SQL ETL Pipeline Simulation Project Report

Abstract

This project simulates an **ETL (Extract, Transform, Load)** workflow using SQL within a lightweight database environment. The process involves extracting data from CSV files, cleaning and transforming it, loading it into structured production tables, and maintaining automated audit logs. Using only SQLite and SQL scripts, the project demonstrates how a complete ETL pipeline can be designed, executed, and automated without external ETL tools.

Introduction

ETL pipelines are core components of data engineering and analytics systems. They ensure that raw, inconsistent data is systematically processed into reliable, structured datasets for reporting and decision-making. This project replicates a real-world ETL workflow using SQL—covering data extraction, transformation, loading, and auditing—to highlight automation, integrity, and reproducibility. The focus is on practical SQL implementation and the use of database triggers for workflow automation.

Tools Used

- **Database:** SQLite
- **Interface:** DB Browser for SQLite
- **Data Source:** CSV file containing sales transactions

Steps Involved in Building the Project

1. Extract

Raw sales data from CSV files was imported into a **staging table** for temporary storage and inspection. The staging table schema included:

```
invoice_no, customer_id, gender, age, category,
quantity, price, payment_method, invoice_date, shopping_mall
```

This stage preserved the original dataset for validation before transformations.

2. Transform

Data cleaning operations were performed on the staging table:

- Removed duplicate and null records.
- Standardized text and date formats.
- Validated numerical fields (e.g., age, quantity, price).

Cleaned data was then enriched with computed columns such as `total_amount = quantity * price` for analytics use.

3. Load

The transformed data was inserted into structured **production tables**. These tables were designed for reporting and analytical queries, ensuring normalized schema and referential integrity. SQL scripts automated the loading process, enabling batch execution.

4. Audit and Automation

An **audit table** tracked each ETL operation by recording:

- Source and destination table names
- Number of rows inserted
- Timestamp of operation

Triggers were implemented to automatically log inserts into the audit table, ensuring full traceability of every ETL batch.

5. Cleanup and Export

Post-processing triggers handled cleanup of temporary staging data after successful load operations. Final production tables and audit logs were exported as CSVs for verification and documentation.

Conclusion

The SQL ETL Pipeline Simulation Project successfully demonstrated an end-to-end ETL workflow using only SQL and SQLite. It showcased how data can be extracted, validated, transformed, loaded, and audited within a single database system. This lightweight, fully SQL-driven approach is suitable for small-scale data warehousing, teaching ETL fundamentals, or prototyping ETL logic before deployment on enterprise systems.