INDEX

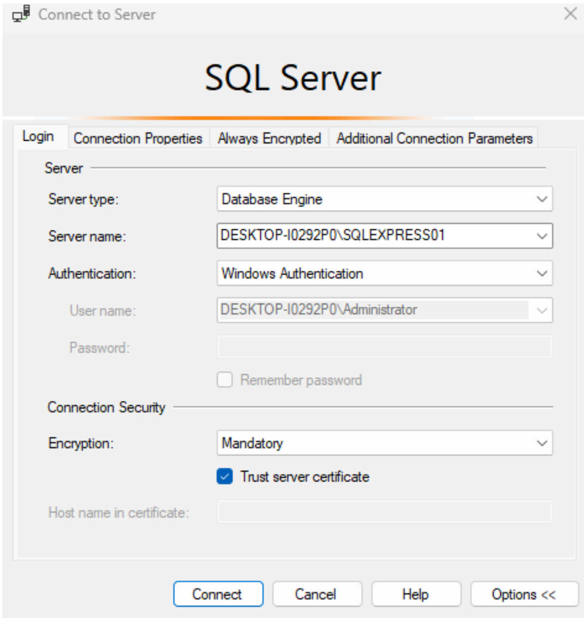
SR NO.	DATE	PRACTICALS	TR. SIGN
1.	17-12-24	Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.	
2.	07-01-25	Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data. Use Excel.	
3.	14-01-25	Perform the data classification using classification algorithm using R/Python.	
4.	21-01-25	Perform the data clustering using clustering algorithm using R/Python.	
5.	04-02-25	Perform the Linear regression on the given data warehouse data using R/Python.	
6.	11-02-25	Perform the logistic regression on the given data warehouse data using R/Python.	
7.	18-02-25	Write a Python program to read data from a CSV file, perform simple data analysis, and generate basic insights. (Use Pandas is a Python library).	
8.	25-02-25	Perform data visualization using Python on any sales data.	

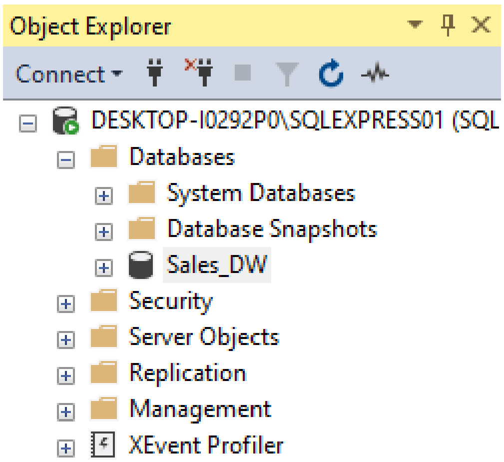
Practical No. 01

<u>Aim:</u> Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

Step 1: SQL Server Management Studio

1. Open SQL Server Management Studio
2. Copy the Server Name and paste it in blank excel
3. Click on Connect



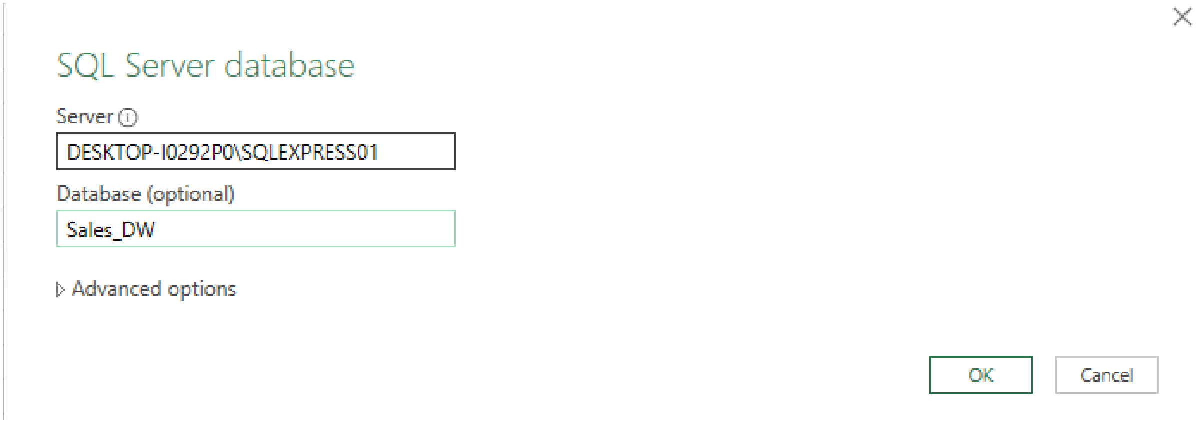


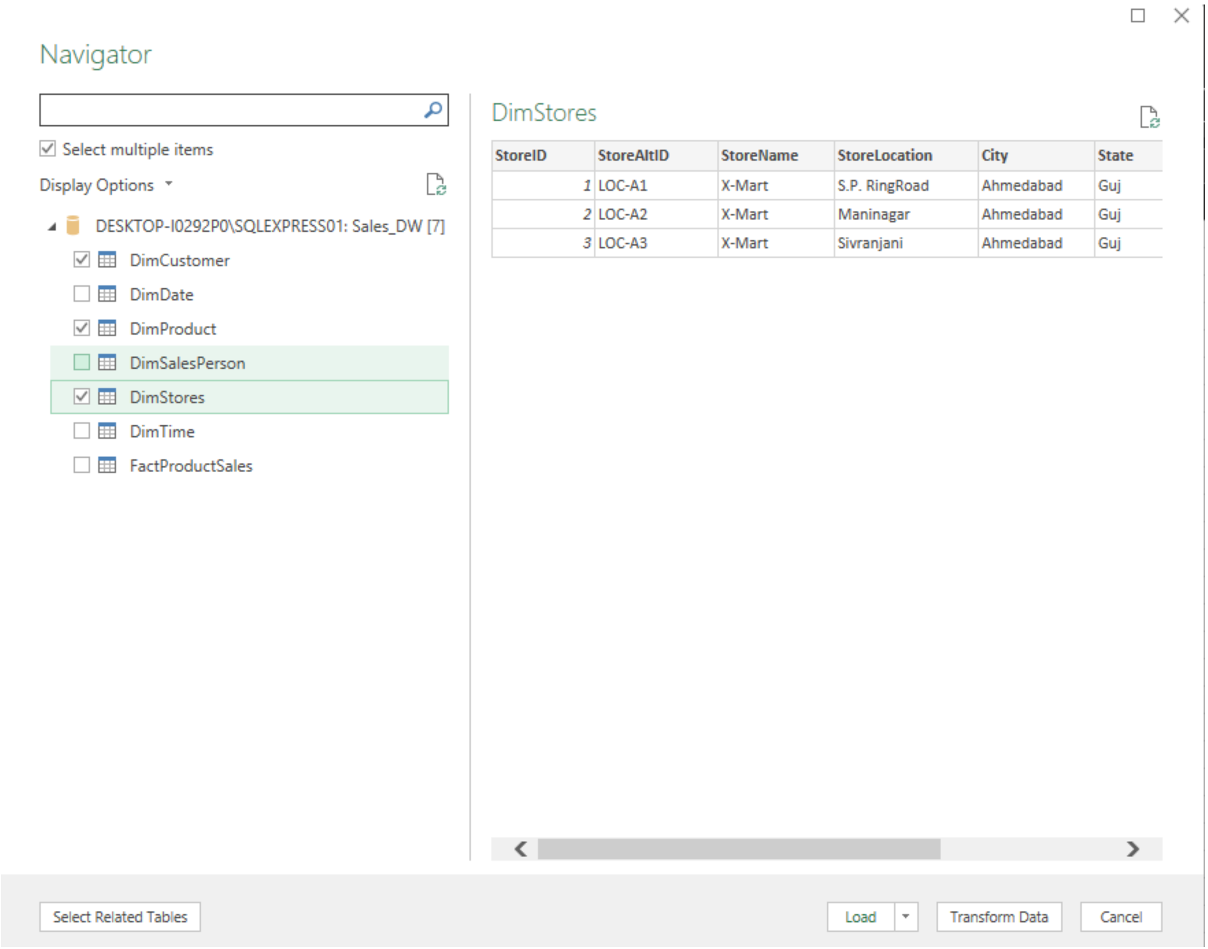
Step 2: Import Data into Excel:

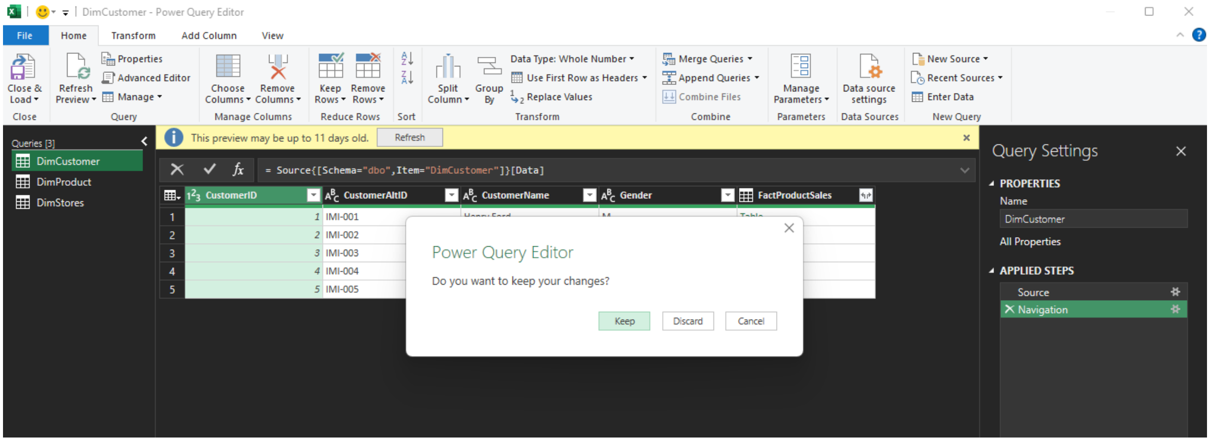
1. Open Microsoft Excel.
2. Go to the **Data** tab.

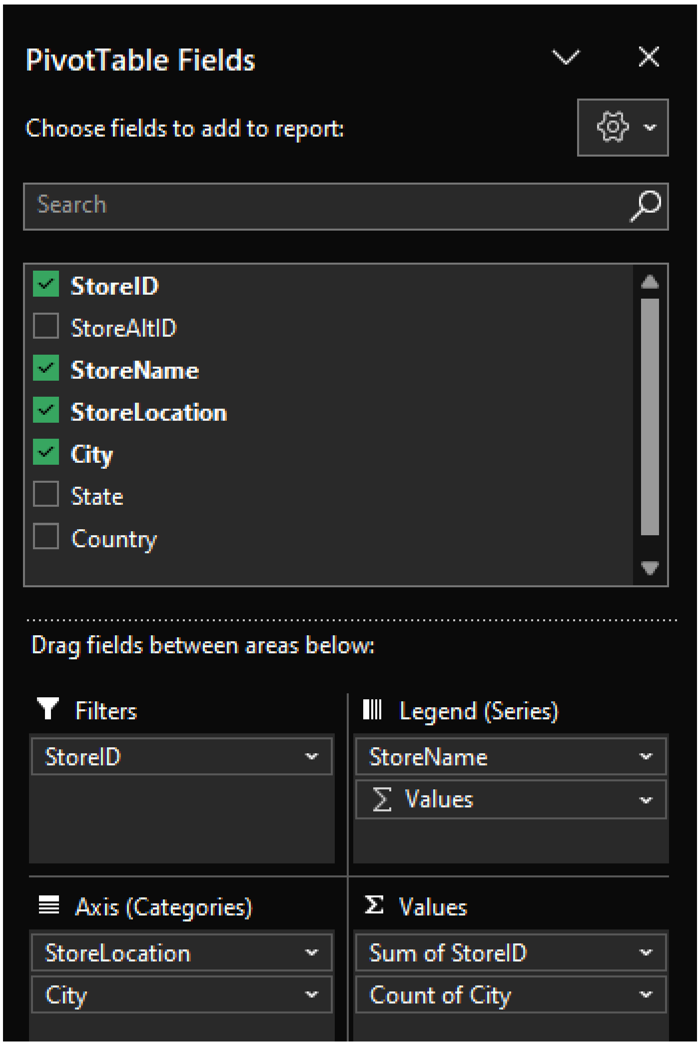
3. Click on **Get Data** (or **Get External Data** depending on your Excel version).
4. Choose the appropriate option based on where your data warehouse is hosted. For example:
    - **From Database**: If your data warehouse is a SQL Server, Oracle, or other databases.
    - **From Azure**: If your data is in Azure.
    - **From Other Sources**: For other types of data warehouses.
5. Follow the prompts to connect to your data warehouse and import the data into Excel.

- **Create a PivotTable:**
    1. Once your data is imported, select the data range.
    2. Go to the **Insert** tab.
    3. Click on **PivotTable**.
    4. In the **Create PivotTable** dialog box, make sure the correct table/range is selected.
    5. Choose whether you want the PivotTable to be placed in a new worksheet or an existing worksheet.
    6. Click **OK**.
- **Create a PivotChart:**
    1. With the PivotTable selected, go to the **PivotTable Analyze** or **Options** tab (depending on your version of Excel).
    2. Click on **PivotChart**.
    3. Select the type of chart you want to create (e.g., Column, Line, Pie, etc.).
    4. Click **OK**.
- **Customize Your PivotTable and PivotChart:**
    1. Drag and drop fields in the PivotTable Field List to arrange your data as needed.
    2. Customize the PivotChart by using chart tools and design options to fit your preferences.

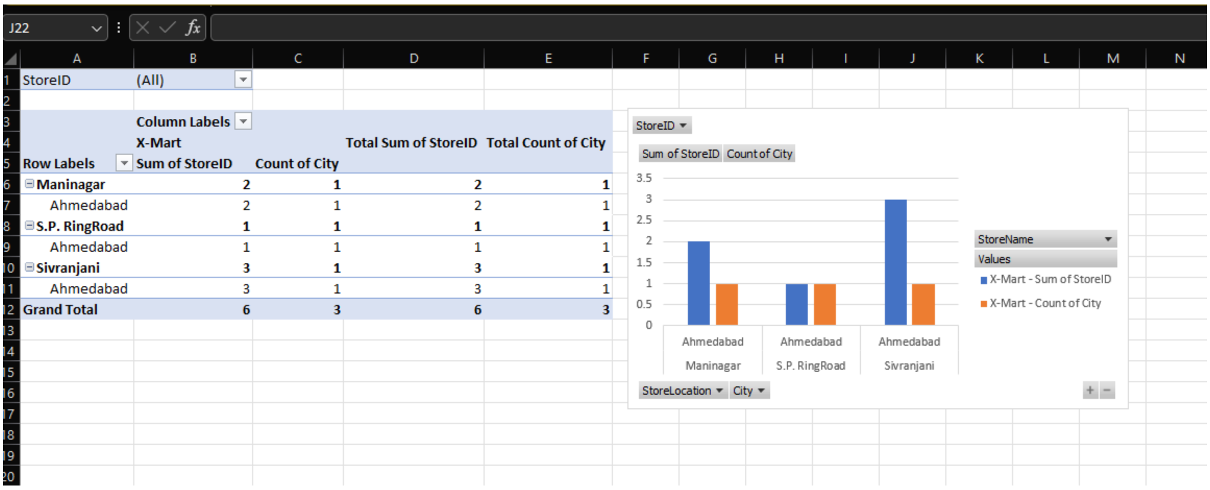

SQL Server database
Server
DESKTOP-I0292P0\SQLEXPRESS01
Database (optional)
Sales_DW
Advanced options
OK
Cancel


Navigator
Select multiple items
Display Options
DESKTOP-I0292P0\SQLEXPRESS01: Sales_DW [7]
DimCustomer
DimDate
DimProduct
DimSalesPerson
DimStores
DimTime
FactProductSales
DimStores
StoreID
StoreAltID
StoreName
StoreLocation
City
State
1 LOC-A1 X-Mart S.P. RingRoad Ahmedabad Guj
2 LOC-A2 X-Mart Maninagar Ahmedabad Guj
3 LOC-A3 X-Mart Sivranjani Ahmedabad Guj
Select Related Tables
Load
Transform Data
Cancel


DimCustomer - Power Query Editor
File
Home
Transform
Add Column
View
Close & Load
Refresh Preview
Properties
Advanced Editor
Manage
Choose Columns
Remove Columns
Keep Rows
Remove Rows
Split Column
Group By
Data Type: Whole Number
Use First Row as Headers
Replace Values
Merge Queries
Append Queries
Combine Files
Manage Parameters
Data source settings
New Source
Recent Sources
Enter Data
Close
Query
Manage Columns
Reduce Rows
Sort
Transform
Combine
Parameters
Data Sources
New Query
Queries [3]
DimCustomer
DimProduct
DimStores
This preview may be up to 11 days old.
Refresh
= Source{[Schema="dbo",Item="DimCustomer"]}[Data]
CustomerID
CustomerAltID
CustomerName
Gender
FactProductSales
IMI-001
IMI-002
IMI-003
IMI-004
IMI-005
Power Query Editor
Do you want to keep your changes?
Keep
Discard
Cancel
Query Settings
PROPERTIES
Name
DimCustomer
All Properties
APPLIED STEPS
Source
Navigation


PivotTable Fields
Choose fields to add to report:
Search
StoreID
StoreAltID
StoreName
StoreLocation
City
State
Country
Drag fields between areas below:
Filters
StoreID
Legend (Series)
StoreName
Values
Axis (Categories)
StoreLocation
City
Values
Sum of StoreID
Count of City

OUTPUT:



StoreID	(All)			
	Column Labels			
	X-Mart		**Total Sum of StoreID**	**Total Count of City**
Row Labels	**Sum of StoreID**	**Count of City**		
Maninagar	**2**	**1**	**2**	**1**
Ahmedabad	2	1	2	1
S.P. RingRoad	**1**	**1**	**1**	**1**
Ahmedabad	1	1	1	1
Sivranjani	**3**	**1**	**3**	**1**
Ahmedabad	3	1	3	1
Grand Total	**6**	**3**	**6**	**3**

Practical No. 02

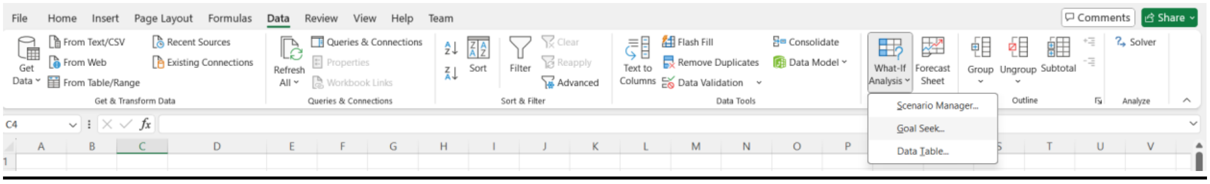
<u>Aim:</u> Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data. Use Excel.

<u>(A) Goal Seek:</u>

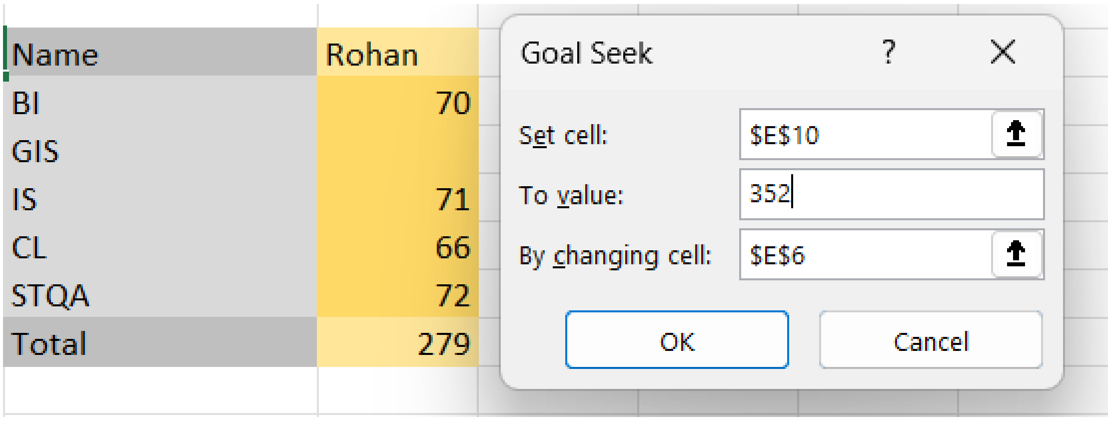
Step 1: In Excel create the table

Name	Rohan
BI	70
GIS	
IS	71
CL	66
STQA	72
Total	279

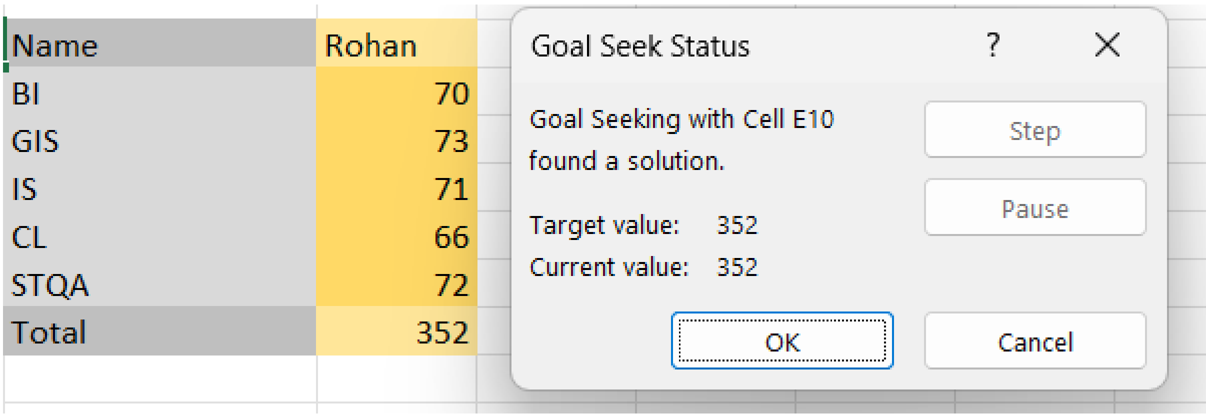
Step 2: In Data, goto What-if Analysis and Select Goal Seek



Step 3:



OUTPUT:

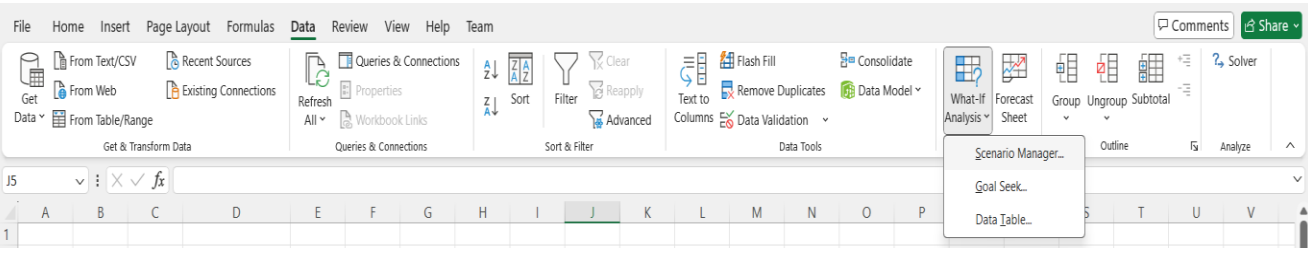


(B) Scenario Manager:

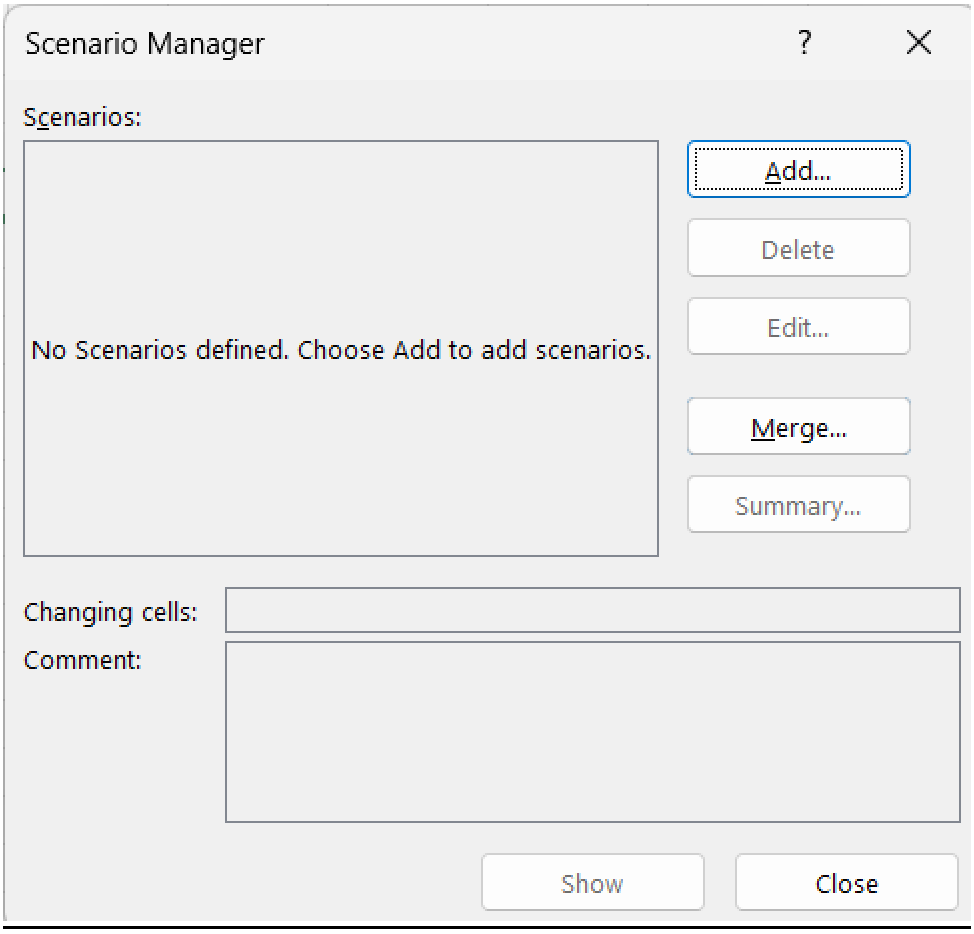
Step 1: In Excel create the table

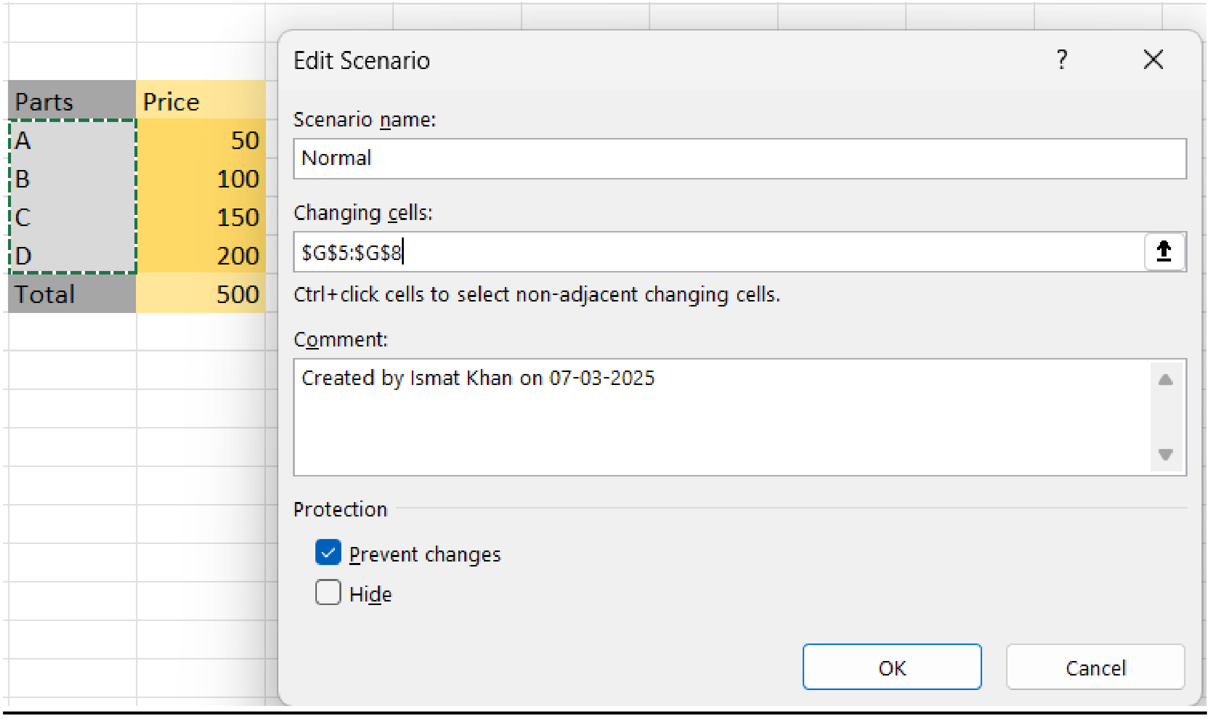
Parts	Price
A	50
B	100
C	150
D	200
Total	500

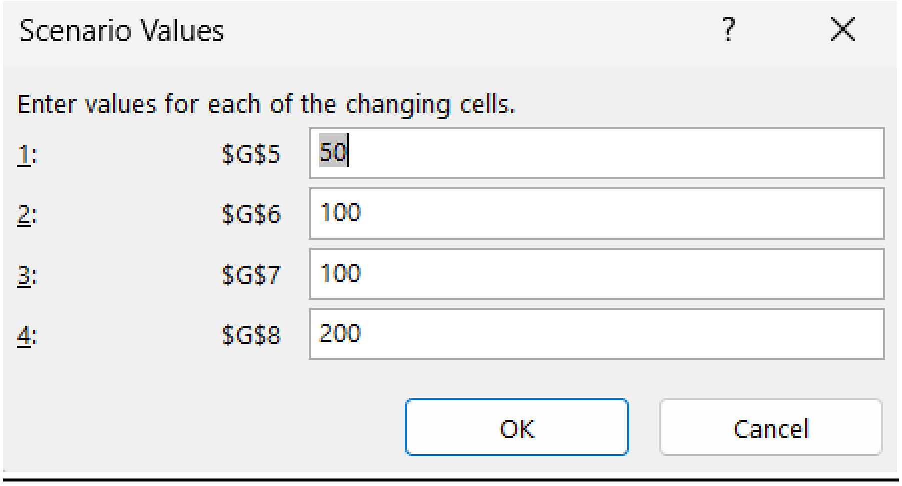
Step 2: In Data, goto What-if Analysis and Select Scenario Manager



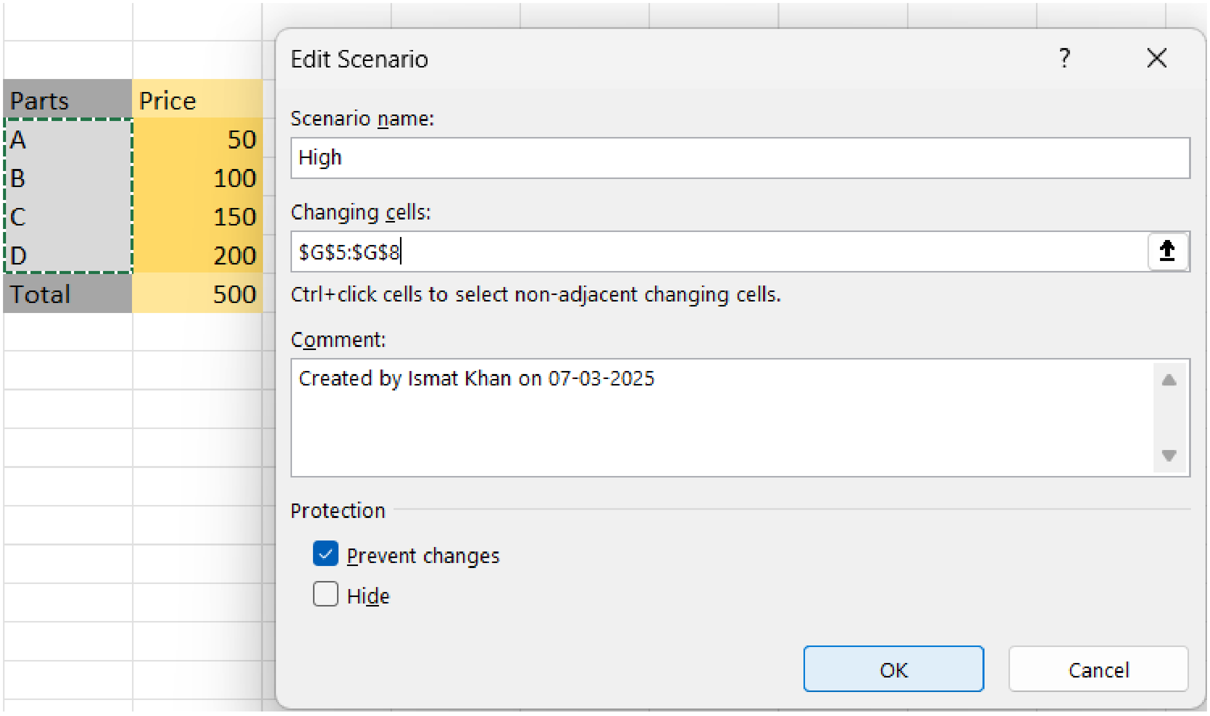
Step 3: Add Scenario (Normal)

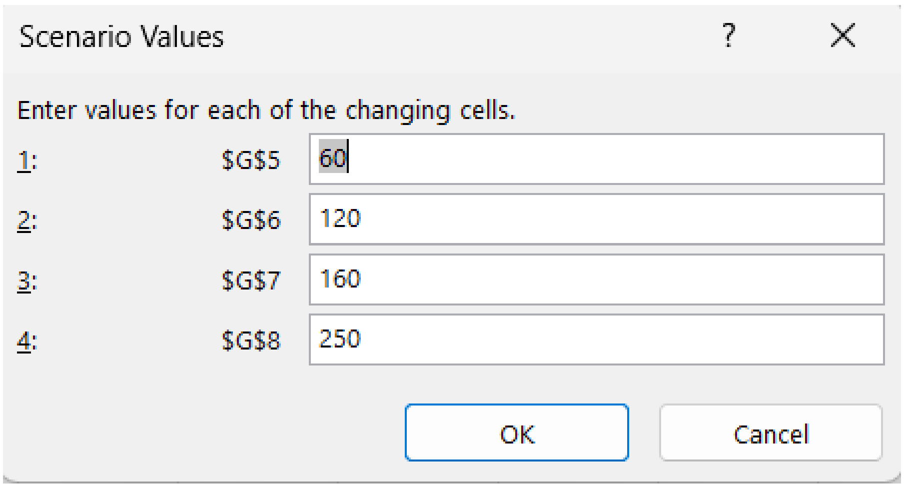




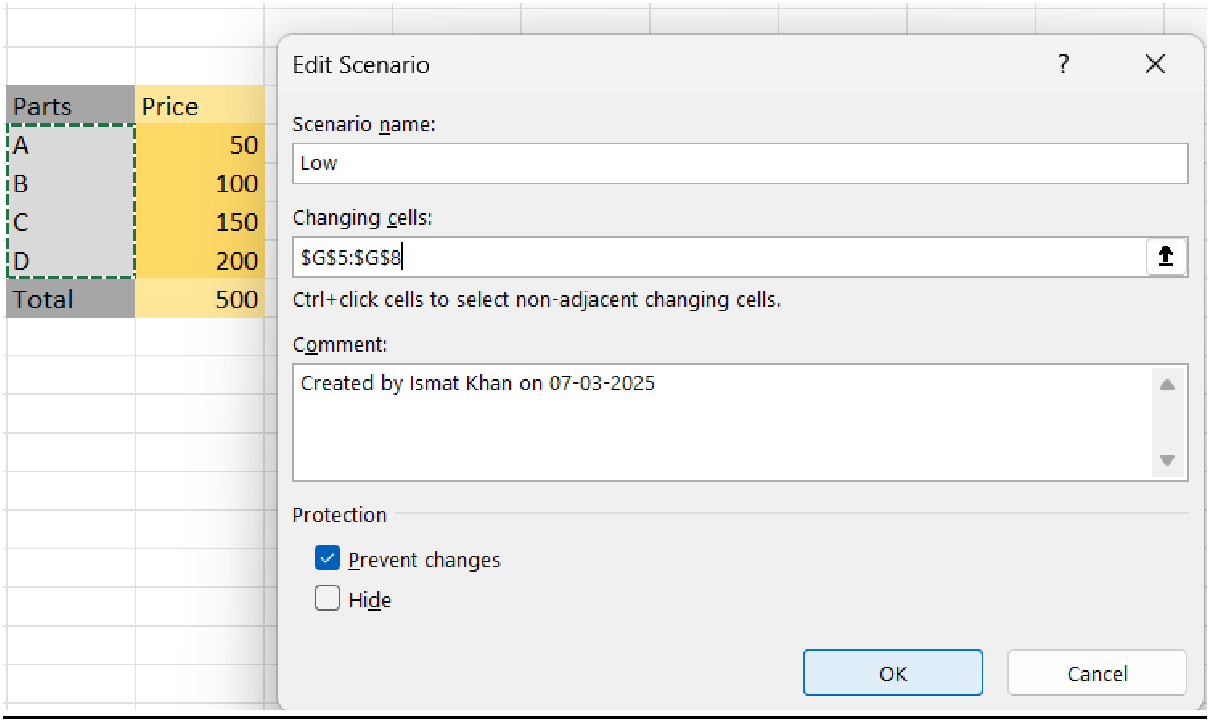


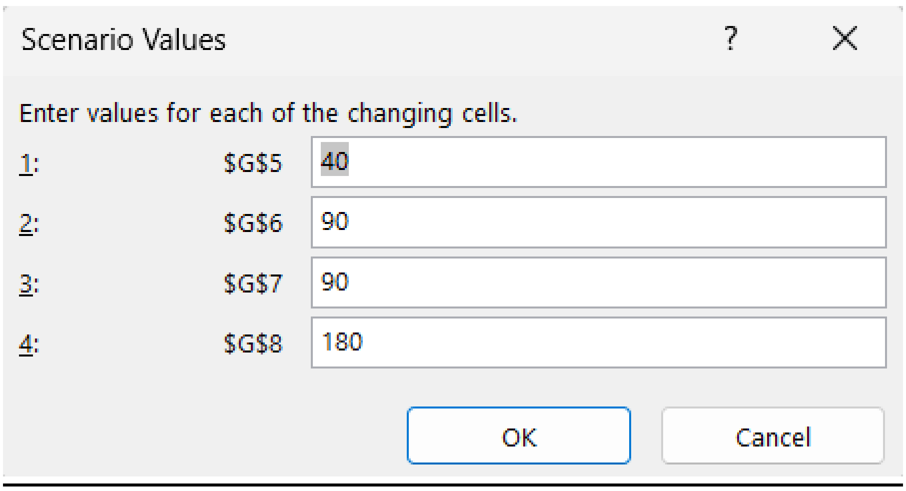
Step 4: Add Scenario (High)



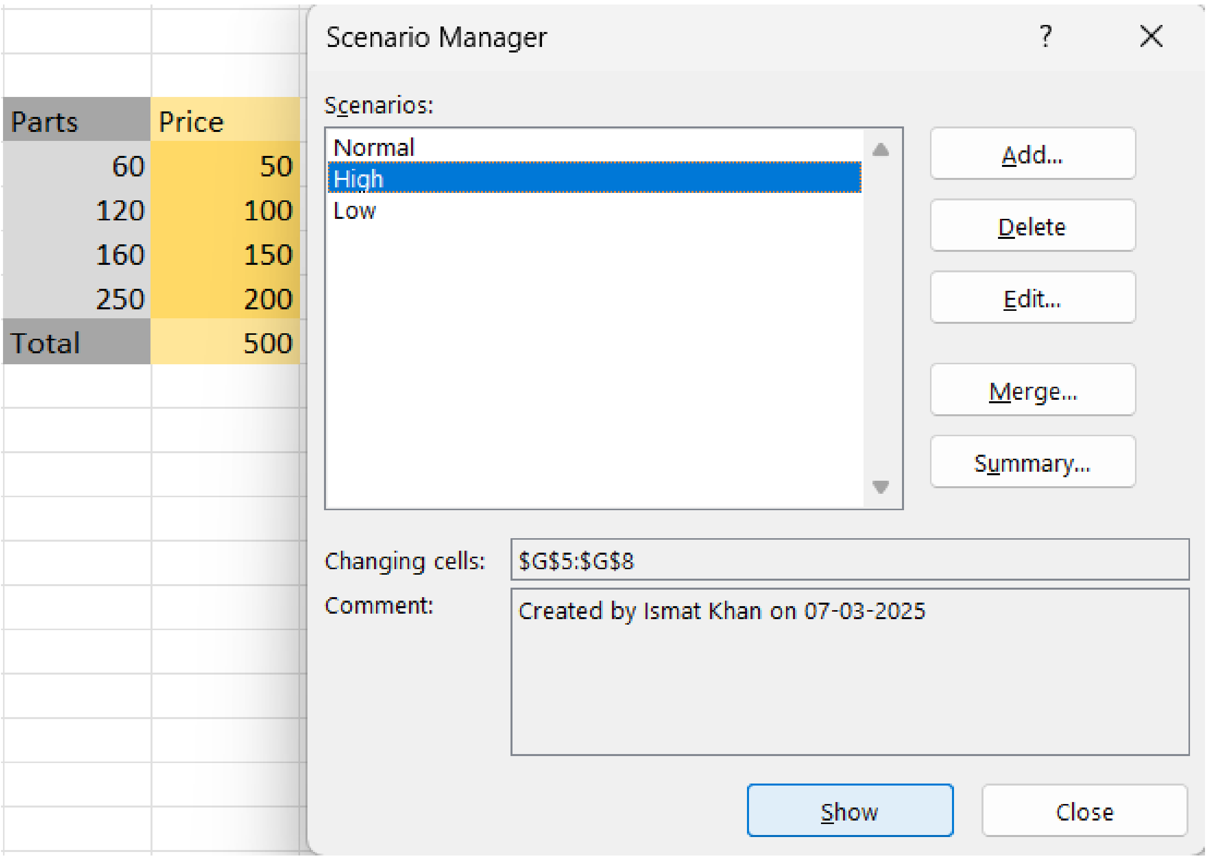


Step 5: Add Scenario (Low)

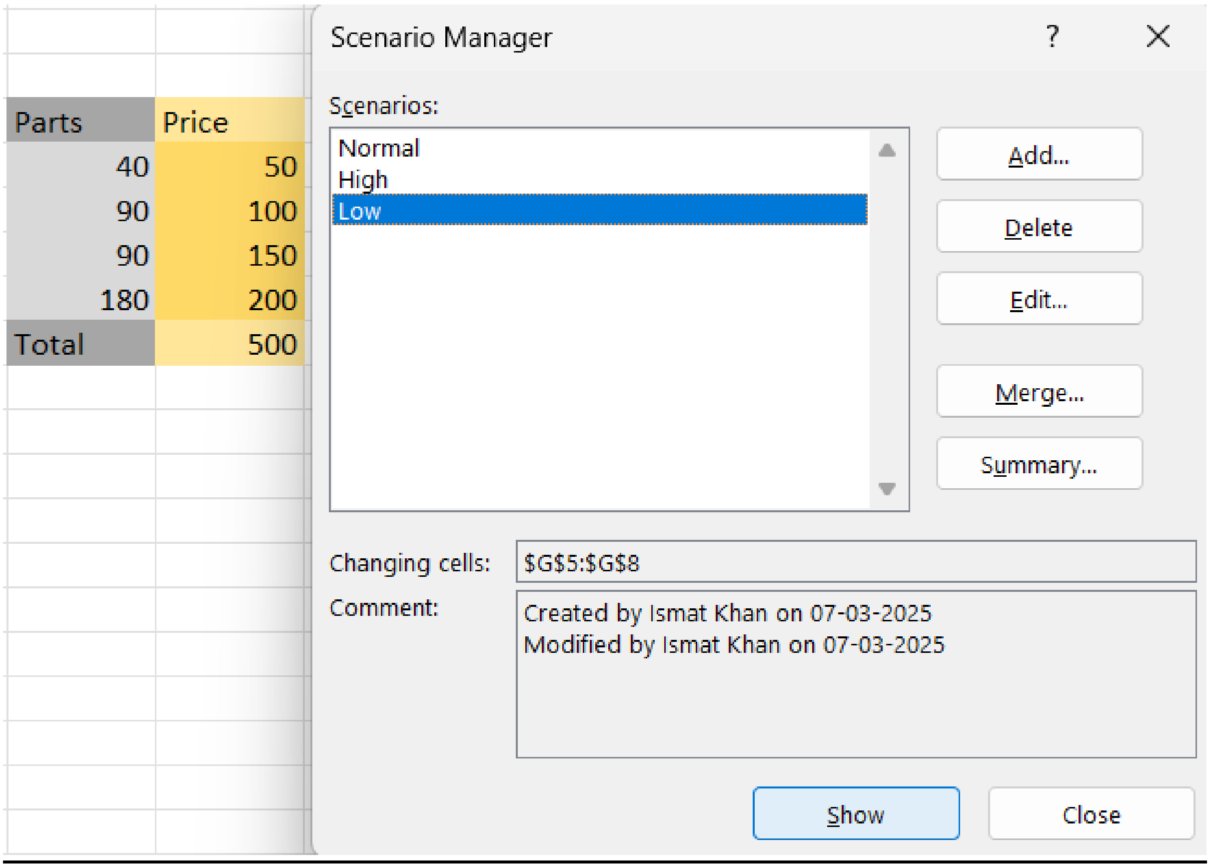




OUTPUT:



Parts
Price
60
50
120
100
160
150
250
200
Total
500
Scenario Manager
?
Scenarios:
Normal
High
Low
Add...
Delete
Edit...
Merge...
Summary...
Changing cells:
$G$5:$G$8
Comment:
Created by Ismat Khan on 07-03-2025
Show
Close


Parts
Price
40
50
90
100
90
150
180
200
Total
500
Scenario Manager
?
Scenarios:
Normal
High
Low
Add...
Delete
Edit...
Merge...
Summary...
Changing cells:
$G$5:$G$8
Comment:
Created by Ismat Khan on 07-03-2025
Modified by Ismat Khan on 07-03-2025
Show
Close

Practical No. 03

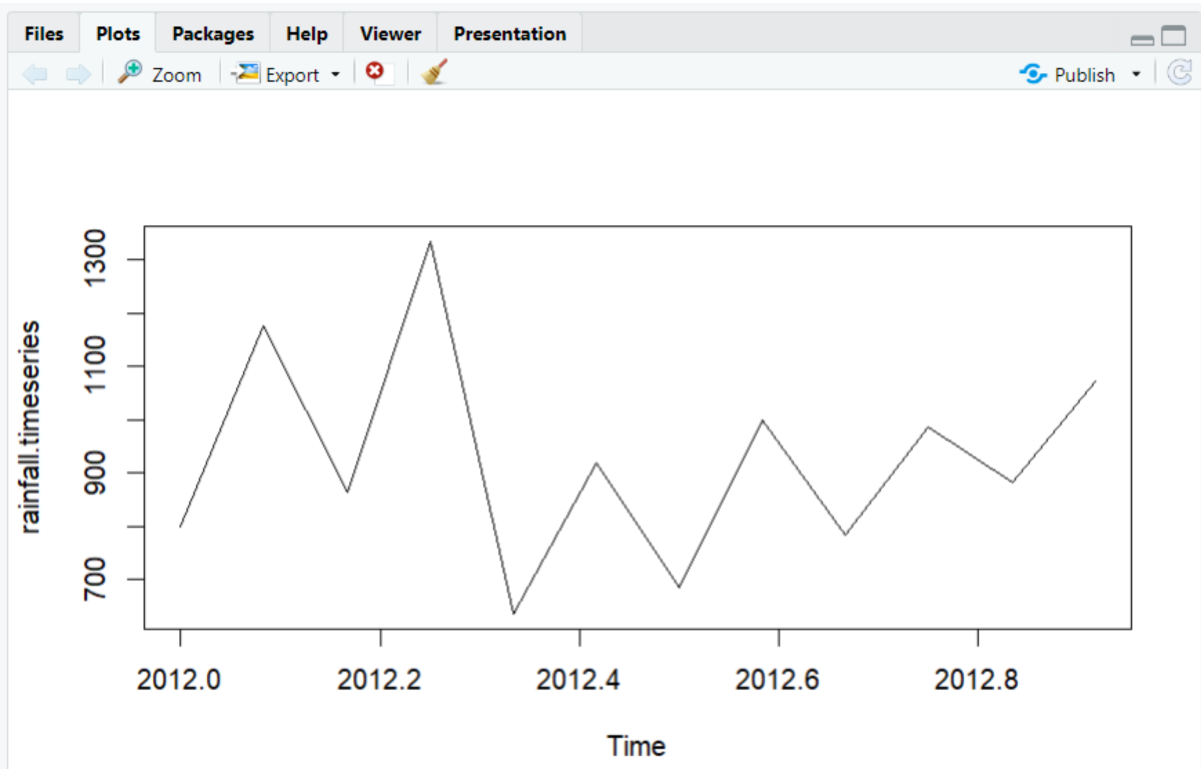
Aim: Perform the data classification using classification algorithm using R.

Perform on R Editor

```
rainfall <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Step 2: Create a time series object starting in January 2012 with a monthly
frequency

rainfall.timeseries <- ts(rainfall, start = c(2012, 1), frequency = 12)

# Step 3: Print the time series object

print(rainfall.timeseries)

# Step 4: Save the plot to a PNG file

png(file = "rainfall.png")

# Step 5: Plot the time series data

plot(rainfall.timeseries)

# Step 6: Close the PNG device

dev.off()

print(rainfall.timeseries)

plot(rainfall.timeseries)
```

OUTPUT:

```
> rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
> # Step 2: Create a time series object starting in January 2012 with a monthly frequency
> rainfall.timeseries <- ts(rainfall, start = c(2012, 1), frequency = 12)
> # Step 3: Print the time series object
> print(rainfall.timeseries)
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
2012  799.0 1174.8  865.1 1334.6  635.4  918.5  685.5  998.6  784.2  985.0  882.8 1071.0
> # Step 4: Save the plot to a PNG file
> png(file = "rainfall.png")
> # Step 5: Plot the time series data
> plot(rainfall.timeseries)
> # Step 6: Close the PNG device
> dev.off()
null device
          1
> print(rainfall.timeseries)
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
2012  799.0 1174.8  865.1 1334.6  635.4  918.5  685.5  998.6  784.2  985.0  882.8 1071.0
> plot(rainfall.timeseries)
> 
```



Practical No. 04

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

Perform on R Console:

iris

newiris<- iris

newiris$Species <- NULL

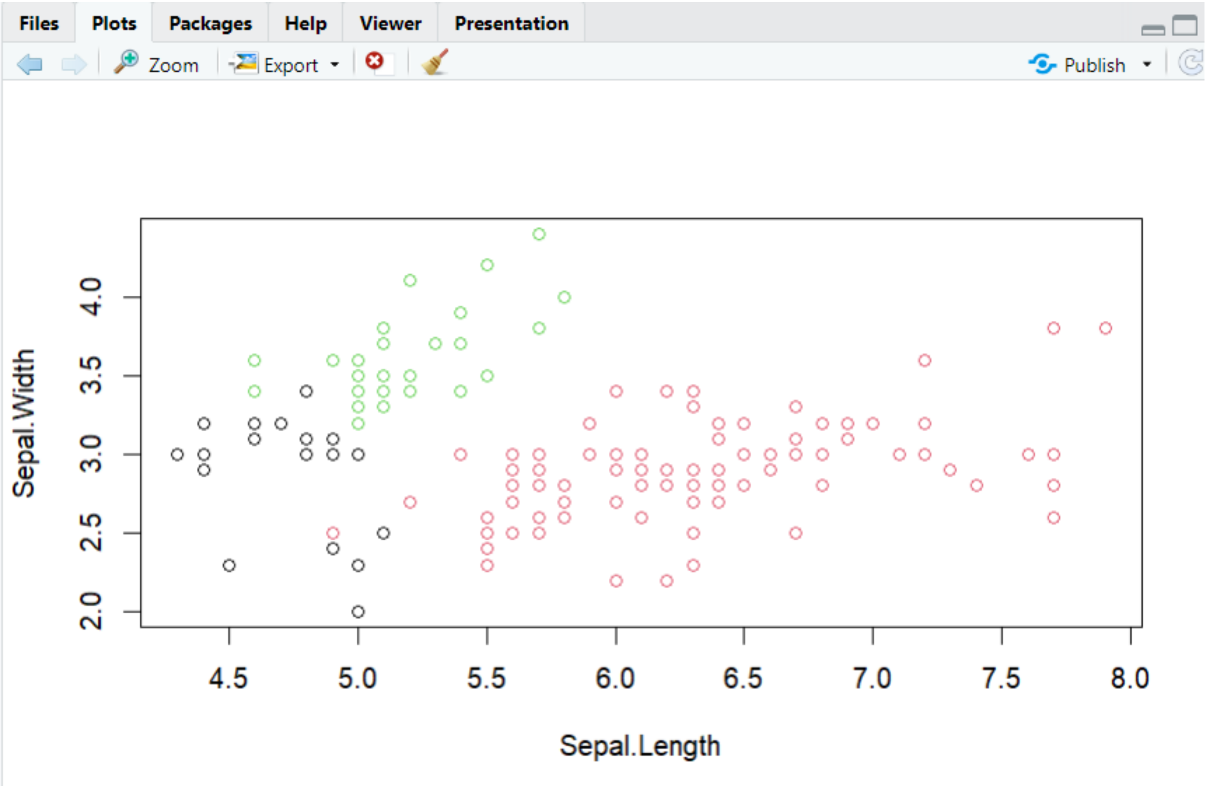
(kc <- kmeans(newiris,3))

table (iris$Species, kc$cluster)

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

OUTPUT:

```
144          6.8         3.2          5.9          2.3  virginica
145          6.7         3.3          5.7          2.5  virginica
146          6.7         3.0          5.2          2.3  virginica
147          6.3         2.5          5.0          1.9  virginica
148          6.5         3.0          5.2          2.0  virginica
149          6.2         3.4          5.4          2.3  virginica
150          5.9         3.0          5.1          1.8  virginica
> 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 3 1 1 3 3 3 1 1 3 3 3 1 3 3 3 1 3 3 1 1 3 3
 [46] 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 2 2 2 2 2 2 2 2 2 2 2 2 2
 [91] 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[136] 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" "betweenss"
[7] "size"         "iter"         "ifault"
> table (iris$Species, kc$cluster)

              1  2  3
  setosa     17  0 33
  versicolor  4 46  0
  virginica   0 50  0
> plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)
> 
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Sepal.Width
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal.Length

Practical No. 05

<u>Aim:</u> Perform the Linear regression on the given data warehouse data using R.

mtcars

input <- mtcars[,c("am","cyl","hp","wt")]

print(head(input))

input <- mtcars[,c("am","cyl","hp","wt")]

am.data=glm(formula = am ~ cyl + hp + wt, data = input, family= binomial)

print(summary(am.data))

<u>OUTPUT:</u>

```
Console   Terminal ×   Background Jobs ×
R 4.4.3 · ~/
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
> input <- mtcars[,c("am","cyl","hp","wt")]
> print(head(input))
                  am cyl  hp    wt
Mazda RX4          1   6 110 2.620
Mazda RX4 Wag      1   6 110 2.875
Datsun 710         1   4  93 2.320
Hornet 4 Drive     0   6 110 3.215
Hornet Sportabout  0   8 175 3.440
Valiant            0   6 105 3.460
> input <- mtcars[,c("am","cyl","hp","wt")]
> am.data=glm(formula = am ~ cyl + hp + wt, data = input, family= binomial)
> print(summary(am.data))

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```

Practical No. 06

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

```
install.packages("tidyverse")
install.packages("caret")
library(tidyverse)
library(caret)
data <- data.frame(
  Date = as.Date(c('2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04', '2025-01-05')),
  Product = c('A', 'B', 'A', 'B', 'A'),
  Sales = c(100, 150, 200, 250, 300),
  Revenue = c(1000, 1500, 2000, 2500, 3000),
  Purchased = c(1, 0, 1, 0, 1) )
print(data)
model <- glm(Purchased ~ Product + Sales + Revenue, data = data, family = binomial)
summary(model)
data$predicted_prob <- predict(model, type = "response")
data$predicted_class <- ifelse(data$predicted_prob > 0.5, 1, 0)
print(data)
confusionMatrix(factor(data$predicted_class), factor(data$Purchased))
```

OUTPUT:

```
Console   Terminal ×   Background Jobs ×
R 4.4.3 · ~/
> library(tidyverse)
> library(caret)
Loading required package: lattice

Attaching package: ‘caret’

The following object is masked from ‘package:purrr’:

    lift

> data <- data.frame(
+   Date = as.Date(c('2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04', '2025-01-05')),
+   Product = c('A', 'B', 'A', 'B', 'A'),
+   Sales = c(100, 150, 200, 250, 300),
+   Revenue = c(1000, 1500, 2000, 2500, 3000),
+   Purchased = c(1, 0, 1, 0, 1)
+ )
> print(data)
        Date Product Sales Revenue Purchased
1 2025-01-01       A   100    1000         1
2 2025-01-02       B   150    1500         0
3 2025-01-03       A   200    2000         1
4 2025-01-04       B   250    2500         0
5 2025-01-05       A   300    3000         1
> model <- glm(Purchased ~ Product + Sales + Revenue, data = data, family = binomial)
> summary(model)

Call:
glm(formula = Purchased ~ Product + Sales + Revenue, family = binomial,
    data = data)

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.457e+01  1.822e+05       0        1
ProductB    -4.913e+01  1.196e+05       0        1
Sales        2.783e-12  8.286e+02       0        1
Revenue             NA         NA      NA       NA
```

Console Terminal Background Jobs

R 4.4.3 · ~/

```
Number of Fisher Scoring iterations: 23

> data$predicted_prob <- predict(model, type = "response")
> data$predicted_class <- ifelse(data$predicted_prob > 0.5, 1, 0)
> print(data)
        Date Product Sales Revenue Purchased predicted_prob predicted_class
1 2025-01-01       A   100    1000         1   1.000000e+00               1
2 2025-01-02       B   150    1500         0   2.143345e-11               0
3 2025-01-03       A   200    2000         1   1.000000e+00               1
4 2025-01-04       B   250    2500         0   2.143345e-11               0
5 2025-01-05       A   300    3000         1   1.000000e+00               1
> confusionMatrix(factor(data$predicted_class), factor(data$Purchased))
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 2 0
         1 0 3

               Accuracy : 1
                 95% CI : (0.4782, 1)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : 0.07776

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.4
         Detection Rate : 0.4
   Detection Prevalence : 0.4
      Balanced Accuracy : 1.0

       'Positive' Class : 0

> 
```

Practical No. 07

Aim: Write a Python program that reads data from a CSV file, performs simple data analysis using Pandas, and generates basic insights:

```
import pandas as pd
# Step 1: Read data from a CSV file
df = pd.read_csv('sales_data.csv')
# Step 2: Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())
# Step 3: Display summary statistics
print("\nSummary statistics:")
print(df.describe())
# Step 4: Display information about the dataset
print("\nDataset information:")
print(df.info())
# Step 5: Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())
# Step 6: Calculate total sales and revenue
total_sales = df['Sales'].sum()
total_revenue = df['Revenue'].sum()
print(f"\nTotal Sales: {total_sales}")
print(f"Total Revenue: {total_revenue}")
# Step 7: Calculate average sales and revenue per product
avg_sales_per_product = df.groupby('Product')['Sales'].mean()
avg_revenue_per_product = df.groupby('Product')['Revenue'].mean()
```

```
print("\nAverage Sales per Product:")
print(avg_sales_per_product)
print("\nAverage Revenue per Product:")
print(avg_revenue_per_product)
# Step 8: Find the product with the highest total sales
highest_sales_product = df.groupby('Product')['Sales'].sum().idxmax()
print(f"\nProduct with the highest total sales: {highest_sales_product}")
# Step 9: Find the product with the highest total revenue
highest_revenue_product = df.groupby('Product')['Revenue'].sum().idxmax()
print(f"Product with the highest total revenue: {highest_revenue_product}")
# Step 10: Generate basic insights
print("\nBasic Insights:")
print(f"The dataset contains sales data for {df['Product'].nunique()} unique products.")
print(f"The average sales per day is {df['Sales'].mean():.2f}.")
print(f"The average revenue per day is {df['Revenue'].mean():.2f}.")
print(f"The product with the highest total sales is {highest_sales_product}.")
print(f"The product with the highest total revenue is {highest_revenue_product}.")
```

OUTPUT:

```
                                                --
First few rows of the dataset:
  Product  Sales  Revenue
0       A    100     1000
1       B    200     2500
2       C    150     1800

Summary statistics:
       Sales      Revenue
count    3.0     3.000000
mean   150.0  1766.666667
std     50.0   750.555350
min    100.0  1000.000000
25%    125.0  1400.000000
50%    150.0  1800.000000
75%    175.0  2150.000000
max    200.0  2500.000000

Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Product  3 non-null      object
 1   Sales    3 non-null      int64 
 2   Revenue  3 non-null      int64 
dtypes: int64(2), object(1)
memory usage: 200.0+ bytes
None

Missing values in each column:
Product    0
Sales      0
Revenue    0
dtype: int64

Total Sales: 450
Total Revenue: 5300

Average Sales per Product:
Product
```

```
A    100
B    200
C    150
Name: Sales, dtype: int64

Average Revenue per Product:
Product
A    1000
B    2500
C    1800
Name: Revenue, dtype: int64

Product with the highest total sales: B
Product with the highest total revenue: B

Basic Insights:
The dataset contains sales data for 3 unique products.
The average sales per day is 150.00.
The average revenue per day is 1766.67.
>>>
```

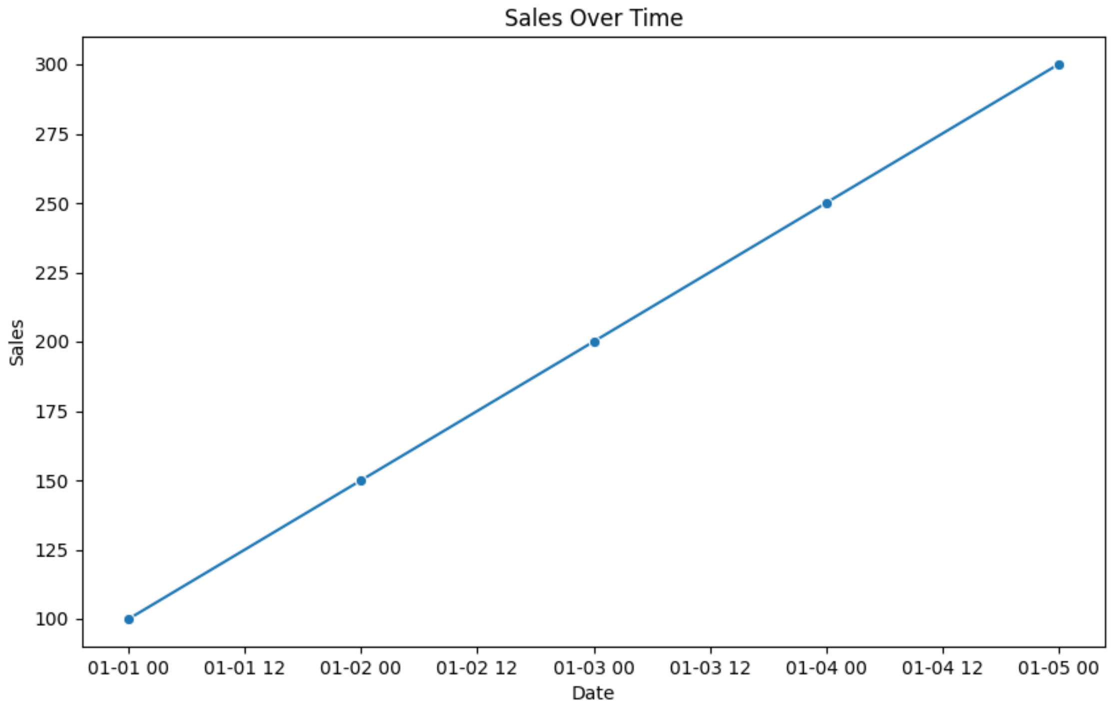
Practical No. 08

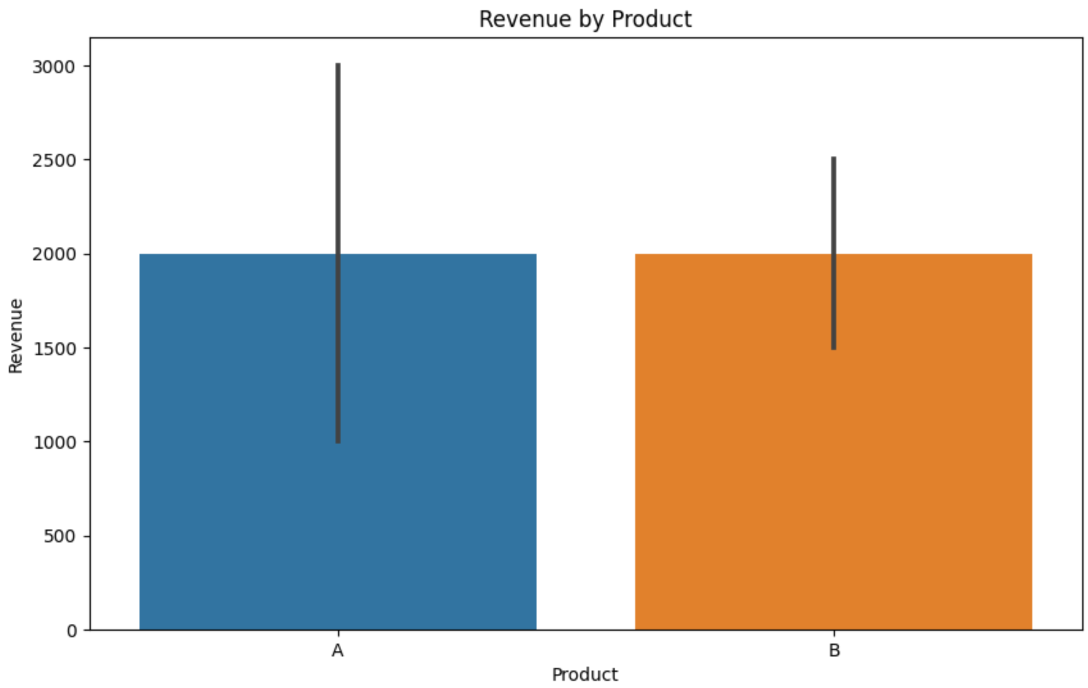
Aim: Perform data visualization using Python on any sales data

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Sample data
data = {
    'Date': ['2025-01-01', '2025-01-02', '2025-01-03', '2025-01-04', '2025-01-05'],
    'Product': ['A', 'B', 'A', 'B', 'A'],
    'Sales': [100, 150, 200, 250, 300],
    'Revenue': [1000, 1500, 2000, 2500, 3000]
}
# Creating DataFrame
df = pd.DataFrame(data)
# Convert 'Date' to datetime
df['Date'] = pd.to_datetime(df['Date'])
# Set the 'Date' column as the index
df.set_index('Date', inplace=True)
# Plotting Sales over Time
plt.figure(figsize=(10,6))
sns.lineplot(data=df, x='Date', y='Sales', marker='o')
plt.title('Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
# Plotting Revenue by Product
```

```
plt.figure(figsize=(10,6))
sns.barplot(data=df, x='Product', y='Revenue')
plt.title('Revenue by Product')
plt.xlabel('Product')
plt.ylabel('Revenue')
plt.show()
```

OUTPUT:




Revenue by Product
3000
2500
2000
1500
1000
500
0
Revenue
A
B
Product