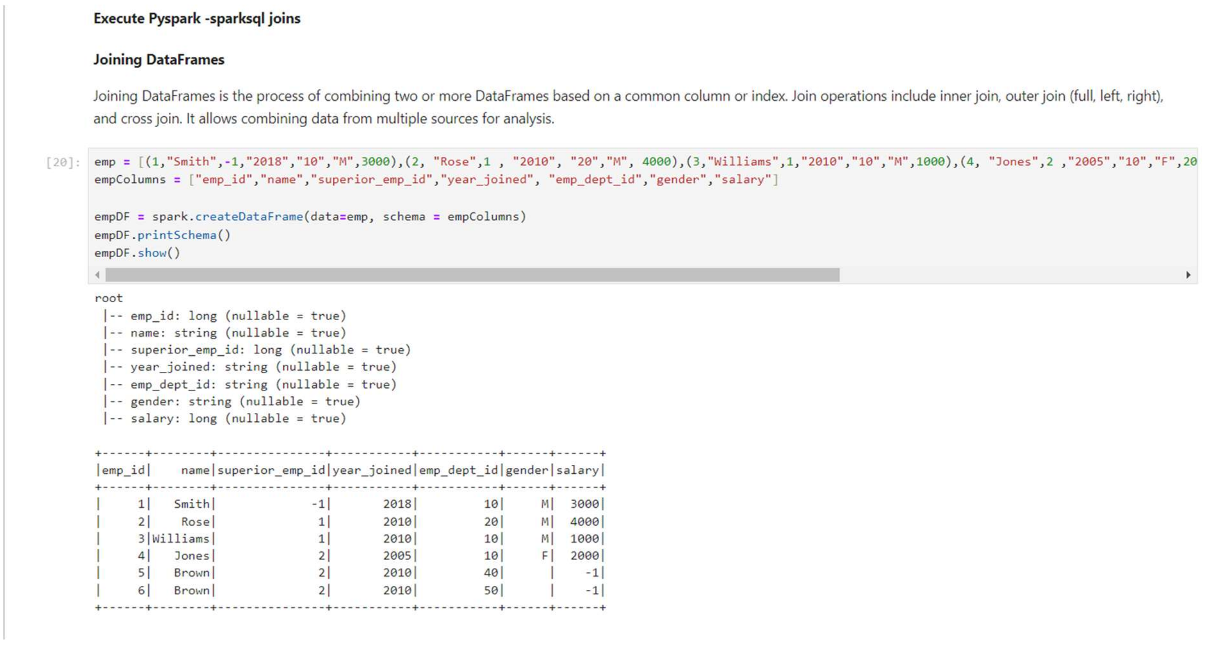
Name: Krushnakumar Patle

Email: krishnapatle128@gmail.com

Batch: Data Engineering Batch-1

Spark and Pyspark Coding Challenge

Q2. Execute Pyspark -sparksql joins & Applying Functions in a Pandas DataFrame



Execute Pyspark -sparksql joins

Joining DataFrames

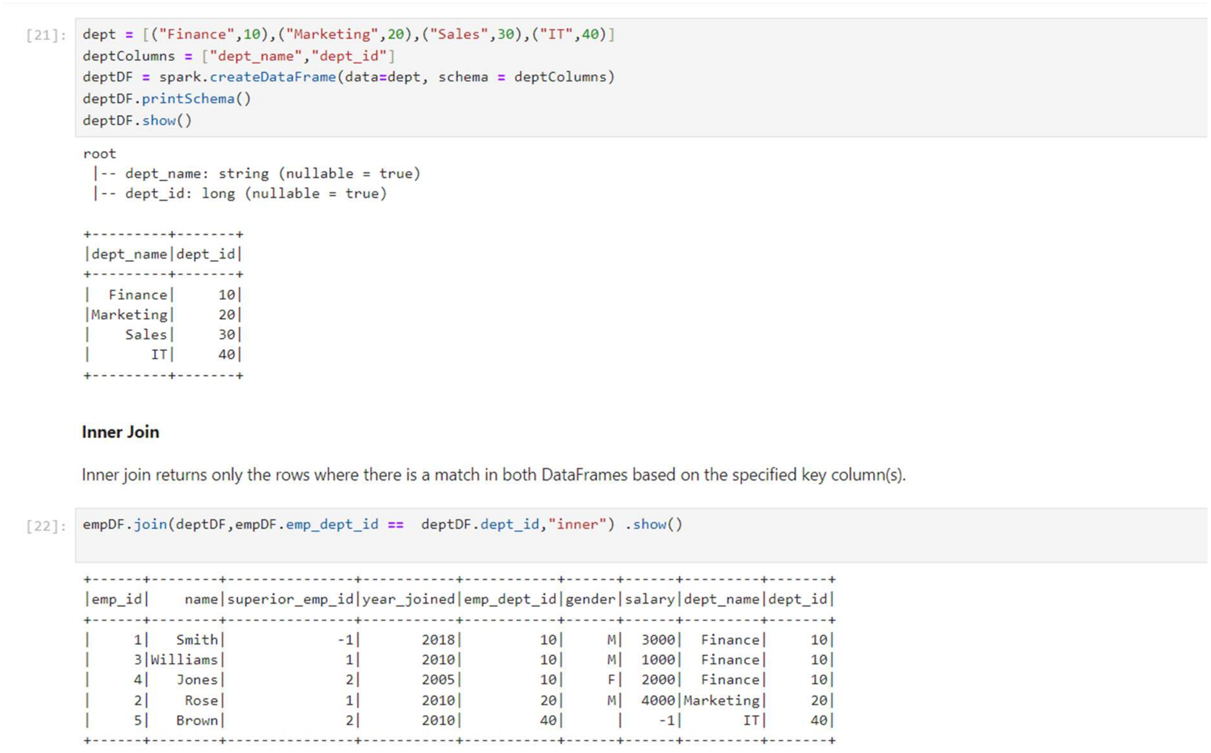
Joining DataFrames is the process of combining two or more DataFrames based on a common column or index. Join operations include inner join, outer join (full, left, right), and cross join. It allows combining data from multiple sources for analysis.

```
[20]: 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",20
      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()
```

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



```
[21]: 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
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

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

Inner Join

Inner join returns only the rows where there is a match in both DataFrames based on the specified key column(s).

```
[22]: empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") .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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Outer Join

Outer join returns all rows from both DataFrames and fills in missing values with null where there is no match based on the specified ke column).

[23]:
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer").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|
|  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

A left join, also known as a left outer join, is a type of join operation in SQL and PySpark that returns all rows from the left DataFrame and only the matching rows from the right DataFrame. If there is no match in the right DataFrame, it fills in the missing values with null.

[24]:
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left").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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

[25]:
```
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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Right join

A right join, also known as a right outer join, is a type of join operation in SQL and PySpark that returns all rows from the right DataFrame and only the matching rows from the left DataFrame. If there is no match in the left DataFrame, it fills in the missing values with null.

[26]:
```
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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

[27]:
```
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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

Left Semi Join

Left semi join returns all the rows from the left DataFrame where there is a match in the right DataFrame based on the specified key column(s). It does not include any columns from the right DataFrame in the result.

[28]:
```
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi").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|
+------+--------+---------------+-----------+-----------+------+------+
```

Left Anti Join

Left anti join returns all the rows from the left DataFrame where there is no match in the right DataFrame based on the specified key column(s). It does not include any columns from the right DataFrame in the result.

[29]:
```
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|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+
```

Applying Functions in a Pandas DataFrame

[30]:
```
data = [(('Ram'), '1991-04-01', 'M', 3000),
        (('Mike'), '2000-05-19', 'M', 4000),
        (('Rohini'), '1978-09-05', 'M', 4000),
        (('Maria'), '1967-12-01', 'F', 4000),
        (('Jenis'), '1980-02-17', 'F', 1200)]

# Column names in dataframe
columns = ["Name", "DOB", "Gender", "salary"]

# Create the spark dataframe
df = spark.createDataFrame(data=data,
                           schema=columns)

# Print the dataframe
df.show()
```

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

Using withColumnRenamed()

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

[31]:
```
df.withColumnRenamed("DOB","DateOfBirth").show()
```

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

[32]:
```
df.withColumnRenamed("Gender","Sex").withColumnRenamed("salary","Amount").show()
```

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

Using selectExpr()

Renaming the column names using selectExpr() method

[33]:
```
data = df.selectExpr("Name as name","DOB","Gender","salary")

# Print the dataframe
data.show()
```

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

Using select() method

[34]:
```
from pyspark.sql.functions import col
data = df.select(col("Name"),col("DOB"),
                 col("Gender"),
                 col("salary").alias('Amount'))
# Print the dataframe
data.show()
```

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

Using toDF()

This function returns a new DataFrame that with new specified column names.

```
[35]: Data_list = ["Emp Name","Date of Birth",
                   " Gender-m/f","Paid salary"]

      new_df = df.toDF(*Data_list)
      new_df.show()
```

```
+--------+-------------+-----------+-----------+
|Emp Name|Date of Birth| Gender-m/f|Paid salary|
+--------+-------------+-----------+-----------+
|     Ram|   1991-04-01|          M|       3000|
|    Mike|   2000-05-19|          M|       4000|
|  Rohini|   1978-09-05|          M|       4000|
|   Maria|   1967-12-01|          F|       4000|
|   Jenis|   1980-02-17|          F|       1200|
+--------+-------------+-----------+-----------+
```