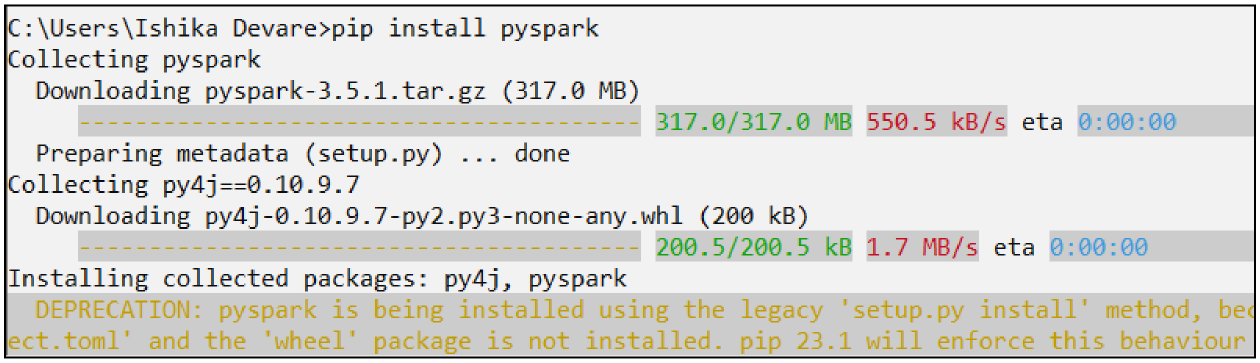
Group 10

14- Ishika Devare
26- Janvi Khatri
45- Sahil Pabale

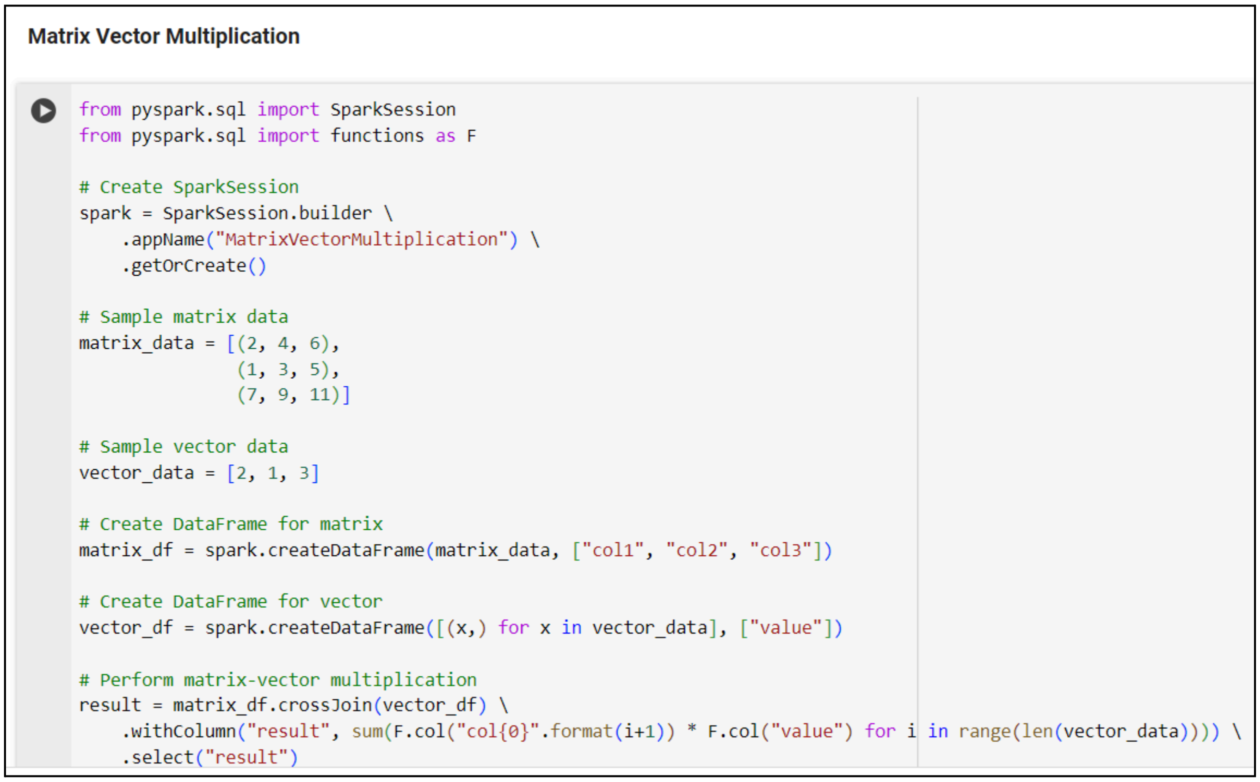
BDA MiniProject CA2

❖ Write MapReduce/Spark Program to perform



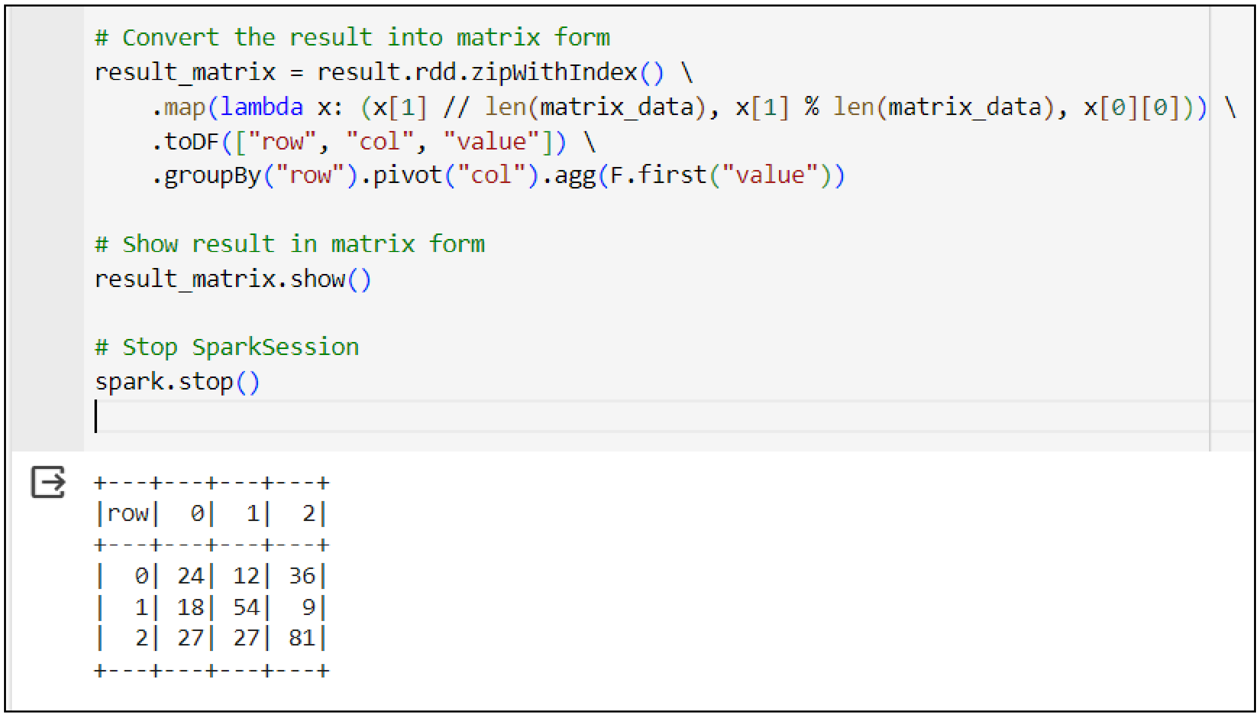
```
C:\Users\Ishika Devare>pip install pyspark
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ---------------------------------------- 317.0/317.0 MB 550.5 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.7
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
     ---------------------------------------- 200.5/200.5 kB 1.7 MB/s eta 0:00:00
Installing collected packages: py4j, pyspark
  DEPRECATION: pyspark is being installed using the legacy 'setup.py install' method, bec
ect.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour
```

1. Matrix Vector Multiplication



Matrix Vector Multiplication

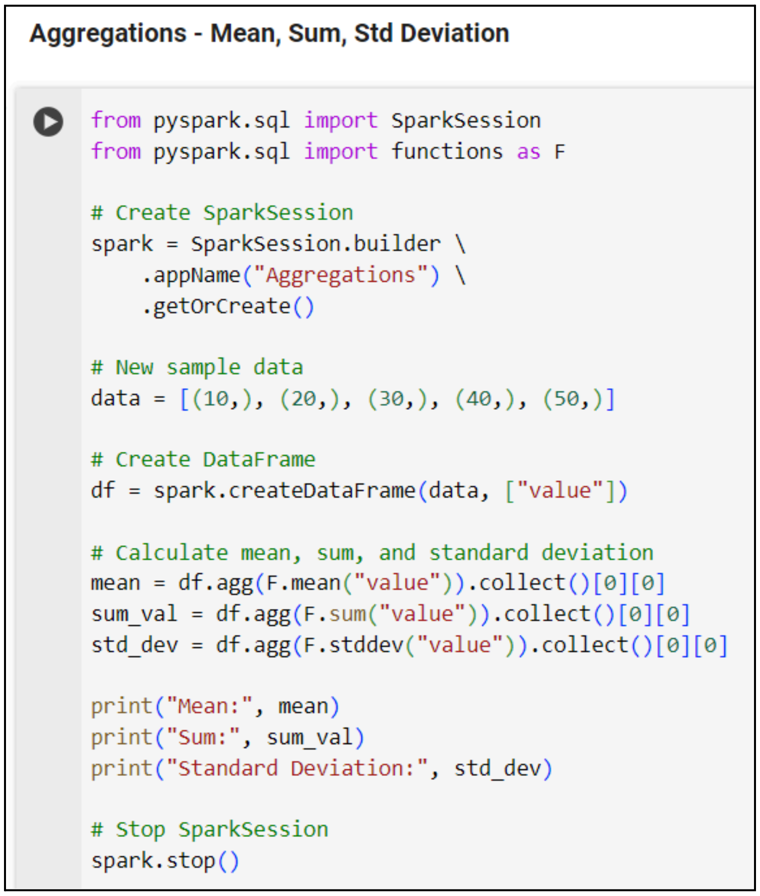
```python
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

# Create SparkSession
spark = SparkSession.builder \
    .appName("MatrixVectorMultiplication") \
    .getOrCreate()

# Sample matrix data
matrix_data = [(2, 4, 6),
               (1, 3, 5),
               (7, 9, 11)]

# Sample vector data
vector_data = [2, 1, 3]

# Create DataFrame for matrix
matrix_df = spark.createDataFrame(matrix_data, ["col1", "col2", "col3"])

# Create DataFrame for vector
vector_df = spark.createDataFrame([(x,) for x in vector_data], ["value"])

# Perform matrix-vector multiplication
result = matrix_df.crossJoin(vector_df) \
    .withColumn("result", sum(F.col("col{0}".format(i+1)) * F.col("value") for i in range(len(vector_data)))) \
    .select("result")
```



```
# Convert the result into matrix form
result_matrix = result.rdd.zipWithIndex() \
    .map(lambda x: (x[1] // len(matrix_data), x[1] % len(matrix_data), x[0][0])) \
    .toDF(["row", "col", "value"]) \
    .groupBy("row").pivot("col").agg(F.first("value"))

# Show result in matrix form
result_matrix.show()

# Stop SparkSession
spark.stop()
```

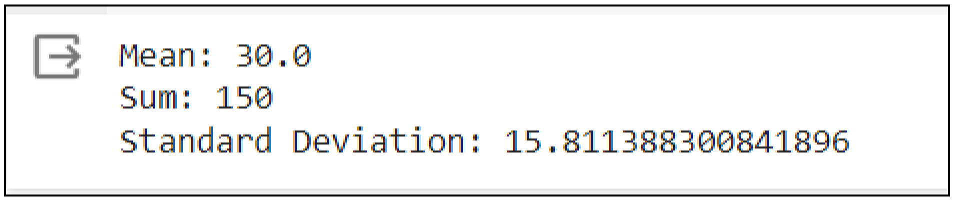
```
+---+---+---+---+
|row|  0|  1|  2|
+---+---+---+---+
|  0| 24| 12| 36|
|  1| 18| 54|  9|
|  2| 27| 27| 81|
+---+---+---+---+
```

2. Aggregations - Mean, Sum, Std Deviation



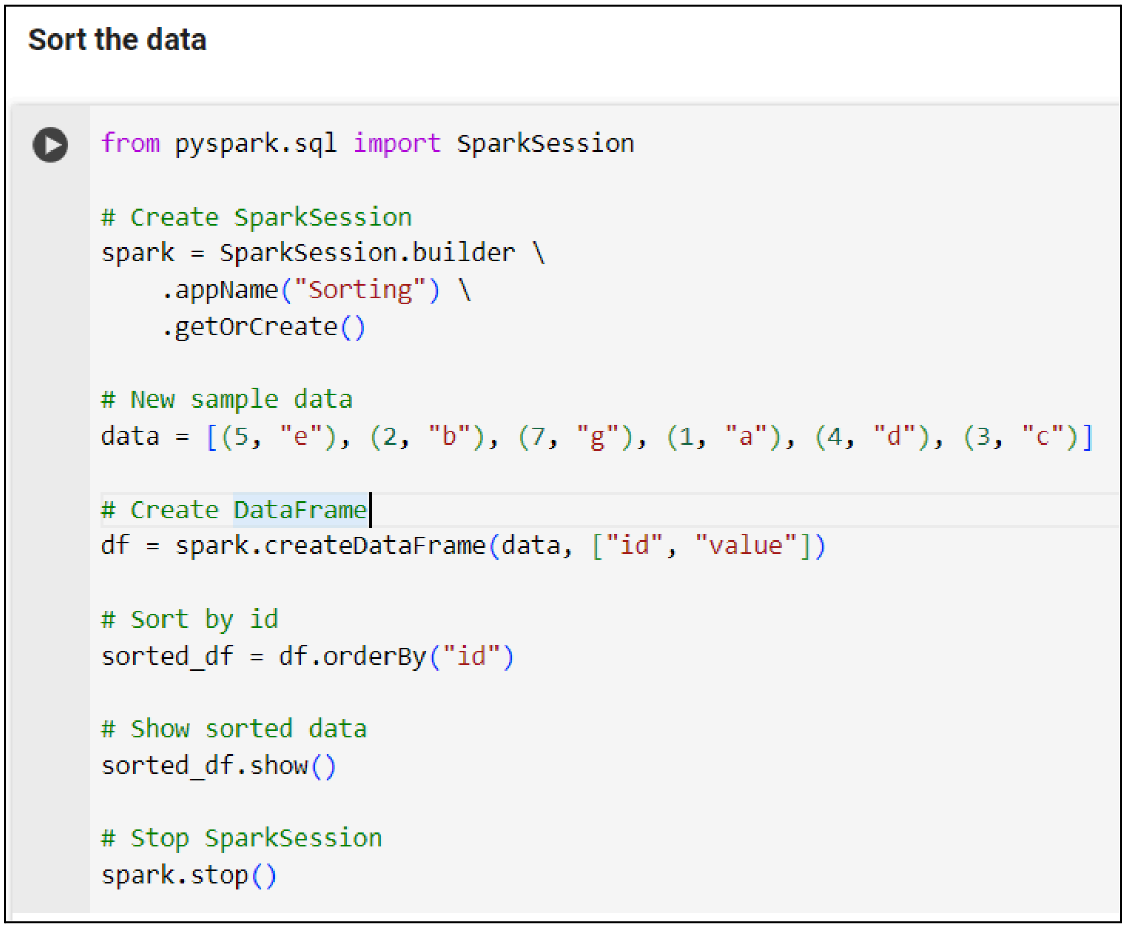
Aggregations - Mean, Sum, Std Deviation

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

# Create SparkSession
spark = SparkSession.builder \
    .appName("Aggregations") \
    .getOrCreate()

# New sample data
data = [(10,), (20,), (30,), (40,), (50,)]

# Create DataFrame
df = spark.createDataFrame(data, ["value"])

# Calculate mean, sum, and standard deviation
mean = df.agg(F.mean("value")).collect()[0][0]
sum_val = df.agg(F.sum("value")).collect()[0][0]
std_dev = df.agg(F.stddev("value")).collect()[0][0]

print("Mean:", mean)
print("Sum:", sum_val)
print("Standard Deviation:", std_dev)

# Stop SparkSession
spark.stop()
```



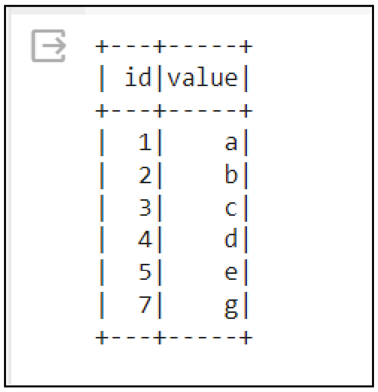
```
Mean: 30.0
Sum: 150
Standard Deviation: 15.811388300841896
```

3. Sort the data



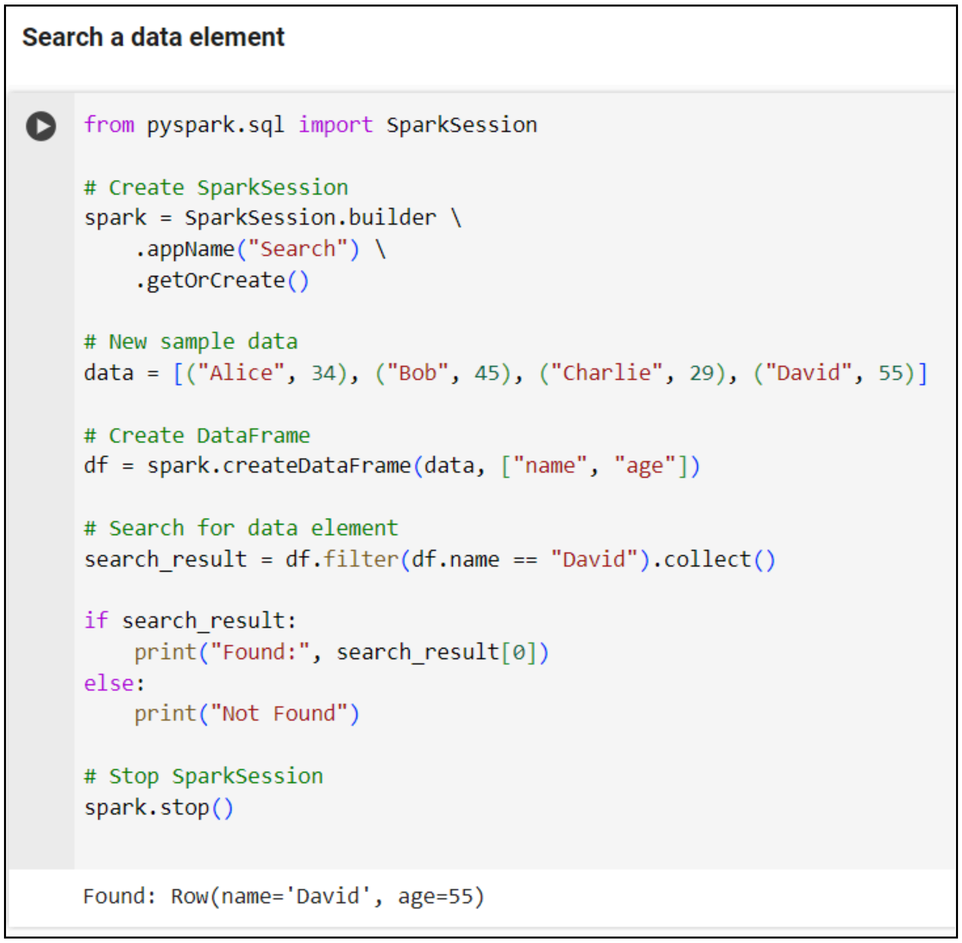
Sort the data

```python
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
    .appName("Sorting") \
    .getOrCreate()

# New sample data
data = [(5, "e"), (2, "b"), (7, "g"), (1, "a"), (4, "d"), (3, "c")]

# Create DataFrame
df = spark.createDataFrame(data, ["id", "value"])

# Sort by id
sorted_df = df.orderBy("id")

# Show sorted data
sorted_df.show()

# Stop SparkSession
spark.stop()
```



```
+---+-----+
| id|value|
+---+-----+
|  1|    a|
|  2|    b|
|  3|    c|
|  4|    d|
|  5|    e|
|  7|    g|
+---+-----+
```

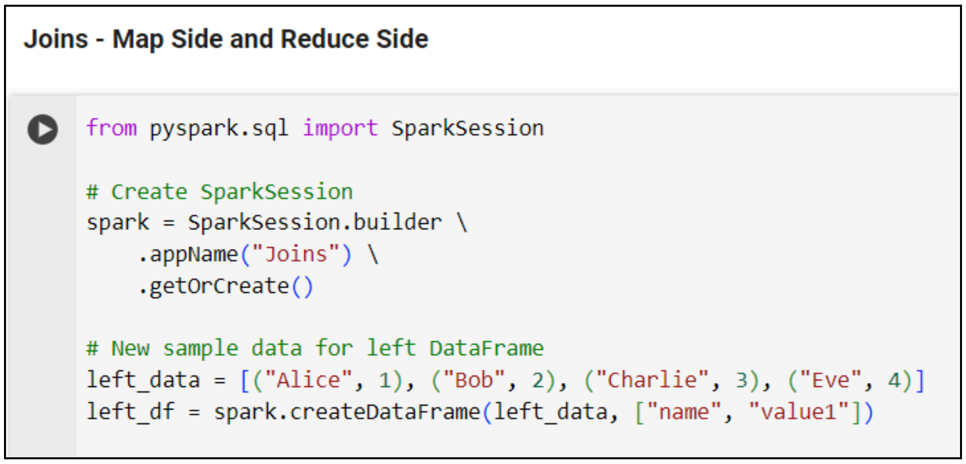
4. Search a data element


Search a data element

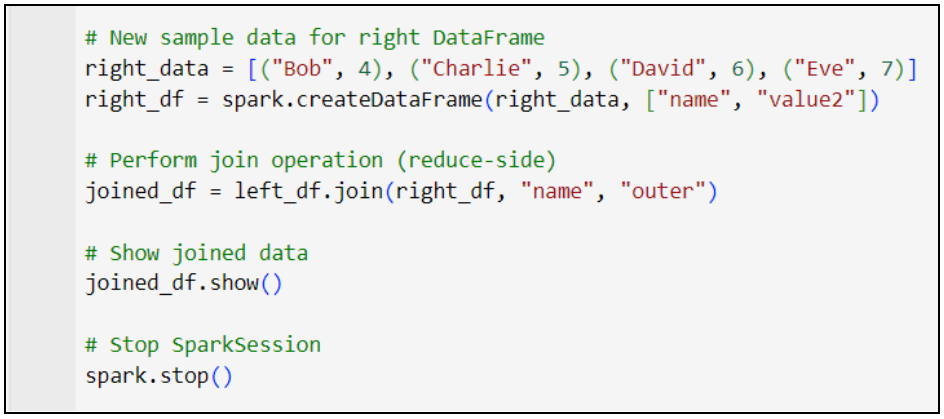
```
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
    .appName("Search") \
    .getOrCreate()

# New sample data
data = [("Alice", 34), ("Bob", 45), ("Charlie", 29), ("David", 55)]

# Create DataFrame
df = spark.createDataFrame(data, ["name", "age"])

# Search for data element
search_result = df.filter(df.name == "David").collect()

if search_result:
    print("Found:", search_result[0])
else:
    print("Not Found")

# Stop SparkSession
spark.stop()
```

```
Found: Row(name='David', age=55)
```

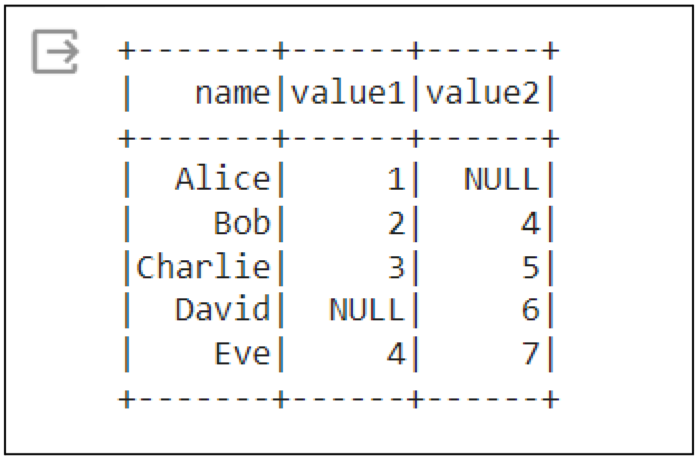
5. Joins - Map Side and Reduce Side


Joins - Map Side and Reduce Side

```
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
    .appName("Joins") \
    .getOrCreate()

# New sample data for left DataFrame
left_data = [("Alice", 1), ("Bob", 2), ("Charlie", 3), ("Eve", 4)]
left_df = spark.createDataFrame(left_data, ["name", "value1"])
```



```python
# New sample data for right DataFrame
right_data = [("Bob", 4), ("Charlie", 5), ("David", 6), ("Eve", 7)]
right_df = spark.createDataFrame(right_data, ["name", "value2"])

# Perform join operation (reduce-side)
joined_df = left_df.join(right_df, "name", "outer")

# Show joined data
joined_df.show()

# Stop SparkSession
spark.stop()
```



```
+-------+------+------+
|   name|value1|value2|
+-------+------+------+
|  Alice|     1|  NULL|
|    Bob|     2|     4|
|Charlie|     3|     5|
|  David|  NULL|     6|
|    Eve|     4|     7|
+-------+------+------+
```