#Request1- 1: City-Level Fare and Trip Summary Report

#Generate a report that displays the total trips, average fare per km, average fare per trip,

and the percentage contribution of each city's trips to the overall trips.

This report will help in assessing trip volume, pricing efficiency, and each city's contribution to the overall trip count

```
SELECT
   dc.city_name,
   COUNT(ft.trip_id) AS total_trips,
   ROUND(SUM(ft.fare_amount) / NULLIF(SUM(ft.distance_travelled_km), 0), 2) AS avg_fare_per_km,
   ROUND(SUM(ft.fare_amount) / NULLIF(COUNT(ft.trip_id), 0), 2) AS avg_fare_per_trip,
   ROUND(
      (COUNT(ft.trip_id) * 100.0) / NULLIF((SELECT COUNT(trip_id) FROM fact_trips), 0),
      2
   ) AS percentage_contribution_to_total_trips
FROM
   dim_city dc
LEFT JOIN
   fact_trips ft ON dc.city_id = ft.city_id
GROUP BY
   dc.city_name
ORDER BY
   total_trips DESC;
```

REQUEST 2: Monthly City-Level Trips Target Performance Report

#e If actual trips are greater than target trips, mark it as "Above Target".

#If actual trips are less than or equal to target trips, mark it as "Below Target".

Additionally, calculate the % difference between actual and target trips to quantify the performance gap.

```
SELECT
    cities.city_name,
    dates.month_name,
    COUNT(trips.trip_id) AS actual_trips,
    targets.total_target_trips AS target_trips,
    CASE
        WHEN COUNT(trips.trip_id) > targets.total_target_trips THEN 'Above Target'
        ELSE 'Below Target'
    END AS performance_status,
    ROUND(
        ((COUNT(trips.trip_id) - targets.total_target_trips) * 100.0) / NULLIF(targets.total_target_trips,
0),
        2
    ) AS percentage_difference
FROM
    trips_db.dim_city AS cities -- Explicitly reference the trips_db database
LEFT JOIN
    trips_db.fact_trips AS trips -- Explicitly reference the trips_db database
    ON cities.city_id = trips.city_id
LEFT JOIN
    trips_db.dim_date AS dates -- Explicitly reference the trips_db database
    ON trips.date = dates.date
LEFT JOIN
    targets_db.monthly_target_trips AS targets -- Explicitly reference the targets_db database
    ON cities.city_id = targets.city_id
      AND dates.start_of_month = targets.month
GROUP BY
    cities.city_name, dates.month_name, targets.total_target_trips
ORDER BY
    cities.city_name, dates.month_name;
```

#Request 3 : City-Level Repeat Passenger Trip Frequency Report

Generate a report that shows the percentage distribution of repeat passengers by the number of trips they have taken in each city.

#Calculate the percentage of repeat passengers who took 2 trips, 3 trips, and so on, up to 10 trips.

```
SELECT
cities.city_name,
ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 2 THEN repeat_trip.repeat_passenger_count ELSE 0 END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
) AS "2-Trips",
ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 3 THEN repeat_trip.repeat_passenger_count ELSE 0 END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
) AS "3-Trips",
ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 4 THEN repeat_trip.repeat_passenger_count ELSE 0 END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
) AS "4-Trips",
ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 5 THEN repeat_trip.repeat_passenger_count ELSE 0 END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
) AS "5-Trips",
ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 6 THEN repeat_trip.repeat_passenger_count ELSE 0 END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
```

```
    2
  ) AS "6-Trips",
  ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 7 THEN repeat_trip.repeat_passenger_count ELSE 0
END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
  ) AS "7-Trips",
  ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 8 THEN repeat_trip.repeat_passenger_count ELSE 0
END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
  ) AS "8-Trips",
  ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 9 THEN repeat_trip.repeat_passenger_count ELSE 0
END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
  ) AS "9-Trips",
  ROUND(
    SUM(CASE WHEN repeat_trip.trip_count = 10 THEN repeat_trip.repeat_passenger_count ELSE 0
END) * 100.0 / NULLIF(SUM(repeat_trip.repeat_passenger_count), 0),
    2
  ) AS "10-Trips"
FROM
  trips_db.dim_city AS cities
LEFT JOIN
  trips_db.dim_repeat_trip_distribution AS repeat_trip
  ON cities.city_id = repeat_trip.city_id
GROUP BY
  cities.city_name
ORDER BY
  cities.city_name asc;
```

---

request 4 Identify Cities with Highest and Lowest Total New Passengers

Generate a report that calculates the total new passengers for each city and ranks them based on this value.

Identify the top 3 cities with the highest number of new passengers as well as the bottom 3 cities with the lowest number of new passengers, categorising them as "Top 3" or "Bottom 3" accordingly.

```
WITH RankedCities AS (
    SELECT
        cities.city_name,
        SUM(passenger_summary.new_passengers) AS total_new_passengers,
        RANK() OVER (ORDER BY SUM(passenger_summary.new_passengers) DESC) AS rank_desc,
        RANK() OVER (ORDER BY SUM(passenger_summary.new_passengers) ASC) AS rank_asc
    FROM
        trips_db.dim_city AS cities
    LEFT JOIN
        trips_db.fact_passenger_summary AS passenger_summary
        ON cities.city_id = passenger_summary.city_id
    GROUP BY
        cities.city_name
),
CategorizedCities AS (
    SELECT
        city_name,
        total_new_passengers,
        CASE
            WHEN rank_desc <= 3 THEN 'Top 3'
            WHEN rank_asc <= 3 THEN 'Bottom 3'
            ELSE NULL
        END AS city_category
```

```
    FROM
        RankedCities
)
SELECT
    city_name,
    total_new_passengers,
    city_category
FROM
    CategorizedCities
WHERE
    city_category IS NOT NULL
ORDER BY
    city_category, total_new_passengers DESC;
    ------------------------------------------------
    #request 5 : Identify Month with Highest Revenue for Each City
 # Generate a report that identifies the month with the highest revenue for each city.
 # For each city, display the month_name, the revenue amount for that month, and the percentage contribution of that month's revenue to the city's total revenue.


   WITH CityMonthlyRevenue AS (
    SELECT
        cities.city_name,
        dates.month_name,
        SUM(trips.fare_amount) AS monthly_revenue
    FROM
        trips_db.dim_city AS cities
    LEFT JOIN
        trips_db.fact_trips AS trips
        ON cities.city_id = trips.city_id
    LEFT JOIN
```

```
        trips_db.dim_date AS dates
        ON trips.date = dates.date
    GROUP BY
        cities.city_name, dates.month_name
),
CityTotalRevenue AS (
    SELECT
        city_name,
        SUM(monthly_revenue) AS total_revenue
    FROM
        CityMonthlyRevenue
    GROUP BY
        city_name
),
HighestMonthlyRevenue AS (
    SELECT
        cmr.city_name,
        cmr.month_name AS highest_revenue_month,
        cmr.monthly_revenue,
        ROUND(
            (cmr.monthly_revenue * 100.0) / ctr.total_revenue,
            2
        ) AS percentage_contribution
    FROM
        CityMonthlyRevenue AS cmr
    INNER JOIN
        CityTotalRevenue AS ctr
        ON cmr.city_name = ctr.city_name
    WHERE
        cmr.monthly_revenue = (
            SELECT MAX(cmr2.monthly_revenue)
```

```
        FROM CityMonthlyRevenue AS cmr2
        WHERE cmr2.city_name = cmr.city_name
    )
)
SELECT
  city_name,
  highest_revenue_month,
  monthly_revenue AS revenue,
  percentage_contribution
FROM
  HighestMonthlyRevenue
ORDER BY
  city_name;
```

#Request 6 : 6: Repeat Passenger Rate Analysis

#Generate a report that calculates two metrics:

#1. Monthly Repeat Passenger Rate: Calculate the repeat passenger rate for each city and month by comparing the number of repeat passengers to the total passengers.

#2. City-wide Repeat Passenger Rate: Calculate the overall repeat passenger rate for each city, considering all passengers across months.

```
WITH MonthlyPassengerStats AS (
  SELECT
    cities.city_name,
    dates.month_name,
    COALESCE(SUM(summary.total_passengers), 0) AS total_passengers,
    COALESCE(SUM(summary.repeat_passengers), 0) AS repeat_passengers
  FROM
    trips_db.dim_city AS cities
  LEFT JOIN
```

```
    trips_db.fact_passenger_summary AS summary
    ON cities.city_id = summary.city_id
  LEFT JOIN
    trips_db.dim_date AS dates
    ON summary.month = dates.start_of_month
  GROUP BY
    cities.city_name, dates.month_name
),
CityRepeatRate AS (
  SELECT
    city_name,
    SUM(total_passengers) AS total_passengers,
    SUM(repeat_passengers) AS repeat_passengers,
    ROUND(
      (SUM(repeat_passengers) * 100.0) / NULLIF(SUM(total_passengers), 0),
      2
    ) AS city_repeat_passenger_rate
  FROM
    MonthlyPassengerStats
  GROUP BY
    city_name
)
SELECT
  mps.city_name,
  mps.month_name AS month,
  mps.total_passengers,
  mps.repeat_passengers,
  ROUND(
    (mps.repeat_passengers * 100.0) / NULLIF(mps.total_passengers, 0),
    2
  ) AS monthly_repeat_passenger_rate,
```

```
    crr.city_repeat_passenger_rate
FROM
    MonthlyPassengerStats AS mps
LEFT JOIN
    CityRepeatRate AS crr
    ON mps.city_name = crr.city_name
ORDER BY
    mps.city_name, mps.month_name;
```