Name: Deep Pawar(A20545137)
Professor: Joseph Rosen
Institute: Illinois Institute of Technology

CSP 554: Big Data Technologies

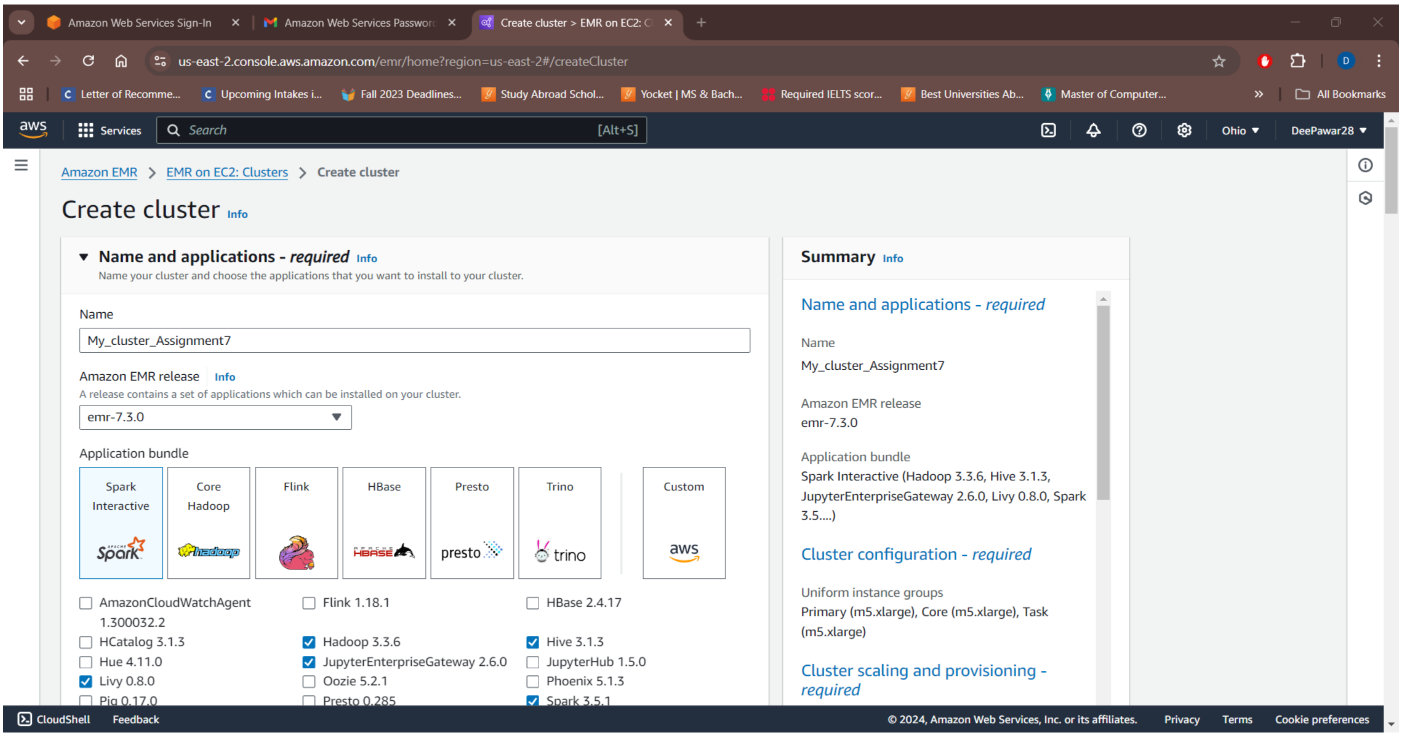
Fall 2024 - Assignment 7

- **Questions and Answers:**

1. Exercise 1)

Step A: Start up an EMR cluster as previously, but instead of choosing the "Core Hadoop" configuration chose the "Spark" configuration (see below), otherwise proceed as before.

Ans:



us-east-2.console.aws.amazon.com/emr/home?region=us-east-2#/createCluster

Amazon EMR > EMR on EC2: Clusters > Create cluster

Create cluster Info

Name and applications - *required* Info

Name your cluster and choose the applications that you want to install to your cluster.

Name
My_cluster_Assignment7

Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.
emr-7.3.0

Application bundle
Spark Interactive | Core Hadoop | Flink | HBase | Presto | Trino | Custom

☐ AmazonCloudWatchAgent 1.300032.2
☐ HCatalog 3.1.3
☐ Hue 4.11.0
☑ Livy 0.8.0
☐ Pig 0.17.0
☐ Flink 1.18.1
☑ Hadoop 3.3.6
☑ JupyterEnterpriseGateway 2.6.0
☐ Oozie 5.2.1
☐ Presto 0.285
☐ HBase 2.4.17
☑ Hive 3.1.3
☐ JupyterHub 1.5.0
☐ Phoenix 5.1.3
☑ Spark 3.5.1

Summary Info

Name and applications - *required*

Name
My_cluster_Assignment7

Amazon EMR release
emr-7.3.0

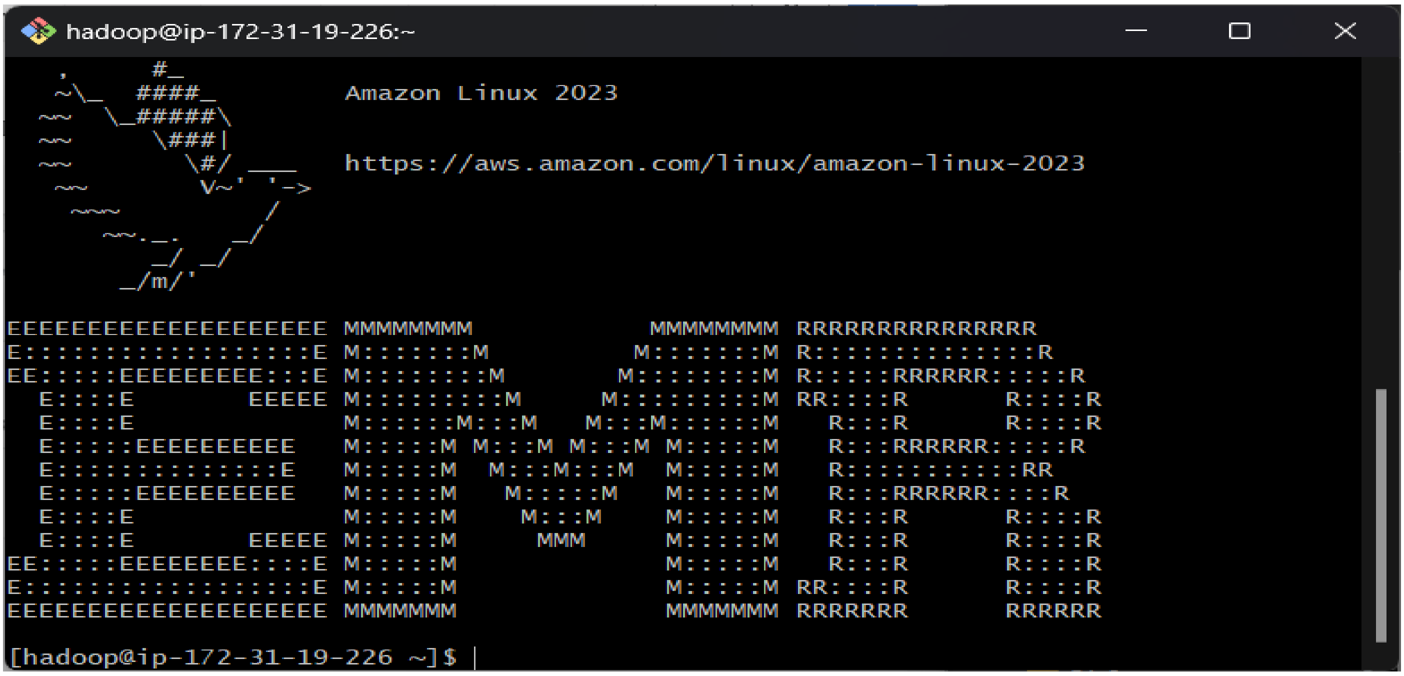
Application bundle
Spark Interactive (Hadoop 3.3.6, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.8.0, Spark 3.5....)

Cluster configuration - *required*

Uniform instance groups
Primary (m5.xlarge), Core (m5.xlarge), Task (m5.xlarge)

Cluster scaling and provisioning - *required*

© 2024, Amazon Web Services, Inc. or its affiliates.



hadoop@ip-172-31-19-226:~

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

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-172-31-19-226 ~]$
```

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

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

Ans:

Magic Number: 232556

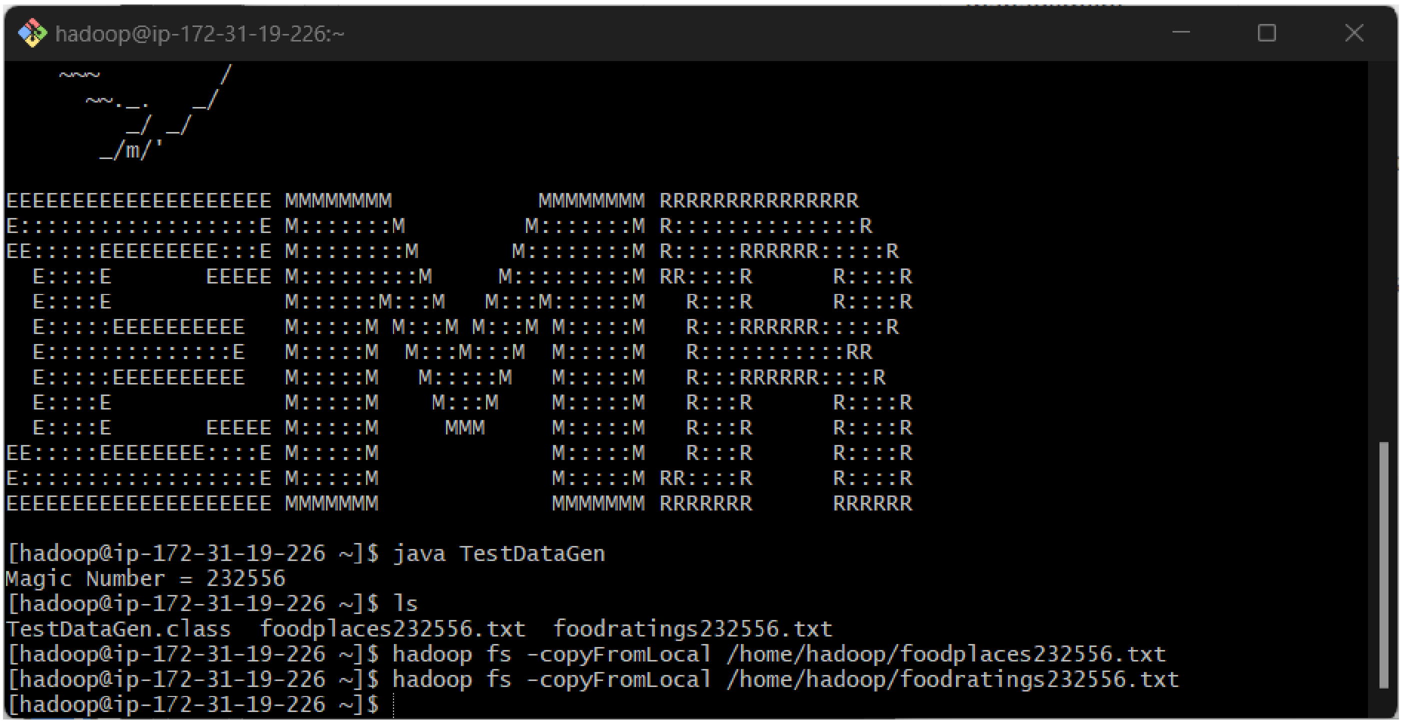
- **Command Used:**

java testDataGen

ls

hadoop fs -copyFromLocal /home/hadoop/foodplaces232556.txt

hadoop fs -copyFromLocal /home/hadoop/foodratings232556.txt



hadoop@ip-172-31-19-226:~

```
    ~~~          /
      ~~._.   _/
          _/ _/
        _/m/'

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-172-31-19-226 ~]$ java TestDataGen
Magic Number = 232556
[hadoop@ip-172-31-19-226 ~]$ ls
TestDataGen.class  foodplaces232556.txt  foodratings232556.txt
[hadoop@ip-172-31-19-226 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodplaces232556.txt
[hadoop@ip-172-31-19-226 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodratings232556.txt
[hadoop@ip-172-31-19-226 ~]$
```

Step C: Load the 'foodratings' file as a 'csv' file into a DataFrame called foodratings.

Ans:

- **Command Used:**

pyspark

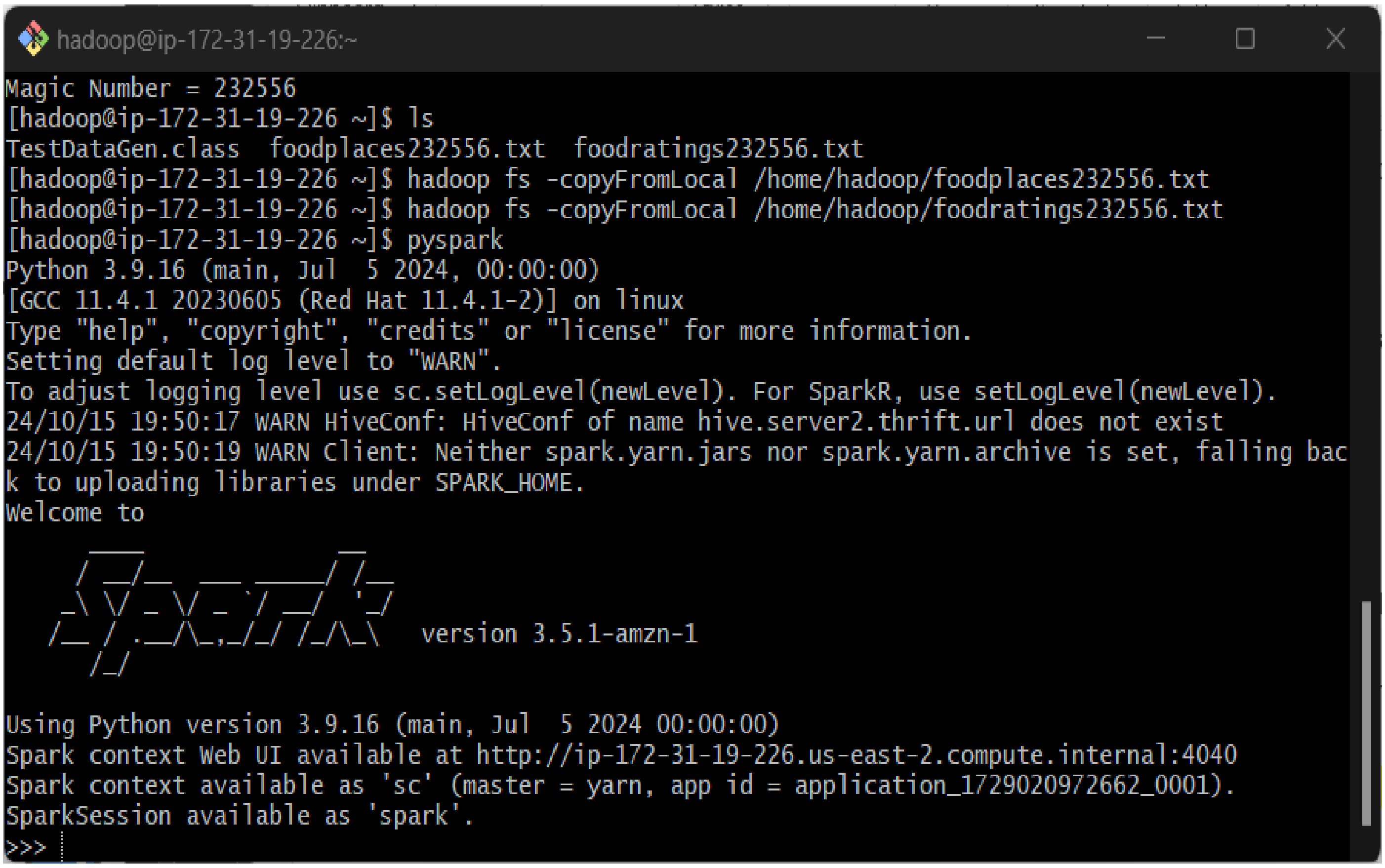
from pyspark.sql.types import *

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(' foodratings232556.txt')

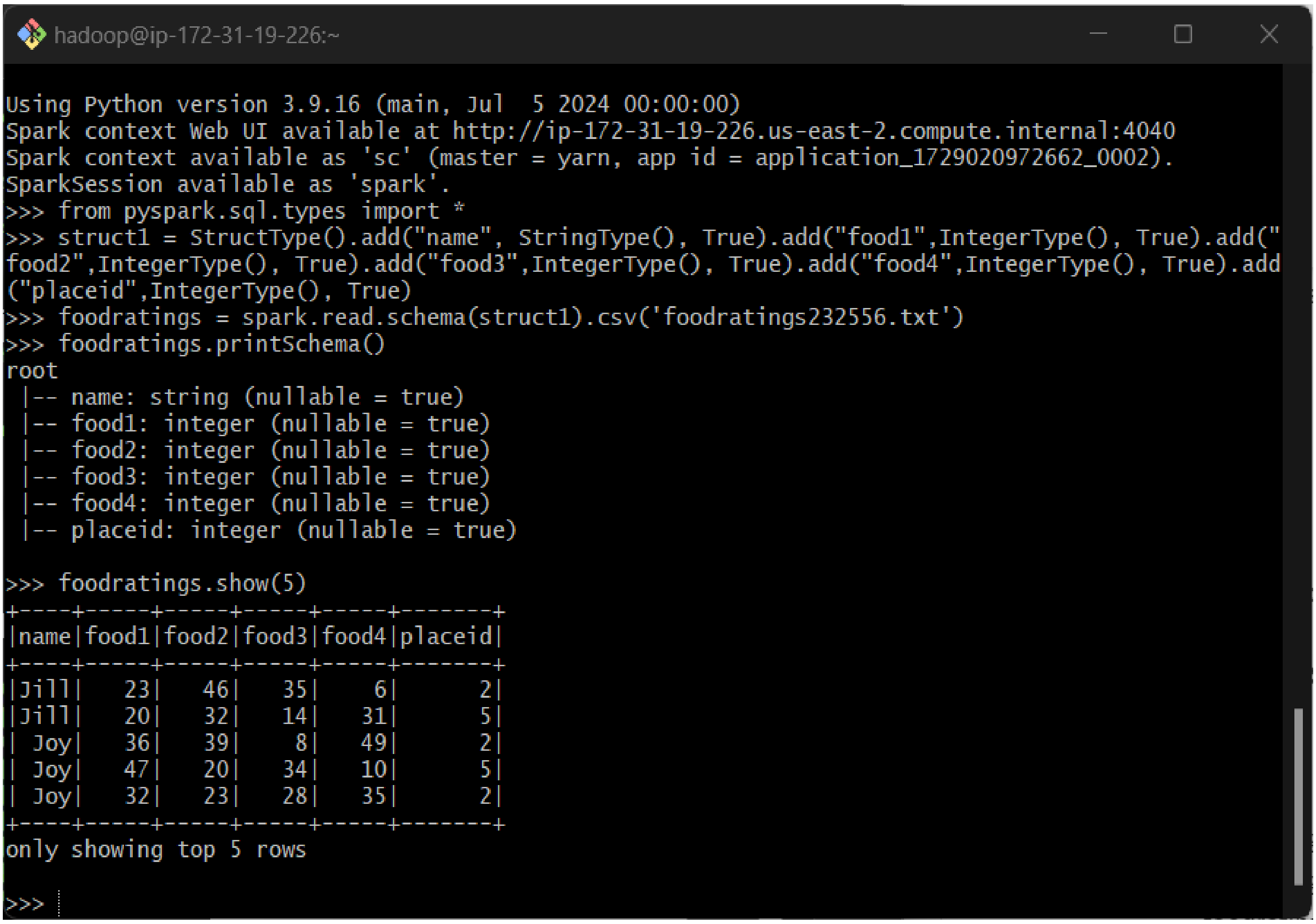
foodratings.printSchema()

foodratings.show(5)



```
Magic Number = 232556
[hadoop@ip-172-31-19-226 ~]$ ls
TestDataGen.class  foodplaces232556.txt  foodratings232556.txt
[hadoop@ip-172-31-19-226 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodplaces232556.txt
[hadoop@ip-172-31-19-226 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodratings232556.txt
[hadoop@ip-172-31-19-226 ~]$ pyspark
Python 3.9.16 (main, Jul  5 2024, 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.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/15 19:50:17 WARN HiveConf: HiveConf of name hive.server2.thrift.url does not exist
24/10/15 19:50:19 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling bac
k to uploading libraries under SPARK_HOME.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.1-amzn-1
      /_/

Using Python version 3.9.16 (main, Jul  5 2024 00:00:00)
Spark context Web UI available at http://ip-172-31-19-226.us-east-2.compute.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1729020972662_0001).
SparkSession available as 'spark'.
>>>
```



```
Using Python version 3.9.16 (main, Jul  5 2024 00:00:00)
Spark context Web UI available at http://ip-172-31-19-226.us-east-2.compute.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1729020972662_0002).
SparkSession available as 'spark'.
>>> 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('foodratings232556.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|
+----+-----+-----+-----+-----+-------+
|Jill|   23|   46|   35|    6|      2|
|Jill|   20|   32|   14|   31|      5|
| Joy|   36|   39|    8|   49|      2|
| Joy|   47|   20|   34|   10|      5|
| Joy|   32|   23|   28|   35|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>>
```

2. Exercise 2)

Load the ‘foodplaces’ file as a ‘csv’ file into a DataFrame called foodplaces.

Ans:

- **Command Used:**

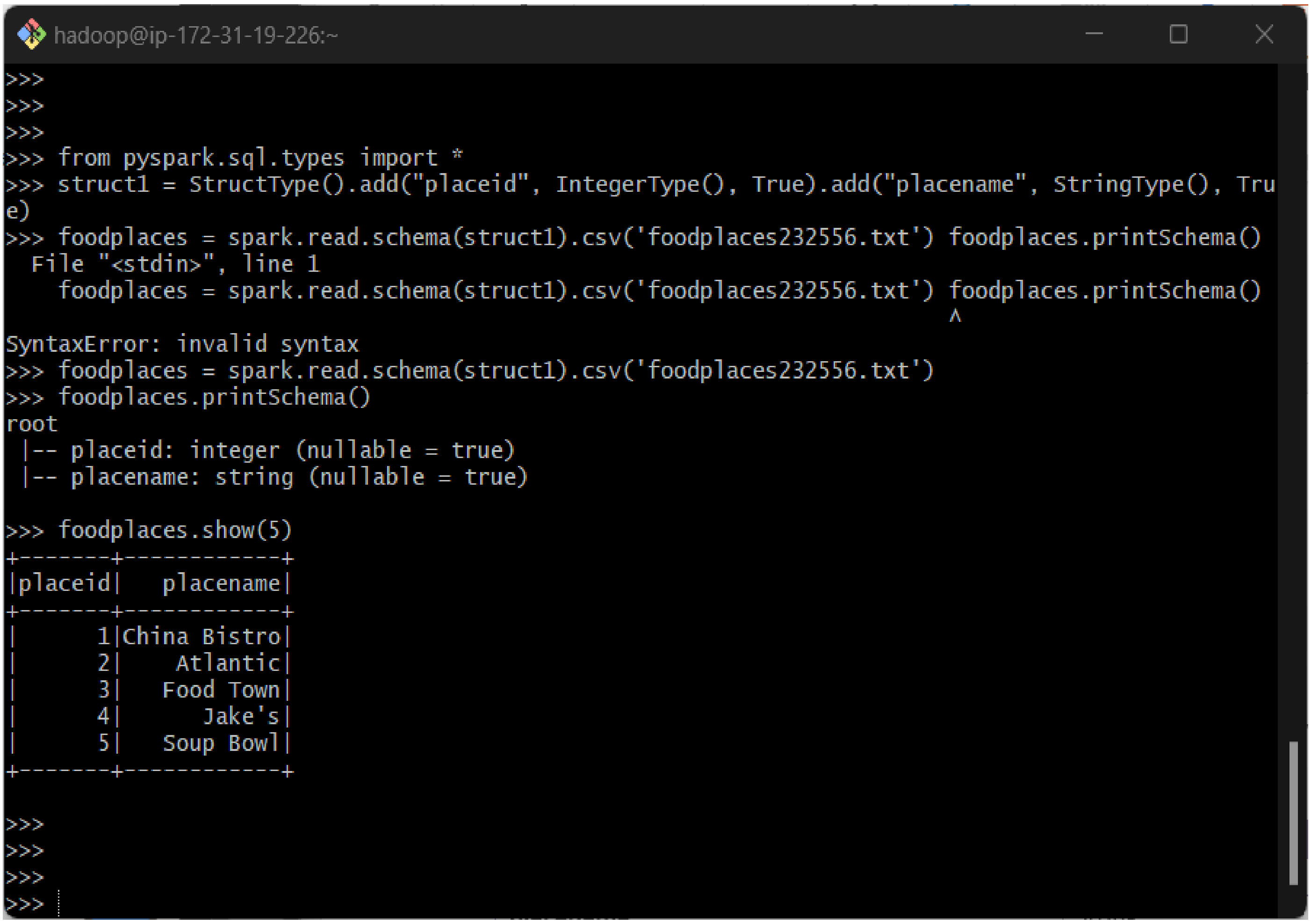
from pyspark.sql.types import *

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

foodplaces = spark.read.schema(struct1).csv('foodplaces232556.txt')

foodplaces.printSchema()

foodplaces.show(5)



```
>>>
>>>
>>>
>>> from pyspark.sql.types import *
>>> struct1 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), Tru
e)
>>> foodplaces = spark.read.schema(struct1).csv('foodplaces232556.txt') foodplaces.printSchema()
  File "<stdin>", line 1
    foodplaces = spark.read.schema(struct1).csv('foodplaces232556.txt') foodplaces.printSchema()
                                                                        ^
SyntaxError: invalid syntax
>>> foodplaces = spark.read.schema(struct1).csv('foodplaces232556.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|
+-------+------------+

>>>
>>>
>>>
>>>
```

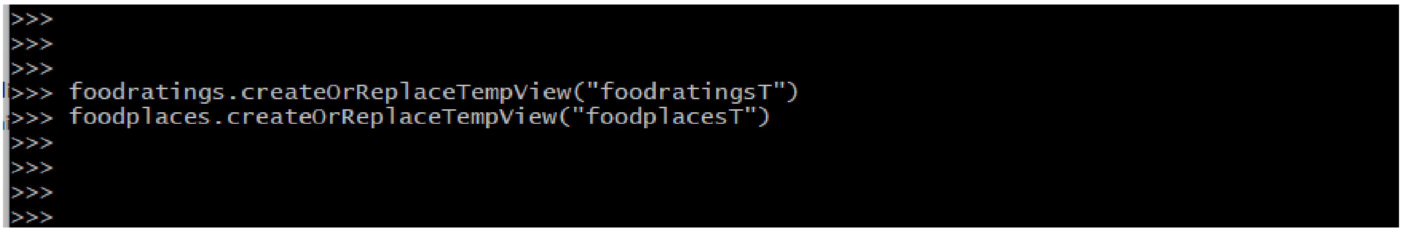
3. Exercise 3)

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

Ans:

- **Command Used:**

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_ex3a holding records which meet the following condition: food2 < 25 and food4 > 40. Remember, when defining conditions in your code use maximum parentheses.

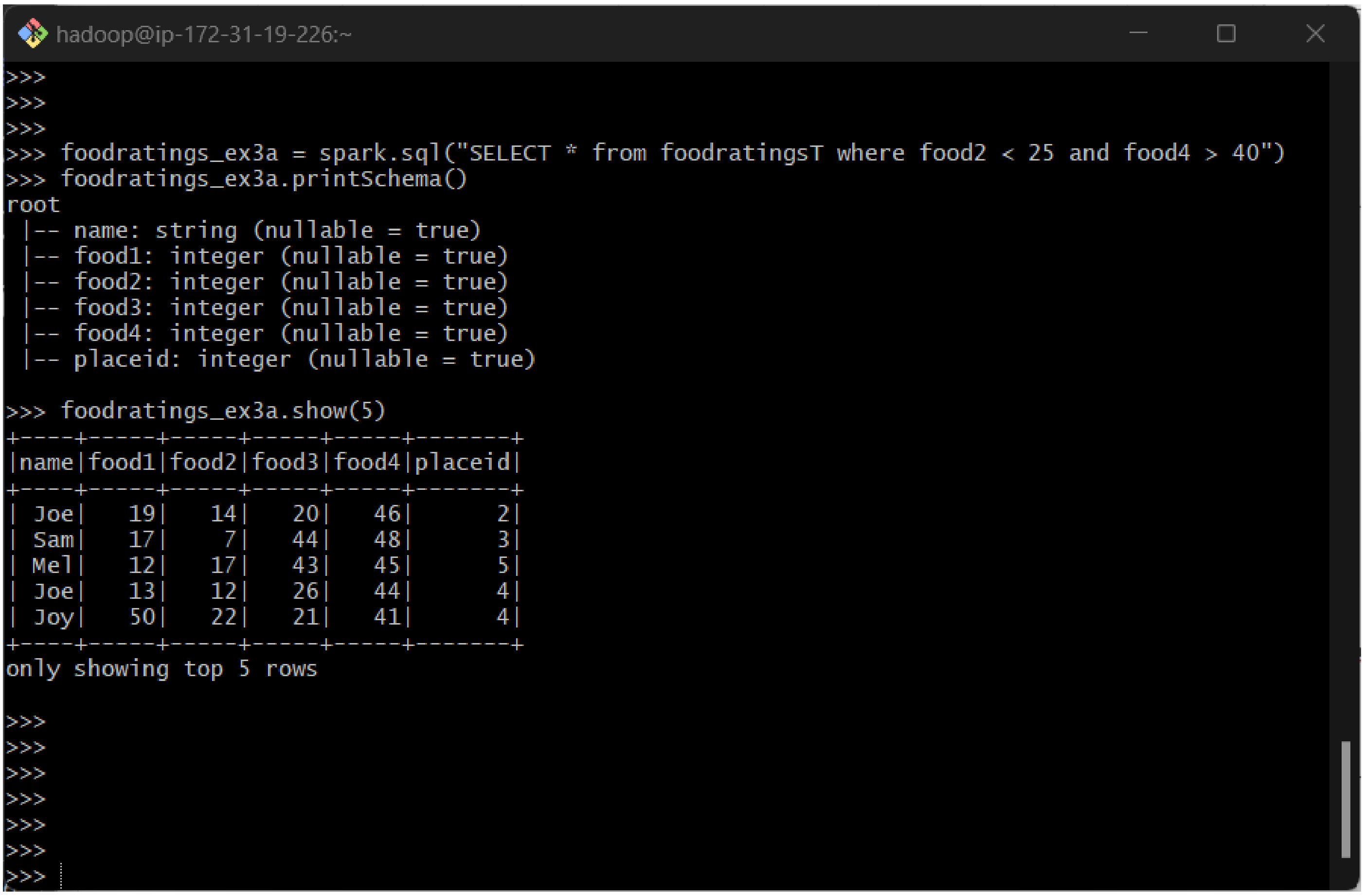
Ans:

- **Command Used:**

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|
+----+-----+-----+-----+-----+-------+
| Joe|   19|   14|   20|   46|      2|
| Sam|   17|    7|   44|   48|      3|
| Mel|   12|   17|   43|   45|      5|
| Joe|   13|   12|   26|   44|      4|
| Joy|   50|   22|   21|   41|      4|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

<u>Step C</u>: 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

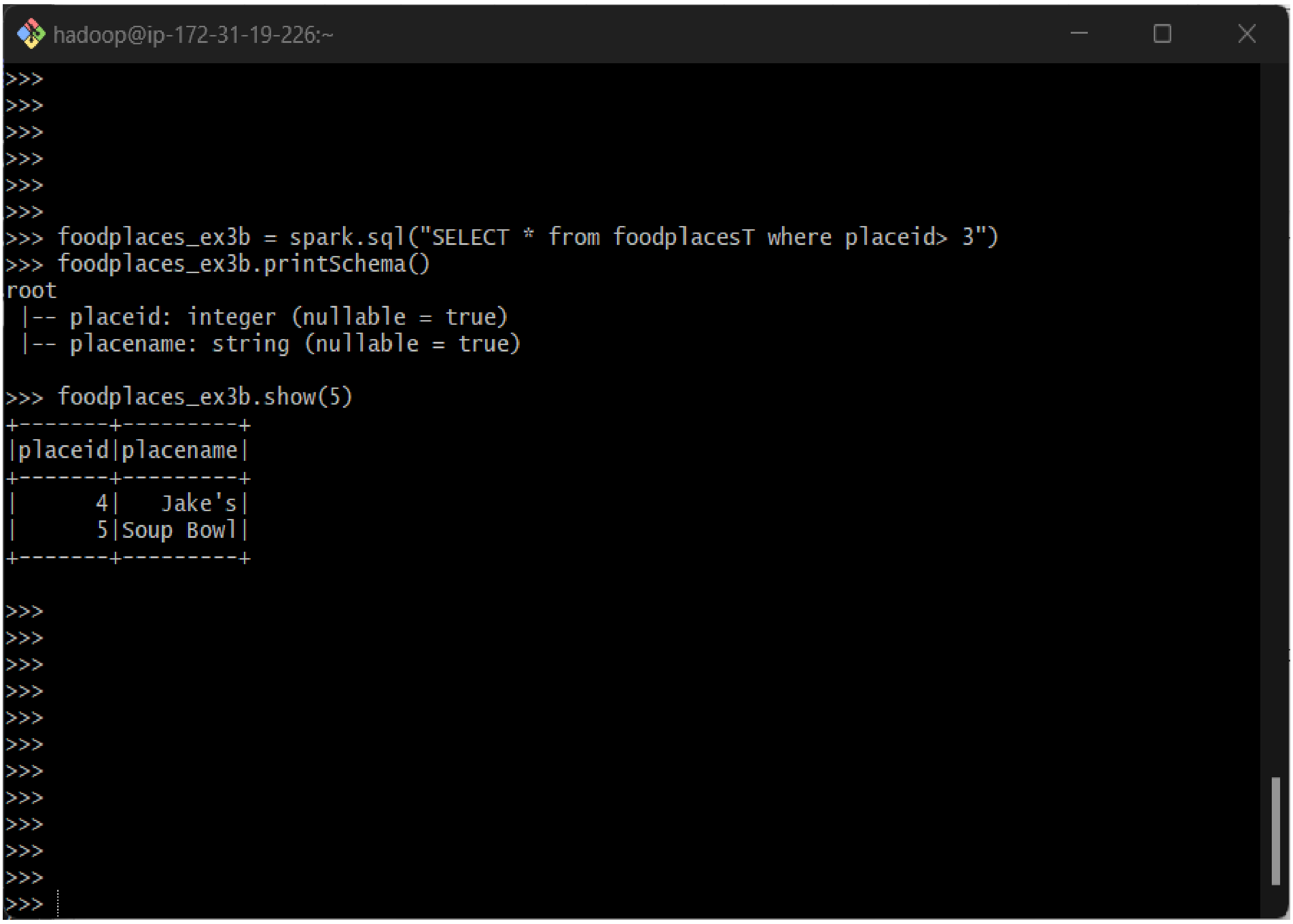
Ans:

- **Command Used:**

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|
+-------+---------+

>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

4. 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.

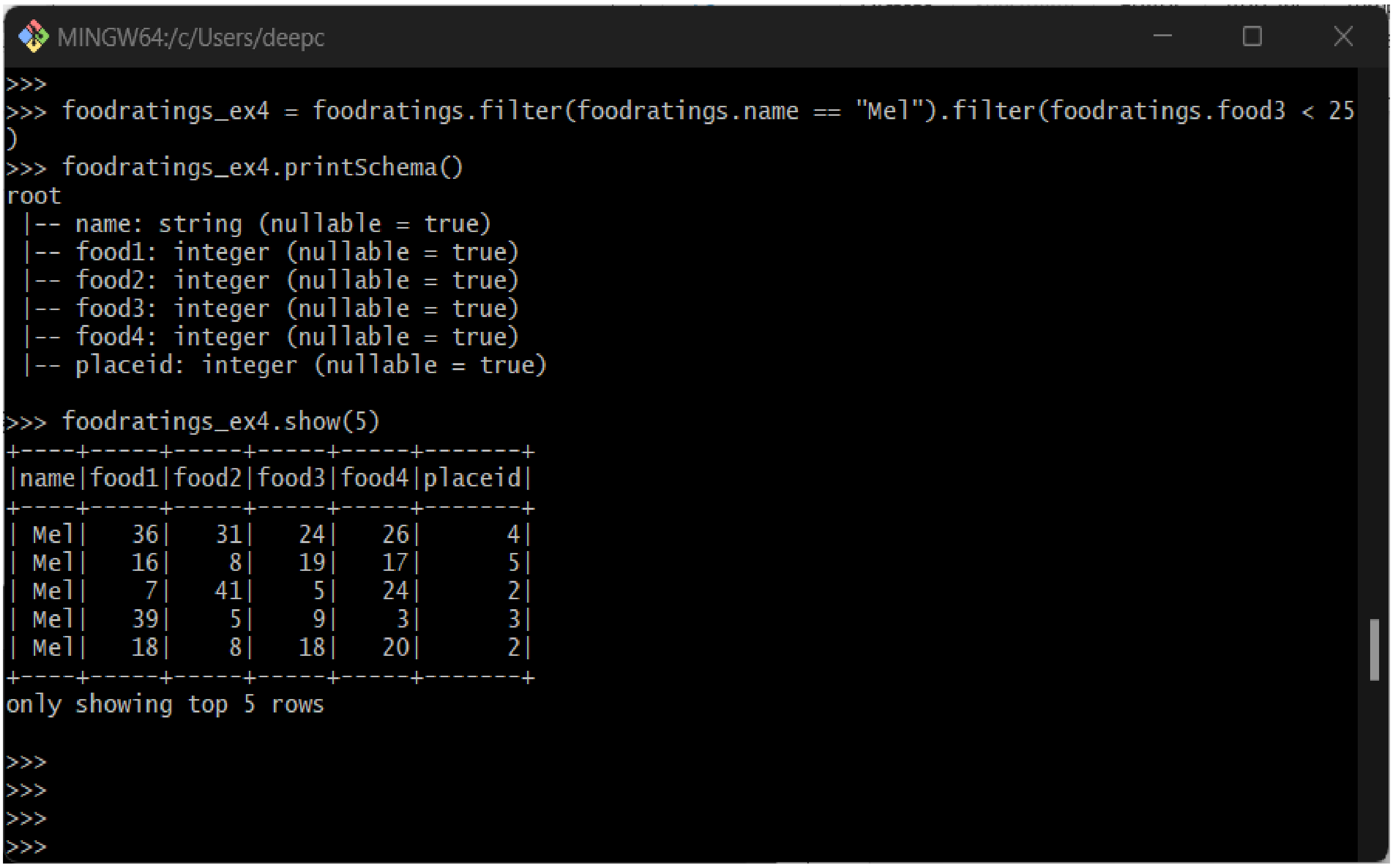
Ans:

- **Command Used:**

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(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|   36|   31|   24|   26|      4|
| Mel|   16|    8|   19|   17|      5|
| Mel|    7|   41|    5|   24|      2|
| Mel|   39|    5|    9|    3|      3|
| Mel|   18|    8|   18|   20|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>>
>>>
>>>
>>>
```

5. 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'

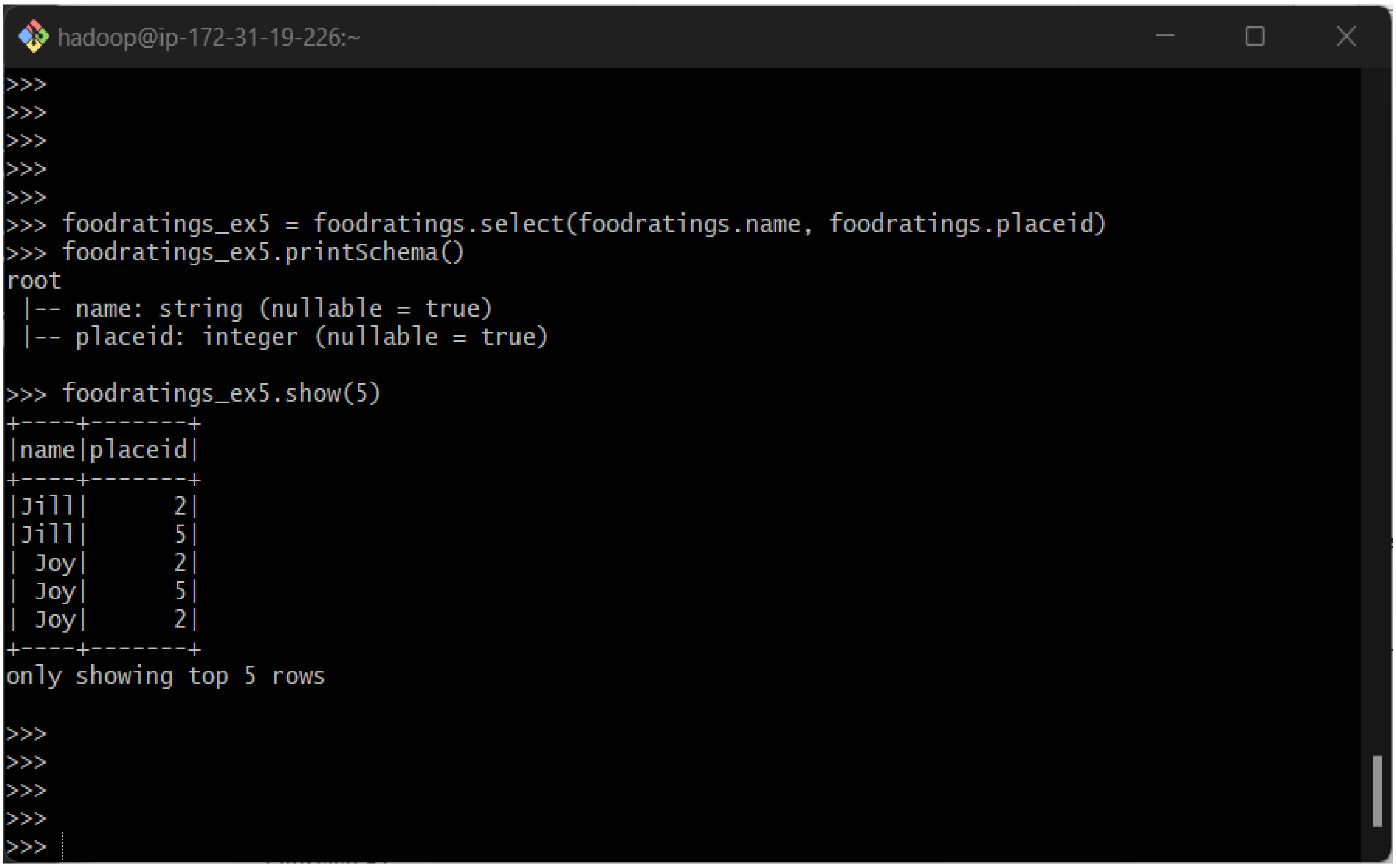
Ans:

- **Command Used:**

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|
+----+-------+
|Jill|      2|
|Jill|      5|
| Joy|      2|
| Joy|      5|
| Joy|      2|
+----+-------+
only showing top 5 rows

>>>
>>>
>>>
>>>
>>>
```

6. 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

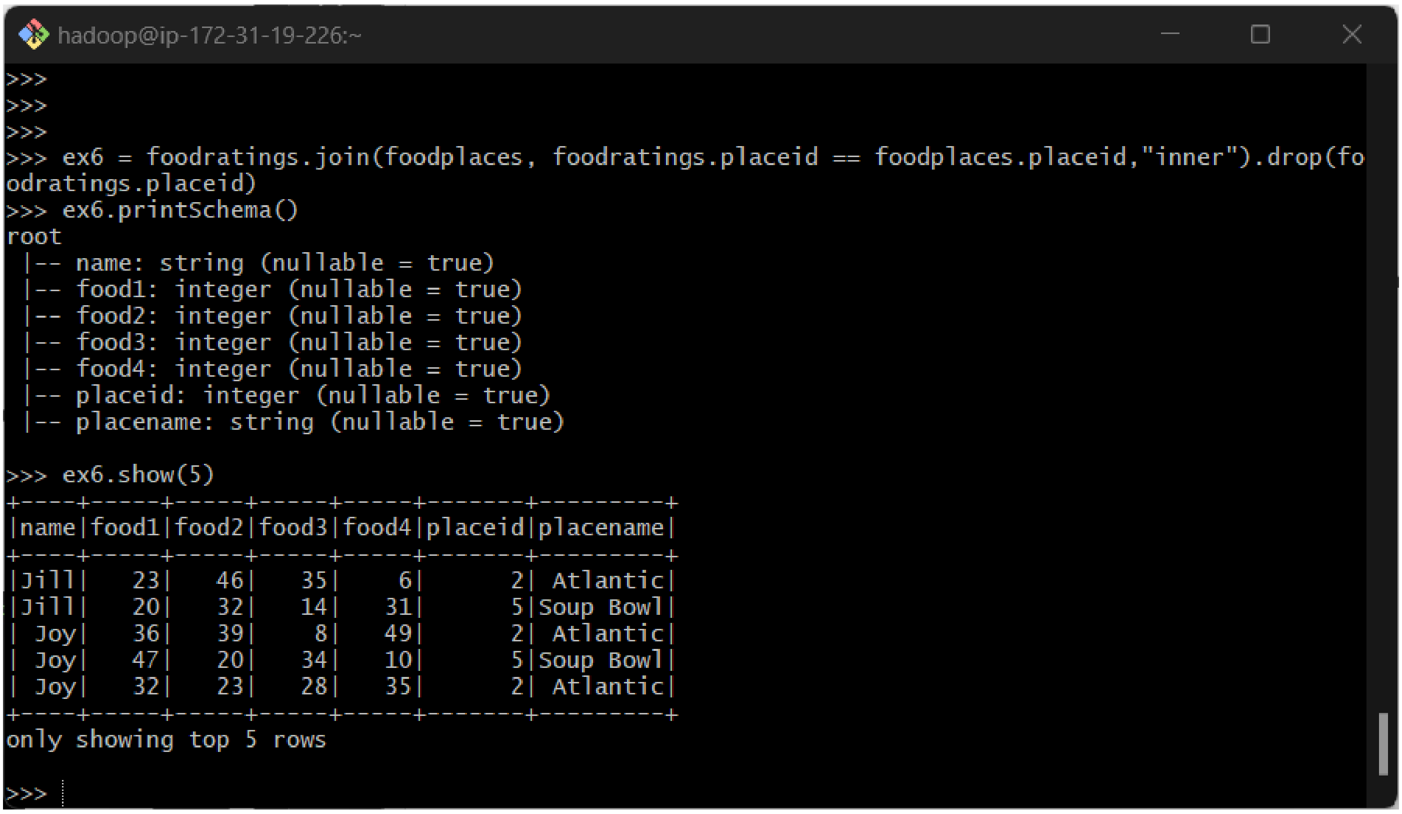
Ans:

- **Command Used:**

ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid,"inner").drop(foodratings.placeid)

ex6.printSchema()

ex6.show(5)



```
>>>
>>>
>>>
>>> ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid,"inner").drop(fo
odratings.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|
+----+-----+-----+-----+-----+-------+---------+
|Jill|   23|   46|   35|    6|      2| Atlantic|
|Jill|   20|   32|   14|   31|      5|Soup Bowl|
| Joy|   36|   39|    8|   49|      2| Atlantic|
| Joy|   47|   20|   34|   10|      5|Soup Bowl|
| Joy|   32|   23|   28|   35|      2| Atlantic|
+----+-----+-----+-----+-----+-------+---------+
only showing top 5 rows

>>>
```