# Data Pipeline Report: Apache Spark and Kafka (Aiven)

## Introduction

This report documents the implementation of a data pipeline using Apache Spark for batch processing and Aiven Kafka for real-time data streaming. It outlines the tools, steps, and challenges encountered during the project.

## 1. Environment and Technologies

The following tools were used for the project:

**Apache Spark**: A powerful distributed computing system used for batch and real-time data processing.

**Aiven Kafka**: A managed cloud-based Kafka service for real-time event streaming. It simplified the setup and scaling of Kafka for handling student data streams.

**Python**: The primary programming language for scripting and integrating Spark and Kafka, automating data ingestion, transformation, and streaming.

**PySpark and Kafka-python Libraries**:

* **PySpark** for interacting with Spark to process and transform data.
* **Kafka-python** for integrating with Kafka, enabling producers to send data and consumers to process real-time messages.

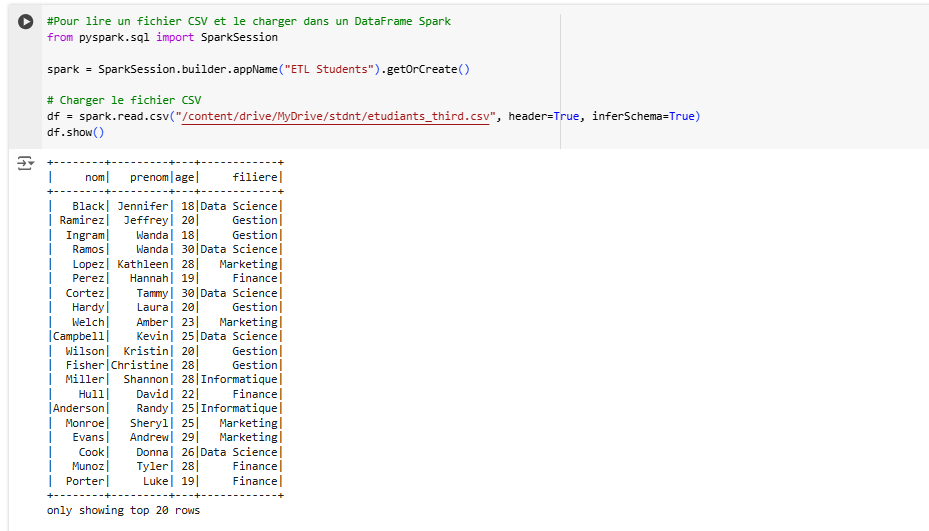
A screenshot of a computer

Description automatically generated

## 2. ETL Pipeline Steps with Spark

The pipeline processes student data through these steps:

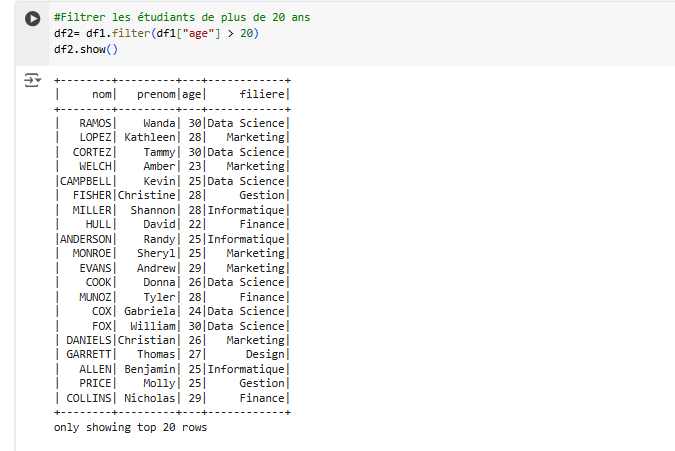
1. **Data Extraction**: Load data from a CSV file into a Spark DataFrame using spark.read.csv(), ensuring automatic schema detection.
2. **Data Transformation**:
   * Convert names to uppercase for consistency.
   * Filter students aged over 20.
   * Calculate the average age per department using groupBy() and avg().
3. **Data Loading**: Save the transformed data and statistics to a new CSV file for further use.



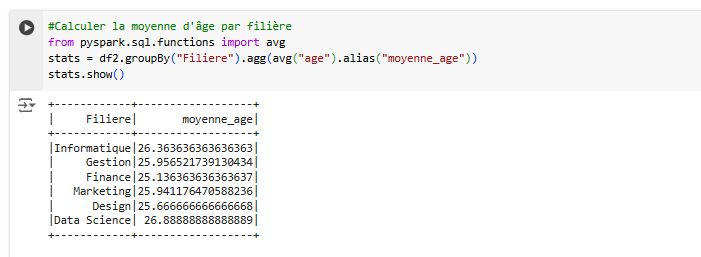
Screenshot: Loading CSV data into Spark DataFrame.



Screenshot: Converting names to uppercase.



Screenshot: Filtering students over 20 years old.



Screenshot: Calculating average age by department.

## 3. Kafka and Spark Streaming Integration

Kafka integration involves setting up a producer to send data to a Kafka topic and a consumer to process the real-time messages using Spark Streaming.



Screenshot: Kafka producer configuration.



Screenshot: Kafka consumer configuration.

## Conclusion

This project demonstrates the effective use of Spark and Kafka for building a robust data pipeline. The pipeline successfully processed batch and streaming data while addressing common challenges like data integration and real-time processing. The tools are well-configured for production use.