Kajol Tanesh Shah
A20496724
Fall 2022

CSP554—Big Data Technologies

Assignment #7

Exercise 1)

Step B
Use the TestDataGen program from previous assignments to generate new data files. Copy both generated files to the HDFS directory "/user/hadoop"

Ans.

```
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~/Desktop/CSP554_Big_Data$ scp -i e
mr-key-pair.pem TestDataGen.class hadoop@ec2-54-197-97-185.compute-1.amazonaws.
com:/home/hadoop
TestDataGen.class

          100% 2189    24.6KB/s   00:00
(base) kajol@kajol-Lenovo-ideapad-310-15IKB:~/Desktop/CSP554_Big_Data$
```

```
[hadoop@ip-172-31-52-250 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-52-250 ~]$ java TestDataGen
Magic Number = 156190
[hadoop@ip-172-31-52-250 ~]$ ls
foodplaces156190.txt  foodratings156190.txt  TestDataGen.class
[hadoop@ip-172-31-52-250 ~]$
```

```
[hadoop@ip-172-31-52-250 ~]$ hadoop fs -copyFromLocal foodplaces156190.txt /user/hadoop/foodplaces156190.csv
[hadoop@ip-172-31-52-250 ~]$ hadoop fs -copyFromLocal foodratings156190.txt /user/hadoop/foodratings156190.csv
[hadoop@ip-172-31-52-250 ~]$ hadoop fs -ls
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2022-10-23 08:24 foodplaces156190.csv
-rw-r--r--   1 hadoop hdfsadmingroup      17439 2022-10-23 08:24 foodratings156190.csv
[hadoop@ip-172-31-52-250 ~]$
```

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 code you execute* and screen shots of the following commands:

foodratings.printSchema()
foodratings.show(5)

Ans.

```
[hadoop@ip-172-31-52-250 ~]$ pyspark
Python 3.7.10 (default, Jun  3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/10/23 08:34:07 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.8-amzn-2
      /_/

Using Python version 3.7.10 (default, Jun  3 2021 00:02:01)
SparkSession available as 'spark'.
```

```
>>> struct1 = StructType(
...     [
...             StructField("name", StringType(), True),
...             StructField("food1", IntegerType(), True),
...             StructField("food2", IntegerType(), True),
...             StructField("food3", IntegerType(), True),
...             StructField("food4", IntegerType(), True),
…             StructField("placeid", IntegerType(), True)
...     ]
... )
>>> foodratings = spark.read.schema(struct1).csv('/user/hadoop/foodratings156190.csv')
```

```
>>> from pyspark.sql.types import *
```

```
>>> struct1 = StructType(
...     [
...             StructField("name", StringType(), True),
...             StructField("food1", IntegerType(), True),
...             StructField("food2", IntegerType(), True),
...             StructField("food3", IntegerType(), True),
...             StructField("food4", IntegerType(), True),
...             StructField("placeid", IntegerType(), True)
...     ]
... )
>>> foodratings = spark.read.schema(struct1).csv('/user/hadoop/foodratings156190.csv')
```

```
>>> 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|
+----+-----+-----+-----+-----+-------+
| Joe|    6|    1|    2|   44|      1|
| Sam|   45|   24|    5|   43|      2|
| Joe|   10|   12|   47|   32|      3|
| Sam|   11|   29|   40|   44|      2|
| Joy|   38|   15|   18|   33|      5|
+----+-----+-----+-----+-----+-------+
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 Name	Field Type
placeid	Integer
placename	String

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

foodplaces.printSchema()
foodplaces.show(5)

Ans.

```
>>> struct2 = StructType(
...     [
...             StructField("placeid", IntegerType(), True),
...             StructField("placename", StringType(), True)
...     ]
... )
>>> foodplaces = spark.read.schema(struct2).csv('/user/hadoop/foodplaces156190.csv')
```

```
>>> struct2 = StructType(
...     [
...             StructField("placeid", IntegerType(), True),
...             StructField("placename", StringType(), True)
...     ]
... )
>>> foodplaces = spark.read.schema(struct2).csv('/user/hadoop/foodplaces156190.csv')
```

```
>>> 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)

<u>Step A</u>
Register the DataFrames created in exercise 1 and 2 as tables called “foodratingsT” and “foodplacesT”

Ans.
foodratings.createOrReplaceTempView("foodratingsT")

foodplaces.createOrReplaceTempView("foodplacesT")

```
>>> foodratings.createOrReplaceTempView("foodratingsT")
>>> foodplaces.createOrReplaceTempView("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.

foodratings_ex3a.printSchema()
foodratings_ex3a.show(5)

Ans.
foodratings_ex3a = spark.sql(“SELECT * FROM foodratingsT WHERE food2 < 25 AND food4 > 40”)

```
>>> foodratings_ex3a = spark.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|
+----+-----+-----+-----+-----+-------+
| Joe|    6|    1|    2|   44|      1|
| Sam|   45|   24|    5|   43|      2|
| Joy|   49|    6|   37|   46|      3|
| Joy|   10|    4|    3|   42|      3|
|Jill|   19|   15|    6|   42|      4|
+----+-----+-----+-----+-----+-------+
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
foodplaces_ex3b.printSchema()

foodplaces_ex3b.show(5)

Ans.

foodplaces_ex3b = spark.sql(“SELECT * FROM foodplacesT WHERE placeid > 3”)

```
>>> foodplaces_ex3b = spark.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)

Ans.

foodratings4_ex4 = foodratings.filter((foodratings[‘name’] == ‘Mel’ & foodratings[‘food3’] < 25))

```
>>> 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|   42|   10|   12|   14|      1|
| Mel|   32|    4|   12|   31|      3|
| Mel|   26|   15|   10|   22|      1|
| Mel|   50|   17|    8|   10|      4|
| Mel|   39|    7|   10|   46|      3|
+----+-----+-----+-----+-----+-------+
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)

Ans.
foodratings_ex5 = foodratings.select(foodratings['name'], foodratings['placeid'])

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

>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
| Joe|      1|
| Sam|      2|
| Joe|      3|
| Sam|      2|
| Joy|      5|
+----+-------+
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)

Ans.
ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid, 'inner')

```
>>> 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|
+----+-----+-----+-----+-----+-------+-------+------------+
| Joe|    6|    1|    2|   44|      1|      1|China Bistro|
| Sam|   45|   24|    5|   43|      2|      2|    Atlantic|
| Joe|   10|   12|   47|   32|      3|      3|   Food Town|
| Sam|   11|   29|   40|   44|      2|      2|    Atlantic|
| Joy|   38|   15|   18|   33|      5|      5|   Soup Bowl|
+----+-----+-----+-----+-----+-------+-------+------------+
only showing top 5 rows

>>>
```