Data Warehouse Documentation

# Overview

This document outlines the schema of the data warehouse, the ETL pipeline, and provides recommendations for further optimizations and improvements to ensure the system is scalable, maintainable, and robust.

# Data Warehouse Schema

## Tables and Schemas

The retail\_schema consists of the following tables:

1. **customers**
   * CustomerID (Text, Primary Key)
   * Country (Text)
2. **products**
   * StockCode (Text, Primary Key)
   * Description (Text)
   * UnitPrice (Float)
3. **sales\_transactions**
   * InvoiceNo (Text, Primary Key)
   * StockCode (Text)
   * Quantity (Int8)
   * InvoiceDate (Timestamp)
   * CustomerID (Text)
4. **raw\_online\_retail**
   * InvoiceNo (Text, Primary Key)
   * StockCode (Text)
   * Description (Text)
   * Quantity (Int8)
   * InvoiceDate (Timestamp)
   * UnitPrice (Float)
   * CustomerID (Text)
   * Country (Text)

# ETL Pipeline

## Overview

The ETL (Extract, Transform, Load) pipeline is responsible for ingesting data from various sources, transforming it into the required format, and loading it into the data warehouse.

## Components

1. **Configuration File**
   * The config.py file is used to configure the pipeline, including database information, file paths, and other parameters.
   * Sensitive information is stored in a .env file, which should be encrypted using encrypt\_env.py and then decrypted when setting environment variables.
   * Each file is used as follows:
     1. config.yml: contains the pipeline.
     2. helpers/crypto: contains the cryptographic package.
     3. db\_helpers.py: all the functions related to the database.
     4. settings.py: setting up the database connection and environment.
     5. steps.py: contains the pipeline steps as a class (you can add steps as classes).
     6. utils.py: helpers that are stand-alone.
     7. etl.py: the script that can be added to the dag.
2. **Data Ingestion**
   * Data is ingested from multiple sources, including CSV files, SQL databases, and potentially APIs.
   * Faster ingestion techniques such as bulk writes are recommended to improve performance.
3. **Transformation**
   * Data is transformed into the appropriate format as per the schema requirements.
   * This includes cleaning, aggregating, and enriching the data.
4. **Loading**
   * Transformed data is loaded into the data warehouse.
5. **Scheduling and Monitoring**
   * Airflow is recommended for scheduling the ETL jobs and handling workflows.
   * Airflow can also send alerts and emails on errors, ensuring that issues are promptly addressed.

### Recommendations for Further Optimizations and Improvements

1. **Dynamic Database Creation**
   * Implement dynamic database creation to support various clients and environments seamlessly.
2. **Bulk Writes**
   * Utilize bulk write operations to improve data ingestion speed and efficiency.
3. **Additional Data Sources**
   * Expand data source options to include API calls, enhancing the pipeline's versatility.
4. **Multi-Database Support**
   * Handle multiple database types (e.g., PostgreSQL, MySQL, SQLite) to cater to different use cases and client requirements.
5. **Advanced ORM Features**
   * Leverage advanced features of ORM frameworks to optimize database queries and interactions.
6. **Enhanced Monitoring and Alerts with Airflow**
   * Implement more sophisticated monitoring and alerting mechanisms within Airflow for better operational oversight.
   * Use Airflow for scheduling and error handling, including sending email notifications on errors (alerts).
7. **Data Quality Monitoring**
   * Implement unit tests for each step of the pipeline to ensure data integrity and correctness.
   * Use comprehensive logging to track the pipeline’s activities and errors.
   * Employ bug catching mechanisms to handle and log errors for better debugging.
8. **Version Control with DVC**
   * Utilize DVC (Data Version Control) to manage versions of datasets and models, ensuring reproducibility and traceability.

### Conclusion

The outlined approach provides a robust framework for building and managing a scalable and maintainable data warehouse. By following these recommendations and leveraging the proposed technologies, the system can efficiently handle growing data volumes and evolving business requirements.