**PYSPARK EXAMPLES**

# Import PySpark and initialize a Spark session

from pyspark.sql import SparkSession

# Initialize SparkSession

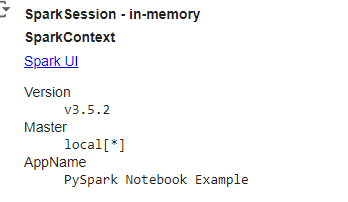
spark = SparkSession.builder \ # \(slash) means still code in new line

    .appName("PySpark Notebook Example") \

    .getOrCreate()

# Verify the Spark session is working

spark



* Output ^



from pyspark.sql import SparkSession

from pyspark.sql.functions import col

spark = SparkSession.builder \

       .appName('Pyspark Dataframe Example') \

       .getOrCreate()

# sample data

data = [

    ("John Doe","Engineering",75000),

    ("Jane Smith","Marketing",60000),

    ("Sam Brown","Marketing",80000),

    ("Emily Davis","HR",50000),

    ("Michael Johnson","Marketing",70000)

]

# schema defining

columns = ["Name","Department","Salary"]

# Creating Dataframe

df = spark.createDataFrame(data,schema=columns)

# showing the DataFrame

df.show()

**output:**

+---------------+-----------+------+

| Name| Department|Salary|

+---------------+-----------+------+

| John Doe|Engineering| 75000|

| Jane Smith| Marketing| 60000|

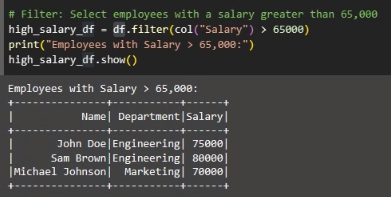
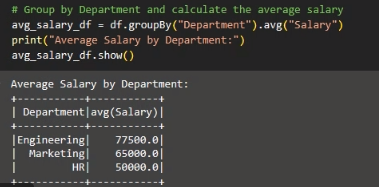
| Sam Brown| Marketing| 80000|

| Emily Davis| HR| 50000|

|Michael Johnson| Marketing| 70000|

+---------------+-----------+------+

---------------------------------------------------------------------------------------------------------

high\_salary\_df = df.filter(col("Salary") > 65000)

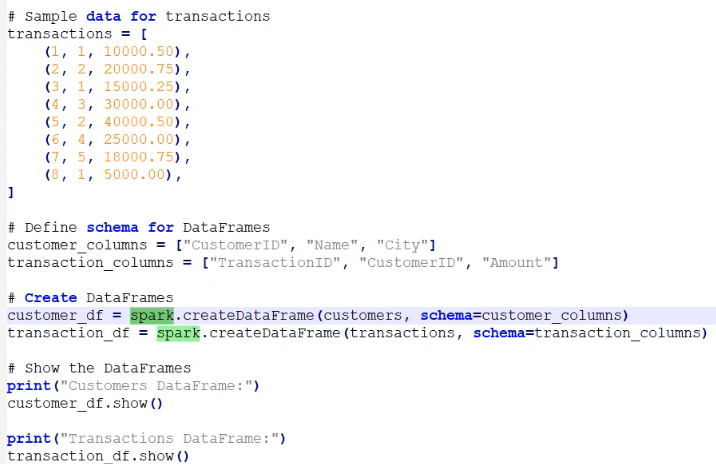
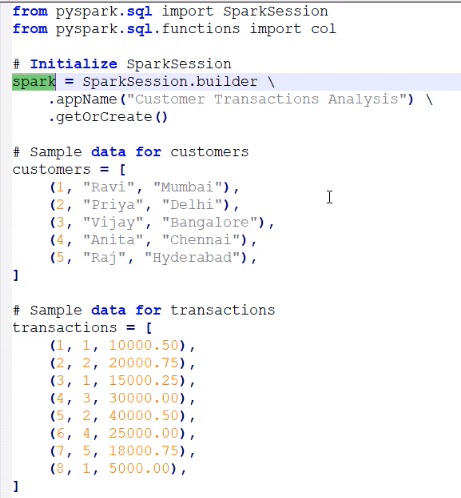
print("High salary above 65000")

high\_salary\_df.show()

average\_salary\_df = df.groupBy("Department").avg("Salary")

print("Average salary by department")

average\_salary\_df.show()



from pyspark.sql import SparkSession

from pyspark.sql.functions import col

# Initialize spark session

spark = SparkSession.builder \

       .appName('Customer Transaction Analysis') \

       .getOrCreate()

# sample data for customers

customes = [

    (1,"Ravi","Mumbai"),

    (2,"Priya","Delhi"),

    (3,"Vijay","Bangalore"),

    (4,"Anita","Chennai"),

    (5,"Raj","Hyderabad"),

]

# sample data for transactions

trasctions = [

    (1,1,10000.50),

    (2,2,20000.75),

    (3,1,15000.25),

    (4,3,30000.00),

    (5,2,40000.50),

    (6,4,25000.00),

    (7,5,18000.75),

    (8,1,5000.00),

]

# Define schema for DataFrames

customer\_columns = ["customerID","Name","City"]

transaction\_columns = ["TransactionID","CustomerID","Amount"]

# Create Dataframes

customer\_df = spark.createDataFrame(customes,schema=customer\_columns)

transaction\_df = spark.createDataFrame(trasctions,schema=transaction\_columns)

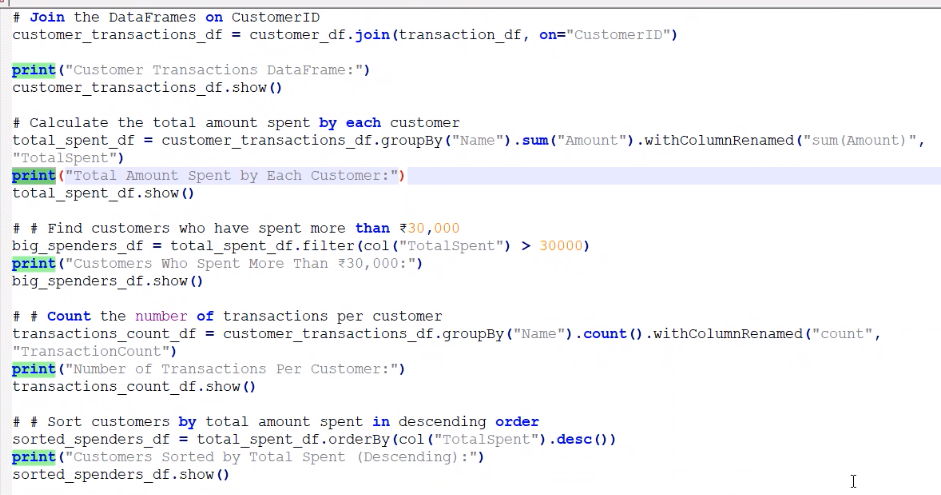
# Show tha DataFrames

print("Customer DataFrame:")

customer\_df.show()

print("Transaction DataFrame:")

transaction\_df.show()



# Join the DataFrame on CustomerId

customer\_transactions\_df = customer\_df.join(transaction\_df,on = "CustomerID")

# Show the joined DataFrame

print("Customer Transactions DataFrame:")

customer\_transactions\_df.show()

# calculate the total amount spent by each customer

totalspent\_df = customer\_transactions\_df.groupBy("CustomerID").sum("Amount").withColumnRenamed("sum(Amount)","TotalSpent")

# Show the total amount spent by each customer

print("Total Amount Spent by Each Customer:")

totalspent\_df.show()

# Total amount spent by each customer

total\_spent\_df = customer\_transactions\_df.groupBy("Name").sum("Amount").withColumnRenamed("sum(Amount)","TotalSpent")

# Show the total amount spent by each customer

print("Total Amount Spent by Each Customer:")

total\_spent\_df.show()

-----------------------------------------------------------------------

# Find customers who have spent more than 35000

big\_spent\_df = total\_spent\_df.filter(col("TotalSpent") > 35000)

# Show the customers who spent more than 35000

print("Customers who spent more than 35000:")

big\_spent\_df.show()

-----------------------------------------------------------------------

# Count the Number of Transactions per customer

transaction\_count\_df = customer\_transactions\_df.groupBy("Name").count().withColumnRenamed("count","TransactionCount")

# Show the number of transactions per customer

print("Number of Transactions per Customer:")

transaction\_count\_df.show()

-----------------------------------------------------------------------

# Sort customers by total amount spent in descending order

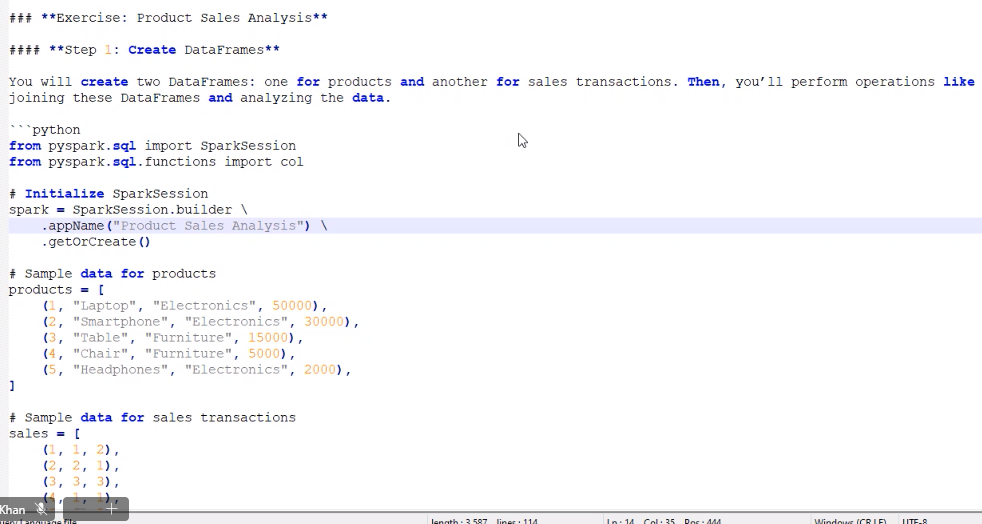
sorted\_spent\_df = total\_spent\_df.orderBy(col("TotalSpent").desc())

# Show the customers sorted by total amount spent

print("Customers Sorted by Total Amount Spent:")

sorted\_spent\_df.show()

-----------------------------------------------------------------------



from pyspark.sql import SparkSession

from pyspark.sql.functions import col

# initialize spark session

spark = SparkSession.builder.appName("Product Sales Analysis").getOrCreate()

# Sample data for products

products = [

    (1,"Laptop","Electronics",50000),

    (2,"SmartPhone","Electronics",30000),

    (3,"Table","Furnitur",15000),

    (4,"Chair","Furniture",5000),

    (5,"Headphone","Electronics",2000),

]

# sample data for sales transactions

sales = [

    (1,1,2),

    (2,2,1),

    (3,3,3),

    (4,1,1),

    (5, 4, 5),

    (6, 2, 2),

    (7, 5, 10),

    (8, 3, 1),

]

# Define schema for DataFrames

product\_columns = ["ProductID", "ProductName", "Category", "Price"]

sales\_columns = ["SaleID", "ProductID", "Quantity"]

# Create DataFrames

product\_df = spark.createDataFrame(products, schema=product\_columns)

sales\_df = spark.createDataFrame(sales, schema=sales\_columns)

# Show the DataFrames

print("Products DataFrame:")

product\_df.show()

print("Sales DataFrame:")

sales\_df.show()

# Joining the tables

joined\_df = sales\_df.join(product\_df, on="ProductId")

print("joined data :")

joined\_df.show()

# Total sales value

Total\_sales\_df = joined\_df.withColumn("Total\_Sales",col("Price")\*col("Quantity"))

print("Total sales value:")

Total\_sales\_df.show()

# Group by Category and calculate total sales value for each category

category\_sales\_df = Total\_sales\_df.groupBy("Category").sum("Total\_Sales")

category\_sales\_df = category\_sales\_df.withColumnRenamed("sum(Total\_Sales)", "TotalSalesValue")

print("Total sales value for each category:")

category\_sales\_df.show()

# Group by ProductName and calculate total sales value for each product

product\_sales\_df = Total\_sales\_df.groupBy("ProductName").sum("Total\_Sales")

product\_sales\_df = product\_sales\_df.withColumnRenamed("sum(Total\_Sales)", "TotalSalesValue")

print("Total sales value for each product:")

product\_sales\_df.show()

# Find the top-selling product

top\_selling\_product\_df = product\_sales\_df.orderBy(col("TotalSalesValue").desc()).limit(1)

print("Top selling product:")

top\_selling\_product\_df.show()

# Sort products by total sales value in descending order

sorted\_products\_df = product\_sales\_df.orderBy(col("TotalSalesValue").desc())

print("Sorted products by total sales value:")

sorted\_products\_df.show()

# Count the number of sales transactions for each product

product\_sales\_count\_df = joined\_df.groupBy("ProductName").count()

print("Number of sales Transactionhs:")

product\_sales\_count\_df.show()

# Filter products with total sales value greater than 50,000

high\_value\_products\_df = product\_sales\_df.filter(col("TotalSalesValue") > 50000)

print("High value products greated than 50000:")

high\_value\_products\_df.show()

-----------------------------------------------------------------------

**EXERCISE QUESTION AND ANSWER**

**---**

**### \*\*Exercise: Analyzing a Sample Sales Dataset Using PySpark\*\***

**In this exercise, you'll work with a simulated sales dataset and perform various data transformations and analyses using PySpark. The dataset includes fields like `TransactionID`, `CustomerID`, `ProductID`, `Quantity`, `Price`, and `Date`. Your task is to generate the dataset, load it into PySpark, and answer specific questions by performing data operations.**

**---**

**### \*\*Part 1: Dataset Preparation\*\***

**#### \*\*Step 1: Generate the Sample Sales Dataset\*\***

**Before starting the analysis, you'll need to create the sample sales dataset. Use the following Python code to generate the dataset and save it as a CSV file.**

**1. \*\*Run the Dataset Preparation Script:\*\***

**```python**

**import pandas as pd**

**from datetime import datetime**

**# Sample sales data**

**data = {**

**"TransactionID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],**

**"CustomerID": [101, 102, 103, 101, 104, 102, 103, 104, 101, 105],**

**"ProductID": [501, 502, 501, 503, 504, 502, 503, 504, 501, 505],**

**"Quantity": [2, 1, 4, 3, 1, 2, 5, 1, 2, 1],**

**"Price": [150.0, 250.0, 150.0, 300.0, 450.0, 250.0, 300.0, 450.0, 150.0, 550.0],**

**"Date": [**

**datetime(2024, 9, 1),**

**datetime(2024, 9, 1),**

**datetime(2024, 9, 2),**

**datetime(2024, 9, 2),**

**datetime(2024, 9, 3),**

**datetime(2024, 9, 3),**

**datetime(2024, 9, 4),**

**datetime(2024, 9, 4),**

**datetime(2024, 9, 5),**

**datetime(2024, 9, 5)**

**]**

**}**

**# Create a DataFrame**

**df = pd.DataFrame(data)**

**# Save the DataFrame to a CSV file**

**df.to\_csv('sales\_data.csv', index=False)**

**print("Sample sales dataset has been created and saved as 'sales\_data.csv'.")**

**```**

**2. \*\*Verify the Dataset:\*\***

**- After running the script, ensure that the file `sales\_data.csv` has been created in your working directory.**

**---**

**### \*\*Part 2: Load and Analyze the Dataset Using PySpark\*\***

**Now that you have the dataset, your task is to load it into PySpark and perform the following analysis tasks.**

**#### \*\*Step 2: Load the Dataset into PySpark\*\***

**1. \*\*Initialize the SparkSession:\*\***

**- Create a Spark session named `"Sales Dataset Analysis"`.**

**2. \*\*Load the CSV File into a PySpark DataFrame:\*\***

**- Load the `sales\_data.csv` file into a PySpark DataFrame.**

**- Display the first few rows of the DataFrame to verify that the data is loaded correctly.**

**#### \*\*Step 3: Explore the Data\*\***

**Explore the data to understand its structure.**

**1. \*\*Print the Schema:\*\***

**- Display the schema of the DataFrame to understand the data types.**

**2. \*\*Show the First Few Rows:\*\***

**- Display the first 5 rows of the DataFrame.**

**3. \*\*Get Summary Statistics:\*\***

**- Get summary statistics for numeric columns (`Quantity` and `Price`).**

**#### \*\*Step 4: Perform Data Transformations and Analysis\*\***

**Perform the following tasks to analyze the data:**

**1. \*\*Calculate the Total Sales Value for Each Transaction:\*\***

**- Add a new column called `TotalSales`, calculated by multiplying `Quantity` by `Price`.**

**2. \*\*Group By ProductID and Calculate Total Sales Per Product:\*\***

**- Group the data by `ProductID` and calculate the total sales for each product.**

**3. \*\*Identify the Top-Selling Product:\*\***

**- Find the product that generated the highest total sales.**

**4. \*\*Calculate the Total Sales by Date:\*\***

**- Group the data by `Date` and calculate the total sales for each day.**

**5. \*\*Filter High-Value Transactions:\*\***

**- Filter the transactions to show only those where the total sales value is greater than ₹500.**

**---**

**### \*\*Additional Challenge (Optional):\*\***

**If you complete the tasks above, try extending your analysis with the following challenges:**

**1. \*\*Identify Repeat Customers:\*\***

**- Count how many times each customer has made a purchase and display the customers who have made more than one purchase.**

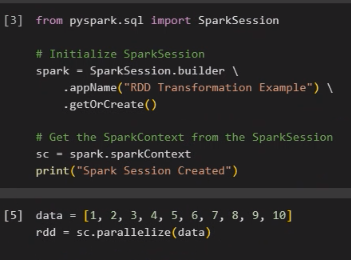
**2. \*\*Calculate the Average Sale Price Per Product:\*\***

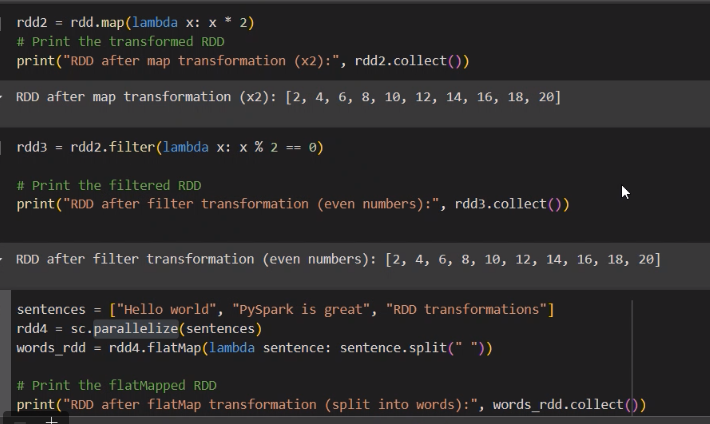
**- Calculate the average price per unit for each product and display the results.**

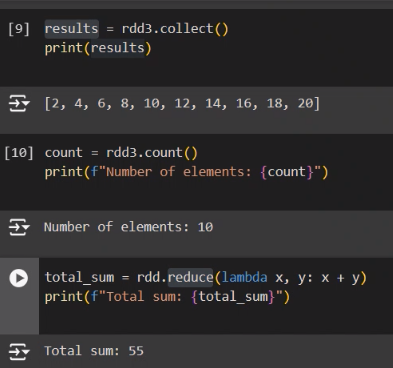
**ANSWERS**

<PySpark_HandsOn_Exercise_2.ipynb>

**RESILIENT DISTRIBUTED DATASET [RDD] :**



+



<RDD_basic_ex.ipynb>

-----------------------------------------------------------------------

**RDD HANDSON QUESTIONS AND ANSWERS**

### \*\*Exercise: Working with Key-Value Pair RDDs in PySpark\*\*

#### \*\*Objective:\*\*

In this exercise, you will work with key-value pair RDDs in PySpark. You will create RDDs, perform operations like grouping, aggregating, and sorting, and extract meaningful insights from the data.

---

### \*\*Dataset:\*\*

You will be working with the following sales data. Each entry in the dataset represents a product and its corresponding sales amount.

```python

sales\_data = [

("ProductA", 100),

("ProductB", 150),

("ProductA", 200),

("ProductC", 300),

("ProductB", 250),

("ProductC", 100)

]

```

You will also be working with an additional dataset for regional sales:

```python

regional\_sales\_data = [

("ProductA", 50),

("ProductC", 150)

]

```

---

### \*\*Step 1: Initialize Spark Context\*\*

1. \*\*Initialize SparkSession and SparkContext:\*\*

- Create a Spark session in PySpark and use the `spark.sparkContext` to create an RDD from the provided data.

---

### \*\*Step 2: Create and Explore the RDD\*\*

2. \*\*Task 1: Create an RDD from the Sales Data\*\*

- Create an RDD from the `sales\_data` list provided above.

- Print the first few elements of the RDD.

---

### \*\*Step 3: Grouping and Aggregating Data\*\*

3. \*\*Task 2: Group Data by Product Name\*\*

- Group the sales data by product name using `groupByKey()`.

- Print the grouped data to understand its structure.

4. \*\*Task 3: Calculate Total Sales by Product\*\*

- Use `reduceByKey()` to calculate the total sales for each product.

- Print the total sales for each product.

5. \*\*Task 4: Sort Products by Total Sales\*\*

- Sort the products by their total sales in descending order.

- Print the sorted list of products along with their sales amounts.

---

### \*\*Step 4: Additional Transformations\*\*

6. \*\*Task 5: Filter Products with High Sales\*\*

- Filter the products that have total sales greater than 200.

- Print the products that meet this condition.

7. \*\*Task 6: Combine Regional Sales Data\*\*

- Create another RDD from the `regional\_sales\_data` list.

- Combine this RDD with the original sales RDD using `union()`.

- Calculate the new total sales for each product after combining the datasets.

- Print the combined sales data.

---

### \*\*Step 5: Perform Actions on the RDD\*\*

8. \*\*Task 7: Count the Number of Distinct Products\*\*

- Count the number of distinct products in the RDD.

- Print the count of distinct products.

9. \*\*Task 8: Identify the Product with Maximum Sales\*\*

- Find the product with the maximum total sales using `reduce()`.

- Print the product name and its total sales amount.

---

### \*\*Challenge Task: Calculate the Average Sales per Product\*\*

10. \*\*Challenge Task:\*\*

- Calculate the average sales amount per product using the key-value pair RDD.

- Print the average sales for each product.

---

### \*\*Expected Outcomes\*\*

- \*\*Task 1:\*\* You should be able to create an RDD with key-value pairs and display its contents.

- \*\*Task 2:\*\* Group the data by key (product name) and explore the structure of grouped data.

- \*\*Task 3:\*\* Aggregate data to calculate total sales per product.

- \*\*Task 4:\*\* Sort the products by total sales and explore the results.

- \*\*Task 5:\*\* Filter the products based on a sales threshold.

- \*\*Task 6:\*\* Combine two RDDs and compute the new total sales for each product.

- \*\*Task 7:\*\* Count the number of distinct products in the dataset.

- \*\*Task 8:\*\* Identify the product with the highest sales.

- \*\*Challenge Task:\*\* Calculate and understand the average sales per product.

**Answer :**

[**RDD\_handson.ipynb**](RDD_handson.ipynb)