Building an ETL Pipeline with Apache Airflow

Assignemnt 3

Big Data

Authors

Florentine Popp

Clàudia Valverde

June 2024

Contents

1 Introduction **2**

2 Overview of the DAG **2**

3 Airflow Tasks **2**

3.1 Download Dataset 2

3.2 Clean Dataset 2

3.3 Transform Dataset 3

3.4 Load into MongoDB 3

3.5 Email Notification 3

4 DAG Schema **4**

5 DAG Structure and Scheduling **5**

6 Conclusions **5**

1 Introduction

This document provides an overview and explanation of an Airflow DAG designed to extract, clean, transform, and load the Online Retail dataset from the UCI Machine Learning Repository into MongoDB. The DAG also includes summary notification thorough email when the pipeline has finished.

2 Overview of the DAG

The DAG, named `OnlineRetailETL`, performs the following steps:

- Downloads the Online Retail dataset.
- Cleans the dataset.
- Transforms the dataset by adding a `TotalPrice` column.
- Loads the transformed dataset into MongoDB.
- Sends a summary report via email.

3 Airflow Tasks

In order to install the necessary packages, we added the required packages to the designated variable in `docker_compose.py`:

```
_PIP_ADDITIONAL_REQUIREMENTS: ${_PIP_ADDITIONAL_REQUIREMENTS:- ucimlrepo pymongo}
```

3.1 Download Dataset

- **Task ID:** `extract_dataset`
- **Operator:** `PythonOperator`
- **Function:** `download_online_retail_dataset`
- **Description:** This task downloads the Online Retail dataset from the UCI Machine Learning Repository using the `ucimlrepo` package. The dataset is saved locally as `X.csv`. The `PythonOperator` runs the `download_online_retail_dataset` Python function to perform this operation.

3.2 Clean Dataset

- **Task ID:** `clean_dataset`
- **Operator:** `PythonOperator`
- **Function:** `clean_online_retail_dataset`

- **Description:** This task cleans the downloaded dataset by handling missing values, removing duplicate entries, and converting data types as needed. For example, it converts the `InvoiceDate` column to a datetime format. The cleaned dataset is then saved as `X_Cleaned.csv`. The `PythonOperator` runs the `clean_online_retail_dataset` Python function to perform this cleaning process.

3.3 Transform Dataset

- **Task ID:** `transform_dataset`
- **Operator:** `PythonOperator`
- **Function:** `transform_online_retail_dataset`
- **Description:** This task transforms the cleaned dataset by adding a new column, `TotalPrice`, which is calculated as the product of the `Quantity` and `UnitPrice` columns. The transformed dataset is saved as `X_Transformed.csv`. The `PythonOperator` runs the `transform_online_retail_dataset` Python function to perform this transformation.

3.4 Load into MongoDB

- **Task ID:** `load_into_mongodb`
- **Operator:** `PythonOperator`
- **Function:** `load_into_mongodb`
- **Description:** This task loads the transformed dataset into MongoDB. It reads the transformed dataset from `X_Transformed.csv`, connects to a MongoDB instance, and inserts the data into the `transactions` collection of the `online_retail` database. The `PythonOperator` runs the `load_into_mongodb` Python function to handle this loading process. See Figure 1 where we ensure the dataframe has been uploaded into MongoDB correctly thourhg MongoCompass.

3.5 Email Notification

- **Task ID:** `send_email`
- **Operator:** `EmailOperator`
- **Description:** The email notification setup in the Airflow DAG involves three main steps. First, it retrieves a list of email addresses from an Airflow Variable named `emails`. This variable had been previously declared as seen in Figure 2.

  Next, the email content is defined in an HTML string. This string includes sections such as "DAG Details," "Task Status," and "Links" to provide a comprehensive summary of the DAG run. The content uses Jinja2 template syntax to dynamically insert information like the DAG run ID, execution date, and task instance details.

  Finally, an `EmailOperator` is used to send the email. This operator is configured with a task ID of 'send_email', the list of email recipients, the subject "Airflow DAG Summary," and the previously defined HTML content. The `dag` parameter links this operator to the defined DAG, ensuring that the email is sent as part of

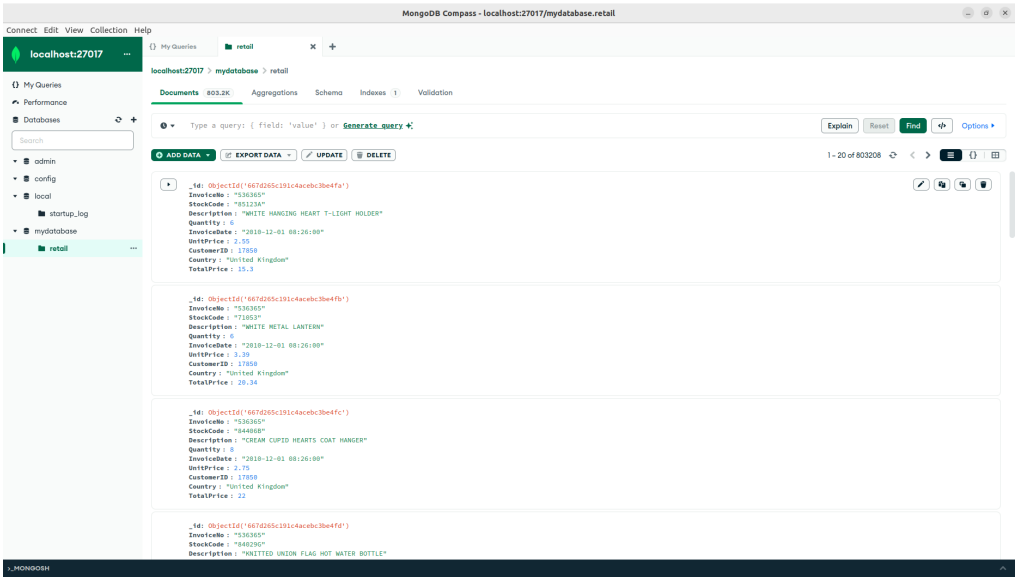

Figure 1: Data loaded into MongoDB

Figure 2: Declaring Email Variable

the DAG's execution process. This setup ensures that recipients receive a detailed and formatted summary of the latest DAG run, including task statuses, start and end times, and relevant links.

Unfortunately althougth we have done all the code and necessary steps for an email notification we have not been able to fully execute the extra exercise, as we encounter an error of connection.

4 DAG Schema

The DAG (Directed Acyclic Graph) diagram represents the workflow of the `OnlineRetailETL` process, showcasing the sequence of tasks and their dependencies. Each node in the diagram corresponds to a task, and the directed edges between nodes indicate the order in which tasks are executed.

The dependencies between tasks ensure that they are executed in the correct order. For example, `transform_dataset` cannot start until `extract_dataset` completes, and `load_dataset` must wait for `transform_dataset` to finish. These dependencies are visualized as directed edges in the DAG diagram 3, forming a clear path from the start of the ETL process to its completion.

The DAG is scheduled to run at specific intervals, as defined in the `schedule_interval` parameter. This ensures that the ETL process is executed regularly, keeping the data warehouse up-to-date with the latest information. When all the functionalities have been done the idea would be to send a summary notification of the overall pipeline performance thorugh email as `send_email` task speicifies.

The DAG diagram provides a clear visual representation of the ETL workflow, highlighting the sequential and dependent nature of the tasks involved.

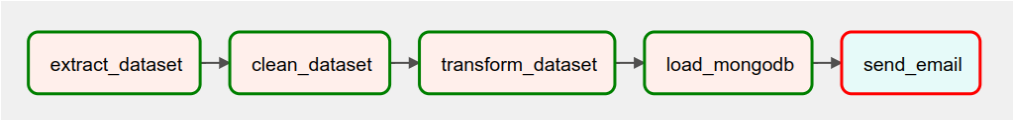

Figure 3: DAG Diagram

5 DAG Structure and Scheduling

- **DAG ID**: `OnlineRetailETL`
- **Description**: A DAG to extract, clean, transform, and load the Online Retail dataset from the UCI Machine Learning Repository.
- **Schedule Interval**: Every day.
- **Default Arguments**:
  - `owner`: `airflow`
  - `start_date`: One day ago (`airflow.utils.dates.days_ago(1)`)
  - `depends_on_past`: `False`
  - `email_on_failure`: `False`
  - `email_on_retry`: `False`
  - `email`: `email_list`
  - `retries`: `1`
  - `retry_delay`: 5 minutes

6 Conclusions

The `OnlineRetailETL` DAG provides a comprehensive and efficient solution for extracting, transforming, and loading online retail data into a data warehouse. The DAG is designed with clear task dependencies, ensuring that each step of the ETL process is executed in the correct order. The key components of the DAG include tasks for data extraction, cleaning, transformation, loading and a task for summary notification.

Throughout the development and deployment of the DAG, several important aspects were emphasized:

Each task is modular and clearly defined, which simplifies maintenance and debugging. The use of Airflow's PythonOperator allows for flexibility in task implementation. At the end an email an email containing the summary of the whole DAG procedure is

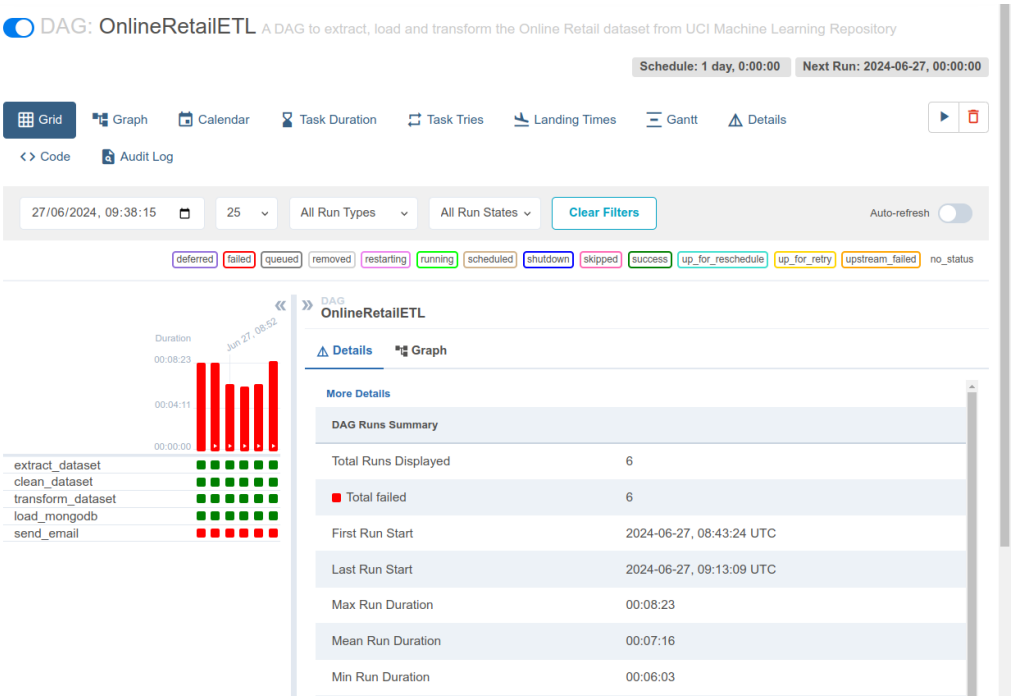

Figure 4: Airflow Pipeline execution

supposed to be sent. Unfortunately, we were not able to properly execute this task due to a connection error.

The DAG is scheduled to run at regular intervals, ensuring continuous data updates. Airflow's scheduling capabilities allow the ETL process to scale with increasing data volumes and complexity.

By implementing this DAG, the ETL process for online retail data was automated, leading to more efficient data management and timely insights. The DAG's design not only ensures data integrity and quality but also provides a foundation for further enhancements and scalability. Overall, the `OnlineRetailETL` DAG is a crucial tool for leveraging data to drive business decisions and improve operational efficiency.