ETL Structure and SQL Queries (Cyclistic BI Capstone)

1️⃣ETL Structure

🔹 Extract:

- **Primary Dataset:** NYC Citi Bike Trips (CSV/SQL dump/Cloud bucket)
- **Secondary Dataset:** Census Bureau US Boundaries (GeoJSON/CSV)

🔹 Transform:

- Parse **datetime fields** into start_time, end_time, year, month, day, hour.
- Clean and anonymize **user identifiers** (if any).
- Geocode **latitude/longitude** to borough/zip using Census data join.
- Flag is_rainy_day by joining with weather data.
- Create trip_duration_minutes = TIMESTAMPDIFF(MINUTE, start_time, end_time).
- Categorize **user type** (subscriber / non-subscriber).
- **Aggregate** for:
  - Total trips by start station, end station.
  - Total trip minutes by destination.
  - Net inflow/outflow per station per day.
  - Trips by hour for peak usage.
  - Year-over-year trip counts for growth.

🔹 Load:

- Load transformed tables into your BI database (Snowflake, BigQuery, PostgreSQL, etc.)
- Create **materialized views** for daily_station_summary, monthly_trends, and

congestion_analysis.

2️⃣SQL Starter Queries

Replace citi_bike_trips with your actual staging table name.

Query 1: Trips by Starting Location

SQL

```sql
SELECT
  start_station_id,
  COUNT(*) AS total_trips,
  AVG(TIMESTAMPDIFF(MINUTE, start_time, end_time)) AS avg_trip_duration_minutes
FROM
  citi_bike_trips
WHERE
  YEAR(start_time) = 2015
GROUP BY
  start_station_id
ORDER BY
  total_trips DESC
LIMIT 20;
```

Query 2: Destination Popularity by Total Trip Minutes

SQL

```
SELECT
  end_station_id,
  SUM(TIMESTAMPDIFF(MINUTE, start_time, end_time)) AS total_trip_minutes,
  COUNT(*) AS trip_count
FROM
  citi_bike_trips
WHERE
  MONTH(start_time) IN (6, 7, 8) -- Summer months
GROUP BY
  end_station_id
ORDER BY
  total_trip_minutes DESC
LIMIT 10;
```

Query 3: Year-over-Year Growth

SQL

```
SELECT
  YEAR(start_time) AS trip_year,
  COUNT(*) AS total_trips
FROM
  citi_bike_trips
GROUP BY
  trip_year
ORDER BY
  trip_year ASC;
```

Query 4: Congestion Analysis (Net Inflow/Outflow)

SQL

```
WITH inflow AS (
  SELECT
    end_station_id AS station_id, DATE(end_time) AS trip_date, COUNT(*) AS trips_in
  FROM
    citi_bike_trips
  GROUP BY
    station_id, trip_date
),
outflow AS (
  SELECT
    start_station_id AS station_id, DATE(start_time) AS trip_date, COUNT(*) AS trips_out
  FROM
    citi_bike_trips
  GROUP BY
    station_id, trip_date
)
SELECT
  COALESCE(inflow.station_id, outflow.station_id) AS station_id,
  COALESCE(inflow.trip_date, outflow.trip_date) AS trip_date,
  IFNULL(trips_in, 0) AS trips_in,
  IFNULL(trips_out, 0) AS trips_out,
  IFNULL(trips_in, 0) - IFNULL(trips_out, 0) AS net_inflow
FROM
  inflow
FULL OUTER JOIN
  outflow
ON
  inflow.station_id = outflow.station_id AND inflow.trip_date = outflow.trip_date
ORDER BY
  trip_date, station_id;
```

Query 5: Peak Usage by Hour of Day

SQL

```
SELECT
  HOUR(start_time) AS trip_hour,
  COUNT(*) AS total_trips
FROM
  citi_bike_trips
GROUP BY
  trip_hour
ORDER BY
  total_trips DESC;
```

Query 6: Weather Impact Analysis

Assuming you have a weather table with date and is_rainy:

SQL

```
SELECT
  cb.start_station_id,
  DATE(cb.start_time) AS trip_date,
  COUNT(*) AS total_trips,
  w.is_rainy
FROM
  citi_bike_trips cb
LEFT JOIN
  weather w
ON
  DATE(cb.start_time) = w.date
GROUP BY
  cb.start_station_id, trip_date, w.is_rainy
ORDER BY
  trip_date, total_trips DESC;
```

3️⃣Recommended Materialized Views:

- ✅ mv_daily_station_summary – daily trips, net inflow/outflow per station
- ✅ mv_monthly_trends – monthly trips, subscriber vs. non-subscriber trends
- ✅ mv_weather_impact – trips on rainy vs. clear days
- ✅ mv_peak_usage_hour – aggregated hourly usage for dashboard insights