In this project, a complete ETL pipeline has been implemented using Python and SQL Server. This process includes initial table cleanup, data loading, data processing and transformation, and storing the output in Parquet format. The main objective of this system is to provide a platform for analyzing user-related data on the platform.

1. **Dropping Existing Tables from the Database**
   At the beginning of the script execution, a class named DropTablesCleaner is defined, which checks whether the tables user_profiles, user_events, and user_activity exist in the database. If they do, they are dropped using the IF OBJECT_ID ... DROP TABLE command. This operation prevents data duplication and ensures a clean start for each pipeline run.
2. **Loading User Profile Data from CSV File**
   The UserProfilesLoader class is responsible for loading user information:

- First, it creates the user_profiles table with the specified structure (user_id, name, registration_date, location) if it does not already exist.
- Then, it reads the CSV file and inserts each row into the table after validation and cleaning.
- The number of successful and error rows are reported separately.

3. **Loading User Event Data from JSON File**
   The UserEventsLoader class handles loading user behavioral data:

- It first creates the user_events table with an appropriate structure if it does not exist. This table includes various fields related to user interactions with the system.
- The JSON file is read, and each event's timestamp is converted using datetime.fromisoformat before insertion.
- The internal structure of the details field is also extracted and inserted as separate parameters into the table.
- The total number of inserted events is displayed at the end.

4. **Joining Tables and Creating the Final Analytical Table**
   The function join_and_save performs a JOIN between user_profiles and

user_events tables, creating a new table called user_activity that contains combined user and behavior information:

- In addition to existing fields, a new column named event_date is extracted from the timestamp.
- This table serves as a foundation for analyzing user behavior and building analytical dashboards.

5. **Exporting Data to Parquet Format**
   At the end of the program, data from the user_activity table is read from the database using the pandas library and SQLAlchemy connection, then saved as Parquet files.

- For optimized storage, the partition_cols = ["event_date"] option is used to partition data by date.
- This format supports usage in analytical tools and data processing platforms such as Hadoop or Spark.

6. **Sample Data Export to Excel File user_activity_sample.xlsx**
   At the end of the program, data from the user_activity table is also exported as a sample Excel file using pandas and SQLAlchemy.

Summary

This script is designed to run fully and independently, covering the following steps:

1. Cleaning up existing tables (if any)
2. Creating necessary database structures
3. Loading data from CSV and JSON files
4. Joining and processing data to produce an analytical view
5. Saving output as time-partitioned Parquet files