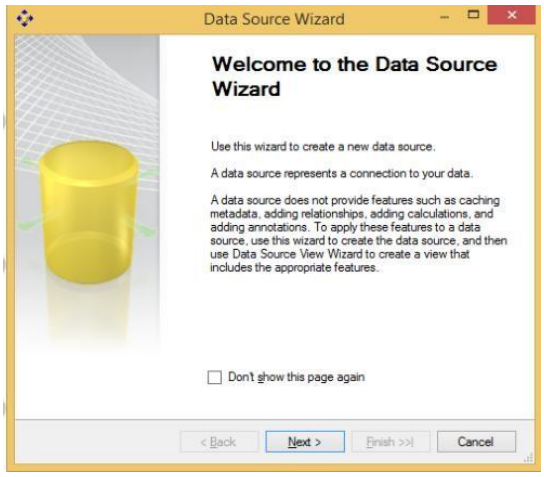
In Business Intelligence → Analysis Services Multidimensional and Data Mining models → appropriate project name → click OK



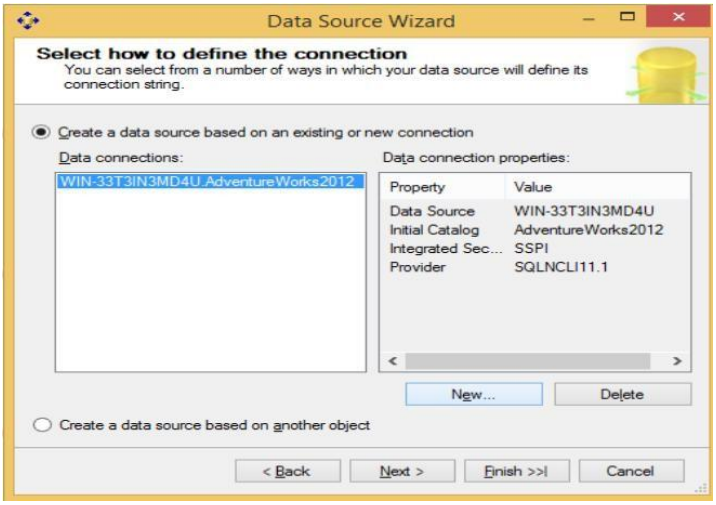
Right click on Data Sources in solution explorer → New Data Source



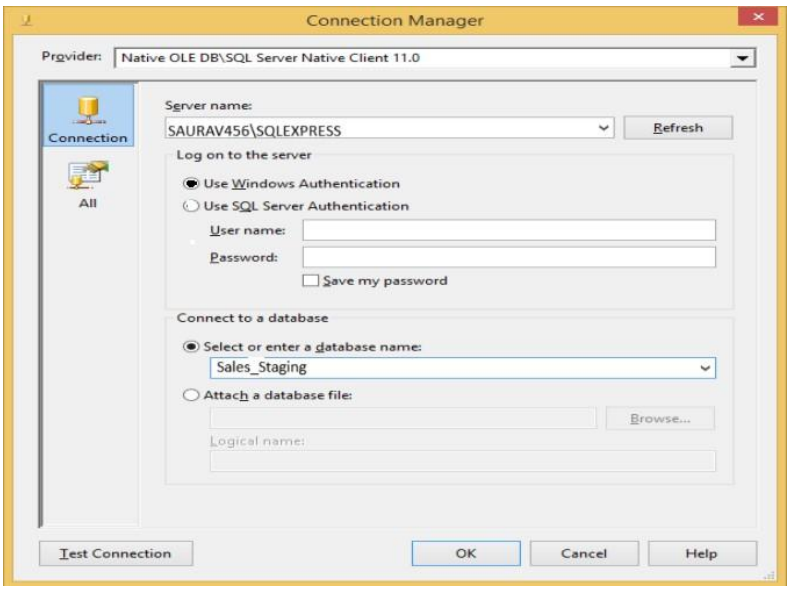
Data Source Wizard appears

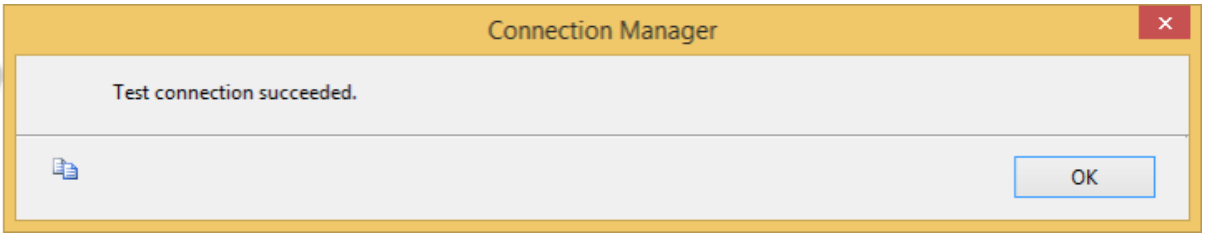


Click on New

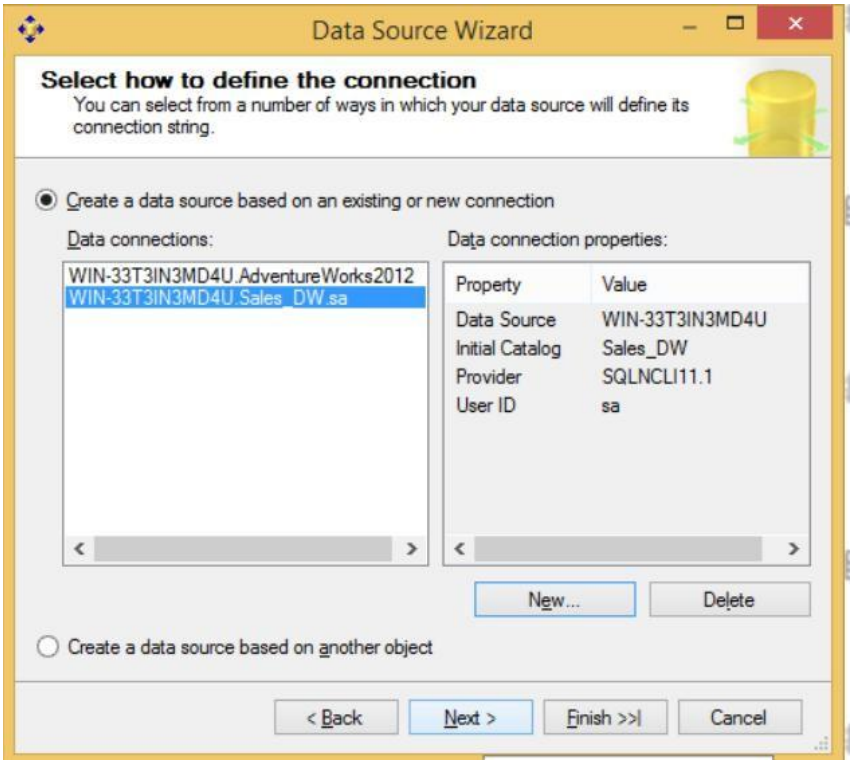


Select Server Name → select Use SQL Server Authentication → Select or enter a database name (Sales_Staging)

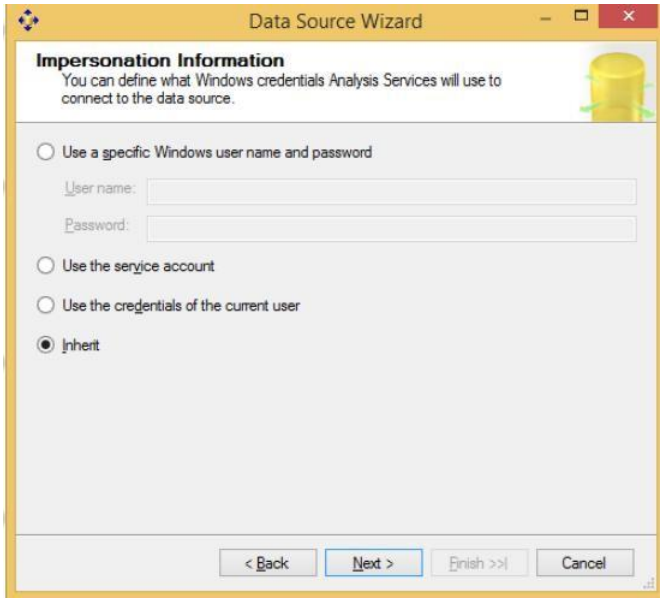




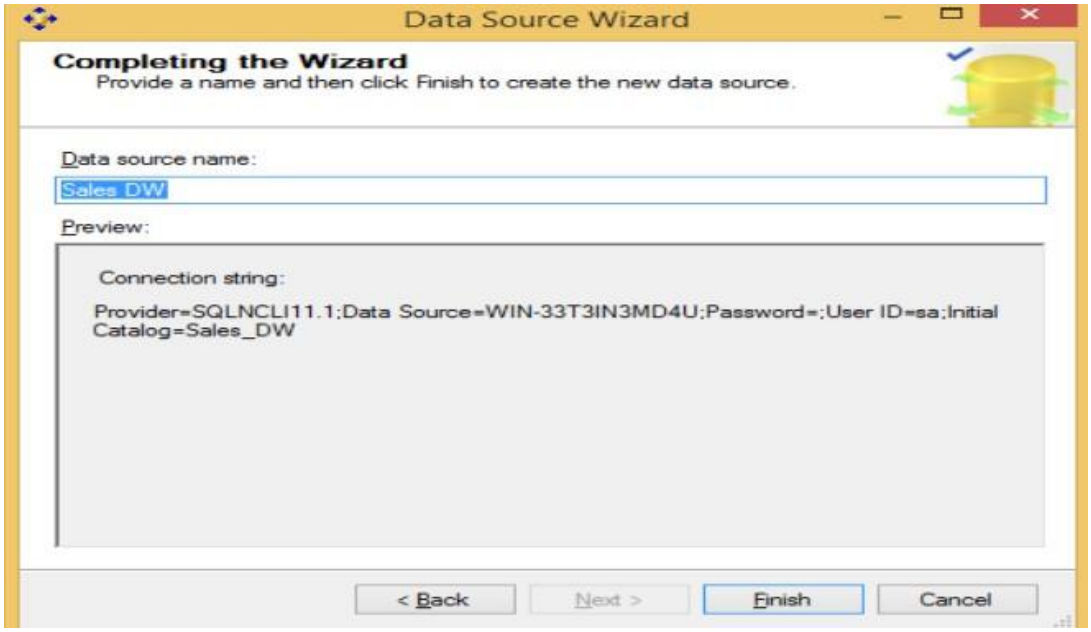
Click ok



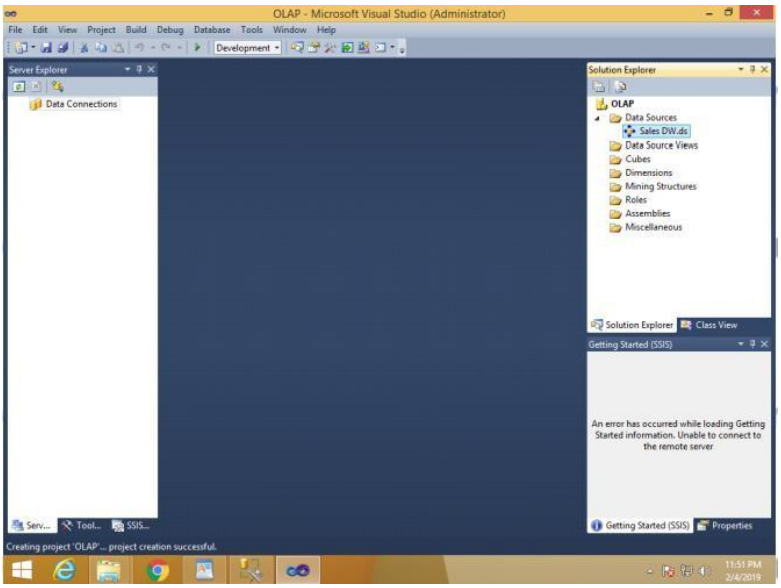
Select Inherit → Next



Click Finish

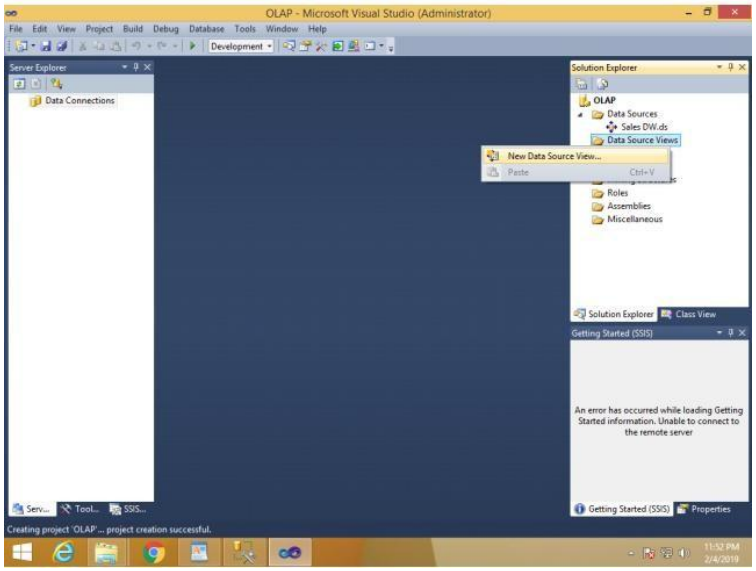


Sales_Staging.ds gets created under Data Sources in Solution Explorer

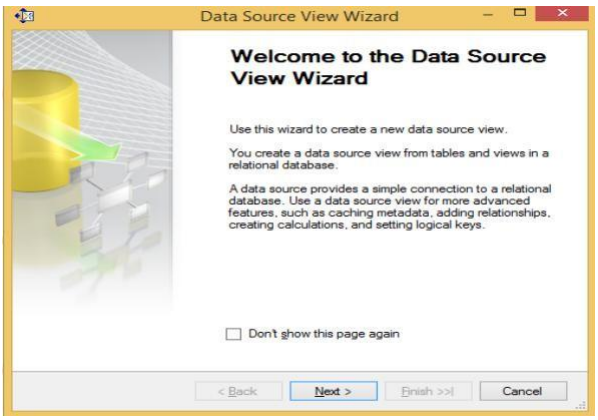


Step 3: Creating New Data Source View

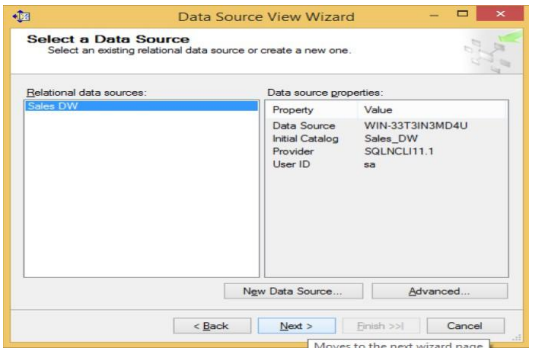
In Solution explorer right click on Data Source View → Select New Data Source View



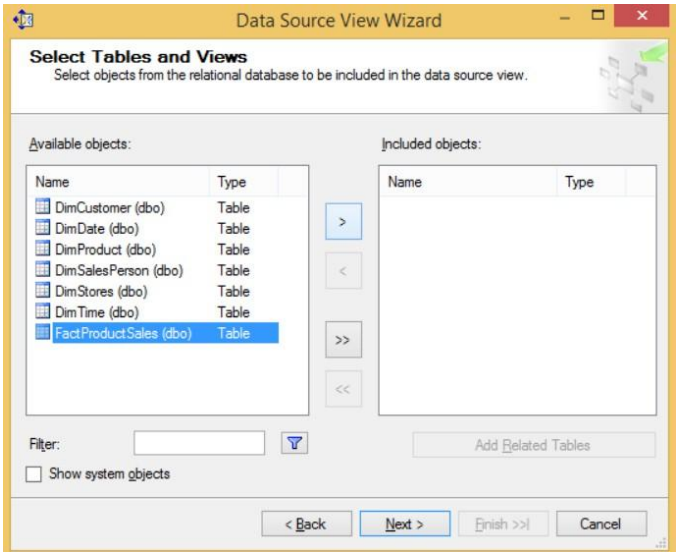
Click Next

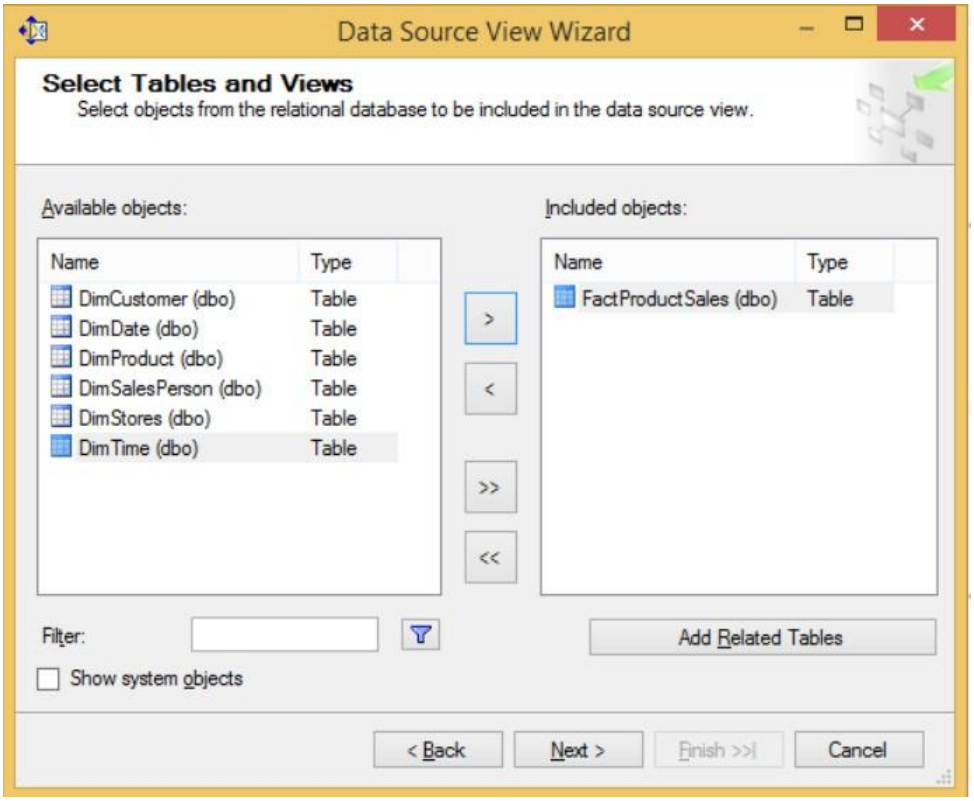


Click Next

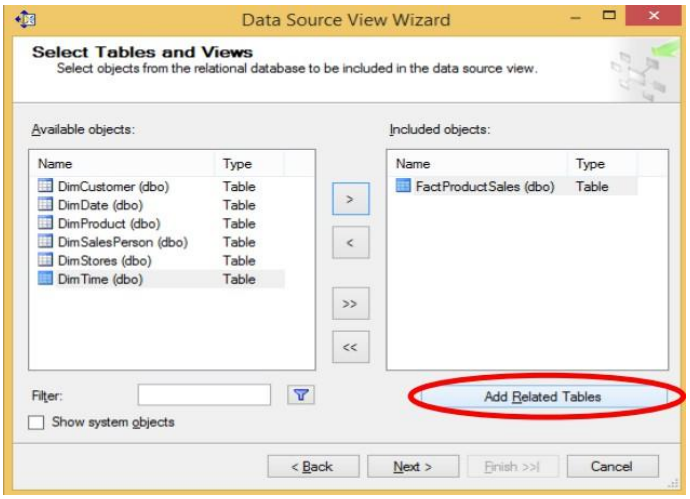


Select FactProductSales(dbo) from Available objects and put in Includes Objects by clicking on

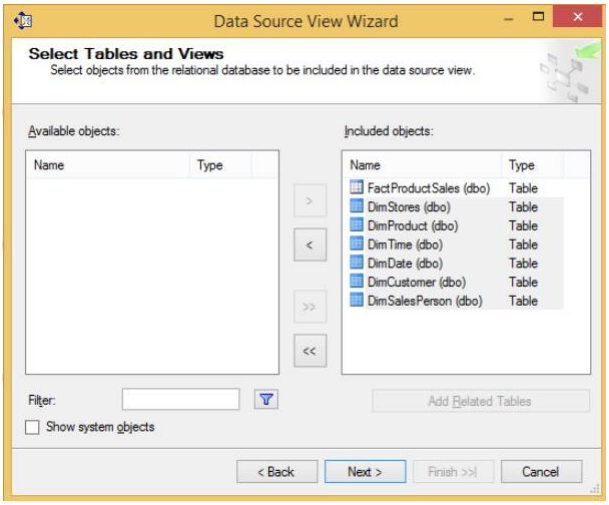

Data Source View Wizard
Select Tables and Views
Select objects from the relational database to be included in the data source view.
Available objects:
Name
Type
DimCustomer (dbo)
Table
DimDate (dbo)
Table
DimProduct (dbo)
Table
DimSalesPerson (dbo)
Table
DimStores (dbo)
Table
DimTime (dbo)
Table
FactProductSales (dbo)
Table
Included objects:
Name
Type
>
<
>>
<<
Filter:
Add Related Tables
Show system objects
< Back
Next >
Finish >>|
Cancel


Data Source View Wizard
Select Tables and Views
Select objects from the relational database to be included in the data source view.
Available objects:
Name
Type
DimCustomer (dbo)
Table
DimDate (dbo)
Table
DimProduct (dbo)
Table
DimSalesPerson (dbo)
Table
DimStores (dbo)
Table
DimTime (dbo)
Table
Included objects:
Name
Type
FactProductSales (dbo)
Table
>
<
>>
<<
Filter:
Add Related Tables
Show system objects
< Back
Next >
Finish >>|
Cancel

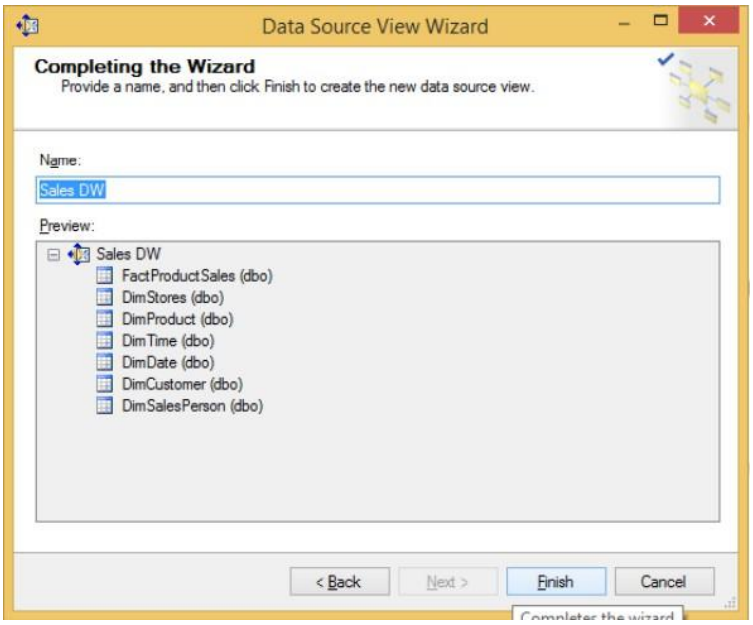
Click on Add Related Tables



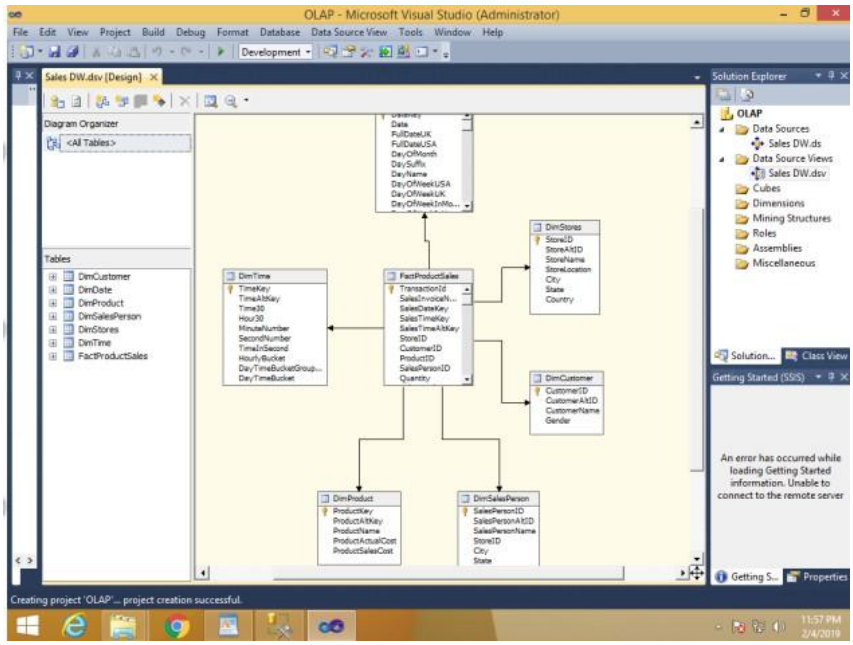
Click Next



Click Finish

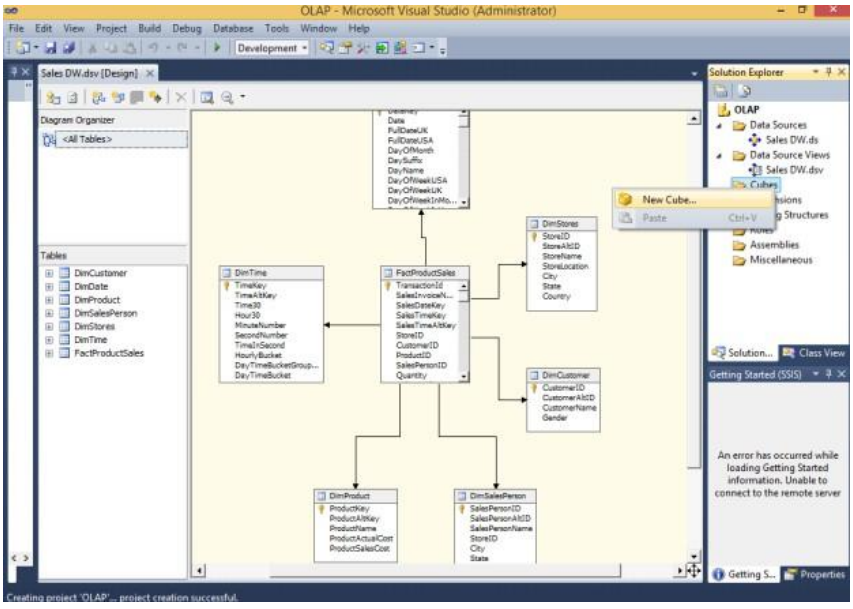


Sales_Staging.dsv appears in Data Source Views in Solution Explorer.

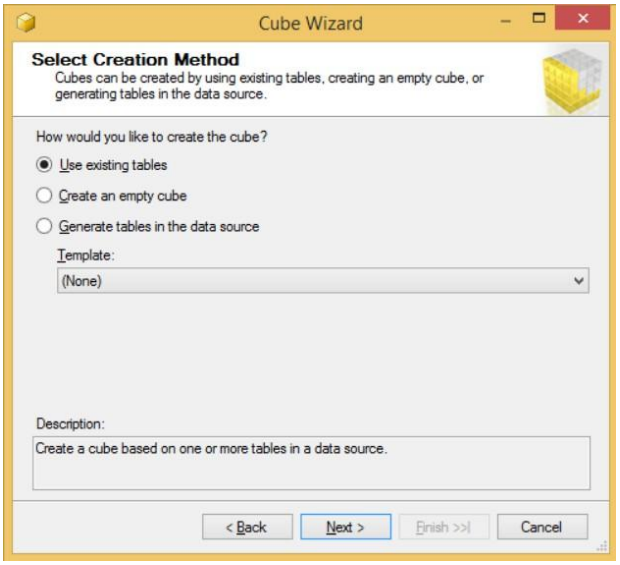


Step 4: Creating new cube

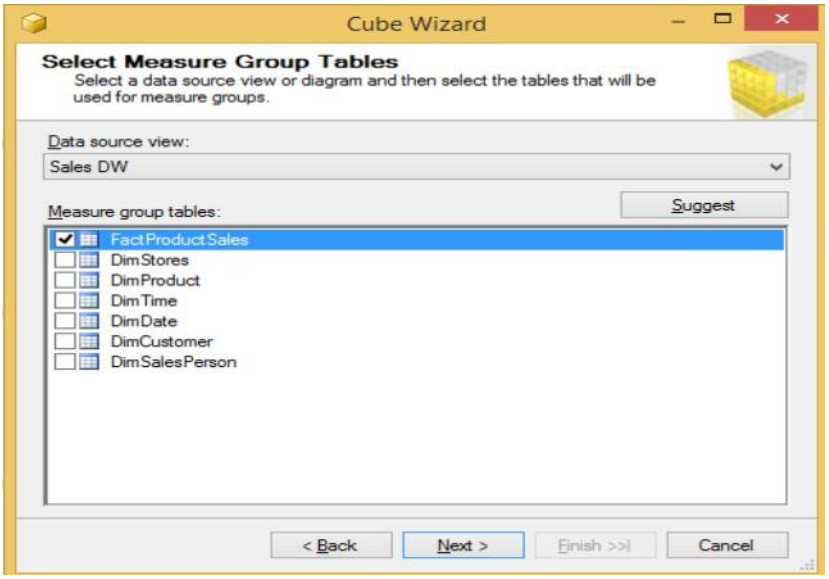
Right click on Cubes → New Cube



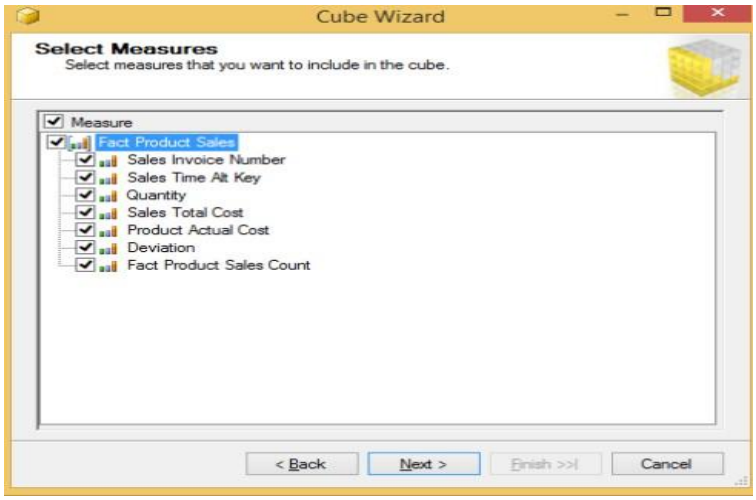
Select Use existing tables in Select Creation Method → Next



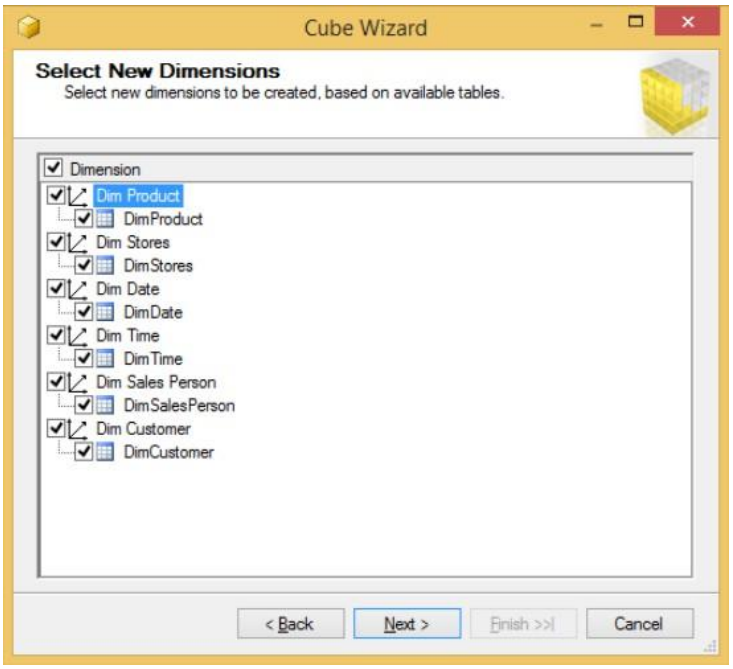
In Select Measure Group Tables → Select FactProductSales → Click Next



In Select Measures → check all measures → Next

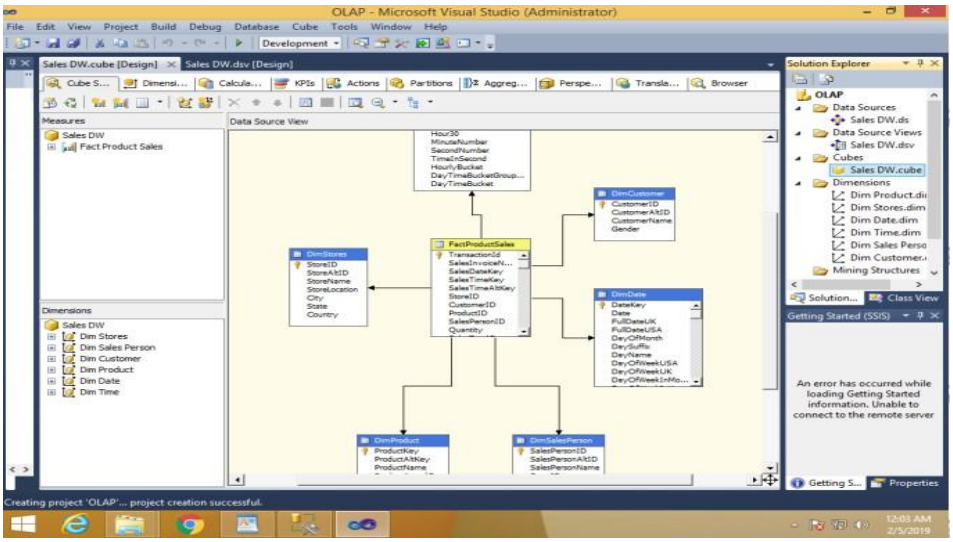


In Select New Dimensions → Check all Dimensions → Next



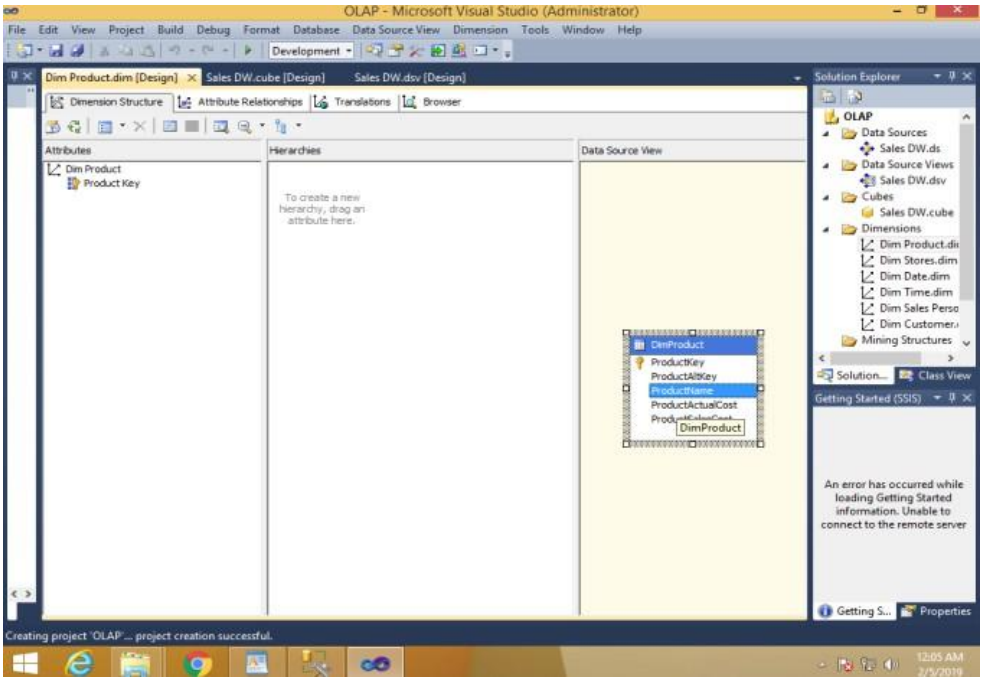
Click on Finish

Sales_Staging.cube is created

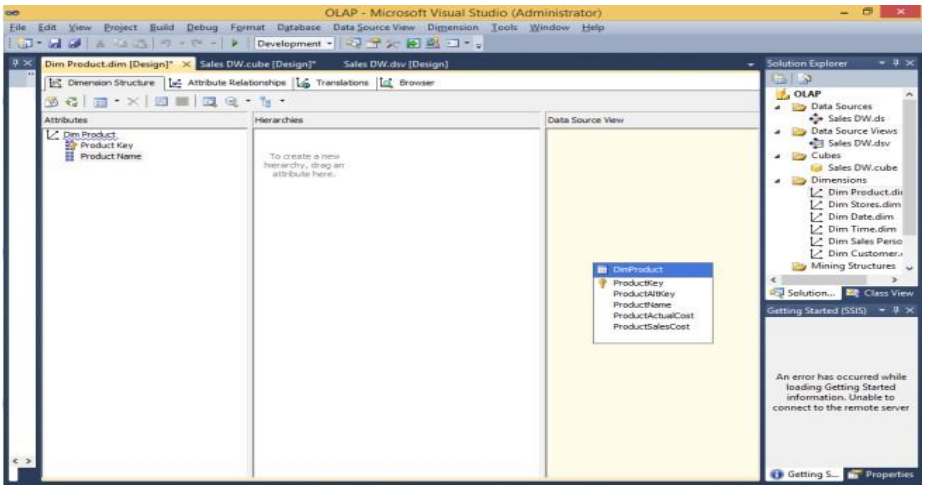


Step 5: Dimension Modification

In dimension tab → Double Click Dim Product.dim

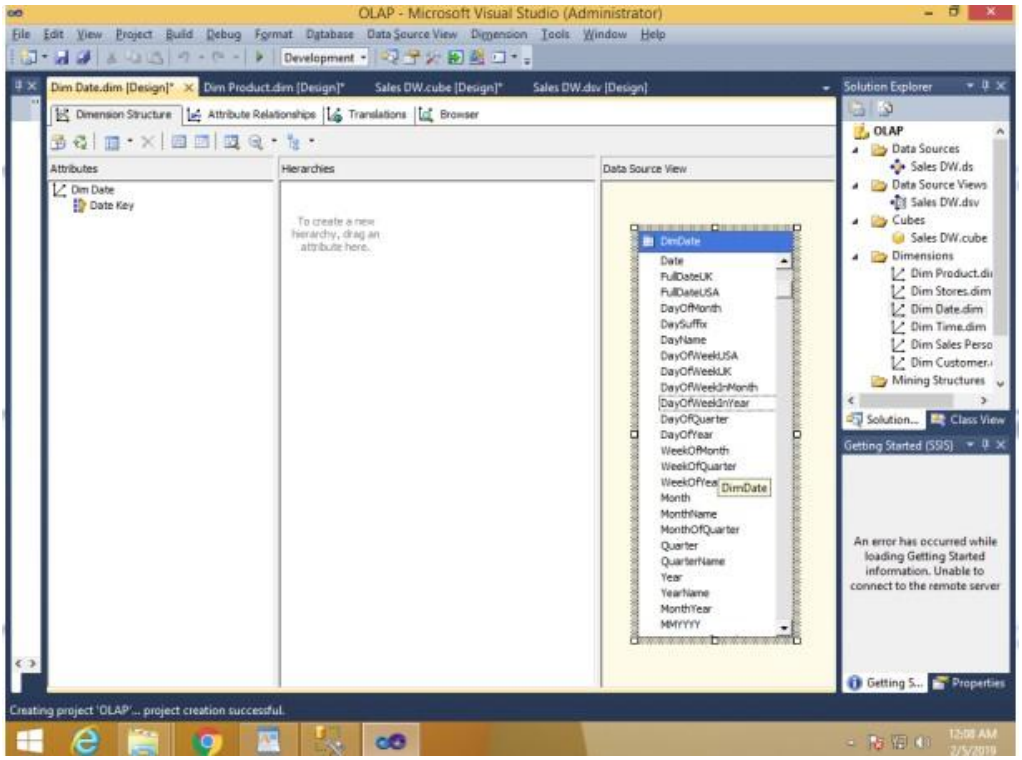


Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side

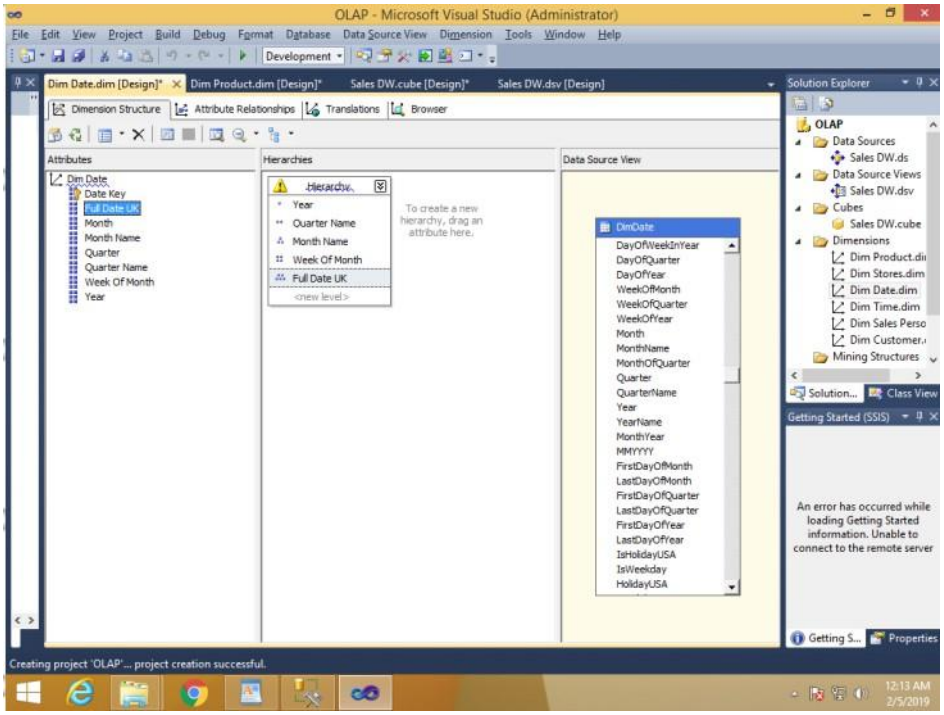


Step 6: Creating Attribute Hierarchy in Date Dimension

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.

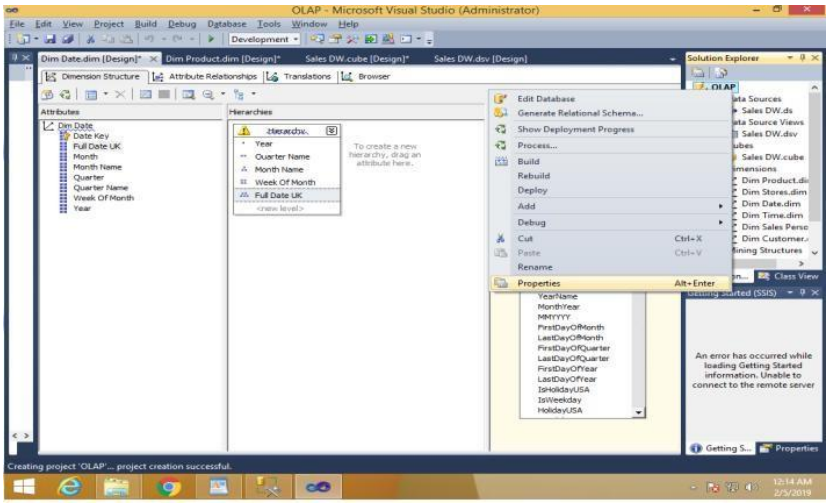


Drag fields in sequence from Attributes to Hierarchy window (Year, Quarter Name, Month Name, Week of the Month, Full D

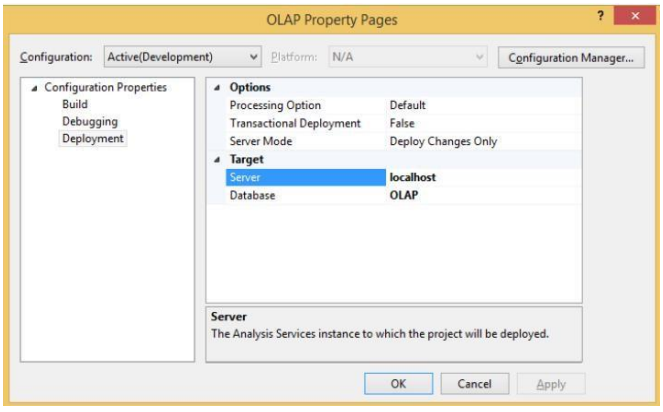


Step 7: Deploy Cube .

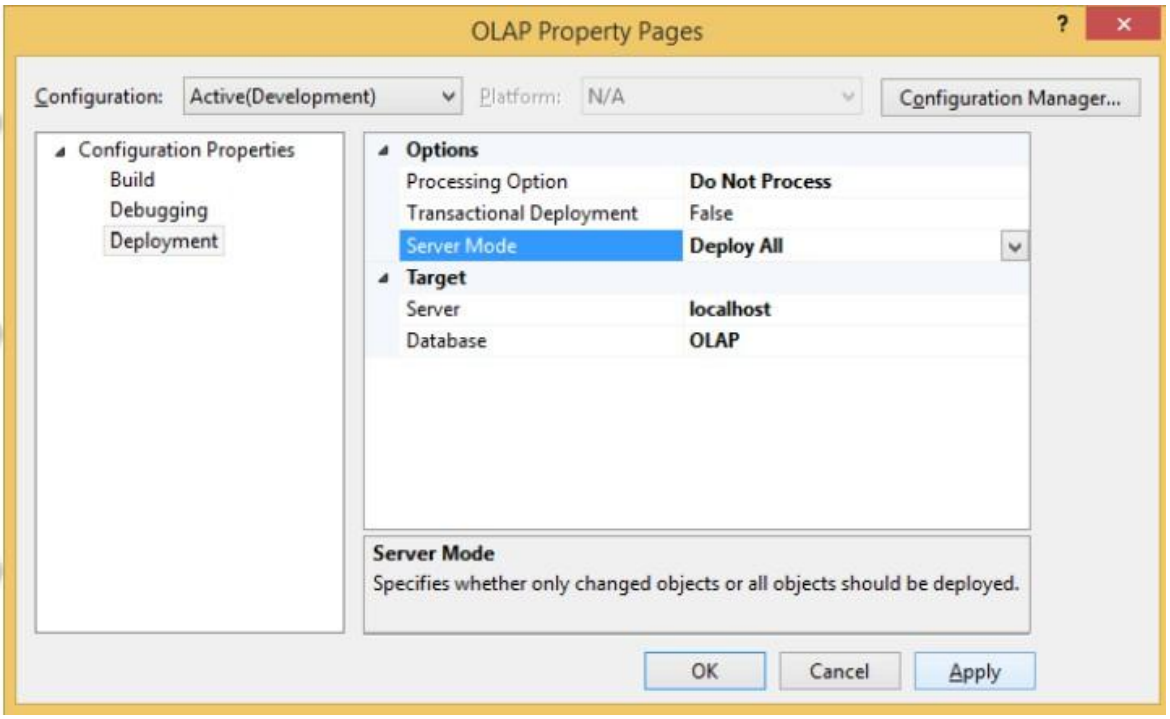
Right click on Project name → Properties



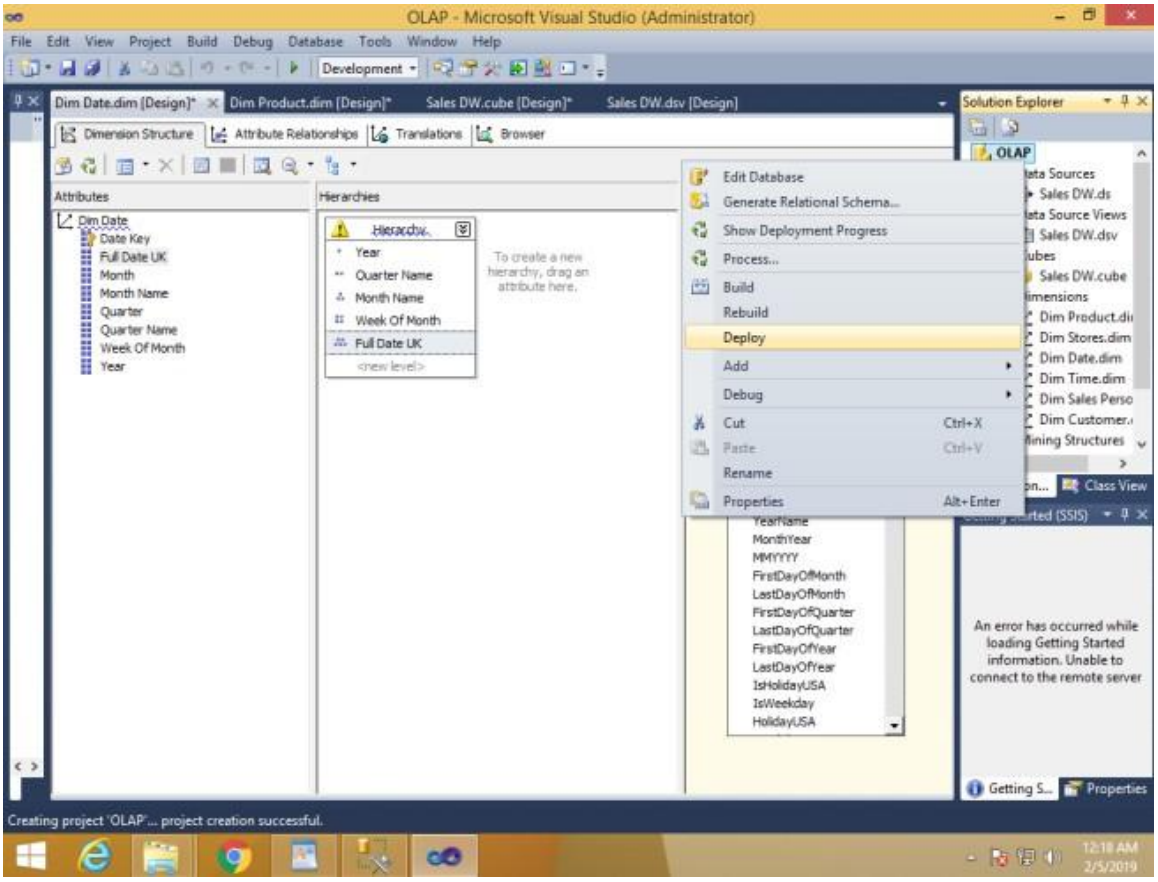
This window appaers



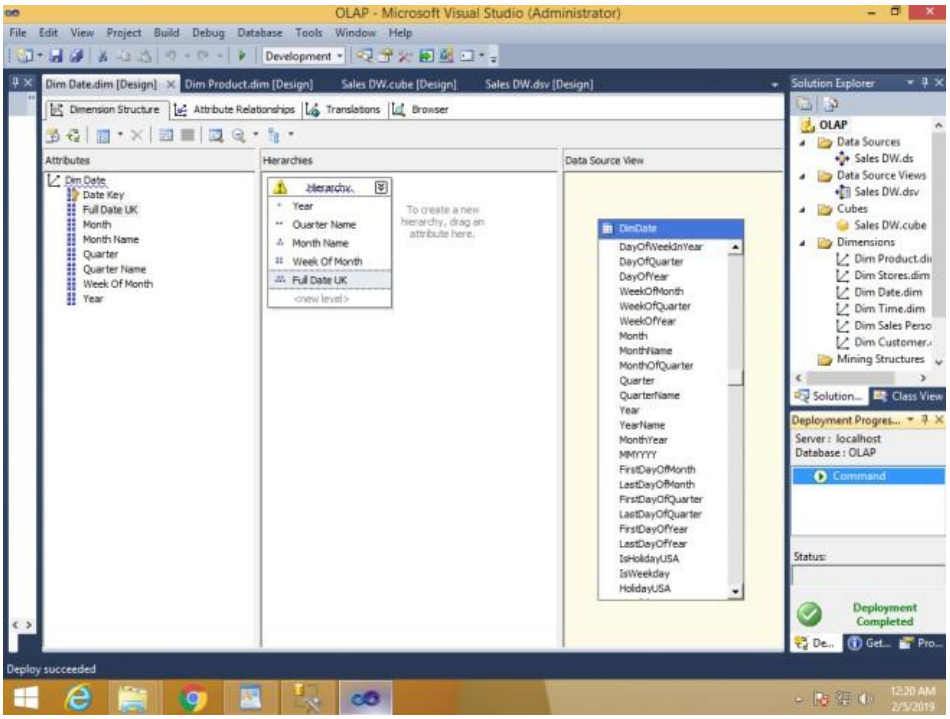
Do following changes and click on Apply & ok



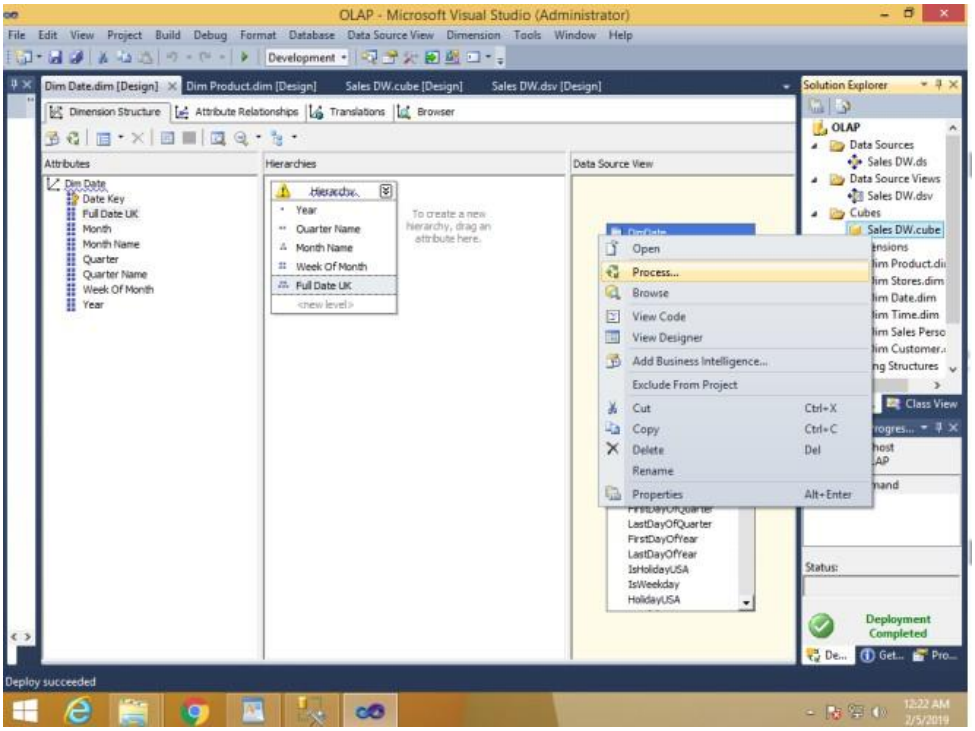
Right click on project name → Deploy



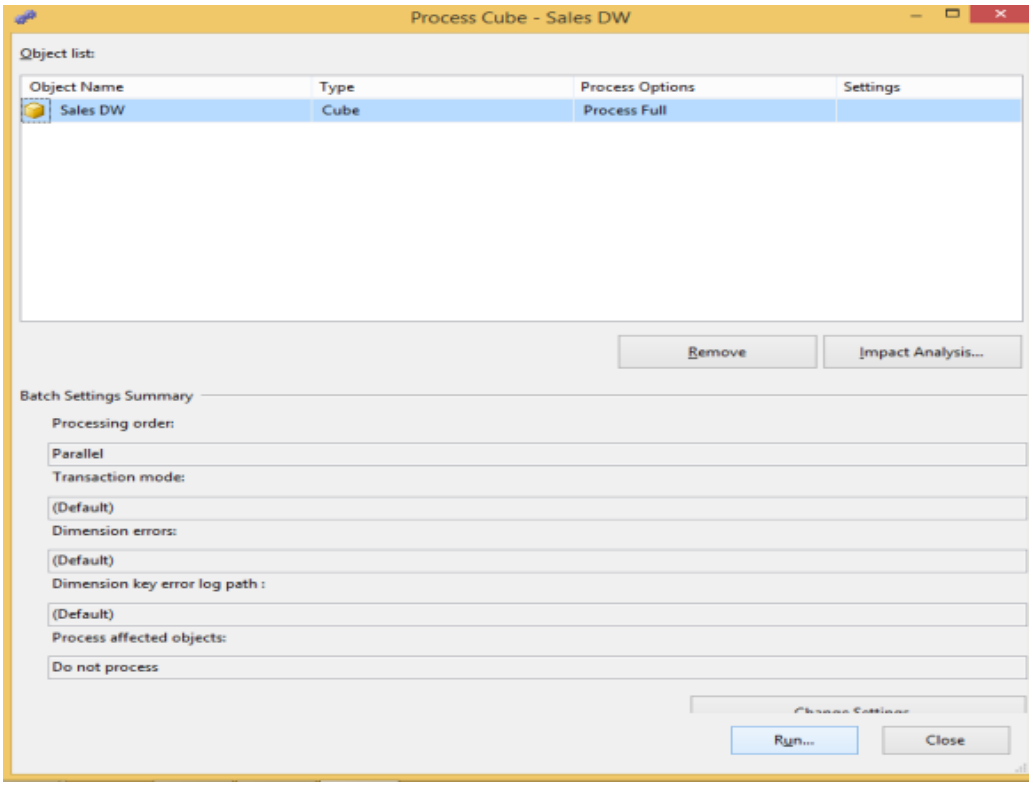
Deployment successful

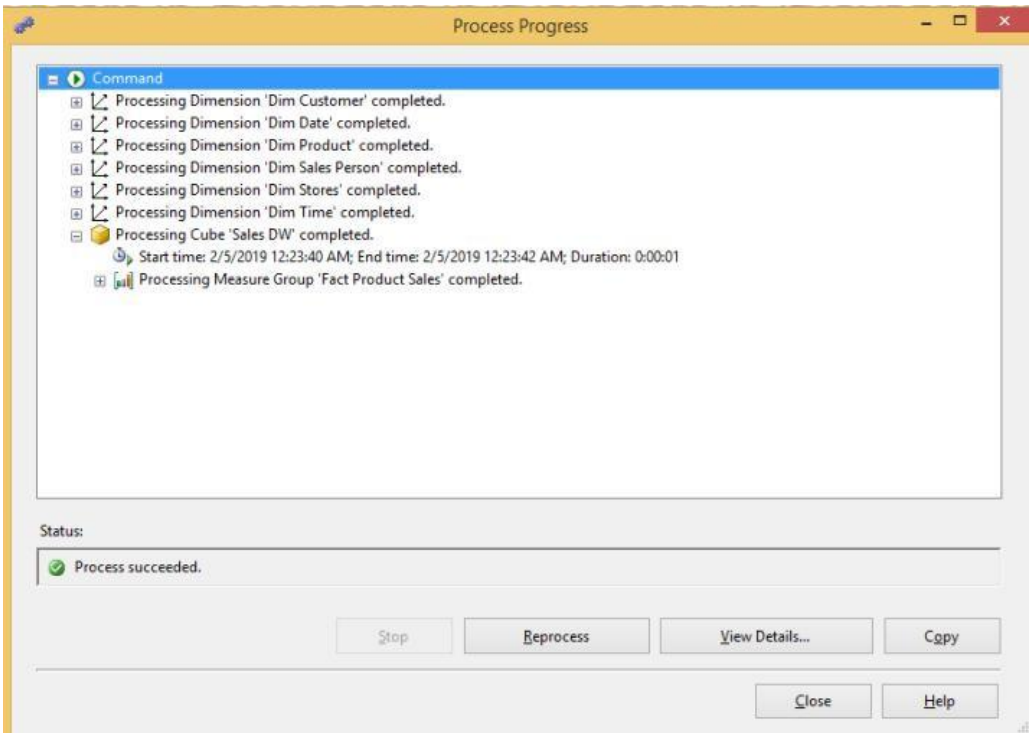


To process cube right click on Sales_DW.cube → Process

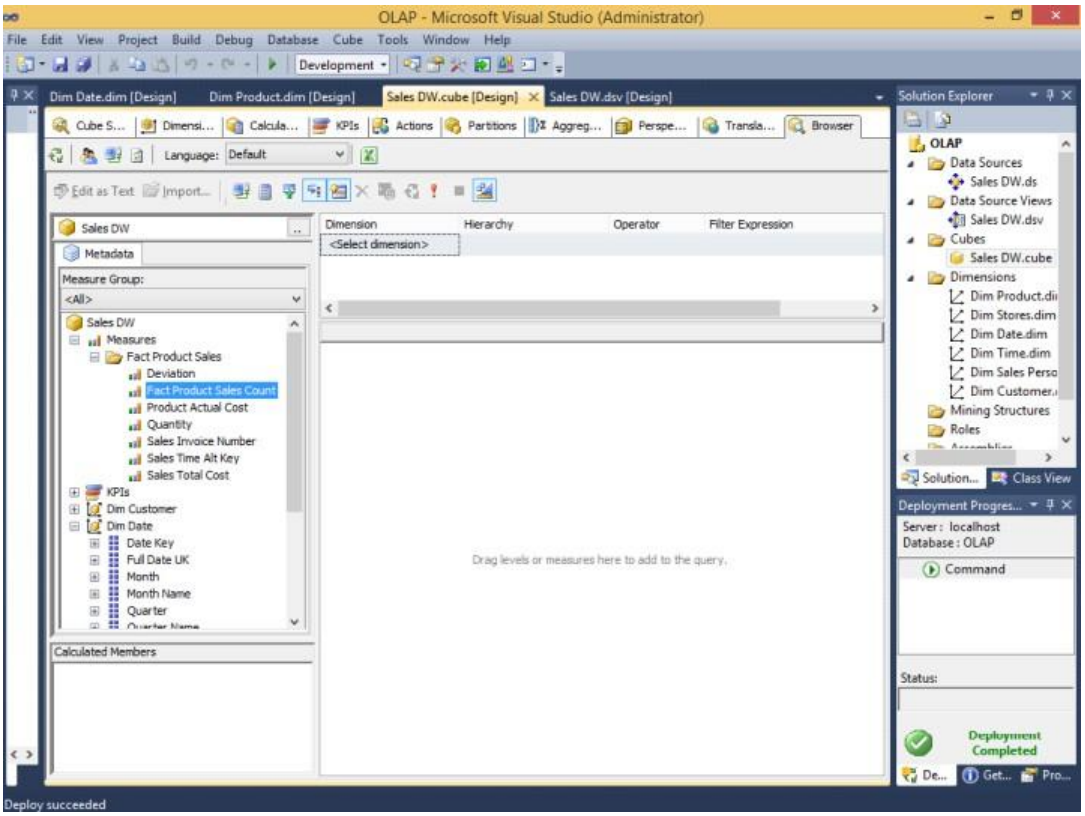


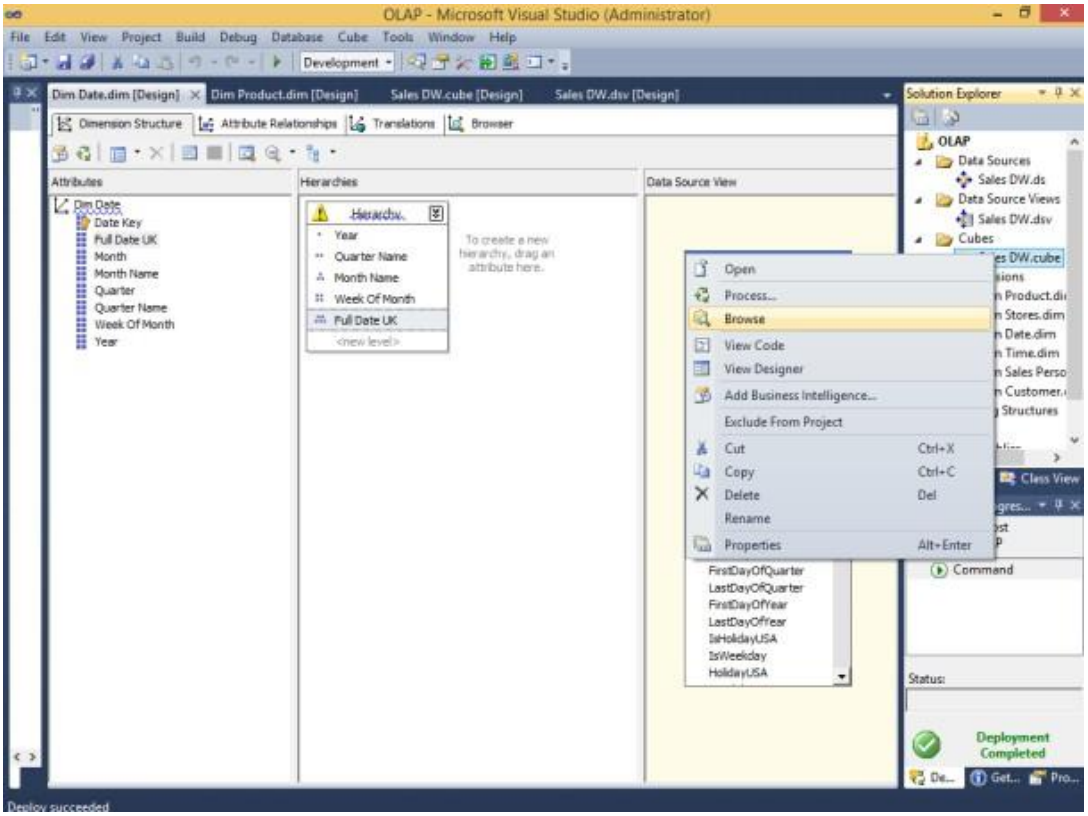
Click run

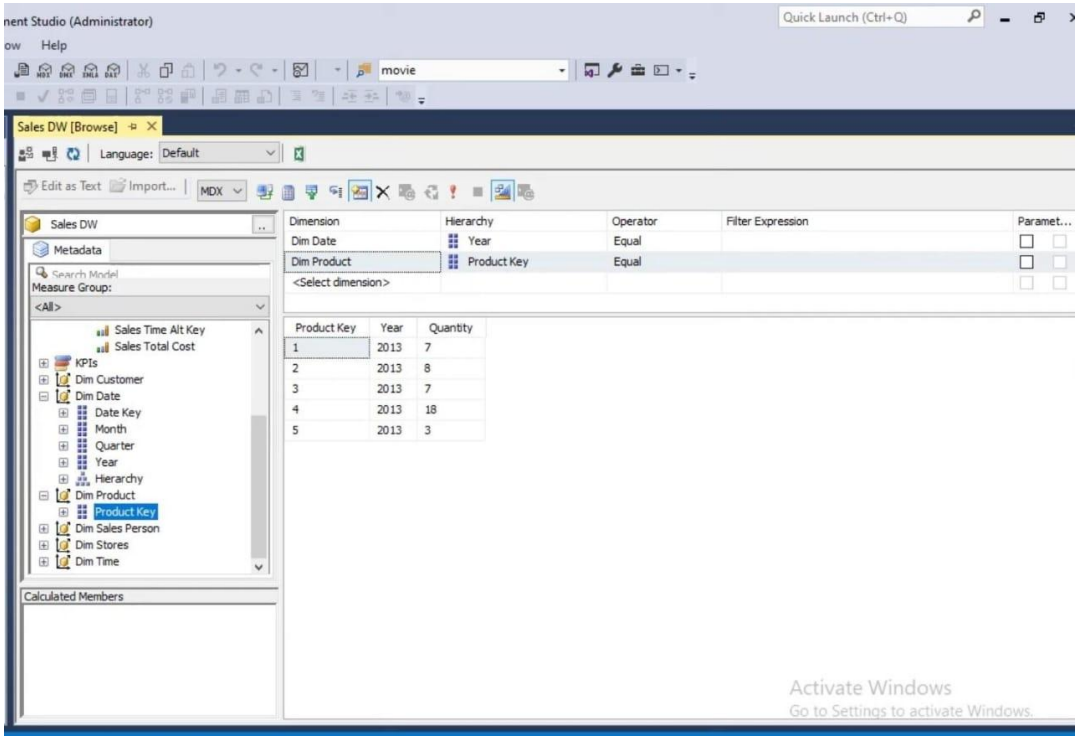




Browse the cube for analysis in solution explorer


OLAP - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Database Cube Tools Window Help
Development
Dim Date.dim [Design]
Dim Product.dim [Design]
Sales DW.cube [Design]
Sales DW.dsv [Design]
Cube S...
Dimensi...
Calcula...
KPIs
Actions
Partitions
Aggreg...
Perspe...
Transla...
Browser
Language: Default
Edit as Text
Import...
Sales DW
Metadata
Measure Group:
<All>
Sales DW
Measures
Fact Product Sales
Deviation
Fact Product Sales Count
Product Actual Cost
Quantity
Sales Invoice Number
Sales Time Alt Key
Sales Total Cost
KPIs
Dim Customer
Dim Date
Date Key
Full Date UK
Month
Month Name
Quarter
Calculated Members
Dimension
Hierarchy
Operator
Filter Expression
<Select dimension>
Drag levels or measures here to add to the query.
Solution Explorer
OLAP
Data Sources
Sales DW.ds
Data Source Views
Sales DW.dsv
Cubes
Sales DW.cube
Dimensions
Dim Product.di
Dim Stores.dim
Dim Date.dim
Dim Time.dim
Dim Sales Perso
Dim Customer.
Mining Structures
Roles
Solution...
Class View
Deployment Progres...
Server : localhost
Database : OLAP
Command
Status:
Deployment Completed
Deploy succeeded


OLAP - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Database Cube Tools Window Help
Development
Dim Date.dim [Design]
Dim Product.dim [Design]
Sales DW.cube [Design]
Sales DW.dsv [Design]
Dimension Structure
Attribute Relationships
Translations
Browser
Attributes
Hierarchies
Data Source View
Dim Date
Date Key
Full Date UK
Month
Month Name
Quarter
Quarter Name
Week Of Month
Year
Hierarchy
Year
Quarter Name
Month Name
Week Of Month
Full Date UK
<new level>
To create a new hierarchy, drag an attribute here.
Open
Process...
Browse
View Code
View Designer
Add Business Intelligence...
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename
Properties Alt+Enter
FirstDayOfQuarter
LastDayOfQuarter
FirstDayOfYear
LastDayOfYear
IsHolidayUSA
IsWeekday
HolidayUSA
Solution Explorer
OLAP
Data Sources
Sales DW.ds
Data Source Views
Sales DW.dsv
Cubes
Class View
Command
Status:
Deployment Completed
Deploy succeeded


nent Studio (Administrator)
Quick Launch (Ctrl+Q)
ow Help
movie
Sales DW [Browse]
Language: Default
Edit as Text
Import...
MDX
Sales DW
Metadata
Search Model
Measure Group:
<All>
Sales Time Alt Key
Sales Total Cost
KPIs
Dim Customer
Dim Date
Date Key
Month
Quarter
Year
Hierarchy
Dim Product
Product Key
Dim Sales Person
Dim Stores
Dim Time
Calculated Members
Dimension
Hierarchy
Operator
Filter Expression
Paramet...
Dim Date
Year
Equal
Dim Product
Product Key
Equal
<Select dimension>
Product Key
Year
Quantity
1
2013
7
2
2013
8
3
2013
7
4
2013
18
5
2013
3
Activate Windows
Go to Settings to activate Windows.