CSP 554 Big Data Technologies

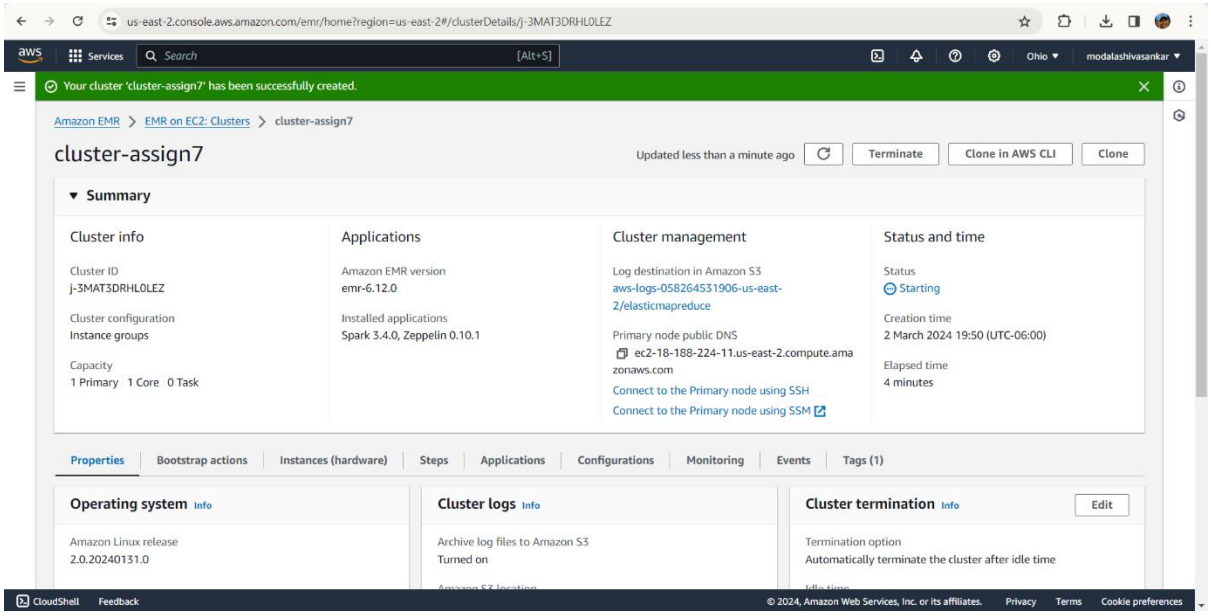
Assignment – #7

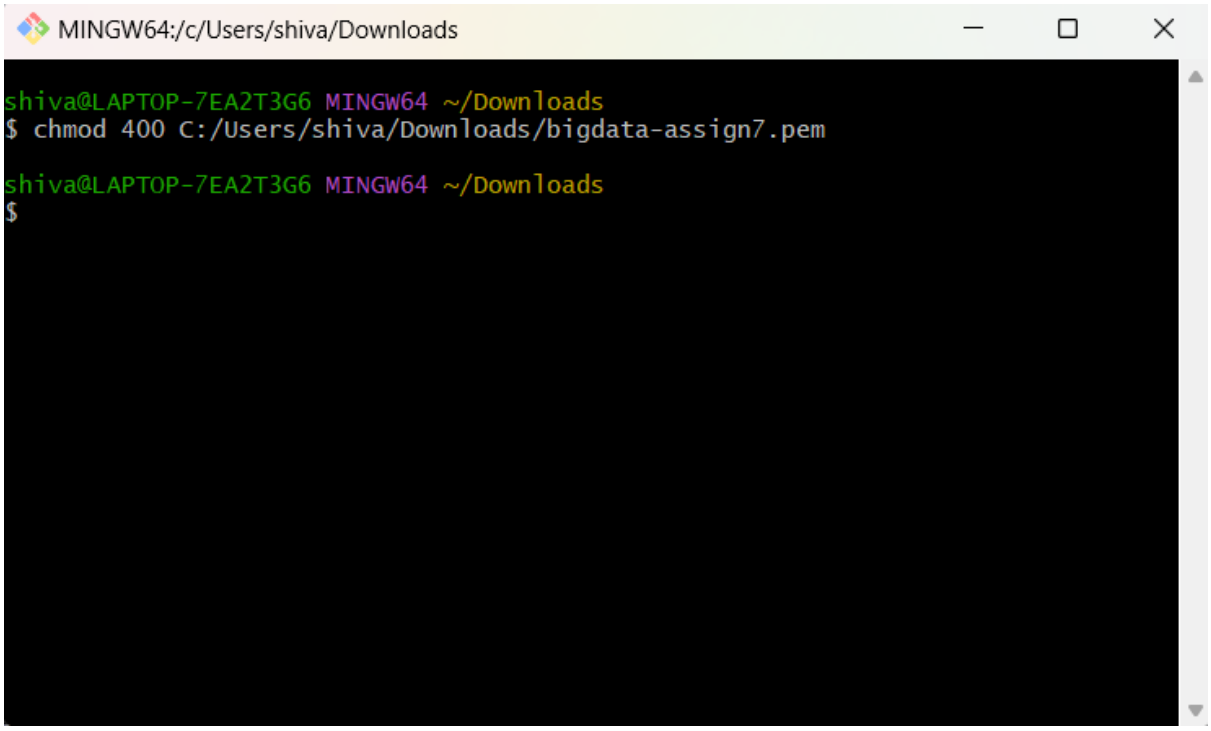
Shiva Sankar Modala(A20517528)

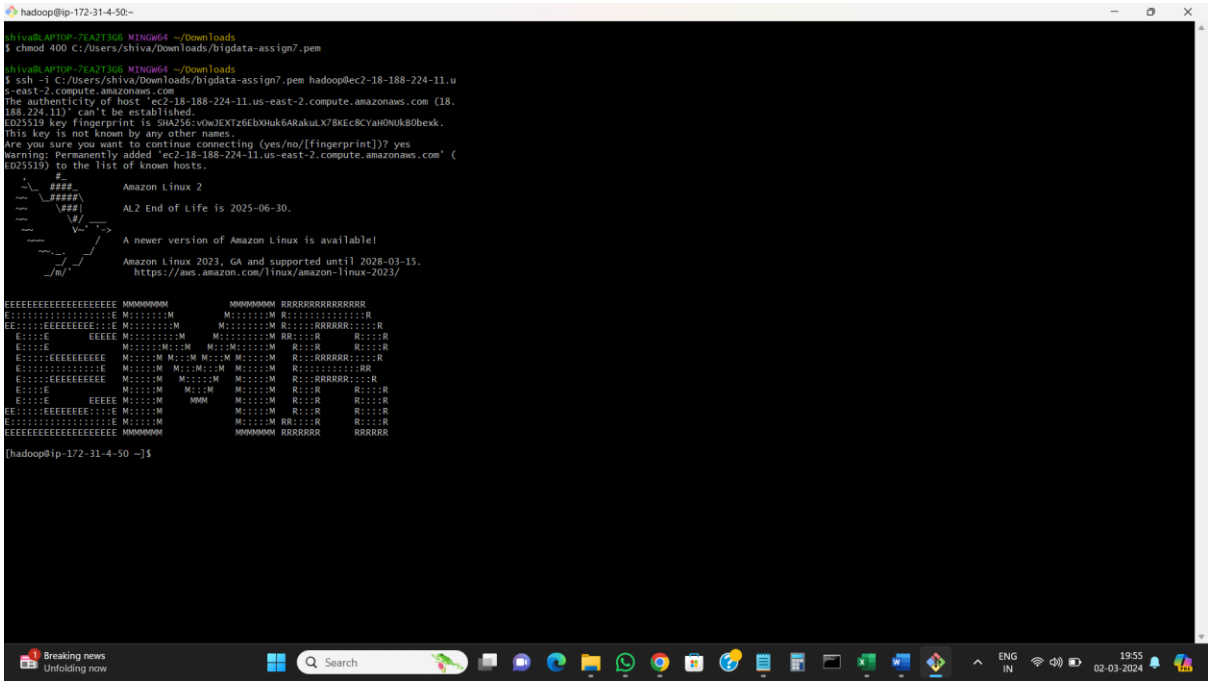
Exercise 1)

Step A

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

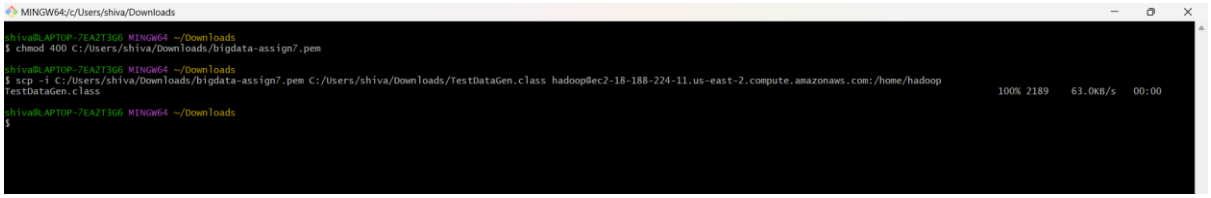






Step B

Use the TestDataGen program from previous assignments to generate new data files. Copy both generated files to the HDFS directory “/user/hadoop”



```
[hadoop@ip-172-31-4-50 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-4-50 ~]$ java TestDataGen
Magic Number = 67831
[hadoop@ip-172-31-4-50 ~]$
```

```
[hadoop@ip-172-31-4-50 ~]$ ls
foodplaces67831.txt  foodratings67831.txt  TestDataGen.class
[hadoop@ip-172-31-4-50 ~]$
```

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 the following commands:

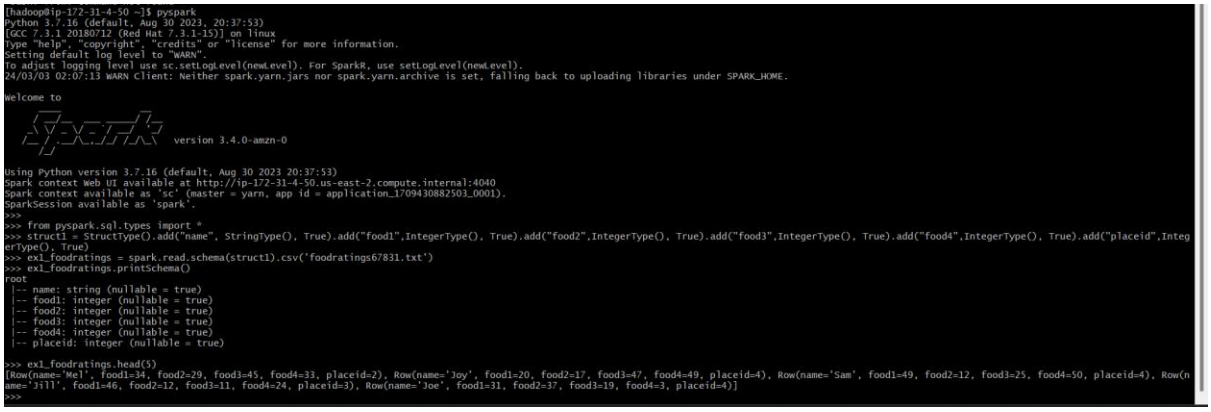
foodratings.printSchema()

foodratings.show(5)

$ hadoop fs -copyFromLocal /home/hadoop/foodratings67831.txt

>>> 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)
>>> ex1_foodratings = spark.read.schema(struct1).csv('foodratings67831.txt')
>>> ex1_foodratings.printSchema()
>>> ex1_foodratings.head(5)

```
[hadoop@ip-172-31-4-50 ~]$ hadoop fs -copyFromLocal /home/hadoop/foodratings67831.txt
[hadoop@ip-172-31-4-50 ~]$
```



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:

foodplaces.printSchema()

foodplaces.show(5)

$ hadoop fs -copyFromLocal /home/hadoop/foodratings120912.txt

$ hadoop fs -ls

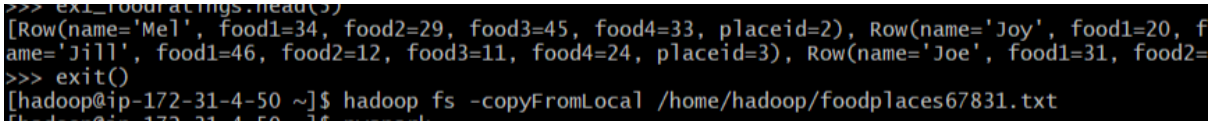
>>> from pyspark.sql.types import *

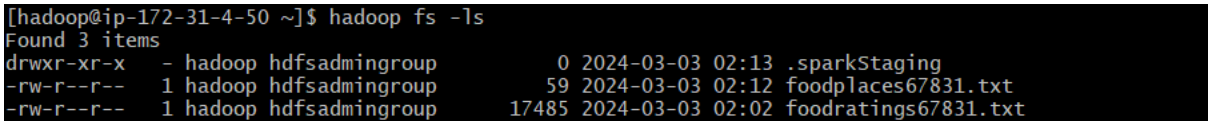
>>> struct1 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), True)

>>> foodplaces = spark.read.schema(struct1).csv('foodplaces120912.txt')

>>> foodplaces.printSchema()

>>> foodplaces.head(5)





```
>>> from pyspark.sql.types import *
>>> struct1 = StructType().add("placeid", IntegerType(), True).add("placename", StringType(), True)
>>> foodplaces = spark.read.schema(struct1).csv('foodplaces67831.txt')
>>> 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')]
>>>
```

Exercise 3)

Step A

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

>>> ex1_foodratings.createOrReplaceTempView("foodratingsT")
>>> foodplaces.createOrReplaceTempView("foodplacesT")

```
>>> ex1_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.printSchema()

foodratings_ex3a.show(5)

>>> foodratings_ex3 = spark.sql("SELECT * from foodratingsT where food2 < 25 and food4 > 40")
>>> foodratings_ex3.printSchema()

```
>>>
>>> foodratings_ex3 = spark.sql("SELECT * from foodratingsT where food2 < 25 and food4 > 40")
>>> foodratings_ex3.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)

>>>
```

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.printSchema()

foodplaces_ex3b.show(5)

>>> foodplaces_ex3 = spark.sql("SELECT * from foodplacesT where placeid> 3")
>>> foodplaces_ex3.printSchema()

```
>>> foodplaces_ex3 = spark.sql("SELECT * from foodplacesT where placeid> 3")
>>> foodplaces_ex3.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>>
```

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.printSchema()

foodratings_ex4.show(5)

>>> foodratings_ex4 = ex1_foodratings.filter(ex1_foodratings.name == "Mel").filter(ex1_foodratings.food3 < 25)
>>> foodratings_ex4.printSchema()
>>> foodratings_ex4.head(5)

```
>>> foodratings_ex4 = ex1_foodratings.filter(ex1_foodratings.name == "Mel").filter(ex1_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=21, food2=4, food3=10, food4=39, placeid=2), Row(name='Mel', food1=18, food2=32, food3=7, food4=4, placeid=1), Row(name='Mel', food1=21, food2=45, food3=22, food4=17, placeid=2), Row(name
='Mel', food1=28, food2=10, food3=3, food4=43, placeid=5), Row(name='Mel', food1=45, food2=48, food3=12, food4=40, placeid=1)]
>>>
```

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.printSchema()

foodratings_ex5.show(5)

>>> foodratings_ex5 = ex1_foodratings.select(ex1_foodratings.name, ex1_foodratings.placeid)
>>> foodratings_ex5.printSchema()
>>> foodratings_ex5.head(5)

```
>>> foodratings_ex5 = ex1_foodratings.select(ex1_foodratings.name, ex1_foodratings.placeid)
>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex5.head(5)
[Row(name='Mel', placeid=2), Row(name='Joy', placeid=4), Row(name='Sam', placeid=4), Row(name='Jill', placeid=3), Row(name='Joe', placeid=4)]
>>>
```

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.printSchema()

ex6.show(5)

>>> ex6 = ex1_foodratings.join(foodplaces, ex1_foodratings.placeid == foodplaces.placeid, "inner").drop(ex1_foodratings.placeid)
>>> ex6.printSchema()
>>> ex6.head(5)

```
>>> ex6 = ex1_foodratings.join(foodplaces, ex1_foodratings.placeid == foodplaces.placeid, "inner").drop(ex1_foodratings.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.head(5)
[Row(name='Mel', food1=34, food2=29, food3=45, food4=33, placeid=2, placename='Atlantic'), Row(name='Joy', food1=20, food2=17, food3=47, food4=49, placeid=4, placename="Jake's"), Row(name='Sam', food1=49, food2
=12, food3=25, food4=50, placeid=4, placename="Jake's"), Row(name='Jill', food1=46, food2=12, food3=11, food4=24, placeid=3, placename='Food Town'), Row(name='Joe', food1=31, food2=37, food3=19, food4=3, placei
d=4, placename="Jake's")]
>>>
```