**Breweries Test - Documentation**

**NAME:** Breweries Test - Documentation

**TYPE:** Documentation

**DESCRIPTION:** Instructions how to use

**AUTHOR:** Thiago Bezerra de sousa

**History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Comments** |
| 29/07/2024 | 1.0 | Document Created |
|  |  |  |
|  |  |  |

**Objective**

The goal of this test is consuming data from an API, transforming and persisting it into a data lake following the medallion architecture with three layers: raw data, curated data partitioned by location, and an analytical aggregated layer.

**Source**

The Source for this test is an api that can be consumed via the endpoint: <api.openbrewerydb.org/breweries> . In this api there is data about breweries of different types and different locations, the file is in json format and has the following schema:

**schema={**

**"id": "string",**

**"name": "string",**

**"brewery\_type": "string",**

**"address\_1": "string",**

**"address\_2": "string",**

**"address\_3": "string",**

**"city": "string",**

**"state\_province": "string",**

**"postal\_code": "string",**

**"country": "string",**

**"longitude": "string",**

**"latitude": "string",**

**"phone": "string",**

**"website\_url": "string",**

**"state": "string",**

**"street": "string"**

**}**

**Solution**

To complete the test, the solution was developed in 3 different steps, each step is an airflow dag.

**Extract**



This step was built to read data from the api using the python requests library, transform this data into a dataframe using and write it in “bronze” layer of the datalake, keeping the original schema and json format properties.

**Transform**



The main objective of this script is to transform and partition brewery data, organizing it in a more structured and efficient way for later analysis. Partitioning by state facilitates quick and specific queries, as well as optimizing data storage and retrieval. After these transformations, the dataframe is converted to parquet type and saved in the “silver” layer, partitioned respectively by: “dt\_process”, “country”, “state” and “city”.

**Aggregate**

The aim of this script is to aggregate brewery data that has previously been transformed and partitioned by state. It reads the partitioned data, aggregates the data by state and brewery type, and saves the result in a Parquet format optimized for analysis. This process allows the creation of a summarized data set that facilitates further analysis of the distribution and types of breweries in different states.

**Orchestration**

The process is scheduled to run once a day, starting at 6 am. The first process starts and when completed the second process starts and so on.

**Monitoring/Alerting**

In a future, I would create a task to send an email to the developers if a task failure, containing the dag id, task id and the error. This measure would make the correction happen more quickly.

Another solution would be to make a report on days when there is a problem with the API and we were unable to read its data, when the day's folder in datalake has no records, it would send an alert to the developers.