Name: Krushnakumar Patle

Email: krishnapatle128@gmail.com

Batch: Data Engineering Batch-1

Spark and Pyspark Coding Challenge

Q1. Execute Manipulating, Droping, Sorting, Aggregations, Joining, GroupeBy DataFrames

```
[2]: from pyspark.sql import SparkSession
spark=SparkSession.builder.appName("coding_challenge_pyspark").getOrCreate()
```

Execute Manipulating, Droping, Sorting, Aggregations, Joining, GroupeBy DataFrames

```
[3]: df_pyspark=spark.read.csv("test.csv",header=True,inferSchema=True)
df_pyspark.show()
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Chandu|Data Science| 50000|
|Rashmi |         IOT| 75000|
|  Rohit|    Big Data| 55000|
| Rohit |    Big Data| 80000|
|  Ronit|         IOT| 60000|
|  Ronit|Data Science| 70000|
| Chandu|Data Science| 45000|
|Krishna|    Big Data| 65000|
| Rashmi|    Big Data| 85000|
|  Rohit|         IOT| 60000|
+-------+------------+------+
```

Manipulating DataFrame

```
[4]: df_pyspark.select("name","salary").show()
```

```
+-------+------+
|   name|salary|
+-------+------+
| Chandu| 50000|
|Rashmi | 75000|
|  Rohit| 55000|
| Rohit | 80000|
|  Ronit| 60000|
|  Ronit| 70000|
```

[5]:
```
df_pyspark.filter(df_pyspark["salary"]>60000).show()
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
|Rashmi |         IOT| 75000|
| Rohit |    Big Data| 80000|
|  Ronit|Data Science| 70000|
|Krishna|    Big Data| 65000|
| Rashmi|    Big Data| 85000|
+-------+------------+------+
```

Droping

Dropping DataFrames refers to removing columns or rows from the DataFrame. This operation is useful when certain columns or rows are not needed for analysis.

[6]:
```
df_pyspark1=spark.read.csv("test2.csv",header=True,inferSchema=True)
df_pyspark1.show()
```

```
+----------+----+----------+------+
|      Name| age|Experience|Salary|
+----------+----+----------+------+
|     Krish|  31|        10| 30000|
|Shudhanshu|  30|         8| 25000|
|     Sunny|  29|         4| 20000|
|      Paul|  24|         3| 20000|
|    Harsha|  21|         1| 15000|
|   Shubham|  23|         2| 18000|
|    Mahesh|NULL|      NULL|  NULL|
|      NULL|NULL|        10| 40000|
|      NULL|  34|        10| 38000|
|      NULL|NULL|      NULL|  NULL|
|      NULL|  36|      NULL|  NULL|
+----------+----+----------+------+
```

[7]:
```
df_pyspark1.na.drop().show()
```

```
+----------+---+----------+------+
|      Name|age|Experience|Salary|
+----------+---+----------+------+
|     Krish| 31|        10| 30000|
|Shudhanshu| 30|         8| 25000|
|     Sunny| 29|         4| 20000|
|      Paul| 24|         3| 20000|
|    Harsha| 21|         1| 15000|
|   Shubham| 23|         2| 18000|
+----------+---+----------+------+
```

[8]:
```
df_pyspark1.na.drop(how="all").show()
 # if all values in rows are null then drop # default any
```

```
+----------+----+----------+------+
|      Name| age|Experience|Salary|
+----------+----+----------+------+
|     Krish|  31|        10| 30000|
|Shudhanshu|  30|         8| 25000|
|     Sunny|  29|         4| 20000|
|      Paul|  24|         3| 20000|
|    Harsha|  21|         1| 15000|
|   Shubham|  23|         2| 18000|
|    Mahesh|NULL|      NULL|  NULL|
|      NULL|NULL|        10| 40000|
|      NULL|  34|        10| 38000|
|      NULL|  36|      NULL|  NULL|
+----------+----+----------+------+
```

[9]:
```
df_pyspark1.na.drop(how="any",thresh=2).show()
#atleast 2 non null values should be present.
```

```
+----------+----+----------+------+
|      Name| age|Experience|Salary|
+----------+----+----------+------+
|     Krish|  31|        10| 30000|
|Shudhanshu|  30|         8| 25000|
|     Sunny|  29|         4| 20000|
|      Paul|  24|         3| 20000|
|    Harsha|  21|         1| 15000|
|   Shubham|  23|         2| 18000|
|      NULL|NULL|        10| 40000|
|      NULL|  34|        10| 38000|
+----------+----+----------+------+
```

[10]:
```
df_pyspark1.na.drop(how="any",subset=["salary"]).show()
 # only in that column rows get deleted
```

```
+----------+----+----------+------+
|      Name| age|Experience|Salary|
+----------+----+----------+------+
|     Krish|  31|        10| 30000|
|Shudhanshu|  30|         8| 25000|
|     Sunny|  29|         4| 20000|
|      Paul|  24|         3| 20000|
|    Harsha|  21|         1| 15000|
|   Shubham|  23|         2| 18000|
|      NULL|NULL|        10| 40000|
|      NULL|  34|        10| 38000|
+----------+----+----------+------+
```

Sorting DataFrames

Sorting DataFrames involves arranging the rows of the DataFrame in a particular order based on the values of one or more columns. This operation helps in organizing the data for better analysis or visualization.

[11]:
```
df_pyspark.sort("salary").show() # Sort based on single column
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Chandu|Data Science| 45000|
| Chandu|Data Science| 50000|
|  Rohit|    Big Data| 55000|
|  Ronit|         IOT| 60000|
|  Rohit|         IOT| 60000|
|Krishna|    Big Data| 65000|
|  Ronit|Data Science| 70000|
|Rashmi |         IOT| 75000|
| Rohit |    Big Data| 80000|
| Rashmi|    Big Data| 85000|
+-------+------------+------+
```

[12]:
```
df_pyspark.sort(df_pyspark["salary"].desc()).show() # sort based on descending order
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Rashmi|    Big Data| 85000|
| Rohit |    Big Data| 80000|
|Rashmi |         IOT| 75000|
|  Ronit|Data Science| 70000|
|Krishna|    Big Data| 65000|
|  Ronit|         IOT| 60000|
|  Rohit|         IOT| 60000|
|  Rohit|    Big Data| 55000|
| Chandu|Data Science| 50000|
| Chandu|Data Science| 45000|
+-------+------------+------+
```

[13]:
```
df_pyspark.sort("salary","name").show() # Sort based on first column then second column
```

```
+-------+------------+------+
|   name|  department|salary|
+-------+------------+------+
| Chandu|Data Science| 45000|
| Chandu|Data Science| 50000|
|  Rohit|    Big Data| 55000|
|  Rohit|         IOT| 60000|
|  Ronit|         IOT| 60000|
|Krishna|    Big Data| 65000|
|  Ronit|Data Science| 70000|
|Rashmi |         IOT| 75000|
| Rohit |    Big Data| 80000|
| Rashmi|    Big Data| 85000|
+-------+------------+------+
```

Aggregations and GroupeBy

Aggregations involve calculating summary statistics or aggregating data across groups. Common aggregation functions include sum, mean, count, min, max, etc. Aggregations are often performed after grouping the data by one or more column

GroupBy DataFrames

GroupBy DataFrames involves splitting the DataFrame into groups based on one or more columns, applying a function to each group independently, and combining the results. GroupBy is typically followed by aggregation or transformation operations.s.

sum()

Compute the sum for each numeric columns for each group.

[14]:
```
df_pyspark.groupBy("department").sum("salary").show()
```

```
+------------+-----------+
|  department|sum(salary)|
+------------+-----------+
|         IOT|     195000|
|    Big Data|     285000|
|Data Science|     165000|
+------------+-----------+
```

min()

Computes the min value for each numeric column for each group.

[15]:
```
df_pyspark.groupBy("department").min("salary").show()
```

```
+------------+-----------+
|  department|min(salary)|
+------------+-----------+
|         IOT|      60000|
|    Big Data|      55000|
|Data Science|      45000|
+------------+-----------+
```

max()

Computes the max value for each numeric columns for each group.

[16]:
```
df_pyspark.groupBy("department").max("salary").show()
```

```
+------------+-----------+
|  department|max(salary)|
+------------+-----------+
|         IOT|      75000|
|    Big Data|      85000|
|Data Science|      70000|
+------------+-----------+
```

avg()

Computes average values for each numeric columns for each group.

[17]:
```
df_pyspark.groupBy("department").avg("salary").show()
```

```
+------------+-----------+
|  department|avg(salary)|
+------------+-----------+
|         IOT|    65000.0|
|    Big Data|    71250.0|
|Data Science|    55000.0|
+------------+-----------+
```

mean()

Computes average values for each numeric columns for each group

mean() is an alias for avg()..

[18]:
```
df_pyspark.groupBy("department").mean("salary").show()
```

```
+------------+-----------+
|  department|avg(salary)|
+------------+-----------+
|         IOT|    65000.0|
|    Big Data|    71250.0|
|Data Science|    55000.0|
+------------+-----------+
```

count()

Counts the number of records for each group.

Counts the number of records for each group.

[19]:
```
df_pyspark.groupBy("department").count().show()
```

```
+------------+-----+
|  department|count|
+------------+-----+
|         IOT|    3|
|    Big Data|    4|
|Data Science|    3|
+------------+-----+
```