CSCE 5300 Introduction to Big data and Data Science

ICE-8

Lesson Title: Spark

Lesson Description: Spark with RDDs (transformation and actions) and Spark with data frames and SQL

Lesson Overview:

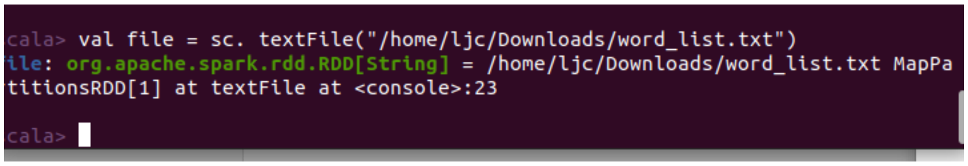
Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

In Class Exercise:

1. Create spark RDD from external dataset(word_list.txt)). Execute transformation and actions by scala.

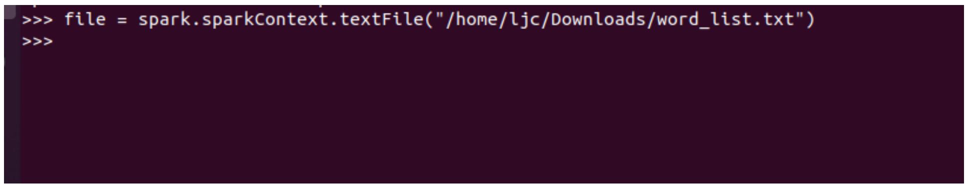
Create RDD from external dataset(word_list.txt).

Scala:



```
cala> val file = sc. textFile("/home/ljc/Downloads/word_list.txt")
ile: org.apache.spark.rdd.RDD[String] = /home/ljc/Downloads/word_list.txt MapPa
titionsRDD[1] at textFile at <console>:23

cala>
```

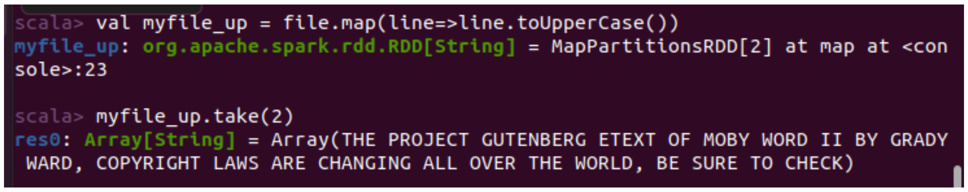
Python:



```
>>> file = spark.sparkContext.textFile("/home/ljc/Downloads/word_list.txt")
>>>
```

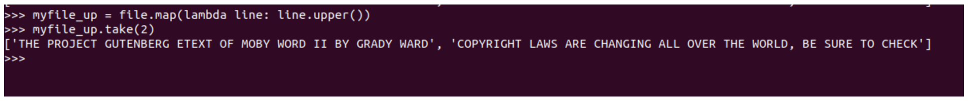
Change all words to uppercase and show the first two lines.

Scala:



```
scala> val myfile_up = file.map(line=>line.toUpperCase())
myfile_up: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at map at <con
sole>:23

scala> myfile_up.take(2)
res0: Array[String] = Array(THE PROJECT GUTENBERG ETEXT OF MOBY WORD II BY GRADY
 WARD, COPYRIGHT LAWS ARE CHANGING ALL OVER THE WORLD, BE SURE TO CHECK)
```

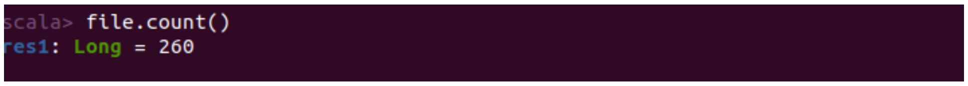
Python:



```
>>> 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']
>>>
```

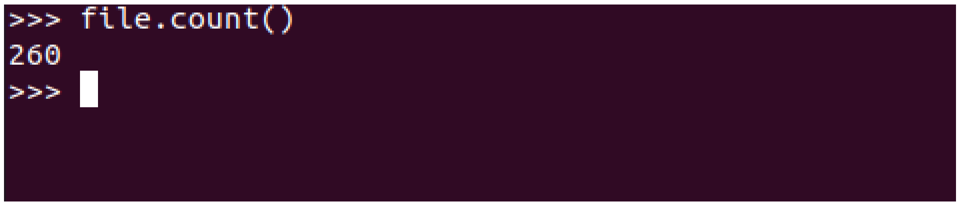
Count the number of lines.

Scala:



```
scala> file.count()
res1: Long = 260
```

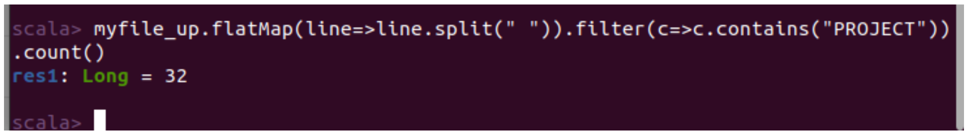
Python:



```
>>> file.count()
260
>>>
```

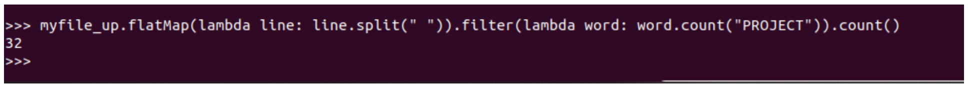
Count the number of word “PROJECT”.

Scala:



```
scala> myfile_up.flatMap(line=>line.split(" ")).filter(c=>c.contains("PROJECT"))
.count()
res1: Long = 32

scala>
```

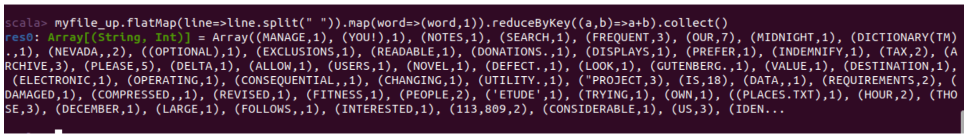
Python:



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

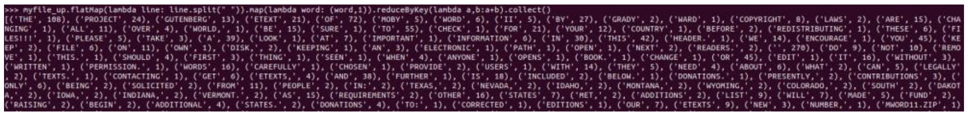
Count the words in the dataset.

Scala:



```
scala> myfile_up.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey((a,b)=>a+b).collect()
res0: Array[(String, Int)] = Array((MANAGE,1), (YOU!),1), (NOTES,1), (SEARCH,1), (FREQUENT,3), (OUR,7), (MIDNIGHT,1), (DICTIONARY(TM)
.,1), (NEVADA,,2), ((OPTIONAL),1), (EXCLUSIONS,1), (READABLE,1), (DONATIONS.,1), (DISPLAYS,1), (PREFER,1), (INDEMNIFY,1), (TAX,2), (A
RCHIVE,3), (PLEASE,5), (DELTA,1), (ALLOW,1), (USERS,1), (NOVEL,1), (DEFECT.,1), (LOOK,1), (GUTENBERG.,1), (VALUE,1), (DESTINATION,1),
 (ELECTRONIC,1), (OPERATING,1), (CONSEQUENTIAL,,1), (CHANGING,1), (UTILITY.,1), ("PROJECT,3), (IS,18), (DATA,,1), (REQUIREMENTS,2), (
DAMAGED,1), (COMPRESSED,,1), (REVISED,1), (FITNESS,1), (PEOPLE,2), ('ETUDE',1), (TRYING,1), (OWN,1), ((PLACES.TXT),1), (HOUR,2), (THO
SE,3), (DECEMBER,1), (LARGE,1), (FOLLOWS,,1), (INTERESTED,1), (113,809,2), (CONSIDERABLE,1), (US,3), (IDEN...
```

Python:



```
>>> myfile_up.flatMap(lambda line: line.split(" ")).map(lambda word: (word,1)).reduceByKey(lambda a,b:a+b).collect()
[('THE', 108), ('PROJECT', 24), ('GUTENBERG', 13), ('ETEXT', 21), ('OF', 72), ('MOBY', 5), ('WORD', 6), ('II', 5), ('BY', 27), ('GRADY', 2), ('WARD', 1), ('COPYRIGHT', 8), ('LAWS', 2), ('ARE', 15), ('CHA
NGING', 1), ('ALL', 11), ('OVER', 4), ('WORLD,', 1), ('BE', 15), ('SURE', 1), ('TO', 55), ('CHECK', 1), ('FOR', 21), ('YOUR', 12), ('COUNTRY', 1), ('BEFORE', 2), ('REDISTRIBUTING', 1), ('THESE', 6), ('FI
LES!!!', 1), ('PLEASE', 5), ('TAKE', 3), ('A', 39), ('LOOK', 1), ('AT', 7), ('IMPORTANT', 1), ('INFORMATION', 6), ('IN', 30), ('THIS', 42), ('HEADER.', 1), ('WE', 14), ('ENCOURAGE', 1), ('YOU', 45), ('KE
EP', 2), ('FILE', 6), ('ON', 11), ('OWN', 1), ('DISK,', 2), ('KEEPING', 1), ('AN', 3), ('ELECTRONIC', 1), ('PATH', 1), ('OPEN', 1), ('NEXT', 2), ('READERS.', 2), ('', 270), ('DO', 9), ('NOT', 10), ('REMO
VE', 1), ('THIS.', 1), ('SHOULD', 4), ('FIRST', 3), ('THING', 1), ('SEEN', 1), ('WHEN', 4), ('ANYONE', 1), ('OPENS', 1), ('BOOK.', 1), ('CHANGE', 1), ('OR', 45), ('EDIT', 1), ('IT', 16), ('WITHOUT', 3),
('WRITTEN', 1), ('PERMISSION.', 1), ('WORDS', 16), ('CAREFULLY', 1), ('CHOSEN', 1), ('PROVIDE', 2), ('USERS', 1), ('WITH', 14), ('THEY', 5), ('NEED', 4), ('ABOUT', 6), ('WHAT', 2), ('CAN', 5), ('LEGALLY'
, 2), ('TEXTS.', 1), ('CONTACTING', 1), ('GET', 6), ('ETEXTS,', 4), ('AND', 38), ('FURTHER', 1), ('IS', 18), ('INCLUDED', 2), ('BELOW.', 1), ('DONATIONS.', 1), ('PRESENTLY,', 2), ('CONTRIBUTIONS', 3), ('
ONLY', 6), ('BEING', 2), ('SOLICITED', 2), ('FROM', 11), ('PEOPLE', 2), ('IN:', 2), ('TEXAS,', 2), ('NEVADA,', 2), ('IDAHO,', 2), ('MONTANA,', 2), ('WYOMING,', 2), ('COLORADO,', 2), ('SOUTH', 2), ('DAKOT
A,', 2), ('IOWA,', 2), ('INDIANA,', 2), ('VERMONT.', 2), ('AS', 15), ('REQUIREMENTS', 2), ('OTHER', 16), ('STATES', 7), ('MET,', 2), ('ADDITIONS', 2), ('LIST', 9), ('WILL', 7), ('MADE', 5), ('FUND', 2),
('RAISING', 2), ('BEGIN', 2), ('ADDITIONAL', 4), ('STATES.', 2), ('DONATIONS', 4), ('TO:', 1), ('CORRECTED', 1), ('EDITIONS', 1), ('OUR', 7), ('ETEXTS', 9), ('NEW', 3), ('NUMBER,', 1), ('MWORD11.ZIP', 1)
```

2. Create spark RDD from external dataset(shakespeare.txt). Execute transformation and actions by scala.

Change all words to lowercase and show the first 5 lines.

Count the total number of words.

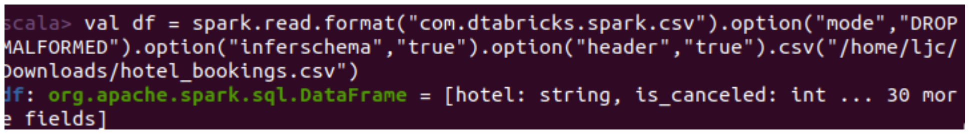
Count the number of word “is”.

Count the number of unique words in the dataset.

3. Create Spark dataframe from hotel_booking data and execute some query.

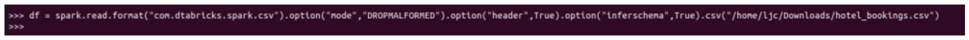
Load data from the hotel_booking.csv.

Scala:



```
cala> val df = spark.read.format("com.dtabricks.spark.csv").option("mode","DROP
MALFORMED").option("inferschema","true").option("header","true").csv("/home/ljc/
Downloads/hotel_bookings.csv")
lf: org.apache.spark.sql.DataFrame = [hotel: string, is_canceled: int ... 30 mor
e fields]
```

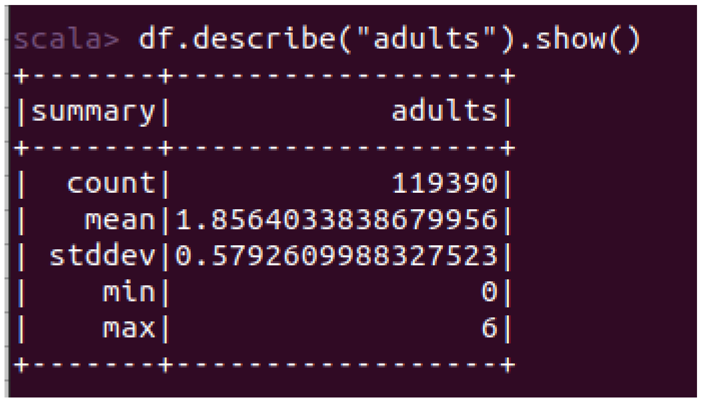
Python:



```
>>> df = spark.read.format("com.dtabricks.spark.csv").option("mode","DROPMALFORMED").option("header",True).option("inferschema",True).csv("/home/ljc/Downloads/hotel_bookings.csv")
>>>
```

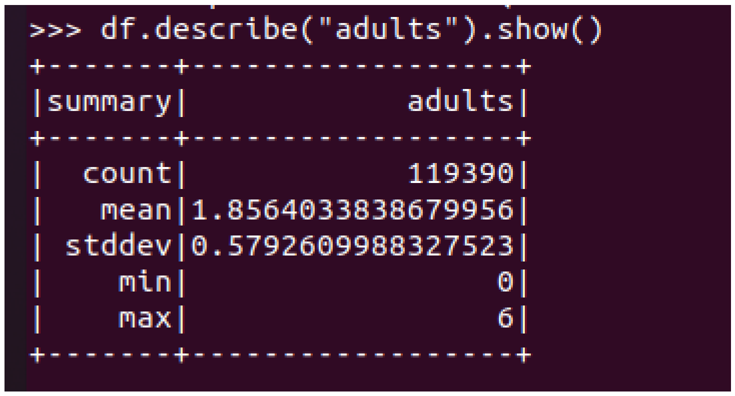
Show some statistical values(mean, max value) of adults column.

Scala:



```
scala> df.describe("adults").show()
+-------+------------------+
|summary|            adults|
+-------+------------------+
|  count|            119390|
|   mean|1.8564033838679956|
| stddev|0.5792609988327523|
|    min|                 0|
|    max|                 6|
+-------+------------------+
```

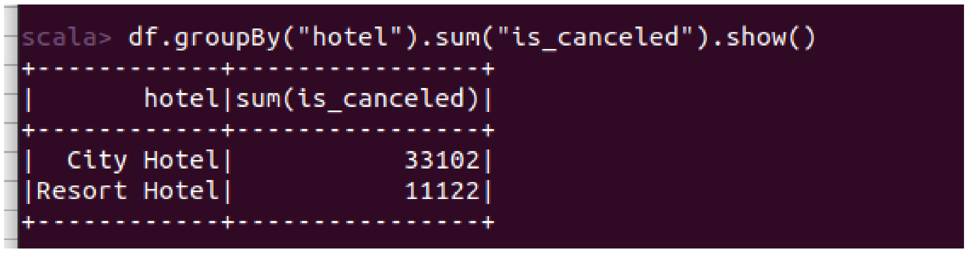
Python:



```
>>> df.describe("adults").show()
+-------+------------------+
|summary|            adults|
+-------+------------------+
|  count|            119390|
|   mean|1.8564033838679956|
| stddev|0.5792609988327523|
|    min|                 0|
|    max|                 6|
+-------+------------------+
```

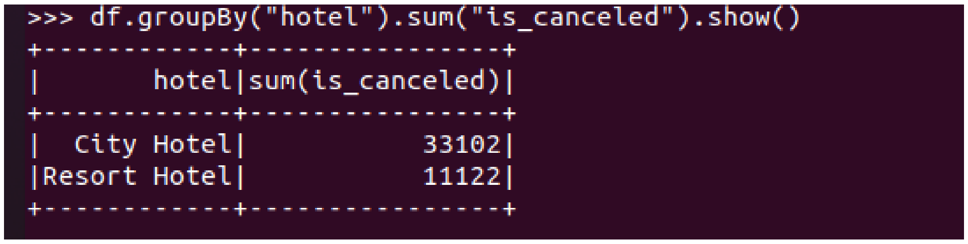
Count total number of canceled by hotel.

Scala:



```
scala> df.groupBy("hotel").sum("is_canceled").show()
+------------+----------------+
|       hotel|sum(is_canceled)|
+------------+----------------+
|  City Hotel|           33102|
|Resort Hotel|           11122|
+------------+----------------+
```

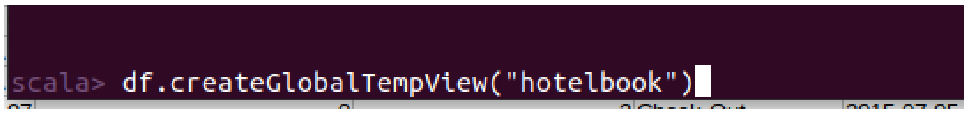
Python:



```
>>> df.groupBy("hotel").sum("is_canceled").show()
+------------+----------------+
|       hotel|sum(is_canceled)|
+------------+----------------+
|  City Hotel|           33102|
|Resort Hotel|           11122|
+------------+----------------+
```

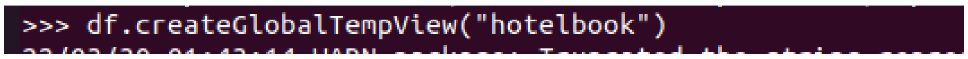
Register the DataFrame as a global temporary view.

Scala:



```
scala> df.createGlobalTempView("hotelbook")
```

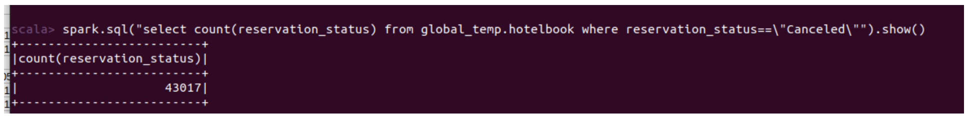
Python:



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

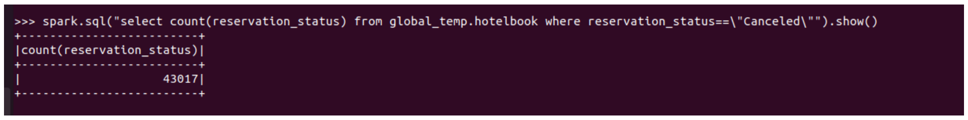
Use query to count number of records is reservation_status=”canceled”.

Scala:



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

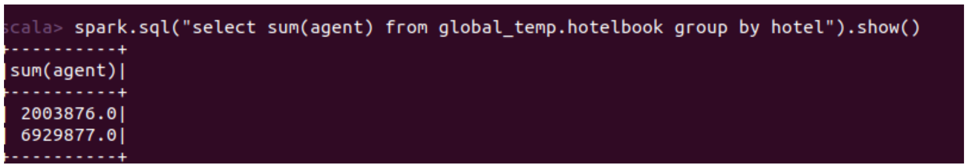
Python:



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

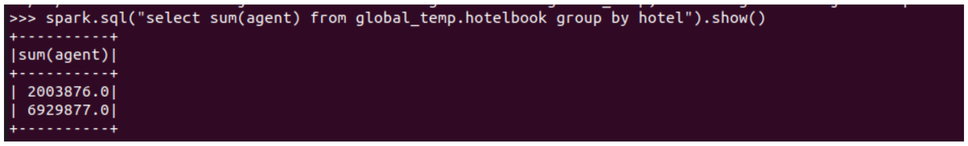
Use query to count the number of agent group by hotel.

Scala:



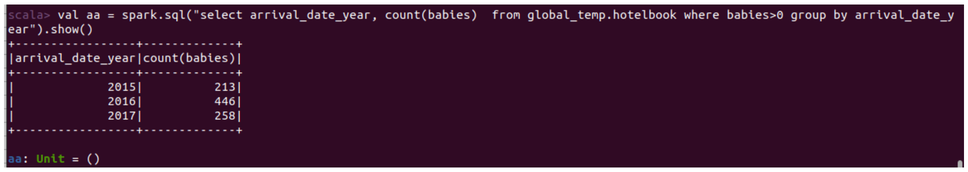
```
cala> spark.sql("select sum(agent) from global_temp.hotelbook group by hotel").show()
----------+
sum(agent)|
----------+
 2003876.0|
 6929877.0|
----------+
```

Python:

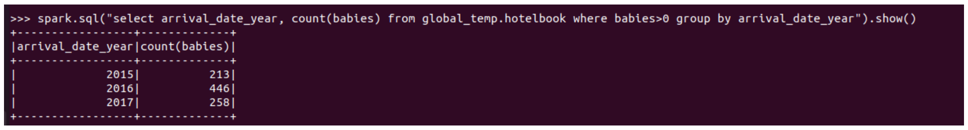

```
>>> spark.sql("select sum(agent) from global_temp.hotelbook group by hotel").show()
+----------+
|sum(agent)|
+----------+
| 2003876.0|
| 6929877.0|
+----------+
```

Use query to count the number of babies when babies are greater than 0 by year.

Scala:

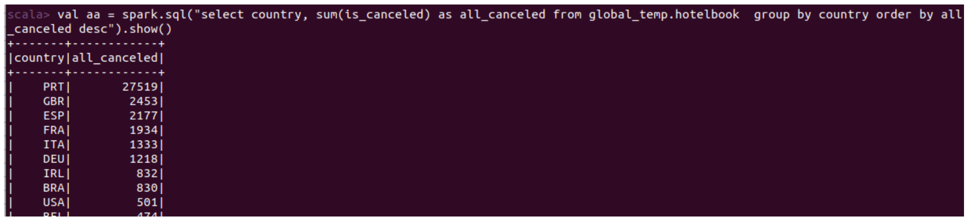

```
scala> val aa = spark.sql("select arrival_date_year, count(babies)  from global_temp.hotelbook where babies>0 group by arrival_date_y
ear").show()
+-----------------+-------------+
|arrival_date_year|count(babies)|
+-----------------+-------------+
|             2015|          213|
|             2016|          446|
|             2017|          258|
+-----------------+-------------+

aa: Unit = ()
```

Python:

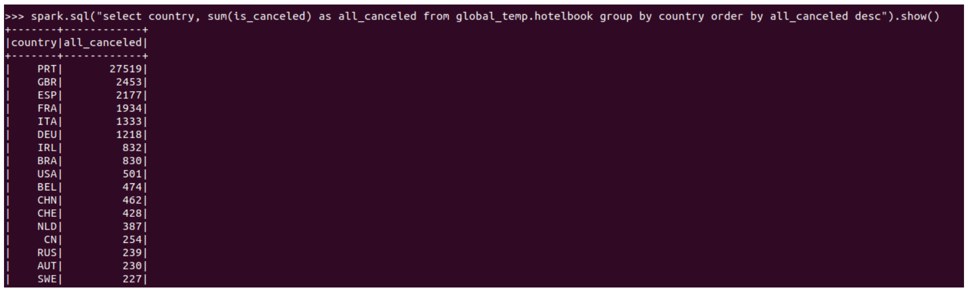

```
>>> spark.sql("select arrival_date_year, count(babies) from global_temp.hotelbook where babies>0 group by arrival_date_year").show()
+-----------------+-------------+
|arrival_date_year|count(babies)|
+-----------------+-------------+
|             2015|          213|
|             2016|        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

Use query to sort the number of canceled by country in decreasing order.

Scala:


```
scala> val aa = spark.sql("select country, sum(is_canceled) as all_canceled from global_temp.hotelbook  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|
```

Python:


```
>>> spark.sql("select country, sum(is_canceled) as all_canceled from global_temp.hotelbook 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|
```

ICE Submission Guidelines

1. ICE Submission is individual.
2. ICE code has to be properly commented.
3. The documentation should include the screenshots of your code/queries and results.
4. Provide the explanation of the exercise for each question as per your understanding.
5. The similarity score for your document should be less than 15%.
6. Submit the source code (if any) properly commented and documentation (.pdf/.doc) with explanation and screenshot of source code/queries having input logic and output results.
7. Submission after the deadline is considered as late submission.

References:

https://spark.apache.org/docs/latest/rdd-programming-guide.html#rdd-operations

https://spark.apache.org/docs/2.2.0/sql-programming-guide.html

https://sparkbyexamples.com/pyspark-rdd

https://sparkbyexamples.com/pyspark/different-ways-to-create-dataframe-in-pyspark/