ETL and Analysis Pipeline
BDA Mid Term

Purpose: This code defines an ETL (Extract, Transform, Load) pipeline to collect, clean, and analyze financial stock data from two sources: MarketStack API and a Kaggle dataset. It then loads the processed data into MongoDB for further analysis and visualization.

Functionality:

1. **Data Collection:**
    - `ETLPipeline` class is responsible for fetching data from both sources.
    - `get_stock_data_from_marketstack` fetches data from MarketStack API (or a mock source).
    - `get_data_from_kaggle_df` loads data from a Kaggle dataset using KaggleHub.
2. **Data Transformation:**
    - `standardize_timestamps` converts date columns to a consistent format.
    - `normalize_column_names` standardizes column names for uniformity.
    - `handle_missing_values` addresses missing data by filling or deriving values.
    - `validate_data` cleans the data by removing rows with invalid financial values.
    - `add_features` calculates new features like daily return and volatility.
    - `aggregate_data` groups data by ticker and date, calculating aggregate metrics.
    - `merge_datasets` combines data from both sources, handling potential duplicates.
3. **Data Loading:**
    - `load_data` loads the final processed data into a MongoDB database for persistence and further analysis.
4. **Data Visualization:**
    - `plot_daily_return_distribution`, `plot_stock_price_vs_volume`, `plot_volatility_over_time`, `plot_average_daily_return`, `plot_stock_prices_over_time` are included for preliminary visualizations of the transformed data. These generate plots for daily return distribution, stock price vs. volume, volatility over time, average daily return by stock, and stock prices over time.

Structure:

- The core logic is encapsulated within the `ETLPipeline` class.
- Data fetching, transformation, and loading are handled by separate methods within the class.
- Data visualization functions are provided to offer initial insights.

Usage:

content_copy

```
# Create pipeline instance
data_collection = ETLPipeline()
# Run the transformation pipeline
df = data_collection.run_transformation()
# Load data to MongoDB
# data_collection.load_data(df)
# Generate some plots
data_collection.plot_daily_return_distribution(df)
```

Use code with caution

Key Improvements:

- **Data Quality:** The pipeline addresses data inconsistencies and errors, leading to improved reliability for analysis.
- **Feature Engineering:** Calculated features enhance the dataset for deeper insights.
- **Data Integration:** Combining data from multiple sources expands the scope of analysis.
- **Scalability:** Using MongoDB for storage allows for handling larger datasets and future expansion.
- **Reproducibility:** The code is organized into a pipeline, making it easier to understand, maintain, and modify.

Further Development:

- Explore more advanced visualization techniques.
- Integrate with other data sources for comprehensive analysis.
- Develop machine learning models using the cleaned dataset.
- Implement data validation checks for real-time data feeds.
- Build an interactive dashboard for monitoring and analysis.