**Project 1: CSV File ETL Read, cleanse, and load data into SQL.**

**Introduction**  
Read data from a CSV file, transform it (clean and enhance the data), and load it into a SQL database.

**Objective project**

Read the CSV file in Python.

Clean and transform:

Convert date fields.

Create new columns if necessary (e.g., actual vs. scheduled time difference).

Handle null values.

Load the cleaned dataset into an SQL database.

Validate that the load was successful.

**Dataset description**

The "Flight Delays" dataset contains 1,048,575 commercial flight records, each detailing information on departure and arrival times, delays, cancellations, diversions, and aircraft technical characteristics.

This dataset is ideal for simulating extraction, cleansing, transformation, and loading (ETL) processes. It reflects common situations when handling large volumes of real-world data, including missing data and the need for normalization.

**Main Columns**

|  |  |
| --- | --- |
| **Column** | **Description** |
| FlightID | Unique flight identifier. |
| Airline | Airline name. |
| FlightNumber | Flight number. |
| Origin | Origin airport code (IATA). |
| Destination | Destination airport code (IATA). |
| ScheduledDeparture | Scheduled departure time. |
| ActualDeparture | Actual departure time. |
| ScheduledArrival | Scheduled arrival time. |
| ActualArrival | Actual arrival time. |
| DelayMinutes | Minutes of delay (can be negative if the flight arrived early). |
| DelayReason | Primary reason for the delay (e.g., "Weather," "Air Traffic Control"). Can contain null values. |
| Cancelled | Boolean indicator if the flight was canceled. |
| Diverted | Boolean indicator if the flight was diverted. |
| AircraftType | Aircraft type (e.g., Boeing 737, Airbus A320). |
| TailNumber | Aircraft registration number. |
| Distance | Total flight distance (in miles). |

**Relevant dataset features**

Date format: The date columns are in text (object) format and must be converted to date (datetime) to facilitate temporal analysis.

Null values: The DelayReason column has missing data, providing an opportunity to practice imputation or data cleaning.

Boolean values: The Cancelled and Diverted columns are in Boolean (True/False) format, ideal for coding and transformation practice.

Data volume: The file size (over 1 million records) allows for simulating work with medium-to-large data, common in real-world data engineering projects.

**Tools used:**

Python - Libraries: pandas, sqlalchemy, psycopg2.

Local database (MySQL)

Jupyter Notebook for documentation.

**Development (code)**

1. Extract

Load the CSV file using pandas.

Display basic statistics and check for data anomalies.

2. Transform

Convert string dates to datetime objects.

Create a new column: TotalDelay (difference between Actual and Scheduled times, if necessary).

Handle missing values in the DelayReason column.

Ensure consistent formatting across all fields.

3. Load

Create a connection to an SQLite database using sqlalchemy.

Insert the cleaned dataset into a new table named flights.

Verify that the data has been correctly inserted into the database.

**Results or conclusions**

Clean and structured flight data stored in a SQL database.

Ready for querying, reporting, or advanced analysis.

The process reflects a realistic ETL pipeline suitable for portfolio presentation.

**README**

***Flight Delays ETL Project***

***Project Overview***

*This project simulates a real-world data engineering workflow by performing an ETL (Extract, Transform, Load) process on a large dataset containing commercial flight information.*

*The main goal is to clean and enrich raw data and then load it into a SQL database for future querying and analysis.*

***Objectives***

*Extract flight delay data from a CSV file.*

*Clean, transform, and enhance the dataset (e.g., date conversions, handling missing values).*

*Load the processed data into a SQL database (SQLite).*

*Validate the data loading process.*

***Tools and Technologies***

*Python (pandas, sqlalchemy, sqlite3)*

*Jupyter Notebook*

*MySQL (local database)*

*GitHub*

***Dataset Description***

|  |  |
| --- | --- |
| ***Column*** | ***Description*** |
| *FlightID* | *Unique flight identifier.* |
| *Airline* | *Airline name.* |
| *FlightNumber* | *Flight number.* |
| *Origin* | *Origin airport code (IATA).* |
| *Destination* | *Destination airport code (IATA).* |
| *ScheduledDeparture* | *Scheduled departure time.* |
| *ActualDeparture* | *Actual departure time.* |
| *ScheduledArrival* | *Scheduled arrival time.* |
| *ActualArrival* | *Actual arrival time.* |
| *DelayMinutes* | *Minutes of delay (can be negative if the flight arrived early).* |
| *DelayReason* | *Primary reason for the delay (e.g., "Weather," "Air Traffic Control"). Can contain null values.* |
| *Cancelled* | *Boolean indicator if the flight was canceled.* |
| *Diverted* | *Boolean indicator if the flight was diverted.* |
| *AircraftType* | *Aircraft type (e.g., Boeing 737, Airbus A320).* |
| *TailNumber* | *Aircraft registration number.* |
| *Distance* | *Total flight distance (in miles).* |

***ETL Process Details***

*1. Extract*

*Load the CSV file using pandas.*

*Display basic statistics and check for data anomalies.*

*2. Transform*

*Convert string dates to datetime objects.*

*Create a new column: TotalDelay (difference between Actual and Scheduled times, if necessary).*

*Handle missing values in the DelayReason column.*

*Ensure consistent formatting across all fields.*

*3. Load*

*Create a connection to an SQLite database using sqlalchemy.*

*Insert the cleaned dataset into a new table named flights.*

*Verify that the data has been correctly inserted into the database.*

***Results***

*Clean and structured flight data stored in a SQL database.*

*Ready for querying, reporting, or advanced analysis.*

*The process reflects a realistic ETL pipeline suitable for portfolio presentation.*

***Project 1: CSV File ETL***

*├── Project\_1\_CSV\_FileETLRead.ipynb*

*├── etl\_to\_postgres.py*

*├── flight\_delays\_eschema.sql*

*├── flight\_delays\_sample.sql*

*├── README.md*