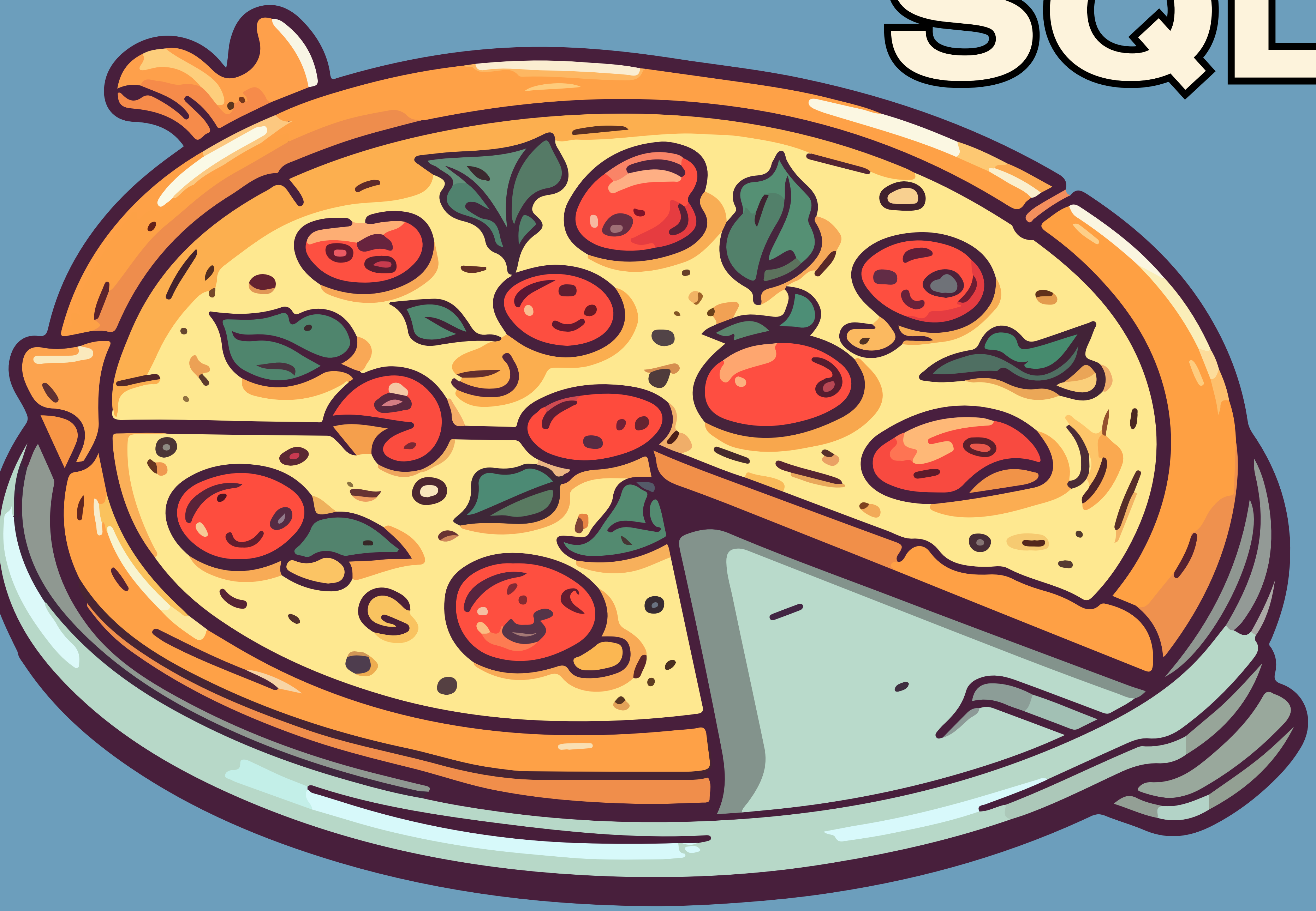
PIZZA SALES SQL PROJECT



HELLO

I’m Krishna singh. In this SQL project, I imported the dataset into MySQL Workbench and executed various SQL queries to analyze the data and find insights based on the given questions. The project mainly focused on using SQL concepts like joins, aggregate functions, and subqueries to extract meaningful information and present clear results from the dataset.

Data Overview

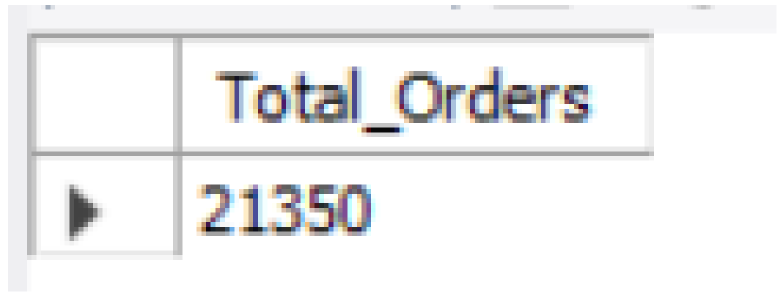
THE DATASET USED IN THIS PROJECT IS BASED ON PIZZA SALES DATA. IT CONTAINS DETAILED INFORMATION ABOUT CUSTOMER ORDERS, INCLUDING ORDER IDS, PIZZA TYPES, SIZES, QUANTITIES, PRICES, AND ORDER TIMESTAMPS. THE DATA IS ORGANIZED ACROSS MULTIPLE TABLES SUCH AS ORDERS, ORDER_DETAILS, PIZZA, AND PIZZA_TYPES. THIS DATASET PROVIDES A COMPLETE VIEW OF SALES TRANSACTIONS, ALLOWING ANALYSIS OF ORDER VOLUMES, REVENUE GENERATION, POPULAR PIZZA TYPES, AND TIME-BASED ORDERING PATTERNS.”

Retrieve the total number of orders placed.

Input:

```
select count(order_id) as Total_Orders from orders;
```

Output:



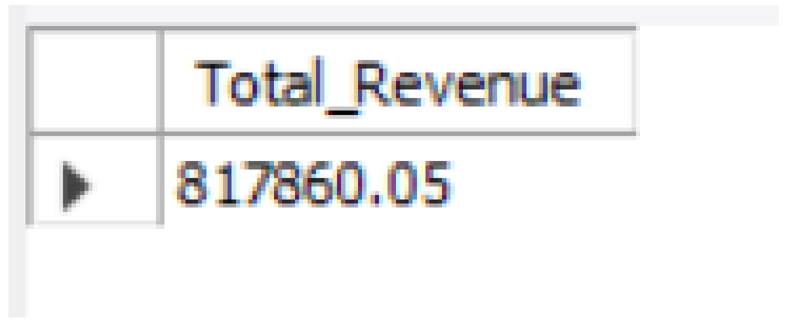
	Total_Orders
▶	21350

Calculate the total revenue generated from pizza sales.

Input:

```
select round(sum(order_details.quantity * pizza.price),2) as Total_Revenue
from  order_details join pizza
on pizza.pizza_id = order_details.pizza_id;
```

Output:



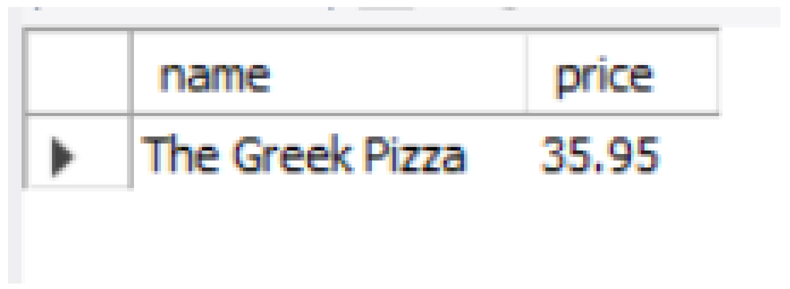
```
Total_Revenue
817860.05
```

Identify the highest-priced pizza.

Input:

```
SELECT pizza_types.name, pizza.price
FROM pizza_types JOIN pizza
ON pizza_types.pizza_type_id = pizza.pizza_type_id
ORDER BY pizza.price desc limit 1;
```

Output:



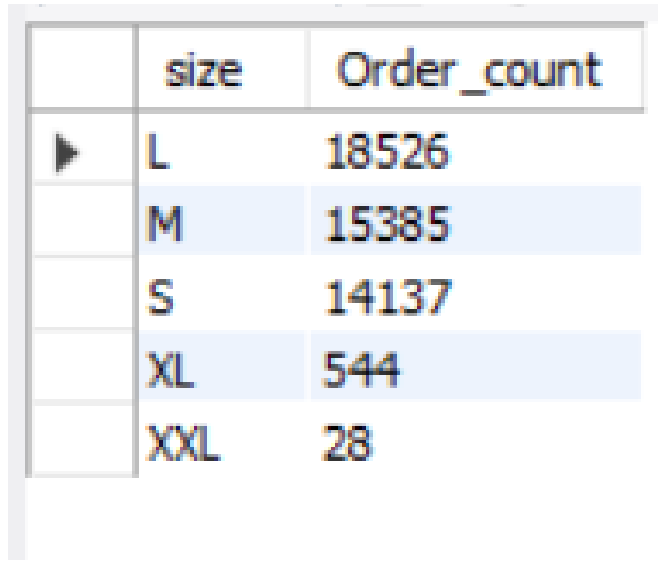
	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

Input:

```
select size, count(order_details_id) as Order_count
from order_details join pizza
on order_details.pizza_id = pizza.pizza_id
group by size
order by Order_count desc;
```

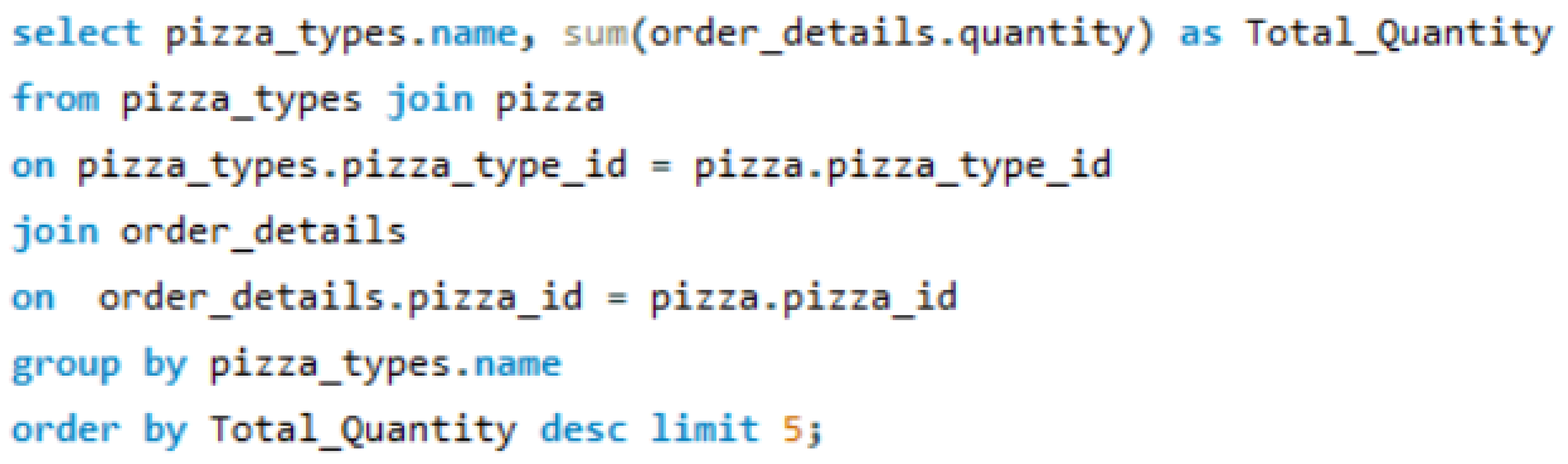
Output:



size	Order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

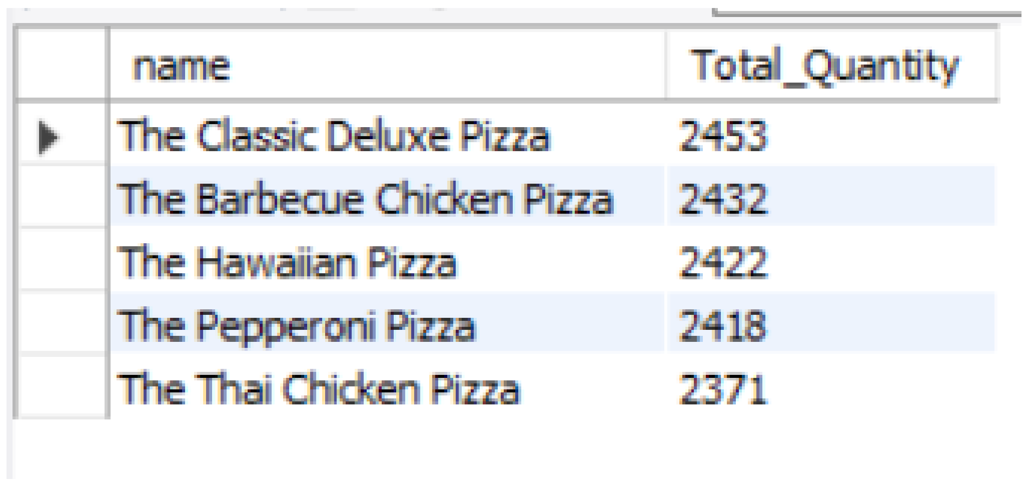
List the top 5 most ordered pizza types along with their quantities.

Input:



```
select pizza_types.name, sum(order_details.quantity) as Total_Quantity
from pizza_types join pizza
on pizza_types.pizza_type_id = pizza.pizza_type_id
join order_details
on  order_details.pizza_id = pizza.pizza_id
group by pizza_types.name
order by Total_Quantity desc limit 5;
```

Output:



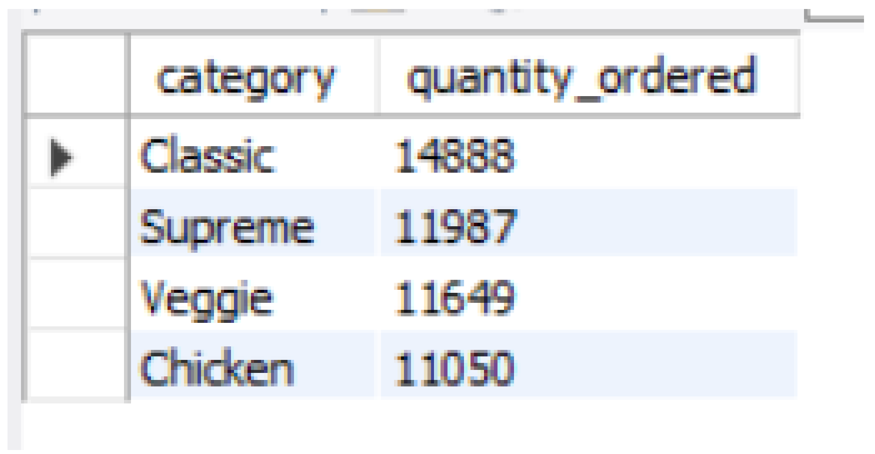
name	Total_Quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Join the necessary tables to find the total quantity of each pizza category ordered.

Input:

```
select category, sum(quantity) as quantity_ordered
from pizza_types join pizza
on pizza_types.pizza_type_id = pizza.pizza_type_id
join order_details
on order_details.pizza_id = pizza.pizza_id
group by category order by quantity_ordered desc;
```

Output:



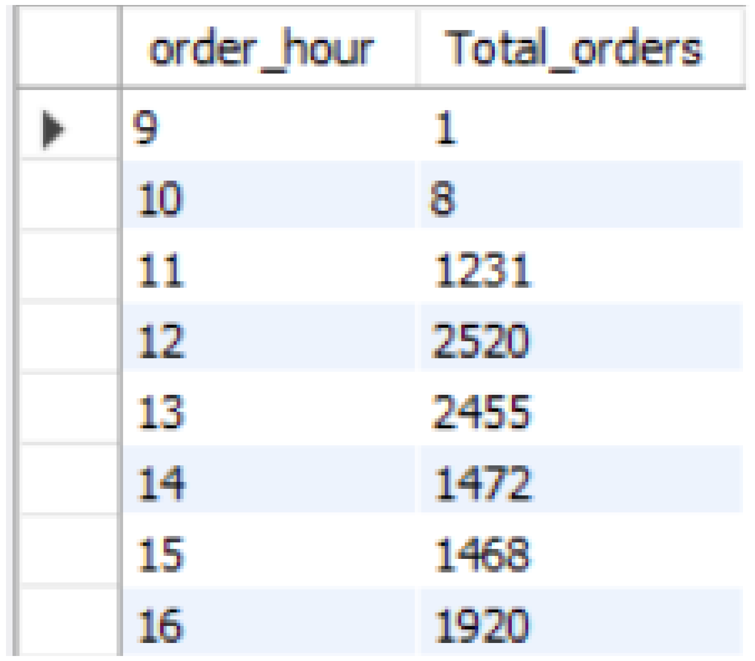
category	quantity_ordered
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

Input:

```
select hour(order_time) as order_hour ,count(order_id)  as Total_orders
from orders
group by order_hour
order by order_hour;
```

Output:



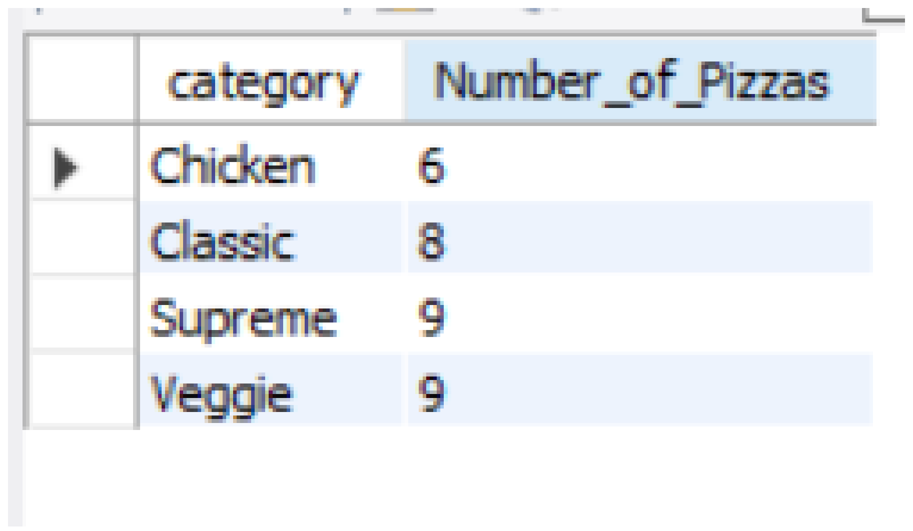
order_hour	Total_orders
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920

Join relevant tables to find the category-wise distribution of pizzas.

Input:

```
select category , count(name) as Number_of_Pizzas
from pizza_types
group by category ;
```

Output:



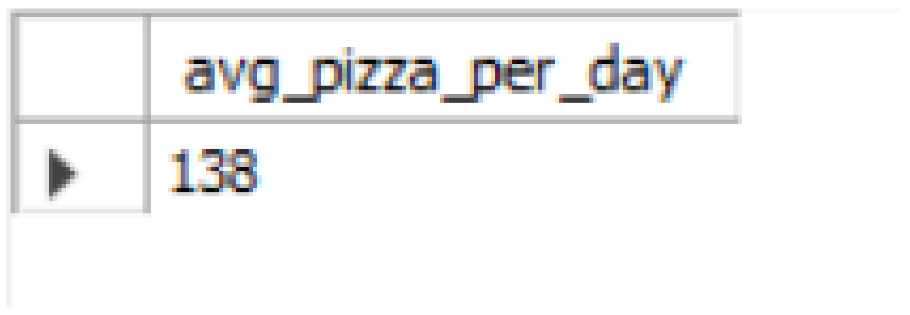
category	Number_of_Pizzas
Chicken	6
Classic	8
Supreme	9
Veggie	9

Group the orders by date and calculate the average number of pizzas ordered per day.

Input:

```
select round(avg(daily.Total_orders)) as avg_pizza_per_day
from(
select order_date, sum(quantity) as Total_orders
from order_details join orders
on order_details.order_id = orders.order_id
group by order_date) as daily;
```

Output:



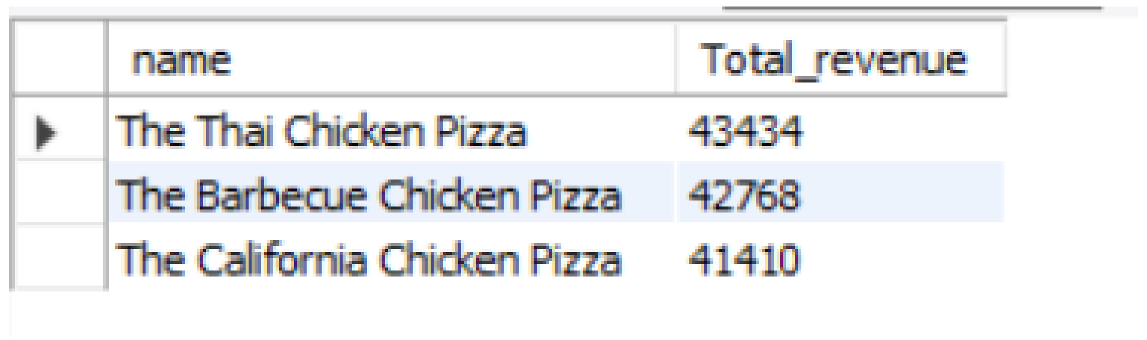
```
avg_pizza_per_day
138
```

Determine the top 3 most ordered pizza types based on revenue.

<u>Input:</u>

```
select pizza_types.name, round(sum((order_details.quantity)*(pizza.price))) as Total_revenue
from pizza_types join pizza
on pizza_types.pizza_type_id = pizza.pizza_type_id
join order_details
on order_details.pizza_id = pizza.pizza_id
group by pizza_types.name
order by Total_revenue desc limit 3;
```

<u>Output:</u>



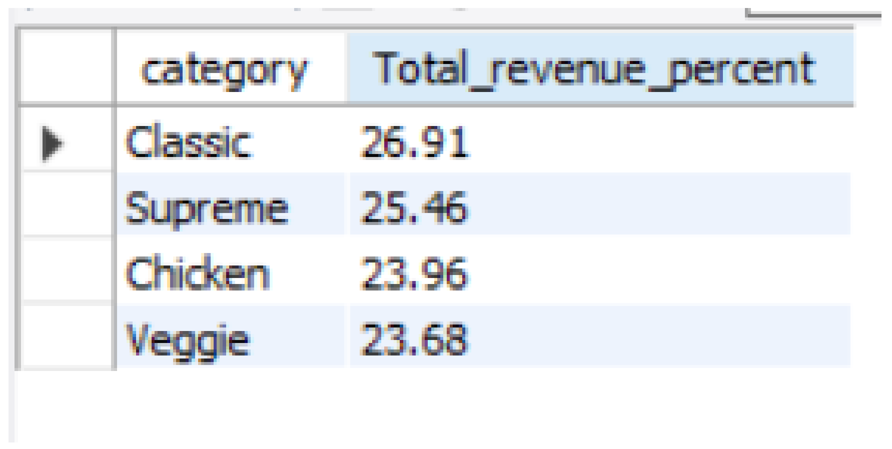
name	Total_revenue
The Thai Chicken Pizza	43434
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41410

Calculate the percentage contribution of each pizza type to total revenue.

Input:

```
select pizza_types.category,
round((sum((order_details.quantity)*(pizza.price))/(select sum(order_details.quantity * pizza.price)
from  order_details join pizza
on pizza.pizza_id = order_details.pizza_id))*100,2) as Total_revenue_percent
from pizza_types join pizza
on pizza_types.pizza_type_id = pizza.pizza_type_id
join order_details
on order_details.pizza_id = pizza.pizza_id
group by pizza_types.category
order by Total_revenue_percent desc;
```

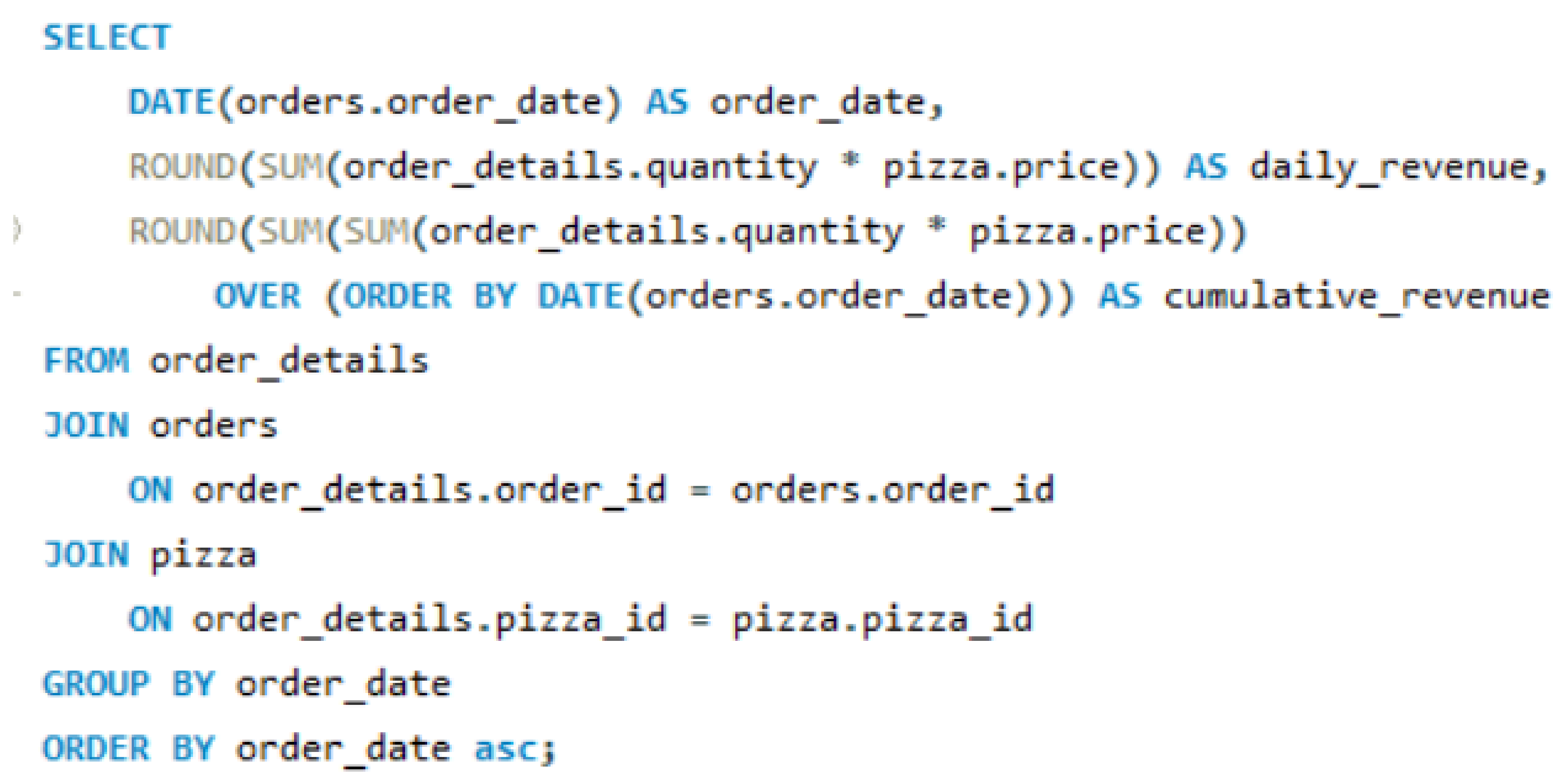
Output:



category	Total_revenue_percent
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

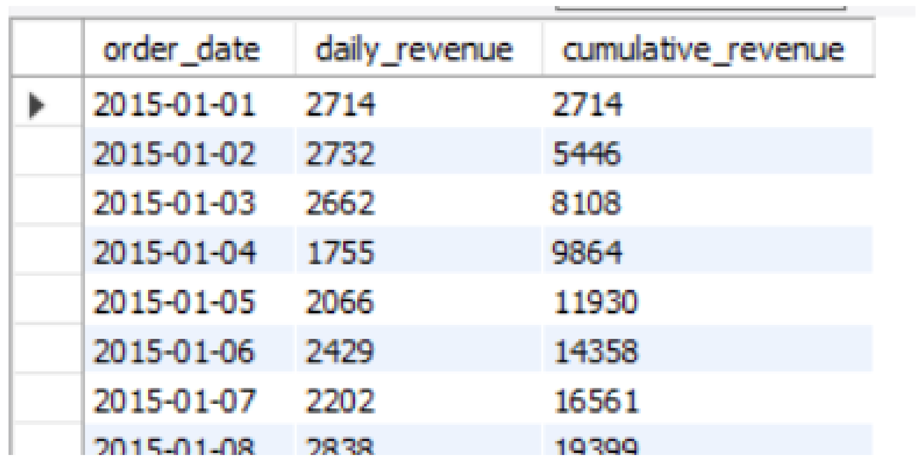
Analyze the cumulative revenue generated over time.

Input:



```
SELECT
    DATE(orders.order_date) AS order_date,
    ROUND(SUM(order_details.quantity * pizza.price)) AS daily_revenue,
    ROUND(SUM(SUM(order_details.quantity * pizza.price))
        OVER (ORDER BY DATE(orders.order_date))) AS cumulative_revenue
FROM order_details
JOIN orders
    ON order_details.order_id = orders.order_id
JOIN pizza
    ON order_details.pizza_id = pizza.pizza_id
GROUP BY order_date
ORDER BY order_date asc;
```

Output:



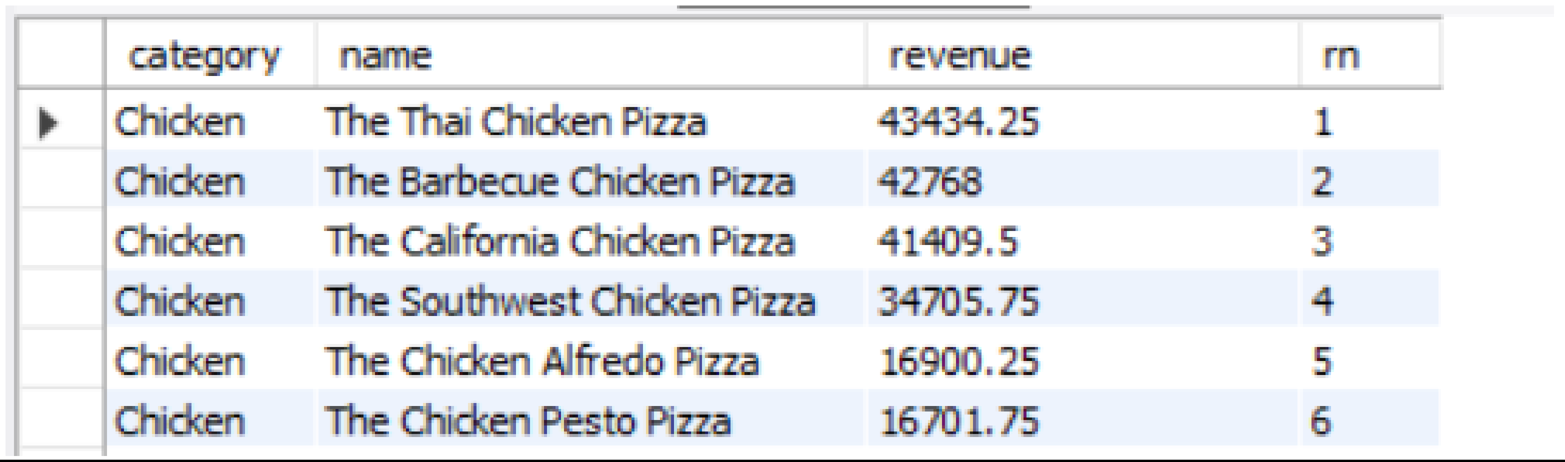
order_date	daily_revenue	cumulative_revenue
2015-01-01	2714	2714
2015-01-02	2732	5446
2015-01-03	2662	8108
2015-01-04	1755	9864
2015-01-05	2066	11930
2015-01-06	2429	14358
2015-01-07	2202	16561
2015-01-08	2838	19399

Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Input:

```
SELECT category, name, revenue,
    RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rn
FROM (
    SELECT category, name,
    SUM(order_details.quantity * pizza.price) AS revenue
    FROM pizza_types JOIN pizza
    ON pizza_types.pizza_type_id = pizza.pizza_type_id
    JOIN order_details
    ON order_details.pizza_id = pizza.pizza_id
    GROUP BY category, name ) AS a
ORDER BY category, rn;
```

Output:



category	name	revenue	rn
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Chicken	The Southwest Chicken Pizza	34705.75	4
Chicken	The Chicken Alfredo Pizza	16900.25	5
Chicken	The Chicken Pesto Pizza	16701.75	6

conclusion

The Pizza Sales SQL Project helped in understanding how SQL can be used to analyze and draw insights from real-world data. By using different SQL queries like joins, aggregate functions, and subqueries, I was able to explore important aspects such as total orders, total revenue, most popular pizzas, and order patterns based on time. This analysis provided a clear view of overall sales performance and customer preferences. The project also improved my practical SQL skills and gave me hands-on experience in data handling, report generation, and business analysis using structured data. Overall, it showed how data-driven insights can help businesses make smarter and more efficient decisions.

THANK YOU