DATA VISUALIZATION LAB MANUAL
MR23-1CS0150
Index

WEEK-1

Import a sales dataset and perform below data manipulation techniques.

1.Add new rows
2.Create new column “total_revenue” by multiplying quantity sold by the price.
3.Delete rows.
4.Delete column.
5.Reaname “Quantity” column to “Quantity_sold”.
6.Create new columns for “day”,“month” and “year” from “Order Date”.
7.Add +2 to “Quantity” variable of South Region.

```
# load sales dataset
data = read.csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/sales_data.cs
v",fileEncoding = "UTF-8-BOM")
#examin the data
head(data)

##   Row.ID       Order.ID Order.Date  Ship.Date       Country Regio
n
## 1      1 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 2      2 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 3      3 CA-2016-138688 12-06-2016 16-06-2016 United States   Wes
t
## 4      4 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 5      5 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 6      6 CA-2014-115812 09-06-2014 14-06-2014 United States   Wes
t
##          Category    Sales Quantity
## 1       Furniture 261.9600        2
## 2       Furniture 731.9400        3
## 3 Office Supplies  14.6200        2
## 4       Furniture 957.5775        5
## 5 Office Supplies  22.3680        2
## 6       Furniture  48.8600        7

# Check the dimentions
dim(data)

## [1] 690   9

#check the structure of the data
str(data)

## 'data.frame':    690 obs. of  9 variables:
##  $ Row.ID    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Order.ID  : Factor w/ 321 levels "CA-2014-101476",..: 157 157
146 273 273 12 12 12 12 12 ...
##  $ Order.Date: Factor w/ 262 levels "01-02-2014","01-03-2014",..:
 60 60 91 86 86 66 66 66 66 66 ...
##  $ Ship.Date : Factor w/ 279 levels "01-05-2016","01-06-2016",..:
 99 99 143 167 167 126 126 126 126 126 ...
##  $ Country   : Factor w/ 1 level "United States": 1 1 1 1 1 1 1 1
 1 1 ...
##  $ Region    : Factor w/ 4 levels "Central","East",..: 3 3 4 3 3
4 4 4 4 4 ...
##  $ Category  : Factor w/ 3 levels "Furniture","Office Supplies
",..: 1 1 2 1 2 1 2 3 2 2 ...
##  $ Sales     : num  262 731.9 14.6 957.6 22.4 ...
##  $ Quantity  : int  2 3 2 5 2 7 4 6 3 5 ...
```

```
tail(data)
```

```
##     Row.ID       Order.ID Order.Date  Ship.Date       Country Reg
ion
## 685    685 US-2017-168116 04-11-2017 04-11-2017 United States   So
uth
## 686    686 CA-2014-157784 05-07-2014 08-07-2014 United States   So
uth
## 687    687 CA-2014-157784 05-07-2014 08-07-2014 United States   So
uth
## 688    688 CA-2014-157784 05-07-2014 08-07-2014 United States   So
uth
## 689    689 CA-2017-161480 25-12-2017 29-12-2017 United States    E
ast
## 690    690 US-2014-117135 21-06-2014 23-06-2014 United States   So
uth
##            Category   Sales Quantity
## 685 Office Supplies 167.440        2
## 686      Technology 479.970        3
## 687 Office Supplies  14.620        2
## 688 Office Supplies  19.440        3
## 689       Furniture 191.984        2
## 690       Furniture 104.010        1
```

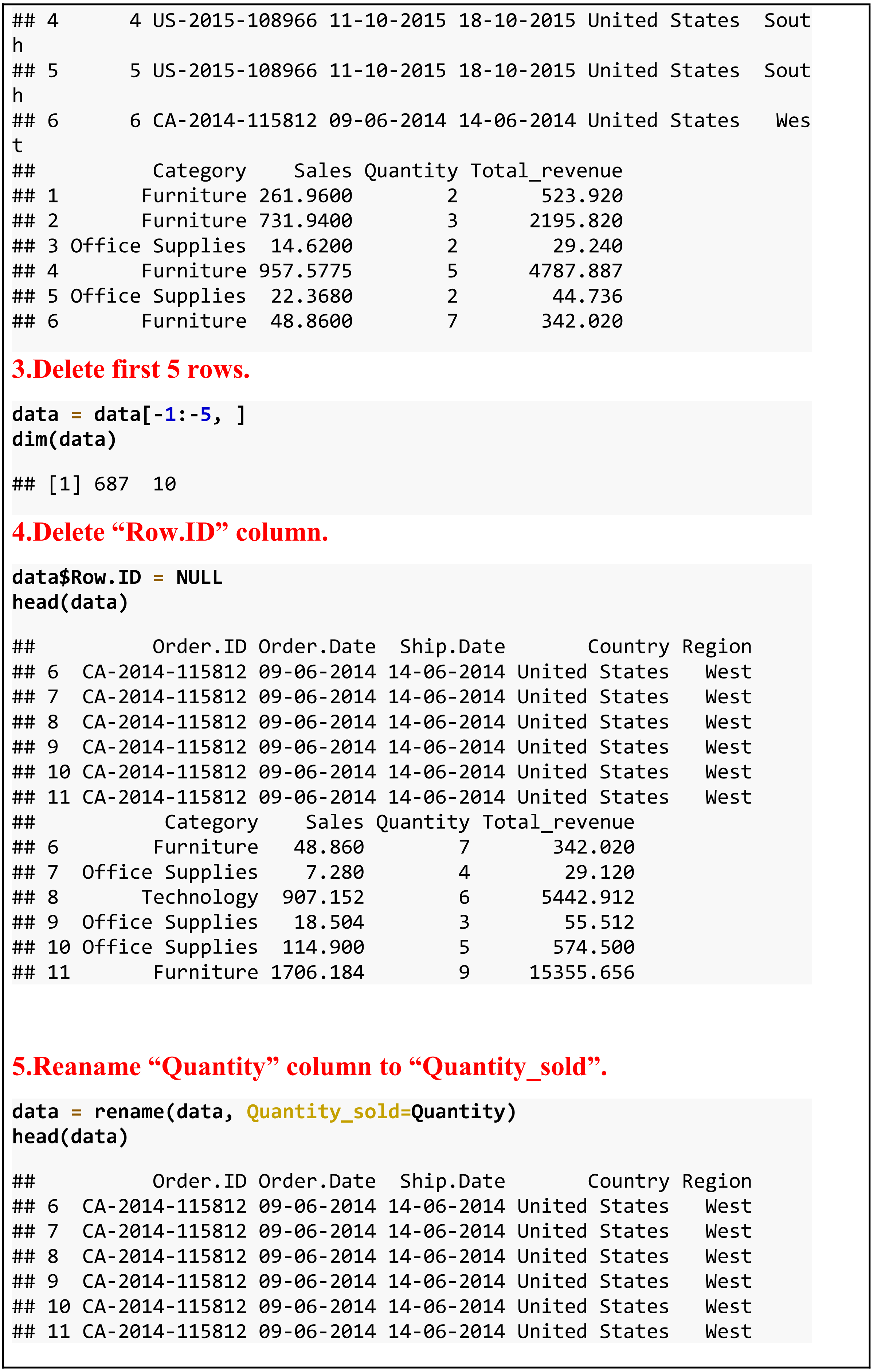
1.ADD rows

```
df <- data.frame(
  Row.ID = c(693L, 694L),
  Order.ID = c("CA-2016-789123", "US-2018-987654"),
  Order.Date = c("05-11-2015", "14-12-2016"),
  Ship.Date = c("12-11-2015", "20-12-2016"),
  Country = c("United States", "United States"),
  Region = c("West", "Central"),
  Category = c("Technology", "Furniture"),
  Sales = c(543.8, 789.6),
  Quantity = c(2L, 7L)
)
```

```
data = rbind(data,df)
head(data)

##   Row.ID       Order.ID Order.Date  Ship.Date       Country Regio
n
## 1      1 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 2      2 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 3      3 CA-2016-138688 12-06-2016 16-06-2016 United States   Wes
t
## 4      4 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 5      5 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 6      6 CA-2014-115812 09-06-2014 14-06-2014 United States   Wes
t
##          Category    Sales Quantity
## 1       Furniture 261.9600        2
## 2       Furniture 731.9400        3
## 3 Office Supplies  14.6200        2
## 4       Furniture 957.5775        5
## 5 Office Supplies  22.3680        2
## 6       Furniture  48.8600        7

dim(data)

## [1] 692   9

print(data[data$Row.ID==693, ])

##     Row.ID       Order.ID Order.Date  Ship.Date       Country Reg
ion
## 691    693 CA-2016-789123 05-11-2015 12-11-2015 United States   W
est
##       Category Sales Quantity
## 691 Technology 543.8        2
```

2.Create new column “Total_revenue” by multiplying quantity sold by the price.

```
library(dplyr)

data = mutate(data, Total_revenue=Sales*Quantity)
head(data)

##   Row.ID       Order.ID Order.Date  Ship.Date       Country Regio
n
## 1      1 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 2      2 CA-2016-152156 08-11-2016 11-11-2016 United States  Sout
h
## 3      3 CA-2016-138688 12-06-2016 16-06-2016 United States   Wes
t
```



```
## 4      4 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 5      5 US-2015-108966 11-10-2015 18-10-2015 United States  Sout
h
## 6      6 CA-2014-115812 09-06-2014 14-06-2014 United States   Wes
t
##          Category    Sales Quantity Total_revenue
## 1        Furniture 261.9600        2       523.920
## 2        Furniture 731.9400        3      2195.820
## 3 Office Supplies  14.6200        2        29.240
## 4        Furniture 957.5775        5      4787.887
## 5 Office Supplies  22.3680        2        44.736
## 6        Furniture  48.8600        7       342.020
```

3.Delete first 5 rows.

```
data = data[-1:-5, ]
dim(data)

## [1] 687  10
```

4.Delete “Row.ID” column.

```
data$Row.ID = NULL
head(data)

##          Order.ID Order.Date  Ship.Date       Country Region
## 6  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 7  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 8  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 9  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 10 CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 11 CA-2014-115812 09-06-2014 14-06-2014 United States   West
##            Category    Sales Quantity Total_revenue
## 6         Furniture   48.860        7       342.020
## 7  Office Supplies    7.280        4        29.120
## 8        Technology  907.152        6      5442.912
## 9  Office Supplies   18.504        3        55.512
## 10 Office Supplies  114.900        5       574.500
## 11        Furniture 1706.184        9     15355.656
```

5.Reaname “Quantity” column to “Quantity_sold”.

```
data = rename(data, Quantity_sold=Quantity)
head(data)

##          Order.ID Order.Date  Ship.Date       Country Region
## 6  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 7  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 8  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 9  CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 10 CA-2014-115812 09-06-2014 14-06-2014 United States   West
## 11 CA-2014-115812 09-06-2014 14-06-2014 United States   West
##            Category    Sales Quantity_sold Total_revenue
## 6         Furniture   48.860             7       342.020
## 7  Office Supplies    7.280             4        29.120
## 8        Technology  907.152             6      5442.912
## 9  Office Supplies   18.504             3        55.512
## 10 Office Supplies  114.900             5       574.500
## 11       Furniture 1706.184             9     15355.656
```

6.Create new columns for “Order_day”,“Order_month” and “Order_year” from “Order.Date”.

```
library(tidyr)
data = data %>% separate(Order.Date, into=c("Order_day","Order_month
","Order_year"), sep='-')
head(data)
```

```
##            Order.ID Order_day Order_month Order_year  Ship.Date## 6
  CA-2014-115812        09          06       2014 14-06-2014
## 7  CA-2014-115812        09          06       2014 14-06-2014
## 8  CA-2014-115812        09          06       2014 14-06-2014
## 9  CA-2014-115812        09          06       2014 14-06-2014
## 10 CA-2014-115812        09          06       2014 14-06-2014
## 11 CA-2014-115812        09          06       2014 14-06-2014
##            Country Region        Category    Sales Quantity_sold
## 6  United States   West       Furniture   48.860             7
## 7  United States   West Office Supplies    7.280             4
## 8  United States   West      Technology  907.152             6
## 9  United States   West Office Supplies   18.504             3
## 10 United States   West Office Supplies  114.900             5
## 11 United States   West       Furniture 1706.184             9
##    Total_revenue
## 6        342.020
## 7         29.120
## 8       5442.912
## 9         55.512
## 10       574.500
## 11     15355.656
```

7.Add +2 to “Quantity” variable of South Region.

```
head(data[data$Region=="South", ])
```

```
##            Order.ID Order_day Order_month Order_year  Ship.Date
## 13 CA-2017-114412        15          04       2017 20-04-2017
## 44 CA-2017-139619        19          09       2017 23-09-2017
## 70 CA-2016-119823        04          06       2016 06-06-2016
## 73 US-2015-134026        26          04       2015 02-05-2015
## 74 US-2015-134026        26          04       2015 02-05-2015
## 75 US-2015-134026        26          04       2015 02-05-2015
##            Country Region        Category   Sales Quantity_sold
## 13 United States  South Office Supplies  15.552             3
## 44 United States  South Office Supplies  95.616             2
## 70 United States  South Office Supplies  75.880             2
## 73 United States  South       Furniture 831.936             8
## 74 United States  South       Furniture  97.040             2
## 75 United States  South Office Supplies  72.784             1
##    Total_revenue
## 13        46.656
## 44       191.232
## 70       151.760
## 73      6655.488
## 74       194.080
## 75        72.784
```

```
data$Quantity_sold[data$Region == "South"] <- data$Quantity_sold[dat
a$Region == "South"] + 2
head(data[data$Region=="South", ])
```

```
##           Order.ID Order_day Order_month Order_year  Ship.Date
## 13 CA-2017-114412        15          04       2017 20-04-2017
## 44 CA-2017-139619        19          09       2017 23-09-2017
## 70 CA-2016-119823        04          06       2016 06-06-2016
## 73 US-2015-134026        26          04       2015 02-05-2015
## 74 US-2015-134026        26          04       2015 02-05-2015
## 75 US-2015-134026        26          04       2015 02-05-2015
##          Country Region        Category   Sales Quantity_sold
## 13 United States  South Office Supplies  15.552             5
## 44 United States  South Office Supplies  95.616             4
## 70 United States  South Office Supplies  75.880             4
## 73 United States  South       Furniture 831.936            10
## 74 United States  South       Furniture  97.040             4
## 75 United States  South Office Supplies  72.784             3
##    Total_revenue
## 13        46.656
## 44       191.232
## 70       151.760
## 73      6655.488
## 74       194.080
## 75        72.784
```

WEEK-2

Perform below data pre-processing techniques on the sales dataset.

1. Delete Unnecessary columns
2. Handle missing values
3. Remove duplicate data
4. Create Country, Order_year and Order_Id from Order_Id variable
5. Remove outliers from sales column

```
# load sales dataset
data = read.csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/sales_data_prepro
cess.csv",fileEncoding = "UTF-8-BOM")
#examin the data
head(data)

##   Row.ID        Order.ID Order.Date  Ship.Date Region        Category
## 1      1 CA-2016-152156 08-11-2016 11-11-2016  South       Furniture
## 2      2 CA-2016-152156 08-11-2016 11-11-2016
## 3      3 CA-2016-138688 12-06-2016 16-06-2016   West Office Supplies
## 4      4 US-2015-108966 11-10-2015 18-10-2015  South       Furniture
## 5      5 US-2015-108966 11-10-2015 18-10-2015  South Office Supplies
## 6      6 CA-2014-115812 09-06-2014 14-06-2014   West
##      Sales Quantity
## 1 261.9600        2
## 2 731.9400        3
## 3  14.6200        2
## 4 957.5775        5
## 5  22.3680        2
## 6  48.8600        7
```

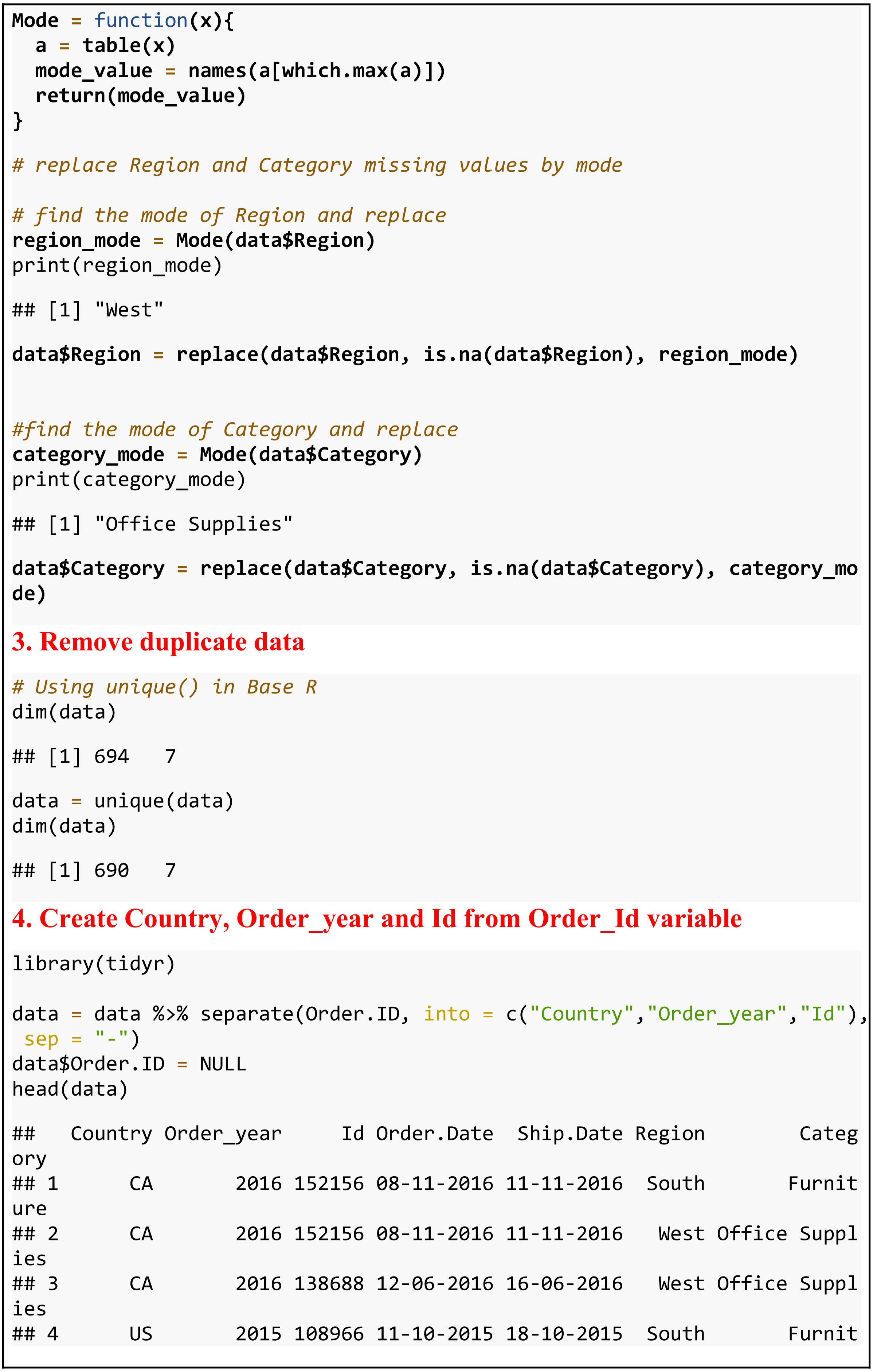
1. Delete Unnecessary columns

```
# Row.ID not required for analysis.Delete Row.ID
data$Row.ID = NULL
head(data)

##         Order.ID Order.Date  Ship.Date Region        Category    Sale
s
## 1 CA-2016-152156 08-11-2016 11-11-2016  South       Furniture 261.960
0
## 2 CA-2016-152156 08-11-2016 11-11-2016                        731.940
0
## 3 CA-2016-138688 12-06-2016 16-06-2016   West Office Supplies  14.620
0
## 4 US-2015-108966 11-10-2015 18-10-2015  South       Furniture 957.577
5
## 5 US-2015-108966 11-10-2015 18-10-2015  South Office Supplies  22.368
0
## 6 CA-2014-115812 09-06-2014 14-06-2014   West                  48.860
```

```
0
##   Quantity
## 1        2
## 2        3
## 3        2
## 4        5
## 5        2
## 6        7
```

2. Handle missing values

```
#replace blank values with NA
data[data == ""] = NA
head(data)

##          Order.ID Order.Date  Ship.Date Region        Category    Sale
s
## 1 CA-2016-152156 08-11-2016 11-11-2016  South       Furniture 261.960
0
## 2 CA-2016-152156 08-11-2016 11-11-2016   <NA>            <NA> 731.940
0
## 3 CA-2016-138688 12-06-2016 16-06-2016   West Office Supplies  14.620
0
## 4 US-2015-108966 11-10-2015 18-10-2015  South       Furniture 957.577
5
## 5 US-2015-108966 11-10-2015 18-10-2015  South Office Supplies  22.368
0
## 6 CA-2014-115812 09-06-2014 14-06-2014   West            <NA>  48.860
0
##   Quantity
## 1        2
## 2        3
## 3        2
## 4        5
## 5        2
## 6        7

# find the percentage of missing values column wise
missing_percentage = colSums(is.na(data))/nrow(data)*100
print(missing_percentage)

##   Order.ID Order.Date  Ship.Date     Region   Category      Sales
##   0.000000   0.000000   0.000000   3.890490   5.187320   1.873199
##   Quantity
##   0.000000

# replace Sales missing values by mean()

#calculate mean of sales
mean_sales = mean(data$Sales, na.rm = TRUE)

#replace by mean
data$Sales = replace(data$Sales, is.na(data$Sales), mean_sales)
```



```
Mode = function(x){
  a = table(x)
  mode_value = names(a[which.max(a)])
  return(mode_value)
}

# replace Region and Category missing values by mode

# find the mode of Region and replace
region_mode = Mode(data$Region)
print(region_mode)

## [1] "West"

data$Region = replace(data$Region, is.na(data$Region), region_mode)


#find the mode of Category and replace
category_mode = Mode(data$Category)
print(category_mode)

## [1] "Office Supplies"

data$Category = replace(data$Category, is.na(data$Category), category_mo
de)
```

3. Remove duplicate data

```
# Using unique() in Base R
dim(data)

## [1] 694   7

data = unique(data)
dim(data)

## [1] 690   7
```

4. Create Country, Order_year and Id from Order_Id variable

```
library(tidyr)

data = data %>% separate(Order.ID, into = c("Country","Order_year","Id"),
 sep = "-")
data$Order.ID = NULL
head(data)

##   Country Order_year     Id Order.Date  Ship.Date Region        Categ
ory
## 1      CA       2016 152156 08-11-2016 11-11-2016  South       Furnit
ure
## 2      CA       2016 152156 08-11-2016 11-11-2016   West Office Suppl
ies
## 3      CA       2016 138688 12-06-2016 16-06-2016   West Office Suppl
ies
## 4      US       2015 108966 11-10-2015 18-10-2015  South       Furnit
```

```
ure
## 5      US       2015 108966 11-10-2015 18-10-2015  South Office Suppl
ies
## 6      CA       2014 115812 09-06-2014 14-06-2014   West Office Suppl
ies
##      Sales Quantity
## 1 261.9600        2
## 2 731.9400        3
## 3  14.6200        2
## 4 957.5775        5
## 5  22.3680        2
## 6  48.8600        7
```

5. Remove outliers from sales column

```
dim(data)

## [1] 690   9

Q1 = quantile(data$Sales, 0.25)
Q3 = quantile(data$Sales, 0.75)

IQR = Q3-Q1

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR

outliers = data$Sales < lower_bound | data$Sales > upper_bound
print(dim(data[outliers, ]))

#remove outlier rows
data = data[!outliers, ]
dim(data)

## [1] 690   9

## [1] 80   9

## [1] 610   9
```

WEEK-3

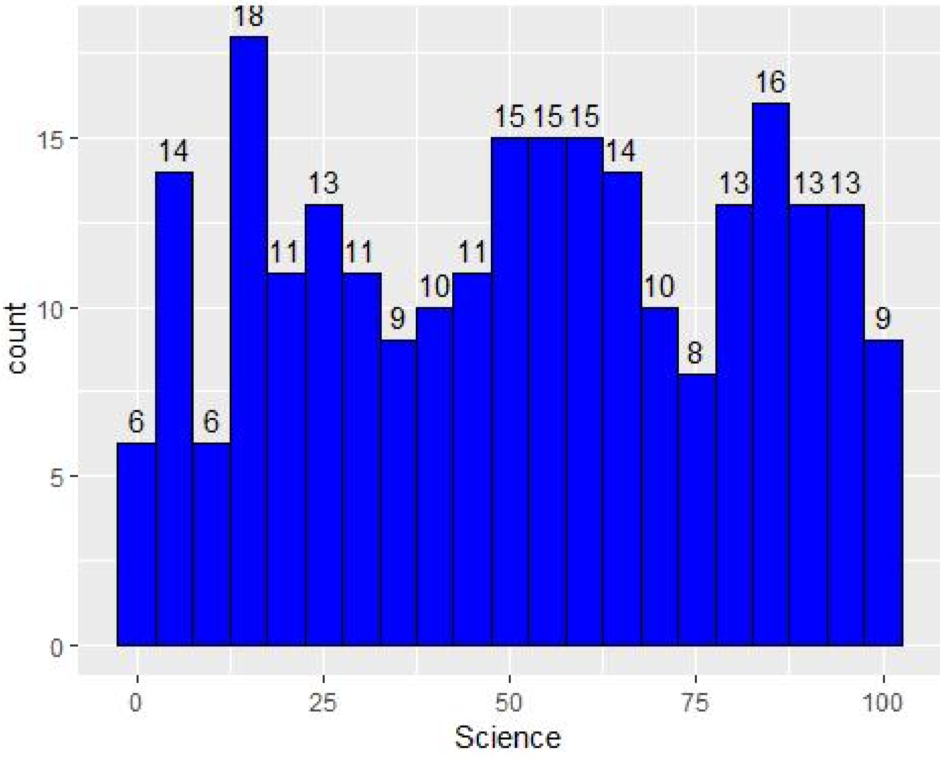
Conduct a complete data analysis on a given student results dataset and derive insights using the ggplot2 package in R.

```
# load sales dataset
data = read.csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/students_marks.cs
v",fileEncoding = "UTF-8-BOM")
#examin the data
head(data)

##   id    Name Gender Age Section Science English History Maths
## 1  1 Bronnie Female  13       C      21      81      62    49
## 2  2  Lemmie   Male  15       B      29      41      17    40
## 3  3   Danya Female  14       C      12      87      16    96
## 4  4   Denna Female  14       B      15      53      82    33
## 5  5 Jocelin   Male  14       A      43       6       3    21
## 6  6 Malissa Female  14       C      98      51      85    76

str(data)

## 'data.frame':    250 obs. of  9 variables:
##  $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Name   : Factor w/ 247 levels "Abel","Adah",..: 47 148 68 73 132 1
57 117 36 62 228 ...
##  $ Gender : Factor w/ 2 levels "Female","Male": 1 2 1 1 2 1 1 2 2 2
 ...
##  $ Age    : int  13 15 14 14 14 14 14 14 15 15 ...
##  $ Section: Factor w/ 3 levels "A","B","C": 3 2 3 2 1 3 2 2 1 3 ...
##  $ Science: int  21 29 12 15 43 98 38 25 39 35 ...
##  $ English: int  81 41 87 53 6 51 74 51 16 25 ...
##  $ History: int  62 17 16 82 3 85 54 41 22 37 ...
##  $ Maths  : int  49 40 96 33 21 76 60 80 49 27 ...

# find the percentage of missing values column wise
missing_percentage = colSums(is.na(data))/nrow(data)*100
print(missing_percentage)

##      id    Name  Gender     Age Section Science English History   Mat
hs
##       0       0       0       0       0       0       0       0
 0
```

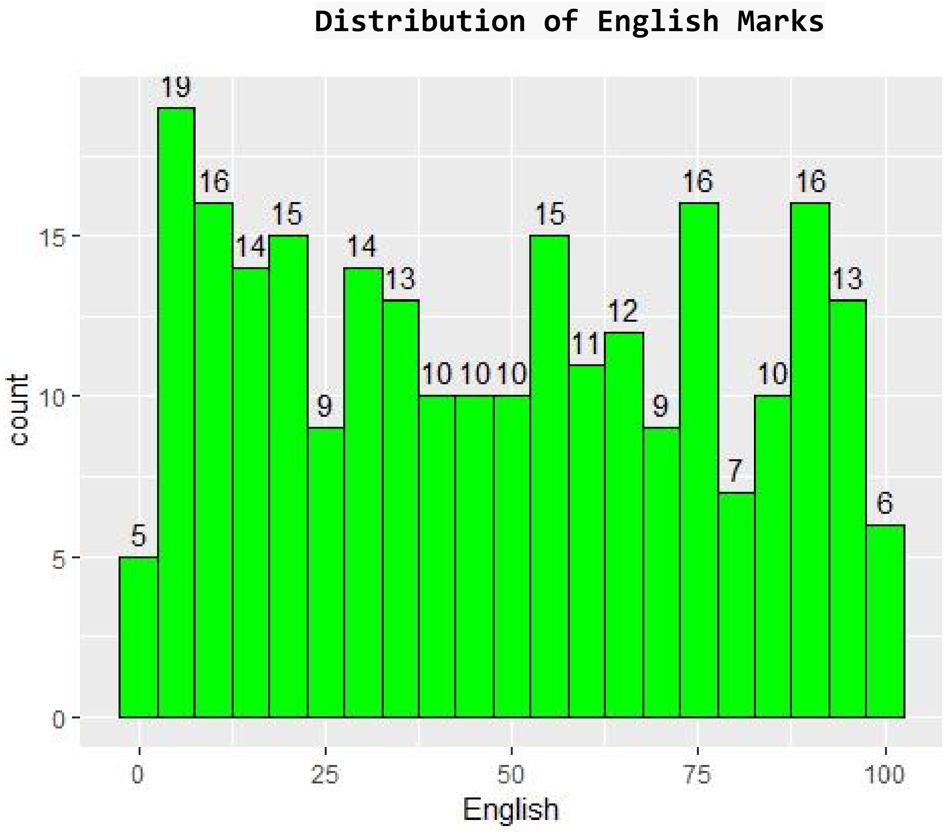
1. Distribution of Science and English Marks

```
library(ggplot2)

# Assuming 'data' is your dataframe
ggplot(data, aes(x = Science)) +
  geom_histogram(binwidth = 5, fill = 'blue', color = 'black') +
  stat_bin(binwidth = 5, geom = "text", aes(label = ..count..), vjust =
-0.5) +
  ggtitle("Distribution of Science Marks")
```

"Distribution of Science Marks"



```
ggplot(data, aes(x = English)) +
  geom_histogram(binwidth = 5, fill = 'green', color = 'black') +
  stat_bin(binwidth = 5, geom = "text", aes(label = ..count..), vjust =
-0.5) +
  ggtitle("Distribution of English Marks")
```



```
##
```

Answer the below questions from the above histogram plots:

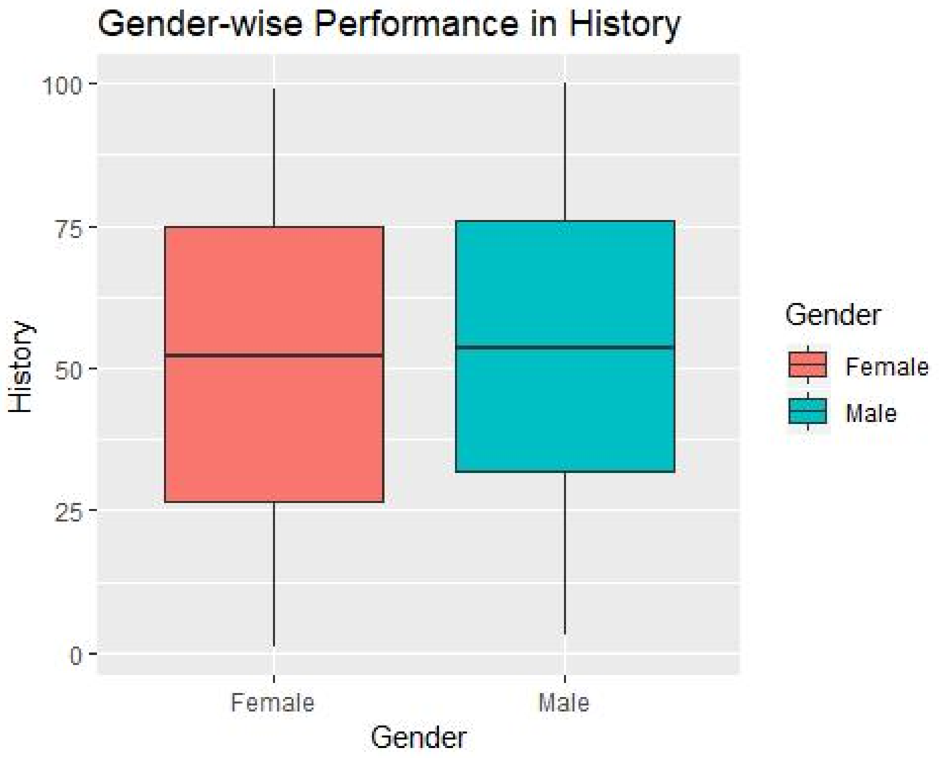
1. How many students are there with science marks > 75 (approximately)?

2. How many students are there with English marks > 75 (approximately)?

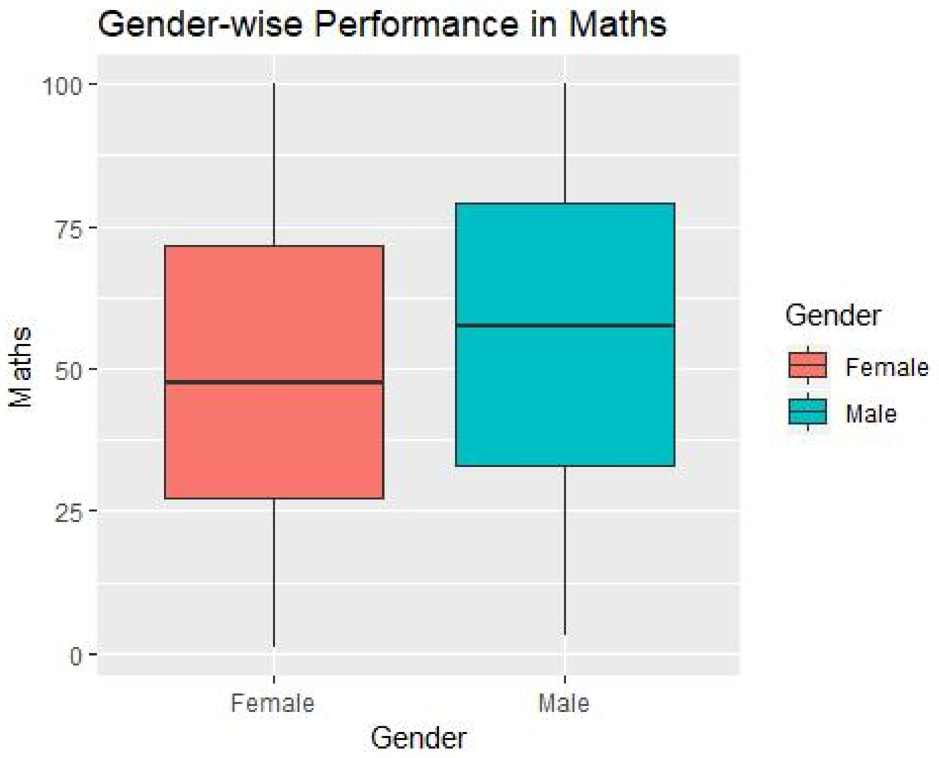
3. How many students are there with science marks < 35 (approximately)?

2. Gender-wise Performance of Maths and History marks

```
ggplot(data, aes(x = Gender, y = History, fill = Gender)) +
  geom_boxplot() +
  ggtitle("Gender-wise Performance in History")
```



```
ggplot(data, aes(x = Gender, y = Maths, fill = Gender)) +
  geom_boxplot() +
  ggtitle("Gender-wise Performance in Maths")
```



Answer the below questions from above box plots:

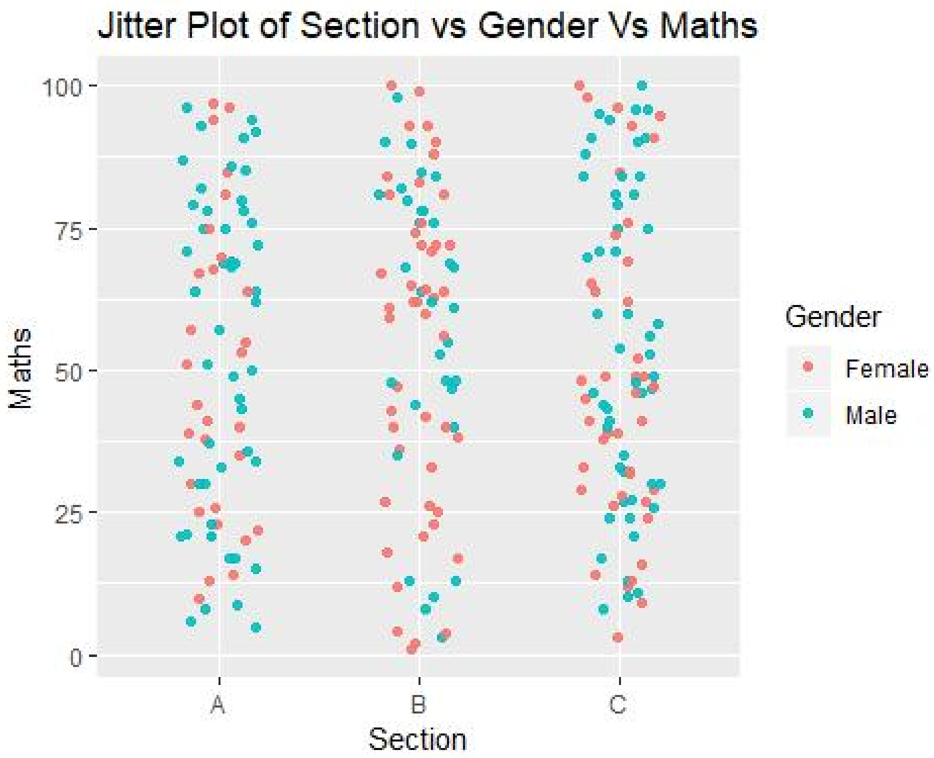
1. Which gender has the highest average math score?

2. Are there any outliers in the math marks?

3. Which gender performed well in the math exam?

3. Section and gender wise Performance of maths subject

```
ggplot(data, aes(x = Section, y = Maths, color = Gender)) +
  geom_jitter(width = 0.2, height = 0.1, alpha = 0.9) +
  labs(title = "Jitter Plot of Section vs Gender Vs Maths",
       x = "Section",
       y = "Maths")
```



Answer the below questions from above jitter plot:

1. Draw jitter plot for remaining subjects also.

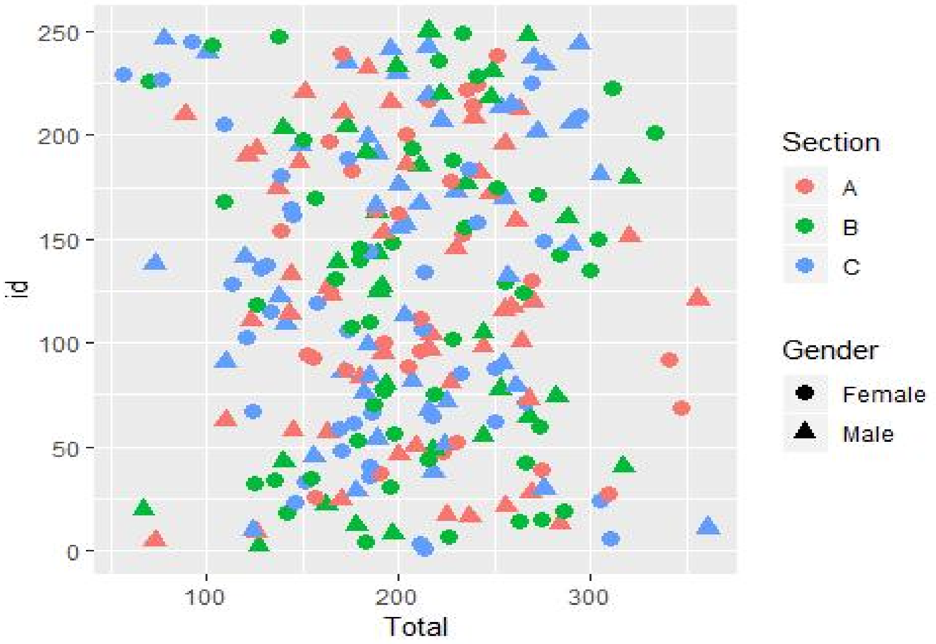
2. Which gender from what section performed well in the math,science,english and History exams?

4. Calculate total marks and analyze them with id,section and gender

```
library(dplyr)

# create total column

data = mutate(data, Total = Maths + Science + English + History)
head(data)
```

```
##   id     Name Gender Age Section Science English History Maths Total
## 1  1 Bronnie Female  13       C      21      81      62    49   213
## 2  2  Lemmie   Male  15       B      29      41      17    40   127
## 3  3   Danya Female  14       C      12      87      16    96   211
## 4  4   Denna Female  14       B      15      53      82    33   183
## 5  5 Jocelin   Male  14       A      43       6       3    21    73
## 6  6 Malissa Female  14       C      98      51      85    76   310
```

```
ggplot(data, aes(x = Total,y = id, shape = Gender, color = Section)) +
  geom_point(size = 3)
```

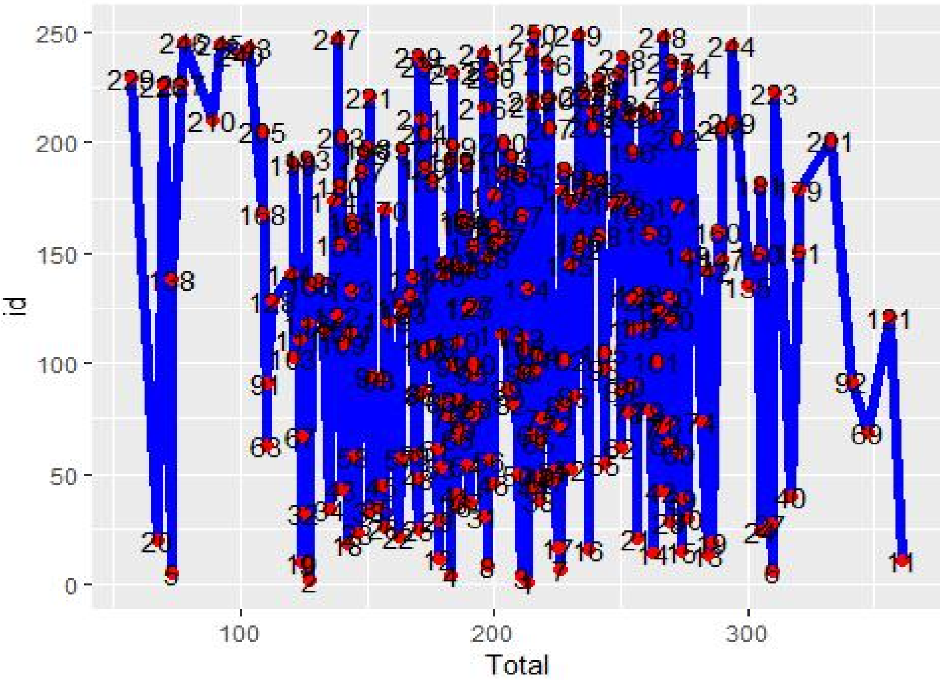


Answer the below questions from above scatter plot:

1. student from which section and gender got the highest total marks.
2. student from which section and gender got the least total marks.

5. Line plot between id and total marks

```
ggplot(data, aes(x = Total, y = id)) +
  geom_line(size = 2, color = "blue") +
  geom_point(color = "red", size = 2) +
  geom_text(aes(label=id))
```



Answer the below questions from above line plot:

What is the ID of the student who got the highest marks?
What is the ID of the student who got the least marks?

WEEK-5

Merge two Data Frames and apply various data manipulation techniques.

Merge two Data Frames

```
im
po
rt
pa
nd
as
as
pd
#
re
ad
th
e
fi
le
s


        data2 =
pd.read_csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/returnsdata.csv")
        data2.head()
```

```
        Returned        Order ID
0       Yes             CA-2017-153822
1       Yes             CA-2017-129707
2       Yes             CA-2014-152345
3       Yes             CA-2015-156440
4       Yes             US-2017-155999
```

```
# merging two dataframes using inner join
data = pd.merge(data1, data2, on='Order ID', how='inner')
data.head()
```

Different data manipulation techniques

1. Delete rows

```
# Delete 2nd and 41th rows
data = data.drop([1,40])
data.shape
```

```
(102, 15)
```

2.Delete columns 'Customer ID', 'Postal Code'.

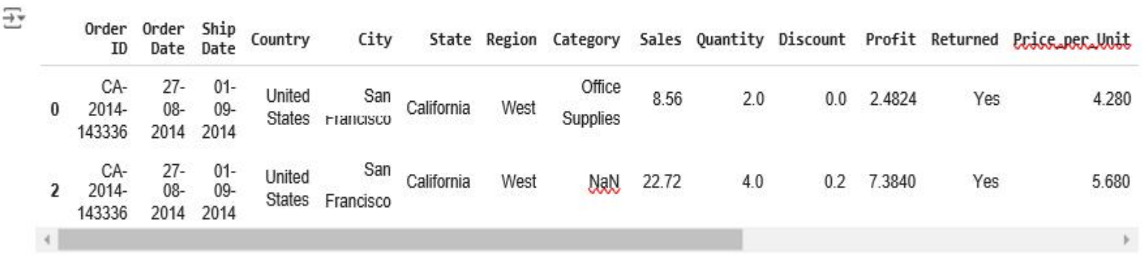
```
data = data.drop(['Customer ID', 'Postal Code'], axis=1)
data.head()
```

3. Modify the values

```
# Round the 'Profit' column to 2 decimal places
data['Profit'] = data['Profit'].round(2)
data.head()
```

4.Create new column from existing columns

```
# Create 'Price_per_Unit' column
data['Price_per_Unit'] = data['Sales'] / data['Quantity']

# Extract the year from 'Order ID'
data['OrYear'] = df['Order ID'].str.split('-').str[1]
data.head()
```



	Order ID	Order Date	Ship Date	Country	City	State	Region	Category	Sales	Quantity	Discount	Profit	Returned	Price_per_Unit
0	CA-2014-143336	27-08-2014	01-09-2014	United States	San Francisco	California	West	Office Supplies	8.56	2.0	0.0	2.4824	Yes	4.280
2	CA-2014-143336	27-08-2014	01-09-2014	United States	San Francisco	California	West	NaN	22.72	4.0	0.2	7.3840	Yes	5.680

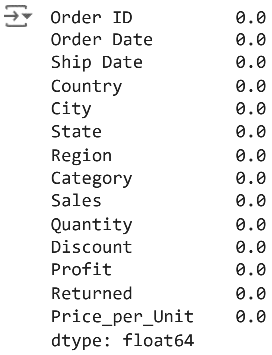
5. Handle missing data

```
import numpy as np
# replace blank
```

```
strings with 'NaN'
data =
data.replace('',np.
nan)

# calculate % of missing values columnwise
missing_percentage =
data.isna().sum()/len(data)*100
missing_percentage
```

```
Order ID        0.000000
Order Date      0.000000
Ship Date       0.000000
Country         0.000000
City            0.000000
State           0.000000
Region          0.000000
Category        6.862745

Sales           0.000000
Quantity        5.882353
Discount        0.000000
Profit          0.000000
Returned        0.000000
Price_per_Unit  5.882353
dtype: float64
```

```
# fill the missing values of Category,Quantity and Price_per_Unit
columns
data['Category'] =
data['Category'].fillna(data['Category'].mode()[0])
data['Quantity'] = data['Quantity'].fillna(data['Quantity'].mean())
data['Price_per_Unit'] =
data['Price_per_Unit'].fillna(data['Price_per_Unit'].mean())

# calculate % of missing values columnwise
missing_percentage = data.isna().sum()/len(data)*100
missing_percentage
```

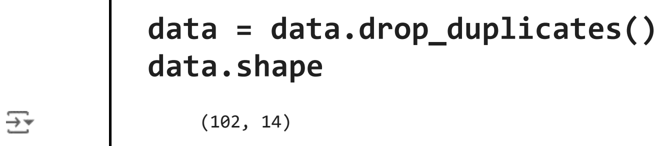


```
Order ID          0.0
Order Date        0.0
Ship Date         0.0
Country           0.0
City              0.0
State             0.0
Region            0.0
Category          0.0
Sales             0.0
Quantity          0.0
Discount          0.0
Profit            0.0
Returned          0.0
Price_per_Unit    0.0
dtype: float64
```

```
data.shape
```

```
(102, 14)
```

6. Remove duplicate entries



```
data = data.drop_duplicates()
data.shape
```

```
(102, 14)
```

No duplicates rows

WEEK-6

Use the Python ‘Matplotlib’ to perform a thorough data analysis and extract insights from a given Housing dataset.

```
import pandas as pd
# read the Housing dataset
data =
pd.read_csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/Housing.csv")

data.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishing
0	6195000	5500	3	2	4	yes	yes	no	no	yes	1	semi-fu
1	6195000	6350	3	2	3	yes	yes	no	no	yes	0	fu
2	6195000	5500	3	2	1	yes	yes	yes	no	no	2	fu
3	[illegible]	450	3	1	4	yes	no	no	no	yes	0	unfu

```
# check the
shape of dataset
data.shape
```

```
(299, 12)
```

```
data.info()
```

```
<class
'pandas.core.f
rame.DataFrame
'>RangeIndex:
299 entries, 0
   to 298
 Data columns
  (total 12
  columns):
    #   Column            Non-Null Count  Dtype
   --   ----------------  --------------  -----
    0   price             299 non-null    int64
    1   area              299 non-null    int64
    2   bedrooms          299 non-null    int64
    3   bathrooms         299 non-null    int64
    4   stories           299 non-null    int64
    5   mainroad          299 non-null    object
    6   guestroom         299 non-null    object
    7   basement          299 non-null    object
    8   hotwaterheating   299 non-null    object
    9   airconditioning   299 non-null    object
    10  parking           299 non-null    int64
    11  furnishingstatus  299 non-null    object
```

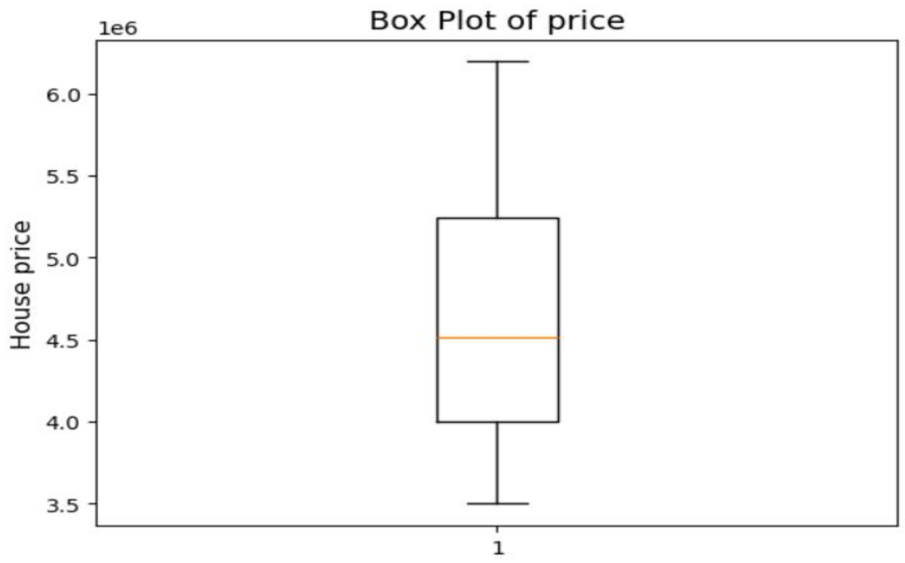
```
# check the missing
values
data.isnull().sum()
```

```
price              0
area               0
bedrooms           0
bathrooms          0
```

```
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
furnishingstatus    0
dtype: int64
```

1. Box plot for price

```
import matplotlib.pyplot as plt

# Create box plot for the 'price'
column plt.boxplot(data['price'])

# Add title and labels
plt.title('Box Plot of price',
fontsize=14) plt.ylabel('House
price', fontsize=12)

# Display the plot
 plt.show()
```



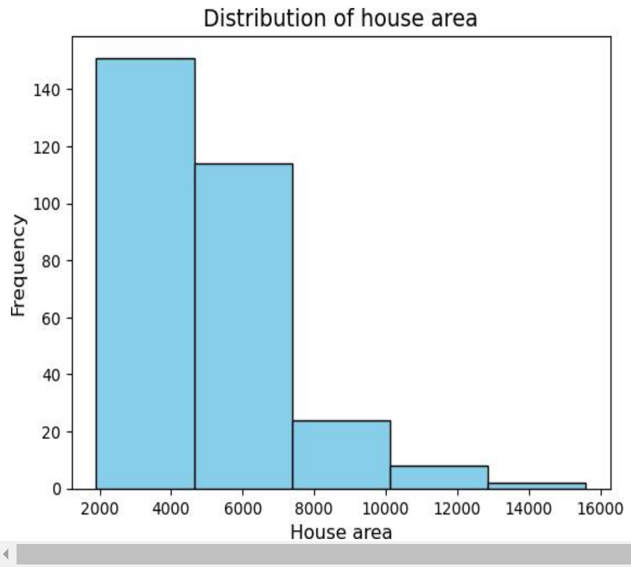
Average house price = 4500000

There are no outliers

range of house price = around 4000000 to 5400000

2. Histogram for area

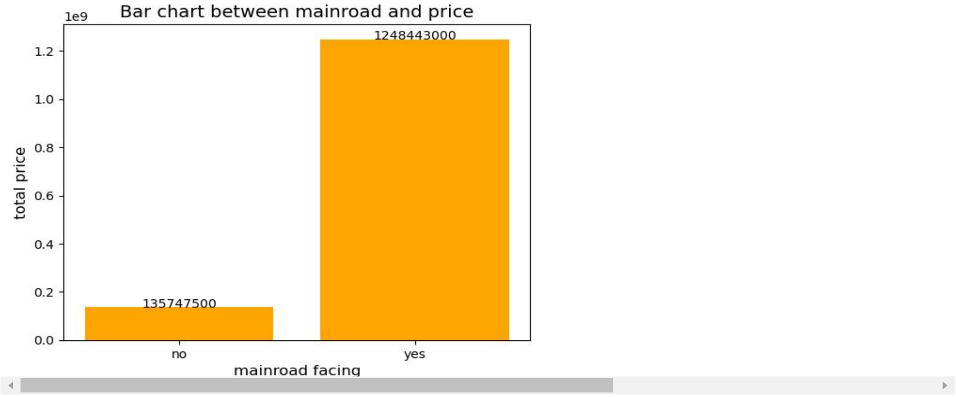
```
#Create histogram for area
plt.hist(data['area'], bins=5, edgecolor='black',
color='skyblue')

# Add labels and title
plt.title('Distribution of house area',
fontsize=14) plt.xlabel('House area',
fontsize=12)
plt.ylabel('Frequency', fontsize=12)

# Display the plot
plt.show()
```



- *most of the house area is in the range from 2000 sqft to 7200 sqft*

3. Bar chart between mainroad and price

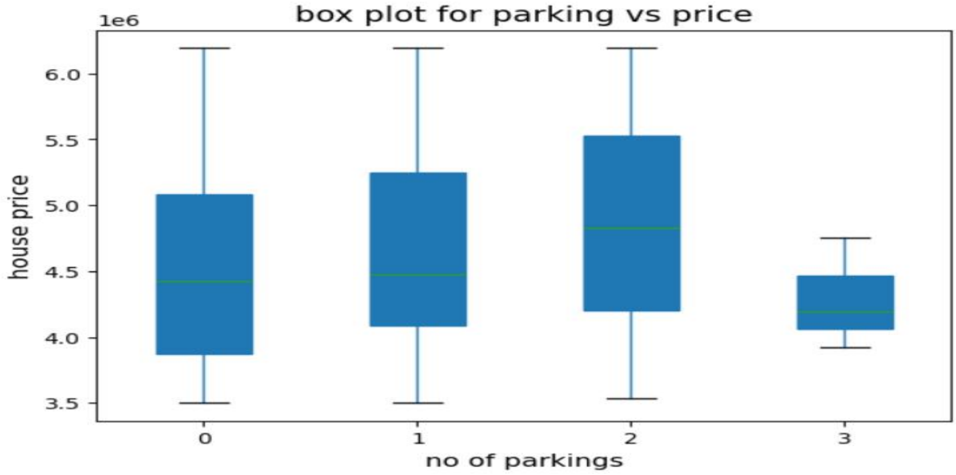
```
# Group data by 'mainroad' and sum the price
grouped_data = data.groupby('mainroad')['price'].sum()
plt.bar(grouped_data.index, grouped_data.values,
color='orange')
# Add labels and title
plt.xlabel('mainroad facing',
fontsize=12) plt.ylabel('total price',
fontsize=12)
plt.title('Bar chart between mainroad and price',
fontsize=14)
# Add data labels on top of the bars
for i, value in enumerate(grouped_data.values):
   plt.text(i, value, str(value), ha='center', fontsize=10)
# Display the
plot
plt.show()
```



- *The houses facing the main road are the most expensive.*

4. box plot for parking vs price

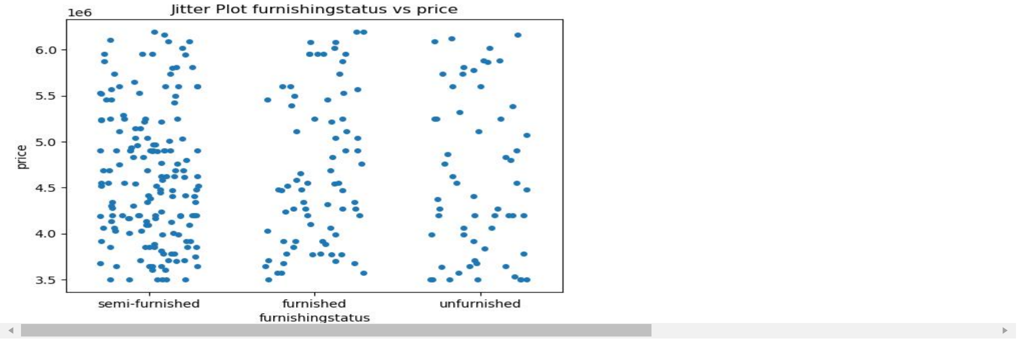
```
# Create box plot for Sales grouped by Region
data.boxplot(column='price', by='parking', grid=False,
patch_artist=True)

# Add title and labels
plt.title('box plot for parking vs price', fontsize=14)
# Remove default 'Boxplot grouped by Region'
plt.suptitle('')
plt.xlabel('no of parkings',
fontsize=12)
 plt.ylabel('house price',
fontsize=12)
# Display the plot
plt.show()
```



- *The houses with 2 parking spaces are the most expensive.*

5. jitter plot for furnishingstatus vs price

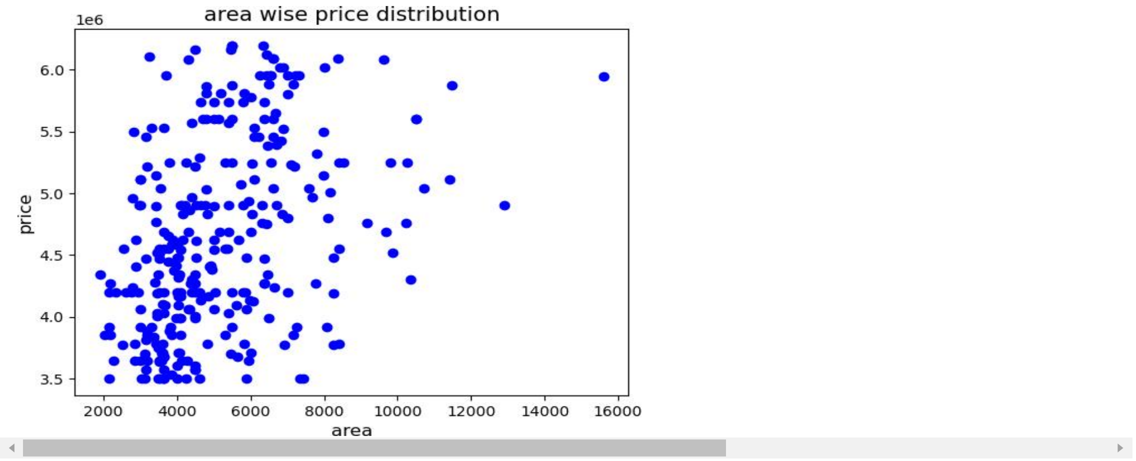
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a jitter plot Region vs Sales
sns.stripplot(x=data['furnishingstatus'], y=data['price'],
jitter=0.3)

# Add labels and title
plt.xlabel('furnishingstatus') plt.ylabel('price')
plt.title('Jitter Plot furnishingstatus vs price')

# Show the plot
plt.show()
```



No insights

6. scatter plot between area and price

```
# Create scatter plot
plt.scatter(data['area'], data['price'], color='blue')

# Add labels and title
plt.title('area wise price distribution', fontsize=14)
plt.xlabel('area', fontsize=12)
plt.ylabel('price', fontsize=12)

# Display the plot
 plt.show()
```



- *There exists a bit positive relation between area and price*

7. subplots among guestroom vs basement vs price

```
data.guestroom.unique()
```

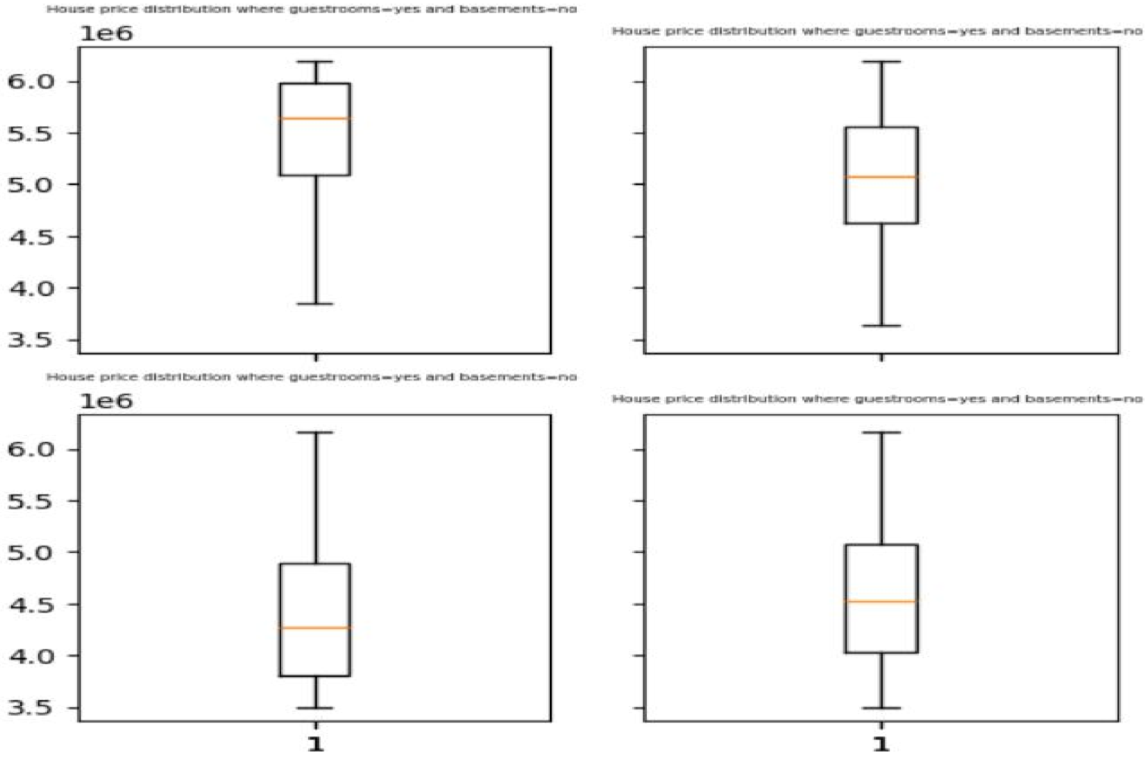
```
array(['yes', 'no'], dtype=object)
```

```
data.basement.unique()
```

```
array(['no', 'yes'], dtype=object)
```

```
import matplotlib.pyplot as plt

# Create a figure with four subplots sharing both x and y axes
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True,
figsize=(10, 10))
```

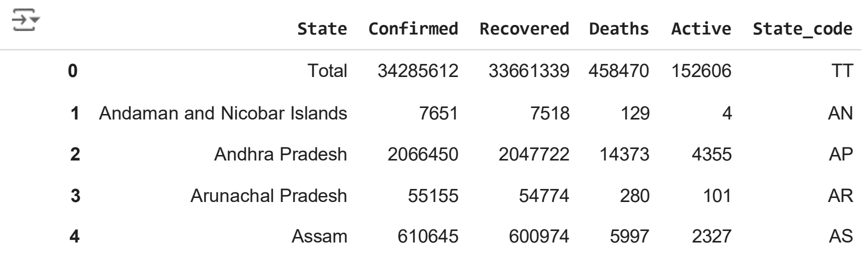
```
# Get the unique regions from the data
guestrooms = data['guestroom'].unique()
basements = data['basement'].unique()

# Plot sales by country for each region
for i, x in enumerate(guestrooms):
    for j, y in enumerate(basements):
        g_data = data[(data['guestroom'] == x) & (data['basement']
== y)]
        axes[i,j].boxplot(g_data['price'])
        axes[i][j].set_title(f'House price distribution where
                guestrooms={x} and basements={y} ',size = 8)

# Display the plots
plt.show()
```



WEEK-7

List 5 findings from the state_wise_covid dataset by making an in-depth data analysis with the help of the 'Matplotlib' library.

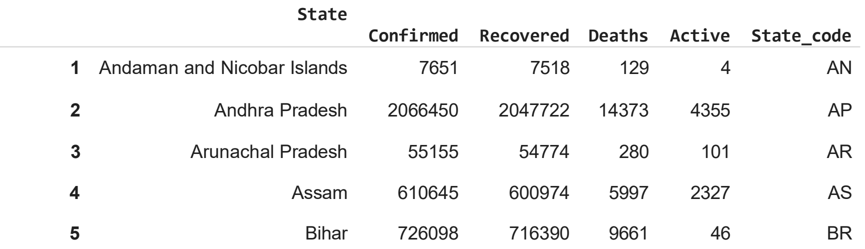
```
import pandas as pd
# read the Housing dataset
 data =
 pd.read_csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/state_data.csv")
data.head()
```



	State	Confirmed	Recovered	Deaths	Active	State_code
0	Total	34285612	33661339	458470	152606	TT
1	Andaman and Nicobar Islands	7651	7518	129	4	AN
2	Andhra Pradesh	2066450	2047722	14373	4355	AP
3	Arunachal Pradesh	55155	54774	280	101	AR
4	Assam	610645	600974	5997	2327	AS

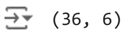
- *The first row in the dataset is the summary row, which is not required, so remove it.*
- *Row 31 has unassigned state, so remove*

```
data = data.drop([0,31])
data.head()
```



	State	Confirmed	Recovered	Deaths	Active	State_code
1	Andaman and Nicobar Islands	7651	7518	129	4	AN
2	Andhra Pradesh	2066450	2047722	14373	4355	AP
3	Arunachal Pradesh	55155	54774	280	101	AR
4	Assam	610645	600974	5997	2327	AS
5	Bihar	726098	716390	9661	46	BR

```
# check the shape of
dataset data.shape
```



```
(36, 6)
```

```
data.info()
```

```
<class
'pandas.core.fra
me.DataFrame'>
Index: 36
entries, 1 to 37
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
--   ------      ----------      -----
 0   State       36 non-null     object
 1   Confirmed   36 non-null     int64
 2   Recovered   36 non-null     int64
 3   Deaths      36 non-null     int64
 4   Active      36 non-null     int64
 5   State_code  36 non-null     object
dtype
s:
int64
(4),
objec
t(2)
memor
y
usage:
2.0+
KB
```

```
# check the missing values
data.isnull().sum()
```

```
State         0
Confirmed     0
Recovered     0
Deaths        0
Active        0
State_code    0
dtype: int64
```

1. Bar plot showing confirmed cases for each state

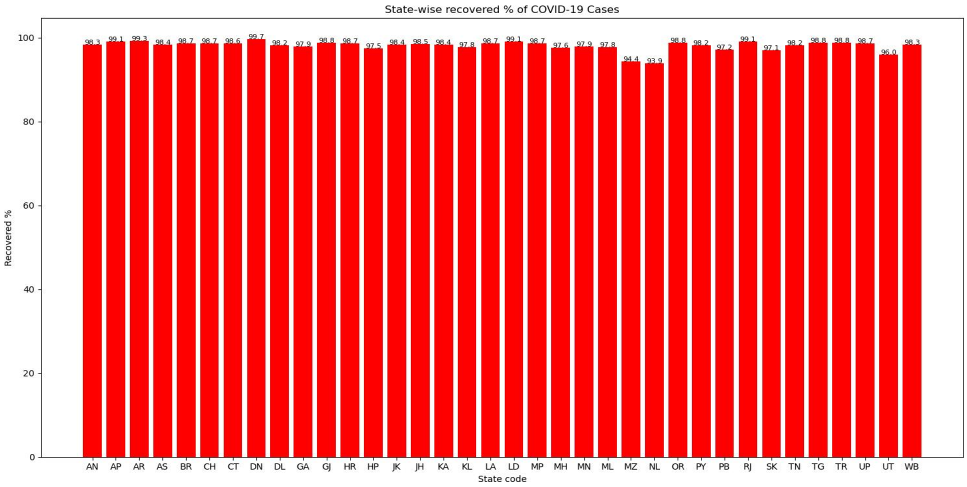
```
import matplotlib.pyplot as plt

# Plotting a bar chart for confirmed cases by state
plt.figure(figsize=(15, 8))
plt.bar(data['State_code'], data['Confirmed'], color='skyblue')
plt.xlabel('State code')
plt.ylabel('No of confirmed cases')
plt.title('State-wise Confirmed COVID-19 Cases')
plt.tight_layout()

# Add data labels on top of the bars
for i, value in enumerate(data['Confirmed']):
   plt.text(i, value, str(value), ha='center', fontsize=8)

# Display the plot
plt.show()
```

- *Maharashtra state had the highest number of confirmed COVID cases.*

- *Andaman and Nicobar Islands had the lowest number of confirmed COVID cases.*

2.Bar plot showing recovered % for each state

```
# Adding a new column for Recovery Rate
data['Recovery Rate (%)'] = (data['Recovered'] /
data['Confirmed']) * 100

# Applying the round function to the 'Recovery Rate (%)'
column and limiting it to 1 decimal place
data['Recovery Rate (%)'] = data['Recovery Rate
(%)'].round(1)

# Plotting a bar chart for confirmed cases by state
plt.figure(figsize=(15, 8))
plt.bar(data['State_code'], data['Recovery Rate (%)'],
color='red') plt.xlabel('State code')
plt.ylabel('Recovered %')
plt.title('State-wise recovered % of COVID-19 Cases')
plt.tight_layout()

# Add data labels on top of the bars
for i, value in enumerate(data['Recovery Rate (%)']):
   plt.text(i, value, str(value), ha='center', fontsize=8)

# Display the plot
plt.show()
```



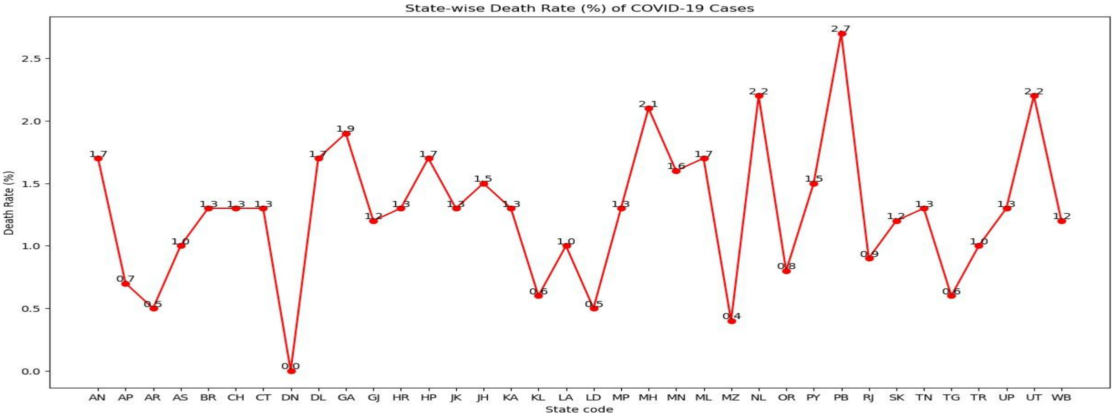
- *DN state had a highest recovered rate = 99.7%*

- *NL state had a least recovered rate = 93.9 %*

3. Line chart between State and Death %

```
# Adding a new column for Death Rate (%)
data['Death Rate (%)'] = (data['Deaths'] / data['Confirmed']) *
data['Death Rate (%)'] = data['Death Rate (%)'].round(1)

#line plot
plt.figure(figsize=(15, 8))
plt.plot(data['State_code'], data['Death Rate (%)'], marker='o',
color='red') plt.xlabel('State code')
plt.ylabel('Death Rate (%)')
plt.title('State-wise Death Rate (%) of COVID-19 Cases')
```

```
# Add data labels
for i, value in enumerate(data['Death Rate (%)']):
   plt.text(i, value, str(value), ha='center',va='bottom',
   fontsize=10)

#disp
lay
the
plot
plt.
show
()
```



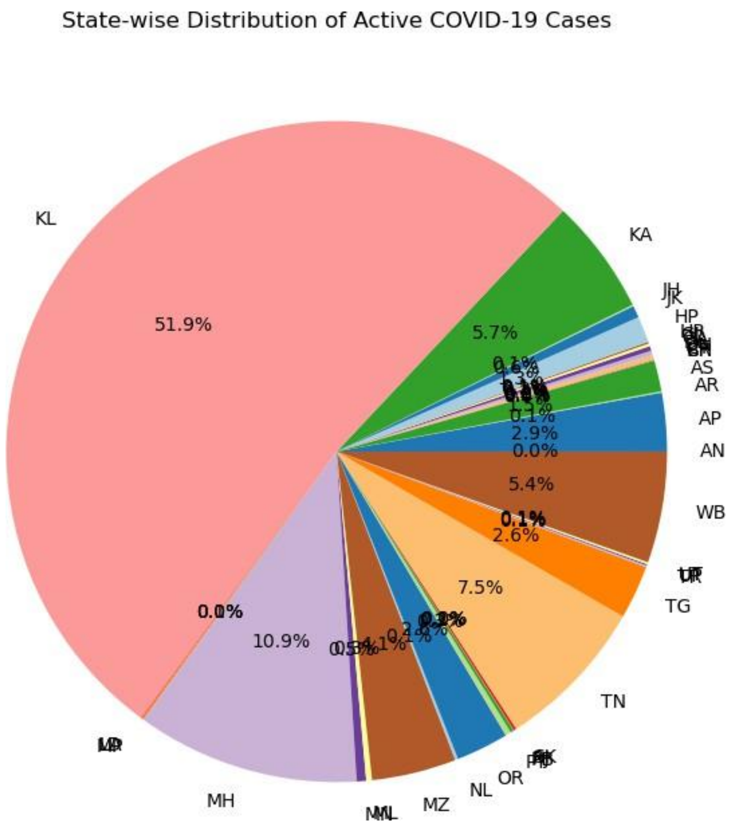
- *PB state had a highest death % = 2.7%*

- *DN state had a least death % = 0 %*

4. Pie plot showing active cases % state wise

```
# Creating a pie chart for Active Cases
across states
plt.figure(figsize=(8,8))

# Creating a pie chart for state wise active cases
plt.pie(data['Active'], labels=data['State_code'],
                            autopct='%1.1f%%',
                            colors=plt.cm.Paired.colors)

# Adding title
plt.title('State-wise Distribution of Active COVID-19 Cases')
```

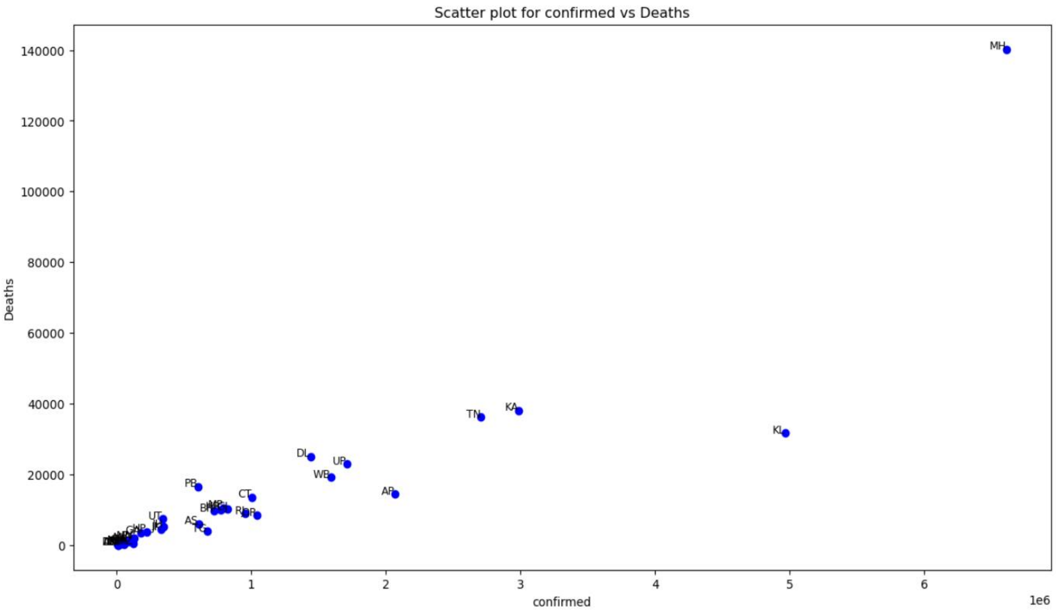
```
# Display
the pie
chart
plt.show
()
```



- *KL state had a highest active cases*

5. Scatter plot for confirmed vs recovered

```
# Create scatter plot
plt.figure(figsize=(15,8))
plt.scatter(data['Confirmed'], data['Deaths'], color='blue')

# Add labels and title
plt.xlabel('confirmed')
plt.ylabel('Deaths')
plt.title('Scatter plot for confirmed vs Deaths')

# Annotating state codes
for i in range(len(data)):
    plt.text(data['Confirmed'].iloc[i], data['Deaths'].iloc[i],
             data['State_code'].iloc[i], fontsize=9, ha='right')

# Show the plot
plt.show()
```



- *There exists a +ve relation between confirmed and deaths variables except for KL state.*
- *From plots 4 and 5 , we can derive that KL state had a less deaths but high active cases.*

Findings from the state_wise_covid dataset:

1. Maharashtra state had the highest number of confirmed COVID cases.%%.
2. Andaman and Nicobar Islands had the lowest number of confirmed COVID cases.
3. DN state had a highest recovered rate = 99.7%
4. NL state had a least recovered rate = 93.9 %
5. PB state had a highest death % = 2.7%
6. DN state had a least death % = 0 %
7. KL state had a highest active cases
8. There exists a +ve relation between confirmed and deaths variables except for KL state.

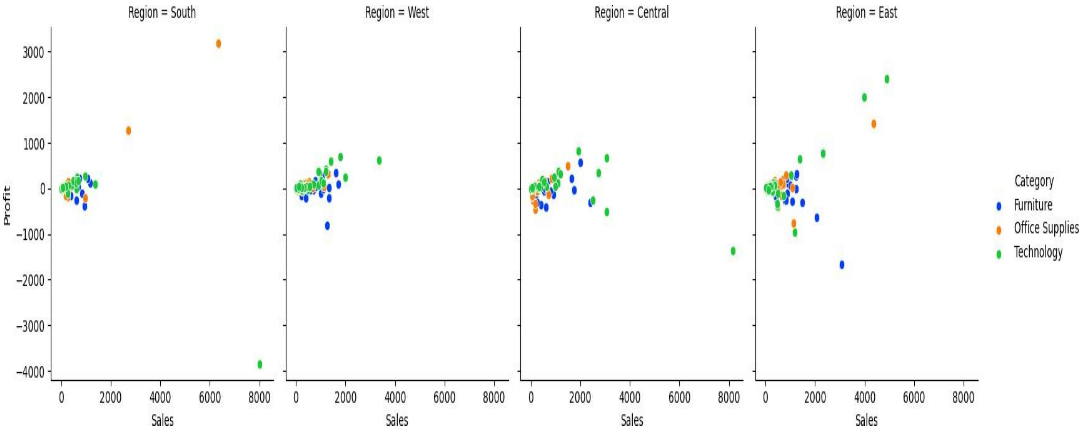
WEEK-8

Customize Heat Maps, Pair Plots, Violin plots and Joint plots with different color palettes using the Seaborn library on any dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# read the Housing dataset
data1 =
pd.read_csv("C:/Users/Dell/Desktop/MRU/DV/Datasets/salesdat
a.csv") data1.head()
```

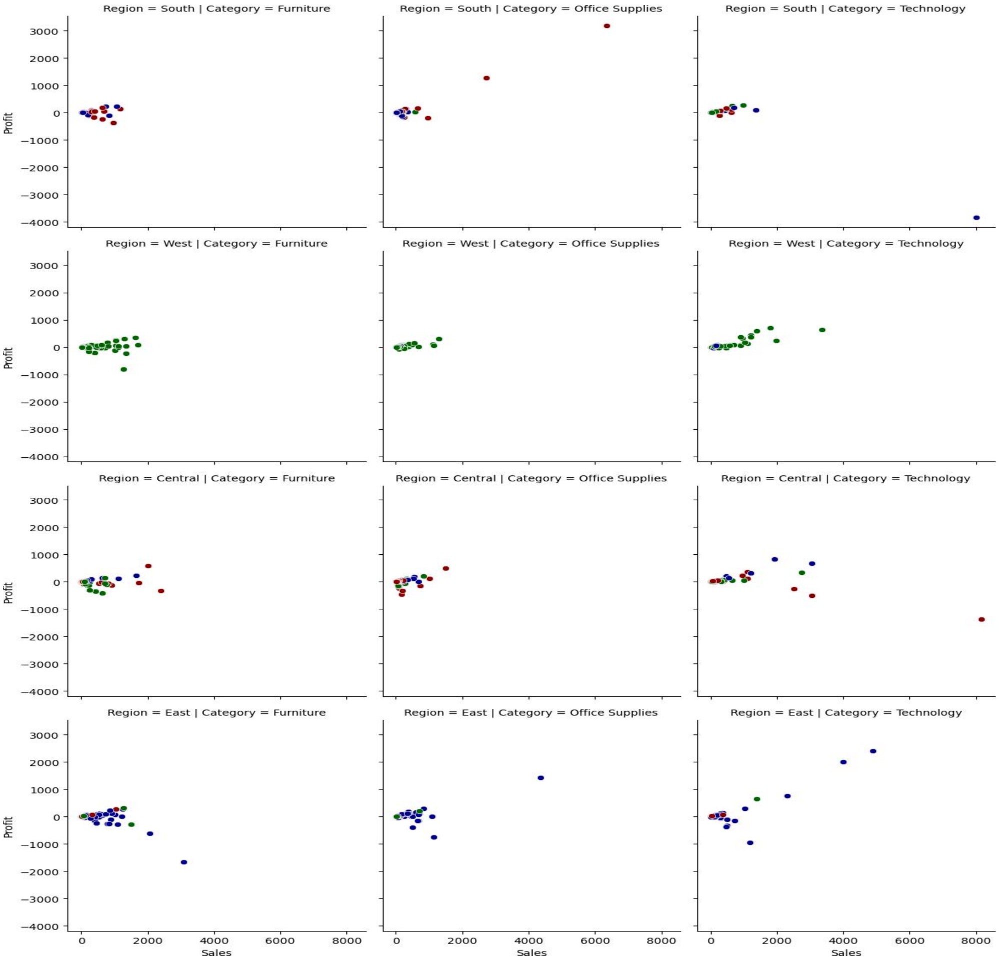
1. Subplots

```
        # Create a FacetGrid with "Region" as columns and "Category" as hue
        g = sns.FacetGrid(data1, col="Region", hue="Category", height=4,
aspect=1,

        # Map a scatter plot to each region with color based on "Category"
        g.map(sns.scatterplot, "Sales", "Profit")

        # Add a legend
        g.add_legend()

        # Display the plot
        plt.show()
```



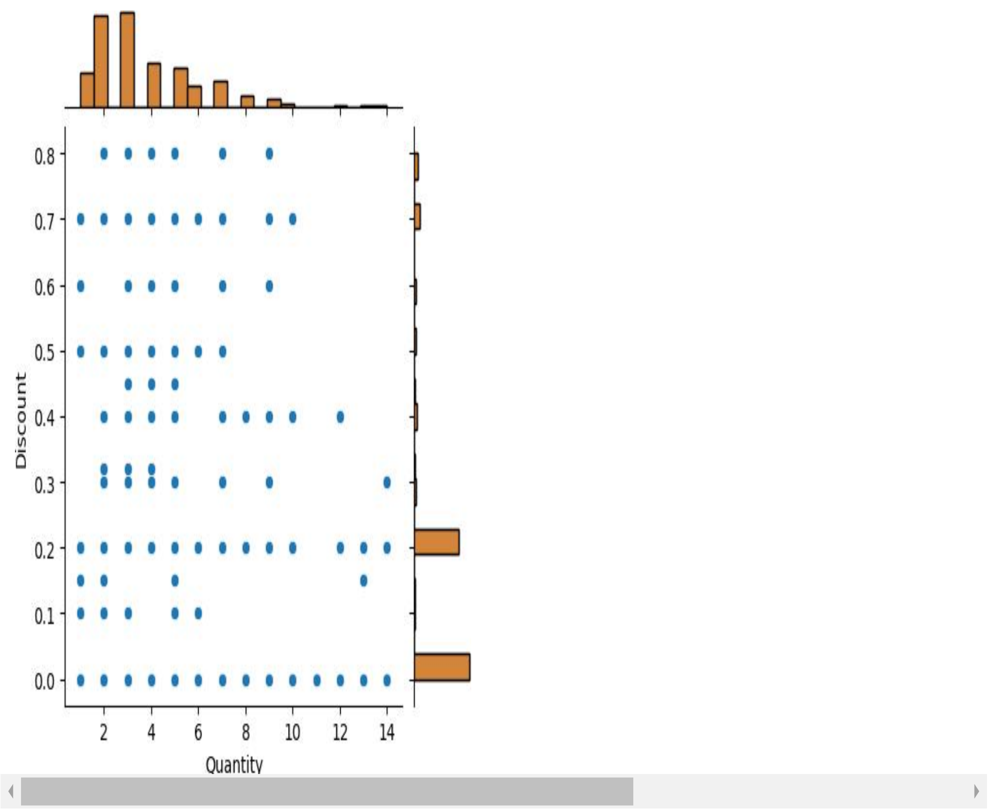
```
        # Define a custom palette with named colors
        custom_palette = sns.color_palette(["darkblue", "darkgreen",
 "darkred"])
```

```
# Create a 2D FacetGrid with rows based on "region" and columns based on
"category"
g = sns.FacetGrid(data1, row="Region", col="Category", hue = "State",
height=4, aspect=1, palette= custom_palette)

# Map a scatter plot with color based on
"State" g.map(sns.scatterplot, "Sales",
"Profit")
```



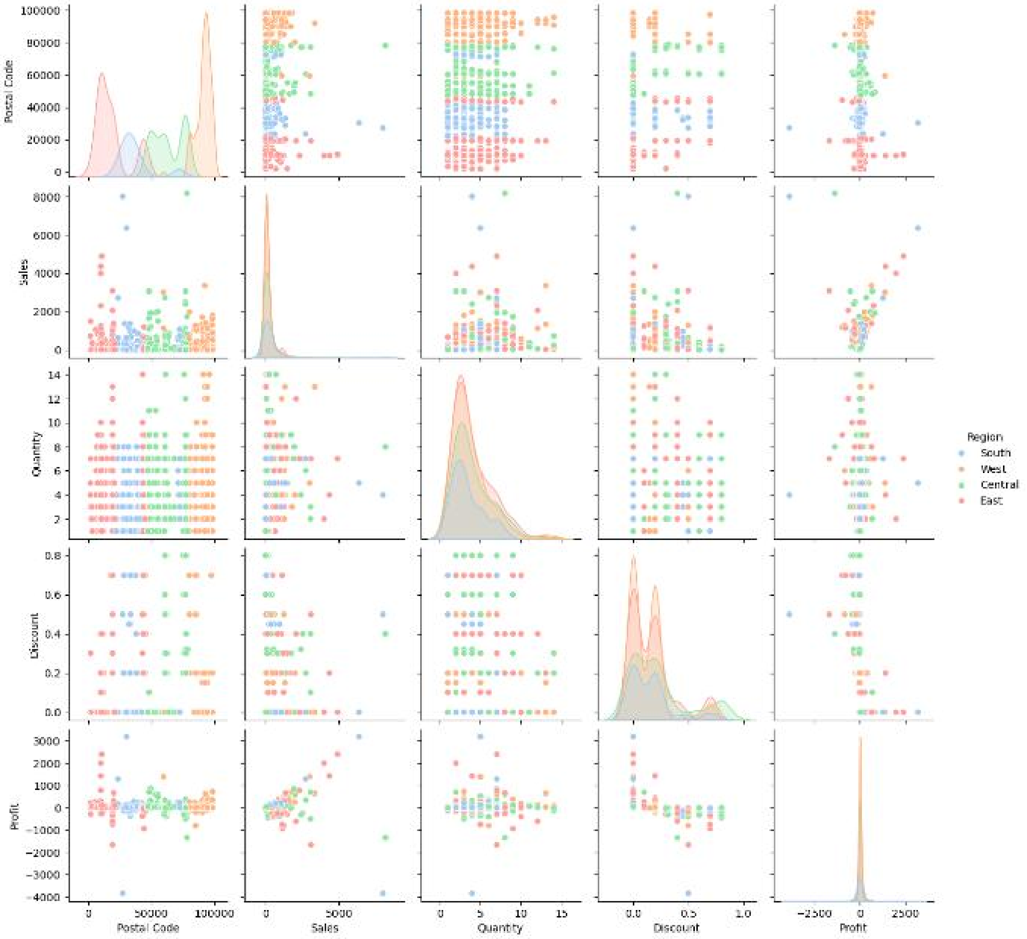
2.Joint plots

```
# Create a joint plot grid
  g = sns.JointGrid(data=data1, x="Quantity", y="Discount", height=5)
  g.plot(sns.scatterplot, sns.histplot)
  g.plot_marginals(sns.histplot)
  plt.show()
```



3. Pair Plots

```
# pair plot
sns.pairplot(data1, hue="Region", palette= 'pastel')
plt.show()
```



WEEK-9

Introduction to Tableau Desktop and Installation. Connecting to Data and preparing data for visualization in Tableau.

Steps to install Tableau Desktop:

1. **Download**:
   Go to tableau.com and download **Tableau Desktop** for your operating system.
2. **Open Installer**:
   Find the downloaded file in your **Downloads** folder and double-click it.
3. **Install**:
   Follow the prompts, accept the license agreement, and click **Install**.
4. **Activate**:
   Open Tableau Desktop, then sign in or enter your license key to activate.
5. **Start Using Tableau**:
   Once activated, you're ready to create visualizations!

Connecting to Data and preparing data:

1. Connecting to Data and Reviewing the Structure

Step 1: Connect to the data source in Tableau by selecting the appropriate file or server connection.

Step 2: Once connected, examine the **Data Source** tab. Here you can see a preview of the data and assess any structural issues like missing headers, extra columns, or incorrectly interpreted data types.

2. Renaming Fields

Renaming fields helps to clarify the purpose of each field in your dataset, making your analysis more accessible to others.

Step 1: Go to the **Data pane** in any worksheet.

Step 2: Locate the field (column) you want to rename.

Step 3: Right-click on the field name and select **Rename**.

Step 4: Type in the new name and press **Enter**.

This new name will appear across your entire workbook, making it easier for you and others to understand what the field represents.

3. Correcting Data Types

Correcting data types in Tableau is an important step in data preparation, ensuring that each field is treated correctly for analysis. Common Data Type Corrections

- Converting Text to Dates**:**
- Converting Numbers Stored as Text
- Boolean Conversion
- Handling Geographic Data
- ➢ Tableau recognizes specific geographic fields (like Country, City, Postal Code) and treats them accordingly.
- ➢ If Tableau doesn't auto-detect geographic data, right-click the field, choose

Geographic Role, and select the appropriate role.

4. Changing Data Types in the Data Pane:

- In the **Data pane** on any worksheet, locate the field you want to adjust.
- Right-click the field name, select **Change Data Type**, and choose the appropriate type from the list (e.g., String, Number, Date, Date & Time, Boolean).

5. Split strings into multiple parts

In Tableau, you can split strings into multiple parts, which is useful when you have data in a single field that needs to be separated into distinct parts.

Steps to splitting "Order ID" into Country, Year and Order ID

Step 1: In the **Data pane**, locate the field you want to split.

Step 2: Right-click the field and select **transform** -> **Split**.

Step 3: Tableau will automatically split the field based on common delimiters and create new fields for each.

Step4: Now rename each field as **Country, Year and Order ID**

6. Replace Values

In Tableau, replacing values is a useful feature to clean and standardize your data. This can be done through **aliases** for categorical fields.

Step 1: Locate the field in the **Data pane** (usually a dimension).

Step 2: Right-click on the field and select **Aliases**.

Step 3: In the **Edit Aliases** dialog, each unique value will appear in a list under **Value**.

Step 4: Click on the cell under **Alias** for any value you want to change and type in the new name.

Step 5: Click **OK** when done.

7. Creating new columns Using Calculated Fields

Calculated Fields: These allow you to create new fields based on custom formulas, which is helpful for data cleaning and transformation.

Create a "Price per Unit" column in Tableau using calculated fields:

Step1: In the Data pane, click on ▾ (arrow mark) and select Create Calculated Field.

Step2: Name the calculated field as Price per Unit.

Step3: Enter the following formula to calculate the price per unit:

```
[Sales] / [Quantity]
```

Step4: Click OK to save the calculated field. The new field Price per Unit will now appear in the Data pane**.**

Step5: you can now drag **Price per Unit** into your views or dashboards to analyze the price per unit.

8. Filtering Data

Filtering allows you to display only relevant data points in your analysis. Tableau offers several ways to filter data:

A. Filtering in the Data Source Tab

B. Filtering in Worksheets

9. creating a duplicate field

In Tableau, creating a **duplicate field** can be helpful if you want to apply different transformations or calculations to the same data field without altering the original.

Duplicating a Field in the Data Pane

Step 1: In the **Data pane** (usually on the left side of the screen), locate the field you want to duplicate.

Step 2: Right-click on the field and select **Duplicate**.

Step 3: Tableau will create a copy of the field, appending the word "(copy)" to the original field name. You can rename the duplicated field if needed.

10. Delete Columns

Step 1: In the **Data Source** tab or any worksheet, locate the column you want to remove.

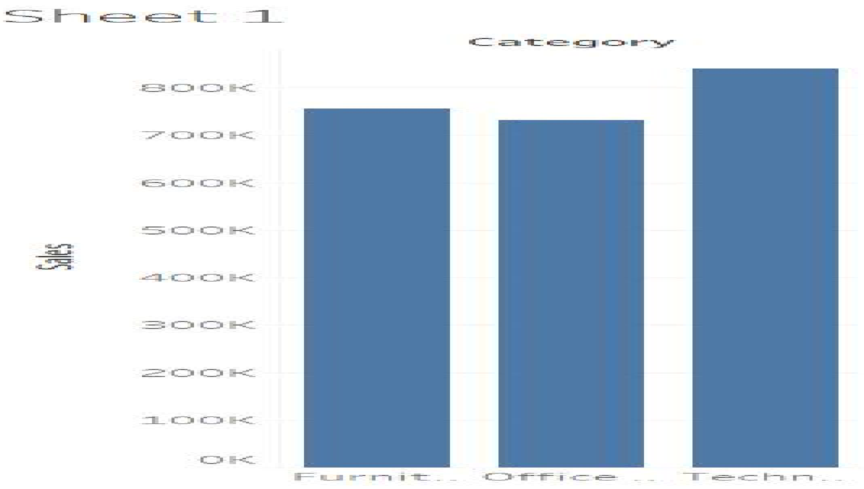
Step 2: Right-click on the field name and select **Hide** to remove it from the view.

WEEK-10

Create bar charts, line charts, tables, heat maps and tree maps on sales data in Tableau.

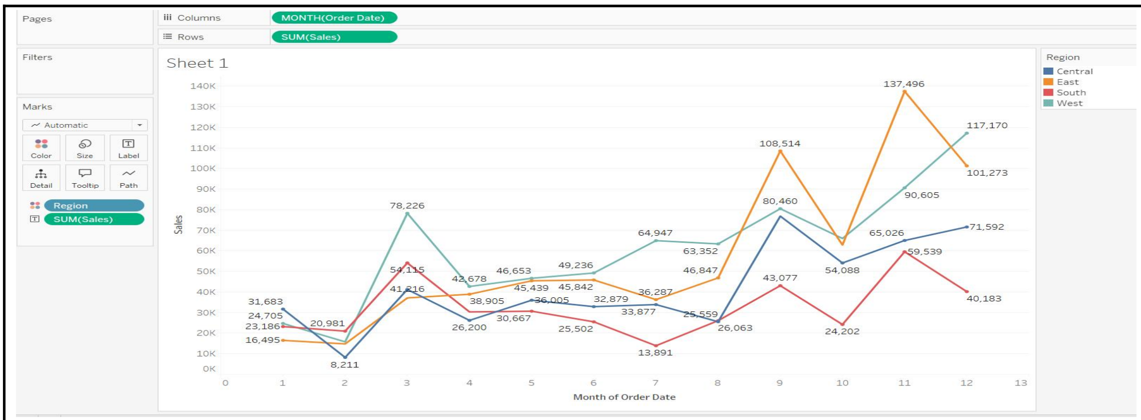
Steps to creating a bar chart in Tableau:

1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open Worksheet**: Click on a new sheet (e.g., "Sheet 1").
3. **Drag Dimension to Columns**: Drag a category **Columns** shelf.
4. **Drag Measure to Rows**: Drag a numerical field to the **Rows** shelf.
5. **Select Bar Chart**:go to **Show Me** and select **Bar Chart**.
6. **Customize (Optional)**: Sort, add labels, or adjust colors as desired.
7. **Save Your Chart**: Save or publish as needed.



Steps to creating a line chart in Tableau:

1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your line chart.
3. **Drag Date to Columns**: Drag a date field (e.g., `Order Date`) to the **Columns** shelf.
4. **Drag Measure to Rows**: Drag a measure (e.g., `Sales`, `Profit`) to the **Rows** shelf.
5. **Select Line Chart or Area Chart in Show Me**: If Tableau doesn’t automatically display a line chart or area chart, go to **Show Me** and select **Line or Area Chart**.
6. **Multi line Chart**: Add any dimension to **Detail** or **Color** in the **Marks** card for multiple lines
7. **Customize (Optional)**: Add **Sales** to **Label** in the **Marks** card to display sales values, or adjust the time granularity by right-clicking on the date field.
8. **Save Your Chart**: Save or publish your line chart.



Columns MONTH(Order Date)

Rows SUM(Sales)

Marks Automatic

Region

SUM(Sales)

Steps to create a table and customization in Tableau:

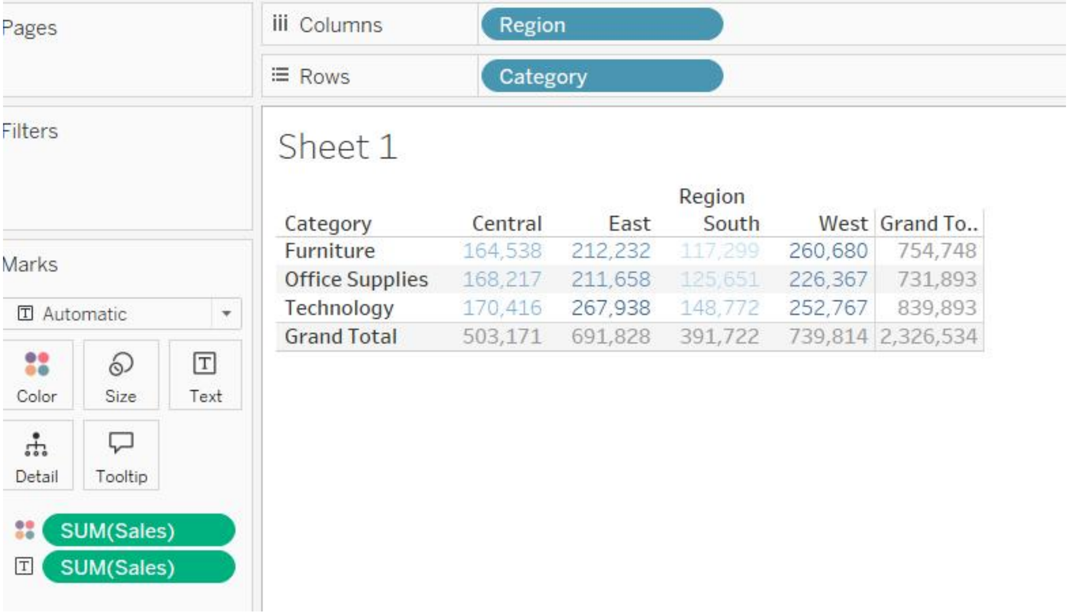
1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your table.
3. **Drag Dimensions to Rows and Columns**: Drag a **dimension** (e.g., Category, Product) to the **Rows** shelf and another **dimension** (e.g., Region, Year) to the **Columns** shelf.
4. **Drag Measure to Text**: Drag a **measure** (e.g., Sales, Profit) to the **Text** shelf in the **Marks** card. Tableau will automatically create a table showing the measure values for each row and column combination.
5. **Set Aggregation Function**:

- Right-click on the measure in the **Text** shelf of the **Marks** card.
- Select **Measure** and choose an aggregation function such as **SUM**, **AVERAGE**, **MIN**, **MAX**, or **COUNT**.

6. **Show Totals and Subtotals**:

- Go to the **Analysis** menu at the top.
- Select **Totals**, then choose **Show Row Grand Totals** and/or **Show Column Grand Totals** to add grand totals at the end of rows and columns.
- To show subtotals, go to **Analysis > Totals** and select **Add All Subtotals**. This will add subtotal rows and columns for each dimension level.

7. **Customize (Optional)**: Sort rows or columns, format text, and add any additional details as needed.



Columns: Region

Rows: Category

Automatic

SUM(Sales)

SUM(Sales)

Sheet 1

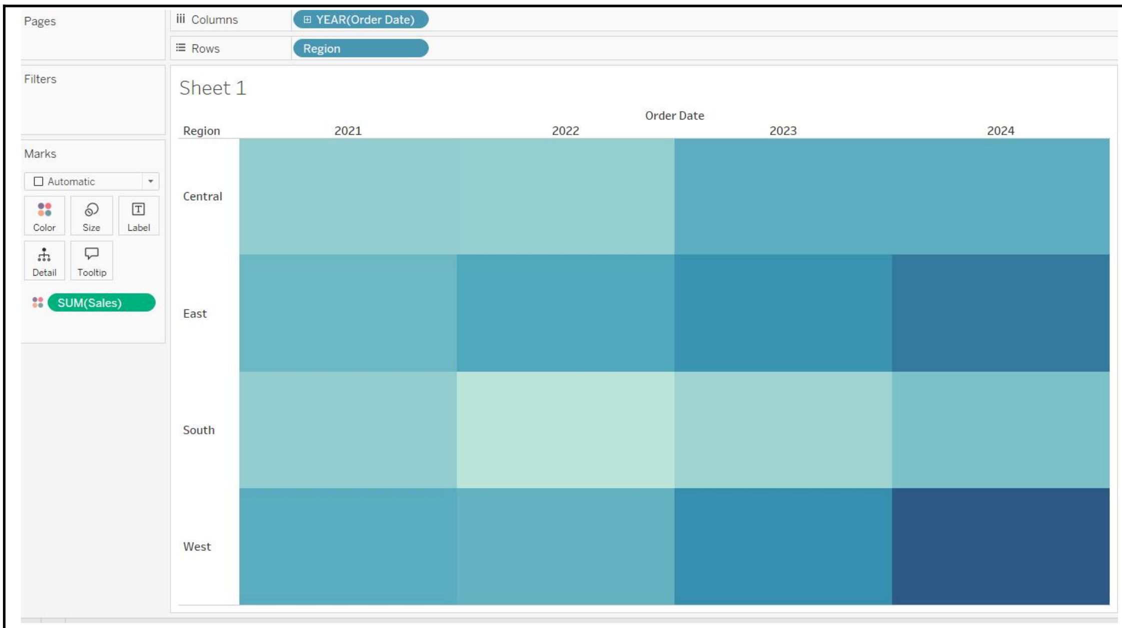
			Region		
Category	Central	East	South	West	Grand To..
Furniture	164,538	212,232	117,299	260,680	754,748
Office Supplies	168,217	211,658	125,651	226,367	731,893
Technology	170,416	267,938	148,772	252,767	839,893
Grand Total	503,171	691,828	391,722	739,814	2,326,534

Steps to create heat maps:

1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your heat map.
3. **Drag Dimensions to Rows and Columns**: Drag one dimension (e.g., Category) to **Rows** and another dimension (e.g., Region) to **Columns**. This will create a grid layout.
4. **Drag Measure to Color**: Drag a measure (e.g., Sales) to the **Color** shelf in the **Marks** card. Tableau will generate a heat map with color intensity representing the measure's values.
5. **Add Measure to Size (Optional)**: Drag the same or a different measure (e.g., Profit) to the **Size** shelf in the **Marks** card. This will adjust the size of each cell based on the measure, providing an additional layer of information.
6. **Customize Color and Size (Optional)**:

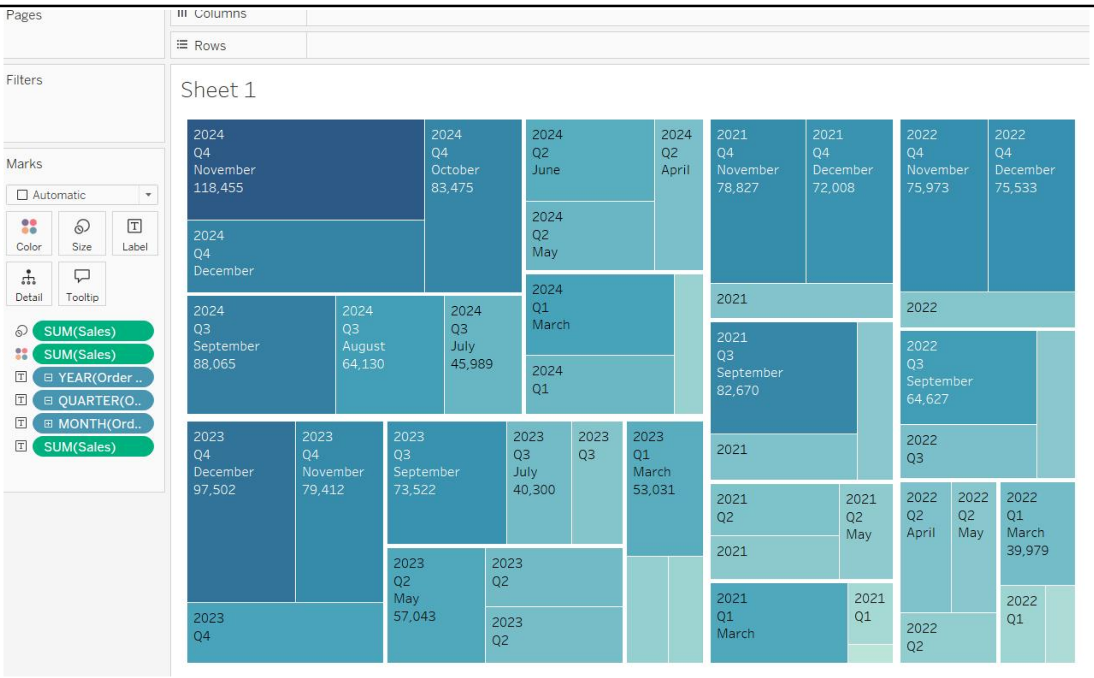
   Edit Colors: Click on **Color** in the **Marks** card to select a color palette that represents your data effectively.

   Adjust Size: Click on **Size** in the **Marks** card to modify the size scale for better readability.
7. **Save Your Heat Map**: Save or publish your heat map once you're satisfied.



Step to creating a tree map in Tableau:

1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your tree map.
3. **Drag Dimension to Rows or Columns**:
   - o Drag a dimension (e.g., Category or Region) to either the **Rows** or **Columns** shelf. This will define the main categories in your tree map.
4. **Drag Measure to Size**:
   - o Drag a measure (e.g., Sales or Profit) to the **Size** shelf in the **Marks** card. This will adjust the size of each rectangle based on the measure's value.
5. **Drag Measure to Color** (Optional):
   - o Drag the same or a different measure to the **Color** shelf in the **Marks** card to represent values with color. This step adds another layer of information by distinguishing each rectangle based on color intensity.
6. **Add Labels (Optional)**:
   - o Drag a dimension or measure (e.g., Product Name or Sales) to the **Label** shelf in the **Marks** card to display labels within each rectangle.
   - o Adjust **Label** settings to show text as desired.
7. **Select Tree map Chart in Show Me**: If Tableau doesn't automatically display a tree map chart, go to **Show Me** and select **Tree map Chart**.

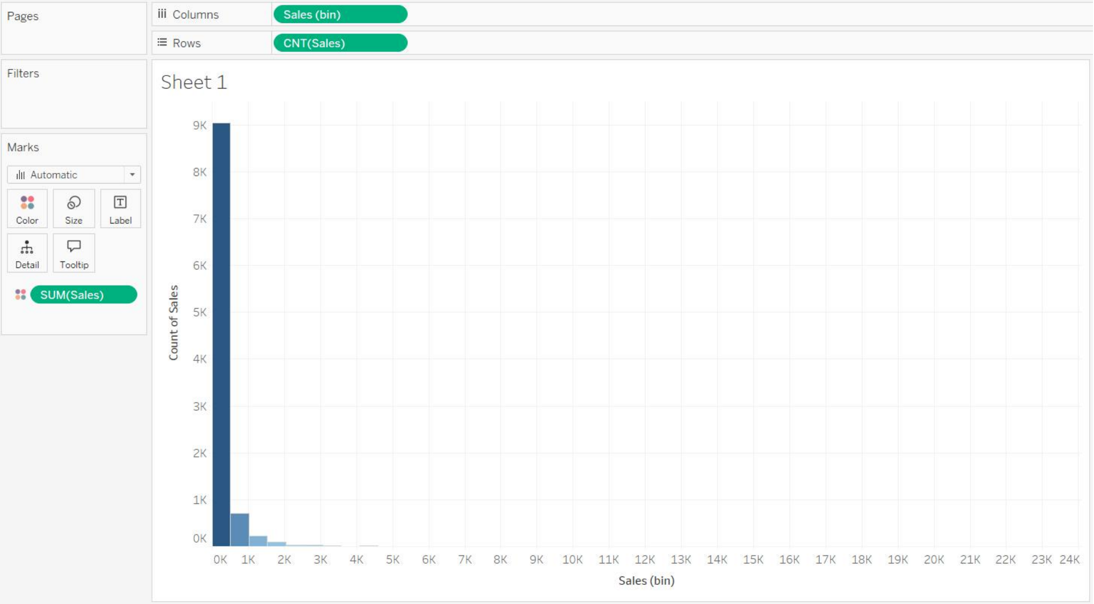

Pages
Columns
Rows
Filters
Sheet 1
Marks
Automatic
Color
Size
Label
Detail
Tooltip
SUM(Sales)
SUM(Sales)
YEAR(Order ..
QUARTER(O..
MONTH(Ord..
SUM(Sales)
2024 Q4 November 118,455
2024 Q4 October 83,475
2024 Q4 December
2024 Q2 June
2024 Q2 April
2024 Q2 May
2024 Q1 March
2024 Q1
2024 Q3 September 88,065
2024 Q3 August 64,130
2024 Q3 July 45,989
2021 Q4 November 78,827
2021 Q4 December 72,008
2021
2021 Q3 September 82,670
2021
2021 Q2
2021 Q2 May
2021
2021 Q1 March
2021 Q1
2022 Q4 November 75,973
2022 Q4 December 75,533
2022
2022 Q3 September 64,627
2022 Q3
2022 Q2 April
2022 Q2 May
2022 Q2
2022 Q1 March 39,979
2022 Q1
2023 Q4 December 97,502
2023 Q4 November 79,412
2023 Q4
2023 Q3 September 73,522
2023 Q3 July 40,300
2023 Q3
2023 Q1 March 53,031
2023 Q2 May 57,043
2023 Q2
2023 Q2

WEEK-11

Create histograms, gantt charts, pie charts and maps on sales data in Tableau.

Steps to creating a histogram in Tableau:

1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your histogram.
3. **Drag Measure to Columns**: Drag the continuous measure you want to analyze (e.g., `Sales`, `Age`, `Income`) to the **Columns** shelf.
4. **Select Histogram Chart in Show Me**
5. **Customize (Optional)**: Adjust bin size, axis labels, or colors as needed.



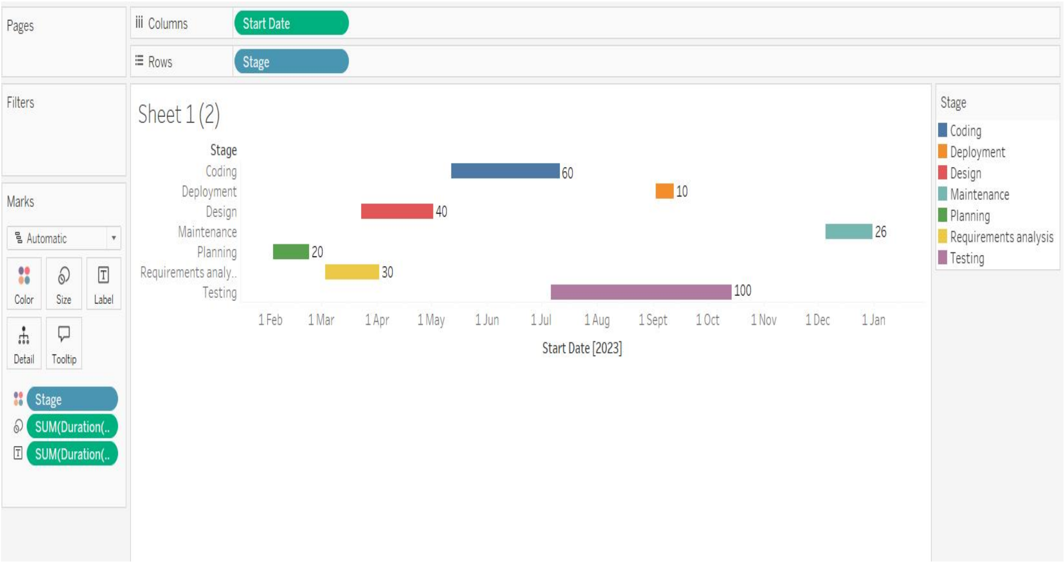
Steps to Create a Gantt Chart in Tableau

1. **Connect to Data**: Open Tableau and load your dataset, ensuring it includes columns for task names, start dates, and durations or end dates.
2. **Open a New Worksheet**: Click on a new sheet to start building your Gantt chart.
3. **Drag Task Dimension to Rows**: Drag the dimension representing tasks or activities (e.g., `Task Name` or `Stage`) to the **Rows** shelf.
4. **Drag Start Date to Columns**:

   - Drag the field representing start dates to the **Columns** shelf. This positions each task along a timeline.
   - Click on dropdown arrow mark on **Year (Start Date)** and select **Exact Date**.

5. **Select Gantt Bar from Marks Card**:

   - In the **Marks** card, select **Gantt Bar** as the mark type.
6. **Drag Duration or End Date to Size**:
   - Drag the duration or end date field to the **Size** shelf. This adjusts each bar’s length to represent the time needed to complete each task.
7. **Customize (Optional)**:
   - Add colors to indicate task categories or phases.
   - Adjust labels to display task names, start dates, or responsible team members for easier interpretation.
8. **Save Your Gantt Chart**: Save or publish your Gantt chart once you’re satisfied.



Creating a pie chart in Tableau:

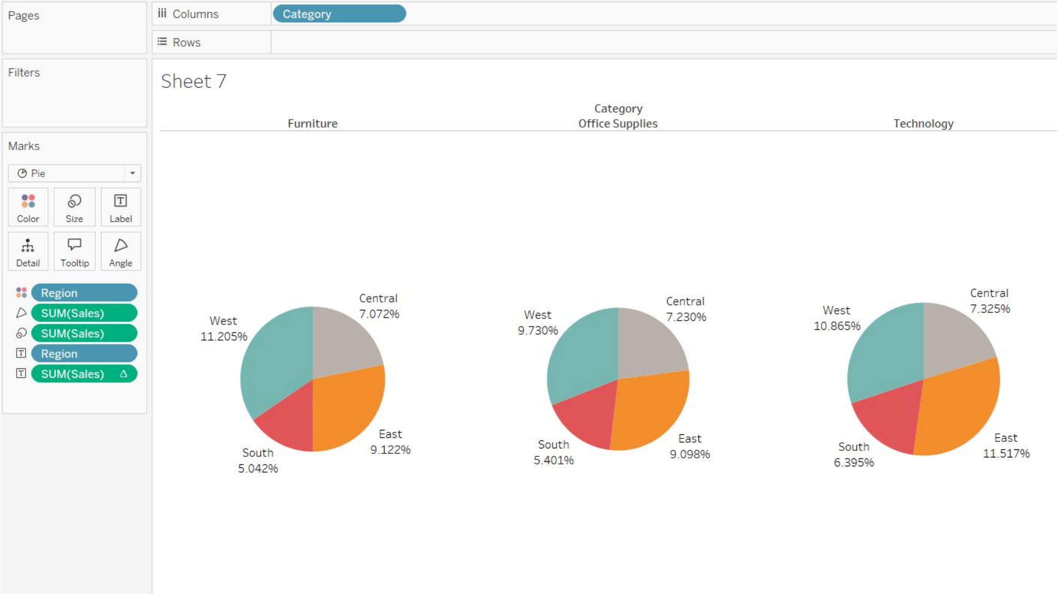
1. **Connect to Data**: Open Tableau and load your dataset.
2. **Open a New Worksheet**: Click on a new sheet to start building your pie chart.
3. **Drag Dimension to rows**: Drag the category field (e.g., `Product Category` or `Region`) to the **row** shelf.
4. **Drag Measure to columns**: Drag the measure you want to visualize (e.g., `Sales` or `Revenue`) to columns shelf.
5. **Select Pie Chart from Show Me**: In **Show Me**, click on the **Pie Chart** icon. Tableau will automatically set up a pie chart with your selected fields.
6. **Drag Dimension to Columns or Rows**: If you want multiple pie charts, drag your chosen dimension (e.g., `Region` or `Category`) to **Columns** or **Rows**. This will create a separate pie chart for each unique value in the dimension.
7. **Adjust Size and Labels**:

   - Use the **Size** slider in the **Marks** card to adjust the size of the pie charts.
   - Optionally, drag fields to **Label** to display values or percentages on each slice.

8. **Save Your Work:** Save or publish your multiple pie charts.

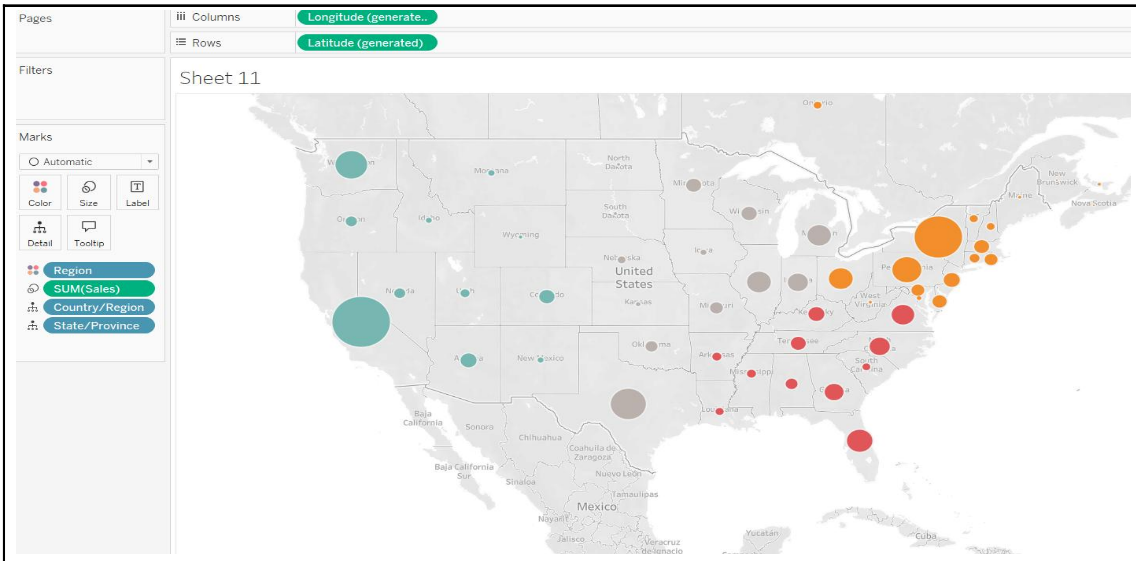
To display the % of Total Sales in each slice of a pie chart in Tableau, follow these steps:

1. In the **Marks** card, drag **Sales** to the **Label** shelf.
2. Click on the **Label** shelf (where you just placed Sales) and select **Quick Table Calculation > Percent of Total**. This will calculate each slice's percentage of the total sales.



Steps to Create a Symbol Map:

1. **Connect to Data**: Open Tableau and load your dataset, ensuring it includes geographic fields (like Country, State, City) or latitude and longitude.
2. **Drag a Geographic Field to the View**: Drag a geographic dimension (e.g., City or State) to visualization area or to **Detail** on the **Marks** card. Tableau will automatically generate a map.
3. **Add a Measure to Size**: Drag a measure (e.g., Sales or Profit) to the **Size** shelf on the **Marks** card to adjust the marker's appearance based on the measure.
4. **Customize**: Adjust the **Size** slider to modify the symbol size and add region to color.
5. **Save**: Save or publish your symbol map.



Pages

Columns Longitude (generate..

Rows Latitude (generated)

Filters

Marks

○ Automatic

Color Size Label Detail Tooltip

Region

SUM(Sales)

Country/Region

State/Province

Sheet 11

WEEK-12

Case study: Create a dashboard that gives in-depth insights into sales data with a minimum of six worksheets.

Here's a step-by-step guide to creating a **dashboard in Tableau** that includes **tables, heat maps, tree maps, line charts, pie charts, and basic maps** across 6 worksheets, culminating in a comprehensive dashboard.

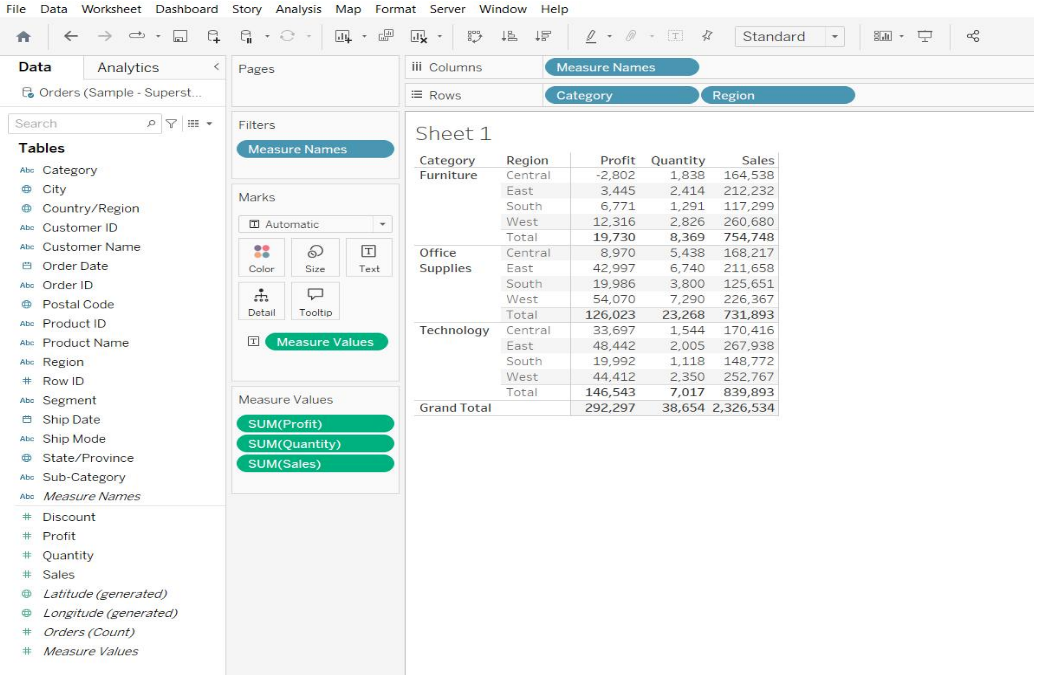
Step 1: Data Preparation

1. **Connect to Data**: Open Tableau and connect to your sales dataset (e.g., CSV, Excel).
2. **Verify Data Fields**: Ensure correct data types (e.g., Order Date as Date, Sales as numeric).
3. **Data Cleaning** (if needed): Check for any missing values or inconsistent data entries, and standardize where necessary.

Step 2: Create Individual Worksheets

Worksheet 1: Table of Key Sales Metrics

1. **Create a New Worksheet**: Click on the + icon to create a new worksheet.
2. **Add Dimensions and Measures**:
3. Drag Region or Product Category to **Rows**.
4. Drag Sales, Profit, and Quantity to **Columns** to show metrics per category.
5. Select Text Table from show me.
6. **Add Totals**:
7. Go to **Analysis > Totals > Show Row Grand Totals** to display totals for each metric.
8. **Format the Table**:
9. Adjust formatting to improve readability, such as using currency formatting for Sales and Profit.



Columns: Measure Names

Rows: Category, Region

Filters: Measure Names

Marks: Automatic

Measure Values

Measure Values: SUM(Profit), SUM(Quantity), SUM(Sales)

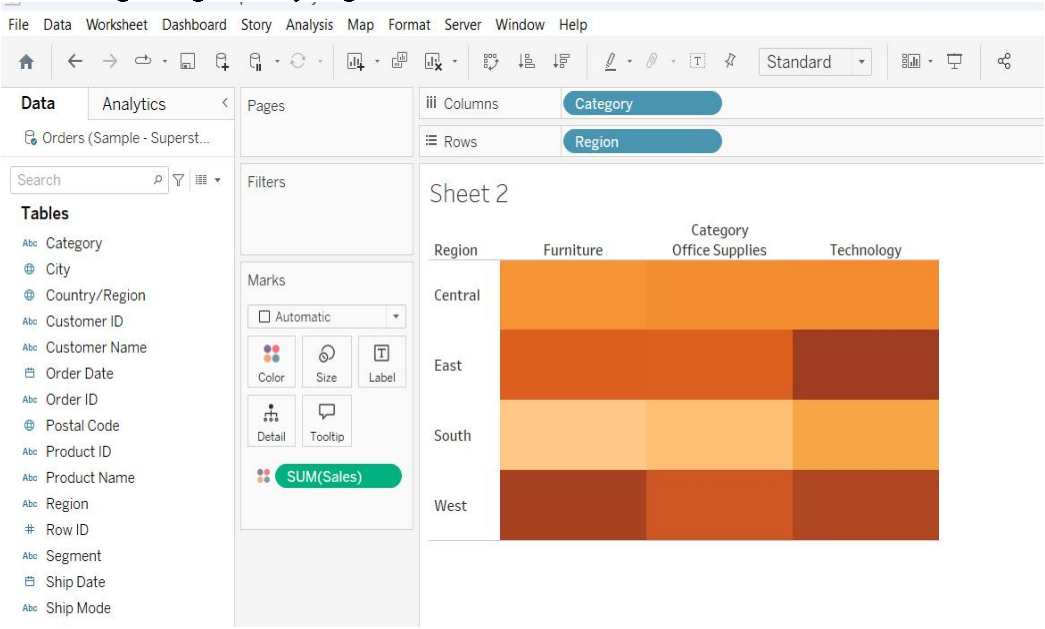
Sheet 1

Category	Region	Profit	Quantity	Sales
Furniture	Central	-2,802	1,838	164,538
	East	3,445	2,414	212,232
	South	6,771	1,291	117,299
	West	12,316	2,826	260,680
	Total	19,730	8,369	754,748
Office Supplies	Central	8,970	5,438	168,217
	East	42,997	6,740	211,658
	South	19,986	3,800	125,651
	West	54,070	7,290	226,367
	Total	126,023	23,268	731,893
Technology	Central	33,697	1,544	170,416
	East	48,442	2,005	267,938
	South	19,992	1,118	148,772
	West	44,412	2,350	252,767
	Total	146,543	7,017	839,893
Grand Total		292,297	38,654	2,326,534

Worksheet 2: Heat Map (Sales by Region and Category)

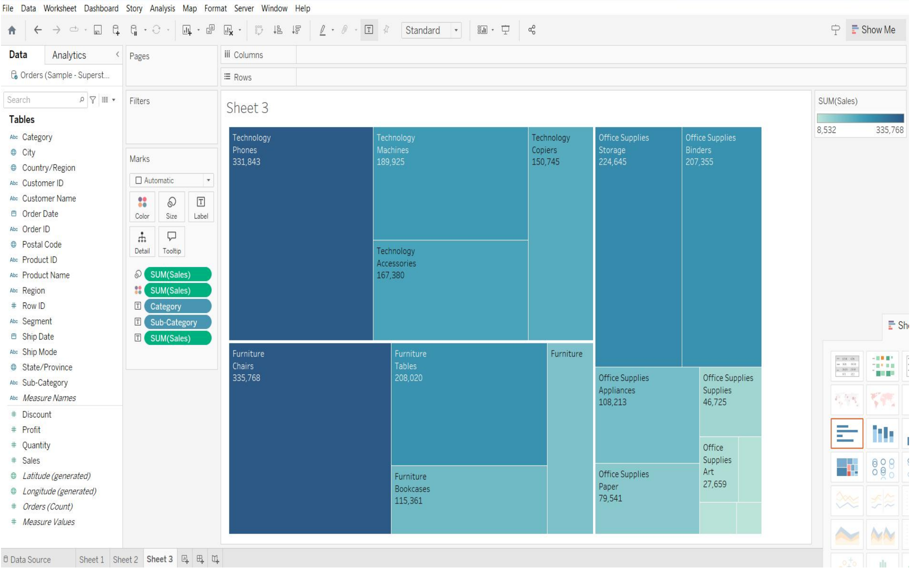
1. **Create a New Worksheet**.

2. **Set Up the Heat Map**:
3. Drag Region to **Rows** and Category to **Columns**.
4. **Color by Sales**:
5. Drag Sales to **Color** in the **Marks** card. Tableau will automatically color the cells based on sales values.
6. **Adjust Color Scheme**:
7. Choose a color gradient to highlight high and low sales values, helping identify top-performing categories by region.



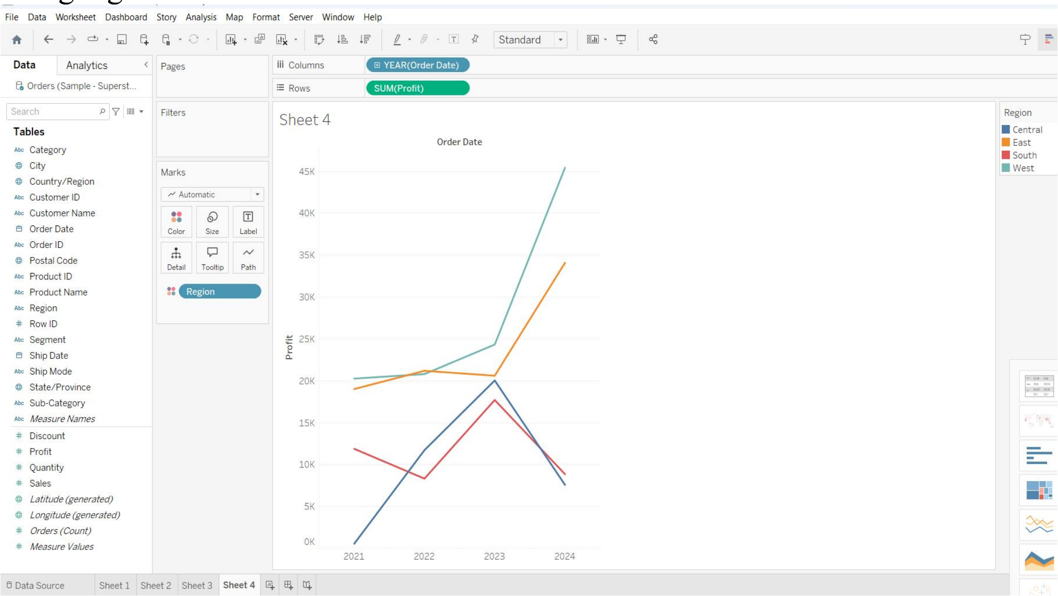
Worksheet 3: Tree Map of Sales by Category and Sub-Category

1. **Create a New Worksheet**.
2. **Build the Tree Map**:
3. Drag Category and Sub-Category to **rows or columns shelf.**
4. Drag Sales to **Size** and **Color** in the **Marks** card.
5. **Select Tree Map**:
6. In the **Marks** dropdown, choose **Treemap** to visualize sales distribution by product sub-category, with size and color representing sales value.



Worksheet 4: Line chart of year wise profits for all regions

1. **Create a New Worksheet**.
2. **Set Up Line Chart**:
3. Drag Order date to **Columns**.
4. Drag profits to **Rows** .
5. **Add Region to color**:
6. Drag region to **color** in the **Marks** card.



Columns: YEAR(Order Date)

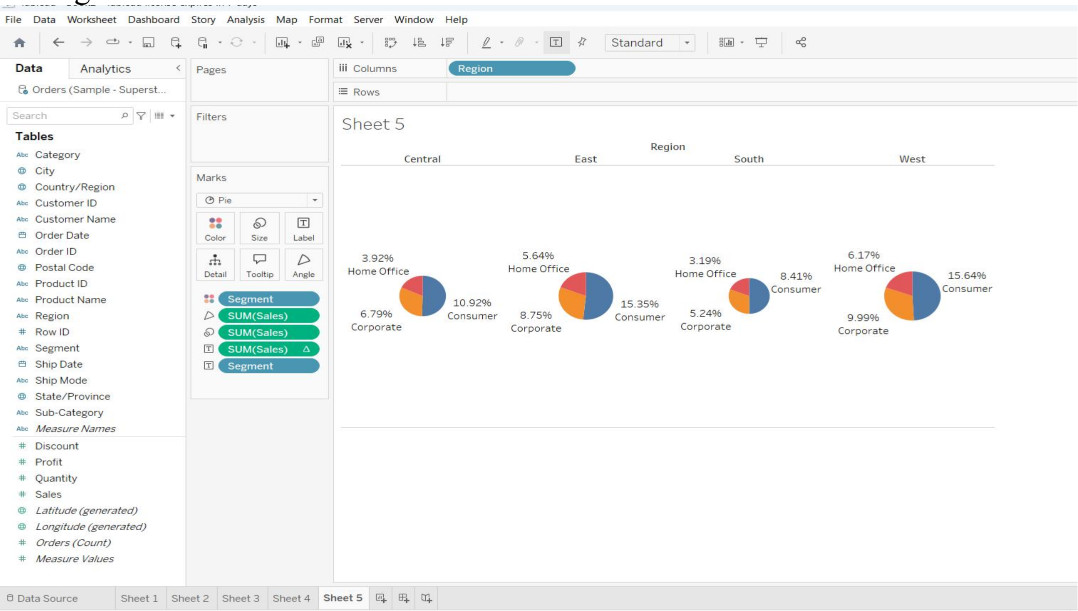
Rows: SUM(Profit)

Marks: Automatic

Color: Region

Worksheet 5: Pie Chart (Sales by region and Customer Segment)

1. **Create a New Worksheet**.
2. **Set Up the Pie Chart**:
3. Drag region and Segment to **rows**.
4. Drag Sales to **columns**.
5. **Choose Pie Chart Type**:
6. In the **Marks** dropdown, select **Pie**.
7. **Adjust Size and Labels**:
8. Use the **Size** slider to adjust the pie size and show percentage labels to make segment contributions clearer.



Columns: Region

Marks: Pie

Segment

SUM(Sales)

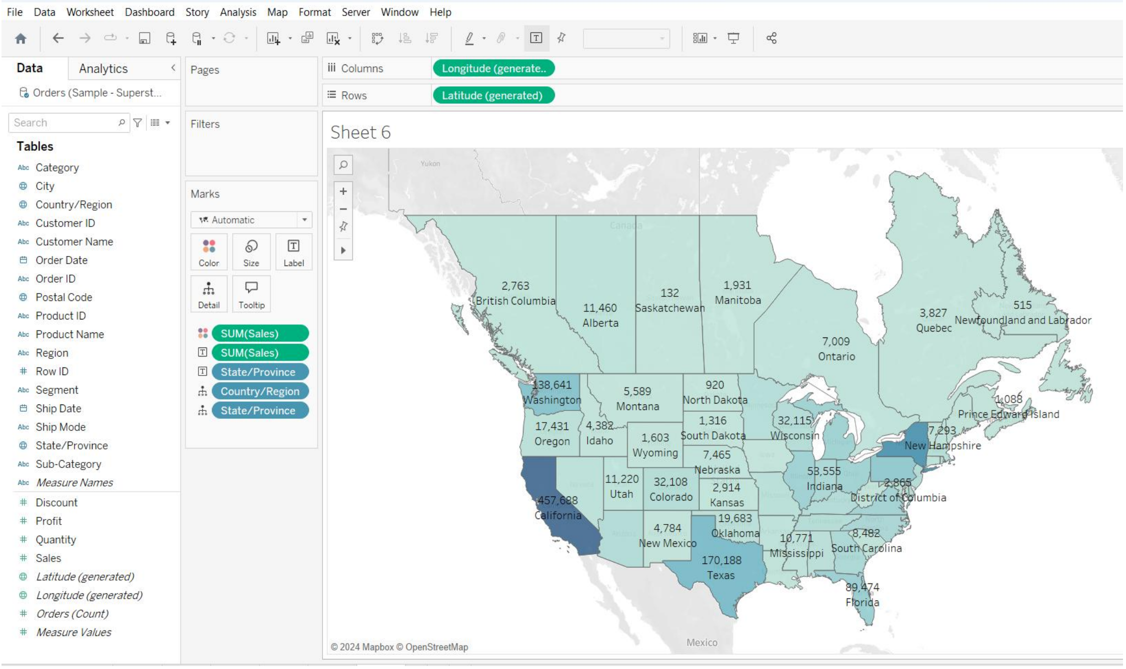
SUM(Sales)

SUM(Sales)

Segment

Worksheet 6: Basic Map (Sales by region and state)

1. **Create a New Worksheet**.
2. **Add Geographic Data**:
3. Drag Country and State to **visualization area**.
4. **Color by Sales**:
5. Drag Sales to **size** and region to **Color** to visualize sales intensity by region.
6. **Add Sales Labels** (Optional):
7. Drag Sales to **Label** to display sales numbers on each region.



Columns: Longitude (generate..

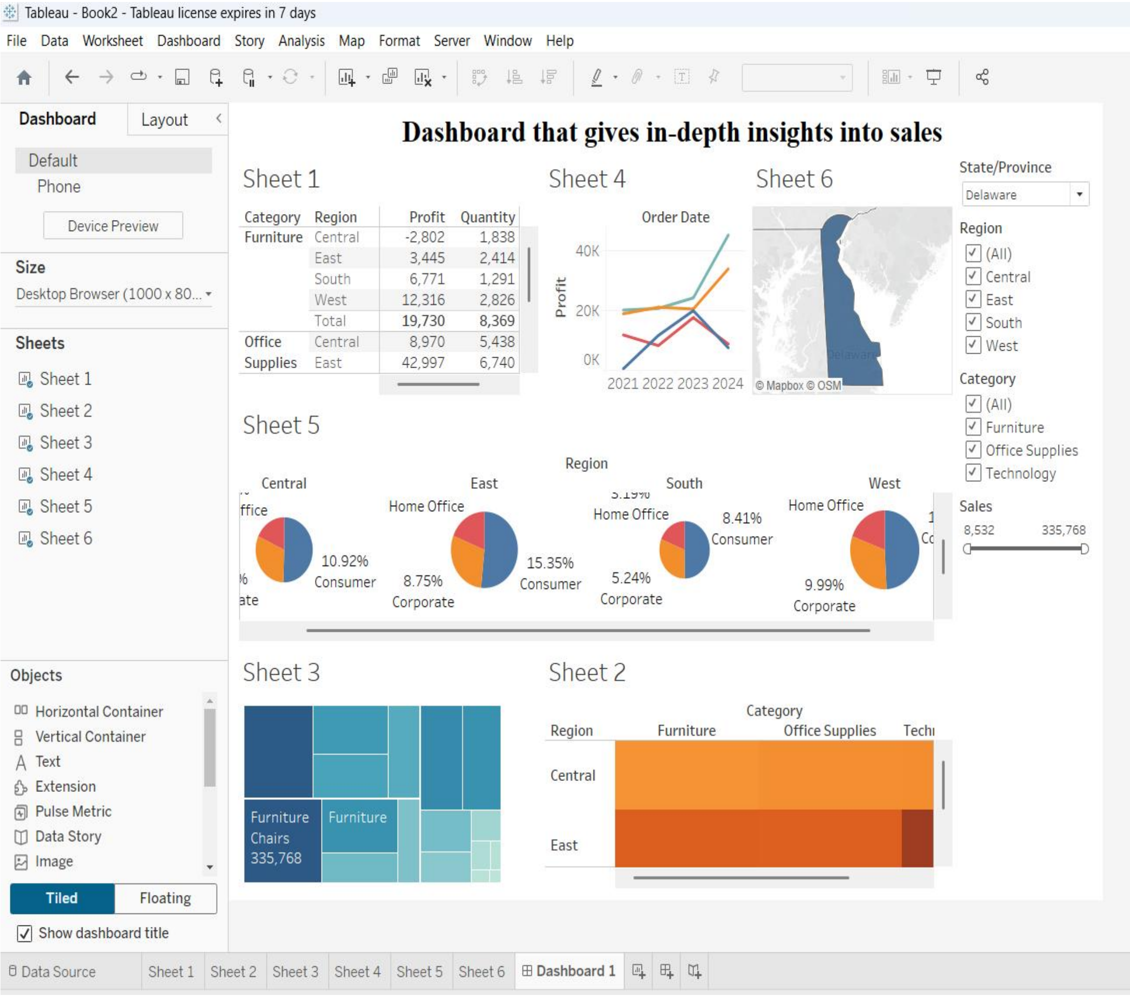
Rows: Latitude (generated)

Marks: Automatic

SUM(Sales)
SUM(Sales)
State/Province
Country/Region
State/Province

Step 3: Build the Dashboard

1. **Create a New Dashboard**:
2. Click on the **New Dashboard** icon.
3. **Add Worksheets**:
4. Drag each worksheet (table, heat map, tree map, line chart, pie chart, and map) onto the dashboard.
5. **Arrange Layout**:
6. Use horizontal and vertical containers to organize the layout for a clean, structured look.
7. Arrange visualizations for category performance, timelines, and geographical insights logically.
8. **Add Filters for Interactivity**:
9. Add filters like Region, Product Category, or Customer Segment to allow users to interact with and customize the view.
10. Set these filters to apply across all relevant sheets within the dashboard for consistency.
11. **Finalize and Publish**:
12. Review the dashboard for clarity, interactivity, and overall appearance.
13. Save and, if needed, publish to Tableau Public, Tableau Online, or Tableau Server.



Size: Desktop Browser (1000 x 80...

Tiled

☑ Show dashboard title

Dashboard that gives in-depth insights into sales

Sheet 1

Category	Region	Profit	Quantity
Furniture	Central	-2,802	1,838
	East	3,445	2,414
	South	6,771	1,291
	West	12,316	2,826
	Total	19,730	8,369
Office Supplies	Central	8,970	5,438
	East	42,997	6,740

State/Province: Delaware

Region
☑ (All)
☑ Central
☑ East
☑ South
☑ West

Category
☑ (All)
☑ Furniture
☑ Office Supplies
☑ Technology

Sales
8,532 335,768