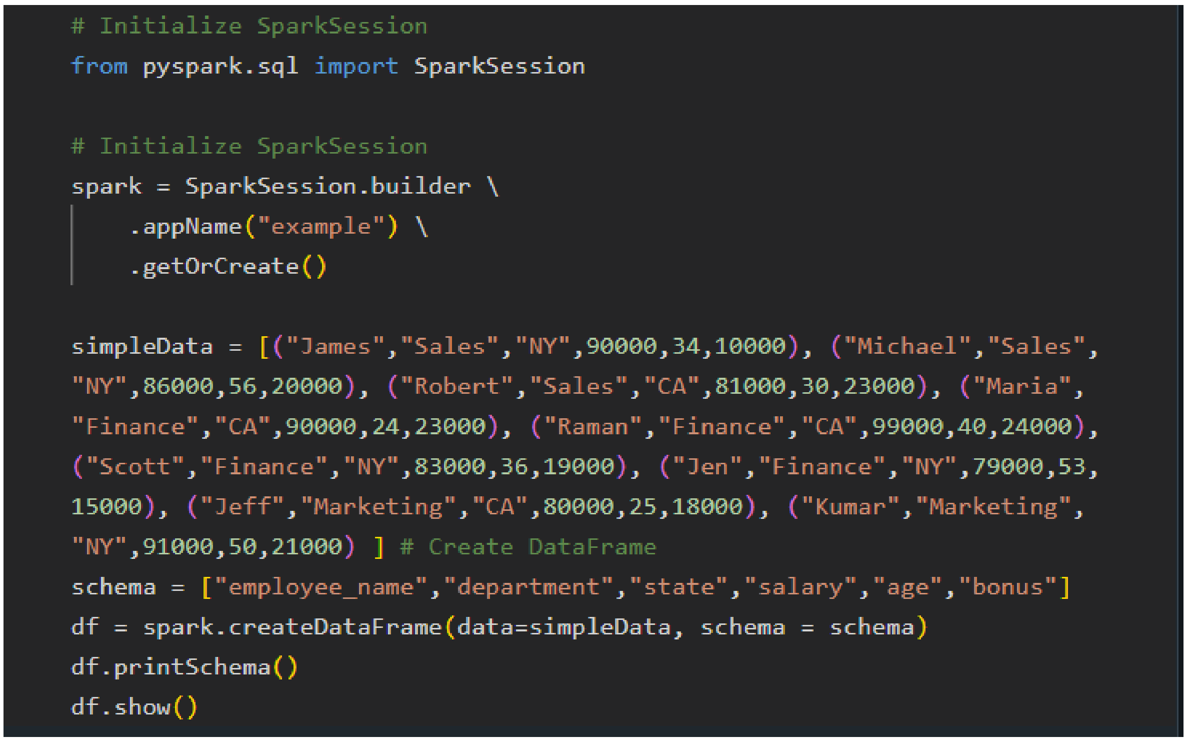
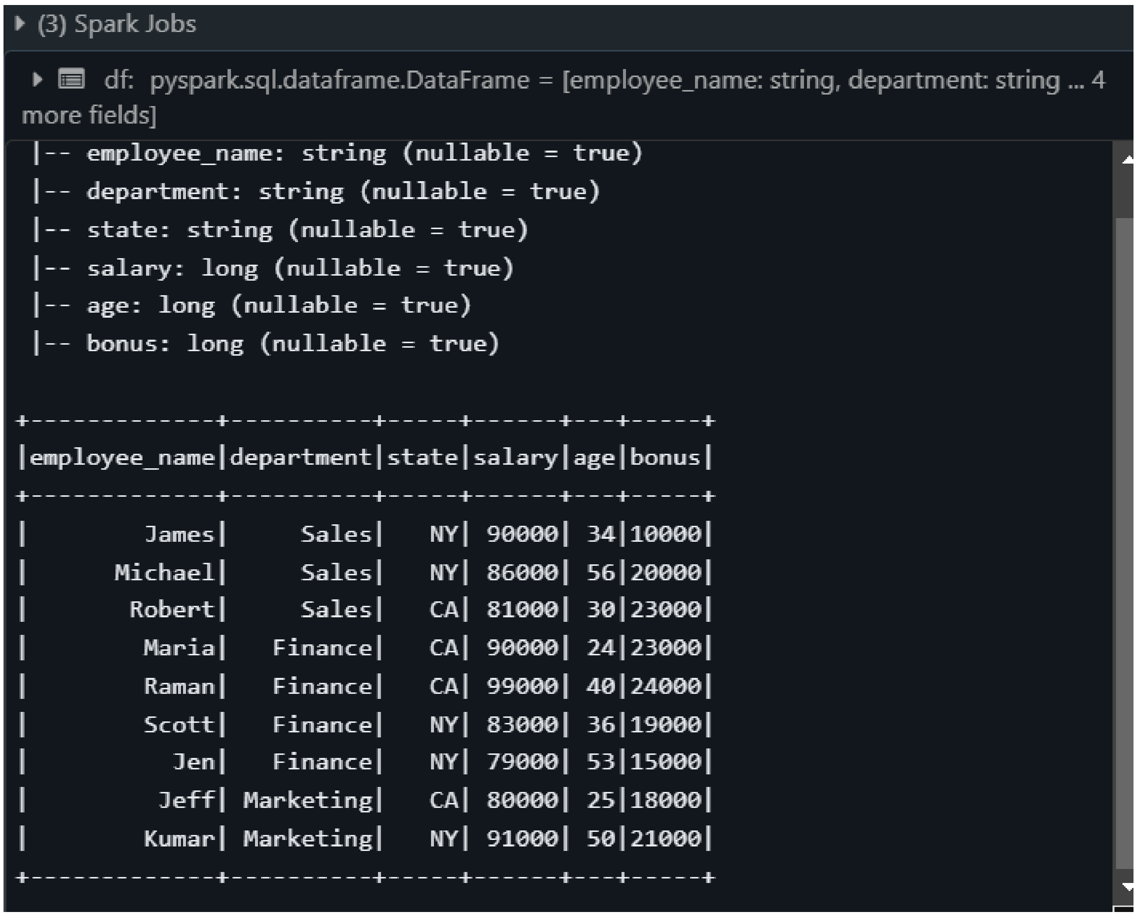
Pyspark day 4 Assignment

1.Creating a pyspark dataframe with sample data



```
# Initialize SparkSession
from pyspark.sql import SparkSession

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("example") \
    .getOrCreate()

simpleData = [("James","Sales","NY",90000,34,10000), ("Michael","Sales",
"NY",86000,56,20000), ("Robert","Sales","CA",81000,30,23000), ("Maria",
"Finance","CA",90000,24,23000), ("Raman","Finance","CA",99000,40,24000),
("Scott","Finance","NY",83000,36,19000), ("Jen","Finance","NY",79000,53,
15000), ("Jeff","Marketing","CA",80000,25,18000), ("Kumar","Marketing",
"NY",91000,50,21000) ] # Create DataFrame
schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
df.printSchema()
df.show()
```

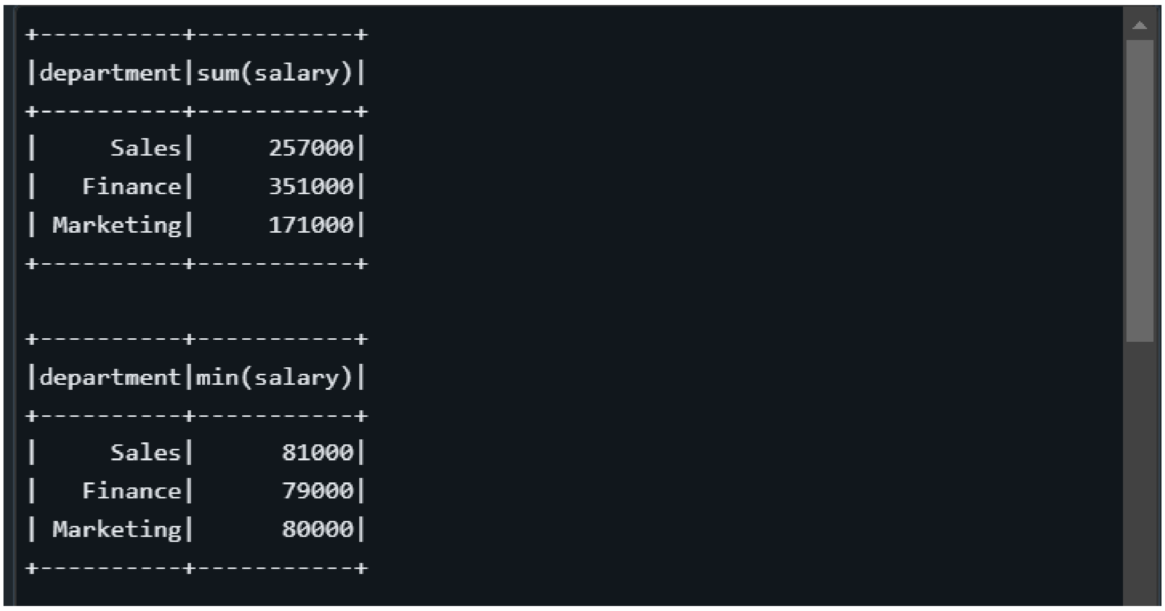


```
(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [employee_name: string, department: string ... 4
more fields]
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- state: string (nullable = true)
 |-- salary: long (nullable = true)
 |-- age: long (nullable = true)
 |-- bonus: long (nullable = true)

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        James|     Sales|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
+-------------+----------+-----+------+---+-----+
```

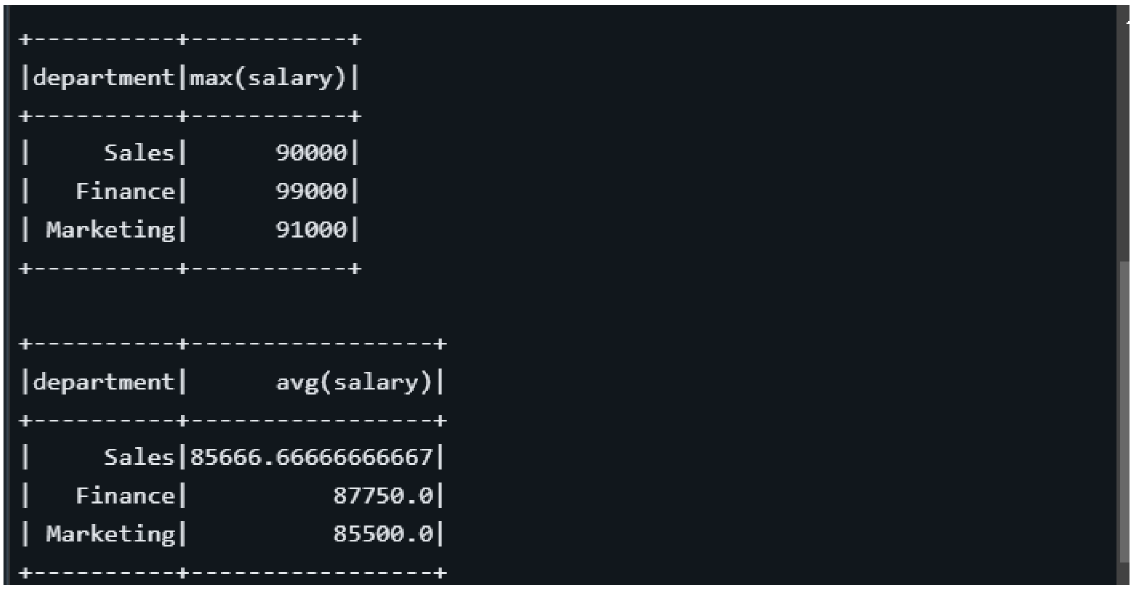
2.Using aggregate functions like sum, mean, avg, max, min, count

Basically these functions are used to add the values, select min of values, maximum of values, average of values and count the number of records respectively based on condition given in group by

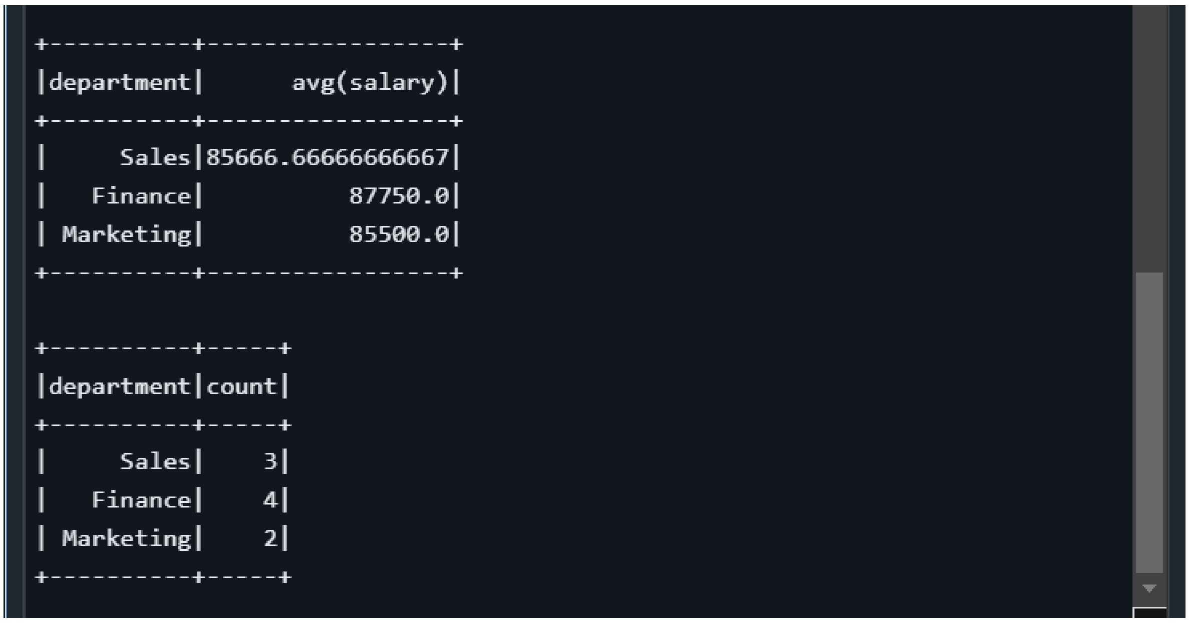
```
data=df.groupBy("department").sum("salary").show()
data=df.groupBy("department").min("salary").show()
data=df.groupBy("department").max("salary").show()
data=df.groupBy("department").avg("salary").show()
data=df.groupBy("department").mean("salary").show()
data=df.groupBy("department").count().show()
```



```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+

+----------+-----------+
|department|min(salary)|
+----------+-----------+
|     Sales|      81000|
|   Finance|      79000|
| Marketing|      80000|
+----------+-----------+
```

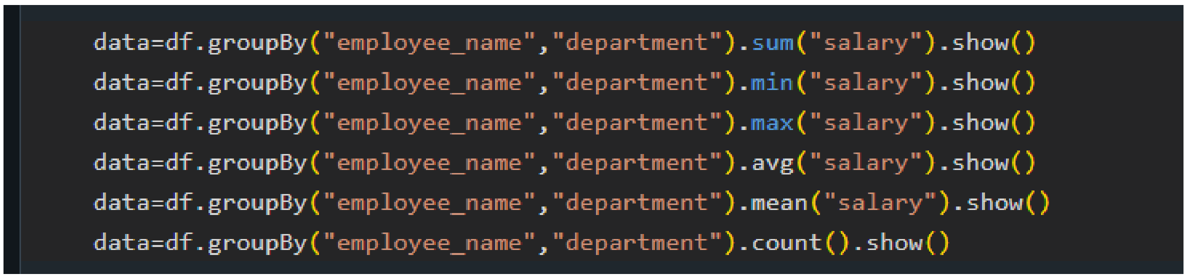


```
+----------+-----------+
|department|max(salary)|
+----------+-----------+
|     Sales|      90000|
|   Finance|      99000|
| Marketing|      91000|
+----------+-----------+

+----------+-----------------+
|department|      avg(salary)|
+----------+-----------------+
|     Sales|85666.66666666667|
|   Finance|          87750.0|
| Marketing|          85500.0|
+----------+-----------------+
```

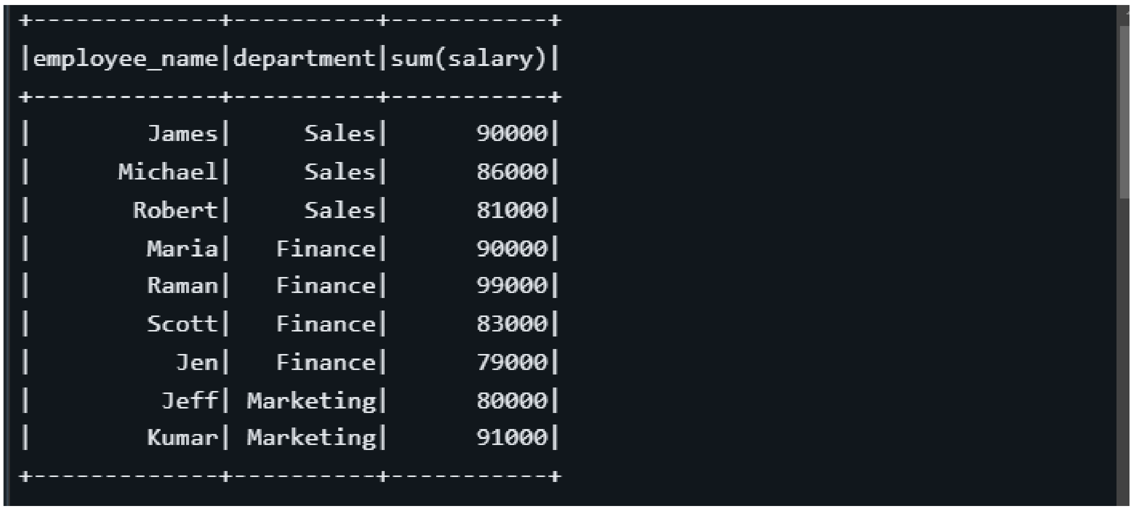


```
+----------+-----------------+
|department|      avg(salary)|
+----------+-----------------+
|     Sales|85666.66666666667|
|   Finance|          87750.0|
| Marketing|          85500.0|
+----------+-----------------+

+----------+-----+
|department|count|
+----------+-----+
|     Sales|    3|
|   Finance|    4|
| Marketing|    2|
+----------+-----+
```

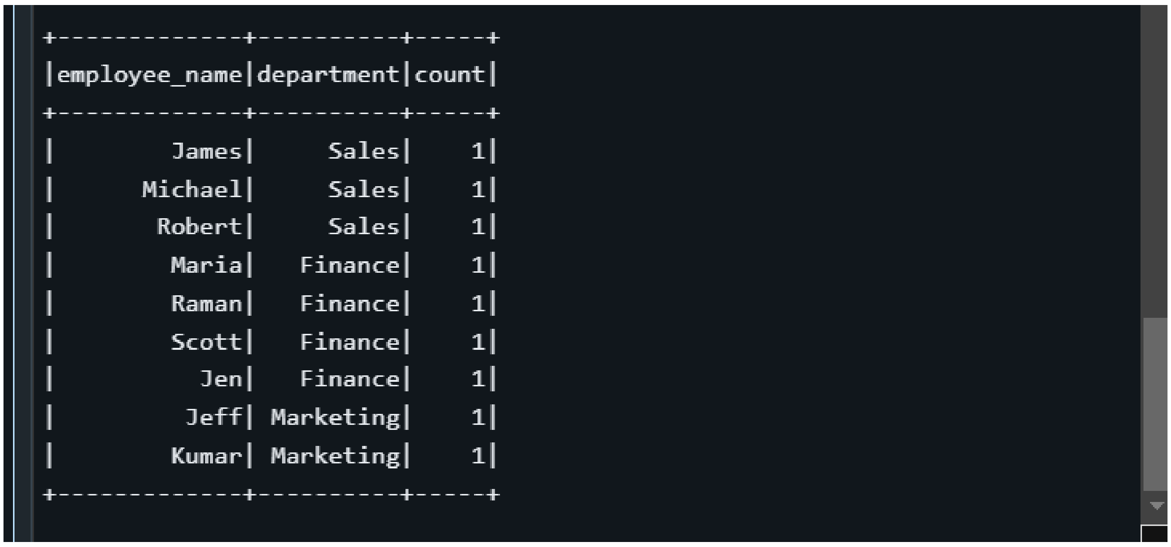
3.Using group by and agg functions with multiple arguments



```
data=df.groupBy("employee_name","department").sum("salary").show()
data=df.groupBy("employee_name","department").min("salary").show()
data=df.groupBy("employee_name","department").max("salary").show()
data=df.groupBy("employee_name","department").avg("salary").show()
data=df.groupBy("employee_name","department").mean("salary").show()
data=df.groupBy("employee_name","department").count().show()
```

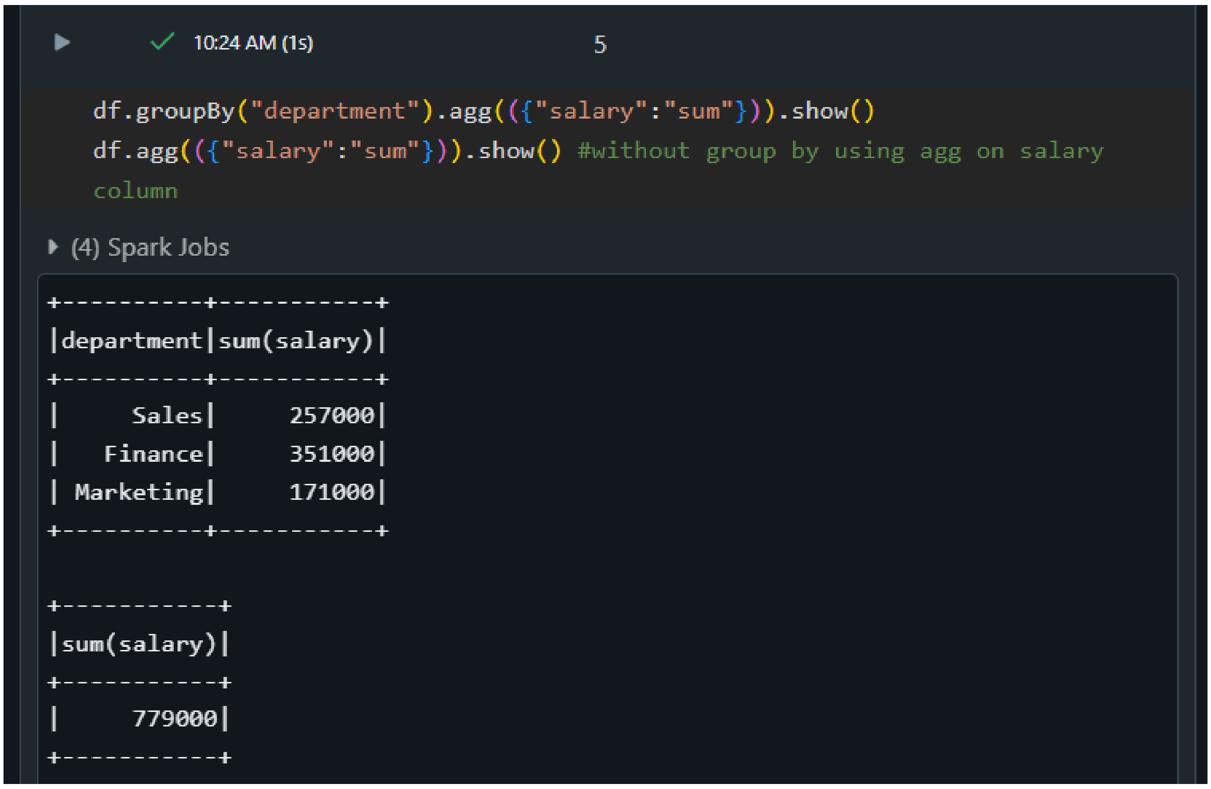


```
+-------------+----------+-----------+
|employee_name|department|sum(salary)|
+-------------+----------+-----------+
|        James|     Sales|      90000|
|      Michael|     Sales|      86000|
|       Robert|     Sales|      81000|
|        Maria|   Finance|      90000|
|        Raman|   Finance|      99000|
|        Scott|   Finance|      83000|
|          Jen|   Finance|      79000|
|         Jeff| Marketing|      80000|
|        Kumar| Marketing|      91000|
+-------------+----------+-----------+
```



```
+-------------+----------+-----+
|employee_name|department|count|
+-------------+----------+-----+
|        James|     Sales|    1|
|      Michael|     Sales|    1|
|       Robert|     Sales|    1|
|        Maria|   Finance|    1|
|        Raman|   Finance|    1|
|        Scott|   Finance|    1|
|          Jen|   Finance|    1|
|         Jeff| Marketing|    1|
|        Kumar| Marketing|    1|
+-------------+----------+-----+
```

4.Aggregate function with and without using group by



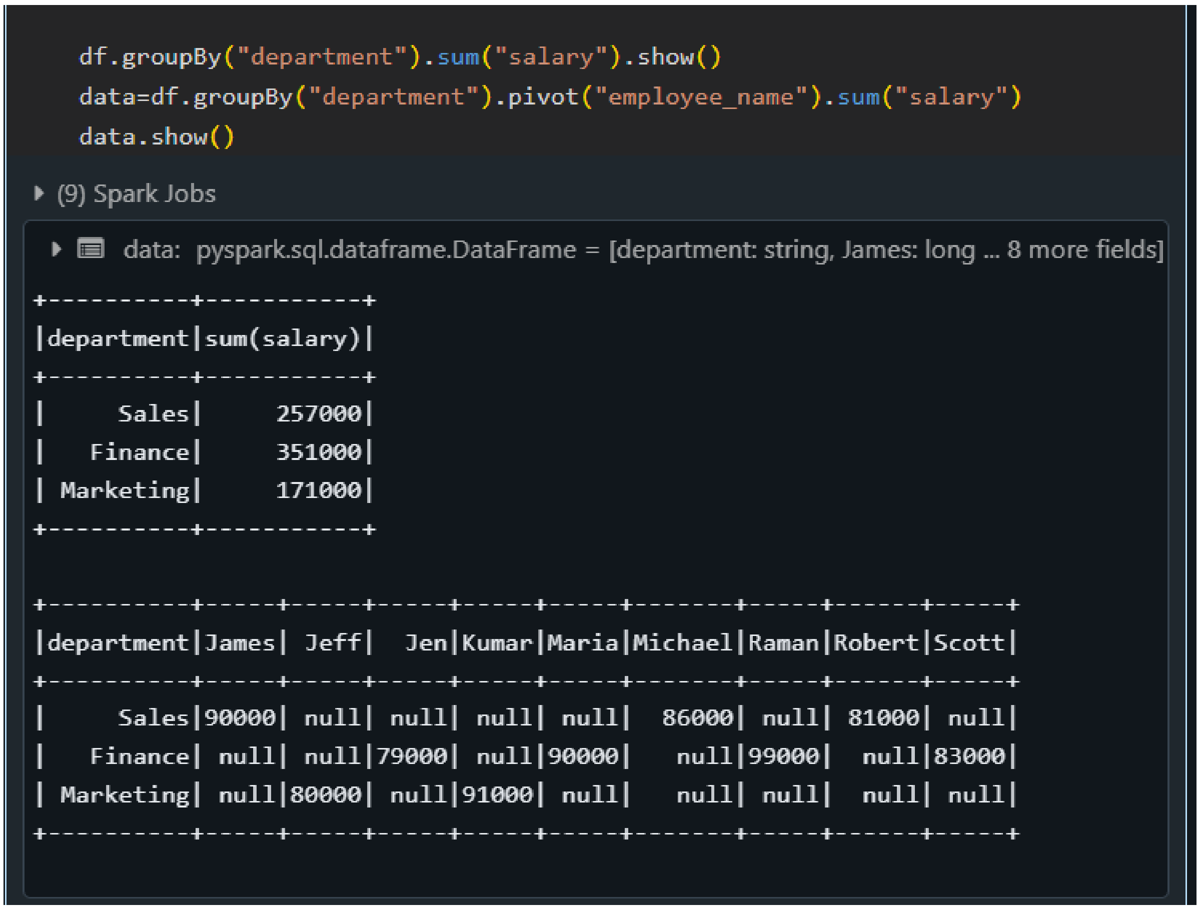
10:24 AM (1s) 5

```
df.groupBy("department").agg(({"salary":"sum"})).show()
df.agg(({"salary":"sum"})).show() #without group by using agg on salary
column
```

(4) Spark Jobs

```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+


+-----------+
|sum(salary)|
+-----------+
|     779000|
+-----------+
```

5. pivot() function is an aggregation function used to rotate data from one column to multiple columns(Transpose row into columns)



```
df.groupBy("department").sum("salary").show()
data=df.groupBy("department").pivot("employee_name").sum("salary")
data.show()
```

(9) Spark Jobs

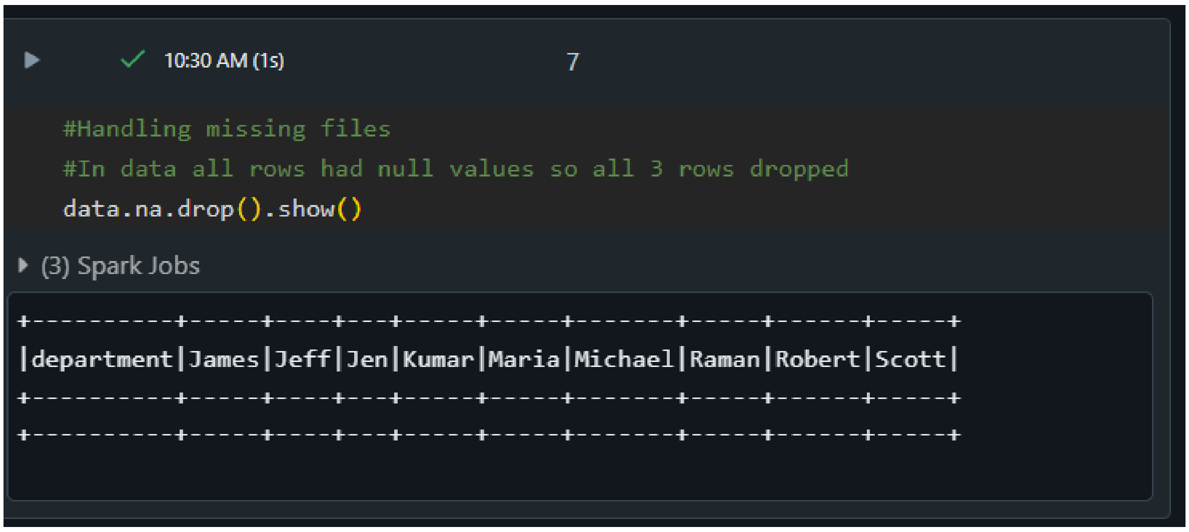
data: pyspark.sql.dataframe.DataFrame = [department: string, James: long ... 8 more fields]

```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+


+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
|department|James| Jeff|  Jen|Kumar|Maria|Michael|Raman|Robert|Scott|
+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
|     Sales|90000| null| null| null| null|  86000| null| 81000| null|
|   Finance| null| null|79000| null|90000|   null|99000|  null|83000|
| Marketing| null|80000| null|91000| null|   null| null|  null| null|
+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
```

6.Missing handling files

This is a process of Data cleaning in order to handle null values.In pyspark databricks,null is not specified,None is used.

.na.drop() functions deletes the row which contains null values

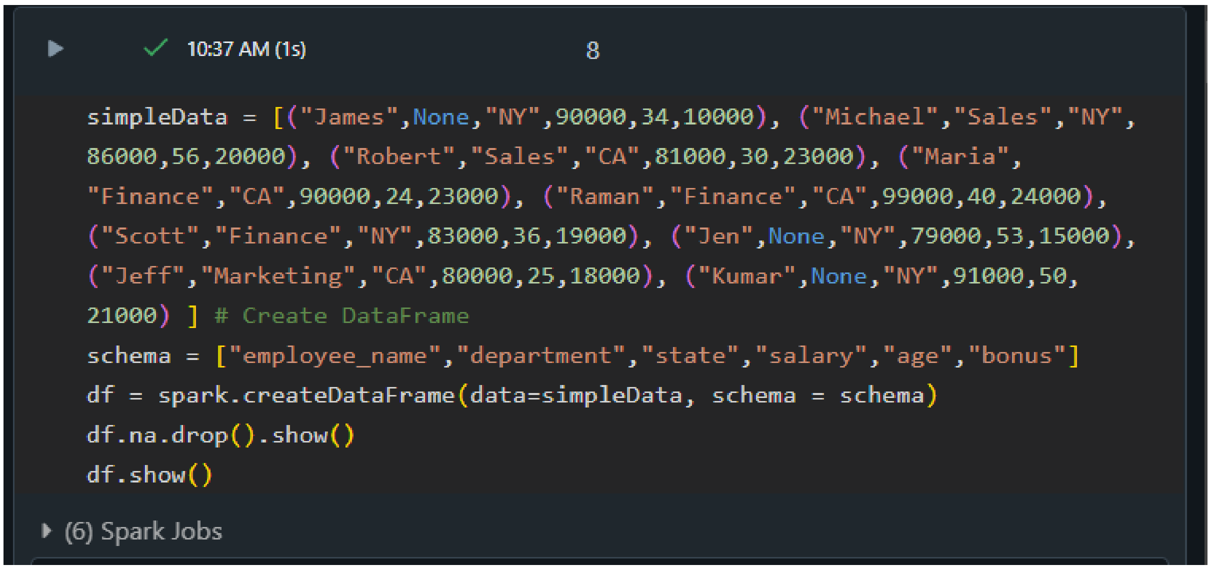


10:30 AM (1s)   7

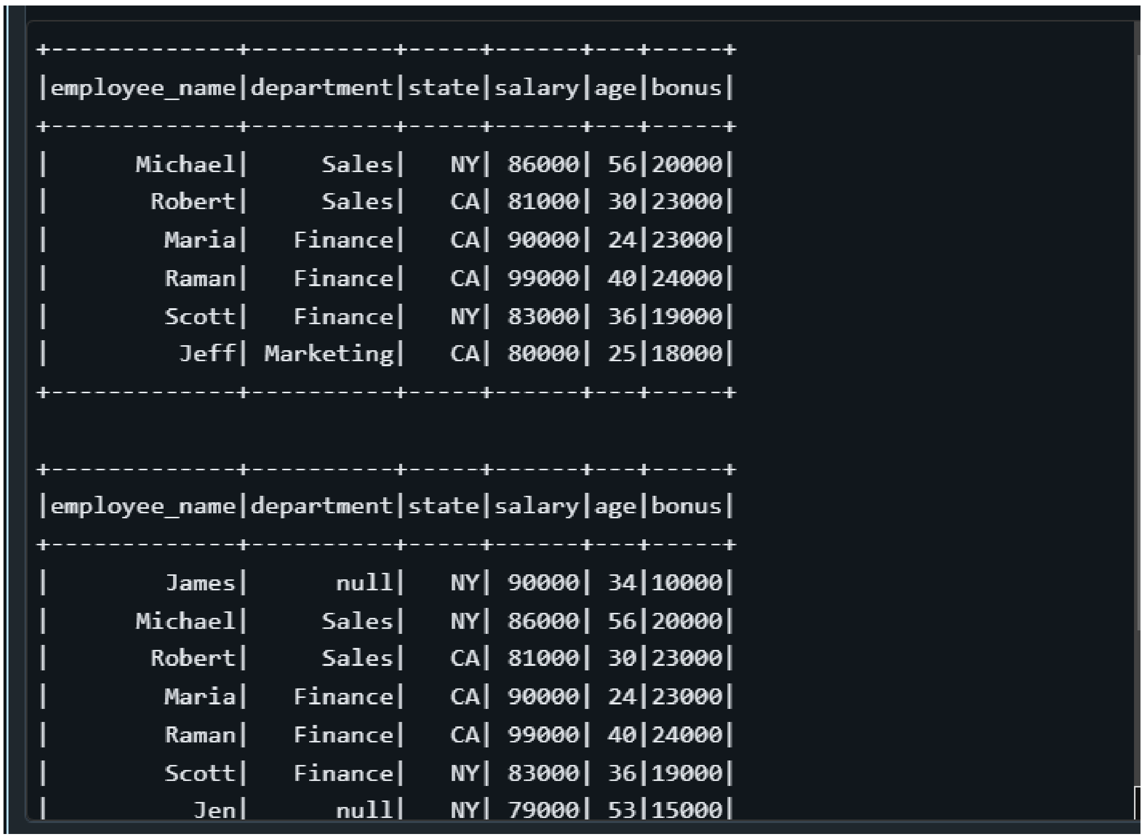
```
#Handling missing files
#In data all rows had null values so all 3 rows dropped
data.na.drop().show()
```

(3) Spark Jobs

```
+----------+-----+----+---+-----+-----+-------+-----+------+-----+
|department|James|Jeff|Jen|Kumar|Maria|Michael|Raman|Robert|Scott|
+----------+-----+----+---+-----+-----+-------+-----+------+-----+
+----------+-----+----+---+-----+-----+-------+-----+------+-----+
```

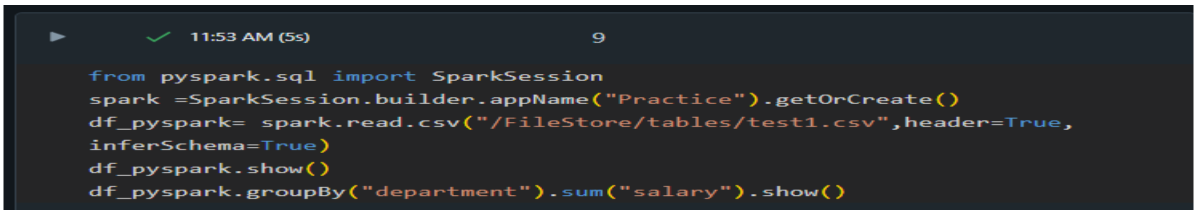


```
simpleData = [("James",None,"NY",90000,34,10000), ("Michael","Sales","NY",
86000,56,20000), ("Robert","Sales","CA",81000,30,23000), ("Maria",
"Finance","CA",90000,24,23000), ("Raman","Finance","CA",99000,40,24000),
("Scott","Finance","NY",83000,36,19000), ("Jen",None,"NY",79000,53,15000),
("Jeff","Marketing","CA",80000,25,18000), ("Kumar",None,"NY",91000,50,
21000) ] # Create DataFrame
schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
df.na.drop().show()
df.show()
```

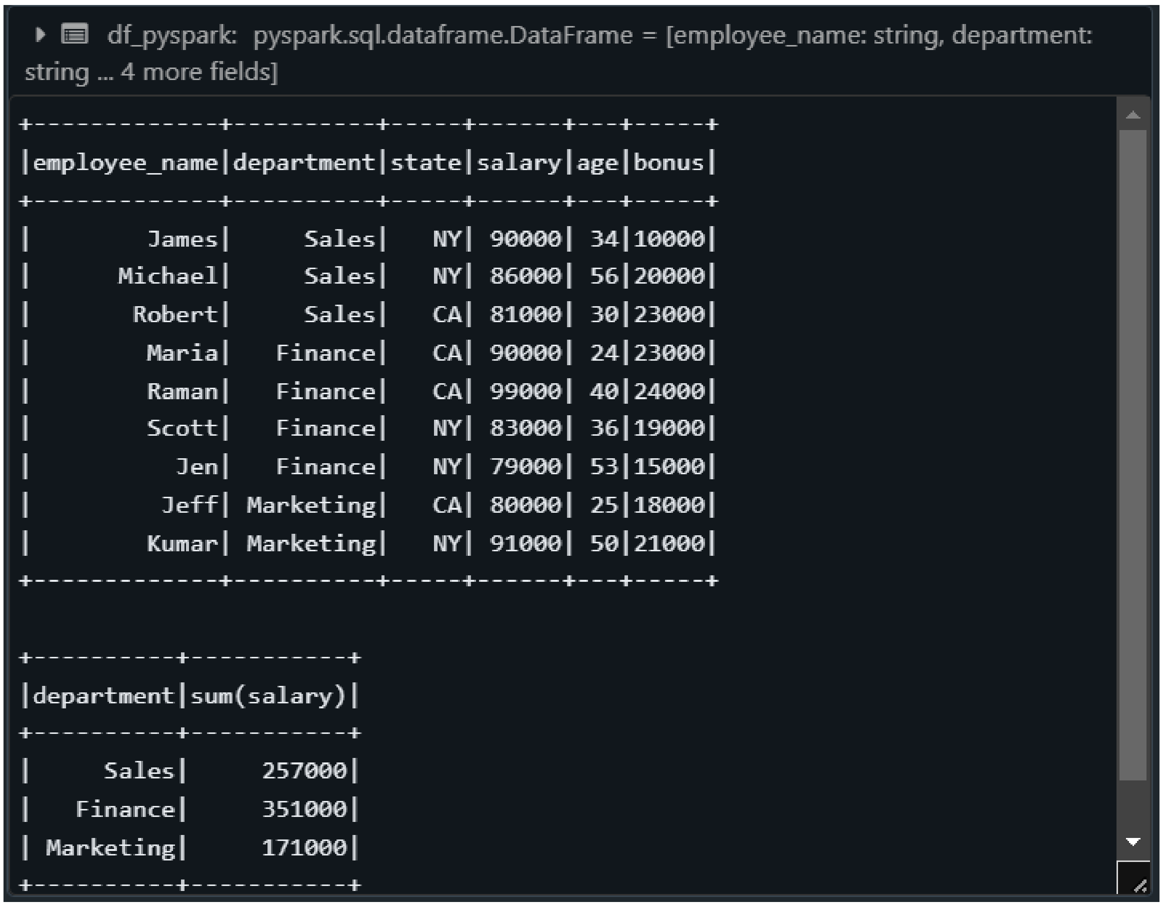


```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        James|      null|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|          Jen|      null|   NY| 79000| 53|15000|
```

7.Importing csv file and performing aggregate function on the data

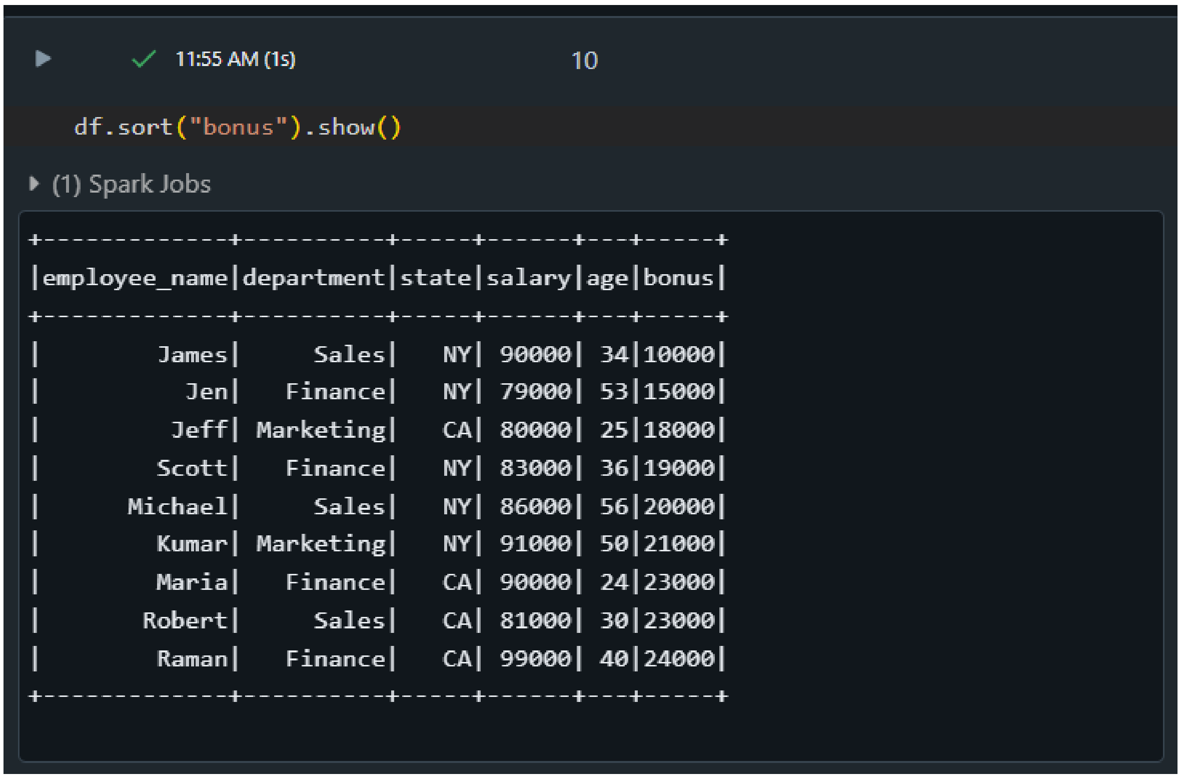


```
from pyspark.sql import SparkSession
spark =SparkSession.builder.appName("Practice").getOrCreate()
df_pyspark= spark.read.csv("/FileStore/tables/test1.csv",header=True,
inferSchema=True)
df_pyspark.show()
df_pyspark.groupBy("department").sum("salary").show()
```


df_pyspark: pyspark.sql.dataframe.DataFrame = [employee_name: string, department: string ... 4 more fields]

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        James|     Sales|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
+-------------+----------+-----+------+---+-----+


+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+
```

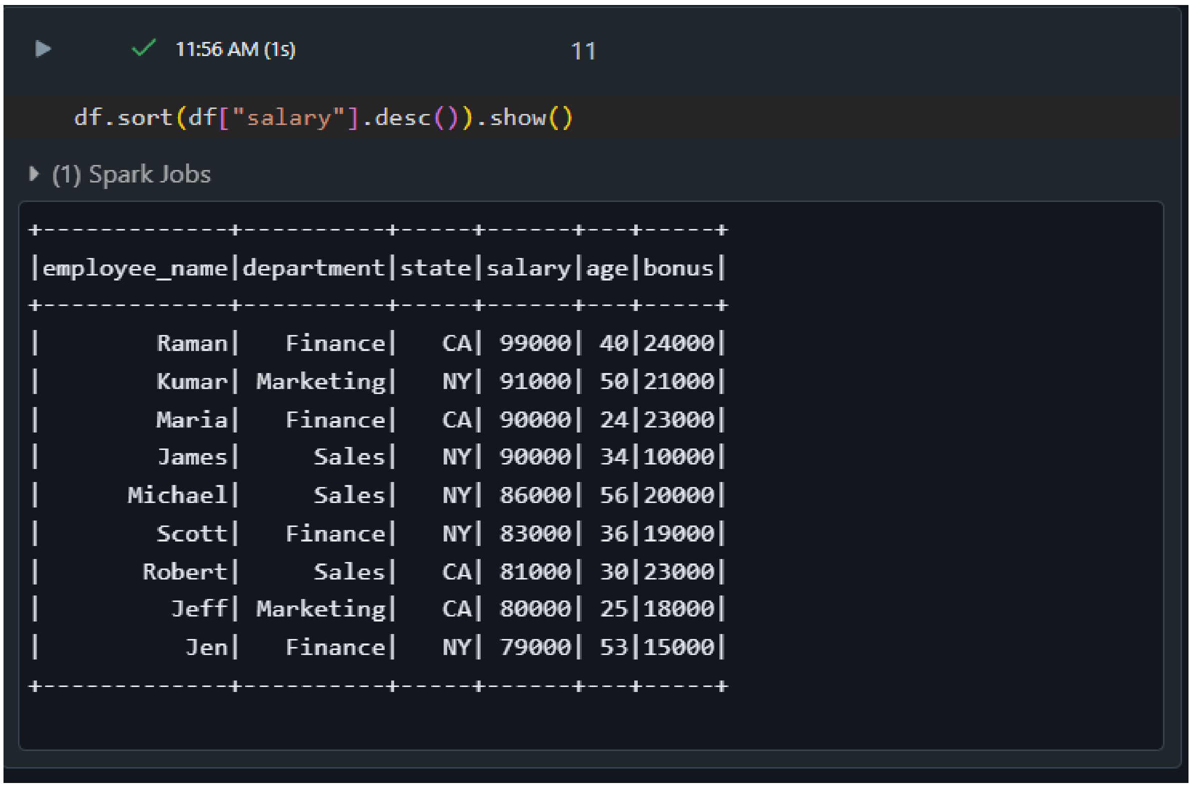
8.Performing sorting function--→used to sort data based on the values of specified column


11:55 AM (1s) 10

```
df.sort("bonus").show()
```

(1) Spark Jobs

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        James|     Sales|   NY| 90000| 34|10000|
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
+-------------+----------+-----+------+---+-----+
```



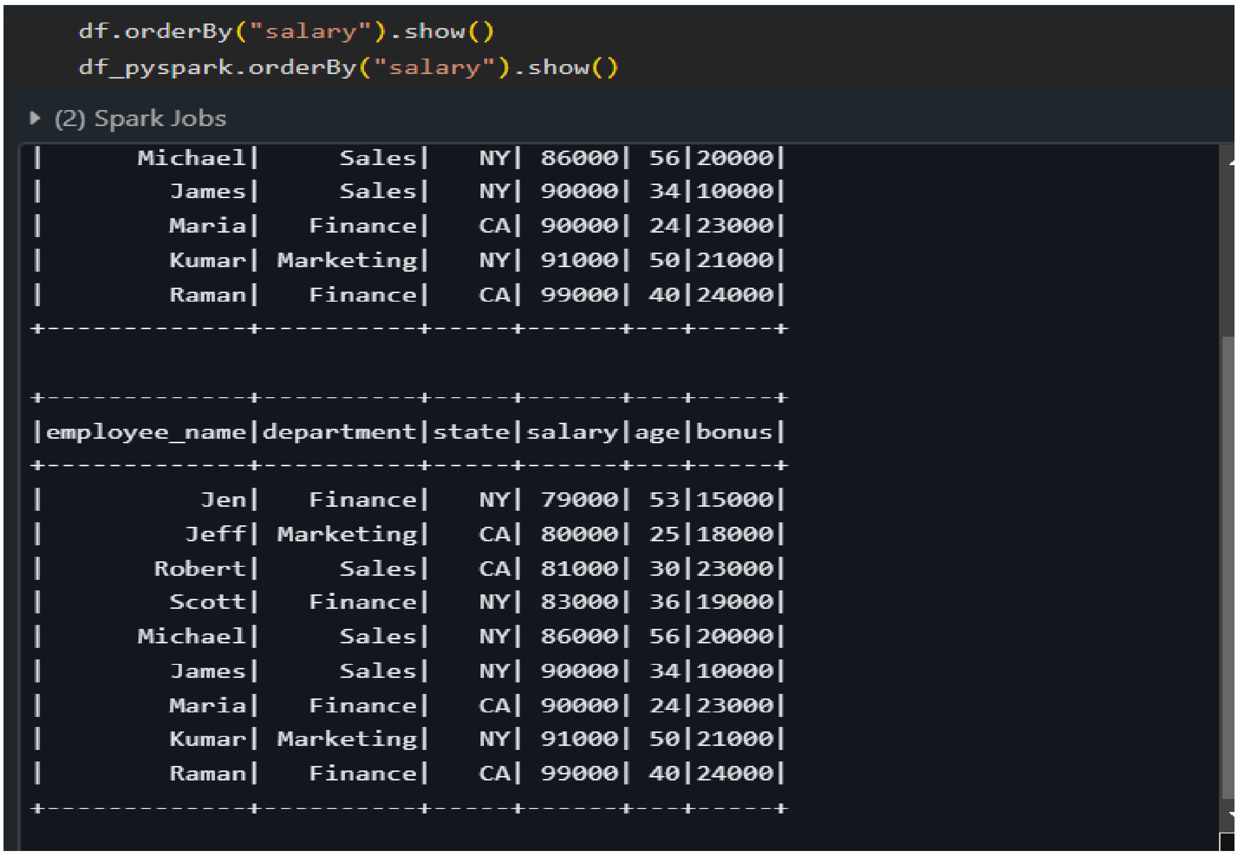
11:56 AM (1s) 11

```
df.sort(df["salary"].desc()).show()
```

(1) Spark Jobs

```
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        James|     Sales|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|          Jen|   Finance|   NY| 79000| 53|15000|
+-------------+----------+-----+------+---+-----+
```

Order By function is same as sort by function

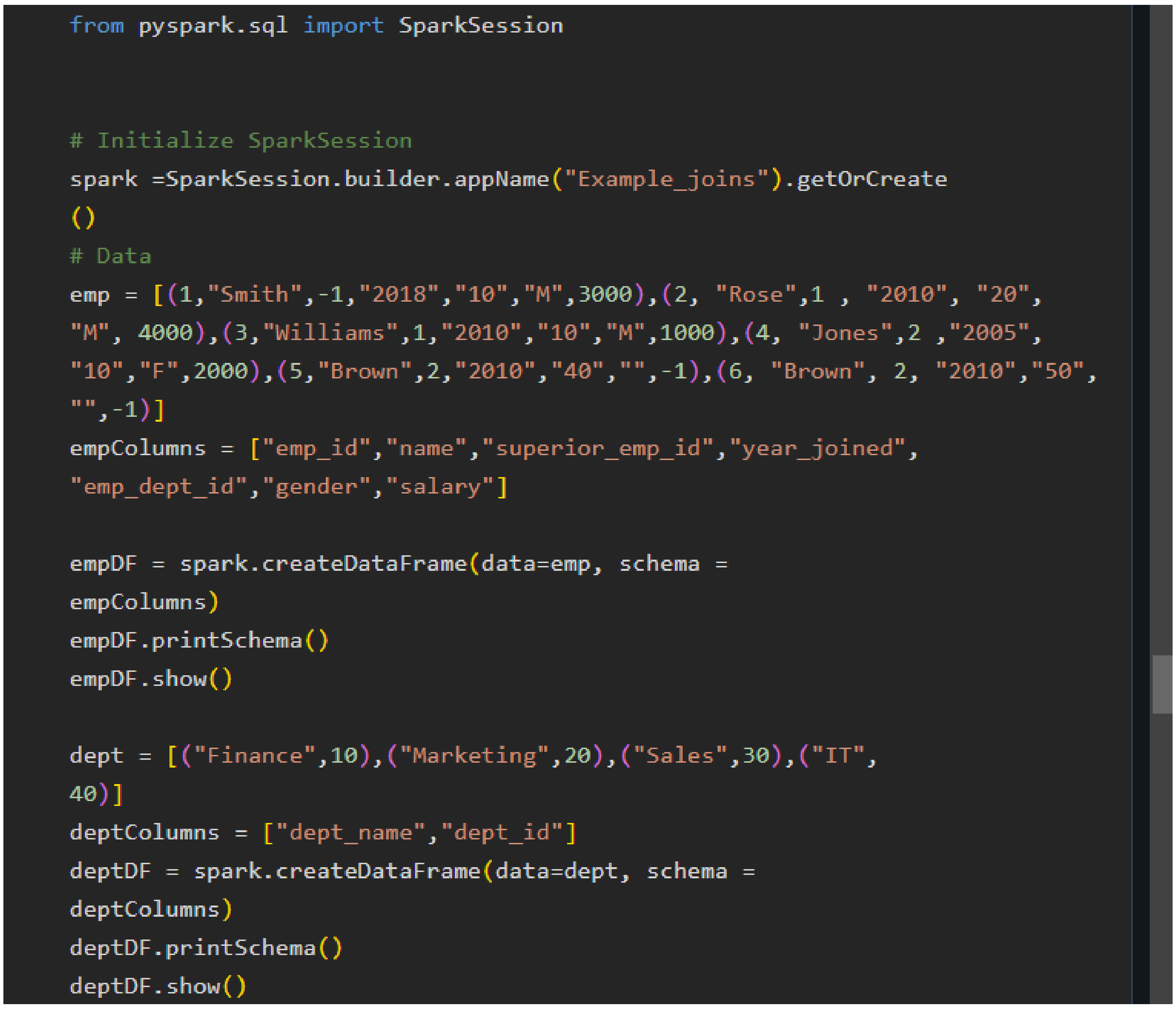


```
df.orderBy("salary").show()
df_pyspark.orderBy("salary").show()
```

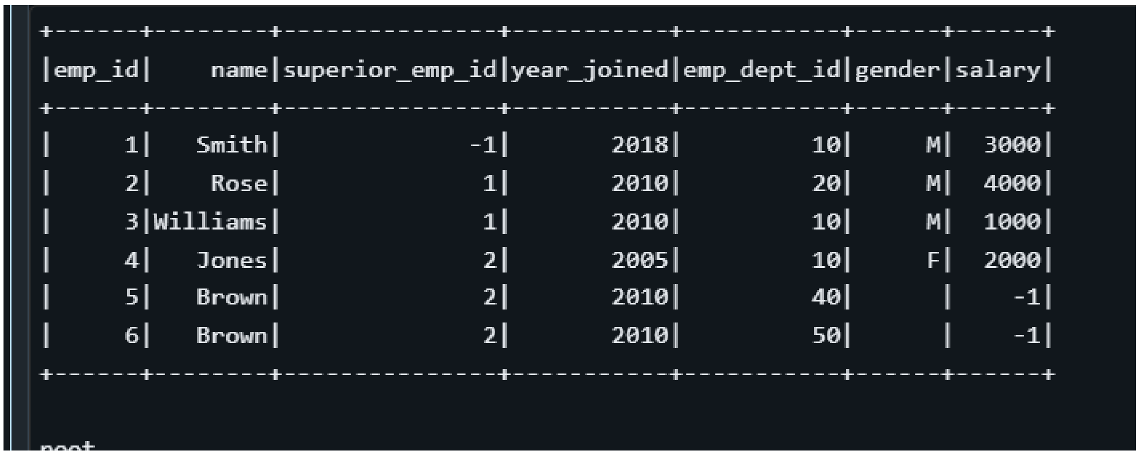
(2) Spark Jobs

```
|      Michael|     Sales|   NY| 86000| 56|20000|
|        James|     Sales|   NY| 90000| 34|10000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
|        Raman|   Finance|   CA| 99000| 40|24000|
+-------------+----------+-----+------+---+-----+

+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|        James|     Sales|   NY| 90000| 34|10000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
|        Raman|   Finance|   CA| 99000| 40|24000|
+-------------+----------+-----+------+---+-----+
```

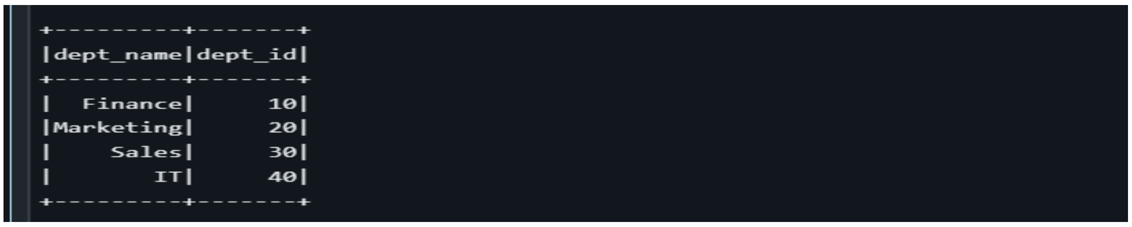
Joins between dataframes



```
from pyspark.sql import SparkSession


# Initialize SparkSession
spark =SparkSession.builder.appName("Example_joins").getOrCreate
()
# Data
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()
```

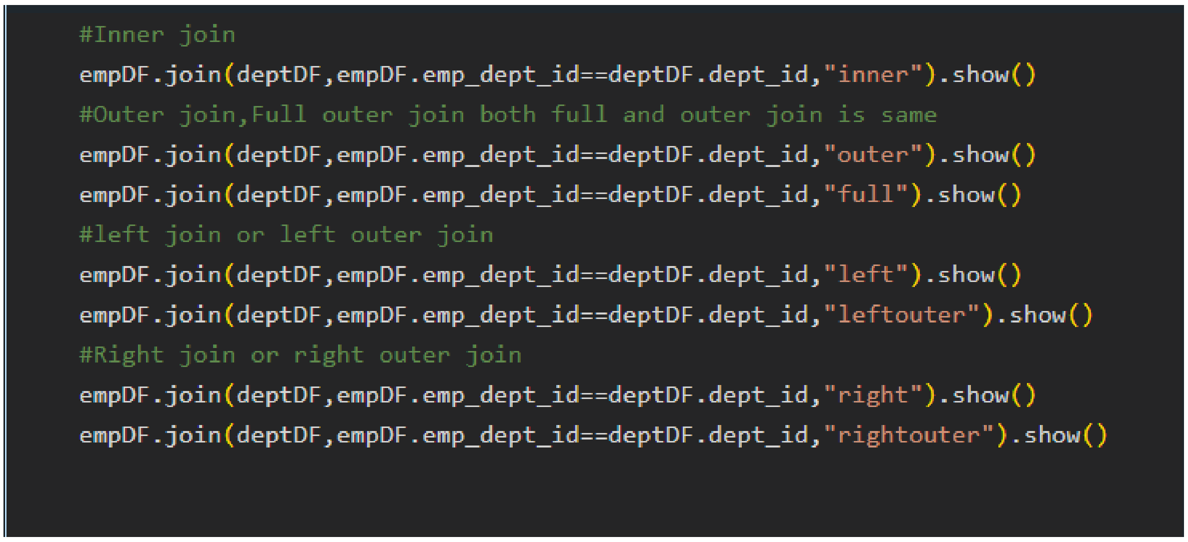


```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     2|    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|
+------+--------+---------------+-----------+-----------+------+------+


root
```



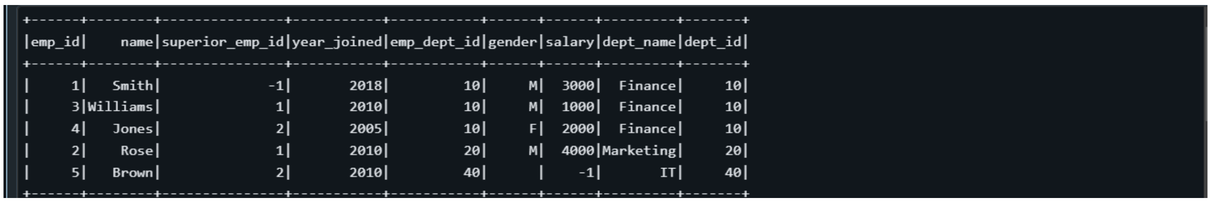
```
+---------+-------+
|dept_name|dept_id|
+---------+-------+
|  Finance|     10|
|Marketing|     20|
|    Sales|     30|
|       IT|     40|
+---------+-------+
```

Join Type	Description
Inner Join	Join records when key columns are matched, and dropped when they are not matched
Outer join	Returns all rows from both datasets, where Join expression doesn't match it returns null or respective columns
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
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
Left Semi Join	Returns columns from the only left dataset for the matched records in the right dataset on join expression
Left Anti Join	Returns only columns from left dataset for non-matched records



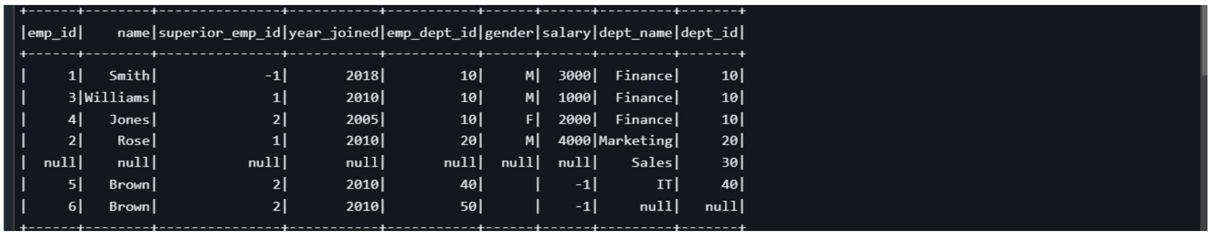
```
#Inner join
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"inner").show()
#Outer join,Full outer join both full and outer join is same
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"outer").show()
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"full").show()
#left join or left outer join
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"left").show()
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"leftouter").show()
#Right join or right outer join
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"right").show()
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"rightouter").show()
```

1.1.Inner join output



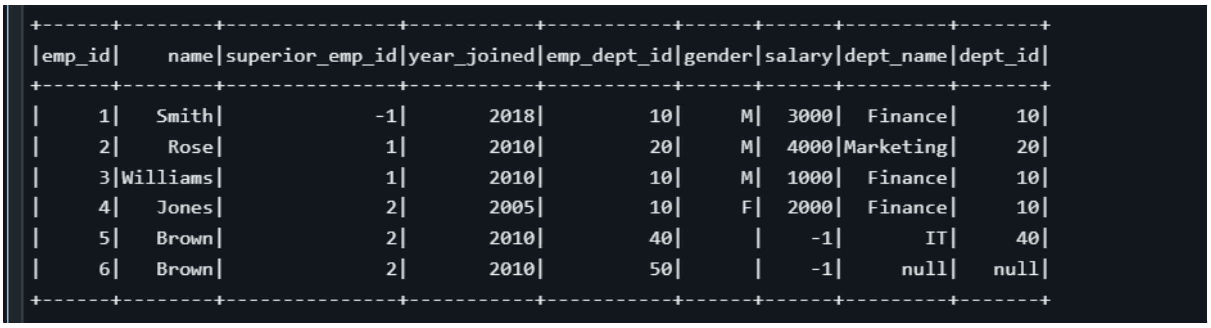
```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

1.2.Outer join output



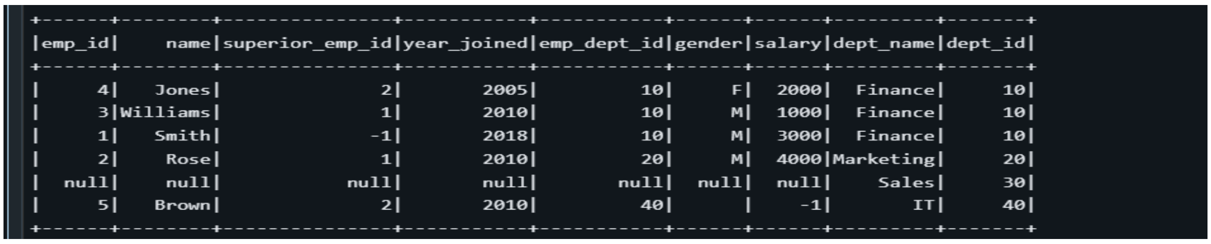
```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

1.3.Left join Output



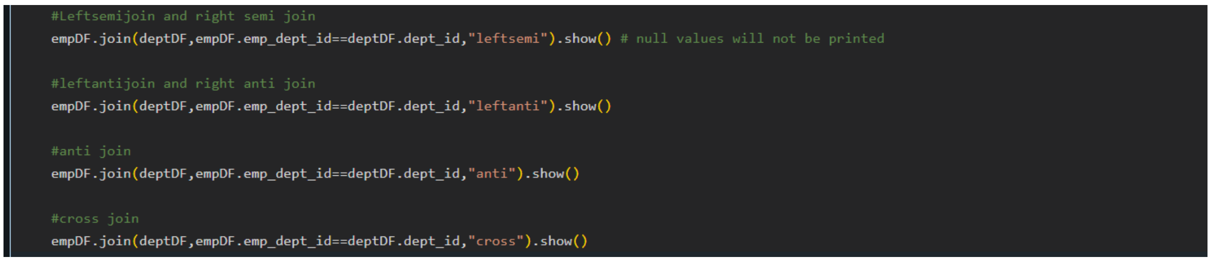
```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|             -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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

1.4.Right join Output



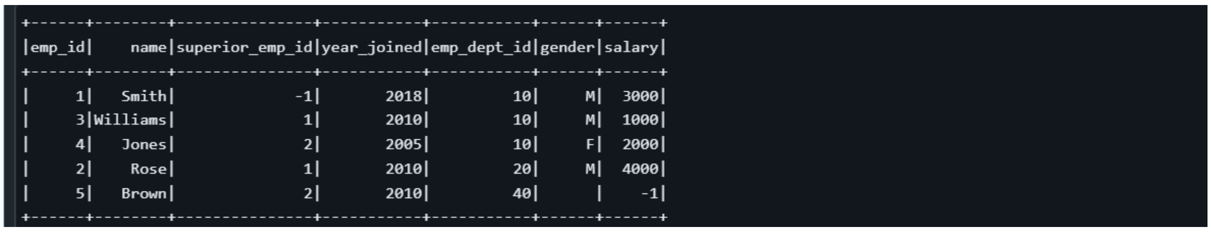
```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|     4|   Jones|              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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Left semi,Left Anti, Anti, Cross joins



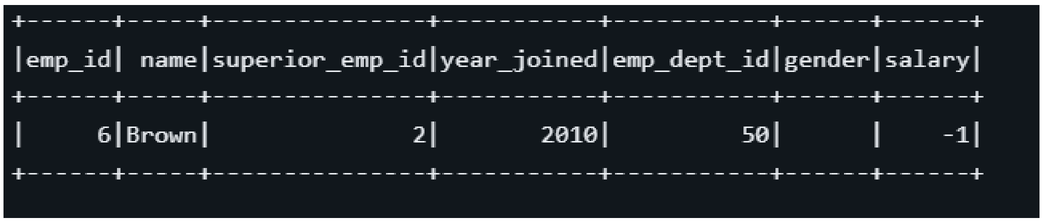
```
#Leftsemijoin and right semi join
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"leftsemi").show() # null values will not be printed

#leftantijoin and right anti join
empDF.join(deptDF,empDF.emp_dept_id==deptDF.dept_id,"leftanti").show()

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

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

1.5.Left semi join



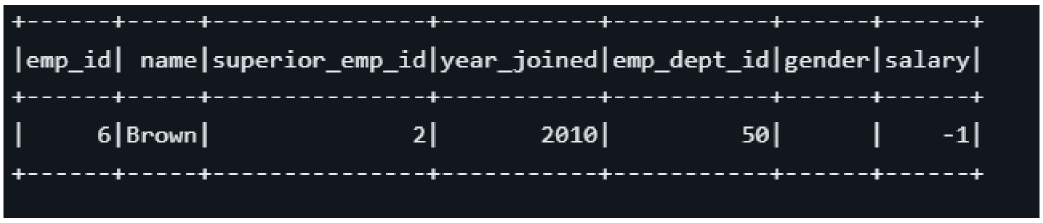
```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     3|Williams|              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|
+------+--------+---------------+-----------+-----------+------+------+
```

1.6.Leftanti join



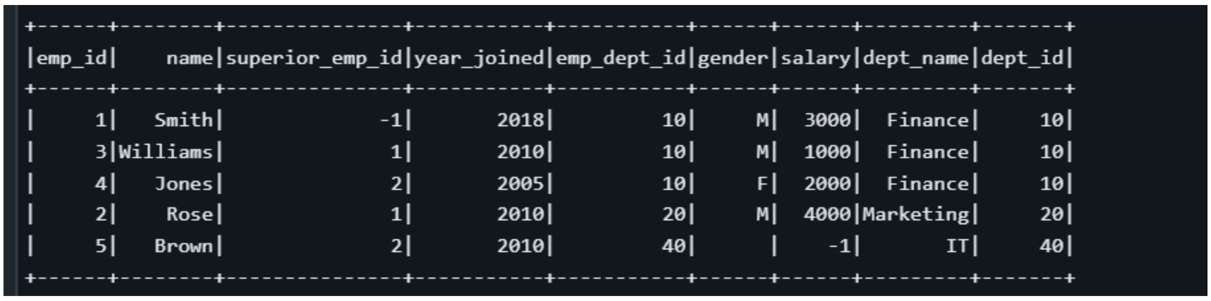
```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id| name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+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

1.7.Anti join



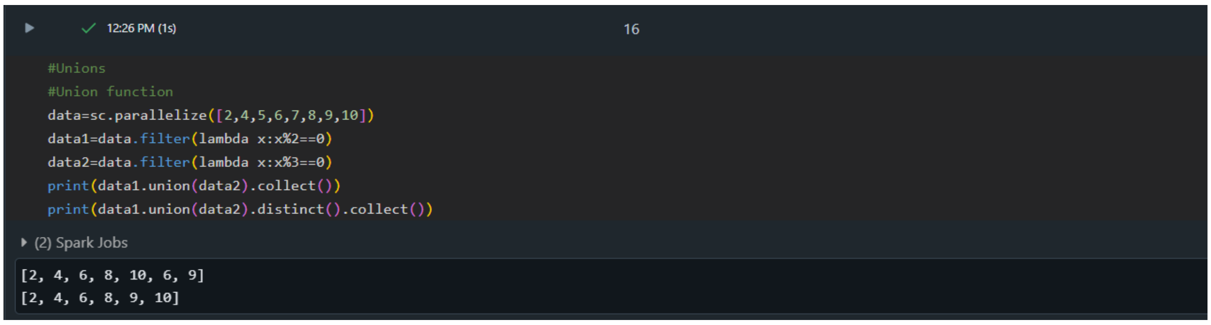
```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id| name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+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

1.8.Cross join



```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

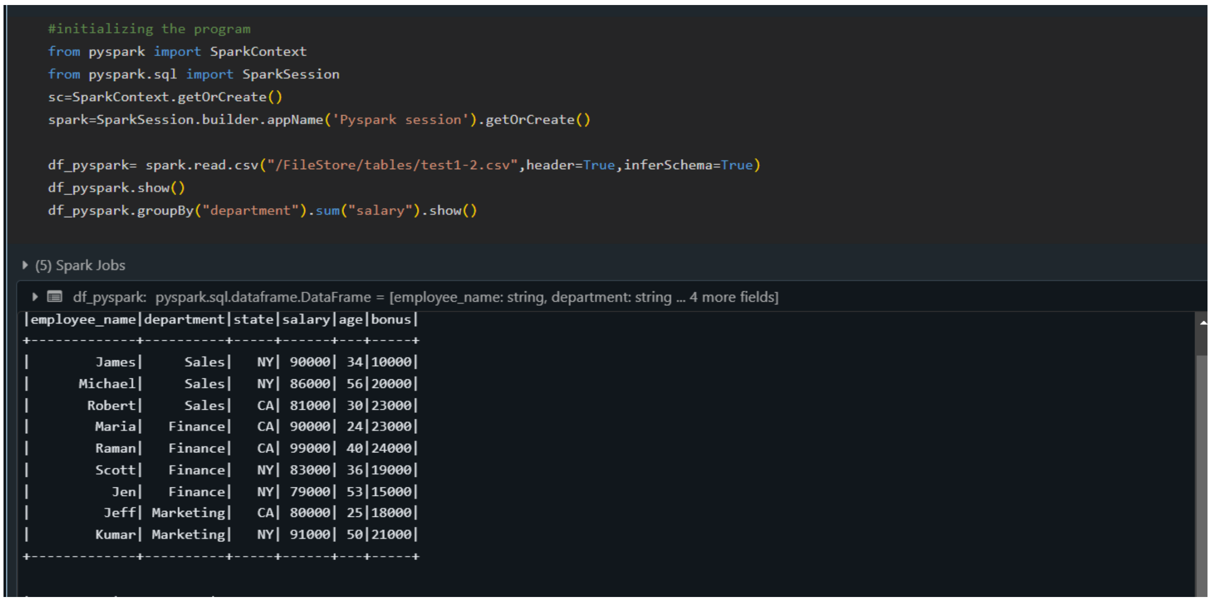
9.Union between two RDDS



```
#Unions
#Union function
data=sc.parallelize([2,4,5,6,7,8,9,10])
data1=data.filter(lambda x:x%2==0)
data2=data.filter(lambda x:x%3==0)
print(data1.union(data2).collect())
print(data1.union(data2).distinct().collect())
```

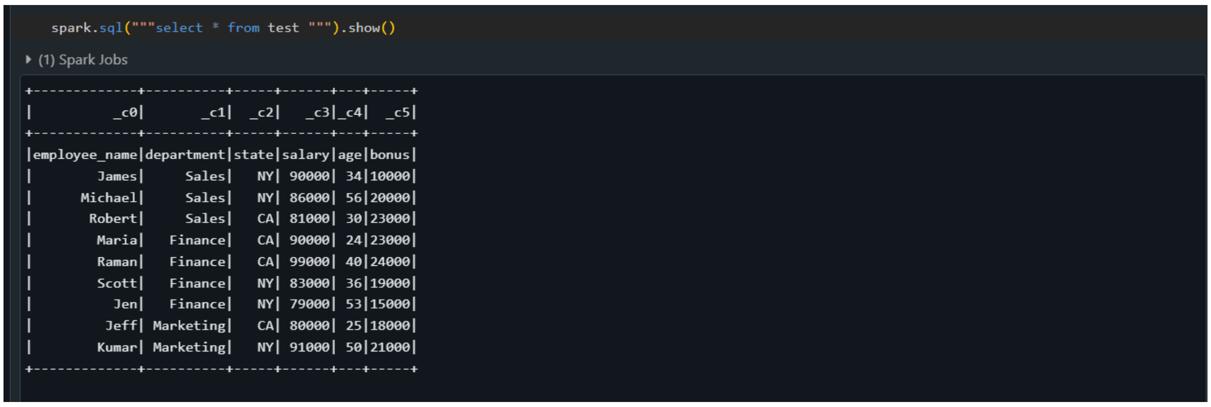
```
[2, 4, 6, 8, 10, 6, 9]
[2, 4, 6, 8, 9, 10]
```

Handson on spark sql


```
#initializing the program
from pyspark import SparkContext
from pyspark.sql import SparkSession
sc=SparkContext.getOrCreate()
spark=SparkSession.builder.appName('Pyspark session').getOrCreate()

df_pyspark= spark.read.csv("/FileStore/tables/test1-2.csv",header=True,inferSchema=True)
df_pyspark.show()
df_pyspark.groupBy("department").sum("salary").show()
```

(5) Spark Jobs
df_pyspark: pyspark.sql.dataframe.DataFrame = [employee_name: string, department: string ... 4 more fields]

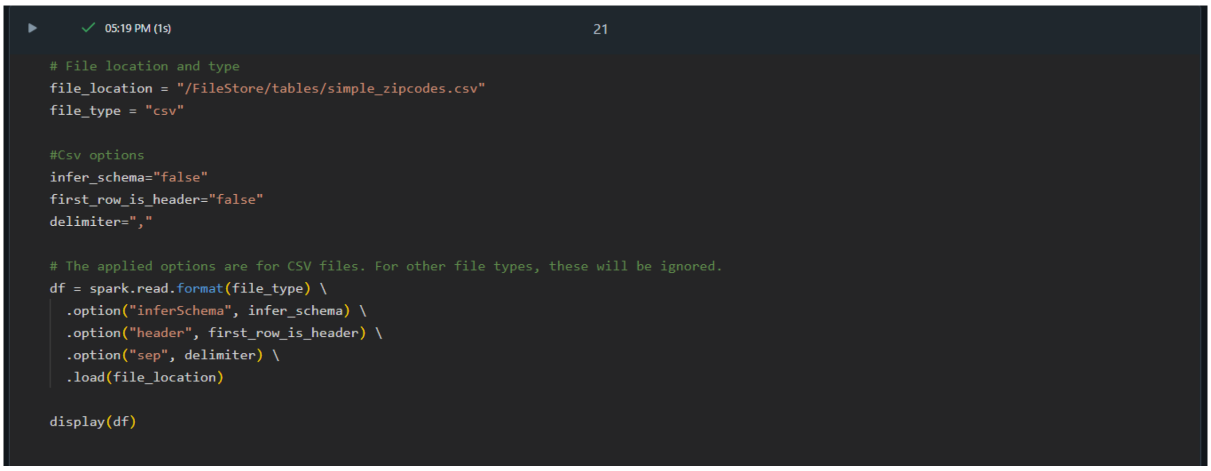
```
|employee_name|department|state|salary|age|bonus|
+-------------+----------+-----+------+---+-----+
|        James|     Sales|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
+-------------+----------+-----+------+---+-----+
```


```
spark.sql("""select * from test """).show()
```

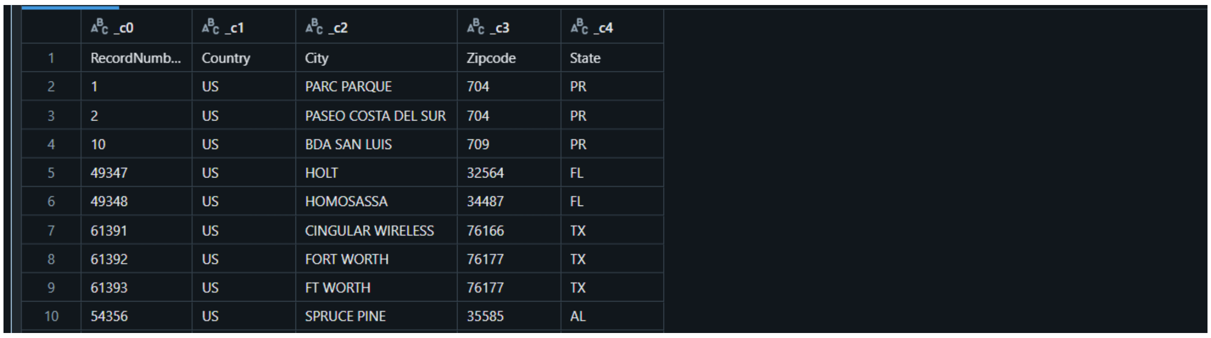
```
+-------------+----------+-----+------+---+-----+
|          _c0|       _c1|  _c2|   _c3|_c4|  _c5|
+-------------+----------+-----+------+---+-----+
|employee_name|department|state|salary|age|bonus|
|        James|     Sales|   NY| 90000| 34|10000|
|      Michael|     Sales|   NY| 86000| 56|20000|
|       Robert|     Sales|   CA| 81000| 30|23000|
|        Maria|   Finance|   CA| 90000| 24|23000|
|        Raman|   Finance|   CA| 99000| 40|24000|
|        Scott|   Finance|   NY| 83000| 36|19000|
|          Jen|   Finance|   NY| 79000| 53|15000|
|         Jeff| Marketing|   CA| 80000| 25|18000|
|        Kumar| Marketing|   NY| 91000| 50|21000|
+-------------+----------+-----+------+---+-----+
```

Reading csv files through spark.

Method 1 : first row is not taken as header

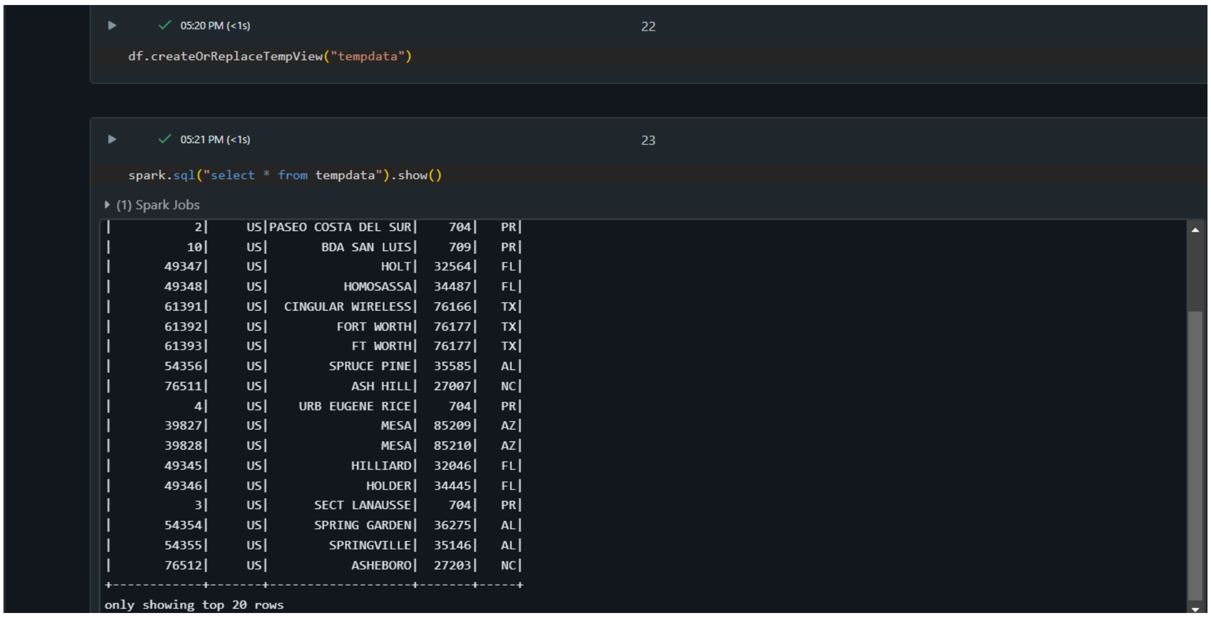


```
# File location and type
file_location = "/FileStore/tables/simple_zipcodes.csv"
file_type = "csv"

#Csv options
infer_schema="false"
first_row_is_header="false"
delimiter=","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```



	_c0	_c1	_c2	_c3	_c4
1	RecordNumb...	Country	City	Zipcode	State
2	1	US	PARC PARQUE	704	PR
3	2	US	PASEO COSTA DEL SUR	704	PR
4	10	US	BDA SAN LUIS	709	PR
5	49347	US	HOLT	32564	FL
6	49348	US	HOMOSASSA	34487	FL
7	61391	US	CINGULAR WIRELESS	76166	TX
8	61392	US	FORT WORTH	76177	TX
9	61393	US	FT WORTH	76177	TX
10	54356	US	SPRUCE PINE	35585	AL

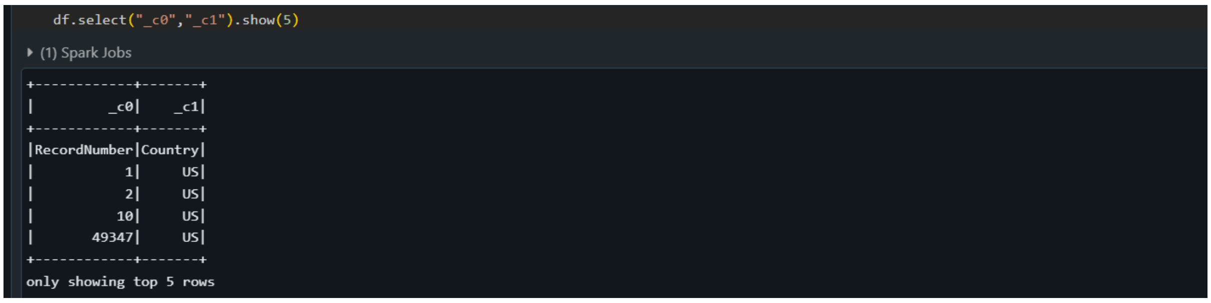
Creating a temp view and obtaining results from the view



```
05:20 PM (<1s)                                    22
df.createOrReplaceTempView("tempdata")
```

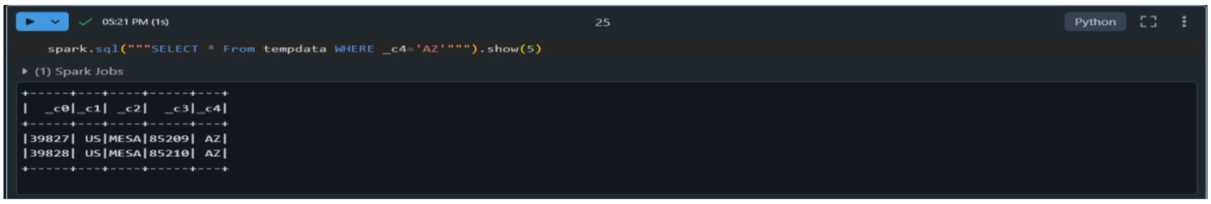
```
05:21 PM (<1s)                                    23
spark.sql("select * from tempdata").show()
(1) Spark Jobs
|           2|     US|PASEO COSTA DEL SUR|    704|   PR|
|          10|     US|       BDA SAN LUIS|    709|   PR|
|       49347|     US|               HOLT|  32564|   FL|
|       49348|     US|          HOMOSASSA|  34487|   FL|
|       61391|     US|  CINGULAR WIRELESS|  76166|   TX|
|       61392|     US|         FORT WORTH|  76177|   TX|
|       61393|     US|           FT WORTH|  76177|   TX|
|       54356|     US|        SPRUCE PINE|  35585|   AL|
|       76511|     US|           ASH HILL|  27007|   NC|
|           4|     US|    URB EUGENE RICE|    704|   PR|
|       39827|     US|               MESA|  85209|   AZ|
|       39828|     US|               MESA|  85210|   AZ|
|       49345|     US|           HILLIARD|  32046|   FL|
|       49346|     US|             HOLDER|  34445|   FL|
|           3|     US|      SECT LANAUSSE|    704|   PR|
|       54354|     US|      SPRING GARDEN|  36275|   AL|
|       54355|     US|        SPRINGVILLE|  35146|   AL|
|       76512|     US|           ASHEBORO|  27203|   NC|
+------------+-------+-------------------+-------+-----+
only showing top 20 rows
```

Show(n) -→n indicates no. of rows



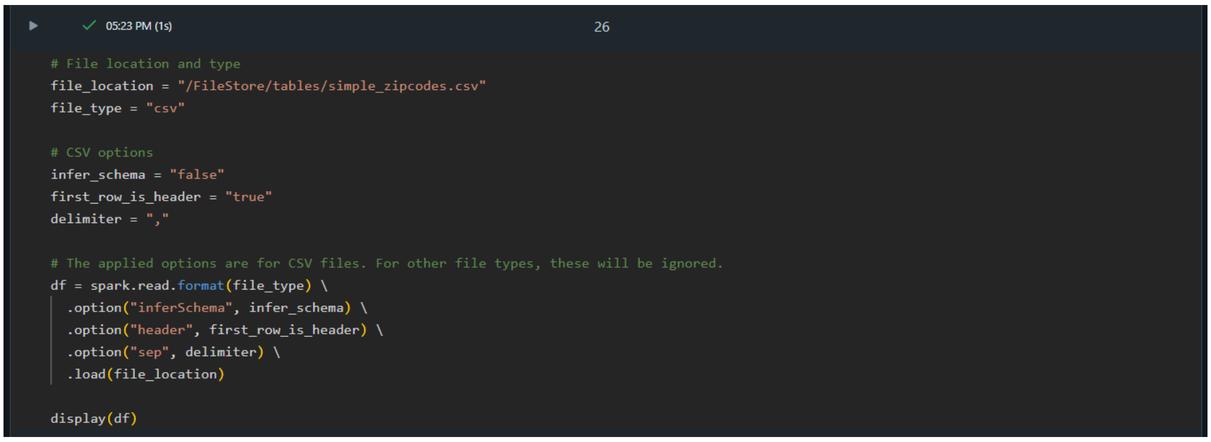
```
df.select("_c0","_c1").show(5)
(1) Spark Jobs
+------------+-------+
|         _c0|    _c1|
+------------+-------+
|RecordNumber|Country|
|           1|     US|
|           2|     US|
|          10|     US|
|       49347|     US|
+------------+-------+
only showing top 5 rows
```

Data manipulation-Using Select and where



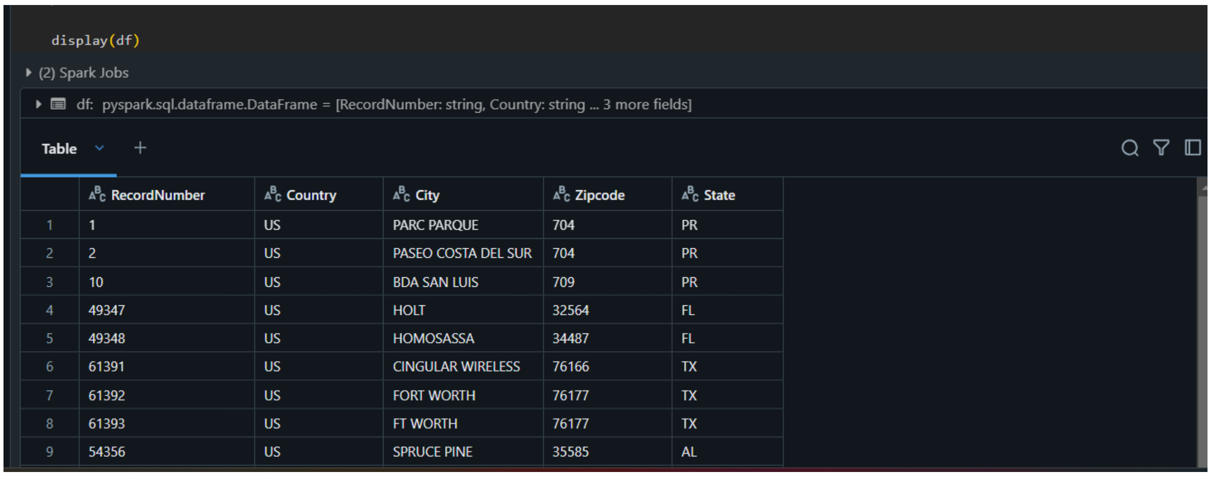
```
05:21 PM (1s)                                    25                                    Python
spark.sql("""SELECT * From tempdata WHERE _c4='AZ'""").show(5)
(1) Spark Jobs
+-----+---+----+-----+---+
|  _c0|_c1| _c2|  _c3|_c4|
+-----+---+----+-----+---+
|39827| US|MESA|85209| AZ|
|39828| US|MESA|85210| AZ|
+-----+---+----+-----+---+
```

Method 2 ---→keeping first row as header



05:23 PM (1s) 26

```
# File location and type
file_location = "/FileStore/tables/simple_zipcodes.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```



```
display(df)
```

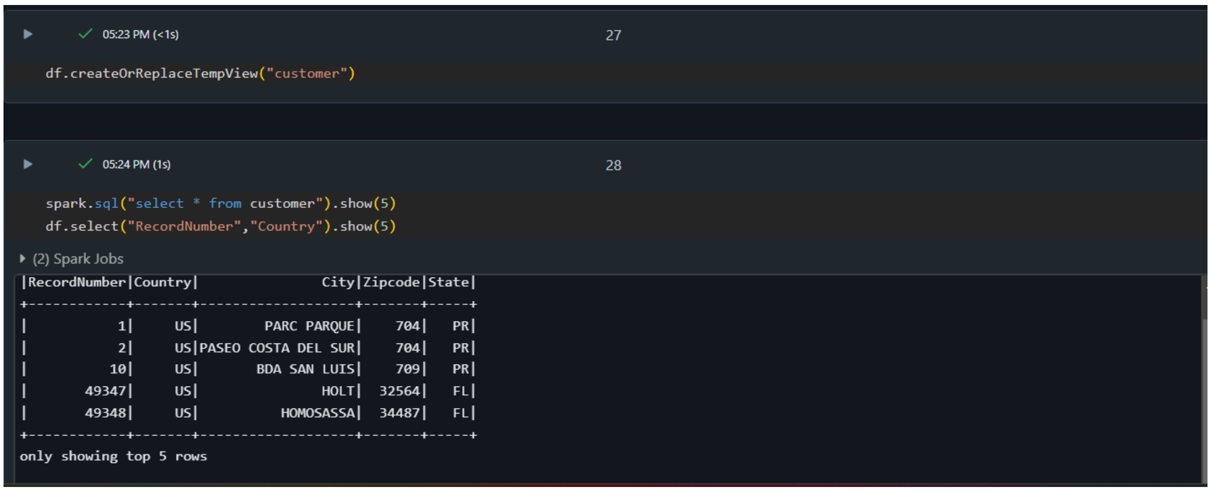
(2) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [RecordNumber: string, Country: string ... 3 more fields]

Table

	RecordNumber	Country	City	Zipcode	State
1	1	US	PARC PARQUE	704	PR
2	2	US	PASEO COSTA DEL SUR	704	PR
3	10	US	BDA SAN LUIS	709	PR
4	49347	US	HOLT	32564	FL
5	49348	US	HOMOSASSA	34487	FL
6	61391	US	CINGULAR WIRELESS	76166	TX
7	61392	US	FORT WORTH	76177	TX
8	61393	US	FT WORTH	76177	TX
9	54356	US	SPRUCE PINE	35585	AL

Creating temp view and accessing data from view



05:23 PM (<1s) 27

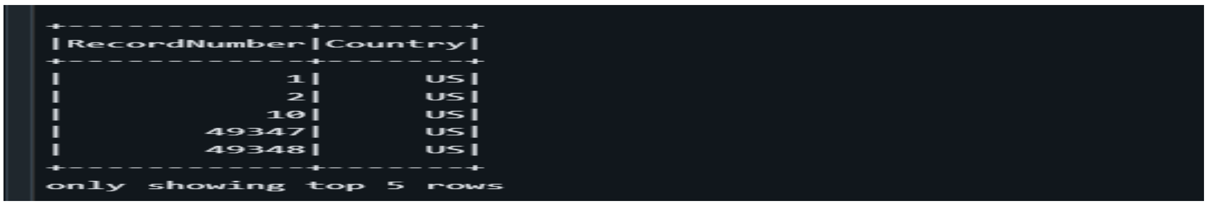
```
df.createOrReplaceTempView("customer")
```

05:24 PM (1s) 28

```
spark.sql("select * from customer").show(5)
df.select("RecordNumber","Country").show(5)
```

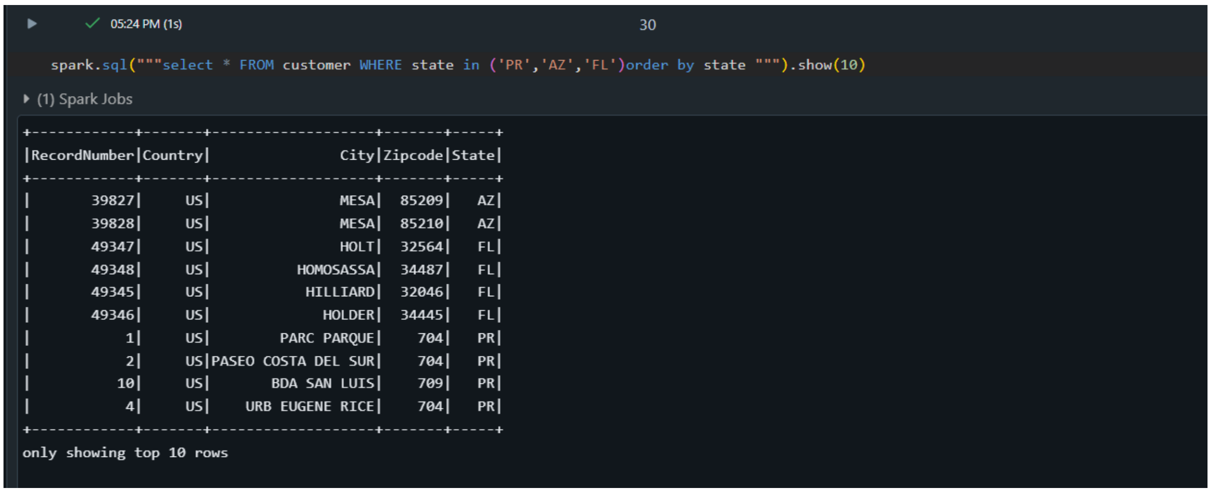
(2) Spark Jobs

```
|RecordNumber|Country|               City|Zipcode|State|
+------------+-------+-------------------+-------+-----+
|           1|     US|        PARC PARQUE|    704|   PR|
|           2|     US|PASEO COSTA DEL SUR|    704|   PR|
|          10|     US|       BDA SAN LUIS|    709|   PR|
|       49347|     US|               HOLT|  32564|   FL|
|       49348|     US|          HOMOSASSA|  34487|   FL|
+------------+-------+-------------------+-------+-----+
only showing top 5 rows
```



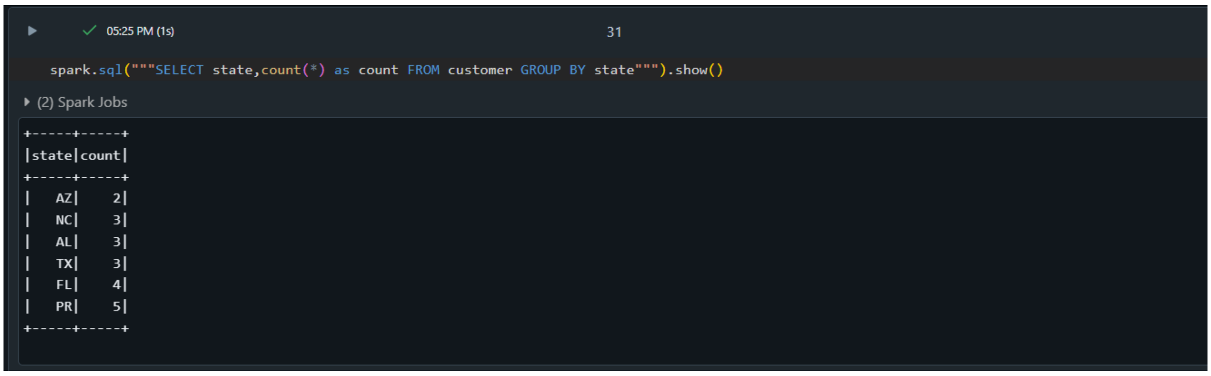
```
+------------+-------+
|RecordNumber|Country|
+------------+-------+
|           1|     US|
|           2|     US|
|          10|     US|
|       49347|     US|
|       49348|     US|
+------------+-------+
only showing top 5 rows
```

Using orderby



```
05:24 PM (1s)                                   30

spark.sql("""select * FROM customer WHERE state in ('PR','AZ','FL')order by state """).show(10)

(1) Spark Jobs

+------------+-------+------------------+-------+-----+
|RecordNumber|Country|              City|Zipcode|State|
+------------+-------+------------------+-------+-----+
|       39827|     US|              MESA|  85209|   AZ|
|       39828|     US|              MESA|  85210|   AZ|
|       49347|     US|              HOLT|  32564|   FL|
|       49348|     US|         HOMOSASSA|  34487|   FL|
|       49345|     US|          HILLIARD|  32046|   FL|
|       49346|     US|            HOLDER|  34445|   FL|
|           1|     US|       PARC PARQUE|    704|   PR|
|           2|     US|PASEO COSTA DEL SUR|    704|   PR|
|          10|     US|      BDA SAN LUIS|    709|   PR|
|           4|     US|   URB EUGENE RICE|    704|   PR|
+------------+-------+------------------+-------+-----+
only showing top 10 rows
```

Using aggregate function like count and group by



```
05:25 PM (1s)                                   31

spark.sql("""SELECT state,count(*) as count FROM customer GROUP BY state""").show()

(2) Spark Jobs

+-----+-----+
|state|count|
+-----+-----+
|   AZ|    2|
|   NC|    3|
|   AL|    3|
|   TX|    3|
|   FL|    4|
|   PR|    5|
+-----+-----+
```