Program 1

Implement map reduce for word count from a given input text file.

Step1. Type mapper.py and reducer.py using any editor

$sudo Nano mapper.py

mapper.py

```
#! /user/bin/envy
python
"""mapper.py"""

import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
# remove leading and trailing whitespace
        line = line.strip()
# split the line into words
        words = line.split()
# increase counters
        for word in words:
# write the results to STDOUT (standard output);
# what we output here will be the input for the
# Reduce step, i.e. the input for reducer.py
#
# tab-delimited; the trivial word count is 1
                print('%s\t%s' % (word, 1))
```

reducer.py

```
#!/usr/bin/env python
"""reducer.py"""
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
# remove leading and trailing whitespace
        line = line.strip()
# parse the input we got from mapper.py
        word, count = line.split('\t', 1)
# convert count (currently a string) to int
        try:
                count = int(count)
        except ValueError:
# count was not a number, so silently
```

```
# ignore/discard this line
                continue
# this IF-switch only works because Hadoop sorts map output
# by key (here: word) before it is passed to the reducer
        if current_word == word:
                current_count += count
        else:
                if current_word:
# write result to STDOUT
                        print('%s\t%s' % (current_word, current_count))
                current_count = count
                current_word = word
# do not forget to output the last word if needed!
if current_word == word:
        print('%s\t%s' % (current_word, current_count))
```

Step 2. Create input.txt file

$sudo nano input.txt

input.txt

Global Academy of Technology (GAT), established in the year 2001, is one of the most sought-after engineering and management colleges in Bengaluru, Karnataka. Located in a sprawling campus of 10-acre land, GAT is a campus ideal for students to hone their academics in an atmosphere of optimism. The institute is enabled with :

GAT provides ample opportunities for various co-curricular and extra-curricular activities for the students. The campus brims with more than 3500 students and 300 experienced staff involved in effective Teaching and Learning Process. Academics is supplemented with mentoring, peer learning and counselling to ensure holistic development of students. GAT has academic alliances with various institutions, industries, and research organizations to provide industry perspective to the students.

Step 3. Run the map-reduce word count program on UNIX platform

$cat input.txt | python3 mapper.py | sort | python3 reducer.py

Step 4. Run the map-reduce word count program on Hadoop

1. $sudo su hduser
2. $ start-dfs.sh
3. $start-yarn.sh
4. $jps
5. $hafs dfs -put /input.txt /
6. hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming 3.3.4.jar -files ./mapper.py,./reducer.py -mapper "python3 mapper.py" -reducer "python3 reducer.py" -input /5000-8.txt -output /0000009
7. Observe the result in localhost:port_number
8. Download outputfile and view it for wordcount

Program 2

Execute a python program to implement map reduce concepts for printing average salary

mapper.py

```
#!/usr/bin/env python
import sys
# Input format: employee_id, name, salary
for line in sys.stdin:
# Split the input line into fields
        fields = line.strip().split(",")
# Check that the input line has the correct number of fields
        if len(fields) != 3:
                continue
# Extract the salary from the input line
        salary = int(fields[2])
# Emit the salary as the output key and a count of 1 as the output value
        print("{0}\t{1}".format(salary, 1))
```

reducer.py

```
#!/usr/bin/env python
import sys
# Initialize variables to hold the sum of salaries and the count of employees
total_salary = 0
employee_count = 0
# Process input from mapper
for line in sys.stdin:
# Split the input line into key and value
        salary, count = line.strip().split('\t')
# Update the sum of salaries and the count of employees
        total_salary += int(salary) * int(count)
        employee_count += int(count)
# Calculate the average salary
average_salary = total_salary / employee_count
# Output the result
print("Average salary: {0}".format(average_salary))
```

input.txt

1	John Doe	50000
2	Jane Smith	60000
3	Bob Johnson	70000
4	Susan Williams	55000
5	Tom Davis	65000
6	Emily Brown	75000
7	Michael Lee	45000
8	Kelly Green	55000
9	David Kim	80000
10	Michelle Wong	90000

Output

Average salary: 64500.0

Program 3

Execute a Map reduce program for printing maximum salary for a given input file.

mapper.py

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
# Remove leading and trailing whitespace
        line = line.strip()
# Split the line into columns
        columns = line.split('\t')
# Extract the salary column
        salary = columns[2]
# Output the salary with a null key
        print("%s\t%s" % (None, salary))
```

reducer.py

```
import sys
# Initialize the maximum salary to 0
max_salary = 0
# Iterate over each line in the input
for line in sys.stdin:
# Split the line into key and value
        key, value = line.strip().split('\t', 1)
# Convert the value to an integer
        try:
                salary = int(value)
        except ValueError:
                continue
# Update the maximum salary if this salary is higher
        if salary > max_salary:
                max_salary = salary
# Output the final maximum salary
print("Max Salary: %s" % max_salary)
```

input.txt

Alice	Engineering	75000
Bob	Sales	60000
Charlie	Engineering	85000
David	Sales	45000
Eve	Marketing	90000
Frank	Engineering	100000

Output

Max Salary: 100000

Program 4

Execute a Map reduce program for printing sales

mapper.py

```
#!/usr/bin/env python
import sys
import csv
for line in csv.reader(iter(sys.stdin.readline, '')):
        year = line[0]
        sales = line
        print(f"{year}\t{sales}")
```

reducer.py

```
#!/usr/bin/env python
import sys
current_year = None
total_sales = 0
for line in sys.stdin:
    year, sales = line.strip().split('\t')
    if current_year is None:
        current_year = year
    if year == current_year:
        total_sales += int(sales)
    else:
        print(f"{current_year}\t{total_sales}")
        current_year = year
        total_sales = int(sales)
if current_year is not None:
    print(f"{current_year}\t{total_sales}")
```

input.txt

2018	**Jan**	**10000**
2018	Feb	12000
2018	Mar	15000
2019	Jan	20000
2019	Feb	22000
2019	Mar	25000
2020	Jan	30000
2020	Feb	32000
2020	Mar	35000
2018	gyu	500000
2019	uyt	750000

Output

2018 37000

2019 67000

2020 97000

Program 5

5. Execute a python program to implement inverted index

Mapper.py

```
import sys

def mapper():
    for line in sys.stdin:
        line = line.strip()
        document, words = line.split(" ", 1)
        word_list = words.split()
        for word in word_list:
            print(f"{word}\t{document}")

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

Reducer.py

```
import sys

def reducer():
    current_word = None
    doc_list = []

    for line in sys.stdin:
        line = line.strip()
        word, document = line.split("\t", 1)

        if current_word == word:
            doc_list.append(document)
        else:
            if current_word:
                print(f"{current_word}\t{', '.join(set(doc_list))}")
            doc_list = [document]
            current_word = word

    if current_word:
        print(f"{current_word}\t{', '.join(set(doc_list))}")

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

Input.txt

```
doc1 apple banana
doc2 banana orange
doc3 apple mango
doc4 banana apple
```

Output

apple: doc1, doc3, doc4

banana: doc1, doc2, doc4

orange: doc2

mango: doc3

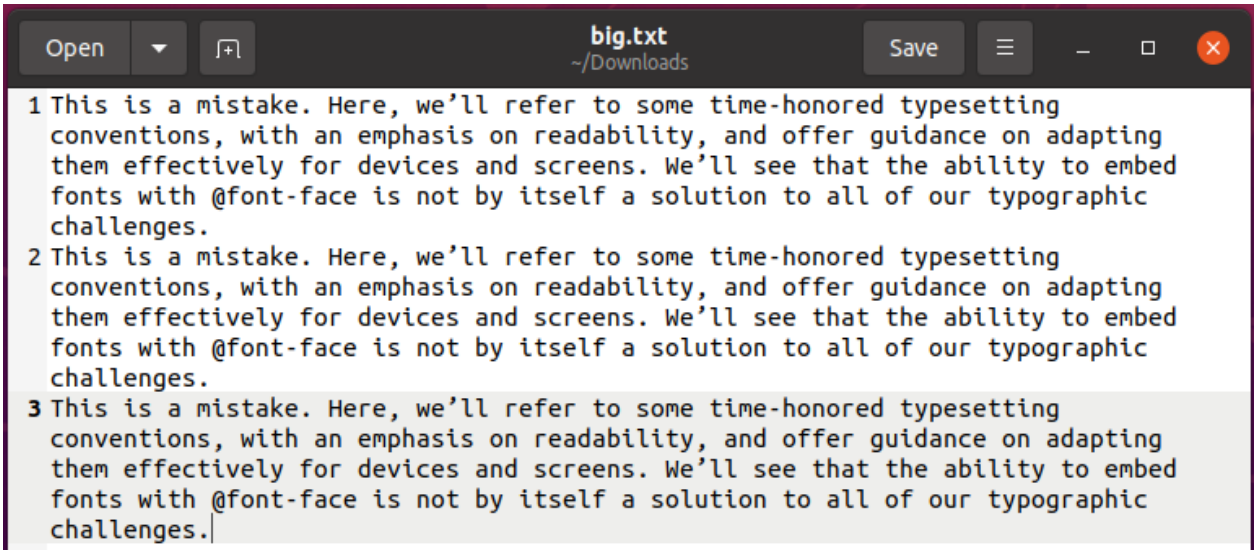
Program 6

Execute python program to implement word count program using spark shell.

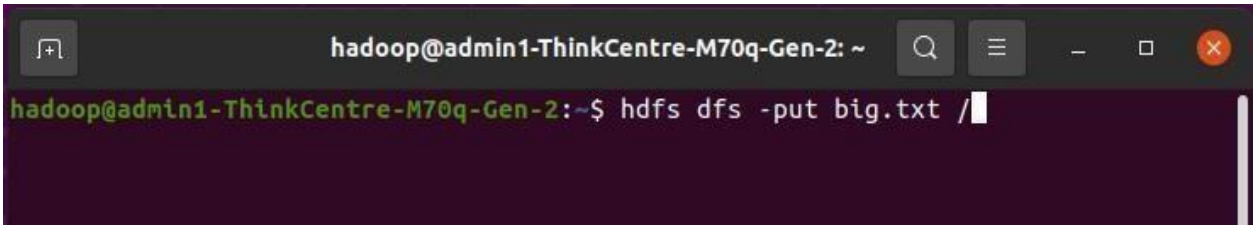
Steps are used to learn how to perform wordcount using spark.

Step 1: Create a Text File and move dfs

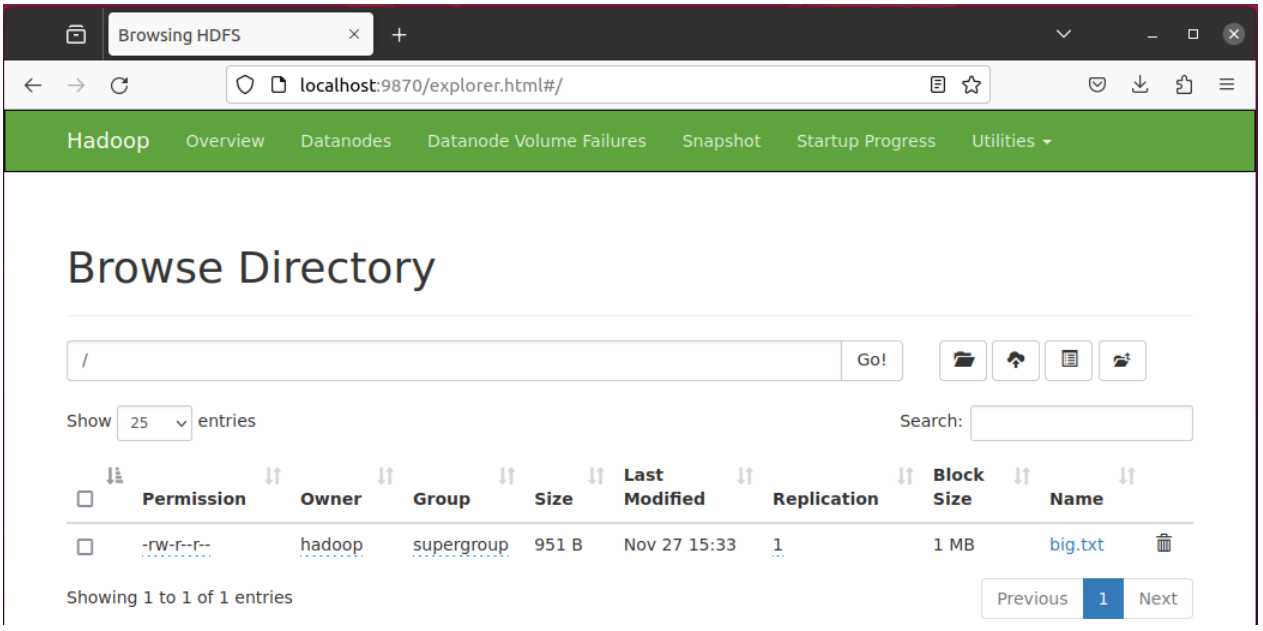
```
gedit big.txt
```




```
hdfs dfs -put big.txt
```

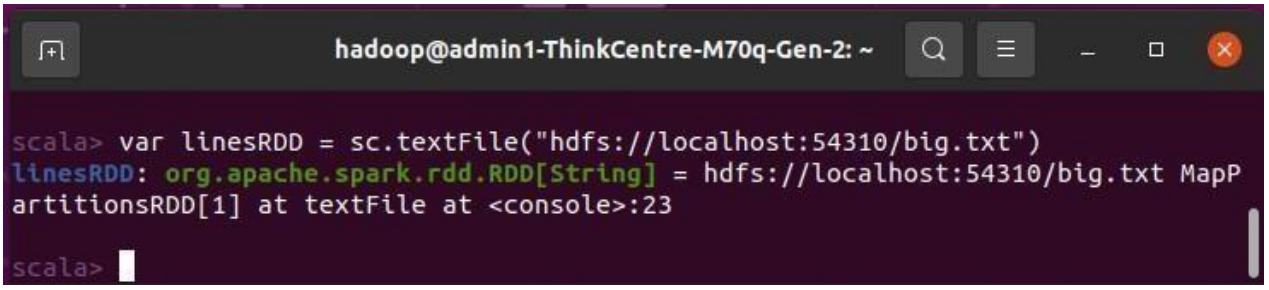



```
localhost:9870
```

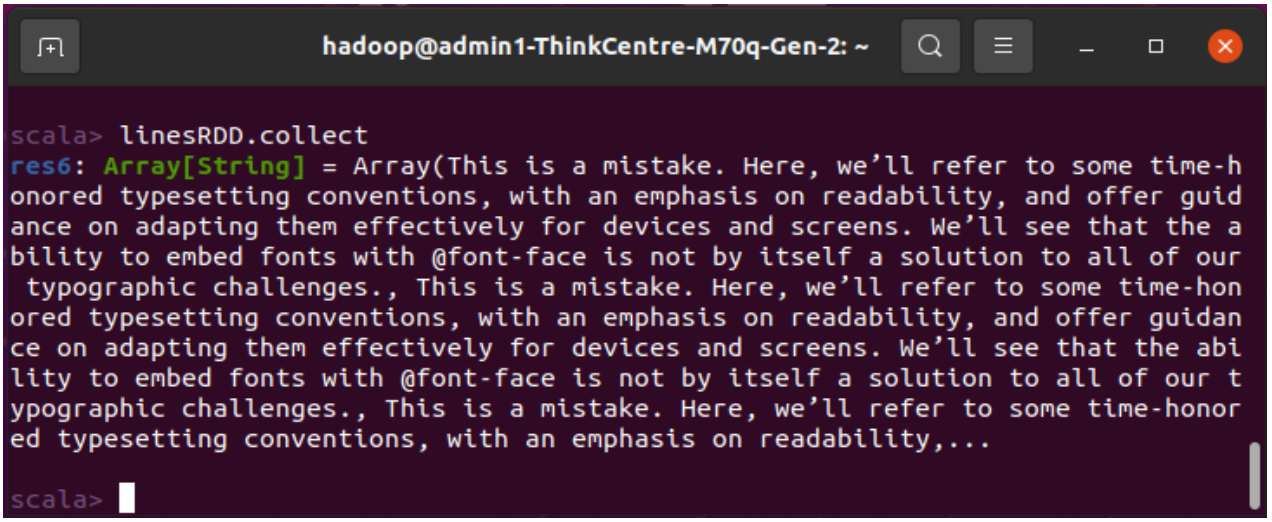


Step 2: Create RDD from a file in HDFS, type the following on spark-shell, and press enter:

```
var linesRDD = sc.textFile("hdfs://localhost:54310/big.txt")
```

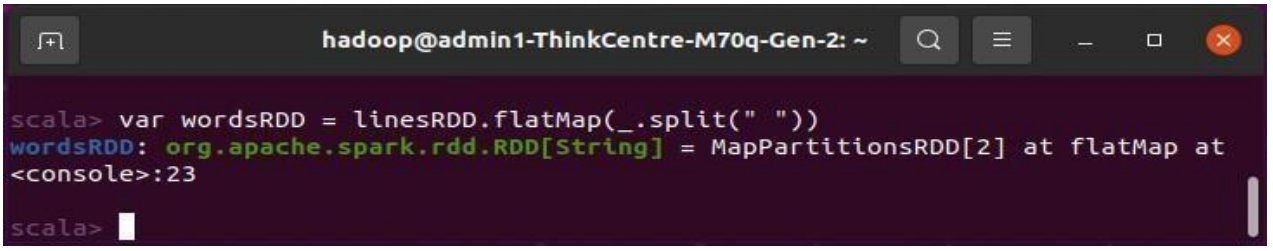



```
linesRDD.collect
```

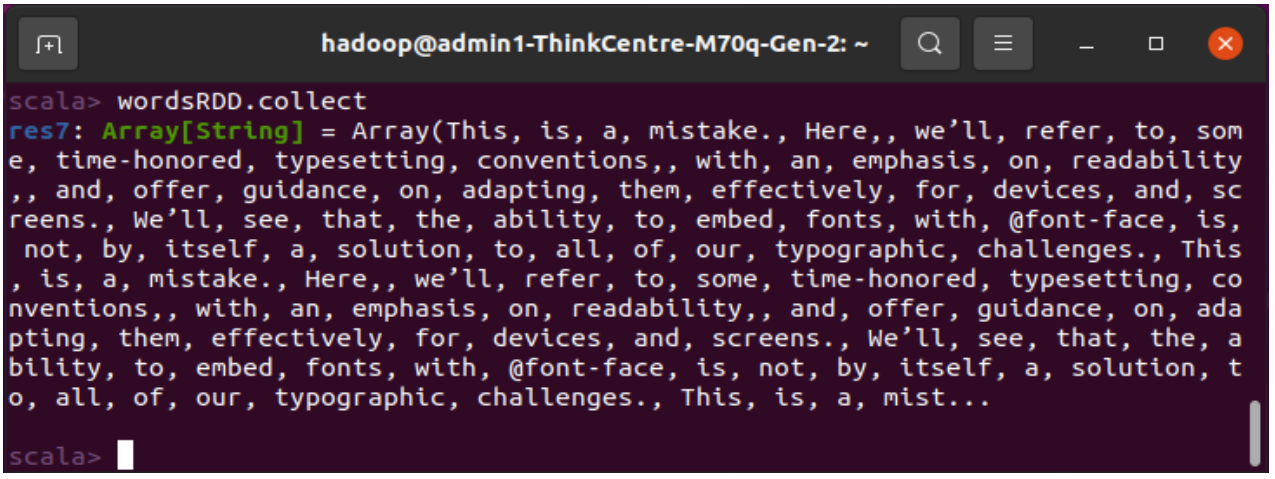


Step 3: Convert each record into word

```
var wordsRDD = linesRDD.flatMap(_.split(" "))
```

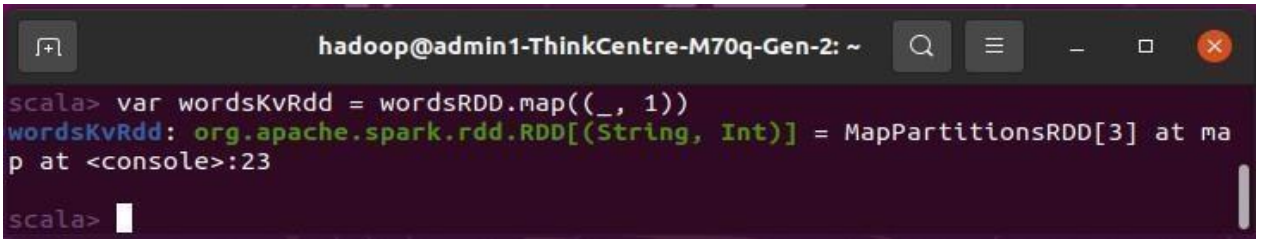



```
wordsRDD.collect
```

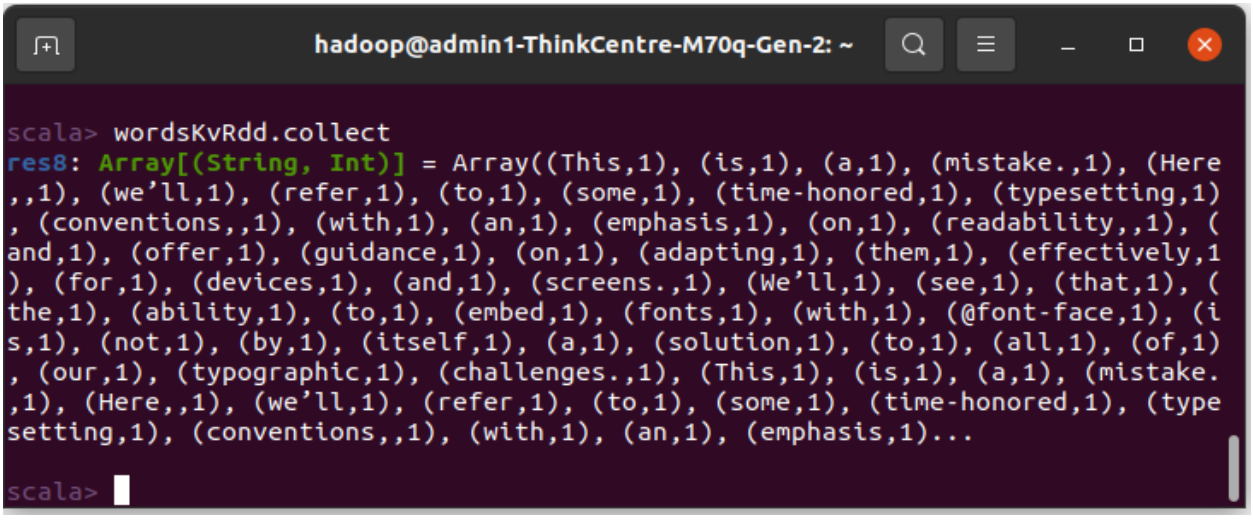


Step 4: Convert each word into key-value pair.

```
var wordsKvRdd = wordsRDD.map((_, 1))
```



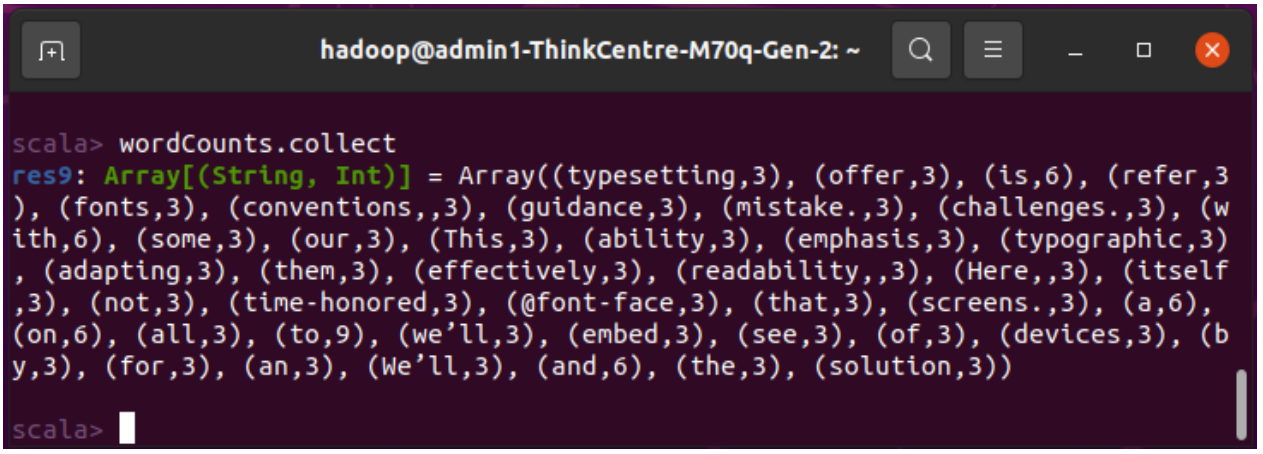

```
wordsKvRdd.collect
```



Step 5: Group By key and perform aggregation on each key:

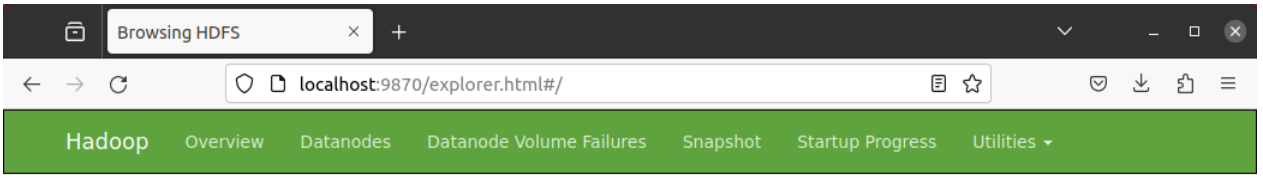
```
var wordCounts = wordsKvRdd.reduceByKey(_ + _ )
```


```
wordCounts.collect
```



Step 6: Save the results into HDFS:

```
wordCounts.saveAsTextFile("hdfs://localhost:54310/output1")
```



Browse Directory

/ Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	hadoop	supergroup	951 B	Nov 27 15:33	1	1 MB	big.txt
☐	drwxr-xr-x	hadoop	supergroup	0 B	Nov 29 09:58	0	0 B	output1

Showing 1 to 2 of 2 entries Previous 1 Next

File information - part-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073741838

Block Pool ID: BP-60461844-127.0.1.1-1700648903486

Generation Stamp: 1014

Size: 253

Availability:

- admin1-ThinkCentre-M70q-Gen-2

File contents

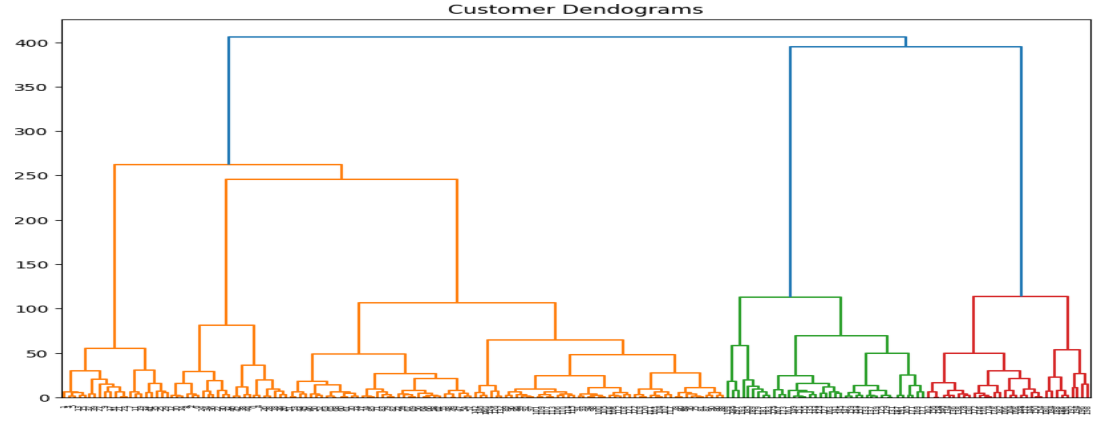
```
(typesetting,3)
(offer,3)
(is,6)
(refer,3)
(fonts,3)
(conventions,,3)
(guidance,3)
(mistake.,3)
```

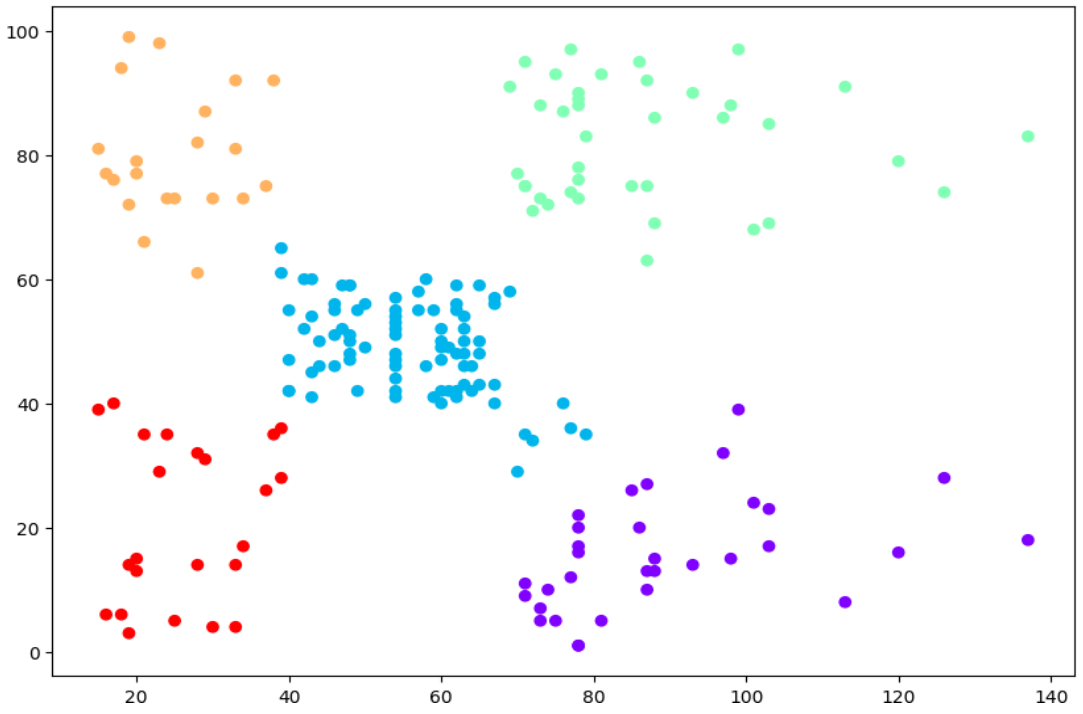
Program 7

Develop a program to Agglomerative Hierarchical clustering.

```
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
import numpy as np

customer_data = pd.read_csv('hierarchical-clustering-with-python-and-scikit-learn-shopping-data.csv')

customer_data.shape
customer_data.head()

data = customer_data.iloc[:, 3:5].values

data

import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
labels_=cluster.fit_predict(data)

labels_

plt.figure(figsize=(10, 7))
plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
```

Output




100
80
60
40
20
0
20
40
60
80
100
120
140

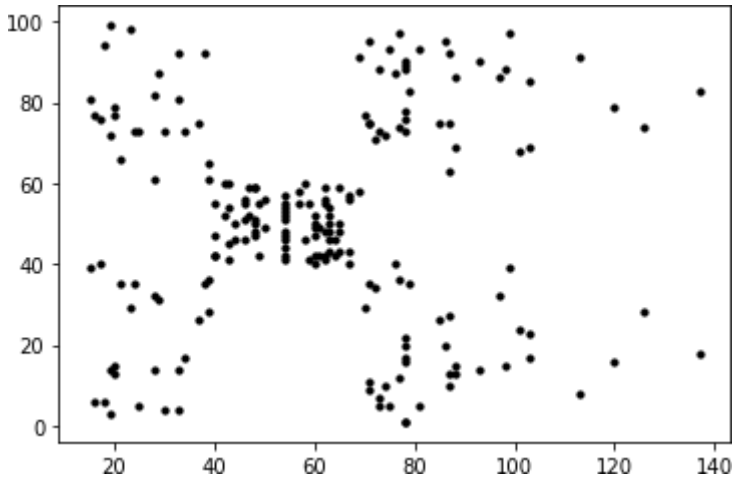
Program 8

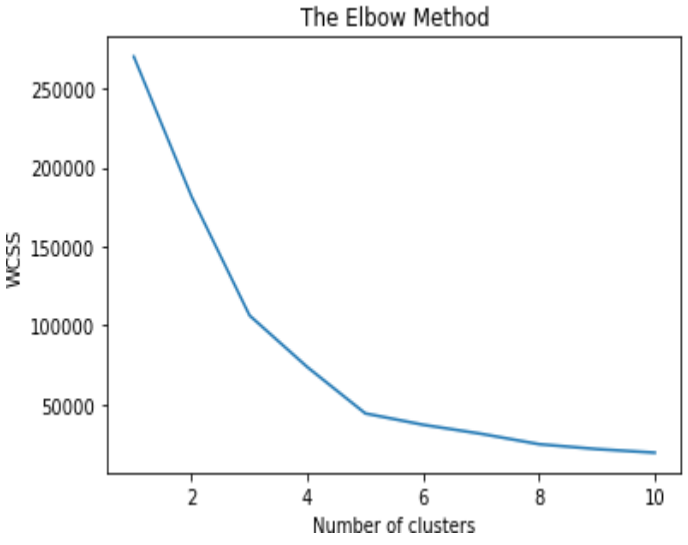
Implement DBSCAN algorithm using appropriate Data sets.

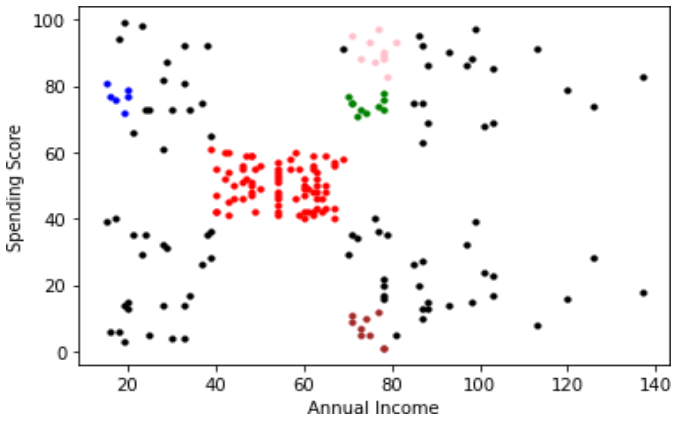
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Mall_customers.csv")

df.head()

df.tail()

df.shape

df = df.iloc[:, [3,4]].values

df

plt.scatter(df[:,0], df[:,1], s=10, c= "black")

from sklearn.cluster import KMeans

wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters= i,
    init = 'k-means++', max_iter= 300, n_init= 10)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11), wcss)
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.show()

from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=5, min_samples=5)

labels = dbscan.fit_predict(df)

np.unique(labels)

# Visualising the clusters
plt.scatter(df[labels == -1, 0], df[labels == -1, 1], s = 10, c = 'black')
plt.scatter(df[labels == 0, 0], df[labels == 0, 1], s = 10, c = 'blue')
plt.scatter(df[labels == 1, 0], df[labels == 1, 1], s = 10, c = 'red')
plt.scatter(df[labels == 2, 0], df[labels == 2, 1], s = 10, c = 'green')
plt.scatter(df[labels == 3, 0], df[labels == 3, 1], s = 10, c = 'brown')
```

```
plt.scatter(df[labels == 4, 0], df[labels == 4, 1], s = 10, c = 'pink')
plt.scatter(df[labels == 5, 0], df[labels == 5, 1], s = 10, c = 'yellow')
plt.scatter(df[labels == 6, 0], df[labels == 6, 1], s = 10, c = 'silver')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```

Output:







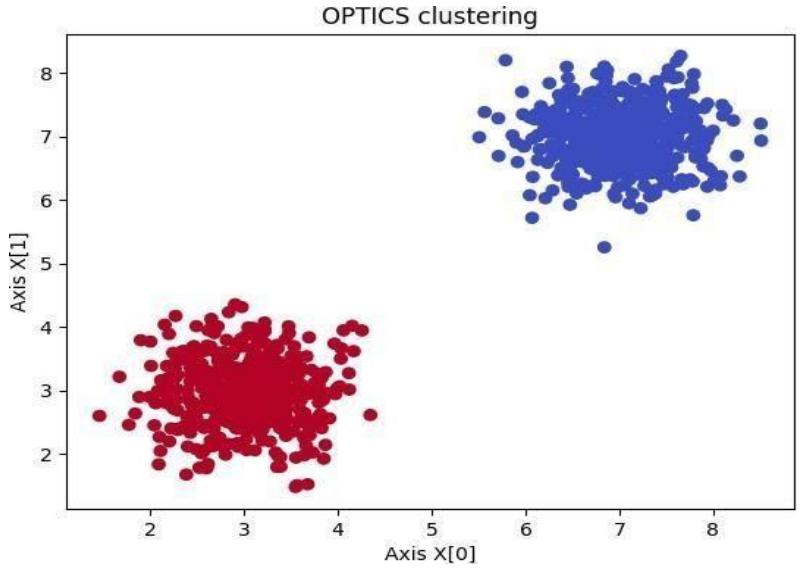
Program 9

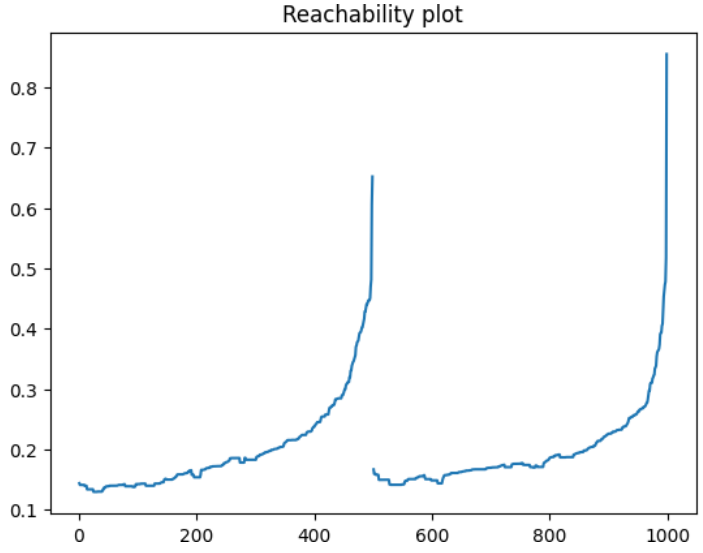
Implement OPTICS algorithm using appropriate Data sets.

```
from sklearn.datasets import make_blobsfrom
sklearn.cluster import OPTICS import numpy
as np
import matplotlib.pyplot as plt

# Configuration options num_samples_total
= 1000
cluster_centers = [(3,3), (7,7)] num_classes
= len(cluster_centers)epsilon = 2.0
min_samples = 22
cluster_method = 'xi'
metric = 'minkowski'

# Generate data
X, y = make_blobs(n_samples = num_samples_total, centers = cluster_centers,n_features =
num_classes, center_box=(0, 1), cluster_std = 0.5)

# Compute OPTICS
db = OPTICS(max_eps=epsilon, min_samples=min_samples,
cluster_method=cluster_method, metric=metric).fit(X)
labels = db.labels_

no_clusters = len(np.unique(labels) )
no_noise = np.sum(np.array(labels) == -1, axis=0)

print('Estimated no. of clusters: %d' % no_clusters)
print('Estimated no. of noise points: %d' % no_noise)

# Generate scatter plot for training data
colors = list(map(lambda x: '#3b4cc0' if x == 1 else '#b40426', labels))
plt.scatter(X[:,0], X[:,1], c=colors, marker="o", picker=True) plt.title(f'OPTICS
clustering')
plt.xlabel('Axis X[0]')
plt.ylabel('Axis X[1]')
plt.show()


#Generate reachability plot
reachability = db.reachability_[db.ordering_]
plt.plot(reachability)
plt.title('Reachability plot')
plt.show()
```

Output:

Estimated no. of clusters: 3
Estimated no. of noise points: 30





Program 10

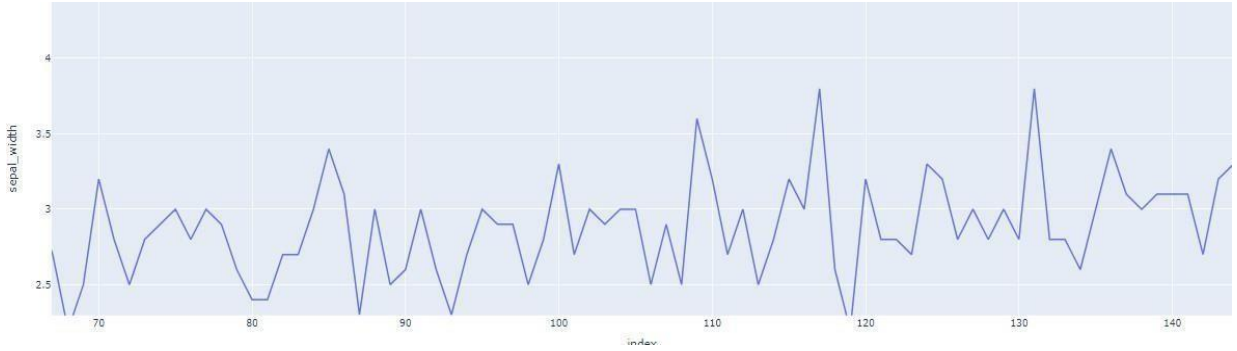
Implement data visualization using Plotly.

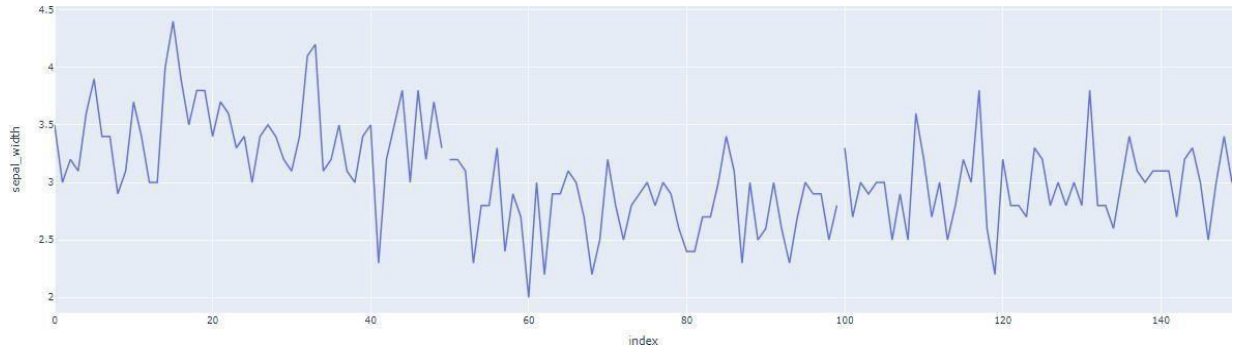
Plotly is an open-source module of Python which is used for data visualization and supports various graphs like line charts, scatter plots, bar charts, histograms, area plot, etc. In this article, we will see how to plot a basic chart with plotly and also how to make a plot interactive. But beforestarting you might be wondering why there is a need to learn plotly, so let's have a look atit.

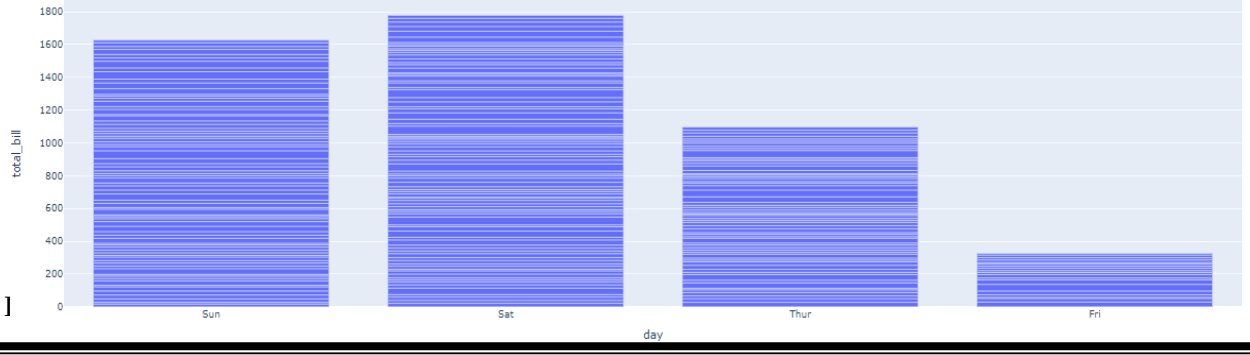
```
import plotly.express as px
# using the iris dataset
df = px.data.iris()

# plotting the line chart
fig = px.line(df, y="sepal_width",)
# showing the plot
fig.show()
# plotting the line chart
fig = px.line(df, y="sepal_width", line_group='species')

# showing the plot
fig.show()

df = px.data.tips()
# Creating the bar chart
fig = px.bar(df, x='day', y="total_bill")
fig.show()
# using the dataset
df = px.data.tips()
 # plotting the histogram
fig = px.histogram(df, x="total_bill")
# showing the plot
fig.show()
df = px.data.tips()
# plotting the histogram
fig = px.histogram(df, x="total_bill", color='sex', nbins=50, histnorm='percent',barmode='overlay')

# showing the plot
fig.show()
df = px.data.tips()
fig = px.pie(df, values="total_bill", names="day")
fig.show()
df = px.data.tips()
# plotting the boxplot
```

```
fig = px.box(df, x="day", y="tip")
# showing the plot
fig.show()
# using the dataset
df = px.data.tips()
# plotting the violin plot
fig = px.violin(df, x="day", y="tip")
# showing the plot
fig.show()
# data to be plotted
df =  px.data.tips()#
plotting the figure
fig = px.scatter_3d(df, x="total_bill", y="sex", z="tip")
fig.show()
```

Output:







]