PROGRAM-1 : IMPLEMENT R-LINE GRAPHS

Description: A line graph is a chart that is used to display information in the form of a series of data points. It utilizes points and lines to represent change over time. Line graphs are drawn by plotting different points on their X coordinates and Y coordinates, then by joining them together through a line from beginning to end. The graph represents different values as it can move up and down based on the suitable variable.

The **plot()** function in R is used to create the line graph.

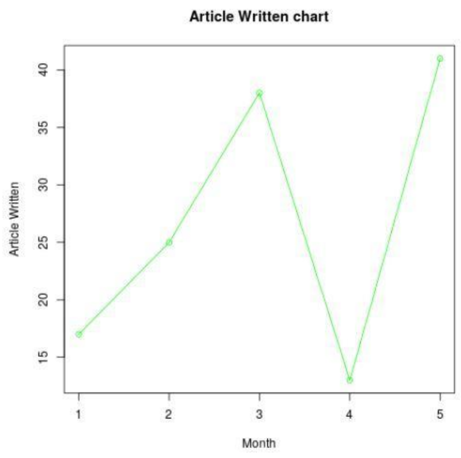
Syntax
The basic syntax to create a line chart in R is −
plot(v,type,col,xlab,ylab)

Following is the description of the parameters used −
v is a vector containing the numeric values.
type takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
xlab is the label for x axis.
ylab is the label for y axis.
main is the Title of the chart.
col is used to give colors to both the points and lines.

PROGRAM CODE: Without dataset

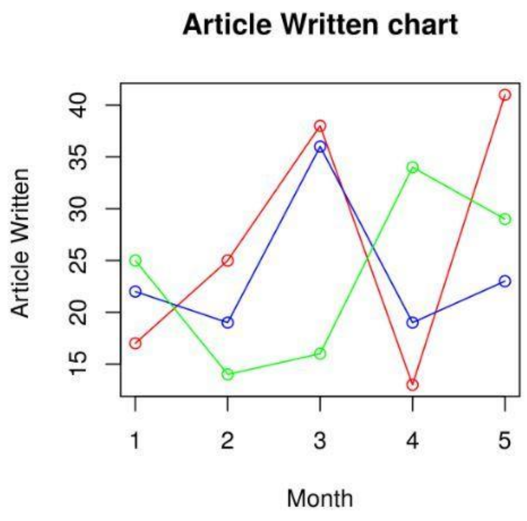
```
# Creating single line plot
# Create the data for the chart.
>v <- c(17, 25, 38, 13, 41)
 #plotting
>plot(v, type = "o", col = "green", xlab = "Month", ylab = "Article Written", main = "Article Written chart")
```

Output:



```
#To create multiple line graphs
# Create the data for the chart.
v <- c(17, 25, 38, 13, 41)
t <- c(22, 19, 36, 19, 23)
m <- c(25, 14, 16, 34, 29)

plot(v, type = "o", col = "red",
    xlab = "Month", ylab = "Article Written ",
    main = "Article Written chart")

lines(t, type = "o", col = "blue")
lines(m, type = "o", col = "green")
```

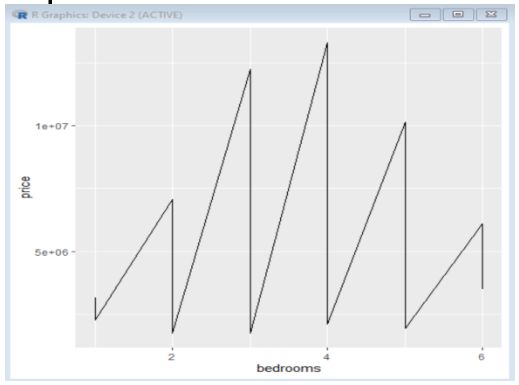
Output:



<u>PROGRAM CODE: With dataset</u>

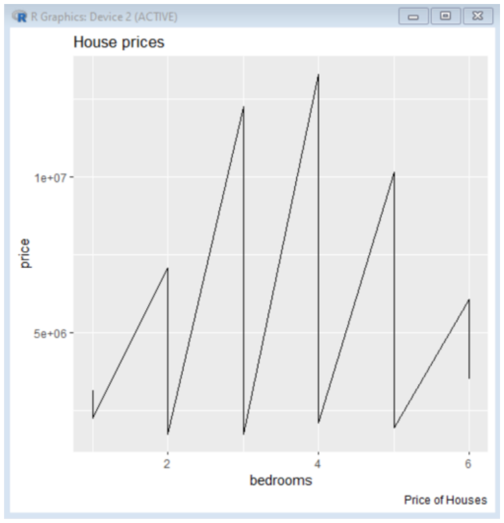
```
install.packages("ggplot2")
install.packages("readr")
install.packages("dplyr")
library(ggplot2)
library(readr)
library(dplyr)

#load the csv file (dataset) in R, storing the dataset in the variable 'k'
k <- read.csv("C:\\Users\\sahithi\\Downloads\\Housing.csv")

#Plotting the data
p <- ggplot() + geom_line(aes(x=bedrooms, y=price), data=k)
p
```

Output:



#Add title, captions, new axis names
p + labs (title = "House prices", x = "bedrooms", y = "price", caption = "Price of Houses")

Output:



PROGRAM-2
IMPLEMENT BAR CHARTS

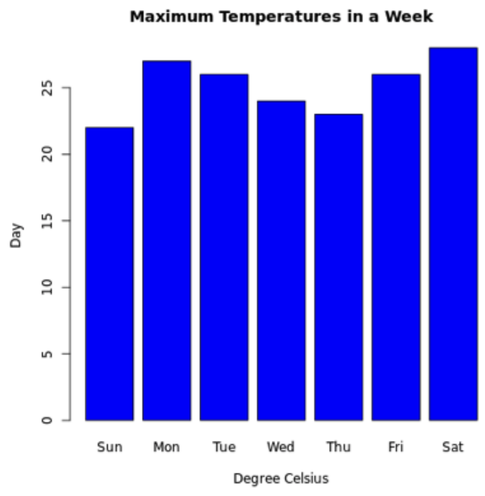
Description: A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts. R can draw both vertical and horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

Syntax
The basic syntax to create a bar-chart in R is −

barplot(H,xlab,ylab,main, names.arg,col)
Following is the description of the parameters used −
H is a vector or matrix containing numeric values used in bar chart.
xlab is the label for x axis.
ylab is the label for y axis.
main is the title of the bar chart.
names.arg is a vector of names appearing under each bar.
col is used to give colors to the bars in the graph.

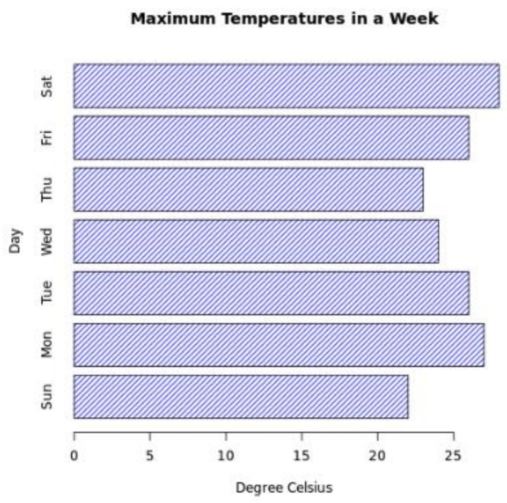
PROGRAM CODE: Without dataset
#vertical bar plot
```
temperatures <- c(22, 27, 26, 24, 23, 26, 28)
result <- barplot (temperatures,
                   main = "Maximum Temperatures in a Week",
                   xlab = "Degree Celsius",
                   ylab = "Day",
                   names.arg = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"),
                   col = "blue"
                   )
print(result)
```

Output:



#Horizontal bar plot:
```
temperatures <- c(22, 27, 26, 24, 23, 26, 28)
result <- barplot(temperatures,
                    main = "Maximum Temperatures in a Week",
                    xlab = "Degree Celsius",
                    ylab = "Day",
                    names.arg = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"),
                    col = "blue",
                    density = 20,
                    horiz = TRUE
                    )
print(result)
```

Output:



<u>PROGRAM CODE: With dataset</u>

Consider **“Titanic dataset”** which is an inbuilt dataset in R. This data set provides information on the fate of passengers on the fatal maiden voyage of the ocean liner ‘Titanic’, summarized according to economic status (class), sex, age and survival

Code:

```
>Titanic
```

Output:

```
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3
```

```
, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes

      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20
```

From the output, we can observe that this dataset has fours dimensions “class, sex, age, survival”
Suppose we wanted to bar plot the count of males and females.
In this case we can use the **margin.table()** function. This function sums up the table entries according to the given index.

Code:

```
> margin.table(Titanic,1)  # count according to class
```

Output:

```
Class
1st  2nd  3rd Crew
325  285  706  885
```

Code:

```
> margin.table(Titanic,4)  # count according to survival
```

Output:

```
Survived
No  Yes
1490  711
```

Code:

```
> margin.table(Titanic)  # gives total count if index is not provided
```

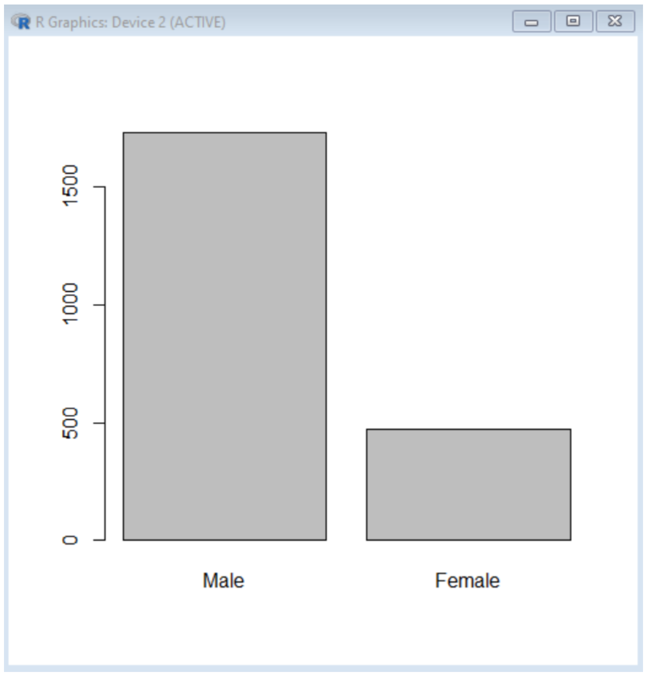
Output:

```
[1] 2201
```

Code

```
> barplot(margin.table(Titanic,2)) #ploting number of male and female survived
```

Output:



PROGRAM-3
IMPLEMENT HISTOGRAMS IN R LANGUAGE

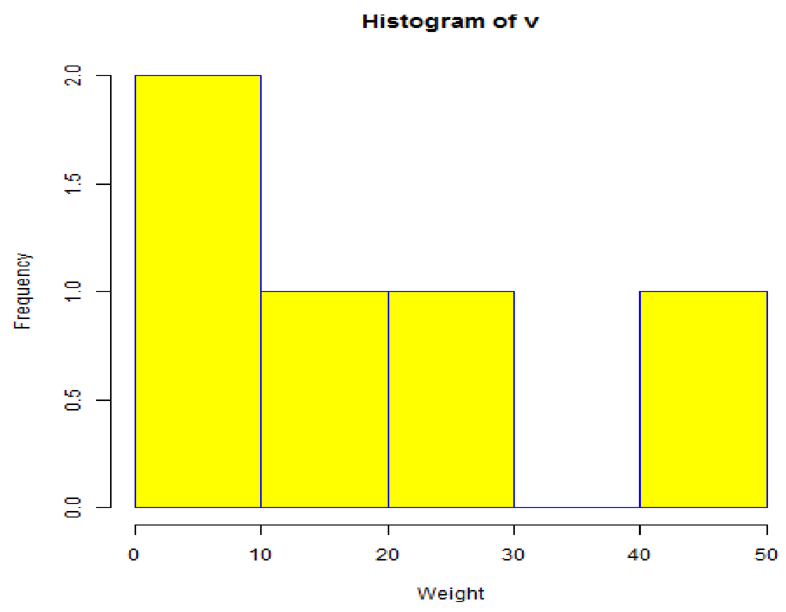
Description: A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.
R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.

Syntax
The basic syntax for creating a histogram using R is −

hist(v,main,xlab,xlim,ylim,breaks,col,border)
Following is the description of the parameters used −
v is a vector containing numeric values used in histogram.
main indicates title of the chart.
col is used to set color of the bars.
border is used to set border color of each bar.
xlab is used to give description of x-axis.
xlim is used to specify the range of values on the x-axis.
ylim is used to specify the range of values on the y-axis.
breaks is used to mention the width of each bar.

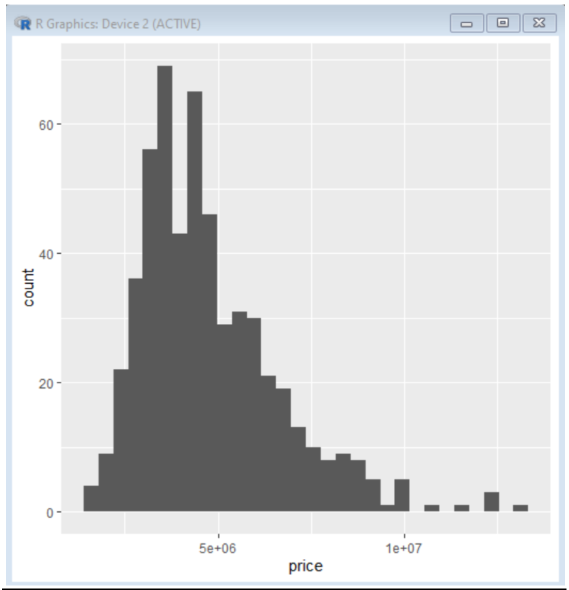
PROGRAM CODE: Without dataset

```
# Create data for the graph.
v <-  c(9,13,21,8,36,22,12,41,31,33,19)

# Create the histogram.
hist(v,xlab = "Weight",col = "yellow",border = "blue")
```



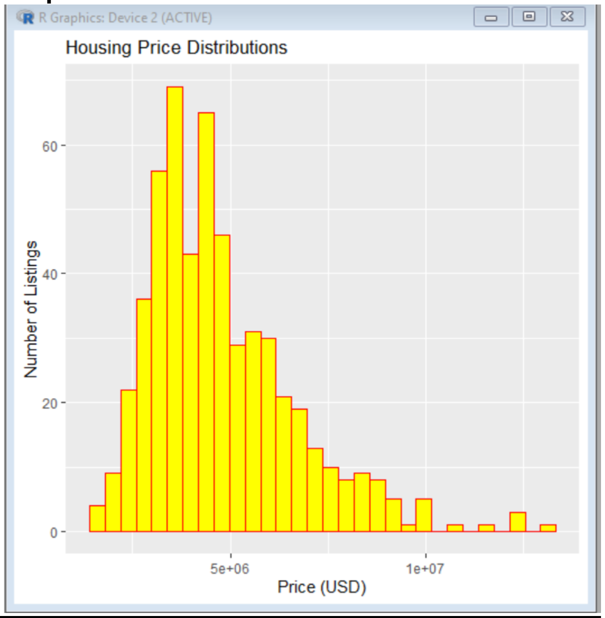
PROGRAM CODE: With dataset

```
#reading dataset
k <- read.csv("C:\\Users\\sahithi\\Downloads\\Housing.csv")

#plotting data
p <- ggplot() + geom_histogram(aes(x=price), data=k)
p
```



```
#Adding colors, titles, labels
p <- ggplot() + geom_histogram(aes(x=price), data=k, color="red", fill="yellow")+ labs(x ='Price (USD)', y='Number of Listings', title = 'Housing Price Distributions')

p
```

Output:



PROGRAM 4
IMPLEMENT R-PIE CHARTS

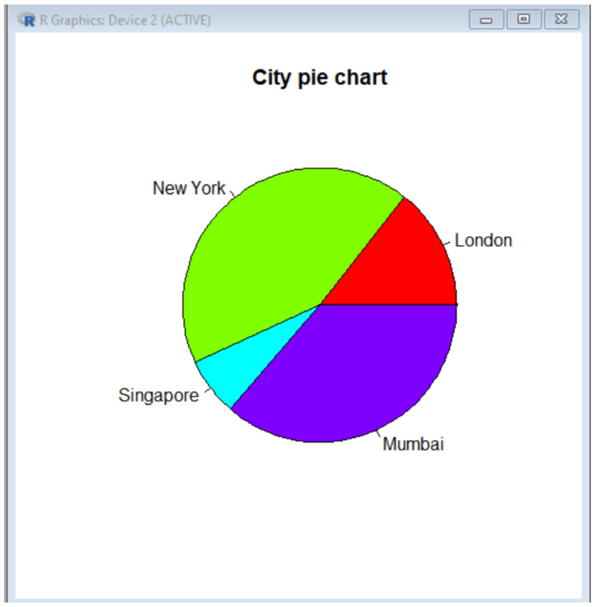
Description: A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportions. It depicts a special chart that uses “pie slices”, where each sector shows the relative sizes of data. A circular chart cuts in a form of radii into segments describing relative frequencies or magnitude also known as a circle graph.
In R the pie chart is created using the **pie()** function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

Syntax
The basic syntax for creating a pie-chart using the R is −

pie(x, labels, radius, main, col, clockwise)
Following is the description of the parameters used −
x is a vector containing the numeric values used in the pie chart.
labels is used to give description to the slices.
radius indicates the radius of the circle of the pie chart.(value between −1 and +1).
main indicates the title of the chart.
col indicates the color palette.
clockwise is a logical value indicating if the slices are drawn clockwise or anti clockwise.

PROGRAM CODE: Without dataset

```
> x <- c(21, 62, 10, 53)
> labels <- c("London", "New York", "Singapore", "Mumbai")
> pie(x, labels, main = "City pie chart", col = rainbow(length(x)))
```



PROGRAM CODE: With dataset
Code:

```
#creating a dataset named “top_ten”
Cities <- c("Visakhapatnam", "Delhi", "Hyderabad", "Mumbai", "Rajahmundry",
          "Chennai", "Bangalore", "Kolkata", "Vijayawada", "Pune")
> Population <- c(1.7, 4.06, 2.68, 2.40, 2.71, 1.58, 1.57, 1.45, 1.40, 1.03 )
> top_ten <- data.frame(Cities, Population)
> top_ten
```

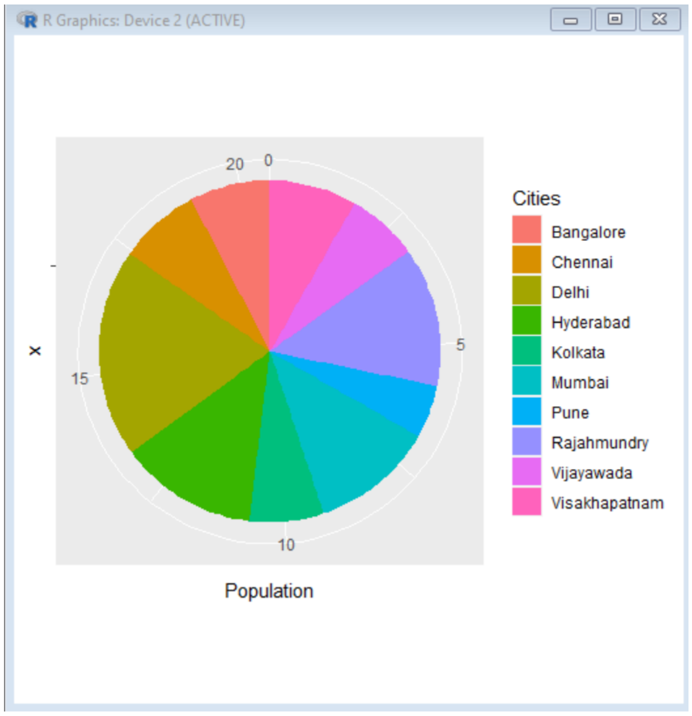
Output:

```
        Cities Population
1  Visakhapatnam       1.70
2          Delhi       4.06
3      Hyderabad       2.68
4         Mumbai       2.40
5    Rajahmundry       2.71
6        Chennai       1.58
7      Bangalore       1.57
8        Kolkata       1.45
9     Vijayawada       1.40
10          Pune       1.03
```

Code:

```
>library(ggplot2)

> ggplot(data = top_ten, aes(x = "", y = Population, fill = Cities)) + geom_bar(stat = "identity") + coord_polar("y")
```

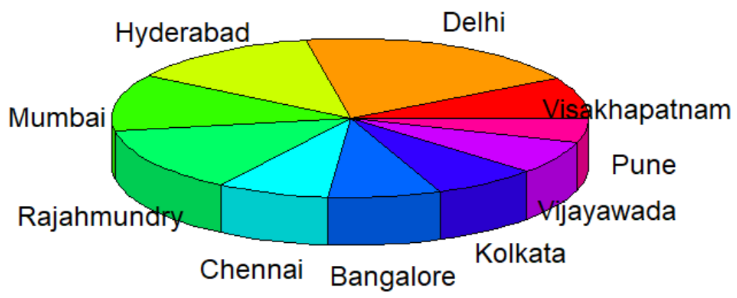
Output:



Code: 3D Pie chart

```
> install.packages("plotrix")
> library(plotrix)
> pie3D(Population, labels=Cities)
```

Output:


Delhi
Hyderabad
Visakhapatnam
Mumbai
Pune
Vijayawada
Rajahmundry
Kolkata
Chennai
Bangalore

PROGRAM 5
IMPLEMENT BOX PLOTS

Description: Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.
Boxplots are created in R by using the **boxplot()** function.

Syntax
The basic syntax to create a boxplot in R is −
boxplot(x, data, notch, varwidth, names, main)

Following is the description of the parameters used −
x is a vector or a formula.
data is the data frame.
notch is a logical value. Set as TRUE to draw a notch.
varwidth is a logical value. Set as true to draw width of the box proportionate to the sample size.
names are the group labels which will be printed under each boxplot.
main is used to give a title to the graph.

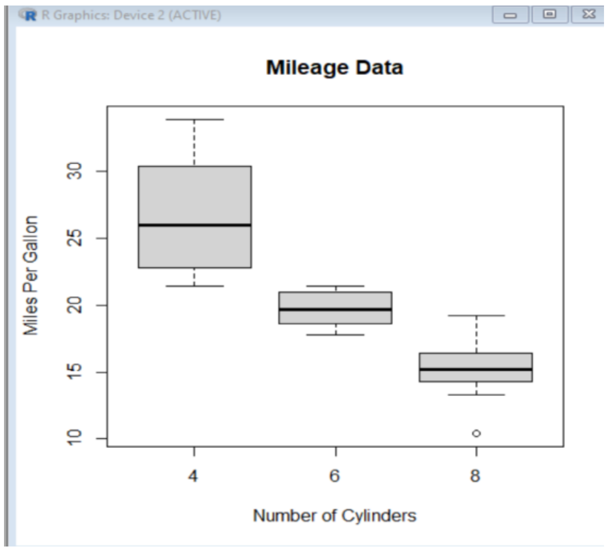
PROGRAM CODE:

Consider **"mtcars dataset"** which is an inbuilt dataset in R.

Code:
```
>mtcars
> p <- mtcars[,c('mpg','cyl')]
> print(head(p))
```

Output:
```
                  mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
```

Code:
```
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab = "Miles Per Gallon", main = "Mileage Data")
```



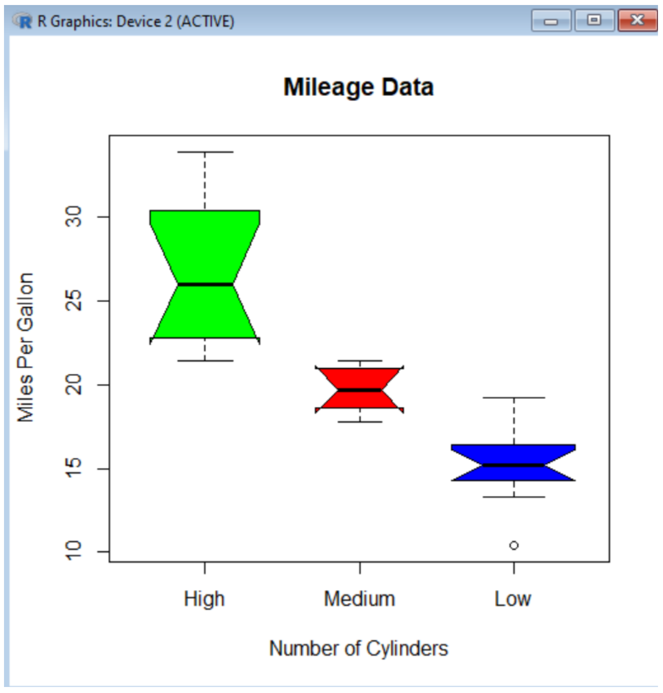
Boxplot using notch

- To draw a boxplot using a notch:
- With the help of notch, we can find out how the medians of different data groups match with each other.
- We are using xlab as “Quantity of Cylinders” and ylab as “Miles Per Gallon”.

Code:

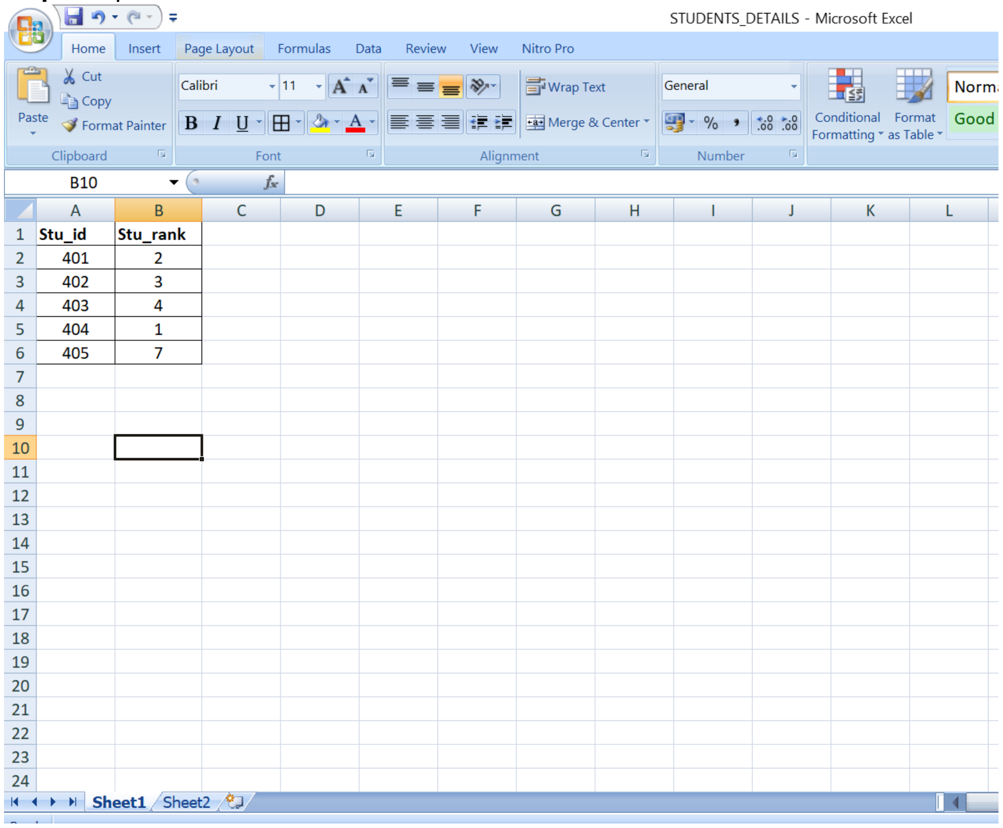
```
> boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab = "Miles Per Gallon",  main = "Mileage Data", notch = TRUE, col = c("green", "red", "blue"), names = c("High", "Medium", "Low"))
```

Output:

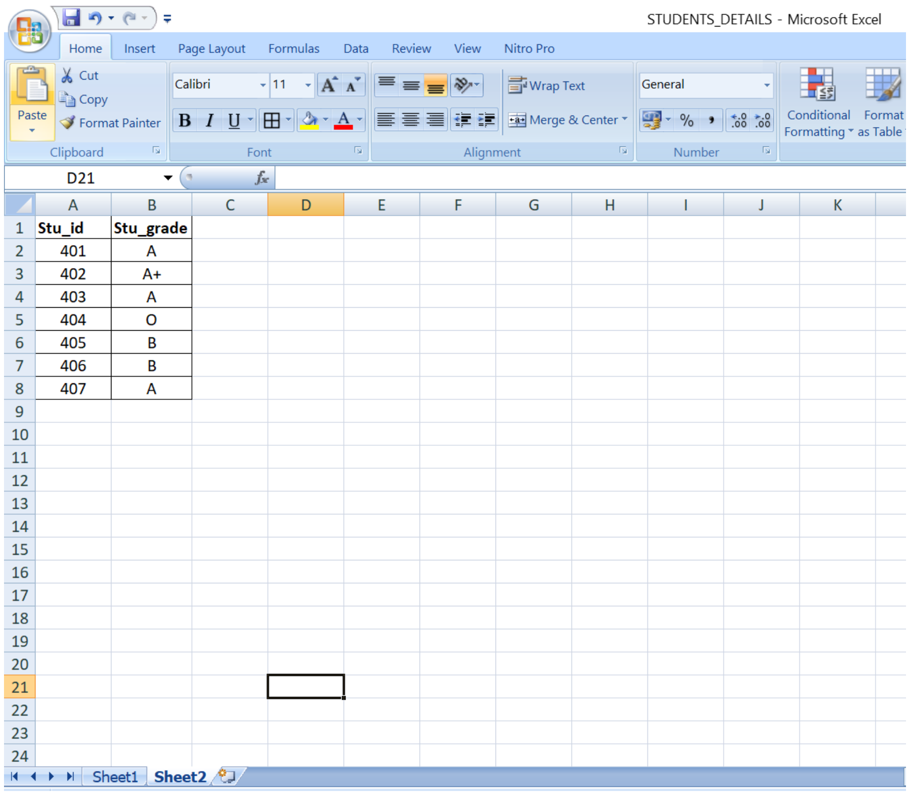


PROGRAM 6
CONNECTING TO MULTIPLE TABLES WITH JOIN IN TABLEAU

Step 1: Open Microsoft Excel and create a dataset which consists of student details, in sheet 1 & sheet 2

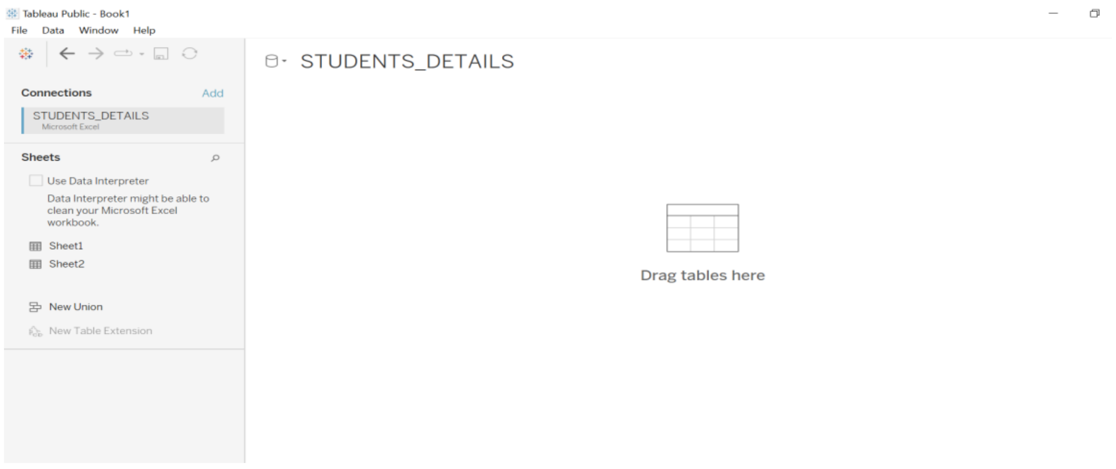


Stu_id	Stu_rank
401	2
402	3
403	4
404	1
405	7

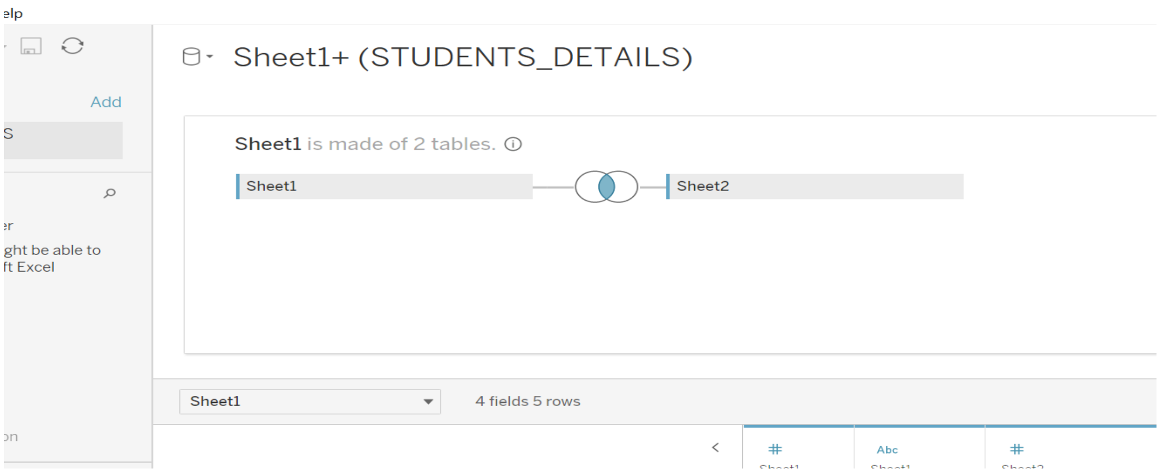


Stu_id	Stu_grade
401	A
402	A+
403	A
404	O
405	B
406	B
407	A

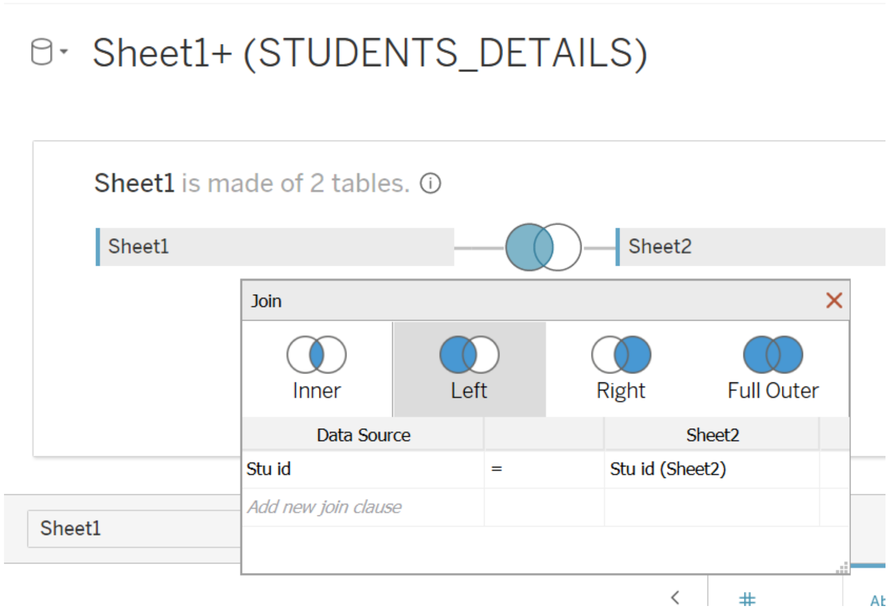
Step 2: Open Tableau, add STUDENT_DETAILS dataset from the data source panel. The sheets which were created in the dataset will be displayed under sheet panel.



Step 3: Drag & drop sheet 1 & 2, then automatically the 2 sheets will be joined.



Step 4: To customize the type of join, we have to double click on the join icon and choose the join type.



Data Source		Sheet2
Stu id	=	Stu id (Sheet2)
Add new join clause		

Step 5: *Left join:* Left join returns all rows from the left and the matching rows from the right table. The result is NULL from the right side if there is no match.

Output:

Stu_id	Stu_rank	Stu_grade
401	2	A
402	3	A+
403	4	A
404	1	O
405	7	B

Right join: When a right join is formed between two tables, the resulting or joined table contains all the values of the right table and only the matching values from the left table. The values that do not find a match in the left table are left as null in the resulting table.

Output:

Stu_id	Stu_rank	Stu_grade
401	2	A
402	3	A+
403	4	A
404	1	O
405	7	B
406	Null	B
407	Null	A

Inner Join: When an inner join is formed between two tables (that is, left and right), the resulting or joined table contains only the values or columns that are common between those tables.

Output:

Stu_id	Stu_rank	Stu_grade
401	2	A
402	3	A+
403	4	A
404	1	O
405	7	B

Outer/Full join: When a full outer join is formed between two tables, the resulting table contains all the data values from both the left and right tables. Also, the values that do not find a match in both the tables are shown as null in the resulting table.

Output:

Stu_id	Stu_rank	Stu_grade
401	2	A
402	3	A+
403	4	A
404	1	O
405	7	B
406	Null	B
407	Null	A

PROGRAM 7
IMPLEMENT DIFFERENT CHARTS IN TABLEAU

Consider “Samplesuper store” dataset downloaded from kaggle and add it to Tableau.
Measure names and Measure values can be used to see the aggregation of all measure present in a data set. These fields can be shown as different types of visualization in Tableau.

Steps:

- Drag ‘Measure Names’ into Columns.
- Drag ‘Measure Values’ into Rows.

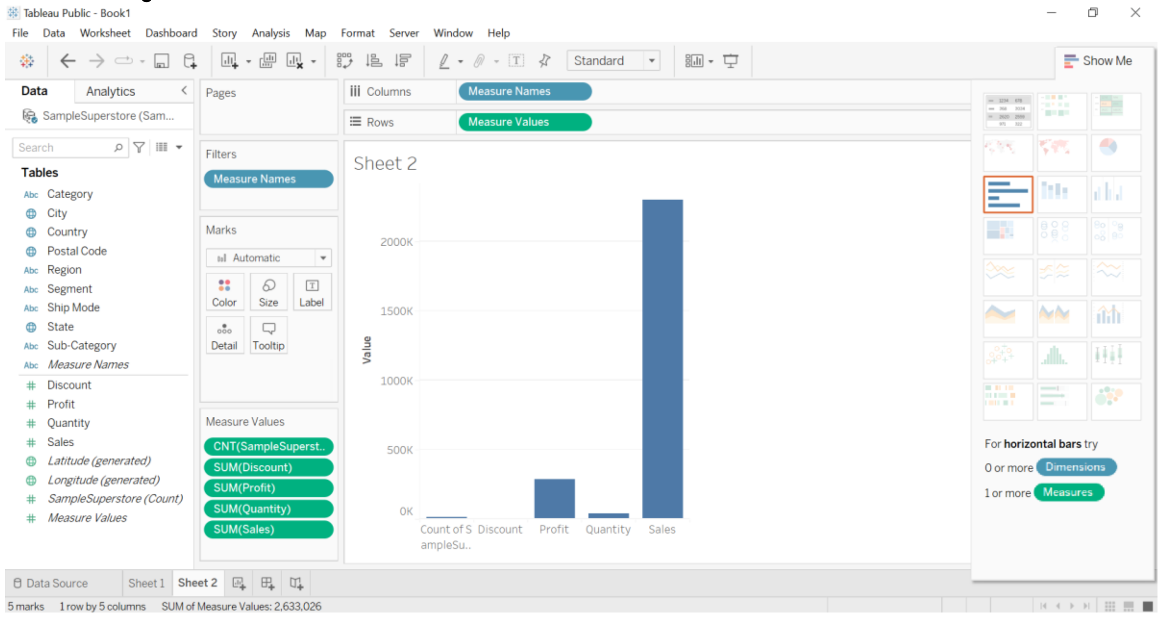


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data Analytics
SampleSuperstore (Sam...
Pages
Columns Measure Names
Rows Measure Values
Filters
Measure Names
Marks
Automatic
Color Size Label Detail Tooltip
Measure Values
CNT(SampleSuperst..
SUM(Discount)
SUM(Profit)
SUM(Quantity)
SUM(Sales)
Tables
Category City Country Postal Code Region Segment Ship Mode State Sub-Category Measure Names
Discount Profit Quantity Sales Latitude (generated) Longitude (generated) SampleSuperstore (Count) Measure Values
Sheet 2
Value
2000K 1500K 1000K 500K 0K
Count of SampleSu.. Discount Profit Quantity Sales
For horizontal bars try
0 or more Dimensions
1 or more Measures
Data Source Sheet 1 Sheet 2
5 marks 1 row by 5 columns SUM of Measure Values: 2,633,026

Chart 1: Horizontal Bar Chart: A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

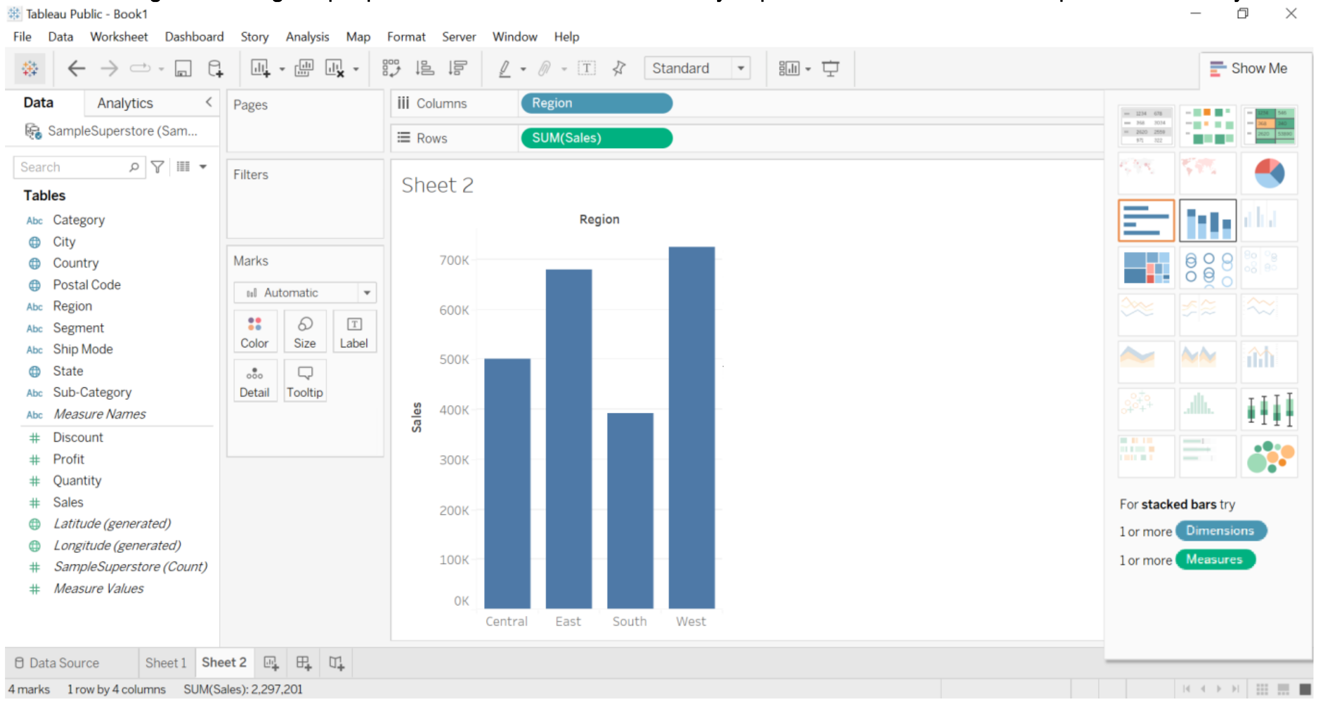


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data Analytics
SampleSuperstore (Sam...
Pages
Columns Region
Rows SUM(Sales)
Filters
Marks
Automatic
Color Size Label Detail Tooltip
Tables
Category City Country Postal Code Region Segment Ship Mode State Sub-Category Measure Names
Discount Profit Quantity Sales Latitude (generated) Longitude (generated) SampleSuperstore (Count) Measure Values
Sheet 2
Region
Sales
700K 600K 500K 400K 300K 200K 100K 0K
Central East South West
For stacked bars try
1 or more Dimensions
1 or more Measures
Data Source Sheet 1 Sheet 2
4 marks 1 row by 4 columns SUM(Sales): 2,297,201

Chart 2: Pie Chart: A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice is proportional to the quantity it represents.

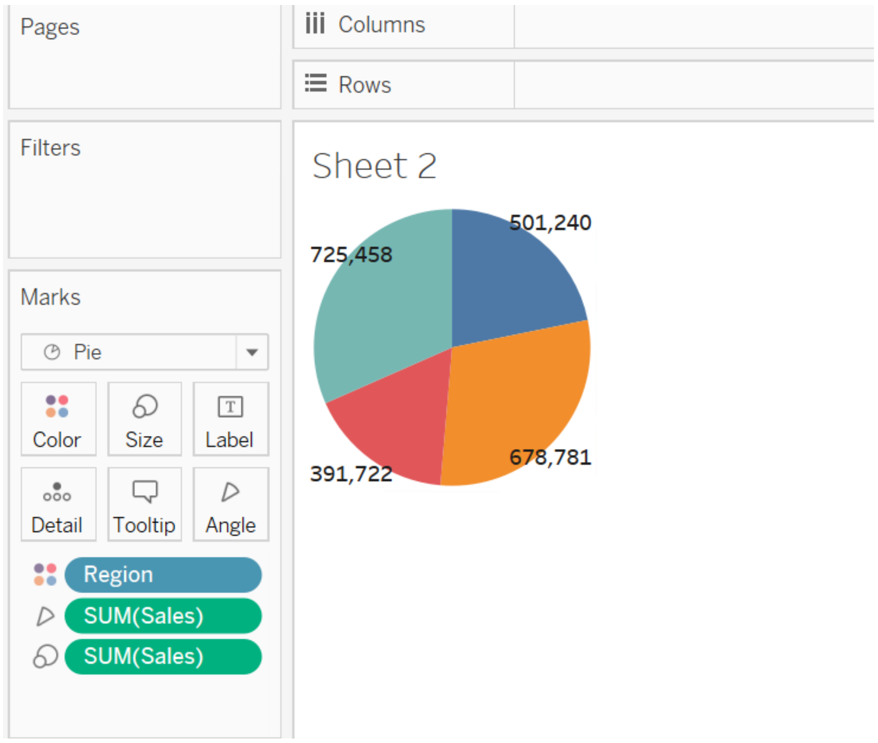


Chart 3: Bubble Chart: A bubble chart is a variation of a scatter chart in which the data points are replaced with bubbles, and an additional dimension of the data is represented in the size of the bubbles. It is the extension of the scatter plot used to look at relationships between three numeric variables.

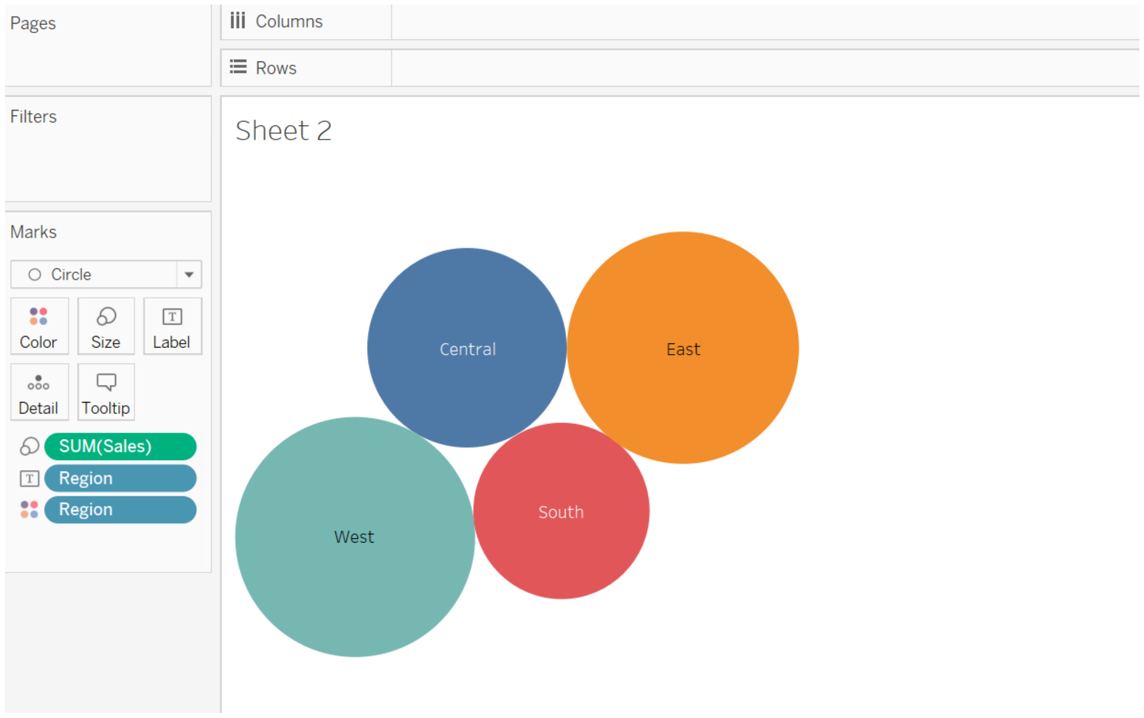
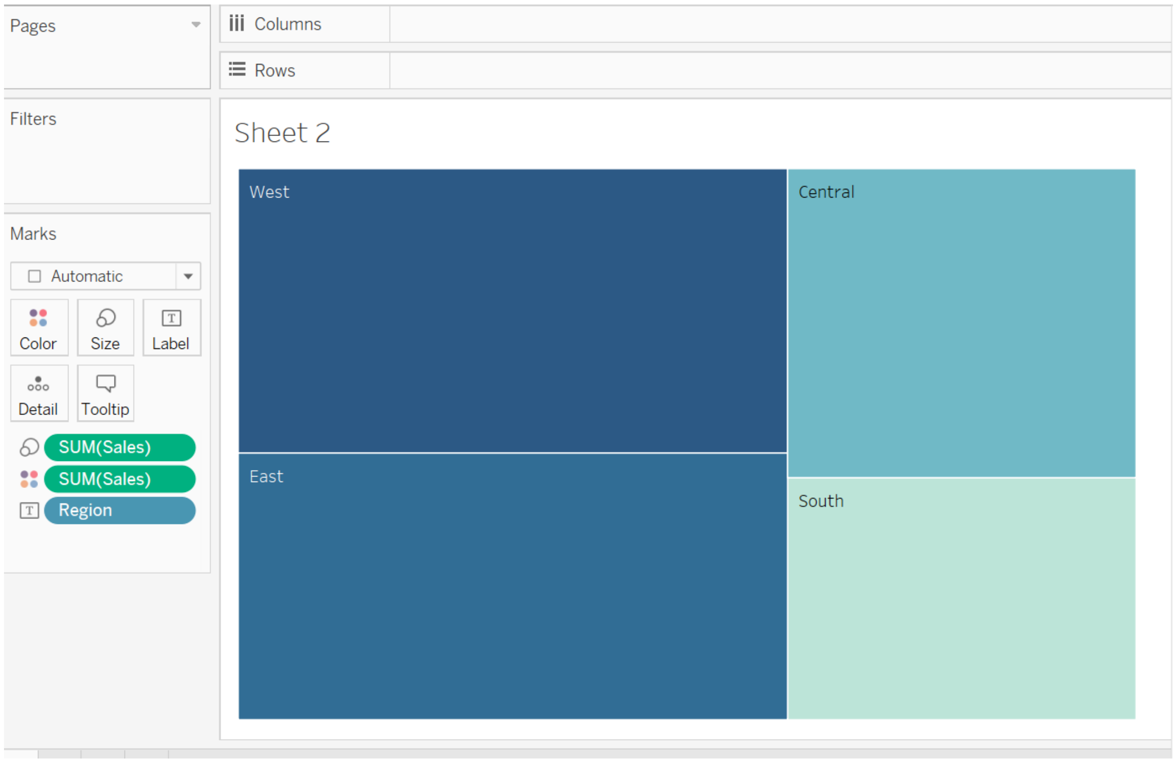


Chart 4: Treemap- Tableau Desktop. Use treemaps to display data in nested rectangles. You use dimensions to define the structure of the treemap, and measures to define the size or color of the individual rectangles. Treemaps are a relatively simple data visualization that can provide insight in a visually attractive format.



PROGRAM-8
DATA PREPARATION IN TABLEAU

Tableau released a brand-new tool to its data visualization product suite, **Tableau Prep**. Tableau Prep is a data preparation tool designed for analysts and business users who try to prepare data for themselves. The tool empowers users to combine, shape, and cleanse data for analysis in Tableau.

Tableau Prep has a clean and friendly user interface. There are a few key panes in the screen:

- Connections pane: Shows the databases and files you are connected to. Add connections to one or more databases and then drag the tables you want to work with into the flow pane.
- Flow pane: As you clean, shape, and combine your data, steps will appear in the flows. This visual indication will allow the user to see an overview of their changes. The user can accomplish a variety of data cleaning tasks in moments, such as fuzzy clustering and other smart features.
- Profile pane: Displays a summary of each field in your data and allows users to see the shape of their data and begin to identify any issues with their data.
- Changes pane: Tableau Prep keeps track of the changes you make, in the order you make them, so you can always go back and review or edit those changes.
- Data grid: Lets the user see row level detail and verify individual records.

The following are the few steps to prepare your data in Tableau:

1. Cleaning Your Data

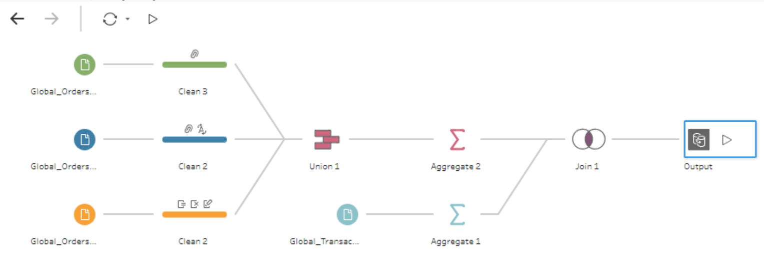
- To create a Flow and begin to clean our data, we need to connect to a data source. Tableau supports a wide variety of on-premise and cloud-based data sources. We'll connect to three text files and create a Flow.
- A Flow is a series of cleaning and transforming steps that users perform on the data and are where you do all of your cleaning, filtering, calculations, and aggregates. We can create calculated fields, split data, and utilize a variety of smart cleaning features built into Tableau Prep.
- We can also profile our data in Prep to spot issues. For example, in one of our datasets we have a lot of NULL Order IDs. We can select those in Prep and begin to understand these NULLs before we move to the analysis phase. For example, we can see all the NULL Order IDs occurred during a few months in early 2013. We can rename these 'Missing Order ID' for further analysis later.

2. Integrating Data

- Once we've cleaned our data, we need to put it together. We can easily union and join datasets in Prep.
- Sometimes the data you join may be at differing levels of granularity. Prep allows users to easily aggregate and dis-aggregate data to get it to the same grain of detail prior to joining.
- Users can customize join conditions, select join types, and view the results of the join all within Prep. Since Prep is iterative, if anything doesn't look right, it's easy to undo changes via the Changes Pane.

3. Loading Data into Tableau

- Loading data into Tableau after you've prepared it is seamless. We simply add an output step at the end of your Flow and click the "play symbol."



- After running our Flow, we can load the Hyper data extract into Tableau Desktop or publish it to Tableau Server as a Published Data Source. We can also choose to generate a .csv file, if you would like.