



GoodCabs

Created by : Punit Kumawat

*Date : 11 April 2025*



**Resume Projects**

**Provided Insights to Chief of Operations in Transportation Domain**

**Goodcabs**

**Ad-hoc Analysis**



**Domain: Transportation & Mobility**


**Function: Operations**




# Goodcabs

## Ad-hoc Analysis




 **Play Store**

 **Goodcabs : Book Cabs**

**Install**


4.8★  
2M review

  
42MB

3+






  
Rated for 3+

100M+  
Download

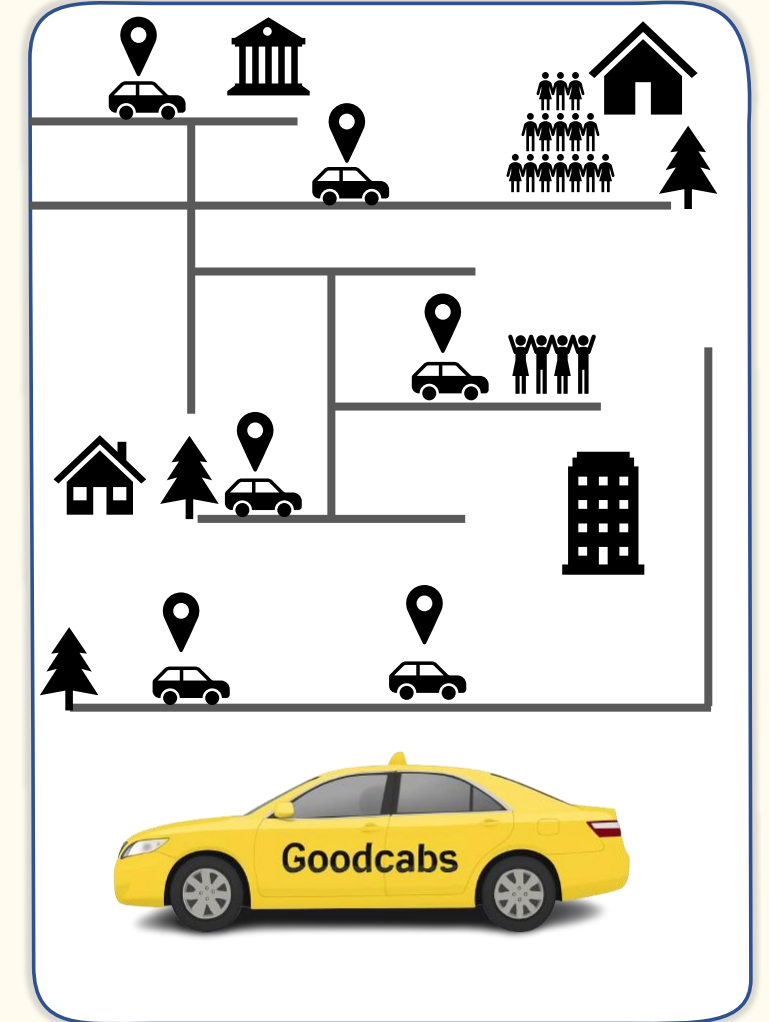
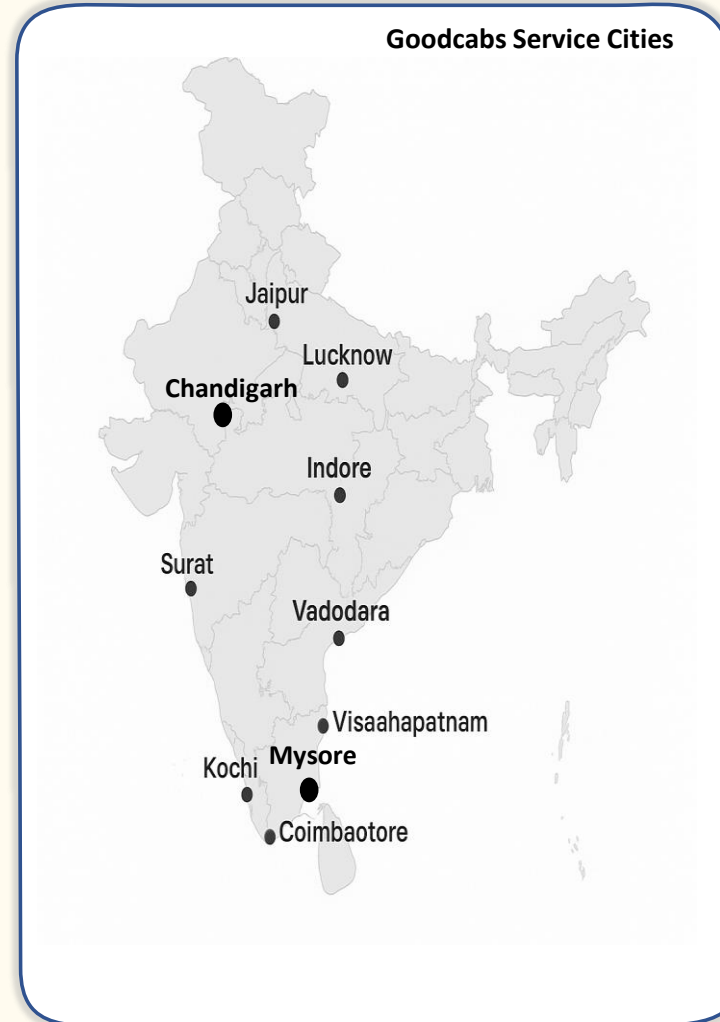
**About this app** 

Easy & Fast Online booking of Cabs.

**Rate this app**

**Write a reviews**



# About Company

- **Established:** 2 years ago
- **Type:** Cab service company
- **Focus Area:** Tier-2 cities across India
- **Mission:**
  - Empower **local drivers** to earn a **sustainable livelihood**
  - Provide **excellent service** to passengers

## •**Unique Approach:**

Unlike other cab companies, Goodcabs focuses on **supporting drivers in their hometowns**, ensuring both **economic growth** and **passenger satisfaction**.

## •**Current Operations:**

Goodcabs operates in **10 Tier-2 cities**: Jaipur, Lucknow, Surat, Kochi, Indore, Chandigarh, Vadodara, Visakhapatnam, Coimbatore, Mysore

## •**Vision for 2024:**

- Set **ambitious performance targets**
- Aim to **expand reach**, enhance **service quality**, and boost **passenger satisfaction**



# Objective of Analysis



Goodcabs has assigned us to perform an **Ad-hoc Analysis** to understand their business needs and support better decision-making.

**The objectives of this analysis are:**

- To understand the **business requirements** provided by the company
- To generate **insights** from the given data
- To identify issues related to **operations, passenger satisfaction, and performance**
- To support Goodcabs' **2024 growth strategy**



# **Business Requirements**

**Business Request – 1** : City-Level Fare and Trip Summary Report.

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

**Business Request –3**: City-Level Repeat Passenger Trip Frequency Report.

**Business Request –4**: Identify Cities with Highest and Lowest Total New Passengers.

**Business Request –5**: Identify Month With Highest Revenue for Each City.

**Business Request –6**: Repeat Passenger Rate Analysis.



# Data Sources

I have been provided with two **SQL text files** for performing ad-hoc analysis. The names of the databases are **trips\_db** and **targets\_db**.

**trips\_db:** This database contains both detailed and aggregated data on trips, passenger types, and repeat trip behavior for Goodcabs' operations across tier-2 cities. It organizes trip data by city, month, and day type (weekday or weekend), enabling comprehensive analysis of travel patterns, passenger demographics, and repeat usage trends.

**Let's take a look at the tables in the trips\_db database:**

dim\_city

dim\_date

fact\_passenger\_summary (Aggregated Data)

dim\_repeat\_trip\_distribution (Aggregated Data)

fact\_trips

**targets\_db:** This database holds Goodcabs' monthly targets for each city, including goals for trip counts, new passenger acquisition, and average passenger ratings. It enables performance evaluations against key benchmarks set by the company.

**Let's take a look at the tables in the targets\_db database:**

city\_target\_passenger\_rating

monthly\_target\_new\_passengers

monthly\_target\_trips

**Aggregated data** is the type of data that is combined based on a category (such as city or month) to create a summarized report.



Let's now go through each of our tables one by one to understand them better:

**dim\_city** ➡ **Purpose:** This table provides city-specific details, enabling location-based analysis of trips and passenger behavior across Goodcabs' operational areas.

- **city\_id:** A unique identifier for each city, using a standardized code (e.g., RJ01 for Jaipur).
- **city\_name:** The name of the city where Goodcabs operates (e.g., Jaipur, Lucknow), used for location-based grouping and insights.

**dim\_date** ➡ **Purpose:** This table provides date-specific details that enable time-based grouping and analysis, helping to identify patterns across days, months, and weekends versus weekdays.

- **date:** The specific date for each entry (formatted as YYYY-MM-DD), used for daily-level analysis and to join with trip data.
- **start\_of\_month:** The first day of the month to which the date belongs, useful for grouping data by month in analyses.
- **month\_name:** The name of the month (e.g., January, February) corresponding to the date, helpful for month-based aggregations.
- **day\_type:** Indicator of whether the date is a weekday or weekend, enabling analysis of demand and trip patterns across different day types.

**fact\_passenger\_summary (Aggregated Data)**

↙ ➡ **Purpose:** This table provides an aggregated summary of passenger counts for each city by month. It includes data on total passengers, new passengers, and repeat passengers, giving a high-level overview of passenger distribution patterns across different locations and times.

- **month:** The start date of the month (formatted as YYYY-MM-DD). All metrics in this table are aggregated for the entire month, aligning with other time-based analyses.
- **city\_id:** Unique identifier for the city.
- **total\_passengers:** The aggregated count of all passengers (both new and repeat) for the specified city and month.
- **new\_passengers:** The count of passengers taking their first ride in the specified city and month.
- **repeat\_passengers:** The count of passengers who have taken at least one ride in previous months and returned to ride again in the specified city and month.





### dim\_repeat\_trip\_distribution (Aggregated Data)



**Purpose:** This table provides a breakdown of repeat trip behavior, aggregated by month and city. It details how many times repeat passengers rode within the given month, categorized by trip frequency. To keep things simple and ensure uniformity across cities, we have included trip frequencies up to a maximum of 10 trips per month. This allows for an analysis of repeat trip patterns at a granular level.

- **month:** The start date of the month (formatted as YYYY-MM-DD). All metrics in this table represent aggregated values for the entire month, allowing alignment with other date-based analyses.
- **city\_id:** Unique identifier for the city.
- **trip\_count:** The number of trips taken by repeat passengers within the specified city and month (e.g., "3-Trips" for passengers who took 3 trips).
- **repeat\_passenger\_count:** The count of repeat passengers associated with each specified trip\_count for the city and month.

### fact\_trips



**Purpose:** This table provides detailed information on each individual trip, including trip-specific metrics like distance, fare, and ratings, which can be used for granular trip-level analysis.

- **trip\_id:** Unique identifier for each trip.
- **date:** The exact date of the trip (formatted as YYYY-MM-DD).
- **city\_id:** Unique identifier for the city where the trip took place.
- **passenger\_type:** Indicates if the passenger is new (first-time rider) or repeat (returning rider).
- **distance\_travelled (km):** Total distance covered in the trip, measured in kilometers.
- **fare\_amount:** Total fare amount paid by the passenger for the trip, in local currency.
- **passenger\_rating:** Rating provided by the passenger for the trip, on a scale from 1 to 10.
- **driver\_rating:** Rating provided by the driver for the passenger, on a scale from 1 to 10.





### city\_target\_passenger\_rating

- **city\_id**: Unique identifier for each city.
- **target\_avg\_passenger\_rating**: The target average passenger rating that Goodcabs aims to achieve for each city, used to monitor customer satisfaction.

### monthly\_target\_new\_passengers

- **month**: The start date of the target month (formatted as YYYY-MM-DD) to align with other time-based data.
- **city\_id**: Unique identifier for each city.
- **target\_new\_passengers**: The target number of new passengers Goodcabs aims to acquire in each city for the specified month, supporting growth tracking.

### monthly\_target\_trips

- **month**: The start date of the target month (formatted as YYYY-MM-DD) for easy alignment with monthly performance data.
- **city\_id**: Unique identifier for each city.
- **total\_target\_trips**: The target number of total trips Goodcabs aims to achieve for each city and month, enabling assessment of trip volume performance.



# Key Insights

Let's proceed with the ad-hoc analysis as per the business requirements and review the SQL queries and their outputs:



★ We are using the MySQL tool to perform the ad-hoc analysis.

★ **Ad-hoc Analysis means** – when a company needs specific information quickly, like “Which city has the highest number of trips?” or “How many new customers joined last month?”, the data analysis done at that moment for that particular purpose is called ad-hoc analysis.



**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

```
SELECT
    c.city_name,
    COUNT(t.trip_id) AS total_trips,
    ROUND(AVG(t.fare_amount / `distance_travelled_km`), 2) AS avg_fare_per_km,
    ROUND(AVG(t.fare_amount), 2) AS avg_fare_per_trip,
    ROUND((COUNT(t.trip_id) * 100.0) / SUM(COUNT(t.trip_id)) OVER (), 2) AS `%_contribution_to_total_trips`
FROM fact_trips t
JOIN dim_city c ON t.city_id = c.city_id
GROUP BY c.city_name;
```

1. Jaipur (76,888 trips) has the highest number of trips. It contributes 18.05% to the total trips.

2. Based on average fare per trip, Jaipur (₹483.92) has the highest average fare per trip. Followed by Kochi (₹335.25) and Visakhapatnam (₹282.67).

3. Surat (₹117.27) and Vadodara (₹118.57) have the lowest average fare per trip.

4. Jaipur (₹16.25/km) is the most expensive in terms of fare per km. Followed by Mysore (₹15.4/km).

5. Mysore (3.81%) and Coimbatore (4.96%) have the lowest contribution.

**6. Top 3 cities by trip contribution:**

- 1. Jaipur – 18.05%
- 2. Lucknow – 15.1%
- 3. Surat – 12.88%

Output

city_name	total_trips	avg_fare_per_km	avg_fare_per_trip	%_contribution_to_total_trips
Chandigarh	38981	12.18	283.69	9.15
Coimbatore	21104	11.3	166.98	4.96
Indore	42456	11.07	179.84	9.97
Jaipur	76888	16.25	483.92	18.05
Kochi	50702	14.13	335.25	11.9
Lucknow	64299	12.14	147.18	15.1
Mysore	16238	15.4	249.71	3.81
Surat	54843	10.92	117.27	12.88
Vadodara	32026	10.54	118.57	7.52
Visakhapatnam	28366	12.7	282.67	6.66



## 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



**Please Check Next Slide**

Note : Sorry, due to the large size of the query output, only the output view is shown here. You can check the full output data on my **GitHub link**. [Click here to view full data on GitHub](#)



```

SELECT
    c.city_name,
    d.month_name,
    COUNT(t.trip_id) AS actual_trips,
    tt.total_target_trips AS target_trips,
    CASE
        WHEN COUNT(t.trip_id) > tt.total_target_trips THEN 'Above Target'
        ELSE 'Below Target'
    END AS performance_status,
    ROUND(((COUNT(t.trip_id) - tt.total_target_trips) / tt.total_target_trips) * 100, 2) AS '%_difference'
FROM trips_db.fact_trips t
JOIN trips_db.dim_city c ON t.city_id = c.city_id
JOIN trips_db.dim_date d ON t.date = d.date
JOIN targets_db.monthly_target_trips tt
    ON t.city_id = tt.city_id
    AND d.start_of_month = tt.month
GROUP BY c.city_name, d.month_name, tt.total_target_trips;

```

### Output

city_name	actual_trips	target_trips	First performance_status	%_difference
Chandigarh	38981	39000	Above Target	0.24
Coimbatore	21104	21000	Above Target	2.97
Indore	42456	43500	Above Target	-13.95
Jaipur				
April	11406	9500	Above Target	20.06
February	15872	13000	Above Target	22.09
January	14976	13000	Above Target	15.20
June	9842	9500	Above Target	3.60
March	13317	13000	Above Target	2.44
May	11475	9500	Above Target	20.79
Kochi	50702	49500	Above Target	17.87
Lucknow	64299	72000	Below Target	-63.21
Mysore	16238	13500	Above Target	127.38
Surat	54843	57000	Above Target	-21.90
Vadodara	32026	37500	Below Target	-87.29
Visakhapatnam	28366	28500	Above Target	-1.26

1. Jaipur consistently performed **above target** in **all months**. It has the **highest positive % differences**, like +22.09% in February and +20.79% in May.

2. Vadodara performed **below target** in **every single month**. It showed very poor performance with -20.42% in January and 27.92% in June.

3. Highest Positive % Difference (Best Month-wise Performance):

- Mysore (February): +33.4%
- Mysore (March): +31.65%
- Even though Mysore has lower targets, it performed extremely well.

4. Highest Negative % Difference (Worst Month-wise Performance):

- Kochi (June): -28.9%
- Vadodara (June): -27.92%
- These cities struggled badly in the month of June.

5. Lucknow is consistently underperforming:

- It is **below target** in **all months**.
- Worst was in January (-16.48%), March (-13.66%), and May (-11.77%).



### **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



**Please Check Next Slide**





```

SELECT
    c.city_name,
    ROUND(SUM(CASE WHEN d.trip_count = '2-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "2-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '3-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "3-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '4-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "4-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '5-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "5-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '6-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "6-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '7-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "7-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '8-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "8-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '9-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "9-Trips",
    ROUND(SUM(CASE WHEN d.trip_count = '10-Trips' THEN d.repeat_passenger_count ELSE 0 END) * 100.0
        / SUM(d.repeat_passenger_count), 2) AS "10-Trips"
FROM trips_db.dim_repeat_trip_distribution d
JOIN trips_db.dim_city c
    ON d.city_id = c.city_id
GROUP BY c.city_name;

```

Output

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	13.4	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	1.9	1.2	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	16.2	18.42	20.18	11.33	6.43	1.91	1.1

1. These cities have a **majority of their repeat passengers doing only 2 or 3 trips.**

- **Visakhapatnam** → 51.25% (2-Trips), 24.96% (3-Trips) = **76.2%** within just 2-3 trips.
- **Mysore** → 48.75% + 24.44% = **73.2%**
- **Kochi** → 47.67% + 24.35% = **72%**
- **Jaipur** → 50.14% + 20.73% = **70.87%**

2. Cities where a significant % of users fall in the **4 to 7 trips range**, showing **consistent engagement**:

- **Surat** → Strong in 4 to 6 trips: 16.55% + 19.75% + 18.45% = **54.75%**
- **Vadodara** → 16.52% + 18.06% + 19.08% = **53.66%**
- **Coimbatore** → 15.56% + 20.62% + 17.64% = **53.82%**
- **Lucknow** → 16.2% + 18.42% + 20.18% = **54.8%**

3. Cities where more passengers are making **8 to 10 trips**:

- **Chandigarh** → 2.33% (9-Trips), 1.79% (10-Trips)
- **Indore** → 2.38% (9-Trips)
- **Vadodara** → 2.05% (9-Trips), 1.61% (10-Trips)

Though these percentages are small, they highlight **super-loyal users** in these cities.

4. **Indore** and **Chandigarh** show a **gradual drop-off** from 2 to 10 trips, indicating a **healthy distribution** of repeat passengers — both casual and loyal.





**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")

Output

city_name	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

```
WITH CityPassengerCounts AS (  
    SELECT  
        c.city_name,  
        SUM(f.new_passengers) AS total_new_passengers  
    FROM trips_db.fact_passenger_summary f  
    JOIN trips_db.dim_city c  
        ON f.city_id = c.city_id  
    GROUP BY c.city_name  
),  
RankedCities AS (  
    SELECT  
        city_name,  
        total_new_passengers,  
        RANK() OVER (ORDER BY total_new_passengers DESC) AS rank_high,  
        RANK() OVER (ORDER BY total_new_passengers ASC) AS rank_low  
    FROM CityPassengerCounts  
)  
SELECT  
    city_name,  
    total_new_passengers,  
    CASE  
        WHEN rank_high <= 3 THEN 'Top 3'  
        WHEN rank_low <= 3 THEN 'Bottom 3'  
        ELSE NULL  
    END AS city_category  
FROM RankedCities  
WHERE rank_high <= 3 OR rank_low <= 3  
ORDER BY total_new_passengers DESC;
```



```

WITH CityRevenue AS (
    SELECT
        c.city_name,
        d.month_name,
        SUM(f.fare_amount) AS revenue
    FROM trips_db.fact_trips f
    JOIN trips_db.dim_city c
        ON f.city_id = c.city_id
    JOIN trips_db.dim_date d
        ON f.date = d.date
    GROUP BY c.city_name, d.month_name
),
RankedRevenue AS (
    SELECT
        city_name,
        month_name AS highest_revenue_month,
        revenue,
        RANK() OVER (PARTITION BY city_name ORDER BY revenue DESC) AS revenue_rank
    FROM CityRevenue
),
TotalCityRevenue AS (
    SELECT
        r.city_name,
        r.highest_revenue_month,
        r.revenue,
        ROUND((r.revenue / t.total_revenue) * 100, 2) AS percentage_contribution
    FROM RankedRevenue r
    JOIN TotalCityRevenue t
        ON r.city_name = t.city_name
    WHERE revenue_rank = 1
    ORDER BY r.city_name;

```

### 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 (%)

#### Output

city_name	highest_revenue_month	revenue	percentage_contribution
Chandigarh	February	2108290	19.07
Coimbatore	April	612431	17.38
Indore	May	1380996	18.09
Jaipur	February	7747202	20.82
Kochi	May	3333746	19.61
Lucknow	February	1777269	18.78
Mysore	May	745170	18.38
Surat	April	1154909	17.96
Vadodara	April	706250	18.6
Visakhapatnam	April	1390682	17.34



## 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



```
SELECT
  c.city_name,
  d.month_name AS month,
  f.total_passengers,
  f.repeat_passengers,
  ROUND((f.repeat_passengers / f.total_passengers) * 100, 2) AS monthly_repeat_passenger_rate,
  ROUND(
    SUM(f.repeat_passengers) OVER (PARTITION BY c.city_name) /
    SUM(f.total_passengers) OVER (PARTITION BY c.city_name) * 100, 2
  ) AS city_repeat_passenger_rate
FROM trips_db.fact_passenger_summary f
JOIN trips_db.dim_city c ON f.city_id = c.city_id
JOIN trips_db.dim_date d ON f.month = d.start_of_month;
```

Note : Sorry, due to the large size of the output data for this query, I have only shown the SQL query here. You can view the complete output data by visiting my **GitHub** link.

[Click here to view full data on GitHub](#)



# Suggestions

**1. Give offers to increase trip frequency (Mysore, Kochi, Jaipur) -**

Offer free rides or discounts to users with 2 - 4 trips to encourage more rides.

**2. Push mid-level users forward (Surat, Vadodara, Coimbatore) -**

Give rewards on every 7th ride to users who usually take 4–6 trips.

**3. Use reminders for balanced cities (Chandigarh, Indore) -**

Send reminders and offers to motivate users for more trips.

**4. Turn loyal users into VIPs -**

Offer special rewards or VIP membership to users with 8+ trips.

**5. Re-target users who stop after 2 trips -**

Bring back 2-trip users with SMS and exclusive offers.

**6. City-wise Strategy Planning -**

- User behavior varies from city to city, so we should create separate marketing strategies for each city.
- When the whole team discusses the data together, many great ideas come up, some of which can be tried to drive growth.





**Thank you so much for  
your time and attention.**

---

