Pratical -1(A)

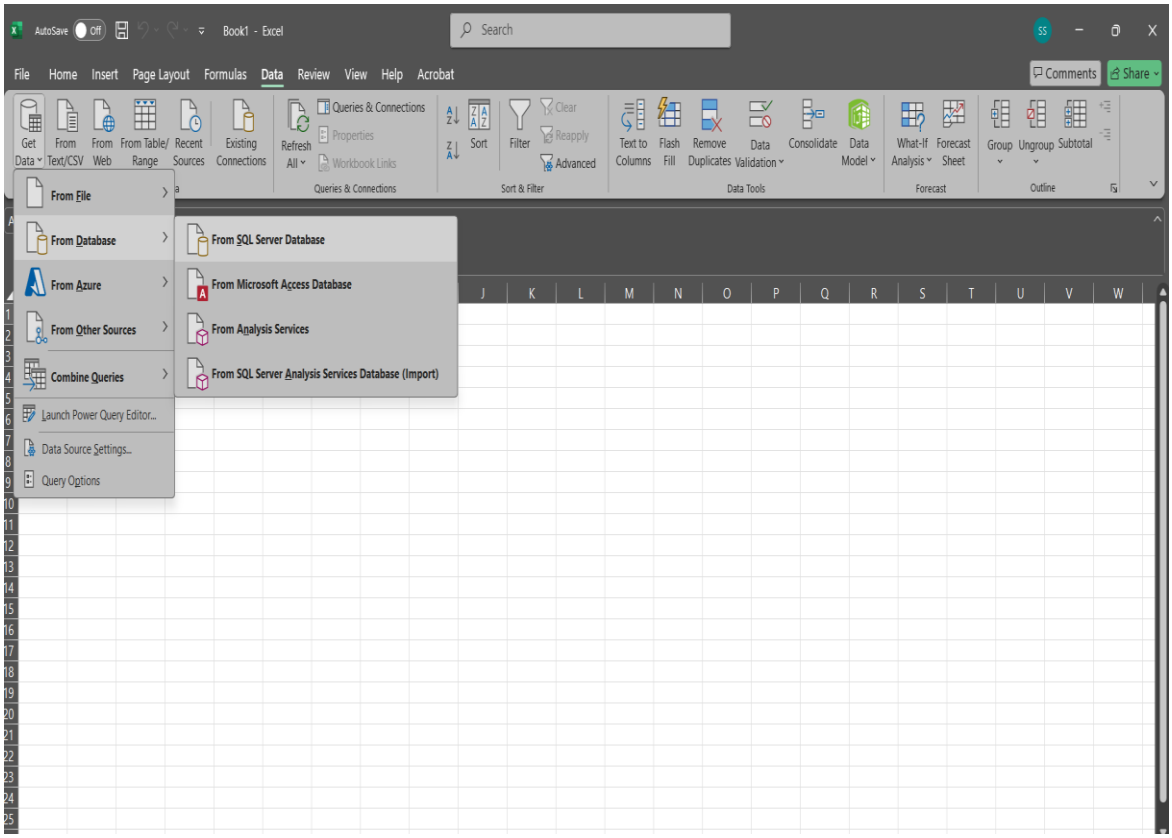
AIM:- Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

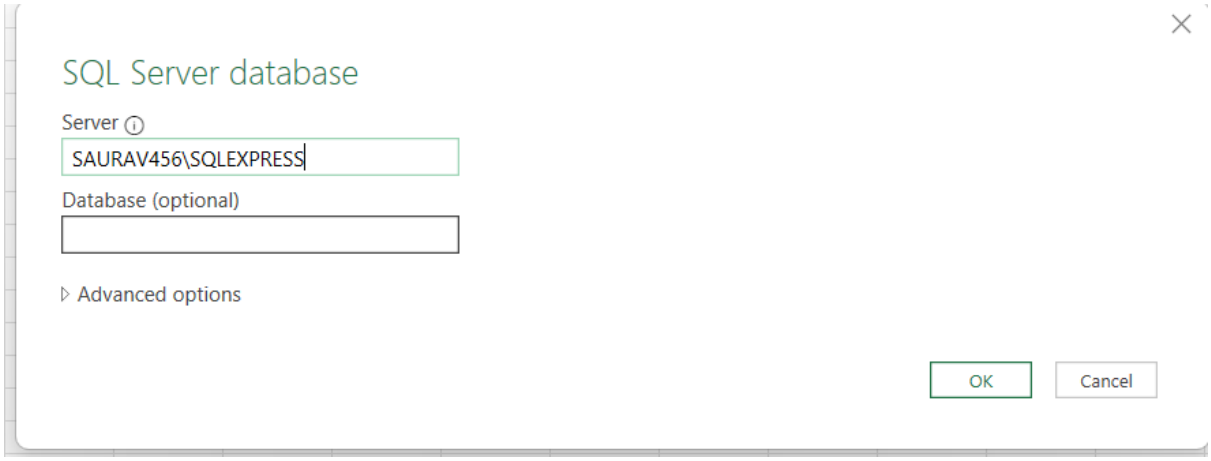
STEPS:-

Step 1: Open Microsoft Excel

Step 2: Connect to Data Warehouse

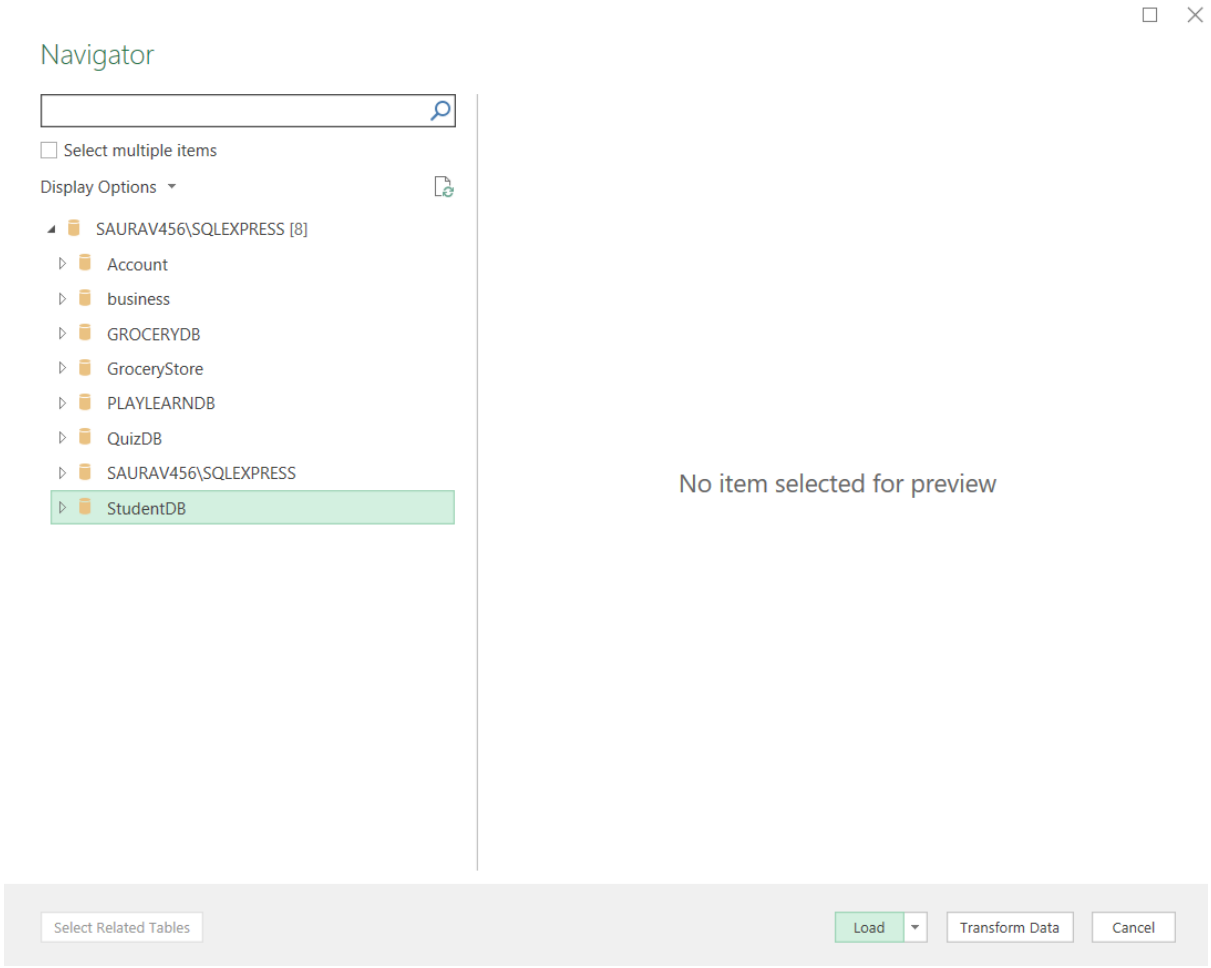
1. Go to the Data tab on the Ribbon.
2. Click Get Data > From Database (Choose the appropriate option based on your database type).
    - From SQL Server Database (for Microsoft SQL Server)
    - From Oracle Database (for Oracle)
    - From MySQL Database (for MySQL)
    - From Other Sources (for other databases)
3. Enter the Server Name and Database Name when prompted.
4. Click OK and then choose the required table or query.





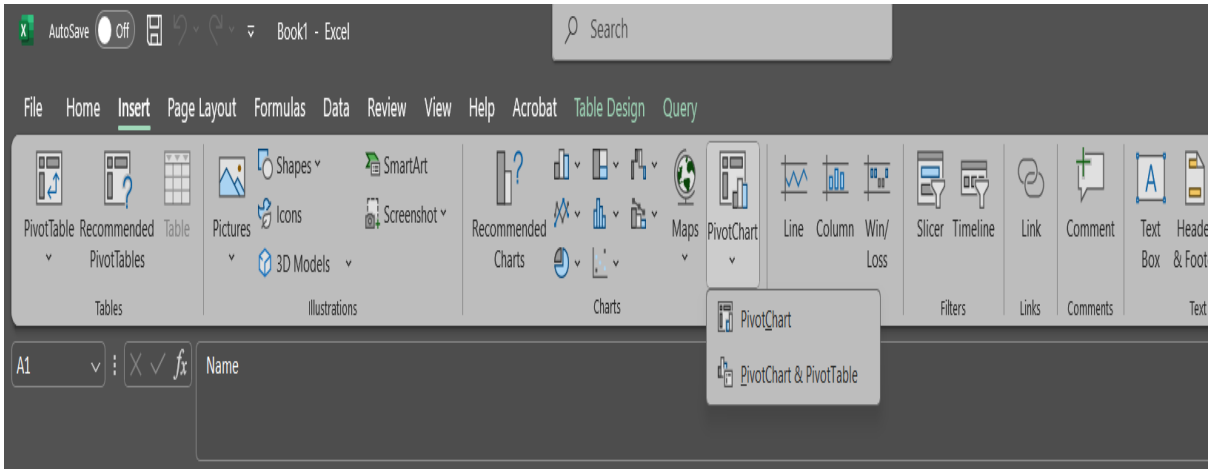
Step 3: Load Data into Excel

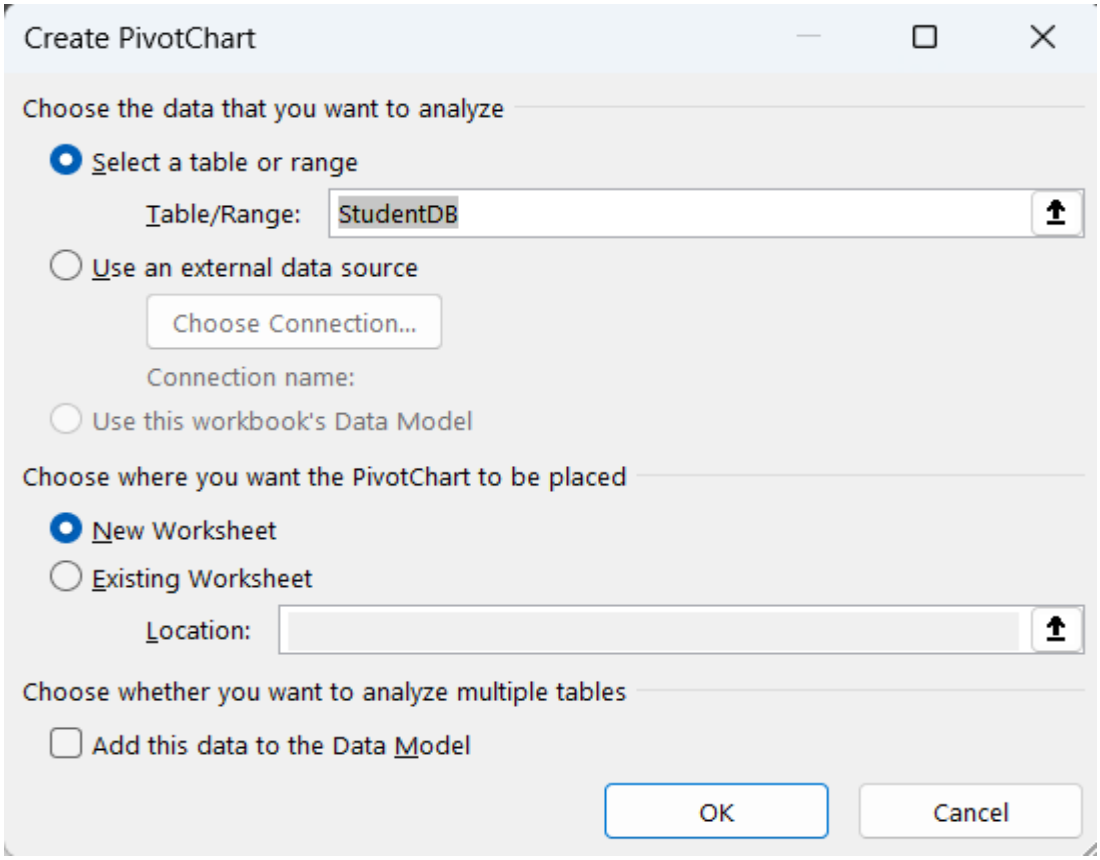
1. After selecting the table, you can either:
    - Click **Load** to import the data directly into an Excel sheet.
    - Click **Transform Data** to clean or filter data before importing (optional).
2. Wait for the data to load into an Excel worksheet.



Step 4: Create a Pivot Table

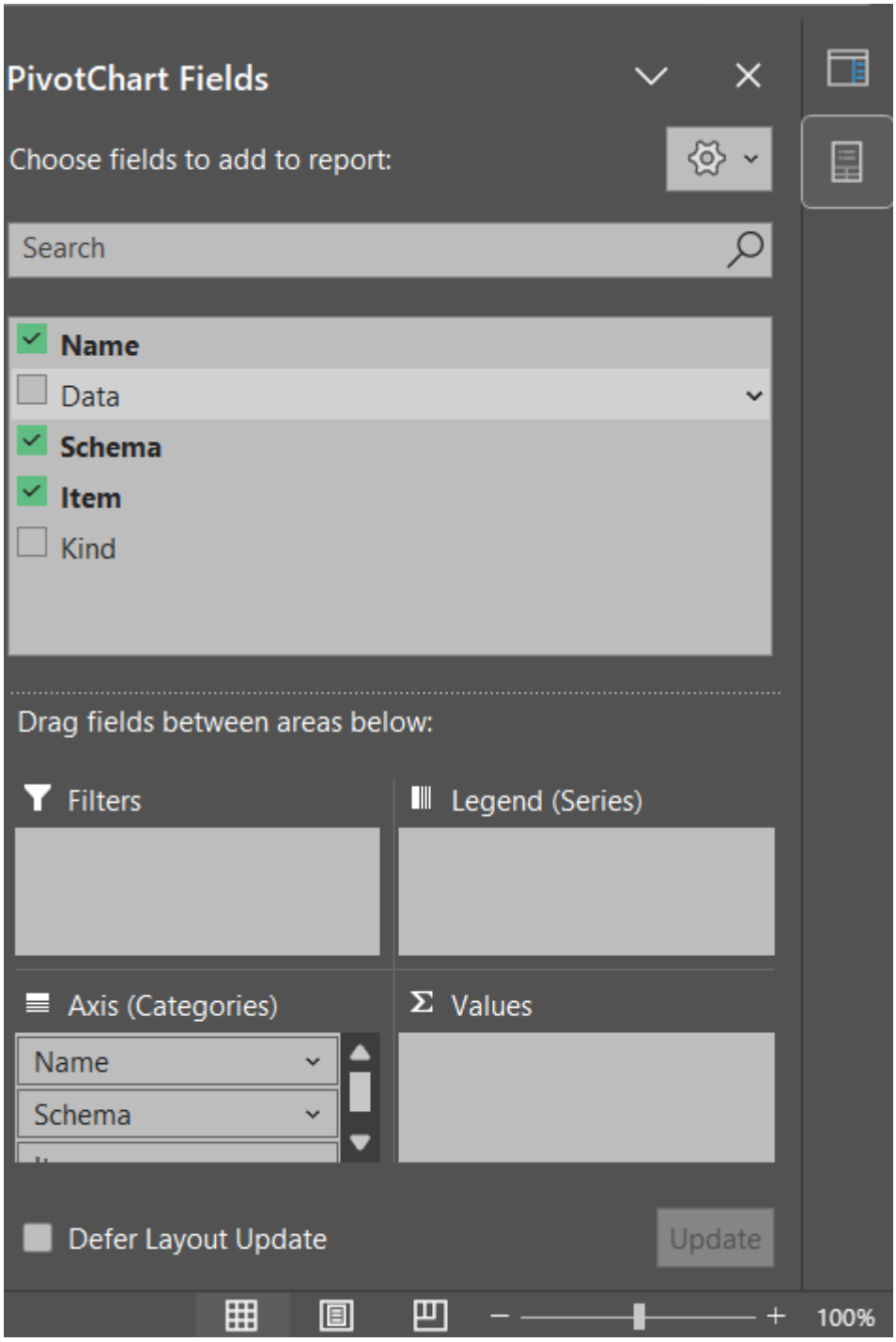
1. Click anywhere in the imported data.
2. Go to the **Insert** tab and select **PivotTable**.
3. In the dialog box:
    - Choose **Select a table or range** (if data is already in the worksheet).
    - Choose **Use an external data source** (if connecting directly to the database).
4. Click **OK** to create the Pivot Table in a new or existing worksheet.





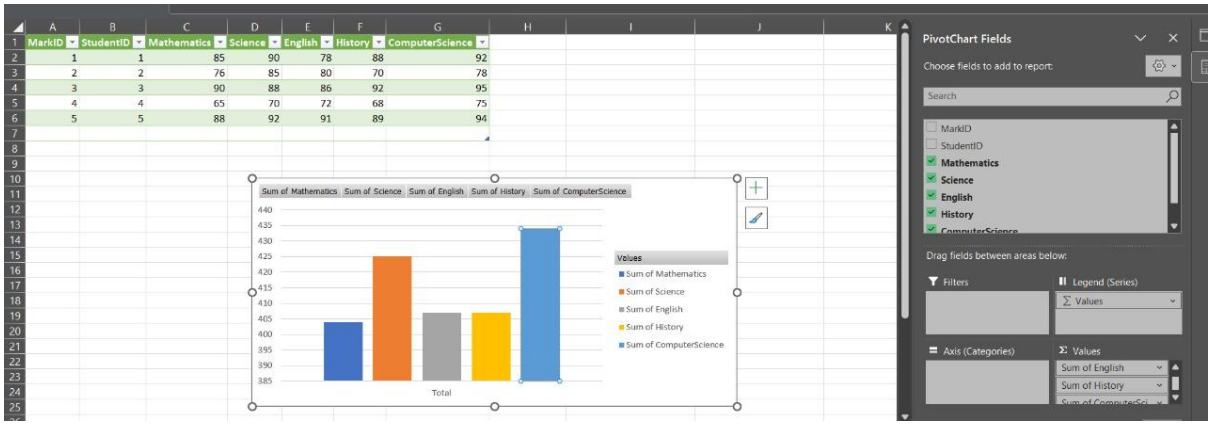
Step 5: Customize the Pivot Table

1. 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

1. Click anywhere in the **Pivot Table**.
2. Go to the **Insert** tab and select **PivotChart**.
3. Choose the chart type (e.g., Column, Bar, Line, Pie).
4. Click **OK**, and the chart will appear in the worksheet.
5. Customize the chart using the **Chart Tools** tab.



Step 7: Save and Refresh Data

1. Save the Excel file for future use.
2. To refresh the Pivot Table and Chart when new data is added, go to **Data** > **Refresh All**.

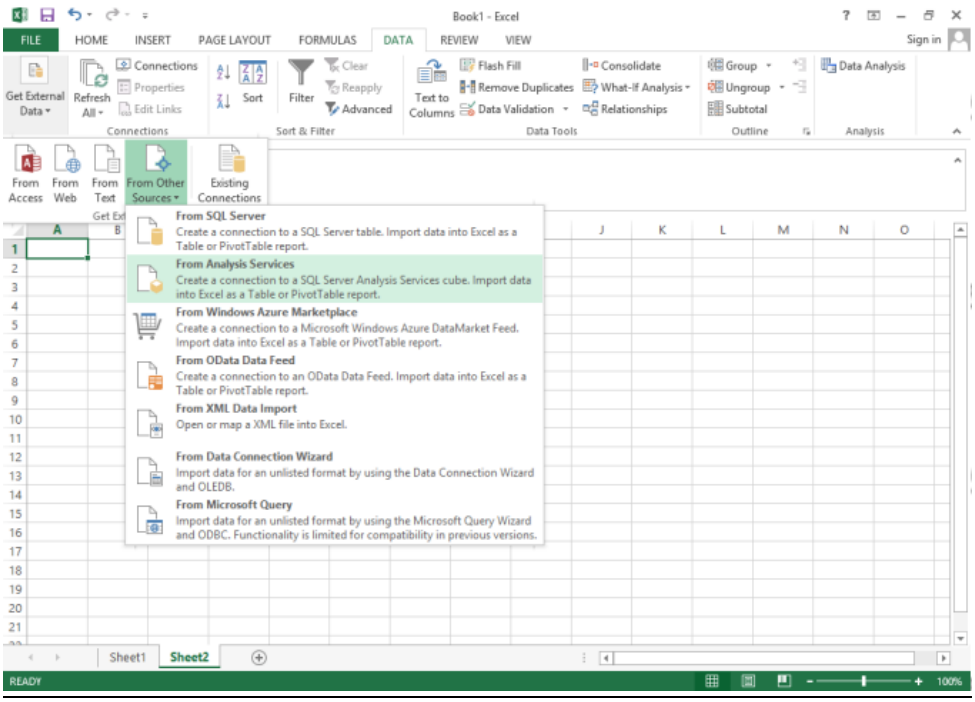
PRATICAL-1(b)

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

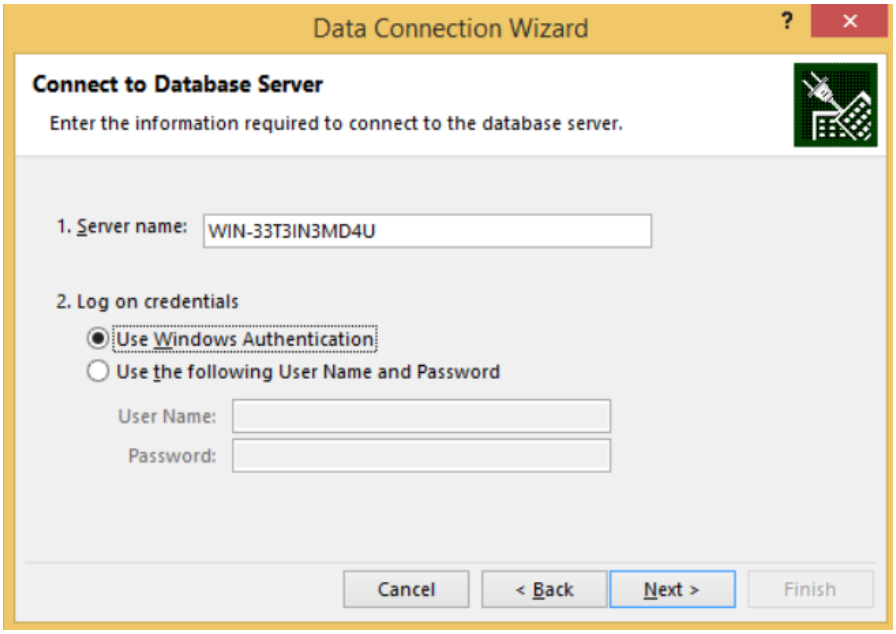
STEPS:-

Step 1:-Open Microsoft Excel

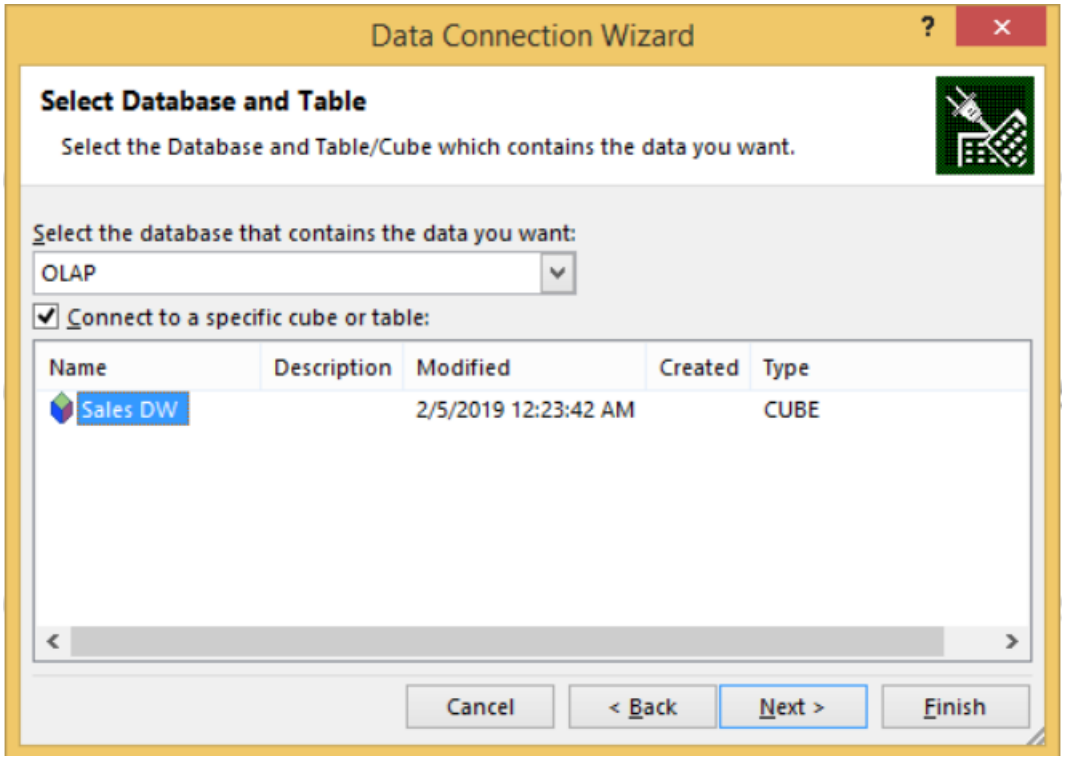
Step2:- Step Go to Data tab → Get External Data → From Other Sources → From Analysis Services



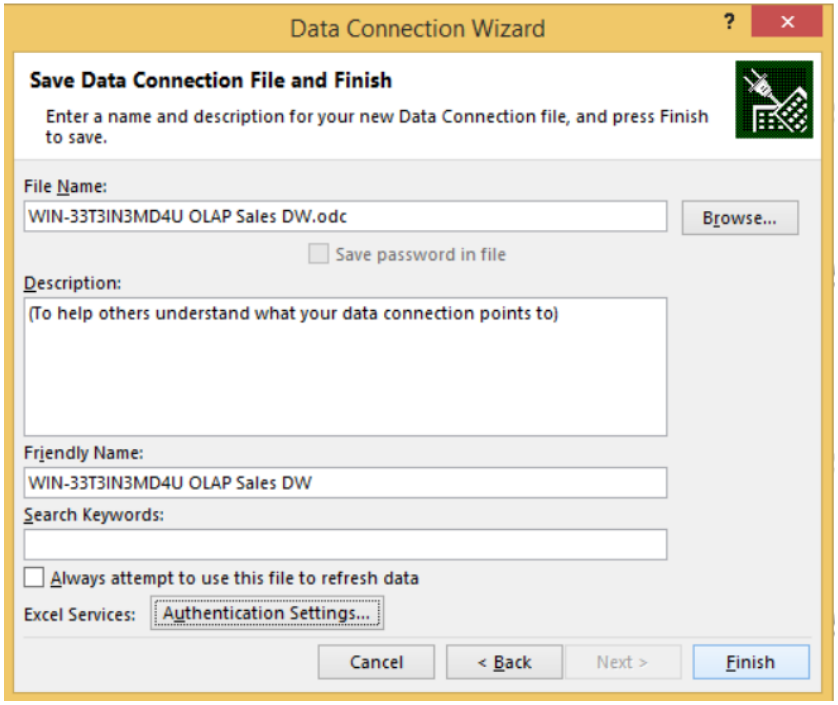
Step 3:- Select Server name and Windows Authentication and click on Next



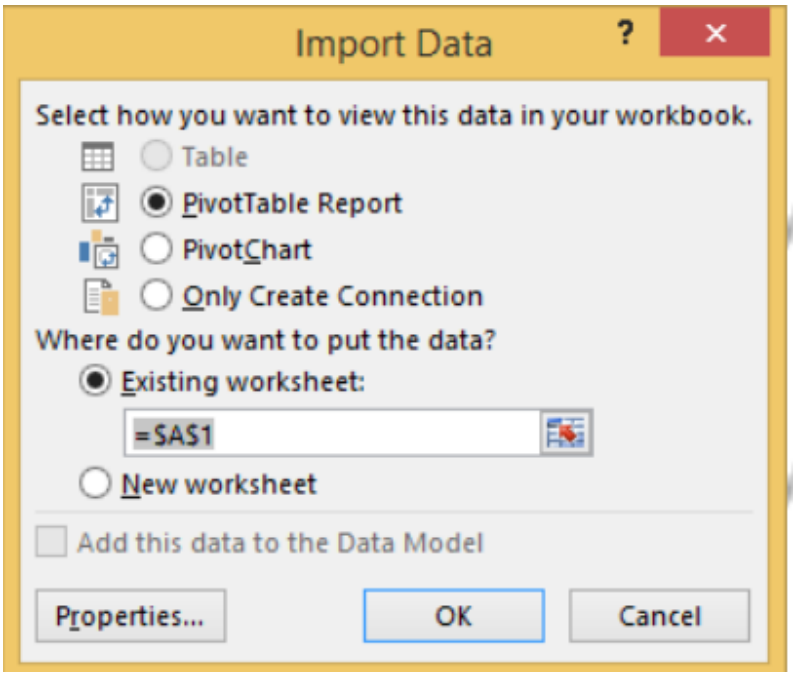
Step 4:- Select OLAP(as per created before) click on Next



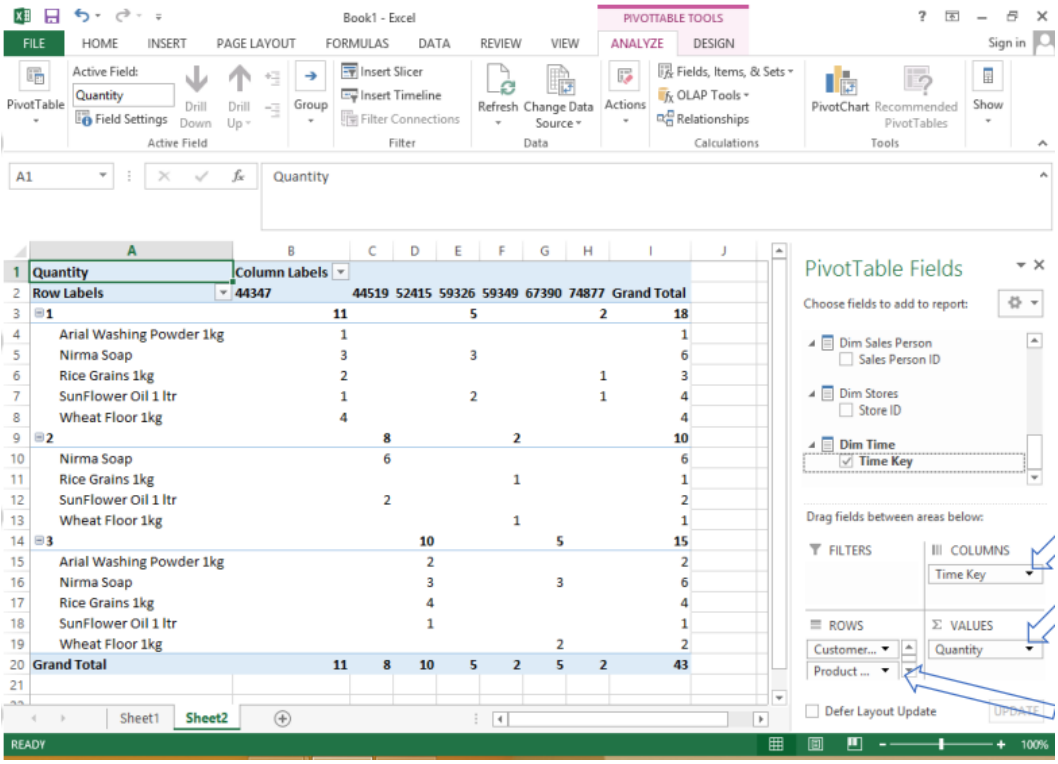
Step5:- Browse and select path name and click on Finish



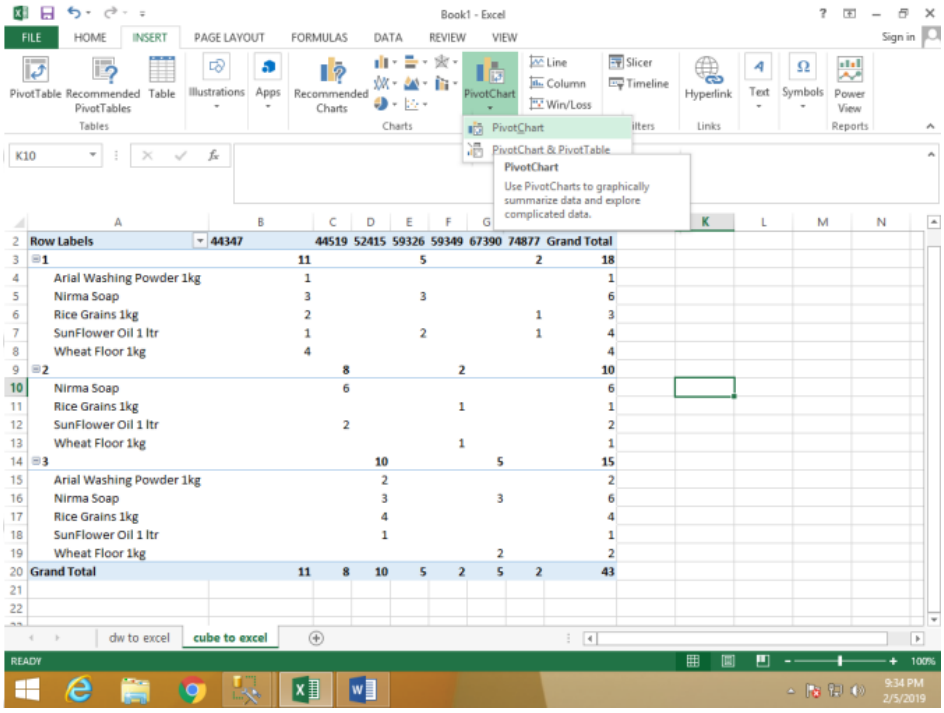
Step 6:- Select Pivot Table Report → OK



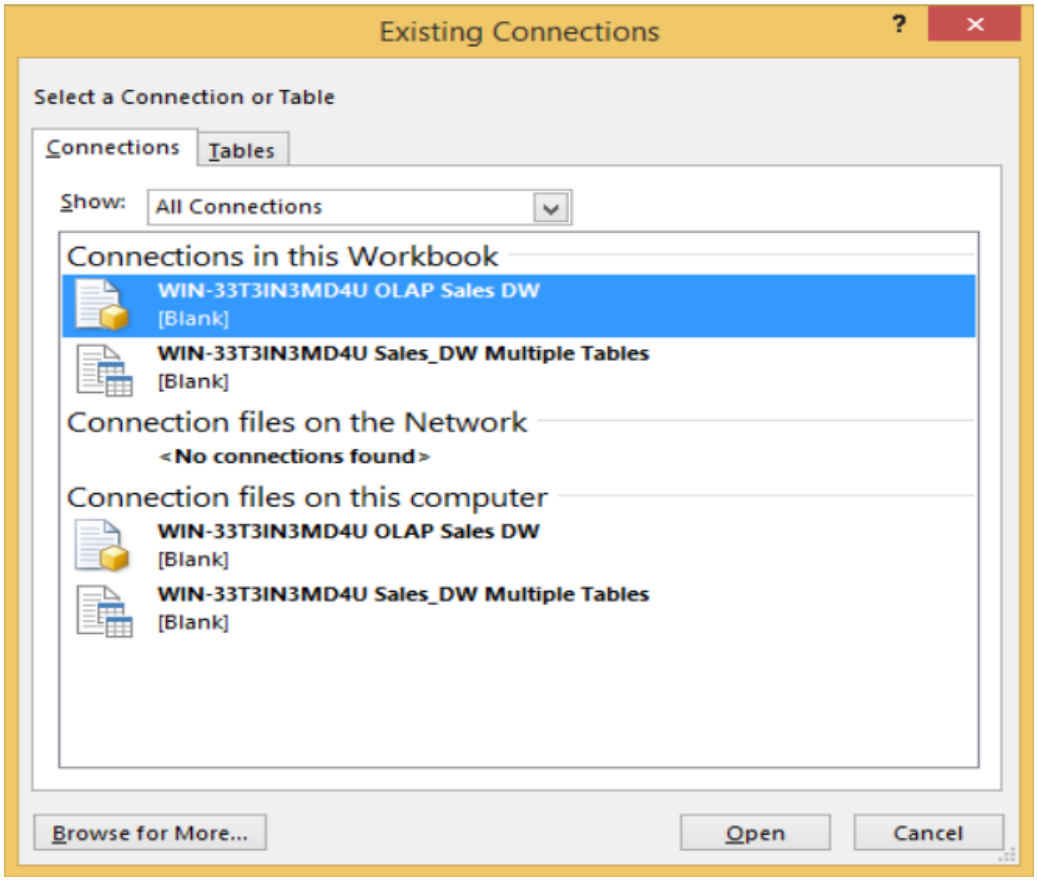
Step 7:- Drag and Drop Fields in rows column and values



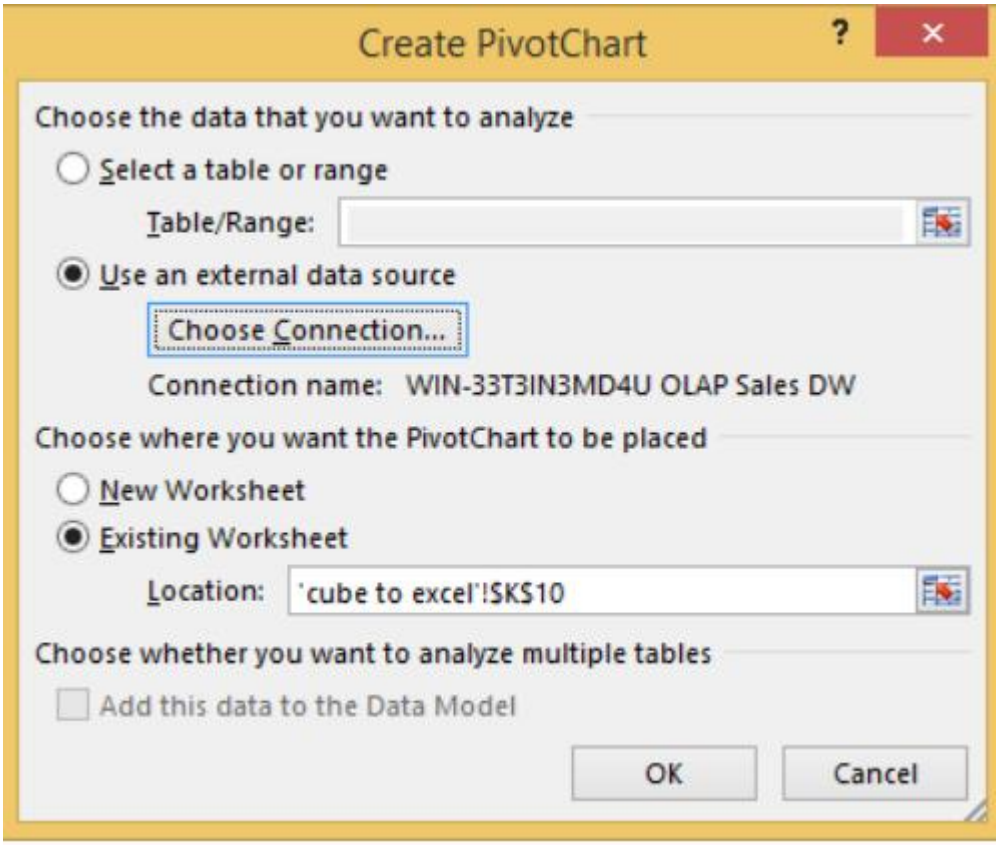
Step 8:-Go to Insert tab → pivot chart and select Pivot Chart from drop down



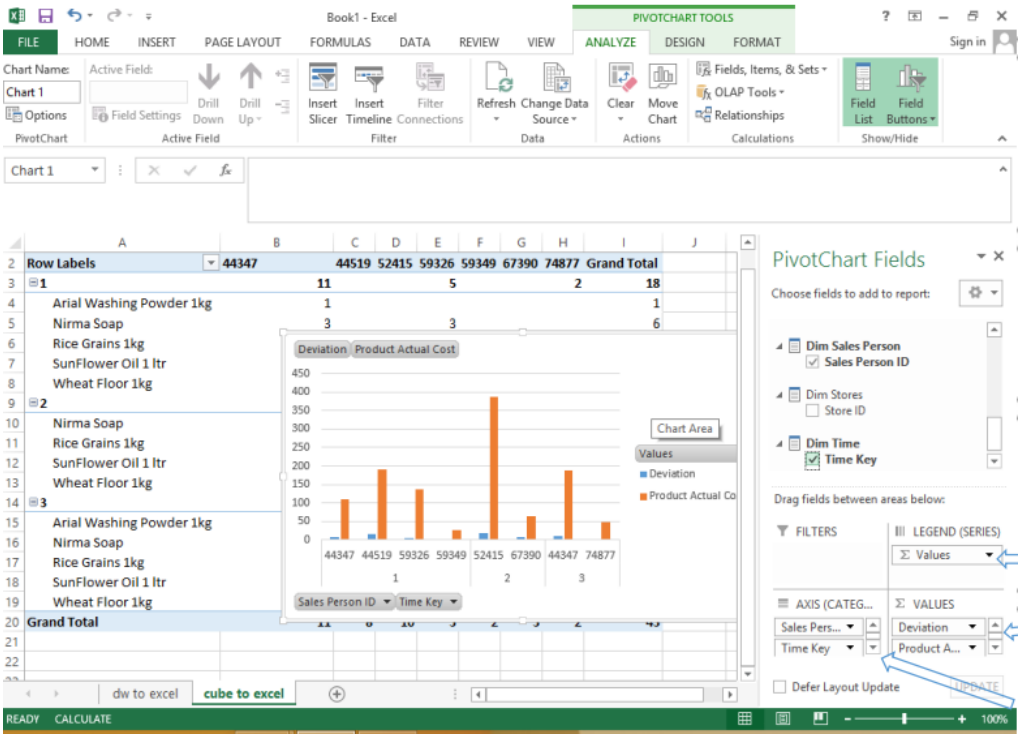
Step 9:- Select existing connection OLAP Sales DW and click on Open



Step 10:- Click on Choose connection to select path



Step 11:- Click on OK

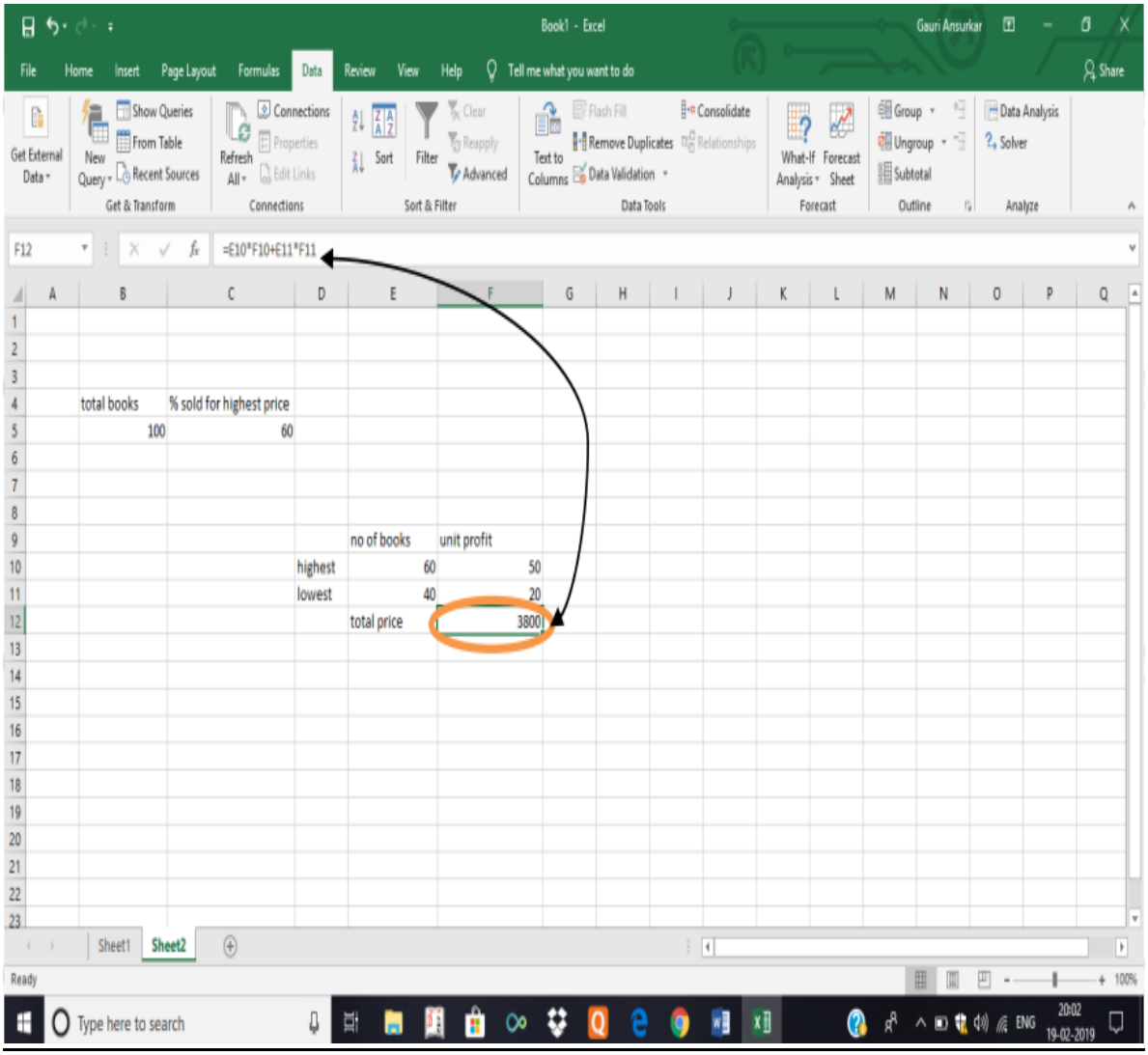


PRATICAL-2

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.



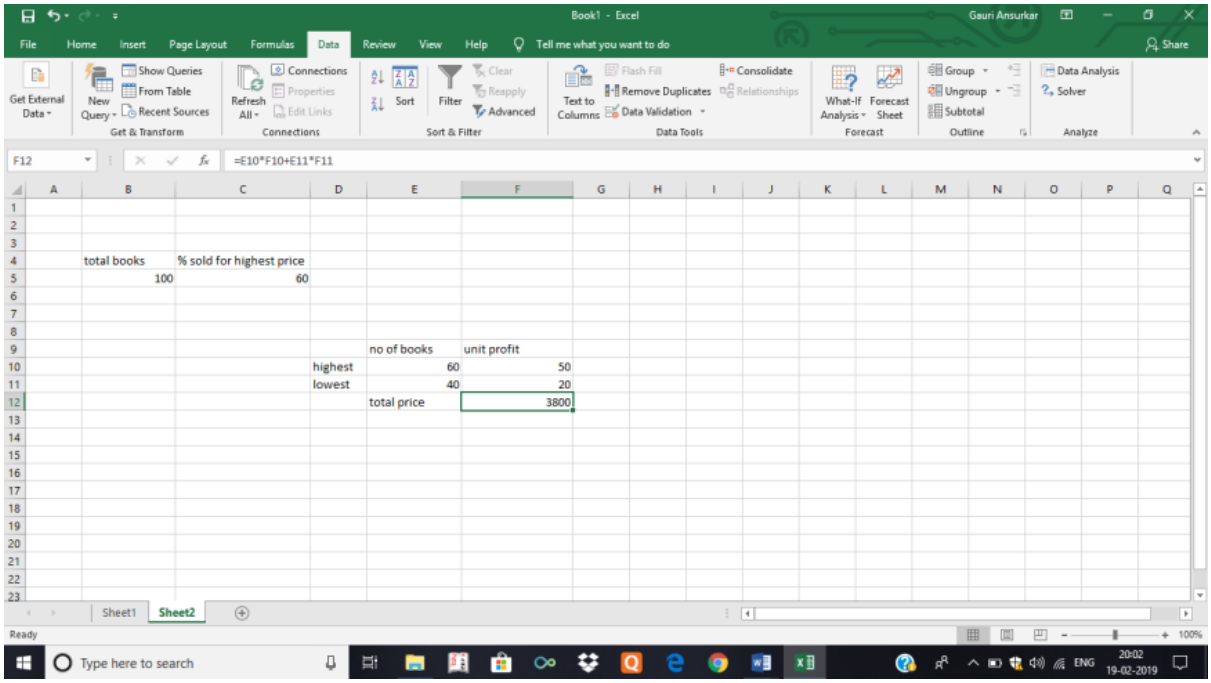
If you sell 60% for the highest price, cell D10 calculates a total profit of 60 * 50 + 40 * 20 = 3800.

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.

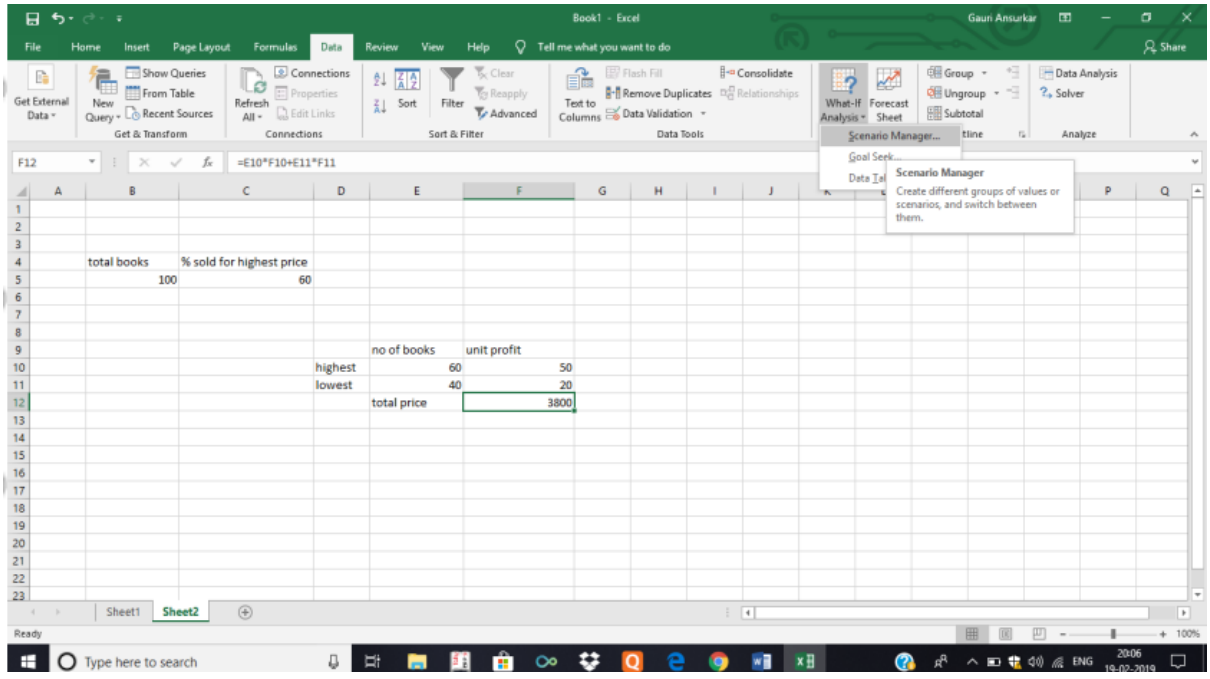
Note: To type different percentage into cell C4 to see the corresponding result of a scenario in cell D10 we use what if analysis.

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

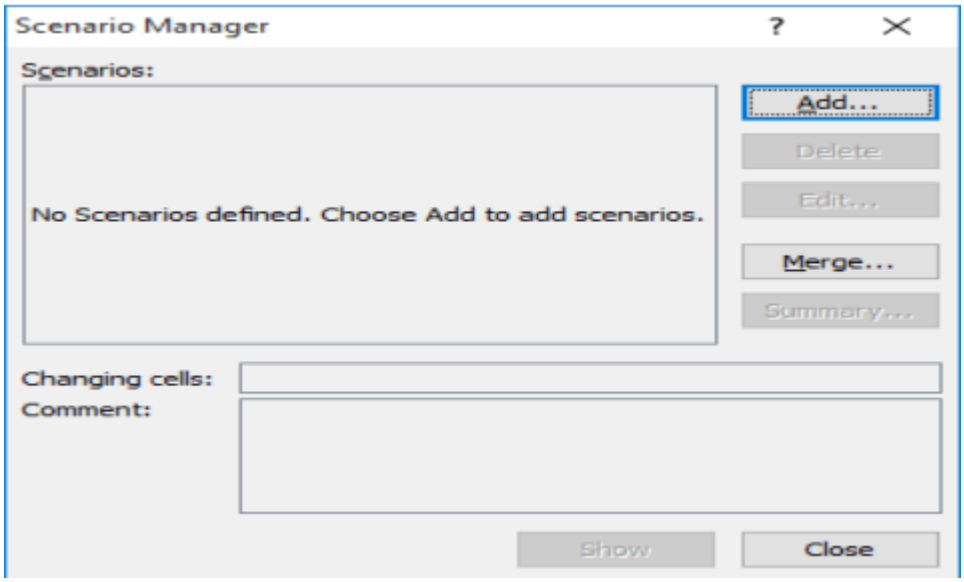


Step 2: Click on What –if-Analysis and select scenario manager.



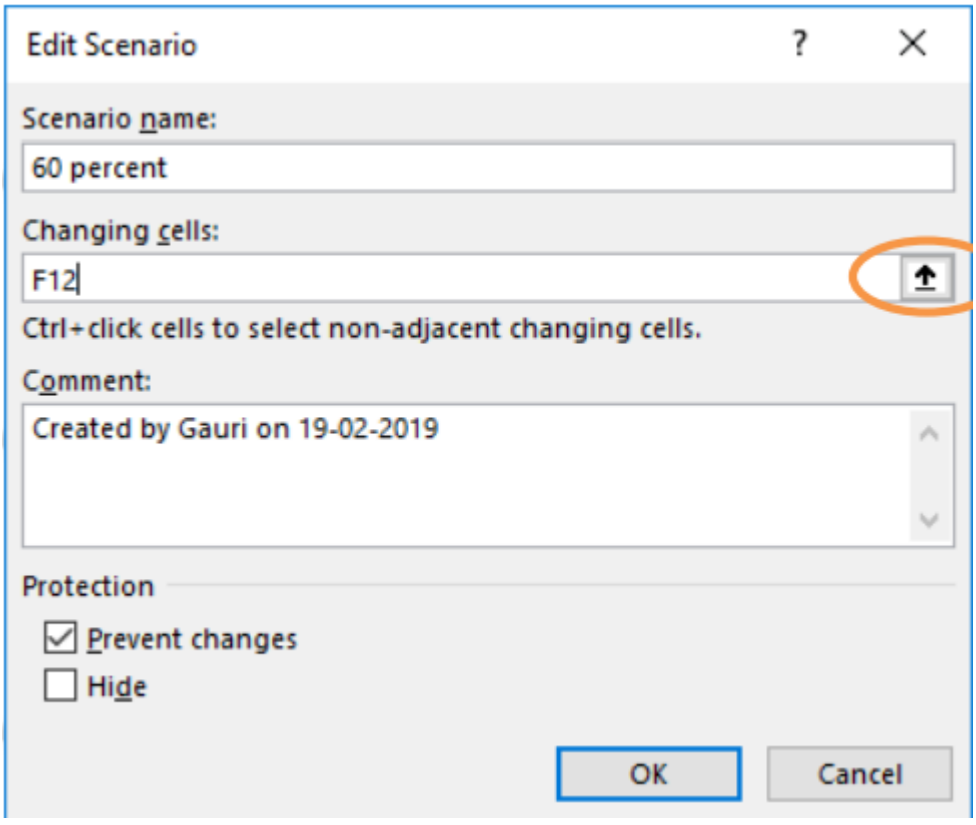
The Scenario Manager Dialog box appears.

Step 3: Add a scenario by clicking on Add.

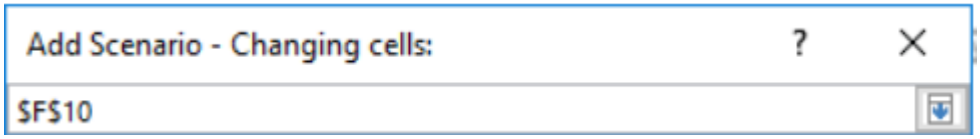


Step 4: Type a name (60percent), select cell F10 (% sold for the highest price) for the Changing cells and click on OK.

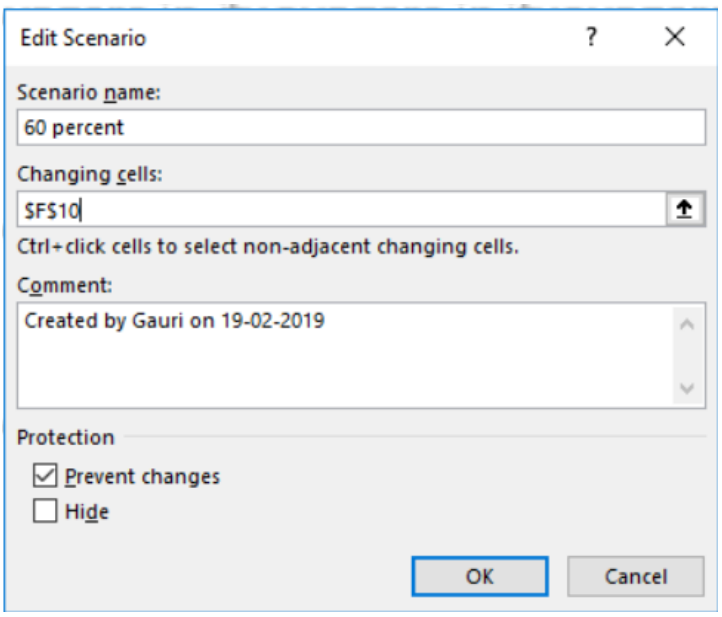
Click on icon which is circled.



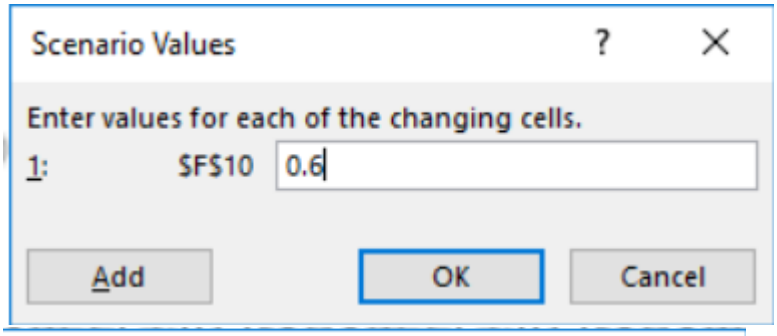
Select F10 cell.

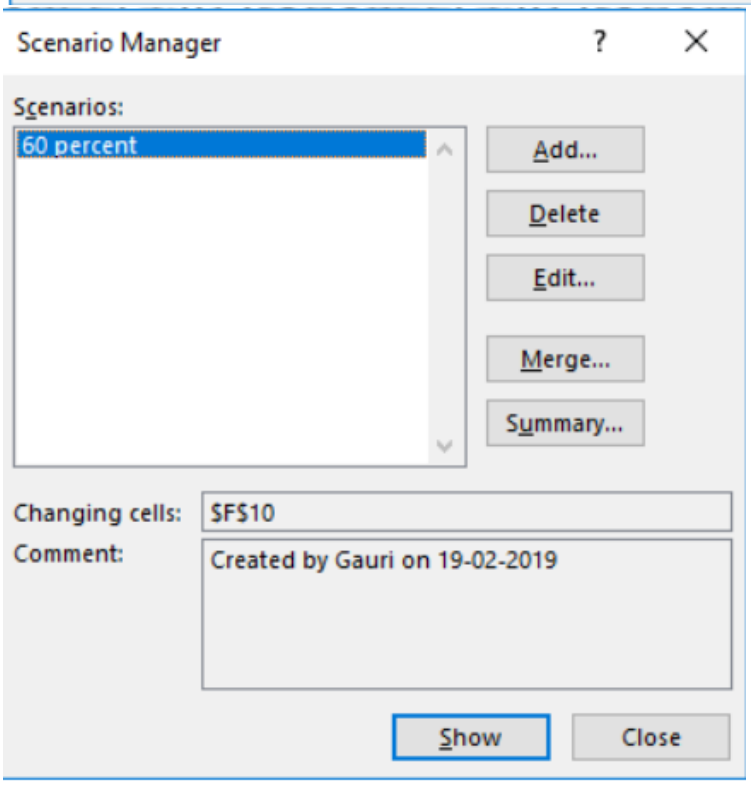


Click back on the icon again and then click OK

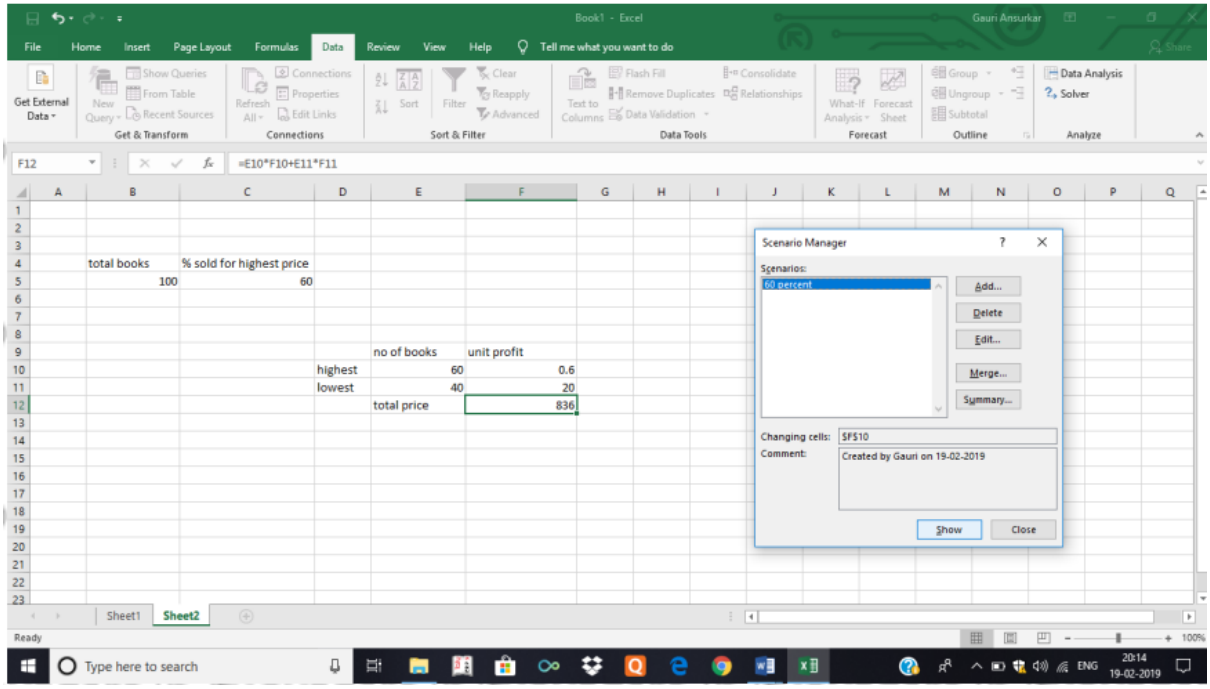


Step 5: Enter the corresponding value 0.6 and click on OK again.

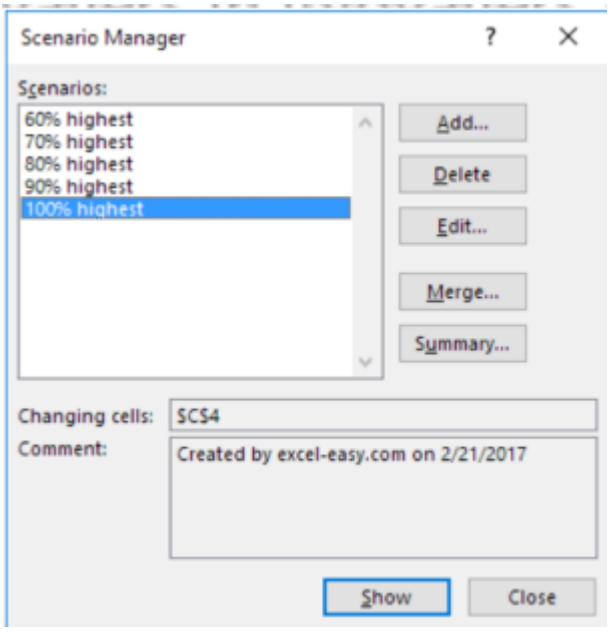




Step 6: To apply scenarios click on Show



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



PRATICAL-3

AIM:- Perform the data classification using classification algorithm using R/Python.

STEPS:-

Software required: R 3.5.1

Step 1:- Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called time-series object. It is also a R data object like a vector or data frame.

Step 2:-The time series object is created by using the ts () function.

Syntax

The basic syntax for ts() function in time series analysis is – timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used –

- **data is a vector or matrix containing the values used in the time series.**
- **start specifies the start time for the first observation in time series.**
- **end specifies the end time for the last observation in time series.**
- **end specifies the end time for the last observation in time series.**

Except the parameter "data" all other parameters are optional

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

Step 3:-Code

```
# 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.

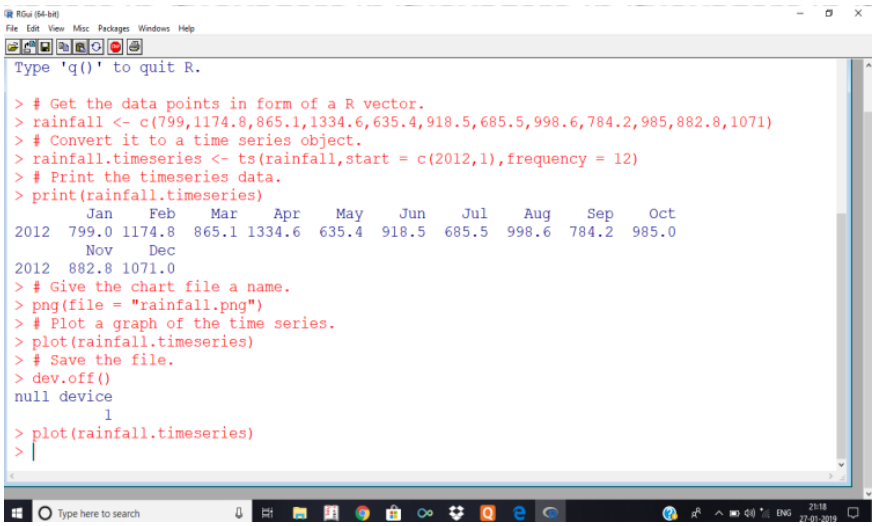
dev.off()

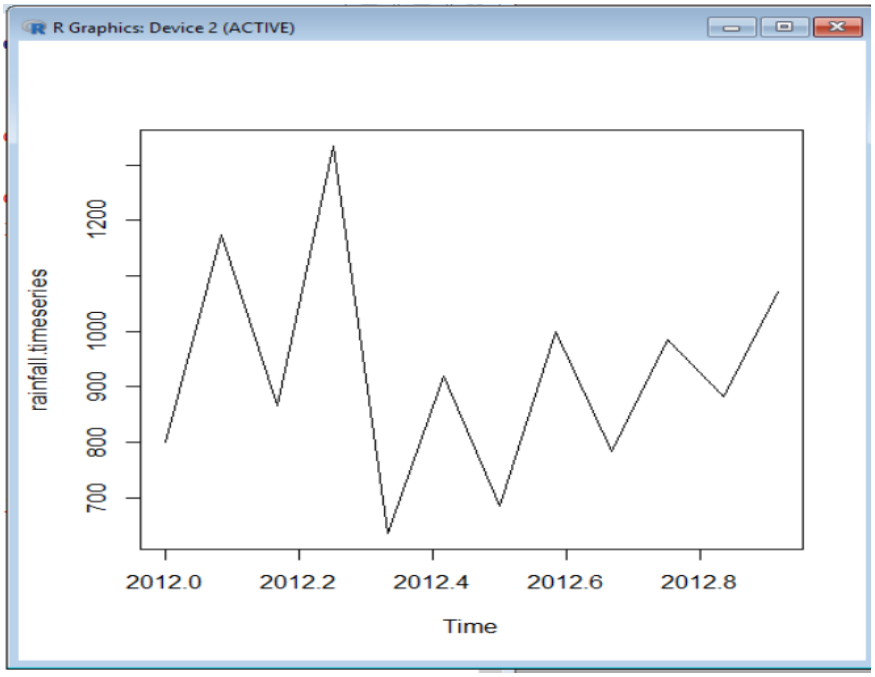
After this again plot to get chart

plot(rainfall.timeseries)

Output:

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





PRATICAL-4

AIM:- **Perform the data clustering using clustering algorithm using R/Python.**

STEPS:-

k-means clustering using R

Step 1:-apply K means to iris and store result

```
newiris <- iris

newiris$Species <- NULL

(kc <- kmeans(newiris,3))
```

```
K-means clustering with 3 clusters of sizes 21, 96, 33

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     4.738095    2.904762     1.790476   0.3523810
2     6.314583    2.895833     4.973958   1.7031250
3     5.175758    3.624242     1.472727   0.2727273

Clustering vector:
  [1] 3 1 1 1 3 3 3 3 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 1 1 3 3 3 1 1 3 3 3 1 3 3 3 1
 [40] 3 3 1 1 3 3 1 3 1 3 3 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [79] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[118] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  17.669524 118.651875   6.432121
 (between_SS / total_SS =  79.0 %)

Available components:

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

Step 2:-Compare the Species label with the clustering result

```
table(iris$Species,kc$cluster)
```

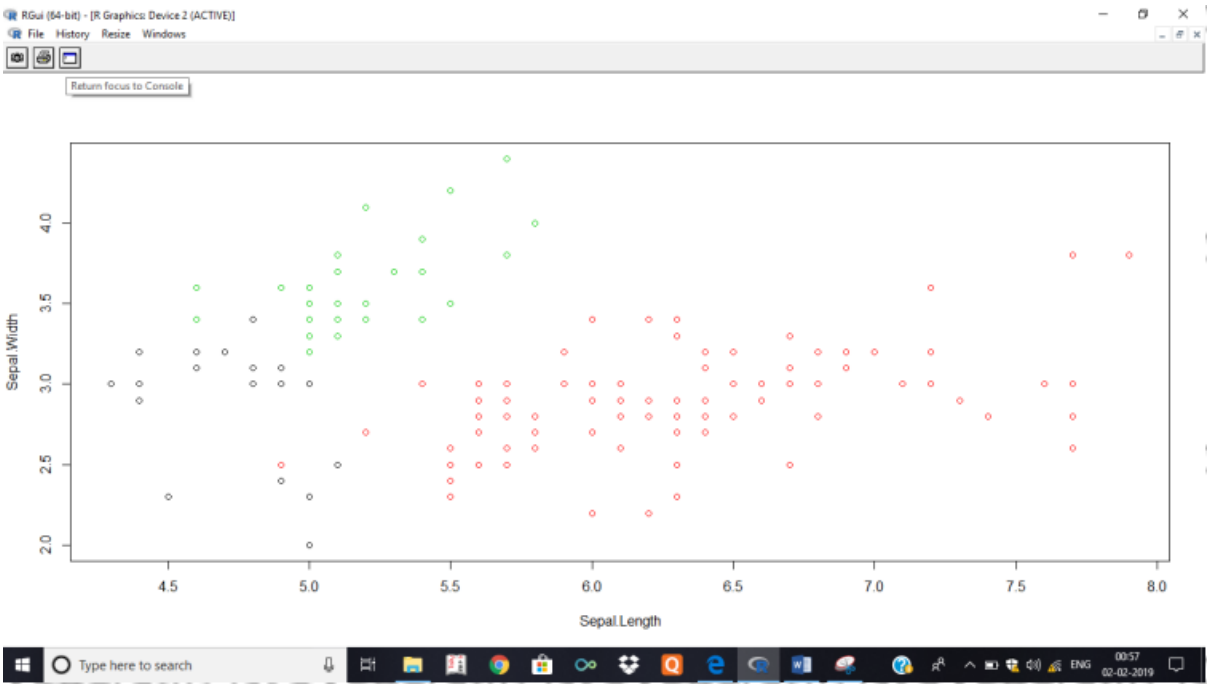
```
            1  2  3
setosa     17  0 33
versicolor  4 46  0
virginica   0 50  0
```

Step 3:-Plot the clusters and their centers

```
plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)

points(kc$centers[,c("Sepal.Length","Sepal.Width")],col=1:3,pch=8,cex=2)

dev.off()
```

Step 4:-Plot the clusters and their centre

plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)



PRATICAL-5

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

STEPS:-

Input Data

Step 1:- Values of height

151, 174, 138, 186, 128, 136, 179, 163, 152, 131

Step 2:-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

Step 3:-The basic syntax for lm() function in linear regression is –

lm(formula,data)

Following is the description of the parameters used :–

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

Step 4:-Create Relationship Model & get the Coefficients

```
# Values of height
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# Values of width
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function.
relation <- lm(y~x)
print(relation)
```

OUTPUT:

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

Coefficients:
(Intercept)            x
   -38.4551       0.6746
```

Step 4:- Get the Summary of the Relationship

```
# Values of height
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# Values of width
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function
relation <- lm(y~x)
print(summary(relation))
```

OUTPUT:

```
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
```

Step 5:- predict() Function

Syntax

predict(object, newdata)

Step 6:- 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)
```

OUTPUT:

```
       1
76.22869
```

Step 6:- Visualize the Regression Graphically

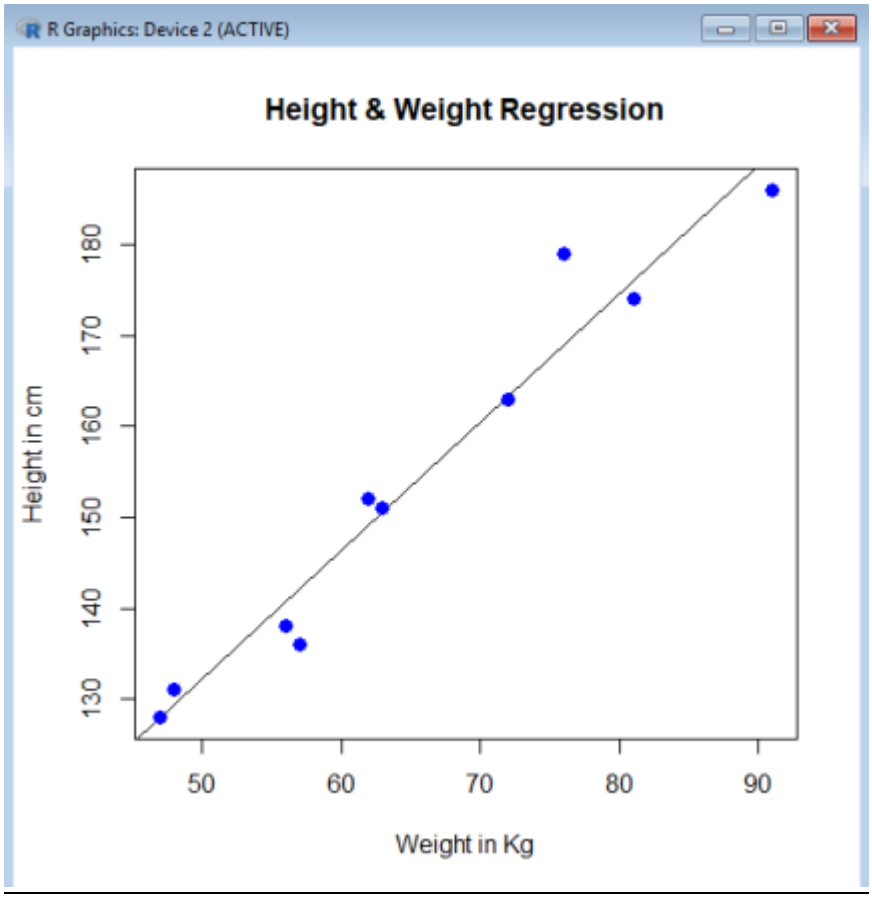
```
# Create the predictor and response variable

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()
```

```
null device
          1
```

Step 7:- 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")
```

OUTPUT:


R Graphics: Device 2 (ACTIVE)
Height & Weight Regression
Height in cm
130
140
150
160
170
180
50
60
70
80
90
Weight in Kg

PRATICAL-6

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

STEPS:-

To perform this you need to download quality.csv file from following link:
https://github.com/TarekDib03/Analytics/tree/master/Week3%20- %20Logistic%20Regression/Data

Step 1:- provide path of file where it is saved on your machine

```
quality <- read.csv('C:/Users/Gauri/Downloads/quality.csv')
> #analysing the quality dataset
> str(quality)
'data.frame': 131 obs. of 14 variables:
$ MemberID : int 1 2 3 4 5 6 7 8 9 10 ...
$ InpatientDays : int 0 1 0 0 8 2 16 2 2 4 ...
$ ERVisits : int 0 1 0 1 2 0 1 0 1 2 ...
 $ OfficeVisits : int 18 6 5 19 19 9 8 8 4 0 ...
$ Narcotics : int 1 1 3 0 3 2 1 0 3 2 ...
$ DaysSinceLastERVisit: num 731 411 731 158 449 ...
 $ Pain : int 10 0 10 34 10 6 4 5 5 2 ...
$ TotalVisits : int 18 8 5 20 29 11 25 10 7 6 ...
$ ProviderCount : int 21 27 16 14 24 40 19 11 28 21 ...
$ MedicalClaims : int 93 19 27 59 51 53 40 28 20 17 ...
 $ ClaimLines : int 222 115 148 242 204 156 261 87 98 66 ...
$ StartedOnCombination: logi FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ AcuteDrugGapSmall : int 0 1 5 0 0 4 0 0 0 0 ...
$ PoorCare : int 0 0 0 0 0 1 0 0 1 0 ...
 > table(quality$PoorCare)
0 1
98 33
> 98/131
[1] 0.7480916
```

> install.packages("caTools")

Installing package into 'C:/Users/Saurav/Documents/R/win-library/3.5' (as 'lib' is unspecified)

--- Please select a CRAN mirror for use in this session ---

Step 2:- installing the dependency 'bitops' from

'http://mirror.its.dal.ca/cran/bin/windows/contrib/3.5/bitops_1.0-6.zip' Content type 'application/zip' length 38894 bytes (37 KB)

installing the dependency 'caTools' from

'http://mirror.its.dal.ca/cran/bin/windows/contrib/3.5/caTools_1.17.1.1.zip'

Step 3:- package 'bitops' successfully unpacked and MD5 sums checked

package 'caTools' successfully unpacked and MD5 sums checked

The downloaded binary packages are in C:\Users\Saurav\AppData\Local\Temp\RtmpmUN9oK\downloaded_package s

Step 4:- > library(caTools) Warning message: package 'caTools' was built under R version 3.5.2

> set.seed(88)

> split = sample.split(quality$PoorCare, SplitRatio = 0.75)

>

> split

[1] TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE

[28] TRUE TRUE FALSE FALSE FALSE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE TRUE FALSE FALSE TRUE

[55] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[82] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE

[109] TRUE FALSE FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE

> qualityTrain = subset(quality, split == TRUE)

> qualityTest = subset(quality, split == FALSE) > nrow(qualityTrain)

[1] 99

> nrow(qualityTest)

[1] 32

> QualityLog = glm(PoorCare ~ OfficeVisits + Narcotics,data=qualityTrain, family=binomial)

```
> summary(QualityLog)

Call:
glm(formula = PoorCare ~ OfficeVisits + Narcotics, family = binomial, data = qualityTrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.06303 -0.63155 -0.50503 -0.09689  2.16686

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.64613    0.52357  -5.054 4.33e-07 ***
OfficeVisits 0.08212    0.03055   2.688  0.00718 **
Narcotics    0.07630    0.03205   2.381  0.01728 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 111.888  on 98  degrees of freedom
Residual deviance:  89.127  on 96  degrees of freedom
AIC: 95.127

Number of Fisher Scoring iterations: 4

> predictTrain = predict(QualityLog, type="response")
> summary(predictTrain)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.06623 0.11912 0.15967 0.25253 0.26765 0.98456
> tapply(predictTrain, qualityTrain$PoorCare, mean)
        0         1
0.1894512 0.4392246
> table(qualityTrain$PoorCare, predictTrain > 0.5)
   FALSE TRUE
0    70    4
1    15   10

> 10/25
[1] 0.4
> 70/74
```

[1] 0.9459459

> table(qualityTrain$PoorCare, predictTrain > 0.7)

FALSE TRUE

0 73 1

1 17 8

> 8/25

[1] 0.32

> 73/74

[1] 0.9864865

> table(qualityTrain$PoorCare, predictTrain> 0.2)

FALSE TRUE

0 54 20

1 9 16

> 16/25

[1] 0.64

> 54/74

[1] 0.7297297

> install.packages("ROCR")

Installing package into 'C:/Users/Saurav/Documents/R/win-library/3.5' (as 'lib' is unspecified) also installing the dependencies 'gtools', 'gdata', 'gplots'

trying URL

'http://mirror.its.dal.ca/cran/bin/windows/contrib/3.5/gtools_3.8.1.zip'

'http://mirror.its.dal.ca/cran/bin/windows/contrib/3.5/gplots_3.0.1.1.zip'

'http://mirror.its.dal.ca/cran/bin/windows/contrib/3.5/ROCR_1.0-7.zip'

> library(ROCR)

Loading required package: gplots

Attaching package: 'gplots'

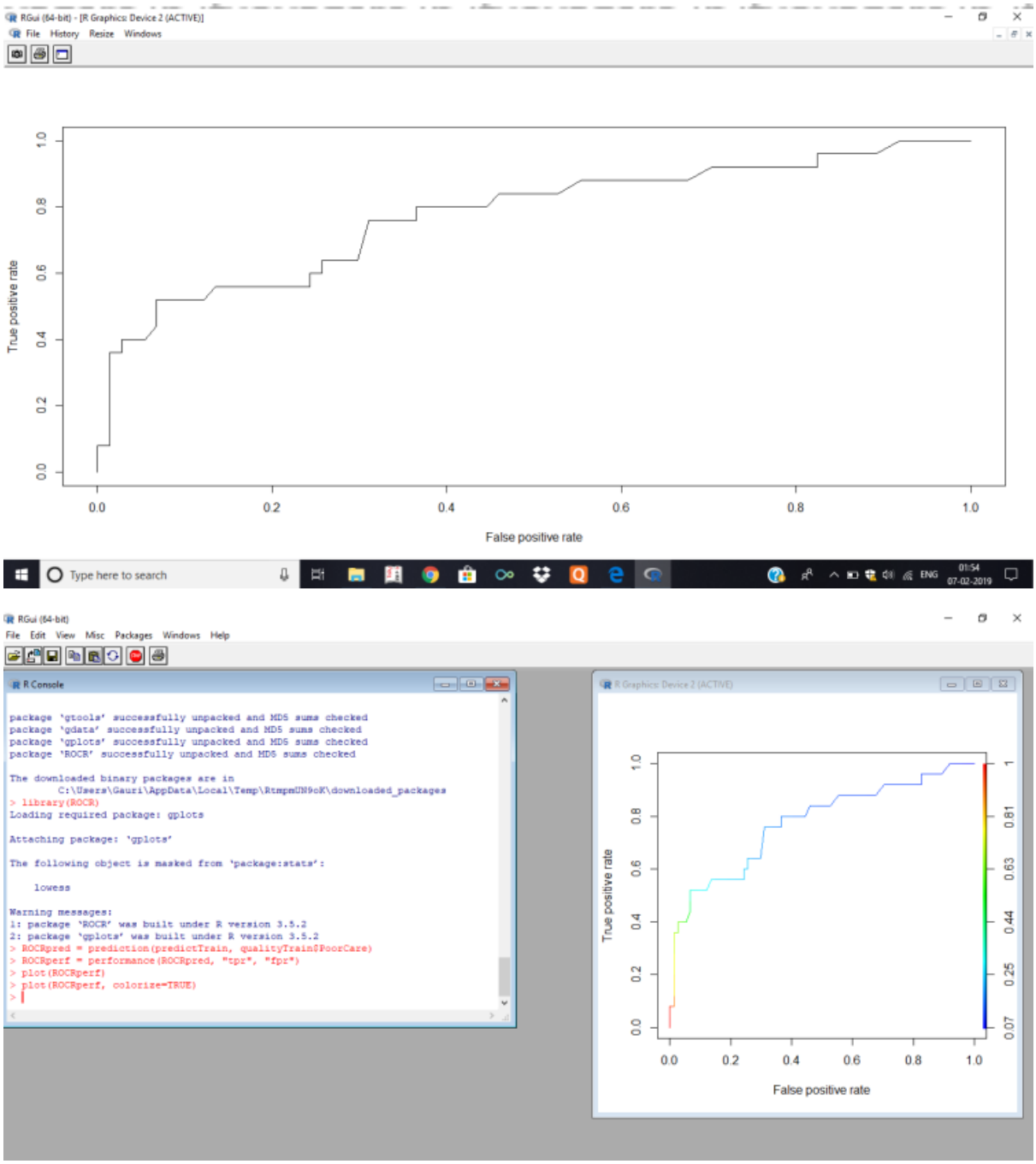
The following object is masked from 'package:stats':

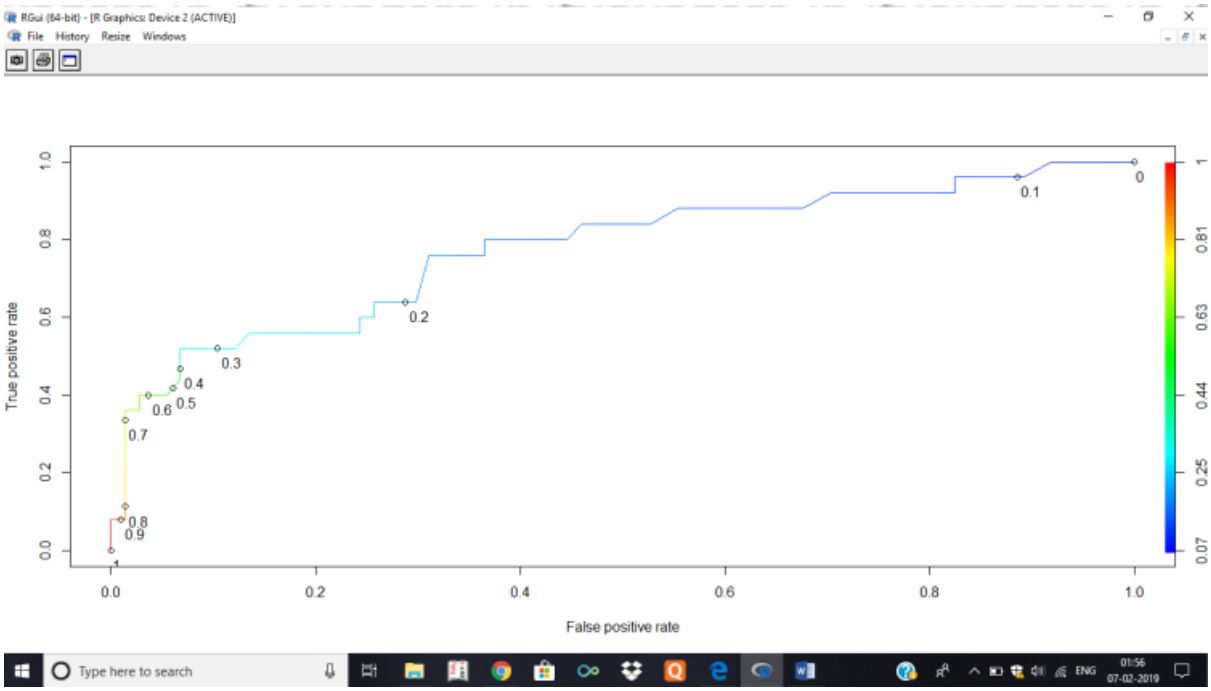
Step 5:- Warning messages:

1: package 'ROCR' was built under R version 3.5.2

2: package 'gplots' was built under R version 3.5.2

```
> ROCRpred = prediction(predictTrain, qualityTrain$PoorCare)
> ROCRperf = performance(ROCRpred, "tpr", "fpr")
> plot(ROCRperf)
> plot(ROCRperf, colorize=TRUE)
> plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
>
```




RGui (64-bit) - [R Graphics: Device 2 (ACTIVE)]
File History Resize Windows
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.81
0.63
0.44
0.25
0.07
Type here to search
ENG
01:56
07-02-2019

PRATICAL-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:-

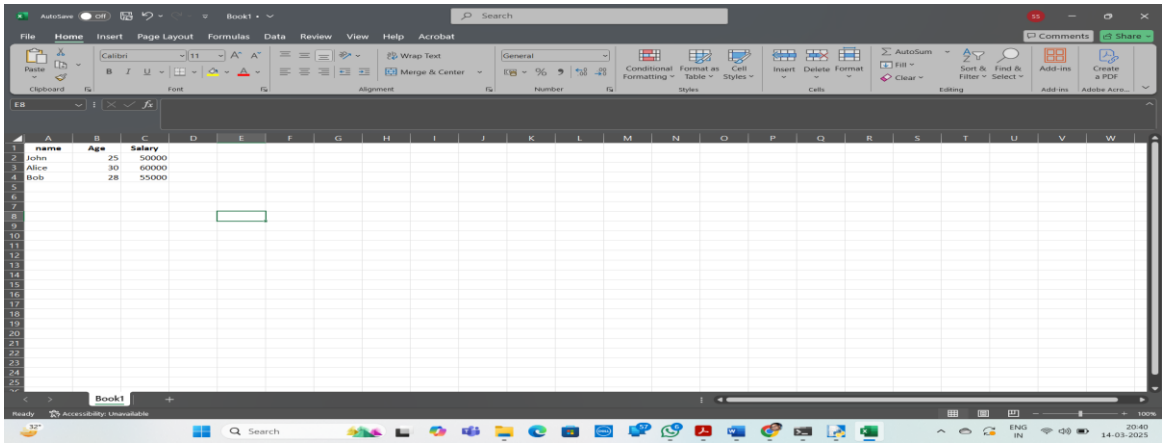
Step 1:- Install Required Library

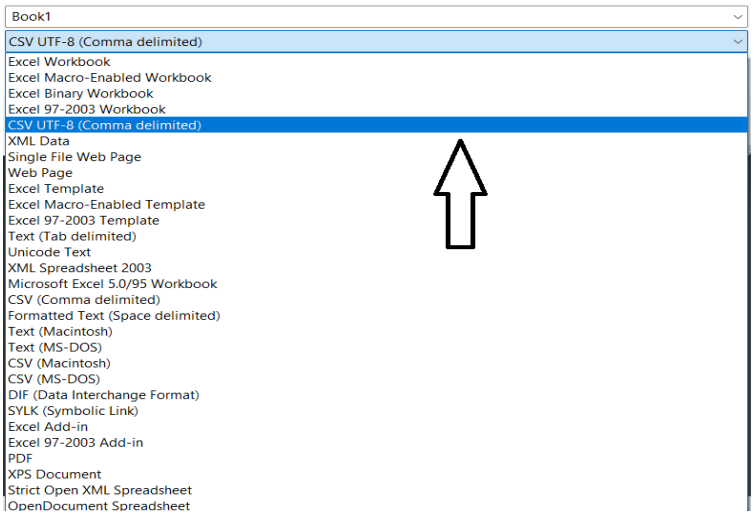
Before running the program, you need to install Pandas.
Open a command prompt or terminal and type:

```
pip install pandas
```

Step 2:- Open MS EXCEL

Prepare a CSV File



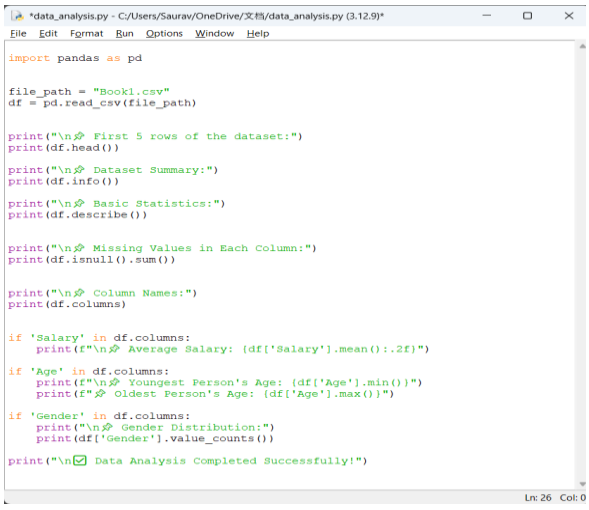


Step 3:- Open PYTHON IDLE

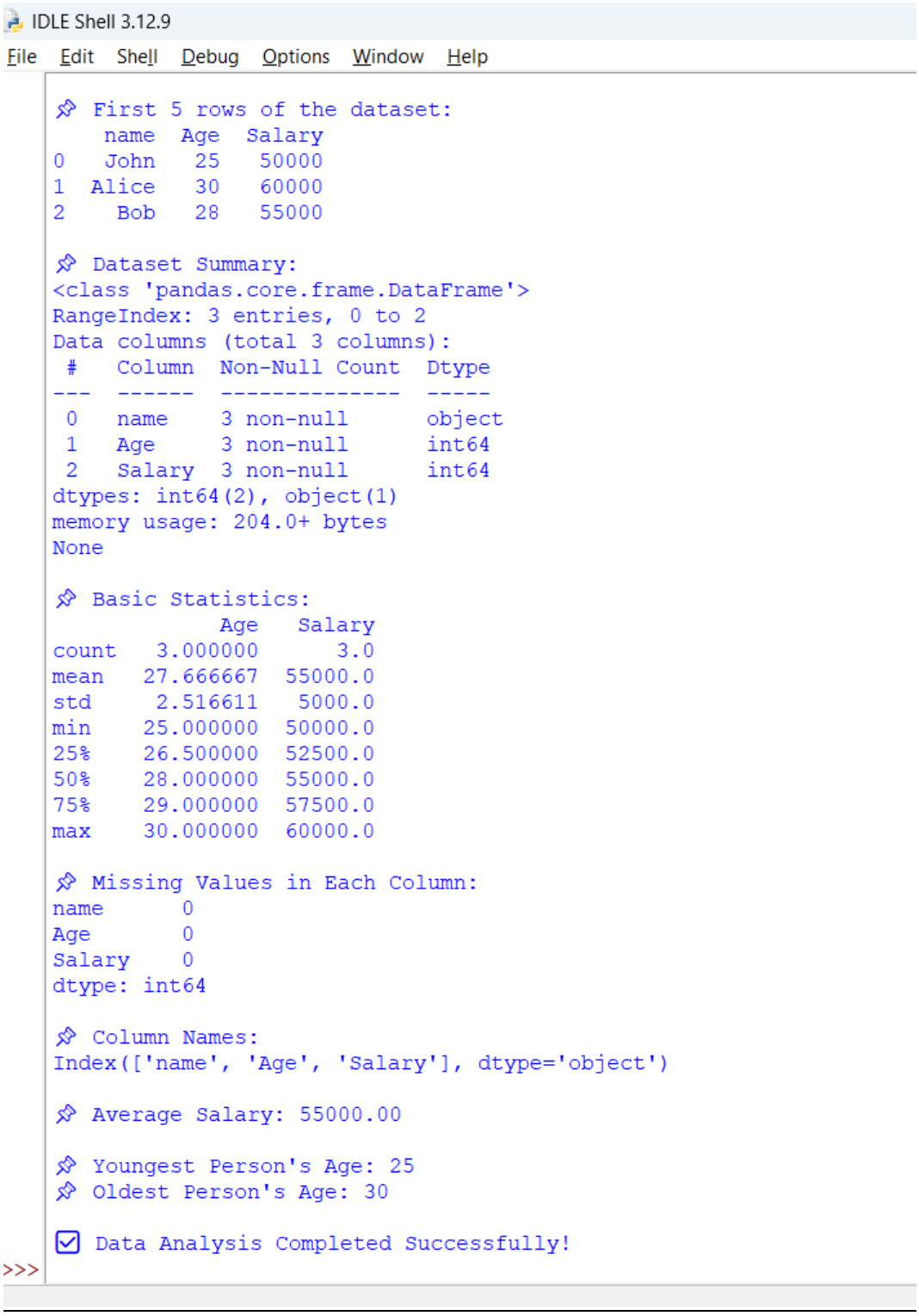
Write the Python Code

```
import pandas as pd

file_path = "Book1.csv”
```

```
df = pd.read_csv(file_path)

print("\n📌 First 5 rows of the dataset:")

print(df.head())  # Show the first 5 rows

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!")
```



Output:-


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!
>>>

PRATICAL-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

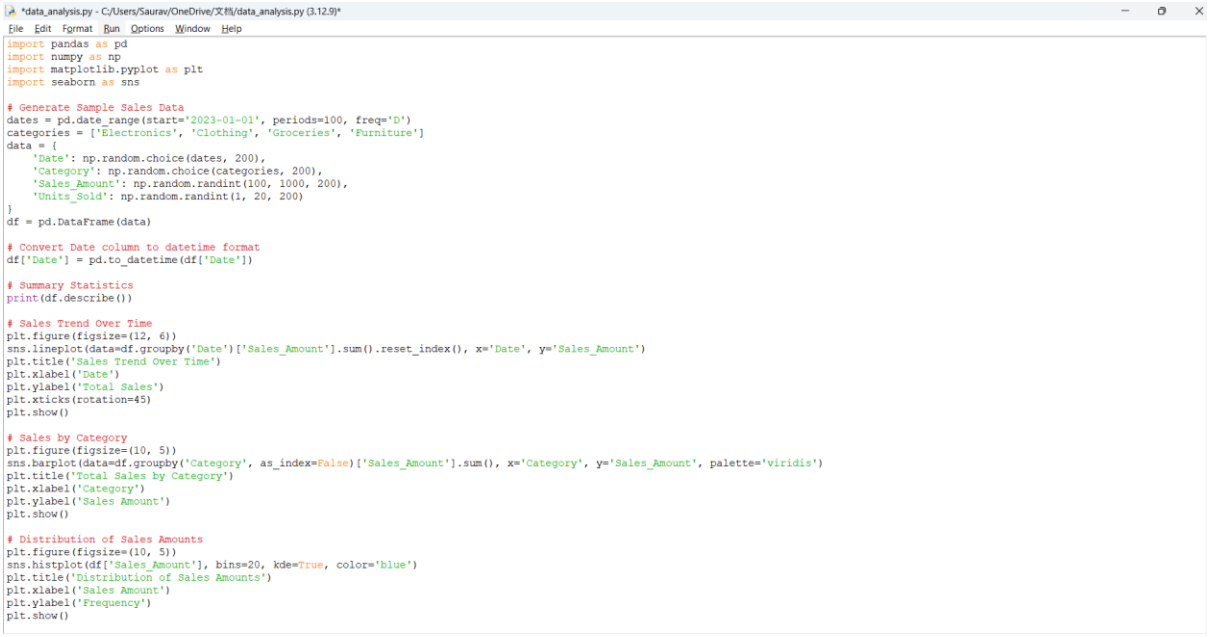
Now, open your Python script and import the required libraries:

```
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:-

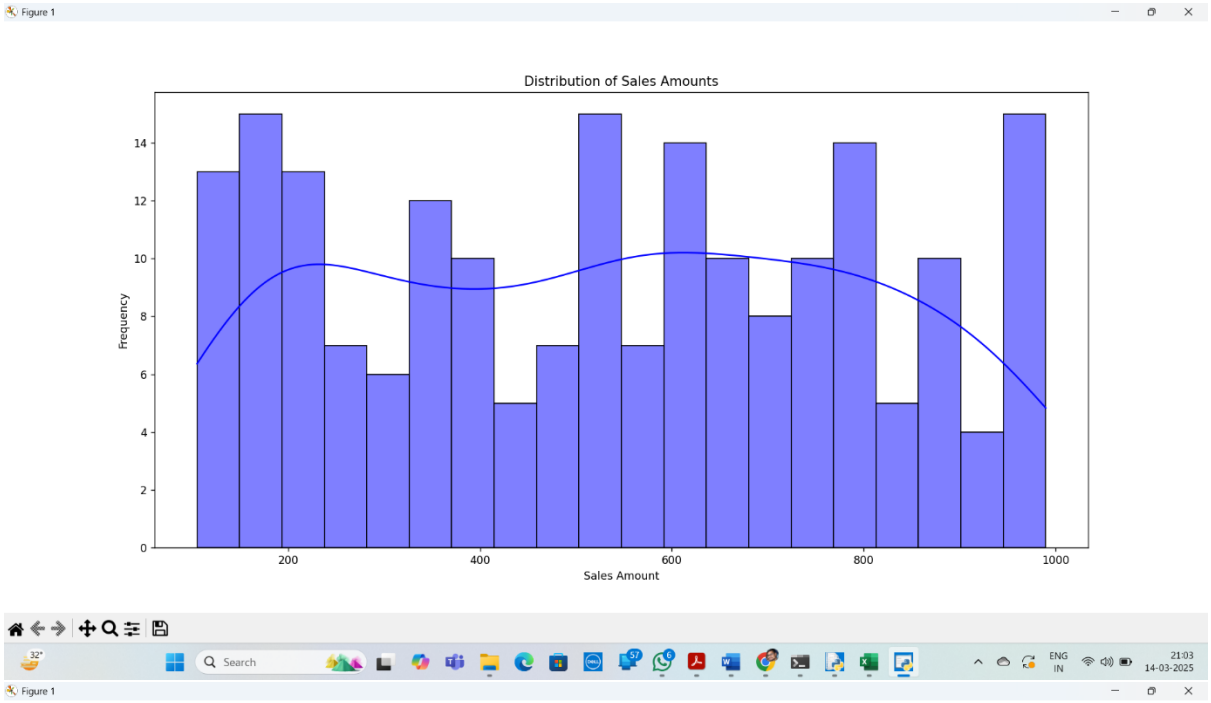


Figure 1
Distribution of Sales Amounts
Frequency
Sales Amount
0
2
4
6
8
10
12
14
200
400
600
800
1000
Search
ENG
IN
21:03
14-03-2025

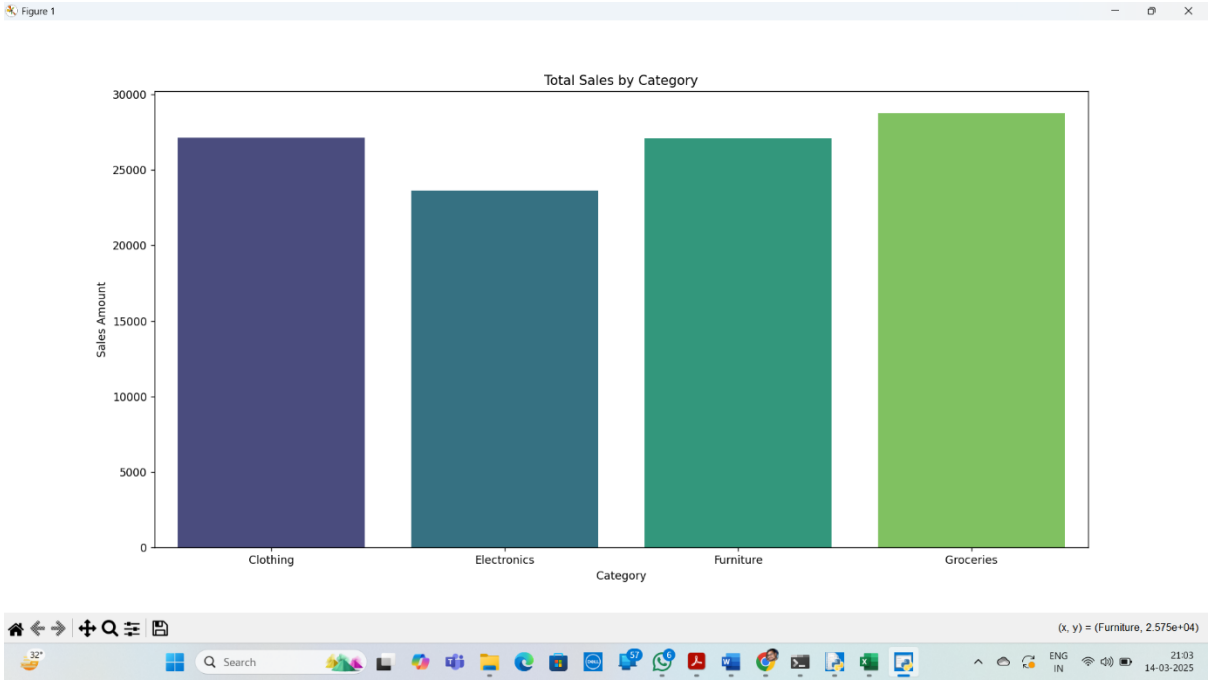


Figure 1
Total Sales by Category
Sales Amount
Category
0
5000
10000
15000
20000
25000
30000
Clothing
Electronics
Furniture
Groceries
(x, y) = (Furniture, 2.575e+04)
Search
ENG
IN
21:03
14-03-2025

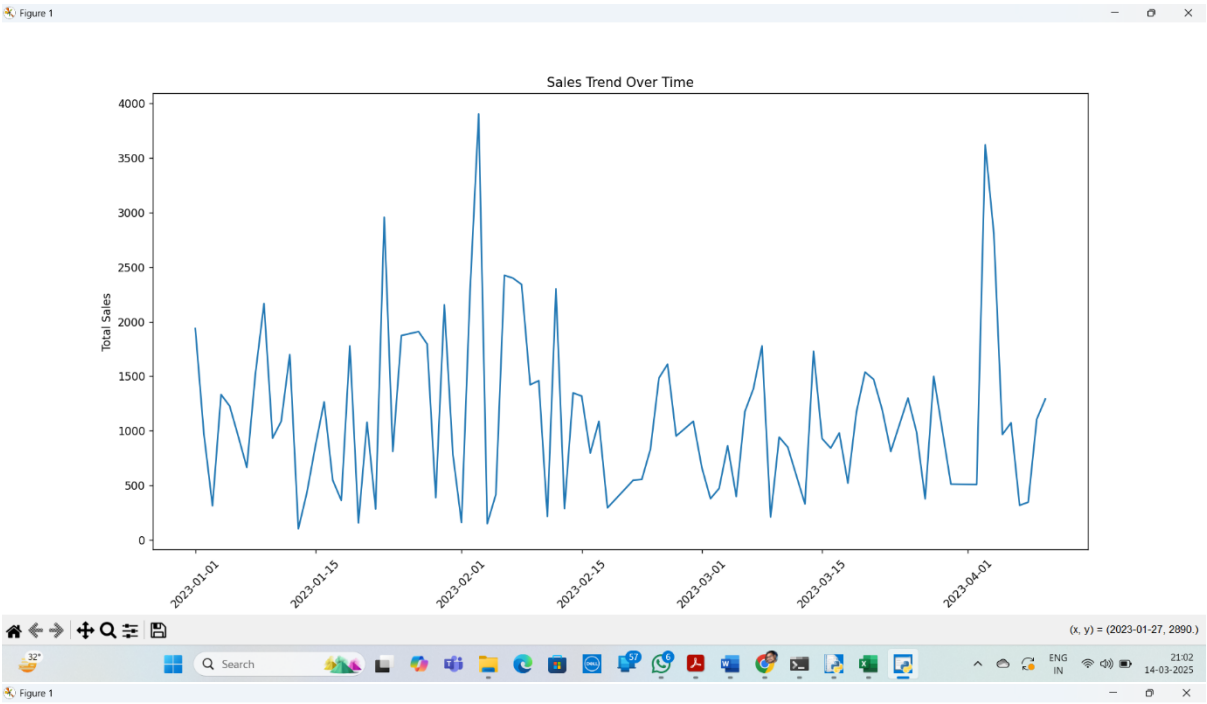


Figure 1
Sales Trend Over Time
Total Sales
4000
3500
3000
2500
2000
1500
1000
500
0
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01
(x, y) = (2023-01-27, 2890.)
Search
ENG
IN
21:02
14-03-2025

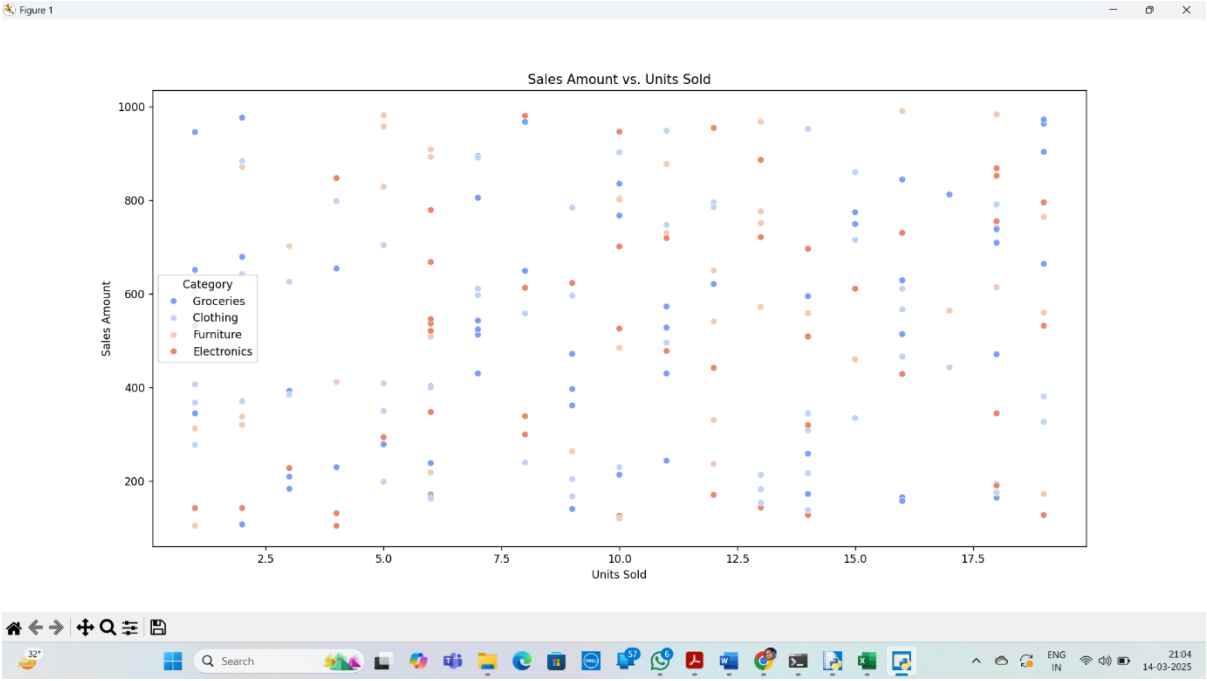


Figure 1
Sales Amount vs. Units Sold
Sales Amount
1000
800
600
400
200
Category
Groceries
Clothing
Furniture
Electronics
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Units Sold
Search
ENG
IN
21:04
14-03-2025

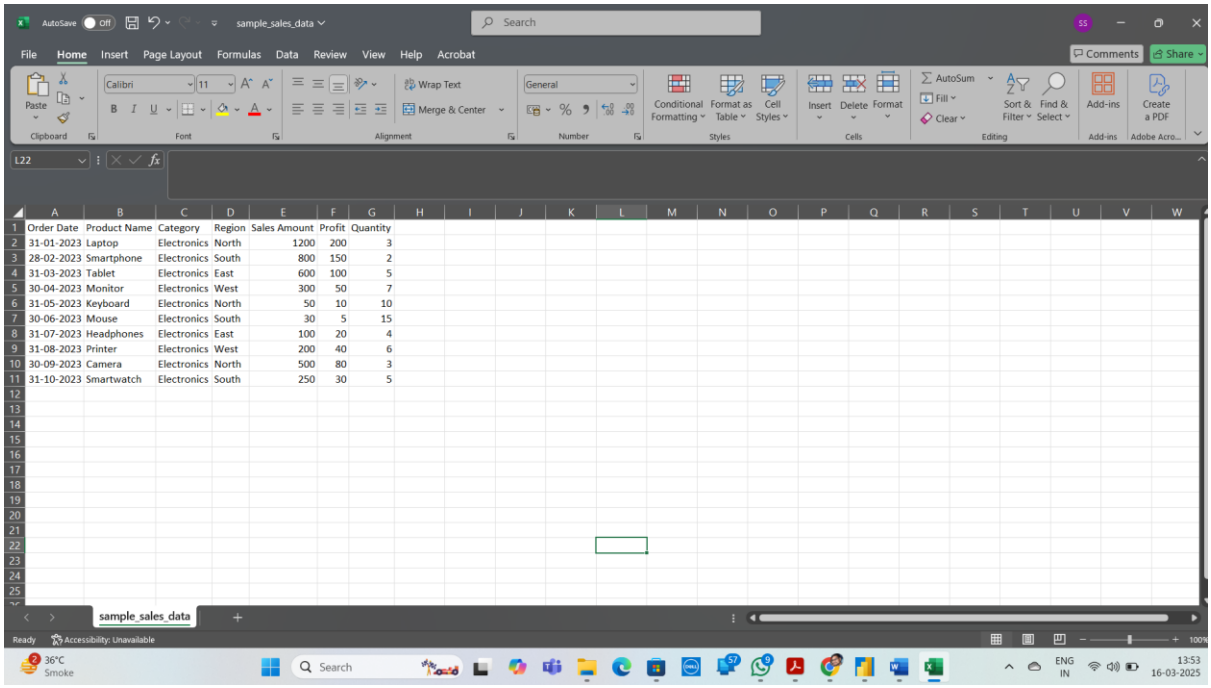
PRATICAL-8(b)

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

Steps:-

Step 1:- Download & Install Power BI

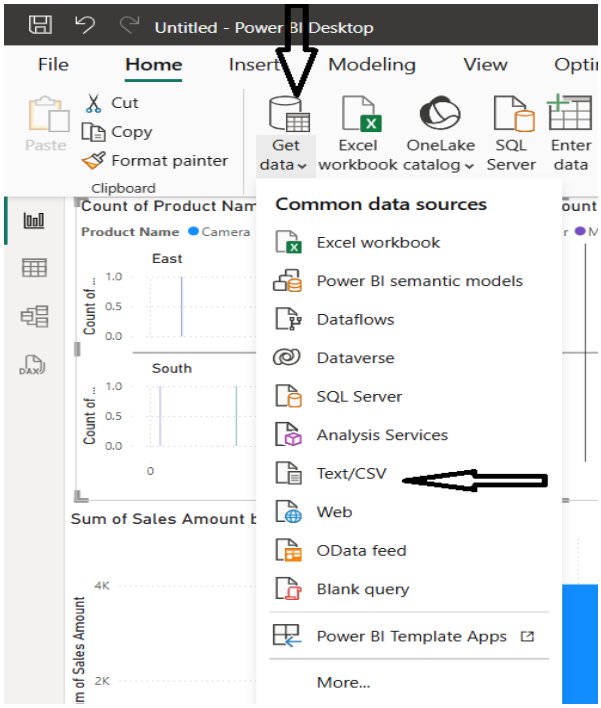
Step 2:- create a sample data



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.



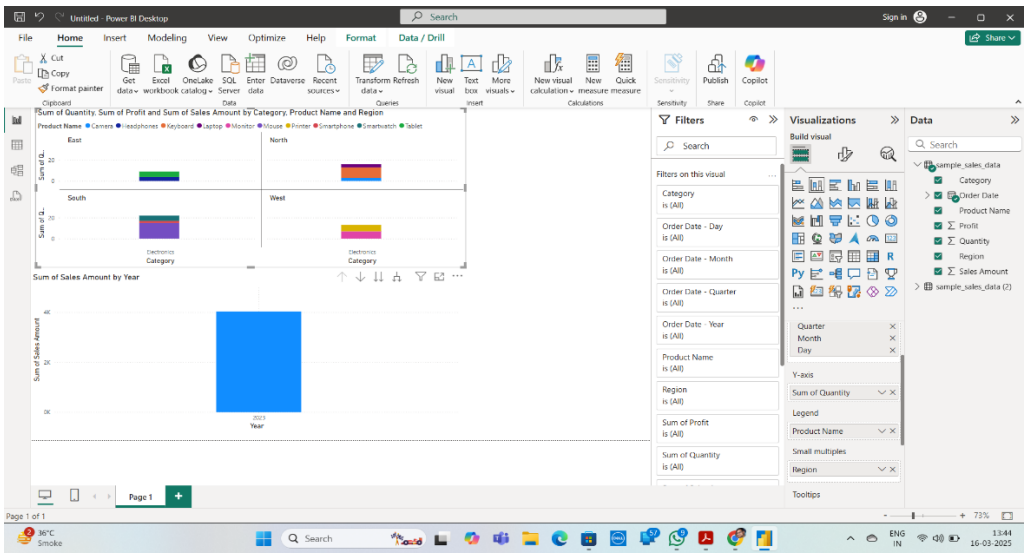
Step 4 :- Clean & Transform Data (Power Query)

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:-



PRATICAL-9

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

STEPS:-

Step 1:- Create a Staging Database

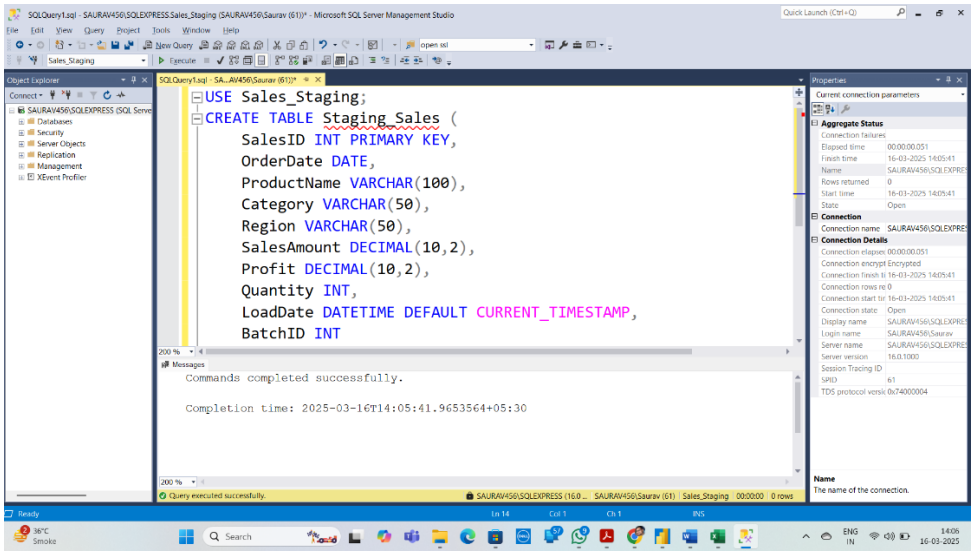
First, create a staging database to store raw sales data.

```
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**:

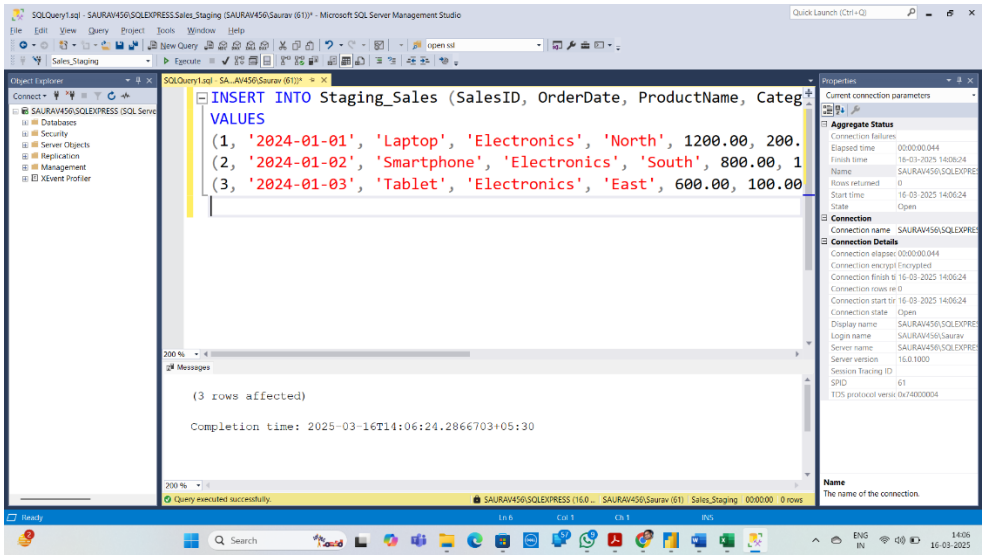
INSERT INTO Staging_Sales (SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity, BatchID)

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

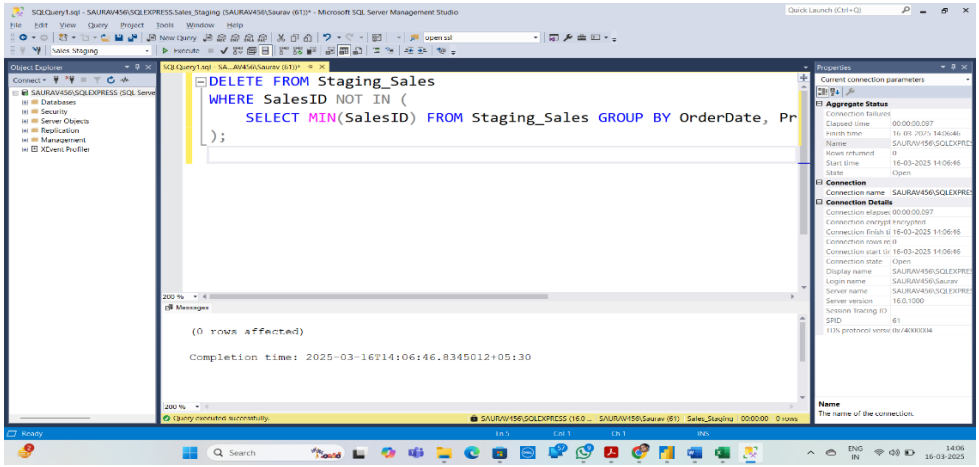
- **Remove Duplicates**

  DELETE FROM Staging_Sales

  WHERE SalesID NOT IN (

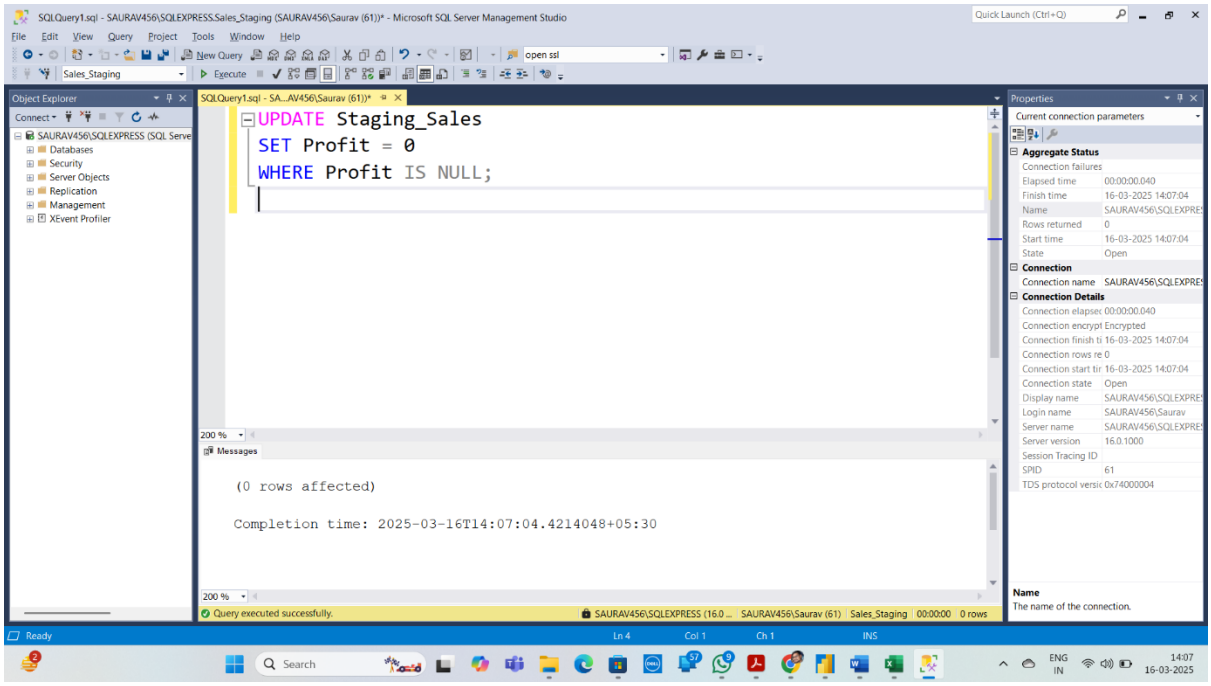
    SELECT MIN(SalesID) FROM Staging_Sales GROUP BY OrderDate, ProductName, Region

  );



- **Handle Null Values**

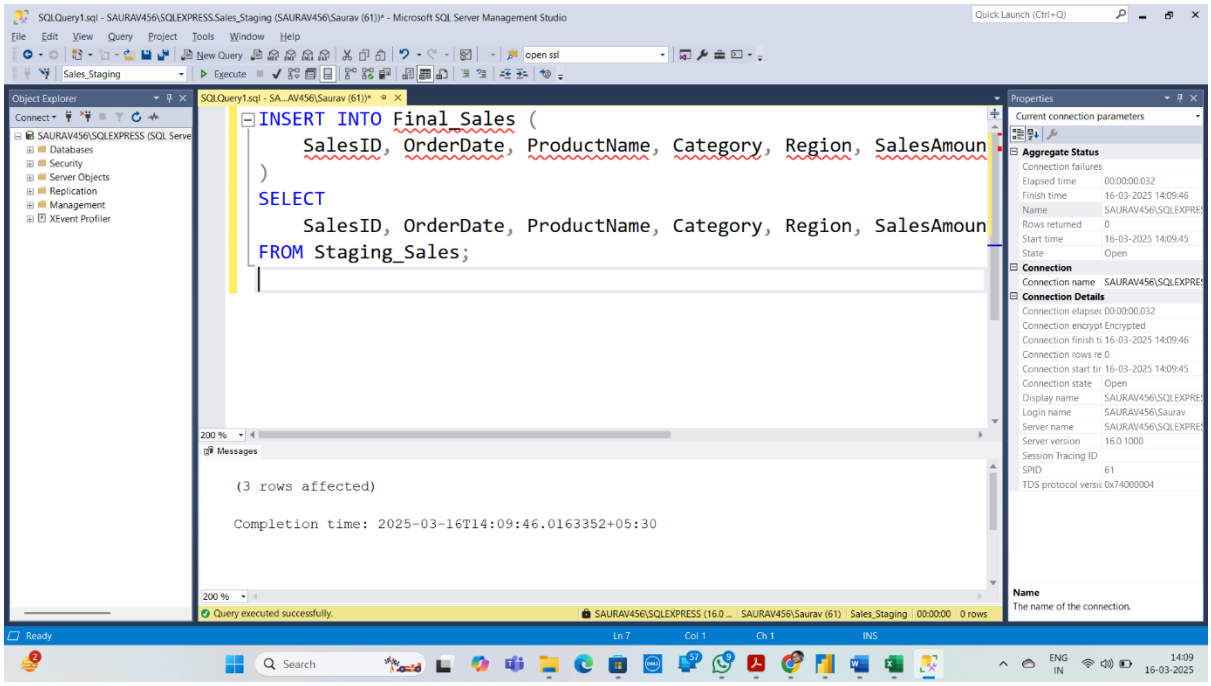
```
UPDATE Staging_Sales
SET Profit = 0
WHERE Profit IS NULL;
```



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

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

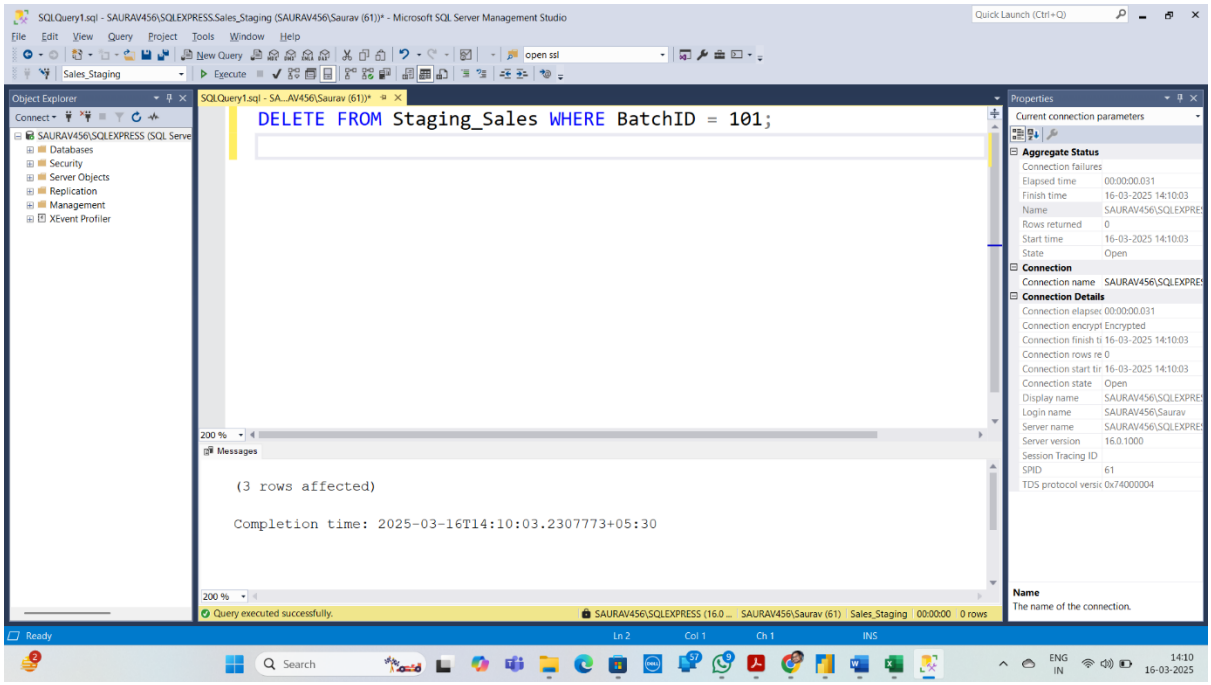
```
INSERT INTO Final_Sales (
   SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity
)
SELECT
   SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity
FROM Staging_Sales;
```



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 = 101;



PRATICAL-10

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

Steps:-

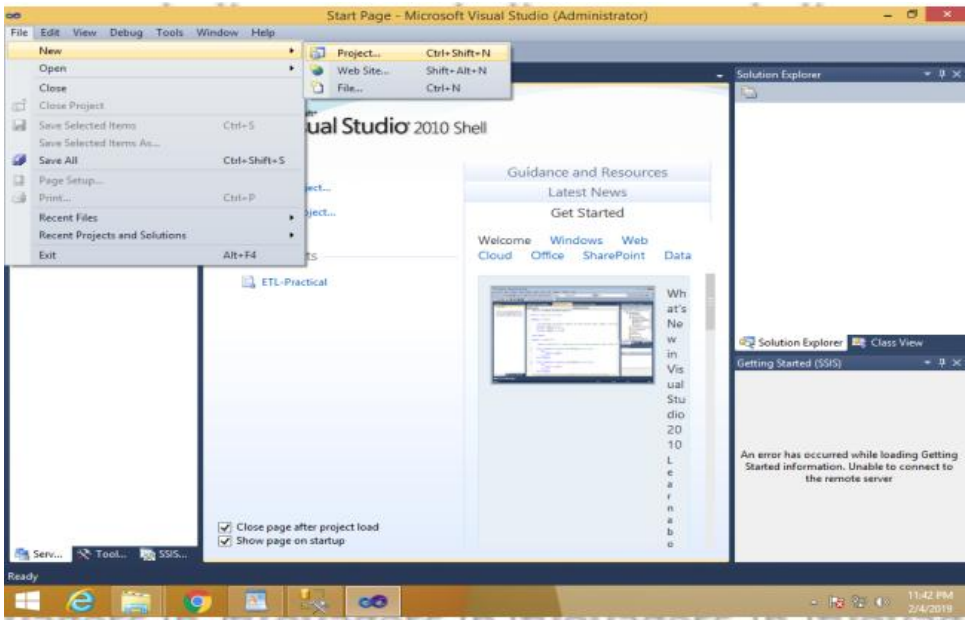
Step 1: Creating Data Warehouse

CREATED IN PRACTICAL 9

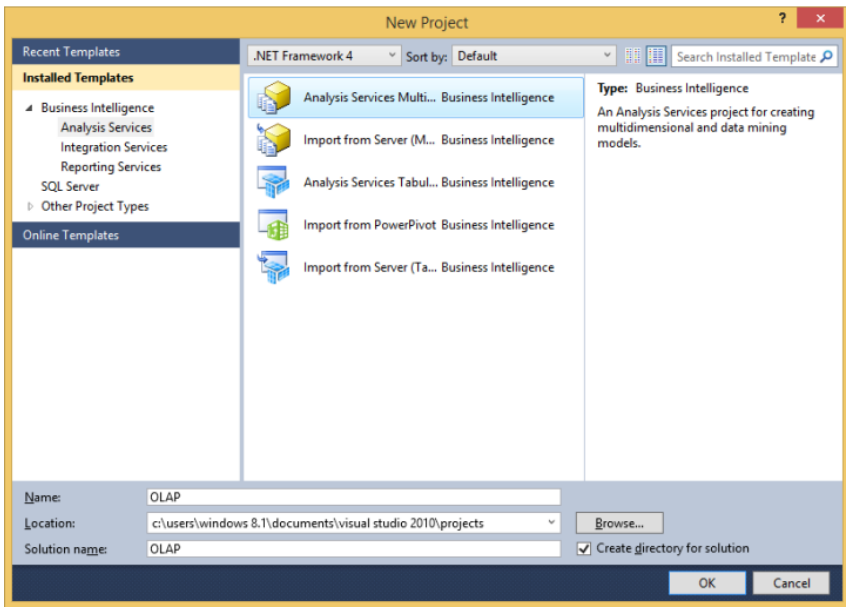
Step 2: Start SSDT environment and create New Data Source

Go to Sql Server Data Tools --> Right click and run as administrator

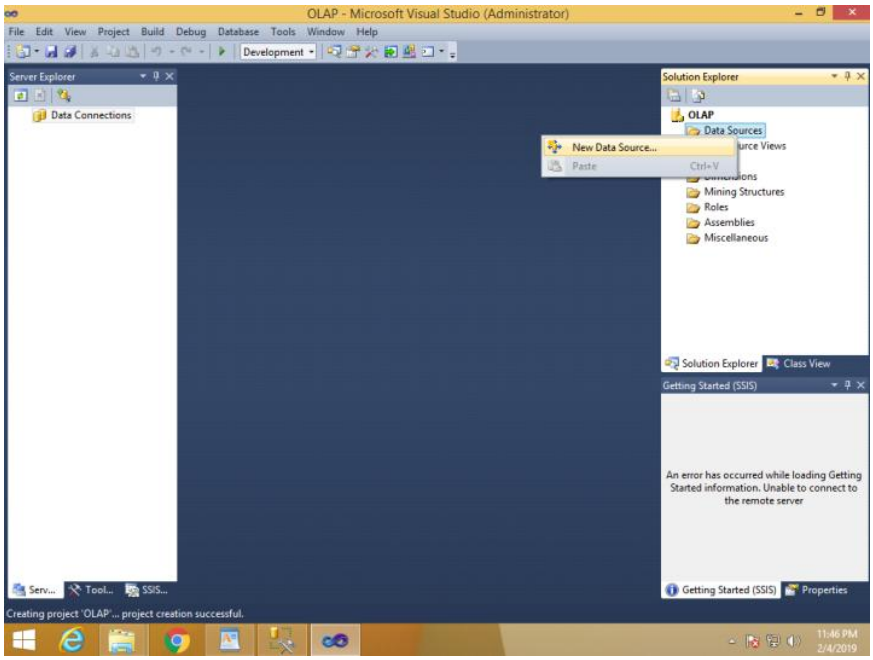
Click on File → New → Project



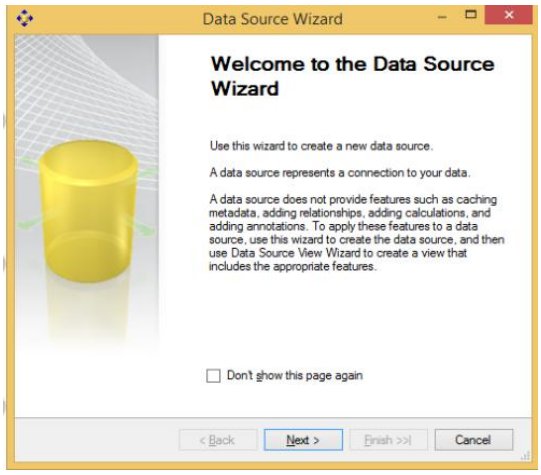
In Business Intelligence → Analysis Services Multidimensional and Data Mining models → appropriate project name → click OK



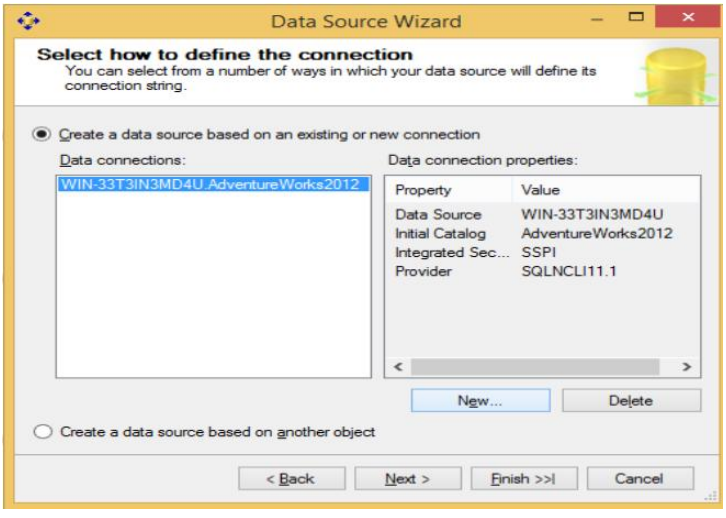
Right click on Data Sources in solution explorer → New Data Source



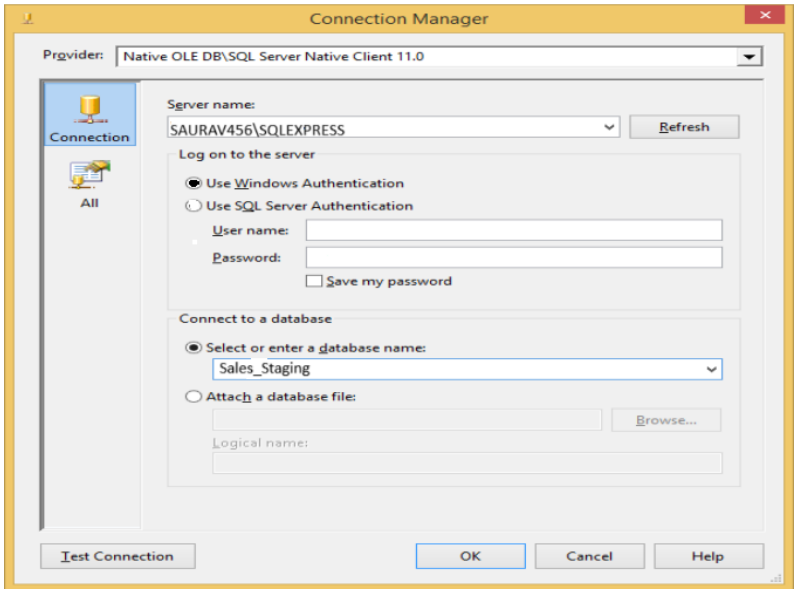
Data Source Wizard appears

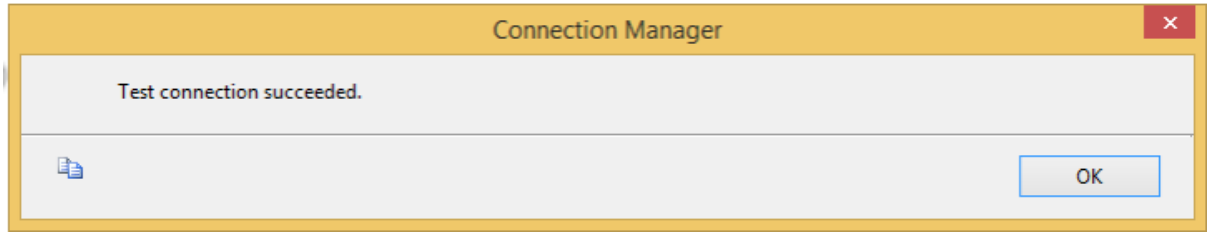


Click on New

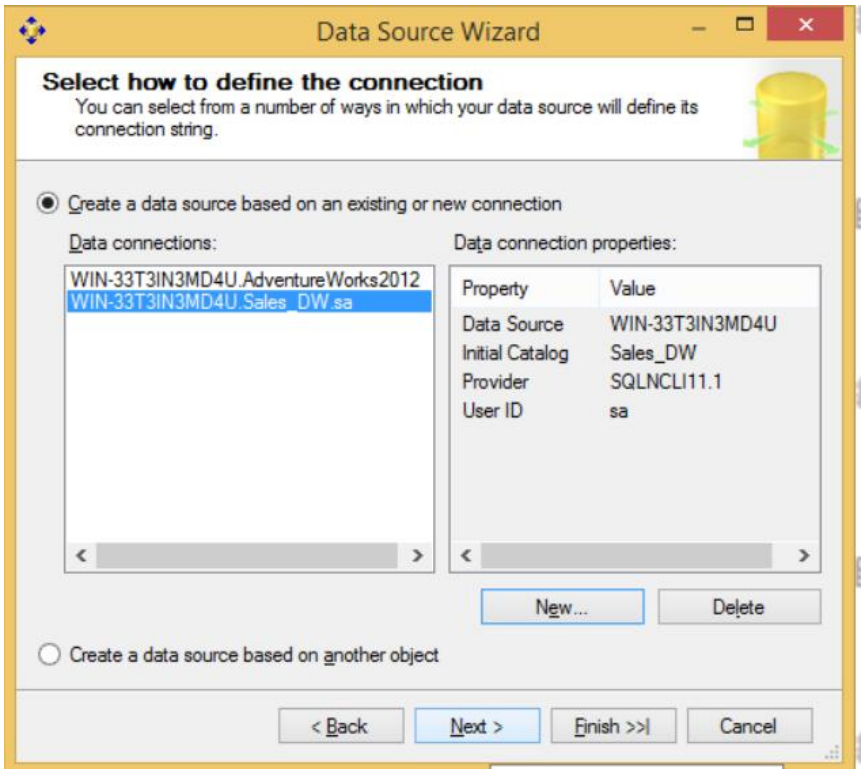


Select Server Name → select Use SQL Server Authentication → Select or enter a database name (Sales_Staging)

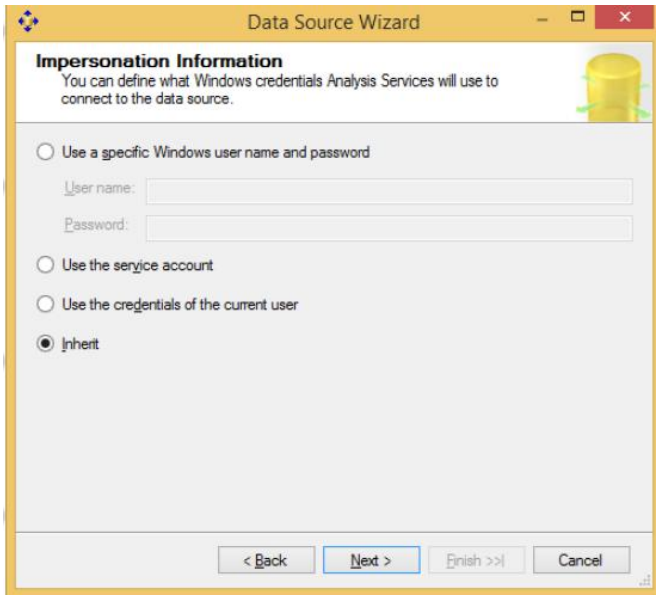




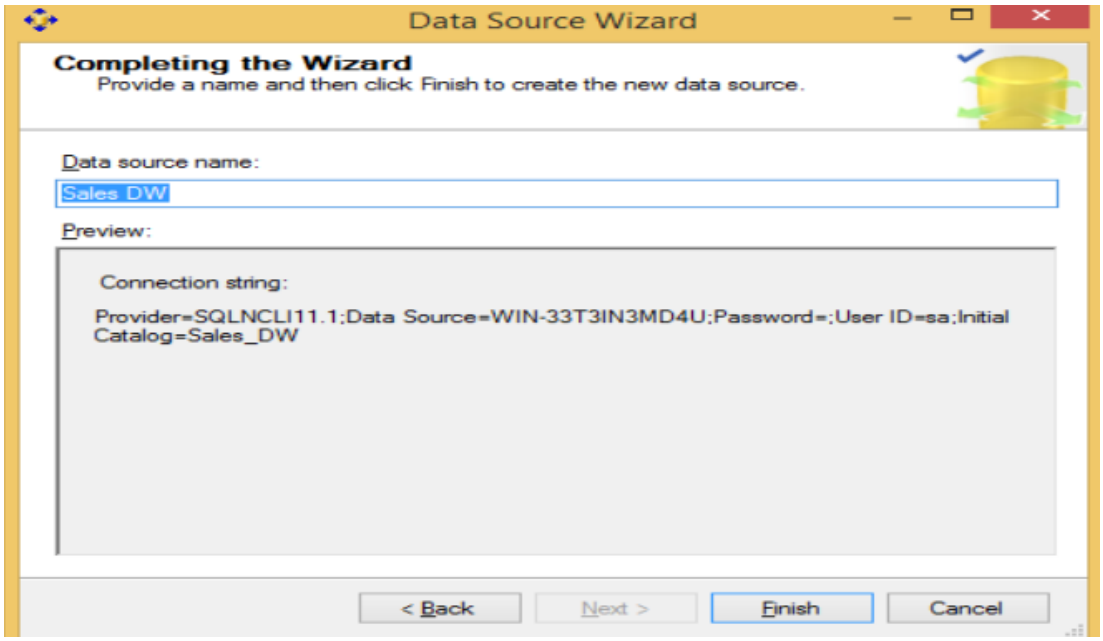
Click ok



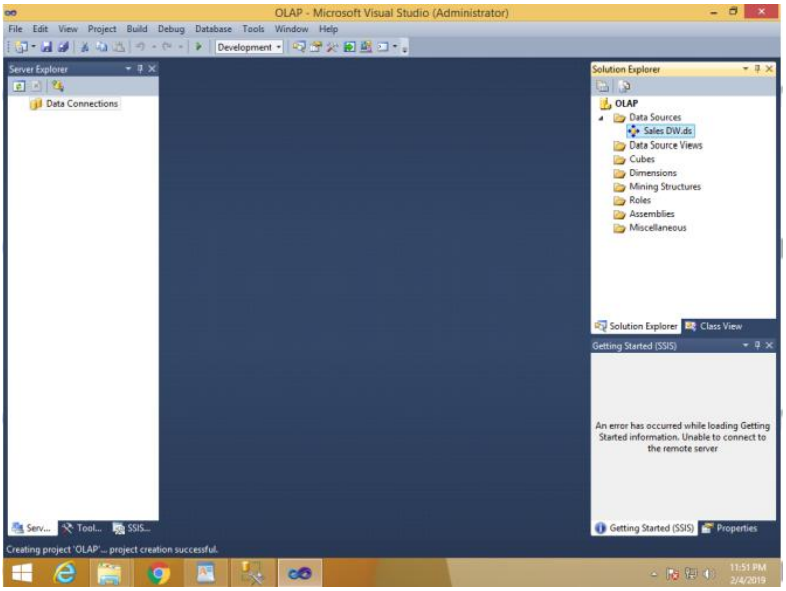
Select Inherit → Next



Click Finish

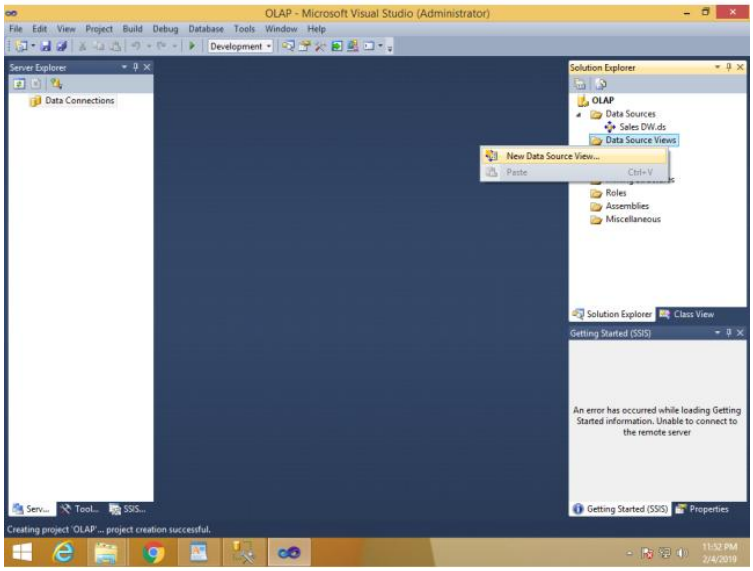


Sales_Staging.ds gets created under Data Sources in Solution Explorer

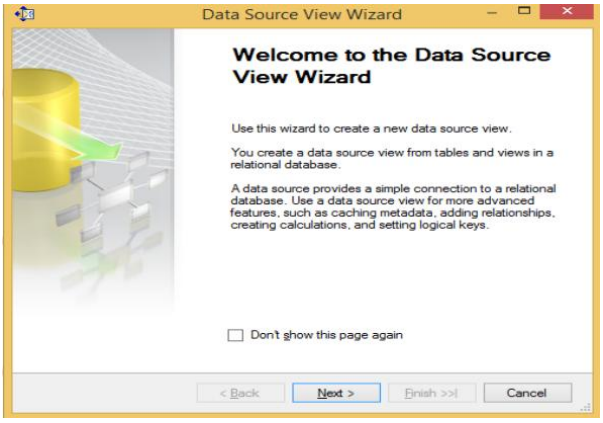


Step 3: Creating New Data Source View

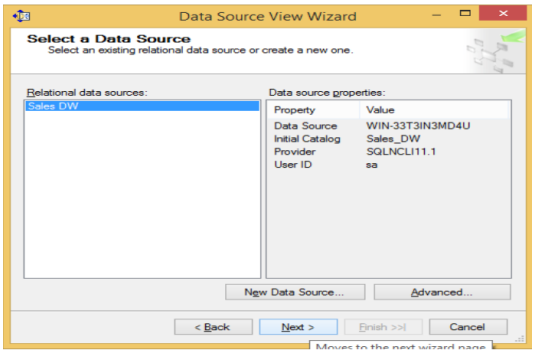
In Solution explorer right click on Data Source View → Select New Data Source View



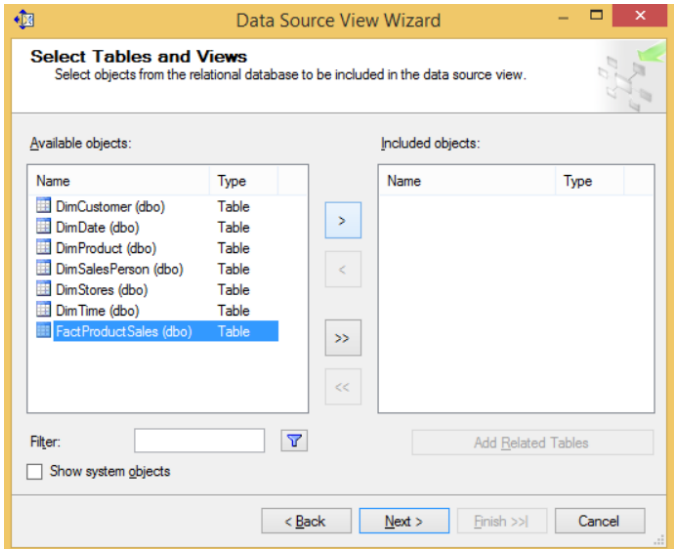
Click Next

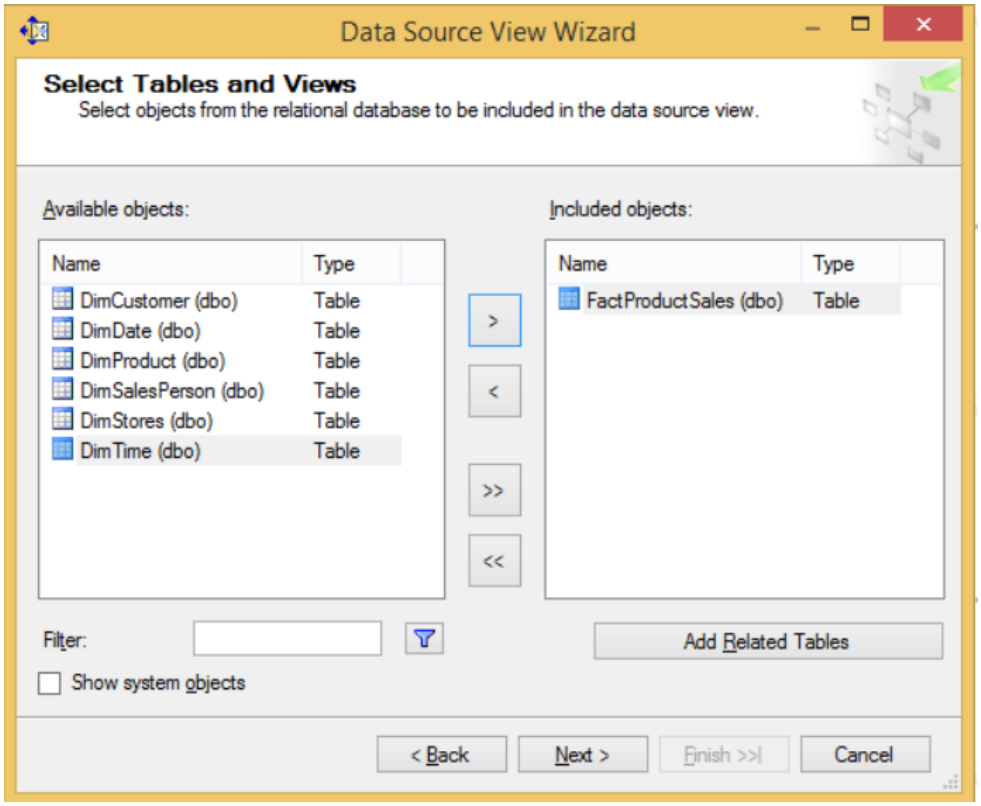


Click Next

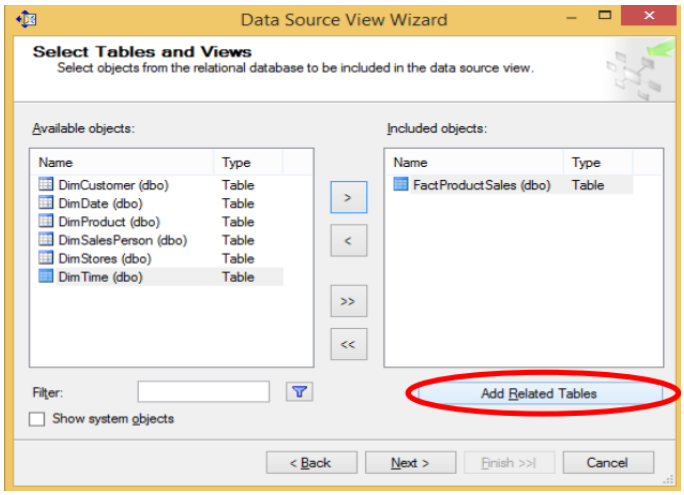


Select FactProductSales(dbo) from Available objects and put in Includes Objects by clicking on

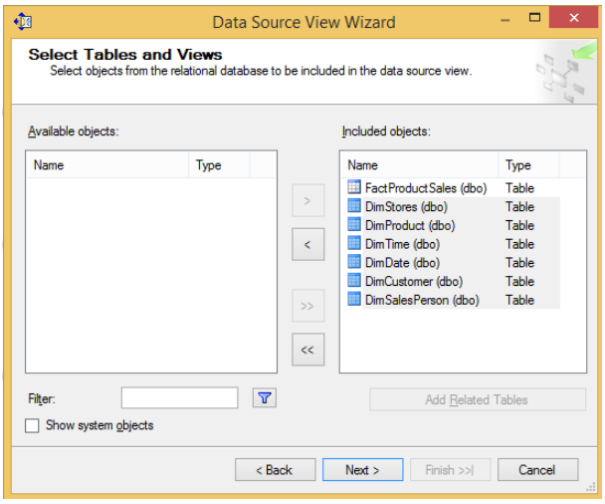




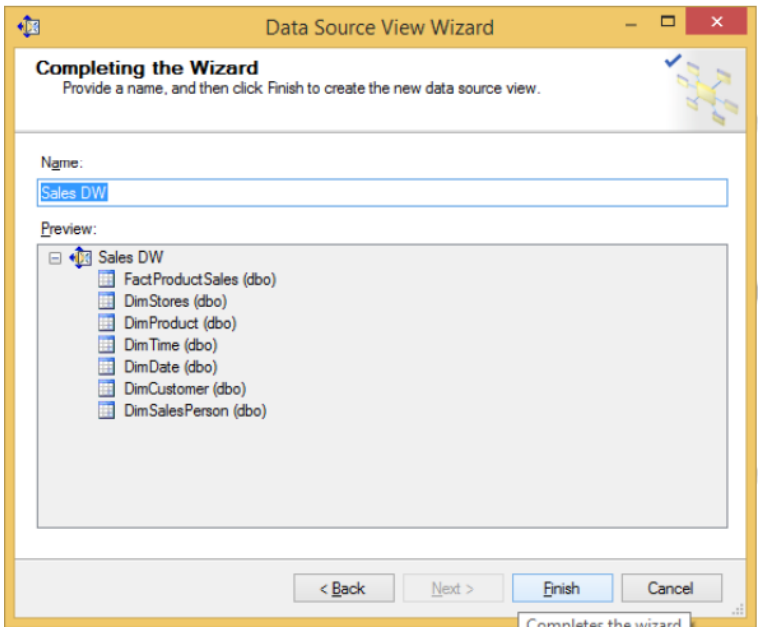
Click on Add Related Tables



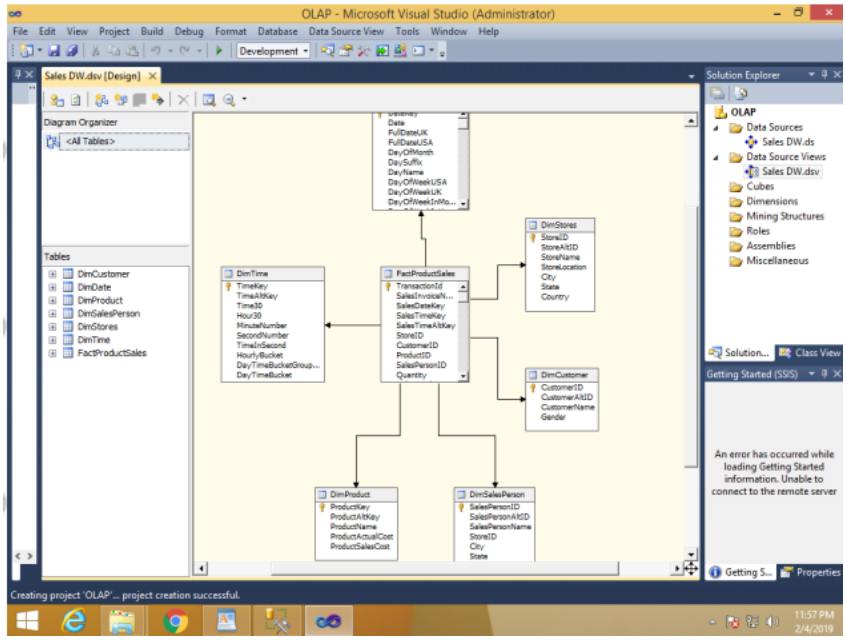
Click Next



Click Finish

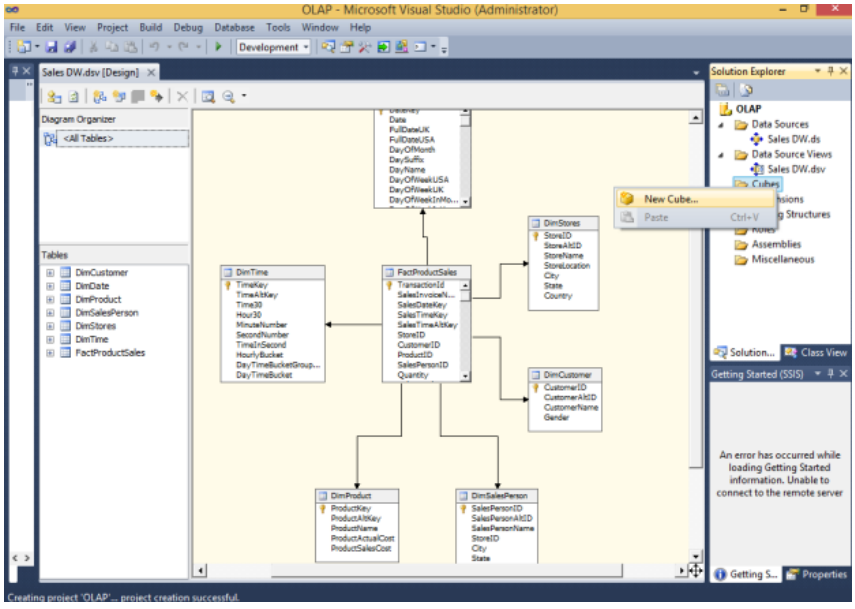


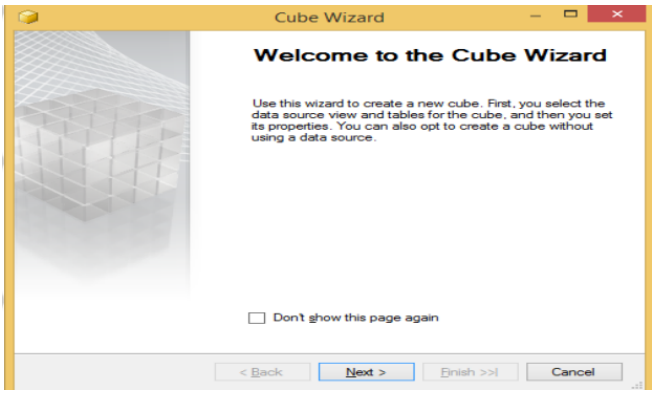
Sales_Staging.dsv appears in Data Source Views in Solution Explorer.



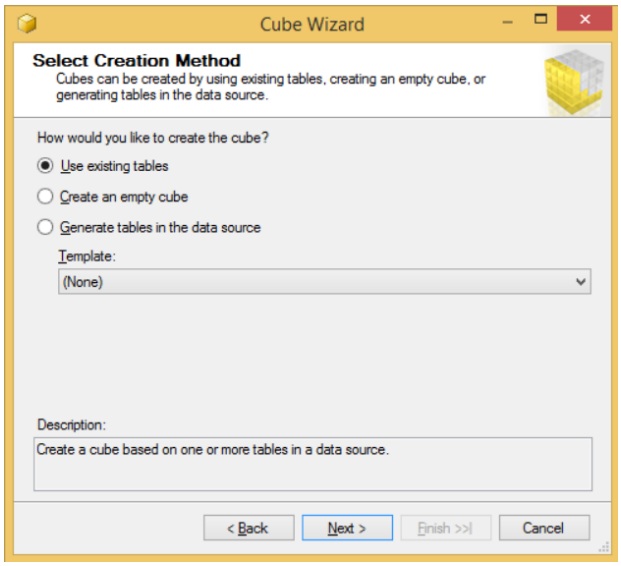
Step 4: Creating new cube

Right click on Cubes → New Cube

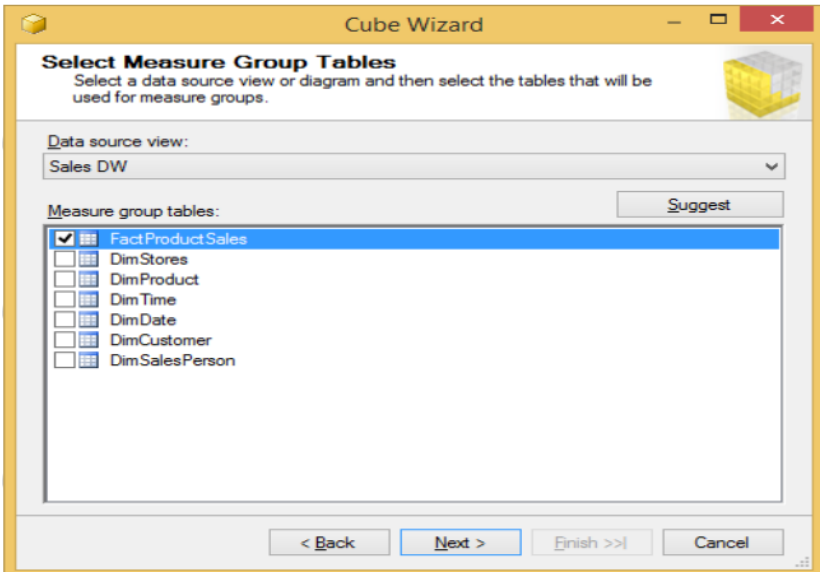




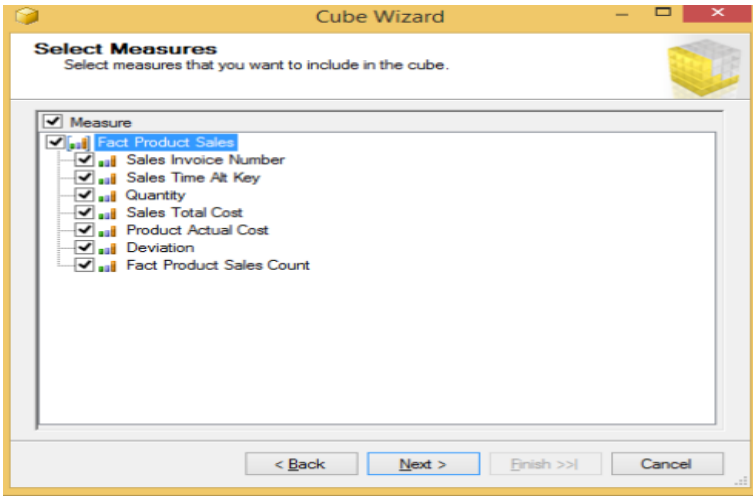
Select Use existing tables in Select Creation Method → Next



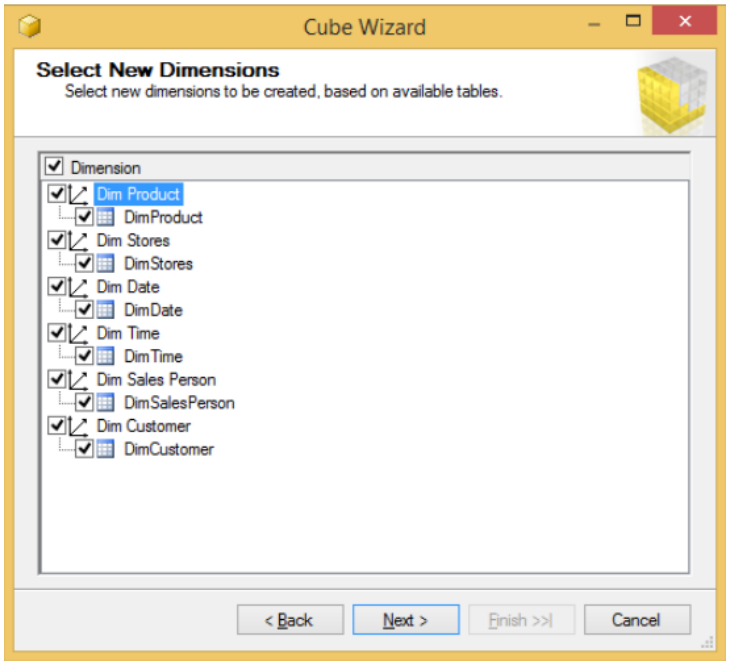
In Select Measure Group Tables → Select FactProductSales → Click Next



In Select Measures → check all measures → Next

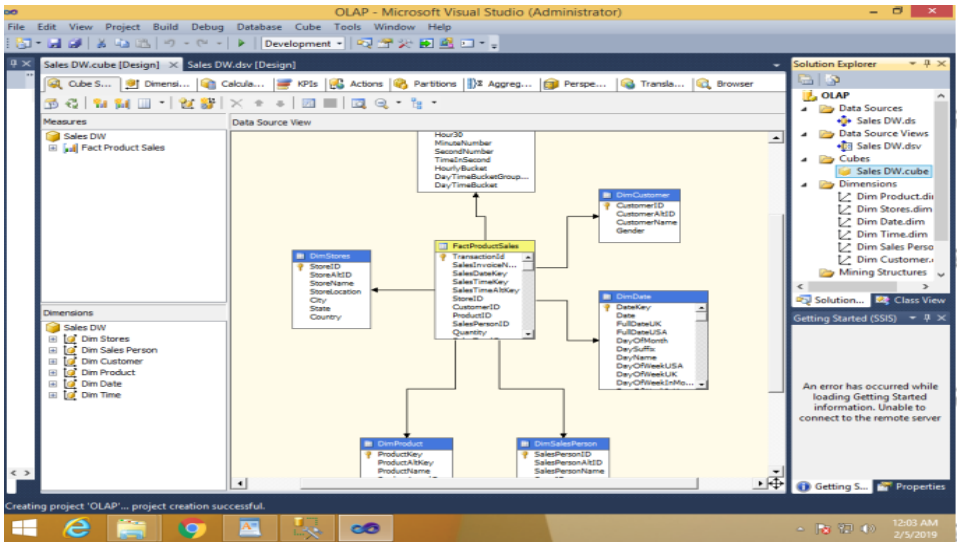


In Select New Dimensions → Check all Dimensions → Next



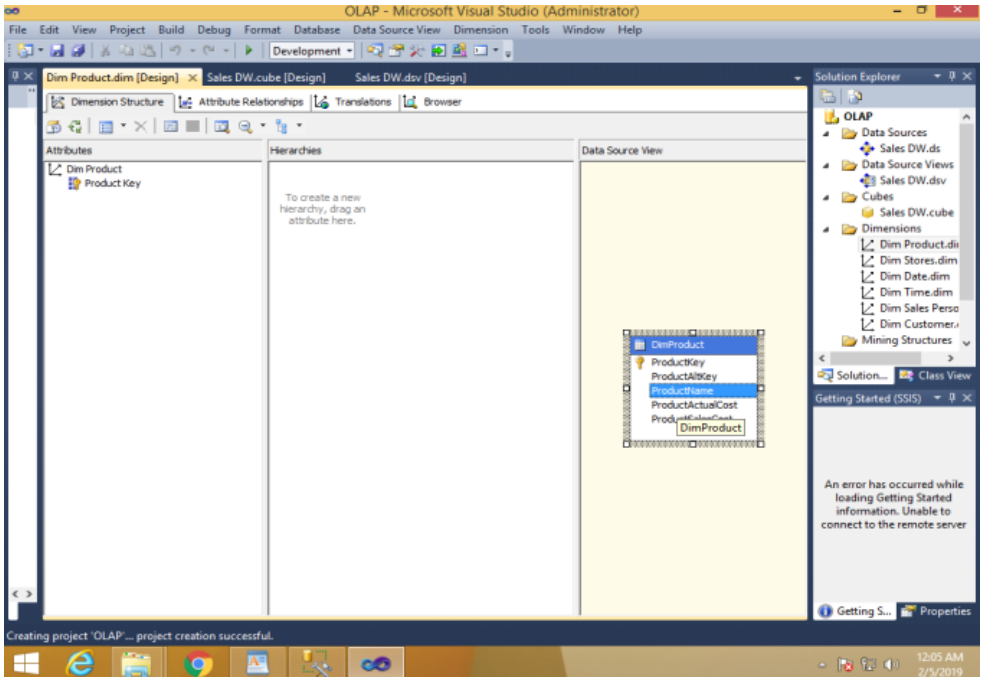
Click on Finish

Sales_Staging.cube is created

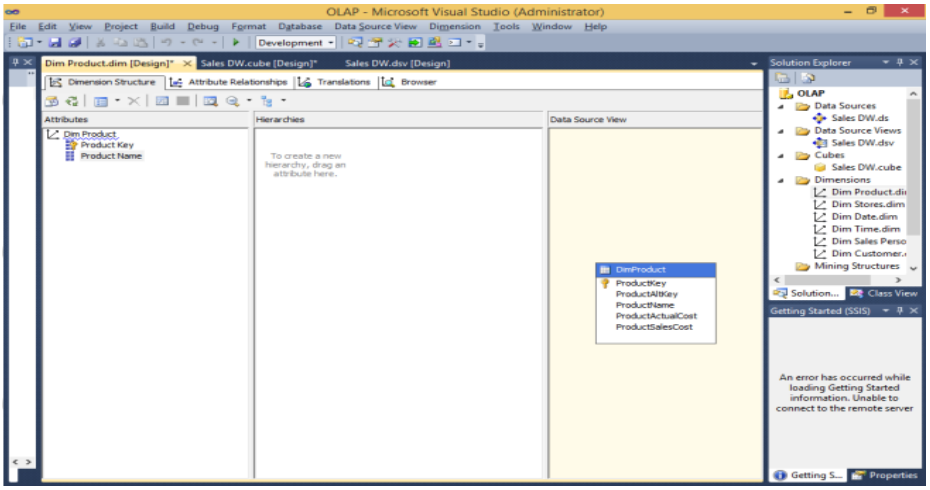


Step 5: Dimension Modification

In dimension tab → Double Click Dim Product.dim



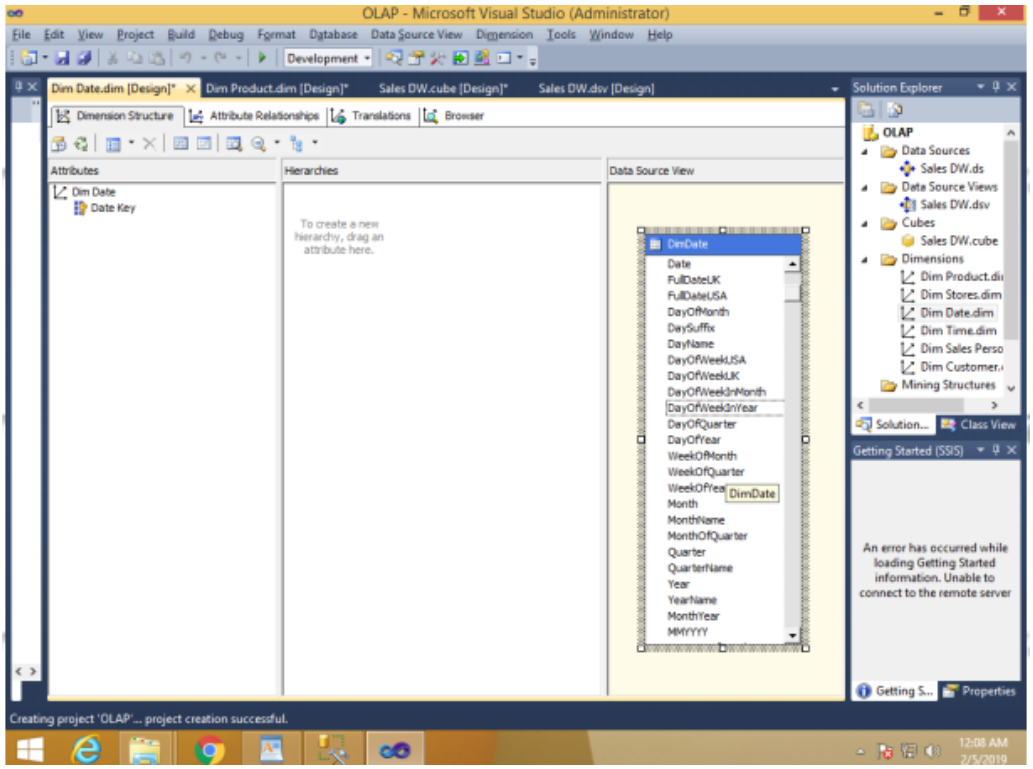
Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side

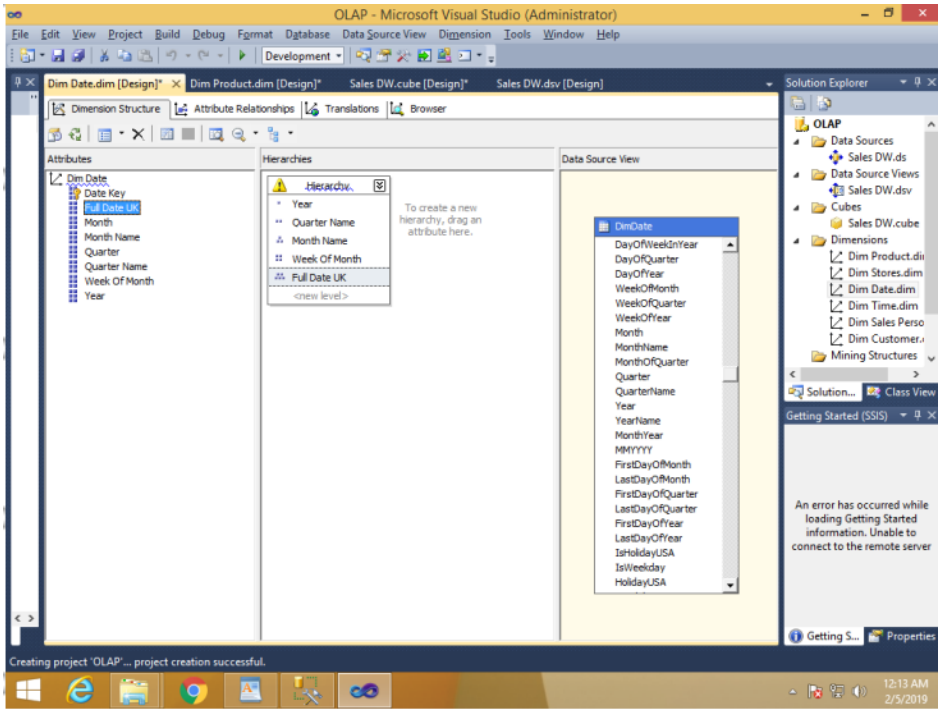


Step 6: Creating Attribute Hierarchy in Date Dimension

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.

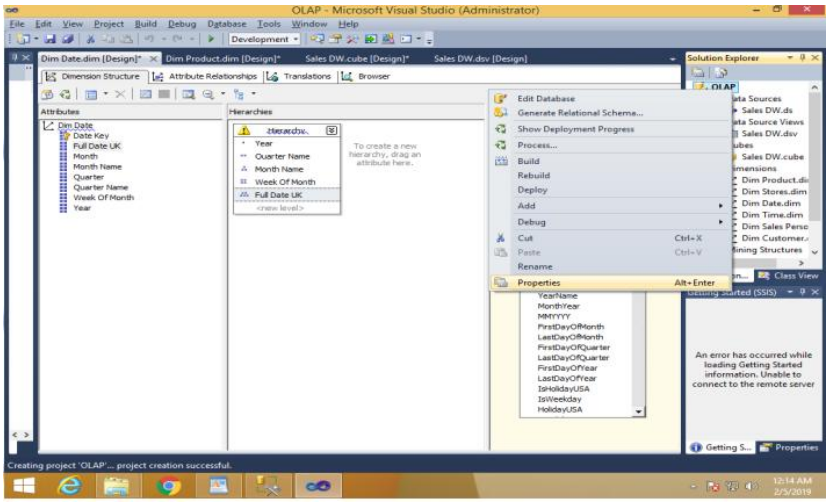
Drag fields in sequence from Attributes to Hierarchy window (Year, Quarter Name, Month Name, Week of the Month, Full Date UK)



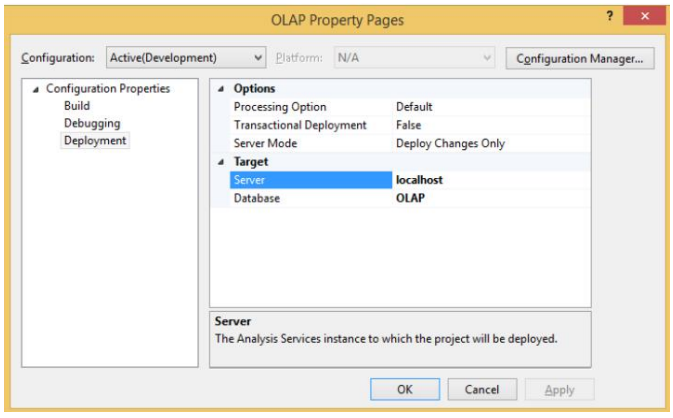


Step 7: Deploy Cube .

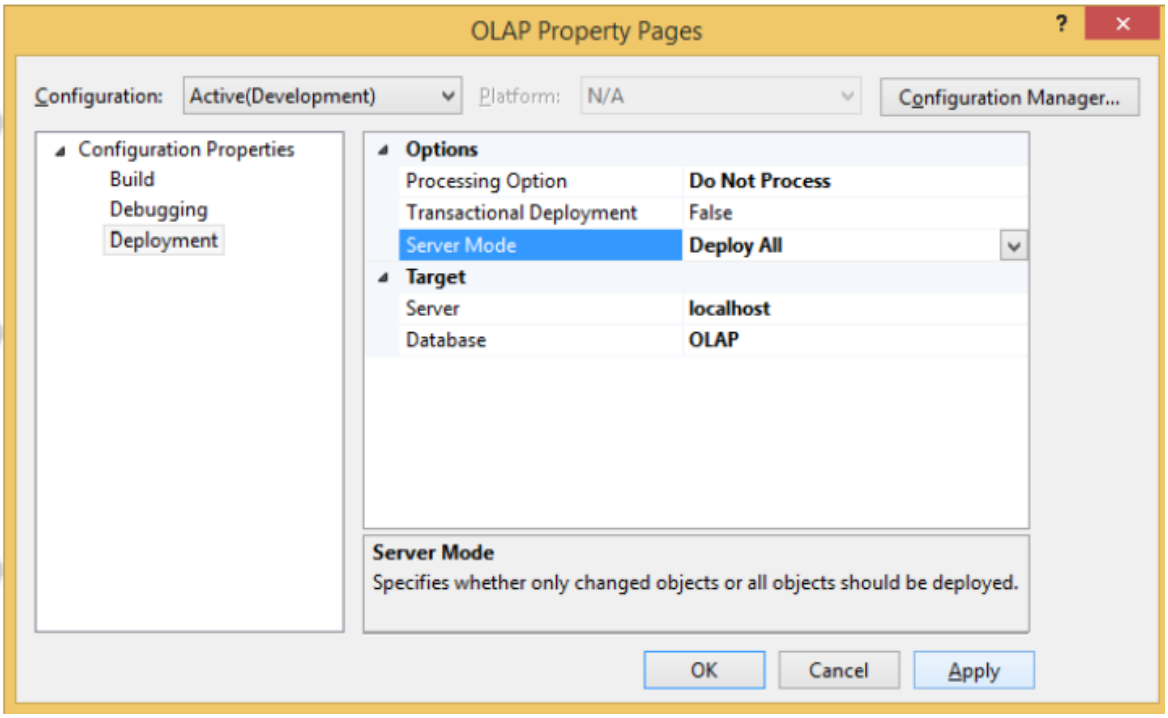
Right click on Project name → Properties



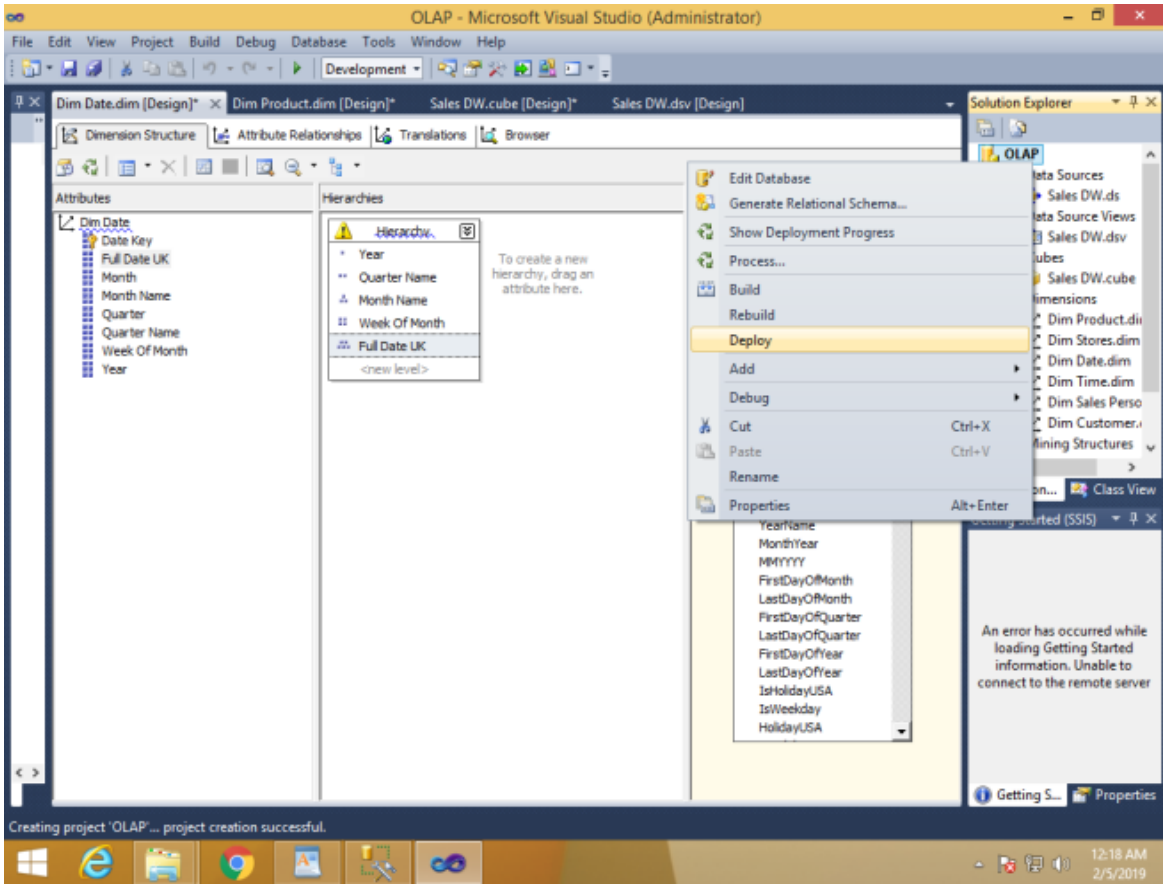
This window appaers



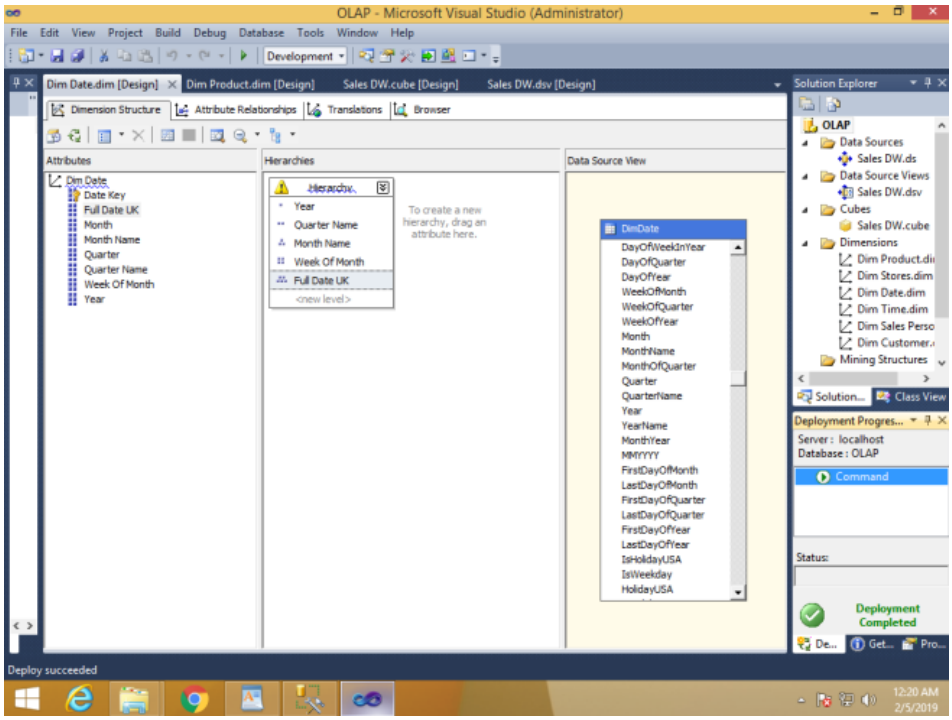
Do following changes and click on Apply & ok



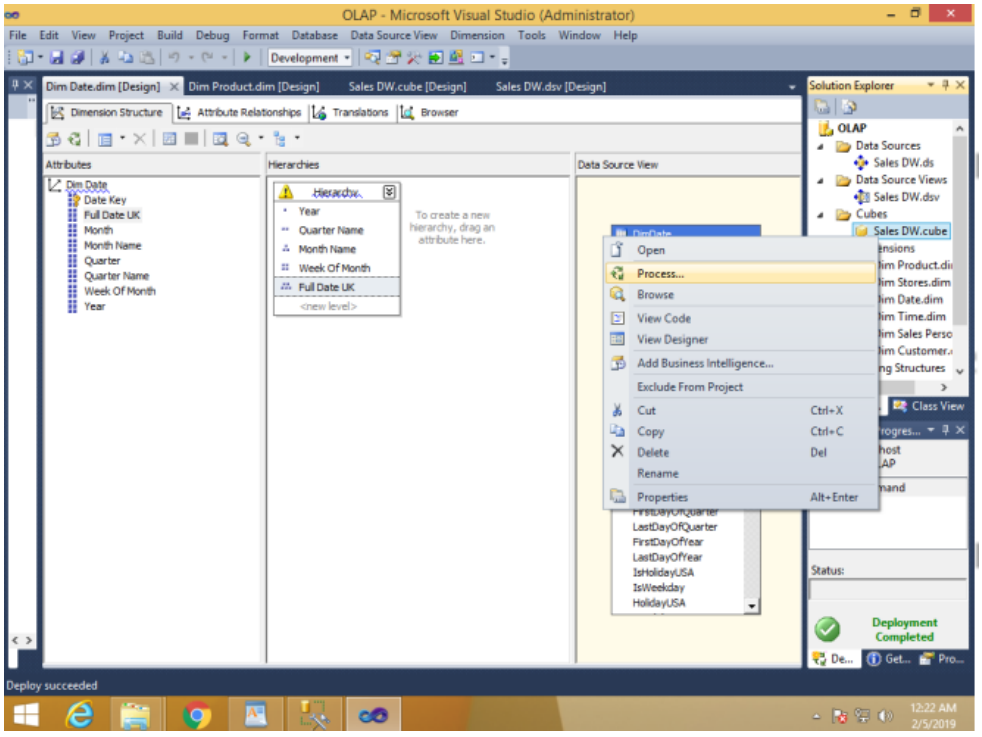
Right click on project name → Deploy



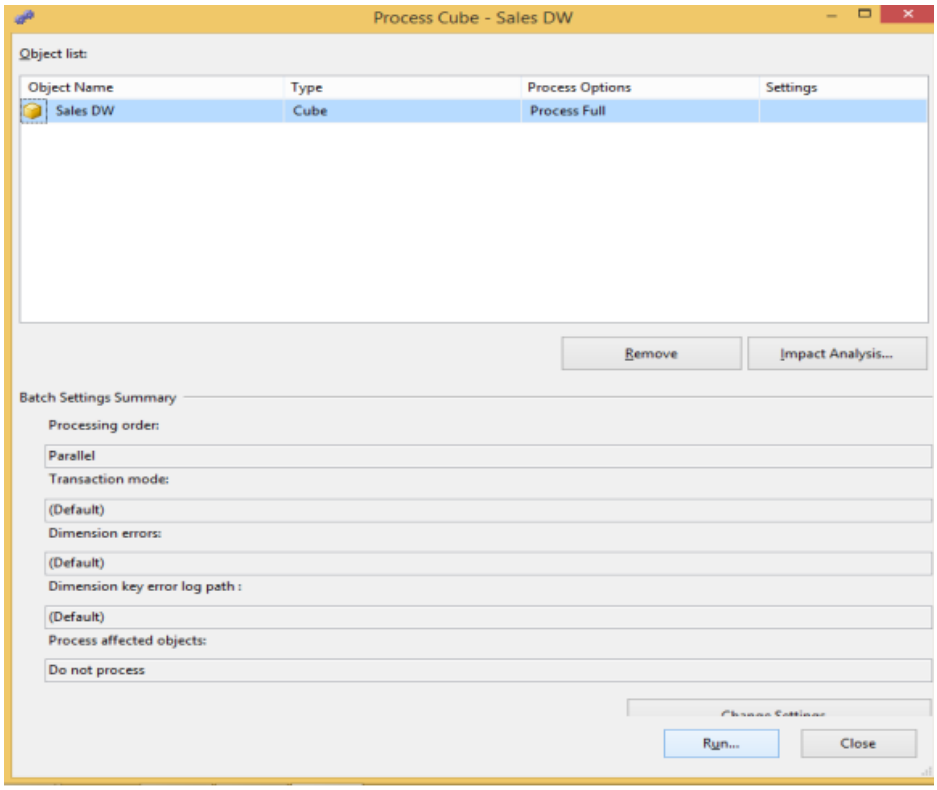
Deployment successful

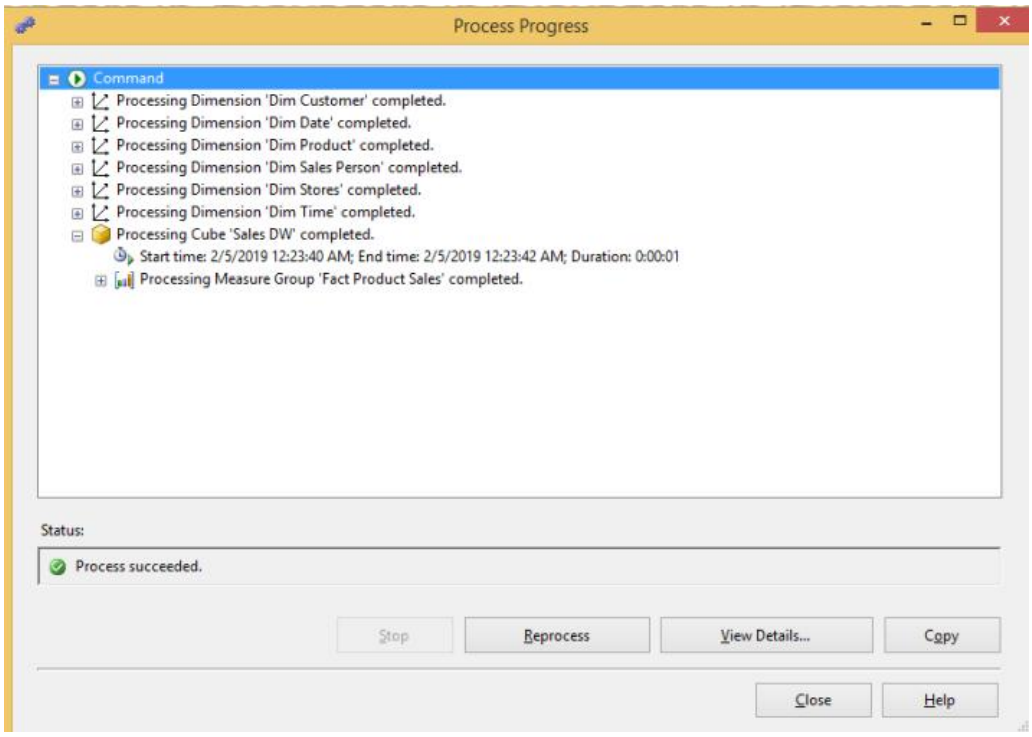


To process cube right click on Sales_DW.cube → Process

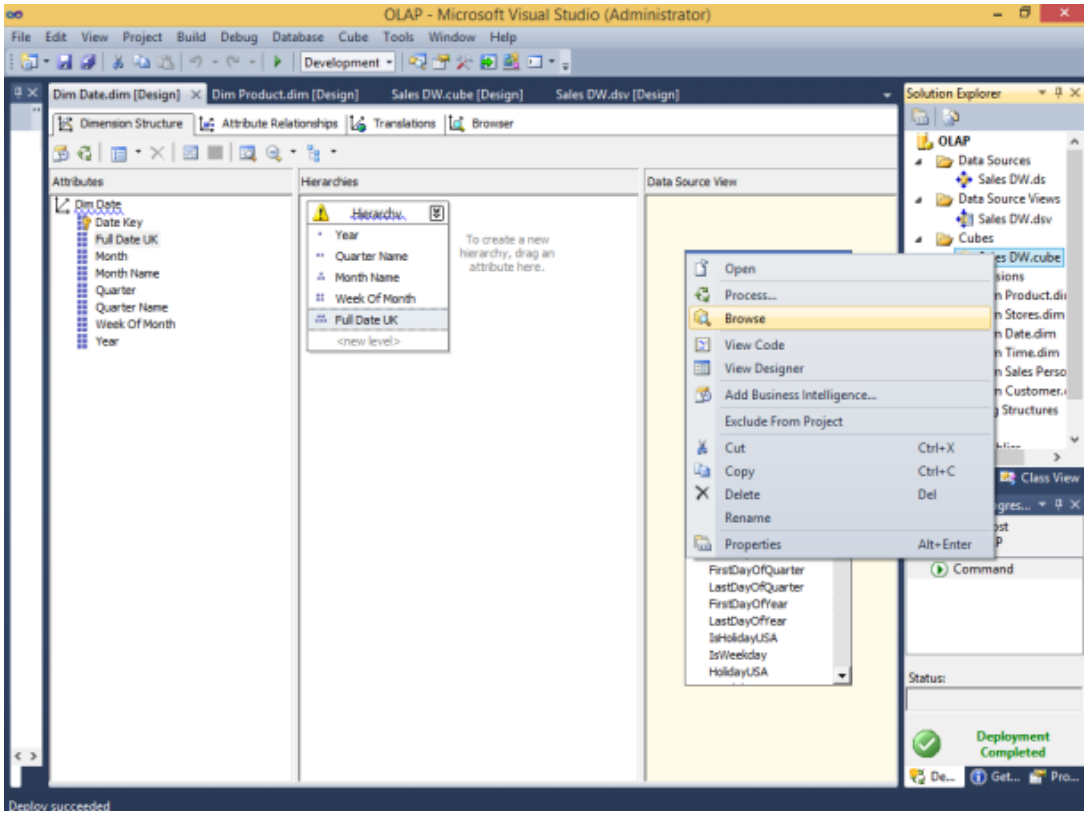


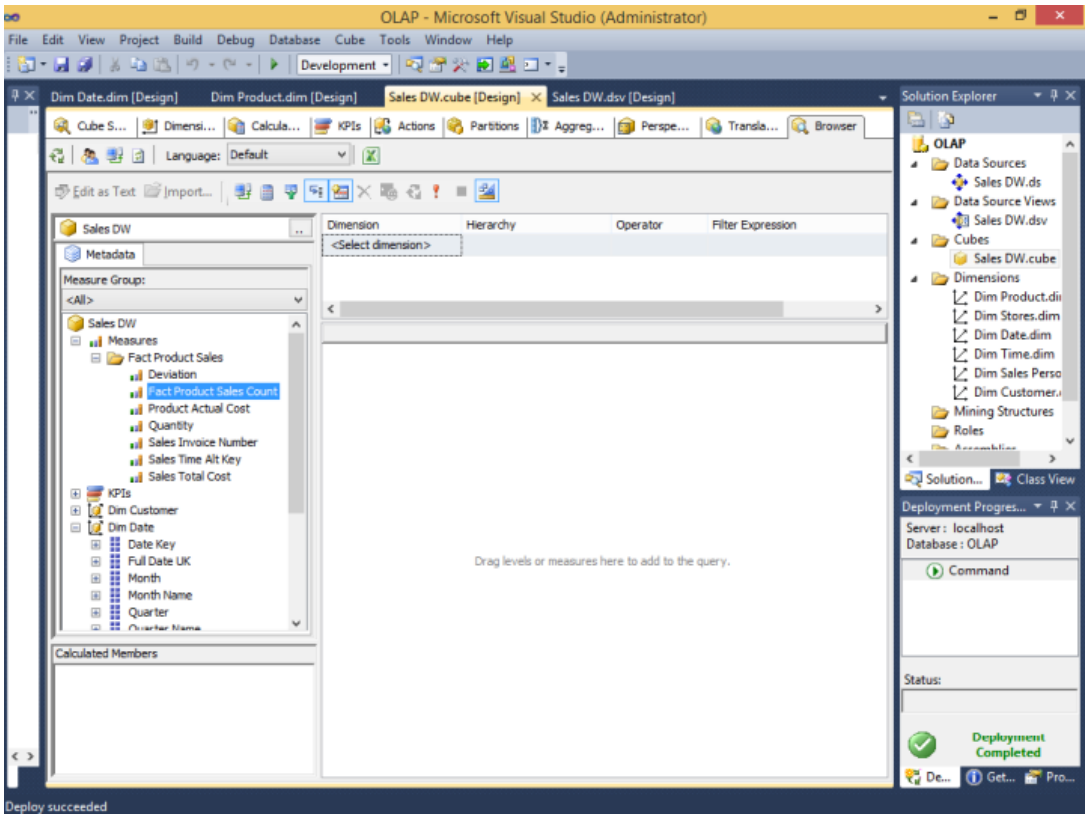
Click run





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]
Dimension Structure
Attribute Relationships
Translations
Browser
Attributes
Dim Date
Date Key
Full Date UK
Month
Month Name
Quarter
Quarter Name
Week Of Month
Year
Hierarchies
Hierarchy
Year
Quarter Name
Month Name
Week Of Month
Full Date UK
<new level>
To create a new hierarchy, drag an attribute here.
Data Source View
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


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.dim
Dim Stores.dim
Dim Date.dim
Dim Time.dim
Dim Sales Perso
Dim Customer.
Mining Structures
Roles
Class View
Deployment Progres...
Server : localhost
Database : OLAP
Command
Status:
Deployment Completed
Deploy succeeded
