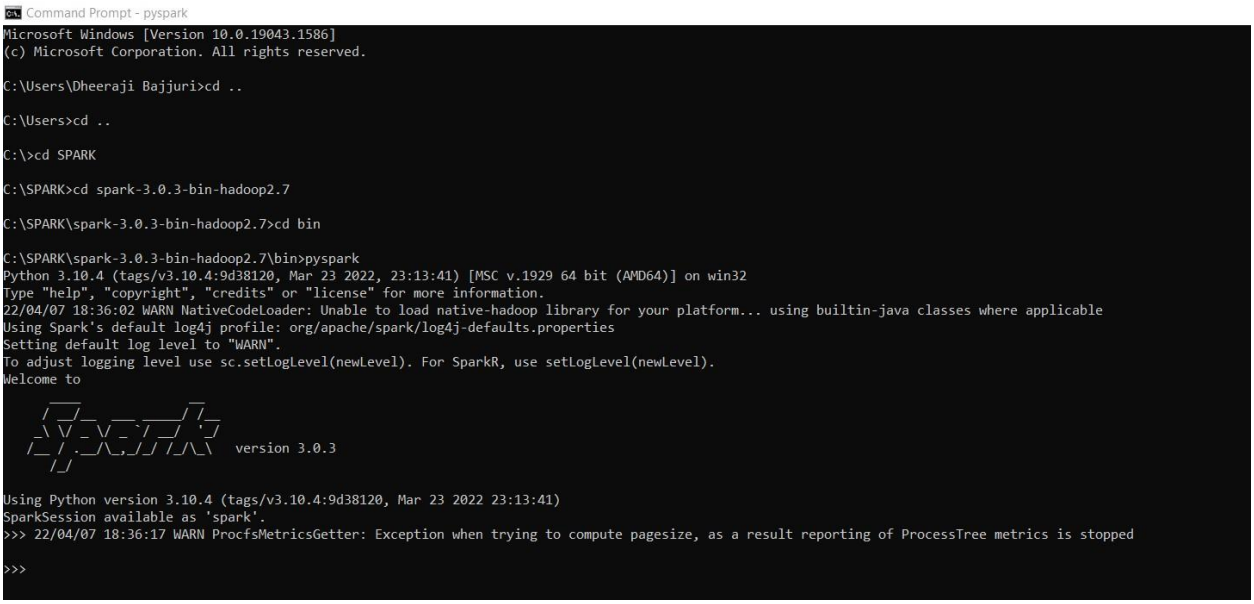
In Class Exercise :

1)

a) Creating dataset



- First we install spark into our system and then we run pyspark in command prompt using ‘pyspark’ command.

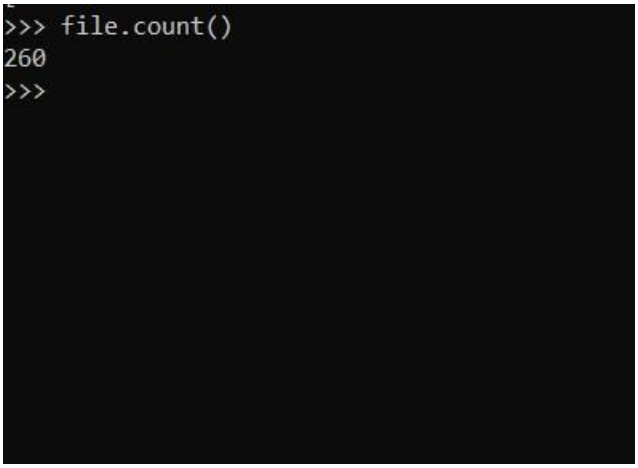
- Then we create an rdd named ‘file’ and load the dataset word_list-3.txt into it by specifying the file path.
- We create the rdd and load it using sparkContext.textFile command.

b)

```
>>> file = spark.sparkContext.textFile("C:/Users/Dheeraji Bajjuri/Downloads/word_list-3.txt")
>>> myfile_up = file.map(lambda line: line.upper())
>>> myfile_up.take(2)
['THE PROJECT GUTENBERG ETEXT OF MOBY WORD II BY GRADY WARD', 'COPYRIGHT LAWS ARE CHANGING ALL OVER THE WORLD, BE SURE TO CHECK']
>>>
```

- Here we create a variable named myfile_up and use map transformation with line.upper command which is used to change all words to uppercase.
- Then we use ‘take’ action and specify 2 as input to display the first two lines of the dataset.

c)



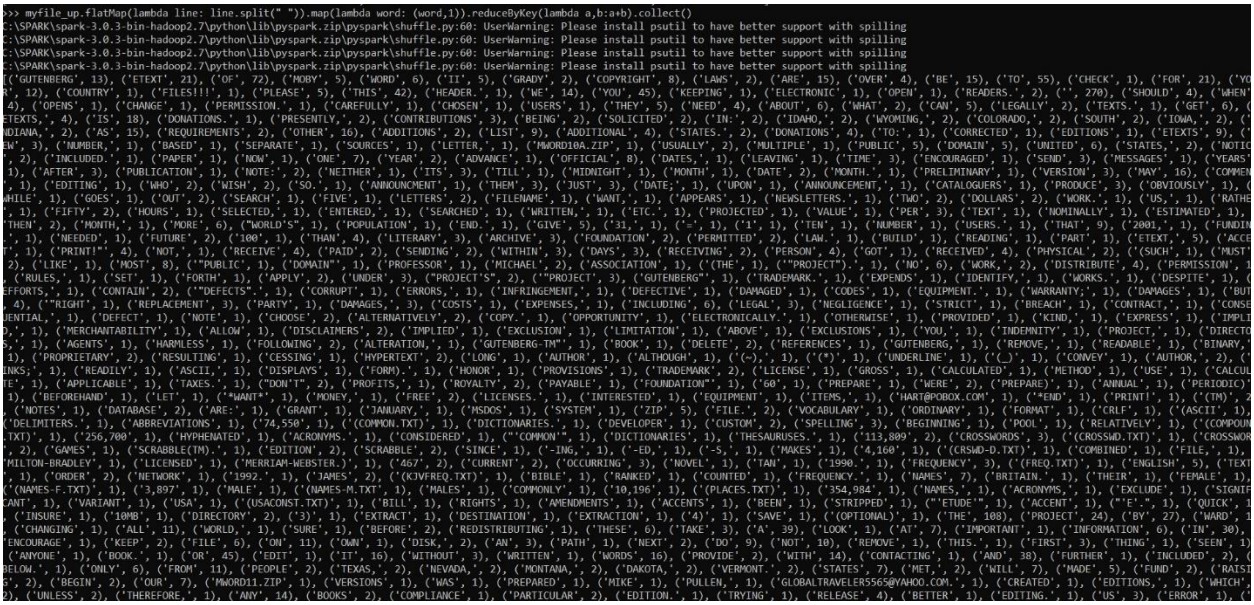
- Here we use ‘count’ action to display the total number of lines present in the dataset.
- In the output, we can see that the total count as 260.

d)

```
>>> myfile_up.flatMap(lambda line: line.split(" ")).filter(lambda word: word.count("PROJECT")).count()
32
>>>
```

- Here we use flatMap to take RDD of lines and convert it into words by splitting each line of input using space “ “ as separator.
- And we use filter transformation to filter the PROJECT word from the dataset and used count() to get the total number of word ‘PROJECT’.
- In the output, we can see that the total count of word PROJECT as 32.

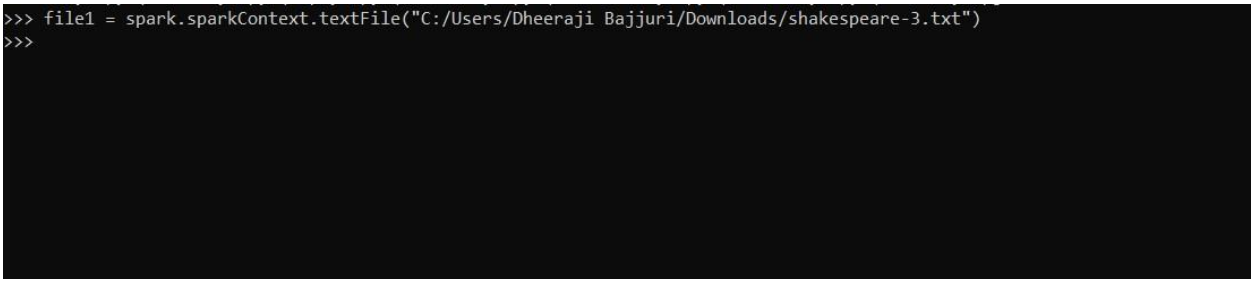
e)



- Here we use flatMap to take RDD of lines and convert it into words by splitting each line of input using space " " as separator.
- And we use map to convert RDD of words into RDD of tuples(word,1) and we use reduceByKey to reduce all the words to a+b based on a,b values and then we call collect() action to display the output of RDD.

2)

a)



- Then we create an rdd named 'file1' and load the dataset shakespeare-3.txt into it by specifying the file path.
- We create the rdd and load it using sparkContext.textFile command.

b)

```
>>> myfile_one = file1.map(lambda line: line.lower())
>>> myfile_one.take(5)
['the project gutenberg ebook of the complete works of william shakespeare, by ', 'william shakespeare', '', 'this ebook is for the use of anyone anywhere at no cost and with', 'almost no restrictions whatsoever
.  you may copy it, give it away or']
>>>
```

- Here we create a variable named myfile_one and use map transformation with line.lower command which is used to change all words to lowercase.
- Then we use ‘take’ action and specify 5 as input to display the first five lines of the dataset.

c)

```
>>> file1.count()
124796
```

- Here we use ‘count’ action to display the total number of words present in the dataset.
- In the output, we can see that the total count as 124796.

d)

```
>>> myfile_one.flatMap(lambda line: line.split(" ")).filter(lambda word: word.count("is")).count()
36799
>>>
```

- Here we use flatMap to take RDD of lines and convert it into words by splitting each line of input using space “ “ as separator.
- And we use filter transformation to filter “is” word from the dataset and used count() to get the total number of word ‘is’.
- In the output, we can see that the total count of word ‘is’ as 36799.

e)

```
>>> file1.distinct().count()
C:\SPARK\spark-3.0.3-bin-hadoop2.7\python\lib\pyspark.zip\pyspark\shuffle.py:60:
C:\SPARK\spark-3.0.3-bin-hadoop2.7\python\lib\pyspark.zip\pyspark\shuffle.py:60:
C:\SPARK\spark-3.0.3-bin-hadoop2.7\python\lib\pyspark.zip\pyspark\shuffle.py:60:
C:\SPARK\spark-3.0.3-bin-hadoop2.7\python\lib\pyspark.zip\pyspark\shuffle.py:60:
111594
>>>
```

- Here we use distinct() which removes duplicates and only gives unique words present in the dataset.
- We use count() action along with distinct() to get the total number of unique words present in the dataset.
- In the output, we can see that the total count of unique words as 111594.

3)

a)

```
>>> df = spark.read.format("com.dtabricks.spark.csv").option("mode","DROPMALFORMED").option("header",True).option("inferschema",True).csv("C:/Users/Dheeraji Bajjuri/Downloads/hotel_bookings-1.csv")
>>>
>>>
```

- Here we load the data into dataframe using read command and specify dropmalformed to drop error rows that do not match the schema and specify header as true to tell that the file has a header row and specify inferschema as true which infers column types based on data and then using .csv(), we specify the path to the csv file in which our data is present.

b)

```
>>> df.describe("adults").show()
+-------+------------------+
|summary|            adults|
+-------+------------------+
|  count|            119390|
|   mean|1.8564033838679956|
| stddev|0.5792609988327535|
|    min|                 0|
|    max|                55|
+-------+------------------+

>>>
```

- Here we use describe command to show statistical values of adults column like count, mean, standard deviation, min and max.

c)

```
>>> df.groupBy("hotel").sum("is_canceled").show()
22/04/07 21:07:03 WARN package: Truncated the string representation of a plan since it was too la
+------------+----------------+
|       hotel|sum(is_canceled)|
+------------+----------------+
|  City Hotel|           33102|
|Resort Hotel|           11122|
+------------+----------------+

>>>
```

- Here we use groupBy to group the results based on hotel name and sum() is used to get the total number of cancelled values and show() is used to display the output.

d)

```
>>> df.createGlobalTempView("hotel")
```

- Then we register the hotel dataframe as a global temporary view using createGlobalTempView().

e)

```
>>> spark.sql("select count(reservation_status) from global_temp.hotel where reservation_status==\"Canceled\"").show()
+------------------------+
|count(reservation_status)|
+------------------------+
|                   43017|
+------------------------+

>>>
```

- Here we use spark.sql to write sql query and we use select command and count() to display the total number of records of reservation status from global temp view and used where clause to specify the condition of reservation_status as cancelled and show() to display the output.

f)

```
>>> spark.sql("select sum(agent) from global_temp.hotel group by hotel").show()
+--------------------------+
|sum(CAST(agent AS DOUBLE))|
+--------------------------+
|                 2003876.0|
|                 6929877.0|
+--------------------------+

>>>
```

- Here we use spark.sql to write sql query and we use select command and sum() to count the number of agents per hotel from global temp view and group by is used to group by hotel and show() to display the output.

g)

```
>>> spark.sql("select arrival_date_year, count(babies) from global_temp.hotel where babies>0 group by arrival_date_year").show()
+-----------------+-------------+
|arrival_date_year|count(babies)|
+-----------------+-------------+
|             2015|          213|
|             2016|          446|
|             2017|          258|
+-----------------+-------------+

>>>
```

- Here we use spark.sql to write sql query and we use select command and specify arrival date_year and count(babies) to display year and count of babies from hotel view and use where clause to specify the condition that babies greater than 0 and group by is used to group data by arrival_date_year and show() to display the output.

h)

```
>>> spark.sql("select country, sum(is_canceled) as all_canceled from global_temp.hotel group by country order by all_canceled desc").show()
+-------+------------+
|country|all_canceled|
+-------+------------+
|    PRT|       27519|
|    GBR|        2453|
|    ESP|        2177|
|    FRA|        1934|
|    ITA|        1333|
|    DEU|        1218|
|    IRL|         832|
|    BRA|         830|
|    USA|         501|
|    BEL|         474|
|    CHN|         462|
|    CHE|         428|
|    NLD|         387|
|     CN|         254|
|    RUS|         239|
|    AUT|         230|
|    SWE|         227|
|    POL|         215|
|    AGO|         205|
|    NOR|         181|
+-------+------------+
only showing top 20 rows

>>>
```

- Here we use spark.sql to write sql query and we use select command and specify country and sum(is_cancelled) to display country and sum of cancelled from hotel view and group by is used to group data by country and order by, desc to order all data by cancelled number in decreasing order and show() to display the output.
- Here we make an alias name for sum(is_cancelled) as all_cancelled.