**Development Diary: Data Handling and Real-Time Processing**

**1. data\_handler.py**

* **Role**: This module is responsible for handling both real-time and historical data. It focuses on rescaling, saving, and organizing data for various assets and intervals.
* **Core Functions**:
  + **Data Rescaling**: Applies scalers to important fields such as open, close, high, and low prices using predefined .pkl scaler files.
  + **Saving Real-Time Data**: Saves fetched real-time data into designated file paths, categorized into raw, processed, or rescaled formats.
  + **Fetch Time Logging**: Maintains a log of the last time data was fetched, preventing data duplication during real-time fetching processes.
  + **Data Transfer**: Manages the transfer of older data to different storage locations to keep the working set organized and current.
* **Key Insights**: This file enables a structured approach to saving and rescaling data, ensuring that all data is categorized into various stages for easier access and further processing.

**2. historical\_data\_handler.py**

* **Role**: This script is dedicated to managing and processing historical data for different assets and time intervals. It ensures that the historical data is correctly formatted, cleaned, and optionally rescaled.
* **Core Functions**:
  + **Loading Configuration Parameters**: Reads cleaning and checking parameters from external JSON files to clean and validate the data.
  + **Data Fetching**: Retrieves historical data chunks based on specified start and end dates for a symbol. Options include fetching raw, cleaned, or rescaled data.
  + **File Path Construction**: Builds file paths dynamically for storing raw, processed, or rescaled historical data files, ensuring data is organized by symbol, interval, and date range.
  + **Batch Processing**: Supports efficient data processing by handling data in chunks, making it ideal for large historical datasets.
  + **Cleaning and Rescaling**: Provides the ability to clean the data using custom rules and apply scalers to standardize it for machine learning models.
* **Key Insights**: This file is essential for processing historical data. It serves as the backbone for preparing clean and structured historical data to be used in training or backtesting.

**3. real\_time\_data\_handler.py**

* **Role**: The core component for real-time data fetching and processing, this script manages the retrieval, cleaning, and saving of real-time data in a way that integrates seamlessly with your trading system.
* **Core Functions**:
  + **Real-Time Data Fetching**: Retrieves live data for multiple symbols at specified intervals, ensuring that the most recent data is available for trading and decision-making.
  + **Log Management**: Tracks the time of the last data fetch and logs any errors or warnings encountered during the fetching or saving processes.
  + **Saving Real-Time Data**: Supports saving fetched data in various formats (raw, processed, rescaled), organizing it in a structured folder system for easy access.
  + **Scheduling Data Fetches**: Implements a time-based scheduling mechanism that calculates the next fetch time and sleeps until that time arrives, ensuring data is fetched consistently.
* **Key Insights**: This module automates the real-time data pipeline, allowing for smooth integration of live data into your system while maintaining the ability to validate and process it.

**Project Integration Insights:**

These three modules (data\_handler.py, historical\_data\_handler.py, and real\_time\_data\_handler.py) are central to the data pipeline of your trading bot. They ensure that data, whether historical or real-time, is fetched, cleaned, rescaled, and saved in an organized manner. Together, they enable the following workflow:

1. **Data Fetching**:
   * Historical data is fetched in chunks, cleaned, and rescaled using historical\_data\_handler.py.
   * Real-time data is fetched continuously and saved in a structured format using real\_time\_data\_handler.py.
2. **Data Processing**:
   * The data\_handler.py module applies scalers and handles different stages of data (raw, processed, rescaled), ensuring it is ready for machine learning models or trading algorithms.
3. **Data Storage**:
   * Both historical and real-time data are stored in dedicated folder structures, making it easy to retrieve and analyze data as required.

By combining historical and real-time data handling, these modules provide a robust system for keeping your models up-to-date and ensuring real-time decisions are based on the latest available information.