Data Engineering Challenge:

Introduction

We want to thank you for having time to work on this challenge. The challenge has three main topics: Streaming, Batch processing, and Dimensional Modelling. We expect the following:

1. You perform very well in at least one of those three topics.
2. You have some knowledge in one of the two remaining topics.

We evaluate your solution based on several factors:

1. You follow a systemic logic to solve the problem.
2. The delivered solution should work on your computer and mine as well.
3. You provide high-quality and clean code and architecture.
4. You can reasonably put assumptions to move forward with the solution.

Following software engineering best practices has a significant impact on the evaluation of your solution.

Part 1: Streaming

You are working on a project whose client requires developing a streaming pipeline that listens to some data sources and dumps the data into storage. You have to deploy the pipeline into two different environments (***Prod-env1*** and ***Prod-env2***). For ***Prod-env1***, the following is expected:

- The pipeline reads **JSON** files as a stream
- It processes and flattens the data
- It stores it as **CSV** files.

And for ***Prod-env2***:

- The pipeline reads the stored **CSV** files from the previous step
- Applies the corresponding transformations
- And stores the data in **Parquet** files.

Your job is to implement the above-described pipeline using **Spark**. You can use **Python**, **Scala**, **JAVA or .Net language**. The attached Python code snippet generates the **JSON** files for the first part of the pipeline and stores them in the *"data/json_files/."*

The first part of the streaming pipeline stores the **CSV** files in a *"data/csv_files/."* The second part of the pipeline that reads data from *"data/csv_files/"* stores the final processed data in *"data/parquet_files/."*

As mentioned above, the first part of the streaming pipeline does flatten job only. It transforms the unstructured **JSON** format to a tabular form. The second job aggregates the values of the columns every *5 minutes* by taking the sum.

To run the **JSON** file generate, download the attached Python file **"main.py"** to your laptop and run the following command in the terminal:

```
python main.py
```

Please use Python 3.6+ to run the script. Python 2. is not supported.

You can terminate the application by pressing **CTRL+C**

main.py

Part 2: Batch processing

Your data team is full of art aficionados and are currently obsessed with the Met Museum collection in New York. So much so, that they want to get deep into it and find insights which interests them, and more.

The dataset they currently drool over is accessible below:

https://github.com/metmuseum/openaccess/

Please go ahead and download the dataset file: "MetObjects.csv", you will need it :)

Note: The above file is stored through Git Large File Storage (LFS), and hence might not be immediately consumable after a git clone. In that case, **please install Git LFS**, or use the **Git Web UI** and download the raw file.

Data Consumption

In this stage, you have to consume the data from the MetObject.csv file and load it into a DataFrame with a defined schema.

- During consumption of large datasets, it's very common to encounter malformed/invalid lines, however, we don't want to stop the pipeline every time that a bad line is found, we need to react to this issue by storing the malformed records in a different path without stopping the pipeline. Without implementing any code, explain how you would approach a solution to this problem.

Pre-Processing

Dimensions: The data contains a column called dimensions with the height, width and length (Ideally) of every object. This data is inconsistent though, the dimensions are not always presented in the same format. Create 3 new columns with the extracted value of height, width and length in cm. If the dimension is not available, fill the value with null.

Examples:

Input: 59 1/4 x 33 x 26 3/8 in. (150.0 x 83.8 x 67 cm)

Output:

height_col	width_col	length_col
150	83.8	67

Input: 59 1/4 x 33 in. (150.0 x 83.8 cm)

Output:

height_col	width_col	length_col
150	83.8	null

Country: The Country column is also messy, instead of having clean country names we found instances like "France or Mexico" or "France/Italy". To clean this data you have to create a new column and extract each country name and store it in an array, so in the case of "France or Mexico" the expected result is [France, Mexico]

Forward and backward fill: The constituent id is null for some of the titles. Fill the null values based on the previous (backward fill) non-null value for that respective Title, in case this doesn't exist, fill it with the next (forward fill) non-null value.

Aggregation

We want to aggregate the data and extract information about each country. For this you have to create another DataFrame with the following information:

- Get the number of artworks per individual country
- Get the number of artists per country
- Average height, width, and length per country
- Collect a unique list of constituent ids per country

Storage

The resulting tables have to be stored in an appropriate format, for example storing the data in a relational database or CSV. You must describe why you chose the storage format you have chosen (How/why is this format helpful, what are the advantages/disadvantages)

Optimization

Imagine that the volume of the data has become very large and that we're working with multiple data sources and tables, the data pipeline is starting to take longer and longer to complete and we need to optimize it. Without implementing any code, explain what approaches you would take to optimize the performance of a data pipeline.

Part 3: Dimensional Modelling

You are working on an E-commerce project where the data is available in different formats. The customer data is **CSV**; the product data is in **XML**; the transactions data is **JSON**. We mocked the data to simplify actual data that you may see in one of our projects.

1. Write an application to load and standardize the schema of the unstructured data and transform it to table format. Store the transformed data in **CSV** files. Use any programming language you like.
2. Evaluate the quality of the data.
3. Design the dimension model of the fact table and the dimension tables for storing the data in the data warehouse.
4. How are you going to partition your tables, and why?
5. Write the SQL queries to answer the following questions:
    a. How many products have been sold with profits?
    b. Calculate the total amount of profits?
    c. Which brand is performing the best?
    d. Sort the customers from the most important to the least important.
    e. Which product do the men buy the most?
    f. Is there a product that both men and women love?

customer_data.csv

products.xml

transactions.json