25-01-24

TASK - 01

Installation of R on windows :

1. To install R, go to cran.r-project.org
2. Click download R to for windows
3. Install R click on install R for the first time.
4. Click download R for windows. Open the downloaded file.
5. Select the language you would like to use during the installation. Then click OK.
6. Click Next
7. Select where you would like R to be installed It will default to your program files on your c drive. Click Next.
8. You can then choose which installation you would like
9. Then specify if you want to customize your startup or just use the default. Then click Next.
10. Then you can choose the folder that you want R to be saved or the default if the R folder that was created. Once you have finished, click Next.
11. You can then select additional shortcuts if you would like. Click Next.
12. Click finish.

13. Next, download Rstudio. Go to https://postit.co/downloads

14. Click Download Rstudio.

15. Once the packet has downloaded, then welcome to Rstudio setup wizard will open. Click Next and go through the installation steps.

16. After the setup wizard finishing the installation. Rstudio will open.

Data types in R -

- Numeric - (10.5, 55, 787)

code: x <- 10.5
class(x)

output: [1] "numeric"

- integer - (1L, 55L, 100L, where "L" declares it as integer)

code: x <- 1000L
class(x)

output: [1] "integer"

- complex - (9+3i, where "i" is the imaginary part)

code: x <- 9i + 3
class(x)

output: [1] "complex"

- character - ("k", "R is exciting")

code :
```
x <- "R is exciting"
class(x)
```

output : [1] "character"

- logical - (TRUE or FALSE)

code :
```
x <- TRUE
class(x)
```

output : [1] "logical"

control statements -

- if condition -

syntax :
```
if (expression) {
    statements
    - - - - -
    - - - - -
}
```

code :
```
x <- 100
if (x>10) {
  print(paste(x, "is greater than 10"))
}
```

output : [1] 100 is greater than 10

12/2/2024

- if - else condition -

syntax :
```
if (expression) {
    statements
    - - - - -
    - - - - -
}
else {
    statements
    - - - - -
    - - - - -
}
```

code :
```
x <- -5
if (x>10) {
    print(paste(x, "is greater than 10"))
} else
  {
  print(paste(x, "is less than 10"))
}
```

output : [1] " 5 is less than 10"

- for loop -

syntax :
```
for (value in vector) {
    statements
    - - - - - -
    - - - - - -
}
```

code :
```
x <- letters[4:10]
for (i in x) {
  print(i)
}
```

output : [1] "d"

[1] "e"

[1] "f"

[1] "g"

[1] "h"

[1] "i"

[1] "j"

- While loop -

Syntax - while (expression) {

statements

- - - - -

- - - - -

}

code: x = 1

while (x <= 5) {

print(x)

x = x + 1

}

output: [1] 1

[1] 2

[1] 3

[1] 4

[1] 5

- Repeated loop -

Syntax: repeat {

statements

- - - - -

- - - - -

if (expression) {

```
    break
    }
    }
```

code :

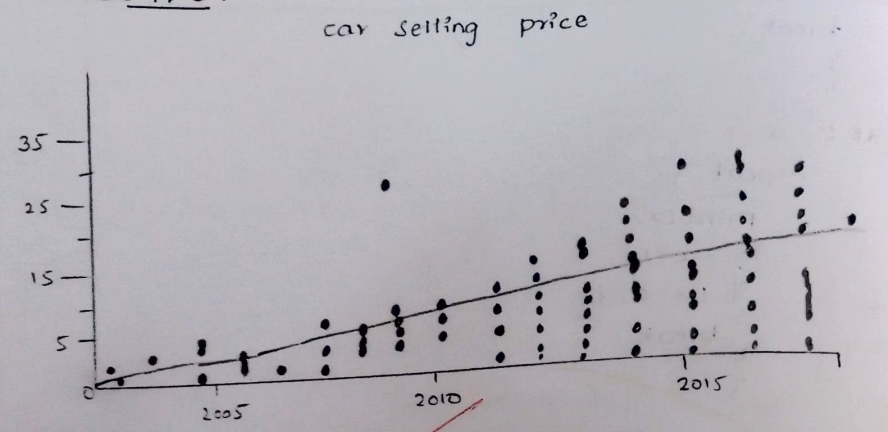
```
x = 1
repeat {
    print(x)
    x = x + 1
    if (x > 5) {
      break
    }
    }
```

output :

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

12/2/24

OUTPUT -



12/2/24

TASK - 02

ACQUIRING AND PLOTTING DATA

Algorithm -

step 1 : create a new folder and download datasets

step 2 : open R studio and install any necessary packages.

step 3 : open the folder in R studio

step 4 : Load the packages you installed.

step 5 : Acquire your data using functions like read_csv()

step 6 : Before plotting, explore and understand your data using functions like head(), summary() and str()

step 7 : create plots using ggplot2 or other plotting libraries based on your requirements.

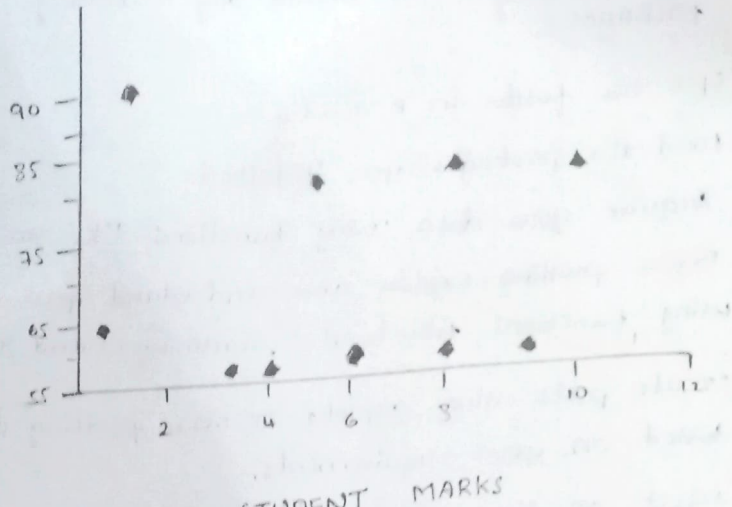
step 8 : select all line and run the program.

step 9 : End

a. program - car selling price

```
install.packages("readxl")
library(readxl)
car_data <- read_excel("D:/DVT LAB/car program/car data")
view(car_data)
X <- car_data$ year
Y <- car_data$ selling_price
plot(X,Y, main = "car selling price",
```

OUTPUT -



12/2/24


```
xlab = "car", ylab = "price",
pch = 19, frame = FALSE)
abline (lm(y~x, data = mtcars), col = "blue")
```

b. program - student marks

```
install.packages ("readxl")
library (readxl)
STUDENT_MARKS_DATA <- read_excel ("D:/DVT LAB/car program
                                  /STUDENT MARKS DATA.xlsx")
View (STUDENT_MARKS_DATA)
X<- STUDENT_MARKS_DATA$S.NO
Y<- STUDENT_MARKS_DATA$SPM

plot (x,y, main = "STUDENT MARKS",
      xlab = "STUDENT", ylab = "MARKS",
      pch = 19, frame = FALSE)
abline (lm(y~x, data = STUDENT_MARKS_DATA), col = "GREEN")
```

RESULT - The program was executed successfully.

15/2/24

22/02/24

TASK - 3

a. STATISTICAL ANALYSIS - SUCH AS MULTIVARIATE ANALYSIS PCA

The R programming language provides a wide range of packages and functions for statistical analysis. Here's an overview of how to perform various statistical analyses using R, along with example programs and interpreting the results.

Multivariate Analysis :

Principal component Analysis (PCA): PCA is used to identify the most important features or patterns in a dataset. The prcomp() function in base R or the prcomp() function from the stats package can be used. Example:

Syntax

```
pca_result <- prcomp(data)
summary(pca_result)
```

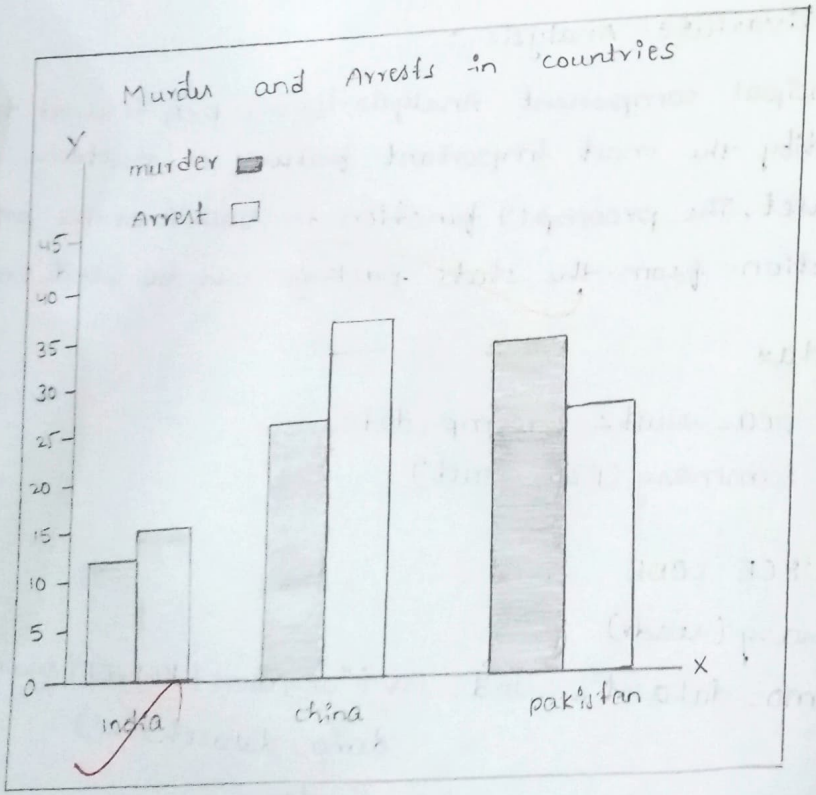
SOURCE CODE

```
library(readr)
demo_dataset <- read_csv("C:/Users/KGRCET/Downloads/
                          demo_dataset.csv")

murder <- demo_dataset$murder
arrest <- demo_dataset$arrest
country = demo_dataset$country
```

OUTPUT :

Importance of components :

	PC1	PC2
Standard deviation	1: 24420.6722	
Proportion of variance	0.77411.2259	
Cumulative proportion	0.77411.0000	




```
# CONVERTING THE DATA IN FRAMES
data <- data.frame (category = country.murder = murder,
                                    arrest = arrest)
twight = t(data[ c("murder", "arrest")])
beside = TRUE
col = c("blue", "green"),
names.arg = data $ category,
main = paste ("reporting "murder and Arrests in countries"),
xlab = "country",
ylab = "Rank",
legend.text = c ("murder", "arrest"),
args.legend = list (x = "topleft", bty = "n"))

# APPLYING THE PCA
numeric_data <- data [supply (data, is numeric)]
pca_result <- prcomp (numeric_data, scale = TRUE)
summary (pca_result)
```

RESULT — The program was compiled and executed succesfully.

29/2/24

TASK - 3B

statistical Analysis - such as Multivariate Analysis, PCA, LDA, correlation regression and analysis of variance.

Source code :

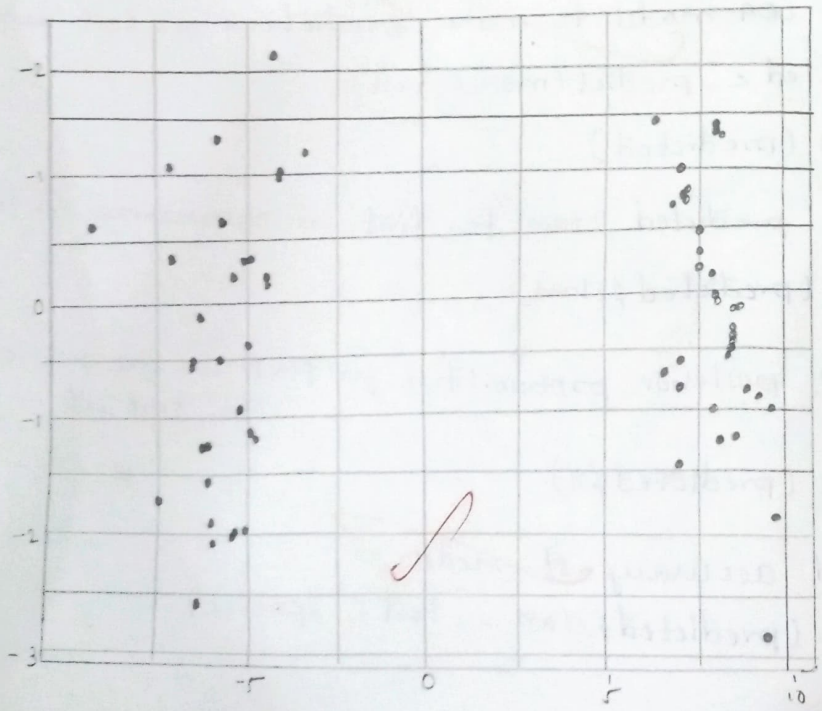
```
#Load the required library
library (MASS)
library (ggplot2)

#load the dataset
 attach (iris)

#view structure of dataset
str (iris)

#scale each predictor variable (i.e first 4 columns)
iris [1:4] <- scale (iris [1:4])

#find mean of each predictor variable
apply (iris [1:4], 2, mean)

#find standard deviation of each predictor variable
apply (iris [1:4], 2, sd)

# make this example reproducible
 set. seed (1)
```

```
#use 70% of dataset as training set and remaining
  30% as testing set

sample <- sample (c(TRUE, FALSE), nrow (iris), replace = TRUE,
                                    prob = c (0.7, 0.3))

train <- iris [sample,]

test <- iris [! sample,]

#fit LDA model
model <- lda (species ~, data = train)

#view model output
model

#use LDA model to make predictions on test data
predicted <- predict (model, test)

names (predicted)

#view predicted class for first six observations in test set
head (predicted $class)

#view posterior probabilities for first six observations
                                    in test set

head (predicted $x)

#find accuracy of model
mean (predicted $class == test $ species)
```

OUTPUT :

#view linear discriminants for first six observations in test set.

	LD1	LD2
4	7.150360	0.7177382
6	7.961533	-1.4839408
7	7.504033	-0.2731178
15	10.170378	-1.9859027
17	8.885168	-2.1026494
18	8.113443	-0.7563902




```
#define data to plot
lda_plot <- cbind(train, predict(model)$x)

# creat plot
ggplot(lda_plot, aes(LD1, LD2)) +
geom_point(aes(color = species))
```

RESULT - The program was compiled and executed completely.

14/3/24

OUTPUT :

```
>clusters_cols
mpg cyl disp hp drat wt qsec vs am gear carb
  1   2    2  2    1  2    3  3  1    2
>cluster_rows
Mazda RX4 Mazda RX4 wag Datsun 710 Hornet 4 Drive
        1             1          2              2
Hornet Sport Valiant Duster 360 Merc 240D
           3       2          2
Merc 230 Merc 280 Merc 280C Merc 450SE
       2        2         3
Merc 450SL Merc 450SLC Cadillac Fletwood
         3          3              3
Chrysler Imperial Fiat 128 Honda Civic Toyota Corolla
                3       2  2          2
Toyota Corona Dodge Challenger AMC Javelin Camaro Z28
            2               3           3          3
Pontiac Firebird Fiat X1-9 Porsche 914-2 Lotus Europa
               3         2  2           2
Ford Pantera L Ferrari Dino Maserati Bora Volvo 142E
             1           1  1           2
```

13/4/24

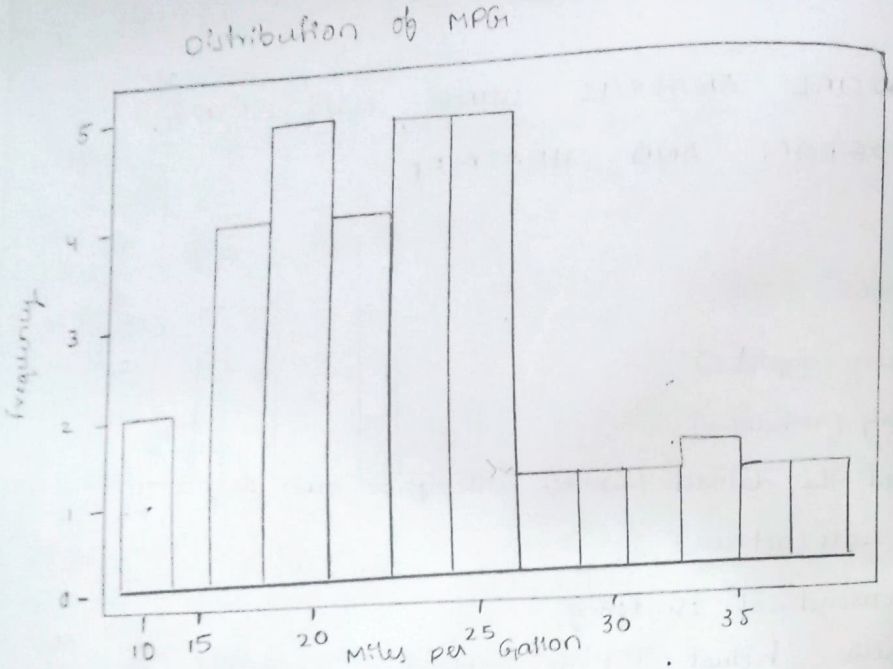
TASK - 04

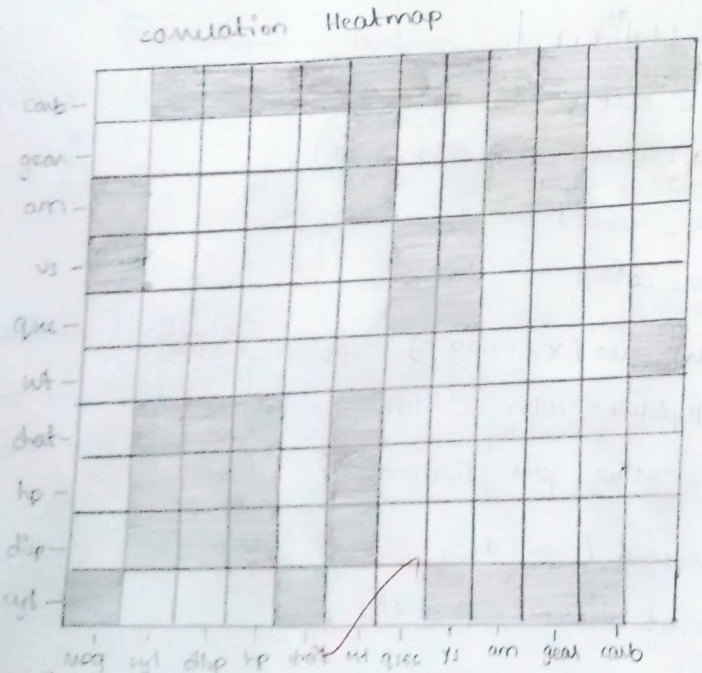
PAGE # 1

FINANCIAL ANALYSIS USING CLUSTERING, HISTOGRAM AND HEATMAP

SOURCE CODE -

```
library(ggplot2)
library(reshape2)
#load the dataset (replace with your own data)
df <- scale(mtcars)
# Hierarchical clustering
hc_rows <- hclust(dist(df), method = "complete")
hc_cols <- hclust(dist(t(df)), method = "complete")
clusters_rows <- cutree(hc_rows, k=3)
clusters_cols <- cutree(hc_cols, k=3)
View(clusters_rows)
View(clusters_cols)
ggplot(mtcars, aes(x=mpg)) + geom_histogram(binwidth
=2, fill = "skyblue", color = "black") + labs(title = "Distribution
of MPG", x = "Miles per Gallon", y = "Frequency")
cor_df <- round(cor(df),2)
melted_cormat <- melt(cor_df)
```






```
ggplot(melted_cormat, aes(x=Var1, y=Var2, fill=value))
+ geom_tile() + scale_fill_gradient(low="white", high=
"red") + labs(title="correlation Heatmap",x="",y="")
clusters_rows <- cutree(hc_rows, k=3)
clusters_cols <- cutree(hc_cols, k=3)
```

Result - The program was compiled and executed successfully.

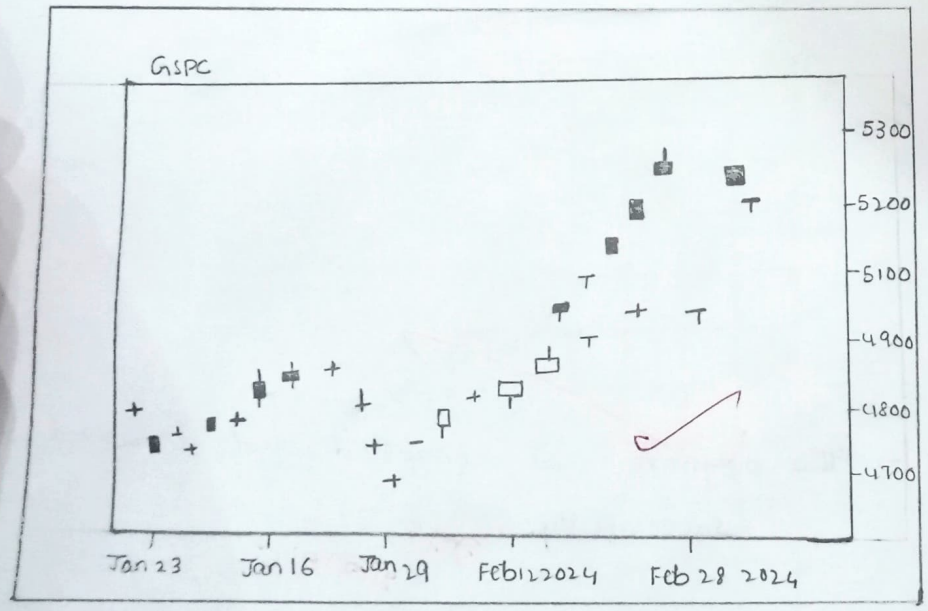
21/3/24.

OUTPUT -

head (GSPC)

GSPC

2024-01-02	4746.20	4754.33	4722.64	742.83	3743050000
2024-01-03	4745.07	4729.29	4699.71	4704.81	3950760000
2024-01-04	4697.42	4728.78	4688.63	4688.63	3715480000
2024-01-05	4690.57	4721.49	4697.24	4697.24	3834437000
2024-01-08	4703.70	4764.54	4699.82	4763.54	3742320000
2024-01-09	4741.93	4768.47	4730.35	4756.50	3629960004



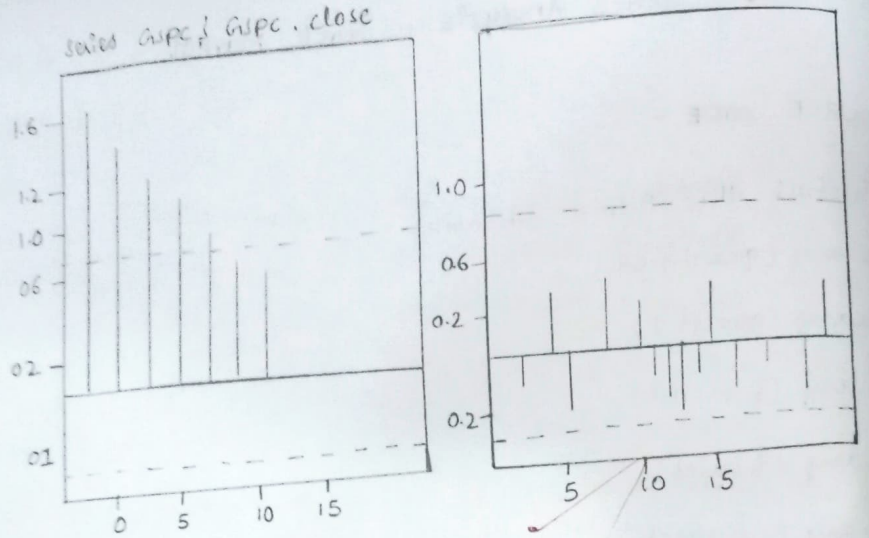
21/7/24

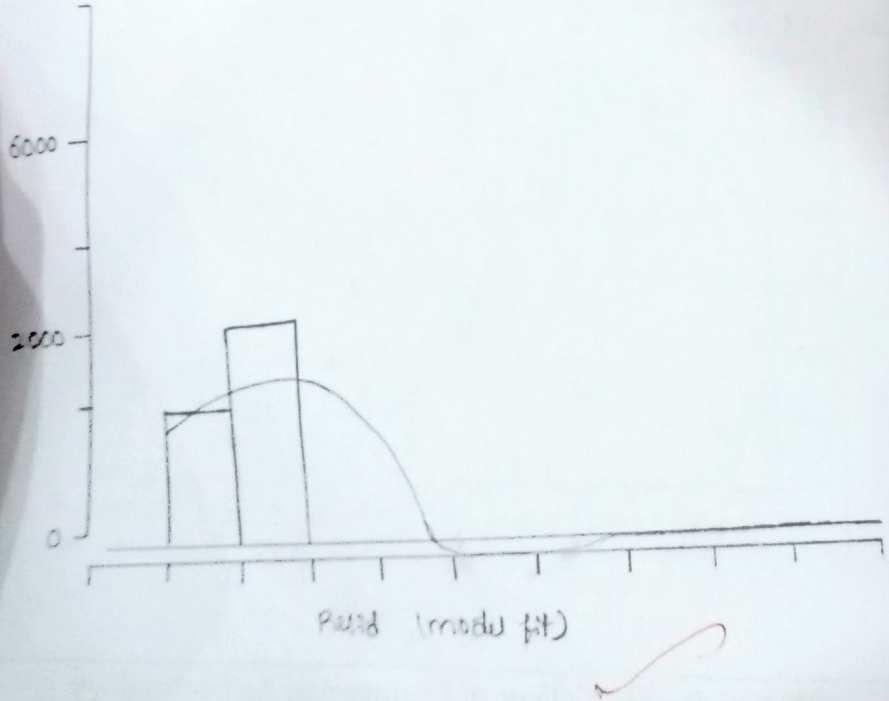
PAGE # 13

5. Time - series Analysis - Stock Market

SOURCE CODE -

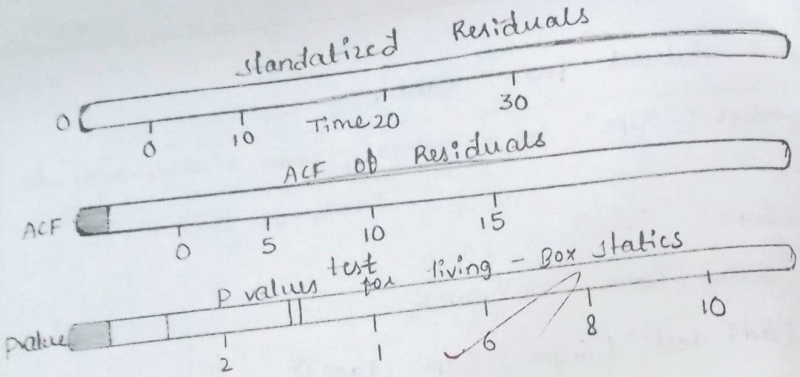
```
#install the Below libraries
library(quantmod)
library(ggplot2)
library(forecast)
library(tseries)
library(rugarch)
library(prophet)
library(tsfknn)
```

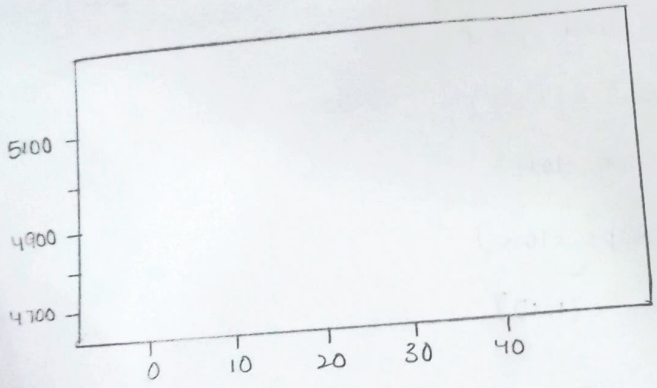


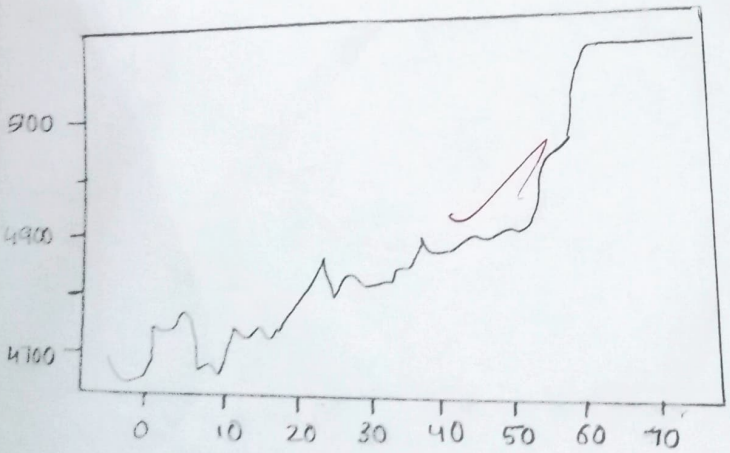


2/3/24


```
#import dataset from yahoo
get symbol ("GSPC", src = "yahoo", from = "2024-0-01", to
                                        "2024-02-29")
head (GSPC)
chart series (GPSC, TA = NULL)
Print (adf test (GSPC $ GSPC.class))

## plot ACF and PACF
Par (m flow = c{1,2})
acf ( GSPC $ GSPC.close)
pact (GSPC $ GSPC.close)
Par (m flow = c (1,1))
```







21/3/24

PAGE # 15

Histogram of Residual of Normality assumption

```
hist (resid (model fit). freq = F, ylim = c(0,0.8))
                    main = "Histograms" of Residuals")
c = Resid (model fit)
curve (dnorm (x, mean = mean (c), sd = sd(c)),
      dd = True, col = "darked")
```

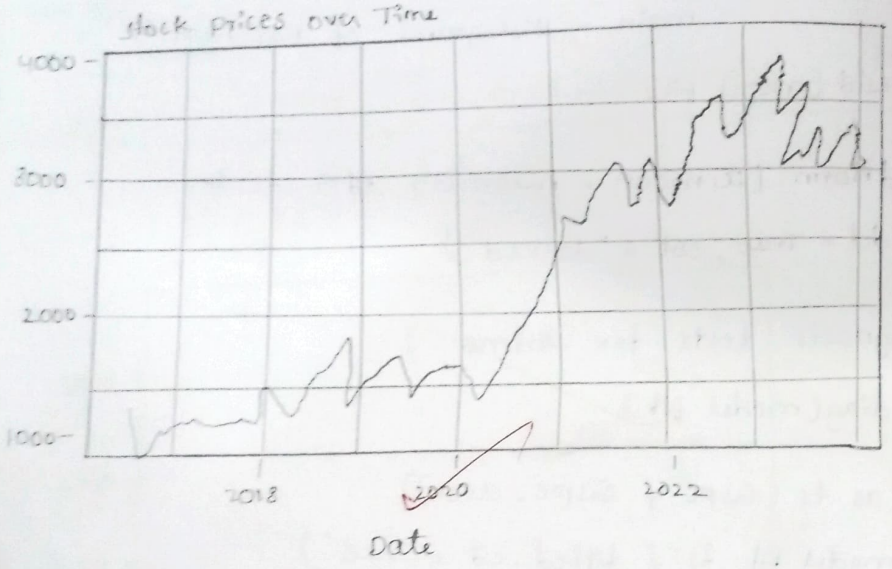
Diagnosis tests for Arima

```
ts diag(model fit)
```

```
Plot (as.ts (GSPC $ GSPC.close))
lines (model fit is $ fitted.cd = "red")
```

RESULT - The program was compiled and executed successfully.

28/3/24

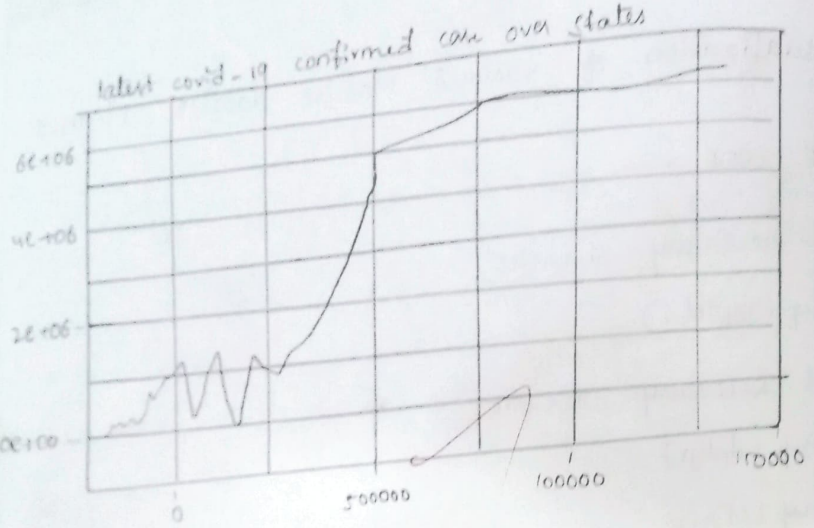


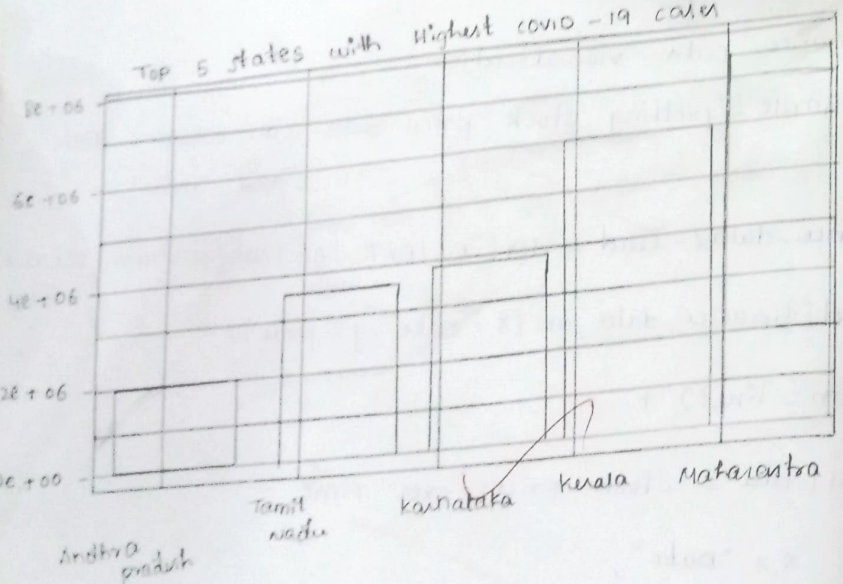
28/3/24

6. Visualization of various Massive dataset - Finance

SOURCE CODE -

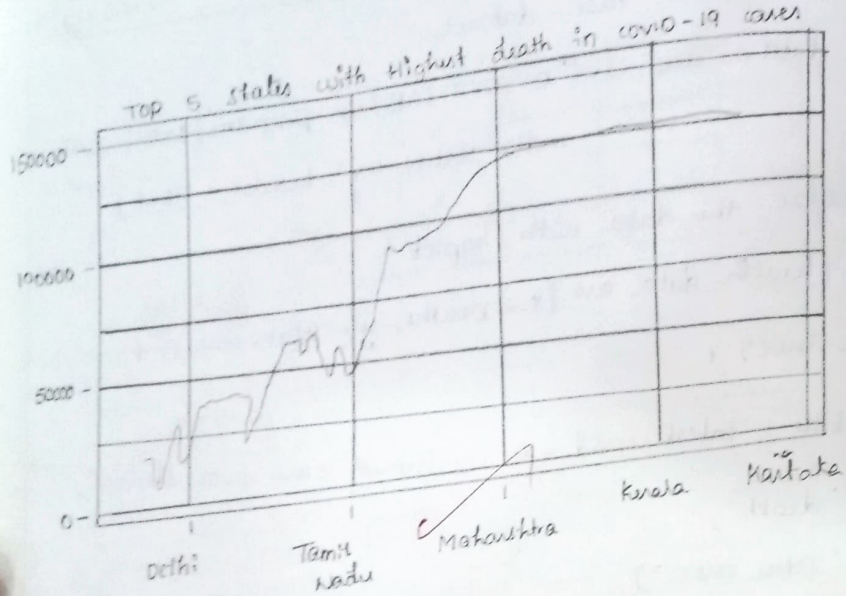
```
# Load necessary libraries
library (ggplot2)
# Load necessary libraries
library (dplyr)
library (sf)
# Finance Data visualization
# Example : plotting stock price data with columns 'Date' and 'price'
finance-data2-read - xlsx ("D:/DVT LAB/car program/TCS.xlsx")
ggplot (finance-data, aes (x = Date, y = price)) +
geom - line() +
labs (title = "stock prices over Time",
      x = "Date",
      y = "price")
```





28/3/24


```
# import health case dataset
health_data <- read.csv("D:/IBVT LAB/car program/latest covid-19
                         india status.csv", header = TRUE)

# visualize the data with ggplot
ggplot(health_data, aes(x = Deaths, y = Total.cases)) +
geom_line() +
labs(title = "latest covid-19 confirmed case over states",
     x = "death",
     y = "Total cases")

# total care over states
top_states <- head(health_data[order(health_data$Total.cases,
                                     decreasing = TRUE),],5)

# create a bar plot
ggplot(top_states, aes(x = reorder(State.UTs, Total.cases),
                       y = Total.cases)) +
geom_bar(stat = "identity", fill = "skyblue") +
labs(title = "Top 5 states with highest COVID-19 cases",
     x = "States",
     y = "Total cases") +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



2/3/24

PAGE # 18

top states over highest death

top_states <- head(health_data[order(health_data$Deaths, decreasing = TRUE),], 5)

create a bar plot

ggplot(top_states, aes(x = reorder(State.UTs, Deaths), y = Deaths)) +

geom_bar(stat = "identity", fill = "skyblue") +

labs(title = "Top 5 states with highest death in covid-19 cases",

x = "States",

y = "Total death") +

theme(axis.text.x = element_text(angle = 45, hjust = 1))

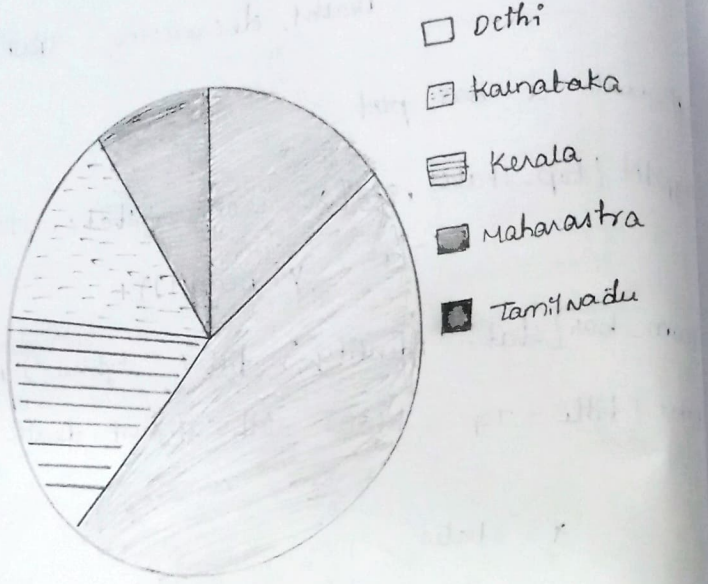
top 5 states with highest deaths

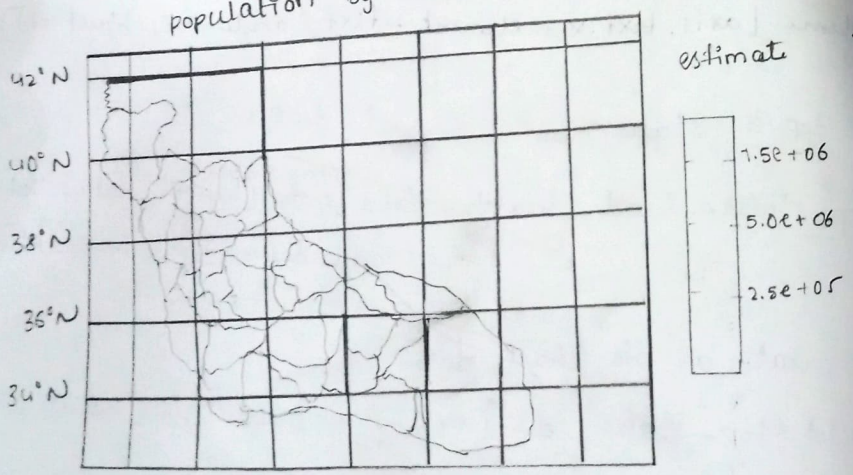
top_states <- head(health_data[order(health_data$Total.Cases, decreasing = TRUE),], 5)

create a pie chart for

ggplot(top_states, aes(x = "", y = Deaths, fill = State.UTs)) +

geom_bar(stat = "identity") +





28/3/24


```
word - polar (theta = "y") +
tabs(title = "Top 5 states with highest COVID 19 Deaths",
     X = NULL
     Y = NULL) +
theme_void()
# install necenary packages
install.packages(c("tidycensus", "sf", "ggplot2", "leaflet"))
library(tidycensus)
library(sf)
library(ggplot2)
## import data set
ca_counties <- get_acs(geography = "county", state = "CA",
                       variables = "B01003_001", geometry = TRUE)
ca_sf <- st_as_sf(ca_counties)
ggplot(ca_sf) + geom_sf(aes(fill = estimate)) +
scale_fill_viridis_c() +
tabs(title = "population by country in california")
leaflet(ca_sf) %>%
addTiles() %>%
addPolygons(fillColor = ~viridis::viridis(estimate),
            fillOpacity = 0.7,
color = "White", weight = 1, smoothFactor = 0.5,
label = ~paste0("country:", NAME, "<br>population:",
                estimate)) %>%
addLegend(position = "bottomright", pal = viridis::viridis(10), values
= ~estimate
title = "population")
```

RESULT - The above Task was executed successfully.

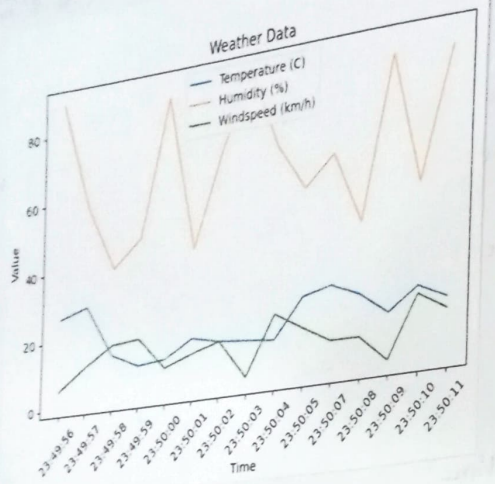
4/12/24

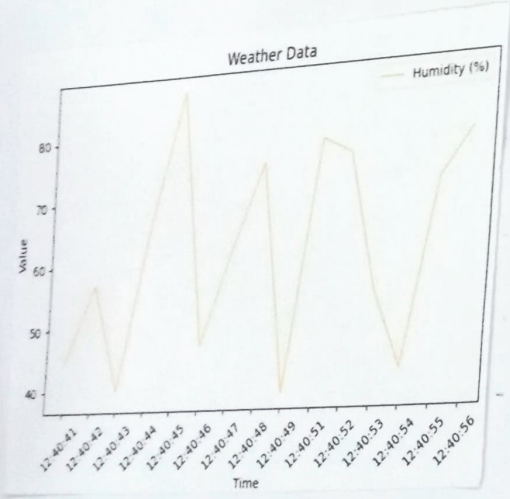
7. Visualization on streaming dataset (stock market, wheather, forecasting)

AIM : Visualization on streaming dataset whether forecasting.

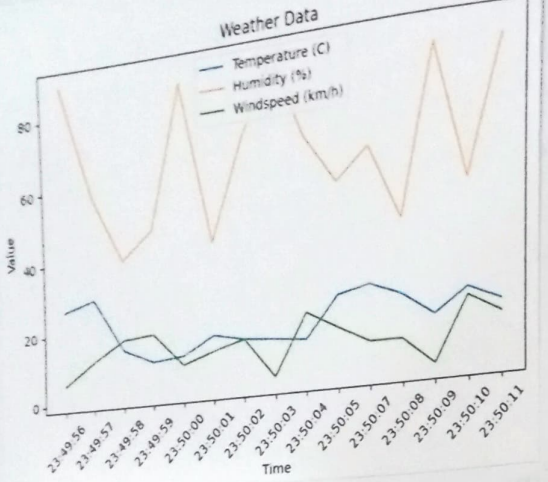
Software Requirements : Python IDE.

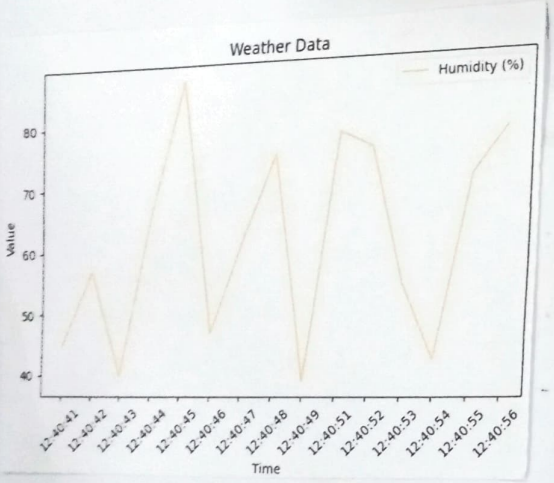
```
import csv
import time
import datetime
import random
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def main():
 timestamps, temperatures, humidities, windspeeds = [],[],[],[]

def update_plot(frame):
 try:
  current_time = datetime.datetime.now()
  date = current_time.strftime("%Y-%m-%d")
  time_now = current_time.strftime("%H:%M:%S")
  temp = round(random.uniform(10,30),2)
```



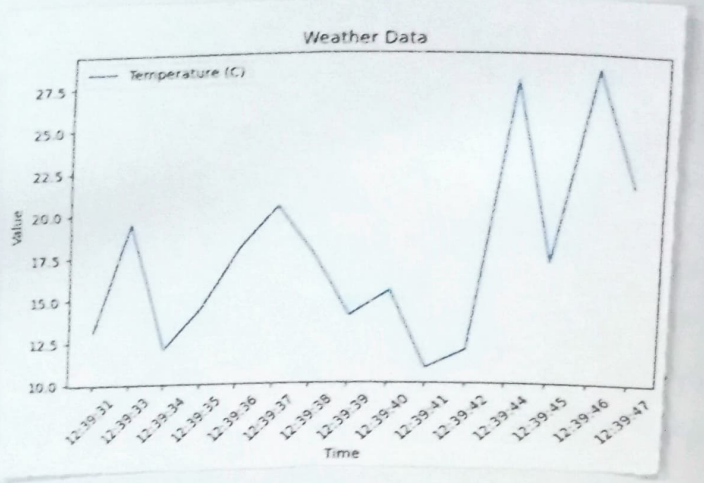


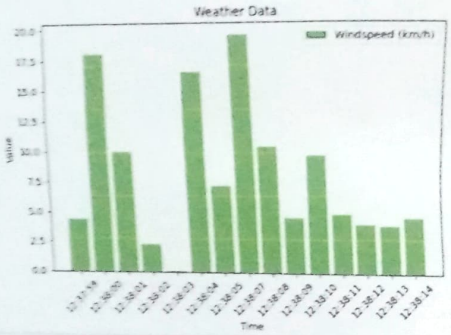

```
humidity = round(random.uniform(30,90),2)
windspeed = round(random.uniform(0,20),2)

timestamps.append(time_now)
temperatures.append(temp)
humidities.append(humidity)
windspeeds.append(windspeed)

if len(timestamps)>15:
 timestamps.pop(0)
 temperatures.pop(0)
 humidities.pop(0)
 windspeeds.pop(0)

#plotting
plt.clf()
plt.plot(timestamps, temperatures, label='Temperature (C)', color='blue',
                                                    alpha=0.7)
plt.plot(timestamps, humidities, label='Humidity (%)', color='orange',
                                                    alpha=0.7)

plt.plot(timestamps, windspeeds, label='Windspeed (km/h)', color='green',
                                                    alpha=0.7)

plt.xlabel('Time')
plt.ylabel('value')
plt.title('Weather Data')
```

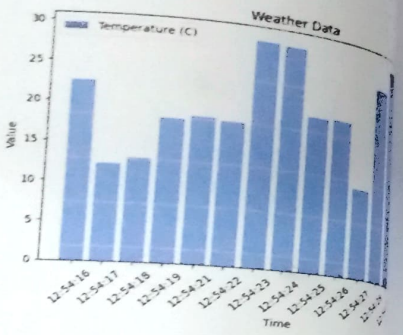


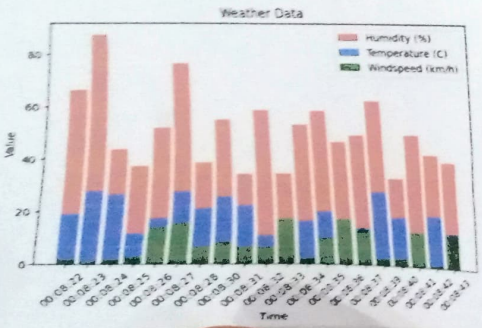


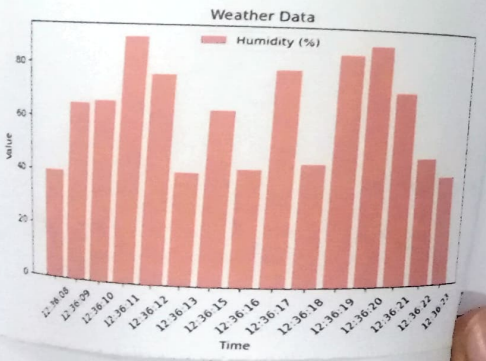

```
humidity = round(random.uniform(30,90),2)
windspeed = round(random.uniform(0,20),2)

timestamps.append(time_now)
temperatures.append(temp)
humidities.append(humidity)
windspeeds.append(windspeed)

if len(timestamps)>15:
 timestamps.pop(0)
 temperatures.pop(0)
 humidities.pop(0)
 windspeeds.pop(0)

#plotting
plt.clf()
plt.plot(timestamps, temperatures, label='Temperature (C)', color='blue',
                                                    alpha=0.7)
plt.plot(timestamps, humidities, label='Humidity (%)', color='orange',
                                                    alpha=0.7)

plt.plot(timestamps, windspeeds, label='Windspeed (km/h)', color='green',
                                                    alpha=0.7)

plt.xlabel('Time')
plt.ylabel('value')
plt.title('Weather Data')
```












```
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
except KeyboardInterrupt:
print("plotting stopped.")
exit()

fig,ax = plt.subplots()
ani = FuncAnimation(fig, update_plot, interval=1000)
plt.show()

if __name__ == "__main__":
main()

Replace # plotting for bargraph
plt.clf()
plt.bar(timestamps, humidities, label='Humidity(%)', color='red',
                                                    alpha=0.7)
plt.bar(timestamps, temperatures, label='Temperature(C)', color='blue',
                                                    alpha=0.8)
plt.bar(timestamps, windspeeds, label='Windspeed(km/h)', color='gree
                                                    alpha=0.9)
plt.xlabel('Time')
plt.ylabel('value')
plt.title('weather data')
```

```
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
```

RESULT - The program was executed succesfully.

18/4/24

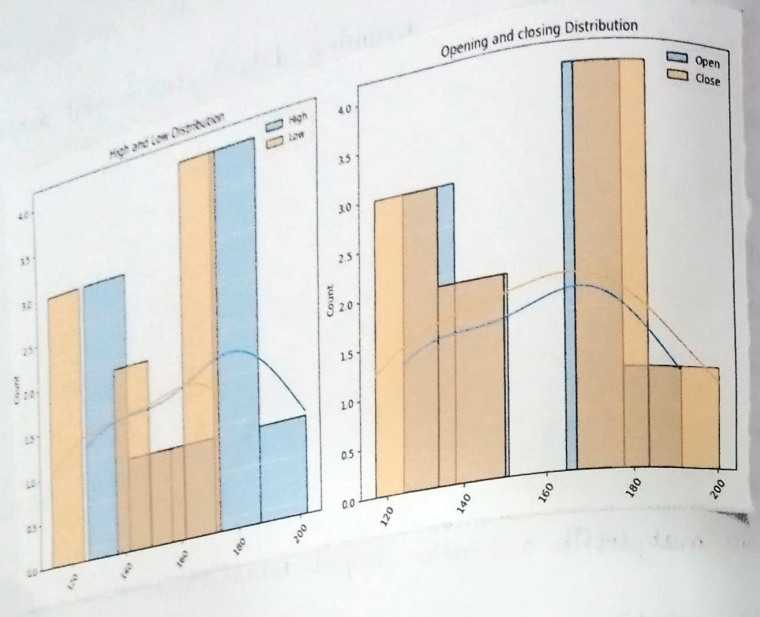
HEAD
Department of CSE (DS)
K.G. College of Engineering & Technology

7b. Visualization on streaming dataset stock market dataset.

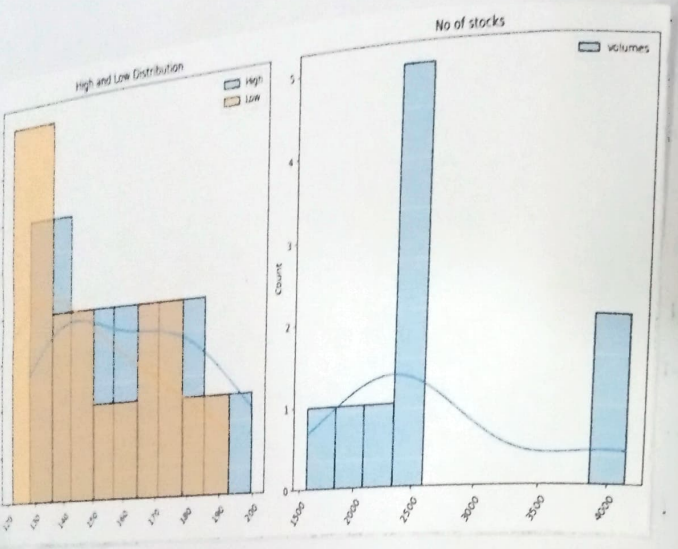
AIM - visualization on streaming dataset stock market dataset.

SOURCE CODE -

```
import csv
import time
import datetime
import random
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def main()
dates, opens, highs, lows, closes, adj_closes, volumes = [], [], [], [], [], []
def update_plot(frame):
try:
current_time = datetime.datetime.now()
date = current_time.strftime("%Y-%m-%d")
time_now = current_time.strftime("%H:%M:%S")
open_price = round(random.uniform(100,200),2)
high_price = round(open_price * random.uniform(1,1,1),2)
low_price = round(open_price * random.uniform(0,9,1),2)
close_price = round(random.uniform(low_price, high_price),2)
```




```
volume = random.randint(1000, 5000)

dates.append(date)
opens.append(open_price)
highs.append(high_price)
lows.append(low_price)
closes.append(close_price)
volumes.append(volume)

if len(highs)>0
dates.pop(0)
opens.pop(0)
highs.pop(0)
lows.pop(0)
closes.pop(0)
volumes.pop(0)

# plot 1
plt.clf()
plt.subplot(1,2,1)
sns.histplot(highs, kde=True, label='High')
sns.histplot(lows, kde=True, label='Low')
plt.title('High and low Distribution')
plt.legend()
plt.xticks(rotation=45)
```




```
#plot 2
plt.subplot(1,2,2)
sns.histplot(opens, kde = True)
sns.histplot(closes, kde = True)
plt.title('opening and closing Distribution')
plt.legend()
plt.xticks(rotation = 45)

plt.tight_layout()
except keyboardinterrupt:
 print("plotting stopped.")
exit()

fig = plt.figure(figsize = (12,6))
ani = FuncAnimation(fig, update_plot, interval = 1000)
plt.show()
if __name__ == "__main__":
main()
```

RESULT – The program was executed succesfully

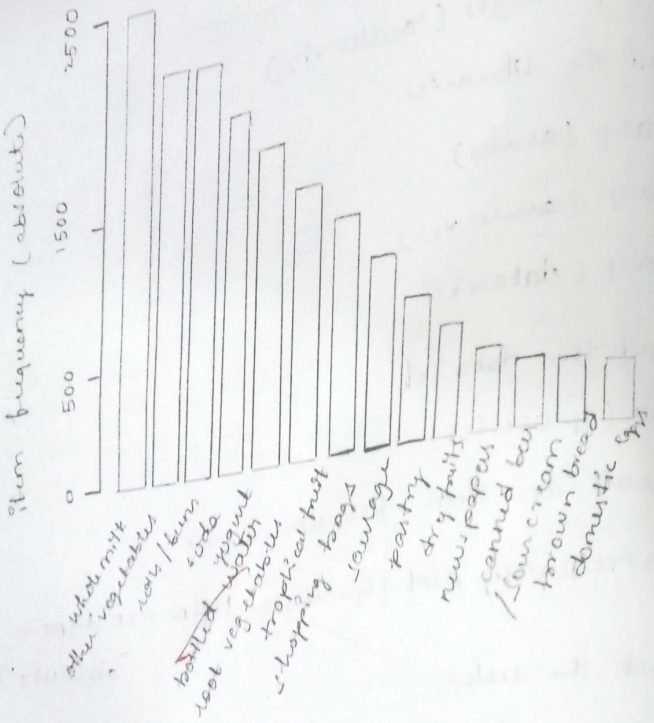
18/4/24

AIM - Market Basket Data Analysis visualization.

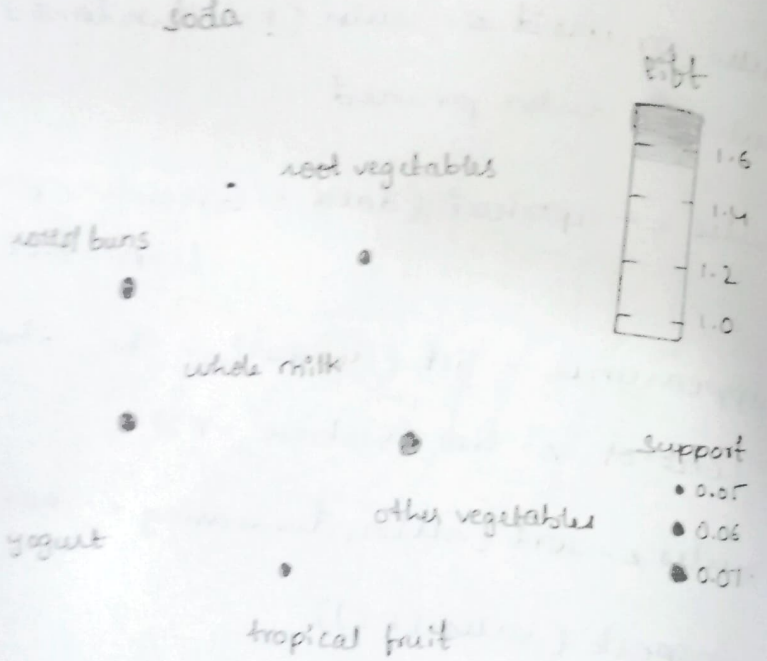
SOURCE CODE -

```
install.packages ('arules')
install.packages ('arules vi2')
#Load the libraries
library (arules)
library (arules vi2)
library ( datasets)
#load the data set
 data (Groceries)
#create an item frequency plot
itemFrequency plot (Groceries, topN = 20, type =
                                    "absolute")
# Get the rules
rules <- apriori (Groceries , parameter = list (supp = 0.01,
                                    conf = 0.8))

options (digits = 2)
inspect (rules [1:5])
rules <- sort (rules, by = "confidence", decreasing = TRUE)
```

OUTPUT -




```
rules <- apriori (Groceries, parameter = list (supp = 0.001,
                  conf = 0.8, maxlen = 3))

subset.matrix <- is.subset (rules, rules)

subset.matrix [lower.tri (subset.matrix, diag = T)] <- NA

redundant <- colSums (subset.matrix, na.rm = T) >= 1

rules.pruned <- rules [! redundant]

rules <- rules.pruned

rules <- apriori (data = Groceries, parameter = list
                  (supp = 0.001, conf = 0.08),

appearance = list (default = "lhs", rhs = "whole milk"),

control <- list (verbose = F))

rules <- sort (rules, decreasing = TRUE, by = "confidence")

inspect (rules [1:5])

rules <- apriori (data = Groceries, parameter = list
                  (supp = 0.001, conf = 0.15, minlen = 2),

appearance = list (default = "rhs", rhs = "whole
                                          milk"),
```




```
control = list (verbose = F))
rules c- sort ( rules, decreasing = TRUE , by =
                                        " confidence ")
inspect (rules [1:5])
plot (rules. method = "graph")
```

RESULT – The Above task was done successfully.

7/3

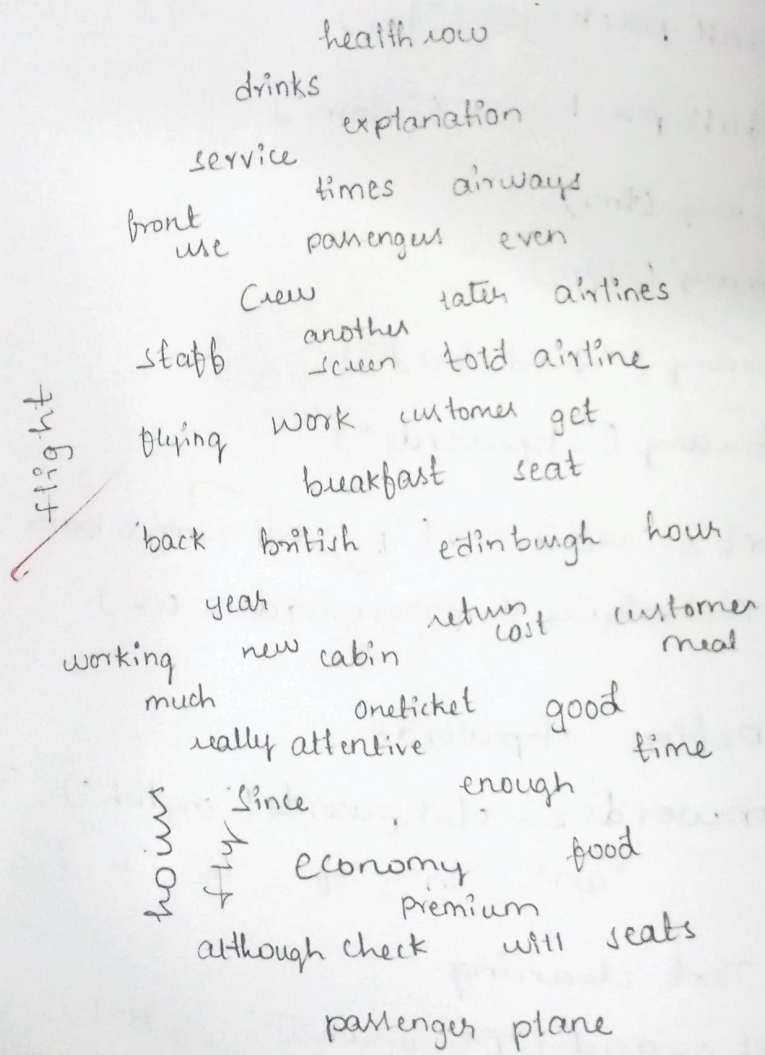
25/4/24

AIM - Text visualization using web analytics.

SOURCE CODE -

```
install.packages ("wordcloud")
install.packages ("stopwords")
install.packages ("tm")
install.packages ("stam")
library (tm)
library (stam)
library ("wordcloud")
library ("stopwords")
text <- read.csv ("F:\\3rd 2sem\\ data visualization
  Techniqus \\ programa data.csv")

# Define stopwords
stopwords <- c(stopwords ("english"), "the", "a",
        "an", "in", "of", "to", "is")
# Text cleaning
text <- gsub ("[^[:alnum:]] ", " ", text)
text <- tolower (text)
```

OUTPUT –




```
wordcloud = WordCloud(words = str.split(text, split = ",")
                      [1]),
    stopwords = stopwords,
    min.frequency = 3,
    scale = c(3, 0.5),
    colors = brewer.pal(8, "Dark2"))
```

313

215/24