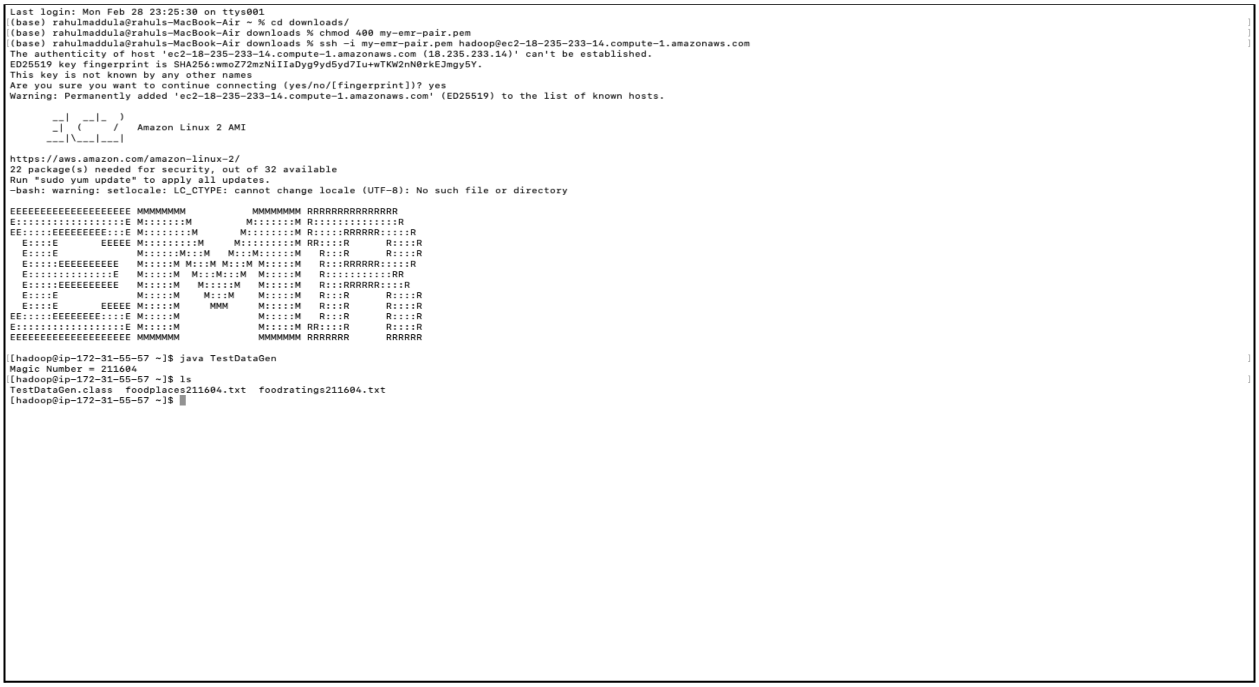
ASSIGNMENT -07

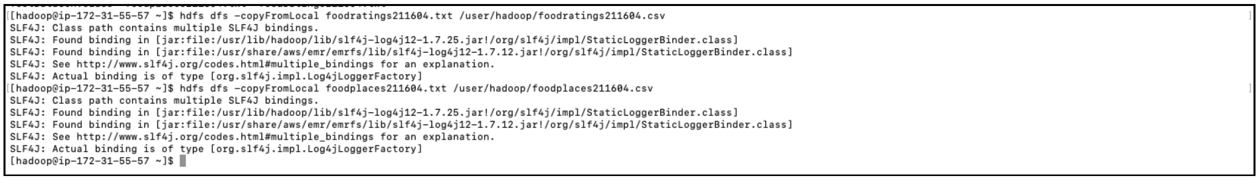
1)

Step B:



```
Last login: Mon Feb 28 23:25:30 on ttys001
(base) rahulmaddula@rahuls-MacBook-Air ~ % cd downloads/
(base) rahulmaddula@rahuls-MacBook-Air downloads % chmod 400 my-emr-pair.pem
(base) rahulmaddula@rahuls-MacBook-Air downloads % ssh -i my-emr-pair.pem hadoop@ec2-18-235-233-14.compute-1.amazonaws.com
The authenticity of host 'ec2-18-235-233-14.compute-1.amazonaws.com (18.235.233.14)' can't be established.
ED25519 key fingerprint is SHA256:wmoZ72mzNiIIaDyg9yd5yd7Iu+wTKW2nN0rkEJmgy5Y.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-235-233-14.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
22 package(s) needed for security, out of 32 available
Run "sudo yum update" to apply all updates.
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory

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-172-31-55-57 ~]$ java TestDataGen
Magic Number = 211604
[hadoop@ip-172-31-55-57 ~]$ ls
TestDataGen.class  foodplaces211604.txt  foodratings211604.txt
[hadoop@ip-172-31-55-57 ~]$ 
```

Magic number=211604



```
[hadoop@ip-172-31-55-57 ~]$ hdfs dfs -copyFromLocal foodratings211604.txt /user/hadoop/foodratings211604.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/aws/emr/emrfs/lib/slf4j-log4j12-1.7.12.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
[hadoop@ip-172-31-55-57 ~]$ hdfs dfs -copyFromLocal foodplaces211604.txt /user/hadoop/foodplaces211604.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/share/aws/emr/emrfs/lib/slf4j-log4j12-1.7.12.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
[hadoop@ip-172-31-55-57 ~]$ 
```

Commands:

```
hdfs dfs -copyFromLocal foodratings211604.txt /user/hadoop/foodratings211604.csv

hdfs dfs -copyFromLocal foodplaces211604.txt /user/hadoop/foodplaces211604.csv
```

Step C:

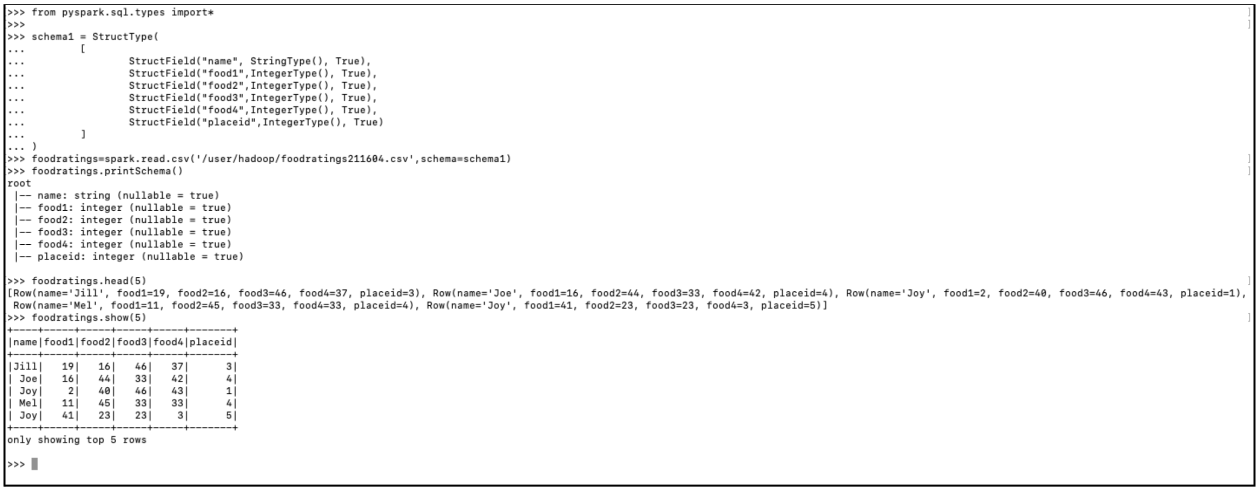
Magic number=211604

Code :

```
from pyspark.sql.types import*
>>>
```

```
>>> schema1 = 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)
...         ]
... )
>>> foodratings=spark.read.csv('/user/hadoop/foodratings211604.csv',schema=schema1)
>>> foodratings.printSchema()
>>> foodratings.head(5)
>>> foodratings.show(5)
```

Outputs:



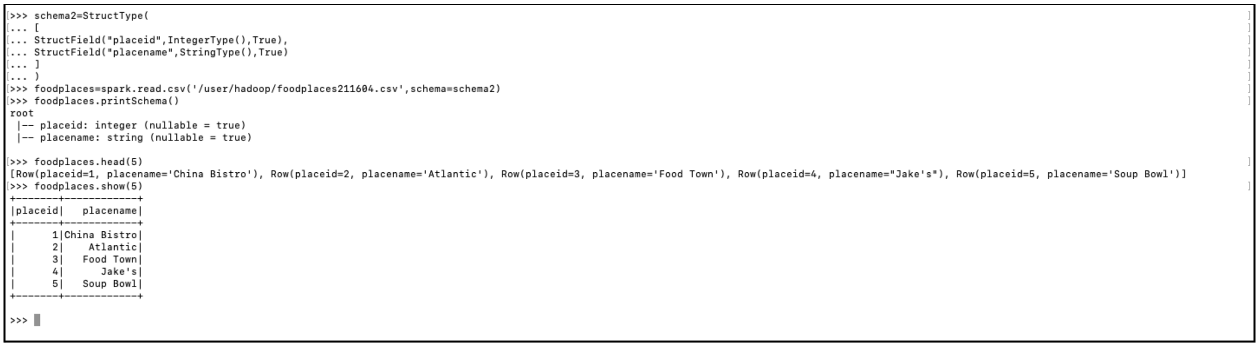
```
>>> from pyspark.sql.types import*
>>>
>>> schema1 = StructType(
...         [
...                 StructField("name", StringType(), True),
...                 StructField("food1",IntegerType(), True),
...                 StructField("food2",IntegerType(), True),
...                 StructField("food3",IntegerType(), True),
...                 StructField("food4",IntegerType(), True),
...                 StructField("placeid",IntegerType(), True)
...         ]
... )
>>> foodratings=spark.read.csv('/user/hadoop/foodratings211604.csv',schema=schema1)
>>> 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.head(5)
[Row(name='Jill', food1=19, food2=16, food3=46, food4=37, placeid=3), Row(name='Joe', food1=16, food2=44, food3=33, food4=42, placeid=4), Row(name='Joy', food1=2, food2=40, food3=46, food4=43, placeid=1),
 Row(name='Mel', food1=11, food2=45, food3=33, food4=33, placeid=4), Row(name='Joy', food1=41, food2=23, food3=23, food4=3, placeid=5)]
>>> foodratings.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
|Jill|   19|   16|   46|   37|      3|
| Joe|   16|   44|   33|   42|      4|
| Joy|    2|   40|   46|   43|      1|
| Mel|   11|   45|   33|   33|      4|
| Joy|   41|   23|   23|    3|      5|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>>
```

2)

Code:

```
schema2=StructType(
... [
... StructField("placeid",IntegerType(),True),
... StructField("placename",StringType(),True)
... ]
... )
>>> foodplaces=spark.read.csv('/user/hadoop/foodplaces211604.csv',schema=schema2)
>>> foodplaces.printSchema()
>>> foodplaces.head(5)
>>> foodplaces.show(5)
```

Output:



```
>>> schema2=StructType(
... [
... StructField("placeid",IntegerType(),True),
... StructField("placename",StringType(),True)
... ]
... )
>>> foodplaces=spark.read.csv('/user/hadoop/foodplaces211604.csv',schema=schema2)
>>> foodplaces.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces.head(5)
[Row(placeid=1, placename='China Bistro'), Row(placeid=2, placename='Atlantic'), Row(placeid=3, placename='Food Town'), Row(placeid=4, placename="Jake's"), Row(placeid=5, placename='Soup Bowl')]
>>> foodplaces.show(5)
+-------+------------+
|placeid|   placename|
+-------+------------+
|      1|China Bistro|
|      2|    Atlantic|
|      3|   Food Town|
|      4|      Jake's|
|      5|   Soup Bowl|
+-------+------------+

>>> 
```

3)

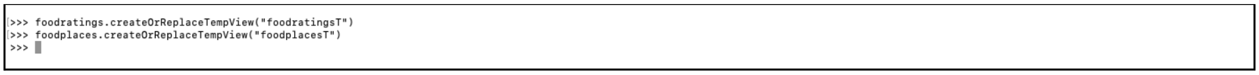
Step A:

Code:

foodratings.createOrReplaceTempView("foodratingsT")

foodplaces.createOrReplaceTempView("foodplacesT")

Output:



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

Step B:

Code:

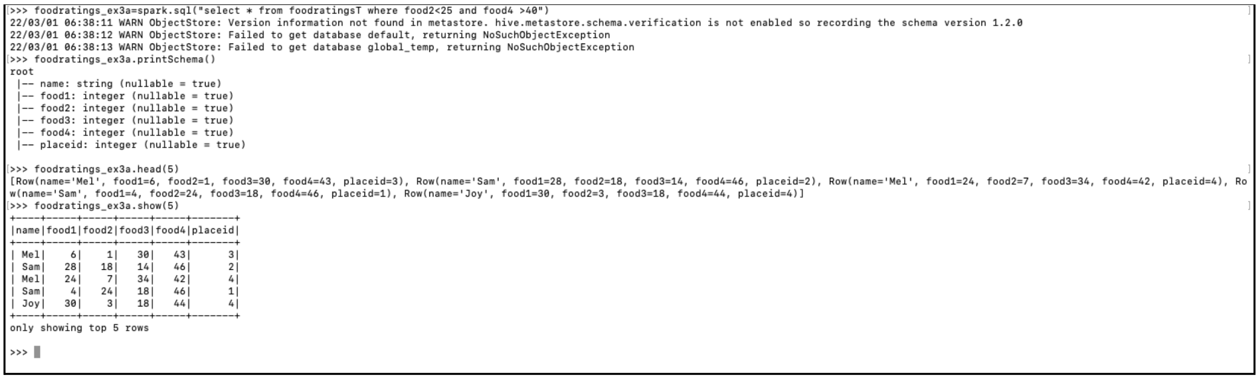
foodratings_ex3a=spark.sql("select * from foodratingsT where food2<25 and food4 >40")

foodratings_ex3a.printSchema()

foodratings_ex3a.head(5)

foodratings_ex3a.show(5)

Output:



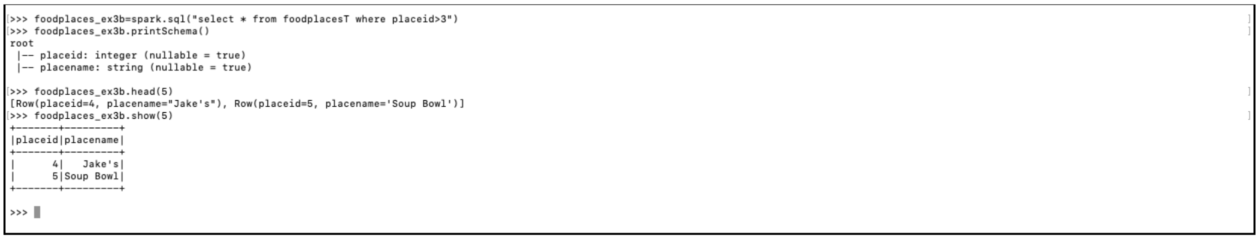
```
>>> foodratings_ex3a=spark.sql("select * from foodratingsT where food2<25 and food4 >40")
22/03/01 06:38:11 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
22/03/01 06:38:12 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
22/03/01 06:38:13 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
>>> 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.head(5)
[Row(name='Mel', food1=6, food2=1, food3=30, food4=43, placeid=3), Row(name='Sam', food1=28, food2=18, food3=14, food4=46, placeid=2), Row(name='Mel', food1=24, food2=7, food3=34, food4=42, placeid=4), Ro
w(name='Sam', food1=4, food2=24, food3=18, food4=46, placeid=1), Row(name='Joy', food1=30, food2=3, food3=18, food4=44, placeid=4)]
>>> foodratings_ex3a.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|    6|    1|   30|   43|      3|
| Sam|   28|   18|   14|   46|      2|
| Mel|   24|    7|   34|   42|      4|
| Sam|    4|   24|   18|   46|      1|
| Joy|   30|    3|   18|   44|      4|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>> 
```

Step C:

Code:

foodplaces_ex3b=spark.sql("select * from foodplacesT where placeid>3")
foodplaces_ex3b.printSchema()
foodrplaces_ex3b.head(5)
foodplaces_ex3b.show(5)

Output:

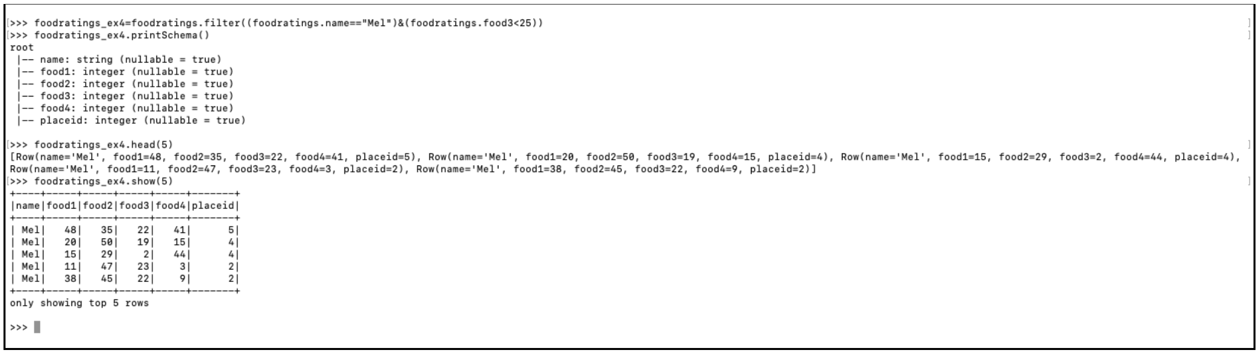


```
>>> foodplaces_ex3b=spark.sql("select * from foodplacesT where placeid>3")
>>> foodplaces_ex3b.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces_ex3b.head(5)
[Row(placeid=4, placename="Jake's"), Row(placeid=5, placename='Soup Bowl')]
>>> foodplaces_ex3b.show(5)
+-------+---------+
|placeid|placename|
+-------+---------+
|      4|   Jake's|
|      5|Soup Bowl|
+-------+---------+

>>> 
```

4)
Code:

foodratings_ex4=foodratings.filter((foodratings.name=="Mel")&(foodratings.food3<25))
foodratings_ex4.printSchema()
foodratings_ex4.head(5)
foodratings_ex4.show(5)

Output:



```
>>> 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.head(5)
[Row(name='Mel', food1=48, food2=35, food3=22, food4=41, placeid=5), Row(name='Mel', food1=20, food2=50, food3=19, food4=15, placeid=4), Row(name='Mel', food1=15, food2=29, food3=2, food4=44, placeid=4),
Row(name='Mel', food1=11, food2=47, food3=23, food4=3, placeid=2), Row(name='Mel', food1=38, food2=45, food3=22, food4=9, placeid=2)]
>>> foodratings_ex4.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   48|   35|   22|   41|      5|
| Mel|   20|   50|   19|   15|      4|
| Mel|   15|   29|    2|   44|      4|
| Mel|   11|   47|   23|    3|      2|
| Mel|   38|   45|   22|    9|      2|
+----+-----+-----+-----+-----+-------+
only showing top 5 rows

>>> 
```

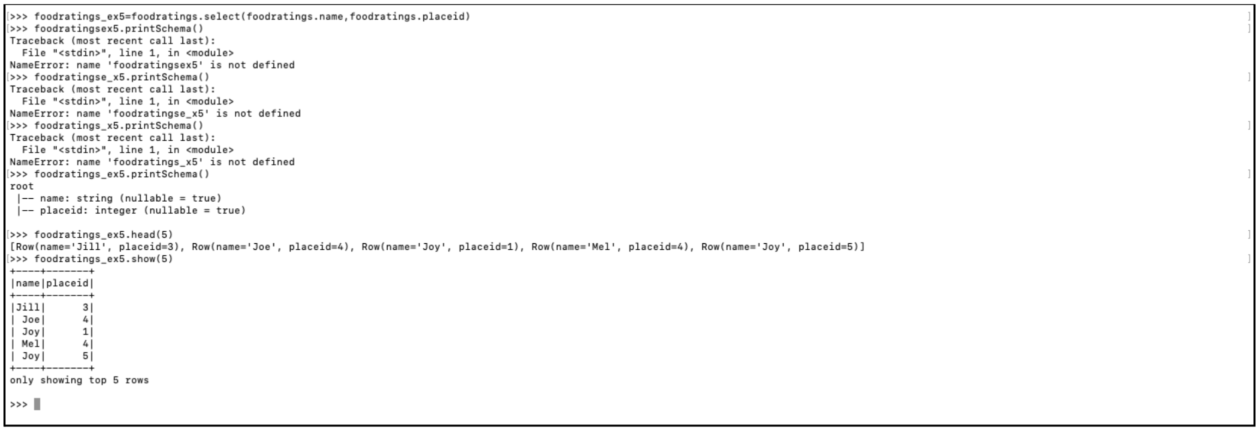
5)

Code:

foodratings_ex5=foodratings.select(foodratings.name,foodratings.placeid)
foodratings_ex5.printSchema()
foodratings_ex5.head(5)

foodratings_ex5.show(5)

Output:



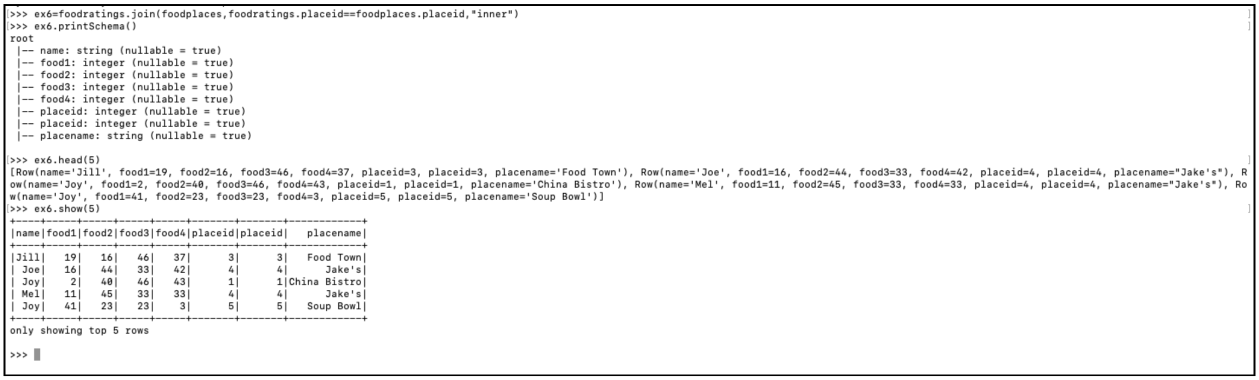
```
>>> foodratings_ex5=foodratings.select(foodratings.name,foodratings.placeid)
>>> foodratingsex5.printSchema()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratingsex5' is not defined
>>> foodratingse_x5.printSchema()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratingse_x5' is not defined
>>> foodratings_x5.printSchema()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratings_x5' is not defined
>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex5.head(5)
[Row(name='Jill', placeid=3), Row(name='Joe', placeid=4), Row(name='Joy', placeid=1), Row(name='Mel', placeid=4), Row(name='Joy', placeid=5)]
>>> foodratings_ex5.show(5)
+----+-------+
|name|placeid|
+----+-------+
|Jill|      3|
| Joe|      4|
| Joy|      1|
| Mel|      4|
| Joy|      5|
+----+-------+
only showing top 5 rows

>>> 
```

6)

Code:

```
ex6=foodratings.join(foodplaces,foodratings.placeid==foodplaces.placeid,"inner")
 ex6.printSchema()
ex6.head(5)
ex6.show(5)
```

Output:



```
>>> 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.head(5)
[Row(name='Jill', food1=19, food2=16, food3=46, food4=37, placeid=3, placeid=3, placename='Food Town'), Row(name='Joe', food1=16, food2=44, food3=33, food4=42, placeid=4, placeid=4, placename="Jake's"), R
ow(name='Joy', food1=2, food2=40, food3=46, food4=43, placeid=1, placeid=1, placename='China Bistro'), Row(name='Mel', food1=11, food2=45, food3=33, food4=33, placeid=4, placeid=4, placename="Jake's"), Ro
w(name='Joy', food1=41, food2=23, food3=23, food4=3, placeid=5, placeid=5, placename='Soup Bowl')]
>>> ex6.show(5)
+----+-----+-----+-----+-----+-------+-------+------------+
|name|food1|food2|food3|food4|placeid|placeid|   placename|
+----+-----+-----+-----+-----+-------+-------+------------+
|Jill|   19|   16|   46|   37|      3|      3|   Food Town|
| Joe|   16|   44|   33|   42|      4|      4|      Jake's|
| Joy|    2|   40|   46|   43|      1|      1|China Bistro|
| Mel|   11|   45|   33|   33|      4|      4|      Jake's|
| Joy|   41|   23|   23|    3|      5|      5|   Soup Bowl|
+----+-----+-----+-----+-----+-------+-------+------------+
only showing top 5 rows

>>> 
```