T.C.

FATIH SULTAN MEHMET VAKIF UNIVERSITY

Faculty of Engineering

Department of Software

Engineering

Big Data Analytics Project Report

Students Name : Zeynep Dağtekin

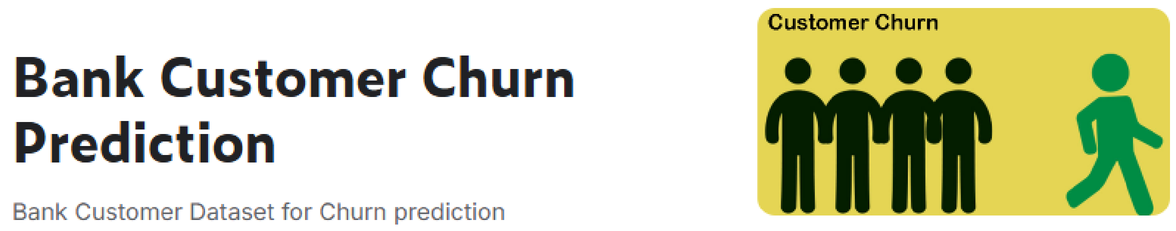
Student ID : 2121251007

Course Instructor : Asst. Prof. Dr. Ümit Demirbağa

INTRODUCTION:

Firstly İ searched about a dataset for my project to implement MapReduce, SQL query and ML. I chose a dataset that i believe is most suitable for the operations i will perform:

Bank Customer Churn Prediction.
The link of dataset:
https://www.kaggle.com/datasets/shantanudhakadd/bank-customer-churn-prediction


Bank Customer Churn Prediction

Bank Customer Dataset for Churn prediction

The reason for choosing this dataset was it had almost 10k rows, which was sufficient for my project and it had enough data diversity. Besides it was such a interesting data that caught my interest.

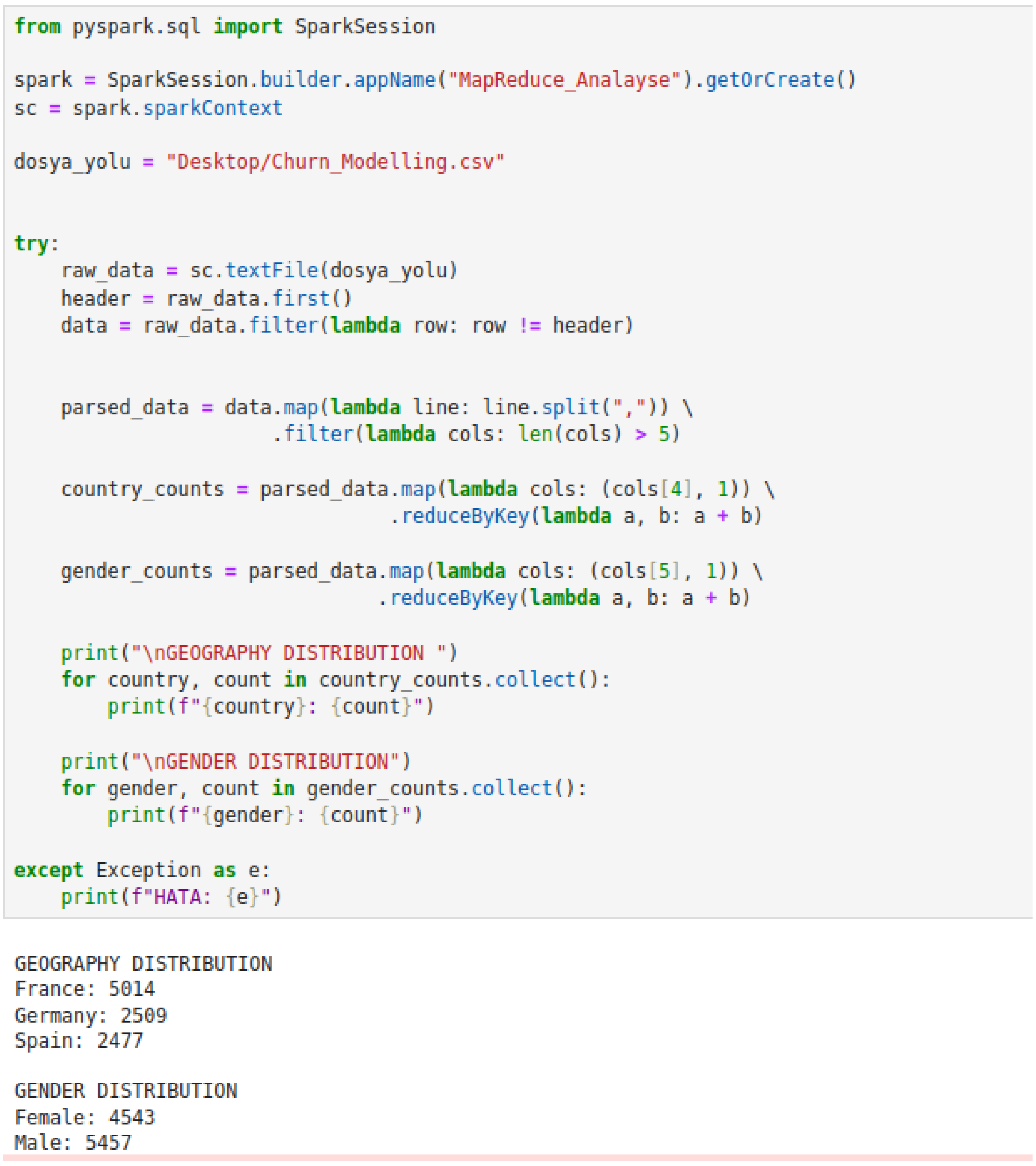
MAPREDUCE IMPLEMENTATION:

MapReduce actually includes two processes: Map and Reduce.

In the Map part, i receive input. This input is each row in our project's dataset. For each input i receive, the number 1 is written as output, meaning it is converted into key-value pairs. For example, I took our dataset here and wanted to find the number of men and women. In the Map part , it outputs (female, 1).

When i look at the Reduce part, the outputs from the Map phase are grouped based on the key and then aggregated and combined. That is, it aggregates all instances of (female, 1) and outputs (female, 4543).

In short, MapReduce handles such a large task by dividing and conquering. In this section, I used MapReduce to calculate both the Gender Distribution and the Geographic Distribution of customers.



```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MapReduce_Analayse").getOrCreate()
sc = spark.sparkContext

dosya_yolu = "Desktop/Churn_Modelling.csv"


try:
    raw_data = sc.textFile(dosya_yolu)
    header = raw_data.first()
    data = raw_data.filter(lambda row: row != header)


    parsed_data = data.map(lambda line: line.split(",")) \
                      .filter(lambda cols: len(cols) > 5)

    country_counts = parsed_data.map(lambda cols: (cols[4], 1)) \
                                .reduceByKey(lambda a, b: a + b)

    gender_counts = parsed_data.map(lambda cols: (cols[5], 1)) \
                               .reduceByKey(lambda a, b: a + b)

    print("\nGEOGRAPHY DISTRIBUTION ")
    for country, count in country_counts.collect():
        print(f"{country}: {count}")

    print("\nGENDER DISTRIBUTION")
    for gender, count in gender_counts.collect():
        print(f"{gender}: {count}")

except Exception as e:
    print(f"HATA: {e}")
```

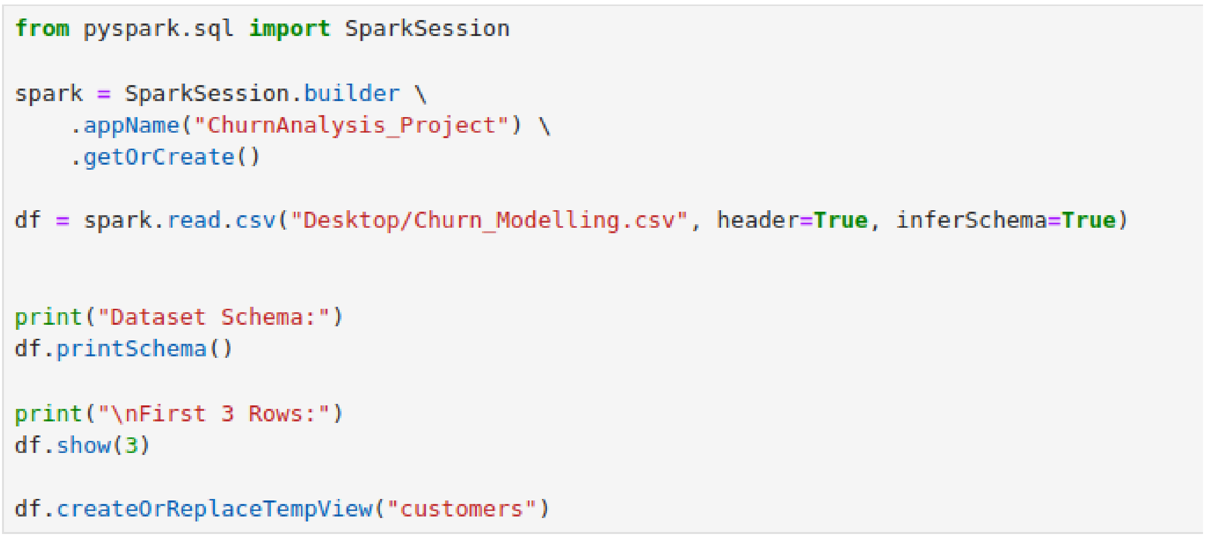
```
GEOGRAPHY DISTRIBUTION
France: 5014
Germany: 2509
Spain: 2477

GENDER DISTRIBUTION
Female: 4543
Male: 5457
```

DATA ANALYSIS WITH SQL

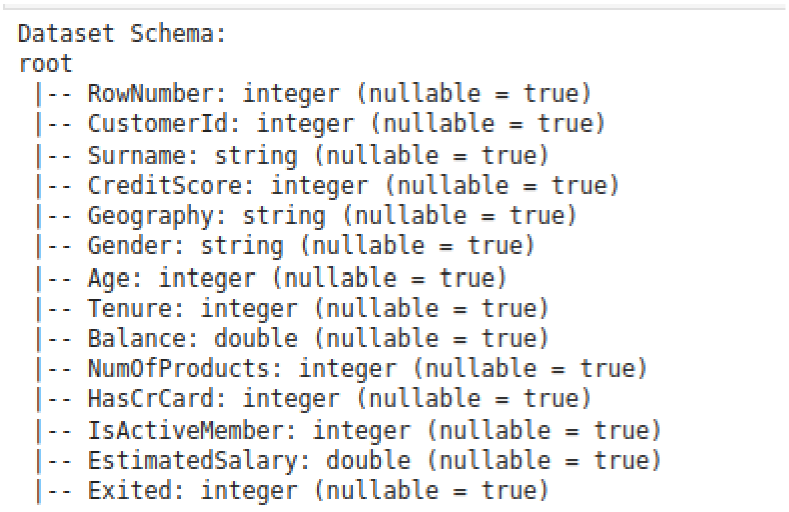
MapReduce is a low level RDD operation, and RDD gives you very specific control over everything. However, using Spark SQL and data frames is simpler and more convenient.

That's why I started a Spark session here. This Spark part allows you to read data and manage machine learning libraries from a single location.

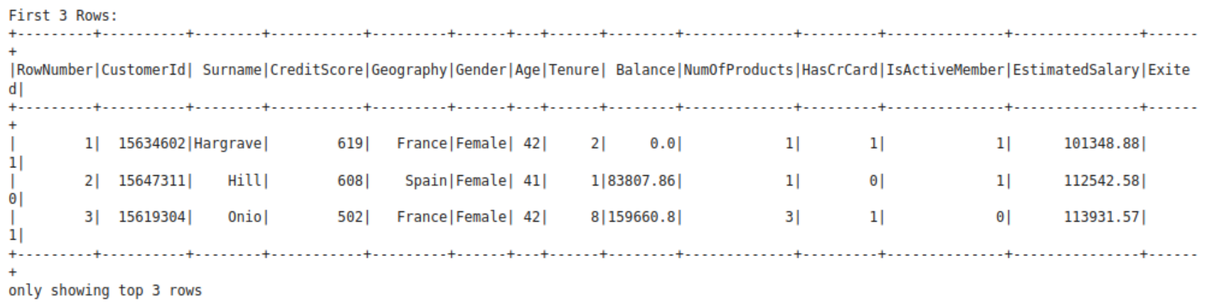
In here i read the dataset using the inferSchema=True parameter, which automatically detects the data type of the columns in the dataset and converts it to the correct schema in dataframe format. To run SQL queries, i created a temporary virtual table named customers using the createOrReplaceTempView("customers") command. The reason for doing this is that SQL cannot directly read an Excel file.



```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("ChurnAnalysis_Project") \
    .getOrCreate()

df = spark.read.csv("Desktop/Churn_Modelling.csv", header=True, inferSchema=True)


print("Dataset Schema:")
df.printSchema()

print("\nFirst 3 Rows:")
df.show(3)

df.createOrReplaceTempView("customers")
```



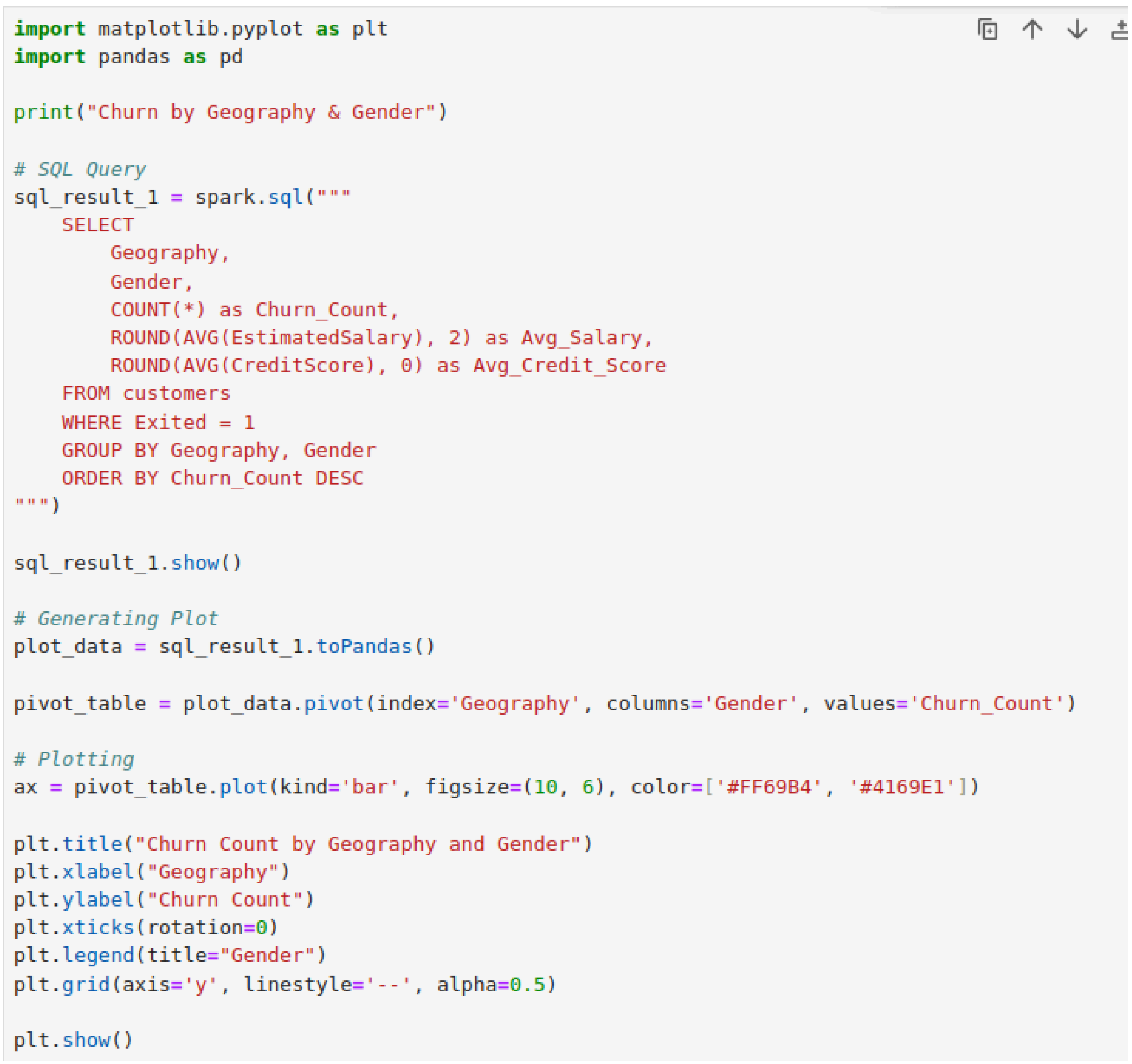
```
Dataset Schema:
root
 |-- RowNumber: integer (nullable = true)
 |-- CustomerId: integer (nullable = true)
 |-- Surname: string (nullable = true)
 |-- CreditScore: integer (nullable = true)
 |-- Geography: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Tenure: integer (nullable = true)
 |-- Balance: double (nullable = true)
 |-- NumOfProducts: integer (nullable = true)
 |-- HasCrCard: integer (nullable = true)
 |-- IsActiveMember: integer (nullable = true)
 |-- EstimatedSalary: double (nullable = true)
 |-- Exited: integer (nullable = true)
```



```
First 3 Rows:
+---------+----------+--------+-----------+---------+------+---+------+--------+-------------+---------+--------------+---------------+------
+
|RowNumber|CustomerId| Surname|CreditScore|Geography|Gender|Age|Tenure| Balance|NumOfProducts|HasCrCard|IsActiveMember|EstimatedSalary|Exite
d|
+---------+----------+--------+-----------+---------+------+---+------+--------+-------------+---------+--------------+---------------+------
+
|        1|  15634602|Hargrave|        619|   France|Female| 42|     2|     0.0|            1|        1|             1|      101348.88|
1|
|        2|  15647311|    Hill|        608|    Spain|Female| 41|     1|83807.86|            1|        0|             1|      112542.58|
0|
|        3|  15619304|    Onio|        502|   France|Female| 42|     8|159660.8|            3|        1|             0|      113931.57|
1|
+---------+----------+--------+-----------+---------+------+---+------+--------+-------------+---------+--------------+---------------+------
+
only showing top 3 rows
```

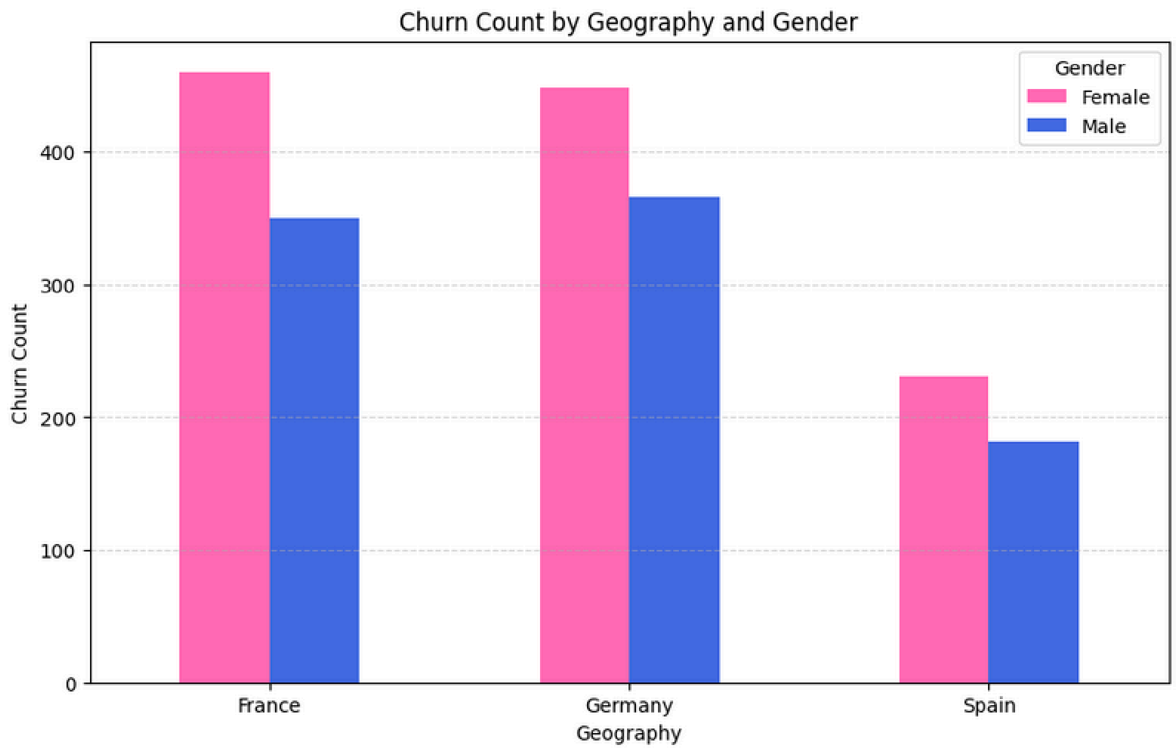
Query 1: Churn Analysis by Geography & Gender

The main purpose of my SQL query here was to create a profile of customers who left the bank. To find the answer to the question of which countries and genders of customers were leaving the bank more, I used WHERE Exited= 1 and the group by command. I also calculated the average salary and credit score to see the quality of the lost customers. For example, the number of female customers lost in France is high. When I looked at their average salaries, they were high. This means i have lost high-quality (i.e., high-income) customers. This is a negative situation for the bank.



```python
import matplotlib.pyplot as plt
import pandas as pd

print("Churn by Geography & Gender")

# SQL Query
sql_result_1 = spark.sql("""
    SELECT
        Geography,
        Gender,
        COUNT(*) as Churn_Count,
        ROUND(AVG(EstimatedSalary), 2) as Avg_Salary,
        ROUND(AVG(CreditScore), 0) as Avg_Credit_Score
    FROM customers
    WHERE Exited = 1
    GROUP BY Geography, Gender
    ORDER BY Churn_Count DESC
""")

sql_result_1.show()

# Generating Plot
plot_data = sql_result_1.toPandas()

pivot_table = plot_data.pivot(index='Geography', columns='Gender', values='Churn_Count')

# Plotting
ax = pivot_table.plot(kind='bar', figsize=(10, 6), color=['#FF69B4', '#4169E1'])

plt.title("Churn Count by Geography and Gender")
plt.xlabel("Geography")
plt.ylabel("Churn Count")
plt.xticks(rotation=0)
plt.legend(title="Gender")
plt.grid(axis='y', linestyle='--', alpha=0.5)

plt.show()
```

```
Churn by Geography & Gender
+---------+------+-----------+----------+----------------+
|Geography|Gender|Churn_Count|Avg_Salary|Avg_Credit_Score|
+---------+------+-----------+----------+----------------+
|   France|Female|        460| 103626.03|           644.0|
|  Germany|Female|        448|  99884.46|           651.0|
|  Germany|  Male|        366|   96591.6|           644.0|
|   France|  Male|        350| 103193.84|           639.0|
|    Spain|Female|        231|  107544.1|           645.0|
|    Spain|  Male|        182|   98661.1|           650.0|
+---------+------+-----------+----------+----------------+
```



When i look at the graph, I see that France is the country where the most women have left the bank, while Germany is the country where the most men have left the bank.

Query 2: Credit Score Segmentation

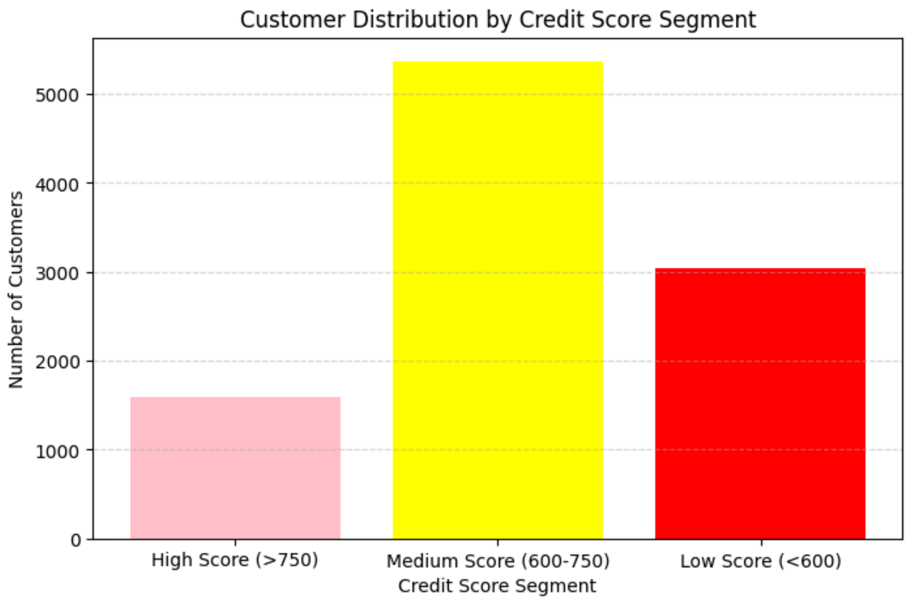
In this SQL query, I targeted to categorize customers into risk groups based on their credit scores. I divided customers into three groups according to their credit scores:

- low score (high risk group)
- medium score (standard group)
- high score (reliable group).

I classified them in three categories. I set scores below 600 as a low score, 600-750 as medium, and 750 and above as high score.

```python
print("Credit Score Segmentation")


sql_result_2 = spark.sql("""
    SELECT
        CASE
            WHEN CreditScore < 600 THEN 'Low Score (<600)'
            WHEN CreditScore BETWEEN 600 AND 750 THEN 'Medium Score (600-750)'
            ELSE 'High Score (>750)'
        END AS Score_Group,
        COUNT(*) as Customer_Count,
        ROUND(AVG(Balance), 2) as Avg_Balance
    FROM customers
    GROUP BY
        CASE
            WHEN CreditScore < 600 THEN 'Low Score (<600)'
            WHEN CreditScore BETWEEN 600 AND 750 THEN 'Medium Score (600-750)'
            ELSE 'High Score (>750)'
        END
    ORDER BY Avg_Balance DESC
""")

sql_result_2.show()

# Generating Plot
plot_data_2 = sql_result_2.toPandas()

plt.figure(figsize=(8,5))
plt.bar(plot_data_2['Score_Group'], plot_data_2['Customer_Count'], color=['pink', 'yellow', 'red'])

plt.title("Customer Distribution by Credit Score Segment")
plt.xlabel("Credit Score Segment")
plt.ylabel("Number of Customers")
plt.grid(axis='y', linestyle='--', alpha=0.5)

plt.show()
```

```
Credit Score Segmentation
+-------------------+--------------+-----------+
|        Score_Group|Customer_Count|Avg_Balance|
+-------------------+--------------+-----------+
|   High Score (>750)|          1598|   78740.13|
|Medium Score (600...|          5368|   76266.67|
|    Low Score (<600)|          3034|   75686.45|
+-------------------+--------------+-----------+
```



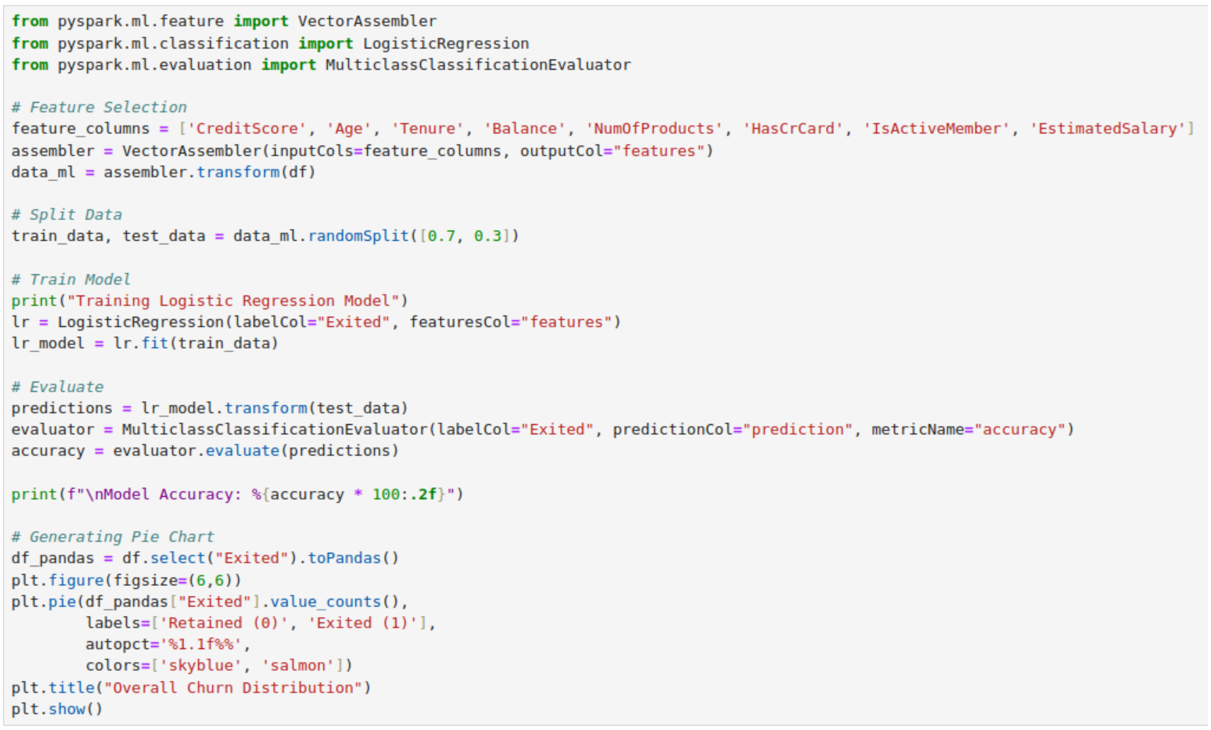
When i check the graph, most customers show an excess in the middle score group. Customers with high scores have the highest average balance. Customers with low scores have the lowest average balance. So, i can say that customers with high credit scores hold the most deposits in the bank, as i can see from the graph.

MACHINE LEARNING

1) Classification Task: Churn Prediction

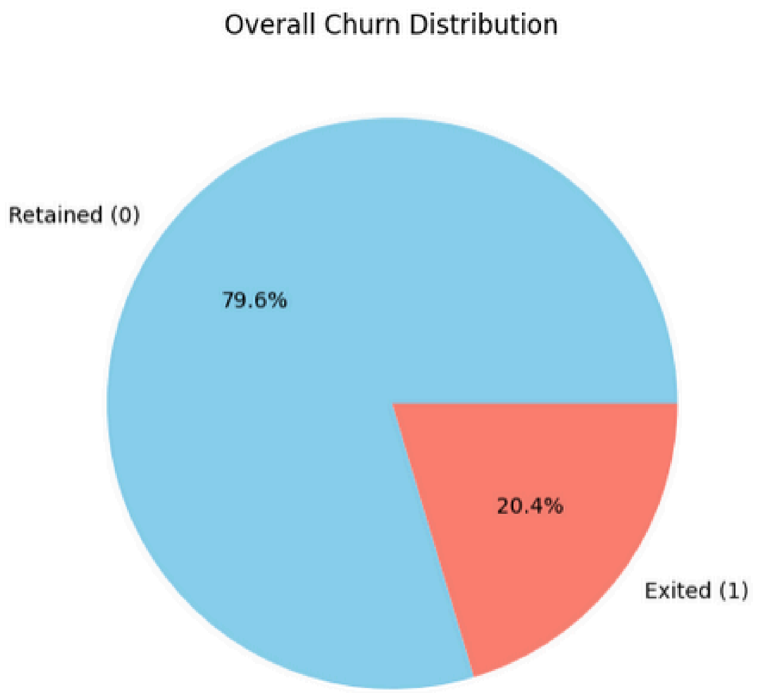
In here i created a classification model to predict a customer will leave the bank or not. I used the logistic regression algorithm from supervised learning algorithms. I used customer features to train the model, like customer's age, balance, and credit score.

I divided dataset in two parts: 70% for training data and 30% for test data. The reason for this split was to prevent overfitting.



```python
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Feature Selection
feature_columns = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
data_ml = assembler.transform(df)

# Split Data
train_data, test_data = data_ml.randomSplit([0.7, 0.3])

# Train Model
print("Training Logistic Regression Model")
lr = LogisticRegression(labelCol="Exited", featuresCol="features")
lr_model = lr.fit(train_data)

# Evaluate
predictions = lr_model.transform(test_data)
evaluator = MulticlassClassificationEvaluator(labelCol="Exited", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)

print(f"\nModel Accuracy: %{accuracy * 100:.2f}")

# Generating Pie Chart
df_pandas = df.select("Exited").toPandas()
plt.figure(figsize=(6,6))
plt.pie(df_pandas["Exited"].value_counts(),
        labels=['Retained (0)', 'Exited (1)'],
        autopct='%1.1f%%',
        colors=['skyblue', 'salmon'])
plt.title("Overall Churn Distribution")
plt.show()
```

```
Training Logistic Regression Model

Model Accuracy: %81.74
```



When i check the graph, 79.6% of customers remained with the bank, while 20.4% left. It achieved an accuracy rate of almost 80%. That means that the model has analyzed customer behavior well to make these predictions.

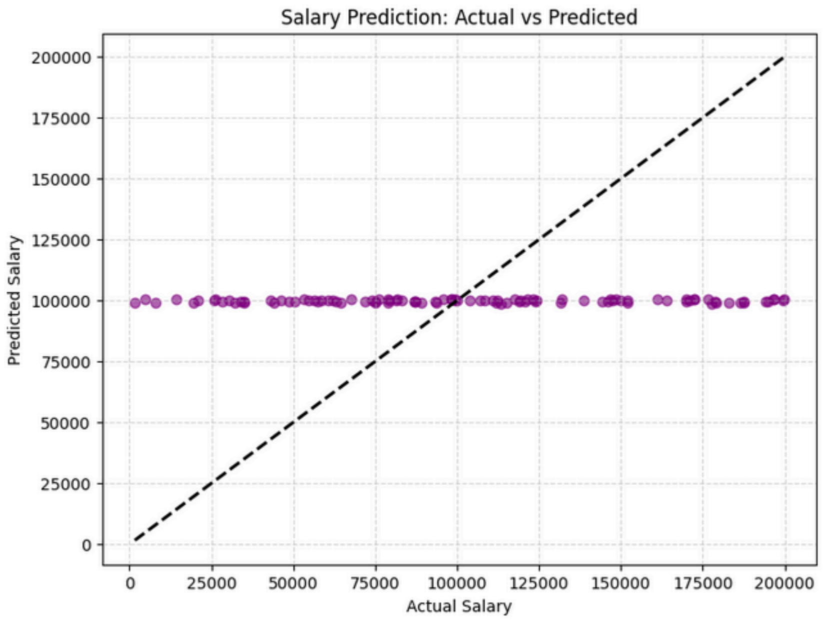
B) Regression Task: Salary Prediction

In here i wanted to test something a little different. I targeted to predict salaries by looking at certain customer characteristics (Age, CreditScore, Tenure). For this, I used the linear regression algorithm.

This time i wanted to test with different rate so i divided the data set into 80% Training and 20% Test. I used RMSE to measure the success of the regression model, and the result was 57.205.

```python
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
import matplotlib.pyplot as plt


# Vector Assembler
assembler_reg = VectorAssembler(inputCols=['Age', 'CreditScore', 'Tenure'], outputCol="features_salary")
df_reg = assembler_reg.transform(df)

# Split Data
train_reg, test_reg = df_reg.randomSplit([0.8, 0.2])

# Train Model
print("Training Linear Regression Model...")
lr_reg = LinearRegression(featuresCol="features_salary", labelCol="EstimatedSalary")
lr_reg_model = lr_reg.fit(train_reg)

# Evaluate (RMSE)
pred_reg = lr_reg_model.transform(test_reg)
evaluator_reg = RegressionEvaluator(labelCol="EstimatedSalary", predictionCol="prediction", metricName="rmse")
rmse = evaluator_reg.evaluate(pred_reg)

print("\n--- Regression Results ---")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# Taking a sample for cleaner visualization
plot_data = pred_reg.select("EstimatedSalary", "prediction").limit(100).toPandas()

plt.figure(figsize=(8,6))
plt.scatter(plot_data["EstimatedSalary"], plot_data["prediction"], color='purple', alpha=0.6)

# Ideal prediction line
plt.plot([plot_data["EstimatedSalary"].min(), plot_data["EstimatedSalary"].max()],
         [plot_data["EstimatedSalary"].min(), plot_data["EstimatedSalary"].max()],
         'k--', lw=2)

plt.xlabel("Actual Salary")
plt.ylabel("Predicted Salary")
plt.title("Salary Prediction: Actual vs Predicted")
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

```
--- Regression Results ---
Root Mean Squared Error (RMSE): 57205.90
```



Wheni looked at the graph, i was waiting something like this to happen. So i see that every dataset or attribute group is not suitable for every type of prediction problem. For example, here I clearly saw that there was no linear relationship between an attribute such as salary and age, credit score, and tenure.

CONCLUSION:

In this project, i learned how to analyze big data and manage machine learning processes using Apache Spark. I first experienced with MapReduce on my dataset, then transitioned to the DataFrame and Spark SQL structure to ensure compatibility with ML libraries. So i believe that Apache Spark's ability to both process such large and voluminous data and produce meaningful results from this data makes it more valuable, and this project helped me understand why Apache Spark is so important.