**Documentation**

**Overview**

This script fetches and processes exchange rate data for a specified currency pair over a given period, providing key insights such as the best, worst, and average exchange rates. It also includes visualization of the exchange rates over time. The script incorporates error handling, logging, caching for flexibility and robustness.

**Approach**

1. **Fetch Exchange Rates**: Connect to the Exchange Rates API to retrieve exchange rate data for a specified date range.
2. **Preprocess Data**: Convert the API response to a pandas DataFrame, ensuring proper formatting and sorting.
3. **Analyze Data**: Calculate the best, worst, and average exchange rates for the given period.
4. **Visualization**: Plot the exchange rates over time using matplotlib.
5. **Error Handling**: Handle various potential errors such as network issues, invalid inputs, and missing data.
6. **Caching**: Implement caching to avoid redundant API calls within a short period.

**Architecture**

* **API Layer**: Handles API requests and responses.
* **Data Processing Layer**: Converts and preprocesses data for analysis.
* **Analysis Layer**: Performs calculations to find key statistics (best, worst, and average rates).
* **Visualization Layer**: Plots the exchange rate data.
* **Error Handling and Logging**: Captures and logs errors for troubleshooting.
* **Caching**: Stores API responses temporarily to optimize performance.

**Best Practices Followed**

1. **Modular Code**: The script is divided into distinct functions for fetching, preprocessing, and analysing data, making it easier to maintain and extend.
2. **Error Handling**: Comprehensive error handling is implemented to manage different failure scenarios gracefully.
3. **Logging**: Detailed logging provides visibility into the script's operations and helps in debugging.
4. **Caching**: Caching improves performance by reducing redundant API calls.
5. **Documentation**: Clear and concise documentation helps in understanding the script's functionality and usage.
6. **Testing**: Unit tests ensure the script handles various scenarios correctly.
7. **Code Readability**: Code is written in a clear and consistent style, following PEP 8 guidelines where applicable.

**Scaling:**

1. A config file or a meta file/table can be used to include any number of currency converters, we can include the number of days, a flag to run it or not.
2. Create a dimensional model with dimensions Currency Dimension and Date Dimension, Fact table called Exchange Rate Fact and include any number of currency conversion and any number of days.
3. Example Schema

 **Fact Table**:

* **Exchange Rate Fact Table**: This table contains measures related to exchange rates and is typically grain-defined per currency pair per day.
  + DateKey: Foreign key to the Date dimension.
  + FromCurrencyKey: Foreign key to the Currency dimension (currency being converted from).
  + ToCurrencyKey: Foreign key to the Currency dimension (currency being converted to).
  + ExchangeRate: The exchange rate value.
  + AverageExchangeRate: Average exchange rate for the period (if applicable).
  + HighExchangeRate: Highest exchange rate for the period (if applicable).
  + LowExchangeRate: Lowest exchange rate for the period (if applicable).

 **Dimension Tables**:

* **Date Dimension**: Contains attributes related to date.
  + DateKey: Primary key.
  + Date: Actual date.
  + Year, Quarter, Month, Day, etc.: Various attributes for filtering and grouping data.
* **Currency Dimension**: Contains attributes related to currencies.
  + CurrencyKey: Primary key.
  + CurrencyCode: ISO code for the currency (e.g., USD, EUR).
  + CurrencyName: Full name of the currency (e.g., US Dollar, Euro).
  + Country: Country or region using the currency.
  + Symbol: Currency symbol (e.g., $, €).