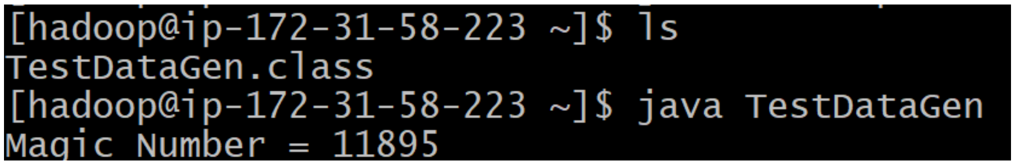
CSP 554 Big Data Technologies

Assignment #4

Student ID: A20549927

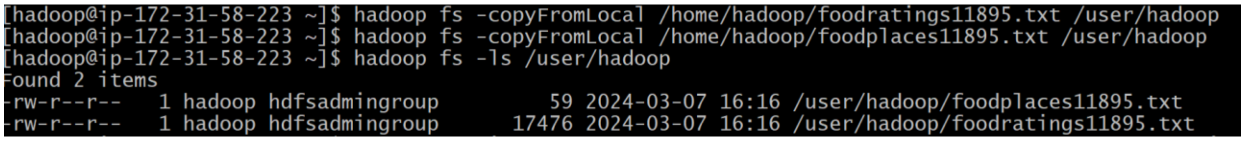
Prachi Kotadia

To execute the TestDataGen file to obtain the magic number 11895



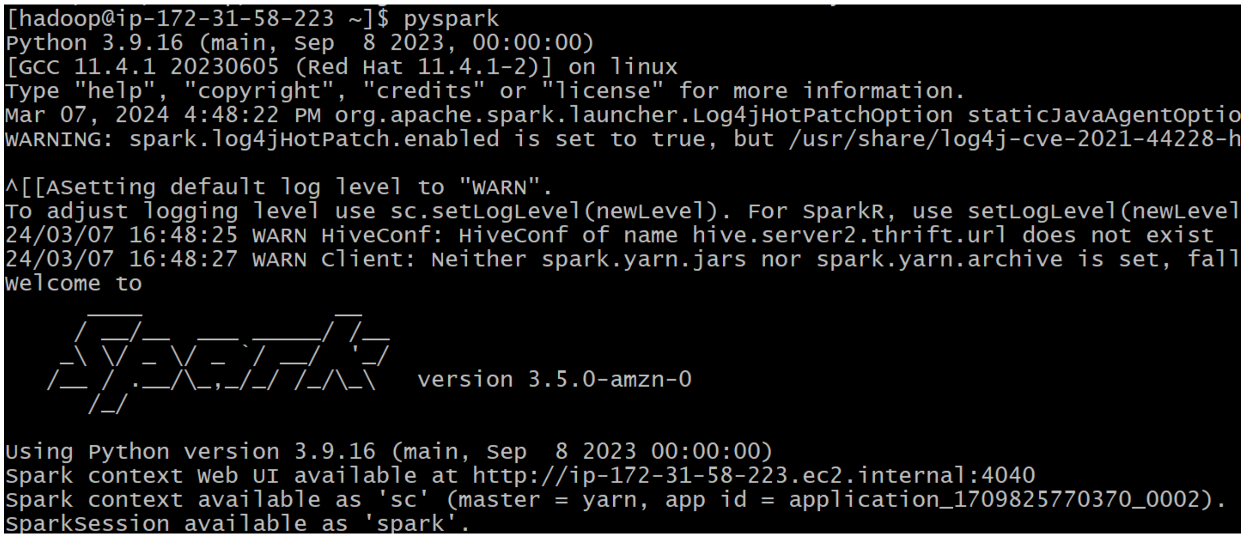
```
[hadoop@ip-172-31-58-223 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-58-223 ~]$ java TestDataGen
Magic Number = 11895
```

To copy foodratings11895.txt and foodplaces11895.txt file into /user/hadoop:



```
[hadoop@ip-172-31-58-223 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodratings11895.txt /user/hadoop
[hadoop@ip-172-31-58-223 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodplaces11895.txt /user/hadoop
[hadoop@ip-172-31-58-223 ~]$ hadoop fs -ls /user/hadoop
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2024-03-07 16:16 /user/hadoop/foodplaces11895.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17476 2024-03-07 16:16 /user/hadoop/foodratings11895.txt
```

For all exercises, load pyspark:



```
[hadoop@ip-172-31-58-223 ~]$ pyspark
Python 3.9.16 (main, Sep  8 2023, 00:00:00)
[GCC 11.4.1 20230605 (Red Hat 11.4.1-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Mar 07, 2024 4:48:22 PM org.apache.spark.launcher.Log4jHotPatchOption staticJavaAgentOptio
WARNING: spark.log4jHotPatch.enabled is set to true, but /usr/share/log4j-cve-2021-44228-h

^[[ASetting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
24/03/07 16:48:25 WARN HiveConf: HiveConf of name hive.server2.thrift.url does not exist
24/03/07 16:48:27 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, fall
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.0-amzn-0
      /_/

Using Python version 3.9.16 (main, Sep  8 2023 00:00:00)
Spark context Web UI available at http://ip-172-31-58-223.ec2.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1709825770370_0002).
SparkSession available as 'spark'.
```

<u>Exercise 1:</u>

To create the foodratings dataframe:

from pyspark.sql.types import StructType, StructField, StringType,IntegerType

foodratings_schema = 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)])

To load the foodratings11895.txt into foodratings dataframe:

foodratings =

spark.read.format("csv").schema(foodratings_schema).load("hdfs:///user/hadoop/foodratings11895.txt")

To load the foodratings11895.txt into foodratings dataframe: foodratings =

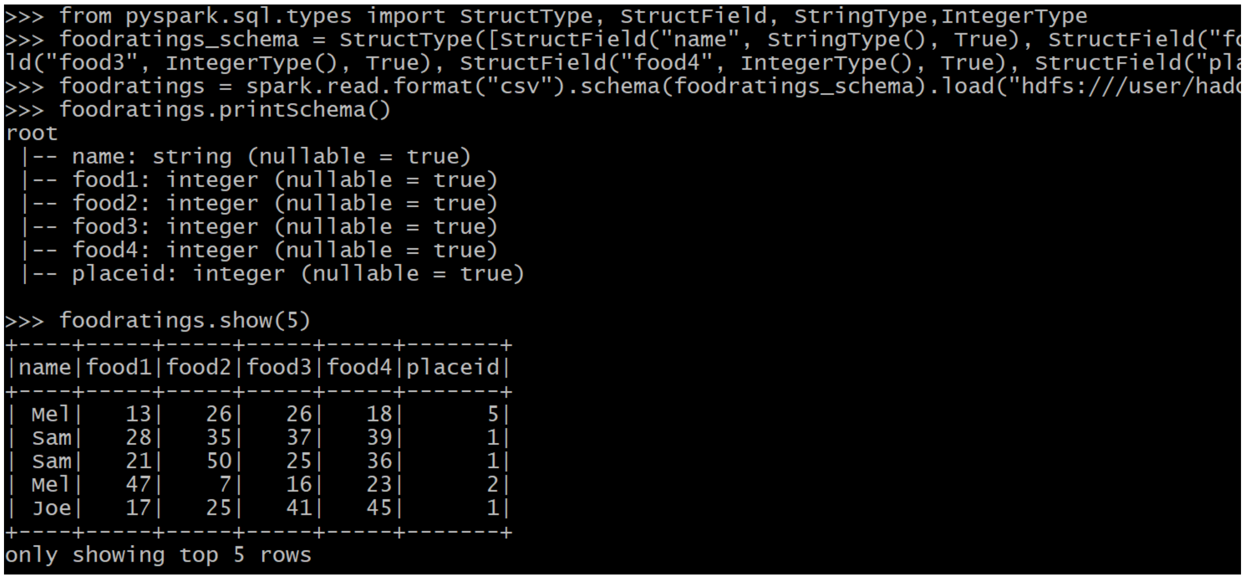
spark.read.format("csv").schema(foodratings_schema).load("hdfs:///user/hadoop/foodratings11895.txt")

To print the schema from foodratings dataframe:

foodratings.printSchema()

To print the top 5 rows from foodratings dataframe:

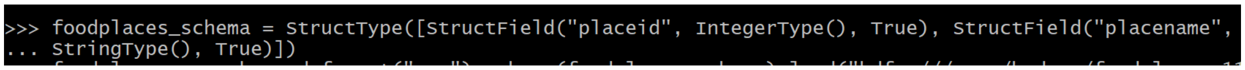
foodratings.show(5)



```
>>> from pyspark.sql.types import StructType, StructField, StringType,IntegerType
>>> foodratings_schema = StructType([StructField("name", StringType(), True), StructField("fo
ld("food3", IntegerType(), True), StructField("food4", IntegerType(), True), StructField("pla
>>> foodratings = spark.read.format("csv").schema(foodratings_schema).load("hdfs:///user/hado
>>> 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|
+----+-----+-----+-----+-----+-------+
| Mel|   13|   26|   26|   18|      5|
| Sam|   28|   35|   37|   39|      1|
| Sam|   21|   50|   25|   36|      1|
| Mel|   47|    7|   16|   23|      2|
| Joe|   17|   25|   41|   45|      1|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Exercise 2:

To create foodplaces dataframe: foodplaces_schema = StructType([StructField("placeid", IntegerType(), True), StructField("placename", StringType(), True)])



```
>>> foodplaces_schema = StructType([StructField("placeid", IntegerType(), True), StructField("placename",
... StringType(), True)])
```

To load foodplaces11895.txt into foodplaces dataframe:

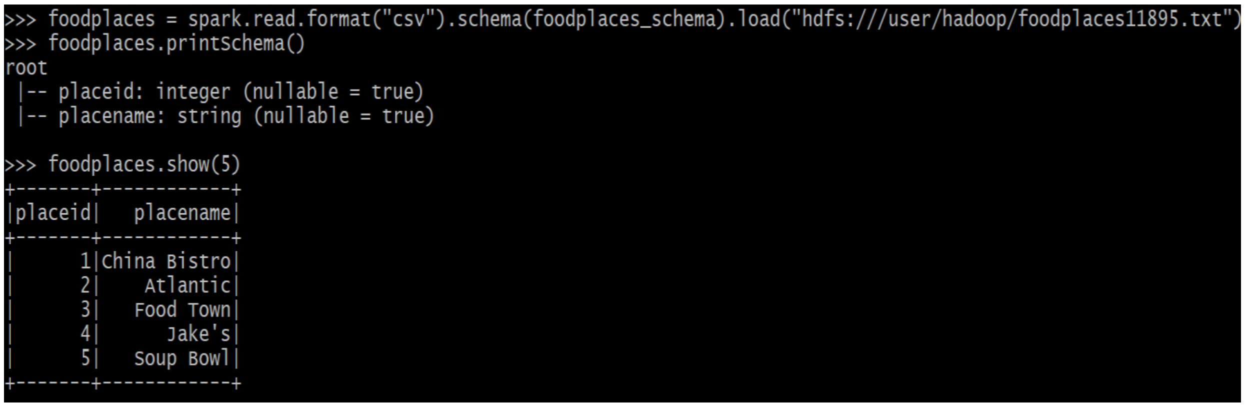
foodplaces = park.read.format("csv").schema(foodplaces_schema).load("hdfs:///user/hadoop/foodplaces11895.txt")

To print the foodplaces dataframe schema:

foodplaces.printSchema()

To print the top 5 rows from foodplaces dataframe:

foodplaces.show(5)



```
>>> foodplaces = spark.read.format("csv").schema(foodplaces_schema).load("hdfs:///user/hadoop/foodplaces11895.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|
+-------+------------+
```

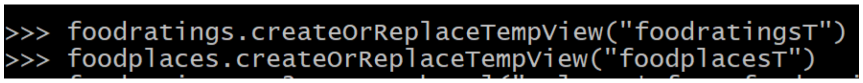
Exercise 3:

Step A:

To register the dataframes from Exercise 1 and Exercise called “foodratingsT” and “foodplacesT”:

foodratings.createOrReplaceTempView("foodratingsT")

foodplaces.createOrReplaceTempView("foodplacesT")



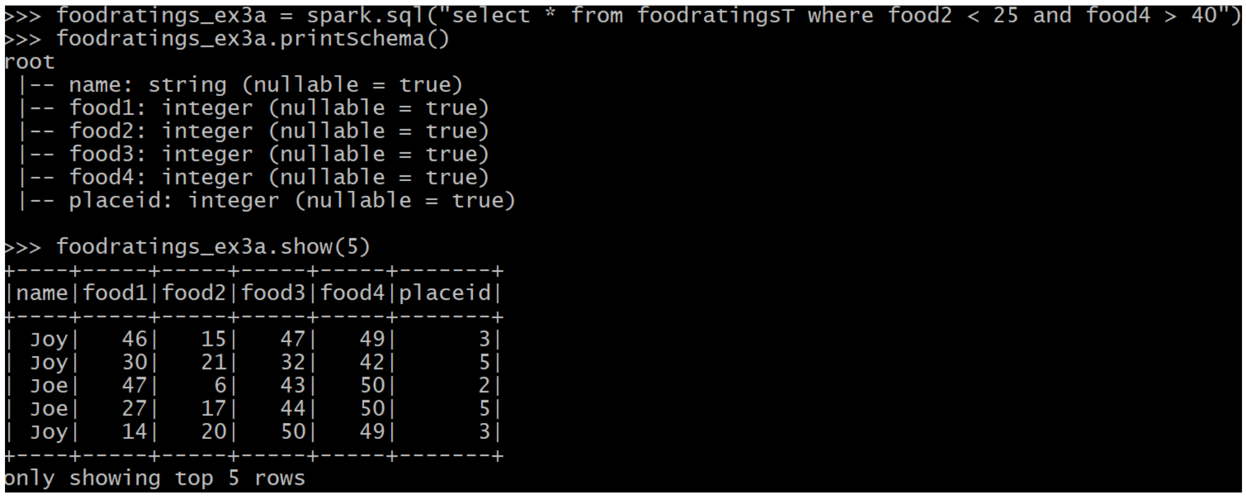
```
>>> foodratings.createOrReplaceTempView("foodratingsT")
>>> foodplaces.createOrReplaceTempView("foodplacesT")
```

Step B:

To create foodratings_ex3a dataframe to print the schema and print the top 5 rows from that dataframe:

foodratings_ex3a = spark.sql("select * from foodratingsT where food2 < 25 and food4 > 40")

foodratings_ex3a.printSchema() foodratings_ex3a.show(5)



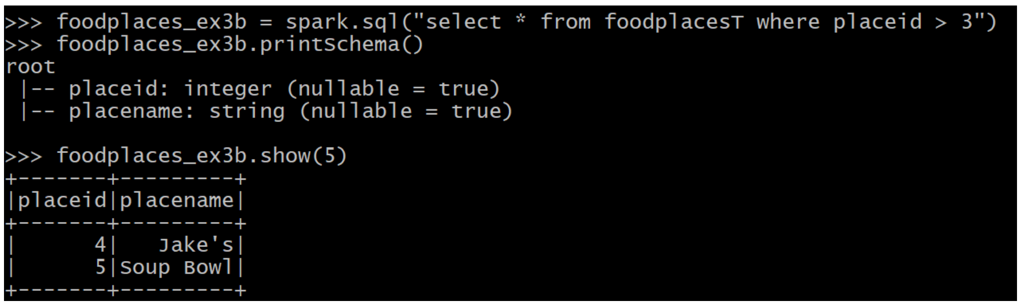
```
>>> foodratings_ex3a = spark.sql("select * from foodratingsT where food2 < 25 and food4 > 40")
>>> foodratings_ex3a.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex3a.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Joy|   46|   15|   47|   49|      3|
| Joy|   30|   21|   32|   42|      5|
| Joe|   47|    6|   43|   50|      2|
| Joe|   27|   17|   44|   50|      5|
| Joy|   14|   20|   50|   49|      3|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

Step C:

To create foodratings_ex3b dataframe to print the schema and print the top 5 rows from that dataframe:

foodplaces_ex3b = spark.sql("select * from foodplacesT where placeid > 3")

foodplaces_ex3b.printSchema() foodplaces_ex3b.show(5)



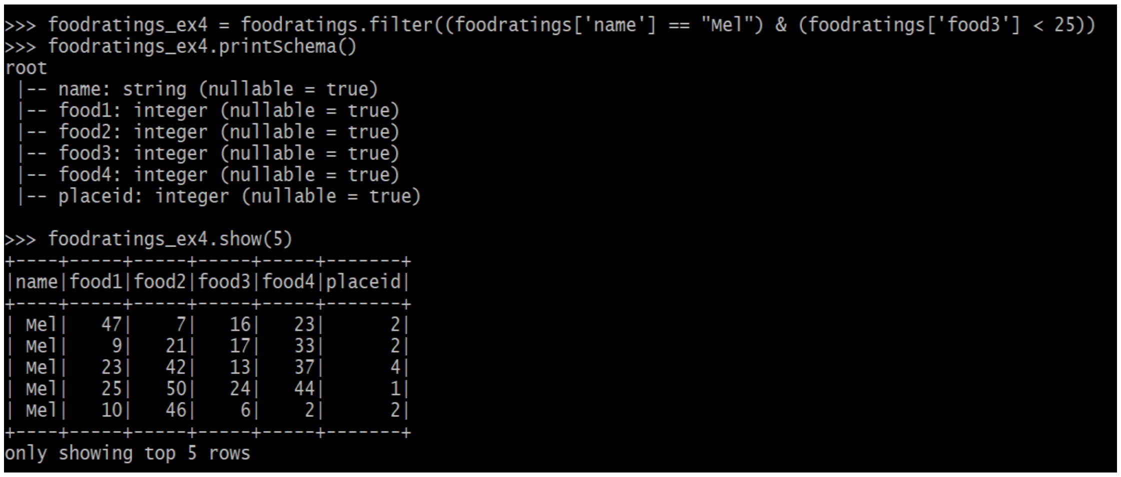
```
>>> foodplaces_ex3b = spark.sql("select * from foodplacesT where placeid > 3")
>>> foodplaces_ex3b.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces_ex3b.show(5)
+-------+---------+
|placeid|placename|
+-------+---------+
|      4|   Jake's|
|      5|Soup Bowl|
+-------+---------+
```

<u>Exercise 4:</u>

To create foodratings_ex4 dataframe using transformation to print the schema and print the top 5 rows from that dataframe:

foodratings_ex4 = foodratings.filter((foodratings['name'] == "Mel") & (foodratings['food3'] < 25))

foodratings_ex4.printSchema() foodratings_ex4.show(5)



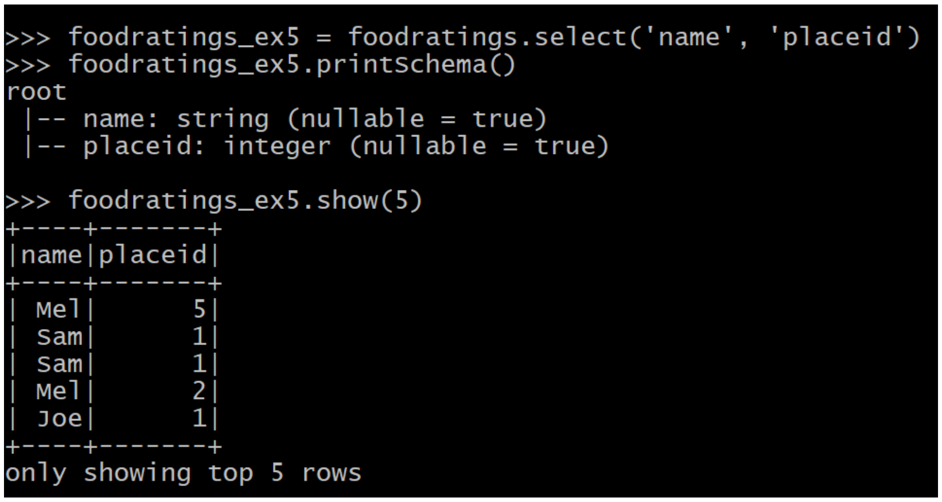
```
>>> foodratings_ex4 = foodratings.filter((foodratings['name'] == "Mel") & (foodratings['food3'] < 25))
>>> foodratings_ex4.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   47|    7|   16|   23|      2|
| Mel|    9|   21|   17|   33|      2|
| Mel|   23|   42|   13|   37|      4|
| Mel|   25|   50|   24|   44|      1|
| Mel|   10|   46|    6|    2|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

<u>Exercise 5:</u>

To create foodratings_ex5 dataframe using transformation to print the schema and print the top 5 rows from that dataframe:

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

foodratings_ex5.printSchema() foodratings_ex5.show(5)



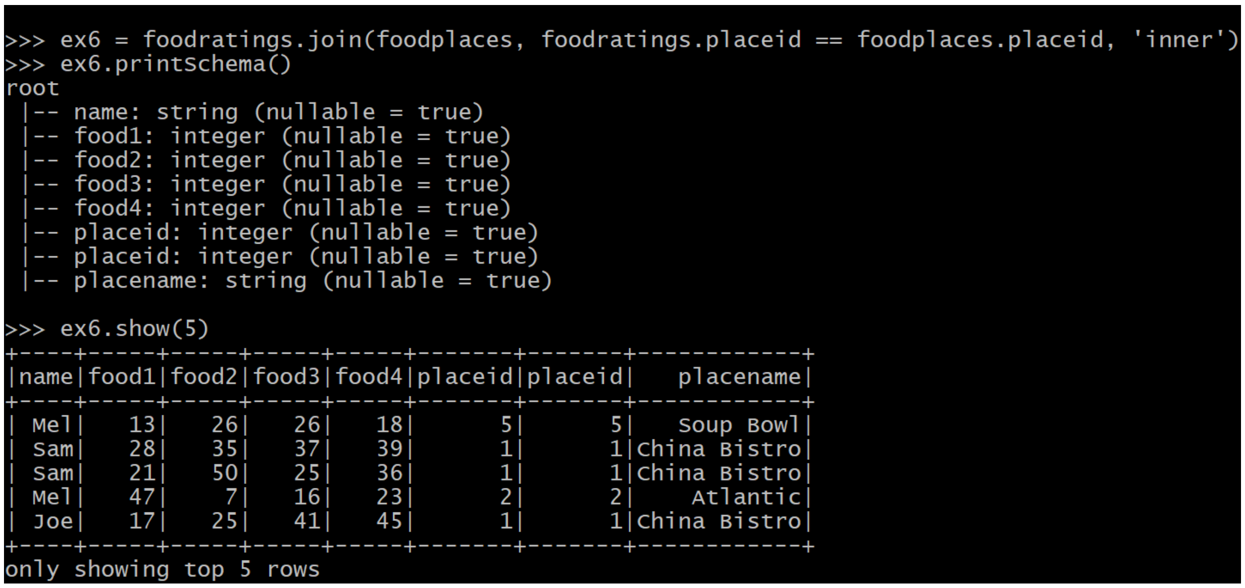
```
>>> foodratings_ex5 = foodratings.select('name', 'placeid')
>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
| Mel|      5|
| Sam|      1|
| Sam|      1|
| Mel|      2|
| Joe|      1|
+----+-------+
only showing top 5 rows
```

<u>Exercise 6:</u>

To create ex6 dataframe using transformation to print the schema and print the top 5 rows from that dataframe:

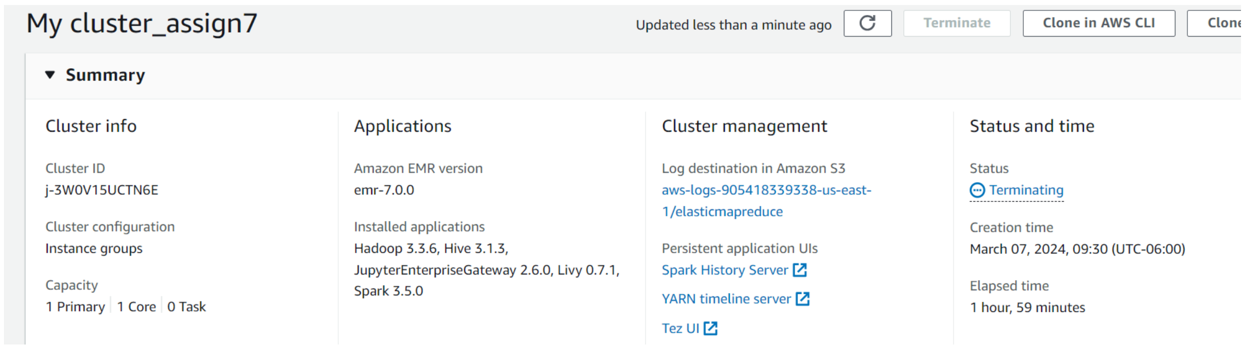
ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid, 'inner')

ex6.printSchema() ex6.show(5)



```
>>> 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|
+----+-----+-----+-----+-----+-------+-------+------------+
| Mel|   13|   26|   26|   18|      5|      5|   Soup Bowl|
| Sam|   28|   35|   37|   39|      1|      1|China Bistro|
| Sam|   21|   50|   25|   36|      1|      1|China Bistro|
| Mel|   47|    7|   16|   23|      2|      2|    Atlantic|
| Joe|   17|   25|   41|   45|      1|      1|China Bistro|
+----+-----+-----+-----+-----+-------+-------+------------+
only showing top 5 rows
```

Terminate the Cluster



My cluster_assign7

Updated less than a minute ago

▼ Summary

Cluster info	Applications	Cluster management	Status and time
Cluster ID j-3W0V15UCTN6E	Amazon EMR version emr-7.0.0	Log destination in Amazon S3 aws-logs-905418339338-us-east-1/elasticmapreduce	Status Terminating
Cluster configuration Instance groups	Installed applications Hadoop 3.3.6, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.7.1, Spark 3.5.0	Persistent application UIs Spark History Server YARN timeline server Tez UI	Creation time March 07, 2024, 09:30 (UTC-06:00)
Capacity 1 Primary | 1 Core | 0 Task			Elapsed time 1 hour, 59 minutes