**Data Engineering Project Report**

**Customer and Traffic Insights with TLC Data**

**A group of yellow taxi cabs in a busy city street

Description automatically generated**

Submitted to:

Prof. (Dr.) Frank Schulz

Prof. Eunnuri Cho

Submitted by:

Sandhya Tripathi (11038709)

Disha Gayathri Umashankar (11038725)

Blessy Evangeline Aaron(11038695)

ADSA(SoSe’24)

**Abstract**

**Introduction:**  
In today’s fast paced data-driven world, analyzing and understanding traffic patterns play a pivotal role in business optimization. Effective management of data enables organizations and authorities to derive actionable insights, leading to improved decision-making processes such as optimized traffic flow, enhanced public transit operations, and accurate demand forecasting during peak periods. For our project we have considered New-York TLC (Taxi and Limousine Commission) data.

**Summary of Own Approach:**

This project focuses on developing a scalable end-to-end data pipeline for traffic analysis using TLC trip data. The approach involves building an Extract-Transform-Load (ETL) pipeline leveraging **Docker** for containerization, **Redis** for caching, **PostgreSQL** for relational data storage with a star schema and **Tableau** for data visualization

**Summary of Own Results:**  
The implemented pipeline successfully ingested and processed over 100,000 records (15MB data) providing insights into trip patterns, customer segmentation, and payment methods. The results highlight key trends, such as peak travel hours, popular pickup/dropoff locations and fare distributions, showcasing the pipeline's ability to support actionable analytics for urban mobility and business strategies.

A screenshot of a computer

Description automatically generated**Bibliography:**

* Big Data Analytics of Taxi Operations in New York City Author: Y. Tang, American Journal of Operations Research, 2019
* A Study of More Than One Billion Taxi Trips in New York City Authors: D. Zhao, K. Musolesi, Scientific Reports, 2020
* Big Data Driven Model for New York Taxi Trips Analysis Authors: S. A. Mostafa, M. M. Hamdan, M. A. Alsmirat, IEEE Access, 2021
* Spatiotemporal Analysis of Ridesourcing and Taxi Demand by Taxi Zones in New York City  
  Authors: P. Toman, J. Zhang, N. Ravishanker, K. Konduri, arXiv preprint, 2020

**Chapter 1: Introduction**

**Application Domain**:

**Data Source Link:** [**https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page**](https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page)

**TLC Trip Record Data:**

Yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs.

**Application Problem:**

In real-world industries such as transportation, data often originates from diverse sources, including GPS systems, fare calculators, and customer information systems. These datasets are typically fragmented, raw, and disconnected, making it challenging to unify and analyze them effectively. For instance, a transportation company might store data such as trip distances, passenger counts, timestamps, and fare amount across multiple files and tables.

The core problem arises when organizations attempt to:

* Integrate these large, disparate datasets.
* Perform real-time or batch data processing.
* Enable structured querying and analysis for actionable insights.

Without a robust solution, the data remains underutilized, and critical insights into customer behavior, revenue analysis, and trip patterns are missed.

**Benefits of the Solution:**

Our proposed solution introduces an end-to-end data pipeline that combines raw, fragmented datasets into a unified, structured format. This pipeline addresses the core challenges of data integration, real-time processing, and visualization.

**Technical Challenges:**

1. **Kafka Producer-Consumer with Airflow Compatibility:**

* Initially, we planned to implement a Kafka-based producer-consumer pipeline for data ingestion.
* However, the integration with Apache Airflow failed due to an OS compatibility issue on Windows, resulting in errors while generating Kafka topics.
* After thorough analysis, we identified Airflow's limitation on Windows systems.

1. **Google Cloud Platform (GCP) Upload Issues:**

* To overcome the previous issue, we switched to Google Cloud Platform (GCP) for data storage.
* Due to a low bandwidth connection (5 MBPS), the upload of the 15 MB dataset was extremely slow.
* Even after uploading, performing transactions was inefficient, leading us to drop this approach.

1. **Switch to Redis for Producer-Consumer:**

* Finally, we opted for Redis as an alternative for the producer-consumer architecture.
* Redis is lightweight and faster to set up compared to Kafka.
* This decision resolved the data ingestion and real-time processing challenges.

1. **Postgres and PGAdmin Configuration in Docker:**

* While implementing PGAdmin for PostgreSQL in the Docker setup, shutting down Docker threw a "configuration error".
* After troubleshooting, we removed the PGAdmin service temporarily and successfully restarted the container without errors.
* The YAML file was updated, and PGAdmin was reconfigured successfully.

1. **Lag in Redis Producer-Consumer Setup:**

* During Redis implementation, the Producer service worked efficiently and started generating records quickly.
* However, the Consumer service experienced an initial lag and took longer to start.
* After troubleshooting, the consumer began functioning as expected. This lag was attributed to the initial setup process.

**Technical Solution Idea:**

To address the above mentioned challenges, our solution leverages a combination of modern tools and technologies to build a scalable and efficient ETL pipeline. The key components of the solution are:

**Docker:** Docker is used to containerize the entire data pipeline, ensuring that all dependencies, libraries, and configurations are packaged into isolated environments.This enables smooth deployment across different systems and platforms, reducing compatibility issues.

**Redis:** Redis acts as an in-memory data store to cache data during transformations, significantly speeding up real-time operations.

**Postgres:** PostgreSQL serves as the relational database for structured data storage. Raw data is transformed into a star schema consisting of fact and dimension tables.

**Tableau:** Tableau is the final component of the pipeline, where the processed data is visualized through interactive dashboards.It enables stakeholders to analyze trends, monitor KPIs, and generate actionable insights using intuitive charts and graphs.

**Chapter 2: Related Work**

**Overview of Existing Solutions**:

* Traditional ETL tools like **Apache Spark**, **AWS Glue**, and **BigQuery** have been widely used for large-scale data processing.
* Redis and Postgres are commonly employed in scalable data storage and management systems.
* Tableau is an industry-standard tool for visualization.

**Comparison**:

* While solutions like **BigQuery** and **AWS Glue** require significant cloud infrastructure, our approach uses Docker containers for flexibility.
* Tools like Redis are optimized for in-memory operations compared to traditional disk-based caching.
* Tableau provides interactive dashboards, making it ideal for visualizing Postgres-stored insights.

**What Makes Our Solution Different**: Our prototype integrates Redis and Postgres into a Dockerized pipeline, enabling faster processing, modularity, and cost-effective local development.

**Chapter 3: Dataset**

**Dataset Information:** The TLC (15MB) dataset is finally having 1,00,000 records for Yellow Taxi services for New York region for 2024. The final dataset contains 19 features. The details are as follows:

|  |  |
| --- | --- |
| **Column Name** | **Description** |
| vendorid | Unique identifier for the taxi provider or vendor. |
| tpep\_pickup\_datetime | Timestamp indicating when the trip started. |
| tpep\_dropoff\_datetime | Timestamp indicating when the trip ended. |
| passenger\_count | The number of passengers during the trip. |
| trip\_distance | Total distance traveled during the trip in miles. |
| pickup\_longitude | Longitude coordinate of the trip's pickup location. |
| pickup\_latitude | Latitude coordinate of the trip's pickup location. |
| RatecodeID | Code indicating the rate type applied for the trip (e.g., standard rate). |
| store\_and\_fwd\_flag | Indicates whether the trip record was stored and forwarded (Y/N). |
| dropoff\_longitude | Longitude coordinate of the trip's drop-off location. |
| dropoff\_latitude | Latitude coordinate of the trip's drop-off location. |
| payment\_type | Method of payment used (e.g., credit card, cash). |
| fare\_amount | The base fare amount charged for the trip. |
| extra | Additional charges such as congestion fees or peak-time surcharges. |
| mta\_tax | Mandatory tax imposed by the Metropolitan Transportation Authority (MTA). |
| tip\_amount | Gratuity amount provided to the driver. |
| tolls\_amount | Total toll fees incurred during the trip. |
| improvement\_surcharge | Additional surcharge for trip improvement or infrastructure funding. |
| total\_amount | The final total amount charged for the trip, including all fees and taxes. |

**Chapter 4: Solution**

**Tools and Technologies Used:**

A screenshot of a computer

Description automatically generated

**Implementation Steps**:

**1. Data Extraction**: raw\_data was loaded in Producer container after preprocessing has been completed in excel.

**2.** **Docker Containerization:** After that YML file was created in order to create various services, for example redis was onboarded on port no 6379, Postgres was on port number 5432, healthcheck was also included to monitor the status of postgres, pgAdmin (interface used for Postgres database) was configured on local host 8080. Images of Producer and consumer was also created. Below in the docker version”3.8” YML file code for creating images.

A screenshot of a computer

Description automatically generated

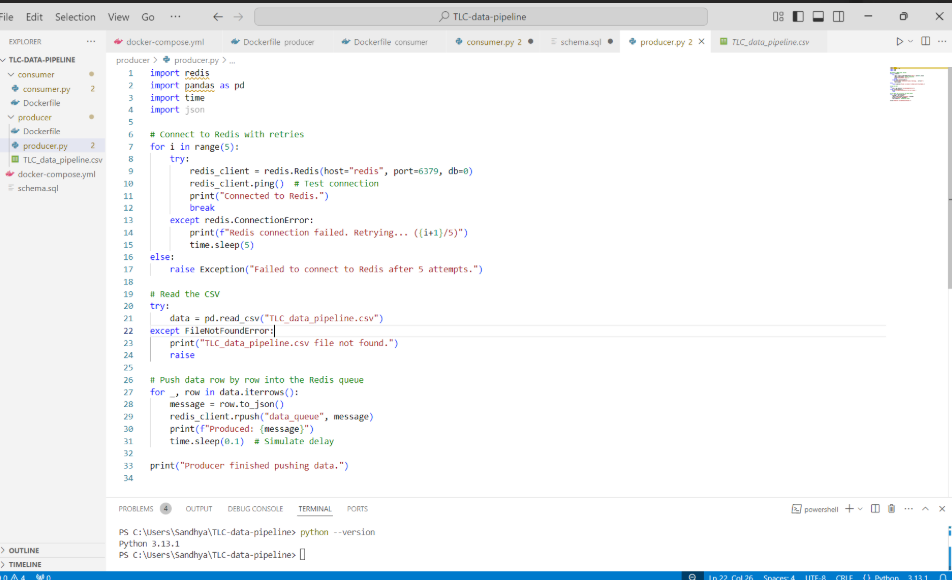
3.**All containers started when the “docker compose build”**

A screenshot of a computer

Description automatically generated

4. As shown in the image above, all **five** servers are actively running, with their status indicated as **green**. The **tlc-data-pipeline** comprises five containers: **Producer**, **Consumer**, **Redis Server**, **Postgres**, and **Postgres Admin**.

5. **Producer Code-Implementation:**

****

6. **Consumer Code-Implementation:**

**A screenshot of a computer

Description automatically generated**

7.**Postgresql Command: Postgres Up and Running**

**A screenshot of a computer

Description automatically generated**

8.**Queries getting executed** PGAdmin Console (Working, all 8 tables and 1 original table are stored in the PostGresSQL Database)

A screenshot of a computer

Description automatically generated

9. **Connection with Tableau and Postgresql Server**

**A screenshot of a computer

Description automatically generated**

**Challenges and Solutions**:

1. **File Parsing**: Resolved inconsistencies in CSV formats during extraction.
2. **Docker Setup**: Addressed container network configuration for Redis and Postgres.
3. **Memory Management**: Optimized Redis for large data caching.

**Decisions Made**:

* Used Docker for modular ETL deployment.
* Redis for faster caching during transformations.
* Postgres for storing relational data.
* Tableau for generating insights effectively.

**Chapter 5: Summary and Outlook**

**Summary of Results**:



**Query 1:**

A computer code with black text

Description automatically generated



Query 2:

A computer code with black text

Description automatically generated



Query 3:

A screenshot of a computer code

Description automatically generated



Query 4:

A computer code with black text

Description automatically generated

Traffic by Day of the Month

A graph of blue lines

Description automatically generated with medium confidence

It highlights the distribution of trip counts across all days, revealing days with **higher traffic** and confirming overall **consistency** in trip volume throughout the month.

Total Revenue by Payment Type

A screenshot of a computer

Description automatically generated

It shows that Credit Card payments account for the highest revenue at 4,149,513, followed by Cash at 2,079,333, while No Charge and Dispute contribute negligible amounts to the total revenue.

Revenue Trends Over Days

A graph with blue lines and numbers

Description automatically generated with medium confidence

It shows that revenue remains relatively consistent across days, with peaks on Days 24 and 25 reaching up to 232,748and 232,553, indicating slightly higher earnings on these days compared to the rest of the month.

**Future Work**:

1. Automate pipeline monitoring and alerts.
2. Expand visualization with advanced Tableau dashboards.

**Ownership of Content:** The setting of Data Pipeline has been carried out by Sandhya Tripathi, Database has been implemented by Disha Gayathri Umashankar and Tableau Dashboard and Date Preprocessing has been completed by Blessy Evangeline Aaron.

**End of Document**

This report includes complete content for all chapters with placeholders for results.