

## Assignment-7

## Exercise 1

## Generating Magic Number

```
[hadoop@ip-172-31-58-111 ~]$ ls
[hadoop@ip-172-31-58-111 ~]$ ls
TestDataGen.class
[hadoop@ip-172-31-58-111 ~]$ java TestDataGen
Magic Number = 34674
[hadoop@ip-172-31-58-111 ~]$ ls
foodplaces34674.txt foodratings34674.txt TestDataGen.class
[hadoop@ip-172-31-58-111 ~]$ |
```

## 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-172-31-58-111 ~]$ hdfs dfs -copyFromLocal foodratings34674.txt /user/hadoop/foodratings34674.csv
[hadoop@ip-172-31-58-111 ~]$ hdfs dfs -copyFromLocal foodplaces34674.txt /user/hadoop/foodplaces34674.csv
[hadoop@ip-172-31-58-111 ~]$ |
```

## 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

```
Using Python version 3.7.10 (default, Jun 3 2021 00:02:01)
SparkSession available as 'spark'.
>>> from pyspark.sql.types import *
>>> struct1 = 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.schema(struct1).csv('/user/hadoop/foodratings34674.csv')
>>> |
```

Magic Number = 34674

foodratings.printSchema()

```
>>> 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)

```
>>> foodratings.show(5)
+-----+-----+-----+-----+-----+
|name|food1|food2|food3|food4|placeid|
+-----+-----+-----+-----+-----+
| Sam|    5|   35|    1|    2|     2|
| Sam|   21|   29|    4|   25|     5|
| Joy|   21|   47|   23|   38|     4|
| Sam|   29|   46|   20|   26|     3|
| Sam|   39|   21|   36|   50|     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:

Field Name	Field Type
placeid	Integer
placename	String

```
>>> struct2 = StructType().add("placeid", IntegerType(), True).add("placename",
StringType(), True)
>>> foodplaces = spark.read.schema(struct2).csv('/user/hadoop/foodplaces34674.csv')
>>> |
```

Magic Number = 34674

foodplaces.printSchema()

```
>>> foodplaces.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)
>>> |
```

foodplaces.show(5)

```
>>> 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")
>>> |
```

#### 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_ex3a = spark.sql("SELECT * FROM foodratingsT WHERE food2 < 25 AND food4 > 40")
22/10/24 04:45:09 WARN ObjectStore: Version information not found in metastore.
hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
22/10/24 04:45:09 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
22/10/24 04:45:09 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
>>> |
```

```
foodratings_ex3a.printSchema()
```

```
>>> 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)
```

```
>>> foodratings_ex3a.show(5)
+-----+-----+-----+-----+-----+
|name|food1|food2|food3|food4|placeid|
+-----+-----+-----+-----+-----+
| Sam|   39|   21|   36|   50|     1|
| Mel|   11|   18|   10|   44|     4|
| Mel|   30|   16|   21|   50|     4|
| Joe|   16|    7|   44|   46|     4|
| Joy|   50|   18|   46|   50|     5|
+-----+-----+-----+-----+-----+
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.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|
+-----+-----+

>>> |
```

#### 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") & (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)
```

```
>>> foodratings_ex4.show(5)
+-----+-----+-----+-----+-----+
|name|food1|food2|food3|food4|placeid|
+-----+-----+-----+-----+-----+
| Mel|   11|   18|   10|   44|      4|
| Mel|   39|   10|   12|   28|      5|
| Mel|   30|   16|   21|   50|      4|
| Mel|   26|   12|    5|   17|      5|
| Mel|   15|   50|   22|   11|      1|
+-----+-----+-----+-----+-----+
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()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)
>>> |
```

```
foodratings_ex5.show(5)
```

```
>>> foodratings_ex5.show(5)
+-----+-----+
|name|placeid|
+-----+-----+
| Sam|      2|
| Sam|      5|
| Joy|      4|
| Sam|      3|
| Sam|      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')
>>> 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.show(5)
```

```
>>> ex6.show(5)
+-----+-----+-----+-----+-----+-----+-----+
|name|food1|food2|food3|food4|placeid|placeid| placename|
+-----+-----+-----+-----+-----+-----+-----+
| Sam| 5| 35| 1| 2| 2| 2| Atlantic|
| Sam| 21| 29| 4| 25| 5| 5| Soup Bowl|
| Joy| 21| 47| 23| 38| 4| 4| Jake's|
| Sam| 29| 46| 20| 26| 3| 3| Food Town|
| Sam| 39| 21| 36| 50| 1| 1|China Bistro|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

>>> |
```

Submitted By: -

Aastha Dhir

CWID-A20468022

adhir2@hawk.iit.edu