Junior Data Engineering Technical Assessment

Table of Contents

1. Data Assignment

DESCRIPTION

The objective of this assignment is to develop a Spark app that handles csv data, performs some cleansing and transformation steps, and finally persists the data to a database, as well as to parquet files. As general guidelines:

- Spark can be installed locally or can be used through a Docker image of your choice.
- SQL database for persisting the data can be an SQLite instance (for simplicity of deployment).
- Spark language of development can be Python (PySpark) - In Optasia we use mostly Scala for this kind of tasks, but if you are not comfortable with it, you can use Python. It does not matter for the evaluation of this task which language you will chose. Regardless of your language choice, give the necessary attention to code quality and readability.

ASSIGNMENT DETAILS

The input data is given in CSV format (the attached **data_transactions.csv** and **subscribers.csv** files).

Each row in the **data_transactions.csv** file is composed of four columns. The first column is the timestamp of a transaction, the second one is the subscriber_id, the third one is the amount of the transaction and the fourth one is the channel through which the transaction was done. To illustrate the format of the file, consider the fragment below:

```
2021-08-16 00:14:01+01,123456,0.3,SMS
2021-08-16 00:54:43+01,123451,0.15,SMS
2021-08-16 00:04:29+01,123452,0.15,SMS
2021-08-16 00:39:05+01,123453,0.15,SMS
```

Regarding the **subscribers.csv** file, it consists of two columns, the subscriber_id and the activation_date of the subscriber. To illustrate the format of the file, consider the fragment below:

```
234593,2021-08-10
234594,2021-08-10
234595,2021-08-10
```

The Spark job should first read the subscribers file, perform any cleansing if required, and push the records in the database (create a table "subscribers") with the below schema:

1. row_key: of data type *String/Varchar* and is the concatenation of "subscriber_id"_"activation_date (format yyyyMMdd)"
2. sub_id: of data type *String/Varchar* from column subscriber_id
3. act_dt: of data type *Date/Timestamp* from column activation_date

Afterwards, the transactions file should be loaded, cleansed if required, and joined with the subscribers table from the database on the column "subscriber_id" in order to retrieve the row_key. The result should be persisted as a parquet file in a target directory of your choice. The output parquet file should have the following schema:

1. timestamp: of data type *TimestampType* produced from the first column of the data_transactions.csv.
2. sub_row_key: of data type *StringType* retrieved from the SQL table "subscribers" column "row_key".
3. sub_id: of data type *StringType* retrieved from the SQL table "subscribers" as well.
4. amount: of data type *DecimalType* with four decimal digits from the third column of the data_transactions.csv.
5. channel: of data type *StringType* from the fourth column of the data_transactions.csv.
6. activation_date: of data type *StringType* (format yyyMMdd) retrieved from the SQL table "subscribers".

All in all, a spark job should be submitted that will read the input csv data and produce the parquet output and the records at database.

Main objective of the assignment ends here. Below are some optional extras.

EXTRA 1

- Automation is a Data Engineer’s best friend. It would be nice to think of some ways that would facilitate the day-to-day operation of this app. Some ideas:
    - DB Init script – If the subscribers table does not already exist (e.g. new deployment), have it be prepared automatically.
    - Retry unmatched subscribers – If we get transactions with subscribers that have not yet been received via the subscribers file (therefore, they are not present in the DB), store them aside and retry joining them on a subsequent run in case we have later received them.

EXTRA 2

- Dockerize the whole project.
- Use PostgreSQL via Docker instead of a local SQLite as a DB.

EXTRA 3

- Spin up a small Hadoop cluster using Docker.
- Store the resulting parquet files in HDFS.

DELIVERABLE

The final deliverable should include the following:

• A **private** repo in GitHub. We only need read-only access to the repo (invite as collaborator fntouskas@outlook.com).

• A **README.md** file with instructions of how to setup, build and execute the spark app. The resulting **parquet file** should also be included in the repo. Your deliverable will be examined based on the requirements of the assignment. Expect to be asked about your design and implementation choices, assumptions and issues that you may have encountered