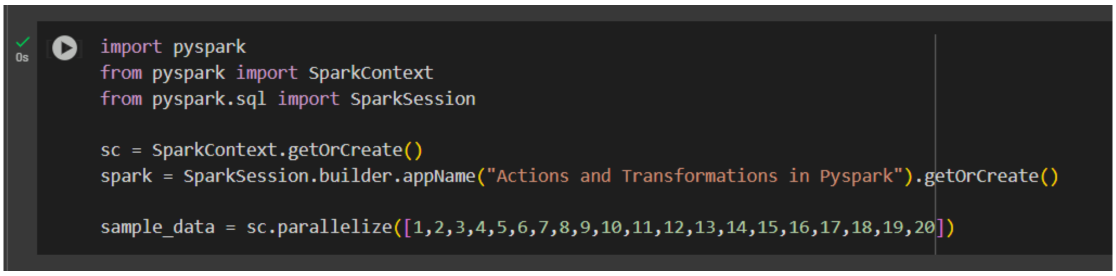
CODING CHALLENGE 4 (PySpark)

TRAINEE NAME: Swathi Baskaran



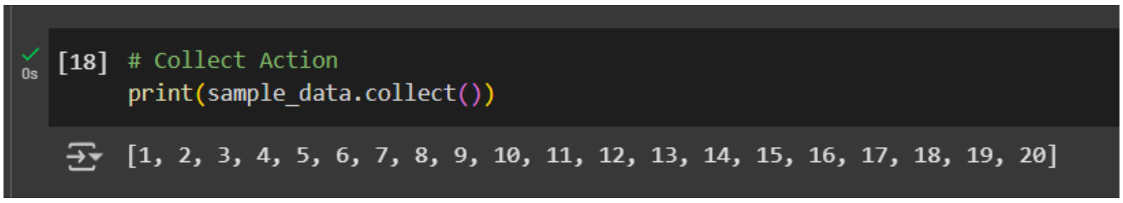
```
import pyspark
from pyspark import SparkContext
from pyspark.sql import SparkSession

sc = SparkContext.getOrCreate()
spark = SparkSession.builder.appName("Actions and Transformations in Pyspark").getOrCreate()

sample_data = sc.parallelize([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
```

Actions:

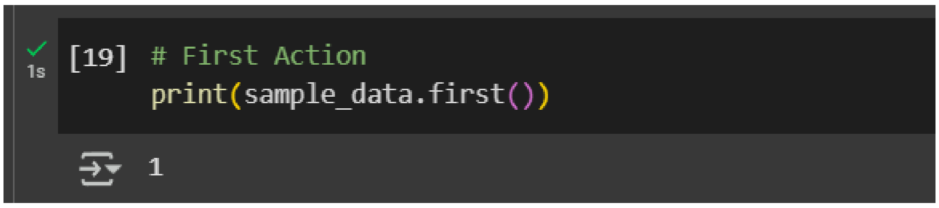
1. Collect



```
[18] # Collect Action
     print(sample_data.collect())

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

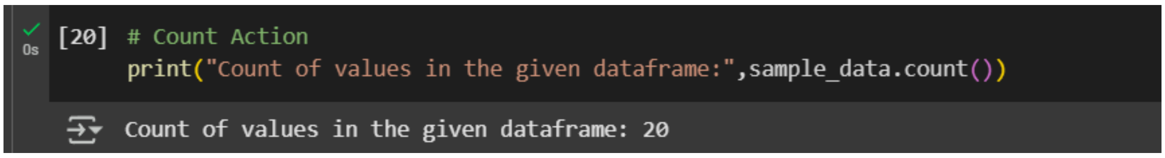
2. First



```
[19] # First Action
     print(sample_data.first())

1
```

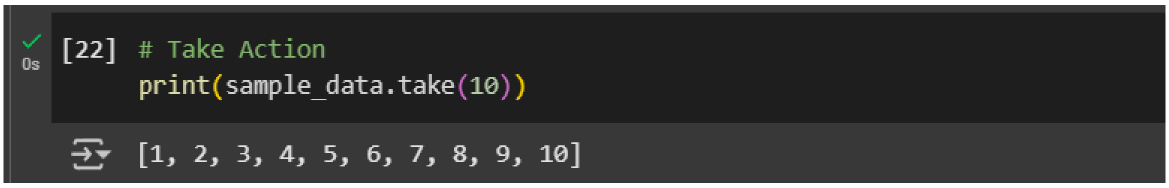
3. Count



```
[20] # Count Action
     print("Count of values in the given dataframe:",sample_data.count())

Count of values in the given dataframe: 20
```

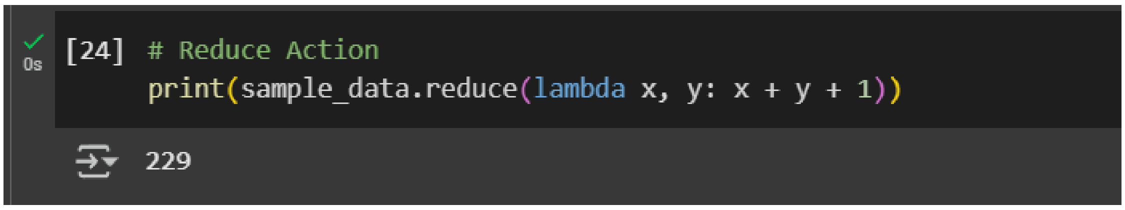
4. Take



```
[22] # Take Action
     print(sample_data.take(10))

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

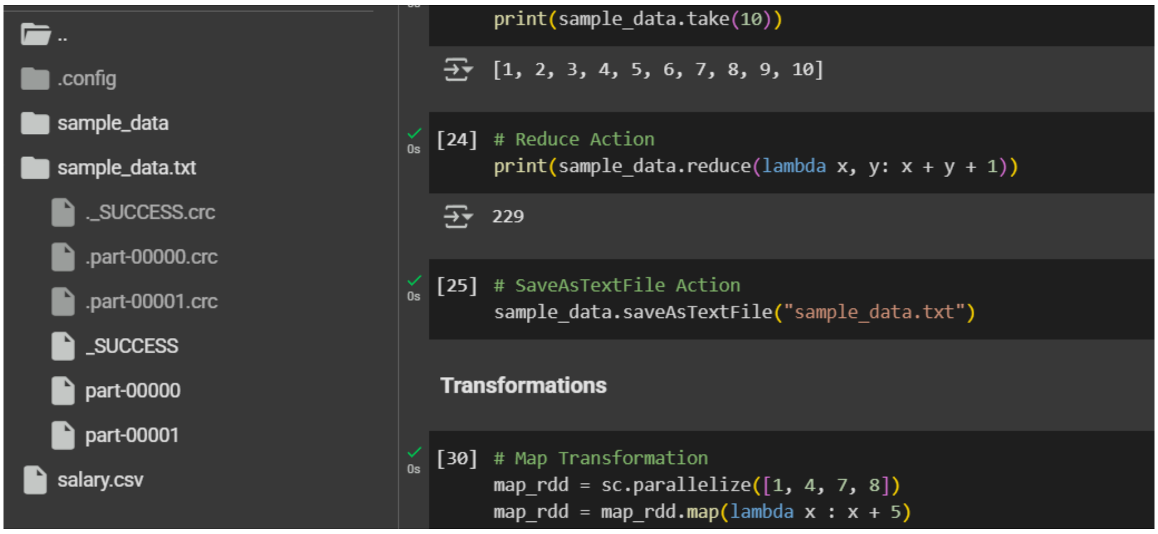
5. Reduce



```
[24] # Reduce Action
     print(sample_data.reduce(lambda x, y: x + y + 1))

229
```

6. Save As Text File



```
..
.config
sample_data
sample_data.txt
    ._SUCCESS.crc
    .part-00000.crc
    .part-00001.crc
    _SUCCESS
    part-00000
    part-00001
salary.csv
```

```
print(sample_data.take(10))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
[24] # Reduce Action
     print(sample_data.reduce(lambda x, y: x + y + 1))
```

```
229
```

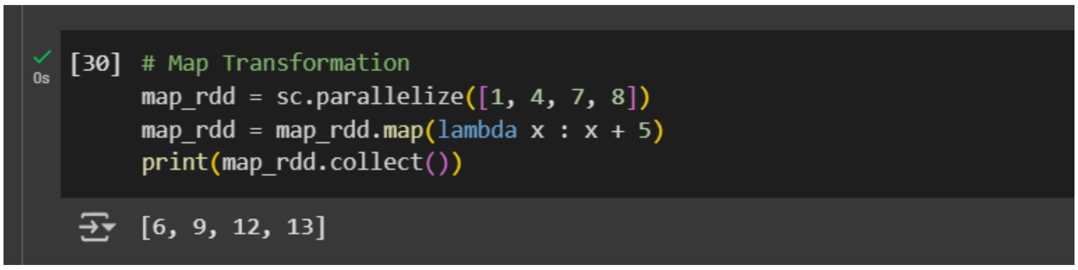
```
[25] # SaveAsTextFile Action
     sample_data.saveAsTextFile("sample_data.txt")
```

Transformations

```
[30] # Map Transformation
     map_rdd = sc.parallelize([1, 4, 7, 8])
     map_rdd = map_rdd.map(lambda x : x + 5)
```

Transformations

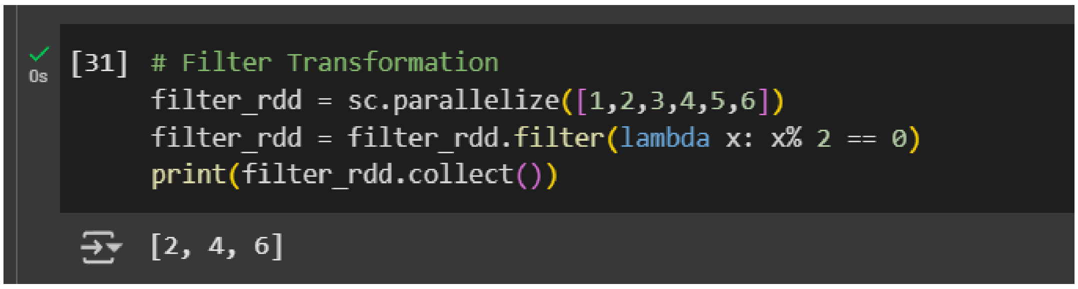
1. Map



```
[30] # Map Transformation
     map_rdd = sc.parallelize([1, 4, 7, 8])
     map_rdd = map_rdd.map(lambda x : x + 5)
     print(map_rdd.collect())
```

```
[6, 9, 12, 13]
```

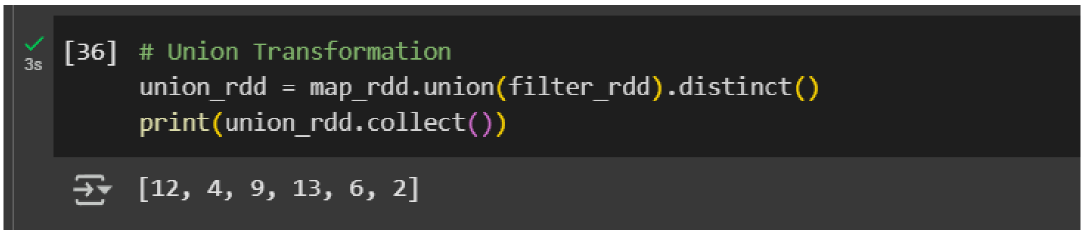
2. Filter



```
[31] # Filter Transformation
     filter_rdd = sc.parallelize([1,2,3,4,5,6])
     filter_rdd = filter_rdd.filter(lambda x: x% 2 == 0)
     print(filter_rdd.collect())
```

```
[2, 4, 6]
```

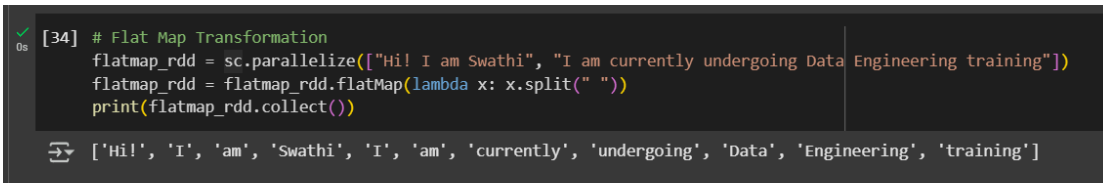
3. Union



```
[36] # Union Transformation
     union_rdd = map_rdd.union(filter_rdd).distinct()
     print(union_rdd.collect())
```

```
[12, 4, 9, 13, 6, 2]
```

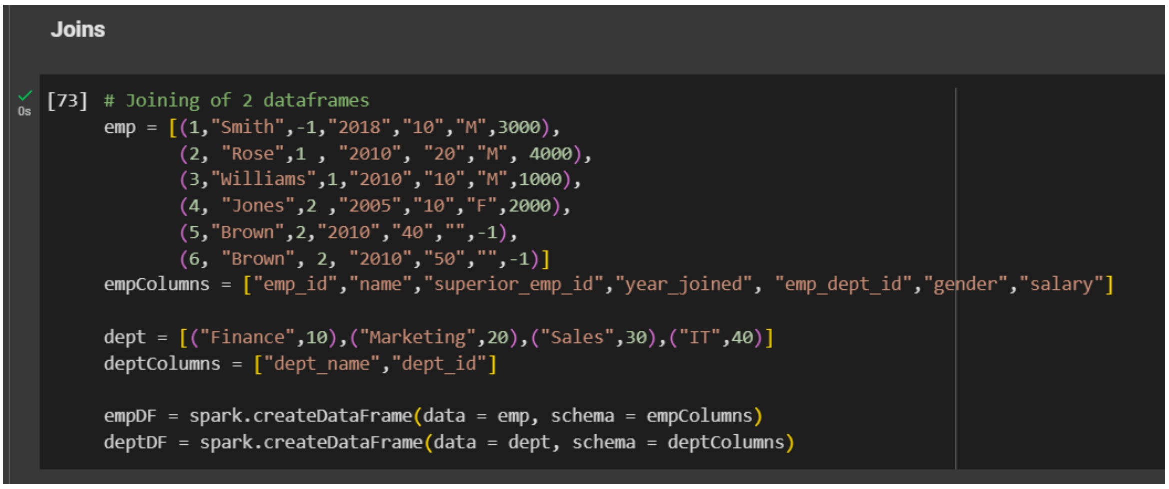
4. FlatMap



```
[34] # Flat Map Transformation
     flatmap_rdd = sc.parallelize(["Hi! I am Swathi", "I am currently undergoing Data Engineering training"])
     flatmap_rdd = flatmap_rdd.flatMap(lambda x: x.split(" "))
     print(flatmap_rdd.collect())
```

```
['Hi!', 'I', 'am', 'Swathi', 'I', 'am', 'currently', 'undergoing', 'Data', 'Engineering', 'training']
```

Joins



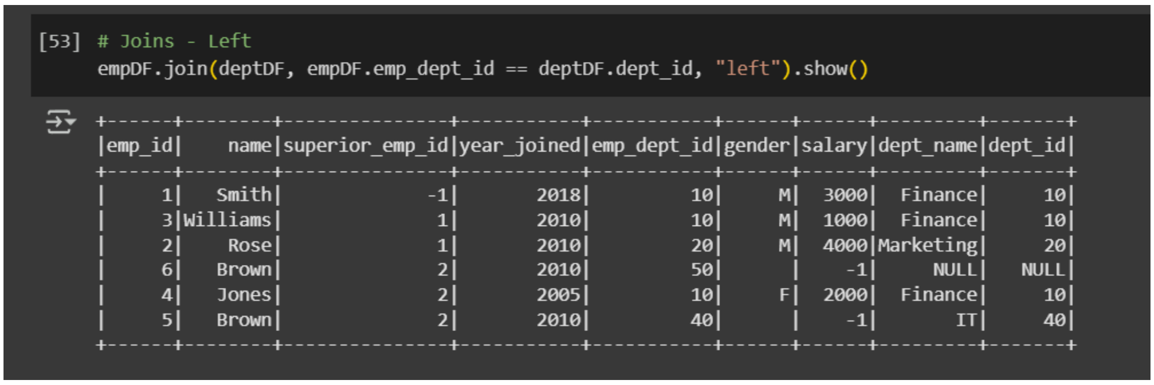
Joins

```
[73] # Joining of 2 dataframes
     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"]

     dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
     deptColumns = ["dept_name","dept_id"]

     empDF = spark.createDataFrame(data = emp, schema = empColumns)
     deptDF = spark.createDataFrame(data = dept, schema = deptColumns)
```

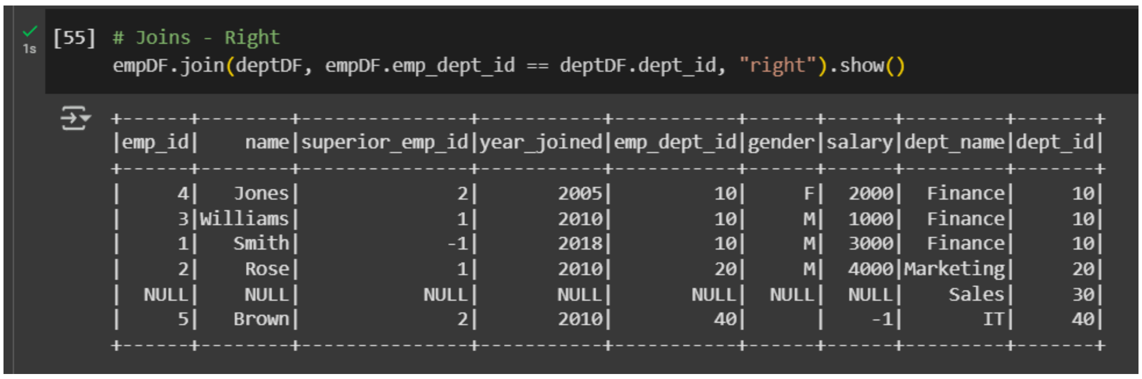
1. Left Join



```
[53] # Joins - Left
     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|
|     3|Williams|              1|       2010|         10|     M|  1000|  Finance|     10|
|     2|    Rose|              1|       2010|         20|     M|  4000|Marketing|     20|
|     6|   Brown|              2|       2010|         50|      |    -1|     NULL|   NULL|
|     4|   Jones|              2|       2005|         10|     F|  2000|  Finance|     10|
|     5|   Brown|              2|       2010|         40|      |    -1|       IT|     40|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

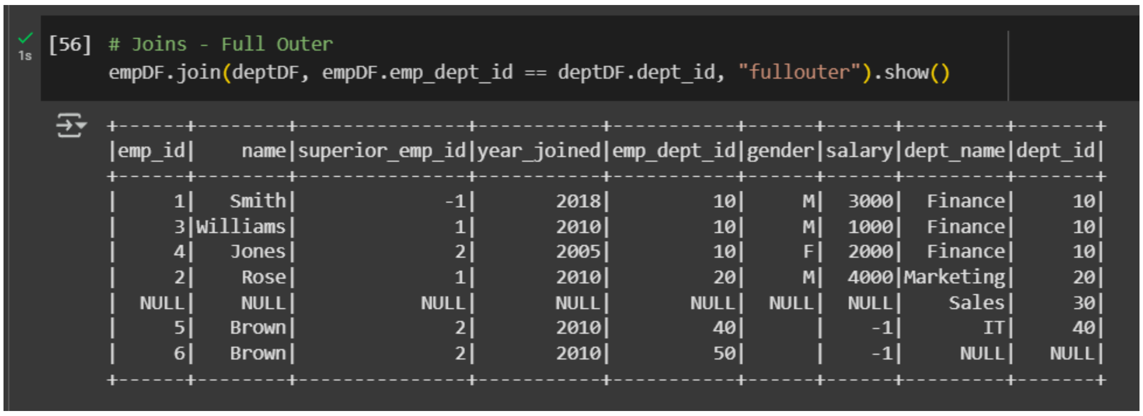
2. Right Join



```
[55] # Joins - Right
     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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

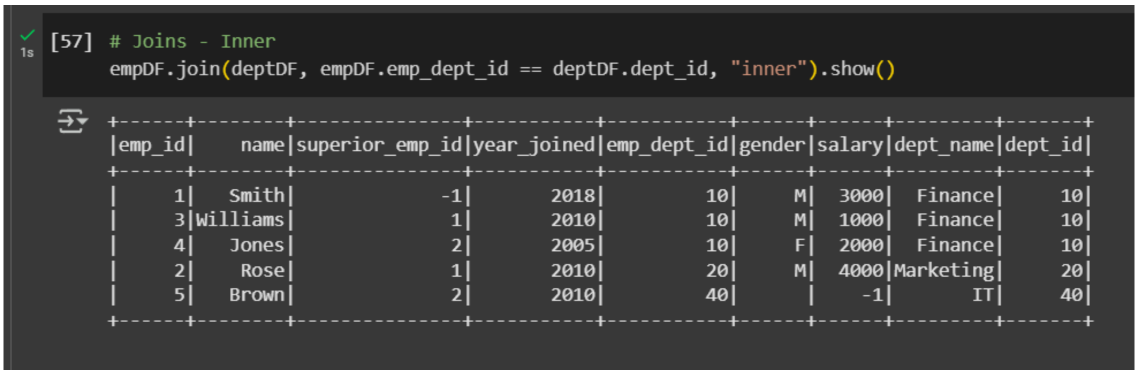
3. Full Outer Join



```
[56] # Joins - Full Outer
     empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "fullouter").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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

4. Inner Join

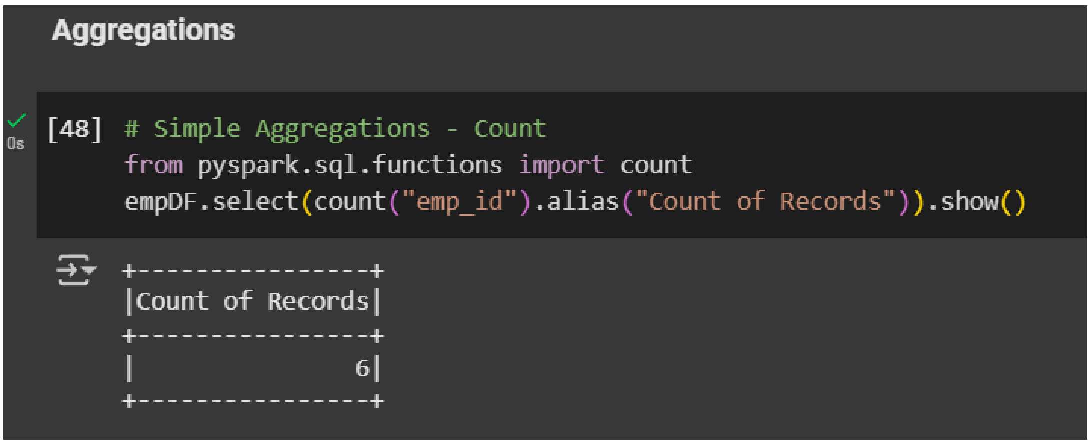


```
[57] # Joins - Inner
     empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "inner").show()
```

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

Aggregations

1. Count

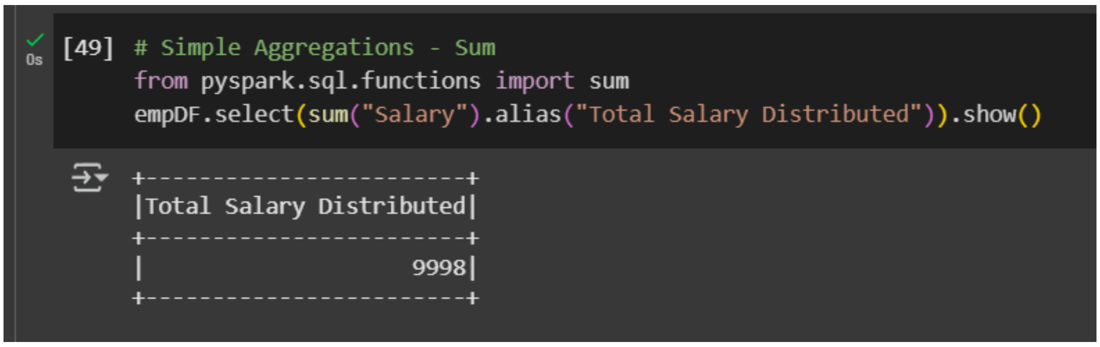


Aggregations

```
[48] # Simple Aggregations - Count
     from pyspark.sql.functions import count
     empDF.select(count("emp_id").alias("Count of Records")).show()
```

```
+----------------+
|Count of Records|
+----------------+
|               6|
+----------------+
```

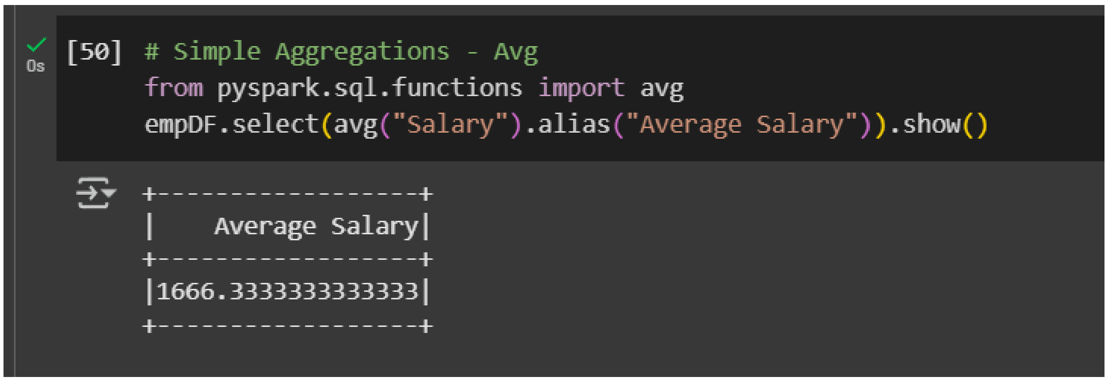
2. Sum



```
[49] # Simple Aggregations - Sum
     from pyspark.sql.functions import sum
     empDF.select(sum("Salary").alias("Total Salary Distributed")).show()
```

```
+------------------------+
|Total Salary Distributed|
+------------------------+
|                    9998|
+------------------------+
```

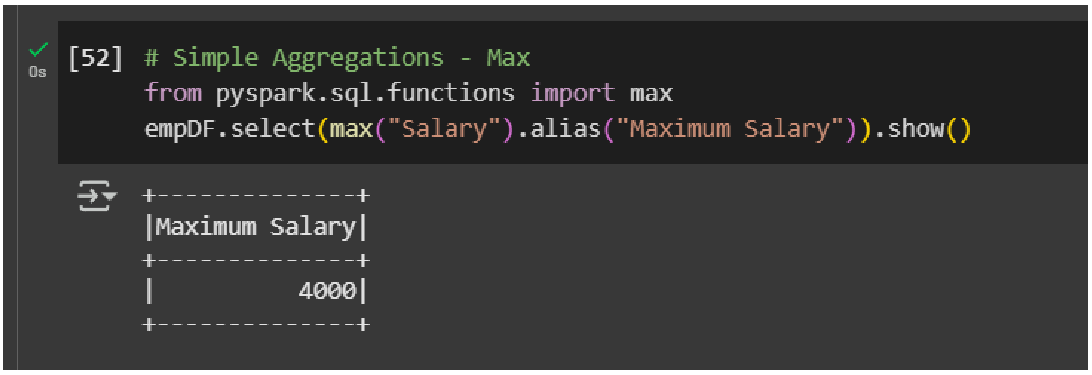
3. Average



```
[50] # Simple Aggregations - Avg
     from pyspark.sql.functions import avg
     empDF.select(avg("Salary").alias("Average Salary")).show()
```

```
+-----------------+
|   Average Salary|
+-----------------+
|1666.333333333333|
+-----------------+
```

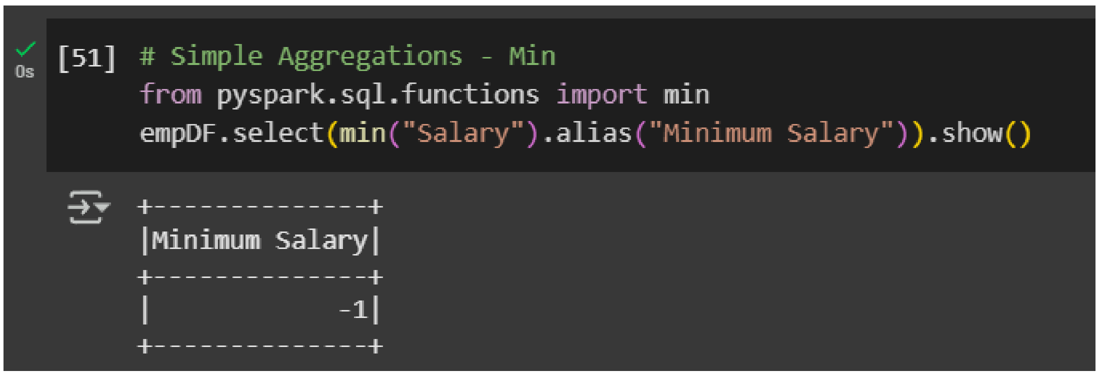
4. Maximum



```
[52] # Simple Aggregations - Max
     from pyspark.sql.functions import max
     empDF.select(max("Salary").alias("Maximum Salary")).show()
```

```
+--------------+
|Maximum Salary|
+--------------+
|          4000|
+--------------+
```

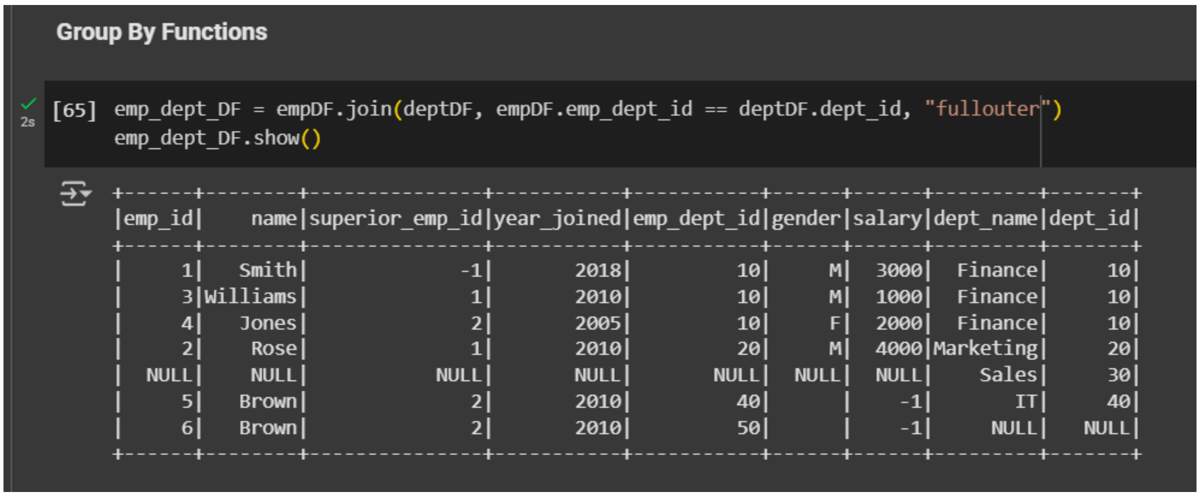
5. Minimum



```
[51] # Simple Aggregations - Min
     from pyspark.sql.functions import min
     empDF.select(min("Salary").alias("Minimum Salary")).show()
```

```
+--------------+
|Minimum Salary|
+--------------+
|            -1|
+--------------+
```

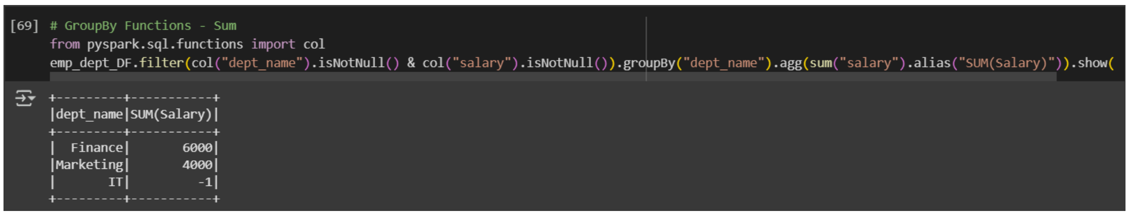
Group By Functions


Group By Functions

```
[65] emp_dept_DF = empDF.join(deptDF, empDF.emp_dept_id == deptDF.dept_id, "fullouter")
     emp_dept_DF.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|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

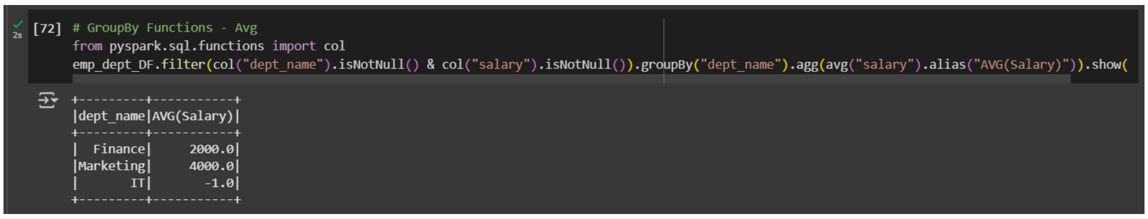
1. Sum



```
[69] # GroupBy Functions - Sum
     from pyspark.sql.functions import col
     emp_dept_DF.filter(col("dept_name").isNotNull() & col("salary").isNotNull()).groupBy("dept_name").agg(sum("salary").alias("SUM(Salary)")).show(
```

```
+---------+-----------+
|dept_name|SUM(Salary)|
+---------+-----------+
|  Finance|       6000|
|Marketing|       4000|
|       IT|         -1|
+---------+-----------+
```

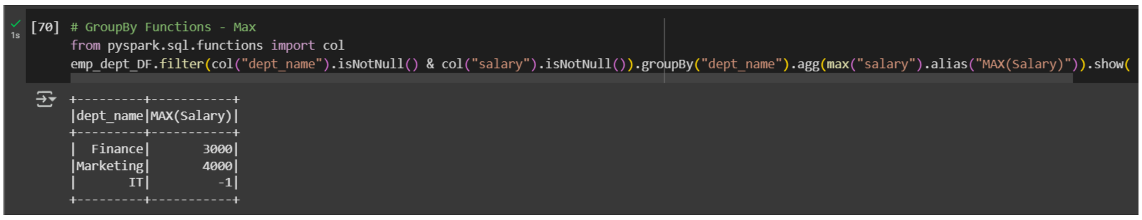
2. Average



```
[72] # GroupBy Functions - Avg
     from pyspark.sql.functions import col
     emp_dept_DF.filter(col("dept_name").isNotNull() & col("salary").isNotNull()).groupBy("dept_name").agg(avg("salary").alias("AVG(Salary)")).show(
```

```
+---------+-----------+
|dept_name|AVG(Salary)|
+---------+-----------+
|  Finance|     2000.0|
|Marketing|     4000.0|
|       IT|       -1.0|
+---------+-----------+
```

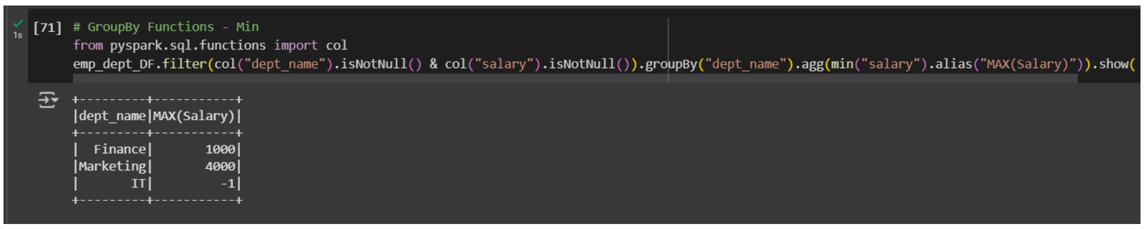
3. Maximum



```
[70] # GroupBy Functions - Max
     from pyspark.sql.functions import col
     emp_dept_DF.filter(col("dept_name").isNotNull() & col("salary").isNotNull()).groupBy("dept_name").agg(max("salary").alias("MAX(Salary)")).show(
```

```
+---------+-----------+
|dept_name|MAX(Salary)|
+---------+-----------+
|  Finance|       3000|
|Marketing|       4000|
|       IT|         -1|
+---------+-----------+
```

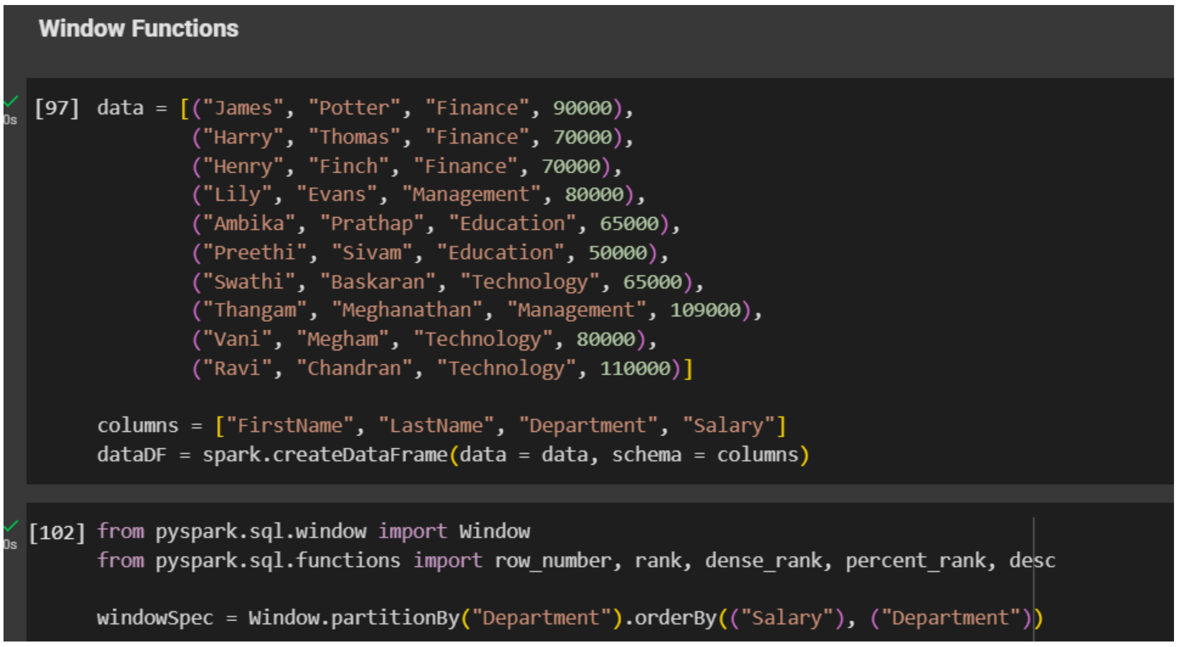
4. Minimum



```
[71] # GroupBy Functions - Min
     from pyspark.sql.functions import col
     emp_dept_DF.filter(col("dept_name").isNotNull() & col("salary").isNotNull()).groupBy("dept_name").agg(min("salary").alias("MAX(Salary)")).show(
```

```
+---------+-----------+
|dept_name|MAX(Salary)|
+---------+-----------+
|  Finance|       1000|
|Marketing|       4000|
|       IT|         -1|
+---------+-----------+
```

Window Functions



Window Functions

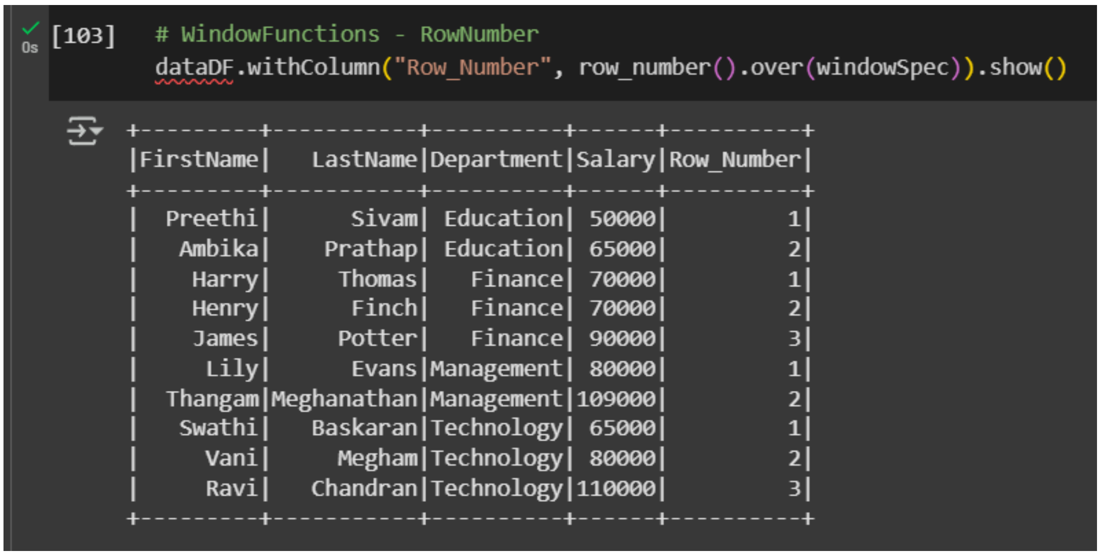
```
[97] data = [("James", "Potter", "Finance", 90000),
            ("Harry", "Thomas", "Finance", 70000),
            ("Henry", "Finch", "Finance", 70000),
            ("Lily", "Evans", "Management", 80000),
            ("Ambika", "Prathap", "Education", 65000),
            ("Preethi", "Sivam", "Education", 50000),
            ("Swathi", "Baskaran", "Technology", 65000),
            ("Thangam", "Meghanathan", "Management", 109000),
            ("Vani", "Megham", "Technology", 80000),
            ("Ravi", "Chandran", "Technology", 110000)]

     columns = ["FirstName", "LastName", "Department", "Salary"]
     dataDF = spark.createDataFrame(data = data, schema = columns)
```

```
[102] from pyspark.sql.window import Window
      from pyspark.sql.functions import row_number, rank, dense_rank, percent_rank, desc

      windowSpec = Window.partitionBy("Department").orderBy(("Salary"), ("Department"))
```

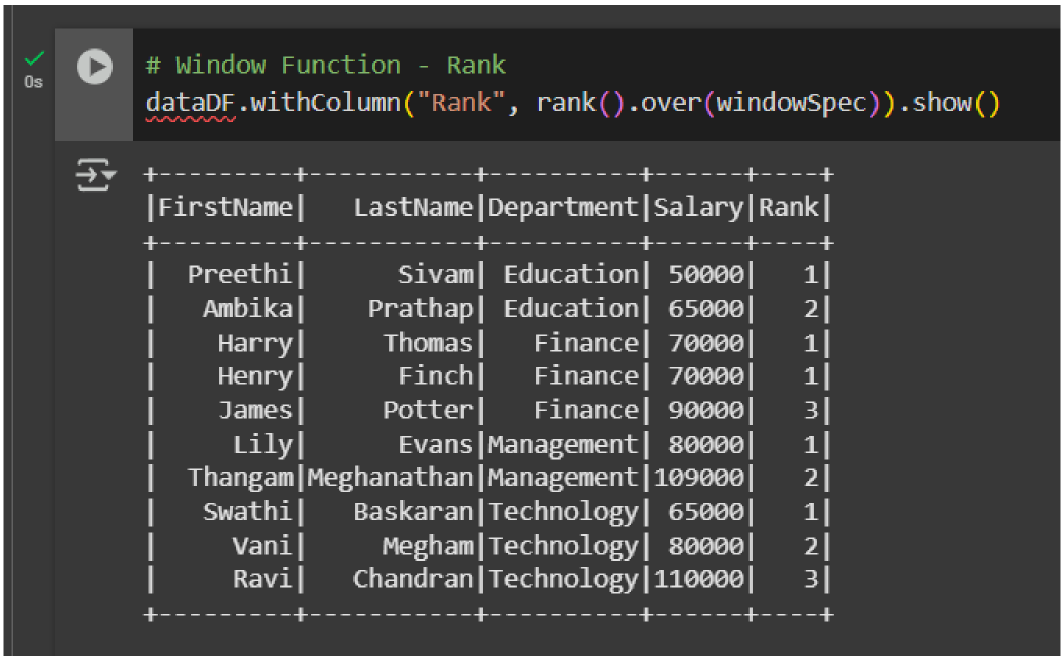
1. Row Number



```
[103]  # WindowFunctions - RowNumber
       dataDF.withColumn("Row_Number", row_number().over(windowSpec)).show()
```

```
+---------+-----------+----------+------+----------+
|FirstName|   LastName|Department|Salary|Row_Number|
+---------+-----------+----------+------+----------+
|  Preethi|      Sivam| Education| 50000|         1|
|   Ambika|    Prathap| Education| 65000|         2|
|    Harry|     Thomas|   Finance| 70000|         1|
|    Henry|      Finch|   Finance| 70000|         2|
|    James|     Potter|   Finance| 90000|         3|
|     Lily|      Evans|Management| 80000|         1|
|  Thangam|Meghanathan|Management|109000|         2|
|   Swathi|   Baskaran|Technology| 65000|         1|
|     Vani|     Megham|Technology| 80000|         2|
|     Ravi|   Chandran|Technology|110000|         3|
+---------+-----------+----------+------+----------+
```

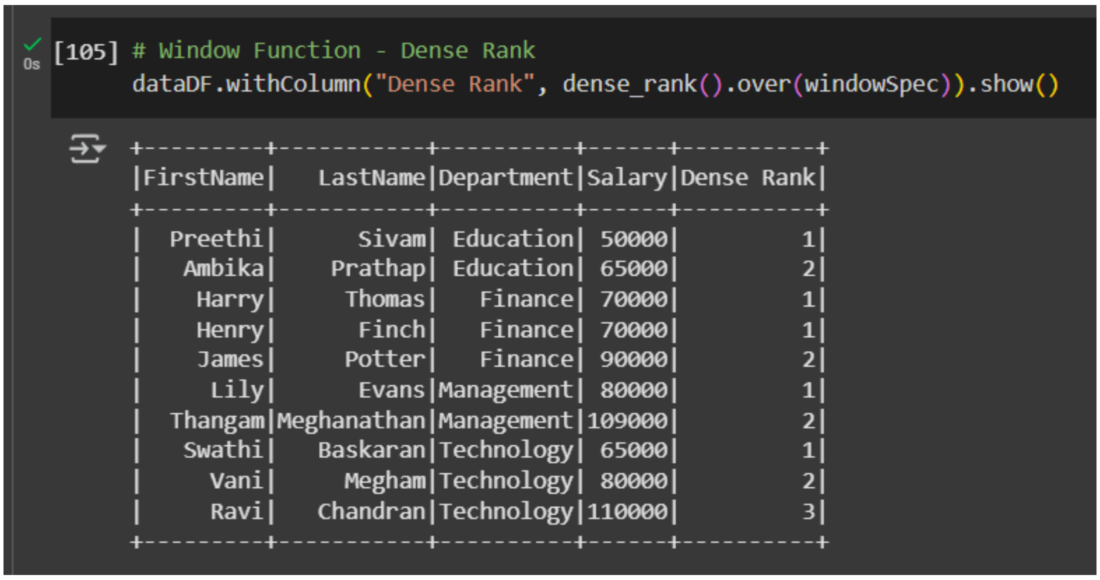
2. Rank



```
# Window Function - Rank
dataDF.withColumn("Rank", rank().over(windowSpec)).show()
```

```
+---------+-----------+----------+------+----+
|FirstName|   LastName|Department|Salary|Rank|
+---------+-----------+----------+------+----+
|  Preethi|      Sivam| Education| 50000|   1|
|   Ambika|    Prathap| Education| 65000|   2|
|    Harry|     Thomas|   Finance| 70000|   1|
|    Henry|      Finch|   Finance| 70000|   1|
|    James|     Potter|   Finance| 90000|   3|
|     Lily|      Evans|Management| 80000|   1|
|  Thangam|Meghanathan|Management|109000|   2|
|   Swathi|   Baskaran|Technology| 65000|   1|
|     Vani|     Megham|Technology| 80000|   2|
|     Ravi|   Chandran|Technology|110000|   3|
+---------+-----------+----------+------+----+
```

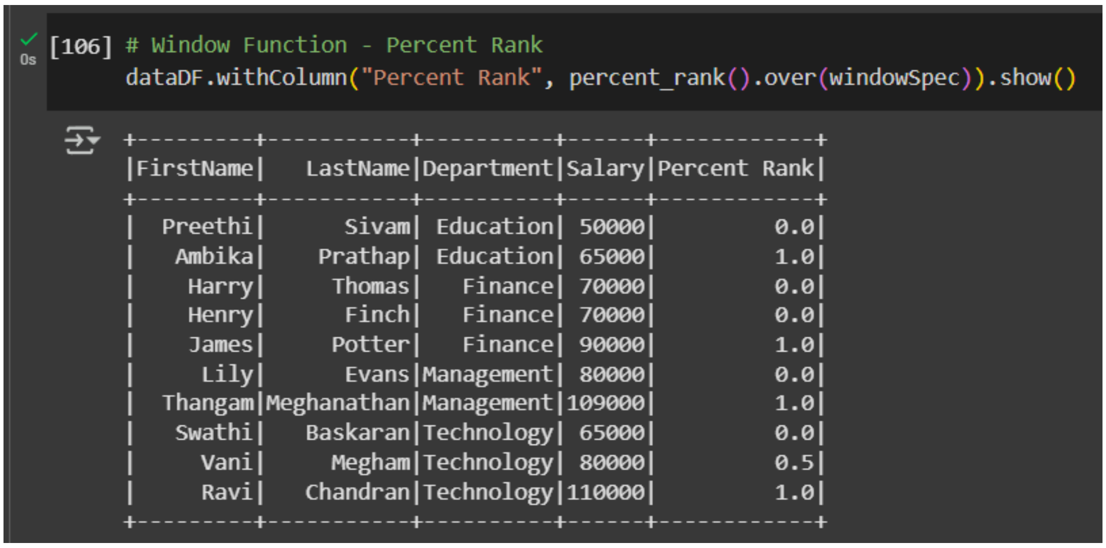
3. Dense Rank



```
[105] # Window Function - Dense Rank
dataDF.withColumn("Dense Rank", dense_rank().over(windowSpec)).show()
```

```
+---------+-----------+----------+------+----------+
|FirstName|   LastName|Department|Salary|Dense Rank|
+---------+-----------+----------+------+----------+
|  Preethi|      Sivam| Education| 50000|         1|
|   Ambika|    Prathap| Education| 65000|         2|
|    Harry|     Thomas|   Finance| 70000|         1|
|    Henry|      Finch|   Finance| 70000|         1|
|    James|     Potter|   Finance| 90000|         2|
|     Lily|      Evans|Management| 80000|         1|
|  Thangam|Meghanathan|Management|109000|         2|
|   Swathi|   Baskaran|Technology| 65000|         1|
|     Vani|     Megham|Technology| 80000|         2|
|     Ravi|   Chandran|Technology|110000|         3|
+---------+-----------+----------+------+----------+
```

4. Percent Rank



```
[106] # Window Function - Percent Rank
      dataDF.withColumn("Percent Rank", percent_rank().over(windowSpec)).show()
```

```
+---------+----------+----------+------+------------+
|FirstName|  LastName|Department|Salary|Percent Rank|
+---------+----------+----------+------+------------+
|  Preethi|     Sivam| Education| 50000|         0.0|
|   Ambika|   Prathap| Education| 65000|         1.0|
|    Harry|    Thomas|   Finance| 70000|         0.0|
|    Henry|     Finch|   Finance| 70000|         0.0|
|    James|    Potter|   Finance| 90000|         1.0|
|     Lily|     Evans|Management| 80000|         0.0|
|  Thangam|Meghanathan|Management|109000|         1.0|
|   Swathi|  Baskaran|Technology| 65000|         0.0|
|     Vani|    Megham|Technology| 80000|         0.5|
|     Ravi|  Chandran|Technology|110000|         1.0|
+---------+----------+----------+------+------------+
```