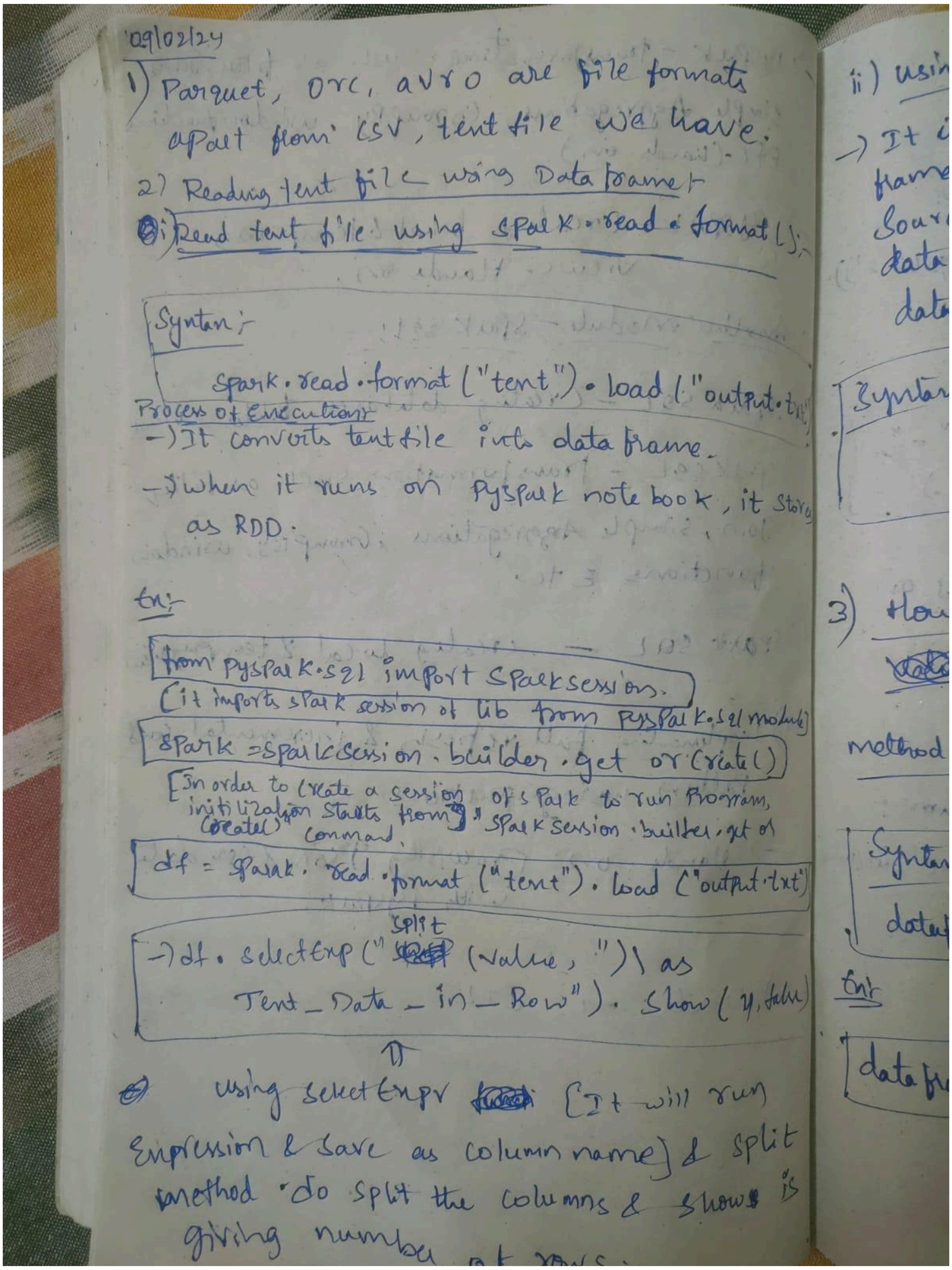
NAME : AKULA SHARATH CHANDRA
BATCH : DATA ENGINEERING
DATE : 09-02-2024
TOPICS: CONCEPTS AND HANDSON FOR READING CSV,GROUP BY,PIVOT,HANDLING FILES,SORT,ORDERBY,JOINS

CONCEPTS:



09/02/24

1) Parquet, orc, avro are file formats apart from csv, text file we have.

2) Reading text file using Data frame

i) Read text file using spark.read.format():-

Syntax:-

```
spark.read.format("text").load("output.txt")
```

Process of Execution:-

-) It converts text file into data frame.

-) when it runs on pyspark note book, it stores as RDD.

Ex:-

```
from pyspark.sql import SparkSession.
```

[it imports spark session of lib from pyspark.sql module]

```
spark = SparkSession.builder.get or Create()
```

[In order to create a session of spark to run program, initilization starts from "SparkSession.builder.get or Create()" command.

```
df = spark.read.format("text").load("output.txt")
```

```
-) df.selectExp("split (value, ' ') as
        Text_Data_in_Row").show(4, false)
```

using selectExpr [It will run Expression & save as column name] & split method do split the columns & show is giving number of rows.

ii) usin

-) It i
frame
Sour
data
data

Syntax

3) How

method

Syntax
data

Ex:-

data fr

ii) using spark.read.format():-

-) It is used to load text files into data frame. The .format() specifies the i/p data source format as "text". The .load() loads data from a data source and returns data frame.

```
Syntax:- spark.read.format("text").
           load(path=None, format=None,
           schema=None, ** options).
```

3) How to add new column ~~with const~~ to a ~~value~~ Pyspark dataframe:-

method 1:- Add new column with constant value.

```
Syntax:
dataframe.withcolumn("column_name", lit(value))
```

Ex:

```
dataframe.withColumn("salary", lit(34000)).show()
```

Method 2:- using concat_ws():-

Syntax:-

```
dataframe.withColumn("column_name", concat_ws
        ("separator", "Existing_column1", 'Existing_col2'))
```

Ex:-

```
dataframe.withColumn("Details", concat_ws("-",
        "Name", "Company")).show()
```

Method 3:- Add a column when not Exists on Data frame

→ Syntax

```
if 'column_name' not in dataframe.colums:
    dataframe.withColumn("column_name', lit(value
```

4) Manipulating :- (groupBy, Aggregate functions)

-) groupBy & aggregate fn used to collect the identical data into groups on DataFrame & performs sum, count, avg, min & max functions on the grouped data.

Ex:

```
import pyspark
from pyspark.sql import sparksession.
#create sparksession
spark = SparkSession.builder.appName('Practice')
                           .getOrCreate()
datalist = [("Java,20000), ("Python,10000)]
rdd = spark.sparkcontent.Parallelize(datlist)
```

Ex2:

```
df2 = spark.read.csv("Path")
df2.show()
```

=) read & shows the data present in csv file.

Ex3:

```
{ Sample = "Path" }
df1 = spark.read.csv(Sample, header = true,
                  inferischema = True)
df1.groupBy("departments").sum("salary").
                              show()
```

shows departments & sum(salary) with groupBy.

En4:-

```
df1.groupBy("Departments").min("salary").show()
```

⇑ Shows min(salary).

En5:-

```
df1.groupBy("dep").max("Salary").show()
```

⇑ Shows max(salary)

En6:-

```
df1.groupBy("dep").avg("sal").show()
```

⇑ Shows avg(sal).

En7:-

```
df1.groupBy("dep").mean("sal").show()
```

calculates avg value or set of values.

En8:-

```
df1.groupBy("dep").count().show()
```

Counts the values of "departments".

En9:

df1

En10

df

-)drop

i) h

df

-)ii

li

ii) df

-) re

R

iii) d

-) r

Ex9:

```
df1.groupBy("dep").Pivot("Name").sum("salary").
                                          show()
```

returns unisue values in Name column into seperate colum. & displays

Ex10:

~~[illegible]~~ dropping rows based on null values

```
df1.na.drop().show
```

-) drop() has 3 Parameters

i) how:-

```
df1.na.drop(how = "all").show().
```

-) it will returns if all values in rows are null then drop # default any.

ii)

```
df1.na.drop(how = "any", subset = ["salary"]).
                 show()
```

-) returns atleast 2 non null values should be Present.

iii)

```
df1.na.drop(how = "any", subset = ["salary"])
                               .show()
```

-) returns only in that column rows get deleted

5) orderby() & sort():-

sort():- is to sort a dataframe by using one or more columns, default - ascending order

i)
```
df1.sort("salary").show()
```
↓
sorts salary in ascending order & Pr

ii) sort based on descending order :-
```
df1.sort(df1["salary"].des()).show()
```
↓
sorts salary in descending order & Prints

iii) sort based on 1st column then 2nd column :-
```
df1.sort("salary", "name").show()
```
sorts Salary & name simultaneously

- orderby:-
```
df1.orderBy("salary").show().
```
Sorts based on single column..

6) PySpark

→ inner join

→ outer jo

→ Left Joi

→ Right

→ Left sem

→ left

Ex syn

```
empDF.
```

Inner

EmpDF

6) Py Spark Joint

-) inner join :- Join records when key columns are matched & droped when they are not match.

-) outer joint returns all rows from both databases, wher Join doesn't match it returns null or respective columns.

-) Left Joint returns all rows from left data set
↑
-) Right Joint oppisite.

-) Left semi joint returns columns from only left data set for matched records.

-) Left Anti Joint returns columns from left dataset for non-matched records

Ex syntax.

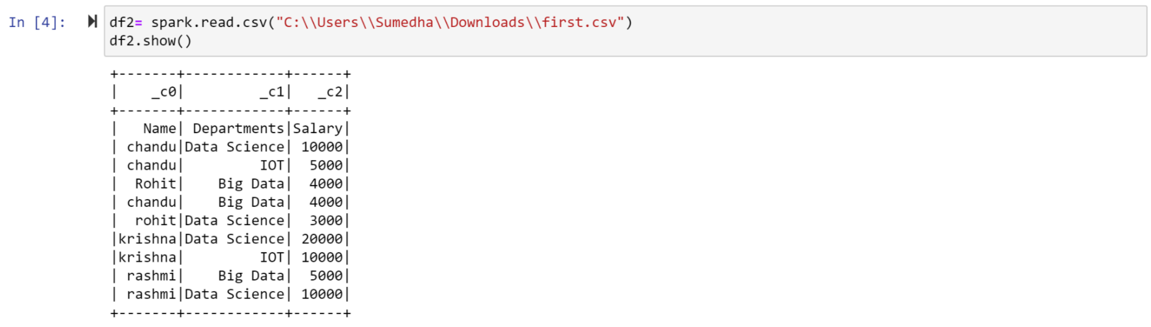
```
EmpDF.Join (deptDF, EmpDF.Emp_dept id ==
                deptDF.dept_id ; "inner").
                                  show()
```

Inner

```
EmpDF.Join (dept DF, EmpDF. Emp dept id
```

HANDS ON:

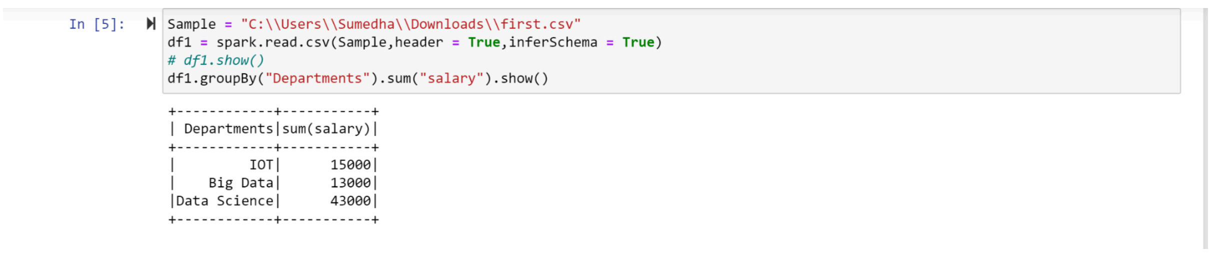
1)READING CSV FILES:



```
In [4]:  df2= spark.read.csv("C:\\Users\\Sumedha\\Downloads\\first.csv")
         df2.show()

+-------+-----------+------+
|    _c0|        _c1|   _c2|
+-------+-----------+------+
|   Name| Departments|Salary|
| chandu|Data Science| 10000|
| chandu|         IOT|  5000|
|  Rohit|    Big Data|  4000|
| chandu|    Big Data|  4000|
|  rohit|Data Science|  3000|
|krishna|Data Science| 20000|
|krishna|         IOT| 10000|
| rashmi|    Big Data|  5000|
| rashmi|Data Science| 10000|
+-------+-----------+------+
```

READING CSV FILE USING GROUP BY:

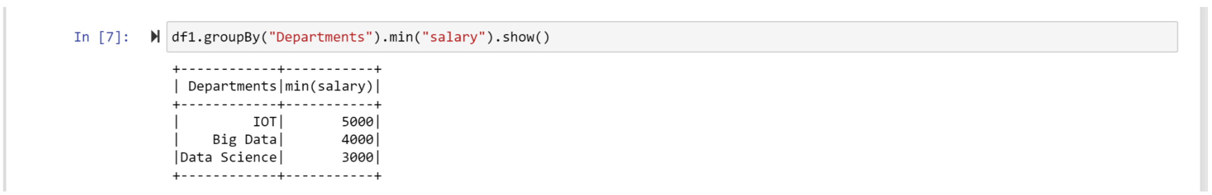


```
In [5]:  Sample = "C:\\Users\\Sumedha\\Downloads\\first.csv"
         df1 = spark.read.csv(Sample,header = True,inferSchema = True)
         # df1.show()
         df1.groupBy("Departments").sum("salary").show()

+------------+-----------+
| Departments|sum(salary)|
+------------+-----------+
|         IOT|      15000|
|    Big Data|      13000|
|Data Science|      43000|
+------------+-----------+
```

GROUP BY USING AGGREGATE FUNCTIONS:

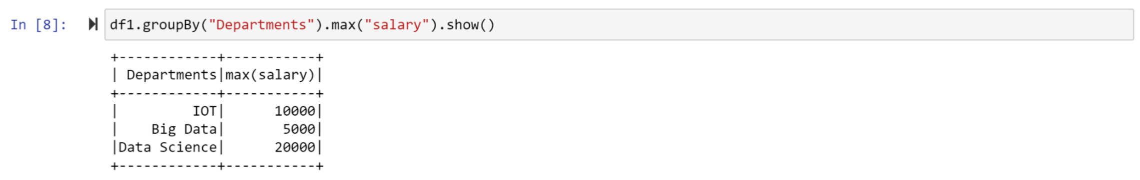
i) using min():



```
In [7]:  df1.groupBy("Departments").min("salary").show()

+------------+-----------+
| Departments|min(salary)|
+------------+-----------+
|         IOT|       5000|
|    Big Data|       4000|
|Data Science|       3000|
+------------+-----------+
```

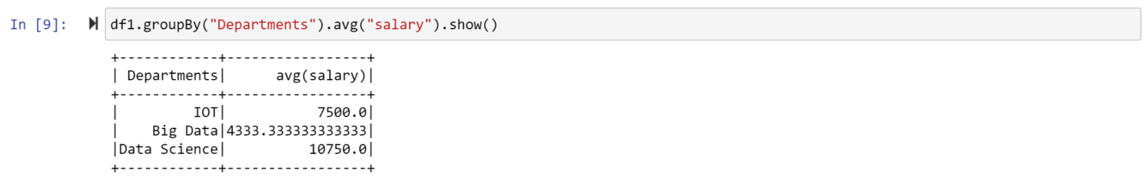
ii) using max():



```
In [8]:  df1.groupBy("Departments").max("salary").show()

+------------+-----------+
| Departments|max(salary)|
+------------+-----------+
|         IOT|      10000|
|    Big Data|       5000|
|Data Science|      20000|
+------------+-----------+
```

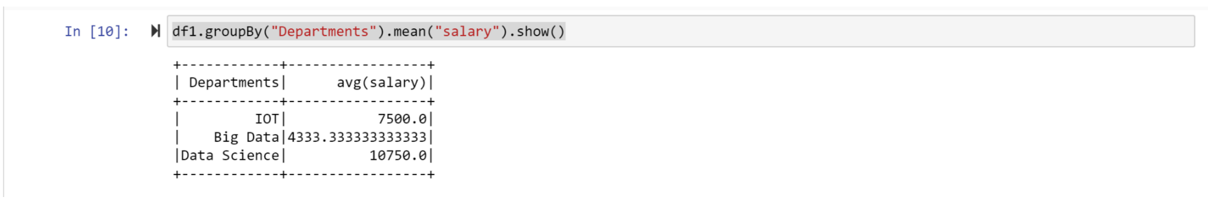
iii) using avg():



```
In [9]:  df1.groupBy("Departments").avg("salary").show()

+------------+-----------------+
| Departments|      avg(salary)|
+------------+-----------------+
|         IOT|           7500.0|
|    Big Data|4333.333333333333|
|Data Science|          10750.0|
+------------+-----------------+
```

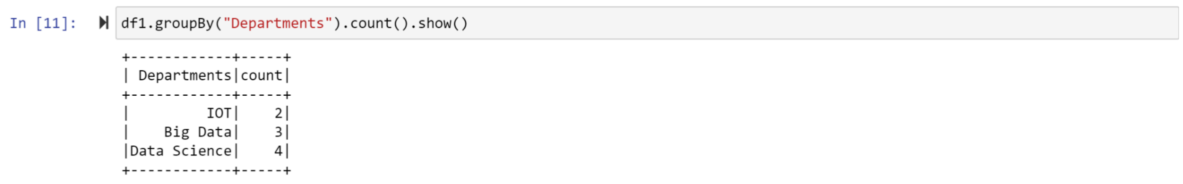
iv) using mean():



```
In [10]:  df1.groupBy("Departments").mean("salary").show()

+------------+-----------------+
| Departments|      avg(salary)|
+------------+-----------------+
|         IOT|           7500.0|
|    Big Data|4333.333333333333|
|Data Science|          10750.0|
+------------+-----------------+
```

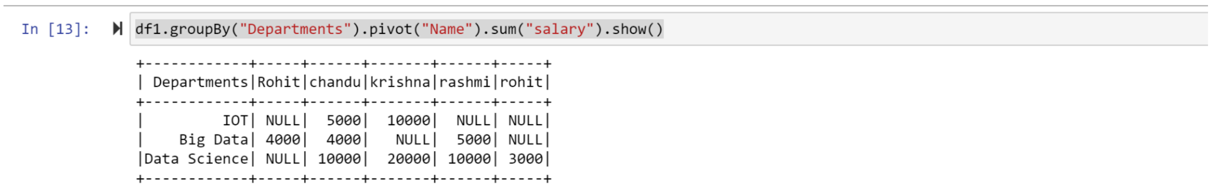
v) using count():



```
In [11]:  df1.groupBy("Departments").count().show()

+------------+-----+
| Departments|count|
+------------+-----+
|         IOT|    2|
|    Big Data|    3|
|Data Science|    4|
+------------+-----+
```

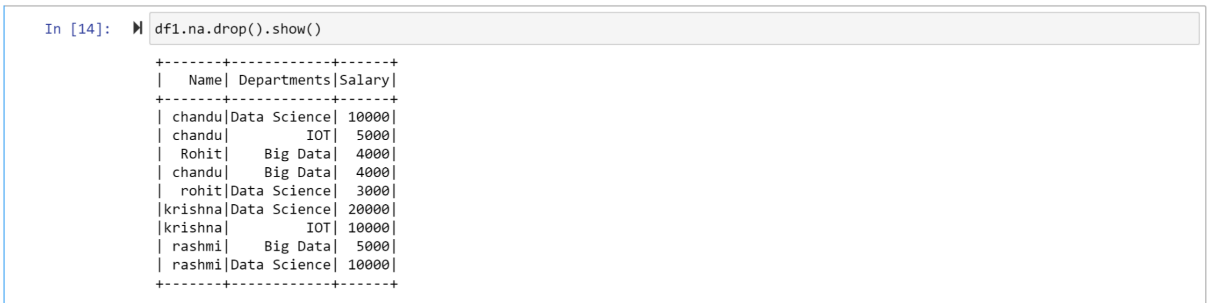
vi) using pivot():



```
In [13]:  df1.groupBy("Departments").pivot("Name").sum("salary").show()

+------------+-----+------+-------+------+-----+
| Departments|Rohit|chandu|krishna|rashmi|rohit|
+------------+-----+------+-------+------+-----+
|         IOT| NULL|  5000|  10000|  NULL| NULL|
|    Big Data| 4000|  4000|   NULL|  5000| NULL|
|Data Science| NULL| 10000|  20000| 10000| 3000|
+------------+-----+------+-------+------+-----+
```

2)HANDLING MISSING VALUES PYSPARK:



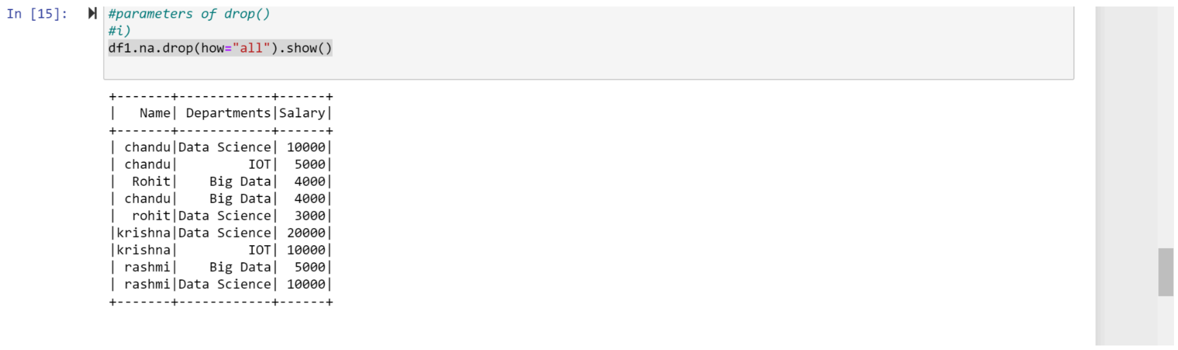
```
In [14]:  df1.na.drop().show()

+-------+------------+------+
|   Name| Departments|Salary|
+-------+------------+------+
| chandu|Data Science| 10000|
| chandu|         IOT|  5000|
|  Rohit|    Big Data|  4000|
| chandu|    Big Data|  4000|
|  rohit|Data Science|  3000|
|krishna|Data Science| 20000|
|krishna|         IOT| 10000|
| rashmi|    Big Data|  5000|
| rashmi|Data Science| 10000|
+-------+------------+------+
```

ii) PARAMETERS OF DROP():

It has three parameters - **how, thresh, and subset**

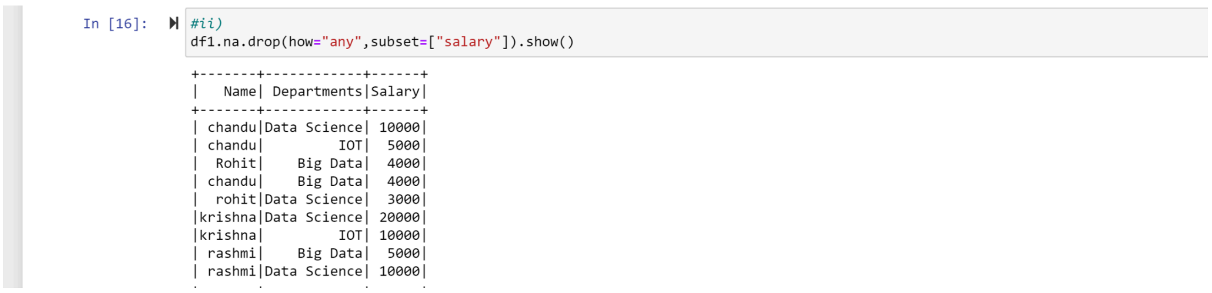
a) if all values in rows are null then drop default any:



```
In [15]:  #parameters of drop()
          #i)
          df1.na.drop(how="all").show()
```

```
+-------+------------+------+
|   Name| Departments|Salary|
+-------+------------+------+
| chandu|Data Science| 10000|
| chandu|         IOT|  5000|
|  Rohit|    Big Data|  4000|
| chandu|    Big Data|  4000|
|  rohit|Data Science|  3000|
|krishna|Data Science| 20000|
|krishna|         IOT| 10000|
| rashmi|    Big Data|  5000|
| rashmi|Data Science| 10000|
+-------+------------+------+
```

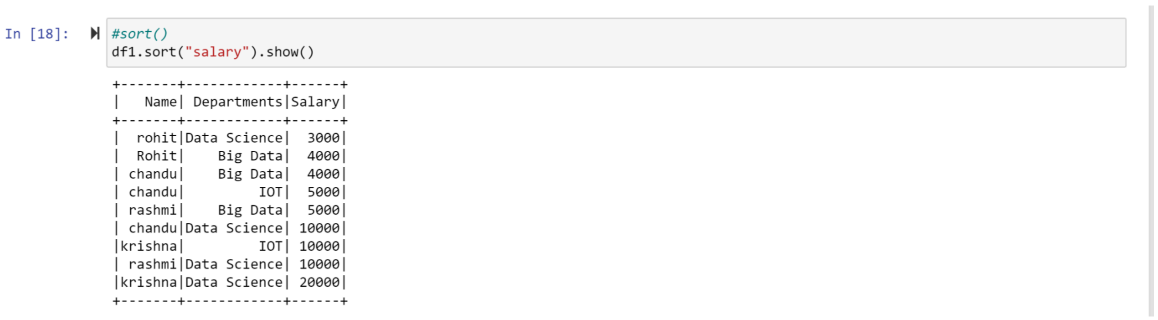
b)at least 2 non null values should be present.:



```
In [16]:  #ii)
          df1.na.drop(how="any",subset=["salary"]).show()
```

```
+-------+------------+------+
|   Name| Departments|Salary|
+-------+------------+------+
| chandu|Data Science| 10000|
| chandu|         IOT|  5000|
|  Rohit|    Big Data|  4000|
| chandu|    Big Data|  4000|
|  rohit|Data Science|  3000|
|krishna|Data Science| 20000|
|krishna|         IOT| 10000|
| rashmi|    Big Data|  5000|
| rashmi|Data Science| 10000|
```

c) only in that column rows get deleted:

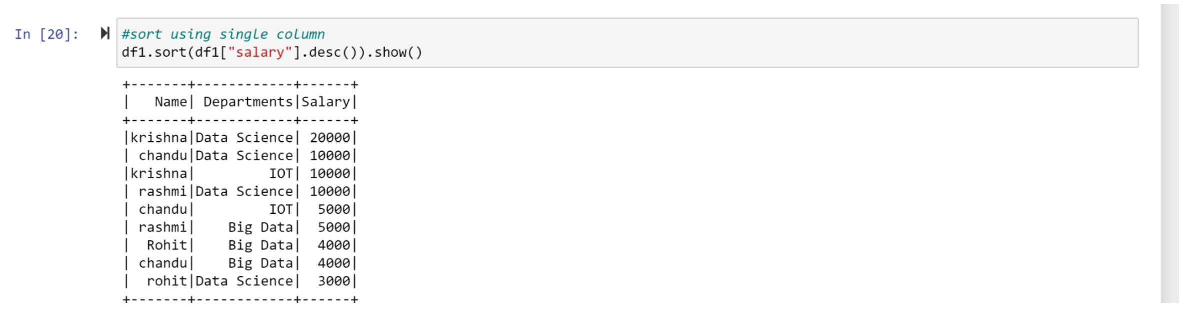


```
In [18]:  #sort()
          df1.sort("salary").show()
```

```
+-------+------------+------+
|   Name| Departments|Salary|
+-------+------------+------+
|  rohit|Data Science|  3000|
|  Rohit|    Big Data|  4000|
| chandu|    Big Data|  4000|
| chandu|         IOT|  5000|
| rashmi|    Big Data|  5000|
| chandu|Data Science| 10000|
|krishna|         IOT| 10000|
| rashmi|Data Science| 10000|
|krishna|Data Science| 20000|
+-------+------------+------+
```

3)ORDERBY() AND SORT() IN PYSPARK DATAFRAME:

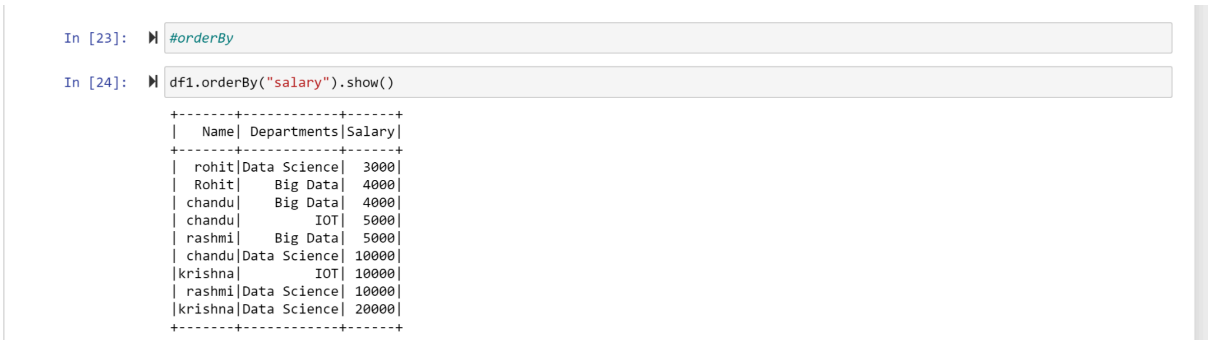
—>sort():



```
In [20]:  #sort using single column
          df1.sort(df1["salary"].desc()).show()

          +-------+------------+------+
          |   Name| Departments|Salary|
          +-------+------------+------+
          |krishna|Data Science| 20000|
          | chandu|Data Science| 10000|
          |krishna|         IOT| 10000|
          | rashmi|Data Science| 10000|
          | chandu|         IOT|  5000|
          | rashmi|    Big Data|  5000|
          |  Rohit|    Big Data|  4000|
          | chandu|    Big Data|  4000|
          |  rohit|Data Science|  3000|
          +-------+------------+------+
```

—>orderBy():

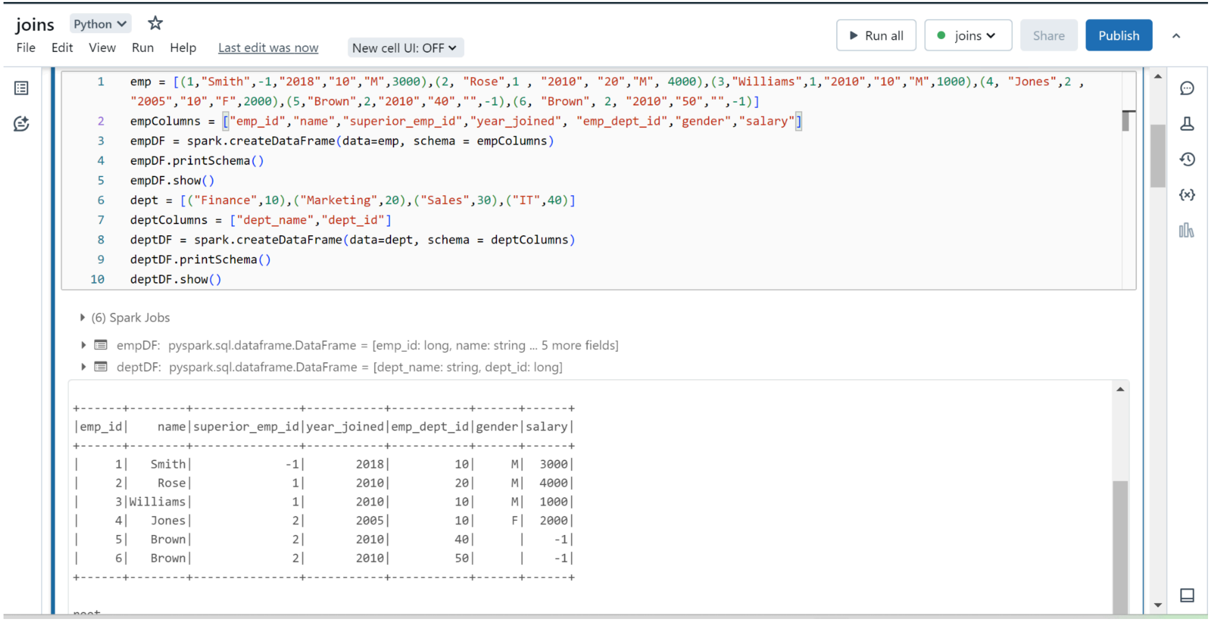


```
In [23]:  #orderBy

In [24]:  df1.orderBy("salary").show()

          +-------+------------+------+
          |   Name| Departments|Salary|
          +-------+------------+------+
          |  rohit|Data Science|  3000|
          |  Rohit|    Big Data|  4000|
          | chandu|    Big Data|  4000|
          | chandu|         IOT|  5000|
          | rashmi|    Big Data|  5000|
          | chandu|Data Science| 10000|
          |krishna|         IOT| 10000|
          | rashmi|Data Science| 10000|
          |krishna|Data Science| 20000|
          +-------+------------+------+
```

4)joins : PySpark Join is used to combine two DataFrames and by chaining these you can join multiple DataFrames



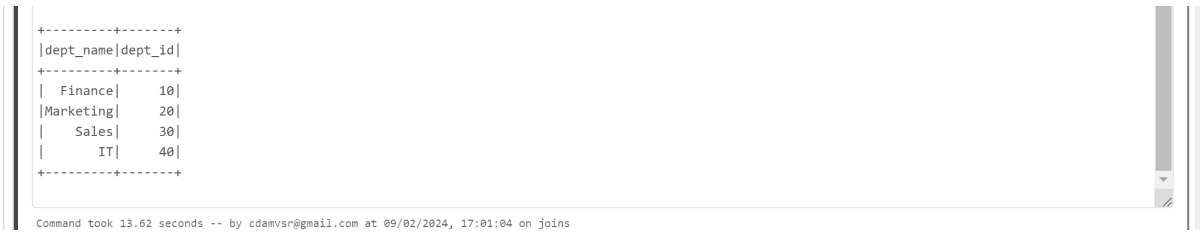
```python
emp = [(1,"Smith",-1,"2018","10","M",3000),(2, "Rose",1 , "2010", "20","M", 4000),(3,"Williams",1,"2010","10","M",1000),(4, "Jones",2 ,
"2005","10","F",2000),(5,"Brown",2,"2010","40","",-1),(6, "Brown", 2, "2010","50","",-1)]
empColumns = ["emp_id","name","superior_emp_id","year_joined", "emp_dept_id","gender","salary"]
empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show()
dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show()
```

(6) Spark Jobs

empDF: pyspark.sql.dataframe.DataFrame = [emp_id: long, name: string ... 5 more fields]

deptDF: pyspark.sql.dataframe.DataFrame = [dept_name: string, dept_id: long]

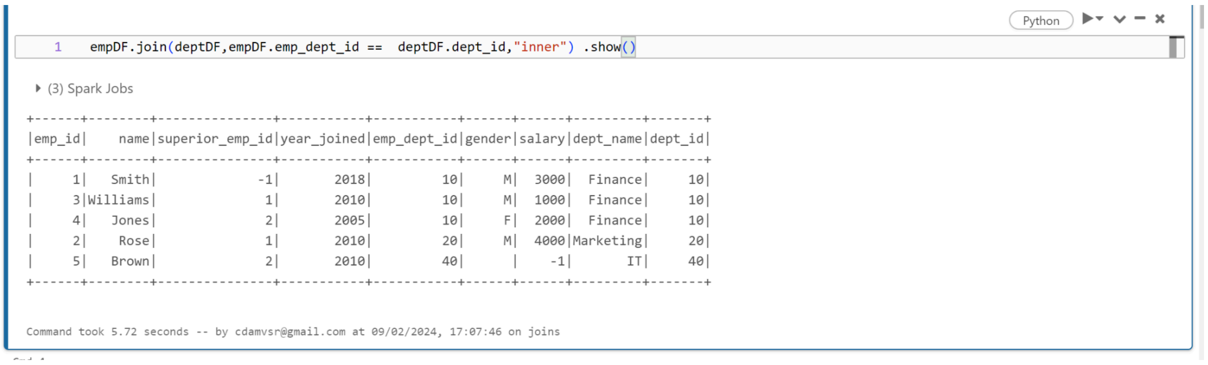
```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     2|    Rose|              1|       2010|         20|     M|  4000|
|     3|Williams|              1|       2010|         10|     M|  1000|
|     4|   Jones|              2|       2005|         10|     F|  2000|
|     5|   Brown|              2|       2010|         40|      |    -1|
|     6|   Brown|              2|       2010|         50|      |    -1|
+------+--------+---------------+-----------+-----------+------+------+
```



```
+---------+-------+
|dept_name|dept_id|
+---------+-------+
|  Finance|     10|
|Marketing|     20|
|    Sales|     30|
|       IT|     40|
+---------+-------+
```

Command took 13.62 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:01:04 on joins

i) INNER JOIN:Join records when key columns are matched, and dropped when they are not matched.



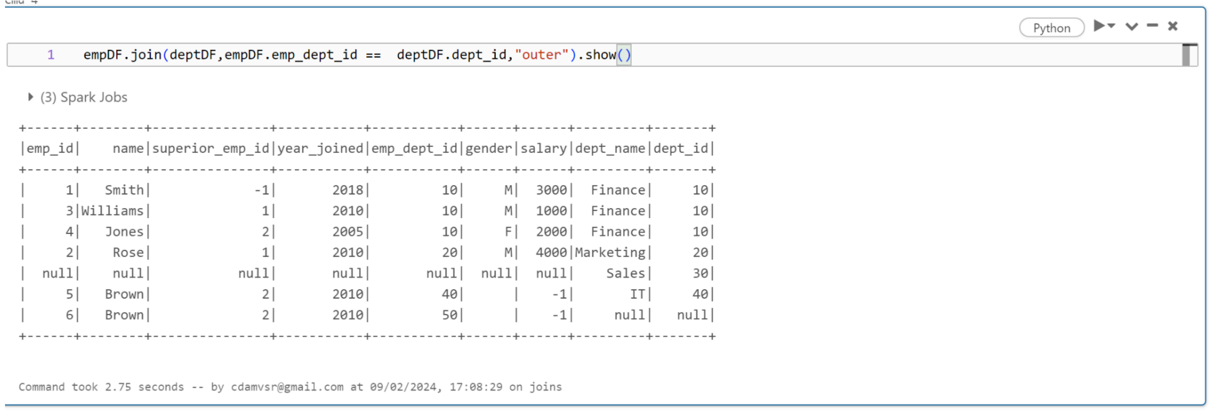
```python
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") .show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Command took 5.72 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:07:46 on joins

ii)OUTER JOIN: Returns all rows from both datasets, where the Join expression doesn't match it returns null or respective columns.



Python

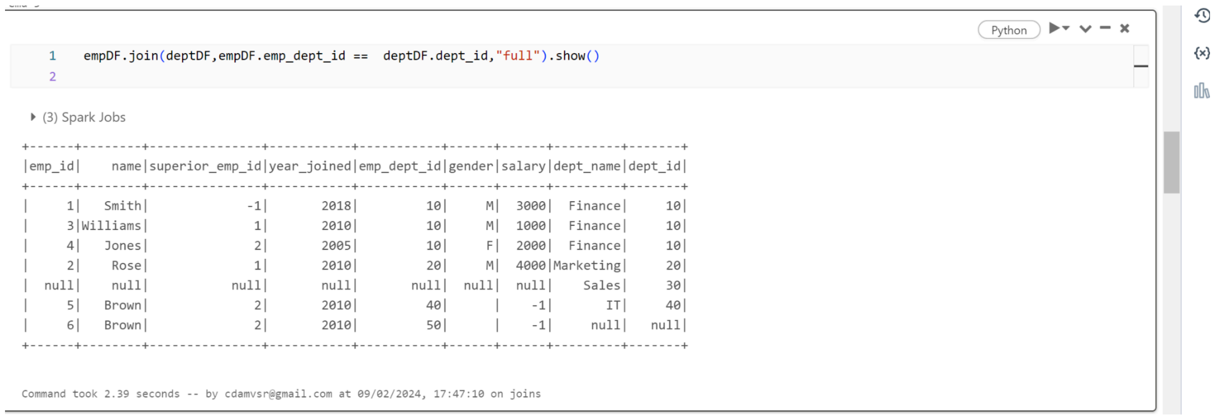
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer").show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|  null|    null|           null|       null|       null|  null|  null|    Sales|     30|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     null|   null|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Command took 2.75 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:08:29 on joins

—>



Python

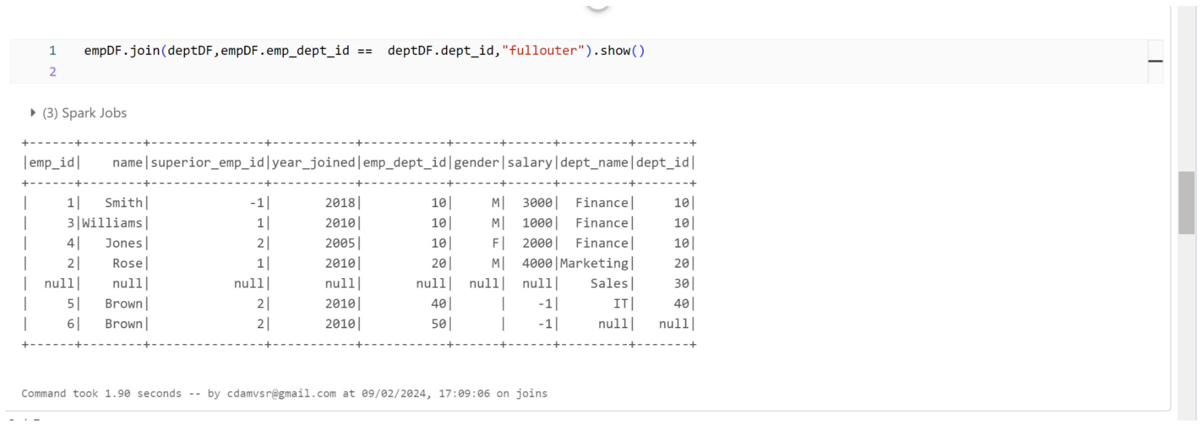
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"full").show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|  null|    null|           null|       null|       null|  null|  null|    Sales|     30|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     null|   null|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Command took 2.39 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:47:10 on joins

—> full outer join:



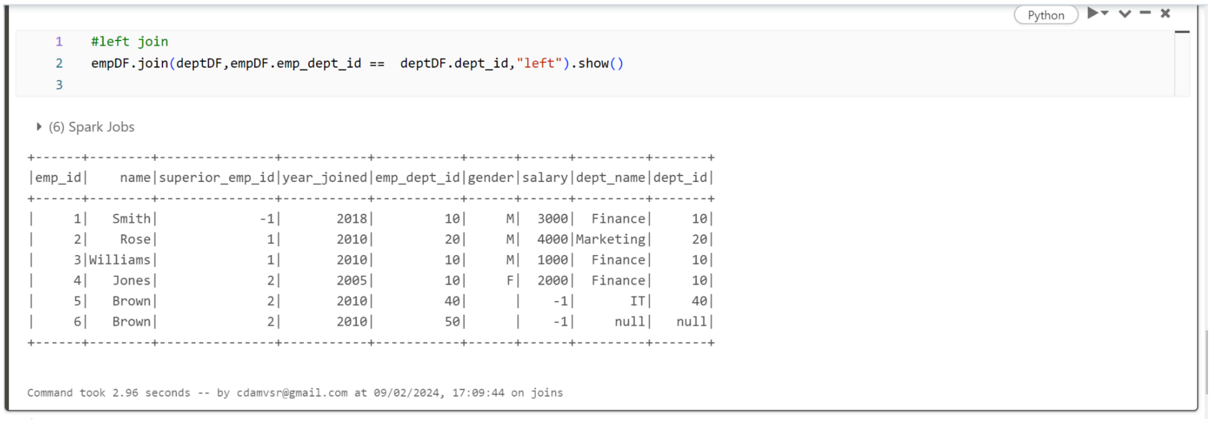
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"fullouter").show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|  null|    null|           null|       null|       null|  null|  null|    Sales|     30|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     null|   null|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Command took 1.90 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:09:06 on joins

iii)LEFT JOIN/ LEFT OUTER JOIN: Returns all rows from left dataset regardless of match found on right dataset, when Join doesn't match - it assigns null for that record.



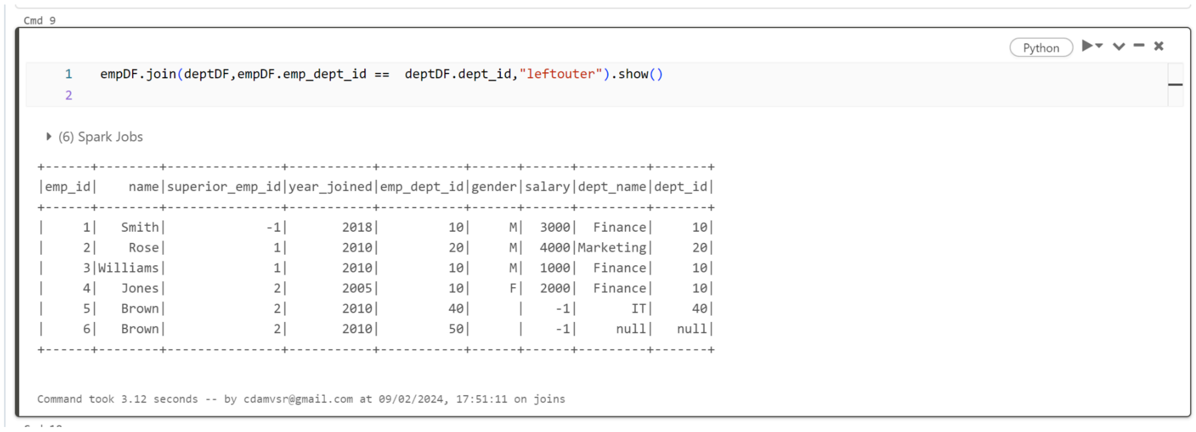
```
#left join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left").show()
```

▸ (6) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     null|   null|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

Command took 2.96 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:09:44 on joins
```

→left outer:



Cmd 9

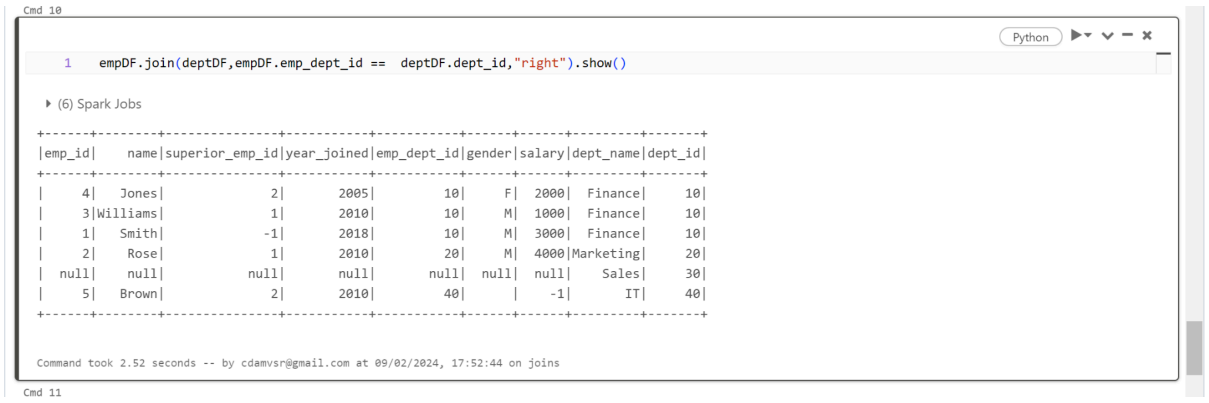
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftouter").show()
```

▸ (6) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
|     6|   Brown|              2|       2010|         50|      |    -1|     null|   null|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

Command took 3.12 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:51:11 on joins
```

iv)RIGHT JOIN/ RIGHT OUTER JOIN: Returns all rows from Right dataset regardless of match found on left dataset, when Join doesn't match - it assigns null for that record.

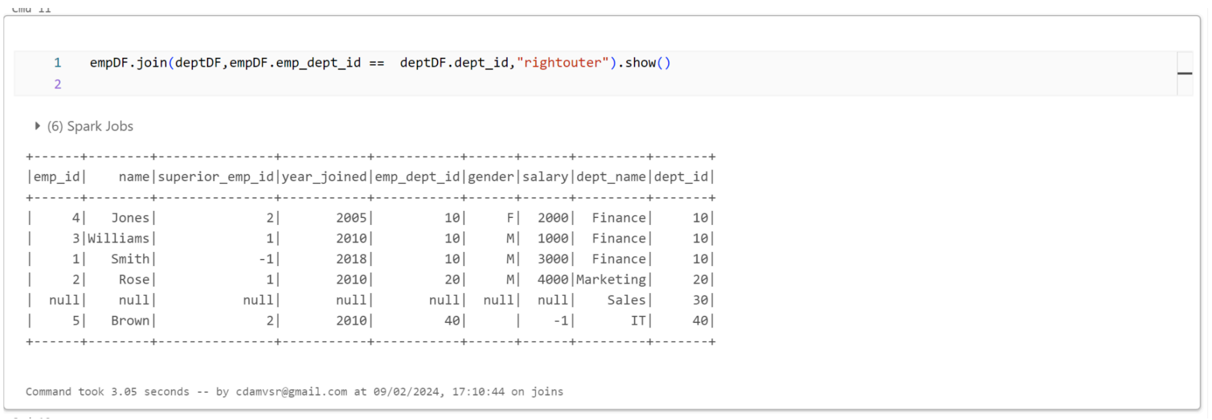


```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"right").show()
```

```
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|     4|   Jones|             2|       2005|         10|     F|  2000|  Finance|     10|
|     3|Williams|             1|       2010|         10|     M|  1000|  Finance|     10|
|     1|   Smith|            -1|       2018|         10|     M|  3000|  Finance|     10|
|     2|    Rose|             1|       2010|         20|     M|  4000|Marketing|     20|
|  null|    null|          null|       null|       null|  null|  null|    Sales|     30|
|     5|   Brown|             2|       2010|         40|      |    -1|       IT|     40|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+

Command took 2.52 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:52:44 on joins
```

—> right outer:

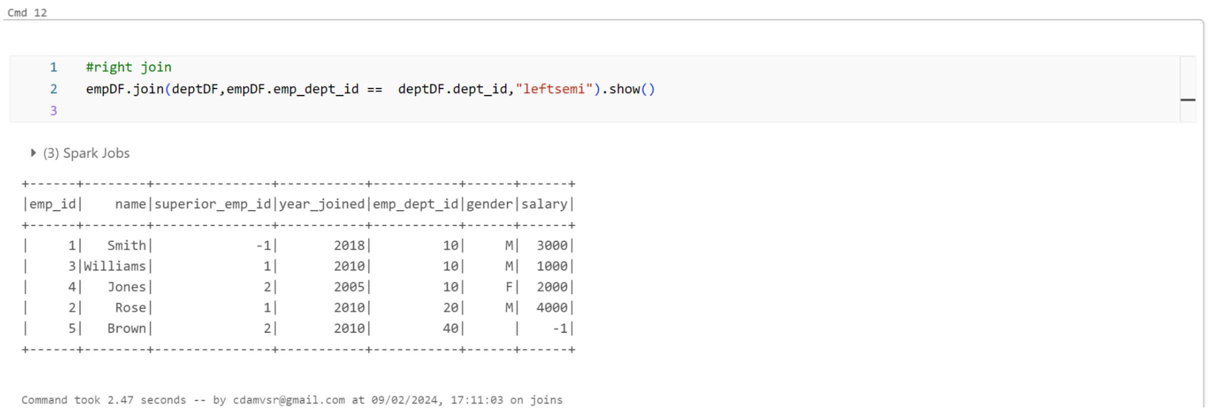


```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"rightouter").show()
```

```
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|     4|   Jones|             2|       2005|         10|     F|  2000|  Finance|     10|
|     3|Williams|             1|       2010|         10|     M|  1000|  Finance|     10|
|     1|   Smith|            -1|       2018|         10|     M|  3000|  Finance|     10|
|     2|    Rose|             1|       2010|         20|     M|  4000|Marketing|     20|
|  null|    null|          null|       null|       null|  null|  null|    Sales|     30|
|     5|   Brown|             2|       2010|         40|      |    -1|       IT|     40|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+

Command took 3.05 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:10:44 on joins
```

v)LEFT SEMI JOIN: Returns columns from the only left dataset for the matched records in the right dataset on join expression.



Cmd 12

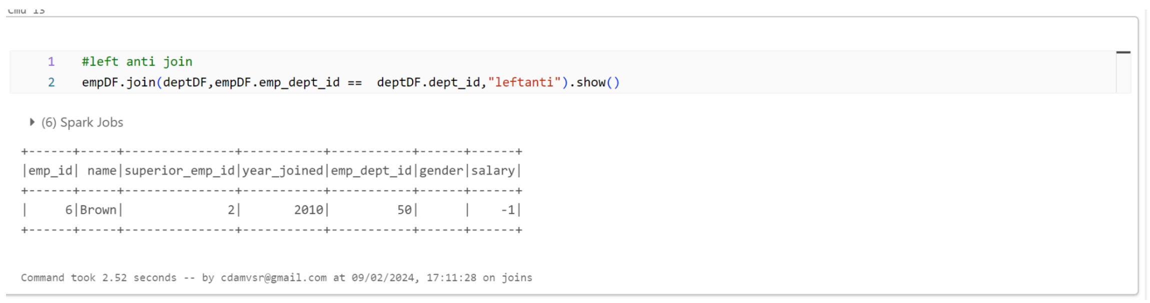
```
#right join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi").show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     3|Williams|              1|       2010|         10|     M|  1000|
|     4|   Jones|              2|       2005|         10|     F|  2000|
|     2|    Rose|              1|       2010|         20|     M|  4000|
|     5|   Brown|              2|       2010|         40|      |    -1|
+------+--------+---------------+-----------+-----------+------+------+
```

Command took 2.47 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:11:03 on joins

VI)LEFT ANTI JOIN: Returns only columns from the left dataset for non-matched records.



```
#left anti join
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftanti").show()
```

(6) Spark Jobs

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id| name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+
```

Command took 2.52 seconds -- by cdamvsr@gmail.com at 09/02/2024, 17:11:28 on joins