AD-HOC-Requests SQL

Business Requests

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 of Total Trips

Query

```sql
select city_name , count(*) as Total_trips ,
    round((sum(fare_amount)/sum(distance_travelled_km)),2) as avg_fare_per_km,
    round((sum(fare_amount)/count(city_name)),2) as avg_fare_per_trip,
    round((count(*) / sum(count(*)) over ()*100),2) as '%_contribution_total'
    from trips_db.fact_trips ft
join trips_db.dim_city dc on ft.city_id = dc.city_id
group by city_name;
```

Output

city_name	Total_trips	avg_fare_per_km	avg_fare_per_trip	%_contribution_total
Visakhapatnam	28366	12.53	282.67	6.66
Chandigarh	38981	12.06	283.69	9.15
Surat	54843	10.66	117.27	12.88
Vadodara	32026	10.29	118.57	7.52
Mysore	16238	15.14	249.71	3.81
Kochi	50702	13.93	335.25	11.90
Indore	42456	10.90	179.84	9.97
Jaipur	76888	16.12	483.92	18.05
Coimbatore	21104	11.15	166.98	4.96
Lucknow	64299	11.76	147.18	15.10

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 categories 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

Query

```sql
with t1 as (
    select city_id , monthname(date) as Month_name ,count(*) as Actual_trips  from trips_db.fact_trips a
    group by city_id , monthname(date)
    ),
t2 as ( select city_id ,monthname(month) as Month_name ,sum(total_target_trips) as Actual_target from targets_db.monthly_target_trips a
group by city_id , monthname(month))

select city_name , t1.Month_name , t1.Actual_trips , t2.Actual_target ,
    (case when t1.Actual_trips > t2.Actual_target then "Above target " else "Below target" end ) as Performance_status,
    round(((t1.Actual_trips -  t2.Actual_target)/( t2.Actual_target))*100,2) as '% Difference'
from t1
join t2 on
    t1.city_id = t2.city_id
and
    t1.Month_name = t2.Month_name
join
trips_db.dim_city c
on
 t1.city_id = c.city_id;
```

Output

city_name	Month_name	Actual_trips	Actual_target	Performance_status	% 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

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.

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

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

Query

```sql
select
    city_name,
    round(sum(case when trip_count = '2-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 2_trip,
    round(sum(case when trip_count = '3-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 3_trip,
    round(sum(case when trip_count = '4-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 4_trip,
    round(sum(case when trip_count = '5-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 5_trip,
    round(sum(case when trip_count = '6-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 6_trip,
    round(sum(case when trip_count = '7-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 7_trip,
    round(sum(case when trip_count = '8-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 8_trip,
    round(sum(case when trip_count = '9-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 9_trip,
    round(sum(case when trip_count = '10-trips' then repeat_passenger_count else 0 end) / sum(repeat_passenger_count) * 100, 2) as 10_trip
from
    trips_db.dim_repeat_trip_distribution a
join
    trips_db.dim_city b
on
    a.city_id = b.city_id
group by
    city_name;
```

Output

city_name	2_trip	3_trip	4_trip	5_trip	6_trip	7_trip	8_trip	9_trip	10_trip
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, categorizing them as "Top 3" or "Bottom 3" accordingly.

•Fields
- city name
- total new passengers
- city category ("Top 3" or "Bottom 3")

Query

```sql
with t1 as (
select city_name , sum(new_passengers) as new_passengers ,
rank() over ( order by sum(new_passengers) desc) as rank_df
 from trips_db.fact_passenger_summary a join
trips_db.dim_city b on a.city_id = b.city_id
group by city_name),

t2 as (select city_name , new_passengers , 'Top' as 'Top/Bottom' , rank_df as Rank_ from t1 where rank_df < 4),
t3 as ( select city_name , new_passengers , 'Bottom' , rank() over (order by new_passengers asc ) as Rank_ from t1
            where rank_df >=(select max(rank_df) - 2 from t1))
select * from t2
union
select * from t3;
```

Output

city_name	new_passengers	Top/Bottom	Rank_
Jaipur	45856	Top	1
Kochi	26416	Top	2
Chandigarh	18908	Top	3
Coimbatore	8514	Bottom	1
Vadodara	10127	Bottom	2
Surat	11626	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 (%)

Query

```sql
with t1 as (
    select city_name,  month_name,  revenue
    from (
        select city_name,  monthname(date) as month_name, sum(fare_amount) as revenue,
            row_number() over (partition by city_name order by sum(fare_amount) desc) as aa
        from trips_db.fact_trips tf
        join trips_db.dim_city dc
            on tf.city_id = dc.city_id
        group by city_name, month_name
    ) a1
    where aa < 2
),

t2 as (
    select city_name,  sum(fare_amount) as total_amount
    from trips_db.fact_trips a
    join trips_db.dim_city b
        on a.city_id = b.city_id
    group by city_name
)

select
    t2.city_name, t1.month_name, t1.revenue as Highest_revenue ,
    (t1.revenue / t2.total_amount) * 100 as revenue_percentage
from t2 join t1
    on t1.city_name = t2.city_name;
```

Output

city_name	month_name	Highest_revenue	revenue_percentage
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

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

Query

```sql
select city_name ,monthname(month) as Month , sum(total_passengers) as Total_passengers ,
sum(repeat_passengers) as Repeat_passengers ,
round(sum(repeat_passengers)/sum(total_passengers)*100,2) as Repeat_passengers_percent
 from trips_db.fact_passenger_summary a
join
trips_db.dim_city b on a.city_id = b.city_id
group by city_name , Month;
```

Output

city_name	Month	Total_passengers	Repeat_passengers	Repeat_passengers_percent
Visakhapatnam	January	3163	650	20.55
Chandigarh	January	4640	720	15.52
Surat	January	3616	1184	32.74
Vadodara	January	2633	544	20.66
Mysore	January	2129	172	8.08
Kochi	January	5660	795	14.05
Indore	January	3876	1033	26.65
Jaipur	January	11845	1422	12.01
Coimbatore	January	2214	392	17.71
Lucknow	January	4896	1431	29.23
Visakhapatnam	February	3170	790	24.92
Chandigarh	February	4957	853	17.21
Surat	February	3567	1313	36.81
Vadodara	February	2756	610	22.13
Mysore	February	2290	183	7.99