ZOMATO SALES ANALYSIS

```sql
-- 1. List the top 3 customers who have placed the highest number of orders.

SELECT c.customer_id, c.customer_name, COUNT(od.order_id) AS total_orders
FROM order_detail od
JOIN customer c on c.customer_id = od.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_orders DESC
LIMIT 3;
```

customer_id	customer_name	total_orders
1001	vaibhav Palkar	11
1008	punam kale	4
1003	sagar amritkar	4

```sql
-- 2. Find the restaurant that has received the highest average rating

SELECT restaurant_id, restaurant_name, rlocation, ROUND(AVG(rrating),2) AS avg_rating
FROM restaurant
GROUP BY restaurant_id, restaurant_name, rlocation
ORDER BY avg_rating DESC
LIMIT 1;
```

restaurant_id	restaurant_name	rlocation	avg_rating
101	hydrabadi spice	aundh	4.50

```sql
-- 3. List all orders from order_detail that were delivered in under 30 minutes.

SELECT *, TIMESTAMPDIFF(MINUTE, order_time, delivered_time) AS delivery_time_in_minutes
FROM order_detail
WHERE TIMESTAMPDIFF(MINUTE, order_time, delivered_time) < 30;
```

order_id	customer_id	restaurant_id	employee_id	order_status	order_time	delivered_time	delivery_time_in_minutes
7719	1009	102	210125	confirmed	2019-10-28 21:36:39	2019-10-28 21:45:25	8
7720	1010	103	210113	confirmed	2019-10-12 12:36:31	2019-10-12 13:01:19	24
7721	1011	104	210121	confirmed	2019-10-14 12:36:31	2019-10-14 13:01:19	24
7724	1004	106	210118	confirmed	2019-10-25 20:01:14	2019-10-25 20:29:44	28
7727	1014	102	210115	confirmed	2019-10-28 21:36:36	2019-10-28 21:45:25	8
7731	1007	104	210113	confirmed	2019-10-21 20:30:45	2019-10-21 20:55:25	24
7734	1005	103	210118	confirmed	2019-10-12 12:36:31	2019-10-12 13:01:19	24
7737	7734	102	210124	confirmed	2019-10-23 13:35:35	2019-10-23 14:02:11	26
7744	1010	103	210118	confirmed	2019-10-23 12:36:31	2019-10-23 13:01:10	24
7746	1007	104	210117	confirmed	2019-10-21 20:30:45	2019-10-21 20:55:25	24
7749	1017	101	210111	confirmed	2019-09-14 10:23:13	2019-09-14 10:51:29	28

```sql
-- 4. Calculate the total revenue generated by each food item. Display food_id, food_name
-- and Total revenue ordered by total revenue in descending order.

SELECT f.food_id, f.food_name, SUM(o.quantity * f.price_per_unit) AS total_revenue
FROM order_food o
JOIN foods f ON f.food_id = o.food_id
GROUP BY f.food_id, f.food_name
ORDER BY total_revenue DESC;
```

food_id	food_name	total_revenue
9999416	veg biryani	2520.00
9999419	mix paratha	1620.00
9999417	mix veg	900.00
9999422	veg roll	900.00
9999411	allo paratha	880.00
9999420	cold drink	665.00
9999414	veg thali	660.00
9999412	veg meal	480.00
9999418	veg pulav	380.00
9999421	paneer roll	360.00
9999415	chicken thali	300.00
9999413	chapati	240.00

```sql
-- 5. Find the second highest revenue-generating restaurant.

SELECT r.restaurant_id, r.restaurant_name, sum(f.price_per_unit * o.quantity) AS total_revenue
FROM order_food o
JOIN foods f ON f.food_id = o.food_id
JOIN restaurant r ON r.restaurant_id = o.restaurant_id
GROUP BY r.restaurant_id, r.restaurant_name
ORDER BY total_revenue DESC
LIMIT 1 OFFSET 1;
```

restaurant_id	restaurant_name	total_revenue
101	hydrabadi spice	1635.00

```sql
-- 6. Find the 5 most popular food items based on the quantity sold.

SELECT f.food_id, f.food_name, SUM(quantity) AS Quantity_Sold
FROM order_food o
JOIN foods f ON f.food_id = o.food_id
GROUP BY f.food_id
ORDER BY Quantity_Sold DESC
LIMIT 5;
```

food_id	food_name	Quantity_Sold
9999420	cold drink	19
9999413	chapati	12
9999411	allo paratha	11
9999416	veg biryani	9
9999419	mix paratha	9

```sql
-- 7. List the top 3 Zomato employees with the highest average delivery ratings.

SELECT employee_id, (employee_avg_rating) AS average_delivery_rating
FROM zomato_employee
ORDER BY average_delivery_rating DESC
LIMIT 3;
```

employee_id	average_delivery_rating
210123	4.7
210118	4.6
210113	4.5

```sql
-- 8. Determine the month with the highest number of total orders placed.

SELECT YEAR(order_time) AS order_year, MONTH(order_time) AS order_month, COUNT(order_id) AS total_orders
FROM order_detail
GROUP BY order_year, order_month
ORDER BY total_orders DESC
LIMIT 1;
```

order_year	order_month	total_orders
2019	10	38

```sql
-- 9. Calculate the average order amount for each customer. Order by average order amount in descending order.

SELECT customer_id, ROUND(AVG(total_amount), 2) AS average_order_amount
FROM (
        SELECT odf.order_id, odf.customer_id,
        SUM(odf.quantity * f.price_per_unit) AS total_amount
        FROM order_food odf
        JOIN foods f ON odf.food_id = f.food_id
        GROUP BY odf.order_id, odf.customer_id
) AS order_totals
GROUP BY customer_id
ORDER BY average_order_amount DESC;
```

customer_id	average_order_amount
1011	395.00
1009	320.00
1017	315.00
1008	301.25
1013	280.00
1001	255.50
1006	247.50
1014	246.67
1004	225.00
1016	195.00
1010	181.67
1007	175.00
1002	150.00
1003	130.00
1005	127.50

```sql
-- 10. Identify the most frequent customer for each restaurant.

WITH customer_orders AS (
    SELECT restaurant_id, customer_id, COUNT(order_id) AS total_orders,
           RANK() OVER (PARTITION BY restaurant_id ORDER BY COUNT(order_id) DESC) AS rank_order
    FROM order_detail
    GROUP BY restaurant_id, customer_id
)
SELECT restaurant_id, customer_id, total_orders
FROM customer_orders
WHERE rank_order = 1;
```

	restaurant_id	customer_id	total_orders
▶	101	1008	4
	102	1002	2
	103	1003	3
	103	1010	3
	104	1007	2
	105	1001	3
	106	1006	2
	106	1001	2
	107	1001	3

```sql
-- 11. Calculate the total number of orders placed on weekends.

SELECT COUNT(order_id)
FROM order_detail
WHERE WEEKDAY(order_time) IN (5,6);
```

	COUNT(order_id)
▶	13

```sql
-- 12. Calculate the average delivery time (in minutes) for orders placed on weekdays versus weekends.

SELECT (CASE
            WHEN WEEKDAY(order_time) IN (5,6) THEN 'WEEKEND'
            ELSE 'WEEKDAYS'
        END) AS day_type,
        AVG(TIMESTAMPDIFF(MINUTE, order_time, delivered_time)) AS average_delivery_time
FROM order_detail
GROUP BY day_type;
```

	day_type	average_delivery_time
▶	WEEKDAYS	37.3000
	WEEKEND	37.0769

```sql
-- 13. List the top 5 most expensive food items. Display food_id, food_name, and
-- price_per_unit for all food items with the highest 5 prices.

WITH rankedFood AS (
    SELECT food_id, food_name, price_per_unit,
                RANK() OVER (ORDER BY price_per_unit DESC) AS ranking
    FROM foods
)
SELECT food_id, food_name, price_per_unit
FROM rankedFood
WHERE ranking <=5;
```

	food_id	food_name	price_per_unit
▶	9999416	veg biryani	280.00
	9999414	veg thali	220.00
	9999418	veg pulav	190.00
	9999417	mix veg	180.00
	9999419	mix paratha	180.00
	9999421	paneer roll	180.00
	9999422	veg roll	180.00

```sql
-- 14. Find the restaurant with the most diverse menu (i.e., the highest number of food items).

SELECT o.restaurant_id, r.restaurant_name, r.rlocation, r.rrating, COUNT(DISTINCT o.food_id) AS menu_count
FROM order_food o
JOIN restaurant r ON r.restaurant_id = o.restaurant_id
GROUP BY o.restaurant_id
ORDER BY menu_count DESC
LIMIT 1;
```

	restaurant_id	restaurant_name	rlocation	rrating	menu_count
▶	105	laa unico	mukund nagar	4.3	8

```sql
-- 15. Calculate total payment amount for each payment type.

SELECT pt.payment_type,
       COUNT(DISTINCT pt.transaction_id) AS total_transactions,
       SUM(o.quantity * f.price_per_unit) AS total_payment
FROM order_food o
JOIN foods f ON f.food_id = o.food_id
JOIN payment_table pt ON pt.order_id = o.order_id
GROUP BY pt.payment_type;
```

	payment_type	total_transactions	total_payment
▶	cod	15	3370.00
	internet banking	28	6535.00