# PySpark Coding Challenge

-Sarthak Niranjan Kulkarni (Maverick)

- sarthakkul2311@gmail.com - (+91) 93256 02791

**26/11/2024 (Tuesday)**

## 1) Explain ETL (Extract, Transform, Load) with PySpark(in your own words):

→ **ETL:** ETL with PySpark means using Python and Apache Spark to extract data from different sources, clean or transform it as needed, and load it into a storage system like a database or data warehouse. It's a fast and efficient way to handle large datasets for analysis or reporting. It helps automate and streamline data workflows, making it easier to manage and process big data in real-time or batches.

**Extract:** Retrieve data from various sources like databases, files or APIs.

**Transform:** Clean, aggregate and manipulate data to fit your analysis needs.

**Load:** Store the transformed data into a database or data warehouse for analysis.

**Advantages of ETL: -**

- Data Centralization: Brings scattered data into a single, unified location.
- Improved Data Quality: Cleans and standardizes data, ensuring accuracy and consistency.
- Faster Decision-Making: Prepares data for quick analysis and insights.
- Time and Cost Efficiency: Automates repetitive data processing tasks, saving resources.
- Flexibility: Handles various data types and formats from multiple sources.
- Scalability: Easily adapts to growing data volumes.

**Example:** Extracting sales data from multiple CSV files, cleaning it to remove duplicates, and loading it into a database for reporting using PySpark.

---

## 2) Using Spark SQL - Transformations such as Filter, Join, Simple Aggregations, GroupBy on the case study dataset: -

**1. Filter: -**

→ # Filter customers older than 40

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("LoanData").getOrCreate()

filtered_df = df.filter(df['Age'] > 40)

filtered_df.show()

```
+------------+---------------+-----------------+
|Customer_ID|Age|Gender|         Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|     Loan Category|Loan Amount|Overdue
| Debt Record| Returned Cheque| Dishonour of Bill|
+-----------+---+------+-------------------+--------------+-----------+------+-----------+-------------+------------------+-----------+-------
+------------+---------------+-----------------+
|    IB14008| 44|  MALE|          PROFESSOR|       MARRIED|          6| 51000|      19999|            4|          SHOPPING|     50,000|      3
|     33,999|              1|                5|
|    IB14024| 55|FEMALE|              NURSE|       MARRIED|          6| 34999|      19888|            4|        AUTOMOBILE|     47,787|      1
|     50,000|              0|                3|
|    IB14027| 51|  MALE|     SYSTEM MANAGER|       MARRIED|          3| 49999|      19111|            5|       RESTAURANTS|     60,676|      8
|     13,000|              2|                5|
|    IB14037| 54|FEMALE|            TEACHER|       MARRIED|          5| 48099|      19999|            4|       RESTAURANTS|     30,999|      1
|     12,000|              7|                5|
|    IB14039| 45|  MALE|    ACCOUNT MANAGER|       MARRIED|          7| 45777|      18452|            4|         GOLD LOAN|  9,87,611 |      7
|     39,999|              8|                1|
|    IB14041| 59|FEMALE|ASSISTANT PROFESSOR|       MARRIED|          4| 50999|      22999|            5|  EDUCATIONAL LOAN|  5,99,934 |      3
|      9,000|              9|                9|
|    IB14049| 49|  MALE|       BANK MANAGER|       MARRIED|          4| 45999|      14500|            4|         TRAVELLING|     79,999|      4
|      6,700|              7|                3|
|    IB14050| 56|  MALE|     CIVIL ENGINEER|       MARRIED|          4|  null|      13999|            3|           HOUSING| 10,65,577 |      6
|     19,999|              4|                2|
```

→ # Filter married customers with Expenditure greater than 20,000

married_high_expenditure_sql_df = spark.sql("SELECT * FROM loan WHERE `Marital Status` = 'MARRIED' AND Expenditure > 20000")

married_high_expenditure_sql_df.show()

```
▸ married_high_expenditure_sql_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 13 more fields]
+------------+---------------+-----------------+
|Customer_ID|Age|Gender|         Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|     Loan Category|Loan Amount|Overdue
| Debt Record| Returned Cheque| Dishonour of Bill|
+-----------+---+------+-------------------+--------------+-----------+------+-----------+-------------+------------------+-----------+-------
+------------+---------------+-----------------+
|    IB14031| 37|FEMALE|  SOFTWARE ENGINEER|       MARRIED|          5| 55999|      23999|            5|        AUTOMOBILE|     60,999|      2
|          0|              5|                3|
|    IB14034| 32|  MALE|   PRODUCT ENGINEER|       MARRIED|          6|  null|      29000|            7|COMPUTER SOFTWARES|     80,660|      6
|      4,500|              5|                4|
|    IB14041| 59|FEMALE|ASSISTANT PROFESSOR|       MARRIED|          4| 50999|      22999|            5|  EDUCATIONAL LOAN|  5,99,934 |      3
|      9,000|              9|                9|
|    IB14054| 58|FEMALE|             DOCTOR|       MARRIED|          5| 60000|      25000|            5|           HOUSING|  9,00,000 |      5
|     21,000|              9|                0|
|    IB14082| 60|FEMALE|            TEACHER|       MARRIED|          5| 70000|      40000|            9|         GOLD LOAN|  2,57,789 |      4
|     10,058|              4|                3|
|    IB14092| 47|  MALE|    SYSTEM ENGINEER|       MARRIED|          4| 52364|      45612|            3|         GOLD LOAN|  6,54,725 |      4
|     67,451|              5|                4|
|    1B14094| 49|  MALE|ASSISTANT PROFESSOR|       MARRIED|          5| 65214|      42589|            5|           HOUSING|  9,85,412 |      5
|     11,254|              1|                2|
|    IB14099| 47|FEMALE|             DOCTOR|       MARRIED|          4| 72154|      45286|            4|        AUTOMOBILE|  7,54,126 |      2
|     19,524|              5|                2|
```

**2. Joins: -** (Performed with Case study Datasets (Loan and Credit_card))

**→ Types Of Joins: -**

- **Inner Join:**

# Load the loan and credit tables as DataFrames

loan_df = spark.table("loan")

credit_df = spark.table("credit")

# Inner join between loan and credit tables on Customer_ID

inner_join_df = loan_df.join(credit_df, loan_df.Customer_ID == credit_df.CustomerId, 'inner')

inner_join_df.show()

```
▸ (2) Spark Jobs
▸ loan_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 13 more fields]
▸ credit_df: pyspark.sql.dataframe.DataFrame = [RowNumber: string, CustomerId: string ... 11 more fields]
▸ inner_join_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 26 more fields]
+-----------+---+------+----------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+------------+---
------------+------------------+---------+----------+-------+-----------+---------+------+---+------+-------+-------------+--------------+------
--------+------+
|Customer_ID|Age|Gender|Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|Loan Category|Loan Amount|Overdue| Debt Record| Re
turned Cheque| Dishonour of Bill|RowNumber|CustomerId|Surname|CreditScore|Geography|Gender|Age|Tenure|Balance|NumOfProducts|IsActiveMember|Estima
tedSalary|Exited|
+-----------+---+------+----------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+------------+---
------------+------------------+---------+----------+-------+-----------+---------+------+---+------+-------+-------------+--------------+------
--------+------+
+-----------+---+------+----------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+------------+---
------------+------------------+---------+----------+-------+-----------+---------+------+---+------+-------+-------------+--------------+------
--------+------+
```

- **Left Join: -**

# Left join between loan and credit tables on Customer_ID

left_join_df = loan_df.join(credit_df, loan_df.Customer_ID == credit_df.CustomerId, 'left')

left_join_df.show()

```
▸ left_join_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 26 more fields]
null|           null|   null|
|    IB14042| 25|FEMALE|          DOCTOR|         SINGLE|          4| 60111|      27111|            5|       TRAVELLING| 12,90,929 |      4
|      18,000|                 1|                 0|      null|       null|    null|        null|       null|  null|null|  null|    null|          null|
null|           null|   null|
|    IB14045| 31|  MALE|    STORE KEEPER|         SINGLE|          5| 40999|      11999|            3|      BOOK STORES|  1,67,654 |      1
|       4,500|                 0|                 1|      null|       null|    null|        null|       null|  null|null|  null|    null|          null|
null|           null|   null|
|    IB14049| 49|  MALE|    BANK MANAGER|        MARRIED|          4| 45999|      14500|            4|       TRAVELLING|     79,999|      4
|       6,700|                 7|                 3|      null|       null|    null|        null|       null|  null|null|  null|    null|          null|
null|           null|   null|
|    IB14050| 56|  MALE|  CIVIL ENGINEER|        MARRIED|          4|  null|      13999|            3|          HOUSING| 10,65,577 |      6
|      19,999|                 4|                 2|      null|       null|    null|        null|       null|  null|null|  null|    null|          null|
null|           null|   null|
|    IB14054| 58|FEMALE|          DOCTOR|        MARRIED|          5| 60000|      25000|            5|          HOUSING|  9,00,000 |      5
|      21,000|                 9|                 0|      null|       null|    null|        null|       null|  null|null|  null|    null|          null|
null|           null|   null|
```

- **Right Join: -**

# Right join between loan and credit tables on Customer_ID

right_join_df = loan_df.join(credit_df, loan_df.Customer_ID == credit_df.CustomerId, 'right')

right_join_df.show()

```
right_join_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 26 more fields]
1|       65951.65|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|       16|  15643966|   Goforth|        616|  Germany|  Male| 45|      3|143129.41|            2|
1|       64327.26|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|       17|  15737452|     Romeo|        653|  Germany|  Male| 58|      1|132602.88|            1|
0|        5097.67|     1|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|       18|  15788218|Henderson|        549|    Spain|Female| 24|      9|        0|            2|
1|       14406.41|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|       19|  15661507|  Muldrow|        587|    Spain|  Male| 45|      6|        0|            1|
0|      158684.81|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|       20|  15568982|      Hao|        726|   France|Female| 24|      6|        0|            2|
1|       54724.03|     0|
+----------+----+-----+---------+------------+----------+-----+----------+-----------+-----------+----------+------+-----------
```

- **Outer Join: -**

# Outer join between loan and credit tables on Customer_ID

outer_join_df = loan_df.join(credit_df, loan_df.Customer_ID == credit_df.CustomerId, 'outer')

outer_join_df.show()

```
outer_join_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: string ... 26 more fields]
0|       151083.8|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|     1463|  15566211|       Hsu|        616|  Germany|Female| 41|      1|103560.57|            1|
0|         236.45|     1|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|     1141|  15566251|   Ferrari|        618|   France|Female| 37|      5| 96652.86|            1|
0|        98686.4|     1|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|     5425|  15566253|   Manning|        580|  Germany|  Male| 44|      9|143391.07|            1|
0|      146891.07|     1|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|     5345|  15566269|  Chialuka|        787|   France|  Male| 25|      5|        0|            2|
0|        47307.9|     0|
|        null|null|  null|      null|          null|        null|  null|        null|          null|          null|        null|   null|         null
|            null|              null|      704|  15566292|Okwuadigbo|        574|    Spain|  Male| 36|      1|        0|            2|
1|       71709.12|     0|
+----------+----+-----+---------+------------+----------+-----+----------+-----------+-----------+----------+------+-----------
```

**3. Simple Aggregate Functions: -**

→ # Count the number of records in the loan table using PySpark

count_records = spark.sql("SELECT COUNT(*) AS total_records FROM loan")

count_records.show()

# Average Income Amount in the loan table using PySpark

average_income = spark.sql("""

SELECT AVG(`Income`) AS avg_income

FROM loan

""")

average_income.show()

```
▸ (4) Spark Jobs
▸ count_records:  pyspark.sql.dataframe.DataFrame = [total_records: long]
▸ average_income:  pyspark.sql.dataframe.DataFrame = [avg_income: double]
+-------------+
|total_records|
+-------------+
|          500|
+-------------+

+-----------------+
|       avg_income|
+-----------------+
|68339.49145299145|
+-----------------+
```

**4. Group By: -**

→ # Group by Marital Status and calculate the total Expenditure using PySpark

```
group_by_marital_status = spark.sql("""
    SELECT `Marital Status`, SUM(Expenditure) AS total_expenditure
    FROM loan
    GROUP BY `Marital Status`
""")
group_by_marital_status.show()
# Group by Loan Category and calculate the average Expenditure using PySpark
group_by_loan_category = spark.sql("""
    SELECT `Loan Category`, AVG(Expenditure) AS avg_expenditure
    FROM loan
    GROUP BY `Loan Category`
""")
group_by_loan_category.show()
```

```
group_by_marital_status:  pyspark.sql.dataframe.DataFrame = [Marital Status: string, total_expenditure: double]
group_by_loan_category:  pyspark.sql.dataframe.DataFrame = [Loan Category: string, avg_expenditure: double]
+--------------+-----------------+
|Marital Status|total_expenditure|
+--------------+-----------------+
|        SINGLE|        3986776.0|
|       MARRIED|        9256684.0|
+--------------+-----------------+

+------------------+------------------+
|     Loan Category|   avg_expenditure|
+------------------+------------------+
|           HOUSING|29052.666666666668|
|        TRAVELLING|         26211.125|
|       BOOK STORES|           21221.0|
|       AGRICULTURE|           30573.5|
|         GOLD LOAN| 26168.61842105263|
|  EDUCATIONAL LOAN|           31088.6|
|        AUTOMOBILE|26787.660714285714|
|          BUSINESS|           31431.0|
|COMPUTER SOFTWARES|26157.363636363636|
|           DINNING|27934.285714285714|
```