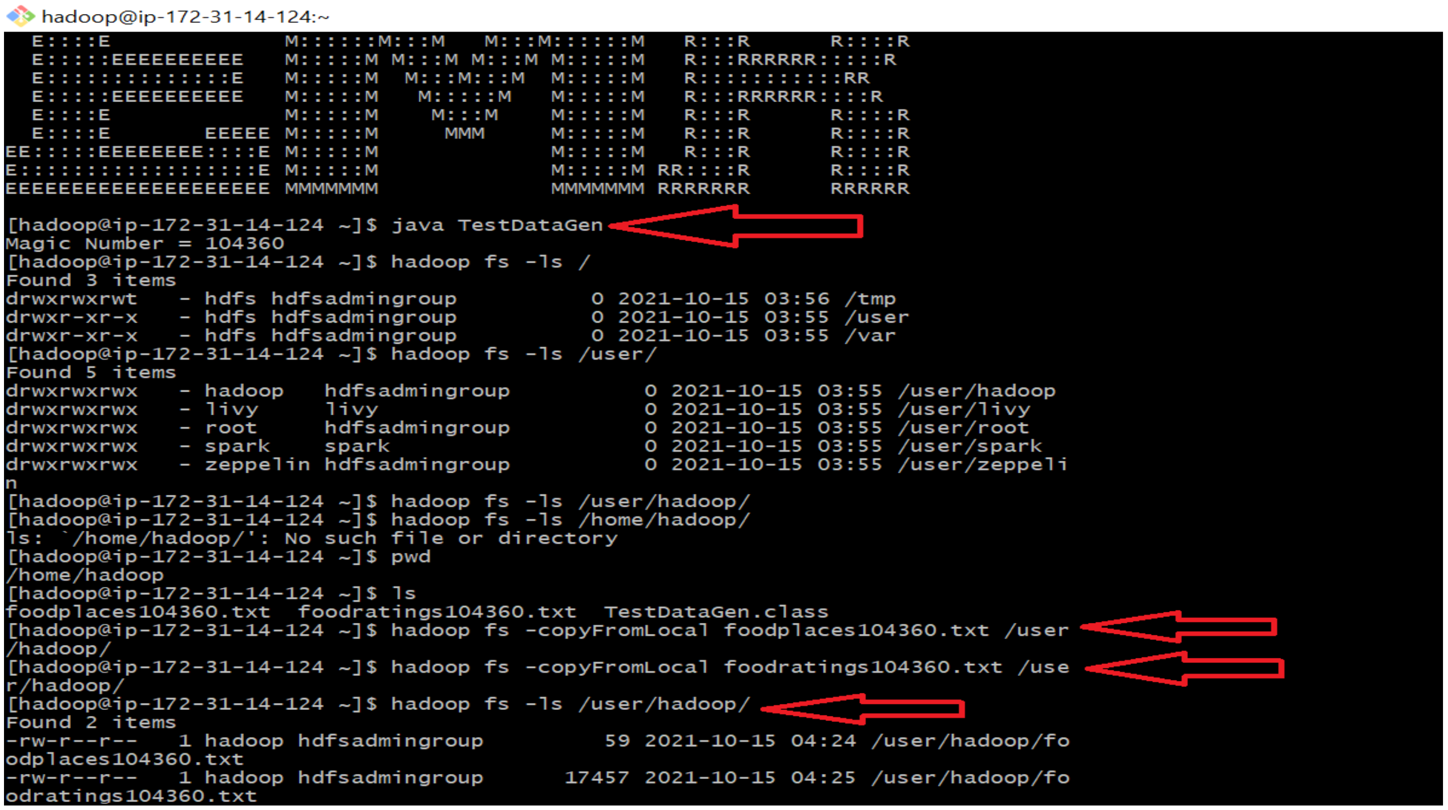
CSP – 554 Big Data Technologies

Assignment #7

Rutul Mehta
A20476293

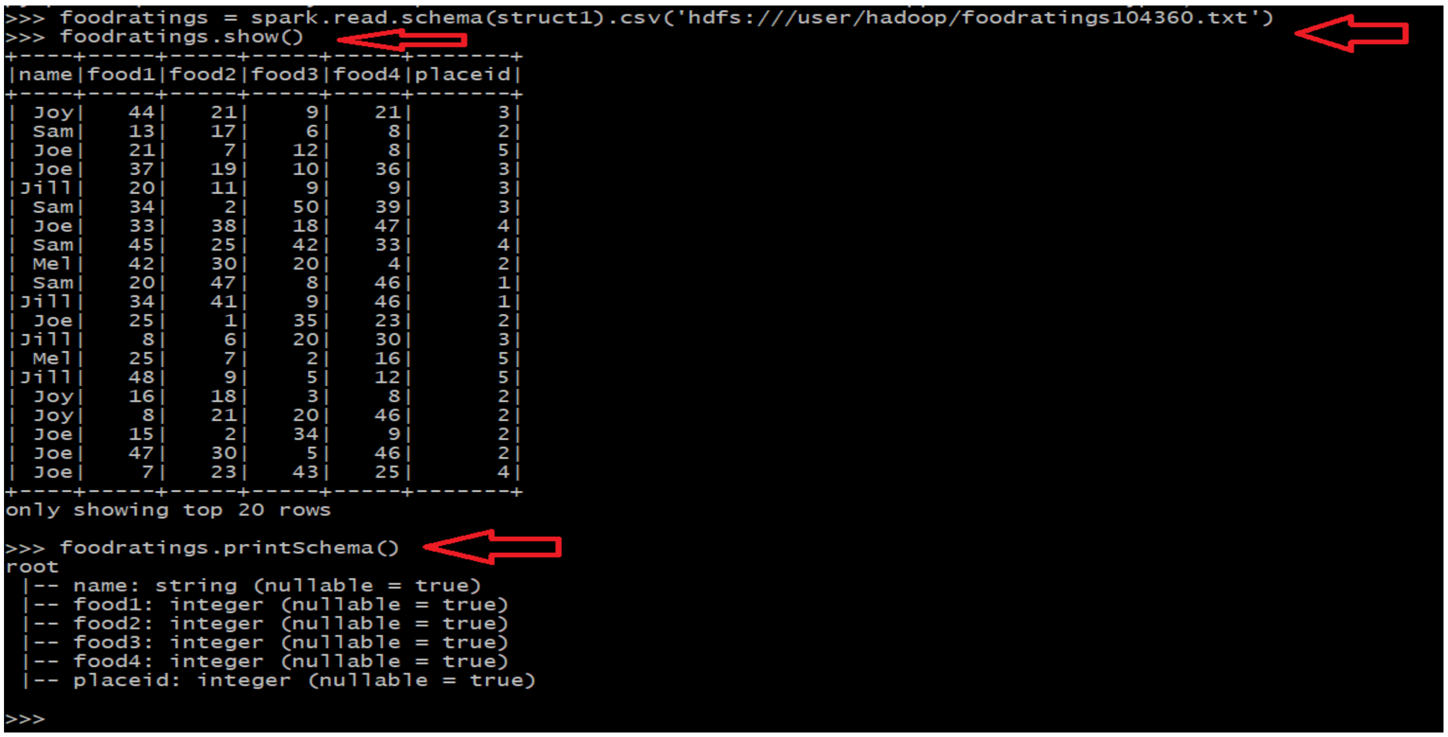
- Copy “TestDataGen.class” file to the /home/Hadoop directory
- Run Java TestDataGen command to generate magic number.
- Magic number is 104360.
- This command also generates two separate files name foodratings104360.txt and foodplaces104360.txt files in /home/Hadoop directory.
- Then I copied both the files from /home/Hadoop directory to /user/Hadoop directory.



hadoop@ip-172-31-14-124:~

```
  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-172-31-14-124 ~]$ java TestDataGen
Magic Number = 104360
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -ls /
Found 3 items
drwxrwxrwt   - hdfs hdfsadmingroup          0 2021-10-15 03:56 /tmp
drwxr-xr-x   - hdfs hdfsadmingroup          0 2021-10-15 03:55 /user
drwxr-xr-x   - hdfs hdfsadmingroup          0 2021-10-15 03:55 /var
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -ls /user/
Found 5 items
drwxrwxrwx   - hadoop   hdfsadmingroup          0 2021-10-15 03:55 /user/hadoop
drwxrwxrwx   - livy     livy                    0 2021-10-15 03:55 /user/livy
drwxrwxrwx   - root     hdfsadmingroup          0 2021-10-15 03:55 /user/root
drwxrwxrwx   - spark    spark                   0 2021-10-15 03:55 /user/spark
drwxrwxrwx   - zeppelin hdfsadmingroup          0 2021-10-15 03:55 /user/zeppeli
n
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -ls /user/hadoop/
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -ls /home/hadoop/
ls: `/home/hadoop/': No such file or directory
[hadoop@ip-172-31-14-124 ~]$ pwd
/home/hadoop
[hadoop@ip-172-31-14-124 ~]$ ls
foodplaces104360.txt  foodratings104360.txt  TestDataGen.class
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -copyFromLocal foodplaces104360.txt /user
/hadoop/
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -copyFromLocal foodratings104360.txt /use
r/hadoop/
[hadoop@ip-172-31-14-124 ~]$ hadoop fs -ls /user/hadoop/
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2021-10-15 04:24 /user/hadoop/fo
odplaces104360.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17457 2021-10-15 04:25 /user/hadoop/fo
odratings104360.txt
```

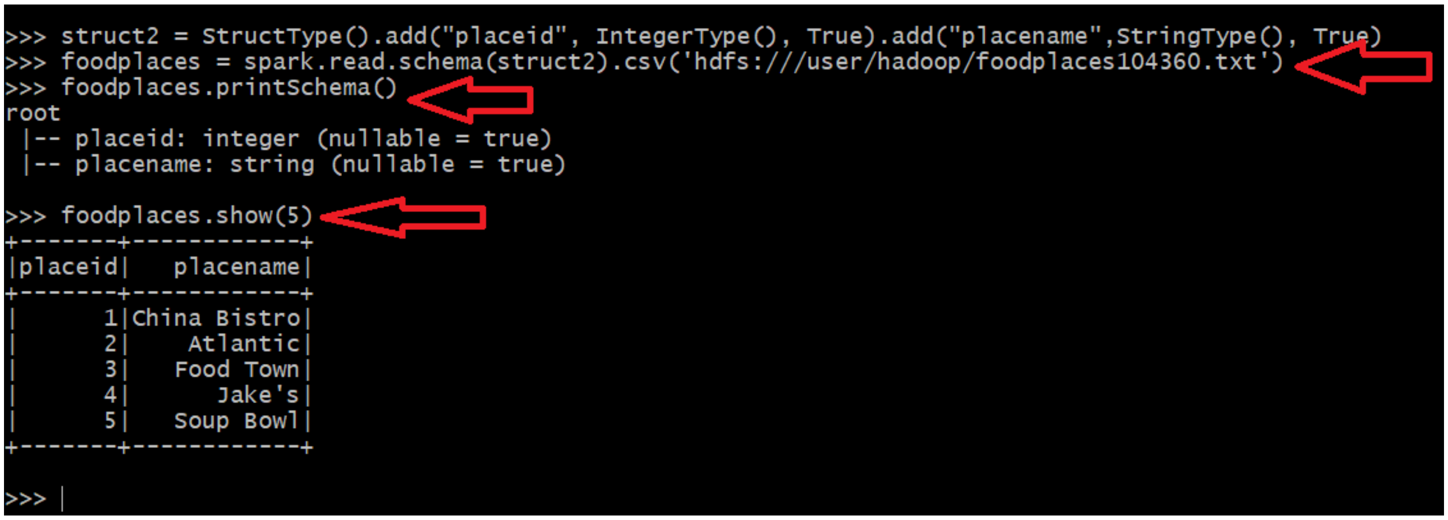
- Then I Load the ‘foodratings’ file as a ‘csv’ file into a DataFrame called foodratings.



```
>>> foodratings = spark.read.schema(struct1).csv('hdfs:///user/hadoop/foodratings104360.txt')
>>> foodratings.show()
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Joy|   44|   21|    9|   21|      3|
| Sam|   13|   17|    6|    8|      2|
| Joe|   21|    7|   12|    8|      5|
| Joe|   37|   19|   10|   36|      3|
|Jill|   20|   11|    9|    9|      3|
| Sam|   34|    2|   50|   39|      3|
| Joe|   33|   38|   18|   47|      4|
| Sam|   45|   25|   42|   33|      4|
| Mel|   42|   30|   20|    4|      2|
| Sam|   20|   47|    8|   46|      1|
|Jill|   34|   41|    9|   46|      1|
| Joe|   25|    1|   35|   23|      2|
|Jill|    8|    6|   20|   30|      3|
| Mel|   25|    7|    2|   16|      5|
|Jill|   48|    9|    5|   12|      5|
| Joy|   16|   18|    3|    8|      2|
| Joy|    8|   21|   20|   46|      2|
| Joe|   15|    2|   34|    9|      2|
| Joe|   47|   30|    5|   46|      2|
| Joe|    7|   23|   43|   25|      4|
+----+-----+-----+-----+-----+-------+
only showing top 20 rows

>>> 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)

>>>
```

Exercise: 2

- Load the ‘foodplaces’ file as a ‘csv’ file into a DataFrame called “foodplaces”

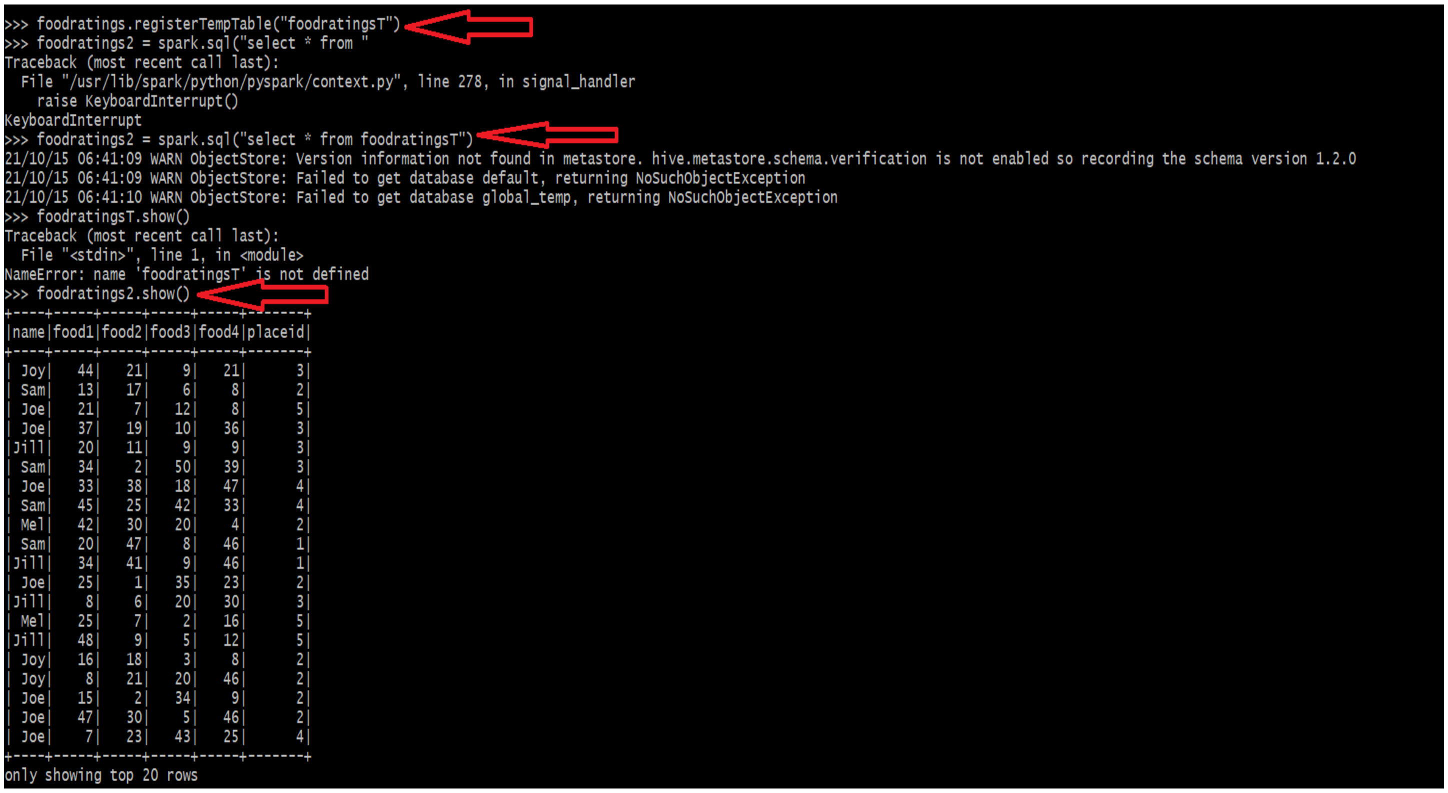


```
>>> struct2 = StructType().add("placeid", IntegerType(), True).add("placename",StringType(), True)
>>> foodplaces = spark.read.schema(struct2).csv('hdfs:///user/hadoop/foodplaces104360.txt')
>>> foodplaces.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

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

>>> |
```

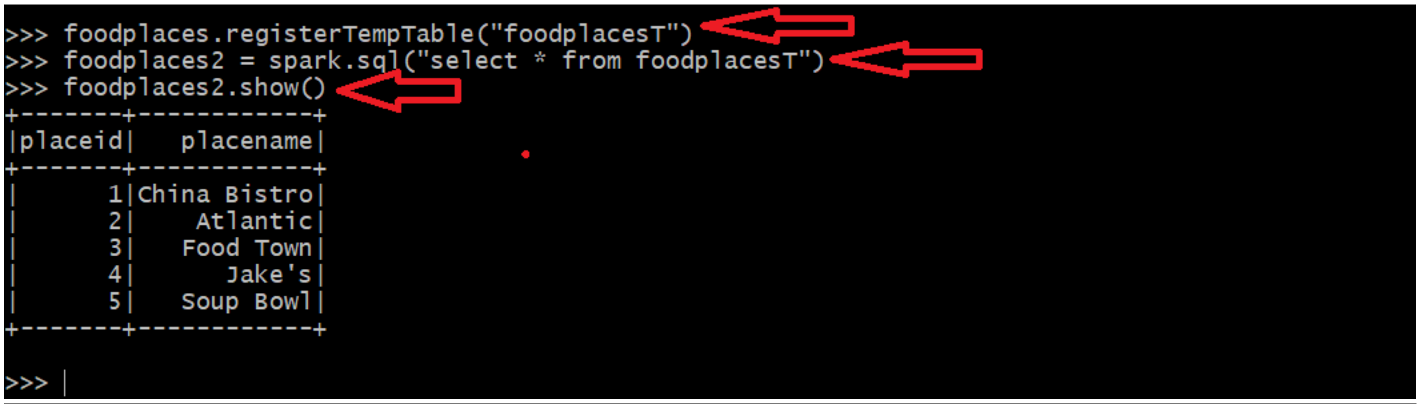
Exercise: 3

- **Step-A**

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



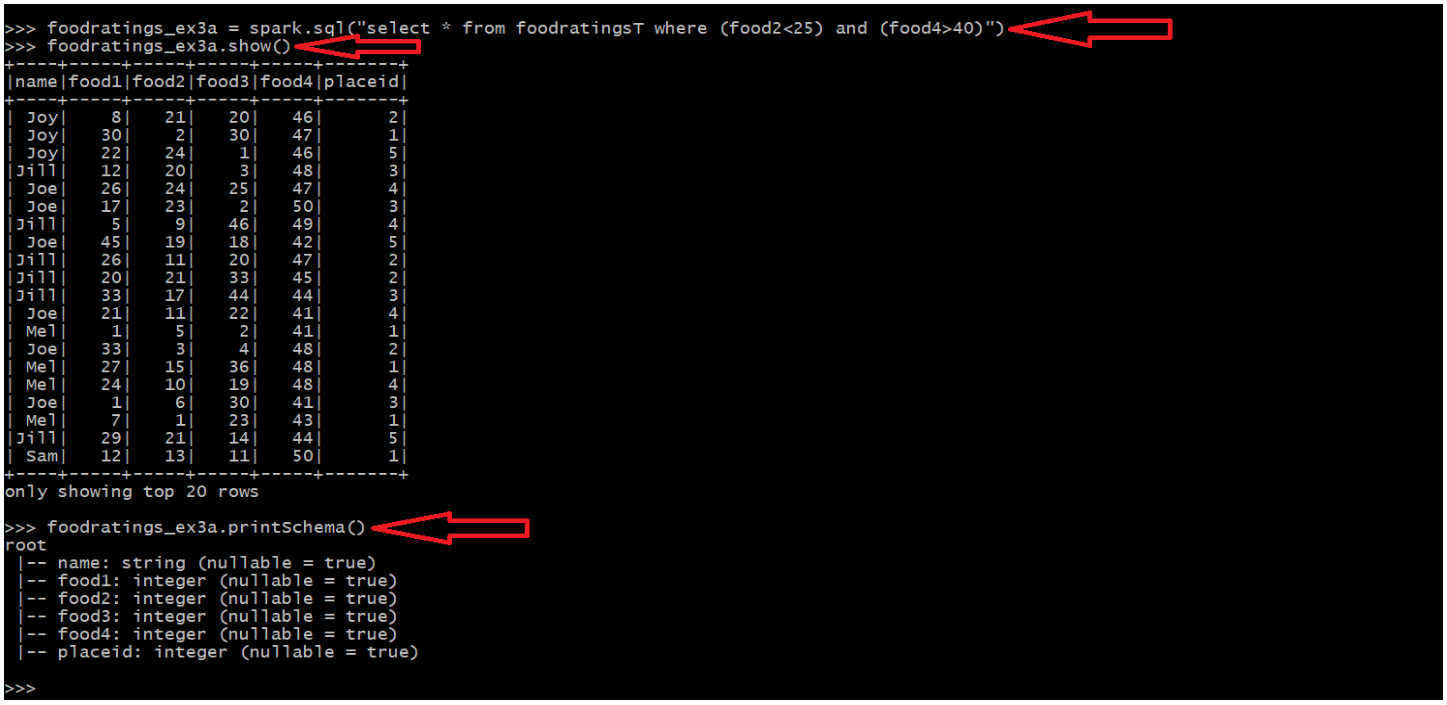
```
>>> foodratings.registerTempTable("foodratingsT")
>>> foodratings2 = spark.sql("select * from "
Traceback (most recent call last):
  File "/usr/lib/spark/python/pyspark/context.py", line 278, in signal_handler
    raise KeyboardInterrupt()
KeyboardInterrupt
>>> foodratings2 = spark.sql("select * from foodratingsT")
21/10/15 06:41:09 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
21/10/15 06:41:09 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
21/10/15 06:41:10 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
>>> foodratingsT.show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratingsT' is not defined
>>> foodratings2.show()
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Joy|   44|   21|    9|   21|      3|
| Sam|   13|   17|    6|    8|      2|
| Joe|   21|    7|   12|    8|      5|
| Joe|   37|   19|   10|   36|      3|
|Jill|   20|   11|    9|    9|      3|
| Sam|   34|    2|   50|   39|      3|
| Joe|   33|   38|   18|   47|      4|
| Sam|   45|   25|   42|   33|      4|
| Mel|   42|   30|   20|    4|      2|
| Sam|   20|   47|    8|   46|      1|
|Jill|   34|   41|    9|   46|      1|
| Joe|   25|    1|   35|   23|      2|
|Jill|    8|    6|   20|   30|      3|
| Mel|   25|    7|    2|   16|      5|
|Jill|   48|    9|    5|   12|      5|
| Joy|   16|   18|    3|    8|      2|
| Joy|    8|   21|   20|   46|      2|
| Joe|   15|    2|   34|    9|      2|
| Joe|   47|   30|    5|   46|      2|
| Joe|    7|   23|   43|   25|      4|
+----+-----+-----+-----+-----+-------+
only showing top 20 rows
```



```
>>> foodplaces.registerTempTable("foodplacesT")
>>> foodplaces2 = spark.sql("select * from foodplacesT")
>>> foodplaces2.show()
+-------+------------+
|placeid|   placename|
+-------+------------+
|      1|China Bistro|
|      2|    Atlantic|
|      3|   Food Town|
|      4|      Jake's|
|      5|   Soup Bowl|
+-------+------------+

>>> |
```

- **Step-B**

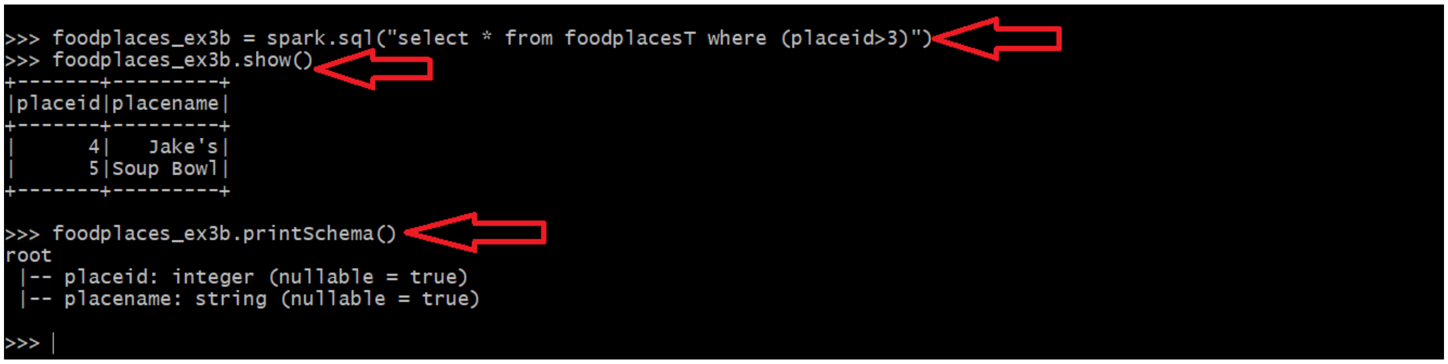
Use a SQL query on the table “foodratingsT” to create a new DataFrame called foodratings_ex3a holding records which meet the following condition: food2 < 25 and food4 > 40. Remember, when defining conditions in your code use maximum parentheses.



```
>>> foodratings_ex3a = spark.sql("select * from foodratingsT where (food2<25) and (food4>40)")
>>> foodratings_ex3a.show()
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Joy|    8|   21|   20|   46|      2|
| Joy|   30|    2|   30|   47|      1|
| Joy|   22|   24|    1|   46|      5|
|Jill|   12|   20|    3|   48|      3|
| Joe|   26|   24|   25|   47|      4|
| Joe|   17|   23|    2|   50|      3|
|Jill|    5|    9|   46|   49|      4|
| Joe|   45|   19|   18|   42|      5|
|Jill|   26|   11|   20|   47|      2|
|Jill|   20|   21|   33|   45|      2|
|Jill|   33|   17|   44|   44|      3|
| Joe|   21|   11|   22|   41|      4|
| Mel|    1|    5|    2|   41|      1|
| Joe|   33|    3|    4|   48|      2|
| Mel|   27|   15|   36|   48|      1|
| Mel|   24|   10|   19|   48|      4|
| Joe|    1|    6|   30|   41|      3|
| Mel|    7|    1|   23|   43|      1|
|Jill|   29|   21|   14|   44|      5|
| Sam|   12|   13|   11|   50|      1|
+----+-----+-----+-----+-----+-------+
only showing top 20 rows

>>> 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)

>>>
```

- **Step-C**

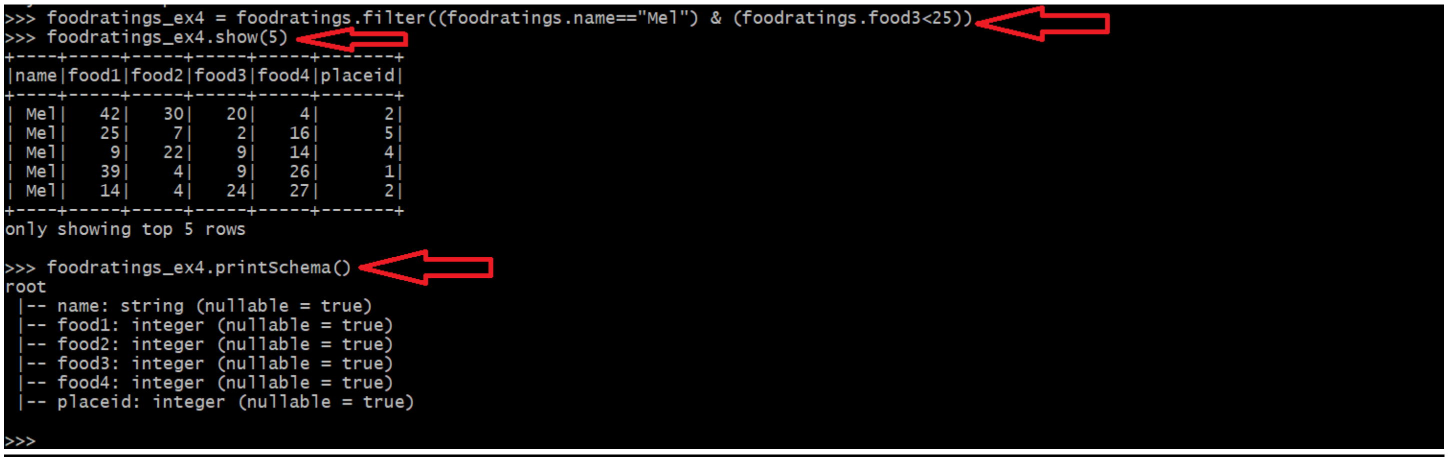
Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces_ex3b holding records which meet the following condition: placeid > 3



```
>>> foodplaces_ex3b = spark.sql("select * from foodplacesT where (placeid>3)")
>>> foodplaces_ex3b.show()
+-------+---------+
|placeid|placename|
+-------+---------+
|      4|   Jake's|
|      5|Soup Bowl|
+-------+---------+

>>> foodplaces_ex3b.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> |
```

Exercise: 4

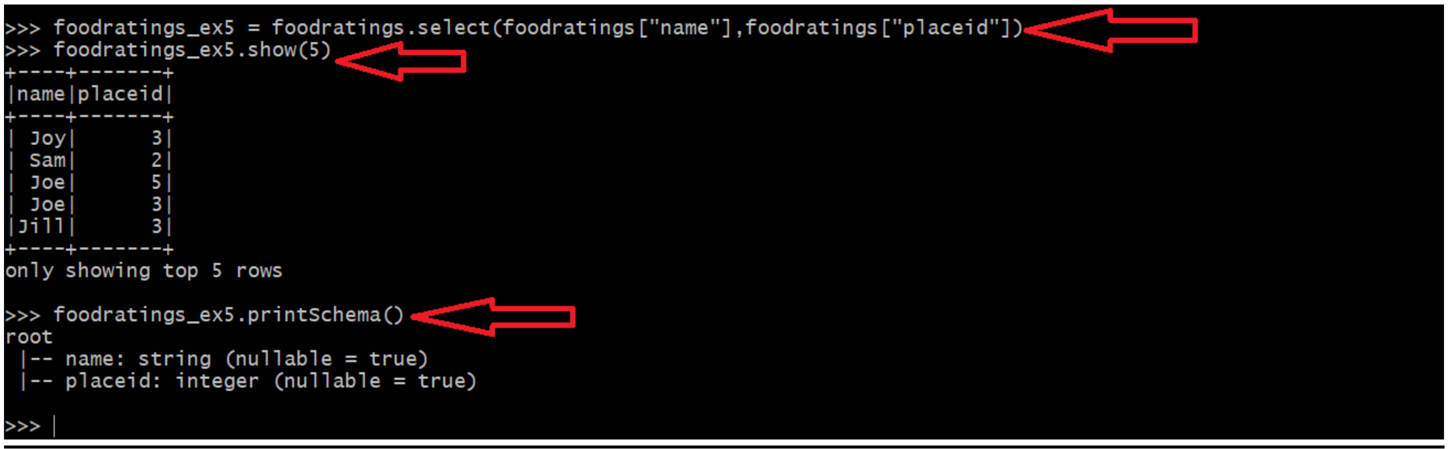
- Use a transformation (not a SparkSQL query) on the DataFrame 'foodratings' created in exercise 1 to create a new DataFrame called foodratings_ex4 that includes only those records (rows) where the 'name' field is "Mel" and food3 < 25.



```
>>> foodratings_ex4 = foodratings.filter((foodratings.name=="Mel") & (foodratings.food3<25))
>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   42|   30|   20|    4|      2|
| Mel|   25|    7|    2|   16|      5|
| Mel|    9|   22|    9|   14|      4|
| Mel|   39|    4|    9|   26|      1|
| Mel|   14|    4|   24|   27|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>> 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)

>>>
```

Exercise: 5

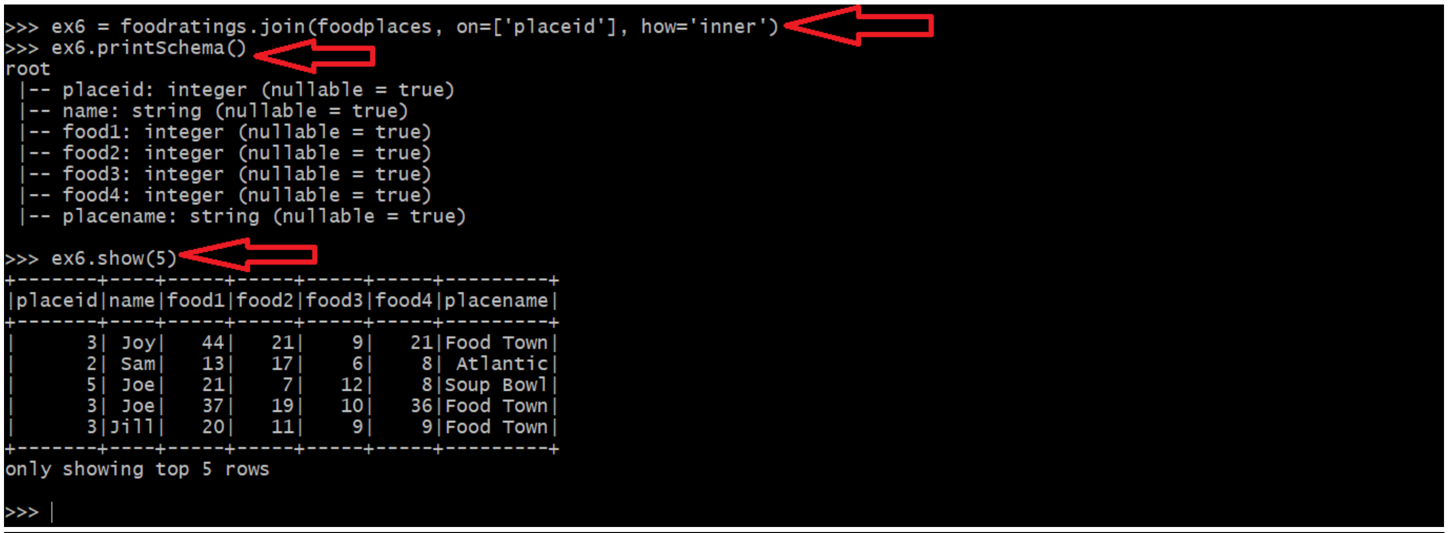
- Use a transformation (not a SparkSQL query) on the DataFrame 'foodratings' created in exercise 1 to create a new DataFrame called foodratings_ex5 that includes only the columns (fields) 'name' and 'placeid'



```
>>> foodratings_ex5 = foodratings.select(foodratings["name"],foodratings["placeid"])
>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
| Joy|      3|
| Sam|      2|
| Joe|      5|
| Joe|      3|
|Jill|      3|
+----+-------+
only showing top 5 rows

>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>>
```

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



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

>>> ex6.show(5)
+-------+----+-----+-----+-----+-----+---------+
|placeid|name|food1|food2|food3|food4|placename|
+-------+----+-----+-----+-----+-----+---------+
|      3| Joy|   44|   21|    9|   21|Food Town|
|      2| Sam|   13|   17|    6|    8| Atlantic|
|      5| Joe|   21|    7|   12|    8|Soup Bowl|
|      3| Joe|   37|   19|   10|   36|Food Town|
|      3|Jill|   20|   11|    9|    9|Food Town|
+-------+----+-----+-----+-----+-----+---------+
only showing top 5 rows

>>> |
```