BUSINESS REQUEST
SQL QUERIES

Business Request - 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.

Fields:

- city_name
- total_trips
- avg_fare_per_km
- avg_fare_per_trip
- %_contribution_to_total_trips

```sql
SELECT
    ct.city_name,
    COUNT(ft.trip_id) AS total_trips,
    SUM(ft.fare_amount) / SUM(ft.distance_travelled_km) AS avg_fare_per_km,
    SUM(ft.fare_amount) / COUNT(ft.trip_id) AS avg_fare_per_trip,
    ROUND(
        (COUNT(ft.trip_id) /
        (SELECT COUNT(*) FROM trips_db.fact_trips) * 100),
        2
    ) AS percentage_contribution_to_total_trips
FROM
    trips_db.fact_trips AS ft
JOIN
    trips_db.dim_city AS ct
ON
    ft.city_id = ct.city_id
GROUP BY
    ct.city_name
ORDER BY
    COUNT(ft.trip_id) DESC;
```

	city_name	total_trips	avg_fare_per_km	avg_fare_per_trip	percentage_contribution_to_total_trips
▶	Jaipur	76888	16.1182	483.9181	18.05
	Lucknow	64299	11.7622	147.1804	15.10
	Surat	54843	10.6638	117.2729	12.88
	Kochi	50702	13.9305	335.2451	11.90
	Indore	42456	10.8977	179.8386	9.97
	Chandigarh	38981	12.0622	283.6870	9.15
	Vadodara	32026	10.2942	118.5662	7.52
	Visakhapatnam	28366	12.5332	282.6723	6.66
	Coimbatore	21104	11.1476	166.9822	4.96
	Mysore	16238	15.1366	249.7072	3.81

Business Request - 2: Monthly City-Level Trips Target Performance Report

Generate a report that evaluates the target performance for trips at the monthly and city level. For each city and month, compare the actual total trips with the target trips and categorise the performance as follows:

- 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.

Fields:

- City_name
- month_name
- actual_trips
- target_trips
- performance_status
- %_difference

```sql
SELECT
    ct.city_name,
    dt.month_name,
    COUNT(TRIP.trip_id) AS Actual_trips,
    Target.total_target_trips AS Target_trips,
    CASE
        WHEN COUNT(TRIP.trip_id) >= Target.total_target_trips
        THEN 'Above Target'
        ELSE 'Below Target'
    END AS Performance_status,
    ROUND(
        ((COUNT(TRIP.trip_id) - Target.total_target_trips) * 100.0 /
Target.total_target_trips),
        2
    ) AS Percentage_difference
FROM
    trips_db.dim_city AS ct
JOIN
    trips_db.fact_trips AS TRIP
    ON ct.city_id = TRIP.city_id
JOIN
    trips_db.dim_date AS dt
    ON TRIP.date = dt.date
JOIN
    trips_db.monthly_target_trips AS Target
    ON TRIP.city_id = Target.city_id
    AND MONTH(TRIP.date) = MONTH(Target.month)
WHERE
    dt.month_name IN ('January', 'February', 'March', 'April', 'May', 'June')
GROUP BY
    ct.city_name,
    dt.month_name,
    Target.total_target_trips,
    Target.month
ORDER BY
    ct.city_name,
    FIELD(dt.month_name, 'January', 'February', 'March', 'April', 'May',
'June');
```

city_name	month_name	Actual_trips	Target_trips	Performance_status	Percentage_difference
Chandigarh	January	6810	7000	Below Target	-2.71
Chandigarh	February	7387	7000	Above Target	5.53
Chandigarh	March	6569	7000	Below Target	-6.16
Chandigarh	April	5566	6000	Below Target	-7.23
Chandigarh	May	6620	6000	Above Target	10.33
Chandigarh	June	6029	6000	Above Target	0.48
Coimbatore	January	3651	3500	Above Target	4.31
Coimbatore	February	3404	3500	Below Target	-2.74
Coimbatore	March	3680	3500	Above Target	5.14
Coimbatore	April	3661	3500	Above Target	4.60
Coimbatore	May	3550	3500	Above Target	1.43
Coimbatore	June	3158	3500	Below Target	-9.77
Indore	January	6737	7000	Below Target	-3.76
Indore	February	7210	7000	Above Target	3.00
Indore	March	7019	7000	Above Target	0.27
Indore	April	7415	7500	Below Target	-1.13
Indore	May	7787	7500	Above Target	3.83
Indore	June	6288	7500	Below Target	-16.16
Jaipur	January	14976	13000	Above Target	15.20

Business 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.

Each column should represent a trip count category, displaying the percentage of repeat passengers who fall into that category out of the total repeat passengers for that city.

This report will help identify cities with high repeat trip frequency, which can indicate strong customer loyalty or frequent usage patterns.

- Fields: city_name, 2-Trips, 3-Trips, 4-Trips, 5-Trips, 6-Trips, 7-Trips, 8-Trips, 9-Trips, 10-Trips

```sql
WITH CityTripFrequency AS (
    SELECT
        ct.city_name,
        drt.trip_count,
        SUM(drt.repeat_passenger_count) AS repeat_passenger_sum
    FROM
        trips_db.dim_city AS ct
    JOIN
        trips_db.dim_repeat_trip_distribution AS drt
        ON ct.city_id = drt.city_id
    WHERE
        drt.trip_count BETWEEN 2 AND 10
    GROUP BY
        ct.city_name, drt.trip_count
),
CityTotalRepeatPassengers AS (
    SELECT
        city_name,
        SUM(repeat_passenger_sum) AS total_repeat_passengers
    FROM
        CityTripFrequency
    GROUP BY
        city_name
),
CityTripPercentage AS (
    SELECT
        ctf.city_name,
        ctf.trip_count,
        ctf.repeat_passenger_sum,
        ROUND((ctf.repeat_passenger_sum / ctrp.total_repeat_passengers) * 100,
2) AS total_percentage
    FROM
        CityTripFrequency AS ctf
    JOIN
        CityTotalRepeatPassengers AS ctrp
        ON ctf.city_name = ctrp.city_name
)
SELECT
    city_name,
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '2-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "2-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '3-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "3-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '4-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "4-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '5-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "5-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '6-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "6-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '7-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "7-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '8-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "8-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '9-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "9-Trips",
    CONCAT(FORMAT(SUM(CASE WHEN trip_count = '10-Trips' THEN total_percentage
ELSE 0 END), 2), '%') AS "10-Trips"
FROM
    CityTripPercentage
GROUP BY
    city_name;
```

city_name	2-Trips	3-Trips	4-Trips	5-Trips	6-Trips	7-Trips	8-Trips	9-Trips	10-Trips
Visakhapatnam	51.25%	24.96%	9.98%	5.44%	3.19%	1.98%	1.39%	0.88%	0.92%
Chandigarh	32.31%	19.25%	15.74%	12.21%	7.42%	5.48%	3.47%	2.33%	1.79%
Surat	9.76%	14.26%	16.55%	19.75%	18.45%	11.89%	6.24%	1.74%	1.35%
Vadodara	9.87%	14.17%	16.52%	18.06%	19.08%	12.86%	5.78%	2.05%	1.61%
Mysore	48.75%	24.44%	12.73%	5.82%	4.06%	1.76%	1.42%	0.54%	0.47%
Kochi	47.67%	24.35%	11.81%	6.48%	3.91%	2.11%	1.65%	1.21%	0.81%
Indore	34.34%	22.69%	13.40%	10.34%	6.85%	5.24%	3.26%	2.38%	1.51%
Jaipur	50.14%	20.73%	12.12%	6.29%	4.13%	2.52%	1.90%	1.20%	0.97%
Coimbatore	11.21%	14.82%	15.56%	20.62%	17.64%	10.47%	6.15%	2.31%	1.22%
Lucknow	9.66%	14.77%	16.20%	18.42%	20.18%	11.33%	6.43%	1.91%	1.10%

Business 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.

Fields

- city_name
- total_new_passengers
- city_category ("Top 3" or "Bottom 3")

```sql
WITH CityNewPassengers AS (
    SELECT
        ct.city_name,
        SUM(ft.new_passengers) AS total_new_passengers
    FROM
        trips_db.dim_city AS ct
    JOIN
        trips_db.fact_passenger_summary AS ft
        ON ct.city_id = ft.city_id
    GROUP BY
        ct.city_name
),
RankedCities AS (
    SELECT
        city_name,
        total_new_passengers,
        RANK() OVER (ORDER BY total_new_passengers DESC) AS Ranking,
        RANK() OVER (ORDER BY total_new_passengers ASC) AS low_rank
    FROM
        CityNewPassengers
)
SELECT
    city_name,
    total_new_passengers,
    Ranking,
    CASE
        WHEN Ranking ⩽ 3 THEN 'Top 3'
        ELSE 'Bottom 3'
    END AS city_category
FROM
    RankedCities
WHERE
    Ranking ⩽ 3 OR low_rank ⩽ 3
ORDER BY
    total_new_passengers DESC,
    city_category ASC;
```

	city_name	total_new_passengers	Ranking	city_category
▶	Jaipur	45856	1	Top 3
	Kochi	26416	2	Top 3
	Chandigarh	18908	3	Top 3
	Surat	11626	8	Bottom 3
	Vadodara	10127	9	Bottom 3
	Coimbatore	8514	10	Bottom 3

Business 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.

Fields

- city_name
- highest_revenue_month
- revenue
- percentage_contribution (%)

```sql
WITH CityMonthlyRevenue AS (
    SELECT
        ct.city_name,
        DATE_FORMAT(ft.date, '%M') AS Month,
        SUM(ft.fare_amount) AS Revenue
    FROM
        trips_db.fact_trips AS ft,
        trips_db.dim_city AS ct
    WHERE
        ct.city_id = ft.city_id
    GROUP BY
        ct.city_name, Month
    ORDER BY
        ct.city_name, Month
),
CityTotalRevenue AS (
    SELECT
        city_name,
        SUM(Revenue) AS total_city_revenue
    FROM
        CityMonthlyRevenue
    GROUP BY
        city_name
),
CityHighestRevenueMonth AS (
    SELECT
        cmr.city_name,
        cmr.Month,
        cmr.Revenue,
        DENSE_RANK() OVER (PARTITION BY cmr.city_name ORDER BY cmr.Revenue
DESC) AS Rank
    FROM
        CityMonthlyRevenue AS cmr
)
SELECT
    chr.city_name,
    chr.Month AS Highest_Revenue_Month,
    chr.Revenue,
    ctr.total_city_revenue,
    CONCAT(FORMAT((chr.Revenue / ctr.total_city_revenue) * 100, 2), '%') AS '%
Contribution'
FROM
    CityHighestRevenueMonth AS chr
JOIN
    CityTotalRevenue AS ctr
    ON chr.city_name = ctr.city_name
WHERE
    chr.Rank = 1
ORDER BY
    chr.Revenue DESC;
```

city_name	Highest_Revenue_Month	Revenue	total_city_revenue	% Contribution
Jaipur	February	7747202	37207497	20.82%
Kochi	May	3333746	16997596	19.61%
Chandigarh	February	2108290	11058401	19.07%
Lucknow	February	1777269	9463551	18.78%
Visakhapatnam	April	1390682	8018282	17.34%
Indore	May	1380996	7635228	18.09%
Surat	April	1154909	6431599	17.96%
Mysore	May	745170	4054745	18.38%
Vadodara	April	706250	3797200	18.60%
Coimbatore	April	612431	3523992	17.38%

Business Request - 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.

These metrics will provide insights into monthly repeat trends as well as the overall repeat behaviour for each city.

Fields:

- city_name
- month
- total_passengers
- repeat_passengers
- monthly_repeat_passenger_rate (%): Repeat passenger rate at the city and month level
- city_repeat_passenger_rate (%): Overall repeat passenger rate for each city, aggregated across months

```sql
WITH MRPR AS (
    SELECT
        ct.city_name,
        MONTH(ft.month) AS month,
        ft.total_passengers,
        ft.repeat_passengers,
        CONCAT(FORMAT((ft.repeat_passengers / ft.total_passengers) * 100, 2),
'%') AS monthly_repeat_passenger_rate
    FROM
        trips_db.dim_city AS ct,
        fact_passenger_summary AS ft
    WHERE
        ct.city_id = ft.city_id
    GROUP BY
        ct.city_name, ft.total_passengers, ft.repeat_passengers, month
),
CRPR AS (
    SELECT
        ct.city_name,
        SUM(ft.total_passengers) AS city_total_passenger,
        SUM(ft.repeat_passengers) AS city_total_repeat_passenger,
        CONCAT(FORMAT((SUM(ft.repeat_passengers) / SUM(ft.total_passengers)) *
100, 2), '%') AS city_repeat_passenger_rate
    FROM
        trips_db.dim_city AS ct,
        fact_passenger_summary AS ft
    WHERE
        ct.city_id = ft.city_id
    GROUP BY
        ct.city_name
)
SELECT
    m.city_name,
    m.month,
    m.total_passengers,
    m.repeat_passengers,
    m.monthly_repeat_passenger_rate,
    c.city_repeat_passenger_rate
FROM
    MRPR AS m
JOIN
    CRPR AS c
ON
    m.city_name = c.city_name;
```

city_name	month	total_passengers	repeat_passengers	monthly_repeat_passenger_rate	city_repeat_passenger_rate
Visakhapatnam	1	3163	650	20.55%	28.61%
Visakhapatnam	2	3170	790	24.92%	28.61%
Visakhapatnam	3	3093	923	29.84%	28.61%
Visakhapatnam	4	2837	992	34.97%	28.61%
Visakhapatnam	5	2890	951	32.91%	28.61%
Visakhapatnam	6	2702	802	29.68%	28.61%
Chandigarh	1	4640	720	15.52%	21.14%
Chandigarh	2	4957	853	17.21%	21.14%
Chandigarh	3	4100	872	21.27%	21.14%
Chandigarh	4	3285	789	24.02%	21.14%
Chandigarh	5	3699	969	26.20%	21.14%
Chandigarh	6	3297	867	26.30%	21.14%
Surat	1	3616	1184	32.74%	42.63%
Surat	2	3567	1313	36.81%	42.63%
Surat	3	3440	1494	43.43%	42.63%
Surat	4	3394	1551	45.70%	42.63%
Surat	5	3217	1606	49.92%	42.63%
Surat	6	3030	1490	49.17%	42.63%