ZOMATO DATA ANALYSIS

```sql
-- Q.1) Write a query to find the top 5 most frequently ordered dishes
SELECT
    customer_name,
    dishes,
    total_orders
FROM
(SELECT
    c.customer_id,
    c.customer_name,
    o.order_item as dishes,
    COUNT(*) as total_orders,
    DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) as rank
FROM orders as o
JOIN
customers as c
ON c.customer_id = o.customer_id
WHERE c.customer_name = 'Leo Martinez'
GROUP BY 1,2,3
ORDER BY 1,4 DESC) as t1
WHERE rank <= 5
```

Showing rows: 1 to 5 Page No: 1 of 1

	customer_name character varying (55)	dishes character varying (55)	total_orders bigint
1	Leo Martinez	Dosa	10
2	Leo Martinez	Vada	5
3	Leo Martinez	Upma	4
4	Leo Martinez	Fried Rice	1
5	Leo Martinez	Idli	1

```sql
--Q2 Pupular time slots
-- Identify the time slots during which most orders are placed

SELECT
    CASE
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 0 AND 1 THEN '00:00 -
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 2 AND 3 THEN '02:00 -
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 4 AND 5 THEN '04:00 -
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 6 AND 7 THEN '06:00 -
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 8 AND 9 THEN '08:00 -
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 10 AND 11 THEN '10:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 12 AND 13 THEN '12:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 14 AND 15 THEN '14:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 16 AND 17 THEN '16:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 18 AND 19 THEN '18:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 20 AND 21 THEN '20:00
       WHEN EXTRACT(HOUR FROM order_time) BETWEEN 22 AND 23 THEN '22:00
   END AS time_slot,
   COUNT(order_id) AS order_count
FROM orders
GROUP BY time_slot
ORDER BY order_count DESC;
```

Showing rows: 1 to 8 Page No: 1 of 1

	time_slot text	order_count bigint
1	14:00 -16:00	75
2	12:00 -14:00	46
3	20:00 -22:00	40
4	18:00 -20:00	36
5	16:00 -18:00	23
6	08:00 -10:00	22
7	10:00 -12:00	19
8	22:00 -00:00	1

```sql
-- Q3. ORDER VALUE ANALYSIS
-- Find the average order value per customer
-- return customer_name and aov(average order value)

SELECT
    c.customer_name,
    AVG(o.total_amount) as aov
FROM orders as o
    JOIN customers as c
    ON c.customer_id = o.customer_id
GROUP BY 1
```

	customer_name character varying (55)	aov double precision
1	Brian Turner	160
2	Zack Mitchell	70
3	Steve Wright	174.1578947368421
4	Jack Thompson	150
5	Felix Collins	320
6	Rachel King	158.57142857142858
7	Isla Howard	250
8	Lavanya Desai	100
9	Pratik Shah	260

```sql
--Q4. high vale customers
--list the customers eho have spent more than 1000 in total on food ord
--return customer_name, and customer_id


SELECT
    c.customer_id,
    c.customer_name,
    SUM(o.total_amount) as total_spent
FROM orders as o
    JOIN customers as c
    ON c.customer_id = o.customer_id
GROUP BY 1
HAVING SUM(o.total_amount) > 1000
```

Showing rows: 1 to 14 Page No: 1 of 1

	customer_id [PK] integer	customer_name character varying (55)	total_spent double precision
1	10	Jack Thompson	2100
2	22	Victor Adams	2340
3	40	Nate Morgan	1220
4	7	Grace Lee	4240
5	15	Olivia Lewis	1550
6	12	Leo Martinez	2065
7	19	Steve Wright	3309
8	30	Daniel Parker	1210
9	21	Uma Hall	4800

```sql
--05. Restaurant Revenue Ranking
--Rank restaurants by their total revenue from last year , including th
--total revenue and rank within their city


SELECT
    r.city,
    r.restaurant_name,
   SUM(o.total_amount) as revenue,
   RANK() OVER(PARTITION BY r.city ORDER BY SUM(o.total_amount)DESC) as
FROM orders as o
JOIN
restaurants as r
ON r.restaurant_id = o.restaurant_id
GROUP BY 1,2
ORDER BY 1,3 DESC
```

	city character varying (25)	restaurant_name character varying (55)	revenue double precision	rank bigint
1	Bengaluru	Tandoori Nights	2410	1
2	Bengaluru	Urban Chulha	440	2
3	Bengaluru	Masaledaar Meals	170	3
4	Bengaluru	Taste of Punjab	150	4
5	Bengaluru	Urban Udupi	120	5
6	Chennai	Masala Magic	3100	1
7	Chennai	Bhojanam Bhavan	320	2
8	Chennai	Chatori Chaat	199	3
9	Chennai	Dakshin Delight	170	4
10	Chennai	Andhra Annapurna	100	5
11	Chennai	Rasoi Rasam	95	6
12	Delhi	Curry Leaf	2280	1

```sql
--Q.7
--Most popular dish by city
-- Identify the most popular dish in each city based on the number of o

SELECT
    r.city,
    o.order_item as dish,
    COUNT(order_id) as total_orders,
    RANK() OVER(PARTITION BY r.city ORDER BY COUNT(order_id)DESC) as r
FROM orders as o
JOIN
restaurants as r
ON r.restaurant_id = o.restaurant_id
GROUP BY 1,2
```

Showing rows: 1 to 85 Page No: 1 of 1

	city character varying (25)	dish character varying (55)	total_orders bigint	rank bigint
1	Bengaluru	Noodles	16	1
2	Bengaluru	Biryani	2	2
3	Bengaluru	Masala Dosa	1	3
4	Bengaluru	Schezwan Rice	1	3
5	Bengaluru	Rajma Chawal	1	3
6	Chennai	Noodles	16	1
7	Chennai	Veg Biryani	1	2
8	Chennai	Mutton Curry	1	2

```sql
--Q.8 find customers who havent placed orders in 2024 but did in 2023

SELECT * FROM orders
WHERE
    EXTRACT(YEAR FROM order_date) = 2023
    AND
    customer_id NOT IN
    (SELECT DISTINCT customer_id FROM orders
      WHERE EXTRACT(YEAR FROM order_date)=2024)
```

order_id [PK] integer	customer_id integer	restaurant_id integer	order_item character varying (55)	order_date date

```sql
--Q.9 Cancellation Rate Comparison
--Calculate and compare the order cancellation rate for each restaurant
-- previous year

SELECT
    o.restaurant_id,
    COUNT(o.order_id) as total_orders,
    COUNT(CASE WHEN d.delivery_status = 'not delivered' THEN 1 END) AS
FROM orders as o
LEFT JOIN
deliveries as d
ON o.order_id = d.order_id
GROUP BY 1
```

Showing rows: 1 to 59 Page No: 1 of 1

	restaurant_id integer	total_orders bigint	not_delivered bigint
1	54	1	0
2	29	1	0
3	4	15	0
4	34	1	0
5	52	1	0
6	10	1	0
7	35	1	0
8	45	1	0

```sql
--Q10) average delivery time taken by rider

SELECT
        o.order_id,
        o.order_time,
        d.delivery_time,
        d.rider_id,
        d.delivery_time - o.order_time AS time_difference,
        EXTRACT(EPOCH FROM(d.delivery_time - o.order_time +
        CASE WHEN d.delivery_time < o.order_time THEN INTERVAL '1days'
        as time_difference_insec
FROM orders as o
JOIN
deliveries as d
ON
o.order_id = d.order_id
WHERE d.delivery_status = 'Delivered'
```

Showing rows: 1 to 212 Page No: 1 of 1

	order_id integer	order_time time without time zone	delivery_time time without time zone	rider_id integer	time_ interv
1	1	12:30:00	11:22:10	1	-01:07
2	2	13:00:00	12:30:45	2	-00:29
3	4	15:45:00	13:10:20	4	-02:34
4	5	11:05:00	15:40:05	5	04:35
5	7	13:00:00	16:10:00	7	03:10
6	8	14:30:00	10:25:18	8	-04:04
7	9	15:00:00	12:05:59	9	-02:54
8	11	12:50:00	09:30:00	11	-03:20
9	12	13:45:00	18:05:11	12	04:20
10	13	14:00:00	14:35:15	13	00:35

```sql
--Q.11 Monthly Restaurant Growth Rate
-- Calculate each restaurant's growth ratio based on the total
--number of delivered orders since its joining
WITH growth_ratio
AS
(
SELECT
    restaurant_id,
    TO_CHAR(o.order_date, 'mm-yy') as month,
    COUNT(o.order_id) as current_month_orders,
    LAG(COUNT(o.order_id), 1) OVER(PARTITION BY o.restaurant_id ORDER B
FROM orders as o
JOIN deliveries as d
ON
o.order_id = d.order_id
WHERE d.delivery_status = 'Delivered'
GROUP BY 1,2
ORDER BY 1,2
)
SELECT
     restaurant_id,
     month,
     current_month_orders,
     prev_month_orders,
     ROUND((current_month_orders::numeric-prev_month_orders::numeric)/p
FROM growth_ratio
```

Showing rows: 1 to 75 Page No: 1 of 1

	restaurant_id integer	month text	current_month_orders bigint	prev_month_orders bigint	month numeri
1	1	01-24	1	[null]	
2	2	01-24	1	[null]	
3	2	03-23	11	1	
4	2	04-23	1	11	
5	3	01-24	1	[null]	
6	3	04-23	10	1	
7	4	01-24	1	[null]	
8	4	02-23	12	1	
9	5	03-23	3	[null]	
10	5	04-23	10	3	
11	6	02-23	1	[null]	
12	6	04-23	6	1	
13	6	05-23	2	6	
14	7	01-23	9	[null]	
15	7	02-24	1	9	
16	8	01-24	1	[null]	
17	9	02-23	6	[null]	
18	9	03-23	14	6	

```sql
--Q12 Customer Segmentation
--Customer segmentarion into gold and siler groups based on their total
--compared to the average order value (AOV). If a customers total spend
--label them as 'gold' otherwise label them as 'silver'

SELECT
    cx_category,
    SUM(total_orders),
    SUM(total_spent)
FROM
(SELECT
        customer_id,
        SUM(total_amount) as total_spent,
        COUNT (order_id) as total_orders,
        CASE
        WHEN SUM(total_amount) > (SELECT AVG(total_amount) FROM orders)
            ElSE 'Silver'
        END as cx_category
FROM orders
group by 1
) as t1
GROUP BY 1


SELECT AVG(total_amount) FROM orders ---156
```

Showing rows: 1 to 2 Page No: 1 of 1

	cx_category text	sum numeric	sum double precision
1	gold	235	38263
2	Silver	27	2845

```sql
--Q13 Riders monthly earnings
--calcualte each riders'total monthly earings , assuming they earn 8% o


SELECT
    d.rider_id,
    TO_CHAR(o.order_date, 'mm-yy')as month,
    SUM(total_amount) as revenue,
    SUM(total_amount)* 0.8 as riders_earnings
FROM orders as o
JOIN deliveries as d
ON o.order_id = d.order_id
GROUP BY 1,2
ORDER BY 1, 2
```

Showing rows: 1 to 191 Page No: 1 of 1

	rider_id integer	month text	revenue double precision	riders_earnings double precision
1	1	01-24	220	176
2	1	02-23	80	64
3	1	03-23	280	224
4	1	04-23	320	256
5	2	01-24	299	239.20000000000002
6	2	03-23	160	128
7	2	04-23	460	368
8	3	01-24	180	144
9	3	03-23	160	128
10	3	04-23	150	120

```sql
--Q.14 order frequency by day
-- analyse order frequency per day of the week and identify the peak da

SELECT * FROM
(SELECT
    r.restaurant_name,
    --o.order_date,
    TO_CHAR(o.order_date,'day') as day,
    COUNT(o.order_id) as total_orders,
    RANK() OVER(PARTITION BY r.restaurant_name ORDER BY COUNT(o.order_i
FROM orders as o
JOIN
restaurants as r
ON o.restaurant_id = r.restaurant_id
GROUP BY 1,2
ORDER  BY 1,3 DESC)

WHERE rank = 1
```

Showing rows: 1 to 71 | Page No: 1 of 1

	restaurant_name character varying (55)	day text	total_orders bigint	rank bigint
1	Andhra Annapurna	thursday	1	1
2	Andhra Kitchen	tuesday	1	1
3	Bhojanam Bhavan	thursday	1	1
4	Bhukkad Adda	sunday	1	1
5	Bombay Bistro	tuesday	1	1
6	Chat Masala Express	friday	1	1
7	Chatkara Junction	friday	1	1
8	Chatori Chaat	wednesday	1	1
9	Chatori Gully	thursday	3	1
10	Chatpata Bhojanalay	wednesday	1	1
11	Chatpata Chatori	saturday	1	1

```sql
--Q.15 Customer lifetime value (CLV)
--Calculate the total revenue generated by each customer over all their

SELECT
    o.customer_id,
    c.customer_name,
    SUM(total_amount) as CLV
FROM orders as o

JOIN customers as c
ON o.customer_id = c.customer_id
GROUP BY 1,2
```

Showing rows: 1 to 59 | Page No: 1 of 1

	customer_id integer	customer_name character varying (55)	clv double precision
1	12	Leo Martinez	2065
2	21	Uma Hall	4800
3	40	Nate Morgan	1220
4	16	Paul Young	130
5	31	Ella Evans	95
6	54	Meera Reddy	130
7	19	Steve Wright	3309

```sql
--q.16 Monthly Sales Trends
--Identify sales trends by comparing each month's total sales to the pr

SELECT
    EXTRACT(YEAR FROM order_date) as year,
    EXTRACT(MONTH FROM order_date) as month,
    SUM(total_amount) as total_sale,
    LAG(SUM(total_amount), 1) OVER(ORDER BY EXTRACT(YEAR FROM order_dat
FROM orders
GROUP BY 1,2
ORDER BY 1,2
```

	year numeric	month numeric	total_sale double precision	previous_month_sale double precision
1	2023	1	520	[null]
2	2023	2	8120	520
3	2023	3	11010	8120
4	2023	4	10430	11010
5	2023	5	1860	10430
6	2024	1	6193	1860
7	2024	2	2975	6193