# PySpark – Coding Challenge

**Submitted By-**

**Subrat Shukla, DE-1**

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

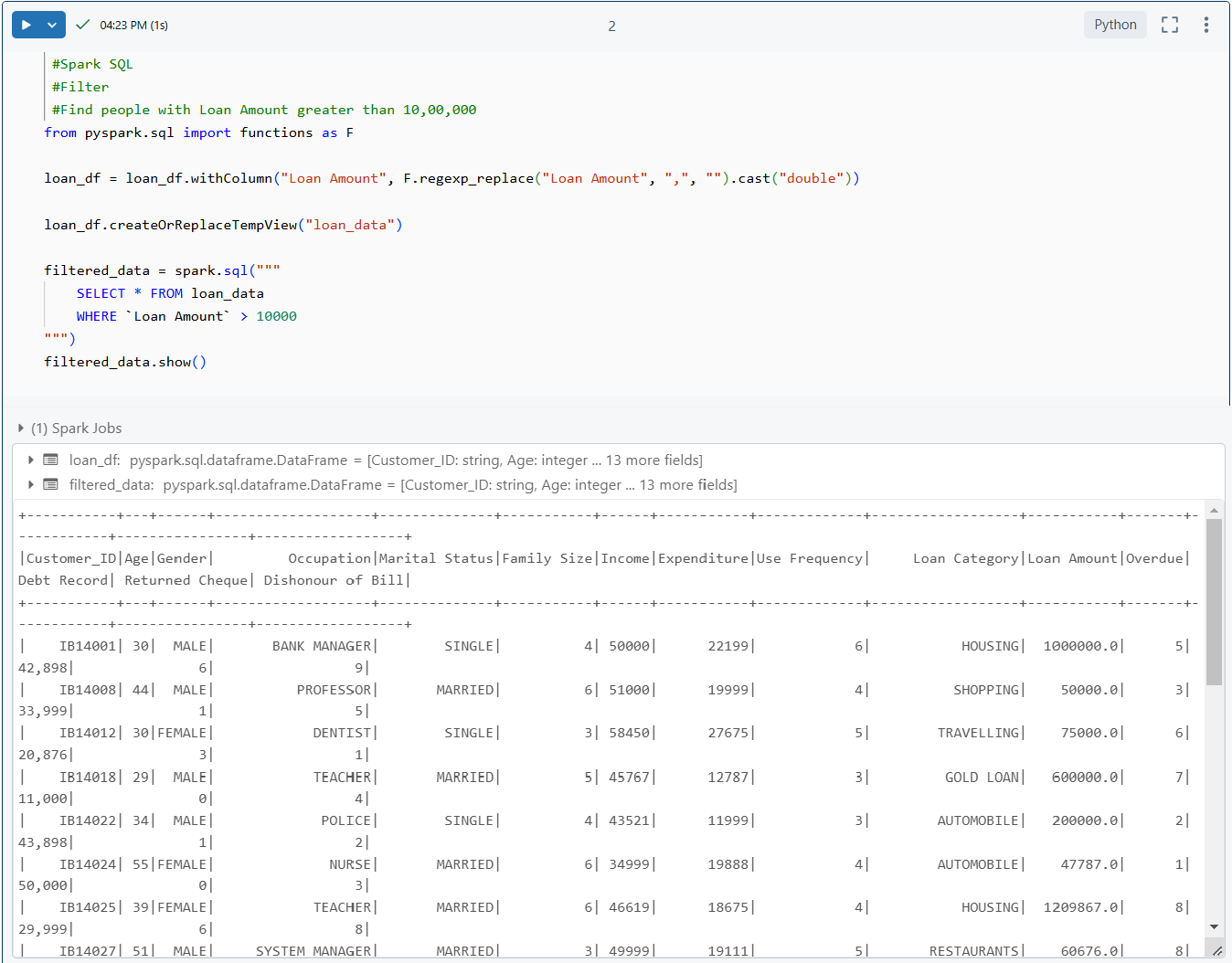
ETL (Extract, Transform, Load) is a fundamental process in data engineering and analytics that preparing data for analysis. The **Extract** phase involves gathering raw data from various source systems, which could be relational databases, files (like CSV, JSON) and is accomplished using methods like 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.

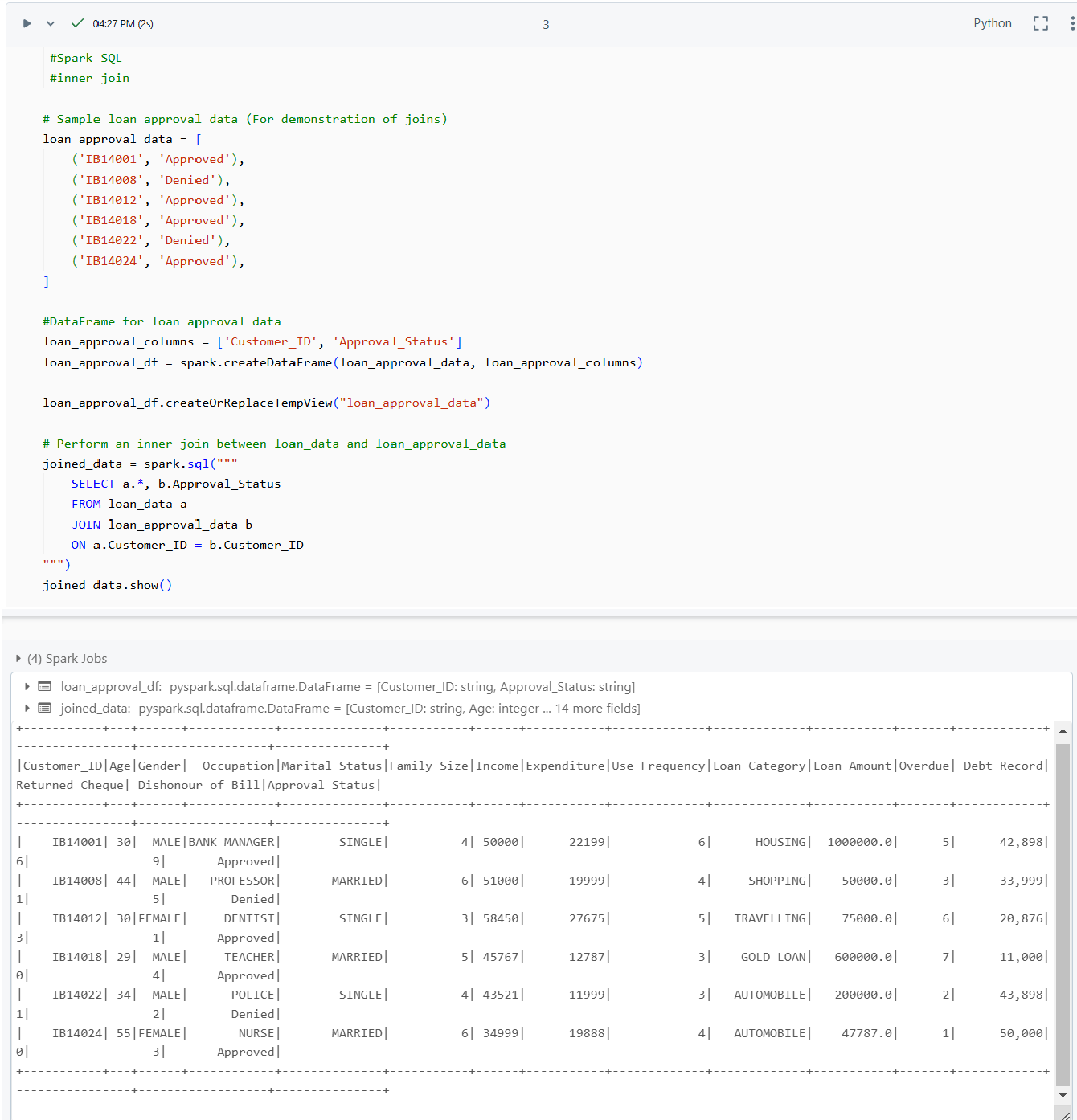
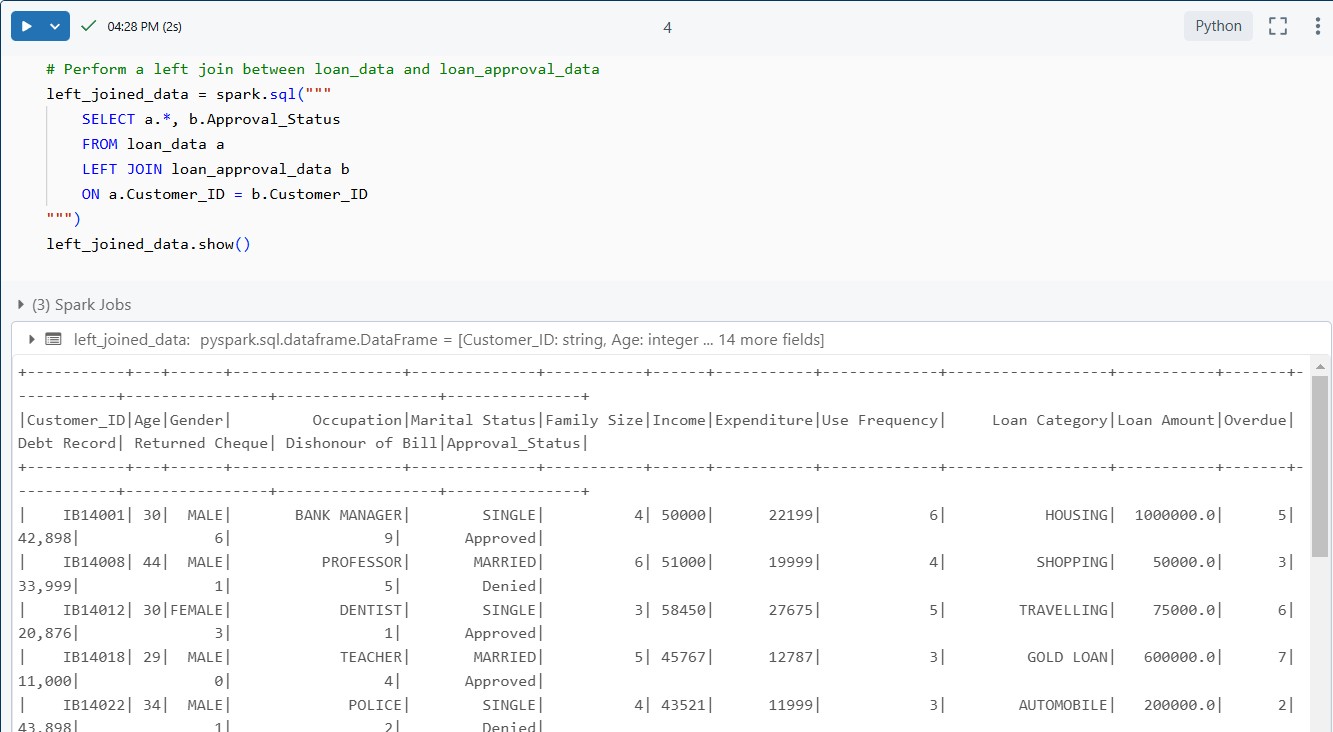
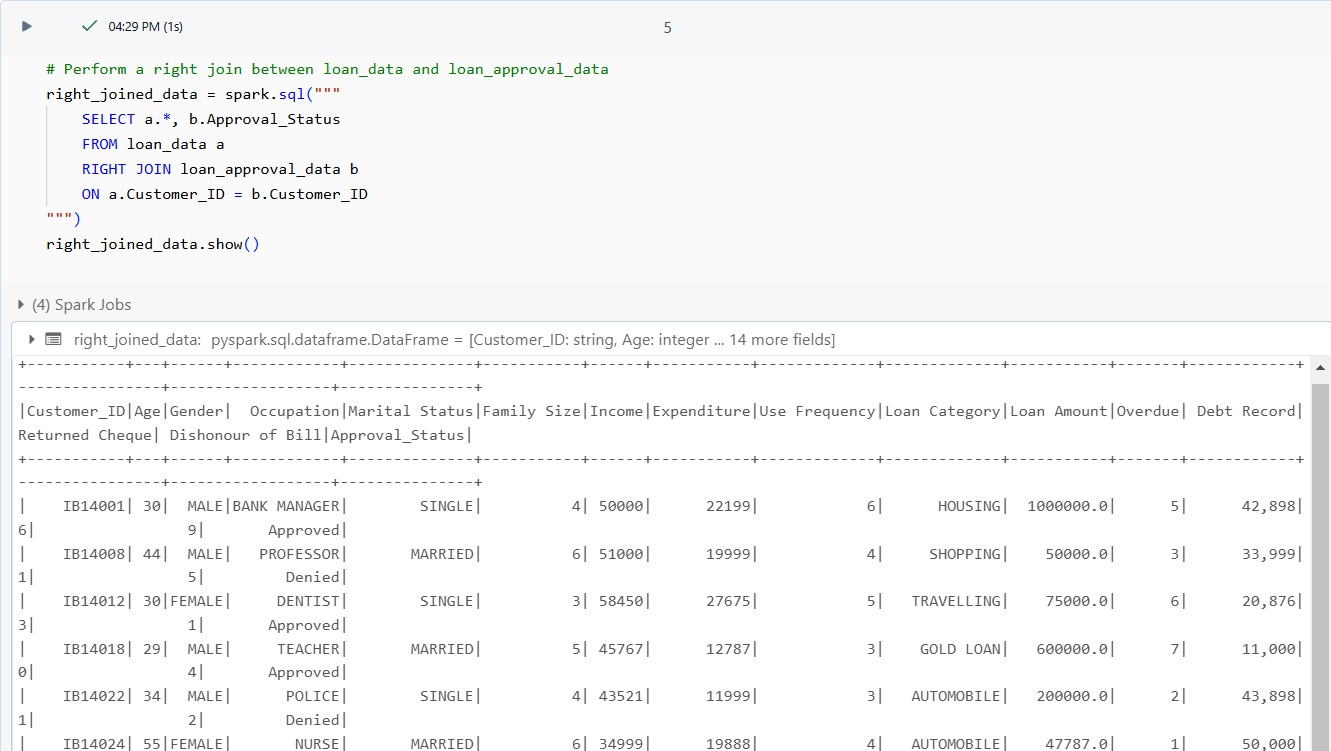
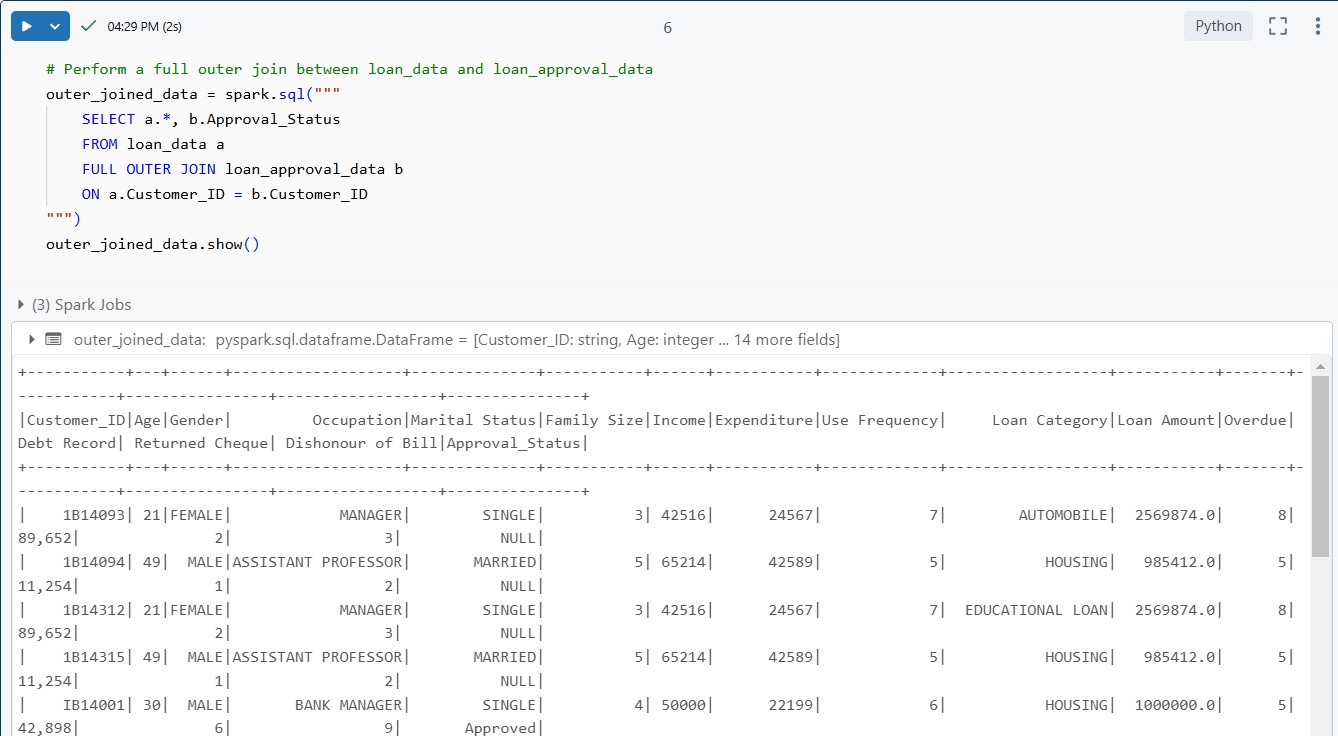
Once the data is extracted, the **Transform** phase begins. This is where PySpark’s allows for large- scale data processing in a distributed manner. In the transformation step, the data is cleaned, enriched, and reshaped to fit the needs of analysis. Common transformations include filtering rows based 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.

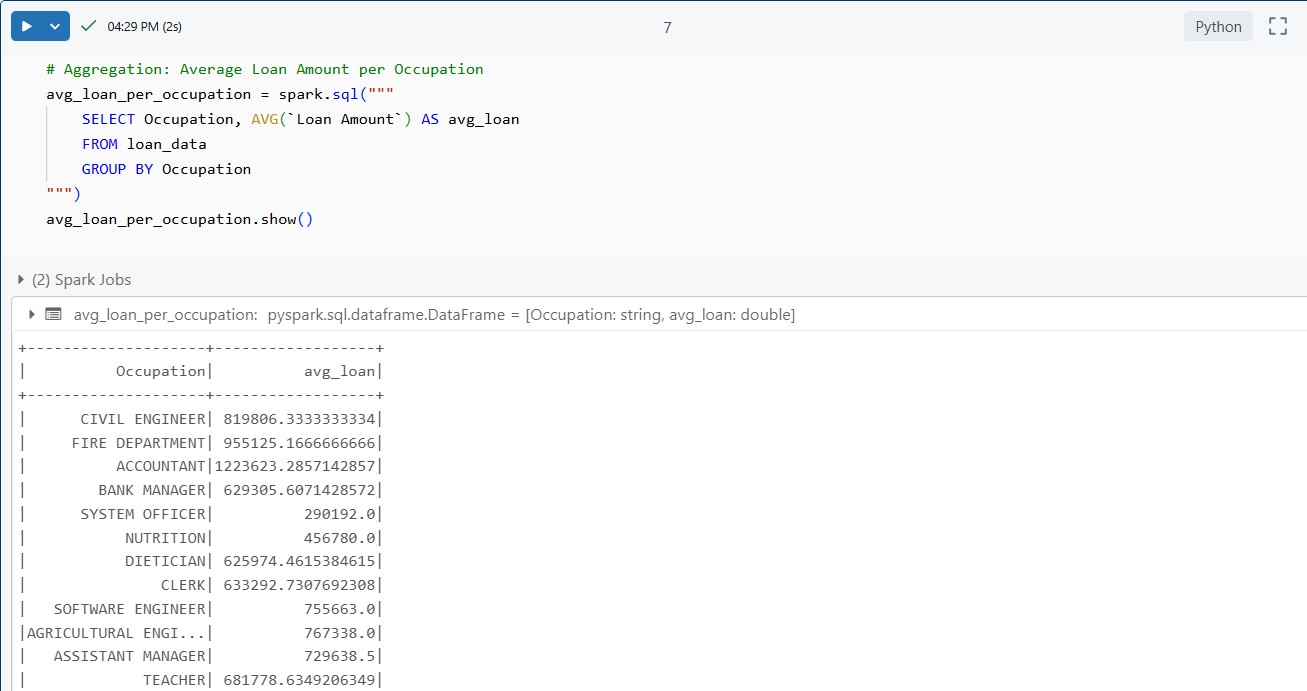
The final stage is **Load**, where the transformed data is loaded into a storage system, such as a relational database, a data lake, or a data warehouse, for future use. This step ensures that the data is available for downstream users or systems that need to access it for reporting, analysis, or further 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.

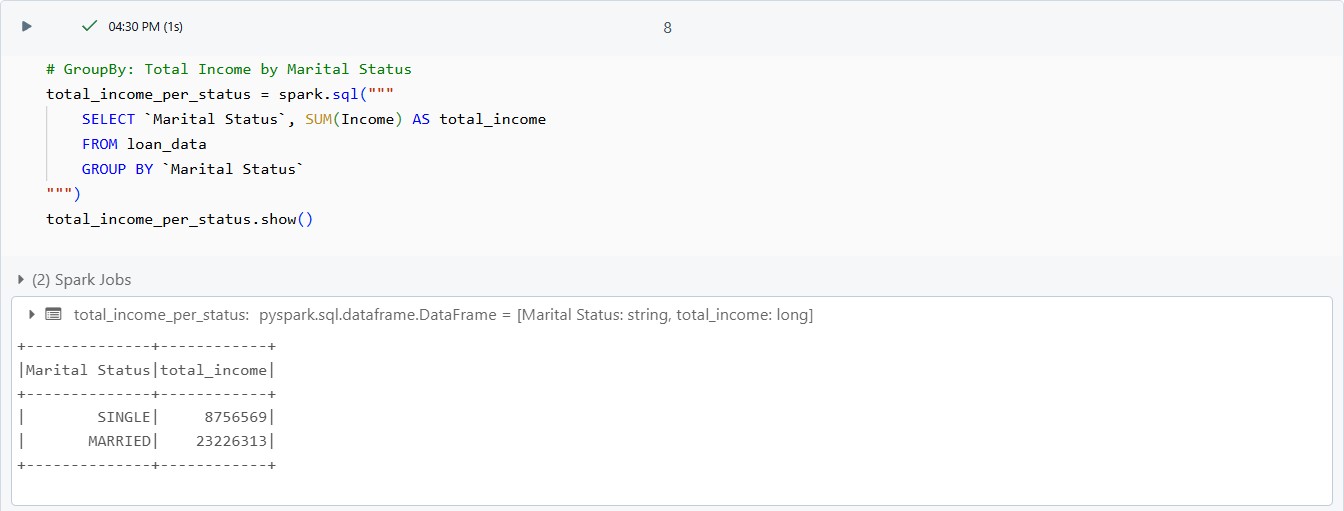
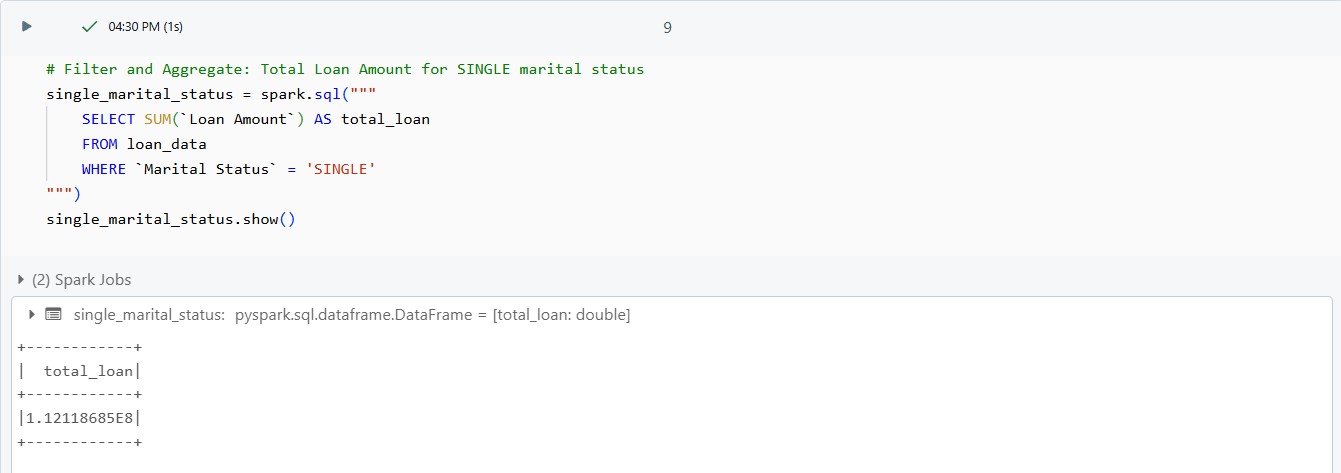
# TASK 2: Using SparkSql and PySpark - Transformations such as Filter, Join, Simple Aggregations, GroupBy on the case study dataset.

## Loading data:

* **Use Spark Sql:**
* **Filter records based on conditions (**e.g., Find people with Loan Amount greater than 10,00,000**):**
* **Joins with another DataFrame (**for demonstration,I created another Dataframe called loan\_approval\_df**):**

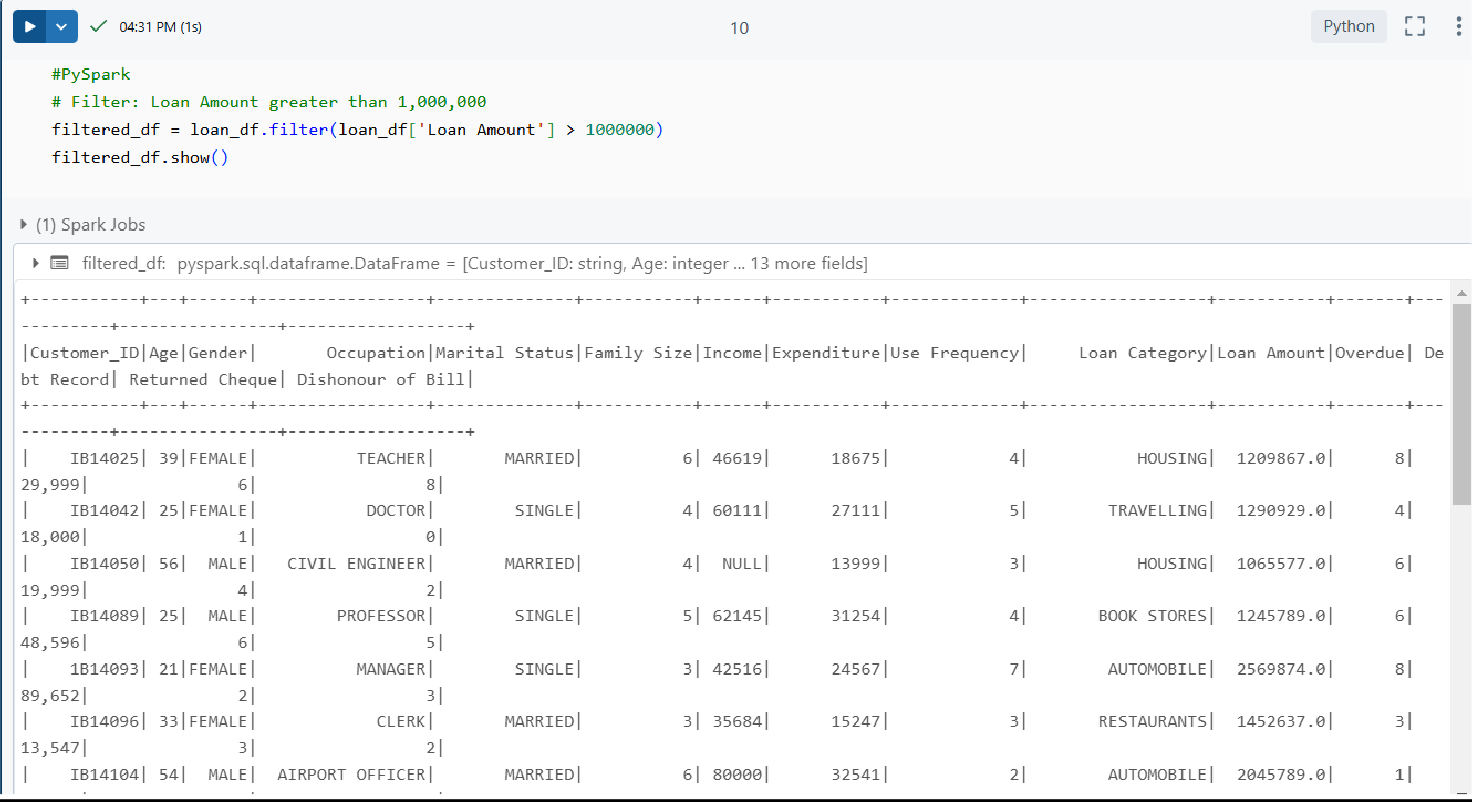
1. **Inner join:**
2. **Left join:**
3. **Right join:**
4. **Outer join:**

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

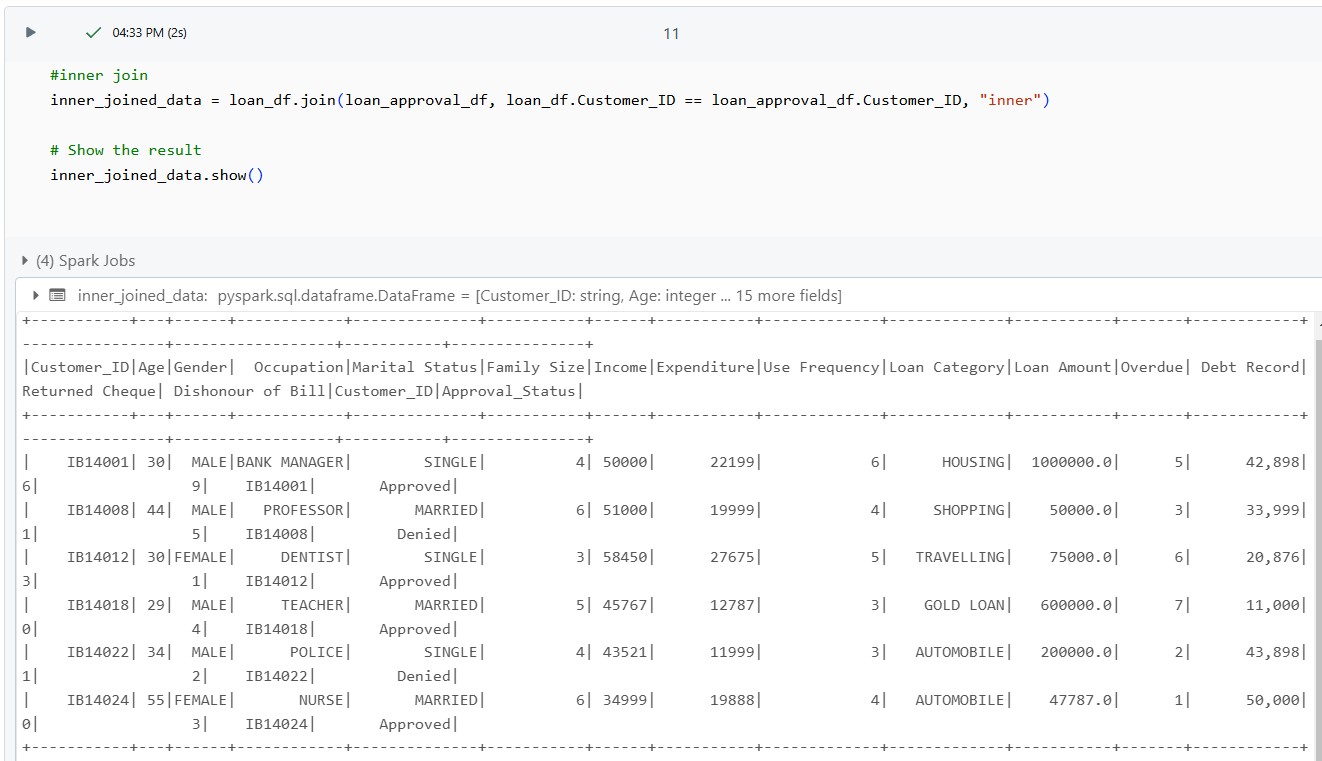
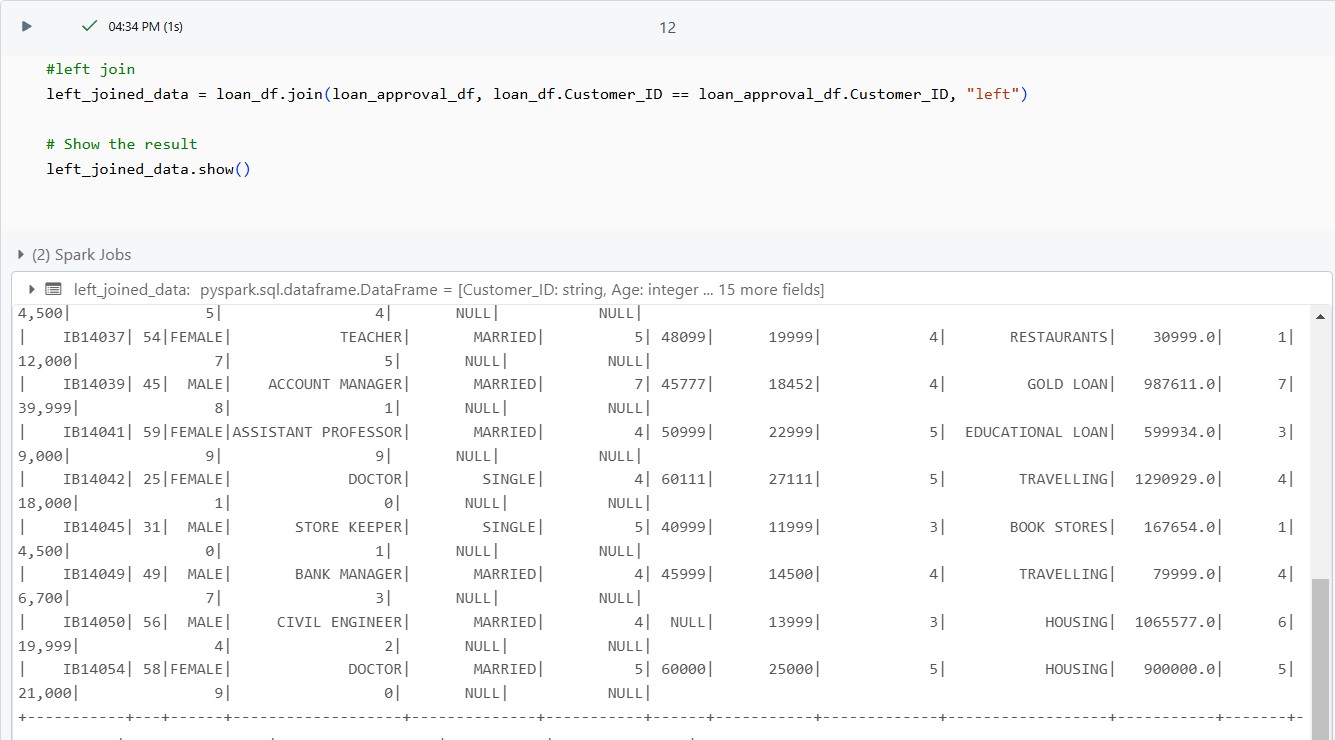
1. **GroupBy and Aggregation** (e.g., total income per marital status):
2. **Filter and Aggregate** (e.g., filter by 'SINGLE' marital status and calculate total loan amount):

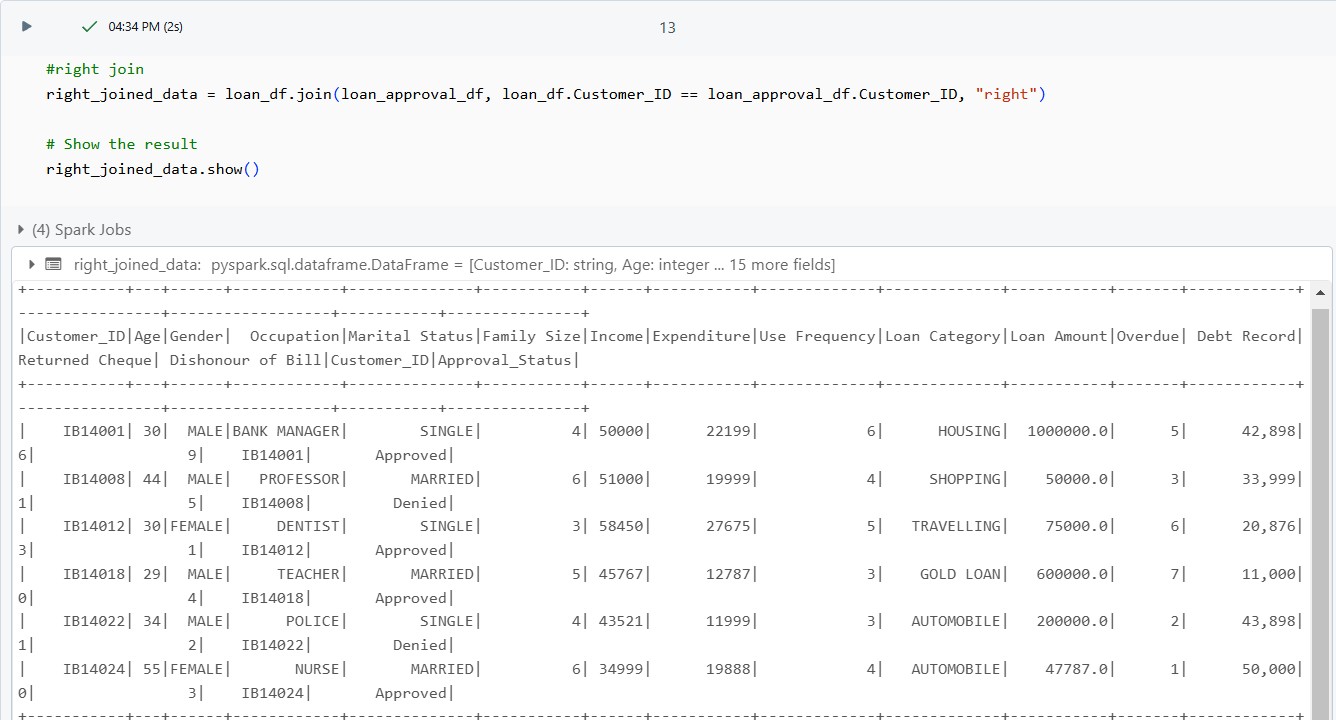
## Use PySpark:

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



* **Joins:**

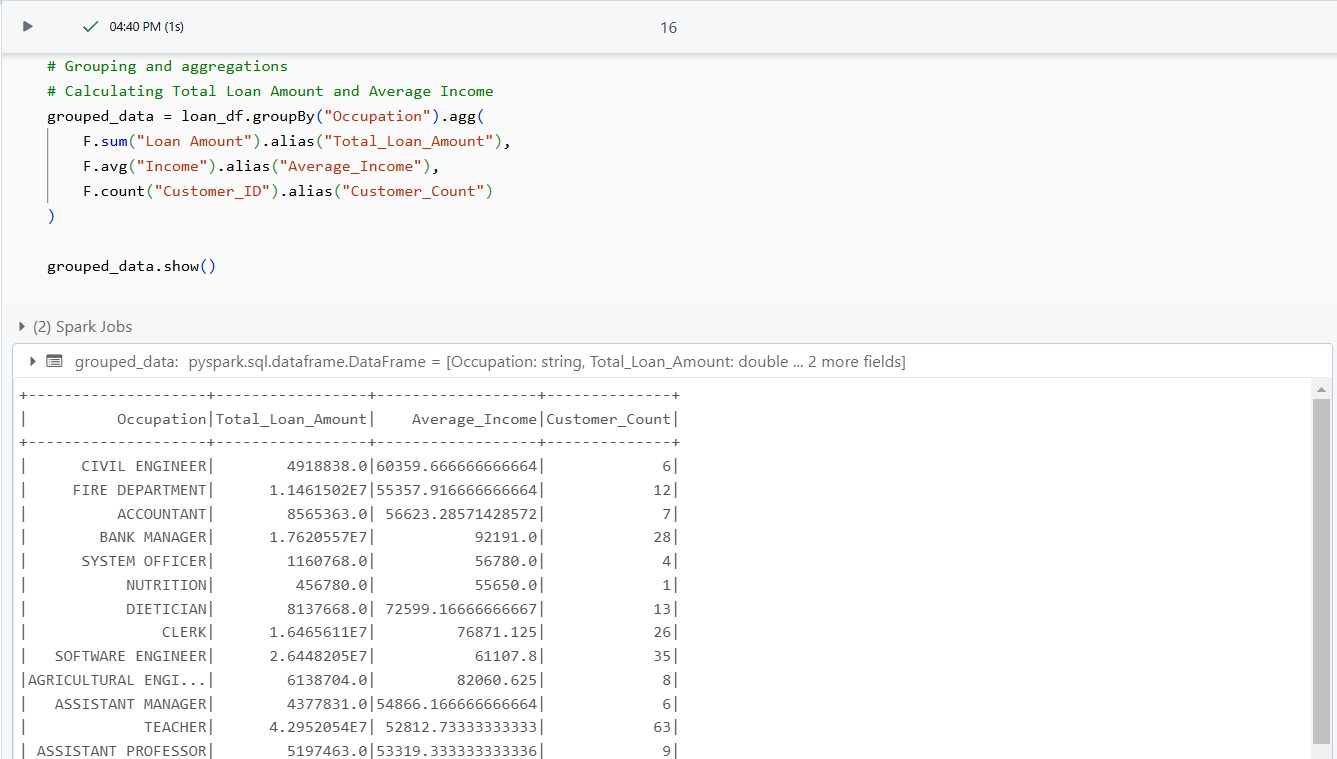
1. **Inner join:**
2. **Left join:**
3. **Right join:**



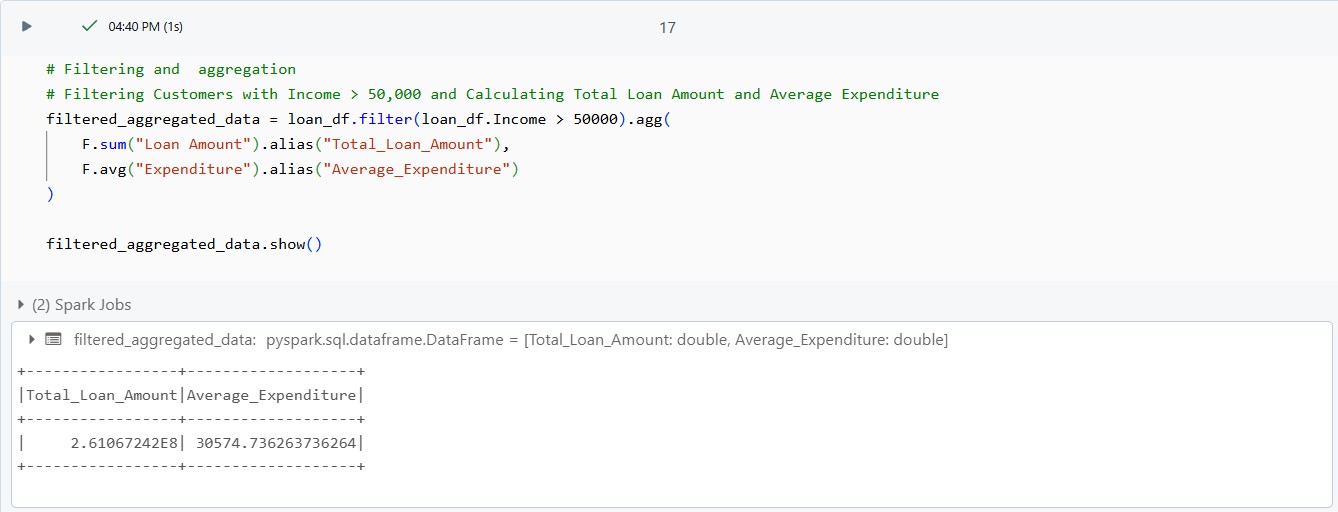
1. **Outer join:**

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

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



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



**--Thank You!**