SQL Task – 4

GROUP BY & Aggregates

1. Find the total sales amount for each restaurant

```
SELECT restaurant, SUM(quantity * price) as total_sales
FROM orders
GROUP BY restaurant;
```

2. Show the average price of food items per category.

```
SELECT category, AVG(price) as avg_price
FROM orders
GROUP BY category;
```

3. Count how many orders were placed for each food category

```
SELECT category, COUNT(*) as order_count
FROM orders
GROUP BY category;
```

4. Find the maximum quantity ordered for each food item.

```
SELECT food_item, MAX(quantity) as max_quantity
FROM orders
GROUP BY food_item;
```

5. Show the total amount spent by each customer.

```
SELECT customer_name, SUM(quantity * price) as total_spent
FROM orders
GROUP BY customer_name;
```

HAVING

6. Find restaurants with total sales greater than 800.

```
SELECT restaurant, SUM(quantity * price) as total_sales
```

```
FROM orders
GROUP BY restaurant
HAVING SUM(quantity * price) > 800;
```

7. Show customers who spent more than 500 in total.

```
SELECT customer_name, SUM(quantity * price) as total_spent
FROM orders
GROUP BY customer_name
HAVING SUM(quantity * price) > 500;
```

8. Find food categories where the average item price > 250.

```
SELECT category, AVG(price) as avg_price
FROM orders
GROUP BY category
HAVING AVG(price) > 250;
```

9. Get restaurants with more than 2 orders.

```
SELECT restaurant, COUNT(*) as order_count
FROM orders
GROUP BY restaurant
HAVING COUNT(*) > 2;
```

10. Show categories where the total quantity ordered > 4.

```
SELECT category, SUM(quantity) as total_quantity
FROM orders
GROUP BY category
HAVING SUM(quantity) > 4;
```

ORDER BY

11. List all orders by price descending.

```
SELECT * FROM orders
ORDER BY price DESC;
```

12. Show customers ordered by their total spending.

```
SELECT customer_name, SUM(quantity * price) as total_spent
FROM orders
```

```
GROUP BY customer_name
ORDER BY total_spent DESC;
```

13. Display restaurants ordered by total quantity sold.

```
SELECT restaurant, SUM(quantity) as total_quantity_sold
FROM orders
GROUP BY restaurant
ORDER BY total_quantity_sold DESC;
```

14. Show the top 3 highest-priced food items.

```
SELECT food_item, price
FROM orders
ORDER BY price DESC
LIMIT 3;
```

15. List orders sorted by order_date (latest first).

```
SELECT * FROM orders
ORDER BY order_date DESC;
```

LIMIT & OFFSET

16. Show the first 5 orders from the table.

```
SELECT * FROM orders
LIMIT 5;
```

17. Get the top 3 most expensive orders.

```
SELECT * FROM orders
ORDER BY price DESC
LIMIT 3;
```

18. Skip the first 3 rows and show the next 4 orders.

```
SELECT * FROM orders
LIMIT 4 OFFSET 3;
```

19. Find the second highest-priced food item using LIMIT + OFFSET

```
SELECT food_item, price FROM orders
```

```
ORDER BY price DESC
LIMIT 1 OFFSET 1;
```

20. Show the top 2 customers by total spending.

```
SELECT customer_name, SUM(quantity * price) as total_spent
FROM orders
GROUP BY customer_name
ORDER BY total_spent DESC
LIMIT 2;
```

Combined Questions (on FoodOrders table)

1. Find the total spending by each customer, but show only those who spent more than ₹500, ordered by spending descending.

```
SELECT customer_name, SUM(quantity * price) as total_spent
FROM orders
GROUP BY customer_name
HAVING SUM(quantity * price) > 500
ORDER BY total_spent DESC;
```

2. Show the top 3 food categories with the highest total sales amount.

```
SELECT category, SUM(quantity * price) as total_sales
FROM orders
GROUP BY category
ORDER BY total_sales DESC
LIMIT 3;
```

3. Find the restaurants where average price > 250, ordered by average price descending.

```
SELECT restaurant, AVG(price) as avg_price
FROM orders
GROUP BY restaurant
HAVING AVG(price) > 250
ORDER BY avg_price DESC;
```

4. Show the top 2 customers who placed the highest quantity of food items (using GROUP BY + ORDER BY + LIMIT).

```
SELECT customer_name, SUM(quantity) as total_quantity
FROM orders
```

```
GROUP BY customer_name
ORDER BY total_quantity DESC
LIMIT 2;
```

5. Find the restaurant with the maximum total sales, but skip the top 1 and show the second highest restaurant (using LIMIT + OFFSET).

   ```
   SELECT restaurant, SUM(quantity * price) as total_sales
   FROM orders
   GROUP BY restaurant
   ORDER BY total_sales DESC
   LIMIT 1 OFFSET 1;
   ```

6. List the categories with more than 2 total orders, ordered by total quantity sold.

   ```
   SELECT category, COUNT(*) as order_count, SUM(quantity) as total_quantity
   FROM orders
   GROUP BY category
   HAVING COUNT(*) > 2
   ORDER BY total_quantity DESC;
   ```

7. Find the top 3 food items by total sales amount, grouped by food_item, ordered descending.

   ```
   SELECT food_item, SUM(quantity * price) as total_sales
   FROM orders
   GROUP BY food_item
   ORDER BY total_sales DESC
   LIMIT 3;
   ```

8. Show customers who ordered more than 2 different categories, ordered by the count of categories descending.

   ```
   SELECT customer_name, COUNT(DISTINCT category) as category_count
   FROM orders
   GROUP BY customer_name
   HAVING COUNT(DISTINCT category) > 2
   ORDER BY category_count DESC;
   ```