Coding Challenge – 3

PySpark & SparkSql

Name: Aathirainathan P
Date: 26-11-2024

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

ETL (Extract, Transform, Load) is a fundamental process in data engineering and analytics that preparing data for analysis. The **Extract** phase involves gathering raw data from various source systems, which could be relational databases, files (like CSV, JSON) and is accomplished using methods like spark.read() for structured data formats, such as CSV. The main objective here is to extract large volumes of data efficiently from various sources and prepare them for further processing.

Once the data is extracted, the **Transform** phase begins. This is where PySpark's allows for large-scale data processing in a distributed manner. In the transformation step, the data is cleaned, enriched, and reshaped to fit the needs of analysis. Common transformations include filtering rows based on specific criteria (.filter()), changing column types (.cast()), handling missing or null values (.fillna()), and applying aggregation or summarization (.groupBy()). This step also includes joining datasets, applying business rules, and performing data enrichment to enhance the value of the data. After transformation, the data is typically structured in a way that is optimized for analytics.

The final stage is **Load**, where the transformed data is loaded into a storage system, such as a relational database, a data lake, or a data warehouse, for future use. This step ensures that the data is available for downstream users or systems that need to access it for reporting, analysis, or further processing. Overall, ETL pipelines in PySpark enable the handling of massive datasets with scalability, speed, and reliability, which is essential in today's data-driven world.

TASK 2: Using SparkSql and PySpark - Transformations such as Filter, Join, Simple Aggregations, GroupBy on the case study dataset.

I. Loading data:

▶ ⌄ ✓ 04:14 PM (3s) 1 Python

```python
#Loading Data
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Case Study").getOrCreate()

loan_file_path = "/FileStore/tables/loan.csv"

loan_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load(loan_file_path)

loan_df.show()
loan_df.createOrReplaceTempView("loan_data")
```

▸ (3) Spark Jobs

▾ loan_df: pyspark.sql.dataframe.DataFrame

```
Customer_ID: string
Age: integer
Gender: string
Occupation: string
Marital Status: string
Family Size: integer
Income: integer
Expenditure: integer
Expenditure: integer
Use Frequency: integer
Loan Category: string
Loan Amount: string
Overdue: integer
Debt Record: string
Returned Cheque: integer
Dishonour of Bill: integer
```

```
4,500|             5|                4|
|   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|
|   IB14042| 25|FEMALE|              DOCTOR|       SINGLE|          4| 60111|     27111|            5|        TRAVELLING| 12,90,929 |       4|
18,000|            1|                0|
|   IB14045| 31|  MALE|         STORE KEEPER|       SINGLE|          5| 40999|     11999|            3|       BOOK STORES|  1,67,654 |       1|
4,500|             0|                1|
|   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|
|   IB14054| 58|FEMALE|              DOCTOR|      MARRIED|          5| 60000|     25000|            5|           HOUSING|  9,00,000 |       5|
21,000|            9|                0|
+----------+---+------+-------------------+-------------+-----------+------+----------+-------------+------------------+-----------+-------+-
----------+---------------+-----------------+
only showing top 20 rows
```

II. Use Spark Sql:

1. Filter records based on conditions (e.g., Find people with Loan Amount greater than 10,00,000**):**

04:23 PM (1s) 2 Python

```python
#Spark SQL
#Filter
#Find people with Loan Amount greater than 10,00,000
from pyspark.sql import functions as F

loan_df = loan_df.withColumn("Loan Amount", F.regexp_replace("Loan Amount", ",", "").cast("double"))

loan_df.createOrReplaceTempView("loan_data")

filtered_data = spark.sql("""
    SELECT * FROM loan_data
    WHERE `Loan Amount` > 10000
""")
filtered_data.show()
```

(1) Spark Jobs

loan_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 13 more fields]

filtered_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 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|
+-----------+---+------+-----------------+--------------+-----------+------+-----------+-------------+-----------------+-----------+-------+
-----------+---------------+-----------------+
|    IB14001| 30|  MALE|     BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|          HOUSING|  1000000.0|      5|
42,898|              6|                9|
|    IB14008| 44|  MALE|        PROFESSOR|       MARRIED|          6| 51000|      19999|            4|         SHOPPING|    50000.0|      3|
33,999|              1|                5|
|    IB14012| 30|FEMALE|          DENTIST|        SINGLE|          3| 58450|      27675|            5|       TRAVELLING|    75000.0|      6|
20,876|              3|                1|
|    IB14018| 29|  MALE|          TEACHER|       MARRIED|          5| 45767|      12787|            3|        GOLD LOAN|   600000.0|      7|
11,000|              0|                4|
|    IB14022| 34|  MALE|           POLICE|        SINGLE|          4| 43521|      11999|            3|       AUTOMOBILE|   200000.0|      2|
43,898|              1|                2|
|    IB14024| 55|FEMALE|            NURSE|       MARRIED|          6| 34999|      19888|            4|       AUTOMOBILE|    47787.0|      1|
50,000|              0|                3|
|    IB14025| 39|FEMALE|          TEACHER|       MARRIED|          6| 46619|      18675|            4|          HOUSING|  1209867.0|      8|
29,999|              6|                8|
|    IB14027| 51|  MALE|   SYSTEM MANAGER|       MARRIED|          3| 49999|      19111|            5|      RESTAURANTS|    60676.0|      8|
```

2. Joins with another DataFrame (for demonstration, I created another DataFrame called loan_approval_df**):**

2.1 Inner join:

▶ ⌄ ✓ 04:27 PM (2s) 3 Python

```python
#Spark SQL
#inner join

# Sample loan approval data (For demonstration of joins)
loan_approval_data = [
    ('IB14001', 'Approved'),
    ('IB14008', 'Denied'),
    ('IB14012', 'Approved'),
    ('IB14018', 'Approved'),
    ('IB14022', 'Denied'),
    ('IB14024', 'Approved'),
]

#DataFrame for loan approval data
loan_approval_columns = ['Customer_ID', 'Approval_Status']
loan_approval_df = spark.createDataFrame(loan_approval_data, loan_approval_columns)

loan_approval_df.createOrReplaceTempView("loan_approval_data")

# Perform an inner join between loan_data and loan_approval_data
joined_data = spark.sql("""
    SELECT a.*, b.Approval_Status
    FROM loan_data a
    JOIN loan_approval_data b
    ON a.Customer_ID = b.Customer_ID
""")
joined_data.show()
```

▸ (4) Spark Jobs

▸ loan_approval_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Approval_Status: string]

▸ joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 14 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|Approval_Status|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
---------------+-----------------+---------------+
|    IB14001| 30|  MALE|BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|      HOUSING|  1000000.0|      5|     42,898|
6|              9|         Approved|
|    IB14008| 44|  MALE|   PROFESSOR|       MARRIED|          6| 51000|      19999|            4|     SHOPPING|    50000.0|      3|     33,999|
1|              5|           Denied|
|    IB14012| 30|FEMALE|     DENTIST|        SINGLE|          3| 58450|      27675|            5|   TRAVELLING|    75000.0|      6|     20,876|
3|              1|         Approved|
|    IB14018| 29|  MALE|     TEACHER|       MARRIED|          5| 45767|      12787|            3|     GOLD LOAN|   600000.0|      7|     11,000|
0|              4|         Approved|
|    IB14022| 34|  MALE|      POLICE|        SINGLE|          4| 43521|      11999|            3|   AUTOMOBILE|   200000.0|      2|     43,898|
1|              2|           Denied|
|    IB14024| 55|FEMALE|       NURSE|       MARRIED|          6| 34999|      19888|            4|   AUTOMOBILE|    47787.0|      1|     50,000|
0|              3|         Approved|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
---------------+-----------------+---------------+
```

2.2 left join:

04:28 PM (2s) 4 Python

```python
# Perform a left join between loan_data and loan_approval_data
left_joined_data = spark.sql("""
    SELECT a.*, b.Approval_Status
    FROM loan_data a
    LEFT JOIN loan_approval_data b
    ON a.Customer_ID = b.Customer_ID
""")
left_joined_data.show()
```

(3) Spark Jobs

left_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 14 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|Approval_Status|
+-----------+---+------+-----------------+--------------+-----------+------+-----------+-------------+-----------------+-----------+-------+-
----------+---------------+------------------+---------------+
|    IB14001| 30|  MALE|     BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|          HOUSING|  1000000.0|      5|
42,898|              6|                 9|       Approved|
|    IB14008| 44|  MALE|        PROFESSOR|       MARRIED|          6| 51000|      19999|            4|         SHOPPING|    50000.0|      3|
33,999|              1|                 5|         Denied|
|    IB14012| 30|FEMALE|          DENTIST|        SINGLE|          3| 58450|      27675|            5|       TRAVELLING|    75000.0|      6|
20,876|              3|                 1|       Approved|
|    IB14018| 29|  MALE|          TEACHER|       MARRIED|          5| 45767|      12787|            3|        GOLD LOAN|   600000.0|      7|
11,000|              0|                 4|       Approved|
|    IB14022| 34|  MALE|           POLICE|        SINGLE|          4| 43521|      11999|            3|       AUTOMOBILE|   200000.0|      2|
43,898|              1|                 2|         Denied|
```

2.3 right join:

04:29 PM (1s) 5

```python
# Perform a right join between loan_data and loan_approval_data
right_joined_data = spark.sql("""
    SELECT a.*, b.Approval_Status
    FROM loan_data a
    RIGHT JOIN loan_approval_data b
    ON a.Customer_ID = b.Customer_ID
""")
right_joined_data.show()
```

(4) Spark Jobs

right_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 14 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|Approval_Status|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
---------------+-----------------+---------------+
|    IB14001| 30|  MALE|BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|      HOUSING|  1000000.0|      5|     42,898|
6|              9|         Approved|
|    IB14008| 44|  MALE|   PROFESSOR|       MARRIED|          6| 51000|      19999|            4|     SHOPPING|    50000.0|      3|     33,999|
1|              5|           Denied|
|    IB14012| 30|FEMALE|     DENTIST|        SINGLE|          3| 58450|      27675|            5|   TRAVELLING|    75000.0|      6|     20,876|
3|              1|         Approved|
|    IB14018| 29|  MALE|     TEACHER|       MARRIED|          5| 45767|      12787|            3|    GOLD LOAN|   600000.0|      7|     11,000|
0|              4|         Approved|
|    IB14022| 34|  MALE|      POLICE|        SINGLE|          4| 43521|      11999|            3|   AUTOMOBILE|   200000.0|      2|     43,898|
1|              2|           Denied|
|    IB14024| 55|FEMALE|       NURSE|       MARRIED|          6| 34999|      19888|            4|   AUTOMOBILE|    47787.0|      1|     50,000|
```

2.4 outer join:

04:29 PM (2s) 6 Python

```python
# Perform a full outer join between loan_data and loan_approval_data
outer_joined_data = spark.sql("""
    SELECT a.*, b.Approval_Status
    FROM loan_data a
    FULL OUTER JOIN loan_approval_data b
    ON a.Customer_ID = b.Customer_ID
""")
outer_joined_data.show()
```

(3) Spark Jobs

outer_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 14 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|Approval_Status|
+-----------+---+------+------------------+--------------+-----------+------+-----------+---------------+------------------+-----------+-------+-
----------+---------------+-----------------+--------------+
|    1B14093| 21|FEMALE|           MANAGER|        SINGLE|          3| 42516|      24567|            7|        AUTOMOBILE|  2569874.0|      8|
89,652|              2|                3|          NULL|
|    1B14094| 49|  MALE|ASSISTANT PROFESSOR|       MARRIED|          5| 65214|      42589|            5|           HOUSING|   985412.0|      5|
11,254|              1|                2|          NULL|
|    1B14312| 21|FEMALE|           MANAGER|        SINGLE|          3| 42516|      24567|            7| EDUCATIONAL LOAN|  2569874.0|      8|
89,652|              2|                3|          NULL|
|    1B14315| 49|  MALE|ASSISTANT PROFESSOR|       MARRIED|          5| 65214|      42589|            5|           HOUSING|   985412.0|      5|
11,254|              1|                2|          NULL|
|    1B14001| 30|  MALE|      BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|           HOUSING|  1000000.0|      5|
42,898|              6|                9|      Approved|
```

3. Simple Aggregations (e.g., average loan amount per occupation**):**

04:29 PM (2s) 7 Python

```python
# Aggregation: Average Loan Amount per Occupation
avg_loan_per_occupation = spark.sql("""
    SELECT Occupation, AVG(`Loan Amount`) AS avg_loan
    FROM loan_data
    GROUP BY Occupation
""")
avg_loan_per_occupation.show()
```

(2) Spark Jobs

avg_loan_per_occupation: pyspark.sql.dataframe.DataFrame = [Occupation: string, avg_loan: double]

```
+--------------------+-----------------+
|          Occupation|         avg_loan|
+--------------------+-----------------+
|      CIVIL ENGINEER| 819806.3333333334|
|     FIRE DEPARTMENT| 955125.1666666666|
|          ACCOUNTANT|1223623.2857142857|
|        BANK MANAGER| 629305.6071428572|
|      SYSTEM OFFICER|          290192.0|
|           NUTRITION|          456780.0|
|           DIETICIAN| 625974.4615384615|
|               CLERK| 633292.7307692308|
|   SOFTWARE ENGINEER|          755663.0|
|AGRICULTURAL ENGI...|          767338.0|
|   ASSISTANT MANAGER|          729638.5|
|             TEACHER| 681778.6349206349|
```

4. GroupBy and Aggregation (e.g., total income per marital status):

04:30 PM (1s) 8

```
# GroupBy: Total Income by Marital Status
total_income_per_status = spark.sql("""
    SELECT `Marital Status`, SUM(Income) AS total_income
    FROM loan_data
    GROUP BY `Marital Status`
""")
total_income_per_status.show()
```

(2) Spark Jobs

total_income_per_status: pyspark.sql.dataframe.DataFrame = [Marital Status: string, total_income: long]

```
+--------------+------------+
|Marital Status|total_income|
+--------------+------------+
|        SINGLE|     8756569|
|       MARRIED|    23226313|
+--------------+------------+
```

5. Filter and Aggregate (e.g., filter by 'SINGLE' marital status and calculate total loan amount):

04:30 PM (1s) 9

```
# Filter and Aggregate: Total Loan Amount for SINGLE marital status
single_marital_status = spark.sql("""
    SELECT SUM(`Loan Amount`) AS total_loan
    FROM loan_data
    WHERE `Marital Status` = 'SINGLE'
""")
single_marital_status.show()
```

(2) Spark Jobs

single_marital_status: pyspark.sql.dataframe.DataFrame = [total_loan: double]

```
+------------+
|  total_loan|
+------------+
|1.12118685E8|
+------------+
```

III. Use PySpark:

1. Filter records based on conditions (e.g., Find people with Loan Amount greater than 10,00,000**):**

04:31 PM (1s) 10 Python

```
#PySpark
# Filter: Loan Amount greater than 1,000,000
filtered_df = loan_df.filter(loan_df['Loan Amount'] > 1000000)
filtered_df.show()
```

(1) Spark Jobs

filtered_df: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 13 more fields]

```
+-----------+---+------+----------------+--------------+-----------+------+-----------+-------------+------------------+-----------+-------+---
---------+---------------+------------------+
|Customer_ID|Age|Gender|      Occupation|Marital Status|Family Size|Income|Expenditure|Use Frequency|     Loan Category|Loan Amount|Overdue| De
bt Record| Returned Cheque| Dishonour of Bill|
+-----------+---+------+----------------+--------------+-----------+------+-----------+-------------+------------------+-----------+-------+---
---------+---------------+------------------+
|    IB14025| 39|FEMALE|         TEACHER|       MARRIED|          6| 46619|      18675|            4|           HOUSING|  1209867.0|      8|
29,999|              6|                 8|
|    IB14042| 25|FEMALE|          DOCTOR|        SINGLE|          4| 60111|      27111|            5|        TRAVELLING|  1290929.0|      4|
18,000|              1|                 0|
|    IB14050| 56|  MALE|  CIVIL ENGINEER|       MARRIED|          4|  NULL|      13999|            3|           HOUSING|  1065577.0|      6|
19,999|              4|                 2|
|    IB14089| 25|  MALE|       PROFESSOR|        SINGLE|          5| 62145|      31254|            4|       BOOK STORES|  1245789.0|      6|
48,596|              6|                 5|
|    IB14093| 21|FEMALE|         MANAGER|        SINGLE|          3| 42516|      24567|            7|        AUTOMOBILE|  2569874.0|      8|
89,652|              2|                 3|
|    IB14096| 33|FEMALE|           CLERK|       MARRIED|          3| 35684|      15247|            3|       RESTAURANTS|  1452637.0|      3|
13,547|              3|                 2|
|    IB14104| 54|  MALE| AIRPORT OFFICER|       MARRIED|          6| 80000|      32541|            2|        AUTOMOBILE|  2045789.0|      1|
```

2. Joins:

2.1 inner join:

04:33 PM (2s) 11

```python
#inner join
inner_joined_data = loan_df.join(loan_approval_df, loan_df.Customer_ID == loan_approval_df.Customer_ID, "inner")

# Show the result
inner_joined_data.show()
```

(4) Spark Jobs

inner_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 15 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|Customer_ID|Approval_Status|
+-----------+---+------+------------+-------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
---------------+-----------------+-----------+--------------+
|    IB14001| 30|  MALE|BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|      HOUSING|  1000000.0|      5|     42,898|
6|                9|    IB14001|      Approved|
|    IB14008| 44|  MALE|   PROFESSOR|       MARRIED|          6| 51000|      19999|            4|     SHOPPING|    50000.0|      3|     33,999|
1|                5|    IB14008|        Denied|
|    IB14012| 30|FEMALE|     DENTIST|        SINGLE|          3| 58450|      27675|            5|   TRAVELLING|    75000.0|      6|     20,876|
3|                1|    IB14012|      Approved|
|    IB14018| 29|  MALE|     TEACHER|       MARRIED|          5| 45767|      12787|            3|    GOLD LOAN|   600000.0|      7|     11,000|
0|                4|    IB14018|      Approved|
|    IB14022| 34|  MALE|      POLICE|        SINGLE|          4| 43521|      11999|            3|   AUTOMOBILE|   200000.0|      2|     43,898|
1|                2|    IB14022|        Denied|
|    IB14024| 55|FEMALE|       NURSE|       MARRIED|          6| 34999|      19888|            4|   AUTOMOBILE|    47787.0|      1|     50,000|
0|                3|    IB14024|      Approved|
+-----------+---+------+------------+-------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
```

2.2 left join:

04:34 PM (1s) 12

```
#left join
left_joined_data = loan_df.join(loan_approval_df, loan_df.Customer_ID == loan_approval_df.Customer_ID, "left")

# Show the result
left_joined_data.show()
```

(2) Spark Jobs

left_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 15 more fields]

```
4,500|              5|               4|      NULL|           NULL|
|    IB14037| 54|FEMALE|          TEACHER|       MARRIED|          5| 48099|      19999|             4|     RESTAURANTS|     30999.0|      1|
12,000|             7|               5|      NULL|           NULL|
|    IB14039| 45|  MALE|  ACCOUNT MANAGER|       MARRIED|          7| 45777|      18452|             4|       GOLD LOAN|    987611.0|      7|
39,999|             8|               1|      NULL|           NULL|
|    IB14041| 59|FEMALE|ASSISTANT PROFESSOR|     MARRIED|          4| 50999|      22999|             5|EDUCATIONAL LOAN|    599934.0|      3|
9,000|              9|               9|      NULL|           NULL|
|    IB14042| 25|FEMALE|           DOCTOR|        SINGLE|          4| 60111|      27111|             5|      TRAVELLING|   1290929.0|      4|
18,000|             1|               0|      NULL|           NULL|
|    IB14045| 31|  MALE|     STORE KEEPER|        SINGLE|          5| 40999|      11999|             3|     BOOK STORES|    167654.0|      1|
4,500|              0|               1|      NULL|           NULL|
|    IB14049| 49|  MALE|     BANK MANAGER|       MARRIED|          4| 45999|      14500|             4|      TRAVELLING|     79999.0|      4|
6,700|              7|               3|      NULL|           NULL|
|    IB14050| 56|  MALE|   CIVIL ENGINEER|       MARRIED|          4|  NULL|      13999|             3|         HOUSING|   1065577.0|      6|
19,999|             4|               2|      NULL|           NULL|
|    IB14054| 58|FEMALE|           DOCTOR|       MARRIED|          5| 60000|      25000|             5|         HOUSING|    900000.0|      5|
21,000|             9|               0|      NULL|           NULL|
+-----------+---+------+-------------------+--------------+-----------+------+-----------+--------------+----------------+------------+-------+
```

2.3 right join:

04:34 PM (2s) 13

```
#right join
right_joined_data = loan_df.join(loan_approval_df, loan_df.Customer_ID == loan_approval_df.Customer_ID, "right")

# Show the result
right_joined_data.show()
```

(4) Spark Jobs

right_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 15 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|Customer_ID|Approval_Status|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
----------------+-----------------+-----------+---------------+
|    IB14001| 30|  MALE|BANK MANAGER|        SINGLE|          4| 50000|      22199|            6|      HOUSING|  1000000.0|      5|     42,898|
6|               9|    IB14001|       Approved|
|    IB14008| 44|  MALE|   PROFESSOR|       MARRIED|          6| 51000|      19999|            4|     SHOPPING|    50000.0|      3|     33,999|
1|               5|    IB14008|         Denied|
|    IB14012| 30|FEMALE|     DENTIST|        SINGLE|          3| 58450|      27675|            5|   TRAVELLING|    75000.0|      6|     20,876|
3|               1|    IB14012|       Approved|
|    IB14018| 29|  MALE|     TEACHER|       MARRIED|          5| 45767|      12787|            3|    GOLD LOAN|   600000.0|      7|     11,000|
0|               4|    IB14018|       Approved|
|    IB14022| 34|  MALE|      POLICE|        SINGLE|          4| 43521|      11999|            3|   AUTOMOBILE|   200000.0|      2|     43,898|
1|               2|    IB14022|         Denied|
|    IB14024| 55|FEMALE|       NURSE|       MARRIED|          6| 34999|      19888|            4|   AUTOMOBILE|    47787.0|      1|     50,000|
0|               3|    IB14024|       Approved|
+-----------+---+------+------------+--------------+-----------+------+-----------+-------------+-------------+-----------+-------+-----------+
```

2.4 outer join:

▶ ✓ 04:35 PM (1s) 14

```
#outer join
outer_joined_data = loan_df.join(loan_approval_df, loan_df.Customer_ID == loan_approval_df.Customer_ID, "outer")

# Show the result
outer_joined_data.show()
```

▸ (3) Spark Jobs

▸ outer_joined_data: pyspark.sql.dataframe.DataFrame = [Customer_ID: string, Age: integer ... 15 more fields]

```
13,000|              2|                 5|         NULL|           NULL|
|    IB14029| 24|FEMALE|           TEACHER|       SINGLE|           3| 45008|      17454|             4|         AUTOMOBILE|    399435.0|       9|
51,987|              4|                 7|         NULL|           NULL|
|    IB14031| 37|FEMALE| SOFTWARE ENGINEER|      MARRIED|           5| 55999|      23999|             5|         AUTOMOBILE|     60999.0|       2|
0|              5|                 3|         NULL|           NULL|
|    IB14032| 24|  MALE|      DATA ANALYST|       SINGLE|           4| 60111|      28999|             6|         AUTOMOBILE|     35232.0|       5|
33,333|              1|                 2|         NULL|           NULL|
|    IB14034| 32|  MALE|  PRODUCT ENGINEER|      MARRIED|           6|  NULL|      29000|             7|COMPUTER SOFTWARES|     80660.0|       6|
4,500|              5|                 4|         NULL|           NULL|
|    IB14037| 54|FEMALE|           TEACHER|      MARRIED|           5| 48099|      19999|             4|        RESTAURANTS|     30999.0|       1|
12,000|              7|                 5|         NULL|           NULL|
|    IB14039| 45|  MALE|   ACCOUNT MANAGER|      MARRIED|           7| 45777|      18452|             4|          GOLD LOAN|    987611.0|       7|
39,999|              8|                 1|         NULL|           NULL|
|    IB14041| 59|FEMALE|ASSISTANT PROFESSOR|     MARRIED|           4| 50999|      22999|             5|     EDUCATIONAL LOAN|    599934.0|       3|
9,000|              9|                 9|         NULL|           NULL|
|    IB14042| 25|FEMALE|            DOCTOR|       SINGLE|           4| 60111|      27111|             5|          TRAVELLING|   1290929.0|       4|
18,000|              1|                 0|         NULL|           NULL|
+-----------+---+------+------------------+-------------+-----------+------+-----------+--------------+------------------+------------+--------+-
----------+-----------------+------------------+-----------+---------------+
only showing top 20 rows
```

3. **Simple Aggregations (**e.g., Aggregating Total Loan Amount, Average Income, and Counting Customers**):**

▶ ✓ 04:39 PM (1s) 15

```
# Simple aggregations
# Aggregating Total Loan Amount, Average Income, and Counting Customers
from pyspark.sql import functions as F

aggregated_data = loan_df.agg(
    F.sum("Loan Amount").alias("Total_Loan_Amount"),
    F.avg("Income").alias("Average_Income"),
    F.count("Customer_ID").alias("Customer_Count"),
    F.min("Income").alias("Min_Income"),
    F.max("Income").alias("Max_Income")
)

aggregated_data.show()
```

▸ (2) Spark Jobs

▸ aggregated_data: pyspark.sql.dataframe.DataFrame = [Total_Loan_Amount: double, Average_Income: double ... 3 more fields]

```
+-----------------+-----------------+--------------+----------+----------+
|Total_Loan_Amount|   Average_Income|Customer_Count|Min_Income|Max_Income|
+-----------------+-----------------+--------------+----------+----------+
|     3.98526449E8|68339.49145299145|           500|     28366|    930000|
+-----------------+-----------------+--------------+----------+----------+
```

4. GroupBy and Aggregation (e.g., Grouping by Occupation and Calculating Total Loan Amount and Average Income**):**

▶ ✓ 04:40 PM (1s) 16

```
# Grouping and aggregations
# Calculating Total Loan Amount and Average Income
grouped_data = loan_df.groupBy("Occupation").agg(
    F.sum("Loan Amount").alias("Total_Loan_Amount"),
    F.avg("Income").alias("Average_Income"),
    F.count("Customer_ID").alias("Customer_Count")
)

grouped_data.show()
```

▸ (2) Spark Jobs

▸ grouped_data: pyspark.sql.dataframe.DataFrame = [Occupation: string, Total_Loan_Amount: double ... 2 more fields]

```
+--------------------+-----------------+-----------------+--------------+
|          Occupation|Total_Loan_Amount|    Average_Income|Customer_Count|
+--------------------+-----------------+-----------------+--------------+
|      CIVIL ENGINEER|        4918838.0|60359.666666666664|             6|
|     FIRE DEPARTMENT|      1.1461502E7|55357.916666666664|            12|
|          ACCOUNTANT|        8565363.0| 56623.28571428572|             7|
|        BANK MANAGER|      1.7620557E7|          92191.0|            28|
|      SYSTEM OFFICER|        1160768.0|          56780.0|             4|
|           NUTRITION|         456780.0|          55650.0|             1|
|           DIETICIAN|        8137668.0| 72599.16666666667|            13|
|               CLERK|      1.6465611E7|        76871.125|            26|
|   SOFTWARE ENGINEER|      2.6448205E7|          61107.8|            35|
|AGRICULTURAL ENGI...|        6138704.0|        82060.625|             8|
|   ASSISTANT MANAGER|        4377831.0|54866.166666666664|             6|
|             TEACHER|      4.2952054E7| 52812.73333333333|            63|
| ASSISTANT PROFESSOR|        5197463.0|53319.333333333336|             9|
```

5. Filter and Aggregate (e.g., filter by 'SINGLE' marital status and calculate total loan amount):

▶ ✓ 04:40 PM (1s) 17

```
# Filtering and  aggregation
# Filtering Customers with Income > 50,000 and Calculating Total Loan Amount and Average Expenditure
filtered_aggregated_data = loan_df.filter(loan_df.Income > 50000).agg(
    F.sum("Loan Amount").alias("Total_Loan_Amount"),
    F.avg("Expenditure").alias("Average_Expenditure")
)

filtered_aggregated_data.show()
```

▸ (2) Spark Jobs

▸ filtered_aggregated_data: pyspark.sql.dataframe.DataFrame = [Total_Loan_Amount: double, Average_Expenditure: double]

```
+-----------------+-------------------+
|Total_Loan_Amount|Average_Expenditure|
+-----------------+-------------------+
|     2.61067242E8| 30574.736263736264|
+-----------------+-------------------+
```

Submitted by:
Aathirainathan P