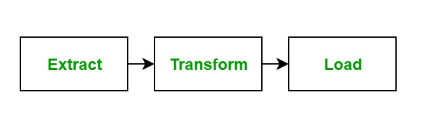
**Spark SQL and PySpark Coding Challenge**

**Name: S.Yazhini**

**Date: 26-11-2024**

**Question 1:**

**Explain ETL (Extract, Transform, Load) with PySpark(in your own words)**

* **Starting with a real time example:**

**ServiceNow Tickets:**

Imagine HR wants to know how much company has spent in this year.

HR (client) will raise a ticket onServiceNow.

**Extract:**

You will first extract the data and clean it.

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

**Transform:**

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

As per the client requirement you will transform and load - power BI visualization or Azure visualization or table data as it is.

**Load:**

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

And then load it (Similar to Publish in databricks). So that you can share the link with anybody (to hr also).

* **Summary of ETL:**

ETL (Extract, Transform, Load) is a data integration process used to collect data from various sources, transform it into a consistent and usable format, and load it into a target system, such as a database or data warehouse. The Extract phase gathers raw data from sources like databases, APIs, or flat files. In the Transform phase, the data is cleaned, standardized, and formatted to meet business or analytical requirements. Finally, the Load phase moves the processed data into the target system for storage and analysis. ETL is essential for ensuring data quality and consistency in analytics and reporting.

**Question 2:**

**Using PySpark and Spark SQL - Transformations such as Filter, Join, Simple Aggregations, GroupBy on the case study dataset**

**Dataset used from case study:**

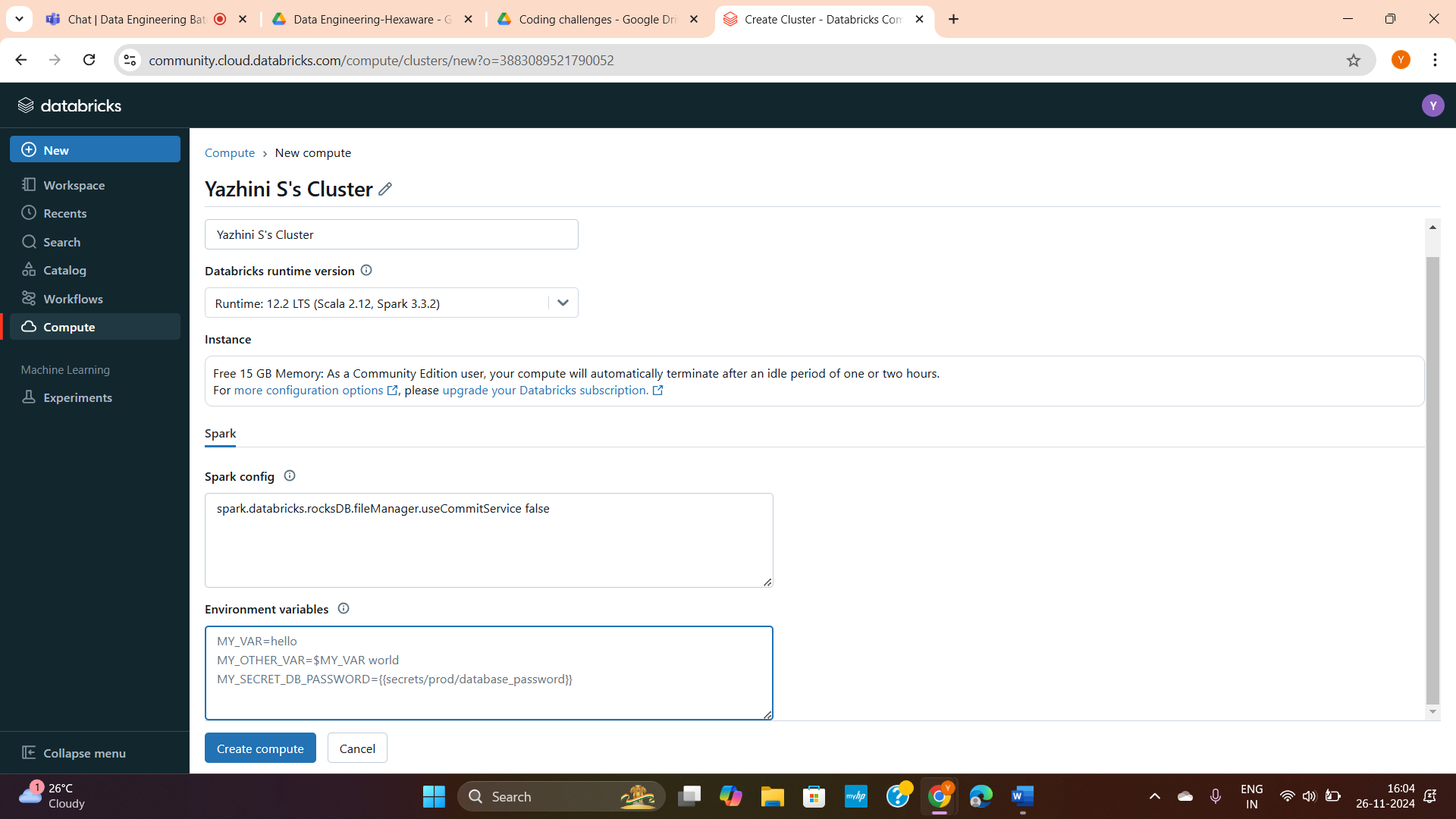
* Loan.csv

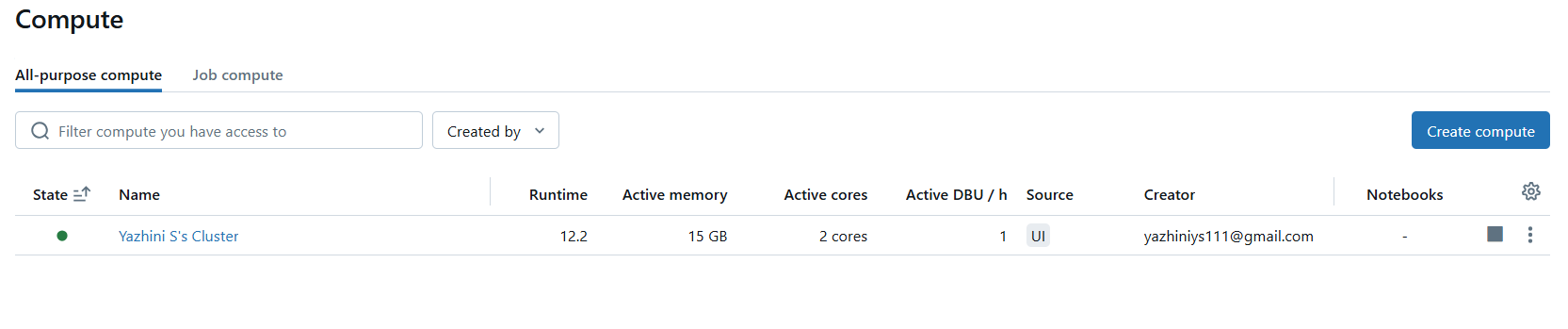
<https://drive.google.com/file/d/1mT00UK41eSsfy__DK3hXtYJ9iNBRLr7-/view?usp=drive_link>

* credit\_card.csv (Only for Joins)

<https://drive.google.com/file/d/1p0HyqEgUal22EWzfNcccFRwJbtBQlsfv/view?usp=drive_link>

**Cluster created images in DataBricks:**



****

**Initialising Spark:**

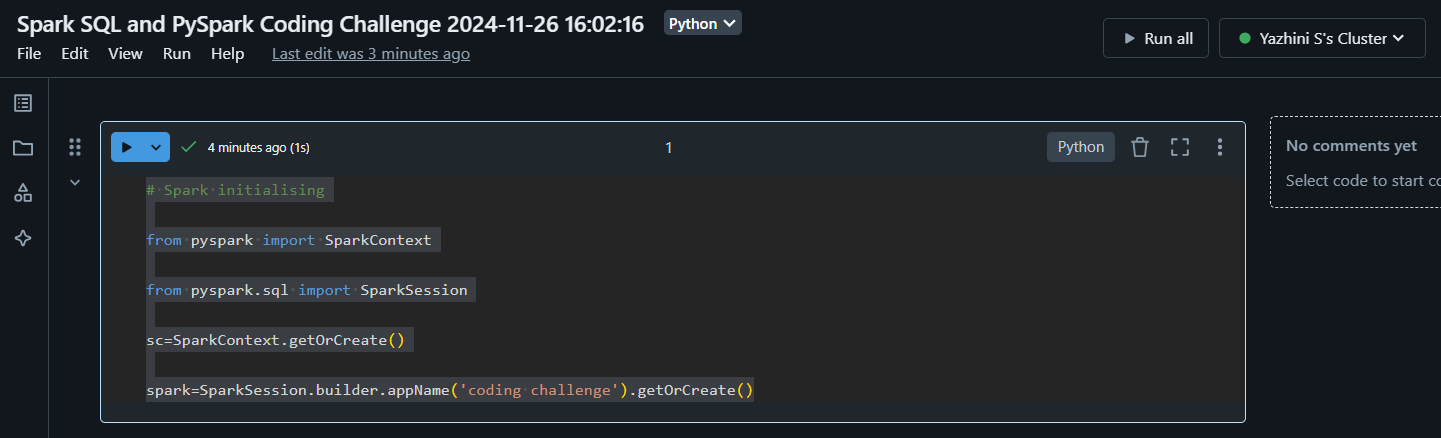
# Spark initialising

from pyspark import SparkContext

from pyspark.sql import SparkSession

sc=SparkContext.getOrCreate()

spark=SparkSession.builder.appName('coding challenge').getOrCreate()



**Checking the Schema and Initialising Data:**

# Loading data

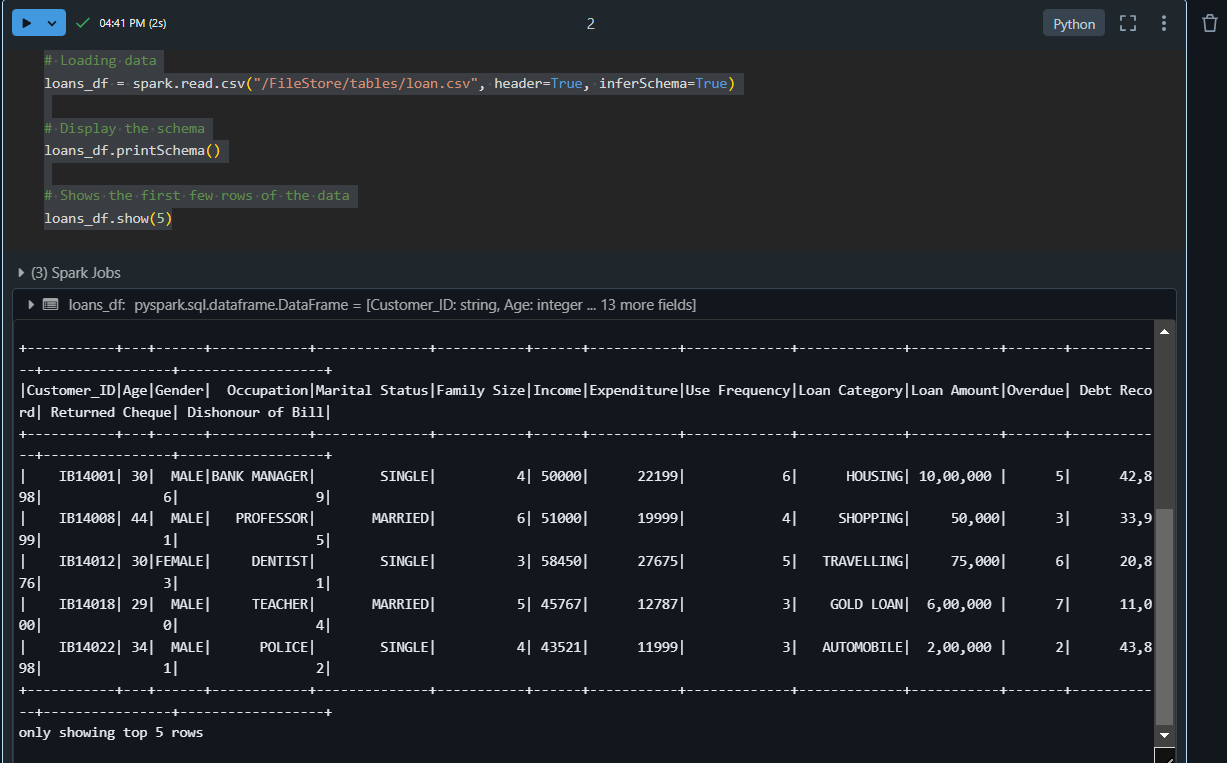
loans\_df = spark.read.csv("/FileStore/tables/loan.csv", header=True, inferSchema=True)

# Display the schema

loans\_df.printSchema()

# Shows the first few rows of the data

loans\_df.show(5)



**Filter transformation:**

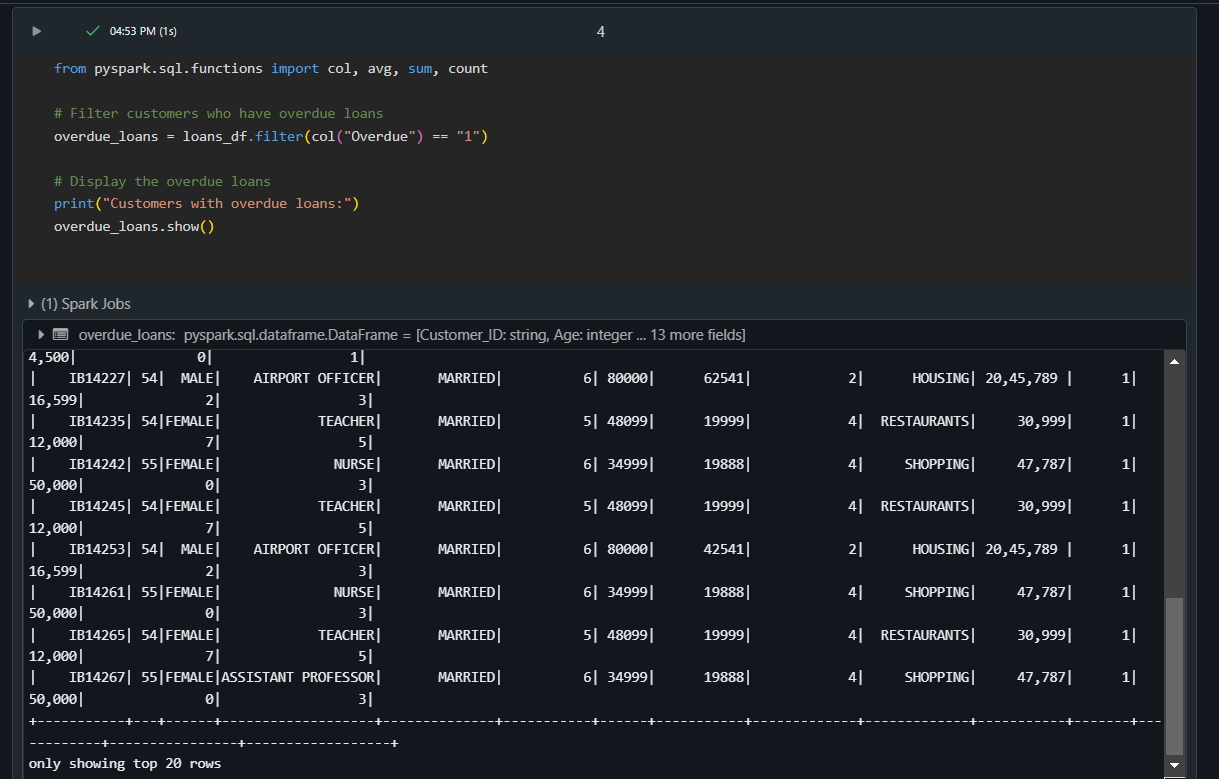
from pyspark.sql.functions import col, avg, sum, count

# Filter customers who have overdue loans

overdue\_loans = loans\_df.filter(col("Overdue") == "1")

# Display the overdue loans

print("Customers with overdue loans:")

overdue\_loans.show()****

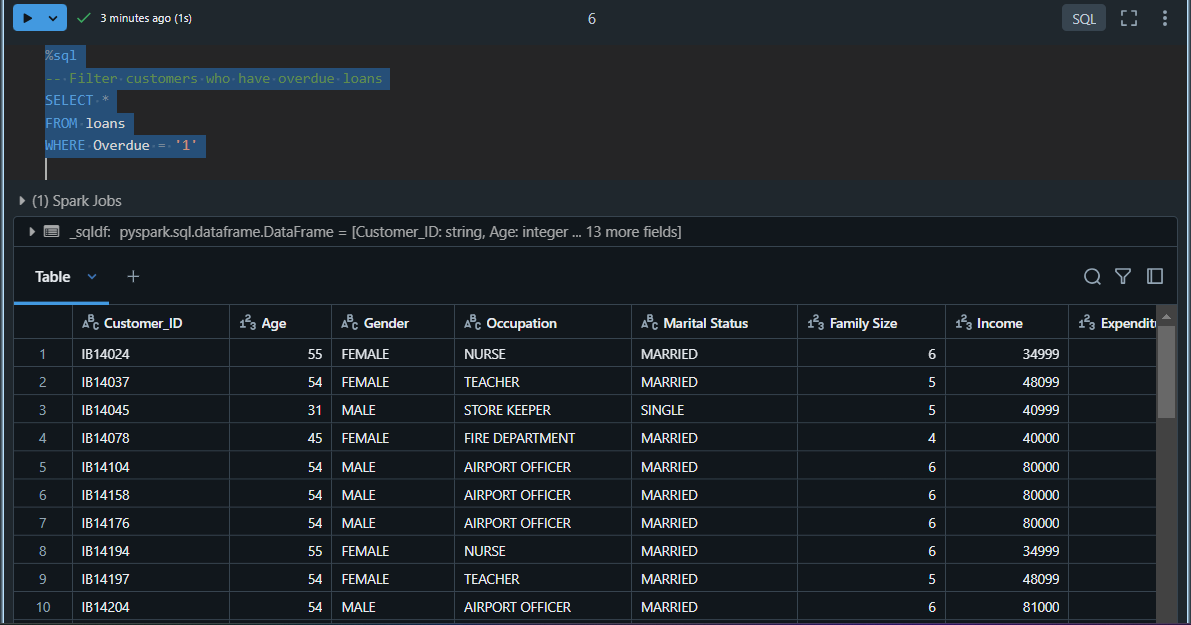
%sql

-- Filter customers who have overdue loans

SELECT \*

FROM loans

WHERE Overdue = '1'



**Filter with Multiple Conditions:**

# Filter for single customers with income > 5000

single\_high\_income\_customers = loans\_df.filter(

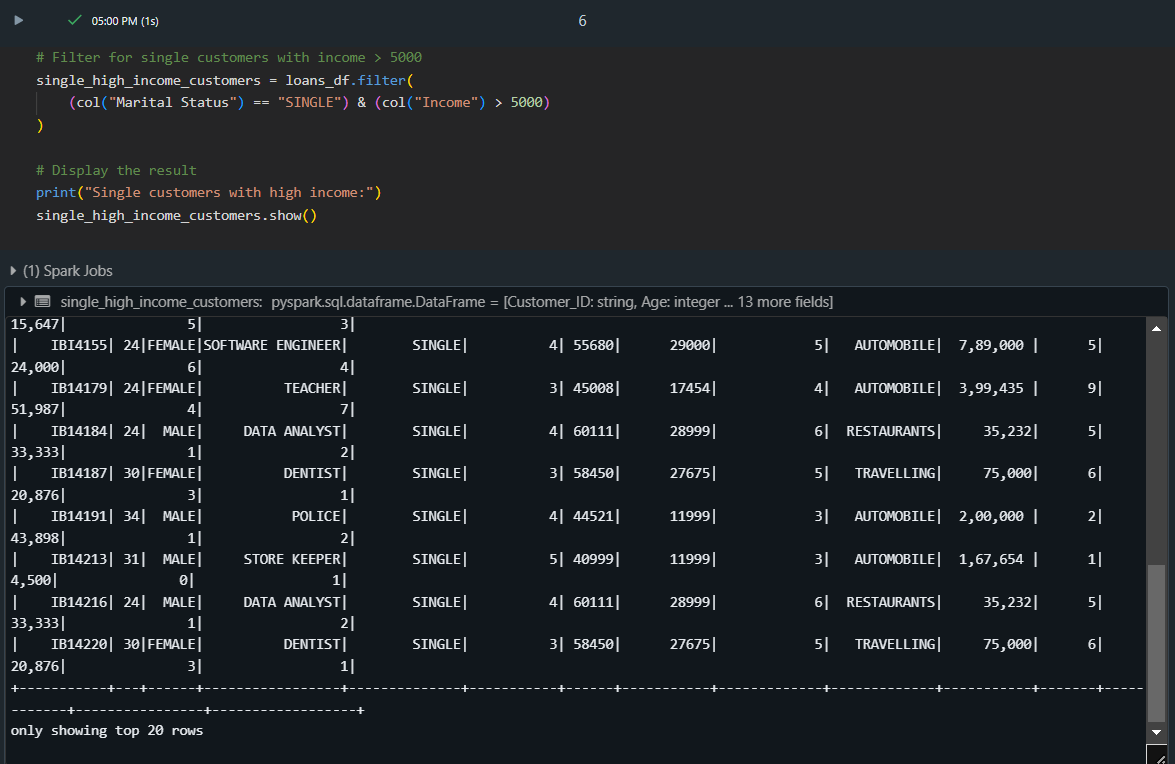
    (col("Marital Status") == "SINGLE") & (col("Income") > 5000)

)

# Display the result

print("Single customers with high income:")

single\_high\_income\_customers.show()

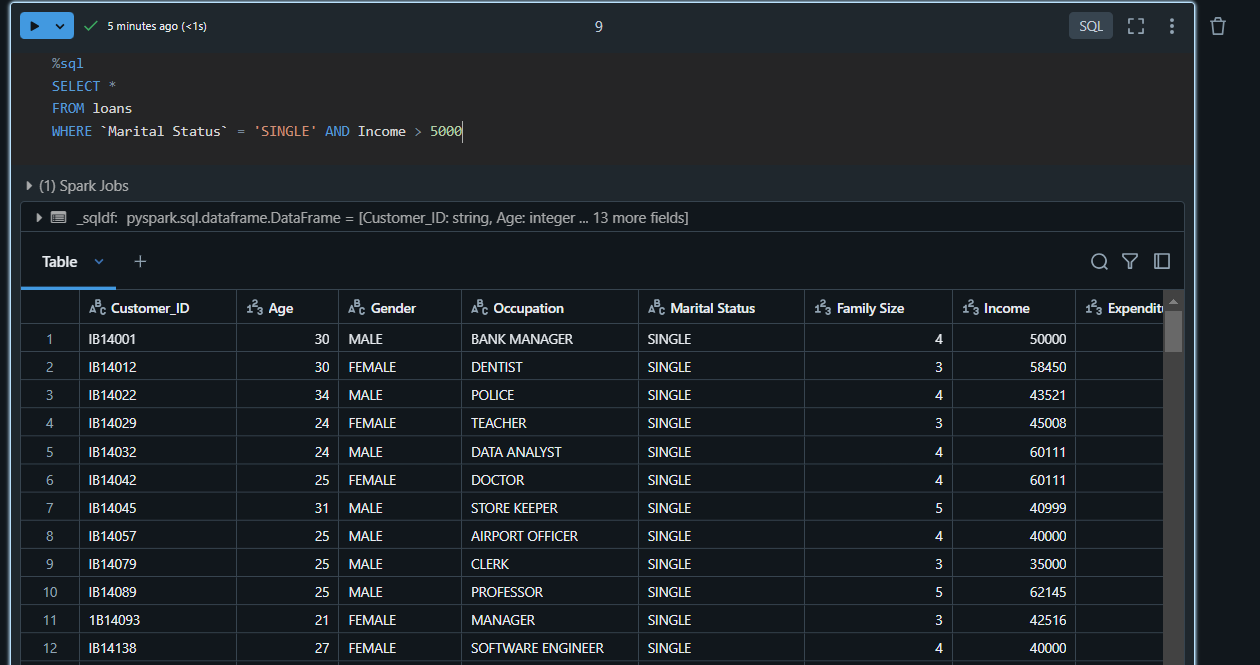


%sql

SELECT \*

FROM loans

WHERE `Marital Status` = 'SINGLE' AND Income > 5000



**GroupBy and Count:**

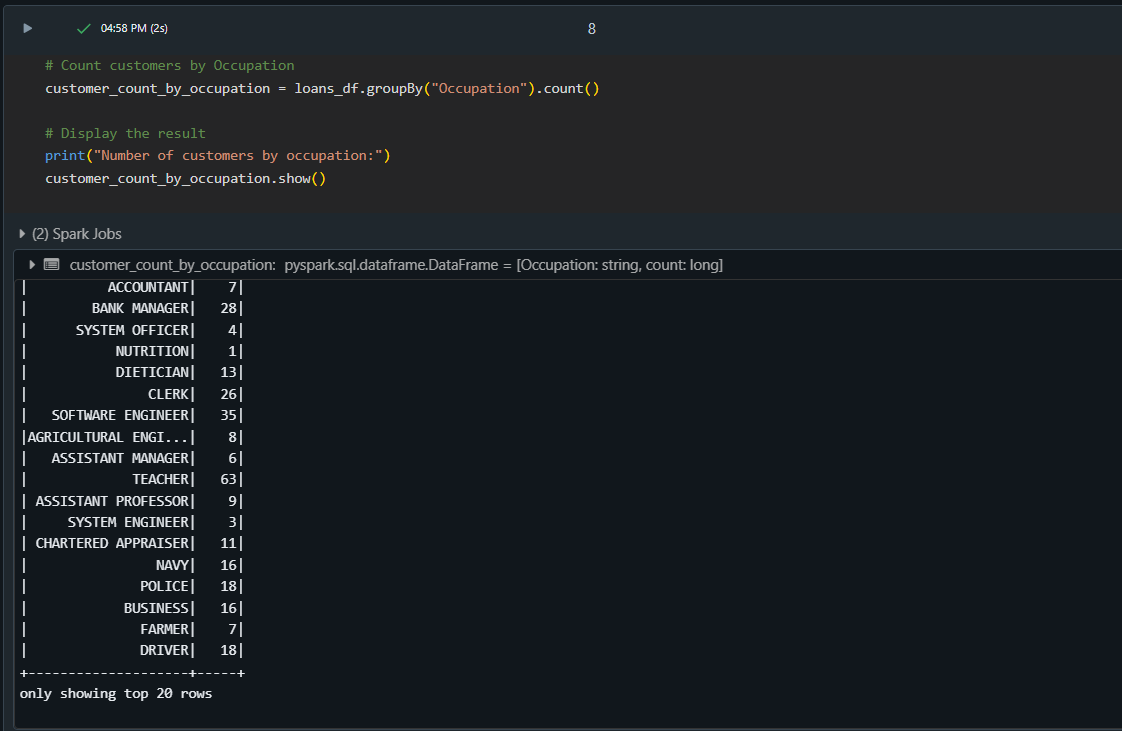
# Count customers by Occupation

customer\_count\_by\_occupation = loans\_df.groupBy("Occupation").count()

# Display the result

print("Number of customers by occupation:")

customer\_count\_by\_occupation.show()

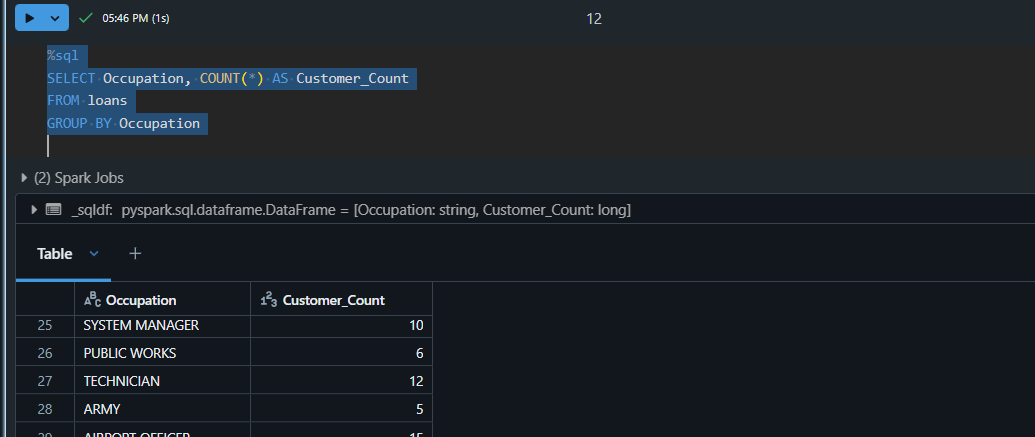


%sql

SELECT Occupation, COUNT(\*) AS Customer\_Count

FROM loans

GROUP BY Occupation



**Aggregation:**

# Total loan amount and average income grouped by marital status

loan\_income\_stats = loans\_df.groupBy("Marital Status").agg(

    sum("Loan Amount").alias("Total\_Loan\_Amount"),

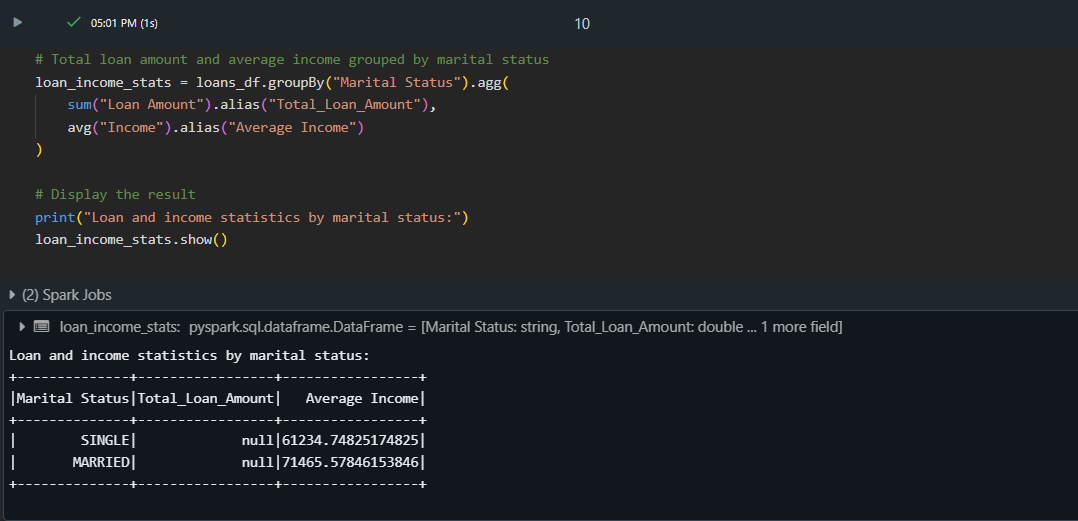
    avg("Income").alias("Average Income")

)

# Display the result

print("Loan and income statistics by marital status:")

loan\_income\_stats.show()



%sql

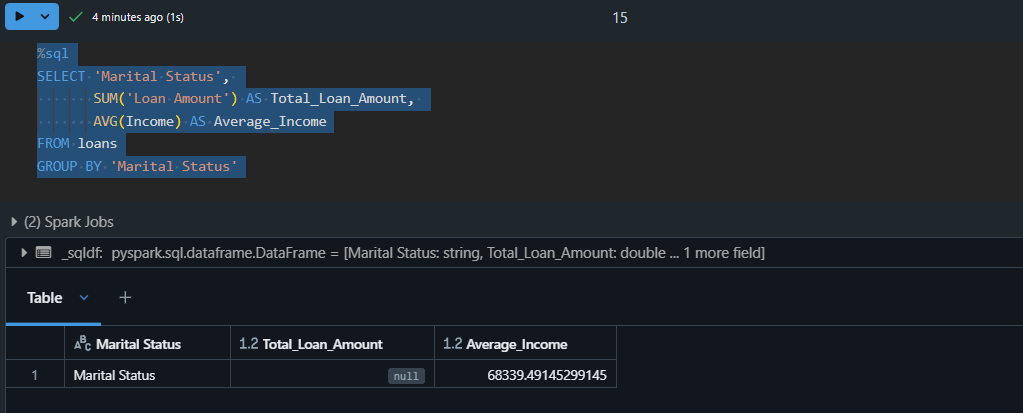
SELECT 'Marital Status',

       SUM('Loan Amount') AS Total\_Loan\_Amount,

       AVG(Income) AS Average\_Income

FROM loans

GROUP BY 'Marital Status'



**Inner join:**

# Load another dataset

credit\_df = spark.read.csv("/FileStore/tables/credit\_card.csv", header=True, inferSchema=True)

# Inner join on CustomerId and Customer\_ID

inner\_join\_df = credit\_df.join(loans\_df, credit\_df.CustomerId == loans\_df.Customer\_ID, how="inner")

# Display 10 rows from the inner join

print("Inner Join Result:")

inner\_join\_df.show(10)



%sql

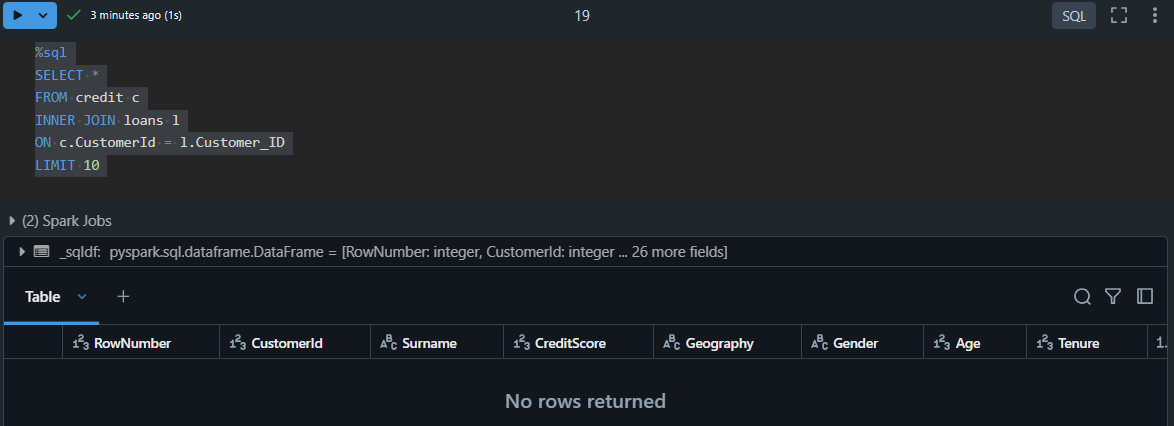
SELECT \*

FROM credit c

INNER JOIN loans l

ON c.CustomerId = l.Customer\_ID

LIMIT 10



**Outer join:**

# Outer join on CustomerId and Customer\_ID

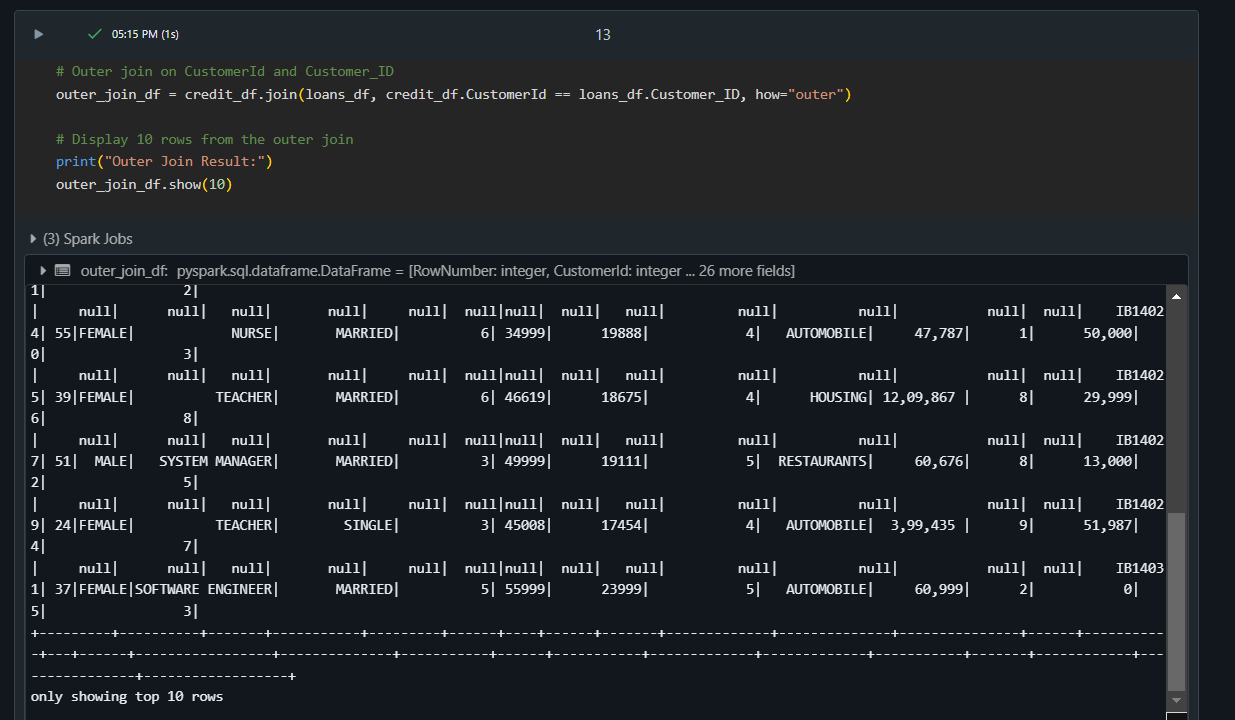
outer\_join\_df = credit\_df.join(loans\_df, credit\_df.CustomerId == loans\_df.Customer\_ID, how="outer")

# Display 10 rows from the outer join

print("Outer Join Result:")

outer\_join\_df.show(10)

NOTE: Some rows may have null values where no match is found.



%sql

SELECT \*

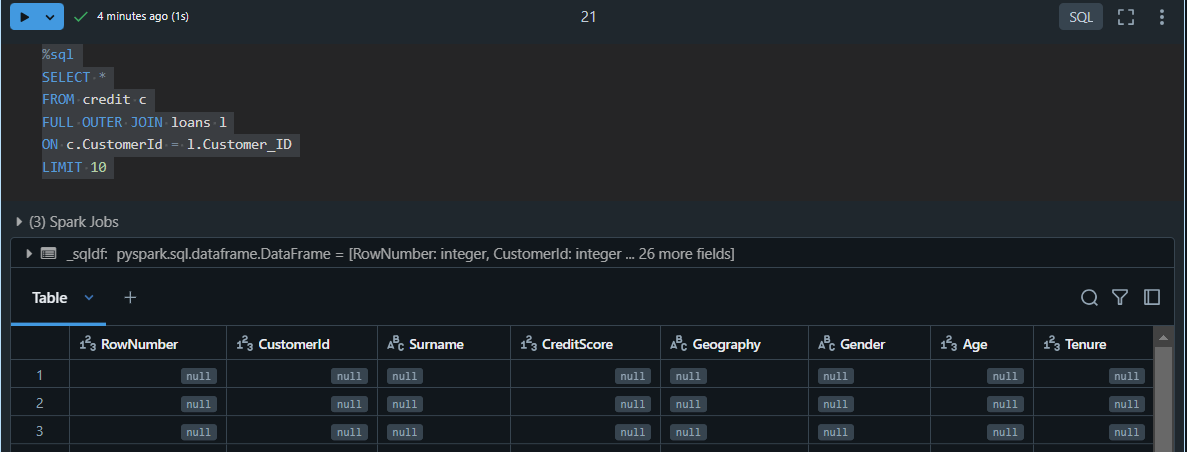
FROM credit c

FULL OUTER JOIN loans l

ON c.CustomerId = l.Customer\_ID

LIMIT 10

NOTE: Some rows may have null values where no match is found.



**Left join:**

# Left join on CustomerId and Customer\_ID

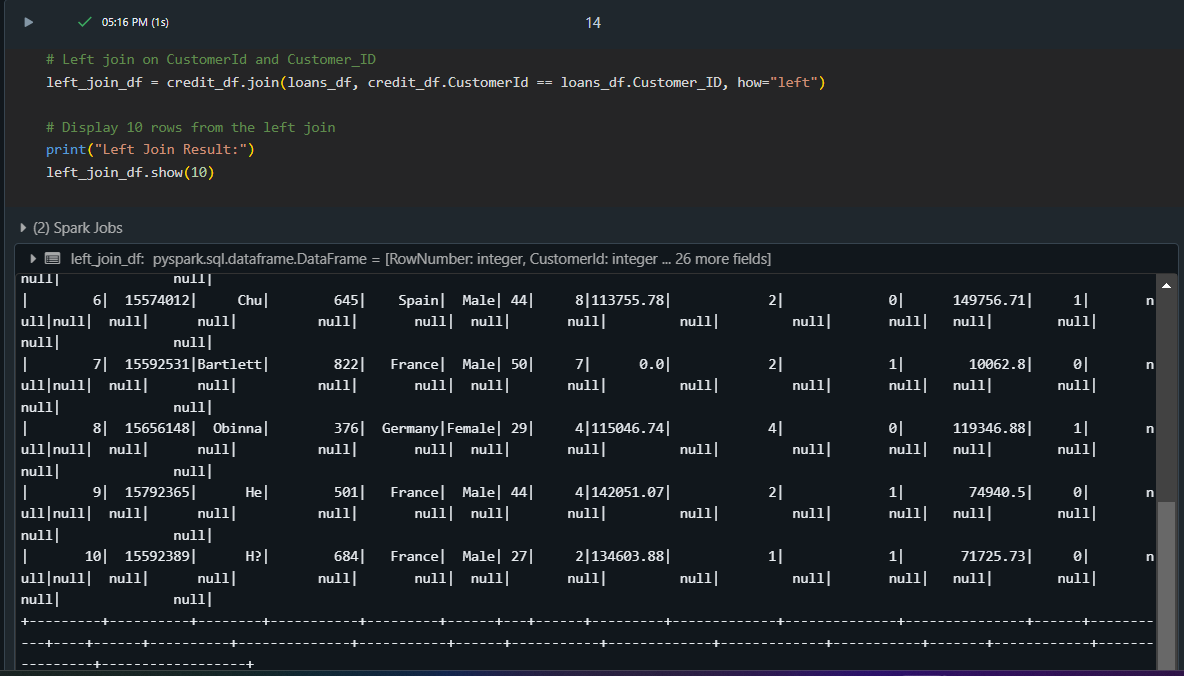
left\_join\_df = credit\_df.join(loans\_df, credit\_df.CustomerId == loans\_df.Customer\_ID, how="left")

# Display 10 rows from the left join

print("Left Join Result:")

left\_join\_df.show(10)

NOTE: Some rows may have null values where no match is found.



%sql

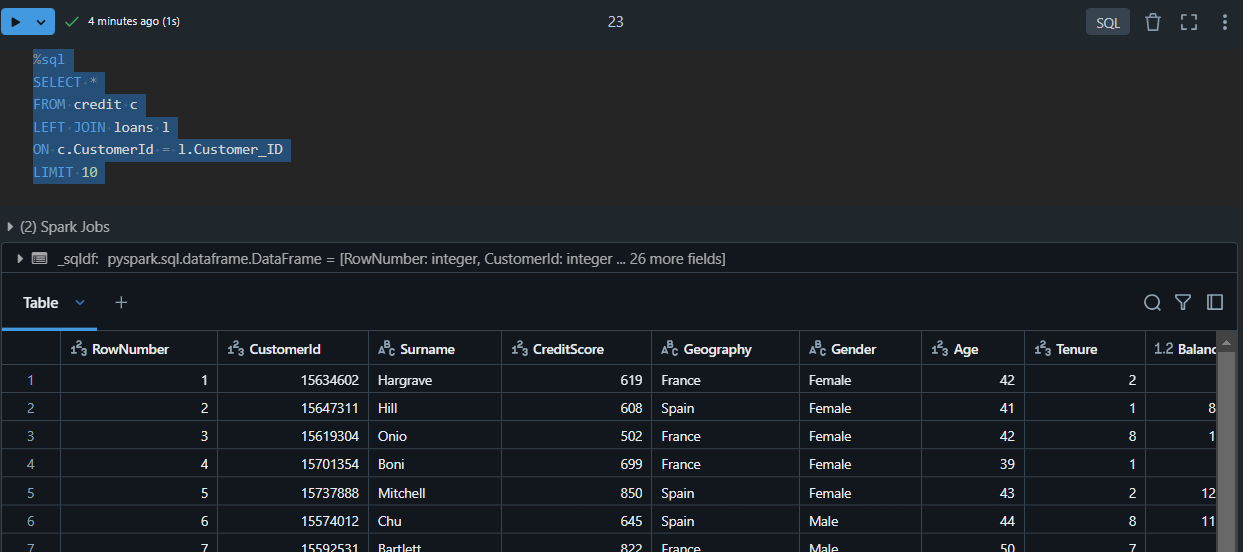
SELECT \*

FROM credit c

LEFT JOIN loans l

ON c.CustomerId = l.Customer\_ID

LIMIT 10



**Right join:**

# Right join on CustomerId and Customer\_ID

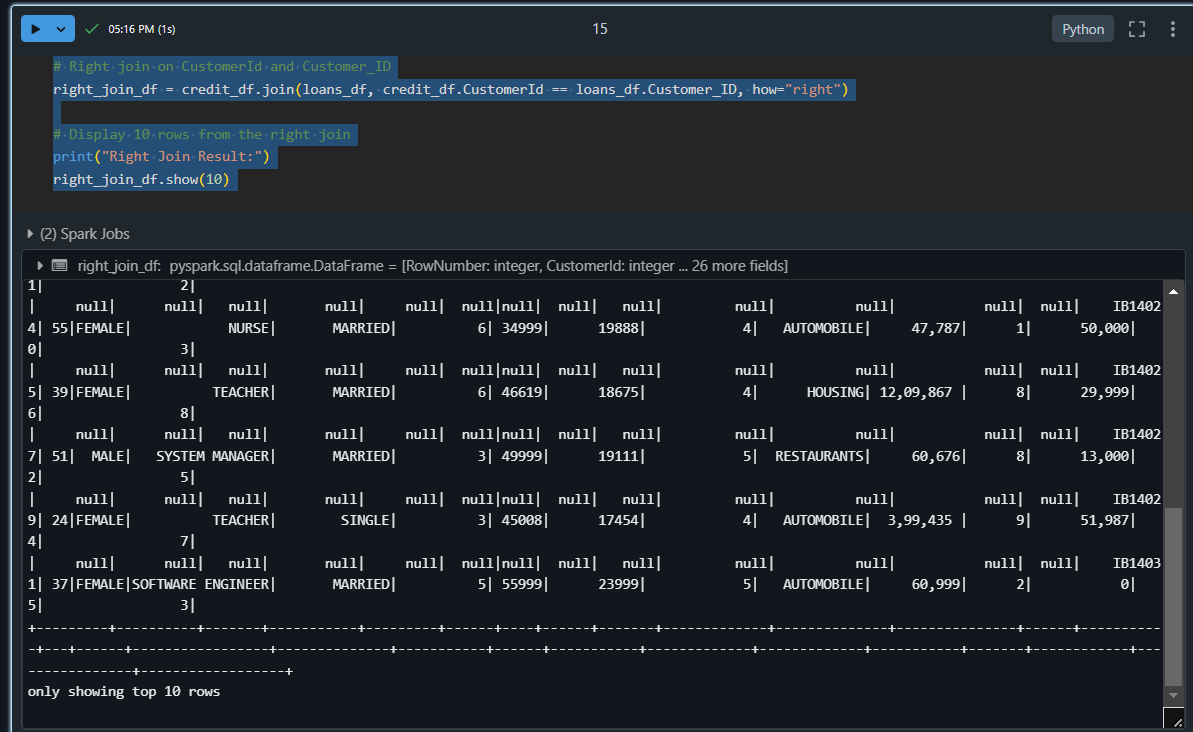
right\_join\_df = credit\_df.join(loans\_df, credit\_df.CustomerId == loans\_df.Customer\_ID, how="right")

# Display 10 rows from the right join

print("Right Join Result:")

right\_join\_df.show(10)

NOTE: Some rows may have null values where no match is found.



%sql

SELECT \*

FROM credit c

RIGHT JOIN loans l

ON c.CustomerId = l.Customer\_ID

LIMIT 10

