Retail Sales Data Processing and MySQL Integration using Python

# Synopsis

This project demonstrates the end-to-end data pipeline from downloading raw retail order data from Kaggle, processing it using Python, and storing it in a MySQL database for future analysis. The project covers critical skills necessary for any entry-level data scientist or data analyst, including API usage, data cleaning, feature engineering, and relational database integration.  
  
It simulates a real-world scenario where a data professional must not only extract and transform data but also prepare it for long-term usage, such as reporting, dashboarding, or modeling.

# Objective

- To extract retail sales data from a publicly available source (Kaggle).  
- To clean and enrich the dataset using Python and Pandas.  
- To load the cleaned data into a MySQL database with proper structure and data types.  
- To demonstrate best practices in database integration for analytics and reporting tasks.

# Technology Stack Used

- Python: For data handling, preprocessing, and scripting.  
- Pandas: For data manipulation and transformation.  
- Kaggle API: To access and download datasets.  
- MySQL: To store the cleaned data in a structured format.  
- SQLAlchemy: To enable Python-MySQL integration and schema definition.

# Project Steps Explained

1. Kaggle API Authentication:  
 The project begins by authenticating with Kaggle using the Kaggle API. This step simulates accessing external data sources in real-world projects.  
  
2. Dataset Download:  
 The required dataset (orders.csv) is downloaded from the Kaggle repository. This shows how automation can be used in the data acquisition phase.  
  
3. File Extraction:  
 The downloaded file is a zip archive, so we extract its contents using Python’s built-in zipfile library.  
  
4. Data Loading and Initial Cleaning:  
 The CSV file is loaded using pandas.read\_csv() while treating entries like 'Not Available' or 'unknown' as missing values.  
  
5. Column Name Cleaning:  
 Column names are standardized by converting them to lowercase and replacing spaces with underscores to avoid issues in querying and coding.  
  
6. Feature Engineering:  
 New columns are derived:  
 - discount = List Price \* Discount Percentage  
 - sale\_price = List Price - Discount  
 - profit = Sale Price - Cost Price  
  
7. Date Formatting:  
 order\_date is converted into proper datetime format to support time-based queries and filtering.  
  
8. Column Pruning:  
 Unnecessary columns (list\_price, cost\_price, discount\_percent) are dropped to reduce storage and processing load.

# Database Connection Setup

Using SQLAlchemy, a connection to a local MySQL database is created. The script defines database credentials and the target database (test). SQLAlchemy abstracts the complexities of raw SQL and enables smooth Python-to-SQL operations.

# Upload to MySQL with Explicit Schema

The final cleaned DataFrame is uploaded to the df\_orders table in MySQL. The key detail here is the explicit definition of each column’s data type, such as:  
- Integer for order\_id, quantity  
- Date for order\_date  
- String for text fields like city, category, product\_id  
- DECIMAL(7,2) for currency-related fields such as discount, sale\_price, and profit

# Why Assigning Data Types Explicitly is Important

- Data Integrity: Ensures that only correct types of data can be stored. For example, dates cannot be mistakenly stored as strings.  
- Storage Efficiency: Ensures that the database allocates the right amount of memory and space.  
- Query Performance: Improves performance of queries and joins, especially when filtering or aggregating.  
- Precision: Especially for monetary values, using DECIMAL maintains precision in financial calculations.  
- Integration Readiness: Proper data types ensure the database is ready for use in business intelligence tools or APIs without further modification.

# Why This Project Matters

This project introduces practical experience in:  
- Working with APIs (Kaggle)  
- Real-world data cleaning and transformation  
- Constructing and deploying structured tables in SQL  
- Understanding how backend data preparation supports frontend reporting and analysis  
  
These skills are universally required in roles involving dashboards, business insights, or machine learning pipelines.

# Conclusion

This project serves as a bridge between theoretical knowledge and industry expectations. It demonstrates a critical pipeline — data extraction, transformation, and loading — using Python and MySQL. The deliberate assignment of data types further simulates how professionals maintain accuracy, speed, and scalability in real-world systems.  
  
By completing such projects, freshers build not just technical competence but also the confidence to handle real-time datasets and infrastructure components in professional environments.