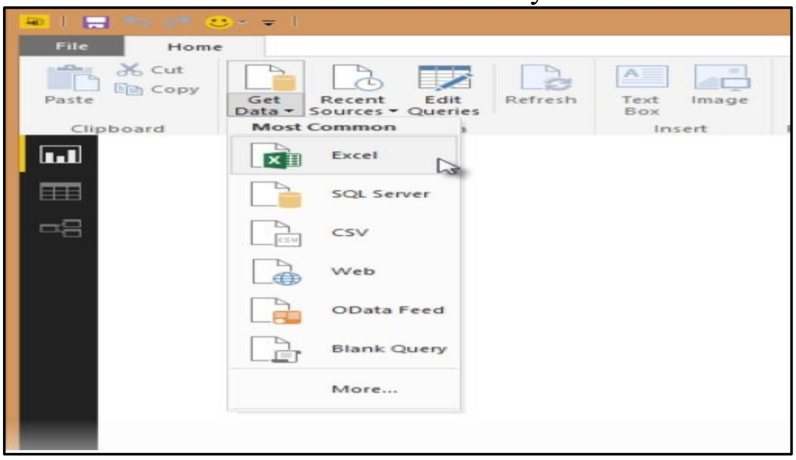
Practical 1: Import the legacy data from Excel and load in the target system

Steps 1: Create an excel sheet with data

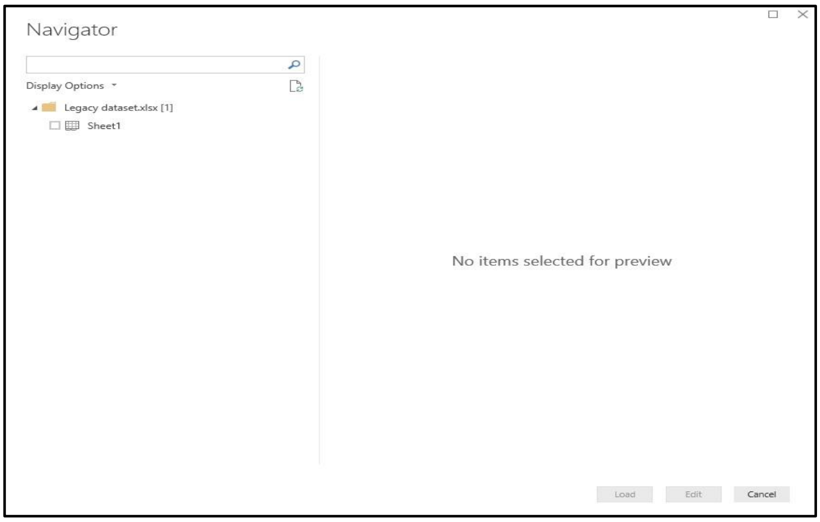
	A	B	C	D	E	F	G
1	OrderDate	Region	Rep	Item	Units	Unit Cost	Total
2	9-1-14	Central	Smith	Desk	2	125.00	250.00
3	6-17-15	Central	Kivell	Desk	5	125.00	625.00
4	9-10-15	Central	Gill	Pencil	7	1.29	9.03
5	11-17-15	Central	Jardine	Binder	11	4.99	54.89
6	10-31-15	Central	Andrews	Pencil	14	1.29	18.06
7	2-26-14	Central	Gill	Pen	27	19.99	539.73
8	10-5-14	Central	Morgan	Binder	28	8.99	251.72
9	12-21-15	Central	Andrews	Binder	28	4.99	139.72
10	2-9-14	Central	Jardine	Pencil	36	4.99	179.64
11	8-7-15	Central	Kivell	Pen Set	42	23.95	1,005.90
12	1-15-15	Central	Gill	Binder	46	8.99	413.54
13	1-23-14	Central	Kivell	Binder	50	19.99	999.50
14	3-24-15	Central	Jardine	Pen Set	50	4.99	249.50
15	5-14-15	Central	Gill	Pencil	53	1.29	68.37
16	7-21-15	Central	Morgan	Pen Set	55	12.49	686.95
17	4-10-15	Central	Andrews	Pencil	66	1.99	131.34
18	12-12-14	Central	Smith	Pencil	67	1.29	86.43
19	4-18-14	Central	Andrews	Pencil	75	1.99	149.25
20	5-31-15	Central	Gill	Binder	80	8.99	719.20
21	2-1-15	Central	Smith	Binder	87	15.00	1,305.00
22	5-5-14	Central	Jardine	Pencil	90	4.99	449.10
23	6-25-14	Central	Morgan	Pencil	90	4.99	449.10
24	12-4-15	Central	Jardine	Binder	94	19.99	1,879.06

Step 2: Open Power BI desktop

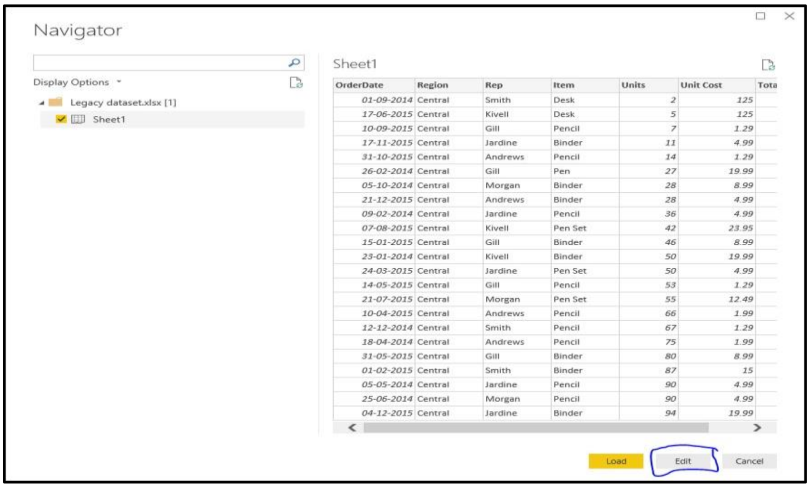
Step 3: Go to Home Ribbon-> Get Data-> Excel and browse your excel file



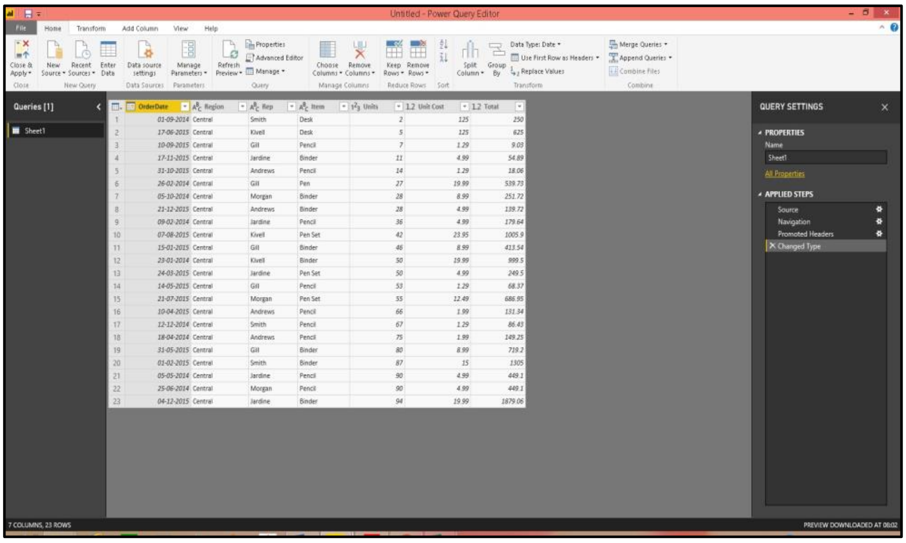
Step 4: In the Navigator tab, select your table(Sheet1) from your dataset(Legacy dataset.xlsx)



Step 5:Click Edit

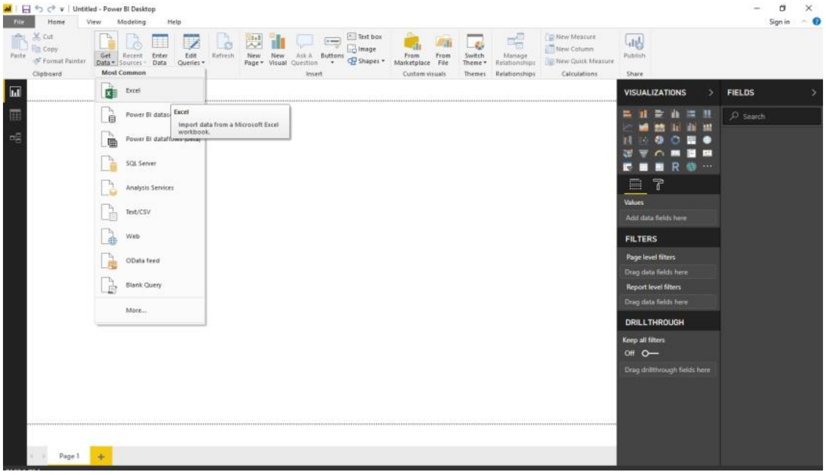


Step 6: You will obtain this screen for queries

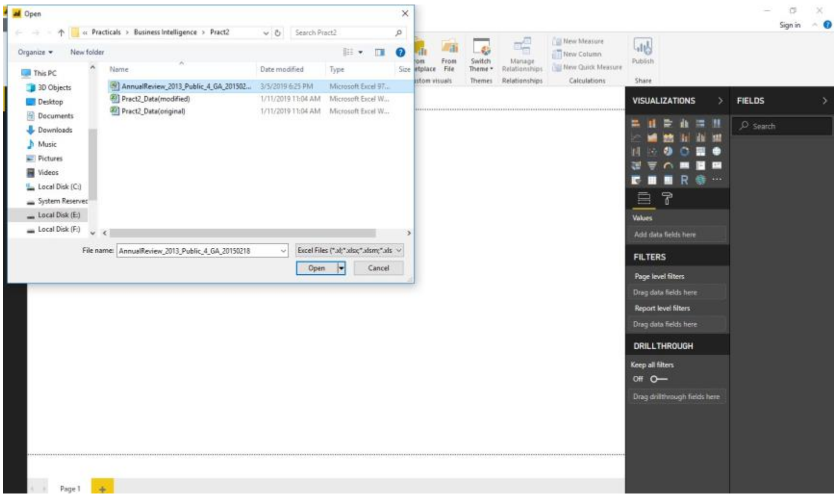


Practical 2: Perform the Extraction Transformation and Loading (ETL) process to construct the database in SQL server.

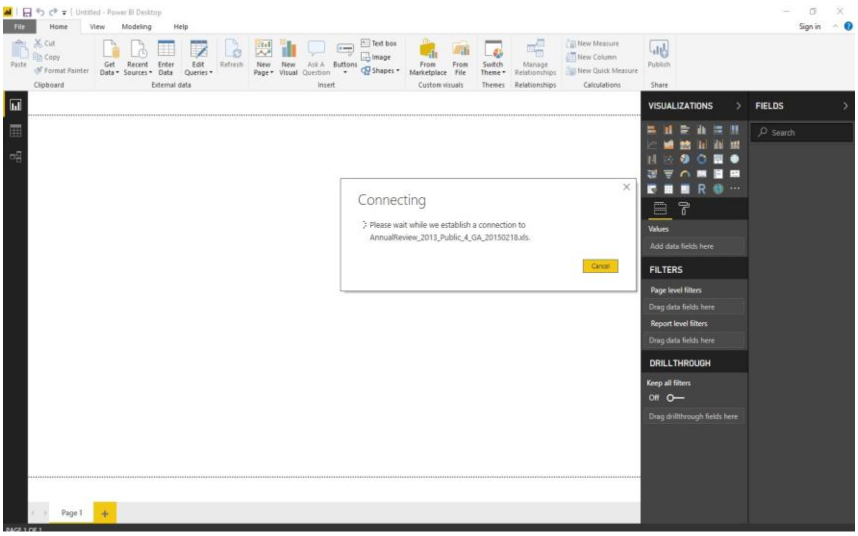
Step 1: Get Data → Excel



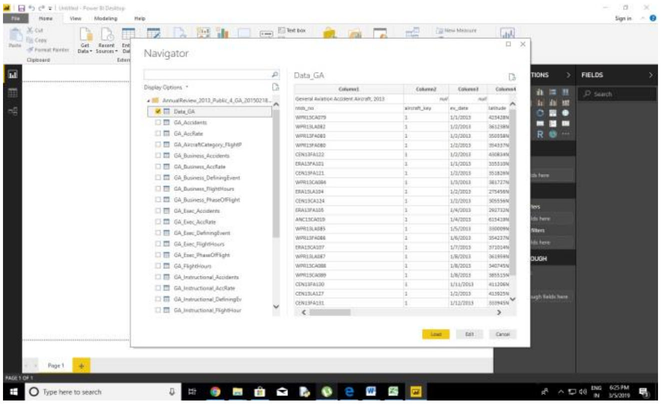
Step 2: Choose you Excel data.



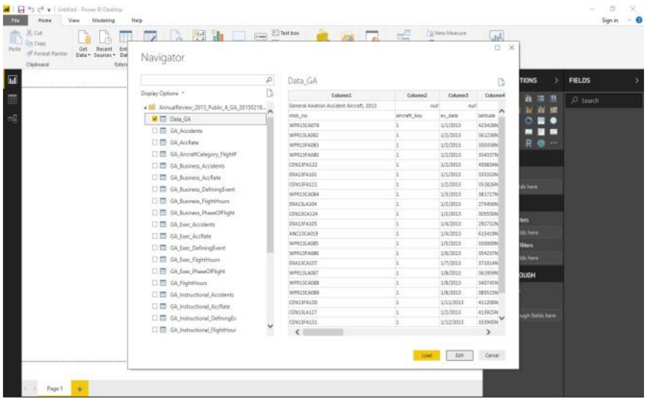
Step 3: Loading might take time.



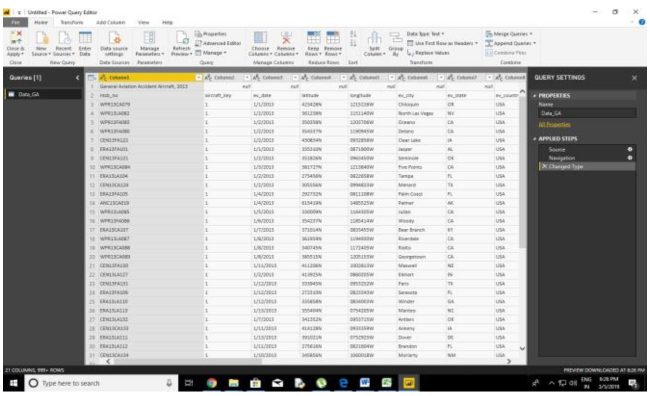
Step 4: Select Data_GA.



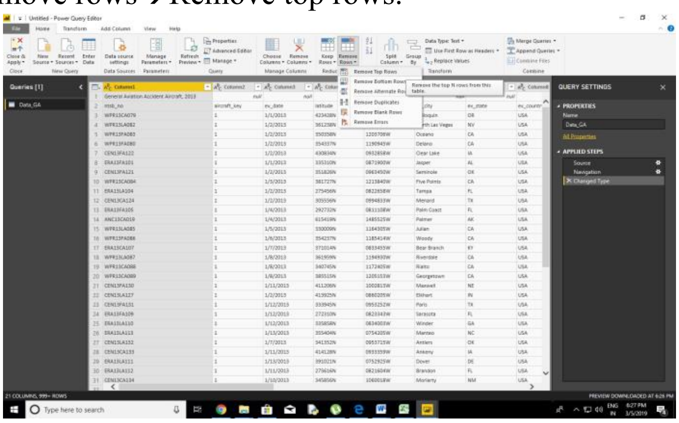
Step 5: Click on Edit button



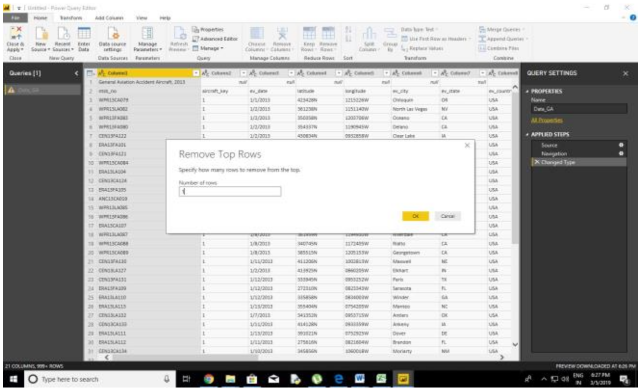
Step 6: Power Query Editor window will open.



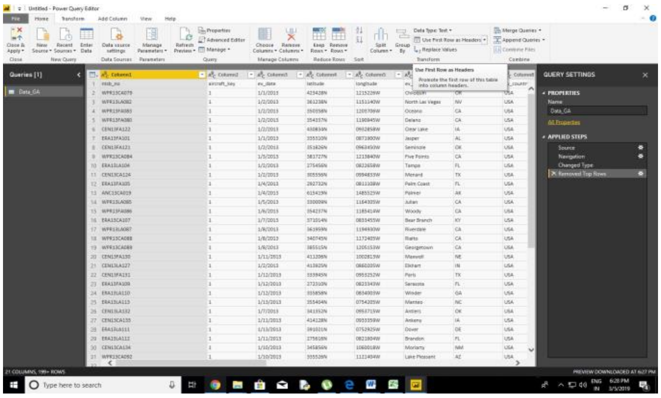
Step 7: Reduce→Remove rows→Remove top rows.



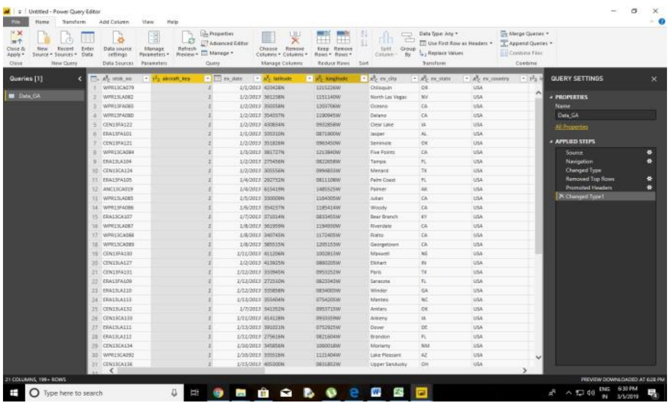
Step 8: Enter the number of rows you want to reduce (here it is just one row from top)



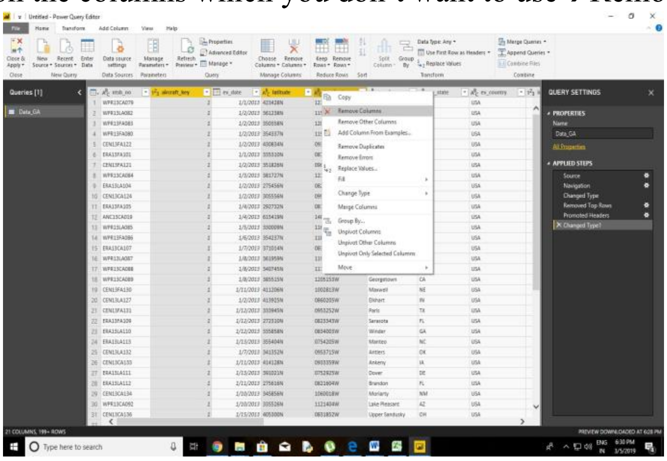
Step 9: Transform→Use first row as headers.



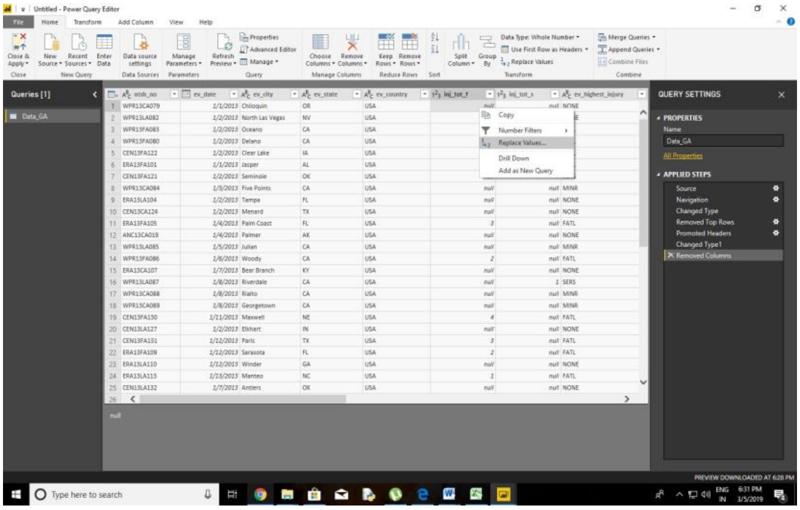
Step 10: Your table has appropriate headers now.



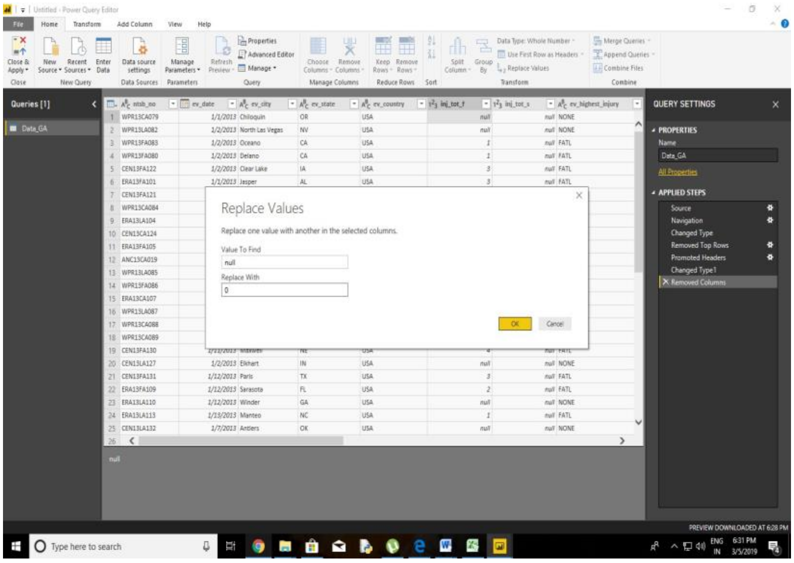
Step 11: Right click on the columns which you don't want to use→Remove Columns.



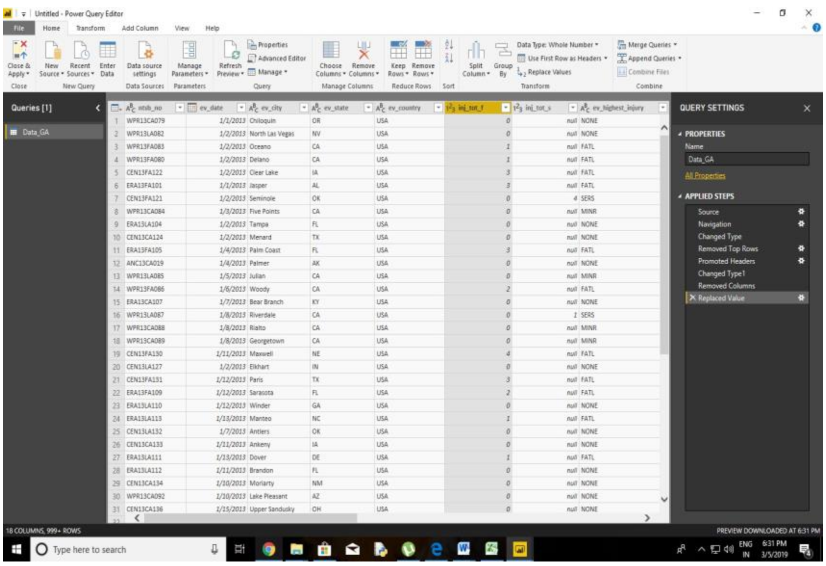
Step 12: Replace values by selecting that cell→right click→replace values…



Step 13: Provide the new value and click on OK.

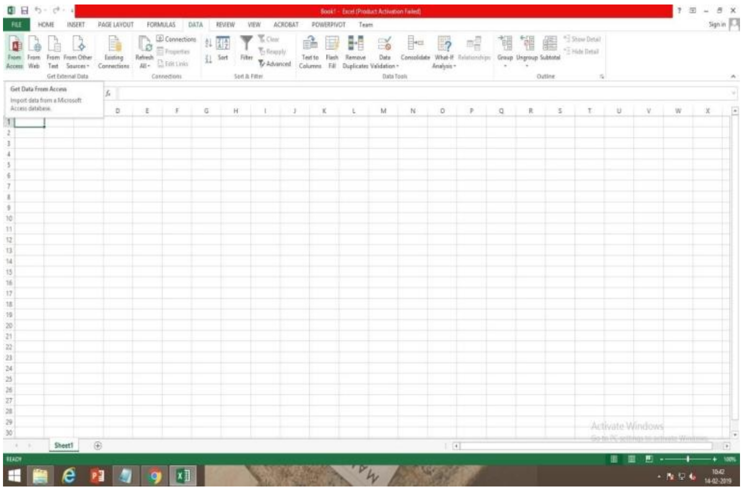


Step 14: All values having the values you wished to replace earlier will be replaced with new values.

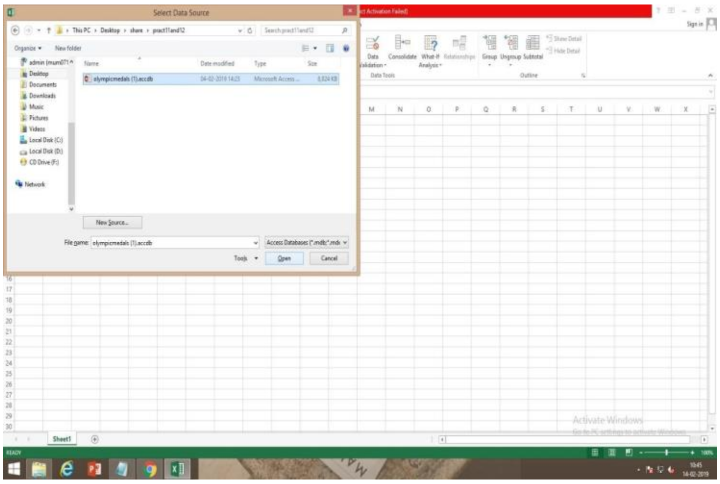


Practical 3(A): Import the datawarehouse data in Microsoft Excel and create the Pivot table.

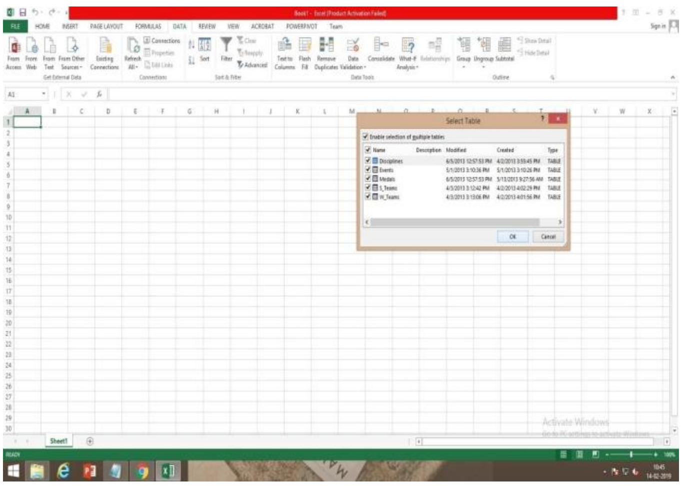
Step 1: Open a blank workbook.
Click Data -> Get External Data -> From Access.



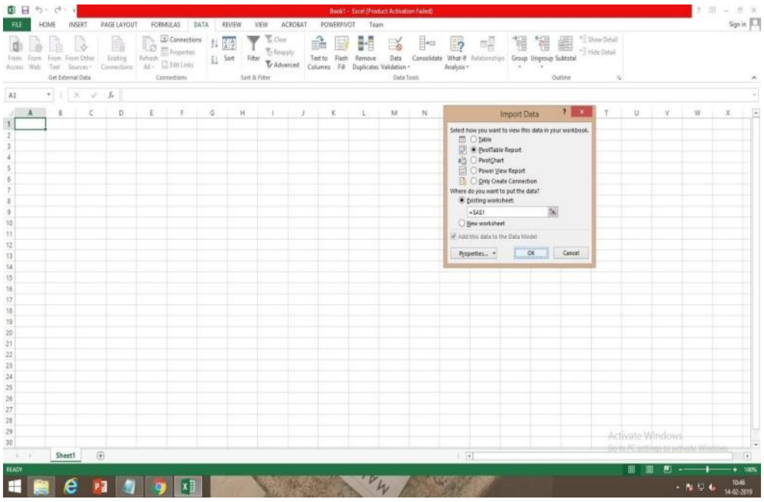
Step 2: Select the OlympicMedals.accdb file and click Open.



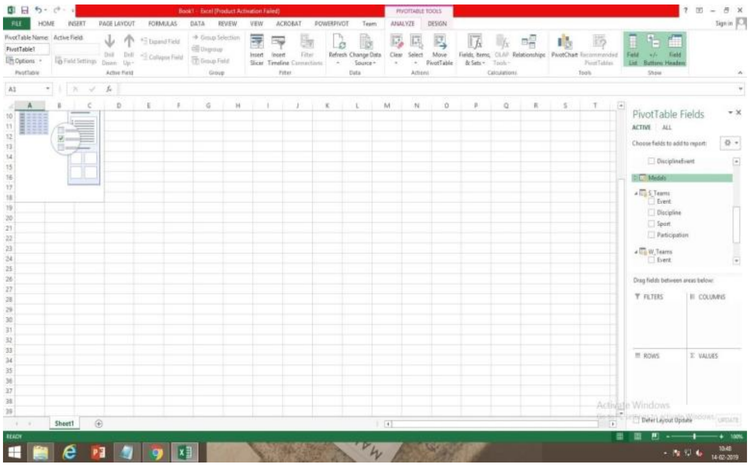
Step 3: Check the Enable Selection of Multiple Tables box and select all the tables. Click OK.



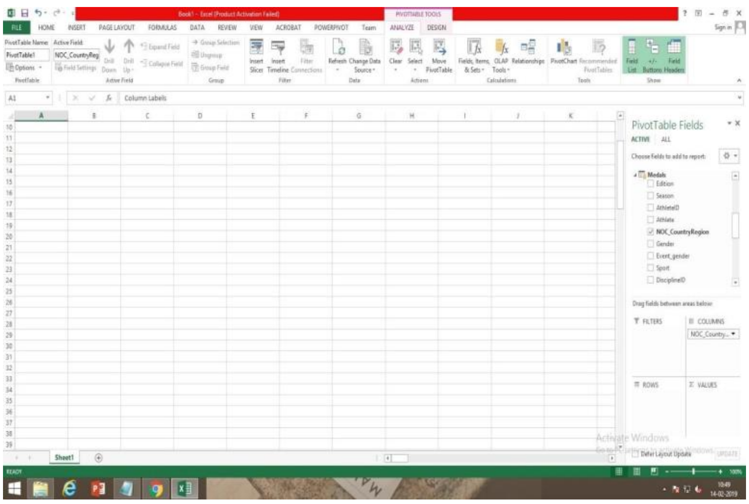
Step 4: The Import Data window appears. Select the PivotTable Report option and click OK.



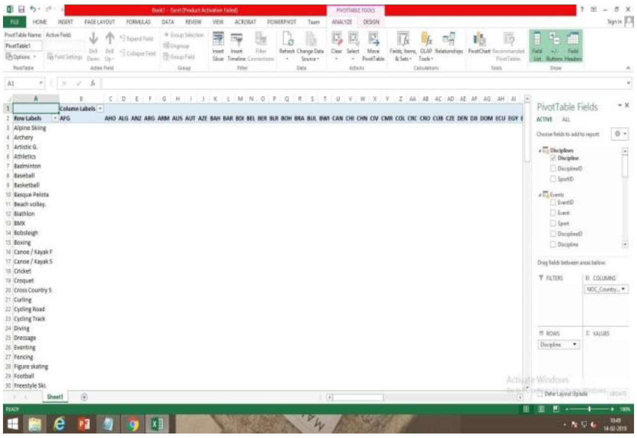
Step 5: A pivot table is created using the imported tables.



Step 6: In PivotTable Fields, expand the Medals table. Find the NOC_CountryRegions and drag it to the Columns area.

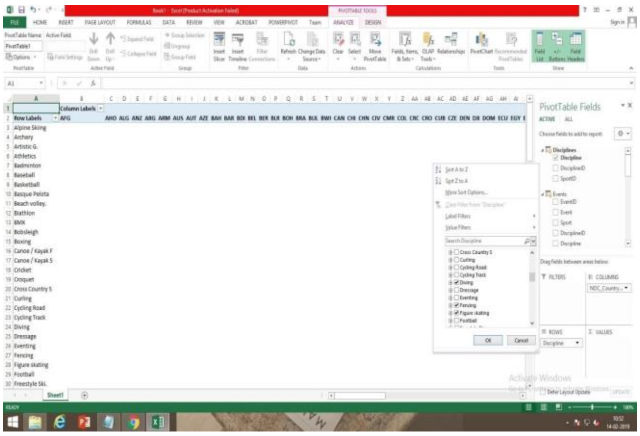


Step 7: Find the Disciplines table and drag it to the Rows area.

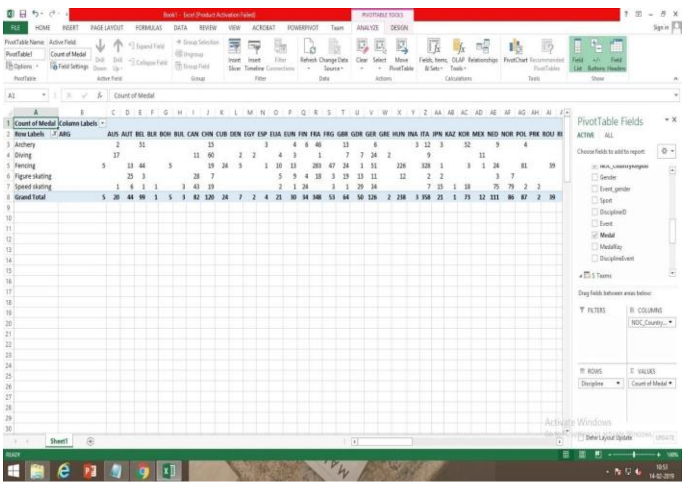


Step 8: filter disciplines to display only five sports: archery, diving, fencing, figure skating and speed skating.

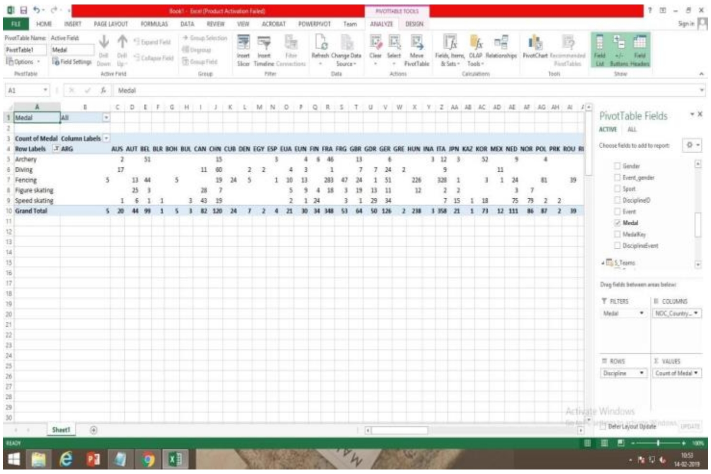
Click anywhere in the PivotTable to ensure the excel PivotTable is selected. In the PivotTable Fields list, where the Disciplines table is expanded, hover over its Discipline field and a drop down arrow appears to the right of the field. Click the dropdown, click "Select All" to remove all selections, then scroll down and select archery, diving, fencing, figure skating and speed skating. Click OK.



Step 9: In PivotTable Fields, from the Medals table, drag Medal to the VALUES area. Since Values must be numeric, Excel automatically changes Medal to Count of Medal.

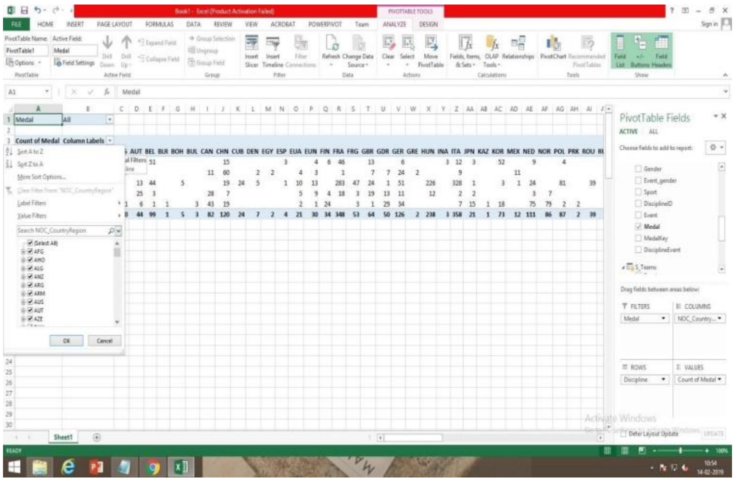


Step 10: From the Medals table, select Medal again and drag it into the FILTERS area.

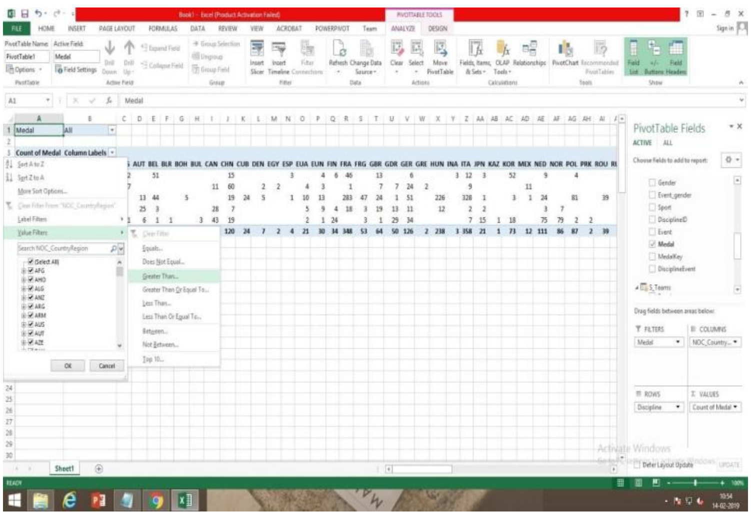


Step 11: Let’s filter the PivotTable to display only those countries or regions with more than 90 total medals.

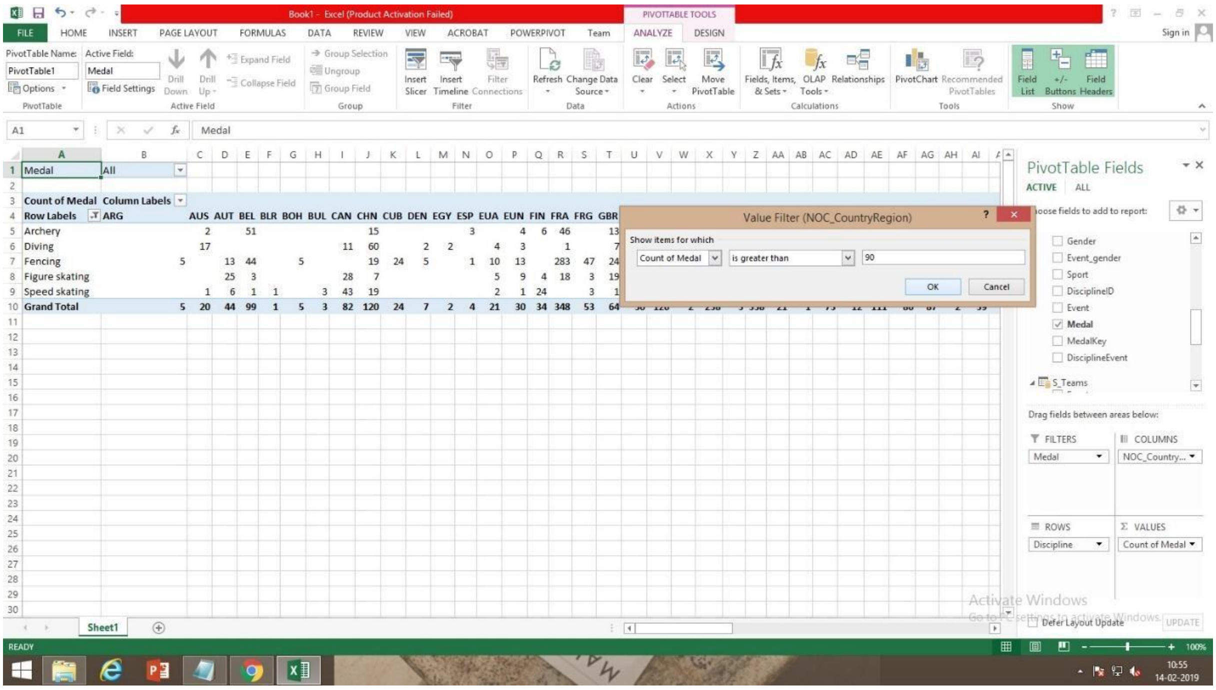
In the PivotTable, click the dropdown to the right of Column Labels.



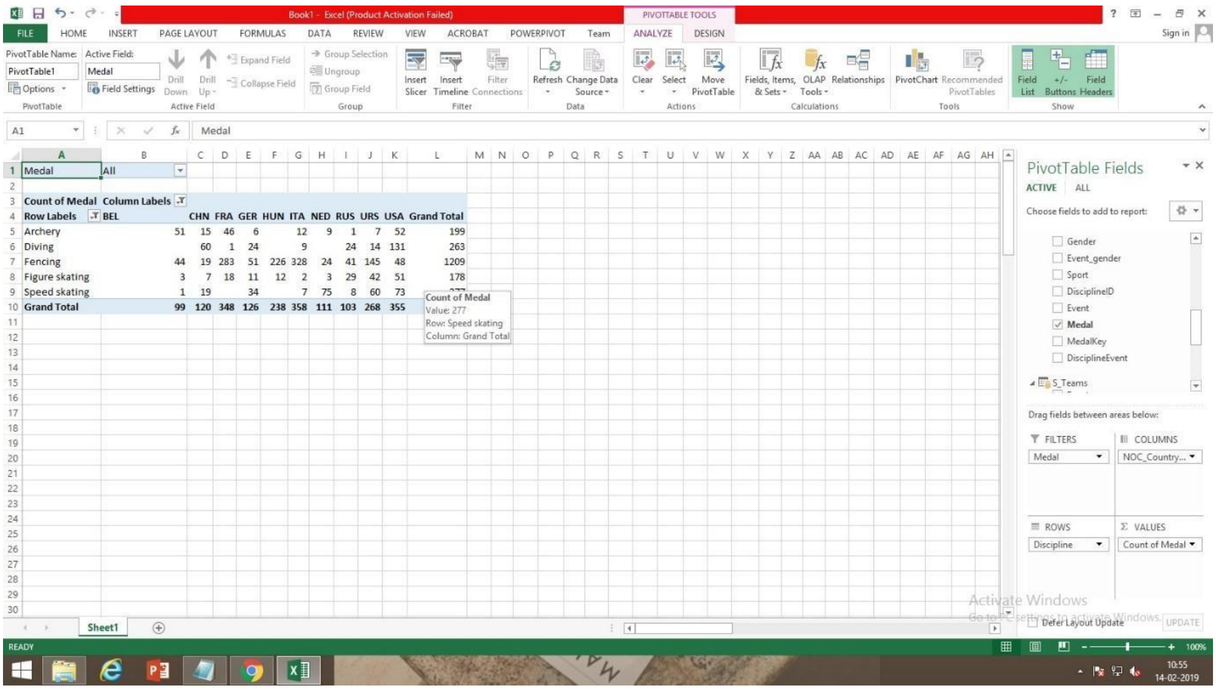
Step 12: Select Value Filters and select Greater Than….



Step 13: Type 90 in the last field (on the right). Click OK.

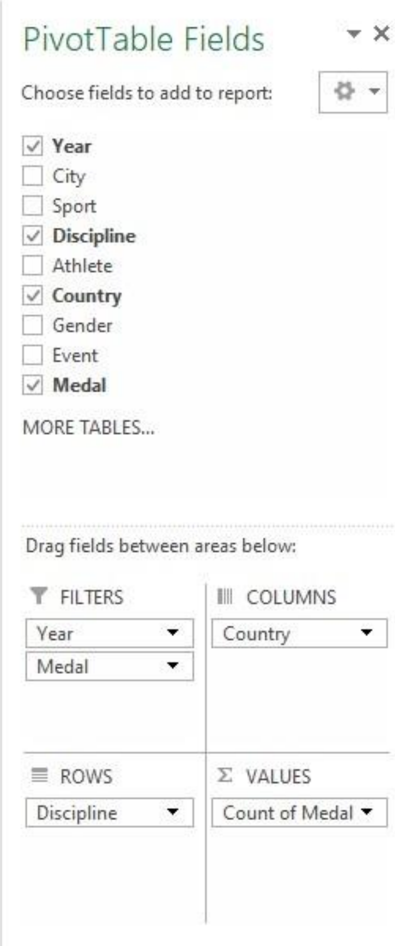


Step 14: Your PivotTable looks like the following screen.

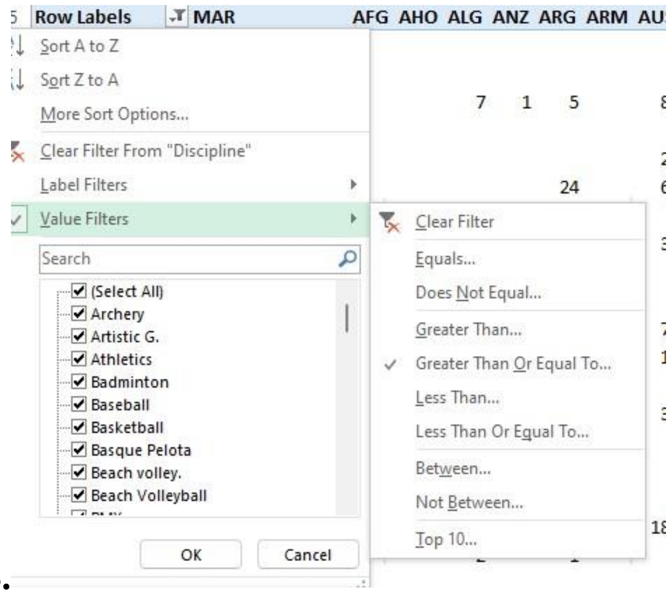


Practical 3(B):

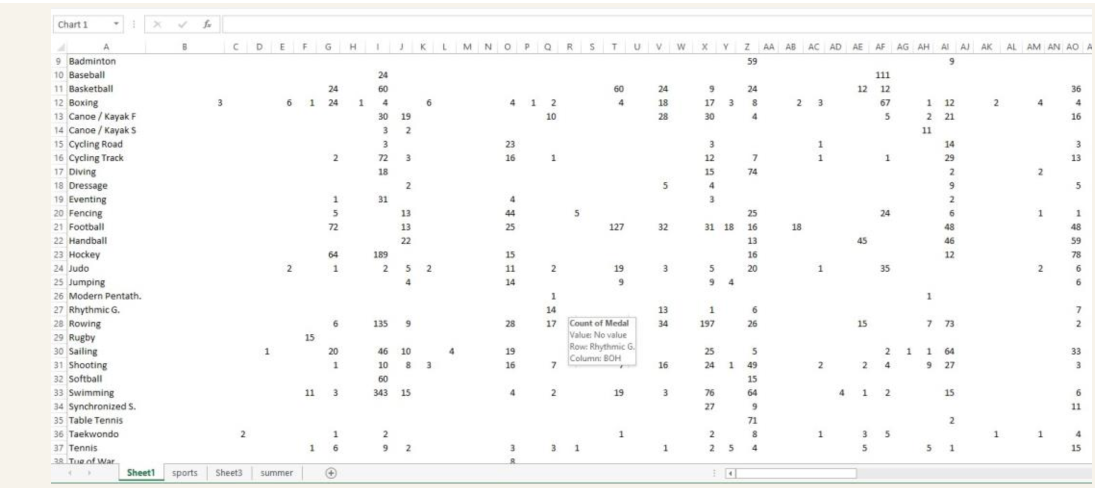
1. Download the dataset(olympics sports and medals database 1896)
2. Open the downloaded file with excel
3. In the insert tab >create pivot table
4. Dragthe following



5.



6.



7.

8. Go to value filters>is greater than or equal to >90 >ok

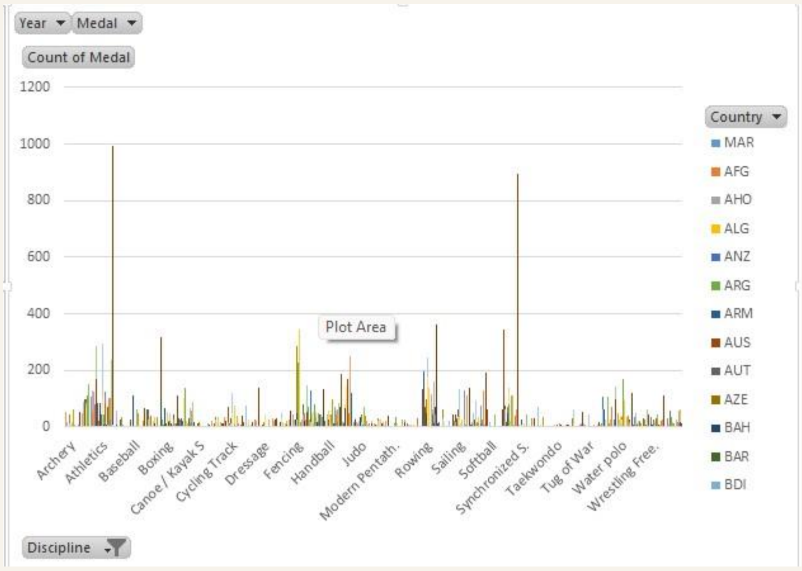
9. In that file only take a new excel sheet amd do ctrl a and ctrl t>it will

<u>Format the table</u>

CREATE PIVOT CHART

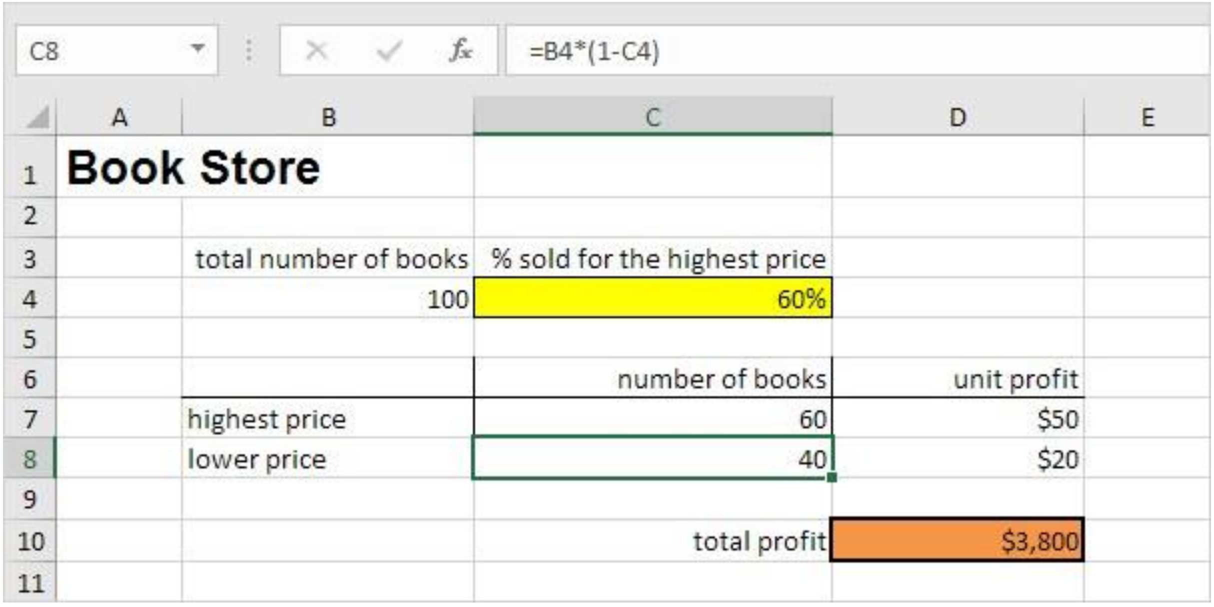
NEW EXCE IN THAT FILE >GO TO FILE >OPTIONS>ADD INS >MANAGE COM ADD INS >GO > SELECT ALL >OKM

10.



Practical 4: Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data.

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.



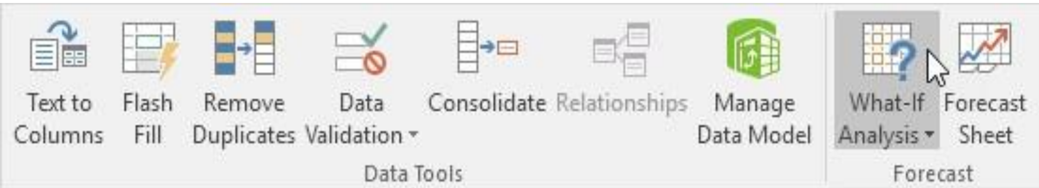
If you sell 60% for the highest price, cell D10 calculates a total profit of 60 * $50 + 40 * $20 = $3800.

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: You can simply type in a different percentage into cell C4 to see the corresponding result of a scenario in cell D10. However, what-if analysis enables you to easily compare the results of different scenarios. Read on.

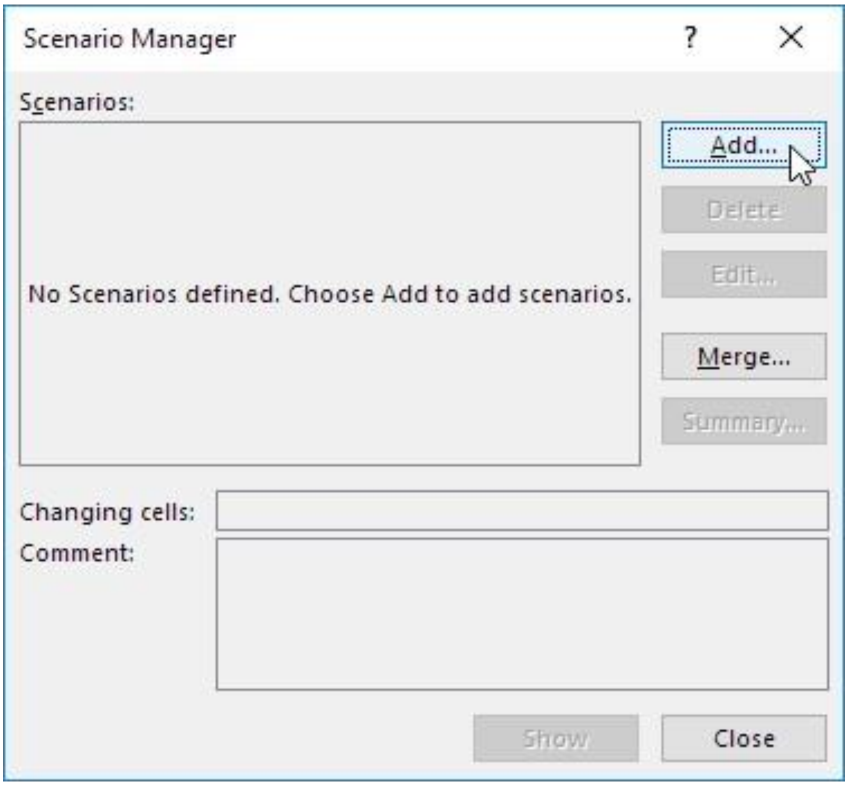
1. On the Data tab, in the Forecast group, click What-If Analysis.



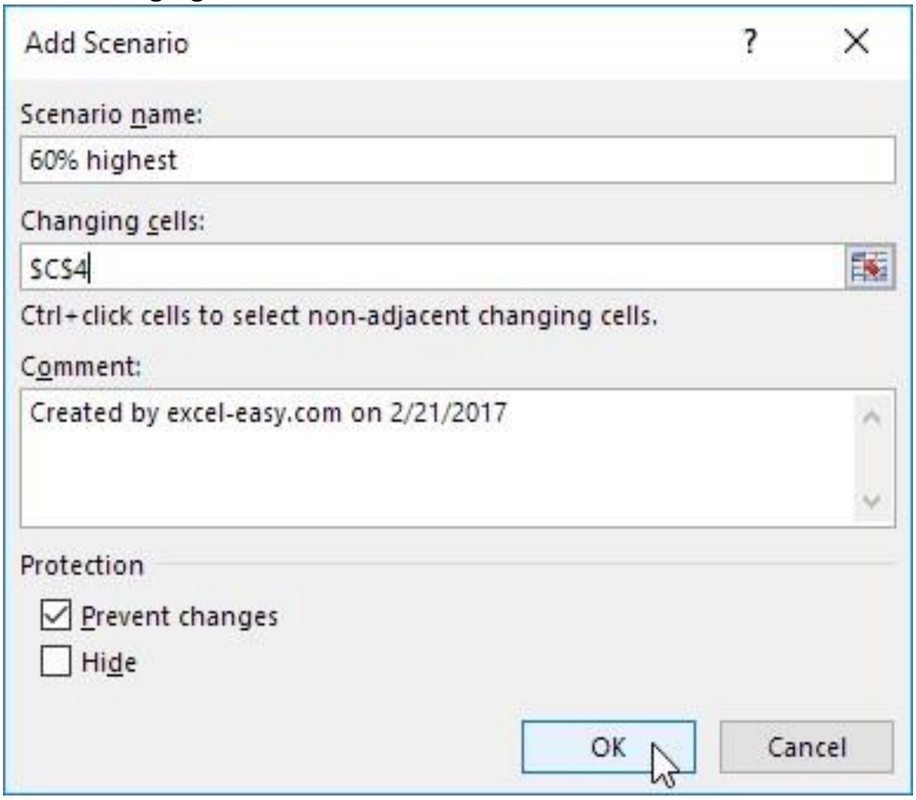
2. Click Scenario Manager.

The Scenario Manager dialog box appears.

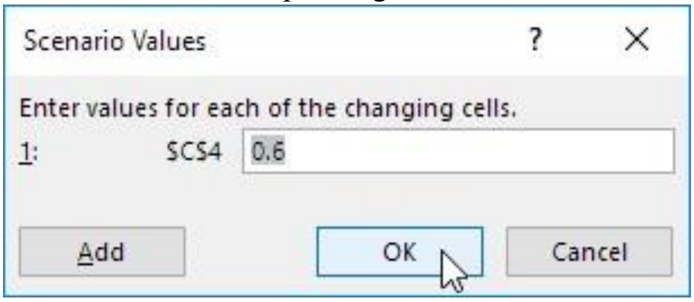
3. Add a scenario by clicking on Add.



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

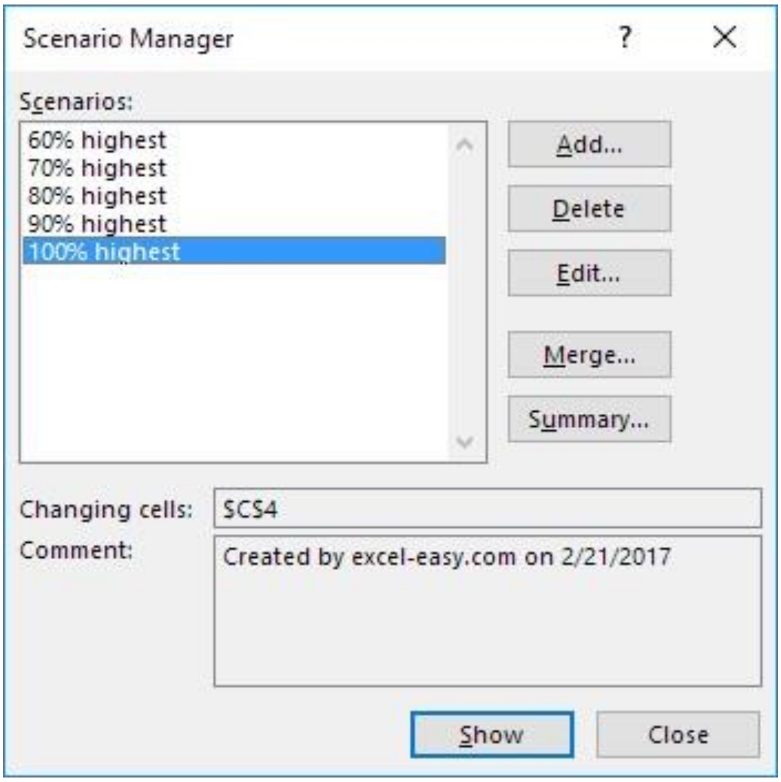


5. Enter the corresponding value 0.6 and click on OK again.



6. Next, add 4 other scenarios (70%, 80%, 90% and 100%).

Finally, your Scenario Manager should be consistent with the picture below:



Practical 5(A): Implementation of Classification algorithm in R Programming.

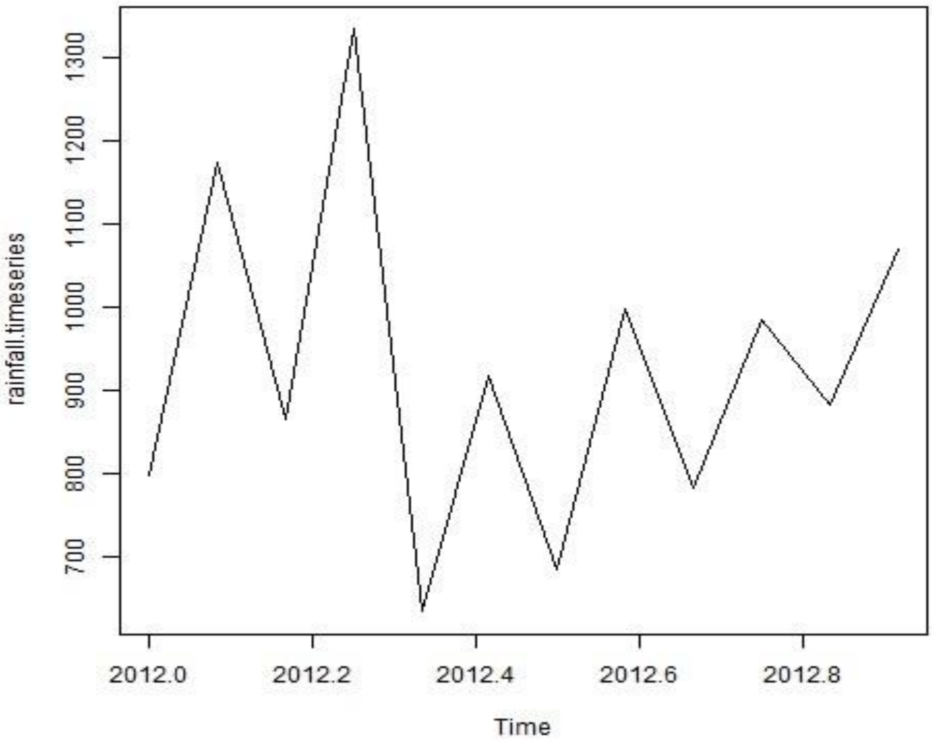
Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

```
# Get the data points in form of a R
vector. rainfall <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,
1071)

# Convert it to a time series object. rainfall.timeseries
<- ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data.
print(rainfall.timeseries)

# Give the chart file a
name. png(file =
"rainfall.png")

# Plot a graph of the time series.
plot(rainfall.timeseries)

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

Output:

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

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



rainfall.timeseries
700
800
900
1000
1100
1200
1300
2012.0
2012.2
2012.4
2012.6
2012.8
Time

Practical 5(B): Practical Implementation of Decision Tree using R Tool

```
install.packages("party")
```

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is −

```
ctree(formula, data)
```

Input Data

We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

Here is the sample data.

```
# Load the party package. It will automatically load other
# dependent packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))
```

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

```
  nativeSpeaker   age   shoeSize
score 1           yes     5
24.83189   32.29385
2            yes     6   25.95238   36.63105
3            no    11   30.42170   49.60593
4            yes     7   28.66450   40.28456
5            yes    11   31.88207   55.46085
6            yes    10   30.07843   52.83124
Loading required package: methods
Loading required package: grid
...............................
...............................
```

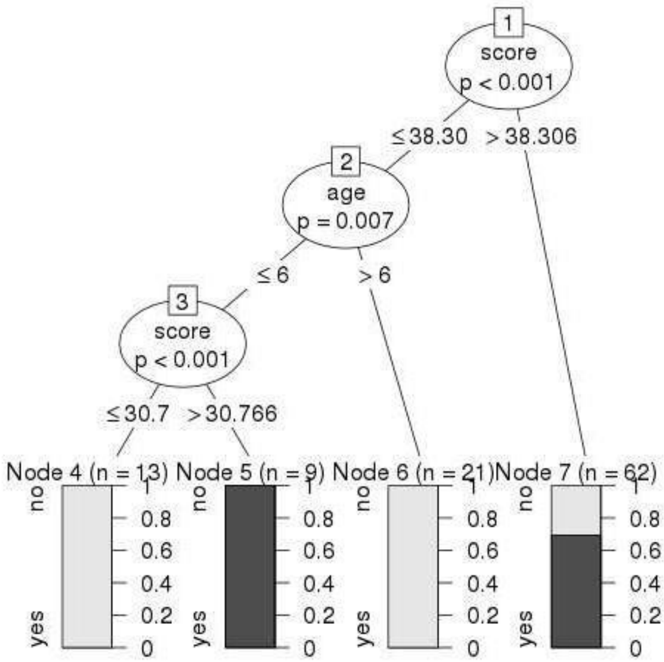
We will use the **ctree()** function to create the decision tree and see its graph.

```
# Load the party package. It will automatically load other
# dependent packages.
library(party)

# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")
```

```
# Create the tree.
output.tree <- ctree( nativeSpeaker ~ age + shoeSize + score, data =
input.dat)

# Plot the tree.
plot(output.tree)

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

Output:-

```
null
devi
ce
1
Loading required package: methods
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

as.Date, as.Date.numeric

Loading required package: sandwich
```



Practical 6: k-means clustering using R

```
R Console
> # Apply K mean to iris and store result
> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris,3))
K-means clustering with 3 clusters of sizes 62, 38, 50

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

Clustering vector:
  [1] 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 3 3 3 3 3 3 3 3
 [47] 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [93] 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2
[139] 1 2 2 2 1 2 2 2 1 2 2 1

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

Available components:

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

Compare the Species label with the clustering result

```
R Console
> #Compare the Species label with the clustering result
> table (iris$Species,kc$cluster)

              1  2  3
  setosa      0  0 50
  versicolor 48  2  0
  virginica  14 36  0
>
```

Plot the clusters and their centre

```
R Console
> # 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)
>
```

Practical 7: Prediction Using Linear Regression

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

y = ax + b is an equation for linear regression.

Where, **y** is the response variable, **x** is the predictor variable and ***a*** and ***b*** are constants which are called the coefficients.

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

The steps to create the relationship is −

- 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 the 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.

Input Data

Below is the sample data representing the observations −

```
# Values of height
151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.
63, 81, 56, 91, 47, 57, 76, 72,
62, 48
```

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for **lm()** function in linear regression is −

```
lm(formula,data)
```

Following is the description of the parameters used −

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

Create Relationship Model & get the Coefficients

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163,
152, 131) y <- c(63, 81, 56, 91, 47, 57, 76, 72,
62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(relation)
```

When we execute the above code, it produces the following result –

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

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

Get the Summary of the Relationship

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163,
152, 131) y <- c(63, 81, 56, 91, 47, 57, 76, 72,
62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(summary(relation))
```

When we execute the above code, it produces the following result –

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

Residuals:
    Min      1Q     Median      3Q     Max
-6.3002    -1.6629  0.0412    1.8944  3.9775

Coefficients:
             Estimate  Std.  Error  t  value
Pr(>|t|)     (Intercept) -38.45509   8.04901
-4.778  0.00139 **  x               0.67461
0.05191  12.997 1.16e-06 *** ---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1
Residual standard error: 3.253 on 8 degrees of freedom
Multiple R-squared:  0.9548,    Adjusted R-squared:
0.9491  F-statistic: 168.9 on 1 and 8 DF,  p-value:
1.164e-06
```

predict() Function

Syntax

The basic syntax for predict() in linear regression is −

```
predict(object, newdata)
```

Following is the description of the parameters used −

- **object** is the formula which is already created using the lm() function.
- **newdata** is the vector containing the new value for predictor variable.

Predict the weight of new persons

```
# The predictor vector. x <- c(151, 174, 138,
186, 128, 136, 179, 163, 152, 131)

# The resposne vector. y <- c(63,
81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

# Find weight of a person with
height 170. a <- data.frame(x =
170)          result          <-
predict(relation,a)
print(result)
Result:
  1
76.22869
```

Visualize the Regression Graphically

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163,
152, 131) y <- c(63, 81, 56, 91, 47, 57, 76,
72, 62, 48) relation <- lm(y~x)

# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart. plot(y,x,col = "blue",main =
"Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab =
"Height in cm")

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

output: