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"

Maagic number:

```
[hadoop@ip-172-31-2-156 ~]$ java TestDataGen
Magic Number = 192572
[hadoop@ip-172-31-2-156 ~]$
```

Command: hadoop fs -ls /user/hadoop

Output:

```
[hadoop@ip-172-31-2-156 ~]$ hadoop fs -ls /user/hadoop
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2023-11-09 20:24 /user/hadoop/foodplaces192572.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17447 2023-11-09 20:24 /user/hadoop/foodratings192572.txt
[hadoop@ip-172-31-2-156 ~]$
```

Step C

Load the 'foodratings' file as a 'csv' file into a DataFrame called foodratings.
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)

Commands:

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/foodratings192572.txt')
foodratings.printScheme( )
foodratings.show(5)

Output:

```
>>> 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/foodratings192572.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|
+----+-----+-----+-----+-----+-------+
| Joe|   25|   15|    5|   41|      5|
|Jill|   13|   16|    3|   26|      5|
| Mel|   36|   14|    7|   17|      1|
| Joe|   24|   27|   30|    7|      1|
| Mel|   10|   40|   19|   20|      3|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Exercise 2)

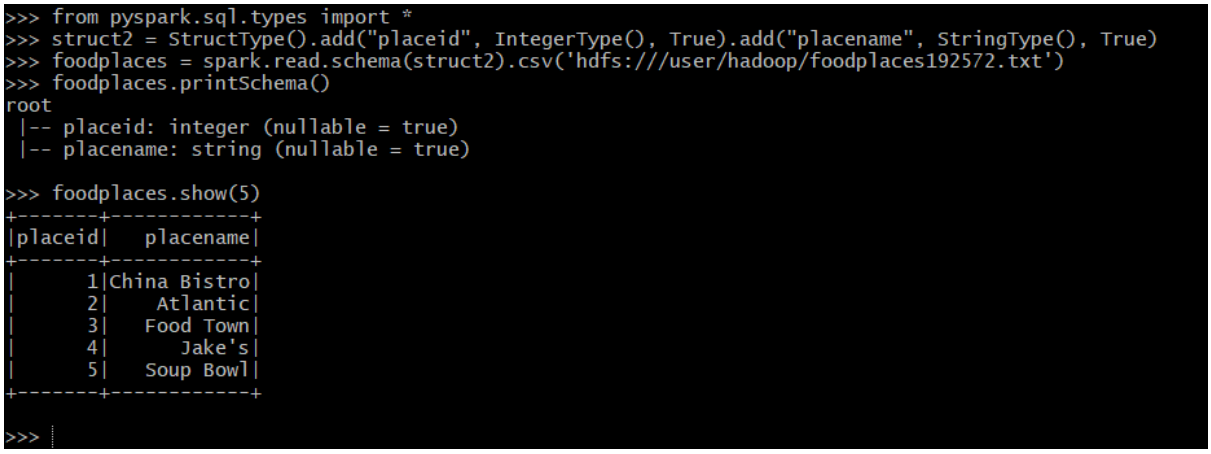
Load the ‘foodplaces’ file as a ‘csv’ file into a DataFrame called foodplaces.

As the results of this exercise provide the code you execute and screen shots of the following commands:
foodplaces.printSchema()
foodplaces.show(5)

Commands:

From pyspark.sql.types import *
struct2 = StructType().add(“placeid”, IntegerType(), True).add(“placename”, StringType(), True)
foodplaces = spark.read.schema(struct2).csv(‘hdfs:///user/hadoop/foodplaces192572.txt’)
foodplaces.printSchema()
foodplaces.show(5)

Output:



Exercise 3)

Step A

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

Commands:

Foodplaces.registerTempTable(“foodplacesT”)
Foodratings.registerTempTable(“foodratingsT”)

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)

Commands:

foodratings_ex3a = sqlContext.sql(“select * from foodratingsT where food2 < 25 and food4 > 40”)
foodratings_ex3a.printSchema()
foodratings_ex3a.show(5)

Output:

```
>>> foodplaces.registerTempTable("foodplacesT")
>>> foodratings.registerTempTable("foodratingsT")
>>> 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|
+----+-----+-----+-----+-----+-------+
| Joe|   25|   15|    5|   41|      5|
| Joy|   18|   23|   11|   46|      5|
| Sam|   14|   19|   43|   41|      2|
| Joe|   12|   22|   14|   41|      4|
| Mel|   34|   20|   44|   48|      1|
+----+-----+-----+-----+-----+-------+
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)

Commands:

foodplaces_ex3b = sqlContext.sql(“select * from foodplacesT where placeid > 3”)
foodplaces_ex3b.printSchema()
foodplaces_ex3b.show(5)

Output:

```
>>> 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)

Commands:

foodratings_ex4 = foodratings.filter((foodratings.name == "Mel") & (foodratings.food3 < 25))
foodratings_ex4.printSchema()
foodratings_ex4.show(5)

Output:

```
>>> 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|   36|   14|    7|   17|      1|
| Mel|   10|   40|   19|   20|      3|
| Mel|   12|   16|    9|   13|      3|
| Mel|   37|   18|    6|   33|      5|
| Mel|   25|   45|   16|   23|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>>
```

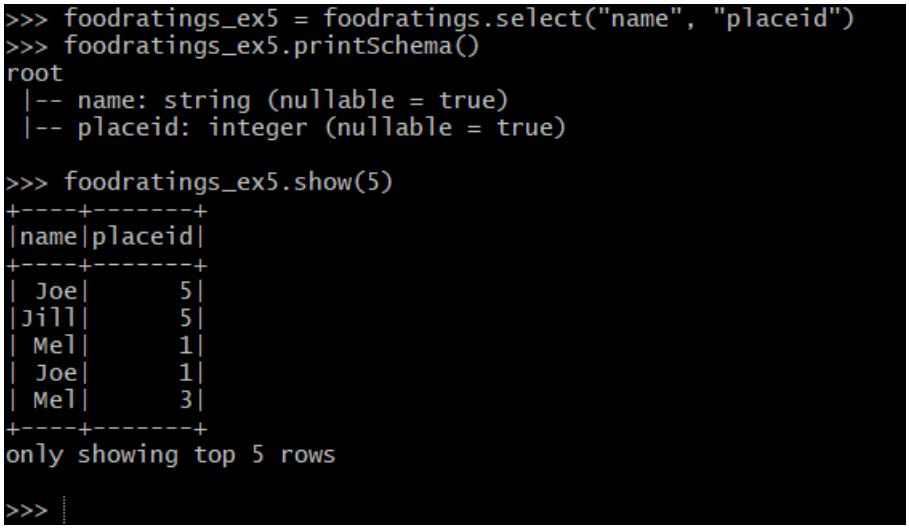
Exercise 5)

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)

Commands:

foodratings_ex5 = foodratings.select("name", "placeid")
foodratings_ex5.printSchema()
foodratings_ex5.show(5)

Output:



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)

Commands:

ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid, "inner")
ex6.printSchema()

Output:

```
>>> 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|   25|   15|    5|   41|      5|      5|   Soup Bowl|
|Jill|   13|   16|    3|   26|      5|      5|   Soup Bowl|
| Mel|   36|   14|    7|   17|      1|      1|China Bistro|
| Joe|   24|   27|   30|    7|      1|      1|China Bistro|
| Mel|   10|   40|   19|   20|      3|      3|   Food Town|
+----+-----+-----+-----+-----+-------+-------+------------+
only showing top 5 rows

>>>
```