DAY-13 (Spark Assignment - 3)

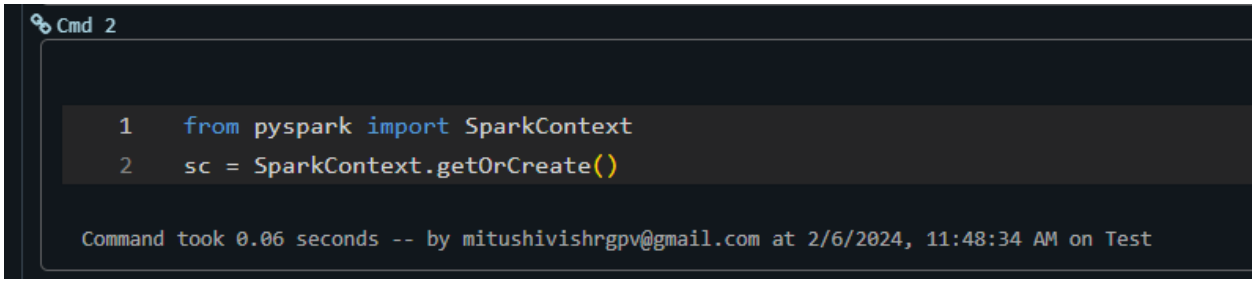
Mitushi Vishwakarma

RDDs

Actions and transformations in PySpark

There are two types of operations that can be performed on RDDs, they are Actions and Transformations. We will initialize a SparkContext to perform the operations. Only one SparkContext can run in the environment.

Initializing SparkContext



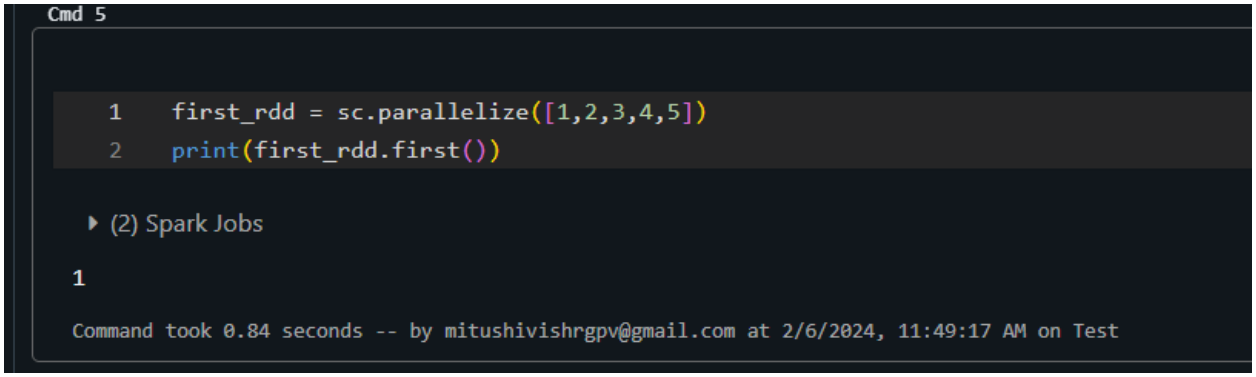
1. Actions :

They are a kind of operation which are applied on an RDD to produce a single value. These methods are applied on a resultant RDD and produces a non-RDD value

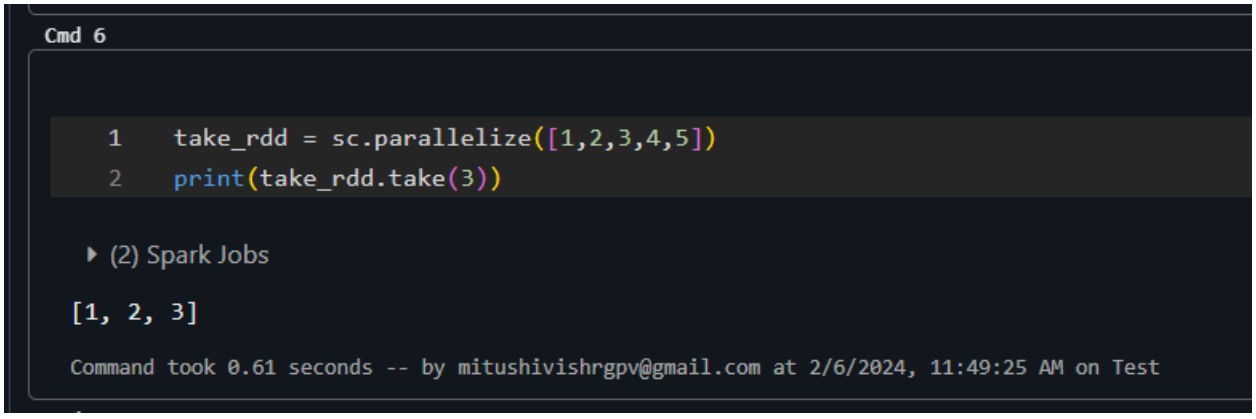
.collect() : returns a list of all the elements of the RDD

.count() : returns the number of elements of our RDD

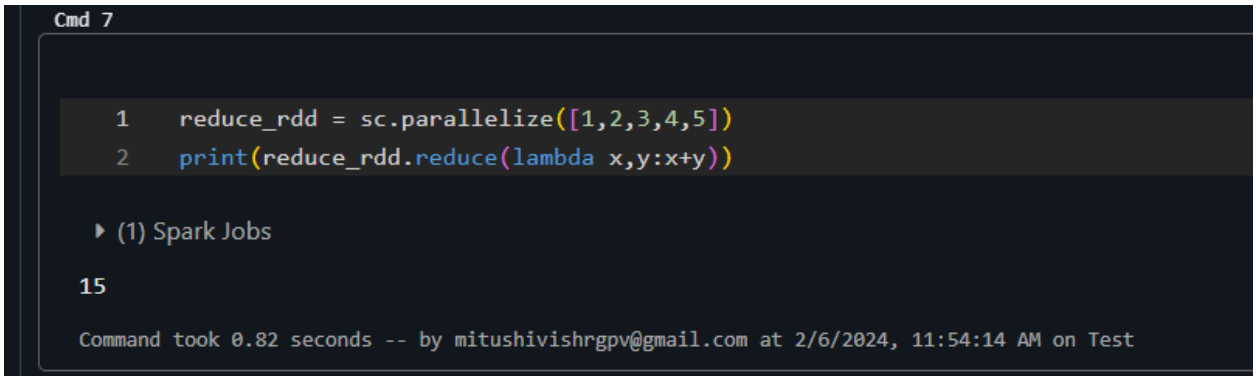
.first() : returns the first element from our RDD



.take(n) : returns n number of elements from the RDD



.reduce() : takes two elements from the given RDD and operates. This operation is performed using an anonymous function or lambda.



.saveAsTextFile() : is used to save the resultant RDD as a text file

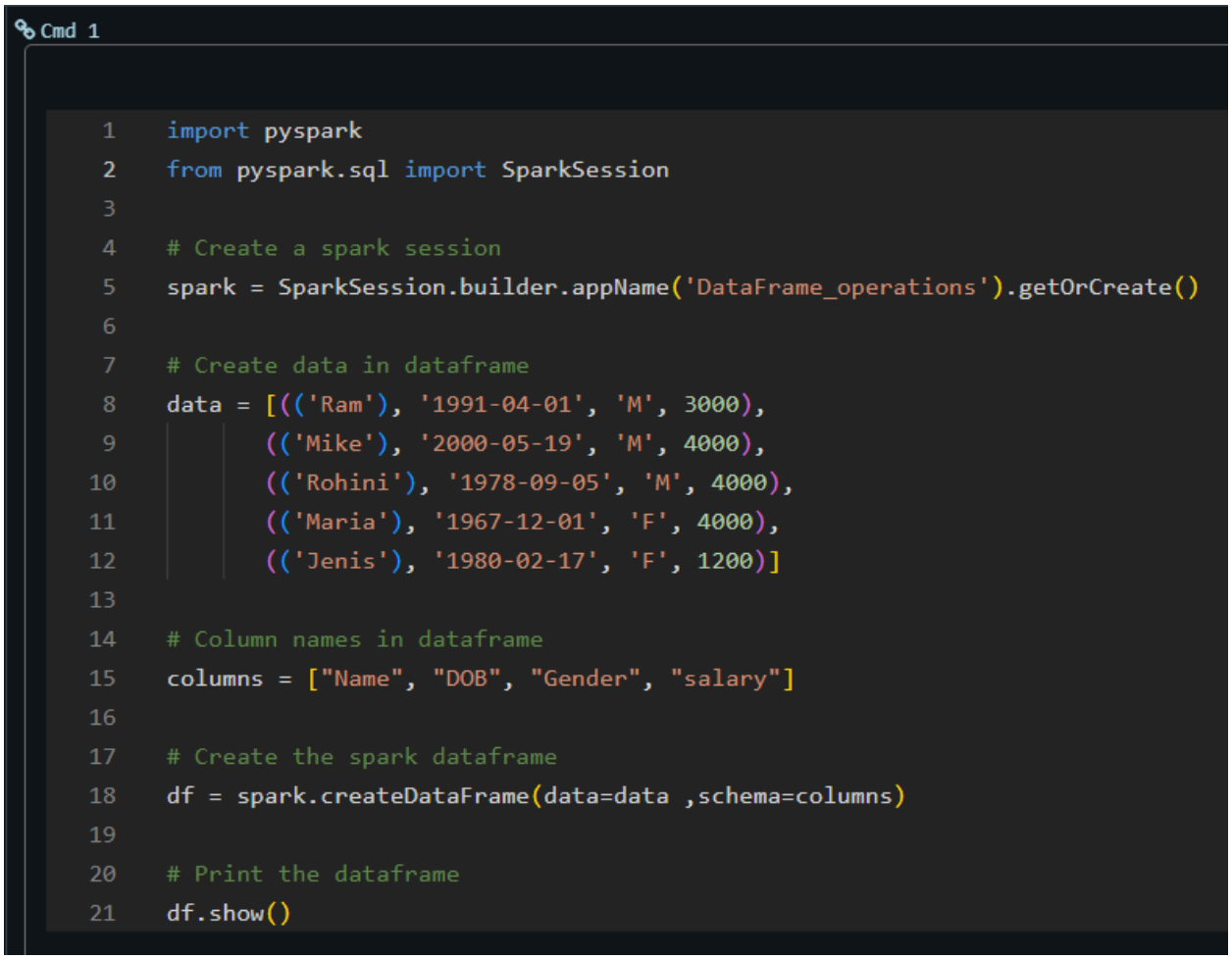
2. Transformations : Transformations are the kind of operations that are performed on an RDD and return a new RDD

.map() : The **.map()** transformation takes in an anonymous function and applies this function to each of the elements in the RDD.

.filter() : for filtering elements from a PySpark RDD. Takes in an anonymous function with a condition

.union() : combines two RDDs and returns the union of the input two RDDs

DataFrames in PySpark : PySpark DataFrames are distributed collections of data organized into named columns. You can create PySpark DataFrames from various data sources such as CSV, JSON, Parquet, or even existing RDDs.



```
▸ (3) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]
+------+----------+------+------+
|  Name|       DOB|Gender|salary|
+------+----------+------+------+
|   Ram|1991-04-01|     M|  3000|
|  Mike|2000-05-19|     M|  4000|
|Rohini|1978-09-05|     M|  4000|
| Maria|1967-12-01|     F|  4000|
| Jenis|1980-02-17|     F|  1200|
+------+----------+------+------+

Command took 4.51 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 3:29:55 PM on Test
```

Renaming Column in dataframe :

There are various methods to rename data in dataframe.

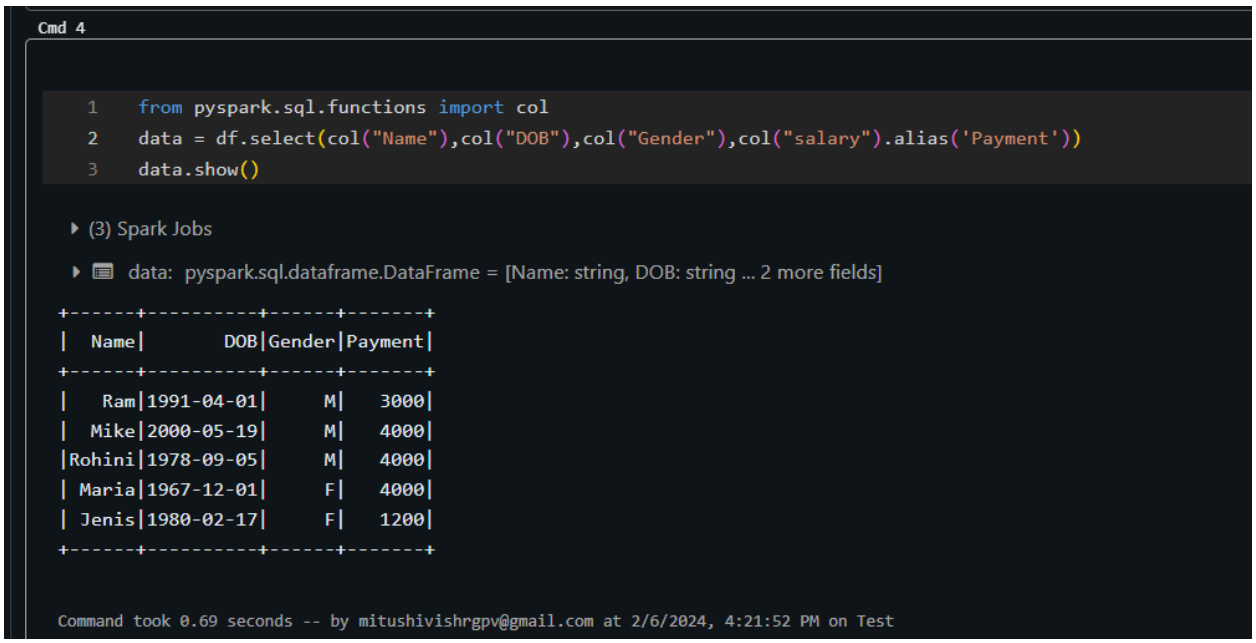
1. **Using withColumnRenamed(existing,new):** Here withColumnRenamed() method takes two argument, existingstr and newstr. Returns dataframe with new column name

```
Cmd 2

1  df_columnRename = df.withColumnRenamed('DOB','DateOfBirth')
2  df_columnRename.printSchema()
3  df_columnRename.show()

▸ (3) Spark Jobs
▸ df_columnRename: pyspark.sql.dataframe.DataFrame = [Name: string, DateOfBirth: string ... 2 more fields]
root
 |-- Name: string (nullable = true)
 |-- DateOfBirth: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+-----------+------+------+
|  Name|DateOfBirth|Gender|salary|
+------+-----------+------+------+
|   Ram| 1991-04-01|     M|  3000|
|  Mike| 2000-05-19|     M|  4000|
|Rohini| 1978-09-05|     M|  4000|
| Maria| 1967-12-01|     F|  4000|
| Jenis| 1980-02-17|     F|  1200|
+------+-----------+------+------+

Command took 0.74 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:19:05 PM on Test
```

Multiple column renaming using withColumnRenamed :

```
Cmd 3

1  df.withColumnRenamed('DOB','DateOfBirth').withColumnRenamed('salary',"Salary").show()

▸ (3) Spark Jobs

+------+-----------+------+------+
|  Name|DateOfBirth|Gender|Salary|
+------+-----------+------+------+
|   Ram| 1991-04-01|     M|  3000|
|  Mike| 2000-05-19|     M|  4000|
|Rohini| 1978-09-05|     M|  4000|
| Maria| 1967-12-01|     F|  4000|
| Jenis| 1980-02-17|     F|  1200|
+------+-----------+------+------+

Command took 0.94 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:04:05 PM on Test
```

2. Using select(cols) : **cols:** List of column names as strings. And new column name can be given using .alias(newstr) with that particular col name.

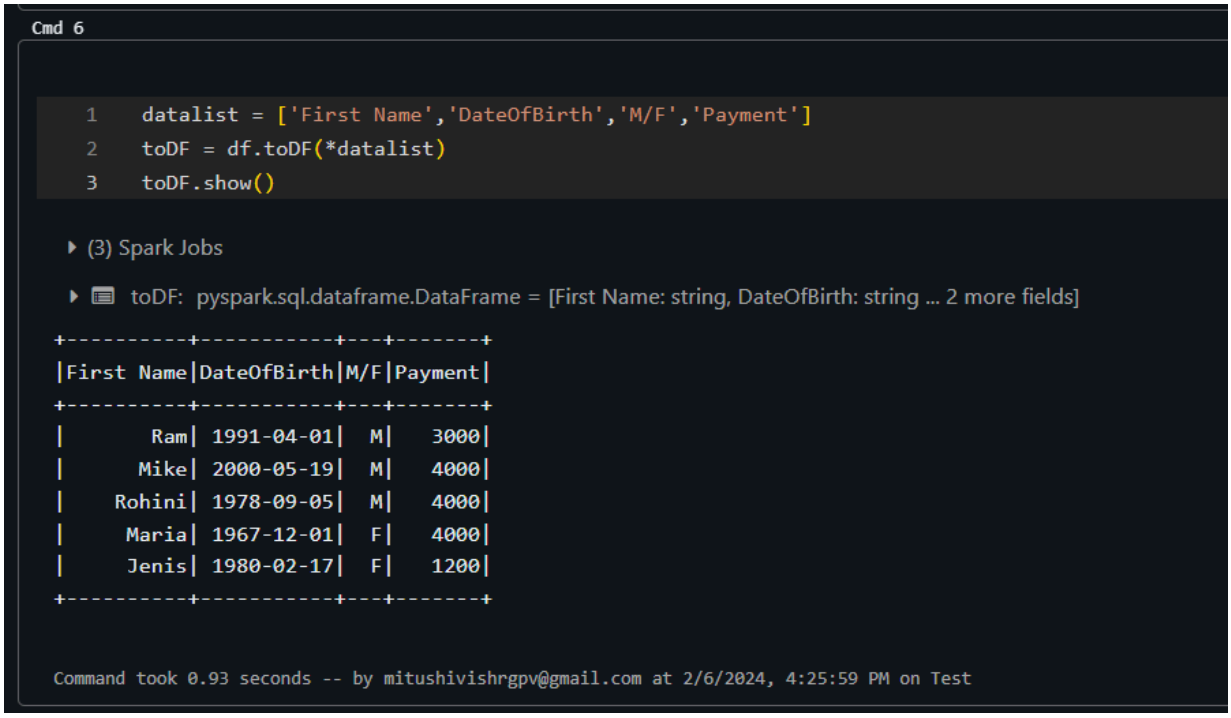


```
Cmd 4

1  from pyspark.sql.functions import col
2  data = df.select(col("Name"),col("DOB"),col("Gender"),col("salary").alias('Payment'))
3  data.show()

▸ (3) Spark Jobs

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

+------+----------+------+-------+
|  Name|       DOB|Gender|Payment|
+------+----------+------+-------+
|   Ram|1991-04-01|     M|   3000|
|  Mike|2000-05-19|     M|   4000|
|Rohini|1978-09-05|     M|   4000|
| Maria|1967-12-01|     F|   4000|
| Jenis|1980-02-17|     F|   1200|
+------+----------+------+-------+

Command took 0.69 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:21:52 PM on Test
```

4 . Using selectExpr () : similar to SQL aliasing. Writing the new column name as expression.

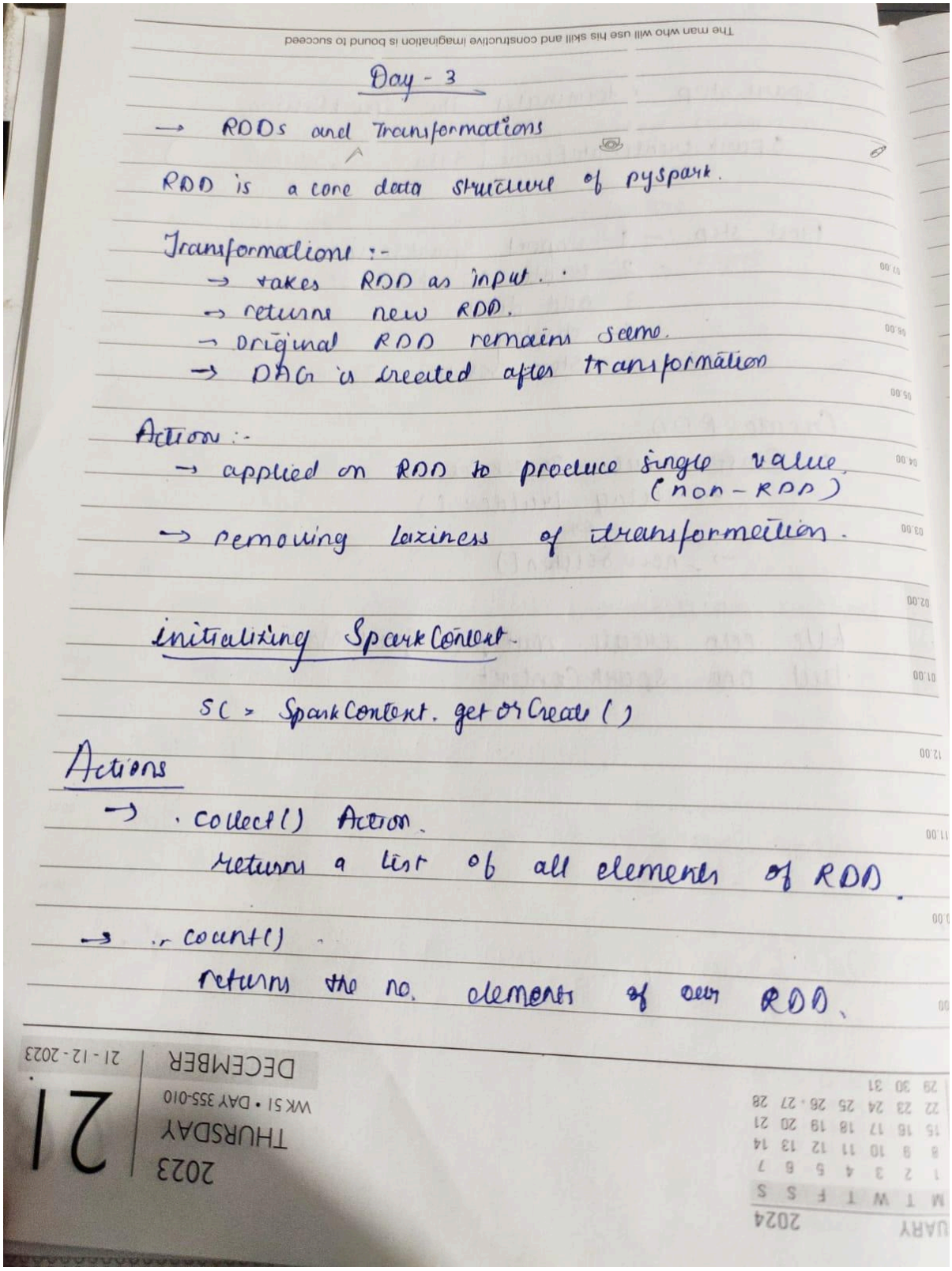
```
Cmd 5

1  df_selectExpr = df.selectExpr("Name as names","DOB",'Gender','salary')
2  df_selectExpr.show()

▸ (3) Spark Jobs
▸ df_selectExpr:  pyspark.sql.dataframe.DataFrame = [names: string, DOB: string ... 2 more fields]

+------+----------+------+------+
| names|       DOB|Gender|salary|
+------+----------+------+------+
|   Ram|1991-04-01|     M|  3000|
|  Mike|2000-05-19|     M|  4000|
|Rohini|1978-09-05|     M|  4000|
| Maria|1967-12-01|     F|  4000|
| Jenis|1980-02-17|     F|  1200|
+------+----------+------+------+

Command took 0.86 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:18:05 PM on Test
```

5. Using toDF(): This function returns a new DataFrame that with new specified column names



```
Cmd 6

1  datalist = ['First Name','DateOfBirth','M/F','Payment']
2  toDF = df.toDF(*datalist)
3  toDF.show()

▸ (3) Spark Jobs
▸ toDF:  pyspark.sql.dataframe.DataFrame = [First Name: string, DateOfBirth: string ... 2 more fields]

+----------+-----------+---+-------+
|First Name|DateOfBirth|M/F|Payment|
+----------+-----------+---+-------+
|       Ram| 1991-04-01|  M|   3000|
|      Mike| 2000-05-19|  M|   4000|
|    Rohini| 1978-09-05|  M|   4000|
|     Maria| 1967-12-01|  F|   4000|
|     Jenis| 1980-02-17|  F|   1200|
+----------+-----------+---+-------+

Command took 0.93 seconds -- by mitushivishrgpv@gmail.com at 2/6/2024, 4:25:59 PM on Test
```

Notes :



The man who will use his skill and constructive imagination is bound to succeed

Day - 3

→ RDDs and Transformations

RDD is a core data structure of pyspark.

Transformations :-
- → takes RDD as input.
- → returns new RDD.
- → Original RDD remains same.
- → DAG is created after transformation

Action :-
- → applied on RDD to produce single value. (non-RDD)
- → removing laziness of transformation.

Initializing SparkContext.

```
SC = SparkContext.getOrCreate()
```

Actions

→ .collect() Action.
returns a list of all elements of RDD.

→ .count()
returns the no. elements of our RDD.

21 | THURSDAY | 2023 | DECEMBER | 21-12-2023
WK 51 • DAY 355-010

JARY 2024
M T W T F S S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

→ . first()
  → returns first element
  → helpful in verifying

→ . take(n) first
  → returns n no. of elements.
  → n is no. of elements we want

→ . reduce()
  → take two elements. and operates.
  → operation lambda

→ . SaveAsTextFile()
  → to save RDD as text file.

Transformations

→ . map()
  → maps a value to elements of RDD.
  → similar to map in python
  → it returns a new RDD so we use . collect() to extract all elements.

→ . filter()
  → filtering the elements
  → takes anonymous func. with a condition

→ .union()
  → combines two RDDs
  → returns union

→ .

Dataframe in spark

Create dataframe :-
  → create spark session.
  → create data
  → column names
  → spark dataframe. . createDataFrame(data=, schema=..)

→ . withColumnRenamed(Existing, new)
  → to change column names
  → returns renamed column in df
  → df is immutable so original remains same.

→ using selectExpr(expr)
  → expr is SQL expression.

→ using .toDF(*col)

Dataframes in pyspark can be created from csv, json, RDD, SQL, python lists/dicts