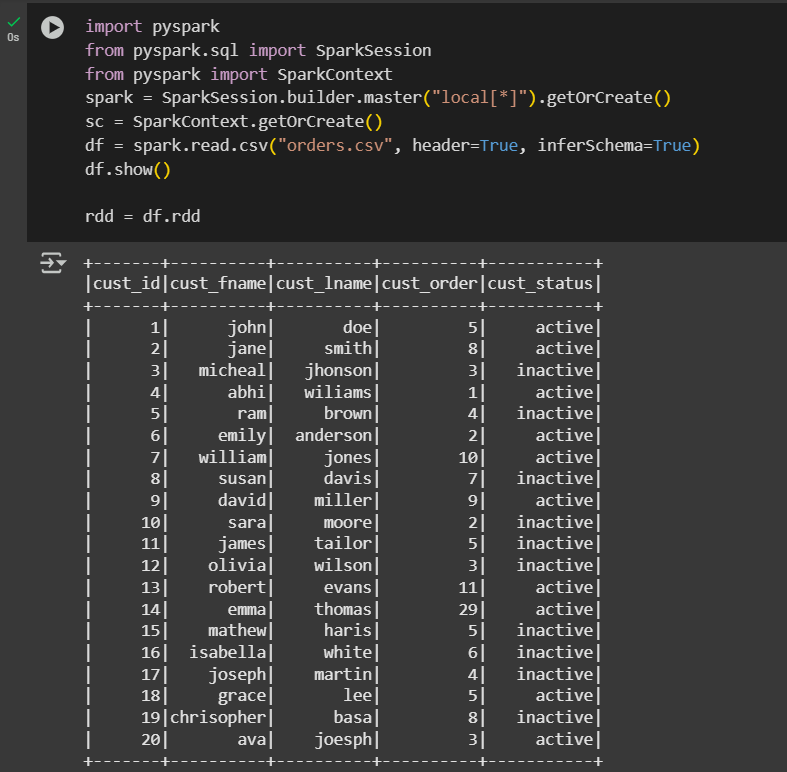
**Coding Assessment – Pyspark**

**Pyspark setup**

****

**Transformations in PySpark**

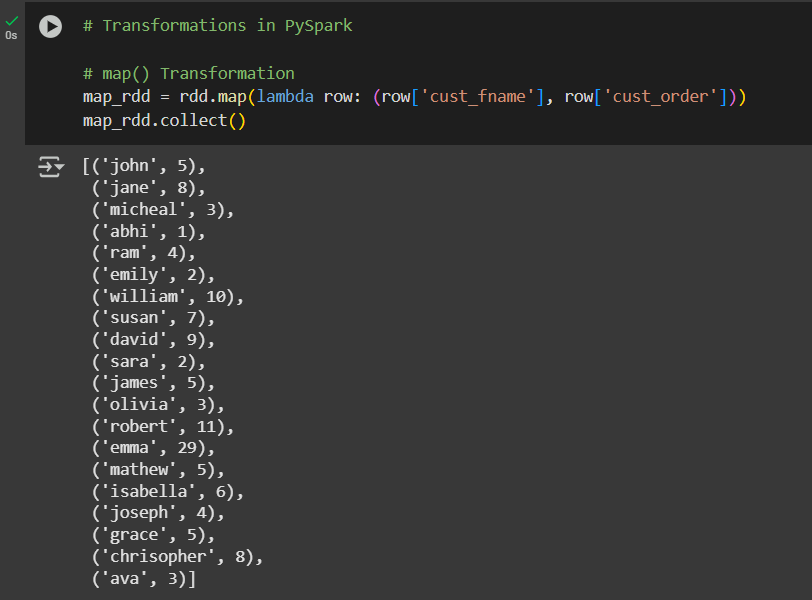
* map() Transformation
* filter() Transformation
* union() Transformation
* flatMap() Transformation

**# Transformations in PySpark**

# map() Transformation

map\_rdd = rdd.map(lambda row: (row['cust\_fname'], row['cust\_order']))

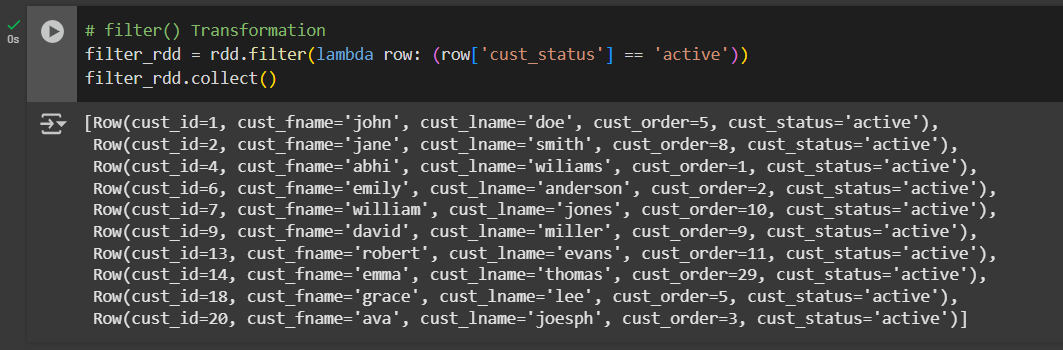
map\_rdd.collect()



# filter() Transformation

filter\_rdd = rdd.filter(lambda row: (row['cust\_status'] == 'active'))

filter\_rdd.collect()



# union() Transformation

from pyspark.sql import Row

test\_orders = spark.sparkContext.parallelize([

Row(cust\_id=21, cust\_fname='demo', cust\_lname='user', cust\_order=1, cust\_status='active')])

union\_rdd = rdd.union(test\_orders)

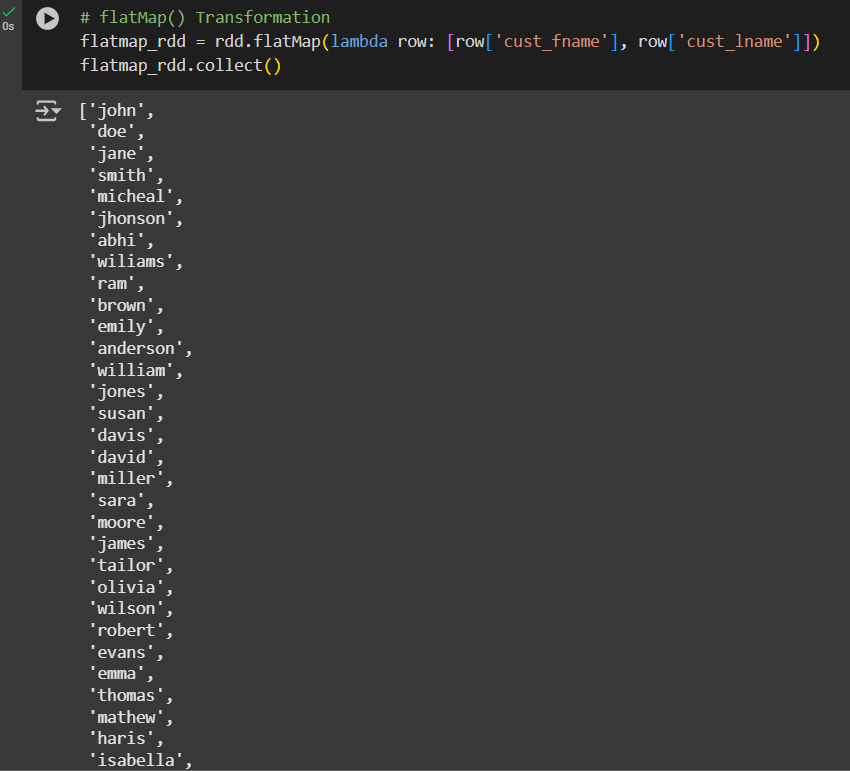
union\_rdd.collect()



# flatMap() Transformation

flatmap\_rdd = rdd.flatMap(lambda row: [row['cust\_fname'], row['cust\_lname']])

flatmap\_rdd.collect()



**Actions in PySpark**

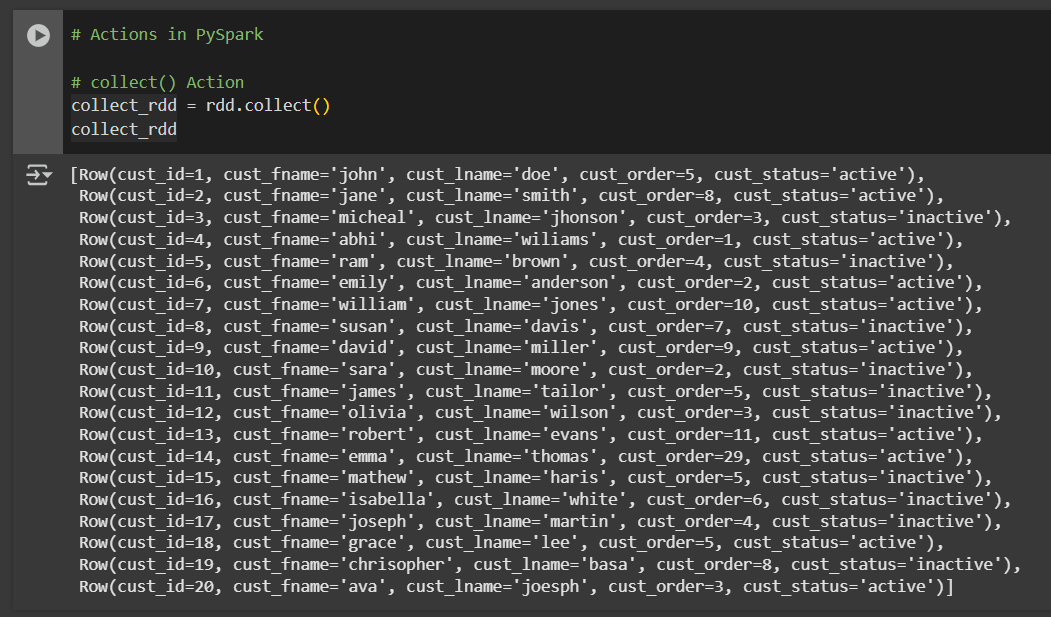
* collect() Action
* count() Action
* first() Action
* take() Action
* reduce() Action
* saveAsTextFile() Action

**# Actions in PySpark**

# collect() Action

collect\_rdd = rdd.collect()

collect\_rdd

****

# count() Action

count\_rdd = rdd.count()

count\_rdd

# first() Action

first\_rdd = rdd.first()

first\_rdd

# take() Action

take\_rdd = rdd.take(5)

take\_rdd

# reduce() Action

reduce\_rdd = rdd.map(lambda row: int(row['cust\_order'])).reduce(lambda a, b: a + b)

print(reduce\_rdd)

# saveAsTextFile

saveAsTextFile\_rdd = rdd.saveAsTextFile("orders.txt")



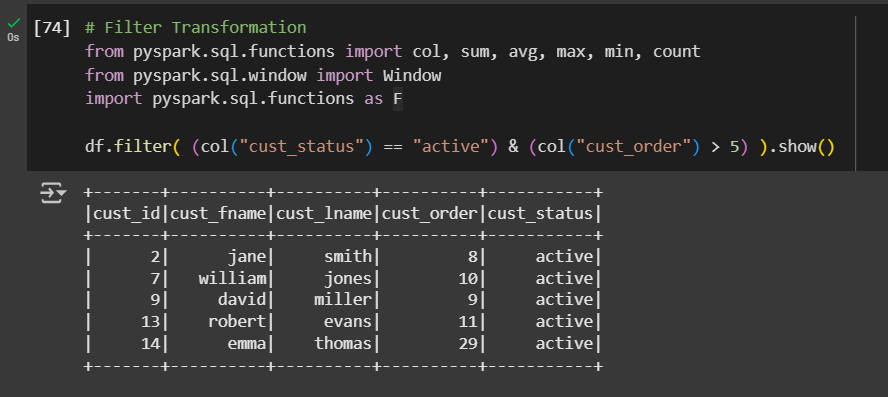
**# Filter Transformation**

from pyspark.sql.functions import col, sum, avg, max, min, count

from pyspark.sql.window import Window

import pyspark.sql.functions as F

df.filter( (col("cust\_status") == "active") & (col("cust\_order") > 5) ).show()



# Joins

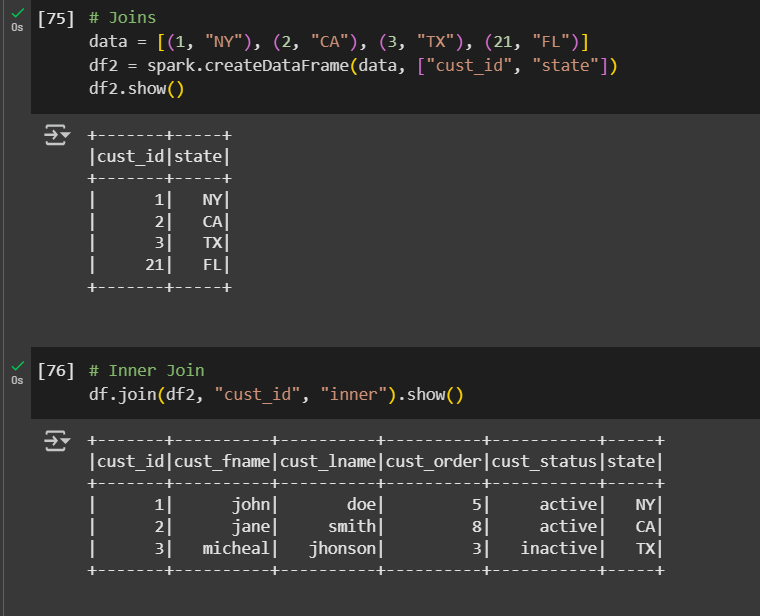
data = [(1, "NY"), (2, "CA"), (3, "TX"), (21, "FL")]

df2 = spark.createDataFrame(data, ["cust\_id", "state"])

df2.show()

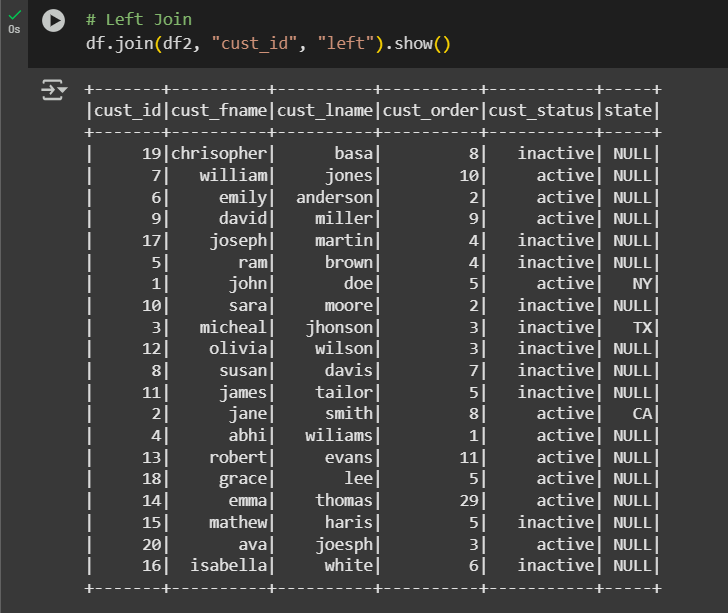
# Inner Join

df.join(df2, "cust\_id", "inner").show()



# Left Join

df.join(df2, "cust\_id", "left").show()

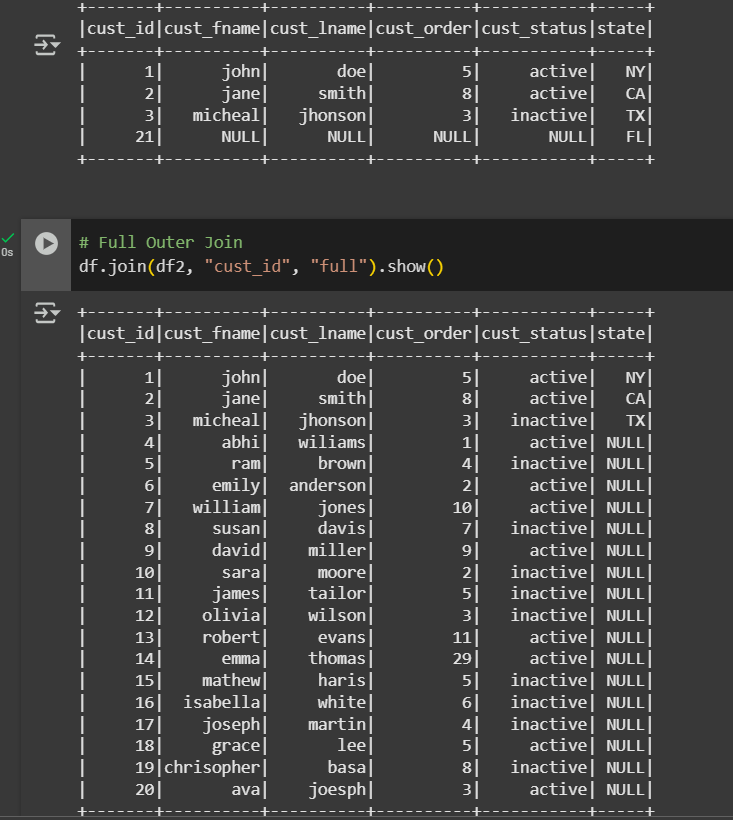


# right join

df.join(df2, "cust\_id", "right").show()

# Full Outer Join

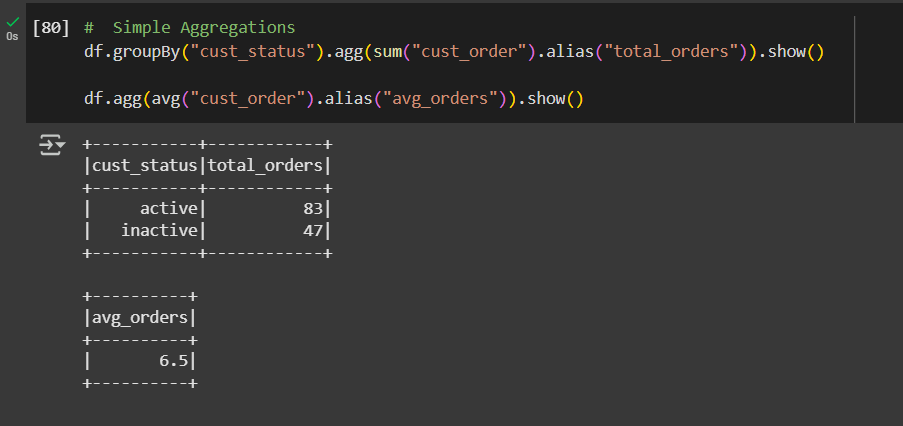
df.join(df2, "cust\_id", "full").show()



#  Simple Aggregations

df.groupBy("cust\_status").agg(sum("cust\_order").alias("total\_orders")).show()

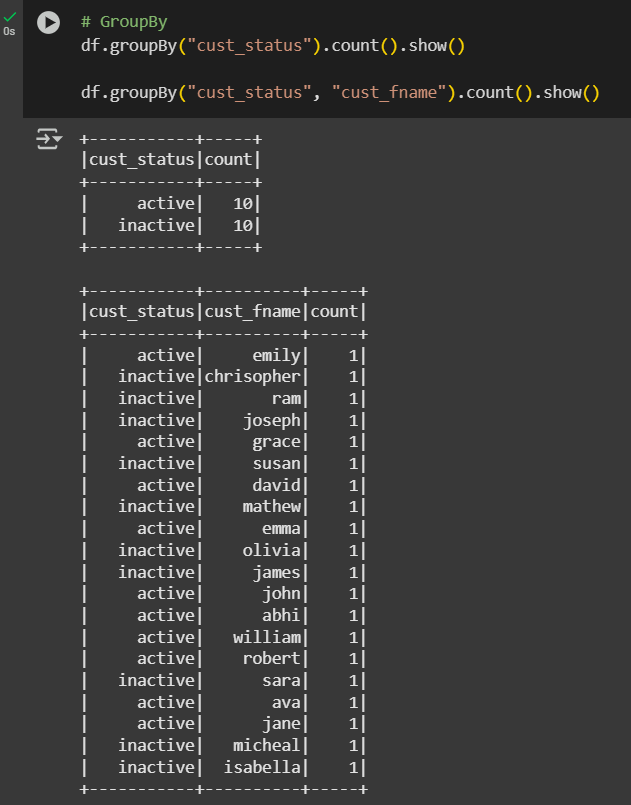
df.agg(avg("cust\_order").alias("avg\_orders")).show()



# GroupBy

df.groupBy("cust\_status").count().show()

df.groupBy("cust\_status", "cust\_fname").count().show()

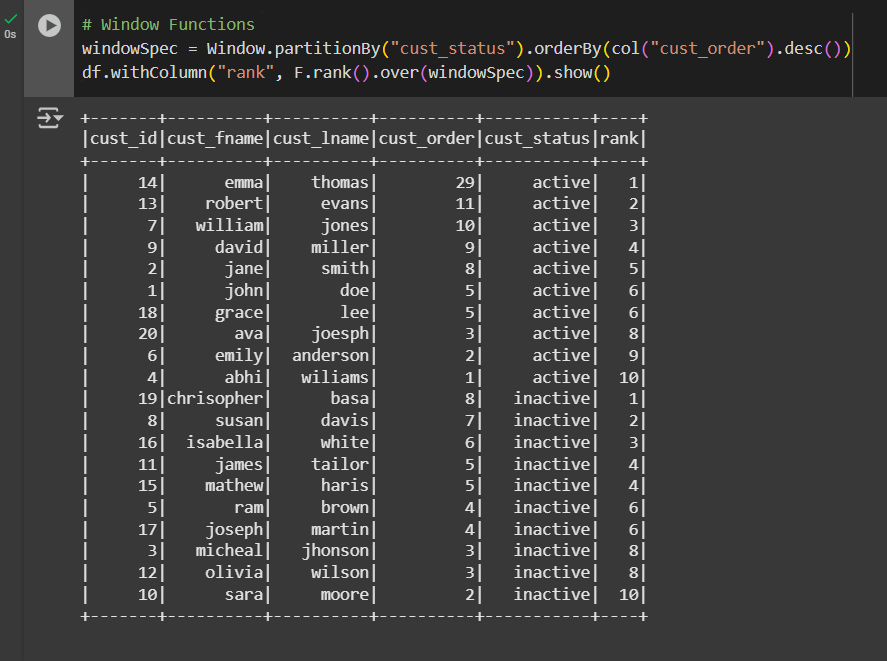


# Window Functions

windowSpec=

Window.partitionBy("cust\_status").orderBy(col("cust\_order").desc())

df.withColumn("rank", F.rank().over(windowSpec)).show()



**Submitted by**

**Siva Balan T**