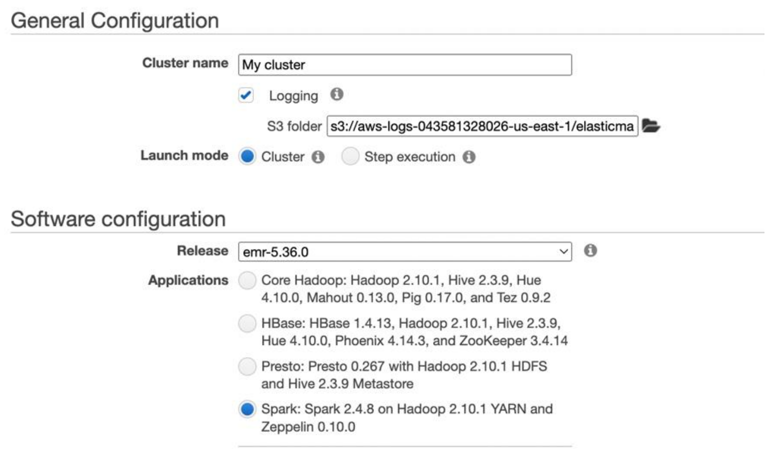
Assignment #7 BigData – Spark | Name: Akshay Jain | CWID: A20502846

Exercise 1) Step A

Start up a Hadoop cluster as previously, but instead of choosing the “Core Hadoop” configuration chose the “Spark” configuration (see below), otherwise proceed as before.



General Configuration

Cluster name: My cluster

[✓] Logging

S3 folder: s3://aws-logs-043581328026-us-east-1/elasticma

Launch mode: [● Cluster] ○ Step execution

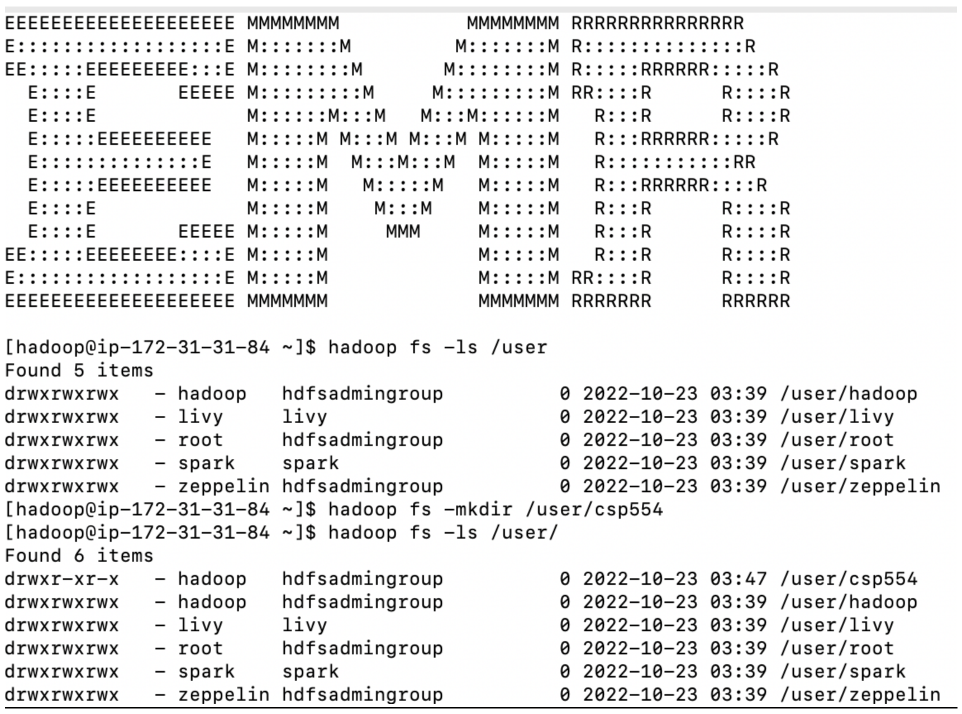
Software configuration

Release: emr-5.36.0

Applications:

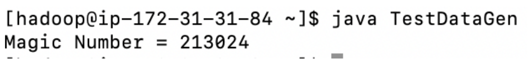
- ○ Core Hadoop: Hadoop 2.10.1, Hive 2.3.9, Hue 4.10.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ○ HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.9, Hue 4.10.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ○ Presto: Presto 0.267 with Hadoop 2.10.1 HDFS and Hive 2.3.9 Metastore
- [● Spark: Spark 2.4.8 on Hadoop 2.10.1 YARN and Zeppelin 0.10.0]

Step B



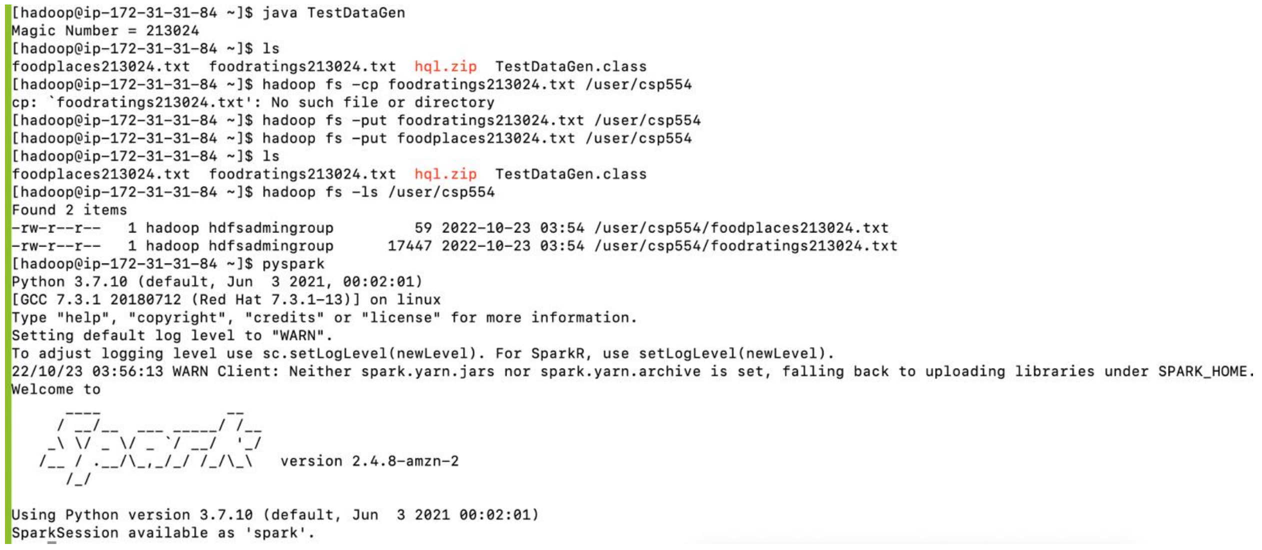
```
EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-31-84 ~]$ hadoop fs -ls /user
Found 5 items
drwxrwxrwx   - hadoop   hdfsadmingroup          0 2022-10-23 03:39 /user/hadoop
drwxrwxrwx   - livy     livy                    0 2022-10-23 03:39 /user/livy
drwxrwxrwx   - root     hdfsadmingroup          0 2022-10-23 03:39 /user/root
drwxrwxrwx   - spark    spark                   0 2022-10-23 03:39 /user/spark
drwxrwxrwx   - zeppelin hdfsadmingroup          0 2022-10-23 03:39 /user/zeppelin
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -mkdir /user/csp554
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -ls /user/
Found 6 items
drwxr-xr-x   - hadoop   hdfsadmingroup          0 2022-10-23 03:47 /user/csp554
drwxrwxrwx   - hadoop   hdfsadmingroup          0 2022-10-23 03:39 /user/hadoop
drwxrwxrwx   - livy     livy                    0 2022-10-23 03:39 /user/livy
drwxrwxrwx   - root     hdfsadmingroup          0 2022-10-23 03:39 /user/root
drwxrwxrwx   - spark    spark                   0 2022-10-23 03:39 /user/spark
drwxrwxrwx   - zeppelin hdfsadmingroup          0 2022-10-23 03:39 /user/zeppelin
```

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



```
[hadoop@ip-172-31-31-84 ~]$ java TestDataGen
Magic Number = 213024
```

Magic Number =213024



```
[hadoop@ip-172-31-31-84 ~]$ java TestDataGen
Magic Number = 213024
[hadoop@ip-172-31-31-84 ~]$ ls
foodplaces213024.txt  foodratings213024.txt  hql.zip  TestDataGen.class
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -cp foodratings213024.txt /user/csp554
cp: `foodratings213024.txt': No such file or directory
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -put foodratings213024.txt /user/csp554
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -put foodplaces213024.txt /user/csp554
[hadoop@ip-172-31-31-84 ~]$ ls
foodplaces213024.txt  foodratings213024.txt  hql.zip  TestDataGen.class
[hadoop@ip-172-31-31-84 ~]$ hadoop fs -ls /user/csp554
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2022-10-23 03:54 /user/csp554/foodplaces213024.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17447 2022-10-23 03:54 /user/csp554/foodratings213024.txt
[hadoop@ip-172-31-31-84 ~]$ 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 03:56:13 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'.
```

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:

hadoop fs -copyFromLocal /home/hadoop/foodratings213024.txt

from pyspark.sql.types import *

struct1 = StructType().add("name", StringType(), True).add("food1",IntegerType(),

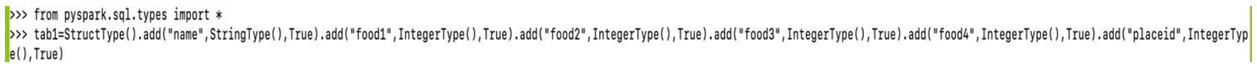
True).add("food2",IntegerType(), True).add("food3",IntegerType(), True).add("food4",IntegerType(),

True).add("placeid",IntegerType(), True) foodratings =

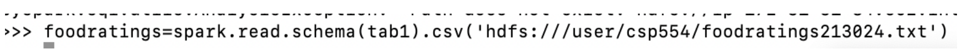
spark.read.schema(struct1).csv('foodratings213024.txt')

foodratings.printSchema()

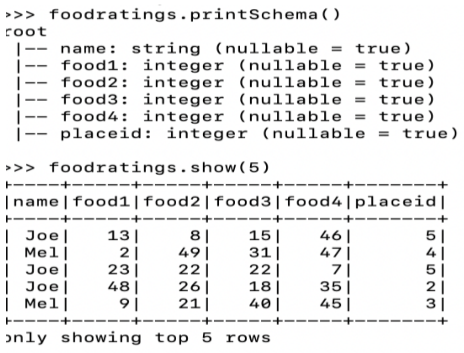
foodratings.show(5)



```
>>> from pyspark.sql.types import *
>>> tab1=StructType().add("name",StringType(),True).add("food1",IntegerType(),True).add("food2",IntegerType(),True).add("food3",IntegerType(),True).add("food4",IntegerType(),True).add("placeid",IntegerTyp
e(),True)
```



```
>>> foodratings=spark.read.schema(tab1).csv('hdfs:///user/csp554/foodratings213024.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|   13|    8|   15|   46|      5|
| Mel|    2|   49|   31|   47|      4|
| Joe|   23|   22|   22|    7|      5|
| Joe|   48|   26|   18|   35|      2|
| Mel|    9|   21|   40|   45|      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

hadoop fs -copyFromLocal /home/hadoop/foodplaces213024.txt

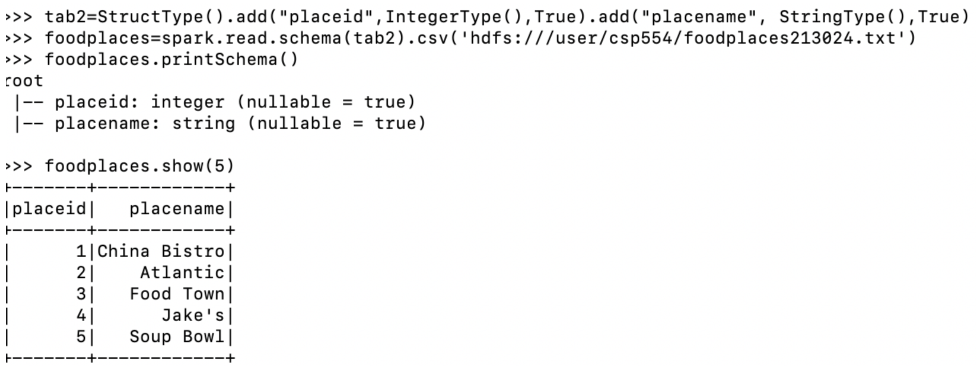
from pyspark.sql.types import *

struct1 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), True)

foodplaces = spark.read.schema(struct1).csv('foodplaces213024.txt')

foodplaces.printSchema()

foodplaces.show(5)



```
>>> tab2=StructType().add("placeid",IntegerType(),True).add("placename", StringType(),True)
>>> foodplaces=spark.read.schema(tab2).csv('hdfs:///user/csp554/foodplaces213024.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|
+-------+------------+
```

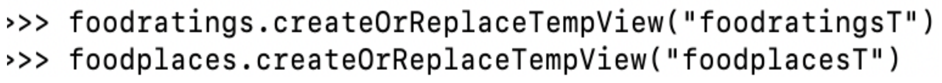
Exercise 3

Step A

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

foodratings.createOrReplaceTempView("foodratingsT")

foodplaces.createOrReplaceTempView("foodplacesT")



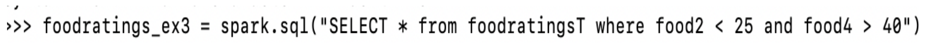
```
>>> foodratings.createOrReplaceTempView("foodratingsT")
>>> foodplaces.createOrReplaceTempView("foodplacesT")
```

Step B

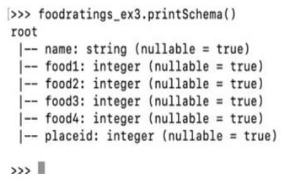
Use a SQL query on the table "foodratingsT" to create a new DataFrame called foodratings_ex3aholding records which meet the following condition: food2 < 25 and food4> 40. Remember, when defining conditions in your code use maximum parentheses

foodratings_ex3 = spark.sql("SELECT * from foodratingsT where food2 < 25 and food4 > 40")
foodratings_ex3.printSchema()

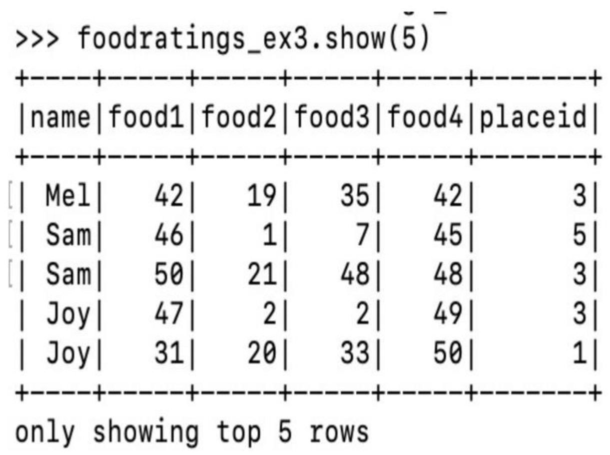
foodratings_ex3.show(5)



```
>>> foodratings_ex3 = spark.sql("SELECT * from foodratingsT where food2 < 25 and food4 > 40")
```



```
>>> foodratings_ex3.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)

>>> 
```



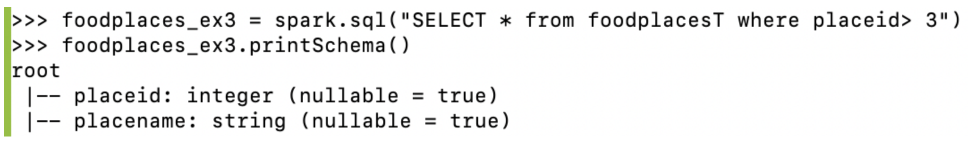
```
>>> foodratings_ex3.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   42|   19|   35|   42|      3|
| Sam|   46|    1|    7|   45|      5|
| Sam|   50|   21|   48|   48|      3|
| Joy|   47|    2|    2|   49|      3|
| Joy|   31|   20|   33|   50|      1|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Step C

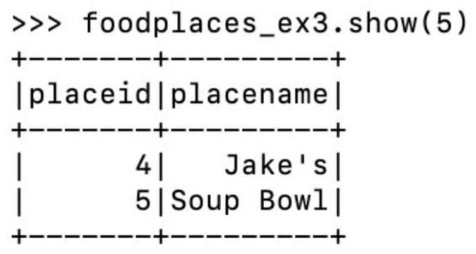
Use a SQL query on the table "foodplacesT" to create a new DataFrame called foodplaces_ex3bholding records which meet the following condition: placeid> 3

foodplaces_ex3 = spark.sql("SELECT * from foodplacesT where placeid> 3")
foodplaces_ex3.printSchema()

foodplaces_ex3.show(5)



```
>>> foodplaces_ex3 = spark.sql("SELECT * from foodplacesT where placeid> 3")
>>> foodplaces_ex3.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)
```



```
>>> foodplaces_ex3.show(5)
+-------+---------+
|placeid|placename|
+-------+---------+
|      4|   Jake's|
|      5|Soup Bowl|
+-------+---------+
```

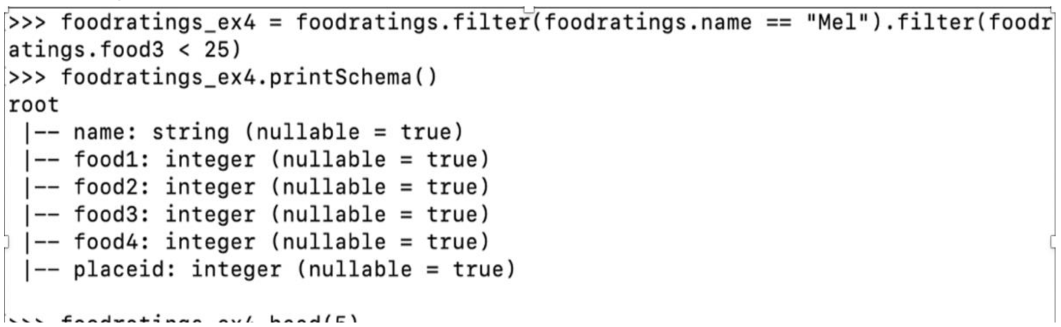
Exercise 4

Use a transformation(not aSparkSQL 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

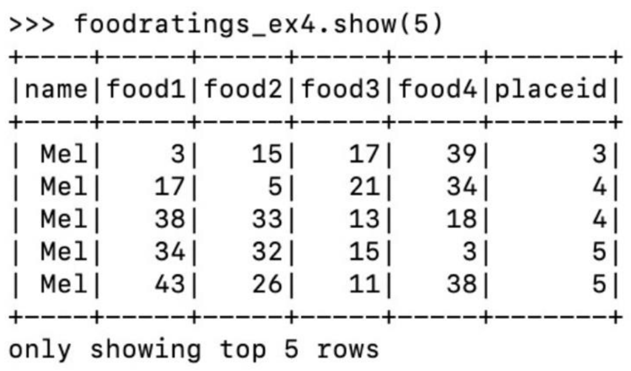
foodratings_ex4 = foodratings.filter(foodratings.name == "Mel").filter(foodratings.food3 < 25)

foodratings_ex4.printSchema()

foodratings_ex4.show(5)



```
>>> foodratings_ex4 = foodratings.filter(foodratings.name == "Mel").filter(foodr
atings.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.head(5)
```



```
>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|    3|   15|   17|   39|      3|
| Mel|   17|    5|   21|   34|      4|
| Mel|   38|   33|   13|   18|      4|
| Mel|   34|   32|   15|    3|      5|
| Mel|   43|   26|   11|   38|      5|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

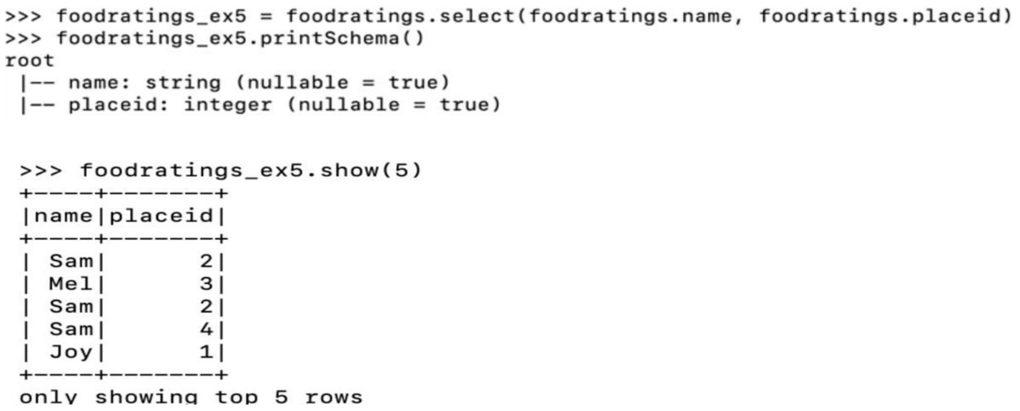
Exercise 5

Use a transformation (not aSparkSQL 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’

foodratings_ex5 = foodratings.select(foodratings.name, foodratings.placeid)

foodratings_ex5.printSchema()

foodratings_ex5.show(5)



```
>>> 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|
 +----+-------+
 | Sam|      2|
 | Mel|      3|
 | Sam|      2|
 | Sam|      4|
 | Joy|      1|
 +----+-------+
 only showing top 5 rows
```

Exercise 6

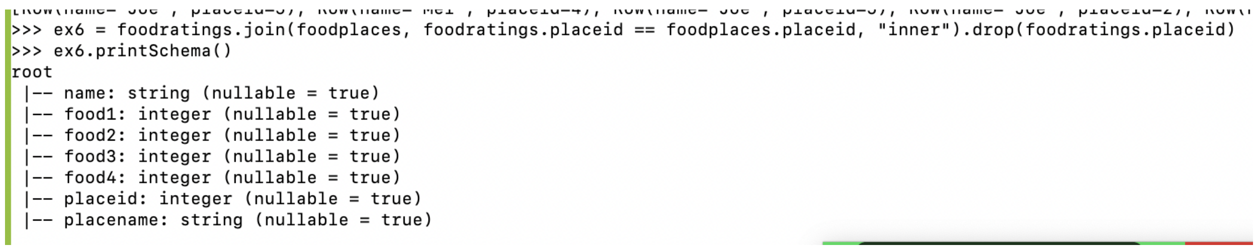
Use a transformation (not aSparkSQL query) to create a newDataFrame called ex6 which is the inner join, on placeid, of the DataFrames ‘foodratings’ and ‘foodplaces’ created in exercises 1 and 2

ex6 = foodratings.join(foodplaces, foodratings.placeid = =

foodplaces.placeid,"inner").drop(foodratings.placeid)

ex6.printSchema()

ex6.show(5)



```
[Row(name='Joe', placeid=5), Row(name='Mel', placeid=4), Row(name='Joe', placeid=5), Row(name='Joe', placeid=2), Row(
>>> ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid, "inner").drop(foodratings.placeid)
>>> 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)
 |-- placename: string (nullable = true)
```

```
>>> ex6.show(5)
+----+-----+-----+-----+-----+-------+------------+
|name|food1|food2|food3|food4|placeid|   placename|
+----+-----+-----+-----+-----+-------+------------+
| Sam|   33|   26|   38|    8|      2|    Atlantic|
| Mel|    3|   15|   17|   39|      3|   Food Town|
| Sam|   22|   14|   36|   14|      2|    Atlantic|
| Sam|   27|   16|   50|   47|      4|      Jake's|
| Joy|    8|   24|    1|   11|      1|China Bistro|
+----+-----+-----+-----+-----+-------+------------+
only showing top 5 rows
```