Introduction

For Assignment 2, I extended my NYC TLC Yellow Taxi data warehouse from Assignment 1 into a complete cloud-based analytics stack. My goals were to design a repeatable pipeline that loads raw January 2023 Yellow Taxi trip data into Amazon Redshift, transforms it into a star schema, and serves it to business users through an online visualization tool and a simple Python "API" that exports analytical CSV files. This report describes the transformation logic, the data warehouse modeling decisions, and the serving layer I built with Tableau and Python.

Cloud Architecture and Technology Stack

All components of this project run in the cloud or connect directly to it:

- **Storage:** An Amazon S3 bucket (taxi-tlc-bucket-abbas) stores the raw TLC Yellow Taxi file for January 2023 in both Parquet and CSV formats under the raw/ prefix.
- **Data Warehouse:** An Amazon Redshift Serverless workgroup (default-workgroup) hosts the dev database and the taxi_dw schema. Redshift reads the raw CSV from S3 using an IAM role (RedshiftS3AccessRole) with AmazonS3ReadOnlyAccess and Redshift command permissions.
- **Compute / Scripting:** Locally, I used Python on my Mac (inside the dw_assignment1_sql project folder) to write extraction and export scripts. These scripts connect securely to Redshift using the JDBC/host endpoint and credentials.
- **Serving / Visualization:** For the BI layer I used Tableau, connecting live to the Redshift taxi_dw schema. The final dashboard is published to Tableau Public and will be shared via link in the submission.

The entire stack is tracked in a GitHub repository, including Python scripts, SQL DDL/DML, and documentation.

Transformation Stage

The first step in Assignment 2 was to upgrade my transformation pipeline so that all business rules are encoded in SQL and can be re-run from scratch.

I created a dedicated Redshift schema called taxi_dw. Inside this schema, I first defined a **staging table**stg_yellow_tripdata with data types chosen to match the raw TLC file but still be friendly to Parquet and Redshift. Most numeric columns, including IDs and measures, are typed as DOUBLE PRECISION, and timestamps are stored as TIMESTAMP. I then used the Redshift COPY command to load the January 2023 CSV from S3:

- Source: s3://taxi-tlc-bucket-abbas/raw/yellow_tripdata_2023-01.csv
- IAM role: RedshiftS3AccessRole

- Options: CSV format, header row ignored, commas as delimiter, blank fields treated as NULL.

From the staging table I created a **cleaned trip table** called taxi_trips_clean. This step is where most of the transformation requirements from the assignment were implemented:

- I enforced a **unified date format** by casting tpep_pickup_datetime to a DATE column called trip_date (implicitly in YYYY-MM-DD format).
- I **split the timestamp into multiple units** to support flexible time-based analysis:
  - trip_year, trip_quarter, trip_month, trip_day, trip_hour, and trip_dow (day of week).
- I **filtered out bad records** by keeping only rows where pickup and dropoff timestamps are not NULL and where trip_distance and total_amount are non-negative. This prevents clearly invalid trips from skewing downstream analysis.
- I added several **derived columns**:
  - trip_duration_minutes computed as the minute difference between pickup and dropoff timestamps.
  - total_surcharges, which is the sum of extra, mta_tax, improvement_surcharge, congestion_surcharge, and airport_fee.
- All numeric measures were cast to appropriate types (DECIMAL or BIGINT) in the later fact table to ensure accurate aggregation.

Throughout this process, I maintained a **data mapping and data dictionary** in a Google Sheet, which I exported as a PDF for submission. For each field in the cleaned table, fact table, and dimensions, the dictionary documents the field name, data type, description, source column, and destination column. This connects the raw TLC schema to the analytical schema in a transparent way and satisfies the assignment's documentation requirement.

Data Warehouse Modeling

With the cleaned trips table in place, I designed a **star schema** with one central fact table and several dimension tables.

The central **fact table** is fact_taxi_trips, which contains one row per taxi trip at the chosen grain. This table has a surrogate primary key trip_key generated by Redshift (IDENTITY(1,1)). The fact table stores all quantitative measures relevant for analysis, including:

- passenger_count, trip_distance, fare_amount, tip_amount, tolls_amount, and total_amount.
- The derived total_surcharges and trip_duration_minutes.
- Location IDs (pulocationid and dolocationid) for optional spatial analysis.

Each row in the fact table links to multiple dimension tables through foreign keys:

- **dim_vendor**: Contains a surrogate vendor_key, the original vendorid, and a descriptive vendor_name (e.g., Creative Mobile Technologies, Verifone, Other/Unknown).
- **dim_rate_code**: Contains ratecode_key, the TLC ratecodeid, and a textual description such as Standard rate, JFK, Newark, Group ride, and so on.
- **dim_payment_type**: Contains payment_type_key, the TLC payment_type, and human-readable labels like Credit card, Cash, No charge, Dispute, and Unknown.
- **dim_datetime**: Functions as a pickup datetime dimension with a surrogate datetime_key, pickup_date, and the split fields for year, quarter, month, day, hour, and day of week.

The dimensions are populated from taxi_trips_clean using INSERT … SELECT DISTINCT … statements so that each combination of code and description appears only once. This separation of fact and dimensions simplifies reporting and allows business users to slice measures by vendor, rate code, payment method, and time.

To make the warehouse easier to consume, I created two **views**:

- v_taxi_trips_star joins the fact table to all its dimensions, returning a denormalized star view that is ideal for BI tools. This view exposes friendly column names such as Pickup Date, Vendor Name, Payment Desc, and the numeric measures.
- v_taxi_trips_for_map reads from taxi_trips_clean and exposes locationid, pickup and dropoff datetimes, passenger count, trip distance, and total amount. This is specifically used to join with the NYC Taxi Zone shapefile in Tableau for geographic visualizations.

The complete data model, including all tables and relationships, is documented in an ERD created with Lucidchart and exported as Assignment 2 ERD.pdf.

Serving Layer: Visualizations and Analytical Access

For the serving layer, I used **Tableau** as my cloud-connected visualization tool. Tableau connects live to the Redshift workgroup using the Redshift endpoint, database dev, and schema taxi_dw. I used the v_taxi_trips_star view as the main data source and joined v_taxi_trips_for_map to the NYC Taxi Zones shapefile for spatial analysis.

I created several visualizations that together satisfy all of the assignment's chart requirements:

- **Line chart – "Daily Yellow Taxi Trips — January 2023"**: This sheet plots the daily count of trips using Pickup Date on the x-axis and COUNT(Trip Key) on the y-axis. It helps reveal weekly and monthly patterns in overall demand.
- **Line chart – "Trips by Pickup Hour — January 2023"**: Here I plot Pickup Hour against COUNT(Trip Key), which highlights peak hours for taxi usage. The chart shows a clear increase from early morning toward late afternoon and evening.

- **Line chart – "Trips by Weekday — January 2023"**: This sheet uses Pickup DOW (Sunday through Saturday) with COUNT(Trip Key) to compare average daily activity across weekdays.
- **Pie chart – "Trips by Payment Type"**: This visualization slices COUNT(Trip Key) by Payment Desc and shows that the majority of trips are paid by credit card, followed by cash and a very small percentage of disputes or no-charge trips.
- **Column chart – "Total Revenue by Day — January 2023"**: This chart uses Pickup Date on the x-axis and SUM(Total Amount) on the y-axis to show daily revenue generated by yellow taxi trips.
- **Heat map – "Trips by Hour and Weekday"**: For this sheet I placed Pickup DOW on rows, Pickup Hour on columns, and used the count of trips as the color. The heat map makes it easy to see which hours are busiest on each day of the week.
- **Geographical map – "Total Revenue by Pickup Zone — January 2023"**: Using the v_taxi_trips_for_mapdata and the NYC Taxi Zones shapefile, I mapped each taxi zone and colored it by total revenue. Areas like JFK Airport stand out with darker shades, indicating higher revenue.

I combined these sheets into a **single dashboard** in Tableau. The dashboard includes a **date filter** based on Pickup Date, so that when the user adjusts the date range, all charts update accordingly. This satisfies the requirement for a global filtering control. The dashboard is published to Tableau Public, and the share link will be submitted along with this report.

Python API and CSV Export

To allow analysts who prefer working with files instead of dashboards, I built a simple "API-style" Python script called export_taxi_summary.py, located in the api/ folder. The script connects from my local machine to the Redshift data warehouse using the Redshift endpoint, database name, username, and password. It executes an aggregated SQL query against v_taxi_trips_star to compute a **daily summary** for January 2023, including:

- date, total number of trips, total revenue, average trip distance, and average total amount.

The script then writes the results to a CSV file at output/taxi_daily_summary_2023_01.csv. Running the script prints confirmation to the terminal and generates a 31-row CSV, one row per day in January. This fulfills the assignment's requirement for a Python script that generates CSV data for analysts as part of the serving data layer.

Version Control and Reproducibility

All logic and documentation for Assignments 1 and 2 are stored in my GitHub repository dw_assignment1_sql. The repository includes:

- Raw data samples in raw_data/.
- Python scripts for downloading TLC data, converting Parquet to CSV, loading into the warehouse, and exporting analytical summaries in scripts/ and api/.
- SQL scripts for creating and populating the Redshift schema, staging, clean, dimension, and fact tables, as well as sanity checks, in the sql/ folder.
- Documentation in docs/, including the Assignment 1 report, Assignment 2 data dictionary and mapping, and the ERD.
- The Tableau workbook is saved in the root of the project for completeness.

Anyone with access to the Redshift credentials and this repository can reproduce the pipeline by running the SQL build script in Redshift, refreshing the Tableau connection, and executing the Python export script.

Conclusion

In Assignment 2, I transformed the single-file TLC Yellow Taxi dataset into a complete cloud-based analytic solution. Starting from raw Parquet and CSV files in S3, I implemented a robust transformation pipeline in Redshift to clean the data, enforce a consistent date structure, derive additional business metrics, and model the results in a star schema with a central fact table and multiple dimensions. I then exposed the data through an online Tableau dashboard and a Python script that exports daily CSV summaries. Together, these components satisfy the transformation, modeling, and serving requirements of the assignment and demonstrate how a cloud data warehouse can support both visual analytics and file-based analysis for business users.