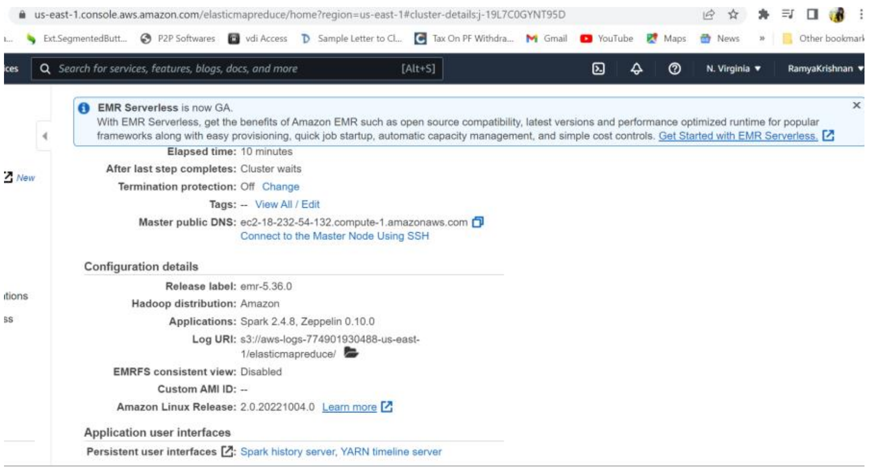
Assignment #07

Exercise 1)

Step A

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



us-east-1.console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-details:j-19L7C0GYNT95D

EMR Serverless is now GA.
With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. Get Started with EMR Serverless.

Elapsed time: 10 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-18-232-54-132.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

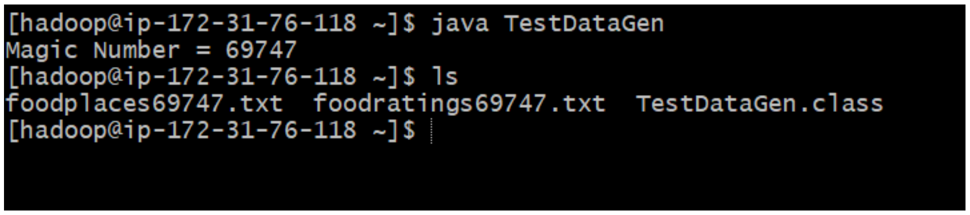
Release label: emr-5.36.0
Hadoop distribution: Amazon
Applications: Spark 2.4.8, Zeppelin 0.10.0
Log URI: s3://aws-logs-774901930488-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --
Amazon Linux Release: 2.0.20221004.0 Learn more

Application user interfaces

Persistent user interfaces: Spark history server, YARN timeline server

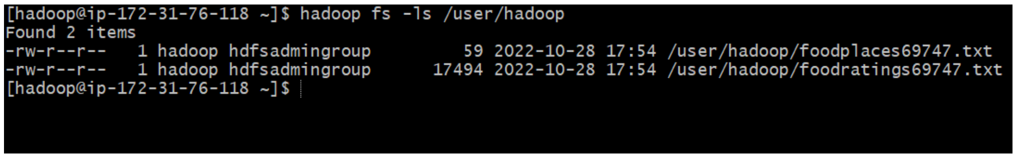
Step B

Use the TestDataGen program from previous assignments to generate new data files.



```
[hadoop@ip-172-31-76-118 ~]$ java TestDataGen
Magic Number = 69747
[hadoop@ip-172-31-76-118 ~]$ ls
foodplaces69747.txt  foodratings69747.txt  TestDataGen.class
[hadoop@ip-172-31-76-118 ~]$
```

Copy both generated files to the HDFS directory "/user/hadoop"



```
[hadoop@ip-172-31-76-118 ~]$ hadoop fs -ls /user/hadoop
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2022-10-28 17:54 /user/hadoop/foodplaces69747.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17494 2022-10-28 17:54 /user/hadoop/foodratings69747.txt
[hadoop@ip-172-31-76-118 ~]$
```

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:

Field Name	Field Type
name	String
food1	Integer
food2	Integer
food3	Integer
food4	Integer
placeid	Integer

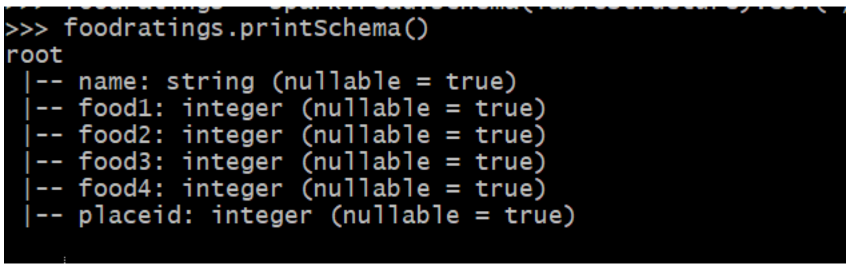
As the results of this exercise provide the magic number, *the code you execute* and screen shots of thefollowing commands:

*from pyspark.sql.types import **

TableStructure = 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)

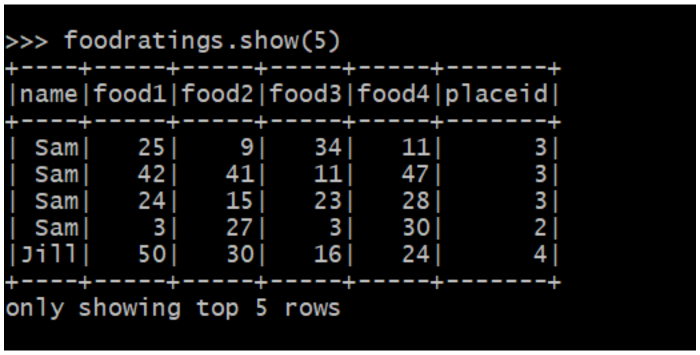
foodratings = spark.read.schema(TableStructure).csv('/user/hadoop/foodratings69747.txt')

foodratings.printSchema()



```
>>> 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)



```
>>> foodratings.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Sam|   25|    9|   34|   11|      3|
| Sam|   42|   41|   11|   47|      3|
| Sam|   24|   15|   23|   28|      3|
| Sam|    3|   27|    3|   30|      2|
|Jill|   50|   30|   16|   24|      4|
+----+-----+-----+-----+-----+-------+
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:

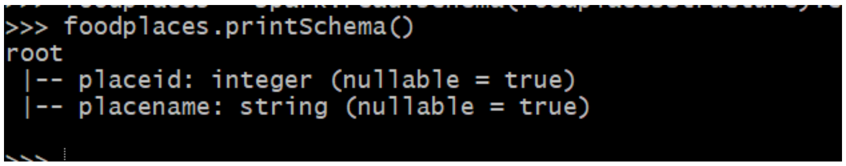
Field Nampee	Field Type
placeid	Integer
placename	String

As the results of this exercise provide *the code you execute* and screen shots of the followingcommands:

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

foodplaces = spark.read.schema(foodplacesStructure).csv('/user/hadoop/foodplaces69747.txt')

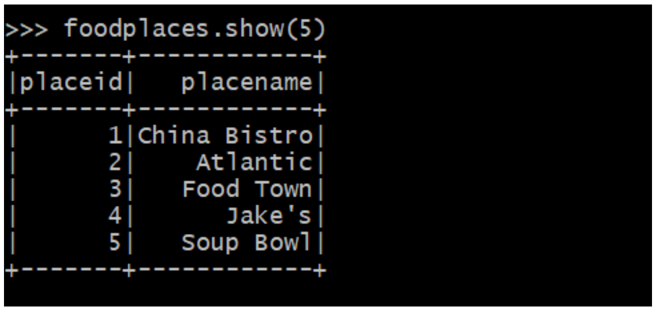
foodplaces.printSchema()



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

>>>
```

foodplaces.show(5)



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

Exercise 3)

<u>Step A</u>

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

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

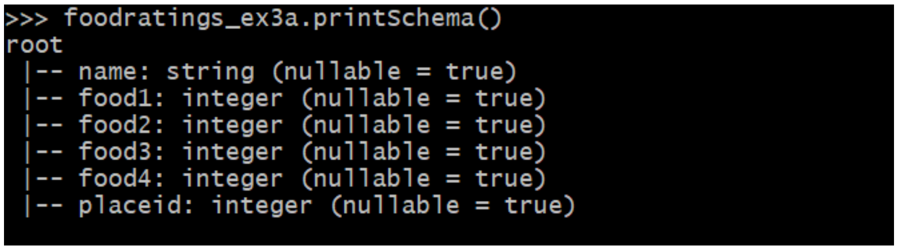
Step B

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:

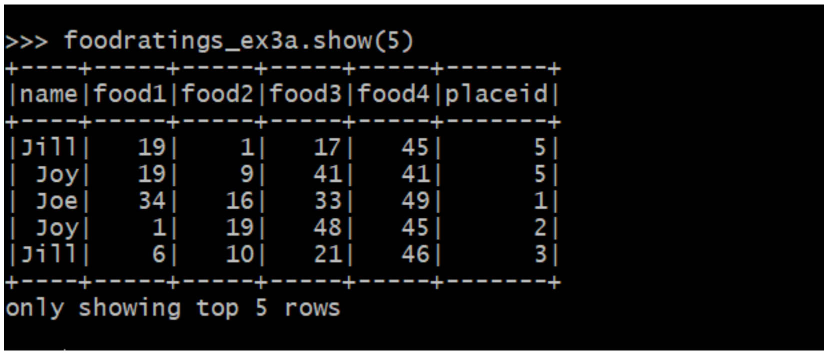
*foodratings_ex3a = spark.sql("SELECT * from foodratingsT WHERE food2 < 25 AND food4 > 40")*

foodratings_ex3a.printSchema()



```
>>> 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)



```
>>> foodratings_ex3a.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
|Jill|   19|    1|   17|   45|      5|
| Joy|   19|    9|   41|   41|      5|
| Joe|   34|   16|   33|   49|      1|
| Joy|    1|   19|   48|   45|      2|
|Jill|    6|   10|   21|   46|      3|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

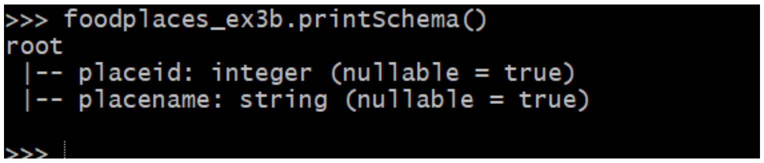
Step C

Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces_ex3b holdingrecords 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 = spark.sql("SELECT * from foodplacesT WHERE placeid>3")*

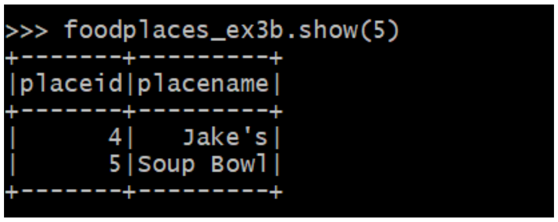
foodplaces_ex3b.printSchema()



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

>>>
```

foodplaces_ex3b.show(5)



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

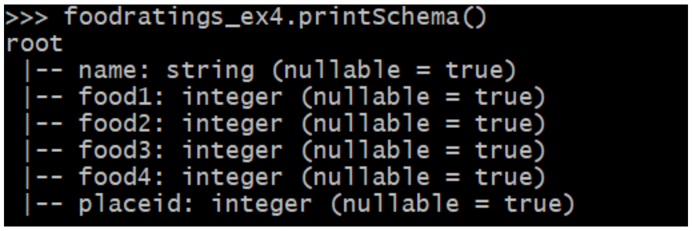
Exercise 4)

Use a transformation (not a SparkSQL query) on the DataFrame 'foodratings' created in exercise 1 tocreate 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:

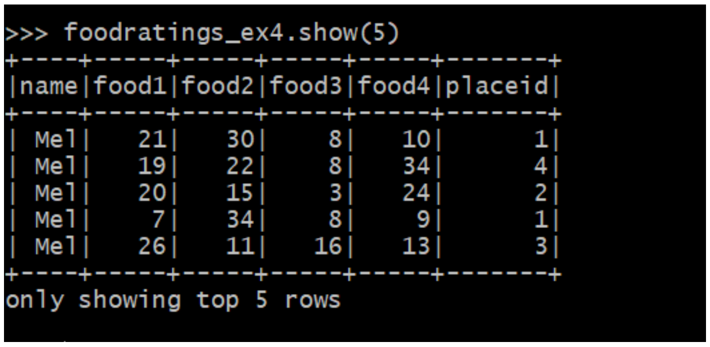
foodratings_ex4 = foodratings.filter(foodratings.name == "Mel").filter(foodratings.food3 < 25)

foodratings_ex4.printSchema()



```
>>> 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)


```
>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   21|   30|    8|   10|      1|
| Mel|   19|   22|    8|   34|      4|
| Mel|   20|   15|    3|   24|      2|
| Mel|    7|   34|    8|    9|      1|
| Mel|   26|   11|   16|   13|      3|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows
```

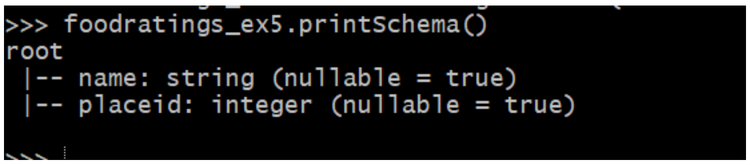
Exercise 5)

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

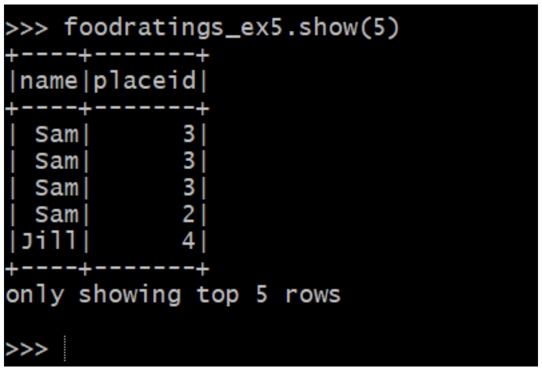
As the results of this step provide the code you execute and screen shots of the following commands:

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

foodratings_ex5.printSchema()


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

foodratings_ex5.show(5)


```
>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
| Sam|      3|
| Sam|      3|
| Sam|      3|
| Sam|      2|
|Jill|      4|
+----+-------+
only showing top 5 rows

>>>
```

Exercise 6)

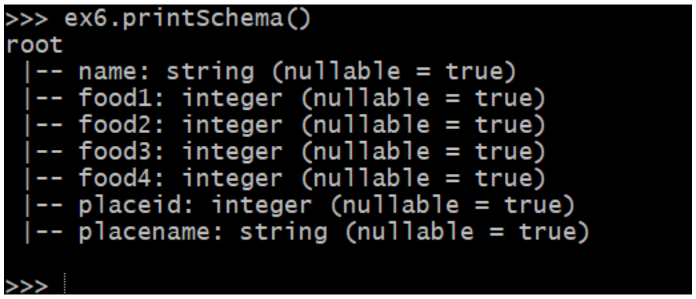
Use a transformation (**not a SparkSQL query**) to create a new DataFrame called ex6 which is the innerjoin, on placeid, of the DataFrames ‘foodratings’ and ‘foodplaces’ created in exercises 1 and 2

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

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

"inner").drop(foodratings.placeid)

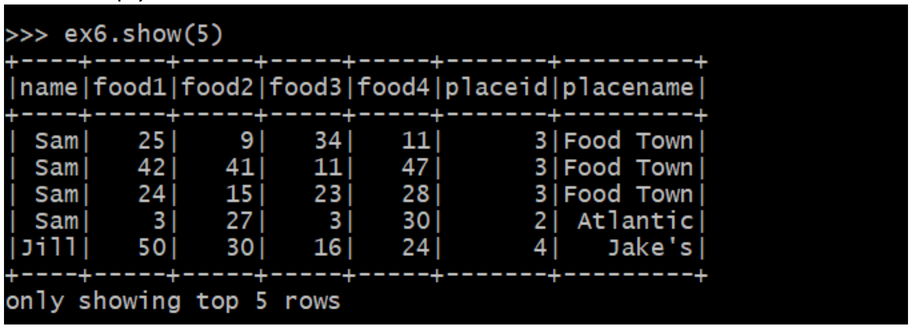
ex6.printSchema()



```
>>> 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)



```
>>> ex6.show(5)
+----+-----+-----+-----+-----+-------+---------+
|name|food1|food2|food3|food4|placeid|placename|
+----+-----+-----+-----+-----+-------+---------+
| Sam|   25|    9|   34|   11|      3|Food Town|
| Sam|   42|   41|   11|   47|      3|Food Town|
| Sam|   24|   15|   23|   28|      3|Food Town|
| Sam|    3|   27|    3|   30|      2| Atlantic|
|Jill|   50|   30|   16|   24|      4|   Jake's|
+----+-----+-----+-----+-----+-------+---------+
only showing top 5 rows
```