```
#Installing pyspark library
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.3.2.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 4.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 3.4 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.2-py2.py3-none-any.whl size=281824028 sha256=65c27268bfbae64baa21c9c4ba839580aeb6
  Stored in directory: /root/.cache/pip/wheels/6c/e3/9b/0525ce8a69478916513509d43693511463c6468db0de237c86
Successfully built pyspark
Installing collected packages: py4j, pyspark
  Attempting uninstall: py4j
    Found existing installation: py4j 0.10.9.7
    Uninstalling py4j-0.10.9.7:
      Successfully uninstalled py4j-0.10.9.7
Successfully installed py4j-0.10.9.5 pyspark-3.3.2
```

Installing pyspark library

```
#Importing SparkSession from pysparkSQL
from pyspark.sql import SparkSession
#Imporrting sum and desc functions from sparkSQL functions
from pyspark.sql.functions import sum, desc

#Creating a spark session under app name 'ICE7' using builder function
spark = SparkSession.builder.appName('ICE7').getOrCreate()
```

By Creating a spark session in app we are using builder method that is present in SparkSession lib. getOrCreate() method is responsible for getting the app with the name passed as arg. If the app does not exisits then create one.

The created session is stored in variable.

```
spark #Spark session details
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.3.2
Master
local[*]
AppName
ICE7

Displaying the spark session

```
#Creating a RDD using the data from word_list-1
word_list_rdd = spark.sparkContext.textFile('/content/word_list-1.txt')
```

Creating a RDD by using the SparkContext.textFile which will read only data from the file located in the given path and returns a RDD of strings which will store in the word_list_rdd

```
word_list_rdd = word_list_rdd.map(lambda line: line.lower()) #Returns a list in which each element is a line from RDD transfromed into lc
word_list_rdd.take(4) #Take the first 4 elements of the RDD.
```

```
['the project gutenberg etext of moby word ii by grady ward',
 'copyright laws are changing all over the world, be sure to check',
 'the laws for your country before redistributing these files!!!',
 'please take a look at the important information in this header.']
```

Transforming all lines into lower case by using map function and passing to an anonymous function which iterates over each element/line in rdd and it will transform it into lower case letters.

```
word_list_rdd.count() #Return the number of elements in word_list_rdd RDD
```

```
260
```

Displaying total number of lines in word_list file or the number of elements in the RDD By using a count() method. The Output gives the count of lines in text file that is 260

```
#Displaying count of ocurrence word 'texas'
word_list_rdd.flatMap(lambda line: line.split(" ")).filter(lambda word : word.count("texas")).count()
```

```
2
```

Displaying the count of the word 'texas' by using the flatMap method of pyspark that takes an anonymous function It splits every line at space and flattens the resulted list by (converting list of lists into a single list) and returns it as a new RDD. On the new RDD filter method with a lambda function is used to get occurence of word 'texas'.

```
#Displaying count of ocurence of word TEXAS
word_list_rdd.flatMap(lambda line: line.split(" ")).filter(lambda word : word.count("TEXAS")).count() #Returns 0 as all the words are cha
```

```
0
```

Displaying the count of the word 'TEXAS' by using flatMap method of pyspark that takes an anonymous function which will splits the every line at space and flattens the resulted list(converting list of lists into a single list) and returns it to new rdd. On the new filter method with a lambda function is used to get occurence of word TEXAS. The output shows the count as 0 because we have to transform the each line in rdd into lower case hence 'TEXAS' is not matched anywhere

```
#Creating a Data Frame with the data from the csv file
data_frame = spark.read.format("com.dtabricks.spark.csv")\
.option("mode", "DROPMALFORMED").option("header" ,True)\
.option("inferschema", True).csv("/content/hotel_bookings.csv")
```

Creating a data frame with data from hotels csv file and with options header as true to skip 1st row in csv, inferSchema as true to define data types of each column in csv and mode as dropmalformed to exclude data from the column that does not match the data type defined in schema. In format we are mentioning the format of the file that we are reading

```
#Displaying & Computing the statistical values of 'children' column in the data frame
data_frame.describe("children").show()
```

```
+-------+------------------+
|summary|          children|
+-------+------------------+
|  count|            119390|
|   mean|0.10388990333874994|
| stddev| 0.3985614447864427|
|    min|                 0|
|    max|                NA|
+-------+------------------+
```

Q3 A: Statistical values of column "children", output displays the 5 basic statistical values
1) count- To Count the total number of values in children column
2)Mean of the column
3)stddev - The standard deviation of the values in children column
4)min value of the column: 0 will be the minmum value in the children column
5)Max value of the column

```
#Displaying the total number of cancelled reservations 2 hotel types
data_frame.groupBy('hotel').agg(sum('is_canceled').alias('Number_Of_Cancelled_Reservations')).show()
```

```
+------------+----------------------+
|       hotel|Canceleled_Reservations|
+------------+----------------------+
|  City Hotel|                 33102|
|Resort Hotel|                 11122|
+------------+----------------------+
```

Q3 B: Now Displaying the number of reservations that is cancelled (under column name 'Number_Of_Cancelled_Reservations') made by the users and by grouping the results by hotel name.

The output displays all the hotels and number of reservations cancelled by users.

grouBy().agg() It is used to do this operation and within the agg funcution sum() it is used to add the values in 'is_Cancelled' column and with alias method the sum is displayed under a column named 'Number_Of_Cancelled_Reservations'.

```
#Registering the data frame with hotels data as a Global temporary view
data_frame.createGlobalTempView('hotel_records')
```

Q3 D: Creating global temporary view of hotels data frame with name hotel_records which will be globally available to all the spark sessions Present in the app until Spark application is live

```
#Displaying the total number of cancelled reservations made by users and grouping them by user's country
data_frame.groupBy('country').agg(sum('is_canceled').alias("Total_Cancellations")).sort(desc("Total_Cancellations"))\
.show(180)
```

```
+-------+-------------------+
|country|Total_Cancellations|
+-------+-------------------+
|    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|
|    ISR|                169|
|    ROU|                134|
|    LUX|                109|
|    MAR|                109|
|    DNK|                109|
|    AUS|                107|
|    TUR|                102|
|    HUN|                 77|
|    FIN|                 69|
|   NULL|                 67|
|    KOR|                 55|
|    ARG|                 54|
|    ARE|                 43|
|    CZE|                 37|
|    GRC|                 35|
|    IND|                 35|
|    SAU|                 33|
|    ZAF|                 31|
|    JPN|                 28|
|    HKG|                 26|
|    HRV|                 25|
|    PHL|                 25|
|    IDN|                 24|
|    SVK|                 24|
|    IRN|                 23|
|    COL|                 23|
|    NGA|                 21|
|    DZA|                 21|
|    UKR|                 20|
|    MOZ|                 19|
|    TUN|                 19|
|    EST|                 18|
|    THA|                 18|
|    CHL|                 16|
|    SGP|                 16|
```

```
Q3 E: Displaying total number of reservations cancelled (under colum
'Total_Cancellations') by users and grouping the results by the coun

The output displays all the countries having the total number of can
reservations made by users from that country.

grouBy().agg() this is used to do this operation and within agg func
is used to add the values in 'is_Cancelled' column and with alias me
sum is displayed under a column named 'Total_Cancellations'.
```

Q3 E: Displaying total number of reservations cancelled (under column name 'Total_Cancellations') by users and grouping the results by the country.

The output displays all the countries having the total number of cancelled reservations made by users from that country.

```
The results can be sorted by using the sort() method and within this
() is used to describe the way results should be sorted. Since we re
results that need to be in the descending order of total cancelled r
we are passing 'Total_Cancellations' column into dec method
```

grouBy().agg() this is used to do this operation and within agg funcution sum() is used to add the values in 'is_Cancelled' column and with alias method the sum is displayed under a column named 'Total_Cancellations'.

The results can be sorted by using the sort() method and within this method desc() is used to describe the way results should be sorted. Since we require the results that need to be in the descending order of total cancelled reservatios we are passing 'Total_Cancellations' column into dec method