

Name: Akula Sharathchandra

Batch: Data Engineering

Date:06/02/2024

TOPICS: PYSPARK RDD, hands on
RDD, CREATING, SELECTING, RENAMING THE
DATAFRAME.

PYSPARK RDD(THEORY):

Syntax:

```
# Import Spark Session  
from pyspark.sql import sparkSession  
  
# Create SparkSession  
spark = SparkSession.builder  
    .master("local[1]")  
    .appName
```

06/02/24

1) Pyspark RDD's & Pyu RDD's

consists of 2 operations

transformations, Actions

* RDD Operations: It is a core data structure of Pyspark.

i) transformations → These are operations that takes an RDD as input & produces another RDD as output.

→ Once transformation is applied to RDD, it returns a new RDD, the original RDD remains the same & thus are immutable.

→ After applying it creates Directed acyclic graph DAG for computations & ends after applying any actions on it.

→ Lazy Evaluation Process

ii) Action applied on RDD to produce a

single value.

→ Here are applied on resultant RDD &

produces a non-RDD value, thus removing laziness of transformation of RDD.

⇒ Layman's terms = transformation will happen

Transformations

applied on RDD & give another RDD

Action

Performed on RDD to give a non-RDD value.

→ Initialize a spark context

from PySpark import SparkContext

SC = SparkContext.getOrCreate()

-) essential PySpark RDD operations

i) ~~•~~ collect() Action

* The collect() action on RDD returns a list of all elements of RDD's

Ex:-

collect_rdd = SC.parallelize([1, 2, 3, 4, 5])

Print (collect_rdd.collect())

O/P = {1, 2, 3, 4, 5}

ii) `Count()`: Returns count of elements from RDD.

Ent
first-odd = sc.parallelize([1, 2, 3, 4, 5])
Print(first-odd.count())

DIP = 10

→ count-odd uses the `Parallelize()` method
& prints the ~~Count~~ count of elements

iii) `first()`: Returns the first element from RDD.

Ent
first-odd = sc.parallelize([1, 2, 3, 4, 5])
Print(first-odd.first())

DIP = 1 (Output) → top = individual X → X

iv) `take()`: Returns n numbers of elements from RDD, returns from starting to no of arguments

Ent
take-odd = sc.parallelize([1, 2, 3, 4, 5])
Print(take-odd.take(3))

DIP = [1, 2, 3]

v) reduce (S) 2 Elements from given RDD

2 Operates.

→ Performed using anonymous functions or lambda.

Syntax:

reduce_rdd = sc.parallelize([1, 3, 4, 6])

Print(reduce_rdd.reduce(lambda x, y: x + y))
O/P:

vi) saveAsTextFile (S) used to save the resultant RDD as text file.

Syntax:

Save_rdd = sc.parallelize([1, 2, 3, 4, 5, 6])

Save_rdd.saveAsTextFile('file.txt')

O/P:

HANDSON OF RDD OPERATIONS IN DATABRICKS AS WELL AS IN JUPYTER:

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 13 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

Cmd 1

```
1 # reduce operation()
2 from pyspark import SparkContext
3 sc = SparkContext.getOrCreate()
4 reduce_rdd = sc.parallelize([1,3,4,6])
5 print(reduce_rdd.reduce(lambda x, y : x + y))
```

▶ (1) Spark Jobs
14
Command took 0.42 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:18:36 on MyCluster3

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 13 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

Cmd 1

```
14  
Command took 0.42 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:18:36 on MyCluster3

Cmd 2



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



▶ (1) Spark Jobs  
10  
Command took 0.55 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:19:36 on MyCluster3



Cmd 3


```

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 13 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

Cmd 1

```
1 # SaveAsTextFile()
2 from pyspark import SparkContext
3 sc = SparkContext.getOrCreate()
4 save_rdd = sc.parallelize([1,2,3,4,5,6])
5 save_rdd.saveAsTextFile('file1.txt')
```

▼ (1) Spark Jobs
▼ Job 6 View (Stages: 1/1)
Stage 6 8/8 succeeded View

Command took 1.76 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:23:24 on MyCluster3

Cmd 4

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 14 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

Job 6 View (Stages: 1/1)
Stage 6 8/8 succeeded View
Command took 1.76 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:23:24 on MyCluster3

Cmd 4

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

▶ (2) Spark Jobs
[1, 2, 3, 4]
Command took 1.01 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:29:54 on MyCluster3

Cmd 5

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 17 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

Command took 1.01 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:29:54 on MyCluster3

Cmd 5

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

▶ (1) Spark Jobs
1
Command took 0.38 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:42:53 on MyCluster3

Cmd 6

RDD OPERATIONS Python ☆

File Edit View Run Help Last edit was 17 minutes ago New cell UI: OFF ▶ Run all MyCluster3 Share Publish

▶ (1) Spark Jobs
1
Command took 0.38 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:42:53 on MyCluster3

Cmd 6

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

▶ (1) Spark Jobs
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9]
Command took 0.21 seconds -- by cdamvsr@gmail.com at 06/02/2024, 12:33:30 on MyCluster3

IN JUPYTER NOTEBOOK:

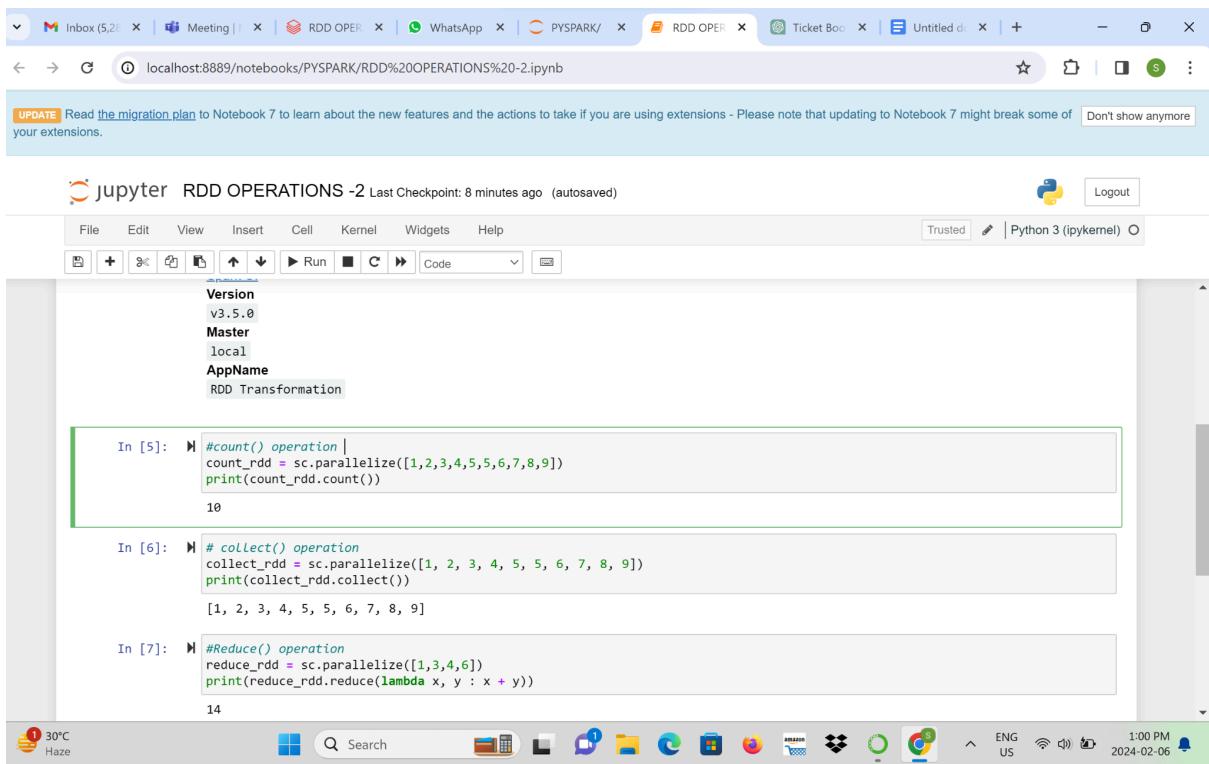
```
In [1]: └─ !pip install findspark
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1

In [2]: └─ import findspark

In [3]: └─ findspark.init()

In [4]: └─ from pyspark import SparkContext
sc = SparkContext("local", "RDD Transformation")
sc

Out[4]: SparkContext
Spark UI
Version
v3.5.0
Master
```



```
In [5]: └─ #count() operation
count_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(count_rdd.count())
10

In [6]: └─ # collect() operation
collect_rdd = sc.parallelize([1, 2, 3, 4, 5, 5, 6, 7, 8, 9])
print(collect_rdd.collect())
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9]

In [7]: └─ #Reduce() operation
reduce_rdd = sc.parallelize([1,3,4,6])
print(reduce_rdd.reduce(lambda x, y : x + y))
14
```



UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. [Don't show anymore](#)

jupyter RDD OPERATIONS -2 Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) Logout

In [8]:

```
reduce_rdd = sc.parallelize([1,3,4,6])
print(reduce_rdd.reduce(lambda x, y : x + y))
```

14

In [8]:

```
#take() operation
take_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(take_rdd.take(4))
```

[1, 2, 3, 4]

In [9]:

```
#first() operation
first_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(first_rdd.first())
```

1

In []:

30°C Haze Search ENG US 1:01 PM 2024-02-06

```
reduce_rdd = sc.parallelize([1,3,4,6])
print(reduce_rdd.reduce(lambda x, y : x + y))
```

14

```
#take() operation
take_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(take_rdd.take(4))
```

[1, 2, 3, 4]

```
#first() operation
first_rdd = sc.parallelize([1,2,3,4,5,5,6,7,8,9])
print(first_rdd.first())
```

1

→ CREATING DATA FRAME:

The screenshot shows a Databricks notebook interface. The title bar says "data frame creation" and "Python". The code cell contains the following Python code:

```
1 import pyspark
2 from pyspark.sql import SparkSession
3 # Create a spark session
4 spark = SparkSession.builder.appName('pyspark - example join').getOrCreate()
5 # Create data in dataframe
6 data = [(['Ram'], '1991-04-01', 'M', 3000),
7         (['Mike'], '2000-05-19', 'M', 4000),
8         (['Rohini'], '1978-09-05', 'M', 4000),
9         (['Maria'], '1967-12-01', 'F', 4000),
10        (['Jenis'], '1980-02-17', 'F', 1200)]
11 # Column names in dataframe
12 columns = ["Name", "DOB", "Gender", "salary"]
13 # Create the spark dataframe
14 df = spark.createDataFrame(data=data,
15                            schema=columns)
16 # Print the dataframe
17 df.show()
```

The output section shows the resulting DataFrame:

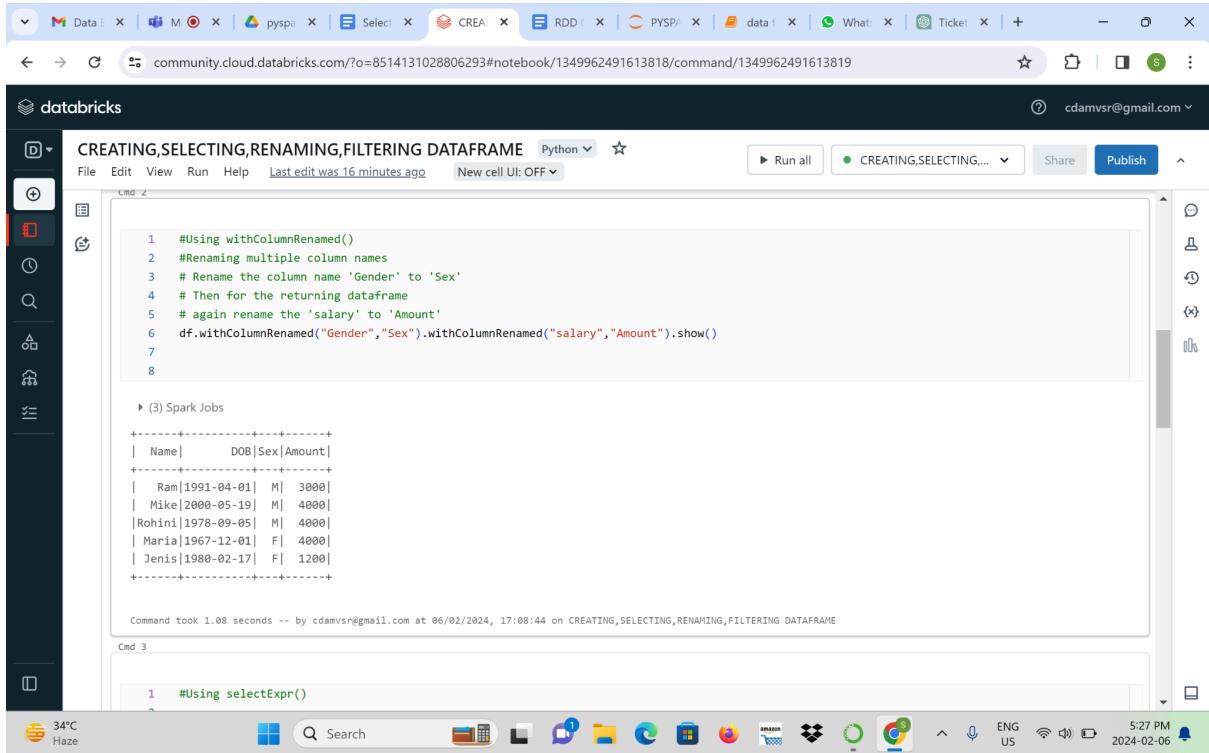
```
+---+---+---+---+
| Name| DOB|Gender|salary|
+---+---+---+---+
| Ram|1991-04-01| M| 3000|
| Mike|2000-05-19| M| 4000|
|Rohini|1978-09-05| M| 4000|
| Maria|1967-12-01| F| 4000|
| Jenis|1980-02-17| F| 1200|
```

The status bar at the bottom indicates "33°C Haze" and the date "2024-02-06".

Method 1: Using withColumnRenamed()

We will use of withColumnRenamed() method to change the column names of pyspark data frame.

Syntax: DataFrame.withColumnRenamed(existing, new)



The screenshot shows a Databricks notebook interface. The top navigation bar includes tabs for Data, M, pyspark, Selection, CREA, RDD, PYSP, dataf, What, Ticket, and a plus sign. The URL in the address bar is community.cloud.databricks.com/?o=8514131028806293#notebook/1349962491613818/command/1349962491613819. The sidebar on the left has icons for file operations, search, and other notebooks. The main area contains a code cell titled "CREATING,SELECTING,RENAMING,FILTERING DATAFRAME" with Python selected. The code is:

```
1 #Using withColumnRenamed()
2 #Renaming multiple column names
3 # Rename the column name 'Gender' to 'Sex'
4 # Then for the returning dataframe
5 # again rename the 'salary' to 'Amount'
6 df.withColumnRenamed("Gender","Sex").withColumnRenamed("salary","Amount").show()
7
8
```

Below the code cell, the output shows a table with columns Name, DOB, Sex, and Amount. The data is:

Name	DOB	Sex	Amount
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

At the bottom of the output, it says "Command took 1.08 seconds -- by cdamvsr@gmail.com at 06/02/2024, 17:08:44 on CREATING,SELECTING,RENAMING,FILTERING DATAFRAME".

Below the output, there is another code cell titled "Cmd 3" with the code:

```
1 #Using selectExpr()
```

Method 2: Using selectExpr()

Renaming the column names using `selectExpr()` method

Syntax : `DataFrame.selectExpr(expr)`

The screenshot shows a Databricks notebook titled "CREATING,SELECTING,RENAMING,FILTERING DATAFRAME" in Python. The code uses `selectExpr` to rename the "Name" column to "name". The output shows a DataFrame with columns "name", "DOB", "Gender", and "salary".

```
1 #Using selectExpr()
2
3 # Select the 'Name' as 'name'
4 # Select remaining with their original name
5 data = df.selectExpr("Name as name","DOB","Gender","salary")
6
7 # Print the dataframe
8 data.show()
9
10
(3) Spark Jobs
data: 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 1.56 seconds -- by cdamvsr@gmail.com at 06/02/2024, 17:07:01 on CREATING,SELECTING,RENAMING,FILTERING DATAFRAME

Method 3: Using select() method

Syntax: `DataFrame.select(cols)`

The screenshot shows a Databricks notebook titled "CREATING,SELECTING,RENAMING,FILTERING DATAFRAME" in Python. The code uses `select` with a list of columns to rename the "Name" column to "Amount". The output shows a DataFrame with columns "Name", "DOB", "Gender", and "Amount".

```
1 # Using select() method
2
3 # Import col method from pyspark.sql.functions
4 from pyspark.sql.functions import col
5
6 # Select the 'salary' as 'Amount' using aliasing
7 # Select remaining with their original name
8 data = df.select(col("Name"),col("DOB"),
9 | | | | col("Gender"),
10 | | | | col("salary").alias("Amount"))
11 # Print the dataframe
12 data.show()
13
14
(3) Spark Jobs
data: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]
+---+---+---+
| Name| DOB|Gender|Amount|
+---+---+---+
| 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|
+---+---+---+
```

Example-2:

The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar says "EXAMPLES OF RENAMING DATAFRAMES" and "Python". The code cell contains Python code to create a DataFrame with Indian names and print it. The output cell shows the DataFrame structure and data.

```
1 import pyspark
2 from pyspark.sql import SparkSession
3
4 # Create a spark session
5 spark = SparkSession.builder.appName('pyspark - example join').getOrCreate()
6
7 # Create data in dataframe with Indian names
8 data = [ ('Arjun', '1990-08-15', 'M', 5000),
9          ('Priya', '1985-03-25', 'F', 6000),
10         ('Amit', '1976-11-10', 'M', 7000),
11         ('Sonia', '1988-06-30', 'F', 8000),
12         ('Rajesh', '1982-09-12', 'M', 9000)]
13
14 # Column names in dataframe
15 columns = ["Name", "DOB", "Gender", "salary"]
16
17 # Create the spark dataframe
18 df = spark.createDataFrame(data=data, schema=columns)
19
20 # Print the dataframe with Indian names
21 df.show()
```

Output:

```
| (3) Spark Jobs
| df: pyspark.sql.dataframe.DataFrame = [Name: string, DOB: string ... 2 more fields]
+----+-----+-----+-----+
| Name|      DOB|Gender|salary|
+----+-----+-----+-----+
| Arjun|1990-08-15|      M|   5000|
| Priya|1985-03-25|      F|   6000|
| Amit|1976-11-10|      M|   7000|
| Sonia|1988-06-30|      F|   8000|
| Rajesh|1982-09-12|      M|   9000|
+----+-----+-----+-----+
```

Command took 2.09 seconds -- by cdamvsr@gmail.com at 06/02/2024, 17:45:47 on CREATING,SELECTING,RENAMING,FILTERING DATAFRAME

The screenshot shows a Jupyter Notebook interface with a dark theme. The code cell contains Python code to rename columns "Gender" to "Sex" and "salary" to "Amount" in the DataFrame. The output cell shows the DataFrame structure and data after renaming.

```
1 #Using withColumnRenamed()
2 #Renaming multiple column name
3 df.withColumnRenamed("Gender","Sex").withColumnRenamed("salary","Amount").show()
```

Output:

```
| (3) Spark Jobs
+----+-----+-----+
| Name|      DOB|Sex|Amount|
+----+-----+-----+
| Arjun|1990-08-15|  M|  5000|
| Priya|1985-03-25|  F|  6000|
| Amit|1976-11-10|  M|  7000|
| Sonia|1988-06-30|  F|  8000|
| Rajesh|1982-09-12|  M|  9000|
+----+-----+-----+
```

Command took 0.96 seconds -- by cdamvsr@gmail.com at 06/02/2024, 17:51:17 on CREATING,SELECTING,RENAMING,FILTERING DATAFRAME

Cmd 3

```
1 #Using selectExpr()
2 # Select the 'Name' as 'name'
3 # Select remaining with their original name
4 data = df.selectExpr("Name as name","DOB","Gender","salary")
5 # Print the dataframe
6 data.show()
```

► (3) Spark Jobs

► [] data: pyspark.sql.DataFrame = [name: string, DOB: string ... 2 more fields]

```
+-----+-----+
| name | DOB |Gender|salary|
+-----+-----+
| Arjun|1990-08-15| M| 5000|
| Priya|1985-03-25| F| 6000|
| Amit|1976-11-10| M| 7000|
| Sonia|1988-06-30| F| 8000|
|Rajesh|1982-09-12| M| 9000|
+-----+-----+
```

Command took 1.46 seconds -- by cdamvsr@gmail.com at 06/02/2024, 17:54:11 on CREATING,SELECTING,RENAMING,FILTERING DATAFRAME

Cmd 4

Python

```
1 # Using select() method
2
3 # Import col method from pyspark.sql.functions
4 from pyspark.sql.functions import col
5
6 # Select the 'salary' as 'Amount' using aliasing
7 # Select remaining with their original name
8 data = df.select(col("Name"),col("DOB"),
9 | | | | col("Gender"),
10 | | | | col("salary").alias('Amount'))
11 # Print the dataframe
12 data.show()
```

► (3) Spark Jobs

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

```
+-----+-----+
| Name | DOB |Gender|Amount|
+-----+-----+
| Arjun|1990-08-15| M| 5000|
| Priya|1985-03-25| F| 6000|
| Amit|1976-11-10| M| 7000|
| Sonia|1988-06-30| F| 8000|
|Rajesh|1982-09-12| M| 9000|
+-----+-----+
```