ASSIGNMENT 1 – DATA WAREHOUSING REPORT

NYC TLC Yellow Taxi Trip Data (January 2023)
Prepared by: Abbas Syed

1. Data Source

For this assignment, I selected the **NYC TLC Yellow Taxi Trip Record Dataset** for **January 2023**.
This dataset is publicly available through the NYC Taxi & Limousine Commission's official website and hosted on AWS CloudFront.

Dataset URL:
https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-01.parquet

A data dictionary was created to fully document each field, including datatype, description, and constraints.

Data Dictionary (Google Sheet):
https://docs.google.com/spreadsheets/d/11K22GrLOxfj3pEcFKnM_dzLQC3pekfMgwXTJjKfXmQo/edit?usp=sharing

All project scripts and artifacts have been stored in a public GitHub repository.

GitHub Repository:
https://github.com/Abbasyed3/dw_assignment1_sql

A complete Entity Relationship Diagram (star schema) was created using Miro.

Miro ERD Diagram:
https://miro.com/app/board/uXjVJnjaEic=/?share_link_id=871598796638

2. Scripts Created

To complete the data sourcing, storage, and modeling steps, I created three main scripts:

1. download_yellow_taxi_to_s3.py
This script downloads the raw TLC parquet file and uploads it to my AWS S3 bucket inside the /raw/ folder.

2. load_taxi_to_postgres.py
This script reads the parquet file and loads it into the AWS RDS PostgreSQL database into the staging table public.raw_yellow_tripdata.

3. create_taxi_dw.sql
This SQL script creates the entire data warehouse star schema, including dimensions and fact tables, and loads them based on the staging table data.

All scripts are stored in the GitHub repository.

3. Storage Setup

The assignment required choosing a cloud storage solution and storing data in an organized manner. I used **Amazon Web Services (AWS)**.

AWS S3 Bucket

Bucket name: **taxi-tlc-bucket-abbas**

This bucket is organized as follows:

- **raw/** – contains the original parquet file:
  yellow_tripdata_2023-01.parquet
- **clean/** – reserved for transformed outputs (Assignment 2)
- **warehouse/** – reserved for serving layer or Redshift outputs (Assignment 2)

AWS RDS PostgreSQL

I created a Postgres instance which contains:

- A staging table: **public.raw_yellow_tripdata**
- A data warehouse schema: **taxi_dw**

The staging table holds the raw data exactly as provided, while the taxi_dw schema contains modeled dimension and fact tables.

4. Data Warehouse Modeling (Star Schema)

A classical star schema was designed.

Dimension Tables

1. **dim_vendor**
   Contains vendor identifiers and vendor descriptions.
2. **dim_rate_code**
   Contains rate code IDs and definitions.
3. **dim_payment_type**
   Contains payment method categories.
4. **dim_datetime**
   Breaks down the pickup timestamp into date, hour, and day-of-week components.

Fact Table

fact_taxi_trips
This table captures all numeric measures for each trip, such as:

- passenger count
- trip distance
- fare amount
- tips
- tolls
- total amount
- surcharges

It also contains foreign keys linking to all four dimensions.

ERD Diagram

A complete ERD showing the fact and dimension tables is available here:
https://miro.com/app/board/uXjVJnjaEic=/?share_link_id=871598796638

5. End-to-End Execution Pipeline

The workflow used to implement Assignment 1 is summarized below:

Step 1: Data Sourcing
Ran the Python script to download and upload the TLC dataset to AWS S3.

Step 2: Loading Raw Data
Executed the loader script to insert the parquet data into the staging Postgres table.

Step 3: Data Warehouse Creation
Executed the SQL script to:

- Create the taxi_dw schema

- Create all dimension and fact tables with surrogate keys
- Populate all tables

Step 4: Validation
Row counts were checked for all tables.
The fact table contains ~2.99M rows, and each dimension is populated correctly.

6. Validation Queries (Reference)

These queries were used to validate the data warehouse:

- Count rows in staging table
- Count rows in each dimension
- Count rows in fact table
- Verify that primary and foreign key relationships work correctly

All validation checks passed.

7. Tools Used

- **Python** (boto3, pandas, psycopg2, requests)
- **AWS S3**
- **AWS RDS PostgreSQL**
- **GitHub** for version control
- **Miro** for the star-schema ERD