Ad hoc analysis

- So far, we have conducted exploratory data analysis, identifying trends in trips, ratings, and repeat passenger behavior.
- Now, we will address six ad hoc business requests from Goodcabs' Chief Operating Officer.
- These requests require SQL-based report generation to support data-driven decision-making.

Business request 1

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.

```sql
WITH CTE AS (
SELECT
    city_name city,
    COUNT(trip_id) AS total_trips,
    ROUND(AVG(fare_amount/distance_travelled_km), 1) AS avg_fare_per_km,
    ROUND(AVG(fare_amount), 1) AS avg_fare_per_trip
FROM
    fact_trips JOIN dim_city USING (city_id)
GROUP BY
    city_name
)
SELECT
    *, ROUND(total_trips * 100/ (SELECT count(*) FROM fact_trips), 1) trips_%_contribution
FROM
    CTE
ORDER BY
    total_trips DESC ;
```

Output : City-Level Fare and Trip Summary Report

city	total_trips	avg_fare_per_km	avg_fare_per_trip	trips_pct_contribution
Jaipur	76888	16.3	483.9	18.1
Lucknow	64299	12.1	147.2	15.1
Surat	54843	10.9	117.3	12.9
Kochi	50702	14.1	335.2	11.9
Indore	42456	11.1	179.8	10.0
Chandigarh	38981	12.2	283.7	9.2
Vadodara	32026	10.5	118.6	7.5
Visakhapatnam	28366	12.7	282.7	6.7
Coimbatore	21104	11.3	167.0	5.0
Mysore	16238	15.4	249.7	3.8

Business request 2

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

```sql
WITH actual_trip AS (
SELECT
  city_id, month(date) AS month, count(trip_id) AS actual_trips
FROM
  fact_trips
GROUP BY
city_id, month
),
target_trip AS (
SELECT
  city_id, month(month) AS month, total_target_trips AS target_trips
FROM
  targets_db.monthly_target_trips
)
SELECT
  city_name AS city, month, actual_trips, target_trips,
  CASE WHEN actual_trips > target_trips THEN "Above Target"
  WHEN actual_trips < = target_trips THEN "Below target"
  END performance_status,
  round((actual_trips - target_trips) *100 / actual_trips, 1) AS pct_difference
FROM
  actual_trip
    JOIN target_trip USING(city_id, month)
    JOIN dim_city USING (city_id)
ORDER BY
  city, month ;
```

Output : Monthly City-Level Trips Target Performance Report

city	month	actual_trips	target_trips	performance_status	pct_difference
Chandigarh	1	6810	7000	Below target	-2.8
Chandigarh	2	7387	7000	Above Target	5.2
Chandigarh	3	6569	7000	Below target	-6.6
Chandigarh	4	5566	6000	Below target	-7.8
Chandigarh	5	6620	6000	Above Target	9.4
Chandigarh	6	6029	6000	Above Target	0.5
Coimbatore	1	3651	3500	Above Target	4.1
Coimbatore	2	3404	3500	Below target	-2.8
Coimbatore	3	3680	3500	Above Target	4.9
Coimbatore	4	3661	3500	Above Target	4.4
Coimbatore	5	3550	3500	Above Target	1.4
Coimbatore	6	3158	3500	Below target	-10.8
Indore	1	6737	7000	Below target	-3.9
Indore	2	7210	7000	Above Target	2.9
Indore	3	7019	7000	Above Target	0.3
Indore	4	7415	7500	Below target	-1.1
Indore	5	7787	7500	Above Target	3.7

Business request 3

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.

```
WITH trip_count AS
(
SELECT city_id,
SUM(CASE WHEN trip_count = '2-Trips' THEN repeat_passenger_count END) AS 2_trips,
SUM(CASE WHEN trip_count = '3-Trips' THEN repeat_passenger_count END) AS 3_trips,
SUM(CASE WHEN trip_count = '4-Trips' THEN repeat_passenger_count END) AS 4_trips,
SUM(CASE WHEN trip_count = '5-Trips' THEN repeat_passenger_count END) AS 5_trips,
SUM(CASE WHEN trip_count = '6-Trips' THEN repeat_passenger_count END) AS 6_trips,
SUM(CASE WHEN trip_count = '7-Trips' THEN repeat_passenger_count END) AS 7_trips,
SUM(CASE WHEN trip_count = '8-Trips' THEN repeat_passenger_count END) AS 8_trips,
SUM(CASE WHEN trip_count = '9-Trips' THEN repeat_passenger_count END) AS 9_trips,
SUM(CASE WHEN trip_count = '10-Trips' THEN repeat_passenger_count END) AS 10_trips,
SUM(repeat_passenger_count) AS TOTAL
FROM
  dim_repeat_trip_distribution
GROUP BY
  city_id
)
SELECT city_name AS city,
ROUND(2_trips * 100/TOTAL,1) 2_trips,
ROUND(3_trips * 100/TOTAL,1) 3_trips,
ROUND(4_trips * 100/TOTAL,1) 4_trips,
ROUND(5_trips * 100/TOTAL,1) 5_trips,
ROUND(6_trips * 100/TOTAL,1) 6_trips,
ROUND(7_trips * 100/TOTAL,1) 7_trips,
ROUND(8_trips * 100/TOTAL,1) 8_trips,
ROUND(9_trips * 100/TOTAL,1) 9_trips,
ROUND(10_trips * 100/TOTAL,1) 10_trips
FROM
  trip_count JOIN dim_city USING (city_id) ;
```

Output : City-Level Repeat Passenger Trip Frequency Report

All values are in percentage

city	2_trips	3_trips	4_trips	5_trips	6_trips	7_trips	8_trips	9_trips	10_trips
Visakhapatnam	51.3	25.0	10.0	5.4	3.2	2.0	1.4	0.9	0.9
Chandigarh	32.3	19.3	15.7	12.2	7.4	5.5	3.5	2.3	1.8
Surat	9.8	14.3	16.6	19.7	18.5	11.9	6.2	1.7	1.4
Vadodara	9.9	14.2	16.5	18.1	19.1	12.9	5.8	2.0	1.6
Mysore	48.7	24.4	12.7	5.8	4.1	1.8	1.4	0.5	0.5
Kochi	47.7	24.4	11.8	6.5	3.9	2.1	1.7	1.2	0.8
Indore	34.3	22.7	13.4	10.3	6.8	5.2	3.3	2.4	1.5
Jaipur	50.1	20.7	12.1	6.3	4.1	2.5	1.9	1.2	1.0
Coimbatore	11.2	14.8	15.6	20.6	17.6	10.5	6.2	2.3	1.2
Lucknow	9.7	14.8	16.2	18.4	20.2	11.3	6.4	1.9	1.1

Business request 4

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, categorizing them as "Top 3" or "Bottom 3" accordingly.

```
WITH new_passengers_count AS(
SELECT
  city_name AS city,
  SUM(new_passengers) AS total_new_passengers,
  dense_rank() over(order by SUM(new_passengers) DESC) AS rnk
FROM
  fact_passenger_summary JOIN dim_city USING(city_id)
GROUP BY
  city_name
),
city_rank AS
(
SELECT
  *,
  CASE
    WHEN rnk < = 3 THEN "Top 3"
    WHEN rnk > = 8 THEN "Bottom 3"
    END AS city_category
FROM
  new_passengers_count
)
SELECT city, total_new_passengers, city_category
FROM city_rank
WHERE city_category IN ("Top 3","Bottom 3")
```

Output : Cities with Highest and Lowest Total New Passengers

city	total_new_passengers	city_category
Jaipur	45856	Top 3
Kochi	26416	Top 3
Chandigarh	18908	Top 3
Surat	11626	Bottom 3
Vadodara	10127	Bottom 3
Coimbatore	8514	Bottom 3

Business request 5

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.

```sql
WITH monthly_revenue AS (
SELECT
  city_id,
  MONTHNAME(date) AS month,
  ROUND(SUM(fare_amount)/1000000,2) AS revenue
FROM
  fact_trips
GROUP BY
  MONTH, city_id
),
monthly_ranking AS (
SELECT
  city_id, month,
    revenue,
  round(revenue * 100/sum(revenue) OVER(PARTITION BY city_id), 1) pct_contribution,
  DENSE_RANK() OVER(PARTITION BY city_id ORDER BY revenue DESC) rnk
FROM
  monthly_revenue
)
SELECT
  city_name AS city, month, revenue AS revenue_in_millions, pct_contribution
FROM
  monthly_ranking JOIN dim_city USING (city_id)
WHERE rnk = 1
ORDER BY city ;
```

Output : Month with Highest Revenue for Each City

city	month	revenue_in_millions	pct_contribution
Chandigarh	February	2.11	19.1
Coimbatore	April	0.61	17.3
Coimbatore	March	0.61	17.3
Coimbatore	January	0.61	17.3
Indore	May	1.38	18.0
Jaipur	February	7.75	20.8
Kochi	May	3.33	19.6
Lucknow	February	1.78	18.8
Mysore	May	0.75	18.5
Surat	April	1.15	17.9
Vadodara	April	0.71	18.7
Visakhapatnam	April	1.39	17.4
Visakhapatnam	March	1.39	17.4

Business request 6

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.

```sql
WITH monthly_rpr AS (
SELECT
    city_name AS city,
    month(month) AS month,
    total_passengers,
    repeat_passengers,
    ROUND(repeat_passengers * 100 / (
                                SELECT
                                    sum(repeat_passengers)
                                FROM
                                    fact_passenger_summary), 1) monthly_repeat_passenger_rate
FROM
    fact_passenger_summary JOIN dim_City USING (city_id)

)
SELECT
*,
ROUND(SUM(monthly_repeat_passenger_rate) OVER(PARTITION BY city), 1) city_repeat_passenger_rate
FROM
    monthly_rpr
ORDER BY
    city, month ;
```

Output : Repeat Passenger Rate Analysis

city_name	month	total_passengers	repeat_passengers	monthly_repeat_passenger_rate	city_repeat_passenger_rate
Chandigarh	1	4640	720	1.2	8.3
Chandigarh	2	4957	853	1.4	8.3
Chandigarh	3	4100	872	1.4	8.3
Chandigarh	4	3285	789	1.3	8.3
Chandigarh	5	3699	969	1.6	8.3
Chandigarh	6	3297	867	1.4	8.3
Coimbatore	1	2214	392	0.6	4.2
Coimbatore	2	1993	346	0.6	4.2
Coimbatore	3	1965	427	0.7	4.2
Coimbatore	4	1722	480	0.8	4.2
Coimbatore	5	1543	504	0.8	4.2
Coimbatore	6	1628	402	0.7	4.2

GOOD CABS

THANK YOU

LinkedIn

Portfolio