Creating a Data Warehouse Using Open Data TLC Taxi Data

**Step 1: Define the Purpose**The goal is to build a data warehouse to consolidate and analyze TLC Yellow Taxi, Uber, and Lyft trip data. This will provide insights into trip patterns, demand, and driver performance. We’ll use **Google Cloud Console** for its scalability, cost-efficiency, and integrated tools.

**Step 2: Data Collection**Data is sourced from the TLC Trip Record Data page:  
<https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>.

**Step 3: Selecting Data**Filter the data by year (2023 and 2024) and ride type (taxi, Uber, Lyft) for meaningful analysis.

The data warehouse will consist of:

1. **Fact Table**: Stores transactional data (e.g., trips, distance).
2. **Dimension Tables**: Contains descriptive information (e.g., locations, time).

Data Mart 1: Trip Metrics by Location and Time

Focuses on analyzing trip volume and patterns based on location and time.

* **Dimensions**:
  + **Time**:
    - Year
    - Month
    - Day of Week
    - Hour of Day
  + **Geography**:
    - Pickup Borough
    - Dropoff Borough
    - Pickup Zone
    - Dropoff Zone
  + **Ride Type**:
    - Taxi, Uber, Lyft
* **Metrics**:
  + Total Trips
  + Average Trip Distance
  + Total Trip Distance
  + Total Revenue (if available)
* **Use Cases**:
  + Analyze trip demand trends over time.
  + Identify high-demand pickup/dropoff zones by hour.

Data Mart 2: Driver Performance

Focuses on tracking and analyzing driver efficiency and performance.

* **Dimensions**:
  + **Driver**:
    - Driver ID (anonymized)
  + **Time**:
    - Year
    - Month
    - Day of Week
    - Hour of Day
  + **Geography**:
    - Pickup Borough
    - Dropoff Borough
    - Pickup Zone
    - Dropoff Zone
* **Metrics**:
  + Total Trips per Driver
  + Average Trip Distance per Driver
  + Total Working Hours (if timestamp data is available)
  + Average Revenue per Trip (if revenue data is available)
* **Use Cases**:
  + Identify top-performing drivers.
  + Assess driver efficiency and identify potential training needs.

Key Steps for Implementation

1. **Extract Data**:
   * Use TLC datasets for Yellow Taxi, Uber, and Lyft for 2023 and 2024.
2. **Transform Data**:
   * Clean data (remove nulls, handle inconsistencies).
   * Add calculated columns (e.g., hour\_of\_day, day\_of\_week).
   * Map PULocationID and DOLocationID to borough and zone using the taxi zone reference file.
3. **Load Data**:
   * Create a star schema in the data warehouse:
     + **Fact Table**: fact\_trips (stores metrics such as trips and distance).
     + **Dimension Tables**: dim\_time, dim\_geography, dim\_ride\_type, and dim\_driver.
4. **Data Mart Creation**:

Use SQL queries or ETL tools to populate the data marts from the star schema.

The data warehouse can deliver valuable insights for strategic decisions, resource allocation, and performance optimization by focusing on these two data marts.

Model 1: Multiple Linear Regression Using Data Marts

CRISP-DM Approach

1. **Business Understanding**
   * **Goal**: Predict the total number of trips for a given location and time based on historical trends and features such as pickup/dropoff locations, time, and ride type.
   * **Use Case**: Help optimize resource allocation (e.g., vehicle availability, driver assignments) by forecasting demand.
2. **Data Understanding**
   * Source: **Data Mart 1: Trip Metrics by Location and Time**
   * Key Variables:
     + **Dependent Variable (Target)**:
       - Trips: Total number of trips for a specific time and location.
     + **Independent Variables (Features)**:
       - hour\_of\_day: Hour of trip start (e.g., 0–23).
       - day\_of\_week: Day of the week (e.g., Monday = 1, Sunday = 7).
       - pickup\_borough: Borough of the pickup location (categorical).
       - dropoff\_borough: Borough of the dropoff location (categorical).
       - ride\_type: Type of ride (Taxi, Uber, Lyft) (categorical).
3. **Data Preparation**
   * **Data Cleaning**:
     + Remove rows with missing or invalid values.
     + Handle outliers (e.g., extreme distances or trips at unlikely times).
   * **Feature Engineering**:
     + Convert categorical variables (pickup\_borough, dropoff\_borough, ride\_type) into one-hot-encoded variables.
     + Normalize or scale numerical features (hour\_of\_day, day\_of\_week) to ensure uniformity.
   * **Train-Test Split**:
     + Split the data into training (70%) and testing (30%) sets.
4. **Modeling**

**Implementation**:

* + - Use a library like Python’s sci-kit-learn or SQL-based ML in BigQuery.
    - Fit the model on the training data.

1. **Evaluation**
   * **Metrics**:
     + **Adjusted R²**:
       - Measures the proportion of variance in trips explained by the independent variables.
     + **Root Mean Squared Error (RMSE)**:
       - Provides the average magnitude of prediction errors.
     + **Mean Absolute Error (MAE)**:
       - Measures the average error in absolute terms, less sensitive to outliers.
   * **Cross-validation**:
     + Use k-fold cross-validation to ensure the model generalizes well.
2. **Deployment**
   * Integrate the model into a dashboard for demand forecasting.
   * Use real-time inputs (e.g., current hour, day, ride type, location) to predict trips dynamically.

How to Measure Model Performance

* **Adjusted R²**:
  + Indicates the percentage of variability in the dependent variable (trips) explained by the features.
  + A high Adjusted R² (close to 1) signifies a good fit.
* **RMSE and MAE**:
  + Measure prediction accuracy.
  + Lower values indicate better performance.
  + Compare these metrics between training and testing data to ensure no overfitting.

Visualization -: <https://developers.google.com/chart/interactive/docs/gallery>

* **Feature Importance**:
  + Bar chart showing the coefficients for each independent variable to understand their impact on trip demand.
* **Predicted vs. Actual Trips**:
  + Scatter plot or line chart comparing predicted trips against actual trips for a specific time/location.