NYC Taxi Data Dimensional Model Documentation

This document lays out a dimensional model for analyzing NYC taxi trip data. It's designed to make queries run efficiently, helping with things like tracking revenue by hour, day, or month, spotting trends in pickup and drop-off locations, and measuring driver performance.

Dimensional Model Design

The data model follows a star schema consisting of one fact table and multiple dimension tables.

Fact Table

- fact_trips: Stores detailed trip records for both Yellow and Green taxis.

Column Name	Data Type	Description
TripID	STRING	Unique identifier for each trip
VendorID	INT	ID of the taxi service provider
PickupDateTime	TIMESTAMP	Date and time when the trip started
DropoffDateTime	TIMESTAMP	Date and time when the trip ended
PULocationID	INT	Pickup location ID
DOLocationID	INT	Drop-off location ID
PassengerCount	INT	Number of passengers
TripDistance	DOUBLE	Distance traveled in miles
FareAmount	DOUBLE	Fare charged for the trip
TipAmount	DOUBLE	Tip given by passengers
TotalAmount	DOUBLE	Total fare including tips and surcharges
PaymentType	INT	Payment method used
TripDuration	DOUBLE	Trip duration in minutes
TripSpeedMPH	DOUBLE	Average speed of the trip

Dimension Tables

- dim_rate_codes:Contains rate code information for different fare calculations.

Column Name	Data Type	Description
RatecodeID	INT	Rate code identifier
Description	STRING	Description of the rate code

- dim_zones:Stores taxi zone details used for trip location analysis.

Column Name	Data Type	Description
LocationID	INT	Unique zone identifier
Borough	STRING	Borough name
Zone	STRING	Specific zone name
Service_Zone	STRING	Type of service zone

- dim_payment_types: Defines payment methods used in taxi transactions.

Column Name	Data Type	Description
PaymentTypeID	INT	Unique ID for payment type
PaymentMethod	STRING	Description (e.g., Credit Card, Cash)

Slowly Changing Dimensions (SCD):

- **dim_zones (Taxi Zones - SCD Type 2):** When a zone name or borough is modified (e.g., a zone is renamed or reassigned to a different borough), a new record is added with effective_start and effective_end timestamps to track these changes. This ensures past trips are linked to the correct zone data at the time they occurred.
- **dim_rate_codes (Rate Codes - SCD Type 1):** Since rate codes rarely change, updates directly overwrite existing values without keeping historical records. This prevents unnecessary data expansion for stable reference data.
- **dim_payment_types (Payment Methods - SCD Type 2):** As new payment methods (such as digital wallets) emerge, a new record is created whenever a change occurs. This approach maintains historical integrity by preserving original payment types for past transactions.

Aggregation Tables

1.)Agg_revenue_by_time

- Stores revenue summaries at different time granularities.
- Used for analyzing hourly, daily, and monthly revenue trends.

Column Name	Data Type	Description
time_period	STRING	Time granularity (hourly/daily/monthly)
time_value	STRING	Actual time value (e.g. '2024-01-01 12:00:00')
total_revenue	DOUBLE	Sum of total_amount for the period

2.) Agg_trip_patterns

- Tracks total trips by day type and month.
- Supports analysis of weekday vs. weekend trip patterns.

Column Name	Data Type	Description
month	STRING	Year-month of trip (e.g., '2024-01')
is_weekend	STRING	'Weekday' or 'Weekend'
total_trips	INT	Count of trips for the period

3.) Agg_long_trips

- Stores long trips (greater than 10 miles) separately for in-depth analysis.
- Helps analyze long-distance travel patterns and pricing.

Column Name	Data Type	Description
pickup_datetime	TIMESTAMP	Start time of long trip
dropoff_datetime	TIMESTAMP	End time of long trip
PULocationID	INT	Pickup location ID
DOLocationID	INT	Dropoff location ID
trip_distance	DOUBLE	Trip distance in miles
total_amount	DOUBLE	Fare amount for long trip

Partitioning Strategy

- **Fact Table Partitioning:**
    - The trip data is divided by **year and month** based on the pickup date, making time-based searches faster.
    - Another partition is created based on the **pickup location**, which helps in running location-based queries more efficiently.
- **Aggregation Table Partitioning:**
    - **Revenue data** is split by **time period** (hourly, daily, monthly) so reports can be generated quickly.
    - **Trip patterns** are grouped by **month**, making it easier to analyze travel trends.
    - **Long trips** (over 10 miles) are stored separately from shorter trips, helping in analyzing extended rides without scanning all trips.

This setup ensures that queries run faster, supports better analysis, and keeps historical data organized for NYC Taxi trips.

Query Optimization Strategy

To ensure efficient processing of the NYC taxi trip data, we apply several optimization techniques in Spark. These help improve query performance, reduce processing time, and make analytics more scalable. Below are the key optimizations used:

- **Indexes on Foreign Keys:** Since the fact table contains references to dimension tables (such as PULocationID and DOLocationID), indexing these columns speeds up joins and lookups, reducing query execution time.
- **Broadcast Joins for Small Dimension Tables:** Some dimension tables, like dim_rate_codes and dim_payment_types, are small in size. Instead of

performing expensive shuffle operations, we use broadcast joins in Spark to improve join efficiency.

- **Filter Pushdown & Predicate Pushdown:** Queries are optimized to push filters as early as possible to ensure that only relevant data is read from storage. This minimizes the amount of data loaded and processed, improving performance.
- **Pre-aggregated Summary Tables:** Since common queries (like revenue by time or trip counts) are frequently used, we store pre-aggregated results in separate tables. This avoids real-time computation and speeds up reporting.