JOINS AND SPARK SQL ASSIGNMENT

- HARSHINI V

MANIPULATING, DROPING, SORTING, AGGREGATIONS, JOINING, GROUPEBY DATAFRAMES:

Manipulating, dropping, sorting, aggregations, joining, and grouping data are essential operations in data analysis and transformation, especially when working with pandas DataFrames in Python.

Manipulating refers to altering or transforming the structure and content of a DataFrame, such as adding or modifying columns and rows.

Dropping involves removing unwanted rows or columns to streamline the dataset. Sorting arranges the data based on specific criteria, such as ascending or descending order, to facilitate analysis.

Aggregation is the process of summarizing data using operations like sum, mean, or count, while joining combines multiple DataFrames to integrate data from different sources.

Lastly, grouping involves organizing data into groups based on column values, followed by applying specific functions to analyze these groups. Together, these operations form the foundation for efficient and insightful data manipulation and analysis.

EXAMPLES:

The data used for the example is as follows.

```
from pyspark.sql import 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)
]
schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
df.printSchema()
df.show()
```

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

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

▸ (2) Spark Jobs

▸ data: pyspark.sql.dataframe.DataFrame = [department: string, sum(salary): long]

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

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

▸ (10) Spark Jobs

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

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

▸ (10) Spark Jobs

```
+-------------+----------+-----------+
|employee_name|department|min(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|max(salary)|
+-------------+----------+-----------+
|        James|     Sales|      90000|
|      Michael|     Sales|      86000|
|       Robert|     Sales|      81000|
|        Maria|   Finance|      90000|
```

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

▸ (9) Spark Jobs

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

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
.appName("example") \
.getOrCreate()
simpleData = [("James","Sales","NY",90000,34,10000),
("Michael","Sales","NY",86000,56,20000),
("Robert","Sales","CA",81000,None,23000),
("Maria","Finance","CA",90000,24,23000),
("Raman","Finance","CA",99000,40,None),
("Scott","Finance","NY",None,36,44000),
("Jen","Finance","NY",55000,53,15000),
("Jeff",None,"CA",80000,25,18000),
("null","Marketing","NY",91000,50,21000)
]
schema = ["employee_name","department","state","salary","age","bonus"]
dfa = spark.createDataFrame(data=simpleData, schema = schema)
dfa.na.drop().show()
```

▸ (3) Spark Jobs

▸ dfa: 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|
|        Maria|   Finance|   CA| 90000| 24|23000|
|          Jen|   Finance|   NY| 55000| 53|15000|
|         null| Marketing|   NY| 91000| 50|21000|
+-------------+----------+-----+------+---+-----+
```

```
dfa.show()
```

▸ (3) Spark Jobs

```
+-------------+----------+-----+------+----+-----+
|employee_name|department|state|salary| age|bonus|
+-------------+----------+-----+------+----+-----+
|        James|     Sales|   NY| 90000|  34|10000|
|      Michael|     Sales|   NY| 86000|  56|20000|
|       Robert|     Sales|   CA| 81000|null|23000|
|        Maria|   Finance|   CA| 90000|  24|23000|
|        Raman|   Finance|   CA| 99000|  40| null|
|        Scott|   Finance|   NY|  null|  36|44000|
|          Jen|   Finance|   NY| 55000|  53|15000|
|         Jeff|      null|   CA| 80000|  25|18000|
|         null| Marketing|   NY| 91000|  50|21000|
+-------------+----------+-----+------+----+-----+
```

```
df.groupBy("department").agg({"salary":"sum"}).show()
df.agg({"salary":"sum"}).show()
```

▸ (4) Spark Jobs

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

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

JOINS IN PYSPARK:

In PySpark, joins are used to combine two or more DataFrames based on a condition, typically involving matching keys from the datasets. Joins are essential for working with distributed datasets, allowing analysts and developers to merge information from multiple sources effectively. PySpark supports various types of joins, including:

1. **Inner Join**: Returns rows with matching keys in both DataFrames.
2. **Left Outer Join**: Returns all rows from the left DataFrame and matching rows from the right; non-matching rows from the right are filled with nulls.
3. **Right Outer Join**: Returns all rows from the right DataFrame and matching rows from the left; non-matching rows from the left are filled with nulls.
4. **Full Outer Join**: Combines all rows from both DataFrames; unmatched rows are filled with nulls.
5. **Cross Join**: Returns the Cartesian product of two DataFrames.
6. **Semi Join**: Filters rows from the left DataFrame where matching keys exist in the right DataFrame.
7. **Anti Join**: Filters rows from the left DataFrame where no matching keys exist in the right DataFrame.

EXAMPLES:

The data used for the example is as follows.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
.appName("example") \
.getOrCreate()
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()
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

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

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

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

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

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

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

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

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

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

```
# Right join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "right").show()
# Right outer join
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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

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

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

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

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

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

SPARK SQL:

Spark SQL is a module of Apache Spark designed for working with structured and semi-structured data using SQL-like queries. It provides a seamless way to interact with data stored in a variety of formats, such as JSON, Parquet, Hive tables, and more, while leveraging the distributed computing power of Spark. Spark SQL is ideal for users familiar with SQL, making it easier to process large datasets without needing to write complex programming code. It is widely used in big data analytics, ETL pipelines, and real-time data processing.

Key Features of Spark SQL:

1. **SQL Queries**: Enables the execution of SQL queries on Spark DataFrames and datasets.
2. **Integration with Spark**: Combines SQL operations with the rich APIs of Spark for tasks like machine learning and streaming.
3. **DataFrames**: Introduces DataFrames, a distributed collection of data organized into named columns, similar to a table in a relational database.

EXAMPLES:

```
file_location = "/FileStore/tables/simple_zipcodes.csv"
file_type = "csv"
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","
df = spark.read.format(file_type) \
.option("inferSchema", infer_schema) \
.option("header", first_row_is_header) \
.option("sep", delimiter) \
.load(file_location)
display(df)
df.createOrReplaceTempView("tempdata")
```

_c0	_c1	_c2	_c3	_c4
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
61391	US	CINGULAR WIRELESS	76166	TX
61392	US	FORT WORTH	76177	TX
61393	US	FT WORTH	76177	TX
54356	US	SPRUCE PINE	35585	AL

```
spark.sql("select * from tempdata").show()
df.select("_c0","_c1").show(5)
```

```
+------------+-------+------------------+-------+-----+
|         _c0|    _c1|               _c2|    _c3|  _c4|
+------------+-------+------------------+-------+-----+
|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|
|       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

+------------+-------+
|         _c0|    _c1|
+------------+-------+
|RecordNumber|Country|
|           1|     US|
|           2|     US|
|          10|     US|
|       49347|     US|
+------------+-------+
only showing top 5 rows
```

```
spark.sql("""SELECT * From tempdata WHERE _c4='AZ'""").show(5)
```

```
+-----+---+-----+-----+---+
|  _c0|_c1| _c2|  _c3|_c4|
+-----+---+-----+-----+---+
|39827| US|MESA|85209| AZ|
|39828| US|MESA|85210| AZ|
+-----+---+-----+-----+---+
```

```
file_location = "/FileStore/tables/simple_zipcodes.csv"
file_type = "csv"
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
    .option("header", first_row_is_header) \
      .option("sep", delimiter) \
        .load(file_location)
display(df)
df.createOrReplaceTempView("customer")
```

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
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

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

```
+------------+-------+--------------------+-------+-----+
|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|
|       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|
|       76513|     US|            ASHEBORO|  27204|   NC|
+------------+-------+--------------------+-------+-----+

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

```
spark.sql("""SELECT * From customer WHERE state='PR'""").show(5)
```

```
+------------+-------+--------------------+-------+-----+
|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|
|           4|     US|     URB EUGENE RICE|    704|   PR|
|           3|     US|       SECT LANAUSSE|    704|   PR|
+------------+-------+--------------------+-------+-----+
```

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

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

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

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