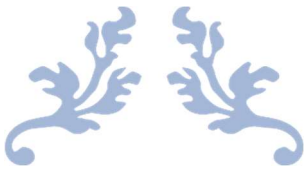
uOttawa



ELG5166

Spark APIs

NOVEMBER 3, 2022

ALI AMIN EL-SAYED MAHMOUD EL-SHERIF
300327246

Question 1:

```scala
/* Q) 1 */
val sales = spark
  .read
  .option("inferSchema", "true")
  .option("header", "true")
  .csv("/FileStore/tables/Data/Sales.csv")
sales.count()
sales.createOrReplaceTempView("Sales")

sales.show(3)
```

▸ (5) Spark Jobs

▸ sales: org.apache.spark.sql.DataFrame = [Transaction_date: string, Product: string ... 10 more fields]

```
+----------------+--------+-----+------------+----------------+--------------------+-------+--------------+--------------+------------+--------+--------
--+
|Transaction_date| Product|Price|Payment_Type|            Name|                City|  State|       Country|Account_Created|  Last_Login|Latitude| Longitu
de|
+----------------+--------+-----+------------+----------------+--------------------+-------+--------------+--------------+------------+--------+--------
--+
|     1/2/09 6:17|Product1| 1200|  Mastercard|         carolina|           Basildon|England|United Kingdom|    1/2/09 6:00| 1/2/09 6:08|    51.5|-1.11666
67|
|     1/2/09 4:53|Product1| 1200|        Visa|           Betina|Parkville         ...|     MO| United States|    1/2/09 4:42| 1/2/09 7:49|  39.195| -94.681
94|
|    1/2/09 13:08|Product1| 1200|  Mastercard|Federica e Andrea|Astoria           ...|     OR| United States|   1/1/09 16:21|1/3/09 12:32|46.18806|   -123.
83|
+----------------+--------+-----+------------+----------------+--------------------+-------+--------------+--------------+------------+--------+--------
--+
only showing top 3 rows
```

```scala
val flightData = spark
  .read
  .option("inferSchema", "true")
  .option("header", "true")
  .csv("/FileStore/tables/flight_data.csv")
 .coalesce(5)
flightData.count()
flightData.cache()
flightData.createOrReplaceTempView("flight")

```

▸ (4) Spark Jobs

▸ flightData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string ... 1 more field]

```
flightData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string ... 1 more field]
```

Cmd 3

```scala
/*Q)1 b-a */
import org.apache.spark.sql.functions.min
var query = spark.sql("select min(count) from flight")
query.show()
```

▸ (2) Spark Jobs

▸ query: org.apache.spark.sql.DataFrame = [min(count): integer]

```
+----------+
|min(count)|
+----------+
|         1|
+----------+

import org.apache.spark.sql.functions.min
query: org.apache.spark.sql.DataFrame = [min(count): int]
```

Cmd 4

```sql
%sql
/*Q)1 b-b */
select min(count) from flight
```

▸ (2) Spark Jobs

Table +

	min(count)
1	1

Showing 1 row. | 0.56 seconds runtime Refreshed now

Question 2:

Cmd 5

```
%python
#Q)2
flight1 = spark.read.option("inferSchema", "true").option("header", "true").csv("/FileStore/tables/flight_data.csv")
flight1.createOrReplaceTempView("flight_data_2015")
```

▸ (2) Spark Jobs

Cmd 6

```
%sql
/*Q)2 a-a */
select sum(count) as total_number_flights,ORIGIN_COUNTRY_NAME from flight_data_2015 group by  ORIGIN_COUNTRY_NAME order by total_number_flights limit 3
```

▸ (2) Spark Jobs

Table +

	total_number_flights	ORIGIN_COUNTRY_NAME
1	1	Singapore
2	1	Lithuania
3	1	Croatia

Showing all 3 rows. | 2.31 seconds runtime Refreshed now

Cmd 7

Scala

```
/*Q)2 a-b */
var query = spark.sql("select sum(count) as total_number_flights,ORIGIN_COUNTRY_NAME from flight_data_2015 group by  ORIGIN_COUNTRY_NAME order by total_number_flights limit 3")
query.show()
```

▸ (2) Spark Jobs

▸ query: org.apache.spark.sql.DataFrame = [total_number_flights: long, ORIGIN_COUNTRY_NAME: string]

```
+-------------------+-------------------+
|total_number_flights|ORIGIN_COUNTRY_NAME|
+-------------------+-------------------+
|                  1|          Singapore|
|                  1|          Lithuania|
|                  1|            Croatia|
+-------------------+-------------------+

query: org.apache.spark.sql.DataFrame = [total_number_flights: bigint, ORIGIN_COUNTRY_NAME: string]
```

Scala

```
/*Q)2 a-c */
var query = spark.sql("select sum(count) as total_number_flights,ORIGIN_COUNTRY_NAME from flight_data_2015 group by  ORIGIN_COUNTRY_NAME order by total_number_flights limit 3")
query.show()

flightData
  .groupBy("ORIGIN_COUNTRY_NAME")
  .sum("count")
  .sort("total_number_flights")
  .limit(3)
```

▸ (2) Spark Jobs

```
+-------------------+-------------------+
|total_number_flights|ORIGIN_COUNTRY_NAME|
+-------------------+-------------------+
|                  1|          Singapore|
|                  1|          Lithuania|
|                  1|            Croatia|
+-------------------+-------------------+

⊞AnalysisException: Column 'total_number_flights' does not exist. Did you mean one of the following? [sum(count), ORIGIN_COUNTRY_NAME];
'Sort ['total_number_flights ASC NULLS FIRST], true
+- Aggregate [ORIGIN_COUNTRY_NAME#192], [ORIGIN_COUNTRY_NAME#192, sum(count#193) AS sum(count)#503L]
   +- Repartition 5, false
      +- Relation [DEST_COUNTRY_NAME#191,ORIGIN_COUNTRY_NAME#192,count#193] csv
```

```scala
/*Q)2 b  */
var query = spark.sql("select sum(count) as total_number_flights,ORIGIN_COUNTRY_NAME from flight_data_2015 group by  ORIGIN_COUNTRY_NAME order by total_number_flights limit 3")
query.show()
query.explain()
```

(2) Spark Jobs

query: org.apache.spark.sql.DataFrame = [total_number_flights: long, ORIGIN_COUNTRY_NAME: string]

```
+--------------------+-------------------+
|total_number_flights|ORIGIN_COUNTRY_NAME|
+--------------------+-------------------+
|                   1|          Singapore|
|                   1|          Lithuania|
|                   1|            Croatia|
+--------------------+-------------------+

== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- TakeOrderedAndProject(limit=3, orderBy=[total_number_flights#507L ASC NULLS FIRST], output=[total_number_flights#507L,ORIGIN_COUNTRY_NAME#421])
   +- HashAggregate(keys=[ORIGIN_COUNTRY_NAME#421], functions=[finalmerge_sum(merge sum#518L) AS sum(count#422)#508L])
      +- Exchange hashpartitioning(ORIGIN_COUNTRY_NAME#421, 200), ENSURE_REQUIREMENTS, [id=#792]
         +- HashAggregate(keys=[ORIGIN_COUNTRY_NAME#421], functions=[partial_sum(count#422) AS sum#518L])
            +- FileScan csv [ORIGIN_COUNTRY_NAME#421,count#422] Batched: false, DataFilters: [], Format: CSV, Location: InMemoryFileIndex(1 paths)[dbfs:/FileStore/tables/flight_data.csv], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<ORIGIN_COUNTRY_NAME:string,count:int>


query: org.apache.spark.sql.DataFrame = [total_number_flights: bigint, ORIGIN_COUNTRY_NAME: string]
```

Question 3:

(2) Spark Jobs

flightsDataFile: org.apache.spark.sql.DataFrame = [DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string ... 1 more field]

flights: org.apache.spark.sql.Dataset[Flight] = [DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string ... 1 more field]

flights_Metadata: org.apache.spark.sql.Dataset[FlightMetadata] = [count: long, random: long]

```
+-----+--------------------+
|count|              random|
+-----+--------------------+
|    0|-5790796081456756646|
|    1|-7406709754318658195|
|    2| 3723754083696420047|
|    3|-3678622977270750094|
|    4|-8820342020588039680|
|    5|  554201728505525082|
|    6|-4210117342112615418|
|    7| 8640157511789465638|
|    8| 5967035757421660660|
|    9| 3968210518744556750|
|   10| 3652462764861634911|
|   11| 4179064758276548709|
|   12| 3928348356421848608|
|   13|-4441871604060770695|
|   14|-5151248049971105193|
|   15|-1522132841203857217|
|   16| 4844185957588460211|
|   17|-5312010116759265553|
```

```scala
/*Q)3 a*/

case class Flight(DEST_COUNTRY_NAME: String,
                  ORIGIN_COUNTRY_NAME: String, count: BigInt)

/* Reads the Parquet file */
val flightsDataFile = spark.read
  .parquet("/FileStore/tables/Data/flight-data/parquet/2010-summary.parquet/part_r_00000_1a9822ba_b8fb_4d8e_844a_ea30d0801b9e_gz.parquet")

/* map the file to a typed dataset using Flight class*/
val flights = flightsDataFile.as[Flight]

/* create a structured dataset of the Type FlightMetadata */
case class FlightMetadata(count: BigInt, random: BigInt)

/* 500 entries with an index column and another with random numbers */
val flights_Metadata = spark.range(500)
  .map(x => (x, scala.util.Random.nextLong))
  .withColumnRenamed("_1", "count")
  .withColumnRenamed("_2", "random")
  .as[FlightMetadata]
flights_Metadata.show()
```

Scala

```scala
/*Q)3 b*/
val flights_2 = flights
  .joinWith(flights_Metadata, flights.col("count") === flights_Metadata.col("count"))
   .withColumnRenamed("_1", "count")
  .withColumnRenamed("_2", "random")
flights_2.show()
```

flights_2: org.apache.spark.sql.DataFrame = [count: struct, random: struct]

```
+--------------------+--------------------+
|               count|              random|
+--------------------+--------------------+
|{United States, U...|{1, -208177811221...|
|{United States, F...|{1, -208177811221...|
|{Bulgaria, United...|{1, -208177811221...|
|{United States, S...|{1, -208177811221...|
|{United States, C...|{1, -208177811221...|
|{United States, B...|{1, -208177811221...|
|{United States, I...|{1, -208177811221...|
|{Vietnam, United ...|{1, -208177811221...|
|{United States, P...|{1, -208177811221...|
|{United States, B...|{1, -208177811221...|
|{United States, G...|{1, -208177811221...|
|{United States, L...|{1, -208177811221...|
|{Malaysia, United...|{1, -208177811221...|
|{United States, S...|{1, -208177811221...|
|{United States, A...|{1, -208177811221...|
|{United States, E...|{1, -208177811221...|
|{United States, V...|{1, -208177811221...|
|{Liberia, United ...|{1, -208177811221...|
```

Question 4:

Cmd 12

SQL

```sql
%sql
/* Q)4 a */

DROP TABLE IF EXISTS flights;
CREATE TABLE flights (
  DEST_COUNTRY_NAME STRING, ORIGIN_COUNTRY_NAME STRING, count LONG)
USING JSON OPTIONS (path '/FileStore/tables/Data/flight-data/json/*.json')
```

OK

```scala
/* Q)4 b*/

flights.rdd.getNumPartitions
```

```
res18: Int = 1
```

Question 5:

```scala
/* Q)5 a*/

val textFile = spark.sparkContext.textFile("/FileStore/tables/Data/Adult.csv")
textFile.count()
```

```
▸ (1) Spark Jobs
textFile: org.apache.spark.rdd.RDD[String] = /FileStore/tables/Data/Adult.csv MapPartitionsRDD[108] at textFile at command-3522973057872627:3
res19: Long = 32562
```

```scala
/* Q)5 b*/

val count = textFile.flatMap(line => line.split(","))
                    .map(word => (word, 1))
                    .reduceByKey(_+_).takeOrdered(5)(Ordering[Int].on(_._2))
count.foreach(println)
```

```
▸ (1) Spark Jobs
(87,1)
(marital_status,1)
(age,1)
(Holand-Nether,1)
(occupation,1)
count: Array[(String, Int)] = Array((87,1), (marital_status,1), (age,1), (Holand-Nether,1), (occupation,1))
```

Question 6:

```scala
/* Q)6 */

val dataframe = spark.read.format("csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .load("/FileStore/tables/2010_12_10.csv")

dataframe.printSchema()

// Create a view.
dataframe.createOrReplaceTempView("dfTable")
```

```
▸ (2) Spark Jobs
▸ dataframe: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)

dataframe: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
```

Cmd 17

```scala
/*Q)6 a*/

import org.apache.spark.sql.functions.col
dataframe.where(col("Quantity").equalTo(12) && col("UnitPrice").gt(2))
  .select("*")
  .show(5, false)
```

(1) Spark Jobs

```
+---------+---------+----------------------------------+--------+-------------------+---------+----------+--------------+
|InvoiceNo|StockCode|Description                       |Quantity|InvoiceDate        |UnitPrice|CustomerID|Country       |
+---------+---------+----------------------------------+--------+-------------------+---------+----------+--------------+
|538172   |22041    |"RECORD FRAME 7"" SINGLE SIZE "   |12      |2010-12-10 09:33:00|2.55     |15805.0   |United Kingdom|
|538172   |84558A   |3D DOG PICTURE PLAYING CARDS      |12      |2010-12-10 09:33:00|2.95     |15805.0   |United Kingdom|
|538174   |22326    |ROUND SNACK BOXES SET OF4 WOODLAND|12      |2010-12-10 09:35:00|2.95     |12471.0   |Germany       |
|538174   |22472    |TV DINNER TRAY DOLLY GIRL         |12      |2010-12-10 09:35:00|4.95     |12471.0   |Germany       |
|538174   |22445    |PENCIL CASE LIFE IS BEAUTIFUL     |12      |2010-12-10 09:35:00|2.95     |12471.0   |Germany       |
+---------+---------+----------------------------------+--------+-------------------+---------+----------+--------------+
only showing top 5 rows

import org.apache.spark.sql.functions.col
```

Cmd 18

```scala
/*Q)6 b*/

import org.apache.spark.sql.functions.split
dataframe.select(split(col("Description"), " ").alias("Detailed description")).show(false)
```

(1) Spark Jobs

```
+----------------------------------------+
|Detailed description                    |
+----------------------------------------+
|[HAWAIIAN, GRASS, SKIRT, ]              |
|[CHILLI, LIGHTS]                        |
|["RECORD, FRAME, 7"", SINGLE, SIZE, "]  |
|[3D, DOG, PICTURE, PLAYING, CARDS]      |
|[60, CAKE, CASES, VINTAGE, CHRISTMAS]   |
|[PAPER, CHAIN, KIT, VINTAGE, CHRISTMAS] |
|[CHRISTMAS, TOILET, ROLL]               |
|["ASSORTED, FLOWER, COLOUR, ""LEIS"""]  |
|[I, CAN, ONLY, PLEASE, ONE, PERSON, MUG]|
|[HAND, WARMER, BABUSHKA, DESIGN]        |
|[BLUE, HARMONICA, IN, BOX, ]            |
|null                                    |
|[SET, OF, 72, RETROSPOT, PAPER, , DOILIES]|
|[PACK, OF, 72, RETROSPOT, CAKE, CASES]  |
|[WOODLAND, DESIGN, , COTTON, TOTE, BAG] |
|[BIG, DOUGHNUT, FRIDGE, MAGNETS]        |
|[RECYCLED, PENCIL, WITH, RABBIT, ERASER]|
```

Cmd 19

```scala
/*Q)6 2-a*/
```

```
%fs
ls /FileStore/tables
```

Table +

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/2010_12_01.csv	2010_12_01.csv	275001	1667513508000
2	dbfs:/FileStore/tables/2010_12_02.csv	2010_12_02.csv	191826	1667513508000
3	dbfs:/FileStore/tables/2010_12_10.csv	2010_12_10.csv	241468	1667513718000
4	dbfs:/FileStore/tables/Data/	Data/	0	0
5	dbfs:/FileStore/tables/flight_data.csv	flight_data.csv	7080	1666034682000

Showing all 5 rows. | 1.13 seconds runtime

Refreshed 2 minutes ago

Cmd 21

```scala
/*Q)6 2-b*/

val dataset = spark.read.format("csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .load("/FileStore/tables/2010_12_02.csv")

dataset.createOrReplaceTempView("dfTable")

import org.apache.spark.sql.functions.{sum,min, max,avg}

dataset.select(sum("UnitPrice"),min("UnitPrice"), max("UnitPrice"),avg("UnitPrice")).show()
```

(4) Spark Jobs

dataset: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
+------------------+--------------+--------------+------------------+
|   sum(UnitPrice)|min(UnitPrice)|max(UnitPrice)|   avg(UnitPrice)|
+------------------+--------------+--------------+------------------+
|6815.679999999958|           0.0|         295.0|3.231711711711692|
+------------------+--------------+--------------+------------------+
```

Question 7:

Cmd 22

```scala
/* Q)7 */

val dataset = spark.read.format("csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .load("/FileStore/tables/2010_12_01.csv")

dataset.createOrReplaceTempView("RetailView")
dataset.count()
```

(4) Spark Jobs

dataset: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
dataset: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
res29: Long = 3108
```

Cmd 23

```scala
/*Q)7 a*/
val  datadescription= dataset.filter(x =>
                  {
                    val allStr = x.getString(2).split(" ")
                    var check = true
                    for(x <- allStr ){
                      if (!x(0).isUpper)
                      {
                          check=false

                      }
                  }
                  (check)
                  })

datadescription.show(5,false)
```

(1) Spark Jobs

datadescription: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
+---------+---------+----------------------------------+--------+------------------+---------+----------+--------------+
|InvoiceNo|StockCode|Description                       |Quantity|InvoiceDate       |UnitPrice|CustomerID|Country       |
+---------+---------+----------------------------------+--------+------------------+---------+----------+--------------+
|536365   |85123A   |WHITE HANGING HEART T-LIGHT HOLDER |6       |2010-12-01 08:26:00|2.55     |17850.0   |United Kingdom|
|536365   |71053    |WHITE METAL LANTERN               |6       |2010-12-01 08:26:00|3.39     |17850.0   |United Kingdom|
|536365   |84406B   |CREAM CUPID HEARTS COAT HANGER    |8       |2010-12-01 08:26:00|2.75     |17850.0   |United Kingdom|
|536365   |84029G   |KNITTED UNION FLAG HOT WATER BOTTLE|6       |2010-12-01 08:26:00|3.39     |17850.0   |United Kingdom|
|536365   |84029E   |RED WOOLLY HOTTIE WHITE HEART.    |6       |2010-12-01 08:26:00|3.39     |17850.0   |United Kingdom|
+---------+---------+----------------------------------+--------+------------------+---------+----------+--------------+
only showing top 5 rows
```

Cmd 24

```
/*Q)7 b*/
/*Drop a any row that contains NULLs using any*/

dataset.na.drop("any")

```

▸ res33: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

res33: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
/*Drop a any row that contains NULLs using all*/

dataset.na.drop("all")
```

▸ res35: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

```
res35: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
```