Design and Implementation of an ETL Pipeline and Data Warehouse for E-Commerce Analytics

Walid K. W. Alsafadi, Ameer T. F. Alzerei, Hamza Obaid and Hazem Muanes

[[1]](#footnote-1)

***Abstract*—This paper presents the design and implementation of an ETL (Extract, Transform, Load) pipeline and a data warehouse tailored for e-commerce analytics. Data was ingested from the Fake Store API, providing comprehensive datasets on products, users, categories, and transactions. The raw data, retrieved in JSON format, underwent extensive transformation processes to ensure accurate data types, flatten nested structures, and derive calculated fields for meaningful analysis. A star schema was designed in PostgreSQL to support efficient querying and align with business goals, enabling seamless integration of fact and dimension tables. The project addresses real-world challenges, including API limitations and data inconsistencies, and demonstrates how structured data engineering workflows can generate valuable insights into sales trends, customer behavior, and product performance. Future improvements aim to extend the schema with additional fact and dimension tables to enhance the depth and accuracy of business analytics.**

***Index Terms*—ETL Pipeline, Data Warehouse, E-Commerce Analytics, PostgreSQL, Star Schema, Data Engineering, API Integration, Business Intelligence, Data Transformation, Data Ingestion.**

# I. INTRODUCTION

H

**arnessing** the power of data has become pivotal for e-commerce businesses aiming to optimize performance and make data-driven decisions. With vast amounts of information generated daily, transforming raw data into actionable insights presents both opportunities and challenges. This paper presents the development of an **ETL (Extract, Transform, Load) pipeline** and a **data warehouse** designed specifically for e-commerce analytics. effectively utilized for analysis. This paper presents the development of an **ETL (Extract, Transform, Load) pipeline** and a **data warehouse** designed specifically for e-commerce analytics.

The **ETL pipeline** automates the process of collecting data from APIs, transforming it into structured formats, and loading it into a PostgreSQL-based data warehouse. Data was sourced from the **Fake Store API**, a simulated e-commerce platform providing datasets on products, users, categories, and sales transactions. The project addresses common data engineering challenges, including handling nested JSON structures, ensuring data consistency, and aligning the data model with business requirements.

A **star schema** was implemented in the data warehouse to facilitate efficient querying and support business intelligence

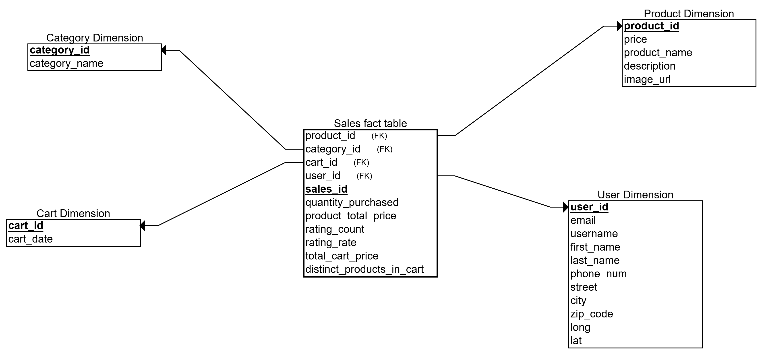
activities. This schema design enables the integration of fact tables containing transactional data with dimension tables that provide contextual information about products, users, and categories. By structuring the data in this way, the project aims to provide actionable insights into **sales trends, customer behavior,** and **product performance**.

The remainder of this paper outlines the methodology used to build the ETL pipeline and data warehouse, discusses the challenges encountered during the project, and highlights the key insights generated from the data. Future work includes extending the data warehouse schema to support more complex business queries and enhancing the ETL pipeline for scalability.

# II. Methodology

This project follows a structured data engineering workflow to develop an ETL pipeline and a data warehouse tailored for e-commerce analytics. The methodology consists of three main phases: **data ingestion**, **data transformation**, and **data warehouse design**. Each phase was carefully implemented using Python for automation, PostgreSQL for data storage, and structured schemas to optimize analytical performance.

The **Entity-Relationship Diagram (ERD)** in **Figure 1** illustrates the relationships between the fact and dimension tables in the star schema. This design ensures efficient querying and facilitates meaningful insights from the data. The following sections describe the detailed steps involved in each phase.



**Fig. 1.** ER Diagram of the Data Warehouse Schema.

## A. Data Ingestion

The data ingestion process involved retrieving data from the **Fake Store API**, which provides datasets in JSON format covering key areas of an e-commerce platform, such as products, users, categories, and carts. A Python script (ingestion.py) was developed to automate the API calls and save the retrieved data into the data/raw/ directory in both **JSON** and **CSV** formats.

The following API endpoints were utilized:

* **Products**: Information on titles, categories, prices, descriptions, images, and ratings.
* **Users**: Customer details including personal information and addresses.
* **Carts**: Transaction history with product IDs, quantities, and dates.
* **Categories**: Classification of products into categories.

## B. Data Transformation

The raw data collected from the API required extensive cleaning and transformation to be suitable for analysis. This phase was executed in two steps:

1. **Exploratory Data Analysis (EDA)** in **exploration.ipynb** to identify missing values, inconsistent data types, and redundant fields.
2. **Data Cleaning and Structuring** using the transformation.py script. Key transformations included:

* **Converting nested JSON structures** (e.g., splitting product ratings into separate rate and count columns).
* **Ensuring consistent data types,** such as converting latitude and longitude fields from strings to floats.
* **Mapping categories** to their respective IDs to maintain referential integrity.
* **Removing redundant fields** and ensuring the data aligned with the data warehouse schema.

The cleaned datasets were saved in the data/processed/ directory, ready for loading into the data warehouse.

## C. Data Warehouse Design

A **star schema** was designed and implemented in **PostgreSQL** to optimize query performance and support business intelligence activities. The schema consists of one **fact table** (sales) and multiple **dimension tables** (products, users, categories, and address).

The following API endpoints were utilized:

* **Fact Table (**sales**):** Contains transactional data, such as quantity purchases, products total price, and rates.
* **Dimension Tables:**
  + **Products:** Product details including title, price, category, and rating.
  + **Users:** Customer information linked to the address dimension.
  + **Categories:** Classification of products.
  + **Cart:** Cart ID and date

The modeling.py script was used to automate the creation of tables in PostgreSQL using the schema defined in dw\_schema.sql, and to load the processed data into the database. The star schema was chosen over a snowflake schema due to its simplicity and better performance in analytical queries, aligning with the business goals.

# III. Challenges and Solutions

Throughout the development of the ETL pipeline and data warehouse, several challenges emerged, particularly during the data ingestion and transformation phases. This section outlines the primary obstacles encountered and the solutions implemented to overcome them.

## A. API Limitations

One of the initial challenges was the **limited availability of free APIs** that provided comprehensive e-commerce data suitable for this project. Many APIs either had restrictive data access policies or lacked the depth required for constructing a meaningful data warehouse. After extensive research, the **Fake Store API** was selected as it provided sufficient product, user, and transaction data to simulate a real-world e-commerce environment. Although this API offered a simplified dataset, it allowed for effective demonstration of ETL processes and data warehousing principles.

## B. Data Transformation Complexities

The raw data retrieved from the API presented several **transformation challenges**:

1. **Nested JSON Structures:** The rating field in the products dataset was stored as a nested JSON object, requiring extraction and normalization.
2. **Inconsistent Data Types:** Certain fields, such as latitude and longitude, were stored as strings instead of numerical types, leading to issues during data analysis.
3. **Fact Table Calculations:** Populating the fact table required deriving additional columns, such as calculating **total sales** from **quantity** and **price**.

The data transformation phase was handled using **Python (Pandas)** for data manipulation. Key steps included:

* **Flattening Nested JSON:** The rating field was split into separate rate and count columns.
* **Data Type Conversion:** String-based numerical fields were converted to appropriate data types (e.g., float for geographic coordinates).
* **Derived Columns:** New columns were created in the fact table by performing calculations on existing data (e.g., ).

## C. Schema Design Alignment

Designing a schema that effectively **aligned with the retrieved data** and supported efficient analytical queries posed another challenge. The initial data did not perfectly match a traditional star schema format, requiring adjustments to both the schema design and the data. A **star schema** was selected due to its simplicity and superior performance for **OLAP (Online Analytical Processing)** queries. The schema was iteratively refined to ensure that the fact and dimension tables were properly normalized and connected via foreign keys. This alignment ensured the warehouse could address key **business questions** such as **sales trends**, **top-performing products**, and **customer behavior**.

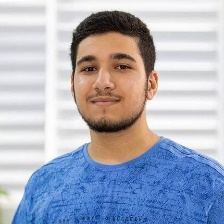
# IV. Results and Insights

V. Future Work and Improvements

VI. Conclusion

References

1. J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility,” *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959, doi: 10.1109/TED.2016.2628402.
2. E. P. Wigner, “Theory of traveling-wave optical laser,” *Phys. Rev*., vol. 134, pp. A635–A646, Dec. 1965.
3. P. Kopyt *et al., “*Electric properties of graphene-based conductive layers from DC up to terahertz range,” *IEEE THz Sci. Technol.,* to be published, doi: 10.1109/TTHZ.2016.2544142. *(Note: If a paper is still to be published, but is available in early access, please follow ref [5]).)*

**Walid K. W. Alsafadi** was born in Ras Al Khaimah, UAE. He is currently pursuing a Bachelor of Science in Data Science and Artificial Intelligence at the University College of Applied Science, Gaza, Palestine. His major field of study focuses on data science and artificial intelligence, with expertise in machine learning, deep learning, and natural language processing.

Walid has previously studied at the American University of Ras Al Khaimah, UAE, where he completed two semesters before transferring to UCAS. His academic achievements include multiple scholarships and recognition on the Dean’s List for academic excellence. He has worked on diverse projects, including sales forecasting, sentiment analysis, and diabetes prediction, utilizing machine learning techniques and frameworks.

Mr. Alsafadi is proficient in multiple programming languages, including Python and SQL, and is a native Arabic speaker with full professional proficiency in English.

**Ameer T. F. Alzerei**, AI and Data Scientist.

**Hamza Obaid**, AI and Data Scientist.

**Hazem Muanes**, AI and Data Scientist.

1. [↑](#footnote-ref-1)