Contents

Introduction

GoodCabs, a cab service company established two years ago, has quickly emerged as a prominent player in India's tier-2 city transportation market. The company's unique approach of empowering local drivers while delivering excellent service has distinguished it from competitors. Operating in ten tier-2 cities across the country.

As part of its growth strategy, GoodCabs has set ambitious performance targets for 2024. These targets focus on key metrics such as trip volume, passenger satisfaction, repeat passenger rates, trip distribution and the balance between new and repeat passengers. To achieve these goals, the company requires a robust data management and analysis system that can provide valuable insights and support informed decision-making.

This project aims to design and implement a comprehensive Database Management System (DBMS) to serve as a central repository for GoodCabs' operational data. The DBMS meticulously stores and tracks diverse data points, enabling stakeholders to make data-driven decisions. It supports key functions such as assessing performance metrics, understanding passenger behavior, and optimizing service quality.

By leveraging stored procedures, Common Table Expressions (CTEs) and custom functions, this system provides the tools to solve critical business problems. From analyzing trip volumes and patterns to evaluating customer satisfaction trends, the DBMS transforms raw data into actionable insights. This initiative not only empowers the GoodCabs management team but also ensures sustainable growth by enhancing operational efficiency and aligning strategies with passenger expectations.

Through this GoodCabs aims to solidify its position in the market, achieve its performance targets for 2024 and continue its mission of delivering excellent service.

Tables

This database consists for 8 tables

fact_trips This table contains information about every ride done by Good Cabs including trip_id, date, ratings, fare amount, trip distance etc.

It has foreign key constraints to link other tables such as date and dim_city to maintain referential integrity. The DDL command mentioned below was used to create the table in database.

```
CREATE TABLE fact_trips  (
trip_id varchar(50) ,
date date,
city_id varchar(5),
passenger_type varchar(10),
distance_travelled int,
fare_amount int,
passenger_rating int,
driver_rating int,
primary key(trip_id),
CONSTRAINT fk_city FOREIGN KEY (city_id) REFERENCES
dim_city(city_id),
CONSTRAINT fk_date FOREIGN KEY (date) REFERENCES dim_date(date) )
ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
+----------------------+-------------+------+-----+---------+-------+
| Field                | Type        | Null | Key | Default | Extra |
+----------------------+-------------+------+-----+---------+-------+
| trip_id              | varchar(50) | NO   | PRI | NULL    |       |
| date                 | date        | YES  | MUL | NULL    |       |
| city_id              | varchar(5)  | YES  | MUL | NULL    |       |
| passenger_type       | varchar(10) | YES  |     | NULL    |       |
| distance_travelled_km | int         | YES  |     | NULL    |       |
| fare_amount          | int         | YES  |     | NULL    |       |
| passenger_rating     | int         | YES  |     | NULL    |       |
| driver_rating        | int         | YES  |     | NULL    |       |
+----------------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

dim_repeat_trip_distribution 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. This allows for an analysis of repeat trip patterns at a granular level.

```
1 CREATE TABLE dim_repeat_trip_distribution (
2 month date,
3 city_id varchar(5),
4 trip_count varchar(10),
5 repeat_passenger_count int,
6 primary key(month),
7 primary key(city_id),
8 primary key(trip_count) )
9 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc dim_repeat_trip_distribution;
+------------------------+-------------+------+-----+---------+-------+
| Field                  | Type        | Null | Key | Default | Extra |
+------------------------+-------------+------+-----+---------+-------+
| month                  | date        | NO   | PRI | NULL    |       |
| city_id                | varchar(5)  | NO   | PRI | NULL    |       |
| trip_count             | varchar(10) | NO   | PRI | NULL    |       |
| repeat_passenger_count | int         | YES  |     | NULL    |       |
+------------------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

fact_passenger_summary 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.

```
1 CREATE TABLE fact_passenger_summary (
2 month date,
3 city_id varchar(5),
4 total_passengers int,
5 new_passengers int,
6 repeat_passengers int,
7 primary key(month),
8 primary key(city_id) )
9 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc fact_passenger_summary;
+-------------------+------------+------+-----+---------+-------+
| Field             | Type       | Null | Key | Default | Extra |
+-------------------+------------+------+-----+---------+-------+
| month             | date       | NO   | PRI | NULL    |       |
| city_id           | varchar(5) | NO   | PRI | NULL    |       |
| total_passengers  | int        | YES  |     | NULL    |       |
| new_passengers    | int        | YES  |     | NULL    |       |
| repeat_passengers | int        | YES  |     | NULL    |       |
+-------------------+------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

dim_date This table provides date-specific details that help to identify patterns across days, months and weekends versus weekdays.

```
1 CREATE TABLE dim_date (
2 date date,
3 start_of_month date,
4 month_name varchar(20),
5 day_type varchar(10),
6 primary key(date) )
7 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc dim_date;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| date           | date        | NO   | PRI | NULL    |       |
| start_of_month | date        | YES  |     | NULL    |       |
| month_name     | varchar(20) | YES  |     | NULL    |       |
| day_type       | varchar(10) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

dim_city This table provides city-specific details, enabling location-based analysis of trips and passenger behavior across Goodcabs' operational areas.

```
1 CREATE TABLE dim_city (
2 city_id varchar(5),
3 city_name varchar(20),
4 primary key (city_name) )
5 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc dim_city;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| city_id   | varchar(5)  | NO   | PRI | NULL    |       |
| city_name | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

city_target_passenger_rate

```
1 CREATE TABLE city_target_passenger_rating (
2 city_id varchar(5),
3 target_avg_passenger_rating decimal(3,2),
4 primary key(city_id) )
5 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc city_target_passenger_rating;
+----------------------------+-------------+------+-----+---------+-------+
| Field                      | Type        | Null | Key | Default | Extra |
+----------------------------+-------------+------+-----+---------+-------+
| city_id                    | varchar(5)  | NO   | PRI | NULL    |       |
| target_avg_passenger_rating | decimal(3,2) | YES  |     | NULL    |       |
+----------------------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

monthly_target_new_passengers

```
1 CREATE TABLE monthly_target_new_passengers (
2 month date,
3 city_id varchar(5),
4 target_new_passenger int,
5 primary key(city_id),
6 primary key(month) )
7 ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
mysql> desc monthly_target_new_passengers;
+----------------------+------------+------+-----+---------+-------+
| Field                | Type       | Null | Key | Default | Extra |
+----------------------+------------+------+-----+---------+-------+
| month                | date       | NO   | PRI | NULL    |       |
| city_id              | varchar(5) | NO   | PRI | NULL    |       |
| target_new_passengers | int        | YES  |     | NULL    |       |
+----------------------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

monthly_target_trips

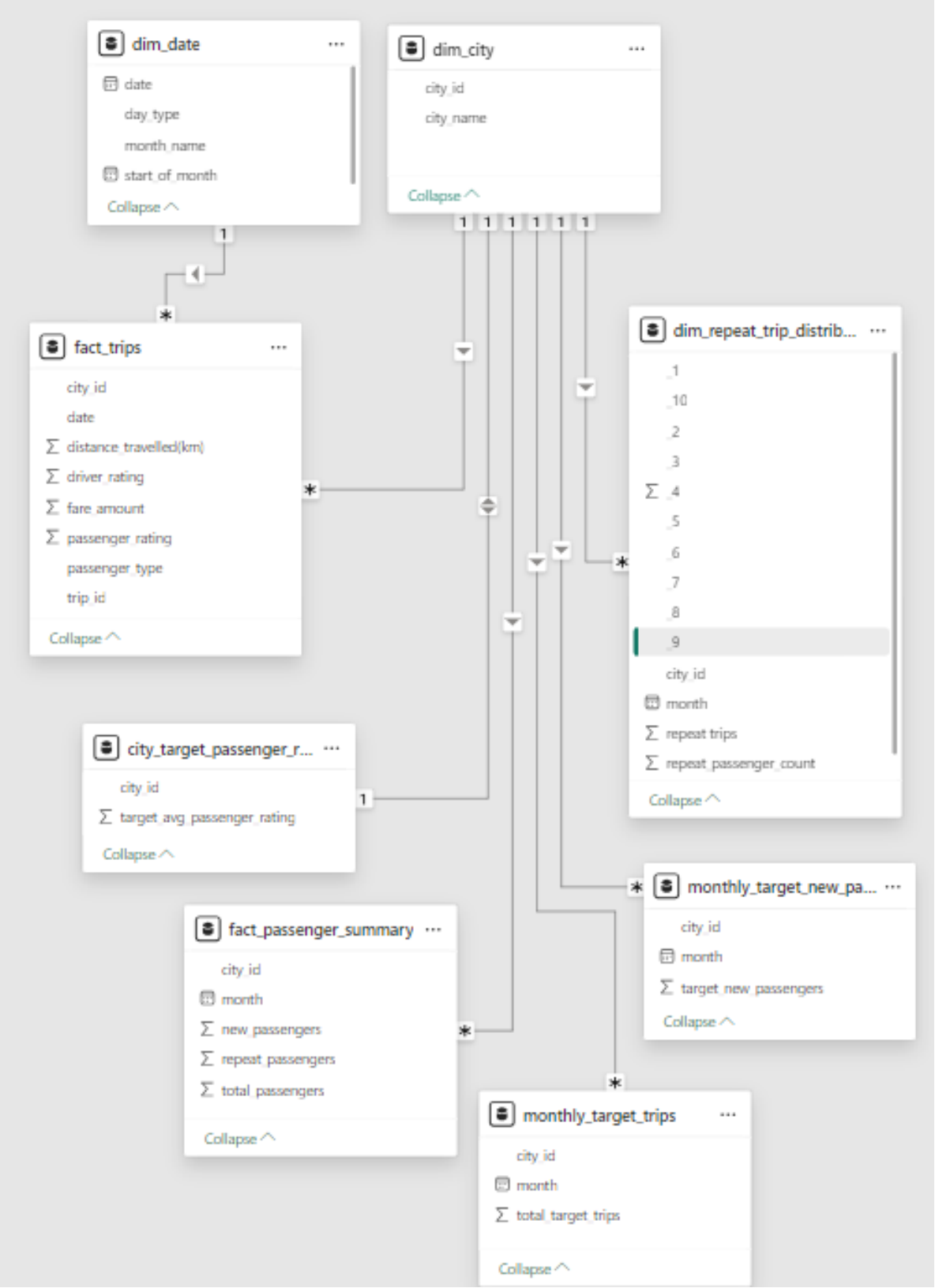
```
1 CREATE TABLE monthly_target_trips (
2 month date,
3 city_id varchar(5),
4 target_target_trips int,
5 primary key(city_id),
6 primary key(month) )
7 ENGINE = InnoDB xDEFAULT CHARSET = latin1;
```

```
mysql> desc monthly_target_trips;
+--------------------+------------+------+-----+---------+-------+
| Field              | Type       | Null | Key | Default | Extra |
+--------------------+------------+------+-----+---------+-------+
| month              | date       | NO   | PRI | NULL    |       |
| city_id            | varchar(5) | NO   | PRI | NULL    |       |
| total_target_trips | int        | YES  |     | NULL    |       |
+--------------------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Record Details

Table Name	No. of Records
fact_trips	425903
dim_repeat_trip_distribution	540
fact_passenger_summary	60
dim_date	182
dim_city	10
city_target_passenger_rating	10
monthly_target_new_passenger	60
monthly_target_trips	60

Table Model Diagram



Flowchart

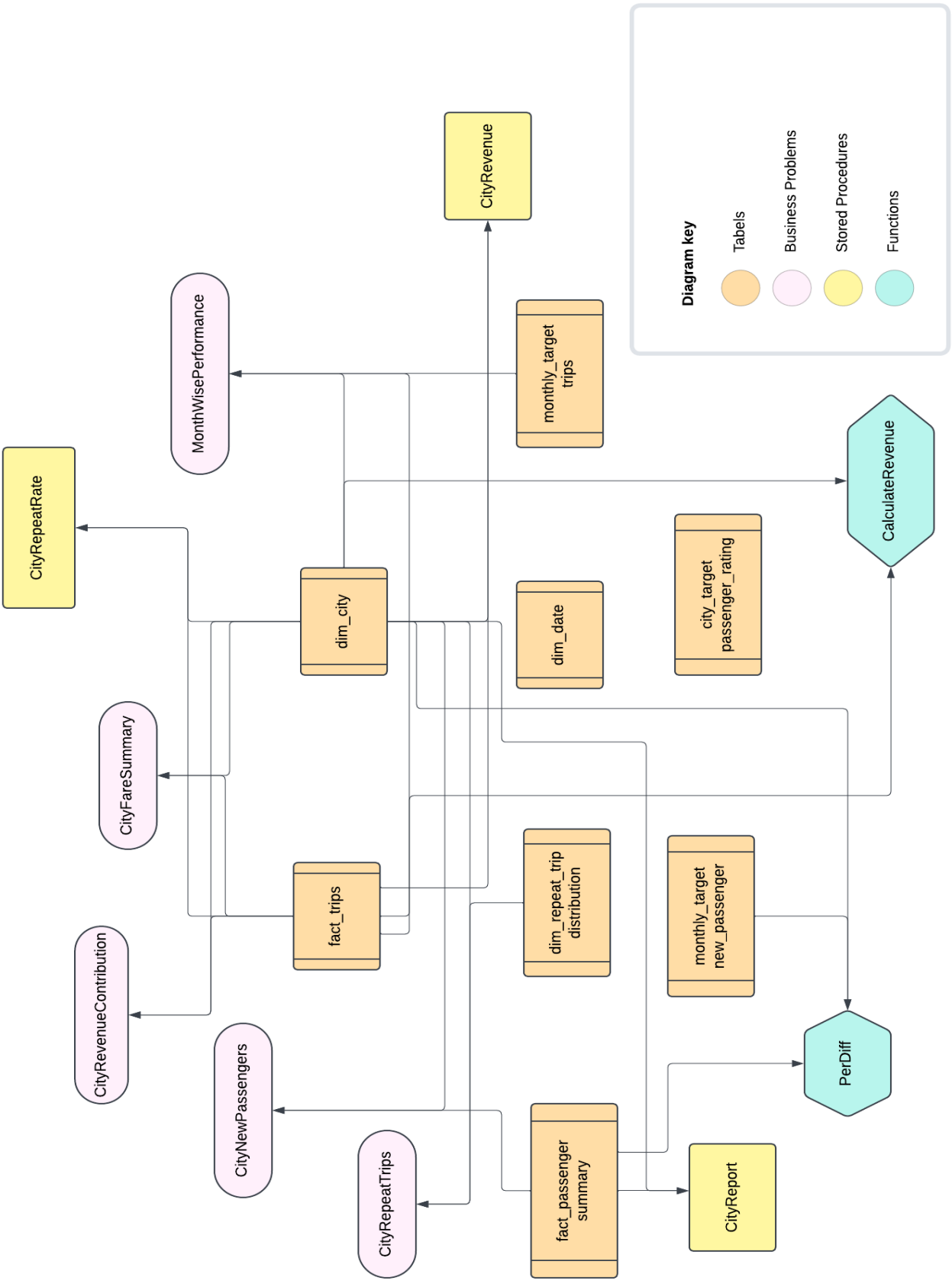


Diagram key
Tabels
Business Problems
Stored Procedures
Functions
CityRevenue
MonthWisePerformance
monthly_target trips
CityRepeatRate
CalculateRevenue
dim_city
dim_date
city_target passenger_rating
CityFareSummary
fact_trips
dim_repeat_trip distribution
monthly_target new_passenger
CityRevenueContribution
PerDiff
CityNewPassengers
fact_passenger summary
CityReport
CityRepeatTrips

Business Questions

Q: 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.

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

```
 0  CREATE VIEW CityFareSummary AS
 1  SELECT
 2      c.*,
        (c.total_trips / (SELECT COUNT(trip_id) FROM fact_trips) * 100) AS
 3  percentage_contribution
 4  FROM (
 5      SELECT
 6          b.city_name,
 7          COUNT(a.trip_id) AS total_trips,
 8          AVG(a.fare_amount / a.distance_travelled_km) AS avg_fare_per_km,
 9          SUM(a.fare_amount) / COUNT(a.trip_id) AS avg_fare_per_trip
10      FROM
11          fact_trips AS a
12      INNER JOIN
13          dim_city AS b
14          ON a.city_id = b.city_id
15      GROUP BY
16          b.city_name
17  ) AS c
18  GROUP BY
19      c.city_name;
```

city_name	total_trips	avg_fare_per_km	avg_fare_per_trip	percentage_contribution
Visakhapatnam	28366	12.70375607	282.6723	6.6602
Chandigarh	38981	12.17599227	283.6870	9.1526
Surat	54843	10.91605974	117.2729	12.8769
Vadodara	32026	10.54417964	118.5662	7.5196
Mysore	16238	15.39967755	249.7072	3.8126
Kochi	50702	14.13468563	335.2451	11.9046
Indore	42456	11.06559860	179.8386	9.9685
Jaipur	76888	16.25218302	483.9181	18.0529
Coimbatore	21104	11.30485444	166.9822	4.9551
Lucknow	64299	12.14309269	147.1804	15.0971

10 rows in set (2.62 sec)

Q: 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 categorize the performance as follows:

--If actual trips are greater than target trips, mark it as "Above Target" else mark it as "Below Target".

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

A:

```
 0  CREATE VIEW MonthWisePerformance AS
 1  select
 2  a.city_name,
 3      b.month_name,
 4      b.actual_trips,
 5      c.total_target_trips,
        case when (b.actual_trips - c.total_target_trips) < 0 then "Below
 6  Target" else "Above Target" end as performance_status,
        (b.actual_trips - c.total_target_trips)/c.total_target_trips*100 as
 7  percentage_difference
 8  from
 9      (select
10          city_id,
11          monthname(date) as month_name,
12          month(date) as month_,
13          count(trip_id) as actual_trips
14          from fact_trips
15      group by
16         city_id,
17          monthname(date),
18          month(date))  as b
19  inner join dim_city as a
20  on a.city_id = b.city_id
    inner join (select *, monthname(month) as month_name from
21  targets_db.monthly_target_trips) as c
22  on c.city_id = b.city_id and c.month_name = b.month_name
23  order by a.city_name, b.month_
24  ;
```

city_name	month_name	actual_trips	total_target_trips	performance_status	percentage_difference
Chandigarh	January	6819	7000	Below Target	-2.7143
Chandigarh	February	7387	7000	Above Target	5.5286
Chandigarh	March	6549	7000	Below Target	-6.4571
Chandigarh	April	5566	6000	Below Target	-7.2333
Chandigarh	May	6629	6000	Above Target	10.4833
Chandigarh	June	6029	6000	Above Target	0.4833
Coimbatore	January	3651	3500	Above Target	4.3143
Coimbatore	February	3484	3500	Below Target	-0.4571
Coimbatore	March	3681	3500	Above Target	5.1714
Coimbatore	April	3664	3500	Above Target	4.6857
Coimbatore	May	3159	3500	Below Target	-9.7714
Coimbatore	June	4737	7000	Below Target	-32.3571
Indore	January	7249	7000	Above Target	3.5571
Indore	February	7819	7000	Above Target	11.7000
Indore	March	7819	7000	Above Target	11.7000
Indore	April	7481	7500	Below Target	-0.2533
Indore	May	7787	7500	Above Target	3.8267
Indore	June	7500	7500	Below Target	-15.1600
Jaipur	January	14976	13000	Above Target	15.2000
Jaipur	February	15975	13000	Above Target	22.8923
Jaipur	March	13317	13000	Above Target	2.4385
Jaipur	April	11486	9500	Above Target	20.8632
Jaipur	May	11475	9500	Above Target	20.7895
Jaipur	June	9842	9500	Above Target	3.6000
Kochi	January	7344	7500	Below Target	-2.0800
Kochi	February	7688	7500	Above Target	2.5067
Kochi	March	8495	7500	Above Target	26.6000
Kochi	April	9762	9000	Above Target	8.4667
Kochi	May	10014	9000	Above Target	11.2667
Kochi	June	6399	9000	Below Target	-28.9000
Lucknow	January	10858	13000	Below Target	-16.4769
Lucknow	February	12069	13000	Below Target	-7.2385
Lucknow	March	11334	13000	Below Target	-13.6615
Lucknow	April	10813	11000	Below Target	-7.1636
Lucknow	May	9785	11000	Below Target	-11.0727
Lucknow	June	10248	11000	Below Target	-6.8909
Mysore	January	2485	2000	Above Target	24.2500
Mysore	February	2668	2000	Above Target	33.4000
Mysore	March	2633	2000	Above Target	31.6500
Mysore	April	2603	2500	Above Target	4.1200
Mysore	May	3087	2500	Above Target	23.4800
Mysore	June	2843	2500	Above Target	13.7200
Surat	January	8358	9000	Below Target	-7.1333
Surat	February	9869	9000	Above Target	9.6556
Surat	March	9267	9000	Above Target	2.9667
Surat	April	9831	10000	Below Target	-1.6900
Surat	May	9734	10000	Below Target	-2.6600
Surat	June	8594	10000	Below Target	-14.0600
Vadodara	January	4775	6000	Below Target	-20.4167
Vadodara	February	5228	6000	Below Target	-12.8667
Vadodara	March	5598	6000	Below Target	-6.7000
Vadodara	April	5941	6500	Below Target	-8.6000
Vadodara	May	5799	6500	Below Target	-10.7846
Vadodara	June	4685	6500	Below Target	-27.9231
Visakhapatnam	January	4468	4500	Below Target	-0.7111
Visakhapatnam	February	4793	4500	Above Target	6.5111
Visakhapatnam	March	4877	4500	Above Target	8.3778
Visakhapatnam	April	4933	5000	Below Target	-1.3400
Visakhapatnam	May	4812	5000	Below Target	-3.7600
Visakhapatnam	June	4478	5000	Below Target	-10.4400

60 rows in set (0.94 sec)

Q: 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.

A: This table allow us the study the passenger repeat ride trends for a city & provide us actionable insights to take decisions regarding marketing, passenger experience, driver training etc.

```
 0 CREATE VIEW CityRepeatTrips AS
 1 select city_name,
       sum(case when trip_count = "2-Trips" then percentage else 0 end) as
 2 "2-Trips",
       sum(case when trip_count = "3-Trips" then percentage else 0 end) as
 3 "3-Trips",
       sum(case when trip_count = "4-Trips" then percentage else 0 end) as
 4 "4-Trips",
       sum(case when trip_count = "5-Trips" then percentage else 0 end) as
 5 "5-Trips",
       sum(case when trip_count = "6-Trips" then percentage else 0 end) as
 6 "6-Trips",
       sum(case when trip_count = "7-Trips" then percentage else 0 end) as
 7 "7-Trips",
       sum(case when trip_count = "8-Trips" then percentage else 0 end) as
 8 "8-Trips",
       sum(case when trip_count = "9-Trips" then percentage else 0 end) as
 9 "9-Trips",
       sum(case when trip_count = "10-Trips" then percentage else 0 end) as
10 "10-Trips"
11 from
       (select city_name, trip_count, trip_sum/city_sum*100 as percentage
12 from
       (select b.city_name, a.trip_count, sum(a.repeat_passenger_count) as
13 trip_sum,
       sum(sum(a.repeat_passenger_count)) over(partition by b.city_name rows
14 between unbounded preceding and unbounded following) as city_sum
15     from dim_repeat_trip_distribution a
16     inner join dim_city b
17     on a.city_id  = b.city_id
18     group by b.city_name, a.trip_count) c ) d
19 group by city_name;
```

city_name	2-Trips	3-Trips	4-Trips	5-Trips	6-Trips	7-Trips	8-Trips	9-Trips	10-Trips
Chandigarh	32.3077	19.2505	15.7396	12.2091	7.4162	5.4832	3.4714	2.3274	1.7949
Coimbatore	11.2113	14.8177	15.5625	20.6194	17.6401	10.4665	6.1544	2.3128	1.2152
Indore	34.3404	22.6857	13.4008	10.3381	6.8459	5.2384	3.2567	2.3836	1.5105
Jaipur	50.1446	20.7292	12.1153	6.2900	4.1314	2.5201	1.9004	1.1981	0.9709
Kochi	47.6659	24.3509	11.8148	6.4778	3.9077	2.1112	1.6522	1.2064	0.8130
Lucknow	9.6593	14.7650	16.2030	18.4224	20.1834	11.3265	6.4291	1.9068	1.1045
Mysore	48.7475	24.4414	12.7285	5.8226	4.0623	1.7603	1.4218	0.5416	0.4739
Surat	9.7592	14.2626	16.5548	19.7499	18.4533	11.8893	6.2399	1.7365	1.3545
Vadodara	9.8711	14.1740	16.5209	18.0626	19.0750	12.8624	5.7754	2.0479	1.6107
Visakhapatnam	51.2529	24.9608	9.9843	5.4424	3.1911	1.9773	1.3900	0.8810	0.9201

10 rows in set (0.01 sec)

Q: 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.

A: This table provide details of the total new passengers obtained that used our services for the given city and categorise them as “Top 3” or “Bottom 3”, allowing us to identify the most under performing cities.

```
0  CREATE VIEW CityNewPassengers AS
1  select * from
2      (select city_name, total_new_passengers,
3            case when rnk <= 3 then "Top 3"
4            when rnk >=8 then "Bottom 3"
             else "Others" end as city_category
5       from
        (select b.city_name, sum(a.new_passengers) as total_new_passengers,
6  rank() over(order by sum(a.new_passengers) desc) as rnk
7  from fact_passenger_summary a
8  inner join  dim_city b
9  on a.city_id = b.city_id
10 group by b.city_name) c) d;
```

city_name	total_new_passengers	city_category
Jaipur	45856	Top 3
Kochi	26416	Top 3
Chandigarh	18908	Top 3
Lucknow	16260	Others
Indore	14863	Others
Visakhapatnam	12747	Others
Mysore	11681	Others
Surat	11626	Bottom 3
Vadodara	10127	Bottom 3
Coimbatore	8514	Bottom 3

Q: 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.

A: This table highlights the month with the highest revenue for each city. Analyzing this data can help us understand the factors contributing to these results and identify opportunities for improvement. By focusing our marketing efforts during these high-performing periods in the following year or similar occasions, we can strategically boost our overall revenue.

```
0  CREATE VIEW CityRevenueContribution AS
1  select city_name, month_name as highest_revenue_month, revenue as
   Revenue_INR, (revenue/total_revenue)*100 as percentage_contirbution from
2      (select *, rank() over(partition by city_name order by revenue desc)
   rnk,
3          sum(revenue) over(partition by city_name rows between unbounded
   preceding and unbounded following) total_revenue
4      from
5          (select b.city_name,
6          monthname(a.date) month_name,
7          sum(fare_amount) as revenue
8         from fact_trips a
9          inner join dim_city b
10             on a.city_id = b.city_id
11         group by b.city_name, month_name) c) d
12 where rnk = 1 ;
```

city_name	highest_revenue_month	Revenue_INR	percentage_contirbution
Chandigarh	February	2108290	19.0651
Coimbatore	April	612431	17.3789
Indore	May	1380996	18.0872
Jaipur	February	7747202	20.8216
Kochi	May	3333746	19.6130
Lucknow	February	1777269	18.7801
Mysore	May	745170	18.3777
Surat	April	1154909	17.9568
Vadodara	April	706250	18.5992
Visakhapatnam	April	1390682	17.3439

10 rows in set (2.76 sec)

Stored Procedures

Q: Generate a report for given city with two metrics:

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

-- City-wide Repeat Passenger Rate: Calculate the overall repeat passenger rate considering all passengers across months.

A: This Procedure returns the overall, performance report of city.

```
DELIMITER //
CREATE PROCEDURE city_report(IN city varchar(20))
BEGIN
    select city_name, month, total_passengers, repeat_passengers,
           monthly_repeat_passenger_rate,
          (sum_repeat_passengers/sum_total_passengers)*100 as
          city_wide_repeat_rate
           from
              (select b.city_name, monthname(a.month) as month,
              month(a.month) as month_num, a.total_passengers,
              a.repeat_passengers,
                  (a.repeat_passengers/a.total_passengers)*100 as
                  monthly_repeat_passenger_rate,
                  sum(a.total_passengers) over(partition by b.city_name
                  rows between unbounded preceding and unbounded following)
                  as sum_total_passengers,
                  sum(a.repeat_passengers) over(partition by b.city_name
                  rows between unbounded preceding and unbounded following)
                  as sum_repeat_passengers
              from fact_passenger_summary as a
              inner join dim_city b
                  on a.city_id=b.city_id) c
           where city_name = city
           order by city_name, month_num;
END//
```

```
mysql> call city_report("Jaipur");
+-----------+----------+-----------------+-----------------+-------------------------------+----------------------+
| city_name | month    | total_passengers | repeat_passengers | monthly_repeat_passenger_rate | city_wide_repeat_rate |
+-----------+----------+-----------------+-----------------+-------------------------------+----------------------+
| Jaipur    | January  |           11845 |            1422 |                       12.0051 |              17.4331 |
| Jaipur    | February |           12450 |            1661 |                       13.3414 |              17.4331 |
| Jaipur    | March    |            9257 |            1840 |                       19.8768 |              17.4331 |
| Jaipur    | April    |            7856 |            1736 |                       22.0978 |              17.4331 |
| Jaipur    | May      |            7174 |            1842 |                       25.6761 |              17.4331 |
| Jaipur    | June     |            6956 |            1181 |                       16.9781 |              17.4331 |
+-----------+----------+-----------------+-----------------+-------------------------------+----------------------+
6 rows in set (0.01 sec)
```

Q: Write a Stored Procedure to present the relation between repeat rate and passenger rating for the give name entered by user.

A:

```
DELIMITER
//
CREATE PROCEDURE city_repeat_rate(IN city varchar(20))
BEGIN
select d.city_name, d.month, d.repeat_rate, e.avg_passenger_rating,
e.avg_driver_rating
from
       (select city_name, month, repeat_rate, month_num
       from
          (select b.city_name, monthname(a.month) as month,
          month(a.month) as month_num,
              (a.repeat_passengers/a.total_passengers)*100 as repeat_rate
          from fact_passenger_summary as a
          inner join dim_city b
              on a.city_id=b.city_id) c ) d
inner
join
       (select b.city_name, monthname(a.date) month,
       avg(a.passenger_rating) avg_passenger_rating, avg(a.driver_rating)
       avg_driver_rating
       from fact_trips a
       inner join dim_city b
          on a.city_id = b.city_id
       group by b.city_name, monthname(a.date) ) e
on d.city_name = e.city_name and d.month = e.month
where d.city_name = city
order by city_name, d.month_num
;
END//
```

```
mysql> call city_repeat_rate("Lucknow");
+-----------+----------+-------------+----------------------+-------------------+
| city_name | month    | repeat_rate | avg_passenger_rating | avg_driver_rating |
+-----------+----------+-------------+----------------------+-------------------+
| Lucknow   | January  |     29.2279 |               6.6207 |            6.6551 |
| Lucknow   | February |     31.9776 |               6.5739 |            6.6316 |
| Lucknow   | March    |     33.9260 |               6.5441 |            6.6390 |
| Lucknow   | April    |     39.2960 |               6.4545 |            6.6140 |
| Lucknow   | May      |     47.6627 |               6.3589 |            6.5943 |
| Lucknow   | June     |     46.7009 |               6.3492 |            6.5647 |
+-----------+----------+-------------+----------------------+-------------------+
6 rows in set (2.71 sec)
```

Q: Write a Procedure to present month & the corresponding revenue for the given city.

A:

```
DELIMITER //
CREATE  PROCEDURE city_revenue (IN city varchar(20))
BEGIN
select
       month(a.date) rnk,
       monthname(a.date) month_name,
       sum(fare_amount) as revenue
       from fact_trips a
inner join dim_city b
       on a.city_id = b.city_id
where city_name = city
group by month_name, rnk
order by city_name, rnk;
END//
```

```
mysql> call city_revenue("Jaipur");
+------+--------------+----------+
| rnk  | month_name   | revenue  |
+------+--------------+----------+
|    1 | January      | 7223310  |
|    2 | February     | 7747202  |
|    3 | March        | 6462092  |
|    4 | April        | 5490146  |
|    5 | May          | 5495976  |
|    6 | June         | 4788771  |
+------+--------------+----------+
6 rows in set (0.49 sec)
```

Database Functions

Q: Write a database function to calculate the percentage difference between the new passengers and the target new passengers for a given city. This will help track the overall performance of the city in terms of marketing and advertisements.

A:

```
DELIMITER //
CREATE FUNCTION per_diff(city VARCHAR(20))
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE avg_difference DECIMAL(10, 2);

    SELECT AVG((a.new_passengers - c.target_new_passengers) / c.target_new_passengers * 100)
    INTO avg_difference
    FROM trips_db.fact_passenger_summary AS a
    INNER JOIN trips_db.dim_city AS b
        ON a.city_id = b.city_id
    INNER JOIN targets_db.monthly_target_new_passengers AS c
        ON c.month = a.month AND c.city_id = a.city_id
    WHERE b.city_name = city
      AND c.target_new_passengers > 0; -- Avoid division by zero

    RETURN avg_difference;
END //
DELIMITER ;
```

```
mysql> SELECT per_diff('Lucknow') AS avg_difference;
+----------------+
| avg_difference |
+----------------+
|           3.77 |
+----------------+
1 row in set (0.01 sec)
```

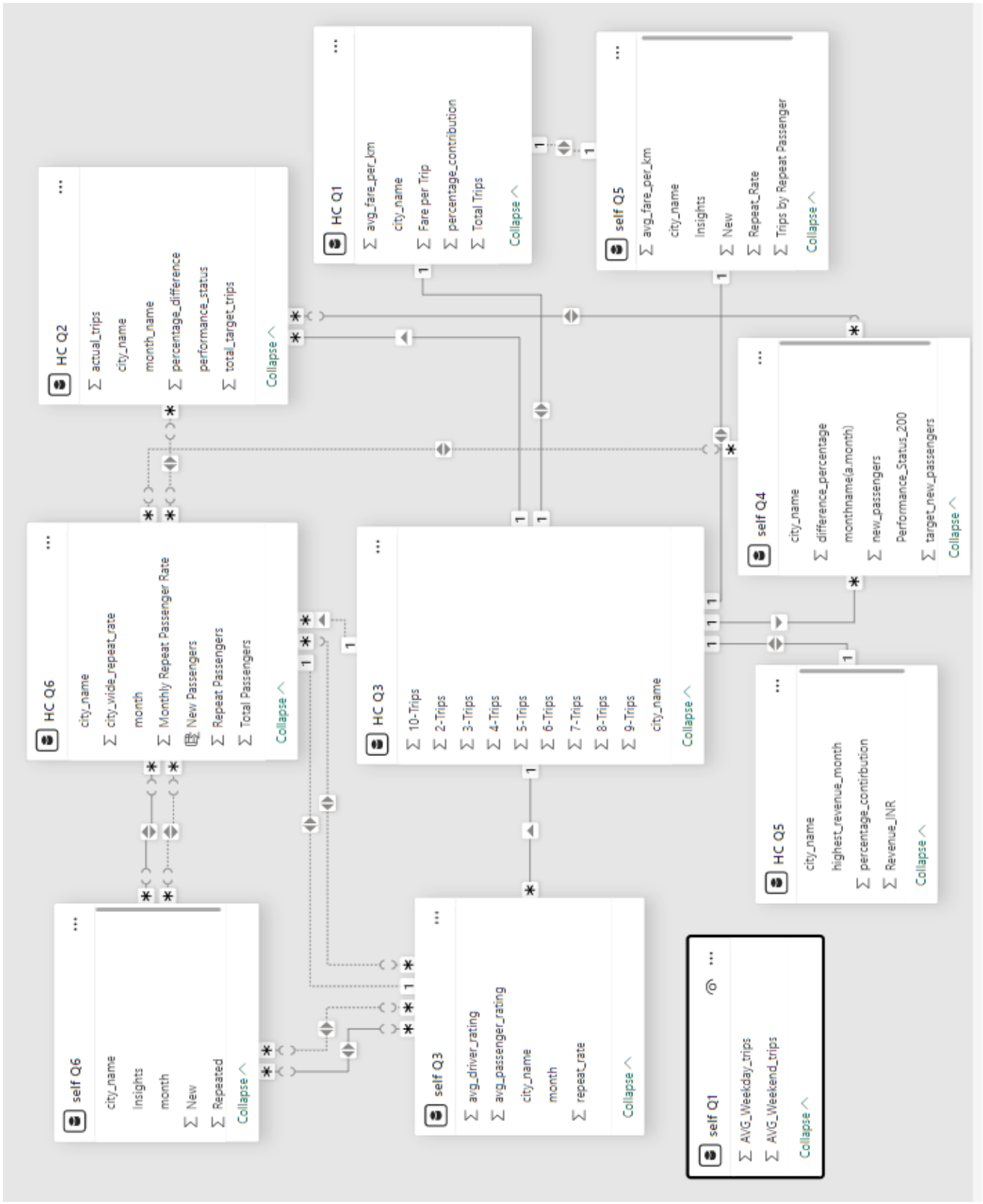
Q: Write a function to calculate total revenue generated by given city and total revenue of Good Cabs and percentage contribution.

A:

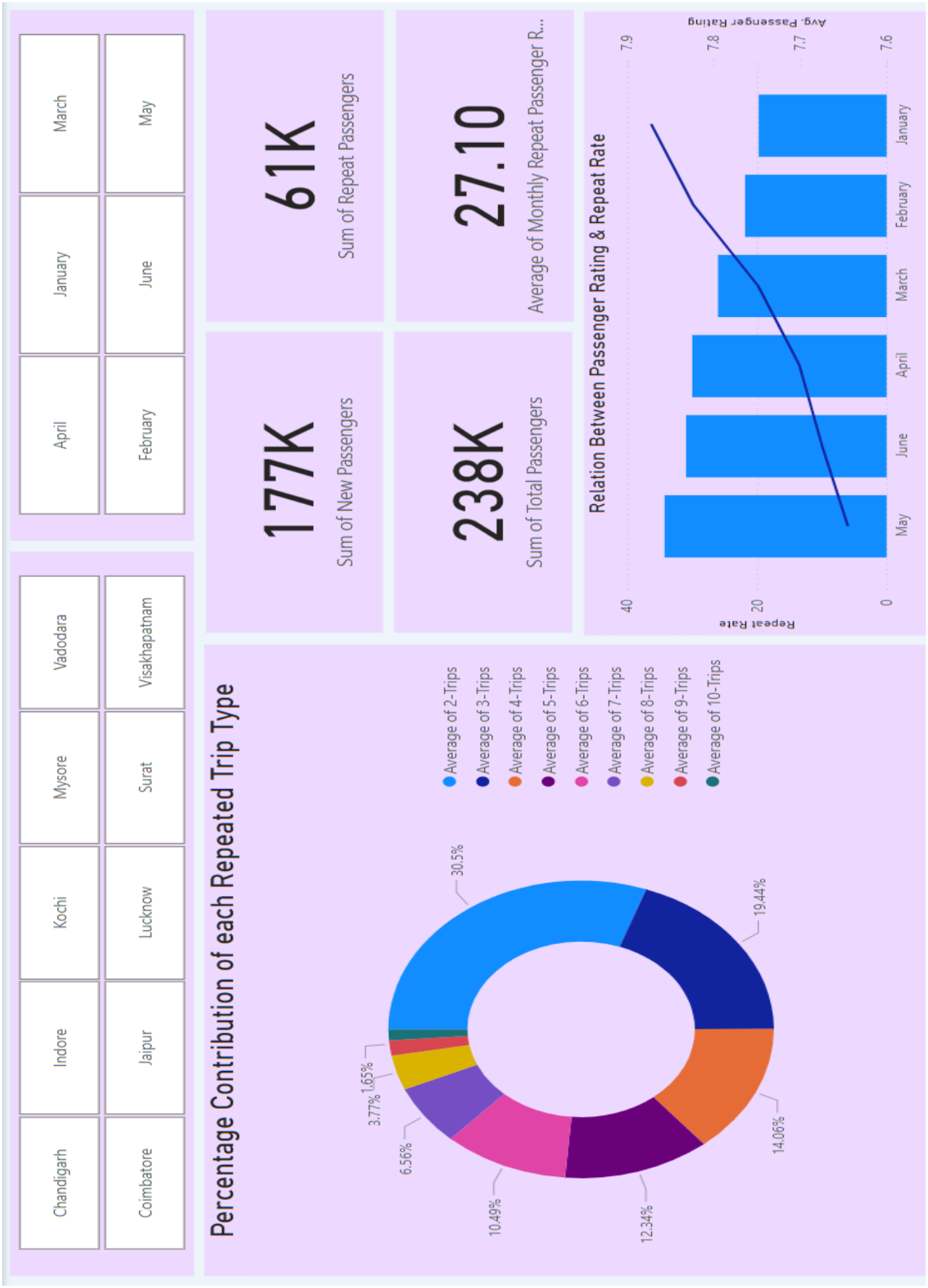
```
DELIMITER //
CREATE FUNCTION calculate_revenue(city VARCHAR(20))
RETURNS JSON
DETERMINISTIC
BEGIN
    DECLARE city_revenue INT;
    DECLARE total_revenue INT;
    DECLARE contribution INT;
    DECLARE result JSON;

    -- Calculate total revenue
    SELECT SUM(fare_amount) INTO total_revenue
    FROM fact_trips a
    INNER JOIN dim_city b
        ON a.city_id = b.city_id;

        SELECT SUM(fare_amount) INTO city_revenue
        FROM fact_trips a
        INNER JOIN dim_city b
            ON a.city_id = b.city_id
        WHERE city_name = city;

        SET contribution = (city_revenue / total_revenue) * 100;
        SET result = JSON_OBJECT(
            'city_name', city,
            'city_revenue', city_revenue,
            'total_revenue', total_revenue,
            'contribution_percentage', contribution
        );
    RETURN result;
END //
DELIMITER ;
```

```
mysql> SET @result = calculate_revenue('Jaipur');
Query OK, 0 rows affected (1.76 sec)

mysql> SELECT @result;
+-----------------------------------------------------------------------------------------------------------------+
| @result                                                                                                         |
+-----------------------------------------------------------------------------------------------------------------+
| {"city_name": "Jaipur", "city_revenue": 37207497, "total_revenue": 108188091, "contribution_percentage": 34} |
+-----------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Dashboard – Table Model



Repeat Rate


March
May
January
June
April
February
61K
Sum of Repeat Passengers
27.10
Average of Monthly Repeat Passenger R...
177K
Sum of New Passengers
238K
Sum of Total Passengers
Relation Between Passenger Rating & Repeat Rate
Avg. Passenger Rating
Repeat Rate
7.9
7.8
7.7
7.6
40
20
0
May
June
April
March
February
January
Vadodara
Visakhapatnam
Mysore
Surat
Kochi
Lucknow
Indore
Jaipur
Chandigarh
Coimbatore
Percentage Contribution of each Repeated Trip Type
Average of 2-Trips
Average of 3-Trips
Average of 4-Trips
Average of 5-Trips
Average of 6-Trips
Average of 7-Trips
Average of 8-Trips
Average of 9-Trips
Average of 10-Trips
30.5%
19.44%
14.06%
12.34%
10.49%
6.56%
3.77%
1.65%

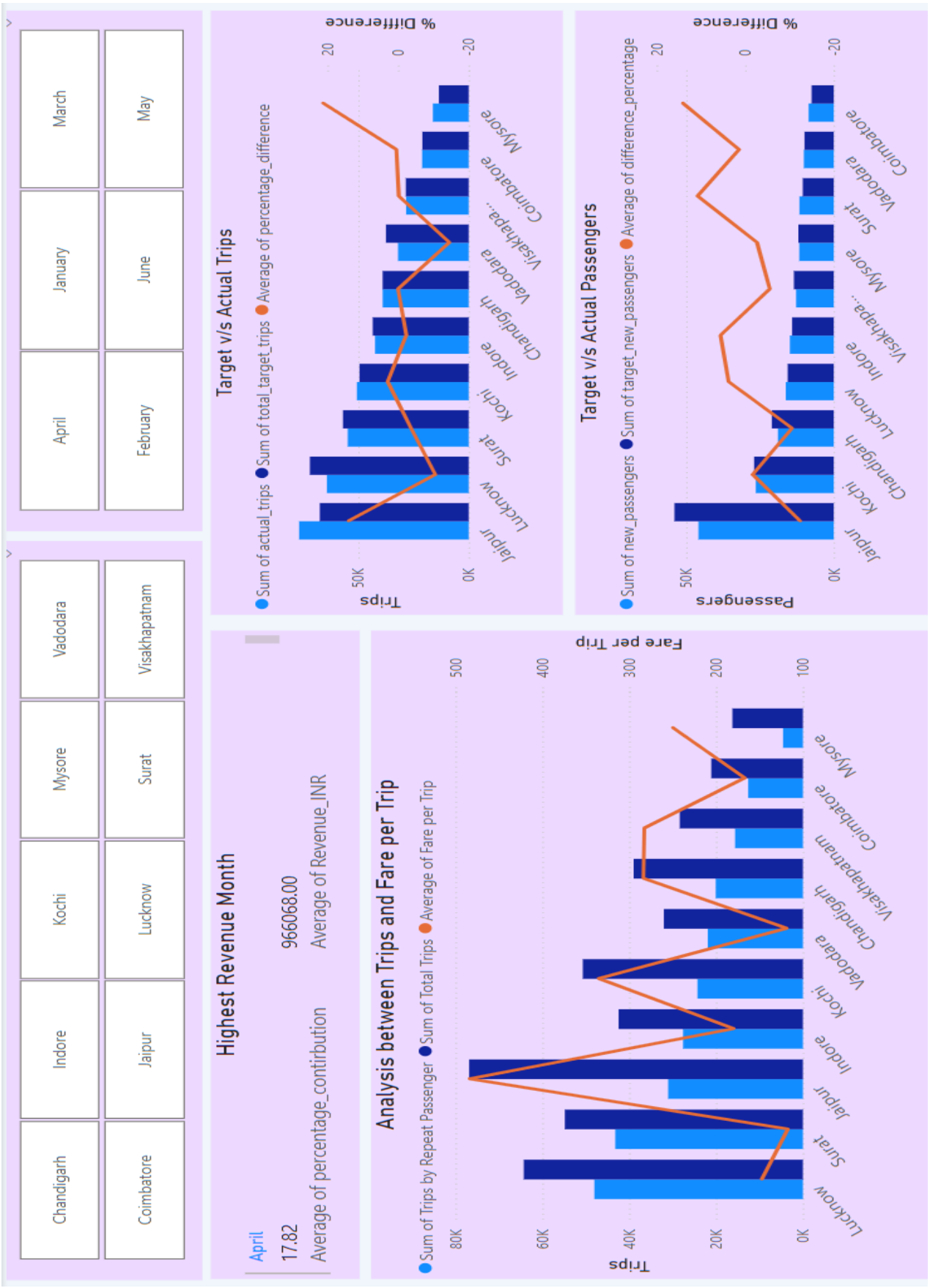
Description

The above dashboard provide following Features & Insights

- Interactive graphs providing Relation between Repeat Rate & Passenger Rating.
- Tiles providing exact numbers of New Passengers, Repeat Passengers, Monthly Repeat Rate.
- Charts presenting percentage contribution of each repeat trip type.
- All there features can be modified for a particular city and month for deeper analysis.

- The Repeat Passengers are higher in Surat & Lucknow.
- April & May have highest repeat rates.
- Highest number of trips taken across all cities is 2-trips.
- Least number of trips taken across all cities are 8,9 and 10 trips.
- Jaipur has highest number of trips.
- Mysore has highest passenger rating 8.7 across the 6 month period.
- Mysore, Coimbatore & Chandigarh has lowest repeat rates.
- Etc.

Revenue & Targets



Description

The above dashboard provide following Features & Insights

- It comes with double bar chart with line graph helps us to analyse Trips vs Fare per Trips.
- Another charts allow us to analyse actual passengers vs target passengers & actual trips vs target trips.

- April was the only month to meet the target for new passengers arrivals, while the other month fell short.
- Mysore, Kochi, Jaipur, met the target for passengers ratings while other cities did not.
- Highest Revenue generating month is February (1.99Cr) and revenue growth is 7.61%.
- Jaipur generated highest revenue (3.72Cr) among all the cities
- Mysore saw significant revenue growth in May (14.9%), Reason is May is holiday season of the year in India.
- Total trips and Target trips are important KPIs here. Both January and June did not meet the target. To reach the target in June, approximately 11.34% more trips are needed, which is higher than the requirement for January.