ELT Pipeline for Currency Exchange Data

Author: Bhavani Kishore
Date: October 6, 2025

1. Project Overview

This project implements an Extract, Load, Transform (ELT) pipeline for fetching, storing, and processing currency exchange data. The pipeline retrieves historical and real-time currency exchange rates from the CurrencyLayer API, loads the raw data into a structured format, applies data quality checks, and stores the cleaned data in a MySQL database. This enables further analysis, dashboarding, and data-driven decision-making related to currency markets.

1.1. Core Functionality

The project is designed to perform the following key functions:

- **Data Extraction:** Fetches both historical and live currency exchange rates from the CurrencyLayer API.
- **Data Storage:** Stores the extracted data in both JSON and CSV formats for archival and processing.
- **Data Loading:** Loads the structured data into a MySQL database.
- **Data Quality:** Implements a series of data quality checks to ensure the integrity and consistency of the data before it is loaded into the database.
- **Automation:** The pipeline is designed to run continuously, fetching live data at regular intervals.

1.2. Project Structure

The project is organized into three main Python scripts:

- `app.py` : The main application script that orchestrates the entire ELT process.
- `dq.py` : A module that contains functions for performing data quality checks.
- `tables.py` : A module for handling all database-related operations, including connection, table creation, and data insertion.

2. Technical Architecture

The ELT pipeline follows a modular architecture, with distinct components for each stage of the process. This separation of concerns makes the system more maintainable and scalable.

2.1. Data Flow

The data flows through the system as follows:

1. **Extraction:** The `app.py` script makes API calls to the CurrencyLayer service to fetch currency exchange data.
2. **Staging:** The raw JSON response from the API is converted into a Pandas DataFrame and saved to both JSON and CSV files in a structured directory format.
3. **Data Quality:** The `dq.py` module reads the staged CSV files and applies a series of data quality rules to clean and validate the data.
4. **Loading:** The `tables.py` module takes the cleaned DataFrame and inserts it into a MySQL database.

2.2. Key Technologies

The project leverages the following key technologies:

- **Python:** The core programming language for the application.
- **Pandas:** Used for data manipulation and analysis.
- **Requests:** For making HTTP requests to the CurrencyLayer API.
- **MySQL:** The relational database used for storing the processed data.
- **`mysql-connector-python`:** The Python driver for connecting to the MySQL database.

3. Code Description

This section provides a detailed description of each of the Python scripts in the project.

3.1. `app.py` - The Main Application

This script serves as the entry point and orchestrator of the ELT pipeline. It imports the `tables` and `dq` modules to perform its functions.

Key Responsibilities:

- **Configuration:** Loads the application configuration from a `config.json` file, which contains API keys, URLs, and file paths.
- **Historical Data Extraction:** Fetches historical data from the CurrencyLayer API, saves it to JSON and CSV files, and then loads it into the database.
- **Live Data Extraction:** Enters an infinite loop to continuously fetch live data from the API at 10-second intervals.

- **Data Orchestration:** For each batch of live data, it iterates through the generated CSV files, applies data quality rules using the `dq.py` module, and then inserts the cleaned data into the database using the `tables.py` module.

3.2. `dq.py` - Data Quality Module

This module is responsible for ensuring the quality and integrity of the data before it is loaded into the database. It contains a suite of functions that perform various validation and cleaning tasks.

Key Functions:

- **`creating_log_and_error_file()` :** Creates log and error files for each run to record data quality issues.
- **`bool_check_and_drop()` :** Checks if a column contains valid boolean values and drops rows that do not.
- **`terms_url_check()` and `privacy_url_check()` :** Validates that the `terms` and `privacy` columns contain valid URLs, and if not, updates them with default values.
- **`timestamp_check_and_drop()` :** Ensures that the `timestamp` column contains valid integer values.
- **`source_check_and_drop()` :** Verifies that the `source` column contains the expected value ("USD").
- **`code_check_and_drop()` :** Validates that the currency codes are valid 3-letter codes.
- **`rate_check_and_update()` :** Checks that the exchange rates are positive floating-point numbers and replaces invalid rates with `NaN` .
- **`applying_dq_rules()` :** A wrapper function that applies all the data quality rules to a given DataFrame.

3.3. `tables.py` - Database Module

This module encapsulates all interactions with the MySQL database. It provides a clean and reusable interface for connecting to the database, creating tables, and inserting data.

Key Functions:

- **`db_connect()` :** Establishes a connection to the MySQL database using the credentials from the `config.json` file.
- **`create_table()` :** Dynamically creates a new table in the database based on the columns of a Pandas DataFrame. It also adds several metadata columns for auditing purposes.
- **`insert_hist_data()` :** Inserts historical data into the specified table.

- **`insert_live_data()` :** Inserts live data into the specified table.

4. Setup and Execution

To run this project, you will need to have Python 3, Pandas, Requests, and `mysql-connector-python` installed. You will also need a MySQL database and a `config.json` file with the appropriate API keys and database credentials.

Once the environment is set up, you can run the application with the following command:

```bash
python3 app.py
```

5. Conclusion

This ELT pipeline provides a robust and automated solution for collecting and processing currency exchange data. The modular design and comprehensive data quality checks ensure that the data stored in the database is accurate and reliable. This project can be extended to include more data sources, advanced analytics, and visualization dashboards to provide deeper insights into the currency market.