Practical No: 01

Aim: Import the data warehouse data in Microsoft Excel and create the pivot table and Pivot chart

Step 1: Import Data from a Data warehouse into Excel

1. Open Excel
   - Launch Microsoft Excel
2. Go to Data Tab
   - Click on "Get Data" > "From Other Sources" > "From SQL Server Database" (or any other relevant source)
3. Enter Server Details
   - In the "SQL Server Database" window:
     - Enter the Server Name
     - Enter the Database Name (optional)
     - click OK
4. Select the Data to Import
   - Choose the tables or views that you need from the data warehouse
   - click load to import the data into Excel

Step 2: Create a Pivot Table

1. Select the Imported Data
   - click anywhere inside the imported data
2. Go to Insert Tab
   - click on "Pivot Table"
3. Choose Pivot Table Options
   - In the "CreatePivotTable" window:

- Ensure the correct table/range is selected
- Choose where to place the PivotTable (New or Existing worksheet)
- Click OK

4. Design the Pivot Table
- Drag and drop fields into the:
  - Rows area (e.g Categories, Regions)
  - Columns area (e.g TimePeriods)
  - Values area (e.g Sales, Revenue)
  - Filters area (optional)

Step 3: Create a Pivot Chart

1 Click on the Pivot Table
- Go to Insert Tab > PivotChart

2 Select chart Type
- Choose a chart type (e.g Column, line, Pie, Bar)
- Click OK

3 Customize Chart
- Add labels, titles and format the chart as needed

Step 4: Refresh data (if needed)

. If data is in warehouse updates, right-click on Pivot Table and select "Refresh" to get the latest data

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

Step 1: Connect to an OLAP Cube in Excel

1. Open Microsoft Excel
   - Launch Excel on your computer
2. Go to Data Tab
   - Click on "Get Data" (Power Query) > "From Database" > "From Analysis Services" (Microsoft's OLAP server)
3. Enter Connection Details
   - In the "Data Connection Wizard"
   - Enter the Server Name where the OLAP cube is hosted
   - Click Next
4. Select the OLAP Cube
   - Choose the appropriate database and cube from the list
   - Click Next and then Finish
5. Import Data into a PivotTable
   - Choose "PivotTable Report" when prompted
   - Click OK to place the PivotTable in a new worksheet

Step 2: Create a PivotTable for Analysis

1. Define Data Fields
   - In the PivotTable Fields Pane drag and drop fields into the respective areas
     - Rows (e.g.: Product, Category, Region)
     - Columns (e.g.: Year, Quarter)
     - Values (e.g.: Sales, Amount, Profit)

- Filters (Optional, e.g. Country, Time Period)

2. Summarize & Analyze Data
- Apply filters, sort, and group data as needed
- Use calculated fields to derive additional insights

Step3: Create a PivotChart for Visualization
1. Click on the PivotTable
- Go the Insert Tab > Click PivotChart
2. Select Chart Type
- Choose a suitable chart (e.g Column, line, Pie, Bar)
- Click OK
3. Customize the chart
- Add titles, labels and format colors
- Apply slicers for interactive filtering

Step4:- Refresh Data for Real-Time Analysis
- If the OLAP cube updates, right-click on the PivotTable and select "Refresh" to pull the latest data.

Practical no 2

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

Step 1: Import Data Warehouse Data into Excel

1. Open Excel
2. Go to Data Tab
   - Click "Get Data" > "From Other Sources" > "From SQL Server Database"
3. Enter Connection Details
   - Provide Server Name and Database Name then click OK
4. Load Data
   - Select required tables/views and click load

Step 2: Create PivotTables and PivotCharts

1. Insert a PivotTable
   - Click anywhere inside the data
   - Go to Insert Tab > Click PivotTable
   - Choose a worksheet and click OK
   - Drag and drop fields into Rows, Columns, Values and Filters
2. Create a PivotChart
   - Select the PivotTable
   - Go to Insert Tab > Click PivotChart
   - Choose an appropriate chart type (Bar, Line, Pie)
   - Format the chart for better visualization

Step 3: Apply What-If Analysis

1. Scenario Manager (Best, Worst and Expected Case Analysis)
   - Go to Data Tab > Click What-If Analysis > Select Scenario Manager

- Click Add and define different scenarios (e.g Sales Increase, Revenue Drop)
- Enter different values for key inputs like sales Growth, Costs, Profit Margins

2. Goal Seek (Find the Required Input for a Target Value)
- Go to Data Tab > Click What-If Analysis > Select Goal Seek
- Set a target value for Revenue or Profit and change sales Growth or price to achieve it
- Click OK to get the results

3. Data Tables (Analyze Multiple Inputs)
- Select a table range with different Price, Sales and Profit
- Go to Data Tab > Click What-If Analysis > Select Data Table
- Define Row Input Cell and Column Input Cell for changing values
- Click OK to see the impact

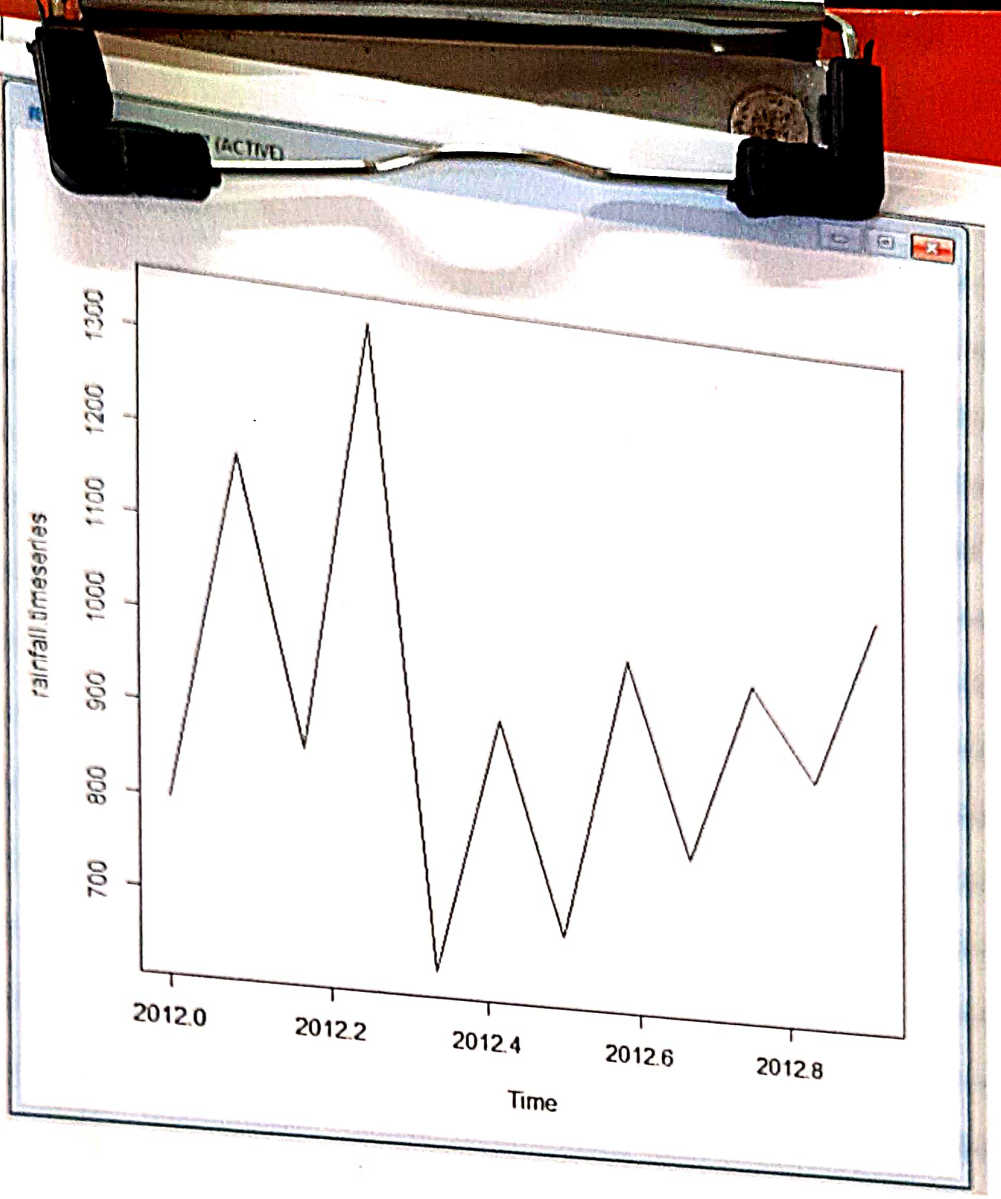
Step 4: Generate Reports Based on Analysis
- Summary Report
  - From Scenario Manager, Click Summary to generate a Comparison report
- Charts for Visualization
  - Use PivotCharts and Conditional Formatting to highlight Insights
- Dashboard Creation
  - Combine PivotTables, Charts and Slicers for an interactive dashboard.

Practical no 3.

Aim :- Perform the data classification using classification algorithm usig R/Python

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

```
#Get the data points in form of a R vector
rainfall <-
c(799, 1174.8, 865.1, 1334.6, 635.4, 918.5, 685.5,
    998.6, 784.2, 985, 882.8, 1071)
# Convert it to a time series object
rainfall.timeseries <- ts(rainfall, start = c(2012,1), frequency = 12)
# Print the timeseries data.
print(rainfall.timeseries)
#Give the chart file a name
png(file = "rainfall.png")
#Plot a graph of the time series
plot(rainfall.timeseries)
# Save the file
dev.off()
```



R R Console

```
> rainfall <-
+ c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
> rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
> print(rainfall.timeseries)
         Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct
2012   799.0 1174.8   865.1 1334.6   635.4   918.5   685.5   998.6   784.2   985.0
         Nov     Dec
2012   882.8 1071.0
> png(file="rainfall.png")
> plot(rainfall.timeseries)
> dev off()
Error: unexpected symbol in "dev off"
> dev.off()
null device
          1
> rainfall <-
+ c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
> rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
> print(rainfall.timeseries)
         Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct
2012   799.0 1174.8   865.1 1334.6   635.4   918.5   685.5   998.6   784.2   985.0
         Nov     Dec
2012   882.8 1071.0
> plot(rainfall.timeseries)
>
```

Practical no 4

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

```
# Apply k mean to iris and store result
newiris <- iris
newiris $Species <- NULL
(kc <- kmeans (newiris, 3))
# Compare the Species label with the clustering result
table (iris$Species , kc$cluster)
# plot the clusters and their centers
plot (newiris [c ("Sepal. Length", "Sepal. width")], col = kc$cluster)
points (kc$ centers[c, ("Sepal. Length", "Sepal. width")], col = 1:3, pch = 8, cex 2)
```

```
> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris,3))
K-means clustering with 3 clusters of sizes 38, 50, 62

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

Clustering vector:
  [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 2 2
 [36] 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 2 2
 [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[141] 3 3 3 3 3 3 3 3 3 3

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

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
```

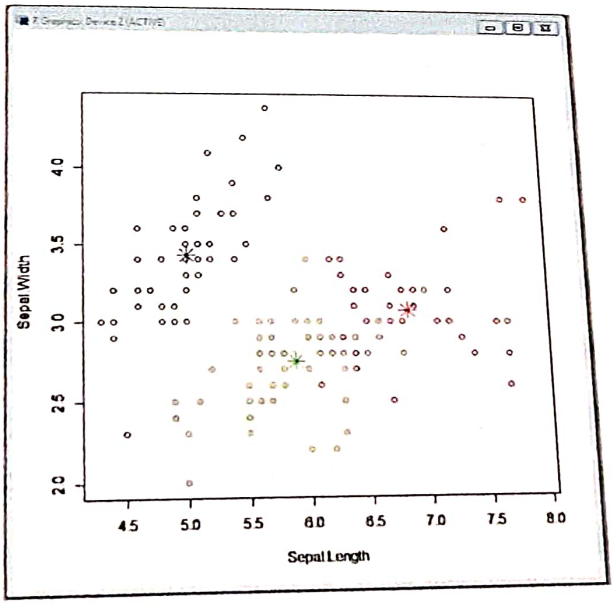
```
 [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[141] 3 3 3 3 3 3 3 3 3 3

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

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> table (iris$Species,kc$cluster)

            1  2  3
  setosa    0 50  0
  versicolor  2  0 48
  virginica  36  0 14
> plot (newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)
> points(kc$centers[,c("Sepal.Lenth","Sepal.Width")],col=1:3,pch=8,cex=2)
Error in kc$centers[, c("Sepal.Lenth", "Sepal.Width")] :
  subscript out of bounds
> points(kc$centers[,c("Sepal.Length","Sepal.Width")],col=1:3,pch=8,cex=2)
>
```



Practical no. 5

* Aim: Perform the Linear Regression on the given data warehouse data using R/Python

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

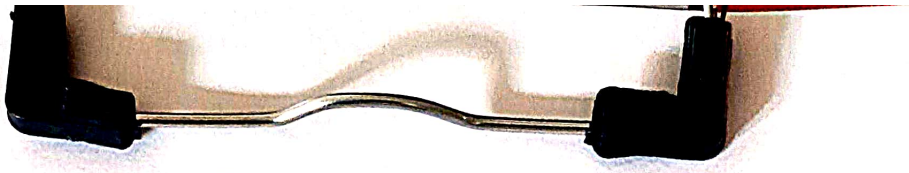
$y = ax + b$ is an equation for linear Regression where y is the response variable, x is the predictor variable and a and b are constants which are called the coefficients

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

The steps to create the relationship is:-

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

```
# Values of height
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
# Values of weight
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Apply the lm() function
relation <-lm(y~x)
# Give the chart file a name
png (file = "linear regression.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()
# plot the chart
plot(y,x, col = "blue" main = "Height & eight Regression",
abline (lm(x~y)), cex = 1.3, pch = 16, xlab = "Weight in kg",
ylab = "Height in cm")
# Save the file
dev.off()
```



Practical no 6

Aim:- Perform the ~~linear~~ logistic regression on the given data warehouse data using R/Python

Steps:- Step 1 : Open R Compiler
Step 2 :- Write above code
Step 3 :- Run the code

```
> mtcars
> input <- mtcars [, c("am", "cyl", "hp", "wt")]
> print(head(input))
> am.data <- glm(formula = am ~ cyl + hp + wt, data = input,
                 family = binomial)
> print(summary(am.data))
```

Practical no7

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)

Step1:- Create Excel file Named as "Elist.csv" & insert data given

Step2:- Open Pycharm IDE And Create a New Project

Step3:- Install the package named as "pandas"

Step4:- Create new python file inside "-venv" file and write above code

Step5:- Copy and paste the excel "Elist.csv" file inside '-venv' folder

Step6:- Run the program

```
import pandas as pd
data =pd. read-csv('prac7.csv')
print ("First 5 rows of data")
print ( data. head)
print ("\n dataset information")
print( data. info ())
print(" summary statistics")
print( data. describe ())
print( "Missing values per column")
print( data. isnull (). sum())
print(" count of unique values in each column")
print(data. nunique ())
```