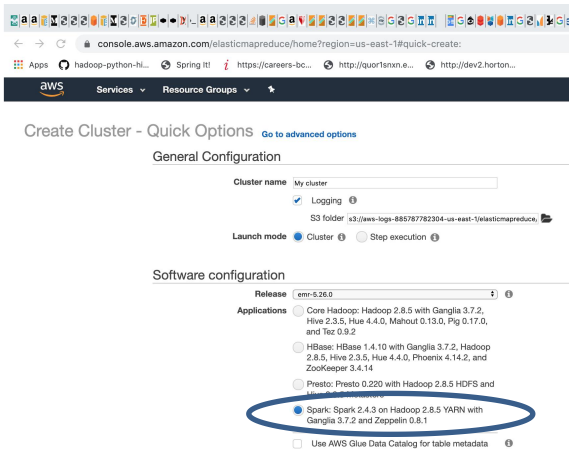


A20512400

Step A

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



Step B

```

pradaappss - hadoop@ip-172-31-9-146: ~ ssh -i ~/Downloads/bdt_4/emr-key-pair.pem hadoop@ec2-44-212-11-157.compute-1.amazonaws.com ...
[pradaappss@Pradaappss-MacBook-Air ~ % ssh -i /Users/pradaappss/Downloads/bdt_4/emr-key-pair.pem hadoop@ec2-44-212-11-157.compute-1.amazonaws.com ~]
 _ _ |   _ _ |
 _ _|| _ _ ||   _ _ |
 _ _|| _ _ ||   _ _ |

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
15 packages(s) needed for security, out of 18 available
Run "sudo yum update" to apply all updates.
--bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory

EEEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRRRRRRRRR
E:::|||||:|||||: M:::||||:~M           M:::||||:~M R:::|||||:|||||:~R
EE:::|||||:EEEEEEEE::: M:::||||:~M     M:::||||:~M R:::|||||:RRRRRR:::~R
 E:::~E      EEEEE M:::||||:~M         M:::||||:~M RR:::~R    R:::~R
 E:::~E      M:::||||:~M:::~M:::~M R:::~R    R:::~R
 E:::|||||:EEEEEEEEEE M:::~M M:::~M M:::~M M:::~M R:::~R RRRRR:::~R
 E:::|||||:|||||: M:::~M M:::~M M:::~M M:::~M R:::~R R:::~R
 E:::|||||:EEEEEEEEEE M:::~M M:::~M M:::~M R:::~R R:::~R RRRRR:::~R
 E:::~E      EEEEE M:::~M M:::~M M:::~M R:::~R R:::~R
 E:::~E      MMM M:::~M M:::~M R:::~R R:::~R
 EE:::|||||:EEEEEEEE::: M:::~M M:::~M R:::~R R:::~R
 E:::|||||:|||||: M:::~M M:::~M RR:::~R R:::~R
 EEEEEEEEEEEEEEEEEEE MMMMMMMM          MMMMMMMM RRRRRRRR    RRRRRR

[hadoop@ip-172-31-9-146 ~]$ hadoop fs -ls /user
Found 5 items
drwxrwxrwx - hadoop hdfsadmingroup 0 2022-10-20 08:18 /user/hadoop
drwxrwxrwx - livy livy 0 2022-10-20 08:18 /user/livy
drwxrwxrwx - root hdfsadmingroup 0 2022-10-20 08:18 /user/root
drwxrwxrwx - spark spark 0 2022-10-20 08:18 /user/spark
drwxrwxrwx - zeppelin hdfsadmingroup 0 2022-10-20 08:18 /user/zeppelin
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -mkdir /user/csp554
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -ls /user/
Found 6 items
drwxr-xr-x - hadoop hdfsadmingroup 0 2022-10-20 08:21 /user/csp554
drwxrwxrwx - hadoop hdfsadmingroup 0 2022-10-20 08:18 /user/hadoop
drwxrwxrwx - livy livy 0 2022-10-20 08:18 /user/livy
drwxrwxrwx - root hdfsadmingroup 0 2022-10-20 08:18 /user/root
drwxrwxrwx - spark spark 0 2022-10-20 08:18 /user/spark
drwxrwxrwx - zeppelin hdfsadmingroup 0 2022-10-20 08:18 /user/zeppelin
```

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

Copy both generated files to the HDFS directory “/user/hadoop”

Magic Number: 187912

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 146x53
[hadoop@ip-172-31-9-146 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-9-146 ~]$ java TestDataGen
Magic Number = 187912
[hadoop@ip-172-31-9-146 ~]$ ls
TestDataGen.class  foodplaces187912.txt  foodratings187912.txt
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -cp foodratings187912.txt /user/csp554
-bash: hadoop: command not found
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -cp foodratings187912.txt /user/csp554
cp: `foodratings187912.txt': No such file or directory
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -put foodratings187912.txt /user/csp554
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -put foodplaces187912.txt /user/csp554
[hadoop@ip-172-31-9-146 ~]$ ls
TestDataGen.class  foodplaces187912.txt  foodratings187912.txt
[hadoop@ip-172-31-9-146 ~]$ hadoop fs -ls /user/csp554
Found 2 items
-rw-r--r-- 1 hadoop hdfsadmingroup 59 2022-10-20 08:32 /user/csp554/foodplaces187912.txt
-rw-r--r-- 1 hadoop hdfsadmingroup 17520 2022-10-20 08:32 /user/csp554/foodratings187912.txt
[hadoop@ip-172-31-9-146 ~]$ pyspark
Python 3.7.10 (default, Jun 3 2021, 00:02:01)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-13)] 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).
22/10/20 08:33:08 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to

  ____      _
 / ___|  __| | | |
| |  | |__| | | |
| |  | |__| | | |
| |  | |__| | | |
|_|  |____|_|_|_|

 version 2.4.8-amzn-2

Using Python version 3.7.10 (default, Jun 3 2021 00:02:01)
SparkSession available as 'spark'.
```

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

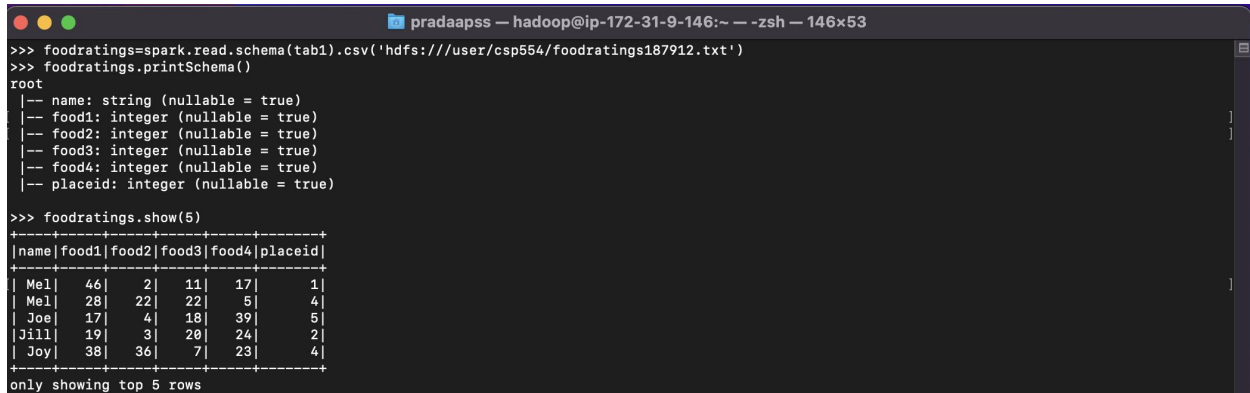
```
>>>from pyspark.sql.types import *
>>>tab1=StructType().add("name",StringType(),True).add("food1",Integer
Type(),True).add("food2",IntegerType(),True).add("food3",IntegerType(),
True).add("food4",IntegerType(),True).add("placeid",IntegerType(),True)
>>>foodratings=spark.read.schema(tab1).csv('hdfs:///user/csp554/foodra
tings187912.txt')
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 110x53
SparkSession available as 'spark'.
>>> from pyspark.sql.types import *
>>> tab1=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)
```

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

```
foodratings.printSchema()
```

```
foodratings.show(5)
```



```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 146x53
>>> foodratings=spark.read.schema(tab1).csv('hdfs:///user/csp554/foodratings187912.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|
+-----+
|Mel|  46|   2|  11|  17|   1|
|Mel|  28|  22|  22|   5|   4|
|Joe|  17|   4|  18|  39|   5|
|Jill| 19|   3|  20|  24|   2|
|Joy|  38|  36|   7|  23|   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 Name	Field Type
placeid	Integer
placename	String

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

```
>>> tab2 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), True)
>>> foodplaces = spark.read.schema(tab2).csv('hdfs:///user/csp554/foodplaces187912.txt')
```

```
foodplaces.printSchema()
```

```
foodplaces.show(5)
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 110x53
[>>> tab2=StructType().add("placeid",IntegerType(),True).add("placename",StringType(),True)
[>>> foodplaces=spark.read.schema(tab2).csv('hdfs:///user/csp554/foodplaces187912.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.createOrReplaceTempView("foodratingsT")
>>> foodplaces.createOrReplaceTempView("foodplacesT")
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 110x53
[>>> 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.show(5)
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 115x53

[>>> 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|   10|    3|   25|   44|      1|
|Jill|   47|   23|   29|   43|      3|
| Joe|   23|   17|   45|   49|      5|
| Joe|   13|   21|    5|   48|      5|
| Joe|   14|   20|   44|   48|      4|
+-----+-----+-----+-----+-----+
only showing top 5 rows
```

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

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.show(5)
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 115x53

[>>> foodplaces_ex3b=spark.sql("select * from foodplacesT where placeid>3")
[>>> foodratings_ex3b.printSchema()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratings_ex3b' is not defined
[>>> foodplaces_ex3b=spark.sql("select * from foodplacesT where placeid>3")
[>>> foodplaces_ex3b.show(5)
+-----+-----+
|placeid|placename|
+-----+-----+
|      4|   Jake's|
|      5| Soup Bowl|
+-----+-----+

[>>> 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|
+-----+-----+
```

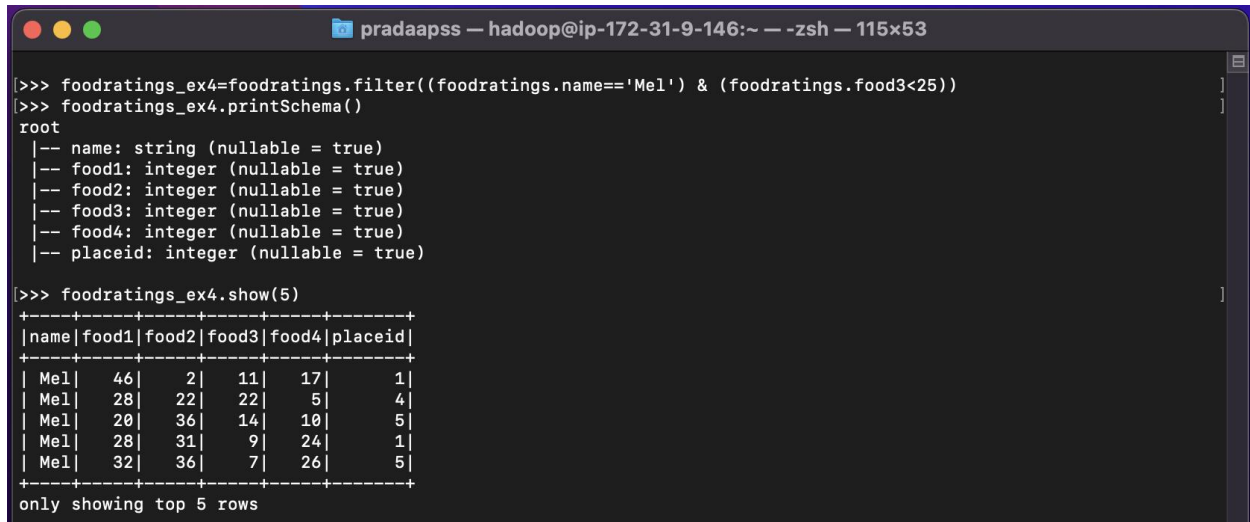
Exercise 4)

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.

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') &
(foodratings.food3<25))
foodratings_ex4.printSchema()

foodratings_ex4.show(5)
```



```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 115x53

[>>> 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|  46|   2|  11|  17|     1|
|Mel|  28|  22|  22|   5|     4|
|Mel|  20|  36|  14|  10|     5|
|Mel|  28|  31|   9|  24|     1|
|Mel|  32|  36|   7|  26|     5|
+-----+
only showing top 5 rows
```

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'

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.show(5)
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 115x45

[>>> 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|
+-----+
|Mel|    1|
|Mel|    4|
|Joe|    5|
|Jill|   2|
|Joy|    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 inner join, 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.pla
ceid,'inner').drop(foodplaces.placeid)
ex6.printSchema()

ex6.show(5)
```

```
pradaapss — hadoop@ip-172-31-9-146:~ — zsh — 115x27

>>> ex6=foodratings.join(foodplaces,foodratings.placeid==foodplaces.placeid,'inner').drop(foodplaces.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|
+-----+-----+-----+-----+-----+-----+
|Mel|   46|    2|   11|   17|    1|China Bistro|
|Mel|   28|   22|   22|    5|    4|   Jake's|
|Joe|   17|    4|   18|   39|    5|  Soup Bowl|
|Jill|  19|    3|   20|   24|    2|  Atlantic|
|Joy|   38|   36|    7|   23|    4|   Jake's|
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> Connection to ec2-44-212-11-157.compute-1.amazonaws.com closed by remote host.
Connection to ec2-44-212-11-157.compute-1.amazonaws.com closed.
pradaapss@Pradaaps-MacBook-Air ~ %
```