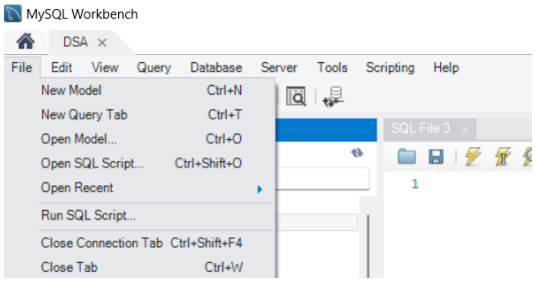
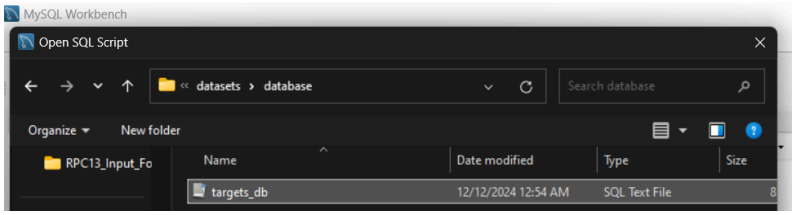
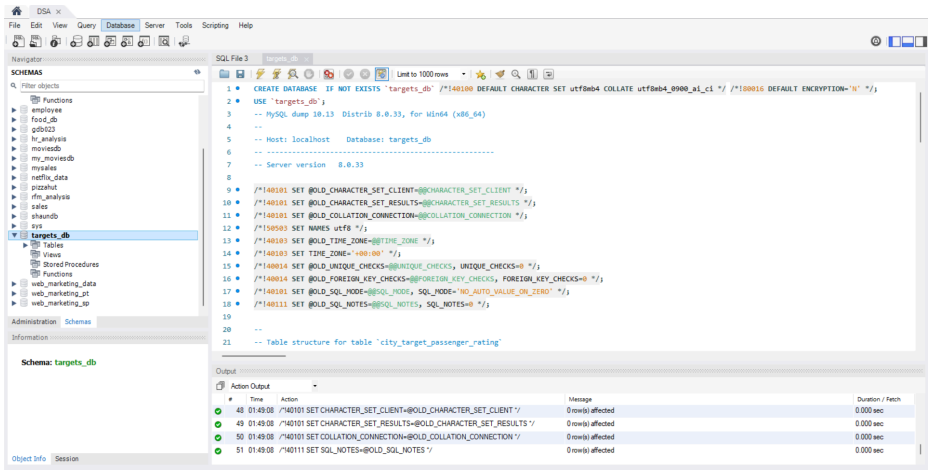
SQL Documentation

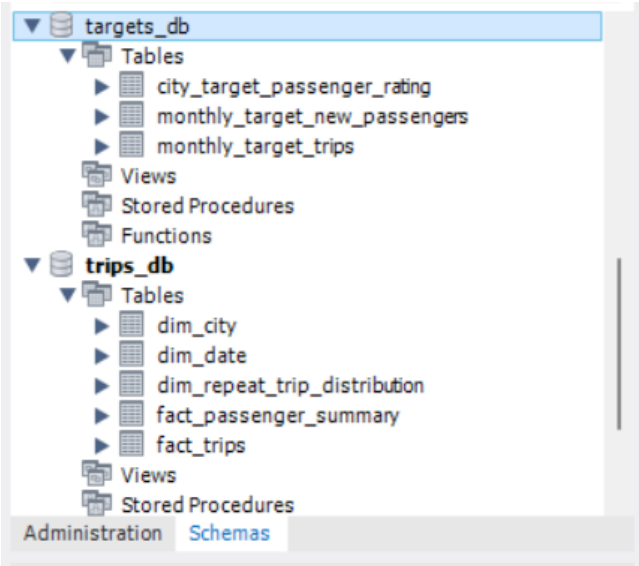
Import database:

importing the 'trips_db' and 'targets_db' databases into MySQL Workbench:









Completed Importing database

Ad_Hoc_Requests

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

Code:

```
use trips_db;
SELECT
 dc.city_name,

 COUNT(ft.trip_id) AS total_trips,

 ROUND(SUM(ft.fare_amount) / SUM(ft.distance_travelled_km),

 1) AS avg_fare_per_km,

 ROUND(SUM(ft.fare_amount) / COUNT(ft.trip_id),

 1) AS avg_fare_per_trip,

 ROUND(COUNT(ft.trip_id) * 100.0 / (SELECT
 COUNT(trip_id)
 FROM
 fact_trips),

 1) AS pct_cont_total_trips
FROM
 fact_trips ft
```

```
 JOIN
 dim_city dc
    ON dc.city_id = ft.city_id
GROUP BY dc.city_name
ORDER BY pct_cont_total_trips DESC;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city_name	total_trips	avg_fare_per_km	avg_fare_per_trip	pct_cont_total_trips
Jaipur	76888	16.1	483.9	18.1
Lucknow	64299	11.8	147.2	15.1
Surat	54843	10.7	117.3	12.9
Kochi	50702	13.9	335.2	11.9
Indore	42456	10.9	179.8	10.0
Chandigarh	38981	12.1	283.7	9.2
Vadodara	32026	10.3	118.6	7.5
Visakhapatnam	28366	12.5	282.7	6.7
Coimbatore	21104	11.1	167.0	5.0
Mysore	16238	15.1	249.7	3.8

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

Code:

```
WITH cte1 AS (
 -- Calculate actual trips per city
        AND month
SELECT
 ft.city_id,

 dc.city_name,

 dd.month_name,

 COUNT(DISTINCT ft.trip_id) AS actual_trips

    FROM fact_trips ft

JOIN dim_city dc

    ON ft.city_id = dc.city_id

JOIN dim_date dd

    ON ft.date = dd.date

    GROUP BY ft.city_id,
        dc.city_name,
        dd.month_name
```

```
)

SELECT
 cte1.city_name,

 cte1.month_name,

 cte1.actual_trips,

 tt.total_target_trips,

 CASE

    WHEN cte1.actual_trips > tt.total_target_trips THEN
    'Above Target'

    ELSE 'Below Target'

    END AS performance_status,
 CONCAT(ROUND((cte1.actual_trips - tt.total_target_trips) * 100.0 /
tt.total_target_trips, 2), '%') AS pct_diff
FROM targets_db.monthly_target_trips tt
JOIN cte1

    ON cte1.city_id = tt.city_id

        AND cte1.month_name = monthname(tt.month)
ORDER BY cte1.city_name, cte1.month_name;
```

Output:

city_name	month_name	actual_trips	total_target_trips	performance_status	pct_diff
Chandigarh	April	5566	6000	Below Target	-7.23%
Chandigarh	February	7387	7000	Above Target	5.53%
Chandigarh	January	6810	7000	Below Target	-2.71%
Chandigarh	June	6029	6000	Above Target	0.48%
Chandigarh	March	6569	7000	Below Target	-6.16%
Chandigarh	May	6620	6000	Above Target	10.33%
Coimbatore	April	3661	3500	Above Target	4.60%

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

Code:

```
SELECT
 c.city_name,
```

```sql
-- Calculate the percentage of repeat passengers who took 2,
        3,
        4... up to 10 trips
ROUND(SUM(
    CASE
    WHEN d.trip_count = '2-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "2-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '3-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "3-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '4-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "4-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '5-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "5-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '6-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "6-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '7-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "7-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '8-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
SUM(d.repeat_passenger_count) * 100, 2) AS "8-Trips",
ROUND(SUM(
    CASE
    WHEN d.trip_count = '9-Trips' THEN
    d.repeat_passenger_count
```

```
    ELSE 0 END) /
 SUM(d.repeat_passenger_count) * 100, 2) AS "9-Trips",
 ROUND(SUM(
    CASE
    WHEN d.trip_count = '10-Trips' THEN
    d.repeat_passenger_count
    ELSE 0 END) /
 SUM(d.repeat_passenger_count) * 100, 2) AS "10-Trips"
FROM
 dim_repeat_trip_distributiON d
JOIN
 dim_city c
    ON d.city_id = c.city_id
WHERE
 d.trip_count IN ('2-Trips', '3-Trips', '4-Trips', '5-Trips', '6-Trips',
'7-Trips', '8-Trips', '9-Trips', '10-Trips')
GROUP BY
 c.city_name
ORDER 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
Chandigarh	32.31	19.25	15.74	12.21	7.42	5.48	3.47	2.33	1.79
Coimbatore	11.21	14.82	15.56	20.62	17.64	10.47	6.15	2.31	1.22
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
Kochi	47.67	24.35	11.81	6.48	3.91	2.11	1.65	1.21	0.81
Lucknow	9.66	14.77	16.20	18.42	20.18	11.33	6.43	1.91	1.10
Mysore	48.75	24.44	12.73	5.82	4.06	1.76	1.42	0.54	0.47

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

Code:

```
WITH Highest
        AND Lowest Total New Passengers

WITH CityPassengerCounts AS (
 -- Calculate total new passengers fOR each city
SELECT
 dc.city_name,

 SUM(fps.new_passengers) AS total_new_passengers

    FROM
 trips_db.fact_passenger_summary fps

JOIN
 trips_db.dim_city dc
```

```
    ON
 fps.city_id = dc.city_id

    GROUP BY
 dc.city_name
),
CityRanking AS (
 -- Rank cities based
    ON total new passengers, both fOR top 3
        AND bottom 3
SELECT
 city_name,

 total_new_passengers,

 DENSE_RANK() OVER(ORDER BY total_new_passengers DESC) AS top_ranking,

 DENSE_RANK() OVER(ORDER BY total_new_passengers ASC) AS bottom_ranking

    FROM
 CityPassengerCounts
)
--SELECT the top 3
        AND bottom 3 cities
SELECT
 city_name,

 total_new_passengers,


    CASE

    WHEN top_ranking <= 3 THEN
    'Top 3'

    WHEN bottom_ranking <= 3 THEN
    'Bottom 3'

    ELSE NULL

    END AS city_category
FROM
 CityRanking
WHERE
 top_ranking <= 3
        OR bottom_ranking <= 3
ORDER BY
 city_category DESC, total_new_passengers DESC;
```

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

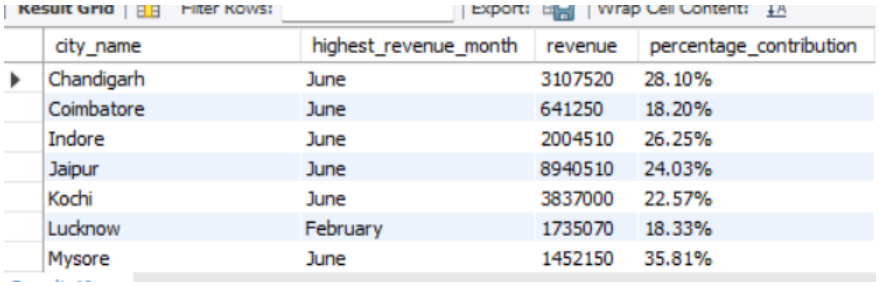
Business Request 5: Identify Month with Highest Revenue for Each City

Code:

```
WITH Highest Revenue fOR Each City
WITH RevenuePerCityMonth AS (
 -- Calculate total revenue fOR each city
        AND month
SELECT
 dc.city_name,

 dd.month_name,

 SUM(ft.fare_amount) AS total_revenue

    FROM
 trips_db.fact_trips ft

JOIN
 trips_db.dim_city dc

    ON
 ft.city_id = dc.city_id

JOIN
 trips_db.dim_date dd

    ON
 ft.date = dd.start_of_month

    GROUP BY
 dc.city_name,
        dd.month_name
),

CityTotalRevenue AS (
 -- Calculate total revenue fOR each city
SELECT
 dc.city_name,
```

```
 SUM(ft.fare_amount) AS total_revenue

    FROM
 trips_db.fact_trips ft

JOIN
 trips_db.dim_city dc

    ON
 ft.city_id = dc.city_id

    GROUP BY
 dc.city_name
)
SELECT
 r.city_name,

 r.month_name AS highest_revenue_month,

 r.total_revenue AS revenue,

 CONCAT(ROUND((r.total_revenue / tr.total_revenue) * 100,
        2),
        '%') AS percentage_contribution
FROM
 RevenuePerCityMonth r
JOIN
 CityTotalRevenue tr
ON
 r.city_name = tr.city_name
WHERE
 r.total_revenue = (
SELECT MAX(total_revenue)

    FROM RevenuePerCityMonth

    WHERE city_name = r.city_name
 )
ORDER BY
 r.city_name;
```

Output:



city_name	highest_revenue_month	revenue	percentage_contribution
Chandigarh	June	3107520	28.10%
Coimbatore	June	641250	18.20%
Indore	June	2004510	26.25%
Jaipur	June	8940510	24.03%
Kochi	June	3837000	22.57%
Lucknow	February	1735070	18.33%
Mysore	June	1452150	35.81%

Business Request 6: Repeat Passenger Rate Analysis

Code:

```
WITH monthly_repeat AS (
SELECT
 dc.city_name,

 dd.month_name,

 SUM(fps.total_passengers) AS total_passengers,

 SUM(fps.repeat_passengers) AS repeat_passengers,

 CONCAT(ROUND((SUM(fps.repeat_passengers) * 100.0 /
SUM(fps.total_passengers)),
        2),
        '%') AS monthly_repeat_passenger_rate

    FROM
 trips_db.fact_passenger_summary fps

JOIN
 trips_db.dim_city dc
    ON fps.city_id = dc.city_id

JOIN
 trips_db.dim_date dd
    ON fps.month = dd.start_of_month

    GROUP BY
 dc.city_name,
        dd.month_name
),


city_repeat AS (
SELECT
 dc.city_name,

 SUM(fps.total_passengers) AS total_passengers,
```

```
 SUM(fps.repeat_passengers) AS repeat_passengers,

 CONCAT(ROUND((SUM(fps.repeat_passengers) * 100.0 /
SUM(fps.total_passengers)),
        2),
        '%') AS city_repeat_passenger_rate

    FROM
 trips_db.fact_passenger_summary fps

JOIN
 trips_db.dim_city dc
    ON fps.city_id = dc.city_id

    GROUP BY
 dc.city_name
)

SELECT
 mr.city_name,

 mr.month_name,

 mr.total_passengers,

 mr.repeat_passengers,

 mr.monthly_repeat_passenger_rate,

 cr.city_repeat_passenger_rate
FROM
 monthly_repeat mr
JOIN
 city_repeat cr
    ON mr.city_name = cr.city_name
ORDER BY
 mr.city_name, mr.month_name;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city_name	month_name	total_passengers	repeat_passengers	monthly_repeat_passenger_rate	city_repeat_passenger_r
Chandigarh	April	98550	23670	24.02%	21.14%
Chandigarh	February	143753	24737	17.21%	21.14%
Chandigarh	January	143840	22320	15.52%	21.14%
Chandigarh	June	98910	26010	26.30%	21.14%
Chandigarh	March	127100	27032	21.27%	21.14%
Chandigarh	May	114669	30039	26.20%	21.14%
Coimbatore	April	51660	14400	27.87%	23.05%

Primary & Secondary_questions_Analysis_Using_SQL

1. **Top and Bottom Performing Cities**

   Code:

```
-- Top 3 Performing Cities by Total Trips
SELECT
 dim_city.city_name,

 SUM(fact_passenger_summary.total_passengers) AS total_trips
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name
ORDER BY
 total_trips DESC
LIMIT 3;
-- Bottom 3 Performing Cities by Total Trips

SELECT
 dim_city.city_name,

 SUM(fact_passenger_summary.total_passengers) AS total_trips
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name
ORDER BY
 total_trips ASC
LIMIT 3;


-- alternative

(SELECT
 city_name,

 SUM(fact_passenger_summary.total_passengers) AS total_trips,

 'Top 3' AS performance
FROM
 fact_passenger_summary
```

```
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name
ORDER BY
 total_trips DESC
LIMIT 3)

UNION ALL

(SELECT
 city_name,

 SUM(fact_passenger_summary.total_passengers) AS total_trips,

 'Bottom 3' AS performance
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name
ORDER BY
 total_trips ASC
LIMIT 3);
```

Output:

city_name	total_trips	performance
Jaipur	55538	Top 3
Kochi	34042	Top 3
Lucknow	25857	Top 3
Coimbatore	11065	Bottom 3
Mysore	13158	Bottom 3
Vadodara	14473	Bottom 3

2. Average Fare per Trip by City

```
SELECT
 dim_city.city_name,

 AVG(fact_trips.fare_amount) AS average_fare,

 AVG(fact_trips.distance_travelled_km) AS average_trip_distance
FROM
 fact_trips
JOIN
```

```
 dim_city
    ON fact_trips.city_id = dim_city.city_id
GROUP BY
 Dim_city.city_name;
```

Output:

city_name	average_fare	average_trip_distance
Chandigarh	283.6870	23.5187
Coimbatore	166.9822	14.9792
Indore	179.8386	16.5025
Jaipur	483.9181	30.0231
Kochi	335.2451	24.0655
Lucknow	147.1804	12.5130

3. Average Ratings by City and Passenger Type

```
        AND Passenger Type
SELECT
 dim_city.city_name,

 fact_trips.passenger_type,

 AVG(fact_trips.passenger_rating) AS avg_passenger_rating,

 AVG(fact_trips.driver_rating) AS avg_driver_rating
FROM
 fact_trips
JOIN
 dim_city
    ON fact_trips.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name, fact_trips.passenger_type;
```

Output:

city_name	passenger_type	avg_passenger_rating	avg_driver_rating
Chandigarh	new	8.4892	7.9921
Chandigarh	repeated	7.4938	7.4728
Coimbatore	new	8.4858	7.9906
Coimbatore	repeated	7.4755	7.4808
Indore	new	8.4858	7.9708
Indore	repeated	7.4740	7.4774

4. Peak and Low Demand Months by City

```
-- Peak DemAND Month by City
SELECT
 dim_city.city_name,

 DATE_FORMAT(fact_passenger_summary.month,
```

```
        '%Y-%m') AS month,

 SUM(fact_passenger_summary.total_passengers) AS total_trips
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name,
        month
ORDER BY
 total_trips DESC
LIMIT 1;

-- Low DemAND Month by City

SELECT
 dim_city.city_name,

 DATE_FORMAT(fact_passenger_summary.month,
        '%Y-%m') AS month,

 SUM(fact_passenger_summary.total_passengers) AS total_trips
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name,
        month
ORDER BY
 total_trips ASC
LIMIT 1;

-- alternative
WITH RankedTrips AS (
SELECT
 dim_city.city_name,

 DATE_FORMAT(fact_passenger_summary.month,
        '%Y-%m') AS month,

 SUM(fact_passenger_summary.total_passengers) AS total_trips,

 ROW_NUMBER()
    OVER (PARTITION BY dim_city.city_name
ORDER BY SUM(fact_passenger_summary.total_passengers) DESC) AS peak_rank,
```

```
 ROW_NUMBER()
    OVER (PARTITION BY dim_city.city_name
ORDER BY SUM(fact_passenger_summary.total_passengers) ASC) AS low_rank

    FROM
 fact_passenger_summary

JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id

    GROUP BY
 dim_city.city_name,
        month
)

SELECT
 city_name,

 month,

 total_trips,

 'Peak Demand' AS demand_type -- Label fOR peak demand
FROM
 RankedTrips
WHERE
 peak_rank = 1 -- Peak demAND month (highest total trips)

UNION ALL

SELECT
 city_name,

 month,

 total_trips,

 'Low Demand' AS demand_type -- Label fOR low demand
FROM
 RankedTrips
WHERE
 low_rank = 1; -- Low demAND month (lowest total trips)
```

Output:

city_name	month	total_trips	demand_type
Surat	2024-01	3616	Peak Demand
Vadodara	2024-02	2756	Peak Demand
Visakhapatnam	2024-02	3170	Peak Demand
Chandigarh	2024-04	3285	Low Demand
Coimbatore	2024-05	1543	Low Demand
Indore	2024-06	3152	Low Demand

5. WeekEND vs. Weekday Trip Demand by City

```
SELECT
 dim_city.city_name,

 dim_date.day_type,

 SUM(fact_passenger_summary.total_passengers) AS total_trips
FROM
 fact_passenger_summary
JOIN
 dim_city
    ON fact_passenger_summary.city_id = dim_city.city_id
JOIN
 dim_date
    ON fact_passenger_summary.month = dim_date.start_of_month
GROUP BY
 dim_city.city_name, dim_date.day_type;
```

Output:

city_name	day_type	total_trips
Lucknow	Weekday	559872
Coimbatore	Weekday	239973
Jaipur	Weekday	1205236
Indore	Weekday	479087
Kochi	Weekday	741101
Mysore	Weekday	284985

6. Repeat Passenger Frequency and City Contribution Analysis

```
SELECT
 dim_city.city_name,

 dim_repeat_trip_distribution.trip_count,

 SUM(dim_repeat_trip_distribution.repeat_passenger_count) AS
repeat_passenger_count
FROM
 dim_repeat_trip_distribution
JOIN
 dim_city
    ON dim_repeat_trip_distribution.city_id = dim_city.city_id
GROUP BY
 dim_city.city_name,
```

```
        dim_repeat_trip_distribution.trip_count
ORDER BY
 SUM(dim_repeat_trip_distribution.repeat_passenger_count) DESC;
```

Output:

city_name	trip_count	repeat_passenger_count
Jaipur	2-Trips	4855
Kochi	2-Trips	3635
Visakhapatnam	2-Trips	2618
Indore	2-Trips	2478
Jaipur	3-Trips	2007
Lucknow	6-Trips	1937

7. Monthly Target Achievement Analysis for Key Metrics

```
SELECT
 dim_city.city_name,

 dim_repeat_trip_distribution.trip_count,

 SUM(dim_repeat_trip_distribution.repeat_passenger_count) AS
repeat_passenger_count
FROM
 trips_db.dim_repeat_trip_distribution
JOIN
 trips_db.dim_city
    ON trips_db.dim_repeat_trip_distribution.city_id =
trips_db.dim_city.city_id
GROUP BY
 dim_city.city_name, dim_repeat_trip_distribution.trip_count
ORDER BY
 repeat_passenger_count DESC
LIMIT 1000;
```

Output:

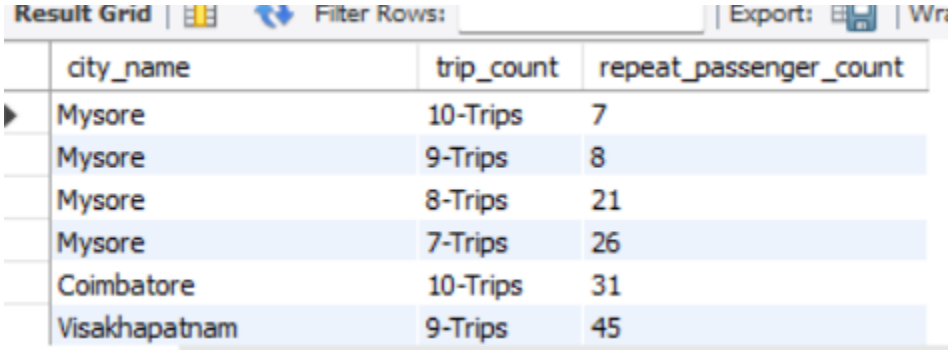
city_name	trip_count	repeat_passenger_count
Jaipur	2-Trips	4855
Kochi	2-Trips	3635
Visakhapatnam	2-Trips	2618
Indore	2-Trips	2478
Jaipur	3-Trips	2007
Lucknow	6-Trips	1937

8. Highest and Lowest Repeat Passenger Rate (RPR%) by City and Month

```
SELECT
 dim_city.city_name,

 dim_repeat_trip_distribution.trip_count,
```

```
 SUM(dim_repeat_trip_distribution.repeat_passenger_count) AS
repeat_passenger_count
FROM
 trips_db.dim_repeat_trip_distribution
JOIN
 trips_db.dim_city
    ON trips_db.dim_repeat_trip_distribution.city_id =
trips_db.dim_city.city_id
GROUP BY
 dim_city.city_name, dim_repeat_trip_distribution.trip_count
ORDER BY
 repeat_passenger_count ASC
LIMIT 1000;
```

Output:



Result Grid | Filter Rows: | Export: | Wr

city_name	trip_count	repeat_passenger_count
Mysore	10-Trips	7
Mysore	9-Trips	8
Mysore	8-Trips	21
Mysore	7-Trips	26
Coimbatore	10-Trips	31
Visakhapatnam	9-Trips	45

The END