Practical 1

Aim: Data preprocessing

Description:
Preprocessing in data analysis and machine learning involves a series of steps to clean, transform, and enhance raw data before it is used for modeling, analysis, or visualization. Preprocessing plays a crucial role in ensuring the quality and suitability of data for subsequent tasks. It aims to address various data-related issues and improve the overall effectiveness of data-driven processes

Code:
```
head(airquality)
mean(airquality)
mean(airquality$Solar.R,na.rm = TRUE)
New_df = airquality

New_df$Ozone = ifelse(is.na(New_df$Ozone),
                median(New_df$Ozone,
                      na.rm = TRUE),
                New_df$Ozone)
head(New_df)
#create a table with parameter marks and roll and save the file in .csv format
dt = read.csv(file.choose())
head(dt)
dt$marks= ifelse(is.na(dt$marks),
              mean(dt$marks,
                  na.rm = TRUE),
              dt$marks)


#Removing outlier using boxplot

data <- iris[,2]
length(data)

boxplot(data)
boxplot(data,plot = FALSE)$out

outlier<- boxplot(data , plot = FALSE)$out
data_no_outlier <- data [-which(data %in% outlier)]

boxplot(data_no_outlier,plot = FALSE)$out
```

length(data_no_outlier)
boxplot(data_no_outlier)

Output:

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> 
```

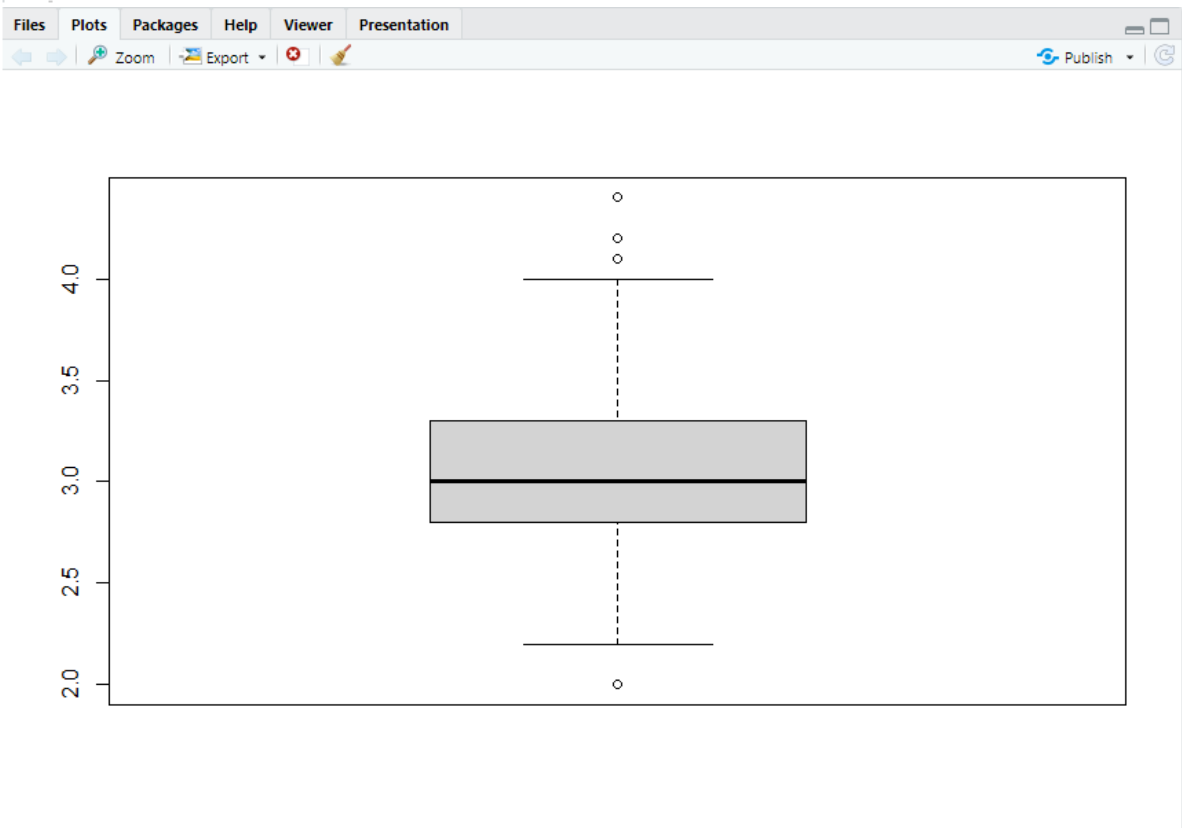
```
> mean(airquality$Solar.R,na.rm = TRUE)
[1] 185.9315
```

```
> New_df = airquality
>
> New_df$Ozone = ifelse(is.na(New_df$Ozone),
+                       median(New_df$Ozone,
+                              na.rm = TRUE),
+                       New_df$Ozone)
> head(New_df)
  Ozone Solar.R Wind Temp Month Day
1  41.0     190  7.4   67     5   1
2  36.0     118  8.0   72     5   2
3  12.0     149 12.6   74     5   3
4  18.0     313 11.5   62     5   4
5  31.5      NA 14.3   56     5   5
6  28.0      NA 14.9   66     5   6
```

```
> dt = read.csv(file.choose())
Qt: Untested Windows version 10.0 detected!
> head(dt)
  roll marks
1   20    75
2   21    55
3   22    65
4   23    47
5   24    NA
6   25    65
> 
```

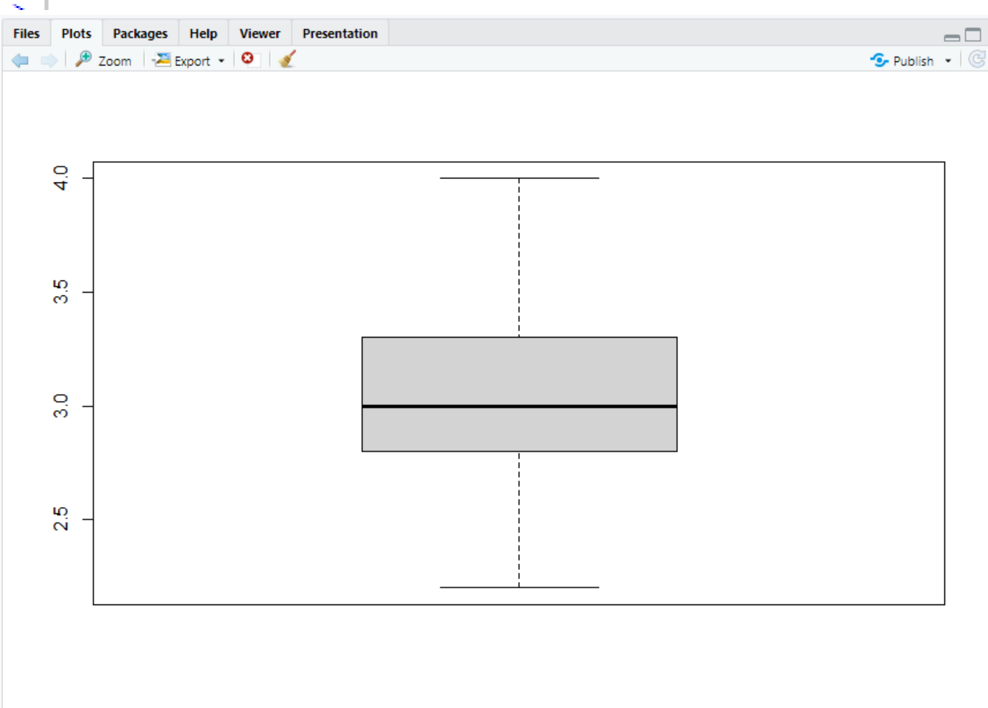
```
> dt$marks= ifelse(is.na(dt$marks),
+                  mean(dt$marks,
+                       na.rm = TRUE),
+                  dt$marks)
> head(dt)
  roll    marks
1   20 75.00000
2   21 55.00000
3   22 65.00000
4   23 47.00000
5   24 64.33333
6   25 65.00000
>
```

```
> data <- iris[,2]
> length(data)
[1] 150
```



```
> boxplot(data)
> boxplot(data,plot = FALSE)$out
[1] 4.4 4.1 4.2 2.0
```

```
> outlier<- boxplot(data , plot = FALSE)$out
> data_no_outlier <- data [-which(data %in% outlier)]
> boxplot(data_no_outlier,plot = FALSE)$out
numeric(0)
> length(data_no_outlier)
[1] 146
> boxplot(data_no_outlier)
```



Practical 2

Aim: Clustering

Description:

Clustering is a machine learning and data analysis technique that involves grouping similar data points together based on certain characteristics or features. It aims to discover inherent patterns, structures, or associations within a dataset by partitioning it into distinct clusters or groups. Clustering is commonly used for tasks such as customer segmentation, anomaly detection, and pattern recognition. It helps uncover hidden insights within data and simplifies the process of understanding and making decisions about complex datasets.

Code:-

Create .csv file with random dataset dt = read.csv(file.choose())
head(dt)

```
> head(dt)
    Rating    Price   Alcohol  Sulphates       pH
1 66.09628 42.35785 12.884478 1.82236164 3.094826
2 26.29778 18.07262 10.788513 1.25833852 3.687788
3 57.55450 27.33195  9.836948 0.02351161 3.463087
4 46.55640 22.75134 12.022274 1.79914031 1.544849
5 49.40503 14.39930 11.442785 1.06218391 3.032852
6 41.32942 18.01701 11.667942 0.85062170 2.191800
```

Create subset of data

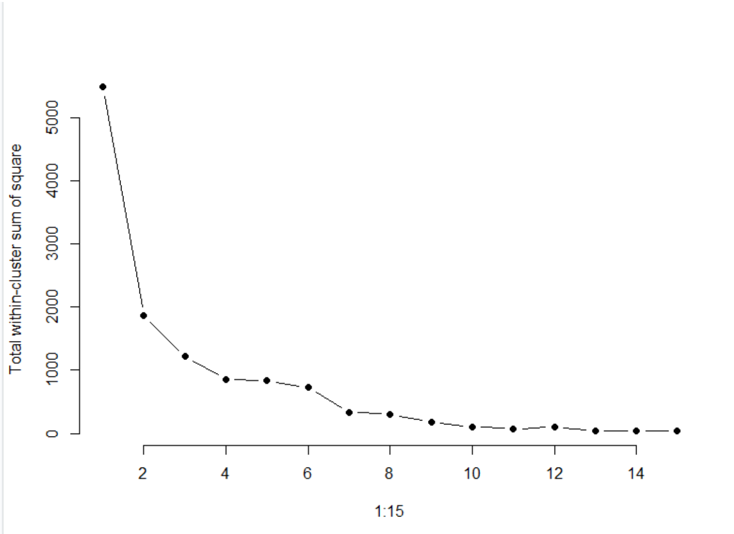
```
> new_data <- dt[, -which(names(dt) == "pH")]
> new_data
     Rating    Price   Alcohol  Sulphates
1  66.09628 42.35785 12.884478 1.82236164
2  26.29778 18.07262 10.788513 1.25833852
3  57.55450 27.33195  9.836948 0.02351161
4  46.55640 22.75134 12.022274 1.79914031
5  49.40503 14.39930 11.442785 1.06218391
6  41.32942 18.01701 11.667942 0.85062170
7  64.17850 39.98592 11.845932 1.77516168
8  35.06645 12.99000 12.054014 0.88979250
```

```
> c1<-kmeans(new_data, 3)
> c1
K-means clustering with 3 clusters of sizes 7, 8, 5

Cluster means:
    Rating    Price  Alcohol Sulphates
1 54.98371 22.23150 11.41921 0.7603628
2 34.98192 19.11217 11.19112 1.0566711
3 63.08960 39.99343 11.47984 1.1505030

Clustering vector:
 [1] 3 2 1 1 1 2 3 2 1 3 1 2 2 2 2 3 3 1 2 1

Within cluster sum of squares by cluster:
[1] 462.4866 645.4911 108.7511
 (between_SS / total_SS =  77.8 %)
```

```
> data<- new_data
> wss <- sapply(1:15, function (k){kmeans(data , k) $tot.withinss})
> wss
 [1] 5484.24452 1866.10652 1216.72880  854.74127  832.38034  727.68515  336.73655  302.45165  185.68252  106.80032   70.87394
[12]  103.66464   45.20735   39.18805   34.37002
```

There are 3 clusters
Plot graph

```r
plot(1:15, wss , type ="b", pch = 19, frame = FALSE,xlab="Number of cluster K",
     ylab="Total within-cluster sum of square")
```

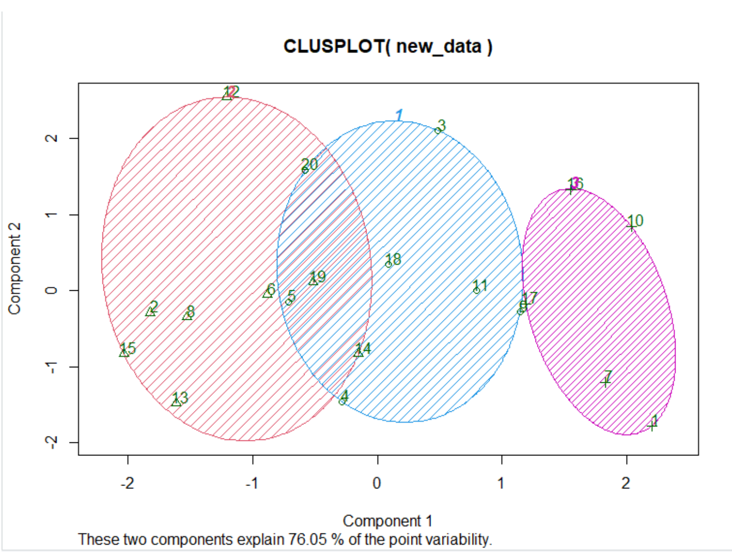


Install package

```
> install.packages("cluster")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/cluster_2.1.4.zip'
Content type 'application/zip' length 586615 bytes (572 KB)
downloaded 572 KB

package 'cluster' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Jagtap\AppData\Local\Temp\Rtmpobo2uz\downloaded_packages
```

Cluster plot

```r
clusplot(new_data, c1$cluster, color=TRUE, shade=TRUE, labels=2,lines=0)
c1$cluster
```

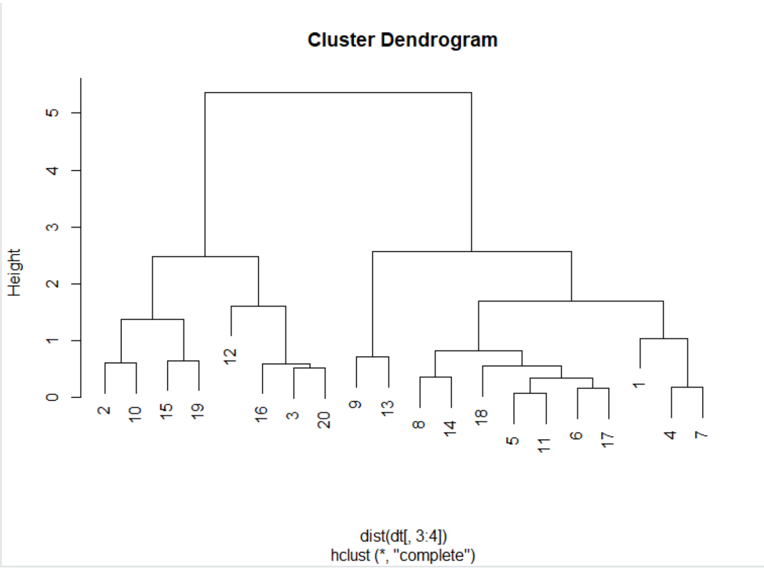


Cluster Plot

```
> clusplot(new_data, c1$cluster, color=TRUE, shade=TRUE, labels=2,lines=0)
> c1$cluster
 [1] 3 2 1 1 1 2 3 2 1 3 1 2 2 2 2 3 3 1 2 1
> c1$centers
    Rating    Price  Alcohol Sulphates
1 54.98371 22.23150 11.41921 0.7603628
2 34.98192 19.11217 11.19112 1.0566711
3 63.08960 39.99343 11.47984 1.1505030
```

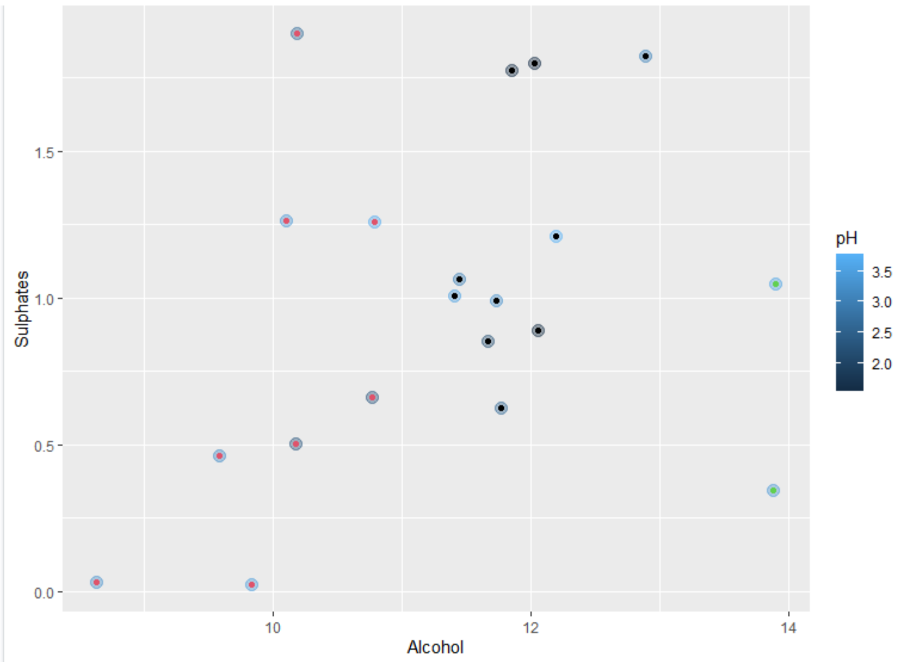
Cluster Dendrogram

```
clusters<- hclust(dist(dt[, 3:4]))
plot(clusters)
```




```
> clusters<- hclust(dist(dt[, 3:4]))
> plot(clusters)
> clusterCut <- cutree(clusters, 3)
> table(clusterCut, dt$pH)

clusterCut 1.544849157 1.559231119 1.571996795 2.19179981 2.313879064 2.315566793 2.641539283 2.673020378 3.03285177 3.094826361
         1           1           1           1          1           0           0           1           0          1           1
         2           0           0           0          0           1           1           0           1          0           0
         3           0           0           0          0           0           0           0           0          0           0

clusterCut 3.226572258 3.29492405 3.300326866 3.401335729 3.410770217 3.463087396 3.563198982 3.574160246 3.687788197 3.773416688
         1           0          1           0           0           0           0           1           0           0           1
         2           1          0           1           1           0           1           0           0           1           0
         3           0          0           0           0           1           0           0           1           0           0
> 
```

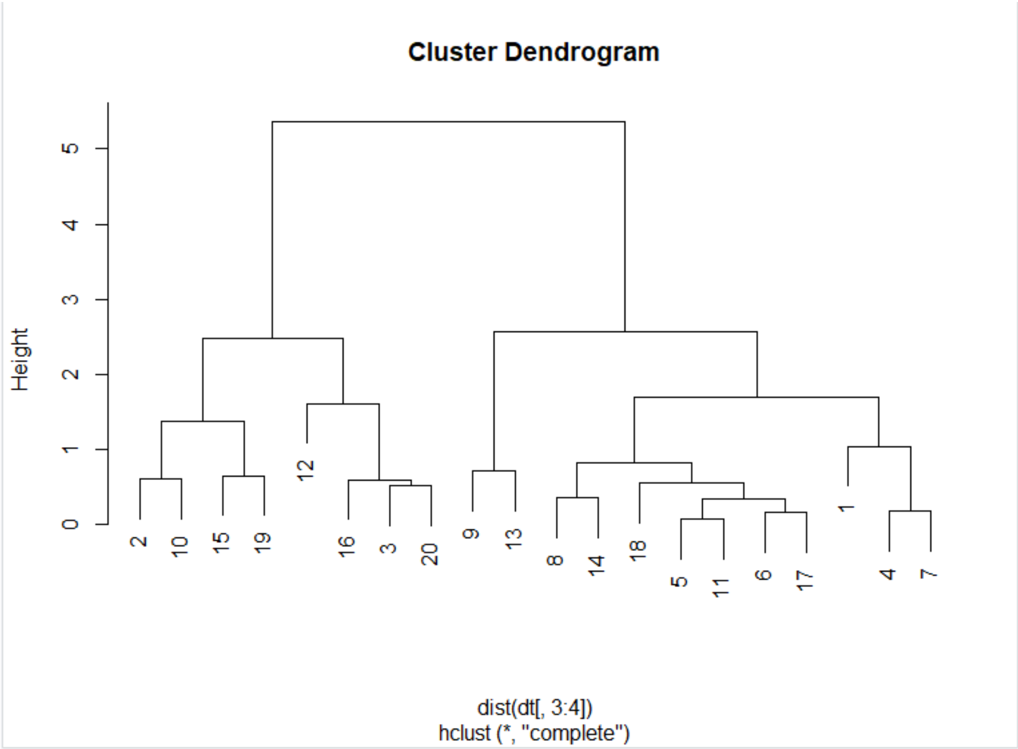
```
ggplot(dt, aes(Alcohol, Sulphates,color = pH)) +
  geom_point(alpha = 0.4,size = 3.5) + geom_point(col = clusterCut)
+ scale_color_manual(values = c('black', 'red' , 'green'))
```



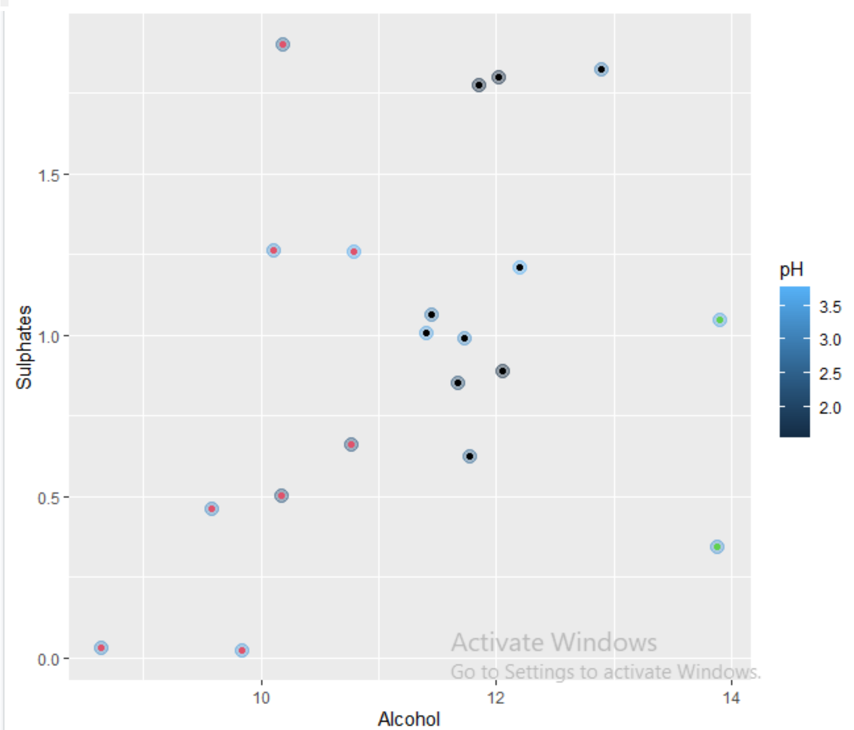
Virginica has maximum clustering

```
> cluster <- hclust(dist(dt[, 3:4]), method = 'average')
> clusterCut1 <- cutree(cluster, 3)
> table(clusterCut1,dt$pH)

clusterCut1 1.544849157 1.559231119 1.571996795 2.19179981 2.313879064 2.315566793 2.641539283 2.673020378 3.03285177 3.094826361 3.226572258 3.29492405 3.300326866
          1           1           1           1          1           0           0           1           0          1           1           0          1           0
          2           0           0           0          0           1           1           0           1          0           0           1          0           1
          3           0           0           0          0           0           0           0           0          0           0           0          0           0

clusterCut1 3.401335729 3.410770217 3.463087396 3.563198982 3.574160246 3.687788197 3.773416688
          1           0           0           0           1           0           0           1
          2           1           0           1           0           0           1           0
          3           0           1           0           0           1           0           0
> |
```

```
> plot(clusters)
> |
```




```
ggplot(dt, aes(Alcohol, Sulphates,color = pH)) +
  geom_point(alpha = 0.4,size = 3.5) + geom_point(col = clusterCut1)
+ scale_color_manual(values = c('black', 'red' , 'green'))
```



```
> install.packages("fpc")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before p
oceeding:

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/fpc_2.2-10.zip'
Content type 'application/zip' length 838741 bytes (819 KB)
downloaded 819 KB

package ‘fpc’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Jagtap\AppData\Local\Temp\RtmpsXlkCi\downloaded_packages
> library(fpc)
> dt_1 <- dt[-5]
> set.seed(220)
> Dbscan_c1 <- dbscan(dt_1, eps = 0.45, MinPts = 5)
> Dbscan_c1$cluster
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> 
```

```
> table(Dbscan_c1$cluster, dt$pH)

    1.544849157 1.559231119 1.571996795 2.19179981 2.313879064 2.315566793 2.641539283 2.673020378 3.03285177 3.094826361 3.226572258 3.29492405
  0           1           1           1          1           1           1           1           1          1           1           1          1

    3.300326866 3.401335729 3.410770217 3.463087396 3.563198982 3.574160246 3.687788197 3.773416688
  0           1           1           1           1           1           1           1           1
```

```
plot(Dbscan_c1, main = "DBSCAN")
plot(Dbscan_c1, main = "Sulphates vs Rating")
```

Practical 3

Aim: Recommendation System

Description:

Recommender Systems, also known as recommendation systems or recommendation engines, are a class of information filtering systems that aim to predict and suggest items or content that a user might be interested in. These systems have gained significant importance in various applications and industries, especially in the digital era where there is an abundance of content and products.

Google Colab Link:

https://colab.research.google.com/drive/1Zi5DkO7TwHn_czaCuGrJ9m8bC-hzvJp4?usp=sharing

Output:

```python
import pandas as pd

metadata = pd.read_csv('movies_metadata.csv', low_memory=False)

# Print the first three rows
metadata.head(3)
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_date	revenue	runtime	spoken_
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	1995-10-30	373554033.0	81.0	[{'iso_6 'name'
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	1995-12-15	262797249.0	104.0	[{'iso_6 'name'
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	...	1995-12-22	0.0	101.0	[{'iso_6 'name'

3 rows × 24 columns

```python
# Calculate mean of vote average column
C = metadata['vote_average'].mean()
print(C)
```

```
5.618207215134185
```

```python
# Calculate the minimum number of votes required to be in the chart, m
m = metadata['vote_count'].quantile(0.90)
print(m)
```

```
160.0
```

```python
# Filter out all qualified movies into a new DataFrame
q_movies = metadata.copy().loc[metadata['vote_count'] >= m]
q_movies.shape
```

```
(4555, 24)
```

```python
metadata.shape
```

```
(45466, 24)
```

```python
[6] # Function that computes the weighted rating of each movie
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
```

```python
[9] # Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

```python
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(20)
```

	title	vote_count	vote_average	score
314	The Shawshank Redemption	8358.0	8.5	8.445869
834	The Godfather	6024.0	8.5	8.425439
10309	Dilwale Dulhania Le Jayenge	661.0	9.1	8.421453
12481	The Dark Knight	12269.0	8.3	8.265477
2843	Fight Club	9678.0	8.3	8.256385
292	Pulp Fiction	8670.0	8.3	8.251406
522	Schindler's List	4436.0	8.3	8.206639
23673	Whiplash	4376.0	8.3	8.205404
5481	Spirited Away	3968.0	8.3	8.196055
2211	Life Is Beautiful	3643.0	8.3	8.187171
1178	The Godfather: Part II	3418.0	8.3	8.180076
1152	One Flew Over the Cuckoo's Nest	3001.0	8.3	8.164256
351	Forrest Gump	8147.0	8.2	8.150272
1154	The Empire Strikes Back	5998.0	8.2	8.132919
1176	Psycho	2405.0	8.3	8.132715
18465	The Intouchables	5410.0	8.2	8.125837
40251	Your Name.	1030.0	8.5	8.112532
289	Leon: The Professional	4293.0	8.2	8.107234
3030	The Green Mile	4166.0	8.2	8.104511

```
#Print plot overviews of the first 5 movies.
metadata['overview'].head()
```

```
0    Led by Woody, Andy's toys live happily in his ...
1    When siblings Judy and Peter discover an encha...
2    A family wedding reignites the ancient feud be...
3    Cheated on, mistreated and stepped on, the wom...
4    Just when George Banks has recovered from his ...
Name: overview, dtype: object
```

```
#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
metadata['overview'] = metadata['overview'].fillna('')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(metadata['overview'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

```
(45466, 75827)
```

```
#Array mapping from feature integer indices to feature name.
tfidf.get_feature_names_out()[5000:5010]
```

```
array(['avails', 'avaks', 'avalanche', 'avalanches', 'avallone', 'avalon',
       'avant', 'avanthika', 'avanti', 'avaracious'], dtype=object)
```

```
movies = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/movies.csv")
movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
n_ratings = len(ratings)
n_movies = len(ratings['movieId'].unique())
n_users = len(ratings['userId'].unique())

print(f"Number of ratings: {n_ratings}")
print(f"Number of unique movieId's : {n_movies}")
print(f"Number of unique unique users : {n_users}")
print(f"Average ratings per user : {round(n_ratings/n_users, 2)}")
print(f"Average ratings per movie : {round(n_ratings/n_movies, 2)}")
```

```
Number of ratings: 100836
Number of unique movieId's : 9724
Number of unique unique users : 610
Average ratings per user : 165.3
Average ratings per movie : 10.37
```

```python
user_freq = ratings[['userId','movieId']].groupby('userId').count().reset_index()
user_freq.columns = ['userId;','n_ratings']
user_freq.head()
```

	userId;	n_ratings
0	1	232
1	2	29
2	3	39
3	4	216
4	5	44

```python
#Find lowest and Highest rated movies:
mean_rating = ratings.groupby('movieId')[['rating']].mean()
mean_rating

#Lowest rated movies
lowest_rated = mean_rating['rating'].idxmin()
movies.loc[movies['movieId'] == lowest_rated]

#Highest rated movies
highest_rated = mean_rating['rating'].idxmax()
movies.loc [ movies['movieId'] == highest_rated]
```

	movieId	title	genres
48	53	Lamerica (1994)	Adventure|Drama

```python
ratings[ratings['movieId']==highest_rated]
```

	userId	movieId	rating	timestamp
13368	85	53	5.0	889468268
96115	603	53	5.0	963180003

```python
ratings[ratings['movieId']==lowest_rated]
```

	userId	movieId	rating	timestamp
13633	89	3604	0.5	1520408880

```python
## the above movies has very low dataset. We will use bayesian average
movie_stats = ratings.groupby('movieId')[['rating']].agg(['count', 'mean'])
movie_stats.columns = movie_stats.columns.droplevel()
```

```python
# Now, we create user-item matrix using scipy csr matrix
from scipy.sparse import csr_matrix
```

```python
# show number of people who rated movies rated movie highest
ratings[ratings['movieId']==highest_rated]
```

	userId	movieId	rating	timestamp
13368	85	53	5.0	889468268
96115	603	53	5.0	963180003

```python
# show number of people who rated movies rated movie lowest
ratings[ratings['movieId']==lowest_rated]
```

	userId	movieId	rating	timestamp
13633	89	3604	0.5	1520408880

```python
## the above movies has very low dataset. We will use bayesian average
movie_stats = ratings.groupby('movieId')[['rating']].agg(['count', 'mean'])
movie_stats.columns = movie_stats.columns.droplevel()
```

```python
# Now, we create user-item matrix using scipy csr matrix
from scipy.sparse import csr_matrix
```

```python
def create_matrix(df):

  N = len(df['userId'].unique())
  M = len(df['movieId'].unique())

  # Map Ids to indices
  user_mapper = dict(zip(np.unique(df["userId"]), list(range(N))))
  movie_mapper = dict(zip(np.unique(df["movieId"]), list(range(M))))

  # Map indices to IDs
  user_inv_mapper = dict(zip(list(range(N)), np.unique(df["userId"])))
  movie_inv_mapper = dict(zip(list(range(M)), np.unique(df["movieId"])))

  user_index = [user_mapper[i] for i in df['userId']]
  movie_index = [movie_mapper[i] for i in df['movieId']]
```

```python
    # https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html
    # https://stackoverflow.com/questions/53254104/cant-understand-scipy-sparse-csr-matrix-example
    X = csr_matrix((df["rating"], (movie_index, user_index)), shape=(M, N))

    return X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper
```

```python
X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper = create_matrix(ratings)
```

```python
print(movie_inv_mapper)
```

Practical 4

Aim:- Collaborative Filtering.

Description: Collaborative Filtering is a technique used in recommendation systems to predict a user's preferences or interests by collecting preferences from many users. It operates on the principle that if a person A has the same opinion as person B on an issue, A is more likely to have B's opinion on a different issue. It relies on the assumption that people who agreed in the past will agree in the future and can make recommendations based on similar users' preferences.

```python
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
ratings = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/ratings.csv")
ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

```python
movies = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/movies.csv")
movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```python
n_ratings = len(ratings)
n_movies = len(ratings['movieId'].unique())
n_users = len(ratings['userId'].unique())

print(f"Number of ratings: {n_ratings}")
print(f"Number of unique movieId's : {n_movies}")
print(f"Number of unique unique users : {n_users}")
print(f"Average ratings per user : {round(n_ratings/n_users, 2)}")
print(f"Average ratings per movie : {round(n_ratings/n_movies, 2)}")
```

```
Number of ratings: 100836
Number of unique movieId's : 9724
Number of unique unique users : 610
Average ratings per user : 165.3
Average ratings per movie : 10.37
```

```python
user_freq = ratings[['userId','movieId']].groupby('userId').count().reset_index()
user_freq.columns = ['userId;','n_ratings']
user_freq.head()
```

	userId;	n_ratings
0	1	232
1	2	29
2	3	39
3	4	216
4	5	44

```
#Find lowest and Highest rated movies:
mean_rating = ratings.groupby('movieId')[['rating']].mean()
mean_rating

#Lowest rated movies
lowest_rated = mean_rating['rating'].idxmin()
movies.loc[movies['movieId'] == lowest_rated]

#Highest rated movies
highest_rated = mean_rating['rating'].idxmax()
movies.loc [ movies['movieId'] == highest_rated]
```

	movieId	title	genres
48	53	Lamerica (1994)	Adventure|Drama

```
ratings[ratings['movieId']==highest_rated]
```

	userId	movieId	rating	timestamp
13368	85	53	5.0	889468268
96115	603	53	5.0	963180003

```
ratings[ratings['movieId']==lowest_rated]
```

	userId	movieId	rating	timestamp
13633	89	3604	0.5	1520408880

```
## the above movies has very low dataset. We will use bayesian average
movie_stats = ratings.groupby('movieId')[['rating']].agg(['count', 'mean'])
movie_stats.columns = movie_stats.columns.droplevel()
```

```
# Now, we create user-item matrix using scipy csr matrix
from scipy.sparse import csr_matrix
```

```python
# show number of people who rated movies rated movie highest
ratings[ratings['movieId']==highest_rated]
```

	userId	movieId	rating	timestamp
13368	85	53	5.0	889468268
96115	603	53	5.0	963180003

```python
# show number of people who rated movies rated movie lowest
ratings[ratings['movieId']==lowest_rated]
```

	userId	movieId	rating	timestamp
13633	89	3604	0.5	1520408880

```python
## the above movies has very low dataset. We will use bayesian average
movie_stats = ratings.groupby('movieId')[['rating']].agg(['count', 'mean'])
movie_stats.columns = movie_stats.columns.droplevel()
```

```python
# Now, we create user-item matrix using scipy csr matrix
from scipy.sparse import csr_matrix
```

```python
def create_matrix(df):

  N = len(df['userId'].unique())
  M = len(df['movieId'].unique())

  # Map Ids to indices
  user_mapper = dict(zip(np.unique(df["userId"]), list(range(N))))
  movie_mapper = dict(zip(np.unique(df["movieId"]), list(range(M))))

  # Map indices to IDs
  user_inv_mapper = dict(zip(list(range(N)), np.unique(df["userId"])))
  movie_inv_mapper = dict(zip(list(range(M)), np.unique(df["movieId"])))

  user_index = [user_mapper[i] for i in df['userId']]
  movie_index = [movie_mapper[i] for i in df['movieId']]
```

```python
  # https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html
  # https://stackoverflow.com/questions/53254104/cant-understand-scipy-sparse-csr-matrix-example
  X = csr_matrix((df["rating"], (movie_index, user_index)), shape=(M, N))

  return X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper
```

```python
X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper = create_matrix(ratings)
```

```python
print(movie_inv_mapper)
```



```python
from sklearn.neighbors import NearestNeighbors
"""
Find similar movies using KNN
"""
def find_similar_movies(movie_id, X, k, metric='cosine', show_distance=False):

  neighbour_ids = []

  movie_ind = movie_mapper[movie_id]
  movie_vec = X[movie_ind]
  k+=1
  kNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
  kNN.fit(X)
  movie_vec = movie_vec.reshape(1,-1)
  neighbour = kNN.kneighbors(movie_vec, return_distance=show_distance)
  for i in range(0,k):
    n = neighbour.item(i)
    neighbour_ids.append(movie_inv_mapper[n])
```

Practical 5

Aim:- Association.

Description:

- **Association:-**

Association is a data mining technique that discovers the probability of the co-occurrence of items in a collection. The relationships between co-occurring items are expressed as Association Rules. Association rule mining finds interesting associations and relationships among large sets of data items. Association rules are "if-then" statements, that help to show the probability of relationships between data items, within large data sets in various types of databases. Here the If element is called antecedent, and then statement is called as Consequent. These types of relationships where we can find out some association or relation between two items is known as single cardinality. Association rule mining has a number of applications and is widely used to help discover sales
correlations in transactional data or in medical data sets.

- **Apriori:**

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.
To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.
Apriori Property - All non-empty subset of frequent itemset must be frequent.

Limitations of Apriori Algorithm

Apriori Algorithm can be slow.
The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets. It will check for many sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficiency when memory capacity is limited with large number of transactions.

Algorithm

• Calculate the support of item sets (of size k = 1) in the transactional database (note that support is the frequency of occurrence of an itemset). This is called generating the candidate set.

OR

• Prune the candidate set by eliminating items with a
support less than the given threshold.
• Join the frequent itemsets to form sets of size k + 1, and repeat the above sets until no more itemsets can be
formed. This will happen when the set(s) formed have a support less than the given support.
1. Set a minimum support and confidence.

2. Take all the subset present in the transactions which have higher support than minimum support.
3. Take all the rules of these subsets which have higher confidence than minimum confidence.
4. Sort the rules by decreasing lift.

Components of Apriori

Support and Confidence:

Support refers to items' frequency of occurrence i.e. x and y items are purchased together, confidence is a conditional probability that y item is purchased given that x item is purchased.
Support(I)=(Number of transactions containing item I)/(Total number of transactions)
Confidence(11 -> 12)=(Number of transactions containing 11 and 12)/ (Number of transactions containing 11)

Support (A) = Number of transaction in which A appears
Total number of transactions
Confidence (A→B) = Support (AUB)
Support(A)

Lift:

Lift gives the correlation between A and B in the rule A=>B. Correlation shows how one item-set A effects the item-set B.
If the rule had a lift of 1,then A and B are independent and no rule can be derived from them.
If the lift is > 1, then A and B are dependent on each other, and the degree of which is given by ift value.
If the lift is < 1, then presence of A will have negative effect on
B.
Lift(11 -> 12) = (Confidence(11 -> 12) / (Support(12))

Coverage:

Coverage (also called cover or LHS-support) is the support of the left-hand-side of the rule X => Y, i.e., supp(X).
It represents a measure of to how often the rule can be applied.
Coverage can be quickly calculated from the rule's quality measures (support and confidence)

Code:

```
# Load required packages
install.packages('arules')
install.packages('arulesViz')
install.packages('RColorBrewer')

library(arules)
library(arulesViz)
```

```
library(RColorBrewer)

# Load the Groceries dataset and explore it
data("Groceries")
str(Groceries)
inspect(head(Groceries, 2))
Groceries@itemInfo$labels

# Apriori analysis on the Groceries dataset
grocery_rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))
inspect(rules[1:10])
inspect(head(sort(grocery_rules, by = 'confidence'), 3))
inspect(tail(sort(grocery_rules, by = 'confidence'), 3))

# Apriori analysis for "whole milk" rules
wholemilk_rules <- apriori(data = Groceries, parameters = list(supp = 0.001, conf = 0.08),
appearance = list(rhs = 'whole milk'))
inspect(head(sort(wholemilk_rules, by = 'confidence'), 3))

# Apriori analysis with increased support and confidence
grocery_rules_increased_support <- apriori(Groceries, parameter = list(support = 0.02,
confidence = 0.5))
inspect(head(sort(grocery_rules_increased_support, by = 'confidence'), 3))

# Item Frequency Plot for Groceries dataset
itemFrequencyPlot(Groceries, topN = 20, type = "absolute", col = brewer.pal(8, 'Pastel2'), main
= "Absolute Item Frequency Plot")

# Import and analyze a transaction dataset (restaurant orders)
txn <- read.transactions(file = "C:/Users/student/Downloads/restaurant-1-orders.csv",
rm.duplicates = TRUE, format = "single", sep = ",", header = TRUE, cols = c("Order Number",
"Item Name"))
str(txn)
inspect(head(txn, 2))
txn@itemInfo$labels

# Apriori analysis on the restaurant orders dataset
rules <- apriori(txn, parameter = list(supp = 0.01, conf = 0.2))
inspect(rules[1:10])
inspect(head(sort(rules, by = "confidence"), 3))

# Apriori analysis for "Pilau Rice" rules
Pilau_Rice_rules <- apriori(data = txn, parameter = list(supp = 0.003, conf = 0.08), appearance
= list(rhs = "Pilau Rice"))
```

```
inspect(head(sort(Pilau_Rice_rules, by = "confidence"), 3))

# Apriori analysis with increased support and confidence for restaurant orders
rules_increased_support <- apriori(txn, parameter = list(support = 0.02, confidence = 0.5))
inspect(head(sort(rules_increased_support, by = "confidence"), 3))

# Import and analyze a transaction dataset (movies)
txn <- read.transactions(file = "D:/Chrome Downloads/movies.csv", rm.duplicates = TRUE,
format = "basket", sep = ",", header = TRUE, cols = 3)
str(txn)
inspect(head(txn, 2))

# Apriori analysis on the movies dataset
rules <- apriori(txn, parameter = list(supp = 0.01, conf = 0.2))
inspect(rules[1:10])
inspect(head(sort(rules, by = "confidence"), 3))

# Apriori analysis for "Children" and "IMAX" rules in the movies dataset
Children_rules <- apriori(data = txn, parameter = list(supp = 0.001, conf = 0.03), appearance =
list(rhs = "Children"))
IMAX_rules <- apriori(data = txn, parameter = list(supp = 0.001, conf = 0.03), appearance =
list(rhs = "IMAX"))

inspect(head(sort(Children_rules, by = "confidence"), 5))
inspect(head(sort(IMAX_rules, by = "confidence"), 5))

# Apriori analysis with increased support and confidence for movies dataset
rules_increased_support <- apriori(txn, parameter = list(support = 0.02, confidence = 0.5))
inspect(head(sort(rules_increased_support, by = "confidence"), 3))
```

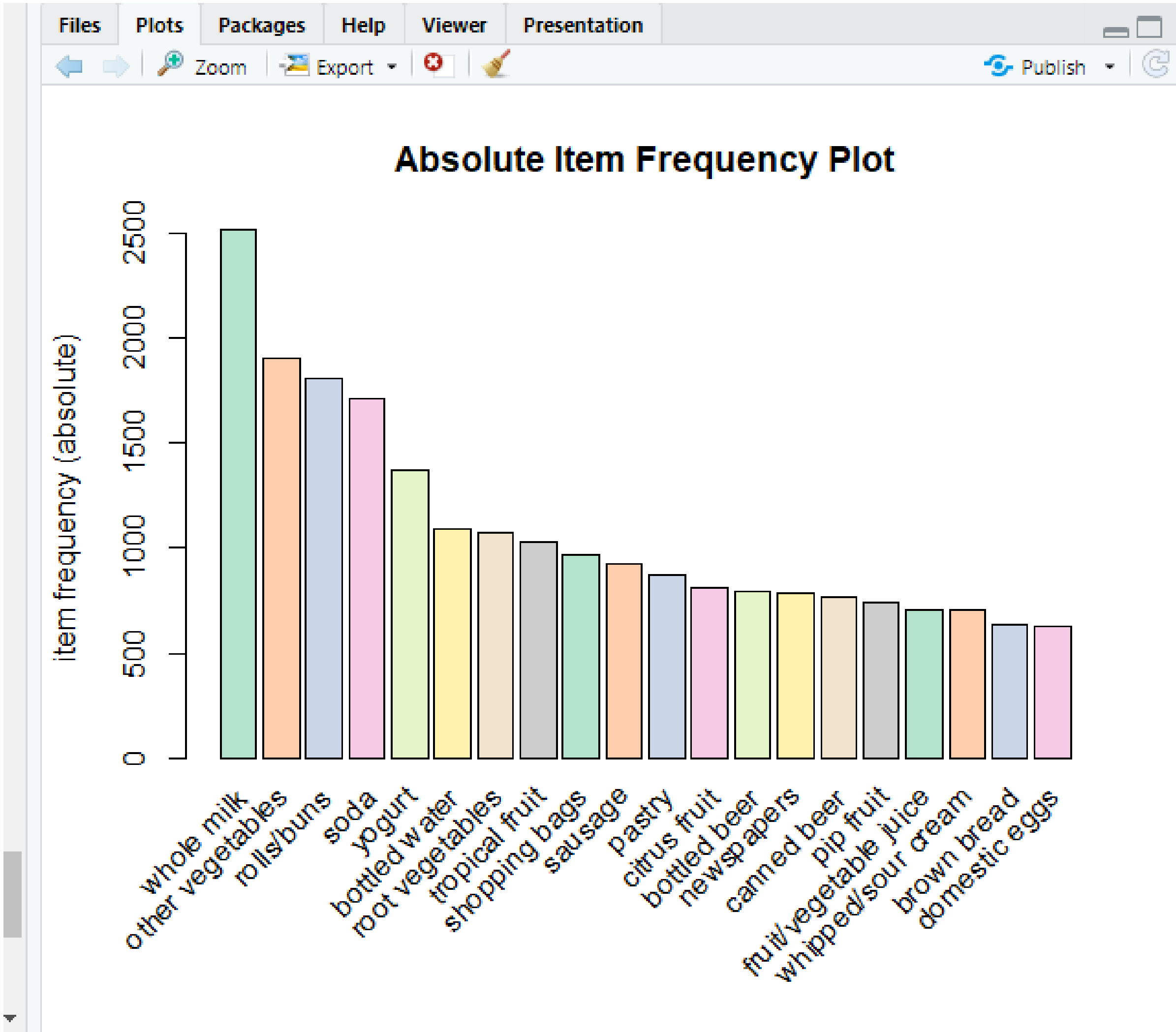
Output:

```
> library(RColorBrewer)
> library(RColorBrewer)
> data("Groceries")
> #displaying data
> str(Groceries)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:43367] 13 60 69 78 14 29 98 24 15 29 ...
  .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
  .. .. ..@ Dim     : int [1:2] 169 9835
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 169 obs. of  3 variables:
  .. ..$ labels: chr [1:169] "frankfurter" "sausage" "liver loaf" "ham" ...
  .. ..$ level2: Factor w/ 55 levels "baby food","bags",..: 44 44 44 44 44 44 44 42
 42 41 ...
  .. ..$ level1: Factor w/ 10 levels "canned food",..: 6 6 6 6 6 6 6 6 6 6 ...
  ..@ itemsetInfo:'data.frame': 0 obs. of  0 variables
> inspect(head(Groceries, 2))
    items
[1] {citrus fruit,
     semi-finished bread,
     margarine,
     ready soups}
[2] {tropical fruit,
     yogurt,
     coffee}
```

```
> #displaying labels i.e. item values
> Groceries@itemInfo$labels
  [1] "frankfurter"                "sausage"
  [3] "liver loaf"                 "ham"
  [5] "meat"                       "finished products"
  [7] "organic sausage"            "chicken"
  [9] "turkey"                     "pork"
 [11] "beef"                       "hamburger meat"
 [13] "fish"                       "citrus fruit"
 [15] "tropical fruit"             "pip fruit"
 [17] "grapes"                     "berries"
 [19] "nuts/prunes"                "root vegetables"
 [21] "onions"                     "herbs"
 [23] "other vegetables"           "packaged fruit/vegetables"
 [25] "whole milk"                 "butter"
 [27] "curd"                       "dessert"
 [29] "butter milk"                "yogurt"
 [31] "whipped/sour cream"         "beverages"
 [33] "UHT-milk"                   "condensed milk"
 [35] "cream"                      "soft cheese"
 [37] "sliced cheese"              "hard cheese"
 [39] "cream cheese "              "processed cheese"
 [41] "spread cheese"              "curd cheese"
 [43] "specialty cheese"           "mayonnaise"
 [45] "salad dressing"             "tidbits"
 [47] "frozen vegetables"          "frozen fruits"
 [49] "frozen meals"               "frozen fish"
 [51] "frozen chicken"             "ice cream"
 [53] "frozen dessert"             "frozen potato products"
 [55] "domestic eggs"              "rolls/buns"
 [57] "white bread"                "brown bread"
 [59] "pastry"                     "roll products "
 [61] "semi-finished bread"        "zwieback"
 [63] "potato products"            "flour"
 [65] "salt"                       "rice"
 [67] "pasta"                      "vinegar"
 [69] "oil"                        "margarine"
 [71] "specialty fat"              "sugar"
 [73] "artif. sweetener"           "honey"
 [75] "mustard"                    "ketchup"
 [77] "spices"                     "soups"
 [79] "ready soups"                "Instant food products"
 [81] "sauces"                     "cereals"
 [83] "organic products"           "baking powder"
 [85] "preservation products"      "pudding powder"
 [87] "canned vegetables"          "canned fruit"
 [89] "pickled vegetables"         "specialty vegetables"
 [91] "jam"                        "sweet spreads"
 [93] "meat spreads"               "canned fish"
```

```
> #applying apriori algorithm
> grocery_rules <- apriori(Groceries, parameter = list(supp = 0.01,
+                                                        conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
        0.2    0.1    1 none FALSE            TRUE       5    0.01      1     10
 target  ext
  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 98

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [232 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
> inspect(head(sort(grocery_rules, by = 'confidence'), 3))
    lhs                                rhs                support    confidence
[1] {citrus fruit, root vegetables}   => {other vegetables} 0.01037112 0.5862069
[2] {tropical fruit, root vegetables} => {other vegetables} 0.01230300 0.5845411
[3] {curd, yogurt}                    => {whole milk}       0.01006609 0.5823529
    coverage   lift     count
[1] 0.01769192 3.029608 102
[2] 0.02104728 3.020999 121
[3] 0.01728521 2.279125  99
> inspect(tail(sort(grocery_rules, by = 'confidence'), 3))
    lhs                        rhs                support    confidence
[1] {fruit/vegetable juice} => {rolls/buns}       0.01453991 0.2011252
[2] {bottled beer}          => {other vegetables} 0.01616675 0.2007576
[3] {tropical fruit}        => {root vegetables}  0.02104728 0.2005814
    coverage   lift     count
[1] 0.07229283 1.093458 143
[2] 0.08052872 1.037546 159
[3] 0.10493137 1.840222 207
> # Apriori analysis for "whole milk" rules
> wholemilk_rules <- apriori(data = Groceries, parameters = list(supp = 0.001, conf
 = 0.08), appearance = list(rhs = 'whole milk'))
Error:
Invalid parameter: parameters
> # Apriori analysis with increased support and confidence
> grocery_rules_increased_support <- apriori(Groceries, parameter = list(support =
 0.02, confidence = 0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
        0.5    0.1    1 none FALSE            TRUE       5    0.02      1     10
 target  ext
  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 196

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [1 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(head(sort(grocery_rules_increased_support, by = 'confidence'), 3))
    lhs                          rhs          support    confidence coverage
[1] {other vegetables, yogurt} => {whole milk} 0.02226741 0.5128806  0.04341637
    lift     count
[1] 2.007235 219
> # Item Frequency Plot for Groceries dataset
> itemFrequencyPlot(Groceries, topN = 20, type = "absolute", col = brewer.pal(8, 'P
astel2'), main = "Absolute Item Frequency Plot")
> # Import and analyze a transaction dataset (restaurant orders)
> txn <- read.transactions(file = "C:/Users/student/Downloads/restaurant-1-orders.c
sv", rm.duplicates = TRUE, format = "single", sep = ",", header = TRUE, cols = c("O
rder Number", "Item Name"))
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Absolute Item Frequency Plot
item frequency (absolute)
0
500
1000
1500
2000
2500
whole milk
other vegetables
rolls/buns
soda
yogurt
bottled water
root vegetables
tropical fruit
shopping bags
sausage
pastry
citrus fruit
bottled beer
newspapers
canned beer
pip fruit
fruit/vegetable juice
whipped/sour cream
brown bread
domestic eggs

Practical 6

```
library(arules)
library(arulesViz)
library(RColorBrewer)
data<-read.transactions('C:/Users/student/Desktop/supermarket.csv', rm.duplicates= TRUE,
format="single",sep=",",header = TRUE,cols=c("City","Product line"))
str(data)
inspect(head(data))
data@itemInfo$labels
data_rules <- apriori(data, parameter = list(supp = 0.01, conf = 0.2))
data_rules
inspect(data_rules[1:20])
inspect(head(sort(data_rules, by = "confidence"),
10)) inspect(tail(sort(data_rules, by =
"confidence"), 10))
fashion_rules <- apriori(data=data, parameter=list (supp=0.001,conf = 0.08), appearance =
list (rhs="Fashion accessories"))
inspect(head(sort(fashion_rules, by = "confidence"), 10))
fashion_rules_increased_support <- apriori(data, parameter = list(support =0.02, confidence =
0.5))
inspect(head(sort(fashion_rules_increased_support, by = "confidence"), 10))
itemFrequencyPlot(data,topN=20,type="absolute",col=brewer.pal(8,'Pastel2'), main="Absolute
Item Frequency Plot")
```

Output:

Console Terminal Background Jobs

R 4.3.1 · ~/

```
> library(arules)
> library(arulesViz)
> library(RColorBrewer)
> data<-read.transactions('C:/Users/student/Desktop/supermarket.csv', rm.duplicates= TRUE, format
="single",sep=",",header = TRUE,cols=c("City","Product line"))
> str(data)
Formal class 'transactions' [package "arules"] with 3 slots
  ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
  .. .. ..@ i       : int [1:18] 0 1 2 3 4 5 0 1 2 3 ...
  .. .. ..@ p       : int [1:4] 0 6 12 18
  .. .. ..@ Dim     : int [1:2] 6 3
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ : NULL
  .. .. .. ..$ : NULL
  .. .. ..@ factors : list()
  ..@ itemInfo   :'data.frame': 6 obs. of  1 variable:
  .. ..$ labels: chr [1:6] "Electronic accessories" "Fashion accessories" "Food and beverages" "Hea
lth and beauty" ...
  ..@ itemsetInfo:'data.frame': 3 obs. of  1 variable:
  .. ..$ transactionID: chr [1:3] "Mandalay" "Naypyitaw" "Yangon"
> inspect(head(data))
    items                       transactionID
[1] {Electronic accessories,
     Fashion accessories,
     Food and beverages,
     Health and beauty,
     Home and lifestyle,
     Sports and travel}             Mandalay
[2] {Electronic accessories,
     Fashion accessories,
     Food and beverages,
     Health and beauty,
     Home and lifestyle,
     Sports and travel}             Naypyitaw
[3] {Electronic accessories,
```

```
     Sports and travel}          Naypyitaw
[3] {Electronic accessories,
     Fashion accessories,
     Food and beverages,
     Health and beauty,
     Home and lifestyle,
     Sports and travel}          Yangon
> data@itemInfo$labels
[1] "Electronic accessories" "Fashion accessories"    "Food and beverages"
[4] "Health and beauty"      "Home and lifestyle"     "Sports and travel"
> data_rules <- apriori(data, parameter = list(supp = 0.01, conf = 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.2    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 3 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [192 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> data_rules
set of 192 rules
> inspect(data_rules[1:20])
     lhs                         rhs                      support confidence coverage lift
[1]  {}                       => {Electronic accessories} 1       1          1        1
```

```
[1]  {}                       => {Electronic accessories} 1       1          1        1
[2]  {}                       => {Fashion accessories}    1       1          1        1
[3]  {}                       => {Food and beverages}     1       1          1        1
[4]  {}                       => {Health and beauty}      1       1          1        1
[5]  {}                       => {Home and lifestyle}     1       1          1        1
[6]  {}                       => {Sports and travel}      1       1          1        1
[7]  {Electronic accessories} => {Fashion accessories}    1       1          1        1
[8]  {Fashion accessories}    => {Electronic accessories} 1       1          1        1
[9]  {Electronic accessories} => {Food and beverages}     1       1          1        1
[10] {Food and beverages}     => {Electronic accessories} 1       1          1        1
[11] {Electronic accessories} => {Health and beauty}      1       1          1        1
[12] {Health and beauty}      => {Electronic accessories} 1       1          1        1
[13] {Electronic accessories} => {Home and lifestyle}     1       1          1        1
[14] {Home and lifestyle}     => {Electronic accessories} 1       1          1        1
[15] {Electronic accessories} => {Sports and travel}      1       1          1        1
[16] {Sports and travel}      => {Electronic accessories} 1       1          1        1
[17] {Fashion accessories}    => {Food and beverages}     1       1          1        1
[18] {Food and beverages}     => {Fashion accessories}    1       1          1        1
[19] {Fashion accessories}    => {Health and beauty}      1       1          1        1
[20] {Health and beauty}      => {Fashion accessories}    1       1          1        1
     count
[1]  3
[2]  3
[3]  3
[4]  3
[5]  3
[6]  3
[7]  3
[8]  3
[9]  3
[10] 3
[11] 3
[12] 3
[13] 3
[14] 3
```

```
Console   Terminal   Background Jobs
R 4.3.1 · ~/
[16] 3
[17] 3
[18] 3
[19] 3
[20] 3
> inspect(head(sort(data_rules, by = "confidence"), 10))
     lhs                          rhs                      support confidence coverage lift
[1]  {}                        => {Electronic accessories} 1       1          1        1
[2]  {}                        => {Fashion accessories}    1       1          1        1
[3]  {}                        => {Food and beverages}     1       1          1        1
[4]  {}                        => {Health and beauty}      1       1          1        1
[5]  {}                        => {Home and lifestyle}     1       1          1        1
[6]  {}                        => {Sports and travel}      1       1          1        1
[7]  {Electronic accessories} => {Fashion accessories}    1       1          1        1
[8]  {Fashion accessories}    => {Electronic accessories} 1       1          1        1
[9]  {Electronic accessories} => {Food and beverages}     1       1          1        1
[10] {Food and beverages}     => {Electronic accessories} 1       1          1        1
     count
[1]  3
[2]  3
[3]  3
[4]  3
[5]  3
[6]  3
[7]  3
[8]  3
[9]  3
[10] 3
> inspect(tail(sort(data_rules, by = "confidence"), 10))
     lhs                          rhs                    support confidence coverage lift count
[1]  {Fashion accessories,
      Food and beverages,
      Health and beauty,
      Sports and travel}       => {Home and lifestyle}       1          1        1    1     3
[2]  {Fashion accessories,
```

```
Console   Terminal   Background Jobs
R 4.3.1 · ~/
[10] 3
> inspect(tail(sort(data_rules, by = "confidence"), 10))
     lhs                          rhs                    support confidence coverage lift count
[1]  {Fashion accessories,
      Food and beverages,
      Health and beauty,
      Sports and travel}       => {Home and lifestyle}       1          1        1    1     3
[2]  {Fashion accessories,
      Food and beverages,
      Home and lifestyle,
      Sports and travel}       => {Health and beauty}        1          1        1    1     3
[3]  {Fashion accessories,
      Health and beauty,
      Home and lifestyle,
      Sports and travel}       => {Food and beverages}       1          1        1    1     3
[4]  {Food and beverages,
      Health and beauty,
      Home and lifestyle,
      Sports and travel}       => {Fashion accessories}      1          1        1    1     3
[5]  {Electronic accessories,
      Fashion accessories,
      Food and beverages,
      Health and beauty,
      Home and lifestyle}      => {Sports and travel}        1          1        1    1     3
[6]  {Electronic accessories,
      Fashion accessories,
      Food and beverages,
      Health and beauty,
      Sports and travel}       => {Home and lifestyle}       1          1        1    1     3
[7]  {Electronic accessories,
      Fashion accessories,
      Food and beverages,
      Home and lifestyle,
      Sports and travel}       => {Health and beauty}        1          1        1    1     3
[8]  {Electronic accessories,
      Fashion accessories,
```

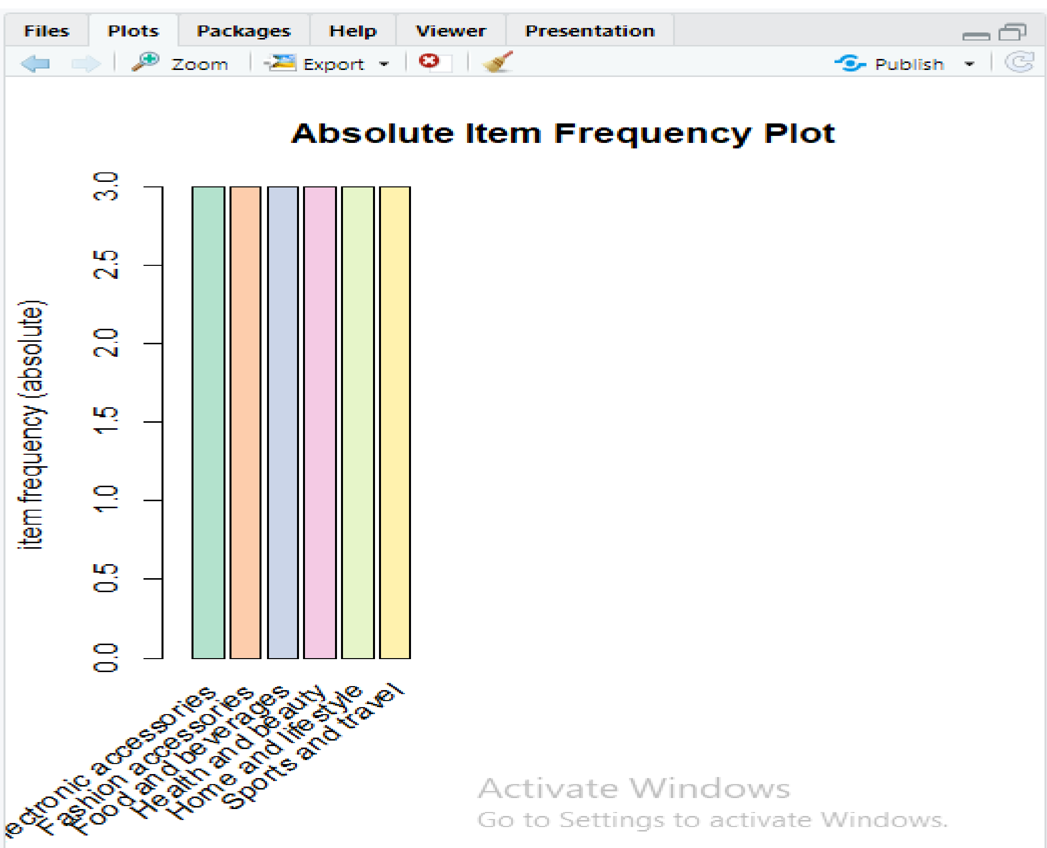
```
Console   Terminal   Background Jobs
R 4.3.1 · ~/
[8]  {Electronic accessories,
      Fashion accessories,
      Health and beauty,
      Home and lifestyle,
      Sports and travel}       => {Food and beverages}          1          1          1    1     3
[9]  {Electronic accessories,
      Food and beverages,
      Health and beauty,
      Home and lifestyle,
      Sports and travel}       => {Fashion accessories}         1          1          1    1     3
[10] {Fashion accessories,
      Food and beverages,
      Health and beauty,
      Home and lifestyle,
      Sports and travel}       => {Electronic accessories}      1          1          1    1     3
> fashion_rules <- apriori(data=data, parameter=list (supp=0.001,conf =  0.08), appearance = list
(rhs="Fashion accessories"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
       0.08    0.1    1 none FALSE            TRUE       5   0.001      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[6 item(s), 3 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [32 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
Console   Terminal   Background Jobs
R 4.3.1 · ~/
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [32 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(head(sort(fashion_rules, by = "confidence"), 10))
     lhs                         rhs                   support confidence coverage lift count
[1]  {}                       => {Fashion accessories}       1          1        1    1     3
[2]  {Electronic accessories} => {Fashion accessories}       1          1        1    1     3
[3]  {Food and beverages}     => {Fashion accessories}       1          1        1    1     3
[4]  {Health and beauty}      => {Fashion accessories}       1          1        1    1     3
[5]  {Home and lifestyle}     => {Fashion accessories}       1          1        1    1     3
[6]  {Sports and travel}      => {Fashion accessories}       1          1        1    1     3
[7]  {Electronic accessories,
      Food and beverages}     => {Fashion accessories}       1          1        1    1     3
[8]  {Electronic accessories,
      Health and beauty}      => {Fashion accessories}       1          1        1    1     3
[9]  {Electronic accessories,
      Home and lifestyle}     => {Fashion accessories}       1          1        1    1     3
[10] {Electronic accessories,
      Sports and travel}      => {Fashion accessories}       1          1        1    1     3
> fashion_rules_increased_support <- apriori(data, parameter = list(support =0.02, confidence = 0.
5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.5    0.1    1 none FALSE            TRUE       5    0.02      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
```

Console Terminal Background Jobs

R 4.3.1 · ~/

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[6 item(s), 3 transaction(s)] done [0.00s].
sorting and recoding items ... [6 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [192 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> inspect(head(sort(fashion_rules_increased_support, by = "confidence"), 10))
     lhs                         rhs                       support confidence coverage lift
[1]  {}                       => {Electronic accessories} 1       1          1        1
[2]  {}                       => {Fashion accessories}    1       1          1        1
[3]  {}                       => {Food and beverages}     1       1          1        1
[4]  {}                       => {Health and beauty}      1       1          1        1
[5]  {}                       => {Home and lifestyle}     1       1          1        1
[6]  {}                       => {Sports and travel}      1       1          1        1
[7]  {Electronic accessories} => {Fashion accessories}    1       1          1        1
[8]  {Fashion accessories}    => {Electronic accessories} 1       1          1        1
[9]  {Electronic accessories} => {Food and beverages}     1       1          1        1
[10] {Food and beverages}     => {Electronic accessories} 1       1          1        1
     count
[1]  3
[2]  3
[3]  3
[4]  3
[5]  3
[6]  3
[7]  3
[8]  3
[9]  3
[10] 3
> itemFrequencyPlot(data,topN=20,type="absolute",col=brewer.pal(8,'Pastel2'), main="Absolute Item F
requency Plot")
>
> |
```



Practical 7

Aim: PageRank Algorithm

Description: PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results**.** PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Code:-

```
ector_dict = {
    "A": [0, 1, 1, 0],
    "B": [0, 0, 0, 0],
    "C": [0, 1, 0, 1],
    "D": [1, 0, 0, 0],
}

df = 0.85

PageRank = {
    "A": 1,
    "B": 1,
    "C": 1,
    "D": 1,
}

columns = {
    "A": 0,
    "B": 1,
    "C": 2,
    "D": 3,
}

def connections(page):
```

```
    column = columns[page]
    incomings = []

    for i in vector_dict.keys():
        if vector_dict[i][column] == 1:
            incomings.append(i)

    return incomings

def outDegree(node):
    count = 0

    for i in vector_dict[node]:
        if i == 1:
            count += 1

    return count

for iteration in range(3):
    for i in PageRank.keys():
        factor = 0
        incoming_nodes = connections(i)

        for node in incoming_nodes:
            factor += PageRank[node] / outDegree(node)

        PageRank[i] = (1 - df) / 4 + df * factor

    print("Iteration", iteration, ":", PageRank)
```

Output:-

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/student/AppData/Local/Programs/Python/Python311/PageRank.py
Iteration 0 : {'A': 0.8875, 'B': 0.8396874999999999, 'C': 0.4146875, 'D': 0.21374218749999999}
Iteration 1 : {'A': 0.21918085937499998, 'B': 0.306894052734375, 'C': 0.130651865234375, 'D': 0.09302704272460938}
Iteration 2 : {'A': 0.11657298631591798, 'B': 0.14257056190887452, 'C': 0.08704351918426514, 'D': 0.07449349565331269
}
>>>
```

Practical 8

Aim:- Topic modelling using LDA

Code:-

```
# Load the necessary libraries
library(topicmodels)
library(tm)

# set the working directory folder of the dataset
setwd("british-fiction-corpus")

# load all the text files in the dataset
filename<-list.files(path=".",pattern="*.txt")
filetext<-lapply(filename,readLines)

# Create a corpus from the dataset
myCorpus<-Corpus(VectorSource(filetext))

# Create a custom stopwords dictionary
custom_stopwords <- c("the", "and", "in", "is", "it", "for", "this", "that")

# Preprocess the text data
myCorpus <- tm_map(myCorpus, content_transformer(tolower))  # Convert to lowercase
myCorpus <- tm_map(myCorpus, removePunctuation)  # Remove punctuation
myCorpus <- tm_map(myCorpus, removeNumbers)  # Remove numbers
myCorpus <- tm_map(myCorpus, removeWords, stopwords("en"))  # Remove common English stopwords
myCorpus <- tm_map(myCorpus, removeWords, custom_stopwords)  # Remove custom stopwords
myCorpus <- tm_map(myCorpus, stripWhitespace)  # Remove extra white spaces

# Create a document-term matrix
dtm<-DocumentTermMatrix(myCorpus)
dtm
```

```
> # Create a document-term matrix
> dtm<-DocumentTermMatrix(myCorpus)
> dtm
<<DocumentTermMatrix (documents: 27, terms: 98167)>>
Non-/sparse entries: 352791/2297718
Sparsity           : 87%
Maximal term length: 54
Weighting          : term frequency (tf)
>
```

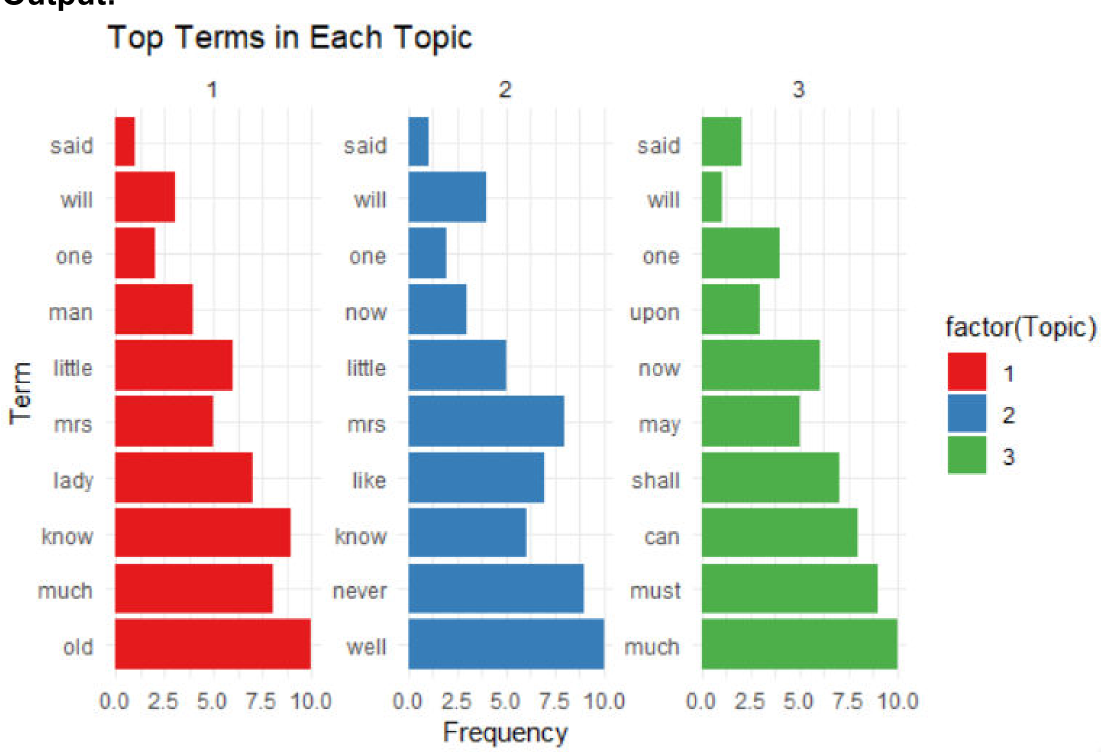
```
# Fit the LDA model
num_topics <- 3  # You can choose the number of topics you want
lda_model <- LDA(dtm, k = num_topics)
```

```r
# Explore topics
topics <- terms(lda_model, 10)  # Get the top 10 terms for each topic

# Print the top terms in each topic
topics
```

```
> topics
      Topic 1  Topic 2  Topic 3
 [1,] "said"   "said"   "will"
 [2,] "one"    "one"    "said"
 [3,] "will"   "now"    "upon"
 [4,] "man"    "will"   "one"
 [5,] "mrs"    "little" "may"
 [6,] "little" "know"   "now"
 [7,] "lady"   "like"   "shall"
 [8,] "much"   "mrs"    "can"
 [9,] "know"   "never"  "must"
[10,] "old"    "well"   "much"
```

```r
# Create a bar plot to visualize the top terms in each topic
library(ggplot2)

topic_terms_df <- data.frame(
  Topic = rep(1:num_topics, each = 10),
  Term = unlist(topics),
  Frequency = rep(1:10, times = num_topics)
)

Term = unlist(topics)
ggplot(topic_terms_df, aes(x = reorder(Term, -Frequency), y = Frequency,
                           fill = factor(Topic))) + geom_bar(stat = "identity") +
  labs(title = "Top Terms in Each Topic", x = "Term", y = "Frequency") +
  theme_minimal() + facet_wrap(~Topic, scales = "free") +
  coord_flip() + scale_fill_brewer(palette = "Set1")
```

Output:



Practical No 9

AIM: To implement MinHashing

DESCRIPTION:

The provided R code employs the textreuse package to perform text similarity analysis and plagiarism detection within a document corpus. It first establishes a Minhash generator with 240 hash functions and generates Minhash signatures for sample text fragments. The code creates a corpus by tokenizing documents into 5-grams, preserving the original tokens. It sets Locality-Sensitive Hashing (LSH) thresholds and probabilities to identify similar documents efficiently. LSH is then applied to cluster similar documents into buckets within the corpus. The code queries LSH to locate documents similar to a specified reference and retrieves potential candidate pairs for similarity assessment using Jaccard similarity. This workflow is valuable for detecting text reuse and potential plagiarism.

ALGORITHM:

1. It sets up a Minhash generator with 240 hash functions for estimating document similarity.
2. Demonstrates Minhash signature generation for sample text fragments.
3. Creates a corpus of documents by tokenizing them into 5-grams, generating Minhash signatures, and retaining original tokens.
4. Defines thresholds and probabilities for Locality-Sensitive Hashing (LSH), a technique for identifying similar documents.
5. Applies LSH to group similar documents into buckets within the corpus.
6. Queries LSH to find documents similar to a specified reference document.
7. Retrieves potential candidate pairs of similar documents.

SOURCE CODE:

```
library(textreuse)
minhash <- minhash_generator(n = 240, seed = 3552)
head(minhash(c("turn tokens into", "tokens into hashes", "into hashes fast")))
```

```
> library(textreuse)
> minhash <- minhash_generator(n = 240, seed = 3552)
> head(minhash(c("turn tokens into", "tokens into hashes", "into hashes fast")))
[1]  -715143991 -1568235737  -501611359 -2123423208  -417352961 -1579395341
```

```
dir <- system.file("extdata/ats", package = "textreuse")
corpus <- TextReuseCorpus(dir = dir, tokenizer = tokenize_ngrams, n = 5,
                   minhash_func = minhash, keep_tokens = TRUE,
                   progress = FALSE)
head(minhashes(corpus[[1]]))
```

```
> dir <- system.file("extdata/ats", package = "textreuse")
> corpus <- TextReuseCorpus(dir = dir, tokenizer = tokenize_ngrams, n = 5,
+                           minhash_func = minhash, keep_tokens = TRUE,
+                           progress = FALSE)
> head(minhashes(corpus[[1]]))
[1] -2147424503 -2147477293 -2147460327 -2147465030 -2147398192 -2147455577
```

```
length(minhashes(corpus[[1]]))
```

lsh_threshold(h = 200, b = 50)
lsh_threshold(h = 240, b = 80)
lsh_probability(h = 240, b = 80, s = 0.25)
lsh_probability(h = 240, b = 80, s = 0.75)

```
> length(minhashes(corpus[[1]]))
[1] 240
> lsh_threshold(h = 200, b = 50)
[1] 0.3760603
> lsh_threshold(h = 240, b = 80)
[1] 0.2320794
> lsh_probability(h = 240, b = 80, s = 0.25)
[1] 0.7163087
> lsh_probability(h = 240, b =  80, s = 0.75)
[1] 1
```

buckets <- lsh(corpus, bands = 80, progress = FALSE)
buckets

```
> buckets <- lsh(corpus, bands = 80, progress = FALSE)
Warning message:
`gather_()` was deprecated in tidyr 1.2.0.
i Please use `gather()` instead.
i The deprecated feature was likely used in the textreuse package.
  Please report the issue at <https://github.com/ropensci/textreuse/issues>.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> buckets
# A tibble: 640 x 2
  doc                buckets
  <chr>              <chr>
1 calltounconv00baxt 73bdd32491559ceccb6dd35862f2e8a1
2 calltounconv00baxt 3357e9e6dc9bfa75bbd33f40d68ed6e5
3 calltounconv00baxt c7a08877e8ff2f7b81099a5be08357bc
```

baxter_matches <- lsh_query(buckets, "calltounconv00baxt")
baxter_matches

```
> baxter_matches <- lsh_query(buckets, "calltounconv00baxt")
> baxter_matches
# A tibble: 1 x 2
  a                  b
  <chr>              <chr>
1 calltounconv00baxt lifeofrevrichard00baxt
```

candidates <- lsh_candidates(buckets)
candidates

```
> candidates <- lsh_candidates(buckets)
> candidates
# A tibble: 3 x 3
  a                     b                      score
  <chr>                 <chr>                  <dbl>
1 calltounconv00baxt    lifeofrevrichard00baxt    NA
2 practicalthought00nev thoughtsonpopery00nevi    NA
3 remember00palm        remembermeorholy00palm    NA
```

lsh_compare(candidates, corpus, jaccard_similarity, progress = FALSE)

```
> lsh_compare(candidates, corpus, jaccard_similarity, progress = FALSE)
# A tibble: 3 x 3
  a                    b                    score
  <chr>                <chr>                <dbl>
1 calltounconv00baxt   lifeofrevrichard00baxt 0.281
2 practicalthought00nev thoughtsonpopery00nevi 0.463
3 remember00palm       remembermeorholy00palm 0.701
> 
```

CONCLUSION:

The overall algorithm utilizes Minhash and Locality-Sensitive Hashing to efficiently identify similar text fragments within a corpus and provides tools for further analysis and comparison of those documents.

Practical 10

Aim: Shingles

Description: Shingles, in the context of data analysis, refer to fixed-length, contiguous subsequences of items within a larger sequence. In text processing, shingling involves the extraction of consecutive words or characters from a document. This technique is commonly used in tasks such as document similarity analysis and information retrieval, allowing for comparisons between documents based on the presence and similarity of their constituent shingles.

Code:

```
readinteger <- function() {
  n <- as.integer(readline(prompt = "Enter value of k-1: "))

  # Check if the file exists
  file_path <- "C:/Users/student/Downloads/a.txt"
  if (!file.exists(file_path)) {
    print("Error: File not found.")
    return(NULL)
  }

  # Read the file using a connection
  con <- file(file_path, open = "r")
  lines <- character(0)

  # Read lines and handle incomplete lines
  while (length(line <- readLines(con, n = 1)) > 0) {
    # Check for incomplete lines and skip them
    if (length(line) > 0) {
      lines <- c(lines, line)
    }
  }

  close(con)  # Close the file connection

  if (length(lines) == 0) {
    print("Error: The file is empty.")
    return(NULL)
  }

  Shingle <- c()

  for (i in 1:(nchar(lines) - n + 1)) {
    Shingle <- append(Shingle, substr(lines, start = i, stop = i + n - 1))
  }

  print(Shingle)
}

# Call the function
```

readinteger()

Output:

```
Enter value of k-1: 2
 [1] "aa" "aa" "aa" "a " " b" "bb" "bb" "bb" "b " " c" "cc" "cc" "cc" "c " " d" "dd" "dd" "dd" "d " " f" "ff" "ff"
[23] "ff" "f " " r" "rr" "rr" "rr" "r," ", " " j" "jj" "jj" "jj" "j " " ." ". " " j" "jj" "jj" "jj" "j," ", " " r"
[45] "rr" "rr" "rr" "r,"
Warning message:
In readLines(con, n = 1) :
  incomplete final line found on 'C:/Users/student/Downloads/a.txt'
> 
```

Practical 11

Aim:- Map Reduce

Software Used: VMware, Cent Os, Eclipse, Cloudera

Description:

Hadoop Word Count is a classic and fundamental example in the world of big data and distributed computing. It demonstrates how to process and analyze a large collection of text documents to count the frequency of each unique word. This example is often used to introduce the Hadoop framework and MapReduce programming model.

Procedure:

On virtual box

- Computer > File System (/) > home > cloudera > downloads > (extract)
- Open Eclipse > create new java project

In project hierarchy

- Src folder right click > build path > configure build path
- New opened window > Libraries >add external JARs..
- Open extracted folder > Select all JAR files
- Right click on project folder in hierarchy > New > Package
- Give package name > finish
- Right click on wc package > New > Class
- Give class name

Program Code:-

```
package wc;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class wcclass {
	public static class Map extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
	private final static IntWritable one = new IntWritable(1);
	private Text word = new Text();

	public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
	String line = value.toString();
	StringTokenizer tokenizer = new StringTokenizer(line);
	while (tokenizer.hasMoreTokens()) {
		word.set(tokenizer.nextToken());
		output.collect(word, one);
	}
```

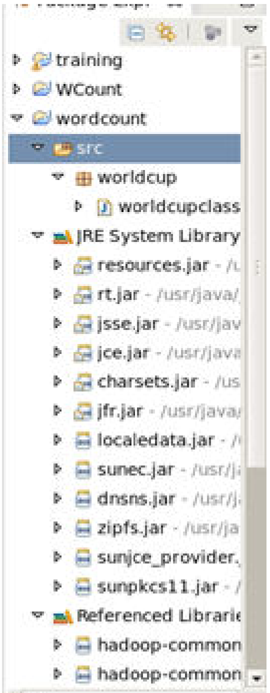
```
        }
        }

        public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
                sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
        }
        }

        public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(wcclass.class);
        conf.setJobName("wordcount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        JobClient.runJob(conf);
        }
}
```

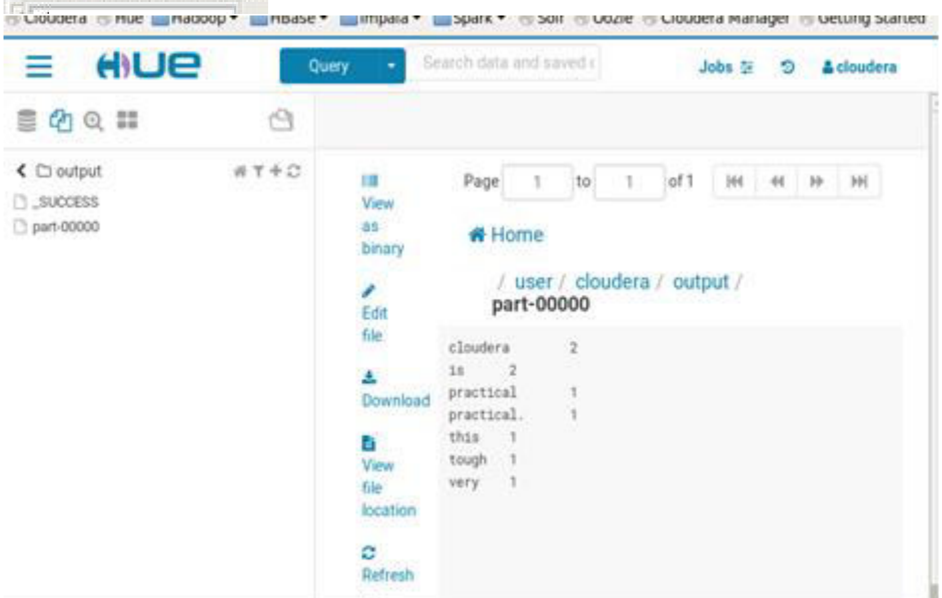
worldcupclass.java

```java
package worldcup;

import java.io.IOException;
import java.util.*;


import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;


public class worldcupclass
{


    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritab
    {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Report
        {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while(tokenizer.hasMoreTokens())
            {
                word.set(tokenizer.nextToken());
                output.collect(word,one);

            }
        }
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritab
    {
        public void reduce(Text key,Iterator<IntWritable>values,OutputCollector<Text,IntWritable>output
        {
            int sum = 0;
            while(values.hasNext())
```

```java
    public static class Reduce extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritab
    {
        public void reduce(Text key,Iterator<IntWritable>values,OutputCollector<Text,IntWritable>output
        {
            int sum = 0;
            while(values.hasNext())
            {
                sum+=values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception
    {
        JobConf conf = new JobConf(worldcupclass.class);
        conf.setJobName("wordcount");


        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);


        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));


        JobClient.runJob(conf);

    }
}
```

Output:-





Conclusion:-

Hadoop has had a profound impact on the field of big data, enabling organizations to efficiently store, process, and analyze vast amounts of data. Its scalability, distributed architecture, and ecosystem of tools make it a crucial component in the big data landscape, helping organizations gain insights and make data-driven decisions. However, it's important to recognize that Hadoop is just one piece of the big data puzzle, and it is often used in conjunction with other technologies to meet specific data processing needs.