Programming Assignment-2

CS5433, Big Data Management

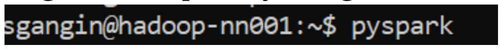
Readme-Part1

Steps for Execution

Line by Line Execution

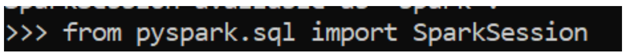
For Method 1

1. Log into the spark by using the below command



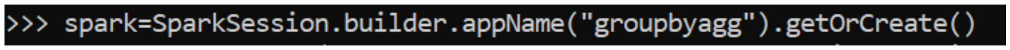
```
sgangin@hadoop-nn001:~$ pyspark
```

2. Then import the Spark Session from pyspark.sql by using below command



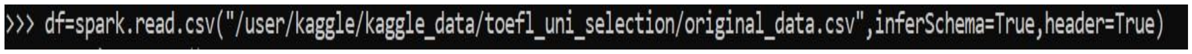
```
>>> from pyspark.sql import SparkSession
```

3. create spark session “groupbyagg” by using below command



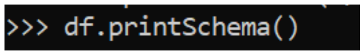
```
>>> spark=SparkSession.builder.appName("groupbyagg").getOrCreate()
```

4. To read a CSV file create data frame Reader



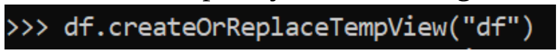
```
>>> df=spark.read.csv("/user/kaggle/kaggle_data/toefl_uni_selection/original_data.csv",inferSchema=True,header=True)
```

5. To view the schema, use the below command



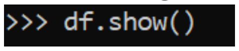
```
>>> df.printSchema()
```

6. To create a temporary view for the given data, use the below command



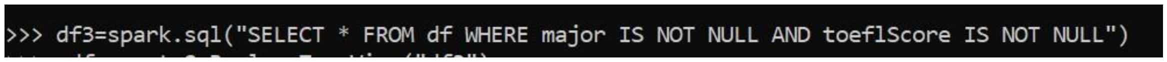
```
>>> df.createOrReplaceTempView("df")
```

7. To view the given data into tabular form, use the below command



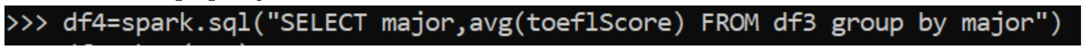
```
>>> df.show()
```

8. To select not null columns and after to create temporary view use below commands



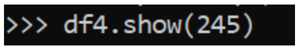
```
>>> df3=spark.sql("SELECT * FROM df WHERE major IS NOT NULL AND toeflScore IS NOT NULL")
```

9. To run the sql query us the below command



```
>>> df4=spark.sql("SELECT major,avg(toeflScore) FROM df3 group by major")
```

10. To view the result use below command



```
>>> df4.show(245)
```

NOTE:-FOR METHOD-1 ONLY UPTO THIS LINE MUST BE EXECUTED BELOW IT IS FOR STORING OUTPUT SHOWN ON SCREEN TO A HDFS FILE. THIS IS BECAUSE IN SPARK-SHELL IT WILL BE COMING INDENTATION ERRORS, IF GIVEN CORRECTLY IT CAN GIVE YOU IN A FILE).

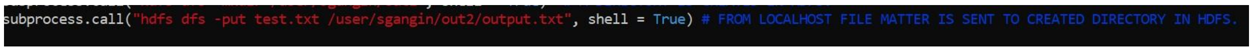
For Method 2 (PYTHON FILE EXECUTION)

1. create a python file "task1.py" in WinSCP for the reference see Group_6_Report_Part1.I attached the screen shorts in that file.

2. For code refer Group_6_Programfile_Part1

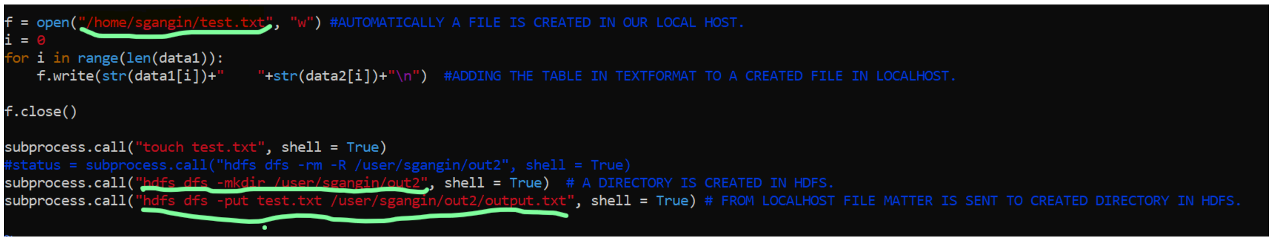
NOTE:- IF THERE IS ALREADY OUTPUT.TXT FILE NAME AS MENTIONED IN THE HDFS DIRECTORY IN SUBPROCESS.CALL PARAMETER AS BELOW, THEN YOU NEED TO DELETE THAT OUTPUT.TXT FILE BY FOLLOWING COMMAND ACCORDING TO YOUR HDFS PATH.

hdfs dfs -rm /user/sgangin/out2/output.txt



```
subprocess.call("hdfs dfs -put test.txt /user/sgangin/out2/output.txt", shell = True) # FROM LOCALHOST FILE MATTER IS SENT TO CREATED DIRECTORY IN HDFS.
```

NOTE :- BEFORE RUNNING SPARK-SUBMIT COMMAND , PLEASE CHANGE PATHS ACCORDING TO YOUR LOCAL AND HDFS PATHS IN THE CODE IN HIGHLIGHTED AREAS.



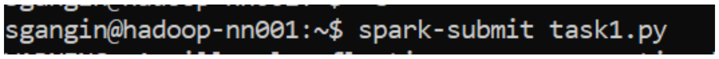
```
f = open("/home/sgangin/test.txt", "w") #AUTOMATICALLY A FILE IS CREATED IN OUR LOCAL HOST.
i = 0
for i in range(len(data1)):
    f.write(str(data1[i])+"    "+str(data2[i])+"\n")  #ADDING THE TABLE IN TEXTFORMAT TO A CREATED FILE IN LOCALHOST.

f.close()

subprocess.call("touch test.txt", shell = True)
#status = subprocess.call("hdfs dfs -rm -R /user/sgangin/out2", shell = True)
subprocess.call("hdfs dfs -mkdir /user/sgangin/out2", shell = True)  # A DIRECTORY IS CREATED IN HDFS.
subprocess.call("hdfs dfs -put test.txt /user/sgangin/out2/output.txt", shell = True) # FROM LOCALHOST FILE MATTER IS SENT TO CREATED DIRECTORY IN HDFS.
```

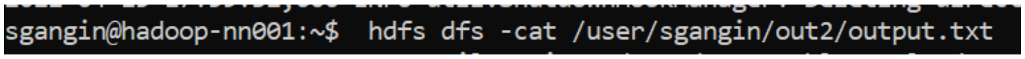
(ALSO REMEMBER THE FILE NAME TEST.TXT GIVEN IN WRITING PATH OF YOUR LOCAL IN LINE 1 MUST BE CONSTANT ONLY AT THEIR RESPECTIVE PLACES)

3. Run the below command in Hadoop cluster to view the result



```
sgangin@hadoop-nn001:~$ spark-submit task1.py
```

4. FOR VIEWING CONTENTS OF FILE NAMED OUTPUT.TXT USE THE FOLLOWING COMMAND.



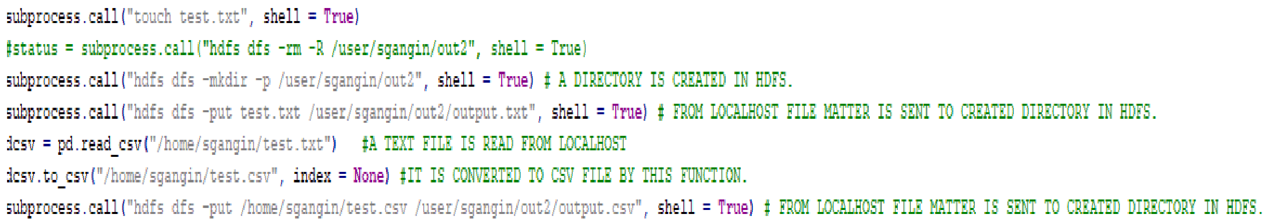
```
sgangin@hadoop-nn001:~$  hdfs dfs -cat /user/sgangin/out2/output.txt
```

(here, until cat same but after cat change path to your hdfs respectively).

IMPORTANT NOTES :- FOR TWO TASKS WE EVEN MANAGED TO HAVE OUTPUT OF A SINGLE FILE IN MULTIPLE PARTITIONS . IF YOU WANT TO SEE ANY OF THE MULTIPLE FILE OUTPUT AS OF THAT AS IN PICTURES IN REPORT . THEN YOU CAN TYPE FOLLOWING COMMAND:-

Hdfs dfs-cat <path to hdfs.

For bonus task :-



```
subprocess.call("touch test.txt", shell = True)
#status = subprocess.call("hdfs dfs -rm -R /user/sgangin/out2", shell = True)
subprocess.call("hdfs dfs -mkdir -p /user/sgangin/out2", shell = True) # A DIRECTORY IS CREATED IN HDFS.
subprocess.call("hdfs dfs -put test.txt /user/sgangin/out2/output.txt", shell = True) # FROM LOCALHOST FILE MATTER IS SENT TO CREATED DIRECTORY IN HDFS.
dcsv = pd.read_csv("/home/sgangin/test.txt")   #A TEXT FILE IS READ FROM LOCALHOST
dcsv.to_csv("/home/sgangin/test.csv", index = None) #IT IS CONVERTED TO CSV FILE BY THIS FUNCTION.
subprocess.call("hdfs dfs -put /home/sgangin/test.csv /user/sgangin/out2/output.csv", shell = True) # FROM LOCALHOST FILE MATTER IS SENT TO CREATED DIRECTORY IN HDFS.
```

(Everything as above is same but for getting out put, you need to cat the file output.csv in out2 folder.