**Project 3: Automating an ETL with Python Automate data tasks.**

**1. Introduction**

In this project, the goal is to automate the process of preparing and loading customer support data into a PostgreSQL database using Python. By building a functional ETL (Extract, Transform, Load) pipeline, we solve the problem of manual data cleaning and formatting, enabling future analysis of customer interactions, satisfaction, and support efficiency.

This automation reduces the time and errors involved in handling raw CSV files, and creates a reliable, repeatable process that prepares data for further reporting and analytics.

**2. Dataset description**

The dataset is a CSV file named Customer\_support\_data.csv, which contains over 85,000 records of customer service interactions. The fields include:

* Contact information (agent, channel, supervisor)
* Support categories and subcategories
* Timestamps of when issues were reported and responded to
* Customer satisfaction (CSAT) scores
* Product and order details (some incomplete or null)

The dataset is assumed to come from a customer support system (e.g., CRM or call center platform).

**3. Tools used**

* **Python**: For automation of the ETL pipeline (using pandas, datetime, sqlalchemy)
* **PostgreSQL**: As the target database for the cleaned data
* **Jupyter Notebook**: For development and testing of the pipeline
* **pgAdmin**: For managing the PostgreSQL database
* **Excel** (optionally): For inspecting and cleaning CSV data manually

**4. Development (code)**

The ETL pipeline follows three main steps:

1. **Extract**

* Load the Customer\_support\_data.csv file using pandas.

1. **Transform**

* Rename columns for consistency (snake\_case naming).
* Parse and normalize date fields (issue\_reported\_at, issue\_responded, survey\_response\_date) as datetime objects.
* Drop columns with excessive null values (e.g., order\_date\_time, product\_category).
* Calculate response time in minutes from issue\_reported\_at to issue\_responded.
* Add a timestamp column (etl\_loaded\_at) to track when the data was loaded.

1. **Load**

* Use SQLAlchemy to connect to a local PostgreSQL database.
* Load the final DataFrame into a table called customer\_support\_cleaned.

**5. Results or conclusions**

* Successfully automated the cleaning and structuring of a large, semi-structured dataset.
* Ensured date consistency and eliminated problematic records using error handling in pandas.
* Created a PostgreSQL table ready for analysis and reporting.
* Improved performance and reproducibility of a previously manual process.
* Gained hands-on experience with pandas, data transformation, and loading into relational databases using Python.

This project demonstrates how Python can be used to streamline real-world data preparation workflows, making the data reliable, consistent, and ready for business analysis or visualization.

*README*

***Automating an ETL with Python***

***Introduction***

*This project automates the process of preparing and loading customer support data into a PostgreSQL database using Python. It addresses the problem of inconsistent, manual data cleaning by building a functional ETL (Extract, Transform, Load) pipeline that prepares the dataset for analysis and reporting.*

***Dataset Description***

*The dataset used is `Customer\_support\_data.csv`, which contains over 85,000 records of customer service interactions. It includes:*

*- Contact channels and categories*

*- Timestamps for reported and resolved issues*

*- Agent and supervisor details*

*- Customer satisfaction (CSAT) scores*

*- Product and order-related fields*

*Some fields contain a high percentage of missing data and were handled or dropped during transformation.*

***Tools Used***

*- Python (pandas, datetime, sqlalchemy)*

*- PostgreSQL*

*- pgAdmin*

*- Jupyter Notebook*

*- Excel (for initial inspection)*

***Development Process***

***1.Extract***

*Data is read from a raw `.csv` file using `pandas`.*

***2.Transform***

*- Column names are normalized.*

*- Date fields are parsed and standardized.*

*- New metrics (e.g., response time) are calculated.*

*- Irrelevant or incomplete fields are removed.*

*- ETL load timestamp is added.*

***3. Load***

*Transformed data is loaded into a PostgreSQL table called `customer\_support\_cleaned` using SQLAlchemy.*

***Results and Conclusions***

*- A complete ETL pipeline was built and tested using real-world customer support data.*

*- The process ensures data consistency, type handling, and structure ready for analysis.*

*- Experience was gained in automating data workflows using Python and PostgreSQL integration.*

*This ETL pipeline serves as a reusable structure for automating future data loading and preprocessing tasks in customer service analytics.*

*Author*

*Monica Prieto — Data Engineer*