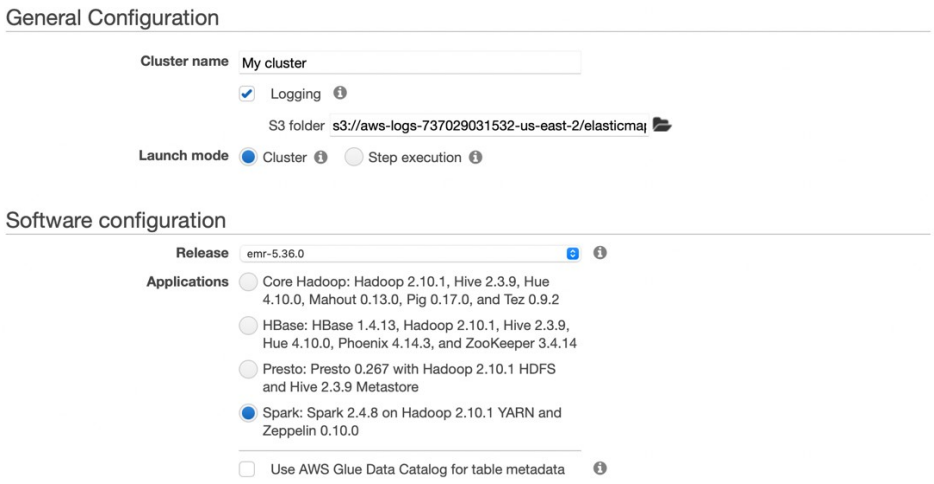
CSP 554 BIG DATA TECHNOLOGIES
MOHAMMED WASIM R D(A20497053)

ASSIGNMENT 7

Step 1 : Assigning the key to private and connecting it to EMR with Spark.



chmod 400 emr-key-pair.pem

ssh -i emr-key-pair.pem hadoop@ec2-54-205-44-5.compute-1.amazonaws.com

```
Last login: Sat Oct 22 21:56:49 on ttys001
(base) mohammedwasimrd@Mohammeds-Air ~ % cd Downloads
(base) mohammedwasimrd@Mohammeds-Air Downloads % chmod 400 keypair.pem
(base) mohammedwasimrd@Mohammeds-Air Downloads % ssh -i keypair.pem hadoop@ec2-3-145-127-254.us-east-2.compute.amazonaws.com

Last login: Sun Oct 23 02:56:59 2022 from 207.237.204.67

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

https://aws.amazon.com/amazon-linux-2/
30 package(s) needed for security, out of 35 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEE 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
EEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR
```

Step 2 : Copy the TestDataGen.class that used in Assignment-4 and run the Java TestDataGen

scp -i emr-key-pair.pem TestDataGen.class hadoop@ec2-3-145-127-254.us-east-2.compute-1.amazonaws.com:/home/hadoop

```
Last login: Sun Oct 23 19:11:18 on ttys000
(base) mohammedwasimrd@Mohammeds-Air ~ % cd Downloads
(base) mohammedwasimrd@Mohammeds-Air Downloads % scp -i keypair-2.pem TestDataGen.class hadoop@ec2-18-191-96-196.us-east-2.compute.amazonaws.com:/home/hadoop

TestDataGen.class                             100% 2189    56.4KB/s   00:00
(base) mohammedwasimrd@Mohammeds-Air Downloads %
```

Magic Number : 76376

```
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -ls /user
Found 5 items
drwxrwxrwx   - hadoop   hdfsadmingroup          0 2022-10-23 03:39 /user/hadoop
drwxrwxrwx   - livy     livy                    0 2022-10-23 03:39 /user/livy
drwxrwxrwx   - root     hdfsadmingroup          0 2022-10-23 03:39 /user/root
drwxrwxrwx   - spark    spark                   0 2022-10-23 03:39 /user/spark
drwxrwxrwx   - zeppelin hdfsadmingroup          0 2022-10-23 03:39 /user/zeppelin
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -mkdir /user/csp554
[hadoop@ip-172-31-21-114 ~]$ java TestDataGen
Magic Number = 76376
[hadoop@ip-172-31-21-114 ~]$
```

```
[hadoop@ip-172-31-21-114 ~]$ ls
foodplaces76376.txt  foodratings76376.txt  hql.zip  TestDataGen.class
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -put foodratings76376.txt/user/csp554
put: `foodratings76376.txt/user/csp554': No such file or directory
[hadoop@ip-172-31-21-114 ~]$
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -put foodratings76376.txt /user/csp554

[hadoop@ip-172-31-21-114 ~]$
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -put foodplaces76376.txt /user/csp554
[hadoop@ip-172-31-21-114 ~]$ hadoop fs -ls /user/csp554
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup         59 2022-10-23 03:54 /user/csp554/foodplaces76376.txt
-rw-r--r--   1 hadoop hdfsadmingroup      17488 2022-10-23 03:54 /user/csp554/foodratings76376.txt
```

Step 3:
Exercise 1

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

```
[hadoop@ip-172-31-21-114 ~]$ 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).
22/10/23 03:56:56 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'.
>>>
```

```
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)
foodratings=spark.read.schema(tab1).csv('hdfs:///user/csp554/foodratings76376.txt')

foodratings.printSchema()

foodratings.show(5)
```

```
>>> 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)
>>> foodratings=spark.read.schema(tab1).csv('hdfs:///user/csp554/foodratings76376.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|
+----+-----+-----+-----+-----+-------+
| Sam|   33|   26|   38|    8|      2|
| Mel|    3|   15|   17|   39|      3|
| Sam|   22|   14|   36|   14|      2|
| Sam|   27|   16|   50|   47|      4|
| Joy|    8|   24|    1|   11|      1|
+----+-----+-----+-----+-----+-------+
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:

tab2=StructType().add("placeid",IntegerType(),True).add("placename", StringType(),True)
foodplaces=spark.read.schema(tab2).csv('hdfs:///user/csp554/foodplaces76376.txt')

foodplaces.printSchema()
foodplaces.show(5)

```
>>> tab2=StructType().add("placeid",IntegerType(),True).add("placename", StringType(),True)
>>> foodplaces=spark.read.schema(tab2).csv('hdfs:///user/csp554/foodplaces76376.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")

```
>>> 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.

foodratings_ex3a=spark.sql("select * from foodratingsT where food2<25 and food4>40")
foodratings_ex3a.printSchema()
foodratings_ex3a.show(5)

```
>>> foodratings_ex3 = spark.sql("SELECT * from foodratingsT where food2 < 25 and food4 > 40")
22/10/23 04:30:17 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
22/10/23 04:30:18 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
22/10/23 04:30:19 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
>>> 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)

>>>
```

```
>>> foodratings_ex3.show(5)
+----+-----+-----+-----+-----+-------+
|name|food1|food2|food3|food4|placeid|
+----+-----+-----+-----+-----+-------+
| Mel|   42|   19|   35|   42|      3|
| Sam|   46|    1|    7|   45|      5|
| Sam|   50|   21|   48|   48|      3|
| Joy|   47|    2|    2|   49|      3|
| Joy|   31|   20|   33|   50|      1|
+----+-----+-----+-----+-----+-------+
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.

foodplaces_ex3 = spark.sql("SELECT * from foodplacesT where placeid> 3")
foodplaces_ex3.printSchema()
foodplaces_ex3.show(5)

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

>>>
```

```
>>> foodplaces_ex3.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.printSchema()
foodratings_ex4.show(5)

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|    3|   15|   17|   39|      3|
| Mel|   17|    5|   21|   34|      4|
| Mel|   38|   33|   13|   18|      4|
| Mel|   34|   32|   15|    3|      5|
| Mel|   43|   26|   11|   38|      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.printSchema()
foodratings_ex5.show(5)

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|
+----+-------+
| Sam|      2|
| Mel|      3|
| Sam|      2|
| Sam|      4|
| Joy|      1|
+----+-------+
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.printSchema()
ex6.show(5)

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(foodratings.placeid)
>>> foodratings_ex6.printSchema()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'foodratings_ex6' is not defined
>>> 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|
+----+-----+-----+-----+-----+-------+-------------+
| Sam|   33|   26|   38|    8|      2|    Atlantic|
| Mel|    3|   15|   17|   39|      3|   Food Town|
| Sam|   22|   14|   36|   14|      2|    Atlantic|
| Sam|   27|   16|   50|   47|      4|     Jake's|
| Joy|    8|   24|    1|   11|      1|China Bistro|
+----+-----+-----+-----+-----+-------+-------------+
only showing top 5 rows

>>>
```