**Implementation Note: ETL Pipeline for Property Data**

**Objective**

The project implements an ETL (Extract, Transform, Load) pipeline to process property data from an API, clean and structure it into a dimensional model, and load it into a PostgreSQL database.

**Implementation Details**

**1. Data Extraction (Extract)**

* **API Call (Commented Out in the Code)**
  + Used the requests library to fetch property data from the RentCast API.
  + API response was stored in a JSON file (PropertyRecords.json) for further processing.
* **Reading Data from File**
  + The JSON file was loaded into a Pandas DataFrame (Propertyrecord\_df).
  + Nested features data was converted to JSON format for proper storage.

**2. Data Transformation (Clean & Prepare)**

* **Handling Missing Values**
  + fillna() method was used to replace NaN values with appropriate defaults.
  + Example: Missing bathrooms, bedrooms, squareFootage were set to 0, while missing addresses were replaced with "unknown".
* **Dimensional Model Construction**
  + Extracted relevant attributes into separate dimension tables and a fact table:
    - **Location Dimension (location\_dim)**: Contains details like city, state, zip code, and geolocation coordinates.
    - **Sales Dimension (sales\_dim)**: Stores sale price and last sale date of properties.
    - **Features Dimension (features\_dim)**: Includes property type, zoning, bedrooms, bathrooms, and other characteristics.
    - **Fact Table (fact\_table)**: Contains property IDs to link other dimensions.
* **Saved Data to CSV Files**
  + Each dimensions and fact table was saved as a CSV file for database insertion.

**3. Data Loading (Load)**

* **Database Connection Setup**
  + Created a get\_db\_connection() function to establish a connection to PostgreSQL using psycopg2.
* **Schema and Table Creation**
  + A function create\_tables() was implemented to:
    - Create schema zapbank.
    - Drop existing tables to avoid duplication.
    - Create tables for location, sales, features, and fact data with appropriate constraints and foreign keys.
* **Loading Data into Database**
  + Implemented functions to read CSV files and insert data into PostgreSQL:
    - load\_data\_from\_csv\_to\_table(): Generic function for loading CSV files into tables.
    - load\_data\_from\_csv\_to\_sales\_table(): Specifically handles date parsing and type conversion for sales data.
* **Inserted Data into PostgreSQL**
  + Called loading functions for each table to transfer data from CSV files to the database.

**Outcome**

* The property data has been successfully extracted, cleaned, structured into a dimensional model, and loaded into PostgreSQL.
* The script ensures proper handling of missing values, data integrity with foreign key constraints, and optimized data storage.

**Next Steps**

* **Schedule ETL Pipeline**: Automate script execution via Windows Task scheduler.