# CSP554—Big Data Technologies

## Assignment #7

Exercise 1)

Step A
Use the TestDataGen program from previous assignments to generate new data files
Copy the files to HDFS.

```
[maria_dev@sandbox-hdp ~]$ java TestDataGen
Magic Number = 107949
[maria_dev@sandbox-hdp ~]$ ls -la | grep 107949
-rw-rw-r--  1 maria_dev maria_dev       59 Nov  1 14:08 foodplaces107949.txt
-rw-rw-r--  1 maria_dev maria_dev    17511 Nov  1 14:08 foodratings107949.txt
[maria_dev@sandbox-hdp ~]$ hadoop fs -put foodratings107949.txt hdfs:///user/maria_dev/foodratings107949.csv
[maria_dev@sandbox-hdp ~]$ hadoop fs -put foodplaces107949.txt hdfs:///user/maria_dev/foodplaces107949.csv
[maria_dev@sandbox-hdp ~]$ hadoop fs -ls hdfs:///user/maria_dev/ | grep 107949
-rw-r--r--   1 maria_dev hdfs         59 2018-11-01 14:10 hdfs:///user/maria_dev/foodplaces107949.csv
-rw-r--r--   1 maria_dev hdfs      17511 2018-11-01 14:10 hdfs:///user/maria_dev/foodratings107949.csv
[maria_dev@sandbox-hdp ~]$
```

```
java TestDataGen
hadoop fs -put foodratings107949.txt hdfs:///user/maria_dev/foodratings107949.csv
hadoop fs -put foodplaces107949.txt hdfs:///user/maria_dev/foodplaces107949.csv
```

**Magic Number = 107949**

Step B
Load the 'foodratings' file as a 'csv' file into a DataFrame called ex1_foodratings. When doing so specify a schema having fields of the following names and types:

| Field Nampee | 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:

ex1_foodratings.printSchema()
ex1_foodratings.head(5)

**Magic Number = 107949**

```
>>> from pyspark.sql.types import *
>>> schemaa = 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)
...         ]
... )
>>>
>>> ex1_foodratings = spark.read.schema(schemaa).csv('/user/maria_dev/foodratings107949.csv')
>>> ex1_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)

>>> print ex1_foodratings.head(5)
[Row(name=u'Joe', food1=23, food2=28, food3=32, food4=24, placeid=1), Row(name=u'Sam', food1=5, food2=28, food3=32, food4=41, placeid=2), Row(nam
od1=30, food2=22, food3=48, food4=39, placeid=1), Row(name=u'Sam', food1=11, food2=31, food3=24, food4=50, placeid=2), Row(name=u'Sam', food1=32,
ood3=50, food4=48, placeid=4)]
```

```
from pyspark.sql.types import *
schemaa = 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)
      ]
)

ex1_foodratings =
spark.read.schema(schemaa).csv('/user/maria_dev/foodratings107949.csv')
ex1_foodratings.printSchema()
print ex1_foodratings.head(5)
```

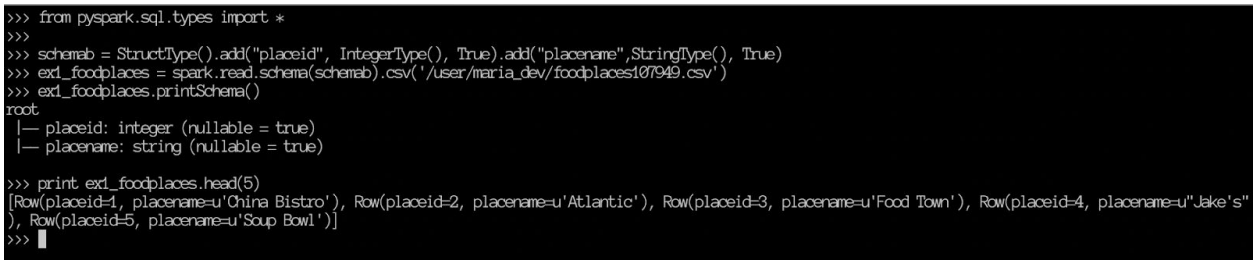
Exercise 2)

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 Ty |
|---|---|
| placeid | integer |
| placename | string |

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

ex1_foodplaces.printSchema()
ex1_foodplaces.head(5)




```
from pyspark.sql.types import *

schemab = StructType().add("placeid", IntegerType(), True).add("placename",StringType(), True)
ex1_foodplaces = spark.read.schema(schemab).csv('/user/maria_dev/foodplaces107949.csv')
ex1_foodplaces.printSchema()
print ex1_foodplaces.head(5)
```

Exercise 3)

Step A

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


```
ex1_foodratings.createOrReplaceTempView("foodratingsT")
ex1_foodplaces.createOrReplaceTempView("foodplacesT")
```

*N.B. - Same pyspark session  so ex1_foodratings, and ex1_foodplaces are available .*

Step B

Use a SQL query on the table “foodratingsT” to create a new DataFrame called foodratings_ex3 holding records which meet the following condition: food2 < 25 and food4 > 40

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

foodratings_ex3.printSchema()

foodratings_ex3.head(5)


```
foodratings_ex3 = spark.sql("SELECT * FROM foodratingsT WHERE food2 < 25 AND food4 > 40")
foodratings_ex3.printSchema()
print foodratings_ex3.head(5)
```

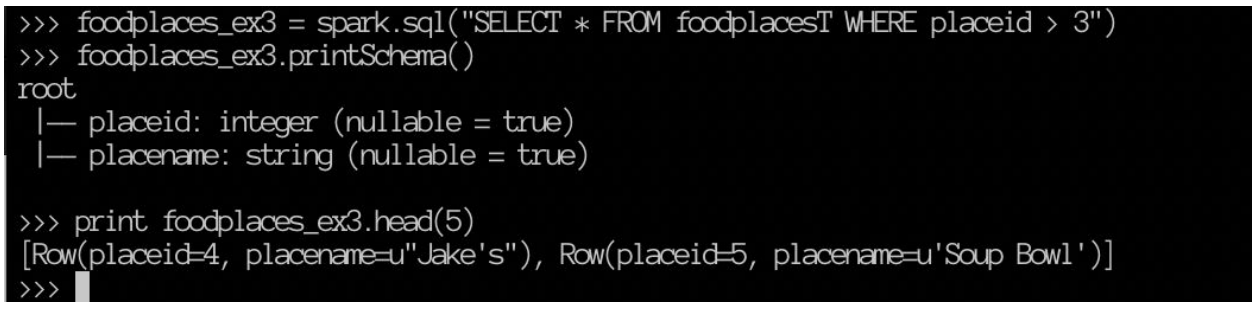
Step C

Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces_ex3 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_ex3.printSchema()

foodplaces_ex3.head(5)




```
foodplaces_ex3 = spark.sql("SELECT * FROM foodplacesT WHERE placeid > 3")
```

```
foodplaces_ex3.printSchema()
print foodplaces_ex3.head(5)
```

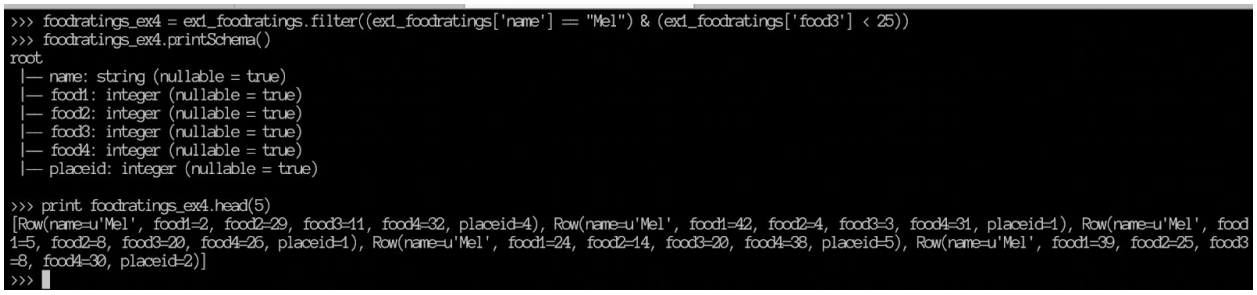
Exercise 4)

Use an operation (not a SQL query) on the DataFrame ‘foodratings’ create 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.head(5)




```
foodratings_ex4 = ex1_foodratings.filter((ex1_foodratings['name'] == "Mel") &
(ex1_foodratings['food3'] < 25))
foodratings_ex4.printSchema()
print foodratings_ex4.head(5)
```

*N.B. - Same pyspark session  so ex1_foodratings, and ex1_foodplaces are available .*

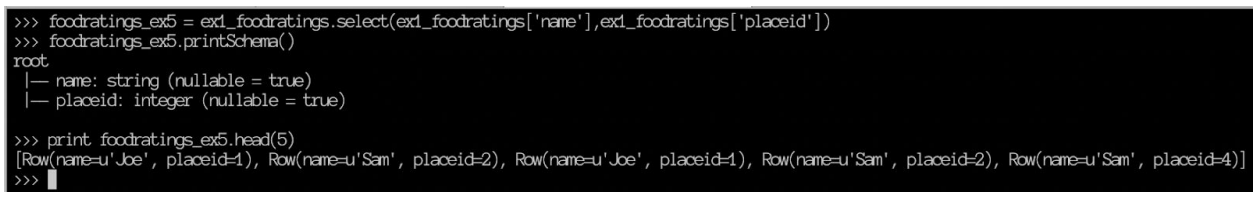
Exercise 5)

Use an operation (not a SQL query) on the DataFrame ‘foodratings’ create 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.head(5)




```
foodratings_ex5 = ex1_foodratings.select(ex1_foodratings['name'],ex1_foodratings['placeid'])
```

```
foodratings_ex5.printSchema()
print foodratings_ex5.head(5)
```

*N.B. - Same pyspark session  so ex1_foodratings, and ex1_foodplaces are available .*

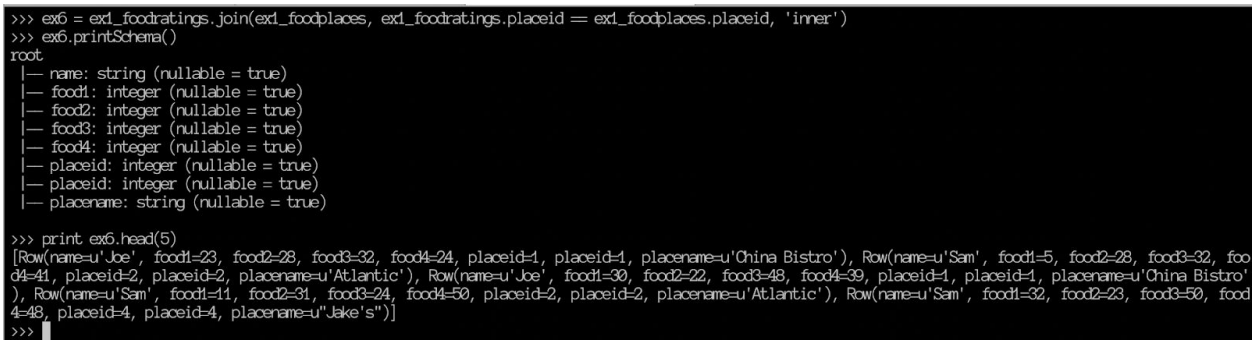
Exercise 6)

Use an operation on the DataFrame ‘to create a new DataFrame called ex6 which is the inner join, on placeid, of the DataFrames ‘ex1_foodratings” and ‘ex1_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.head(5)




```
ex6 = ex1_foodratings.join(ex1_foodplaces, ex1_foodratings.placeid ==
ex1_foodplaces.placeid, 'inner')
ex6.printSchema()
print ex6.head(5)
```