Name:
Student Reference Number:

Module Code: PUSL 2076	Module Name: Data Programming R
Coursework Title:	
Deadline Date: 10/01/2024	Member of staff responsible for coursework:
Programme: BSc. (hons) in Data Science	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group: C. Paboda

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed :

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software………………………………………………………………

Overall mark _____% **Assessors Initials ______** **Date_________**

PUSL2076 Data Programming in R (23/AU/M)

Final Report

PUSL 2076 -Group Assignment

Group Member Details

PU index No.	Student Name	Degree Program
10899479	Kekulawala	Data Science
10899280	Adikaram Adikar	Data Science
10899495	Weerasiri	Data Science
10899488	Senanayake Senanayake	Data Science

Table of Contents

Introduction

The dataset used for the analysis is based on the sales and transactions of a supermarket. Every row in this dataset represents a unique transaction and each column gives an idea about a specific feature that is related to the sales.

Furthermore, this dataset contains data about the products sold, customer demographics, transaction details and different payment methods which were used by the customers. This data was used for various types of analyses to observe the behaviors of customers, to evaluate the performance of various products and for some predictions regarding sales of the supermarket.

Study summary of this part is to learn the outcomes of data visualization and data analysis part using R language. When we going to do this analysis parts we can use regression, classification, clustering methods.

We are going to analyze our data set by using based on 09 main columns.

1. Branch
2. City
3. Customer Type
4. Gender
5. Product Line
6. Unit price, Quantity, Tax, Total
7. Payment
8. Cost of goods sales, Gross Margin Percentage, Gross Income
9. Rating

Data Exploration and Cleaning

- The lubridate package is imported into R with the line library(lubridate). A package in R called lubridate makes it easier to handle and work with date and time objects.

- The below code line loads the CSV file "supermarket_sales - Sheet1.csv" from the given file directory into a variable called supermarket using the read.csv function. Using this command, the CSV file is read and saved as a supermarket data frame.

```
library(lubridate)
supermarket<-read.csv("C:\\Users\\ASUS\\Desktop\\supermarket_sales - Sheet1.csv")
```

```
> supermarket<-read.csv("C:\\Users\\ASUS\\Desktop\\supermarket_sales - Sheet1.csv")
>
```

- The supermarket data frame's columns' summary statistics are provided for each by the summary() function. A brief synopsis of the factors (like counts of unique values) and numerical columns (like mean, median, min, and max) is provided.

```
supermarket
summary(supermarket)
```

```
> summary(supermarket)
  Invoice.ID            Branch              City           Customer.type          Gender           Product.line      
 Length:1000        Length:1000        Length:1000        Length:1000        Length:1000        Length:1000       
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character  


   Unit.price       Quantity         Tax.5.            Total              Date               Time          
 Min.   :10.08   Min.   : 1.00   Min.   : 0.5085   Min.   :  10.68   Length:1000        Length:1000       
 1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249   1st Qu.: 124.42   Class :character   Class :character  
 Median :55.23   Median : 5.00   Median :12.0880   Median : 253.85   Mode  :character   Mode  :character  
 Mean   :55.67   Mean   : 5.51   Mean   :15.3794   Mean   : 322.97                                        
 3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453   3rd Qu.: 471.35                                        
 Max.   :99.96   Max.   :10.00   Max.   :49.6500   Max.   :1042.65                                        
   Payment               cogs         gross.margin.percentage  gross.income         Rating      
 Length:1000        Min.   : 10.17   Min.   :4.762           Min.   : 0.5085   Min.   : 4.000  
 Class :character   1st Qu.:118.50   1st Qu.:4.762           1st Qu.: 5.9249   1st Qu.: 5.500  
 Mode  :character   Median :241.76   Median :4.762           Median :12.0880   Median : 7.000  
                    Mean   :307.59   Mean   :4.762           Mean   :15.3794   Mean   : 6.973  
                    3rd Qu.:448.90   3rd Qu.:4.762           3rd Qu.:22.4453   3rd Qu.: 8.500  
                    Max.   :993.00   Max.   :4.762           Max.   :49.6500   Max.   :10.000  
```

- The head function shows the first few rows of the supermarket dataset (by default, the first six rows), giving you an idea of how the data initially appears.

  The tail function shows you the data near the end of the supermarket dataset by displaying the last few rows (again, by default, the last 6 rows). It works similarly to head().

```
5 head(supermarket)
```

```
> head(supermarket)
   Invoice.ID Branch      City Customer.type Gender            Product.line Unit.price Quantity  Tax.5.    Total
1 750-67-8428      A    Yangon        Member Female       Health and beauty      74.69        7 26.1415 548.9715
2 226-31-3081      C Naypyitaw        Normal Female Electronic accessories      15.28        5  3.8200  80.2200
3 631-41-3108      A    Yangon        Normal   Male      Home and lifestyle      46.33        7 16.2155 340.5255
4 123-19-1176      A    Yangon        Member   Male       Health and beauty      58.22        8 23.2880 489.0480
5 373-73-7910      A    Yangon        Normal   Male       Sports and travel      86.31        7 30.2085 634.3785
6 699-14-3026      C Naypyitaw        Normal   Male Electronic accessories      85.39        7 29.8865 627.6165
       Date  Time     Payment   cogs gross.margin.percentage gross.income Rating
1  1/5/2019 13:08     Ewallet 522.83                4.761905      26.1415    9.1
2  3/8/2019 10:29        Cash  76.40                4.761905       3.8200    9.6
3  3/3/2019 13:23 Credit card 324.31                4.761905      16.2155    7.4
4 1/27/2019 20:33     Ewallet 465.76                4.761905      23.2880    8.4
5  2/8/2019 10:37     Ewallet 604.17                4.761905      30.2085    5.3
6 3/25/2019 18:30     Ewallet 597.73                4.761905      29.8865    4.1
```

```
6 tail(supermarket)
```

```
> tail(supermarket)
     Invoice.ID Branch      City Customer.type Gender            Product.line Unit.price Quantity  Tax.5.     Total
995  652-49-6720      C Naypyitaw        Member Female Electronic accessories      60.95        1  3.0475   63.9975
996  233-67-5758      C Naypyitaw        Normal   Male      Health and beauty      40.35        1  2.0175   42.3675
997  303-96-2227      B  Mandalay        Normal Female     Home and lifestyle      97.38       10 48.6900 1022.4900
998  727-02-1313      A    Yangon        Member   Male     Food and beverages      31.84        1  1.5920   33.4320
999  347-56-2442      A    Yangon        Normal   Male     Home and lifestyle      65.82        1  3.2910   69.1110
1000 849-09-3807      A    Yangon        Member Female    Fashion accessories      88.34        7 30.9190  649.2990
          Date  Time Payment   cogs gross.margin.percentage gross.income Rating
995  2/18/2019 11:40 Ewallet  60.95                4.761905       3.0475    5.9
996  1/29/2019 13:46 Ewallet  40.35                4.761905       2.0175    6.2
997   3/2/2019 17:16 Ewallet 973.80                4.761905      48.6900    4.4
998   2/9/2019 13:22    Cash  31.84                4.761905       1.5920    7.7
999  2/22/2019 15:33    Cash  65.82                4.761905       3.2910    4.1
1000 2/18/2019 13:28    Cash 618.38                4.761905      30.9190    6.6
```

- The is.na(supermarket) tool is used to search the supermarket dataset for any missing entries. When applied on a supermarket-style dataframe, it yields a similar-sized dataframe with each cell changed with either FALSE (when the value is not missing) or TRUE (if the value is missing, NA).

```
7  is.na(supermarket)
```

```
> is.na(supermarket)
      Invoice.ID Branch  City Customer.type Gender Product.line Unit.price Quantity Tax.5. Total  Date  Time Payment
 [1,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [2,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [3,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [4,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [5,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [6,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [7,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [8,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
 [9,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[10,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[11,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[12,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[13,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[14,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[15,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[16,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[17,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[18,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[19,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
[20,]      FALSE  FALSE FALSE         FALSE  FALSE        FALSE      FALSE    FALSE  FALSE FALSE FALSE FALSE   FALSE
```

- In the below code, sum() makes use of is.na(supermarket). This computes the total TRUE values in the supermarket dataframe as a consequence of the is.na() method. This provides an overall count of all the missing values in the collection.

```
sum(is.na(supermarket))
```

```
> sum(is.na(supermarket))
[1] 0
```

- The below line retrieves the supermarket dataset's 'Date' column's class (data type). The purpose of this line is to verify the 'Date' column's initial class.

```
class(supermarket$Date)
```

```
> class(supermarket$Date)
[1] "character"
```

- With the use of the as.Date() function, this line changes the 'Date' column in the supermarket dataset from its current format—which might be a character or factor—to a correct Date format. The original dataset's date format is specified by the format = "%m/%d/%Y" argument, where %m denotes the month, %d the day, and %Y the year.

  print(supermarket$Date): This line outputs the converted 'Date' column, updated. It displays the 'Date' column in the format of the Date once it has been converted.

```
supermarket$Date <- as.Date(supermarket$Date,format = "%m/%d/%y")
print(supermarket$Date)
```

```
> supermarket$Date <- as.Date(supermarket$Date,format = "%m/%d/%y")
> print(supermarket$Date)
  [1] "2020-01-05" "2020-03-08" "2020-03-03" "2020-01-27" "2020-02-08" "2020-03-25" "2020-02-25" "2020-02-24"
  [9] "2020-01-10" "2020-02-20" "2020-02-06" "2020-03-09" "2020-02-12" "2020-02-07" "2020-03-29" "2020-01-15"
 [17] "2020-03-11" "2020-01-01" "2020-01-21" "2020-03-11" "2020-02-25" "2020-03-05" "2020-03-15" "2020-02-17"
 [25] "2020-03-02" "2020-03-22" "2020-02-08" "2020-03-10" "2020-01-25" "2020-03-15" "2020-02-25" "2020-01-28"
 [33] "2020-01-10" "2020-03-15" "2020-02-06" "2020-01-07" "2020-03-10" "2020-01-15" "2020-03-23" "2020-03-03"
 [41] "2020-01-17" "2020-02-02" "2020-02-08" "2020-03-04" "2020-03-16" "2020-03-09" "2020-02-27" "2020-02-06"
 [49] "2020-02-10" "2020-03-19" "2020-02-03" "2020-02-10" "2020-03-22" "2020-01-25" "2020-03-07" "2020-02-28"
 [57] "2020-03-27" "2020-02-07" "2020-01-20" "2020-03-12" "2020-02-15" "2020-02-24" "2020-02-03" "2020-03-06"
 [65] "2020-02-14" "2020-03-13" "2020-02-10" "2020-01-07" "2020-01-24" "2020-02-02" "2020-01-06" "2020-02-11"
 [73] "2020-03-05" "2020-03-09" "2020-01-22" "2020-01-13" "2020-01-09" "2020-01-12" "2020-03-05" "2020-01-22"
 [81] "2020-01-21" "2020-01-26" "2020-01-23" "2020-02-23" "2020-03-09" "2020-03-05" "2020-03-25" "2020-03-27"
 [89] "2020-01-02" "2020-02-27" "2020-01-23" "2020-01-26" "2020-01-10" "2020-03-12" "2020-02-06" "2020-03-08"
 [97] "2020-03-29" "2020-02-09" "2020-03-23" "2020-03-05" "2020-03-26" "2020-03-01" "2020-02-01" "2020-03-28"
[105] "2020-03-19" "2020-01-12" "2020-01-05" "2020-03-22" "2020-03-24" "2020-03-03" "2020-02-05" "2020-02-05"
[113] "2020-02-15" "2020-01-19" "2020-02-01" "2020-03-02" "2020-03-05" "2020-01-16" "2020-02-02" "2020-01-20"
[121] "2020-02-14" "2020-01-12" "2020-03-09" "2020-03-13" "2020-03-09" "2020-03-10" "2020-01-27" "2020-01-08"
[129] "2020-01-08" "2020-02-08" "2020-01-25" "2020-03-06" "2020-02-10" "2020-02-17" "2020-03-08" "2020-02-18"
[137] "2020-01-18" "2020-02-18" "2020-02-16" "2020-03-16" "2020-01-23" "2020-01-25" "2020-02-05" "2020-02-22"
[145] "2020-01-21" "2020-03-08" "2020-02-10" "2020-03-19" "2020-03-06" "2020-03-27" "2020-03-23" "2020-03-11"
[153] "2020-01-29" "2020-02-07" "2020-01-28" "2020-02-20" "2020-01-04" "2020-03-07" "2020-03-30" "2020-03-27"
[161] "2020-01-19" "2020-02-25" "2020-03-13" "2020-01-30" "2020-02-20" "2020-02-25" "2020-01-16" "2020-02-08"
[169] "2020-01-19" "2020-02-01" "2020-01-03" "2020-01-26" "2020-03-03" "2020-01-19" "2020-01-18" "2020-03-21"
[177] "2020-03-03" "2020-02-13" "2020-03-23" "2020-01-28" "2020-02-09" "2020-01-23" "2020-03-23" "2020-01-25"
[185] "2020-03-04" "2020-03-05" "2020-03-03" "2020-02-08" "2020-02-10" "2020-01-28" "2020-02-11" "2020-01-15"
[193] "2020-03-16" "2020-01-26" "2020-03-19" "2020-01-13" "2020-03-26" "2020-03-23" "2020-03-12" "2020-02-17"
[201] "2020-01-29" "2020-03-15" "2020-01-14" "2020-02-06" "2020-02-15" "2020-01-03" "2020-01-04" "2020-03-18"
```

- This line, like the last one, determines the class of the supermarket dataset's "Time" column, giving R insight into how it is interpreting this column.

- Then by the below code it changes the 'Time' column in the supermarket dataset to the correct POSIXct format—a class in R that represents dates and times. The original dataset's time format, with %H denoting the hour and %M the minute, is specified by the format = "%H:%M" option.

- This line outputs the modified "Time" column following the conversion, illustrating the appearance of the "Time" column following its conversion to the POSIXct format.

```r
class(supermarket$Time)
supermarket$Time <- as.POSIXct(supermarket$Time,format = "%H:%M")
print(supermarket$Time)
```

```
> supermarket<-read.csv("C:\\Users\\ASUS\\Desktop\\supermarket_sales - Sheet1.csv")
> supermarket$Time <- as.POSIXct(supermarket$Time,format = "%H:%M")
> print(supermarket$Time)
  [1] "2024-02-13 13:08:00 +0530" "2024-02-13 10:29:00 +0530" "2024-02-13 13:23:00 +0530" "2024-02-13 20:33:00 +0530"
  [5] "2024-02-13 10:37:00 +0530" "2024-02-13 18:30:00 +0530" "2024-02-13 14:36:00 +0530" "2024-02-13 11:38:00 +0530"
  [9] "2024-02-13 17:15:00 +0530" "2024-02-13 13:27:00 +0530" "2024-02-13 18:07:00 +0530" "2024-02-13 17:03:00 +0530"
 [13] "2024-02-13 10:25:00 +0530" "2024-02-13 16:48:00 +0530" "2024-02-13 19:21:00 +0530" "2024-02-13 16:19:00 +0530"
 [17] "2024-02-13 11:03:00 +0530" "2024-02-13 10:39:00 +0530" "2024-02-13 18:00:00 +0530" "2024-02-13 15:30:00 +0530"
 [21] "2024-02-13 11:24:00 +0530" "2024-02-13 10:40:00 +0530" "2024-02-13 12:20:00 +0530" "2024-02-13 11:15:00 +0530"
 [25] "2024-02-13 17:36:00 +0530" "2024-02-13 19:20:00 +0530" "2024-02-13 15:31:00 +0530" "2024-02-13 12:17:00 +0530"
 [29] "2024-02-13 19:48:00 +0530" "2024-02-13 15:36:00 +0530" "2024-02-13 19:39:00 +0530" "2024-02-13 12:43:00 +0530"
 [33] "2024-02-13 14:49:00 +0530" "2024-02-13 10:12:00 +0530" "2024-02-13 10:42:00 +0530" "2024-02-13 12:28:00 +0530"
 [37] "2024-02-13 19:15:00 +0530" "2024-02-13 17:17:00 +0530" "2024-02-13 13:24:00 +0530" "2024-02-13 13:01:00 +0530"
 [41] "2024-02-13 18:45:00 +0530" "2024-02-13 10:11:00 +0530" "2024-02-13 13:03:00 +0530" "2024-02-13 20:39:00 +0530"
 [45] "2024-02-13 19:47:00 +0530" "2024-02-13 18:00:00 +0530" "2024-02-13 17:24:00 +0530" "2024-02-13 15:47:00 +0530"
 [49] "2024-02-13 12:45:00 +0530" "2024-02-13 17:08:00 +0530" "2024-02-13 10:19:00 +0530" "2024-02-13 15:10:00 +0530"
 [53] "2024-02-13 14:42:00 +0530" "2024-02-13 15:46:00 +0530" "2024-02-13 11:49:00 +0530" "2024-02-13 19:01:00 +0530"
 [57] "2024-02-13 11:26:00 +0530" "2024-02-13 11:28:00 +0530" "2024-02-13 15:55:00 +0530" "2024-02-13 20:36:00 +0530"
 [61] "2024-02-13 17:47:00 +0530" "2024-02-13 10:55:00 +0530" "2024-02-13 13:40:00 +0530" "2024-02-13 12:27:00 +0530"
 [65] "2024-02-13 14:35:00 +0530" "2024-02-13 16:40:00 +0530" "2024-02-13 15:43:00 +0530" "2024-02-13 15:01:00 +0530"
 [69] "2024-02-13 10:04:00 +0530" "2024-02-13 18:50:00 +0530" "2024-02-13 12:46:00 +0530" "2024-02-13 16:19:00 +0530"
 [73] "2024-02-13 18:17:00 +0530" "2024-02-13 18:21:00 +0530" "2024-02-13 10:55:00 +0530" "2024-02-13 17:04:00 +0530"
 [77] "2024-02-13 14:20:00 +0530" "2024-02-13 15:48:00 +0530" "2024-02-13 16:24:00 +0530" "2024-02-13 18:56:00 +0530"
 [81] "2024-02-13 14:42:00 +0530" "2024-02-13 19:56:00 +0530" "2024-02-13 18:37:00 +0530" "2024-02-13 18:45:00 +0530"
 [85] "2024-02-13 10:17:00 +0530" "2024-02-13 14:31:00 +0530" "2024-02-13 10:23:00 +0530" "2024-02-13 20:35:00 +0530"
 [89] "2024-02-13 16:57:00 +0530" "2024-02-13 17:55:00 +0530" "2024-02-13 10:25:00 +0530" "2024-02-13 19:54:00 +0530"
 [93] "2024-02-13 16:42:00 +0530" "2024-02-13 12:09:00 +0530" "2024-02-13 20:05:00 +0530" "2024-02-13 20:38:00 +0530"
 [97] "2024-02-13 10:25:00 +0530" "2024-02-13 13:11:00 +0530" "2024-02-13 10:16:00 +0530" "2024-02-13 18:14:00 +0530"
[101] "2024-02-13 19:20:00 +0530" "2024-02-13 13:22:00 +0530" "2024-02-13 11:27:00 +0530" "2024-02-13 16:44:00 +0530"
```

Data Analysis and Visualization

- Data analysis and visualization is done in various methods such as ;
  a) Bar charts
  b) Histograms
  c) Pie charts
  d) Scatterplots
  e) Box plots

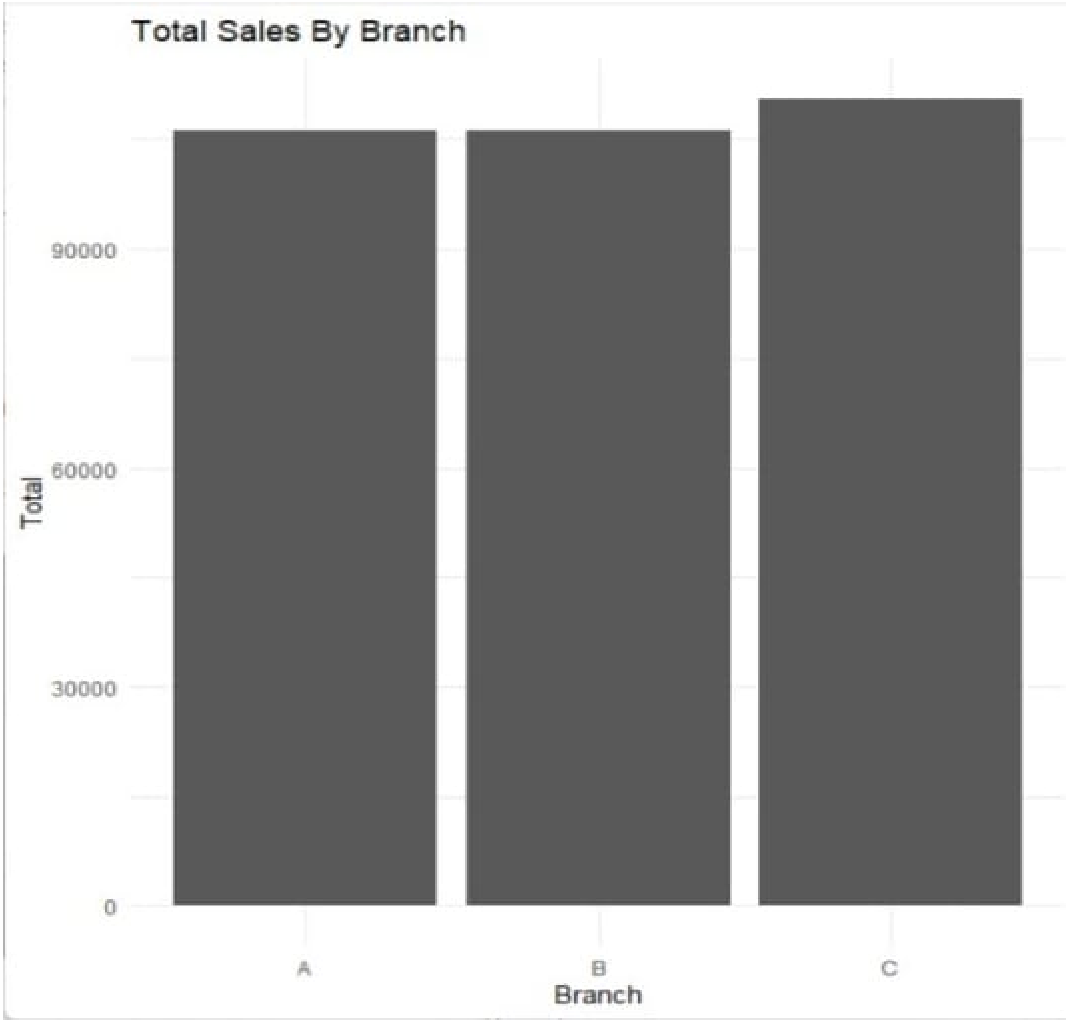
a) Bar charts

Bar chart for the Total Sales by Branch :

Input (Code) :

```
library(ggplot2)# drawing plots

getwd()
dataset <- read.csv("supermarket.csv")

dataset

summary(dataset)
plot(dataset)


ggplot(dataset,aes(Branch,Total,TotalSalesByBranch))+
  geom_point()
ggplot(dataset,aes(Branch,Total,TotalSalesByBranch))+
  geom_bar(stat = "identity") +
  ggtitle("Total Sales By Branch")+
  ylab("Total")+
  theme_minimal()
```

Output :



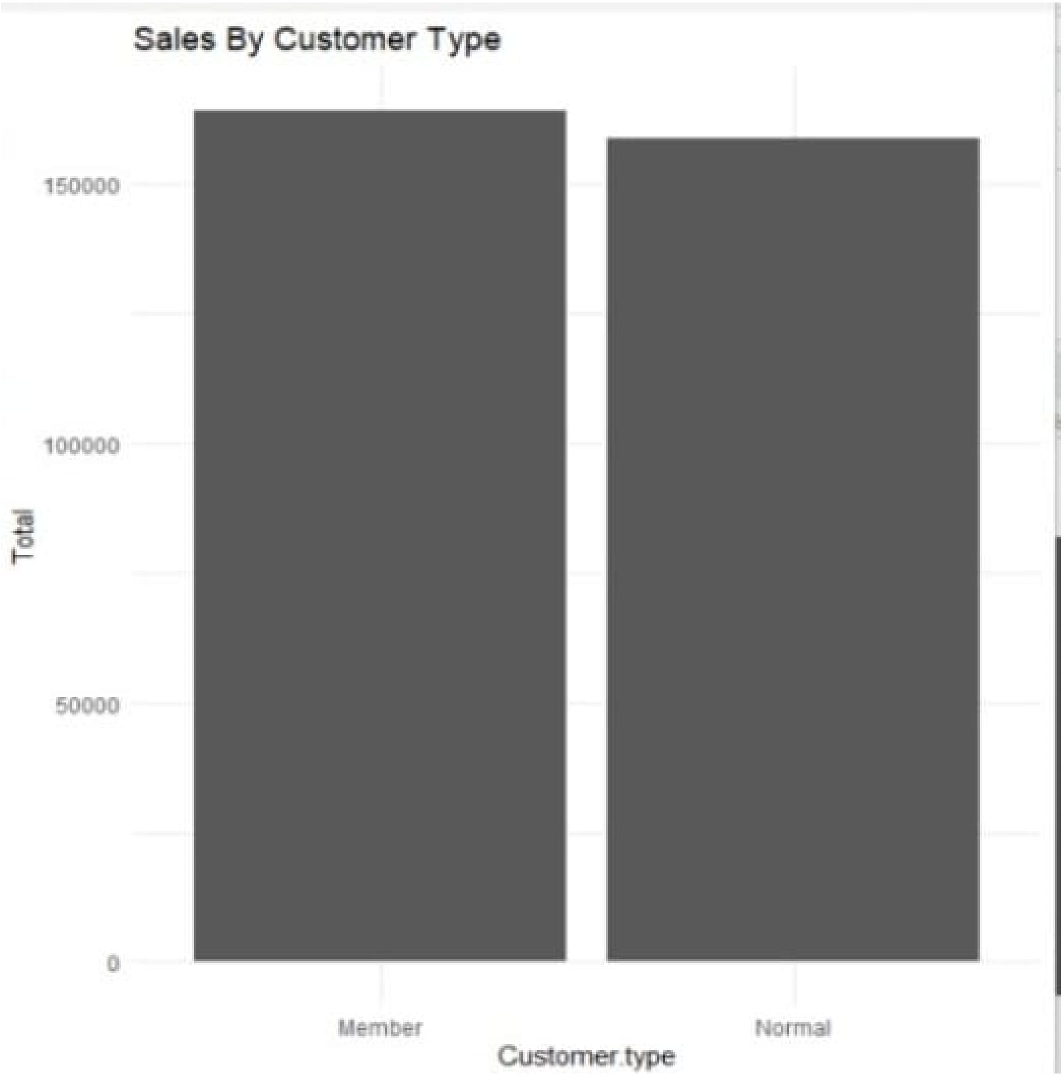
- This graph explains a line graph showing total sales per branch of a company, with rupees on the y-axis and branch labels on the x-axis. Sales increase steadily from B1 to B10. Text above suggests a 114% increase from the previous day, and the date is shown at the bottom-left corner. Overall, it demonstrates sales growth across branches, with B10 having the highest sales and B1 the lowest.

Bar chart for the Sales By Customer Type :

Input (Code):

```
dataset

summary(dataset)
plot(dataset)


ggplot(dataset,aes(Branch,Total,TotalSalesByBranch))+
  geom_point()
ggplot(dataset,aes(Branch,Total,TotalSalesByBranch))+
  geom_bar(stat = "identity") +
  ggtitle("Total Sales By Branch")+
  ylab("Total")+
  theme_minimal()

ggplot(dataset,aes(Customer.type,Total,SalesByCustomerType))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Customer Type")+
  ylab("Total")+
  theme_minimal()
```

Output :



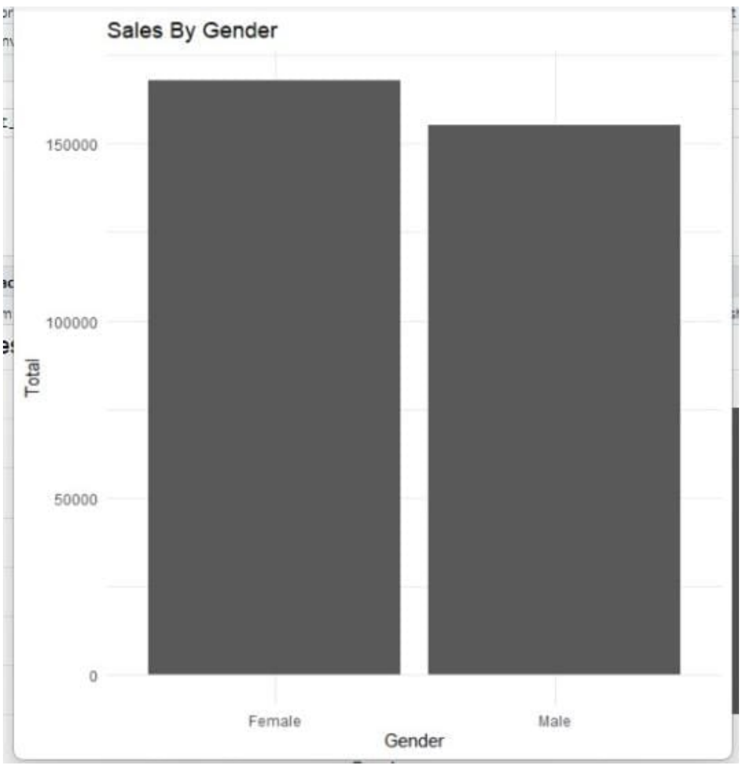
- The graph summarizes a bar chart showing sales by customer type. Members generated much higher sales than normal customers, with a blue bar reaching 150,000 Rupees compared to a green bar reaching only 50,000 Rupees, indicating a threefold difference. Text suggests a 123% increase in sales compared to the previous day, with the date displayed at the bottom-left corner. Overall, the graph emphasizes members as more valuable customers, with significantly higher sales revenue and a notable increase from the previous day.

Bar chart for the Sales by Gender :

Input (Code):

```
ggplot(dataset,aes(Branch,Total,TotalSalesByBranch))+
  geom_bar(stat = "identity") +
  ggtitle("Total Sales By Branch")+
  ylab("Total")+
  theme_minimal()

ggplot(dataset,aes(Customer.type,Total,TotalSalesByPaymentMethod))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Customer Type")+
  ylab("Total")+
  theme_minimal()

ggplot(dataset,aes(Gender,Total,TotalSalesByGender))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Gender")+
  ylab("Total")+
  theme_minimal()
```

Output:



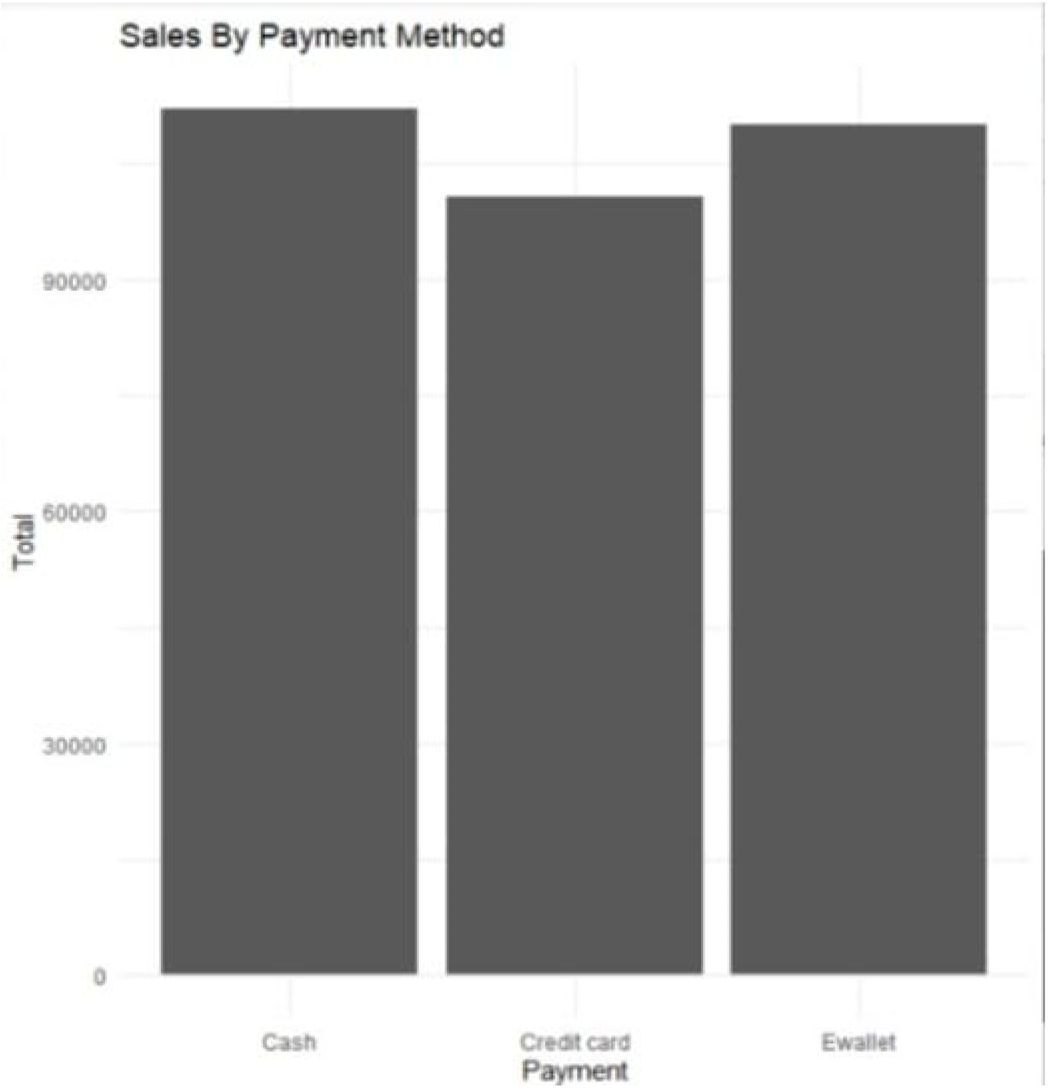
- The bar chart shows sales by gender, with males generating significantly more sales than females. Male sales total $1,500,000, represented by a blue bar, while female sales total $500,000, represented by a green bar, indicating a threefold difference. However, the text "20% 1219%" lacks context. The date "2/13/2024" is displayed in the bottom-left corner. It's crucial to consider other factors such as products, marketing, and demographics, as this data may not be universally applicable.

Bar chart for the Sales by Payment Method:

Input (Code):

```
  ylab("Total")+
  theme_minimal()

ggplot(dataset,aes(Customer.type,Total,TotalSalesByCustomerType))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Customer Type")+
  ylab("Total")+
  theme_minimal()

ggplot(dataset,aes(Gender,Total,TotalSalesByGender))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Gender")+
  ylab("Total")+
  theme_minimal()
ggplot(dataset,aes(Payment,Total,TotalSalesByPaymentMethod))+
  geom_bar(stat = "identity") +
  ggtitle("Sales By Payment Method")+
  ylab("Total")+
  theme_minimal()
```

Output :



- The bar chart illustrates sales by payment method, indicating Cash as the most popular with 90,000 sales, followed by Credit Card with 50,000, and enwalled with 30,000. Cash payments are three times more common than Enwalled. The text "3 of 10" and "123%" suggests a 123% increase in sales from the previous day, and it's the 3rd of 10 visualizations. Overall, Cash is the most popular payment method, followed by Credit Card and Enwalled, with a notable 123% increase in sales compared to the previous day.

Explanation.

1) Reading data

```
dataset <- read.csv("supermarket.csv")
```

- In passing through the intricacies of textual data, the algorithm navigates a CSV file bearing the nomenclature "supermarket.csv," orchestrating the assimilation of information into the designated variable, aptly titled dataset.

2) Summary Statistics

```
summary(dataset)
```

- In elucidating the dataset's characteristics, a compendium of statistical insights is proffered, encapsulating metrics encompassing the mean, median, minimum, and maximum values for every individual variable.

3) Scatter Plot:

```
plot(dataset)
```

- This line engenders a scatter plot for every conceivable pairing of variables within the dataset.

4)Bar charts:

```
ggplot(dataset, aes(Branch, Total, TotalSalesByBranch)) +
geom_bar(stat = "identity") +
ggtitle("Total Sales By Branch") +
ylab("Total") +theme_minimal()
```

- These segments give rise to a bar chart, elucidating the cumulative sales across different branches. Employing the ggplot function, the code meticulously delineates the dataset and aesthetic mappings, introducing bars through geom_bar, defining the title and y-axis label, all under the aegis of a minimalistic theme.

- Conforming patterns follow for bar charts contingent on customer classification, gender distribution, and payment methodology.

- In synopsis, the code leverages ggplot2 to visually represent total sales categorized by distinct facets such as branch, customer classification, gender, and payment methodology. Each code excerpt begets an individualized bar chart catering to the specified classification.

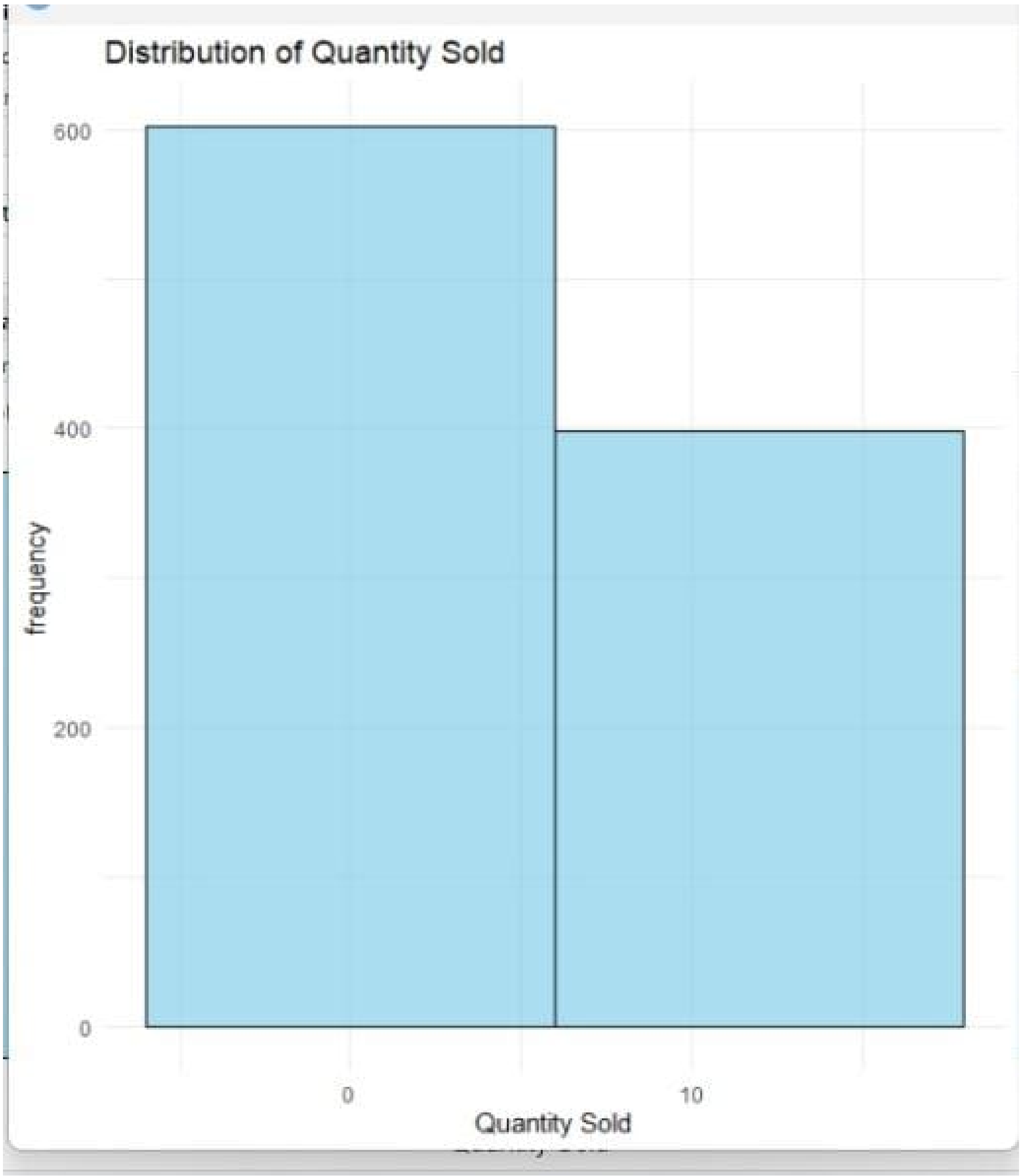
b) Histograms

Histogram for Quantity Sold :

Input (Code) :

```
  geom_bar(stat = "identity") +
  ggtitle("Sales By Payment Method")+
  ylab("Total")+
  theme_minimal()
ggplot(dataset, aes(x = Quantity)) +
  geom_histogram(binwidth = 12, fill = "skyblue", color = "black",
  ggtitle("Distribution of Quantity Sold") +
  xlab("Quantity Sold") +
  ylab("frequency") +
  theme_minimal()

ggplot(dataset, aes(x = Quantity)) +
  geom_histogram(binwidth = 12, fill = "skyblue", color = "black",
  ggtitle("Distribution of ") +
  xlab("Gross Income") +
  ylab("frequency") +
  theme_minimal()
```

Output :

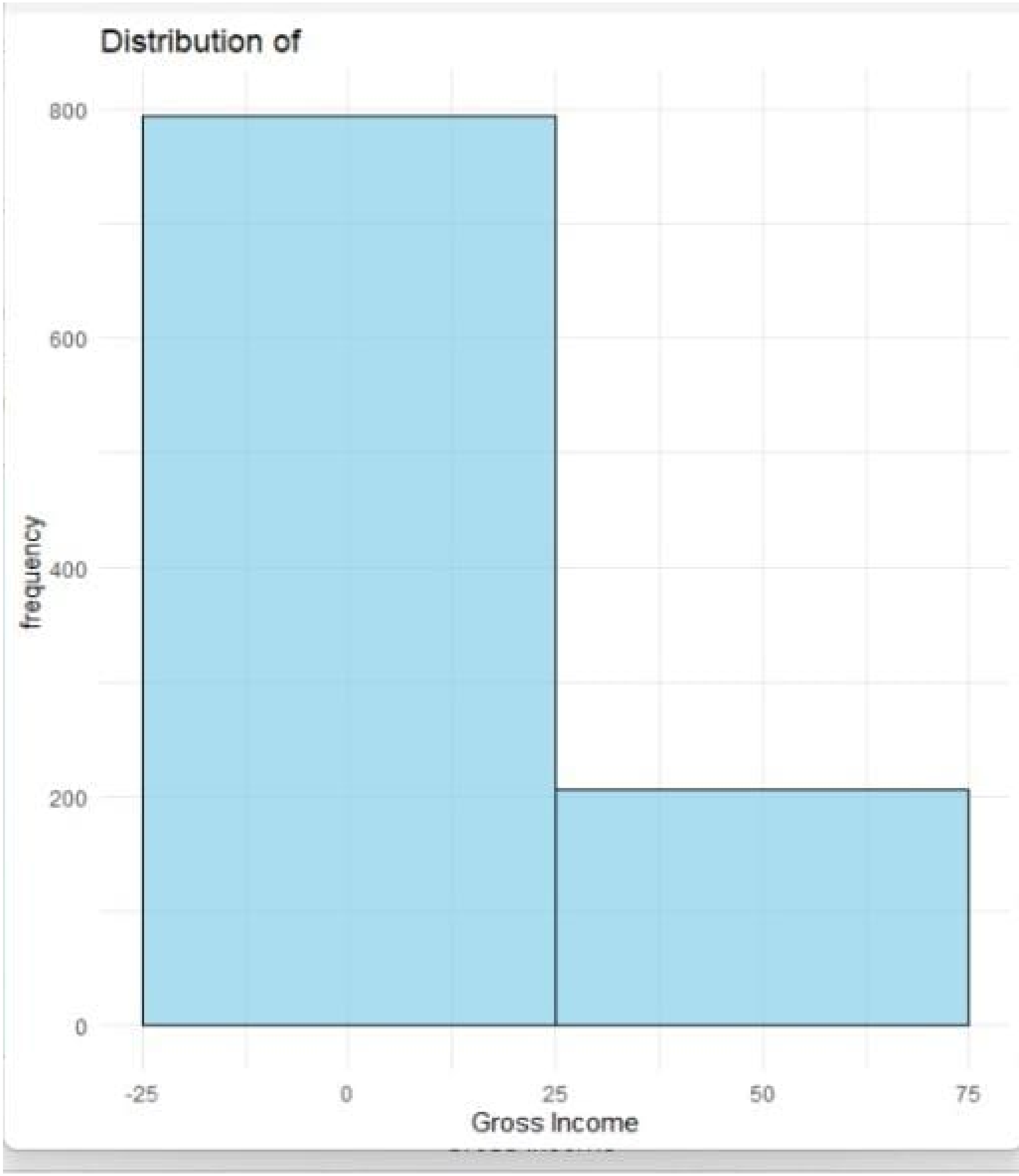

Distribution of Quantity Sold
600
400
200
0
frequency
0
10
Quantity Sold

Distribution of gross income :

Input(Code) :

```
  ggtitle("Sales By Payment Method")+
  ylab("Total")+
  theme_minimal()
ggplot(dataset, aes(x = Quantity)) +
  geom_histogram(binwidth = 12, fill = "skyblue", color = "black", 
  ggtitle("Distribution of Quantity Sold") +
  xlab("Quantity Sold") +
  ylab("frequency") +
  theme_minimal()

ggplot(dataset, aes(x = gross.income)) +
  geom_histogram(binwidth = 50, fill = "skyblue", color = "black", 
  ggtitle("Distribution of ") +
  xlab("Gross Income") +
  ylab("frequency") +
  theme_minimal()
```

Output :



- The paragraph describes a line graph illustrating the distribution of grades in a school, ranging from "F" to "A+". It notes the arrangement of grades on the x-axis and observes a peak in frequency at "C", indicating that most students fall within this range. However, without precise y-axis values, it's challenging to determine the exact number of students for each grade.

Explanation.

1) Histogram for Quantity Sold:

```
ggplot(dataset, aes(x = Quantity)) +
geom_histogram(binwidth = 12, fill = "skyblue", color = "black", alpha =
0.7) +
ggtitle("Distribution of Quantity Sold") +
xlab("Quantity Sold") + ylab("Frequency") +
theme_minimal()
```

aes(x = Quantity): Specifies that the x-axis should represent the "Quantity" variable.
geom_histogram(binwidth = 12, fill = "skyblue", color = "black", alpha = 0.7): Creates a histogram with a bin width of 12, filled with sky-blue color, black borders, and 70% transparency.
ggtitle("Distribution of Quantity Sold"): Sets the title of the plot.
xlab("Quantity Sold") + ylab("Frequency"): Labels the x and y axes.
theme_minimal(): Applies a minimal theme to the plot.

2) Histogram for Gross Income:

```
ggplot(dataset, aes(x = gross.income)) +

geom_histogram(binwidth = 50, fill = "skyblue", color = "black", alpha =
0.7)  +

ggtitle("Distribution of Gross Income") +

xlab("Gross Income") + ylab("Frequency") +

theme_minimal()
```

aes(x = gross.income): Specifies that the x-axis should represent the "gross.income" variable.

geom_histogram(binwidth = 50, fill = "skyblue", color = "black", alpha = 0.7): Creates a histogram with a bin width of 50, filled with sky-blue color, black borders, and 70% transparency.

ggtitle("Distribution of Gross Income"): Sets the title of the plot.

xlab("Gross Income") + ylab("Frequency"): Labels the x and y axes.

theme_minimal(): Applies a minimal theme to the plot.

These visualizations aim to show the distribution of values for "Quantity Sold" and "Gross Income" in your dataset. If you have specific questions or need further clarification, feel free to ask!

c) <u>Pie Charts</u>

Input (Code) :

```
  ggtitle("Distribution of Quantity Sold") +
  xlab("Quantity Sold") +
  ylab("frequency") +
  theme_minimal()

ggplot(dataset, aes(x = gross.income)) +
  geom_histogram(binwidth = 50, fill = "skyblue", color = "black",
  ggtitle("Distribution of ") +
  xlab("Gross Income") +
  ylab("frequency") +
  theme_minimal()

pie_chart <- ggplot(dataset, aes(x = "Quantity", y = Unit.price, fi
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  theme_minimal() +
  labs(title = "Product Line Distribution", fill = "Product.line")


print(pie_chart)
```

Output :

- The paragraph describes a pie chart showing the distribution of products across six categories. "Fashion accessories" and "Electronic accessories" are the largest segments, followed by "Home and lifestyle." Additional information includes the chart title, ambiguous text in the center, and the date displayed. Overall, the chart indicates the dominance of fashion and electronic accessories in product distribution.

Explanation.

In the aes function, it appears you're endeavoring to correlate the x-axis with the string "Quantity," a choice that may not align with your intended outcome. Additionally, the coord_polar function lacks a closing parenthesis after 'y''. Here's a rectified rendition of your code:

```
pie_chart <- ggplot(dataset, aes(x = "", y = Unit.price, fill = Product.line)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar('y', start = 0) +
  theme_minimal() +
  labs(title = "Product Line Distribution", fill = "Product Line")

print(pie_chart)
```

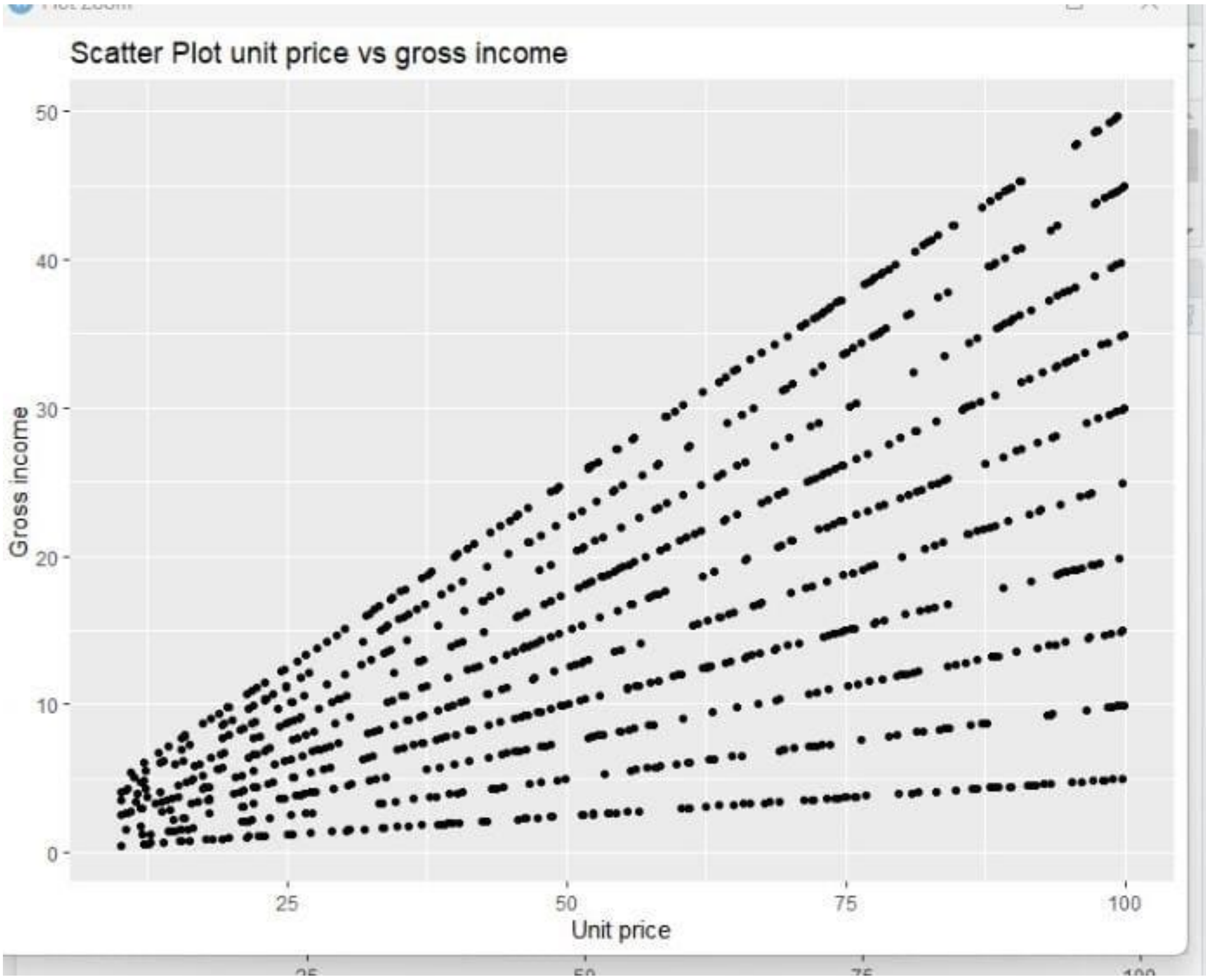
d) Scatter Plots

Scatter Plot for unit vs gross income :

Input (Code) :

```
  ylab("frequency") +
  theme_minimal()

#Pie Charts
pie_chart <- ggplot(dataset, aes(x = "Quantity", y = Unit.price, fi
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  theme_minimal() +
  labs(title = "Product Line Distribution", fill = "Product.line")


print(pie_chart)


#Scatter Plot
scatter_plot <- ggplot(dataset, aes(x = Unit.price, y = gross.income
  geom_point() +
  labs(title = "Scatter Plot unit price vs gross income", x = "Unit


print(scatter_plot)
```

Output :

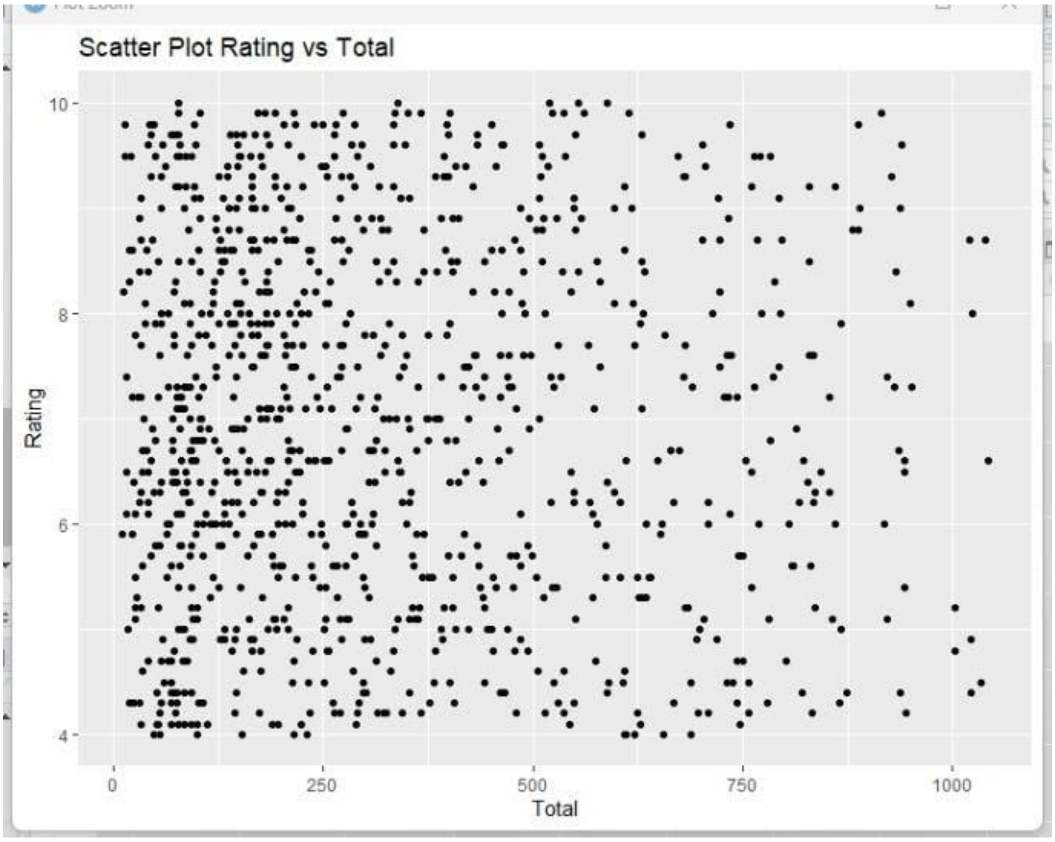

Scatter Plot unit price vs gross income
Gross income
Unit price
0
10
20
30
40
50
25
50
75
100

Scatter plot for rating vs total :

Input (Code) :

```
  coord_polar("y", start = 0) +
  theme_minimal() +
  labs(title = "Product Line Distribution", fill = "Product.line")


print(pie_chart)


#Scatter Plot
scatter_plot <- ggplot(dataset, aes(x = Unit.price, y = gross.income
  geom_point() +
  labs(title = "Scatter Plot unit price vs gross income", x = "Unit


print(scatter_plot)

scatter_plot <- ggplot(dataset, aes(x = Total, y = Rating)) +
  geom_point() +
  labs(title = "Scatter Plot Rating vs Total", x = "Total", y = "Ra


print(scatter_plot)
```

Output :



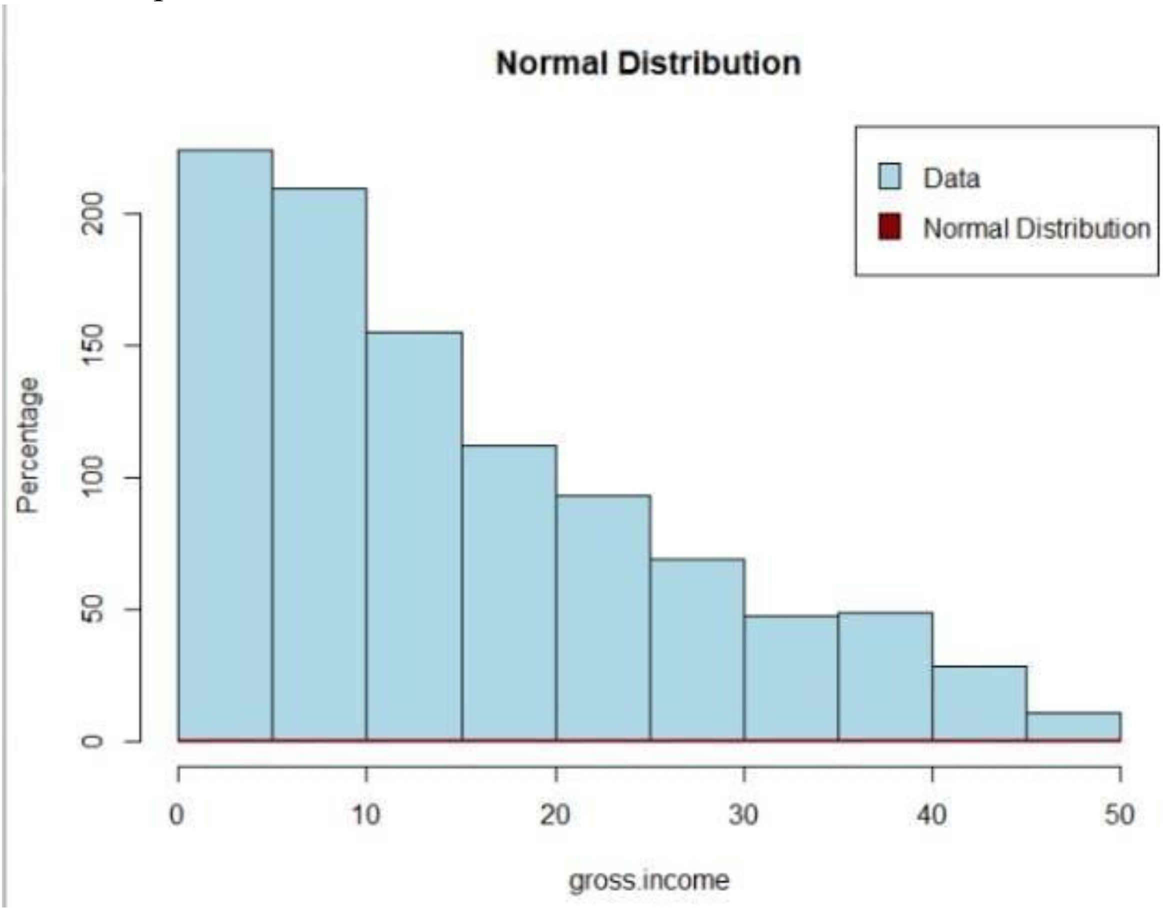
- A scatter plot illustrating ratings versus total review counts for various apps. It observes a general trend of higher ratings correlating with more reviews, though there are outliers. Specific examples are given of apps with 5-star ratings and high review counts, as well as those with 1-star ratings and low review counts. The scatter plot's title and date are mentioned, along with a reminder that the data represents only a snapshot and may not capture the entire app population.

Normal Distribution

Input (Code) :

```
print(scatter_plot)

scatter_plot <- ggplot(dataset, aes(x = Total, y = Rating)) +
  geom_point() +
  labs(title = "Scatter Plot Rating vs Total", x = "Total", y = "Ra

print(scatter_plot)

hist(dataset$gross.income, col = "lightblue", main = "Normal Distri

curve(dnorm(x, mean = mean(dataset$gross.income), sd = sd(dataset$g

legend("topright", legend = c("Data", "Normal Distribution"), fill
```

Output :



Explanation

Scatter Plot: Unit price vs Gross income

```
scatter_plot <- ggplot(dataset, aes(x = Unit.price, y = gross.income)) +
  geom_point() +
  labs(title = "Scatter Plot Unit price vs Gross income", x = "Unit price", y = "Gross income")

print(scatter_plot)
```

Scatter Plot: Total vs Rating

```
scatter_plot <- ggplot(dataset, aes(x = Total, y = Rating)) +
  geom_point() +
  labs(title = "Scatter Plot Rating vs Total", x = "Total", y = "Rating")

print(scatter_plot)
```

Histogram with Normal Distribution curve for gross income

```
hist(dataset$gross.income, col = "lightblue", main = "Normal Distribution", xlab = 'Gross income', ylab = "Percentage")

curve(dnorm(x, mean = mean(dataset$gross.income), sd = sd(dataset$gross.income)), col = "darkred", lwd = 2, add = TRUE)

legend("topright", legend = c("Data", "Normal Distribution"), fill = c("lightblue", "darkred"))
```

Custom percentage axis function

```
percent_axis <- function(x, pos) {
  paste0(format(x * 100, digits = 2), "%")
}
```

Plotting histogram with custom percentage labels

```
hist(dataset$gross.income, col = "lightblue", main = "Normal Distribution", xlab = 'Gross income',
   ylab = "Percentage", labels = TRUE, ylim = c(0, max(hist(dataset$gross.income, plot = FALSE)$counts) / length(dataset$gross.income)))


curve(dnorm(x, mean = mean(dataset$gross.income), sd = sd(dataset$gross.income)),
     diff(hist(dataset$gross.income, plot = FALSE)$breaks[1:2]) * length(dataset$gross.income),
     col = "darkred", lwd = 2, add = TRUE)


axis(2, at = axTicks(2), labels = percent_axis)


legend("topright", legend = c("Data", "Normal Distribution"), fill = c("lightblue", "darkred"))
```

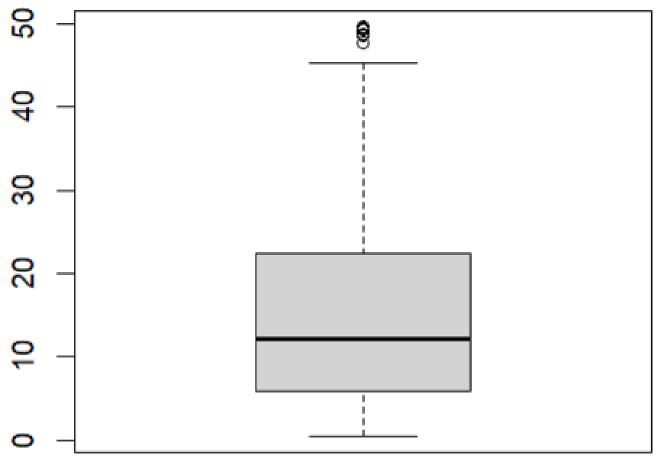
e) **Boxplots**

- These plots represented the
  - Median
  - Q_3
  - Q_1
  - Min value
  - Max value
  - Out liars of the each column.
- The following code is used to create box plots for the relevant columns in the dataset.

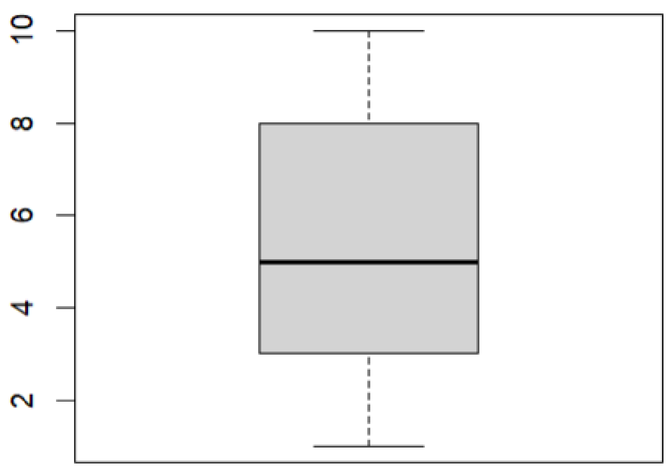
```
library(ggplot2)
library(e1071)
library(caret)
library(caTools)


boxplot(data$Unit.price)
boxplot(data$Quantity)
boxplot(data$Tax.5.)
boxplot(data$Total)
boxplot(data$cogs)
boxplot(data$gross.income)
```

- The following are the outputs of the above codes for box plots.

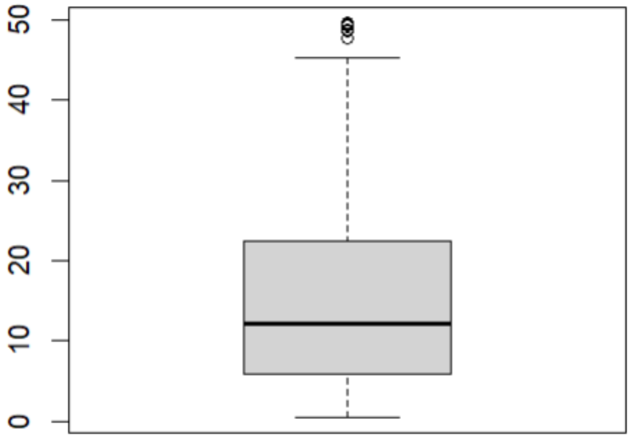
1. Boxplot(data$Unit.price)



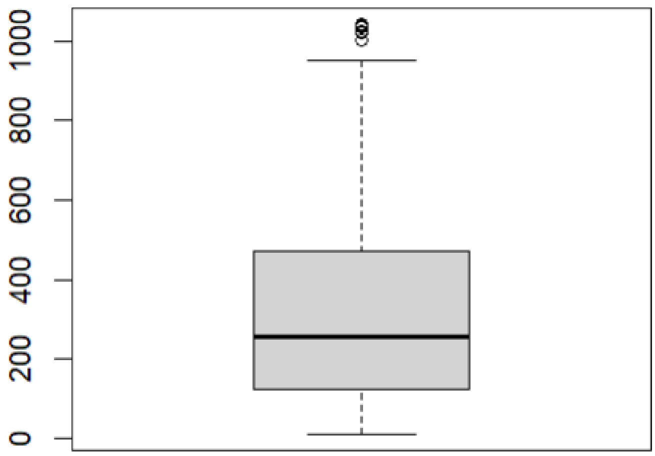
2. Boxplot(data$Quantity)



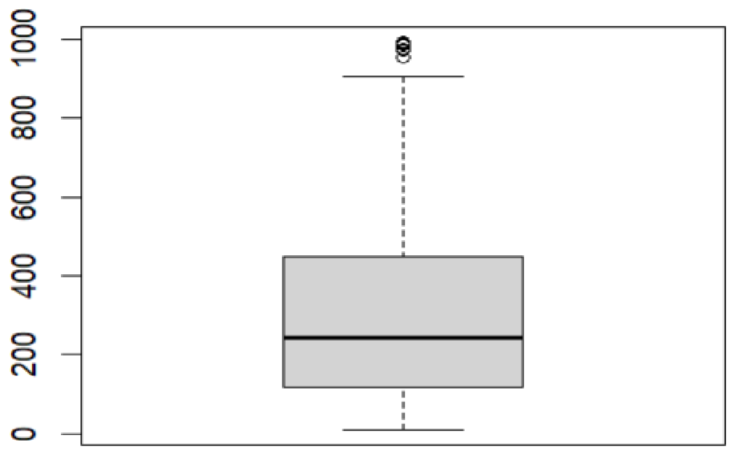
3. Boxplot(data$Tax.5.)



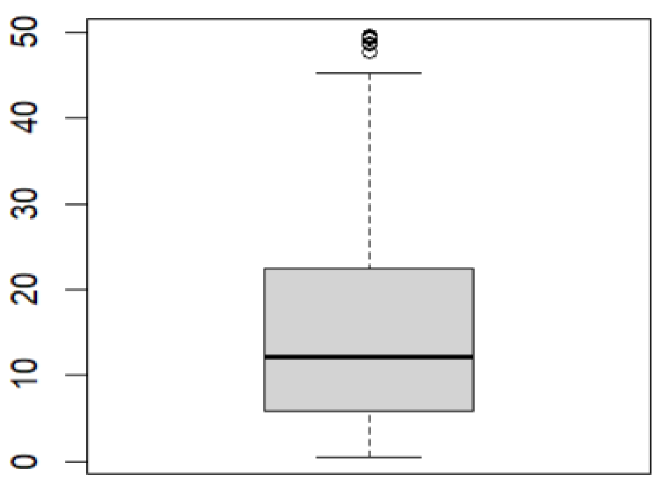
4. Boxplot(data$Total)



5. Boxplot(data$cogs)



6. Boxplot(data$gross.income)



Data analytics through models

A. Clustering

Implementing K-means clustering for the Gender, gross income and rating :

- Use the gender column as categorical data. And using this column we cluster the gross income and rating columns. Blue one represents the male and red one represents the female.

- Above figure shows the scatter plot of below code

```
data <- read.csv("D:\\education\\year2\\Assigenment in r2.csv")
head(data)
library(ggplot2)
ggplot(data,aes(gross.income,Rating))+geom_point(aes(col=Gender),size = 3)
```

- Applying clustering Algorithm for the Gender

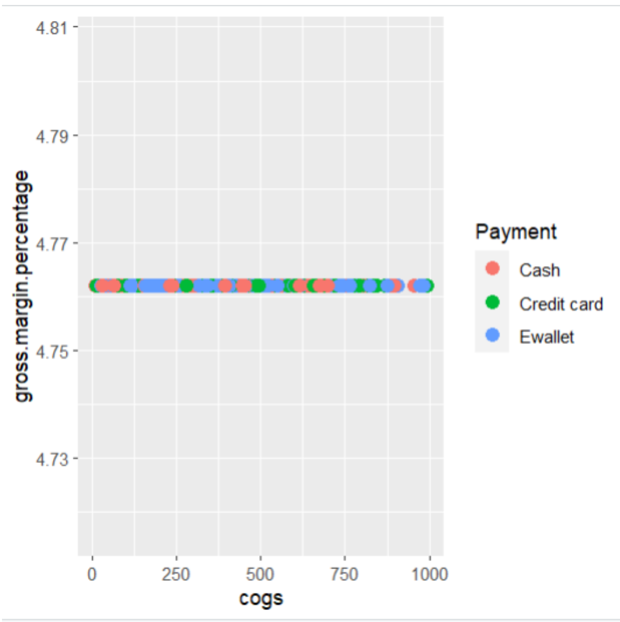
```
data <- read.csv("D:\\education\\year2\\Assigenment in r2.csv")
head(data)
library(ggplot2)
ggplot(data,aes(gross.income,Rating))+geom_point(aes(col=Gender),size = 3)

#k means cluster
set.seed(150)
cluster_result <- kmeans(data[ ,15:16],centers = 3, nstart = 25)
cluster_result

table(cluster_result$cluster,data$Gender)
```

```
> table(cluster_result$cluster,data$Gender)

    Female Male
  1     86   83
  2    168  144
  3    247  272
```

- The table shows the cluster result of gender through gross income and rating. Basically its divided in to 3 main cluster points as male and female.

Implementing K-means clustering for the payment , cogs and gross margin percentage :

- Use the payment column as categorical data. And using this column we cluster the cogs and gross margin percentage columns. Blue one represents the Ewallet and red one represents the cash and green one represents the credit cards.



- Above figure shows the scatter plot of below code

```
data <- read.csv("D:\\education\\year2\\Assigenment in r2.csv")
head(data)
library(ggplot2)
ggplot(data,aes(cogs,gross.margin.percentage))+geom_point(aes(col=Payment),size = 3)
```

- Applying clustering Algorithm for the Payment

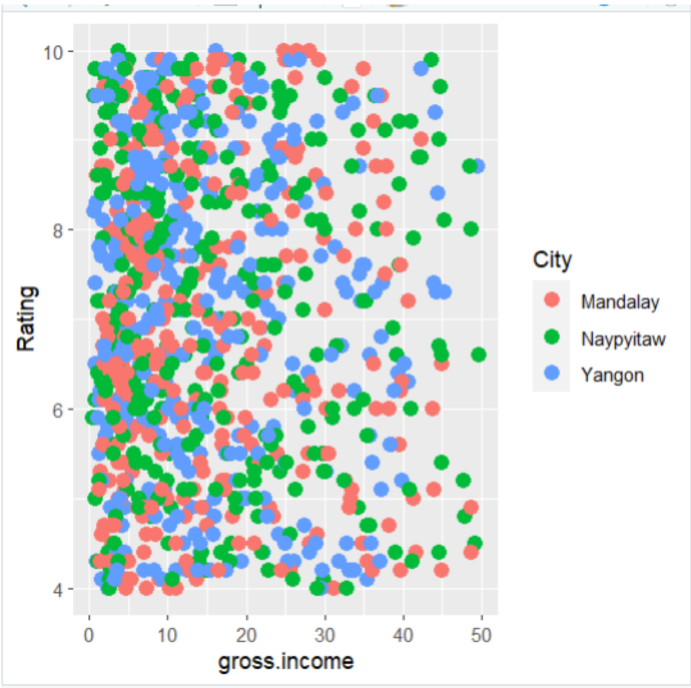
```
#k means cluster
set.seed(150)
cluster_result <- kmeans(data[ ,14:15],centers = 3, nstart = 25)
cluster_result

table(cluster_result$cluster,data$Payment)
```

```
> table(cluster_result$cluster,data$Payment)

    Cash Credit card Ewallet
  1  175         166     178
  2  110          87     115
  3   59          58      52
```

- The table shows the cluster result of gender through cogs and gross margin percentage. Basically its divided in to 3 main cluster points as cash , credit card and ewallet.

Implementing K-means clustering for the City , gross income and rating :

- Use the city column as categorical data. And using this column we cluster the gross income and rating columns. Blue one represents the Yangon and red one represents the Mandalay and green one represents the Naypyitaw.



- Above figure shows the scatter plot of below code

```
data <- read.csv("D:\\education\\year2\\Assigenment in r2.csv")
head(data)
library(ggplot2)
ggplot(data,aes(gross.income,Rating))+geom_point(aes(col=City),size = 3)
```

- Applying clustering Algorithm for the City

```
#k means cluster
set.seed(150)
cluster_result <- kmeans(data[ ,15:16],centers = 3, nstart = 25)
cluster_result

table(cluster_result$cluster,data$City)
```

```
> table(cluster_result$cluster,data$City)

    Mandalay Naypyitaw Yangon
  1       56        63     50
  2      101       103    108
  3      175       162    182
```

- The table shows the cluster result of gender through gross income and rating. Basically its divided in to 3 main cluster points as Mandalay , Naypyitaw and Yangon.

B. <u>Linear Regression</u>

- Firstly, write the code to the create plot of linear regression and secondly write code of finding coefficient value of each plot of linear regression.

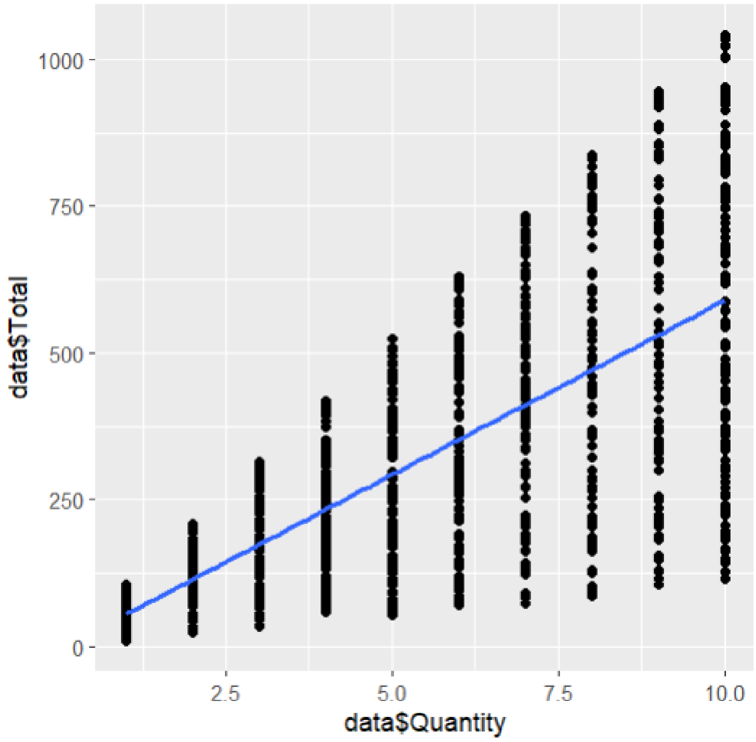
```r
library(ggplot2)
library(dplyr)


data <- read.csv("D:\\education\\year2\\Assigenment in r2.csv")


summary(data)
str(data)
ggplot(data = data,aes(x=data$Quantity,y=data$Total))+geom_point()+geom_smooth(method = lm, se=FALSE)

regmodel1 <- lm(Total ~ Quantity,
                data=data)
coef(regmodel1)



ggplot(data = data,aes(x=data$Unit.price,y=data$gross.income))+geom_point()+geom_smooth(method = lm, se=FALSE)

regmodel1 <- lm(gross.income ~ Unit.price,
                data=data)
coef(regmodel1)



ggplot(data = data,aes(x=data$Total,y=data$Tax.5.))+geom_point()+geom_smooth(method = lm, se=FALSE)

regmodel1 <- lm(Tax.5. ~ Total,
                data=data)
coef(regmodel1)


ggplot(data = data,aes(x=data$Quantity,y=data$cogs))+geom_point()+geom_smooth(method = lm, se=FALSE)

regmodel1 <- lm(cogs ~ Quantity,
                data=data)
coef(regmodel1)
```

1. Prediction total sales based on Quantity

- The below graph shows the prediction of total sales based on quality .
- In the scatter plot,
  X - Quantity
  Y – Total
- The coefficient value is representing as below figure .

```
> coef(regmodel1)
(Intercept)    Quantity
  -3.993328   59.339397
```

- The linear regression graph is represented as in the below figure.



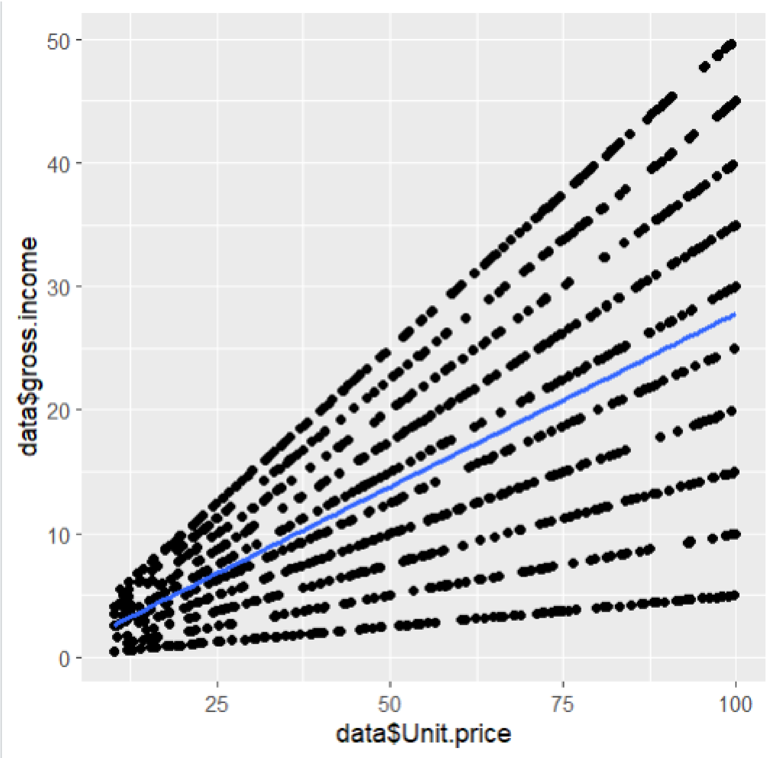
- The blue straight line represents the best fitted line of graph.

2. Analyzing the impact of Unit price on Gross income

- The below graph shows the analyzing the impact of unit price on gross income.
- In the scatter plot,
  X - Unit price
  Y - Gross income
- The coefficient value is representing as below figure

```
> coef(regmodel1)
(Intercept)  Unit.price
 -0.2181896   0.2801682
```

- The linear regression graph is represent as below figure.

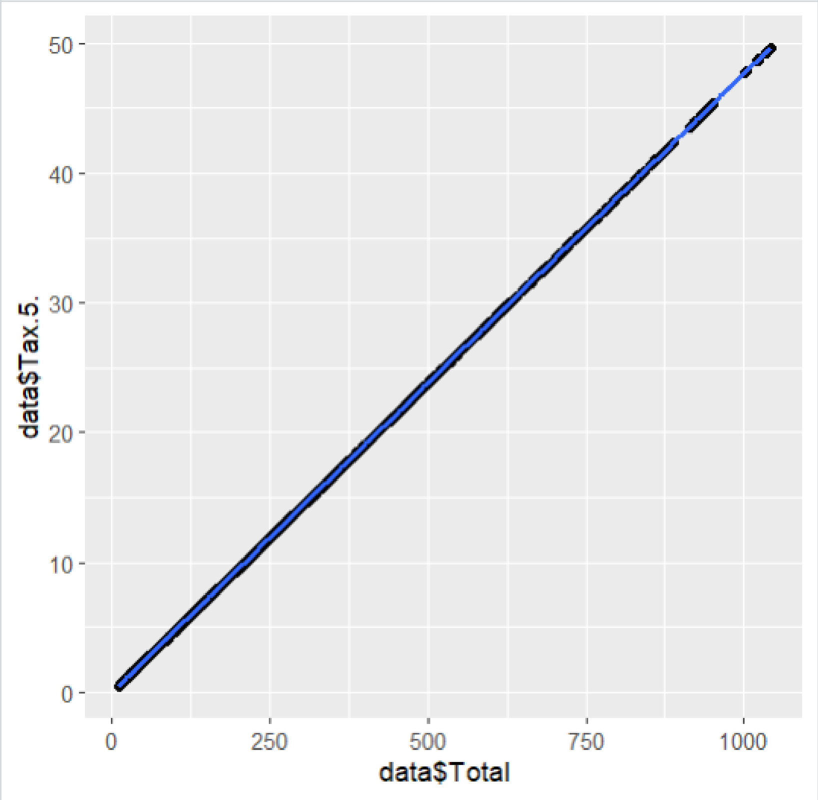


3. Predicting Tax amount based on Total sales

- The below graph shows the predicting tax amount based on total sales.
- In the scatter plot,
  X -Total
  Y - Tax 5%
- The coefficient value is representing as below figure

```
> coef(regmodel1)
  (Intercept)         Total
-6.111659e-14  4.761905e-02
```

- The linear regression graph is represent as below figure.

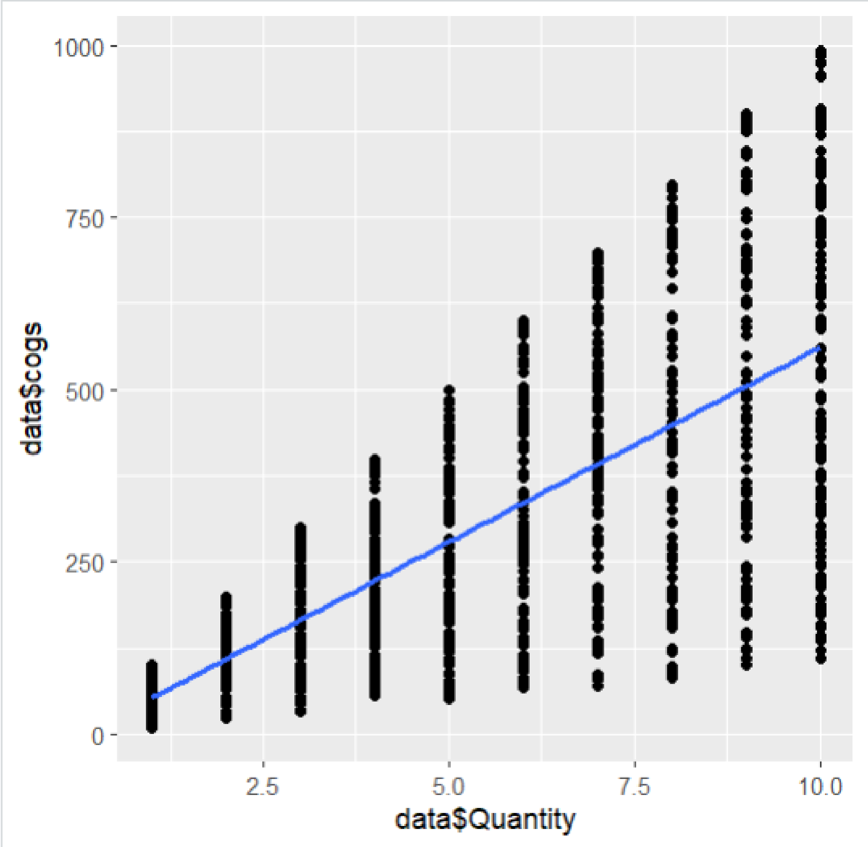


4. Prediction Cogs (Cost of Goods Sold) based on Quantity

- The below graph shows the prediction cogs (Cost of Goods Sold)based on quantity
- In the scatter plot,
  X -Quantity
  Y - Cogs
- The coefficient value is representing as below figure

```
> coef(regmodel1)
(Intercept)    Quantity
   -3.80317    56.51371
```

- The linear regression graph is represent as below figure.


1000
750
500
250
0
data$cogs
2.5
5.0
7.5
10.0
data$Quantity

C. Decision Tree Classifier

Overview:

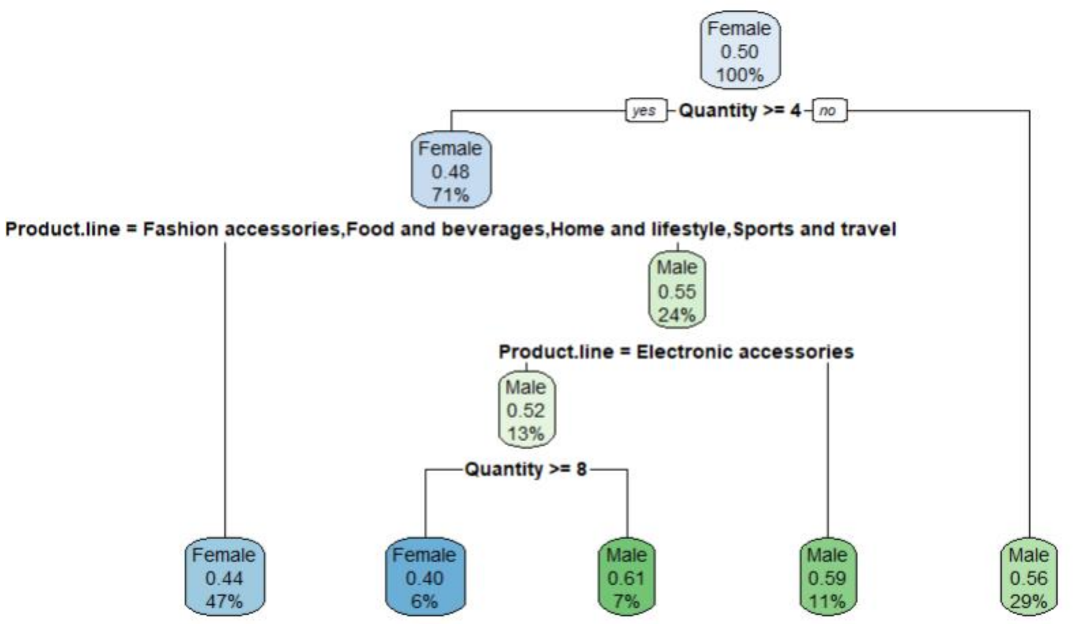
- Decision Tree Classifier is a modelling type used with datasets. With decision trees it's easy to understand the classification by having an initial glance. We can clearly see the flow of the tree. Also decision trees can handle both numerical data and categorical data which is a good advantage. Here in the dataset decision tree modeling was used for predictions based on different variables. Libraries such as rpart, rpart.plot were used in this classification.

Decision Tree 1

Code and Output:

```
library(rpart)
library(rpart.plot)
predata <- read.csv("D:\\Downloads\\sp\\supermarket_sales - Sheet1.csv")
predata
tree <- rpart(Gender ~Product.line+Quantity,predata)
rpart.plot(tree)
```

- This decision tree was used to predict the gender based on the product line and quantity purchased.



Explanation:

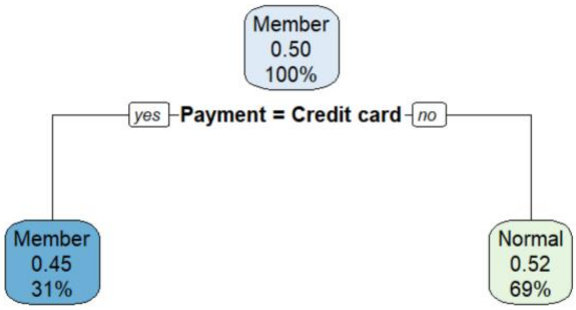
- When referring to the code, first of all the libraries and the dataset is loaded into R. Then a variable is created named "Tree" with the code for the tree. Here the columns used to predict the gender is included in the code. At last the tree is plotted using the rpart.plot function.
- In the first root node, all the instances are considered as 1 with a majority of females with the probability of 0.501 and males with the probability of 0.499. In the second node, the quantity is considered and the females have a higher probability. When moving to the fourth and fifth nodes, the decision tree further splits based on product line of purchase. For the product lines fashion accessories, food and beverages, home and lifestyle or sports and travel it has a prediction of females as the majority and the rest of the product lines electronic accessories or health and beauty it has predicted male as the majority. Further node 5 splits based on product line which is electronic accessories based on quantity. Here if the quantity is greater than 8 it has predicted males as the majority. So we can understand that female customers of this supermarket mostly buy products related to fashion ,food ,home and sports and travel. On the other hand males most of the time buy products related to electronics and health.

Decision Tree 2

Code and Output:

```
library(rpart)
library(rpart.plot)
predata2 <- read.csv("D:\\Downloads\\sp\\supermarket_sales - Sheet1.csv")
predata2
tree2 <- rpart(Customer.type ~Payment,predata2)
print(tree2)
rpart.plot(tree2)

```

- This decision tree was used to predict the customer type based on the payment method.



Explanation:

- When referring to the code, first of all the libraries and the dataset is loaded into R. Then a variable is created named “Tree2” with the code for the tree. Here the column “Payment” which is used to predict the customer type is included. At last the tree is plotted using the rpart.plot function.
- In the first root node, all the instances are considered as 1 with a majority of females with the probability of 0.501 and males with the probability of 0.499. In the second node credit card was considered as the payment method and majority were members. In the third node cash or e-wallet was considered as the payment method and majority were normal customers without membership. So the model has predicted that if the payment method is credit card, the customer is a member otherwise if the payment method is cash or e-wallet, the customer is not a member.

D. Naive Baye’s Classifier

Overview:

- Naive Baye’s Classifier is a modelling type used with datasets. This classifier is used for predictions as it’s simple and understandable. This classifier is computationally efficient when dealing with massive datasets. Naïve Baye’s classifier suits well with categorical data with minimum preprocessing. Here in the dataset Naive Baye’s Classifier was used for predictions based on different variables. Libraries such as naivebayes,mlbench,e1071,caret,caTools were used in this classification.

Naive Baye’s Classifier - 1

Code and Output:

```
library(naivebayes)
library(mlbench)
library(e1071)
library(caret)
library(caTools)
sp <- read.csv("D:\\Downloads\\sp\\supermarket_sales - Sheet1.csv")
sp
set.seed(200)
splitg <- sample.split(sp$Gender, SplitRatio = 0.75)
traindatag <- subset(sp, split_ratio == TRUE)
testdatag <- subset(sp, split_ratio == FALSE)
#naive baye's classifier
Naivegender <- naiveBayes(Gender ~ Product.line + Quantity + Payment + Customer.type, data = traindatag)
prediresultsg <- predict(Naivegender, newdata = testdatag)
conmatrixgender <- table(prediresultsg, testdatag$Gender)
accugender <- mean(prediresultsg == testdatag$Gender)
table(prediresultsg)
print(conmatrixgender)
print(accugender)
```

- This Naive Baye’s Classifier was used to predict the gender of the customers based on the product line , quantity, payment method and customer type.

```
> table(prediresultsg)
prediresultsg
Female   Male
   154    126
> print(conmatrixgender)

prediresultsg Female Male
       Female     79   75
       Male       55   71
> print(accugender)
[1] 0.5357143
```

Explanation:

- When referring to the code, first of all the required libraries and the dataset is loaded into R. Then the seed rate is set to 200 and the dataset is splitted based on the gender column with a split ratio of 0.75. Then the training dataset and the testing datasets are created for the model we are creating based on the supermarket dataset. Afterwards using the "Naivebayes" function is used required columns that are needed to make the prediction. The results of the model is inserted in a table and the confusion matrix is created to analyze how the model performed in predicting the gender.
- Here as in the confusion matrix the model has mis predicted 55 instances as Female where the actual gender was Male and on the other hand the model has mis predicted 75 instances as Male where the actual gender was Female. Talking about the sensitivity, true positive rate is 51.3% and the true negative rate is 56.3%. Overall accuracy of the model is 53.57% which is slightly above marginal. So what we can conclude is the model is performing well in a certain way but not too accurate considering the accuracy we got when predicting the gender.

Contribution

1) 10899479 (Kekulawala) – Did data exploration part and data cleaning part.

2) 10899280 (Adikaram Adikar) – Created report and Did Analysis and Visualization part. (pie charts, bar charts, histograms, scatterplots)

3) 10899495(Weerasiri) – Did classification and prediction part (clustering, linear regression) and done descriptive statistics part (mean, mode, median, range,box plots)

4) 10899488(Senanayaka Senanayake) – Did classification and prediction part (naïve bayes, decision tree)