ASSIGNMENT SOLUTION

The following Common Boilerplate code to create a Spark Session has to be executed before running the queries.

```
from pyspark.sql import SparkSession
import getpass
username = getpass.getuser()
spark= SparkSession. \
builder. \
config('spark.ui.port','0'). \
config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \
enableHiveSupport(). \
master('yarn'). \
getOrCreate()
```

Question 1

```
#creating schema for reading the json file
from pyspark.sql.types import StructType, StructField, StringType,
IntegerType, ArrayType
users_schema = StructType([
    StructField("user_id", IntegerType(), nullable=False),
    StructField("user_first_name", StringType(), nullable=False),
    StructField("user_last_name", StringType(), nullable=False),
    StructField("user_email", StringType(), nullable=False),
    StructField("user_gender", StringType(), nullable=False),
    StructField("user_phone_numbers", ArrayType(StringType()),
nullable=True),
    StructField("user_address", StructType([
        StructField("street", StringType(), nullable=False),
        StructField("city", StringType(), nullable=False),
        StructField("state", StringType(), nullable=False),
        StructField("postal_code", StringType(), nullable=False),
    ]), nullable=False)
])
```

#reading the files

users_df = spark.read \
.format("json") \
.schema(users_schema) \
.load("/public/sms/users/")

#number of partitions
users_df.rdd.getNumPartitions()

```
#creating schema for reading the json file

from pyspark.sql.types import StructType, StructField, StringType, IntegerType, ArrayType
users_schema = StructType([
    StructField("user_id", IntegerType(), nullable=False),
    StructField("user_first_name", StringType(), nullable=False),
    StructField("user_last_name", StringType(), nullable=False),
    StructField("user_email", StringType(), nullable=False),
    StructField("user_gender", StringType(), nullable=False),
    StructField("user_phone_numbers", ArrayType(StringType()), nullable=True),
    StructField("user_address", StructType([
        StructField("street", StringType(), nullable=False),
        StructField("city", StringType(), nullable=False),
        StructField("state", StringType(), nullable=False),
        StructField("postal_code", StringType(), nullable=False),
    ]), nullable=False)
])
```

```
#reading the files

users_df = spark.read \
.format("json") \
.schema(users_schema) \
.load("/public/sms/users/")
```

```
#Ans 1 - number of partitions
users_df.rdd.getNumPartitions()
```

```
3
```

Question 2

2a.

#Total count of records in the Datarame

users_df.count()

```
#Ans 2a - Count of records
users_df.count()
```

```
1000000
```

2b.

#extracting columns from nested json and creating views

```
from pyspark.sql.functions import col,size
users_df.withColumn("user_street",col("user_Address.street")) \
.withColumn("user_city",col("user_address.city")) \
.withColumn("user_state", col("user_Address.state")) \
.withColumn("user_postal_code", col("user_address.postal_code")) \
.withColumn("num_phn_numbers",
size(col("user_phone_numbers"))).createOrReplaceTempView("users_vw")
```

#finding distinct user count for New York State

```
spark.sql("""select count(distinct user_id) as user_cnt from users_vw
          where user_state='New York'""")
```

```
#extracting columns from nested json and creating views

from pyspark.sql.functions import col,size
users_df.withColumn("user_street",col("user_Address.street")) \
.withColumn("user_city",col("user_address.city")) \
.withColumn("user_state", col("user_Address.state")) \
.withColumn("user_postal_code", col("user_address.postal_code")) \
.withColumn("num_phn_numbers", size(col("user_phone_numbers"))).createOrReplaceTempView("users_vw")
```

```
#Ans 2b - finding distinct user count for New York State
spark.sql("""select count(distinct user_id) as user_cnt from users_vw
           where user_state='New York'""")
```

user_cnt
49576

2c.

#State with max postal codes

```
spark.sql("""select user_state,count(distinct user_postal_Code) as postal_cnt
from users_vw
group by user_state order by postal_cnt desc limit 1""")
```

```
#ans 2c - State with max postal codes
spark.sql("""select user_state,count(distinct user_postal_Code) as postal_cnt from users_vw
group by user_state order by postal_cnt desc limit 1""")
```

user_state	postal_cnt
California	206

2d.

#City with maximum users

spark.sql("""select user_city,count(distinct user_id) as user_cnt from users_vw
where user_city is not null
group by user_city order by user_cnt desc limit 1""")

```
#ans 2d - City with maximum users
spark.sql("""select user_city,count(distinct user_id) as user_cnt from users_vw
where user_city is not null
group by user_city order by user_cnt desc limit 1""")
```

user_city	user_cnt
Washington	28504

2e.

#Count of users having email domain as bizjournals.com

spark.sql("""select count(distinct user_id) as user_cnt from users_vw
where user_email like '%bizjournals.com'""")

```
#ans 2e - users having email domain as bizjournals.com
spark.sql("""select count(distinct user_id) as user_cnt from users_vw
where user_email like '%bizjournals.com'""")
```

user_cnt
2015

2f.

#Users with 4 phone numbers

spark.sql("select count(distinct user_id) as user_cnt from users_vw where num_phn_numbers=4")

```
#ans 2f - users with 4 phone numbers
spark.sql("select count(distinct user_id) as user_cnt from users_vw where num_phn_numbers=4")
```

user_cnt
179041

2g.

#Users with no phone numbers mentioned

#ans2g - users with no phone number
spark.sql("""select count(distinct user_id) as user_cnt from users_vw
where user_phone_numbers is null""")

```
#ans2g - users with no phone number
spark.sql("""select count(distinct user_id) as user_cnt from users_vw
where user_phone_numbers is null""")
```

user_cnt
108981

Question 3

#Saving back the base dataframe

users_df.write.format("parquet").mode("overwrite").option("path","/user/<your-username>/week9/assignment").save()

```
#Ans 3 saving back the base dataframe
users_df.write.format("parquet").mode("overwrite").option("path","/user/itv006753/week9/assignment").save()
```

Check for the no.of files and the file size by opening a terminal and executing the following command

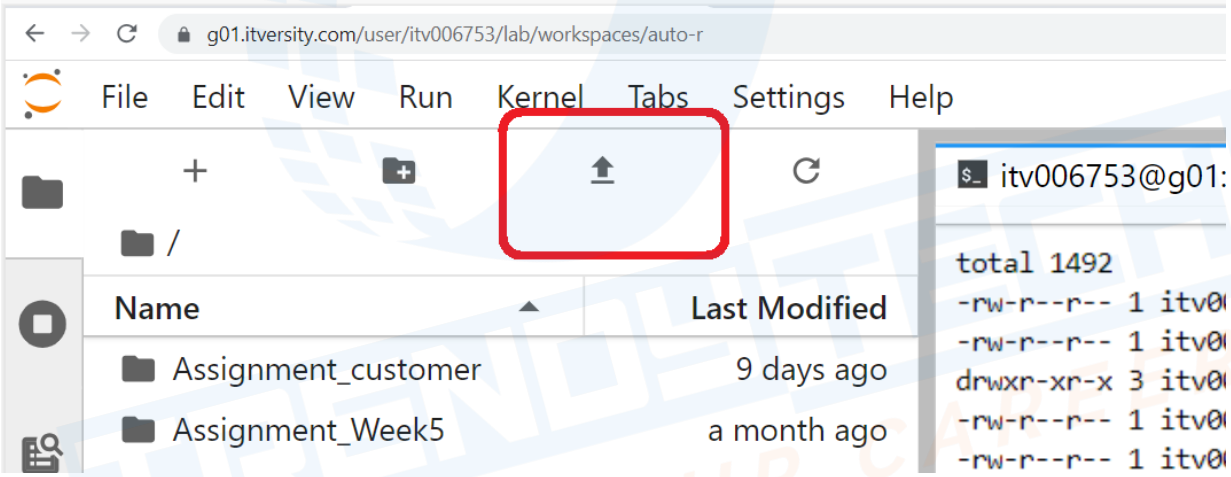
hadoop fs -ls -h /user/<your-username>/week9/assignment

```
[itv006753@g01 ~]$ hadoop fs -ls -h  /user/itv006753/week9/assignment
Found 4 items
-rw-r--r--   3 itv006753 supergroup          0 2023-06-30 03:57 /user/itv006753/week9/assignment/_SUCCESS
-rw-r--r--   3 itv006753 supergroup     25.9 M 2023-06-30 03:57 /user/itv006753/week9/assignment/part-00000-3b750244-3aa1-47de-bebe-e9506fa6168f-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup     25.9 M 2023-06-30 03:56 /user/itv006753/week9/assignment/part-00001-3b750244-3aa1-47de-bebe-e9506fa6168f-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup     13.1 M 2023-06-30 03:57 /user/itv006753/week9/assignment/part-00002-3b750244-3aa1-47de-bebe-e9506fa6168f-c000.snappy.parquet
```

Question 4

In a notepad, copy the complete solution of Question 1 and 2 along with the spark session creation code and import statements and save it as “question4.py”. Files will be available in the downloadable section for your reference.

Upload this file to the gateway node of the Lab.



Open a terminal and execute the code file using spark-submit utility as given below :

```
spark-submit \
--master yarn \
--num-executors 2 \
--executor-cores 2 \
--executor-memory 4G \
--conf spark.dynamicAllocation.enabled=false \
question4.py
```

```
[itv006753@g01 ~]$ spark-submit \
> --master yarn \
> --num-executors 2 \
> --executor-cores 2 \
> --executor-memory 4G \
> --conf spark.dynamicAllocation.enabled=false \
> question4.py
Multiple versions of Spark are installed but SPARK_MAJOR_VERSION is not set
Spark2 will be picked by default
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/spark-2.4.7-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/Stat
SLF4J: Found binding in [jar:file:/opt/hadoop-3.3.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/imp
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
5
1000000
+--------+
|user_cnt|



+----------+----------+
|user_state|postal_cnt|
+----------+----------+
|California|       206|
+----------+----------+

+----------+--------+
| user_city|user_cnt|
+----------+--------+
|Washington|   28504|
+----------+--------+

+--------+
|user_cnt|
+--------+
|    2015|
+--------+


+--------+
|user_cnt|
+--------+
|  179041|
+--------+

+--------+
|user_cnt|
+--------+
|  108981|
+--------+
```

Question 5

```
#State-wise gender count


spark.sql("""select user_state,
        sum(Male_cnt) as Male,
        sum(Female_cnt) as Female
        from
        (select user_state,
        case when user_gender='Male' then count(user_id) end as Male_cnt,
        case when user_gender='Female' then count(user_id) end as
Female_cnt
        from users_vw
        where user_state is not null and user_phone_numbers is not null
        group by user_state,user_gender)
        group by user_state
        order by user_state""")
```

```
#Ans 5 state-wise gender count
spark.sql("""select user_state,
            sum(Male_cnt) as Male,
            sum(Female_cnt) as Female
            from
            (select user_state,
            case when user_gender='Male' then count(user_id) end as Male_cnt,
            case when user_gender='Female' then count(user_id) end as Female_cnt
            from users_vw
            where user_state is not null and user_phone_numbers is not null
            group by user_state,user_gender)
            group by user_state
            order by user_state""")
```

user_state	Male	Female
Alabama	9307	9178
Alaska	1882	1938
Arizona	9406	9543
Arkansas	2420	2416
California	49120	48716
Colorado	10128	10125
Connecticut	5797	5917

Question 6

In a notepad, copy the complete solution of Question 5 along with the spark session creation code and import statements and save it as “question6.py”. Files will be available in the downloadable section for your reference.

Upload this file to the gateway node of the Lab.

Create a folder named “pivot_assignment_result” in HDFS home directory using the following command :

hadoop fs -mkdir /user/<your-username>/pivot_assignment_result

Open a terminal and execute the following :

```
spark-submit \
--deploy-mode cluster \
--master yarn \
--num-executors 4 \
--executor-cores 1 \
--executor-memory 2G \
--driver-memory 2G \
--driver-cores 1 \
--conf spark.dynamicAllocation.enabled=false \
--verbose \
question6.py
```

```
[itv006753@g01 ~]$ spark-submit \
> --deploy-mode cluster \
> --master yarn \
> --num-executors 4 \
> --executor-cores 1 \
> --executor-memory 2G \
> --driver-memory 2G \
> --driver-cores 1 \
> --conf spark.dynamicAllocation.enabled=false \
> --verbose \
> question6.py
Multiple versions of Spark are installed but SPARK_MAJOR_VERSION is not set
Spark2 will be picked by default
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/spark-2.4.7-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinc
SLF4J: Found binding in [jar:file:/opt/hadoop-3.3.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLogg
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Using properties file: /opt/spark2-client/conf/spark-defaults.conf
Adding default property: spark.yarn.jars=hdfs:///spark2-jars/*.jar
Adding default property: spark.history.fs.logDirectory=hdfs:///spark2-logs
Adding default property: spark.eventLog.enabled=true
Adding default property: spark.shuffle.service.enabled=true
Adding default property: spark.history.fs.update.interval=10s
Adding default property: spark.yarn.historyServer.address=m01.itversity.com:18081
Adding default property: spark.history.fs.cleaner.enabled=true
Adding default property: spark.dynamicAllocation.maxExecutors=10
Adding default property: spark.driver.extraJavaOptions=-Dderby.system.home=/tmp/derby/
Adding default property: spark.master=yarn
Adding default property: spark.sql.repl.eagerEval.enabled=true
Adding default property: spark.history.provider=org.apache.spark.deploy.history.FsHistoryProvider
Adding default property: spark.executor.extraLibraryPath=/opt/hadoop/lib/native
Adding default property: spark.eventLog.dir=hdfs:///spark2-logs
```

Check the results by navigating to the results directory in HDFS home :
hadoop fs -ls user/<your-username>/pivot_assignment_result

```
[itv006753@g01 ~]$ hadoop fs -ls /user/itv006753/pivot_assignment_result
Found 52 items
-rw-r--r--   3 itv006753 supergroup          0 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/_SUCCESS
-rw-r--r--   3 itv006753 supergroup        967 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00000-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        958 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00001-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        967 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00002-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        976 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00003-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        994 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00004-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        976 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00005-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup       1003 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00006-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        976 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00007-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup       1084 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00008-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        967 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00009-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        967 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00010-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        958 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00011-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        949 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00012-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        976 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00013-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        967 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00014-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
-rw-r--r--   3 itv006753 supergroup        938 2023-06-30 05:37 /user/itv006753/pivot_assignment_result/part-00015-d23c80c0-5213-4b09-b104-2aa351151df5-c000.snappy.parquet
```

Question 7

```
#setting up spark session - default configuration

from pyspark.sql import SparkSession
import getpass

username = getpass.getuser()

spark = SparkSession. \
    builder. \
        config('spark.shuffle.useOldFetchProtocol', 'true'). \
    config("spark.sql.warehouse.dir", "/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()

airlines_df = spark.read \
.format("csv") \
.load("/public/airlines_all/airlines/")
```

- **Try seeing how many initial partitions are there in your dataframe.**

```
airlines_df.rdd.getNumPartitions()
```

```
#Answer 7
#setting up spark session - default configuration

from pyspark.sql import SparkSession
import getpass

username = getpass.getuser()

spark = SparkSession. \
    builder. \
    config('spark.shuffle.useOldFetchProtocol', 'true'). \
    config("spark.sql.warehouse.dir", "/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()
```

```
airlines_df = spark.read \
.format("csv") \
.load("/public/airlines_all/airlines/*")
```

```
airlines_df.rdd.getNumPartitions()
```

```
1919
```

- Why do you see these many partitions, what is the logic?

```
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 11:29 /public/airlines_all/airlines/part-01850
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 07:53 /public/airlines_all/airlines/part-01851
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:09 /public/airlines_all/airlines/part-01852
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:15 /public/airlines_all/airlines/part-01853
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:45 /public/airlines_all/airlines/part-01854
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:26 /public/airlines_all/airlines/part-01855
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:24 /public/airlines_all/airlines/part-01856
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:53 /public/airlines_all/airlines/part-01857
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:50 /public/airlines_all/airlines/part-01858
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:55 /public/airlines_all/airlines/part-01859
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:53 /public/airlines_all/airlines/part-01860
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:49 /public/airlines_all/airlines/part-01861
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:32 /public/airlines_all/airlines/part-01862
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:35 /public/airlines_all/airlines/part-01863
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:26 /public/airlines_all/airlines/part-01864
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 11:21 /public/airlines_all/airlines/part-01865
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:36 /public/airlines_all/airlines/part-01866
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:26 /public/airlines_all/airlines/part-01867
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:59 /public/airlines_all/airlines/part-01868
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:26 /public/airlines_all/airlines/part-01869
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:41 /public/airlines_all/airlines/part-01870
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 11:31 /public/airlines_all/airlines/part-01871
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:57 /public/airlines_all/airlines/part-01872
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:10 /public/airlines_all/airlines/part-01873
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:17 /public/airlines_all/airlines/part-01874
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:31 /public/airlines_all/airlines/part-01875
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:43 /public/airlines_all/airlines/part-01876
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:51 /public/airlines_all/airlines/part-01877
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:36 /public/airlines_all/airlines/part-01878
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:27 /public/airlines_all/airlines/part-01879
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 11:00 /public/airlines_all/airlines/part-01880
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:34 /public/airlines_all/airlines/part-01881
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:53 /public/airlines_all/airlines/part-01882
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:08 /public/airlines_all/airlines/part-01883
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 08:03 /public/airlines_all/airlines/part-01884
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 09:40 /public/airlines_all/airlines/part-01885
-rw-r--r--   2 hdfs supergroup     64.0 M 2021-01-28 10:45 /public/airlines_all/airlines/part-01886
```

We can’t merge another file with it, as it will exceed 128Mb

/public/airlines_all/airlines/ -> This path has 1920 files

but when we create a dataframe on top of this we have 1919 files:

airlines_df=spark.read\
.format("csv")\
.load("/public/airlines_all/airlines/")

airlines_df.rdd.getNumPartitions()

1919

Here, the last file is of size : 48MB which gets added to previous file of 64MB

so (64 + 4) + (48 + 4) = 120

But the same way the two files of 64MB cannot be added:

(64 + 4) + (64 + 4) = 136

As, the file will be 136MB which exceeds 128MB

- **Now change the maxPartitionBytes to 140 mb**

To do that covert 140mb into Bytes and add this statement to that sparksession code

```
config("spark.sql.files.maxPartitionBytes", "146800640"). \
```

- **Try creating the same dataframe again by loading all the files. (Again, No need to infer the schema, nor you have to define it.)**
- **Try seeing how many initial partitions there in this new dataframe are.**

```
#setting up spark session - default configuration

from pyspark.sql import SparkSession
import getpass

username = getpass.getuser()

spark = SparkSession. \
    builder. \
        config('spark.shuffle.useOldFetchProtocol', 'true'). \
    config("spark.sql.files.maxPartitionBytes", "146800640"). \
    config("spark.sql.warehouse.dir", "/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()

airlines_df_changed = spark.read \
.format("csv") \
.load("/public/airlines_all/airlines/")

airlines_df_changed.rdd.getNumPartitions()
```

```
#Answer 7
#setting up spark session - default configuration

from pyspark.sql import SparkSession
import getpass

username = getpass.getuser()

spark = SparkSession. \
    builder. \
   *config('spark.shuffle.useOldFetchProtocol', 'true'). \
    config("spark.sql.files.maxPartitionBytes", "146800640"). \
    config("spark.sql.warehouse.dir", "/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()
```

```
airlines_df = spark.read \
.format("csv") \
.load("/public/airlines_all/airlines/*")
```

```
airlines_df.rdd.getNumPartitions()
```

```
960
```

You will see a different number of partitions now, why?
In this case, with the partition size set to 140MB and each file adding up to 68MB, it is possible to merge another file within the partition size limit. Consequently, we would have 960 partitions in this scenario.

Question 8

hadoop fs -rm -R /user/<your-username>/pivot_assignment_result