Assignment – Day 13

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

20/11/2024 (Wednesday)

Transaction & Action Practice:-

1. **Creating RDDs and DataFrame in PySpark with a Schema: -**

```
#to create rdds and  dataframe

from pyspark import SparkContext
from pyspark.sql import  SparkSession

sc =SparkContext.getOrCreate()
spark = SparkSession.builder.appName('pyspark first program').getOrCreate()

#create the rdd

rdd = sc.parallelize([('C',85,76,87,91), ('B',85,76,87,91), ("A", 85,78,96,92), ("A",
92,76,89,96)], 4)
mydata = ['Division','English','Mathematics','Physics','Chemistry']
marks_df = spark.createDataFrame(rdd, schema=mydata)
print(rdd.collect())
print(rdd) #---Transformation which gives rdd value

rdd.collect() #----Action gives non rdd value
```

```
▸ (3) Spark Jobs
▸ marks_df: pyspark.sql.dataframe.DataFrame = [Division: string, English: long ... 3 more fields]
[('C', 85, 76, 87, 91), ('B', 85, 76, 87, 91), ('A', 85, 78, 96, 92), ('A', 92, 76, 89, 96)]
ParallelCollectionRDD[181] at readRDDFromInputStream at PythonRDD.scala:435
Out[14]: [('C', 85, 76, 87, 91),
 ('B', 85, 76, 87, 91),
 ('A', 85, 78, 96, 92),
 ('A', 92, 76, 89, 96)]
```

2. **Creating RDDs, Converting to DataFrame, and Performing Actions in PySpark: -**

```
#to create rdds and  dataframe


from pyspark import SparkContext
```

```
from pyspark.sql import  SparkSession

sc =SparkContext.getOrCreate()
spark = SparkSession.builder.appName('pyspark first program').getOrCreate()

#create the rdd

rdd = sc.parallelize([('C',85,76,87,91), ('B',85,76,87,91), ("A", 85,78,96,92), ("A",
92,76,89,96)], 4)
mydata = ['Division','English','Mathematics','Physics','Chemistry']
marks_df = spark.createDataFrame(rdd, schema=mydata)
print(rdd.count())

rdd.take(2) ##Action gives non rdd value
```

```
▸ (4) Spark Jobs
  ▸ marks_df:  pyspark.sql.dataframe.DataFrame = [Division: string, English: long ... 3 more fields]
4
Out[13]: [('C', 85, 76, 87, 91), ('B', 85, 76, 87, 91)]
```

---

3. Counting the Number of Elements in an RDD Using PySpark: -

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.count())
count_rdd.count()
```

```
▸ (2) Spark Jobs
10
Out[3]: 10
```

---

4. Using count() and first() Actions to Analyze an RDD in PySpark: -

```
from pyspark import SparkContext

sc = SparkContext.getOrCreate()

first_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])

print(first_rdd.count())

first_rdd.count()

first_rdd.first() #First method is action
```

```
▸ (3) Spark Jobs
10
Out[4]: 1
```

---

5. Filtering Elements of an RDD Based on a Condition in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
filter_rdd_2 = sc.parallelize(['Rahul', 'Sarthak', 'Rohan',
'Shreya', 'Priya'])
print(filter_rdd_2.filter(lambda x: x.startswith('S')).collect())
```

```
▸ (1) Spark Jobs
['Sarthak', 'Shreya']
```

---

6. Applying Filters and Performing Union Operation on RDDs in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
uninon_inp = sc.parallelize([2,4,5,6,7,8,9,10])
uninon_rdd_1 = uninon_inp.filter(lambda x:x % 2 == 0)
uninon_rdd_2 = uninon_inp.filter(lambda x:x % 3 == 0)
print(uninon_rdd_1.union(uninon_rdd_2).collect())
```

```
▸ (1) Spark Jobs
[2, 4, 6, 8, 10, 6, 9]
```

---

7. Using flatMap() Transformation to Split and Flatten Data in an RDD in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
flatmap_rdd = sc.parallelize(["Hey there", "This is PySpark RDD
Transformations"])
print(flatmap_rdd.flatMap(lambda x :x.split(" ").collect()))
flatmap_rdd.flatMap(lambda x :x.split(" ").collect())
```

```
PythonRDD[217] at RDD at PythonRDD.scala:58
Out[17]: PythonRDD[218] at RDD at PythonRDD.scala:58
```

---

8. Using flatMap() to Split and Flatten Strings into Words in an RDD in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
flatmap_rdd = sc.parallelize(["Hey there", "This is PySpark RDD
Transformations"])
(flatmap_rdd.flatMap(lambda x: x.split(" ")).collect())
```

```
▸ (1) Spark Jobs
Out[18]: ['Hey', 'there', 'This', 'is', 'PySpark', 'RDD', 'Transformations']
```

---

9. Using reduce() Action to Aggregate Elements in an RDD in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
reduce_rdd = sc.parallelize([1,2,3])
print(reduce_rdd.reduce(lambda x,y:x+y))
```

```
▸ (1) Spark Jobs
6
```

---

10. Saving an RDD to a Text File Using saveAsTextFile() in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
save_rdd = sc.parallelize([1,2,3,4,5,5])
save_rdd.saveAsTextFile('file3.txt')
```

```
▸ (1) Spark Jobs
```

---

11. Using map() Transformation to Modify Elements in an RDD in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
map_rdd = sc.parallelize([1,2,3])
print(map_rdd.map(lambda x:x+10))
print(map_rdd.map(lambda x:x+10).collect())
```

```
▸ (1) Spark Jobs
PythonRDD[147] at RDD at PythonRDD.scala:58
[11, 12, 13]
```

---

12. Using filter() Transformation to Select Even Numbers from an RDD in PySpark

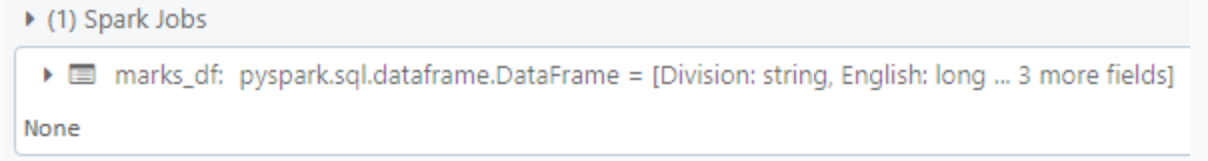
```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
filter_rdd = sc.parallelize([1,2,3])
print(filter_rdd.filter(lambda x:x%2==0))
print(filter_rdd.filter(lambda x:x%2==0).collect())
```

```
▸ (1) Spark Jobs
PythonRDD[150] at RDD at PythonRDD.scala:58
[2]
```

---

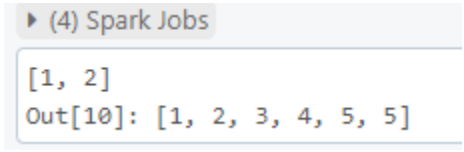
13. Creating an RDD, Converting it to a DataFrame, and Registering it as a Temp View in PySpark

```
from pyspark import SparkContext
from pyspark.sql import  SparkSession
sc =SparkContext.getOrCreate()
spark = SparkSession.builder.appName('pyspark first
program').getOrCreate()

#create the rdd
rdd = sc.parallelize([('C',85,76,87,91), ('B',85,76,87,91), ("A",
85,78,96,92), ("A", 92,76,89,96)], 4)
mydata = ['Division','English','Mathematics','Physics','Chemistry']
marks_df = spark.createDataFrame(rdd, schema=mydata)
print(marks_df.createOrReplaceTempView("dataofmarks"))
```



14. Using take() Action to Retrieve the First N Elements from an RDD in PySpark

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.take(2))
count_rdd.take(6)
```



Summary of Transaction & Action:-

1. **Creating RDDs:**

- RDDs are created using sc.parallelize() with sample data.

2. **Creating DataFrames:**

- RDDs are converted to DataFrames using spark.createDataFrame() with a predefined schema.

3. **RDD Transformations:**

- map(): Applies a function to each element of the RDD.
- filter(): Filters elements based on a condition.
- flatMap(): Flattens the results of applied functions (e.g., splitting strings into words).
- union(): Combines two RDDs into one.

4. **RDD Actions:**

    - count(): Counts the number of elements in an RDD.
    - first(): Returns the first element of the RDD.
    - reduce(): Reduces the RDD to a single value (e.g., summing elements).
    - collect(): Collects all elements from the RDD to the driver.
    - saveAsTextFile(): Saves the RDD to a text file.

5. **DataFrame Operations:**

- A DataFrame can be registered as a temporary SQL view using createOrReplaceTempView(), enabling SQL queries on the data.

6. **File Operations:**

- The code includes file I/O operations like saving an RDD to a text file using saveAsTextFile().

7. **Lazy and Eager Execution:**

- Transformations are lazy (not executed immediately), while actions are eager (trigger computation and return results).

Assignment – Day 13

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

20/11/2024 (Wednesday)

Summary of View in Spark:-

1. **Spark Session Creation**:

- The SparkSession is created using .builder.appName("SparkByExamples.com").enableHiveSupport().getOrCreate(). This initializes a Spark session that can interact with Hive if needed.

2. **Data and Schema Setup**:

- A list of tuples (data) is created, containing sample personal information such as first name, last name, country, and state.
- A list of column names (columns) is defined: "firstname", "lastname", "country", and "state".

3. **DataFrame Creation**:

- The data is converted into a DataFrame (sampleDF) by using spark.sparkContext.parallelize(data).toDF(columns). The parallelize function distributes the data across the Spark cluster, and toDF(columns) converts the list of data into a structured DataFrame with specified columns.

4. **Creating Temporary Views**:

- The sampleDF DataFrame is registered as two temporary SQL views: "Person" and "mydata", using createOrReplaceTempView(). These views allow Spark SQL queries to be executed against the DataFrame.

5. **Displaying Data**:

- sampleDF.show() is used to display the contents of the DataFrame in a tabular format.
- spark.sql("select * from person").show() and spark.sql("select * from mydata").show() run SQL queries on the two views and display the same data since both views are based on the same sampleDF DataFrame.

Views Practice: -

1. **Creating a Spark DataFrame and registering it as temporary SQL views for querying.**

```
from pyspark.sql import SparkSession
# Create spark session
spark = SparkSession \
.builder \
.appName("SparkByExamples.com") \
.enableHiveSupport() \
.getOrCreate()
data = [("Sarthak","Kulkarni","IND","MH"),
("Lakshita","Sathe","IND","MP"),
("Harsh","Choudhari","USA","COL"),
("Pratik","Pathak","IRE","DUB")]
columns = ["firstname","lastname","country","state"]
# Create dataframe
sampleDF = spark.sparkContext.parallelize(data).toDF(columns)
sampleDF.createOrReplaceTempView("Person")
sampleDF.createOrReplaceTempView("mydata")
sampleDF.show()
```

```
▸ (5) Spark Jobs
▸ sampleDF: pyspark.sql.dataframe.DataFrame = [firstname: string, lastname: string ... 2 more fields]
+---------+---------+-------+-----+
|firstname| lastname|country|state|
+---------+---------+-------+-----+
|  Sarthak| Kulkarni|    IND|   MH|
| Lakshita|    Sathe|    IND|   MP|
|    Harsh|Choudhari|    USA|  COL|
|   Pratik|   Pathak|    IRE|  DUB|
+---------+---------+-------+-----+
```

---

2. Executing SQL queries on temporary views in Spark to display data

```
spark.sql("select * from person").show()
spark.sql("select * from mydata").show()
```

▸ (6) Spark Jobs

```
+---------+---------+-------+-----+
|firstname| lastname|country|state|
+---------+---------+-------+-----+
|  Sarthak| Kulkarni|    IND|   MH|
| Lakshita|    Sathe|    IND|   MP|
|    Harsh|Choudhari|    USA|  COL|
|   Pratik|   Pathak|    IRE|  DUB|
+---------+---------+-------+-----+

+---------+---------+-------+-----+
|firstname| lastname|country|state|
+---------+---------+-------+-----+
|  Sarthak| Kulkarni|    IND|   MH|
| Lakshita|    Sathe|    IND|   MP|
|    Harsh|Choudhari|    USA|  COL|
|   Pratik|   Pathak|    IRE|  DUB|
+---------+---------+-------+-----+
```

Assignment – Day 13

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

20/11/2024 (Wednesday)

Summary of Modifying DataFrames in PySpark:-

1. **Spark Session Creation:**

- A SparkSession is created using .builder.appName('pyspark - example join').getOrCreate(), allowing the execution of PySpark commands.

2. **DataFrame Creation:**

- A list of tuples (data) is defined with sample information (names, dates of birth, gender, and salary), and this data is loaded into a DataFrame (df) with specified column names ("Name", "DOB", "Gender", "salary").

3. **Column Renaming:**

- The column "DOB" is renamed to "date of birth".
- The column "Name" is renamed to "personname", and the updated DataFrame is shown.

4. **Selecting and Renaming Columns Using selectExpr:**

- "Gender" is renamed to "category".
- "Name" is renamed to "name".
- The resulting DataFrame data is displayed.

5. **Using col() for Column Selection and Aliasing:**

- The select() function with col() is used to select columns explicitly and rename the "salary" column to "Amount" using the alias() function.
- The DataFrame with the renamed column (Amount) is displayed.

Modifying DataFrames in PySpark Practice: -

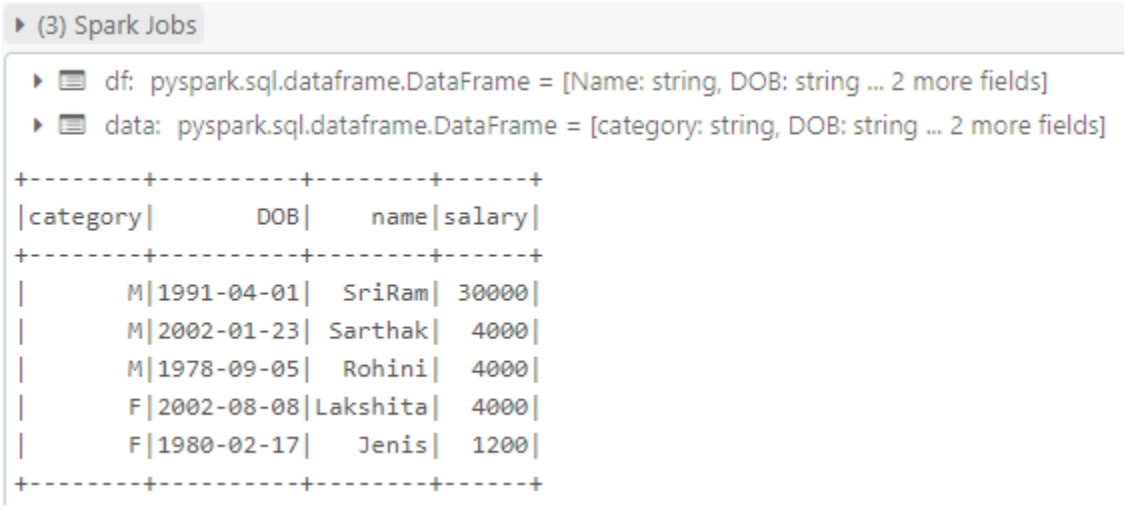
1. **Renaming columns in a PySpark DataFrame using withColumnRenamed.**

```
# Importing necessary libraries
from pyspark.sql import SparkSession

# Create a spark session
spark = SparkSession.builder.appName('pyspark - example
join').getOrCreate()

# Create data in dataframe
data = [(('SriRam'), '1991-04-01', 'M', 30000),
        (('Sarthak'), '2002-01-23', 'M', 4000),
        (('Rohini'), '1978-09-05', 'M', 4000),
        (('Lakshita'), '2002-08-08', 'F', 4000),
        (('Jenis'), '1980-02-17', 'F', 1200)]

# Column names in dataframe
columns = ["Name", "DOB", "Gender", "salary"]

# Create the spark dataframe
df = spark.createDataFrame(data=data,
                           schema=columns)
df.withColumnRenamed("DOB","date of birth").show()
df.withColumnRenamed("DOB","date of
birth").withColumnRenamed("Name","personname").show()
```

```
▸ (6) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]
+--------+-------------+------+------+
|    Name|date of birth|Gender|salary|
+--------+-------------+------+------+
|  SriRam|   1991-04-01|     M| 30000|
| Sarthak|   2002-01-23|     M|  4000|
|  Rohini|   1978-09-05|     M|  4000|
|Lakshita|   2002-08-08|     F|  4000|
|   Jenis|   1980-02-17|     F|  1200|
+--------+-------------+------+------+

+----------+-------------+------+------+
|personname|date of birth|Gender|salary|
+----------+-------------+------+------+
|    SriRam|   1991-04-01|     M| 30000|
|   Sarthak|   2002-01-23|     M|  4000|
|    Rohini|   1978-09-05|     M|  4000|
|  Lakshita|   2002-08-08|     F|  4000|
|     Jenis|   1980-02-17|     F|  1200|
+----------+-------------+------+------+
```

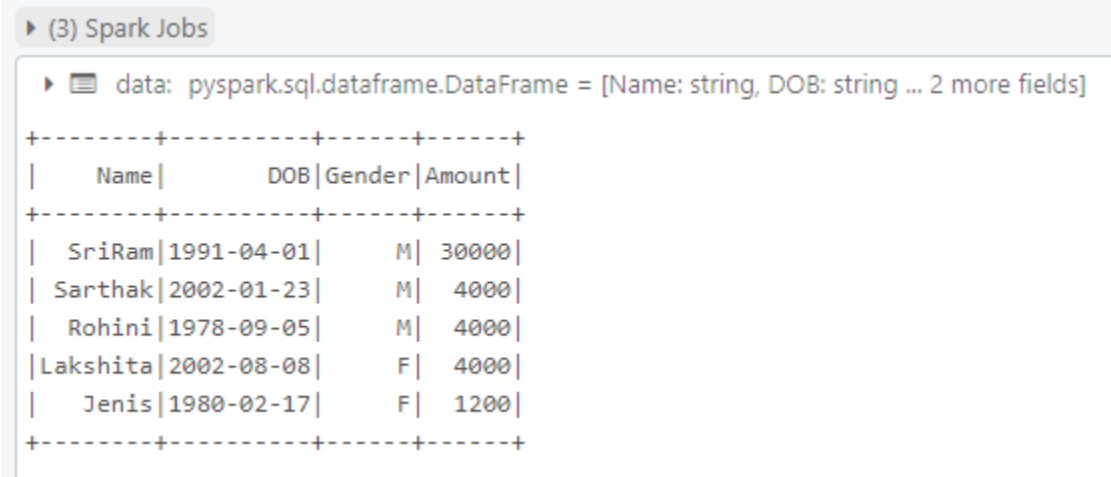
2. Selecting and renaming columns in a PySpark DataFrame using selectExpr.

```
# Importing necessary libraries using select exp
from pyspark.sql import SparkSession

# Create a spark session
spark = SparkSession.builder.appName('pyspark - example
join').getOrCreate()

# Create data in dataframe
data = [(('SriRam'), '1991-04-01', 'M', 30000),
        (('Sarthak'), '2002-01-23', 'M', 4000),
        (('Rohini'), '1978-09-05', 'M', 4000),
        (('Lakshita'), '2002-08-08', 'F', 4000),
        (('Jenis'), '1980-02-17', 'F', 1200)]

# Column names in dataframe
columns = ["Name", "DOB", "Gender", "salary"]

# Create the spark dataframe
df = spark.createDataFrame(data=data,
                           schema=columns)
data = df.selectExpr("Gender as category","DOB","Name as
name","salary")
data.show()
```



3. Selecting and aliasing columns in a PySpark DataFrame using col() and alias().

```
from pyspark.sql.functions import col

# Select the 'salary' as 'Amount' using aliasing
# Select remaining with their original name
data = df.select(col("Name"),col("DOB"),
                 col("Gender"),
                 col("salary").alias('Amount'))
data.show()
```



(3) Spark Jobs

data: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]

```
+--------+----------+------+------+
|    Name|       DOB|Gender|Amount|
+--------+----------+------+------+
|  SriRam|1991-04-01|     M| 30000|
| Sarthak|2002-01-23|     M|  4000|
|  Rohini|1978-09-05|     M|  4000|
|Lakshita|2002-08-08|     F|  4000|
|   Jenis|1980-02-17|     F|  1200|
+--------+----------+------+------+
```