CSP554—Big Data Technologies

Assignment #7

Worth: 12 points (2 points for each problem)

Exercise 1)

Step B

Use the TestDataGen program from previous assignments to generate new data files.

Copy the files to the directory "/user/hadoop" in HDFS

```
[hadoop@ip-172-31-4-238 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-4-238 ~]$ java TestDataGen
Magic Number = 3531
[hadoop@ip-172-31-4-238 ~]$ ls
foodplaces3531.txt  foodratings3531.txt  TestDataGen.class
```

```
[hadoop@ip-172-31-4-238 ~]$ hadoop fs -copyFromLocal *.txt /user/hadoop

[hadoop@ip-172-31-4-238 ~]$
[hadoop@ip-172-31-4-238 ~]$ hadoop fs -ls /user/hadoop
Found 2 items
-rw-r--r--   1 hadoop hadoop         59 2021-03-10 03:01 /user/hadoop/foodplaces3531.txt
-rw-r--r--   1 hadoop hadoop      17513 2021-03-10 03:01 /user/hadoop/foodratings3531.txt
```

Magic Number = 3531

Step C

Load the 'foodratings' file as a 'csv' file into a DataFrame called foodratings. When doing so specify a schema having fields of the following names and types:

Field Name	Field Type
Name	String
food1	Integer
food2	Integer
food3	Integer
food4	Integer
Placeid	Integer

As the results of this exercise provide the magic number, *the code you execute* and screen shots of the following commands:

foodratings.printSchema()

foodratings.show(5)

```
>>> from pyspark.sql.types import *
>>> struct1 = StructType().add("name", StringType(), True).add("food1", IntegerType(), True).add("food2", IntegerType(), True).add("food3", IntegerType(), True).add("food4", IntegerType(), True).add("placeid",
IntegerType(), True)
>>> foodratings = spark.read.schema(struct1).csv('hdfs:///user/hadoop/foodratings3531.txt')
>>>
>>>
>>> foodratings.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>>
>>> foodratings.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|    2|   37|   36|    6|      2|
|Jill|   44|   29|    8|   16|      3|
| Joy|   11|   16|   19|   28|      5|
| Mel|   17|   37|   32|    5|      1|
| Joe|   22|   28|   42|   35|      3|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Exercise 2)

Load the 'foodplaces' file as a 'csv' file into a DataFrame called foodplaces. When doing so specify a schema having fields of the following names and types:

Field Nampee	Field Type
placeid	Integer
placename	String

As the results of this exercise provide *the code you execute* and screen shots of the following commands:

foodratings.printSchema()

foodratings.show(5)

Magic Number = 3531

```
>>> from pyspark.sql.types import *
>>> struct2 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), True)
>>> foodplaces = spark.read.schema(struct2).csv('hdfs:///user/hadoop/foodplaces3531.txt')
>>>
>>>
>>> foodplaces.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces.show(5)
+-------+-----------+
|placeid|  placename|
+-------+-----------+
|      1|China Bistro|
|      2|    Atlantic|
|      3|  Food Town|
|      4|     Jake's|
|      5|  Soup Bowl|
+-------+-----------+
```

Exercise 3)

Magic Number = 3531

Step A

Register the DataFrames created in exercise 1 and 2 as tables called “foodratingsT” and “foodplacesT”

```
>>> foodratings.registerTempTable("foodratingsT")
>>> foodplaces.registerTempTable("foodplacesT")
```

Step B

Use a SQL query on the table “foodratingsT” to create a new DataFrame called foodratings_ex3a holding records which meet the following condition: food2 < 25 and food4 > 40. Remember, when defining conditions in your code use maximum parentheses.

As the results of this step *provide the code you execute* and screen shots of the following commands:

foodratings_ex3a.printSchema()

foodratings_ex3a.show(5)

```
>>> foodratings_ex3a = sqlContext.sql("select * from foodratingsT where food2 < 25 and food4 > 40")
>>> foodratings_ex3a.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex3a.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Joy|   21|    5|   29|   45|      5|
| Sam|    2|    5|   42|   42|      1|
| Sam|   27|   20|   33|   49|      5|
| Joy|    9|   15|   45|   42|      3|
| Sam|   43|    7|   50|   46|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Step C

Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces_ex3b holding records which meet the following condition: placeid > 3

As the results of this step *provide the code you execute* and screen shots of the following commands:

foodplaces_ex3b.printSchema()

foodplaces_ex3b.show(5)

```
>>> foodplaces_ex3b = sqlContext.sql("select * from foodplacesT where placeid > 3")
>>> foodplaces_ex3b.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces_ex3b.show(5)
+-------+---------+
|placeid|placename|
+-------+---------+
|      4|   Jake's|
|      5|Soup Bowl|
+-------+---------+
```

Exercise 4)

Use a transformation (not a SparkSQL query) on the DataFrame 'foodratings' created in exercise 1 to create a new DataFrame called foodratings_ex4 that includes only those records (rows) where the 'name' field is "Mel" and food3 < 25.

As the results of this step provide the code you execute and screen shots of the following commands:

foodratings_ex4.printSchema()

foodratings_ex4.show(5)

Magic Number = 3531

```
>>> foodratings_ex4 = foodratings.filter( (foodratings.name == "Mel") & (foodratings.food3 < 25) )
>>> foodratings_ex4.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|    9|   47|   21|   24|      5|
| Mel|    5|    1|   13|   17|      4|
| Mel|   32|   12|    2|    2|      4|
| Mel|   39|   27|   23|   23|      3|
| Mel|   34|   36|   21|   21|      5|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Exercise 5)

Use a transformation (not a SparkSQL query) on the DataFrame 'foodratings' created in exercise 1 to create a new DataFrame called foodratings_ex5 that includes only the columns (fields) 'name' and 'placeid'

As the results of this step provide the code you execute and screen shots of the following commands:

foodratings_ex5.printSchema()

foodratings_ex5.show(5)

Magic Number = 3531

```
>>> foodratings_ex5 = foodratings.select("name", "placeid")
>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
| Mel|      2|
|Jill|      3|
| Joy|      5|
| Mel|      1|
| Joe|      3|
+----+-------+
only showing top 5 rows
```

Exercise 6)

Use a transformation (not a SparkSQL query) to create a new DataFrame called ex6 which is the inner join, on placeid, of the DataFrames 'foodratings; and 'foodplaces' created in exercises 1 and 2

As the results of this step provide the code you execute and screen shots of the following commands:

ex6.printSchema()

ex6.show(5)

Magic Number = 3531

```
>>> ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid, "inner")
>>> ex6.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> ex6.show(5)
+----+-----+-----+-----+-----+-------+-------+------------+
|name|food1|food2|food3|food4|placeid|placeid|   placename|
+----+-----+-----+-----+-----+-------+-------+------------+
| Mel|    2|   37|   36|    6|      2|      2|    Atlantic|
|Jill|   44|   29|    8|   16|      3|      3|   Food Town|
| Joy|   11|   16|   19|   28|      5|      5|   Soup Bowl|
| Mel|   17|   37|   32|    5|      1|      1|China Bistro|
| Joe|   22|   28|   42|   35|      3|      3|   Food Town|
+----+-----+-----+-----+-----+-------+-------+------------+
only showing top 5 rows
```