Ex: No: 1 **Design and implement a Data Warehouse.**

Date: 7.7.22

Aim:

To Identify source tables and populate sample data., Create the dimension table and fact table in the data warehouse, Design multi-dimensional data models namely Star, Snowflake and Fact Constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, manufacturing, Automobiles, sales etc.).

Procedure:

Step 1: To identify the source tables and populate sample data using Microsoft SQL server Management Studio.
Step2: Create the dimension table and fact table in the data warehouse
Step 3: Design multi-dimensional data models namely Star, Snowflake and Fact Constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, manufacturing, Automobiles, sales etc.).

Coding:

Star Schema:

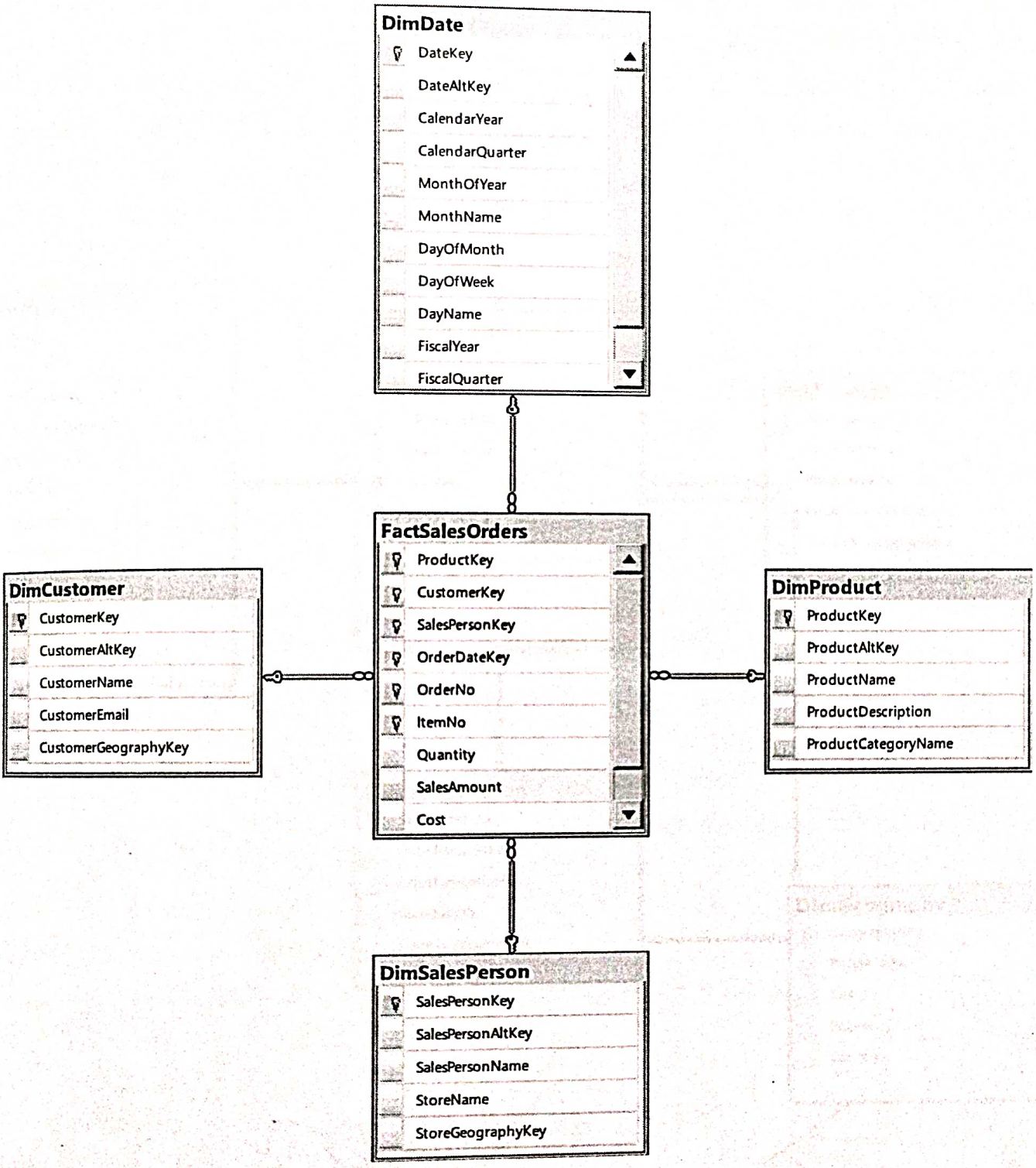
```
use DemoDw
go

create table DimProduct
(ProductKey int identity not null primary key nonclustered,
ProductAltKey nvarchar(10) not null,
ProductName nvarchar(50) null,
ProductDescription nvarchar(100) null,
ProductCategoryName nvarchar(50))
go

create table DimCustomer
(CustomerKey int identity not null primary key nonclustered,
CustomerAltKey nvarchar(10) not null,
CustomerName nvarchar(50) null,
CustomerEmail nvarchar(100) null,
CustomerGeographyKey int null)
go

create table DimSalesPerson
(SalesPersonKey int identity not null primary key nonclustered,
SalesPersonAltKey nvarchar(10) not null,
SalesPersonName nvarchar(50) null,
StoreName nvarchar(100) null,
StoreGeographyKey int null)
go

create table DimDate
(DateKey int not null primary key nonclustered,
DateAltKey datetime not null,
```

```
CalendarYear int not null,
CalendarQuarter int not null,
MonthOfYear int not null,
[MonthName]nvarchar(15)not null,
[DayOfMonth]int not null,
[DayOfWeek]int not null,
[DayName]nvarchar(15)not null,
FiscalYear int not null,
FiscalQuarter int not null)
go

Create table FactSalesOrders
(ProductKey int not null references DimProduct(ProductKey),
CustomerKey int not null references DimCustomer(CustomerKey),
SalesPersonKey int not null references DimSalesPerson(SalesPersonKey),
OrderDateKey int not null references DimDate(DateKey),
OrderNo int not null,
ItemNo int not null,
Quantity int not null,
SalesAmount money not null,
Cost money not null
constraint[PK_FactSalesOrder]primary key nonclustered
(
[ProductKey],[CustomerKey],[SalesPersonKey],[OrderDateKey],[OrderNo],[Item
No]
)
)
```

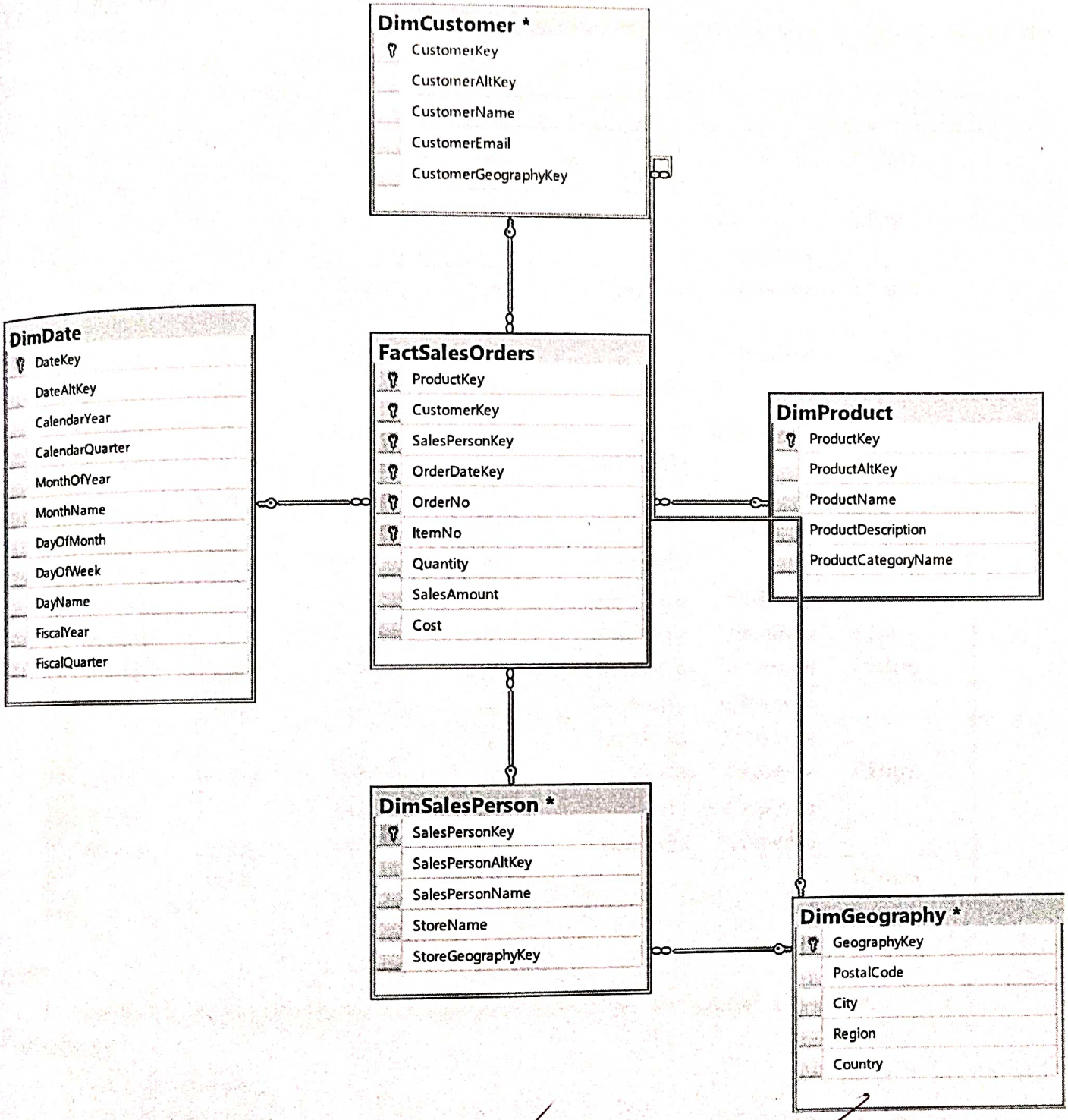
Snow Flake Schema:

```
use DemoDw
go

create table DimGeography
(GeographyKey int identity not null primary key nonclustered,
PostalCode nvarchar(15) null,
City nvarchar(50) null,
Region nvarchar(50) null,
Country nvarchar(50) null)
go
```

Output:
Star Schema:



Snowflake Schema:



Result:

Thus the program for design and emplement a data warehouse was executed successfully.

Date : 14-7-22

Ex. No. 2 APRIORI ALGORITHMS ON DATA SETS

Problem Statement:

The manager of a store is trying to find, which items are bought together the most, out of the given 7. Below is the given dataset.

	A	B	C	D	E	F	G
1	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
2	Bread		Butter	Jam	Nutella		
3			Butter	Jam	Nutella		Chips
4		Milk				Cheese	
5	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
6	Bread	Milk			Nutella		
7	Bread	Milk	Butter	Jam		Cheese	Chips
8	Bread	Milk			Nutella		
9	Bread		Butter			Cheese	
10	Bread		Butter	Jam	Nutella		
11		Milk	Butter	Jam		Cheese	
12	Bread			Jam	Nutella	Cheese	Chips
13	Bread	Milk	Butter	Jam	Nutella		
14	Bread		Butter		Nutella	Cheese	
15	Bread		Butter	Jam	Nutella	Cheese	Chips
16	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
17		Milk	Butter	Jam	Nutella	Cheese	
18		Milk		Jam	Nutella	Cheese	
19	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
20	Bread		Butter	Jam	Nutella	Cheese	
21	Bread	Milk	Butter		Nutella	Cheese	
22		Milk					Chips
23							

Aim:

To identify the frequent itemset for given dataset using an Apriori algorithm.

Procedure:

1. Import the libraries.
2. Load the data set.
3. Display the loaded data.
4. Generate the Apriori model.

Coding:

In [1]:
```
import numpy as np
import pandas as pd
from apyori import apriori
```

In [2]:
```
book_data=pd.read_csv('Book1.csv',header=None)
```

In [4]:
```
book_data
```

Out[4]:

	0	1	2	3	4	5	6
0	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
1	Bread	NaN	Butter	Jam	Nutella	NaN	NaN
2	NaN	NaN	Butter	Jam	Nutella	NaN	Chips
3	NaN	Milk	NaN	NaN	NaN	Cheese	NaN
4	Bread	Milk	Butter	Jam	Nutella	Cheese	Chips
5	Bread	Milk	NaN	NaN	Nutella	NaN	NaN
6	Bread	Milk	Butter	Jam	NaN	Cheese	Chips
7	Bread	Milk	NaN	NaN	Nutella	NaN	NaN
8	Bread	NaN	Butter	NaN	NaN	Cheese	NaN
9	Bread	NaN	Butter	Jam	Nutella	NaN	NaN
10	NaN	Milk	Butter	Jam	NaN	Cheese	NaN
11	Bread	NaN	NaN	Jam	Nutella	Cheese	Chips
12	Bread	Milk	Butter	Jam	Nutella	NaN	NaN

In [6]:
```
#converting dataframe into list
items=[]
for i in range(0,22):
    items.append([str(book_data.values[i,j]) for j in range(0,6)])
#generating apriori model
final_rule=apriori(items,min_support=0.5,min_confidence=0.7,min_lift=1.2,min_length=2)
#grouping all the rules into a list
final_results=list(final_rule)
```

Output:

In [7]:
```
final_results
```

Out[7]:
```
[RelationRecord(items=frozenset({'Butter', 'Jam', 'Nutella'}), support=0.5, ordered_statistics=[OrderedStatistic(items_base=fro
zenset({'Butter', 'Nutella'}), items_add=frozenset({'Jam'}), confidence=0.8461538461538461, lift=1.241025641025641)])]
```

The final rule shows that confidence of the rule is 0.846; it means that out of all transactions that contain 'Butter' and 'Nutella', 84.6% contains 'Jam' too. The lift of 1.24 tells us that 'Jam' is 1.24 times likely to be bought by customers who bought 'Butter' and 'Nutella' compared to the customers who bought 'Jam' separately.

Result:

14/7/22

Thus the Program for Apriori Algorithms on Data sets was executed Successfully.

Date: 21.7.22

Ex.No. 3 TREE BASED CLASSIFICATION ALGORITHMS ON DATA SETS

Aim:

Classify the dataset using Decision tree.

Procedure:

1. Install the necessary packages
2. Load the dataset
3. Implement the decision tree
4. While implementing the decision tree we will go through the following two phases:

Building Phase

- Preprocess the dataset.
- Split the dataset from train and test using Python sklearn package.
- Train the classifier.

Operational Phase

- Make predictions.
- Calculate the accuracy

Coding:

```
pip install graphviz

from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
# Prepare the data data
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Fit the classifier with default hyper-parameters
clf = DecisionTreeClassifier(random_state=1234)
model = clf.fit(X, y)
Print Text Representation
text_representation = tree.export_text(clf)
print(text_representation)
```

If you want to save it to the file, it can be done with following code:

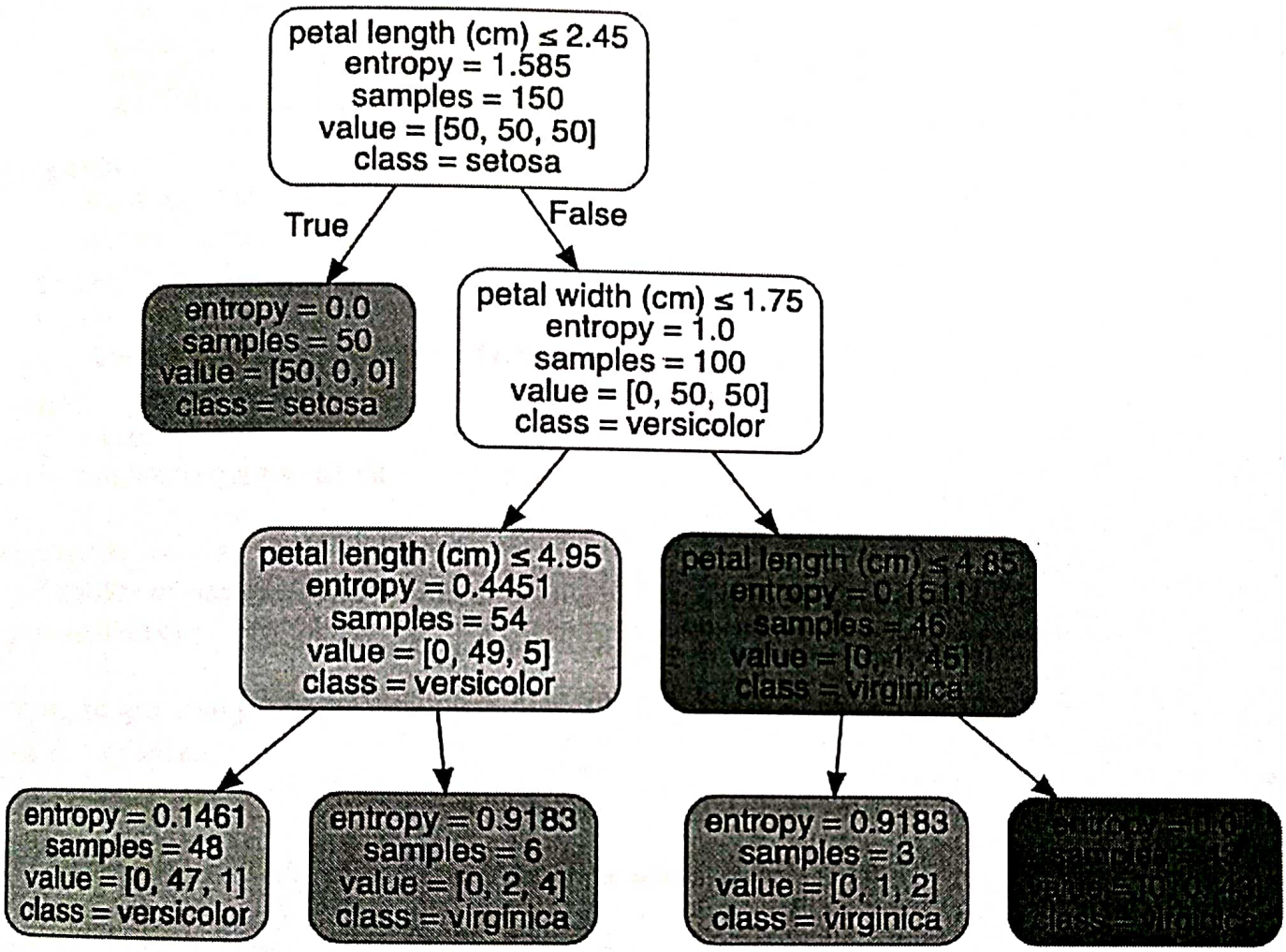
```
with open("decistion_tree.log", "w") as fout:
    fout.write(text_representation)
Plot Tree with plot_tree
fig = plt.figure(figsize=(25,20))
```

= tree.plot_tree(clf, feature_names=iris.feature_names, class_names=iris.target_names, filled=True)

To save the figure to the .png file:

fig.savefig("decistion_tree.png")

Output:



Result:

21/7/22

Thus the Program for Tree Based classification Algorithms on Data sets was executed successfully.

Date:

Ex.No. 4 REGRESSION CLASSIFICATION ALGORITHMS ON DATA SETS

Aim: Classify the given dataset using linear regression.

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

Procedure:

1. Importing the dataset.
2. Visualising the Data.
3. Data Cleaning.
4. Build the Model and Train it.
5. Make Predictions on Unseen Data.

Coding:

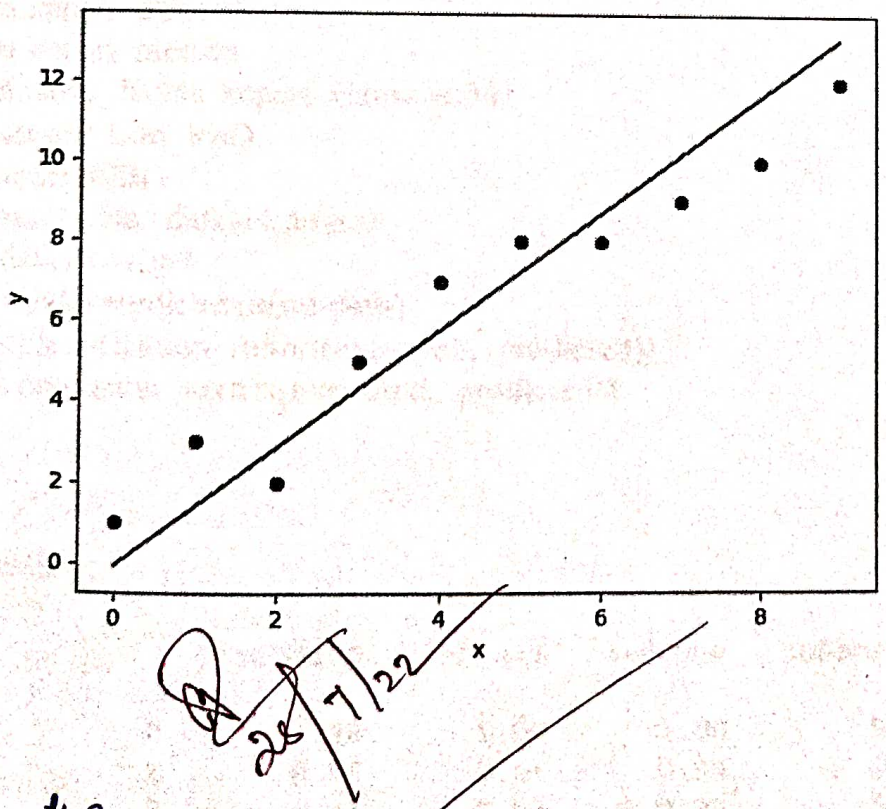
```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x, y):
    # Number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m", marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
```

```
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    # observations / data
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \\nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)

if __name__ == "__main__":
    main()
```

Output:

Estimated coefficients:

- b_0 = -0.0586206896552
- b_1 = 1.45747126437



Result:

Thus the Program was Regression classification Algorithms on Data sets was executed Successfully.

25/7/22

Date: 24.8.22

Ex.No. 5 NAIVE BAYES CLASSIFICATION ALGORITHMS ON DATA SETS

Aim:

Generate the classification using Navie Bayes Classification Algorithm on iris dataset.

Python Packages:

1. **sklearn :**
   - In python, sklearn is a machine learning package which includes a lot of ML algorithms.
   - Here, we are using some of its modules like train_test_split, DecisionTreeClassifier and accuracy_score.
2. **NumPy :**
   - It is a numeric python module which provides fast maths functions for calculations.
   - It is used to read data in numpy arrays and for manipulation purpose.
3. **Pandas :**
   - Used to read and write different files.
   - Data manipulation can be done easily with dataframes.

4. **pip install -U scikit-learn**

Procedure:

1. Importing the dataset.
2. Create Navie Bayes Model using Sklearn.
3. Build the Model and Train it.
4. Get the accuracy of the model.
5. Make Predictions on Unseen Data.

Coding:

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
dataset = datasets.load_iris()
model = GaussianNB()
model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

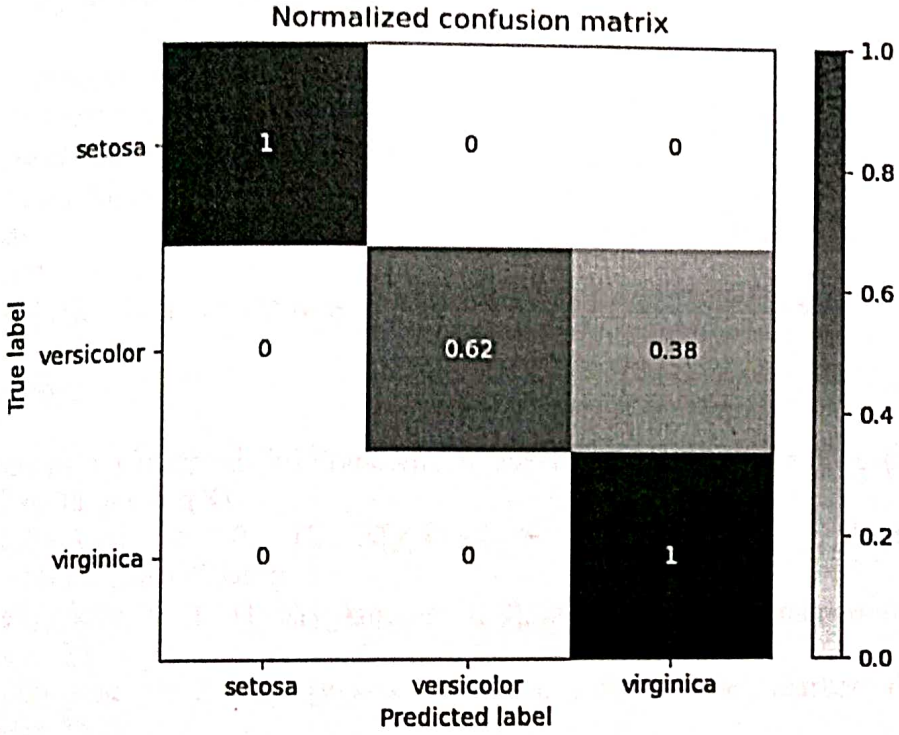
Output:

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.94	0.94	0.94	50
2	0.94	0.94	0.94	50
avg / total	0.96	0.96	0.96	150

Confusion Matrix:

```
[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
```



Result:

Thus the Program for Naive Bayes Classification algorithm Data sets was executed successfully.

Date: 11.8.22

Ex.No. 6 (a) CLUSTERING CLASSIFICATION ALGORITHMS ON DATA SETS

Aim:

Identify the cluster of data objects in a dataset using K-means algorithm.

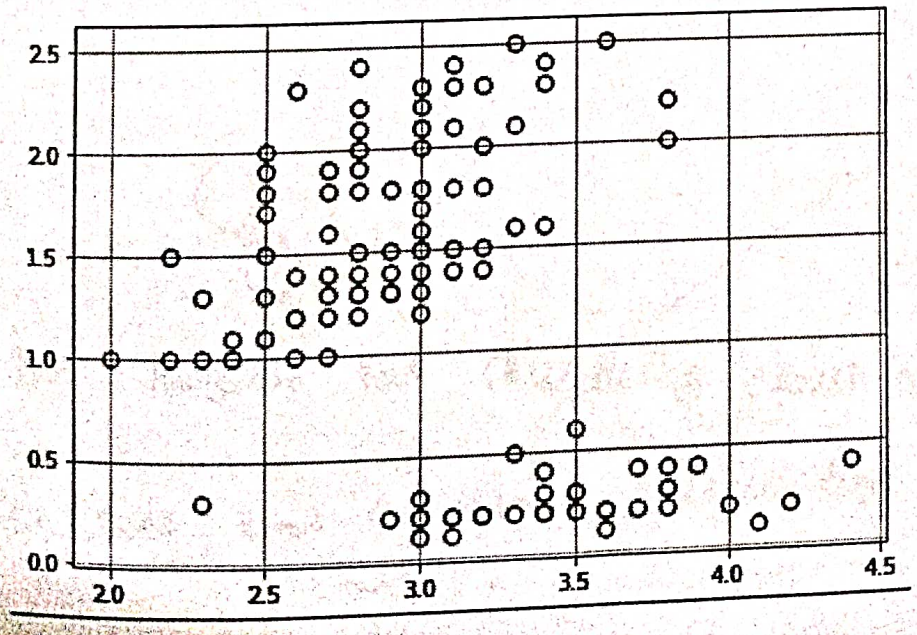
Procedure:

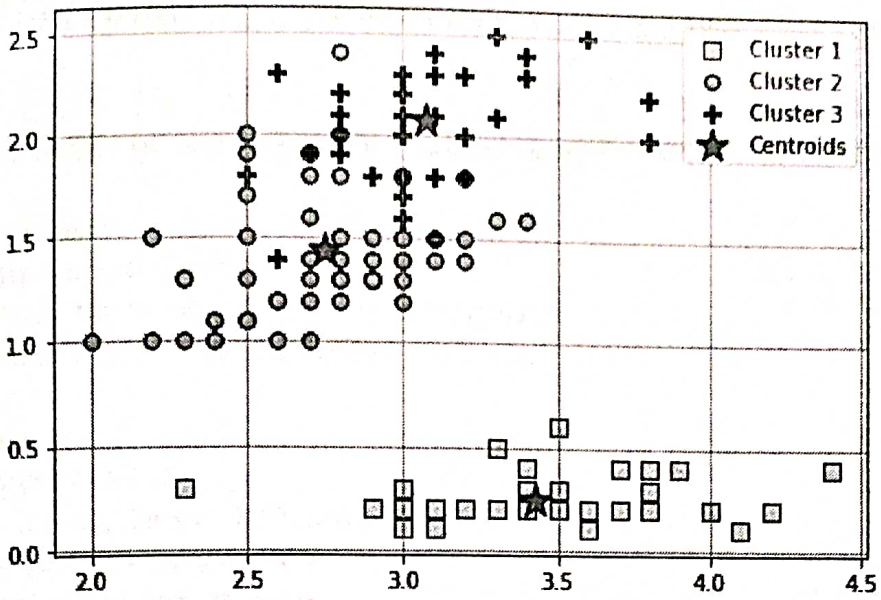
1. Import the dataset
2. Visualize the imported data.
3. Build and train the model using K-means algorithm.
4. Plot the predictions against the original data set.

Coding:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
iris = datasets.load_iris()
X = iris.data
y = iris.target
plt.scatter(X[:,1], X[:,3], color='white', marker='o', edgecolor='red', s=50)
plt.grid()
plt.tight_layout()
plt.show()
kmc = KMeans(n_clusters=3, init='random', n_init=10, max_iter=300,tol=1e-04, random_state=0)
y_kmc = kmc.fit_predict(X)
plt.scatter(X[y_kmc == 0, 1], X[y_kmc == 0, 3], s=50, c='lightgreen', marker='s',
edgecolor='black', label='Cluster 1')
plt.scatter(X[y_kmc == 1, 1], X[y_kmc == 1, 3], s=50, c='orange', marker='o', edgecolor='black',
label='Cluster 2')
plt.scatter(X[y_kmc == 2, 1], X[y_kmc == 2, 3], s=50, c='blue', marker='P', edgecolor='black',
label='Cluster 3')
plt.scatter(kmc.cluster_centers_[:, 1], kmc.cluster_centers_[:, 3], s=250, marker='*', c='red',
edgecolor='black', label='Centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
plt.show()
```

Output:





Result:

Thus the Program for Clustering classification Algorithms on Data sets was executed successfully.

Date: 11-8-22

Ex.No. 6 (b) CLUSTERING CLASSIFICATION ALGORITHMS ON DATA SETS

Aim: Identify the cluster of data objects in a dataset using K-mediods algorithm.

Procedure:

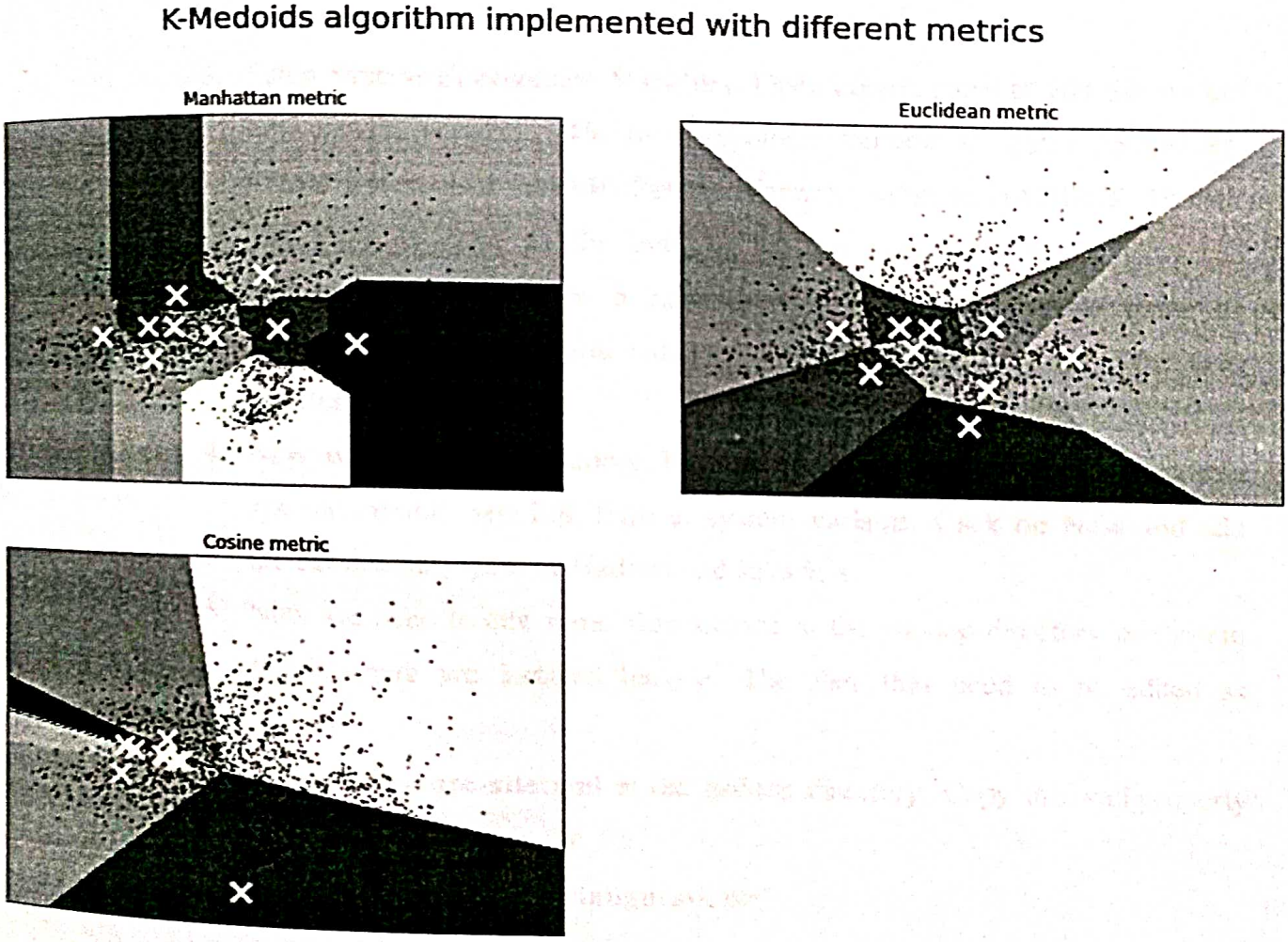
1. Import the dataset
2. Visualize the imported data.
3. Build and train the model using K-means algorithm.
4. Plot the predictions against the original data set.

Coding:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn_extra.cluster import KMedoids
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.datasets import load_digits
dataset = load_digits()
digit_data = scale(dataset.data)
num_digits = len(np.unique(dataset.target))
red_data = PCA(n_components=2).fit_transform(digit_data)
h = 0.02
xmin, xmax = red_data[:, 0].min() - 1, red_data[:, 0].max() + 1
ymin, ymax = red_data[:, 1].min() - 1, red_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))
models = [(KMedoids(metric="manhattan", n_clusters=num_digits, init="heuristic",
max_iter=2),"Manhattan metric",),(KMedoids(metric="euclidean", n_clusters=num_digits,
    init="heuristic", max_iter=2),"Euclidean metric",),(KMedoids(metric="cosine",
n_clusters=num_digits, init="heuristic",max_iter=2), "Cosine metric", ),]
num_rows = int(np.ceil(len(models) / 2.0))
num_cols = 2
plt.clf()
plt.figure(figsize=(15,10))
for i, (model, description) in enumerate(models):
  model.fit(red_data)
  Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
  Z = Z.reshape(xx.shape)
  plt.subplot(num_cols, num_rows, i + 1)
  plt.imshow(
    Z,  #data to be plotted
    interpolation="nearest",extent=(xx.min(), xx.max(), yy.min(), yy.max()),
    cmap=plt.cm.Paired, #colormap
    aspect="auto", #aspect ratio of the axes
    origin="lower",  #set origin as lower left corner of the axes
  )
  plt.plot(
    red_data[:, 0], red_data[:, 1], "k.", markersize=2, alpha=0.3
  )
  centroids = model.cluster_centers_
  plt.scatter(
    centroids[:, 0],
```

```
    centroids[:, 1],
    marker="x",
    s=169, linewidths=3,color="w", zorder=10)
plt.title(description)
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.xticks(())
plt.yticks(())
plt.suptitle("K-Medoids algorithm implemented with different metrics\n\n",fontsize=20)
plt.show()
```

Output:



Result:

Thus the Program for Clustering Classification algorithm Data sets was executed successfully.

11/8/22

Date : 18.8.22

Ex.No. 7 INSTALLATION AND USE OF HADOOP ON WINDOWS OS

Aim: Install Hadoop on Windows OS.

Procedure:

1. To install Hadoop, you should have Java version 1.8 in your system. Check your java version through this command on command prompt.

   java –version

2. After downloading java version 1.8, download **hadoop version 3.1**. Extract it to a folder.
3. Setup System Environment Variables. Open control panel to edit the system environment variable. Go to environment variable in system properties. Create a new user variable. Put the Variable_name as HADOOP_HOME and Variable_value as the path of the bin folder where you extracted hadoop. Likewise, create a new user variable with variable name as JAVA_HOME and variable value as the path of the bin folder in the Java directory.
4. Now we need to set Hadoop bin directory and Java bin directory path in system variable path.Edit Path in system variable. Click on New and add the bin directory path of Hadoop and Java in it.
5. Now we need to edit some files located in the hadoop directory of the etc folder where we installed hadoop. The files that need to be edited as follows.
6. Edit the file **core-site.xml** in the hadoop directory. Copy this xml property in the configuration in the file.

```
<configuration>
  <property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

7. Edit **mapred-site.xml** and copy this property in the configuration.

```
<configuration>
  <property>
```

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

8. Create a folder 'data' in the hadoop directory. Create a folder with the name '**datanode**' and a folder '**namenode**' in this data directory
9. Edit the file **hdfs-site.xml** and add below property in the configuration.

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>C:\Users\hp\Downloads\hadoop-3.1.0\hadoop

3.1.0\data\namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value> C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-

3.1.0\data\datanode</value>

</property>

</configuration>

10. Edit the file **yarn-site.xml** and add below property in the configuration.

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

</configuration>

11. Edit hadoop-env.cmd and replace %JAVA_HOME% with the path of the java folder where your jdk 1.8 is installed.
12. Hadoop needs windows OS specific files which do not come with default download of hadoop.
13. To include those files, replace the bin folder in hadoop directory with the bin folder provided in this github link.

    https://github.com/s911415/apache-hadoop-3.1.0-winutils
14. Download it as zip file. Extract it and copy the bin folder in it. If you want to save the old bin folder, rename it like bin_old and paste the copied bin folder in that directory.
15. Check whether hadoop is successfully installed by running this command on cmd.

hadoop version

Result:

Thus the program for installation and use of Hadoop on windows OS was executed successfully.

Date: 25.8.22

Ex.No. 8 EXECUTE HDFS COMMANDS IN HADOOP ENVIRONMENT

Aim: Execute the Hadoop Commands on Windows OS.

Hadoop Commands:

1. Start namenode and datanode with this command.

   start-dfs.cmd

2. Start yarn through this command.

   start-yarn.cmd

3. Create a directory named 'sample' in my hadoop directory using the following command.

   hdfs dfs –mkdir /sample

4. To verify if the directory is created in hdfs, use **'ls' command** which will list the files present in hdfs.

   hdfs dfs –ls /

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hp supergroup          0 2019-04-07 23:39 /sample

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

5. Copy a text file named 'potatoes' from my local file system to this folder that I just created in hdfs using **copyFromLocal command.**

   hdfs dfs -copyFromLocal C:\Users\hp\Downloads\potatoes.txt /sample

6. To verify if the file is copied to the folder, use **'ls' command** by specifying the folder name which will read the list of files in that folder.

   hdfs dfs –ls /sample

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -ls /sample
Found 1 items
-rw-r--r--   1 hp supergroup       3736 2019-04-07 23:39 /sample/potatoes.txt

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

7. To view the contents of the file we copied, use **cat command.**

   hdfs dfs –cat /sample/potatoes.txt

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -cat /sample/potatoes.txt
area    temp    size    storage method  texture flavor  moistness
1       1       1       1       1       2.9     3.2     3.0
1       1       1       1       2       2.3     2.5     2.6
1       1       1       1       3       2.5     2.8     2.8
1       1       1       1       4       2.1     2.9     2.4
1       1       1       1       5       1.9     2.8     2.2
1       1       1       2       1       1.8     3.0     1.7
1       1       1       2       2       2.6     3.1     2.4
1       1       1       2       3       3.0     3.0     2.9
1       1       1       2       4       2.2     3.2     2.5
1       1       1       2       5       2.0     2.8     1.9
1       1       1       3       1       1.8     2.6     1.5
1       1       1       3       2       2.0     2.8     1.9
1       1       1       3       3       2.6     2.6     2.6
1       1       1       3       4       2.1     3.2     2.1
1       1       1       3       5       2.5     3.0     2.1
1       1       1       4       1       2.6     3.1     2.4
```

8. To Copy file from hdfs to local directory, use **get command.**

hdfs dfs -get /sample/potatoes.txt C:\Users\hp\Desktop\priyanka

9. Displays free space at given hdfs destination use **df command.**

$ hdfs dfs -df hdfs:/

10. Count the number of directories, files and bytes under the paths that match the specified file pattern, use **count command.**

~$ hdfs dfs -count hdfs:/

11. HDFS Command to check the health of the Hadoop files system use **fsck command.**

~$ hdfs fsck /

12. Run a cluster balancing utility use **balancer Command**

~$ hdfs balancer

13. Remove files in hadoop use **rm command**

$hadoop fs -rm /file-name

Or

$hdfs dfs -rm /file-name

14. Remove directories in hadoop use **rmr command**

$hadoop fs -rmr /directory-name

Or

$hdfs dfs -rm / directory-name

15. **Du command** is used to how much file occupied in the disk. The field is the base size of the file or directory before replication.

$ hadoop fs -du /hdfs-file-path

Or

$ hdfs dfs -du /hdfs-file-path

Result:

Thus the Program For Execute HDFS Commands in Hadoop Environment was executed successfully.

Date: 8.9.22

Ex.No. 9 **IMPLEMENTATION OF A MAP REDUCE ALGORITHM**

Problem Statement: Using mapreduce framework, find the frequency of characters in a very large file (running into a few terabytes!). The output consists of two columns - The ASCII character and the number of occurrences of the character in the input file.

We solve this problem using three classes - mapper, reducer and the driver. The driver is the entry point for the mapreduce program. Hadoop mapreduce will use the configured mapper and reducer to compute the desired output.

Prerequisites for Java MapReduce Program

- Java 1.8 or above
- Gradle 3.x or above

Creating the MapReduce Java Project in Gradle

Run the following command on console to create a simple Java project in gradle. Ensure that gradle and java is already installed on the system.

```
gradle init --type java-application
```

This creates an initial set of files for the Java gradle project. Delete App.java and AppTest.java from the new project (contained in src/main/java and src/test/java folders).

Configuring the MapReduce Gradle Build

Replace the build.gradle in the project with the following,

```
apply plugin: 'java-library'
apply plugin: 'application'
mainClassName = "AlphaCounter"
jar {
  manifest { attributes 'Main-Class': "$mainClassName" }
}
repositories { jcenter() }
dependencies { compile 'org.apache.hadoop:hadoop-client:2.7.3' }
```

Writing the Mapper Class

Copy the following class to the src/main/java folder. This is the mapper class for our mapreduce program. The mapreduce framework will pass each line of data as the value variable to the map function. Our program will convert it into a key/value pair where each character becomes a key and the value is set as 1.

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
// A mapper class converting each line of input into a key/value pair
// Each character is turned to a key with value as 1
public class AlphaMapper extends Mapper<Object, Text, Text, LongWritable> {
  private final static LongWritable one = new LongWritable(1);
  private Text character = new Text();
  @Override
```

```
    public void map(Object key, Text value, Context context) throws IOException,
  InterruptedException {
      String v = value.toString();
      for (int i = 0; i < v.length(); i++) {
        character.set(v.substring(i, i + 1));
        context.write(character, one);
      }
    }
  }
```

Writing the Reducer Class

Now copy the following reducer function to src/main/java folder. The mapreduce program will collect all the values for a specific key (a character and its occurrence count in our example) and pass it to the reduce function. Our function computes the total number of occurrences by adding up all the values.

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
// Calculate occurrences of a character
public class AlphaReducer extends Reducer<Text, LongWritable, Text,
LongWritable> {
  private LongWritable result = new LongWritable();

  public void reduce(Text key, Iterable<LongWritable> values, Context context)
      throws IOException, InterruptedException {
    long sum = 0;
    for (LongWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

Writing the MapReduce Entry point Class

Finally copy the main entry point class for our mapreduce program. This sets up the mapreduce job including the name of mapper and reducer classes.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
```

```
import org.apache.hadoop.util.ToolRunner;

// The driver program for mapreduce job.
public class AlphaCounter extends Configured implements Tool {

   public static void main(String[] args) throws Exception {
      int res = ToolRunner.run(new Configuration(), new AlphaCounter(), args);
      System.exit(res);
   }

   @Override
   public int run(String[] args) throws Exception {

      Configuration conf = this.getConf();

      // Create job
      Job job = Job.getInstance(conf, "Alpha Counter Job");
      job.setJarByClass(AlphaCounter.class);

      job.setMapperClass(AlphaMapper.class);
      job.setReducerClass(AlphaReducer.class);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(LongWritable.class);

      FileInputFormat.addInputPath(job, new Path(args[0]));
      job.setInputFormatClass(TextInputFormat.class);

      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      job.setOutputFormatClass(TextOutputFormat.class);

      return job.waitForCompletion(true) ? 0 : 1;
   }
}
```

Output:

Run the following command from the project folder to create a jar file for our project,

```
gradle jar
```

Copy the jar created to the hadoop home folder. Open a command window and navigate to the hadoop home folder.

First create a simple text file with the content "Hello World" and save it as input.txt. Upload the file to HDFS file system using the following command. This will copy the file to hdfs home folder.

```
bin/hdfs dfs -put input.txt .
```

Finally run the mapreduce program from the command line,

```
bin/hadoop jar mapreducedemo.jar ./input.txt output
```

Viewing the MapReduce Output

Run the following command to view the output of the mapreduce program,

```
bin/hdfs dfs -cat output/*
```

The console output consists of every character in "Hello World" and the number of occurrences of each character as shown below.

```
      1
H     1
W     1
d     1
e     1
l     3
o     2
r     1
```

Using a Reducer Program as Combiner

It is possible in mapreduce to configure the reducer as a combiner. A combiner is run locally immediately after execution of the mapper function. Since it is run locally, it substantially improves the performance of the mapreduce program and reduces the data items to be processed in the final reducer stage. Note that combiner can only be used in functions which are commutative and associative.

Add the following line to AlphaCounter.java to configure the reducer as the combiner,

```
job.setCombinerClass(AlphaReducer.class);
```

Result:

Thus the program for implementation of a map reduce Algorithm was executed sucessfully.

Date: 22.9.22

Ex.No. 10 HIVE INSTALLATION AND RUN COMMANDS ON GIVEN DATA

Procedure to install Apache Hive 3.1.2 on Windows 10:

Prerequisites:

1. Java 8
2. 7zip
3. Installing Hadoop
4. Apache Derby 10.14.2.0 - Apache Hive requires a relational database to create its Metastore (where all metadata will be stored).

Steps:

1. Once downloaded, we must extract twice *(using 7zip: the first time we extract the .tar.gz file, the second time we extract the .tar file)* the content of the db-derby-10.14.2.0-bin.tar.gz archive into the desired installation directory. Since in the previous guide we have installed Hadoop within "E:\hadoop-env\hadoop-3.2.1\" directory, we will extract Derby into "E:\hadoop-env\db-derby-10.14.2.0\" directory.

2. **Cygwin :** Since there are some Hive 3.1.2 tools that aren't compatible with Windows (such as schematool). We will need the Cygwin tool to run some Linux commands.

3. **Downloading Apache Hive binaries :** In order to download Apache Hive binaries, you should go to the following website: **https://downloads.apache.org/hive/hive-3.1.2/.** Then, download the **apache-hive-3.1.2.-bin.tar.gz** file.

Name	Last modified	Size	Description
Parent Directory		-	
apache-hive-3.1.2-bin.tar.gz	2019-08-26 20:20	266M	
apache-hive-3.1.2-bin.tar.gz.asc	2019-08-26 20:20	833	
apache-hive-3.1.2-bin.tar.gz.sha256	2019-08-26 20:20	95	
apache-hive-3.1.2-src.tar.gz	2019-08-26 20:20	24M	
apache-hive-3.1.2-src.tar.gz.asc	2019-08-26 20:20	833	
apache-hive-3.1.2-src.tar.gz.sha256	2019-08-26 20:20	95	

4. When the file download is complete, we should extract twice *(as mentioned above)* the apache-hive.3.1.2-bin.tar.gz archive into "E:\hadoop-env\apache-hive-3.1.2" directory (Since we decided to use E:\hadoop-env\" as the installation directory for all technologies used in the previous guide.

5. **Setting environment variables :** After extracting Derby and Hive archives, we should go to Control Panel > System and Security > System. Then Click on "Advanced system settings". In the advanced system settings dialog, click on "**Environment variables**" button. Now we should add the following user variables:

- **HIVE_HOME:** "E:\hadoop-env\apache-hive-3.1.2\"
- **DERBY_HOME:** "E:\hadoop-env\db-derby-10.14.2.0\"
- **HIVE_LIB:** "%HIVE_HOME%\lib"
- **HIVE_BIN:** "%HIVE_HOME%\bin"
- **HADOOP_USER_CLASSPATH_FIRST:** "true"

Besides, we should add the following system variable:

- **HADOOP_USER_CLASSPATH_FIRST: "true"**

Now, we should edit the **Path user variable** to add the following paths:

- **%HIVE_BIN%**
- **%DERBY_HOME%\bin**

6. **Configuring Hive :**

- **Copy Derby libraries**

Now, we should go to the Derby libraries directory (E:\hadoop-env\db-derby-10.14.2.0\lib) and copy all *.jar files. Then, we should paste them within the Hive libraries directory (E:\hadoop-env\apache-hive-3.1.2\lib).

- **Configuring hive-site.xml**

Now, we should go to the Apache Hive configuration directory (E:\hadoop-env\apache-hive-3.1.2\conf) create a new file "hive-site.xml". We should paste the following XML code within this file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration><property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby://localhost:1527/metastore_db;create=true</value>
<description>JDBC connect string for a JDBC metastore</description>
</property><property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.ClientDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>hive.server2.enable.doAs</name>
<description>Enable user impersonation for HiveServer2</description>
<value>true</value>
</property>
<property>
```

```
<name>hive.server2.authentication</name>
<value>NONE</value>
<description> Client authentication types. NONE: no authentication
check LDAP: LDAP/AD based authentication KERBEROS: Kerberos/GSSAPI
authentication CUSTOM: Custom authentication provider (Use with
property hive.server2.custom.authentication.class) </description>
</property>
<property>
<name>datanucleus.autoCreateTables</name>
<value>True</value>
</property>
</configuration>
```

7. **Starting Services:**
   - **Hadoop Services**

   To start Apache Hive, open the command prompt utility as administrator. Then, start the Hadoop services using start-dfs and start-yarn commands.
   - **Derby Network Server**

   Then, we should start the Derby network server on the localhost using the following command:

```
E:\hadoop-env\db-derby-10.14.2.0\bin\StartNetworkServer -h 0.0.0.0
```

8. **Starting Apache Hive:**

   Now, let try to open a command prompt tool and go to the Hive binaries directory (E:\hadoop-env\apache-hive-3.1.2\bin) and execute the following command:

```
Hive
```

9. This error is thrown since the Hive 3.x version is not built for Windows (only in some Hive 2.x versions). To get things working, we should download the necessary *.cmd files from the following link: https://svn.apache.org/repos/asf/hive/trunk/bin/. Note that, you should keep the folder hierarchy (bin\ext\util). You can download all *.cmd files from the following GitHub repository. https://github.com/HadiFadl/Hive-cmd

10. Now if we try to execute the "hive" command, we will receive the following error: As mentioned in the comments, this issue can be solved by replacing the guava-19.0.jar stored in "E:\hadoop-env\apache-hive-3.1.2\lib" with Hadoop's guava-27.0-jre.jar found in "E:\hadoop-env\hadoop-3.2.1\share\hadoop\hdfs\lib".

11. Now, if we run hive command again, then Apache Hive will start successfully.

DDL Commands in Hive

1. Create Database in Hive:

Syntax:

```
CREATE (DATABASE) [IF NOT EXISTS] database_name

[COMMENT database_comment]

[LOCATION hdfs_path]

[WITH DBPROPERTIES (property_name=property_value, ...)];
```

Example:

```
hive> create database if not exists firstDB comment "This is my first

demo" location '/user/hive/warehouse/newdb' with DBPROPERTIES

('createdby'='abhay','createdfor'='dezyre');

OK

Time taken: 0.092 seconds
```

2. Drop Database in Hive

This command is used for deleting an already created database in Hive and the syntax is as follows –

Syntax:

```
DROP (DATABASE) [IF EXISTS] database_name [RESTRICT|CASCADE];
```

Example:

```
hive> drop database if exists firstDB CASCADE;

OK

Time taken: 0.099 seconds
```

In Hadoop Hive, the mode is set as **RESTRICT** by default and users cannot delete it unless it is non-empty. For deleting a database in Hive along with the existing tables, users must change the mode from **RESTRICT to CASCADE.**

3. Describe Database Command in Hive

This command is used to check any associated metadata for the databases.

```
hive> describe database extended firstDB;
OK
firstdb This is my first demo   hdfs://Boss-Machine:9000/user/hive/warehouse/new
db      abhay   USER    {createdby=abhay, createdfor=dezyre}
Time taken: 0.027 seconds, Fetched: 1 row(s)
```

4. Alter Database Command in Hive

Syntax:

```
ALTER (DATABASE) database_name SET DBPROPERTIES
(property_name=property_value, ...);
```

```
hive> alter database firstDB set OWNER ROLE admin;
OK
Time taken: 0.058 seconds
hive> describe database extended firstDB;
OK
firstdb This is my first demo   hdfs://Boss-Machine:9000/user/hive/warehouse/new
db      admin   ROLE    {createdby=abhay, createdfor=dezyre}
Time taken: 0.05 seconds, Fetched: 1 row(s)
```

5. Show Database Command in Hive

Syntax:

```
Show databases;
```

```
hive> Show databases;
OK
default
firstdb
Time taken: 0.039 seconds, Fetched: 2 row(s)
```

6. Use Database Command in Hive

```
hive> use firstdb;
OK
Time taken: 0.027 seconds
```

7. Create Table Command in Hive

Syntax:

```
CREATE  TABLE [IF NOT EXISTS] [db_name.]table_name     --
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
   [LOCATION hdfs_path]
```

```
hive> create  database IF NOT EXISTS college;
OK
Time taken: 0.062 seconds
hive> use college;
OK
Time taken: 0.021 seconds
hive>
    > CREATE TABLE IF NOT EXISTS college.students (
    > ID BIGINT COMMENT 'unique id for each student',
    > name STRING COMMENT 'student name',
    > age INT COMMENT 'sudent age between 16-26',
    > fee DOUBLE COMMENT 'student college fee',
    > city STRING COMMENT 'cities to which students belongs',
    > state STRING COMMENT 'student home address streets',
    > zip BIGINT COMMENT 'student address zip code'
    > )
    > COMMENT 'This table holds the demography info for each student'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE
    > LOCATION '/user/hive/warehouse/college.db/students';
OK
Time taken: 0.317 seconds
```

8. DROP Table Command in Hive

Drops the table and all the data associated with it in the Hive metastore.

Syntax:

```
DROP TABLE [IF EXISTS] table_name [PURGE];
```

```
hive> drop table if exists senior_students PURGE;
OK
Time taken: 0.58 seconds
```

DROP table command removes the metadata and data for a particular table. Data is usually moved to .Trash/Current directory if Trash is configured. If PURGE option is specified then the table data will not go to the trash directory and there will be no scope to retrieve the data in case of erroneous DROP command execution.

9. TRUNCATE Table Command in Hive

This hive command is used to truncate all the rows present in a table i.e. it deletes all the data from the Hive meta store and the data cannot be restored.

Syntax:

```
TRUNCATE TABLE [db_name].table_name
```

```
hive>  truncate table college.senior_students;
OK
Time taken: 0.141 seconds
```

```
hive> create  database IF NOT EXISTS college;
OK
Time taken: 0.062 seconds
hive> use college;
OK
Time taken: 0.021 seconds
hive>
    > CREATE TABLE IF NOT EXISTS college.students (
    > ID BIGINT COMMENT 'unique id for each student',
    > name STRING COMMENT 'student name',
    > age INT COMMENT 'sudent age between 16-26',
    > fee DOUBLE COMMENT 'student college fee',
    > city STRING COMMENT 'cities to which students belongs',
    > state STRING COMMENT 'student home address streets',
    > zip BIGINT COMMENT 'student address zip code'
    > )
    > COMMENT 'This table holds the demography info for each student'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '|'
    -> LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE
    > LOCATION '/user/hive/warehouse/college.db/students';
OK
Time taken: 0.317 seconds
```

8. DROP Table Command in Hive

Drops the table and all the data associated with it in the Hive metastore.

Syntax:

```
DROP TABLE [IF EXISTS] table_name [PURGE];
```

```
hive> drop table if exists senior_students PURGE;
OK
Time taken: 0.58 seconds
```

DROP table command removes the metadata and data for a particular table. Data is usually moved to .Trash/Current directory if Trash is configured. If PURGE option is specified then the table data will not go to the trash directory and there will be no scope to retrieve the data in case of erroneous DROP command execution.

9. TRUNCATE Table Command in Hive

This hive command is used to truncate all the rows present in a table i.e. it deletes all the data from the Hive meta store and the data cannot be restored.

Syntax:

```
TRUNCATE TABLE [db_name].table_name
```

```
hive>  truncate table college.senior_students;
OK
Time taken: 0.141 seconds
```

10. ALTER Table Command in Hive

Using ALTER Table command, the structure and metadata of the table can be modified even after the table has been created. Let's try to change the name of an existing table using the ALTER command –

Syntax:

```
ALTER TABLE [db_name].old_table_name RENAME TO
[db_name].new_table_name;
```

```
hive> ALTER TABLE college.senior_students RENAME TO college.college_students;
OK
Time taken: 0.247 seconds
```

Syntax:

```
ALTER TABLE [db_name].tablename SET TBLPROPERTIES
('property_key'='property_new_value')
```

```
hive> ALTER TABLE college.college_students SET TBLPROPERTIES ('creator' = 'abhay
');
OK
Time taken: 0.148 seconds
```

In the above step, we have set the creator attribute for the table and similarly we can later or modify other table properties also.

```
hive> ALTER TABLE college.college_students SET TBLPROPERTIES ('comment' = 'chang
ed the creater to abhay');
OK
Time taken: 0.161 seconds
```

11. DESCRIBE Table Command in Hive

Gives the information of a particular table and the syntax is as follows –

Syntax:

```
DESCRIBE [EXTENDED|FORMATTED]  [db_name.] table_name[.col_name (
[.field_name]
```

```
hive> describe college.college_students;
OK
id                      bigint                  unique id for each student
name                    string                  student name
age                     int                     sudent age between 16-26
fee                     double                  student college fee
city                    string                  cities to which students belongs
state                   string                  student home address streets
zip                     bigint                  student address zip code
Time taken: 0.064 seconds, Fetched: 7 row(s)
```

```
hive> show tables;
OK
college_students
students
Time taken: 0.018 seconds, Fetched: 2 row(s)
```

DML Commands in Hive

DML (Data Manipulation Language) commands in Hive are used for inserting and querying the data from hive tables once the structure and architecture of the database has been defined using the DDL commands listed above. Data can be loaded into Hive tables using –

- LOAD command
- Insert command

LOAD Command for Inserting Data Into Hive Tables

Syntax:

```
LOAD DATA [LOCAL] INPATH 'hdfsfilepath/localfilepath' [OVERWRITE] INTO
TABLE existing_table_name
```

Example:

```
hive> CREATE TABLE IF NOT EXISTS college.students (
    > ID BIGINT COMMENT 'unique id for each student',
    > name STRING COMMENT 'student name',
    > age INT COMMENT 'sudent age between 16-26',
    > fee DOUBLE COMMENT 'student college fee',
    > city STRING COMMENT 'cities to which students belongs',
    > state  STRING COMMENT 'student home address state s',
    > zip BIGINT COMMENT 'student address zip code'
    > )
```

Let's load the file into the student table

```
hive> LOAD DATA LOCAL INPATH 'student_data.txt' OVERWRITE INTO TABLE students;
Loading data to table college.students
OK
Time taken: 1.056 seconds
```

Let's check if the data has been inserted into the table

```
hive> select * from students;
```

Result:

Thus the Program For Hive installation and run Commands ON Given Data was executed successfully.

Date: 24.9.22

Ex.No. 11 INSTALL HBASE AND APPLY VARIOUS TABLE QUERIES

Procedure to install Hbase on Windows 10:

Prerequisites:

1. Java 8
2. Download Hbase - Download Apache Hbase from this link. (https://hbase.apache.org/downloads.html)

Steps:

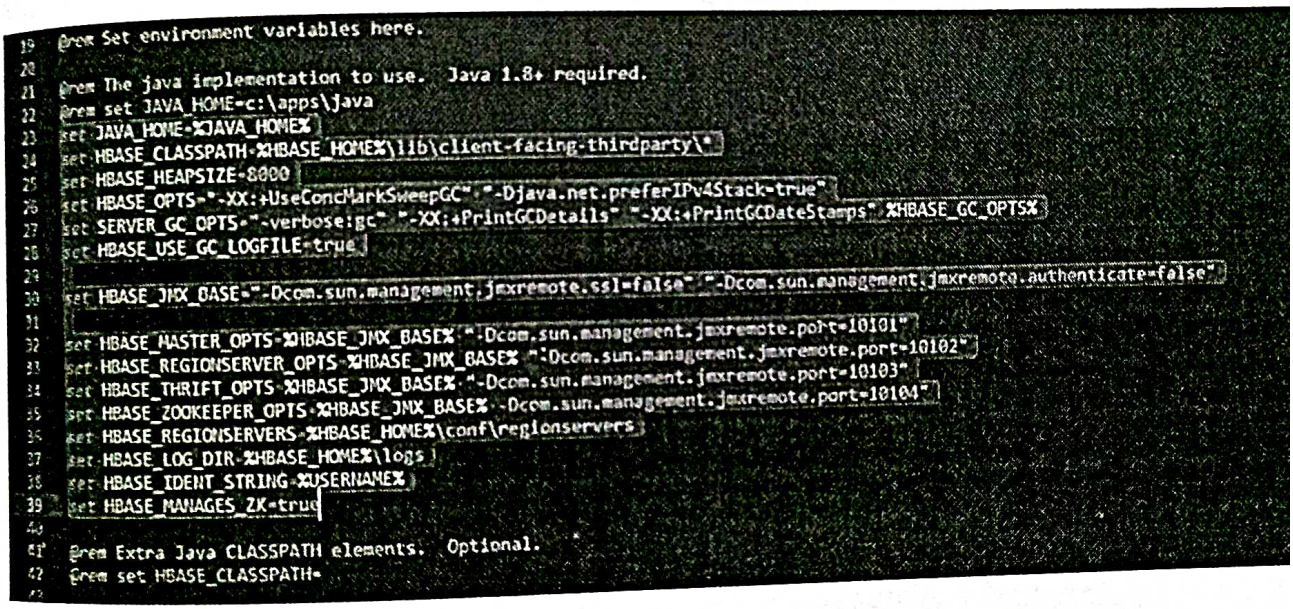
1. Extract all the files in C drive.
2. Create folders named "**hbase**" and "**zookeeper.**"
3. Deleting line in **HBase.cmd.**
   - Open hbase.cmd in any text editor.
   - Search for line **%HEAP_SETTINGS% and remove it.**

```
  )
)
set HBASE_OPTS=%HBASE_OPTS% -Dhbase.security.logger="%HBASE_SECURITY_LOGGER%"

set HEAP_SETTINGS=%JAVA_HEAP_MAX% %JAVA_OFFHEAP_MAX%
set java_arguments=%HBASE_OPTS% -classpath "%CLASSPATH%" %CLASS% %hbase-command-arguments%

if defined service_entry (
  call :makeServiceXml %java_arguments%
) else (
  call %JAVA% %java_arguments%
)

endlocal
goto :eof
```

4. Add lines in **hbase-env.cmd**
   - Now open hbase-env.cmd, which is in the conf folder in any text editor.
   - Add the below lines in the file after the comment session.

```
set JAVA_HOME=%JAVA_HOME%
set HBASE_CLASSPATH=%HBASE_HOME%\lib\client-facing-thirdparty\*
set HBASE_HEAPSIZE=8000
set HBASE_OPTS="-XX:+UseConcMarkSweepGC" "-
Djava.net.preferIPv4Stack=true"
set SERVER_GC_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-
XX:+PrintGCDateStamps" %HBASE_GC_OPTS%
set HBASE_USE_GC_LOGFILE=true
set HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false" "-
Dcom.sun.management.jmxremote.authenticate=false"
set HBASE_MASTER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10101"
```

```
set HBASE_REGIONSERVER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10102"
set HBASE_THRIFT_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10103"
set HBASE_ZOOKEEPER_OPTS=%HBASE_JMX_BASE% -
Dcom.sun.management.jmxremote.port=10104"
set HBASE_REGIONSERVERS=%HBASE_HOME%\conf\regionservers
set HBASE_LOG_DIR=%HBASE_HOME%\logs
set HBASE_IDENT_STRING=%USERNAME%
set HBASE_MANAGES_ZK=true
```



5. Add the line in **Hbase-site.xml**
   - Open **hbase-site.xml**, which is in the conf folder in any text editor.
   - Add the lines inside the <configuration> tag.

```
<property>
   <name>hbase.rootdir</name>
   <value>file:///C:/Documents/hbase-2.2.5/hbase</value>
 </property>
 <property>
   <name>hbase.zookeeper.property.dataDir</name>
   <value>/C:/Documents/hbase-2.2.5/zookeeper</value>
 </property>
 <property>
    <name> hbase.zookeeper.quorum</name>
   <value>localhost</value>
 </property>
```

```
    <value>false</value>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>file:///C:/Documents/hbase-2.2.5/hbase</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/C:/Documents/hbase-2.2.5/zookeeper</value>
  </property>
  <property>
    <name> hbase.zookeeper.quorum</name>
    <value>localhost</value>
  </property>
</configuration>
```

6. **Setting Environment Variables**
   - Variable name: **HBASE_HOME**
   - Variable Value: Put the path of the Hbase folder.
   - We have completed the HBase Setup on Windows 10 procedure.

HBASE DDL COMMANDS

These are the commands that operate on the tables in HBase.

- **create** - Creates a table.
- **list** - Lists all the tables in HBase.
- **disable** - Disables a table.
- **is_disabled** - Verifies whether a table is disabled.
- **enable** - Enables a table.
- **is_enabled** - Verifies whether a table is enabled.
- **describe** - Provides the description of a table.
- **alter** - Alters a table.
- **exists** - Verifies whether a table exists.
- **drop** - Drops a table from HBase.
- **drop_all** - Drops the tables matching the 'regex' given in the command.
- **Java Admin API** - Prior to all the above commands, Java provides an Admin API to achieve DDL functionalities through programming. Under **org.apache.hadoop.hbase.client** package, HBaseAdmin and HTableDescriptor are the two important classes in this package that provide DDL functionalities.

HBASE DML COMMANDS

- **put** - Puts a cell value at a specified column in a specified row in a particular table.
- **get** - Fetches the contents of row or a cell.
- **delete** - Deletes a cell value in a table.
- **deleteall** - Deletes all the cells in a given row.
- **scan** - Scans and returns the table data.
- **count** - Counts and returns the number of rows in a table.
- **truncate** - Disables, drops, and recreates a specified table.
- **Java client API** - Prior to all the above commands, Java provides a client API to achieve DML functionalities, **CRUD** (Create Retrieve Update Delete) operations and more through programming, under org.apache.hadoop.hbase.client package. **HTable Put** and **Get** are the important classes in this package.

Result:

Thus the Program For Install Hbase and Apply Various tables Queries was executed successfully.

24/12/22

HBASE DML COMMANDS

- **put** - Puts a cell value at a specified column in a specified row in a particular table.
- **get** - Fetches the contents of row or a cell.
- **delete** - Deletes a cell value in a table.
- **deleteall** - Deletes all the cells in a given row.
- **scan** - Scans and returns the table data.
- **count** - Counts and returns the number of rows in a table.
- **truncate** - Disables, drops, and recreates a specified table.
- **Java client API** - Prior to all the above commands, Java provides a client API to achieve DML functionalities, **CRUD** (Create Retrieve Update Delete) operations and more through programming, under org.apache.hadoop.hbase.client package. **HTable Put** and **Get** are the important classes in this package.

Result:

Thus the Program For Install Hbase and Apply Various tables Queries was executed successfully.

29/12/22