Pizza Sales Analytics using SQL

This self-initiated project uses structured SQL queries to extract insights from a fictional pizza sales database. It demonstrates practical SQL skills for data analysis, joins, aggregations, and window functions. In this project I wrote structured queries to find important business insights like total orders, revenue, most popular pizza types and sizes, and time-based order patterns.

Core SQL Commands Used

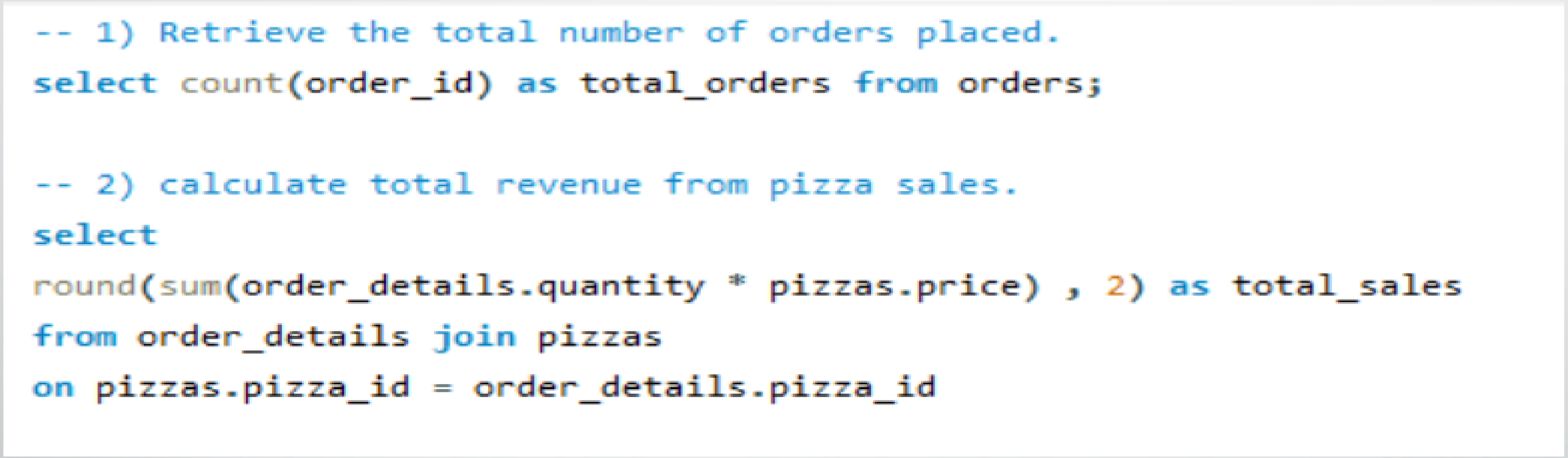
- Used **JOIN** to combine data from multiple related tables.
- Applied **SUM( )** and **COUNT( )** functions to calculate revenue and order details.
- Used **GROUP BY** to analyse data by category, size, date , and hour.
- Used **ORDER BY** and **LIMIT** to identify top- performing pizzas and sizes.
- Applied **ROUND()** to format numerical outputs.
- Used **subqueries** to calculate overall metrics like total revenue.
- Used **SUM( ) OVER(ORDER BY…)** to compute cumulative revenue over time.

Project Objectives

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.
- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

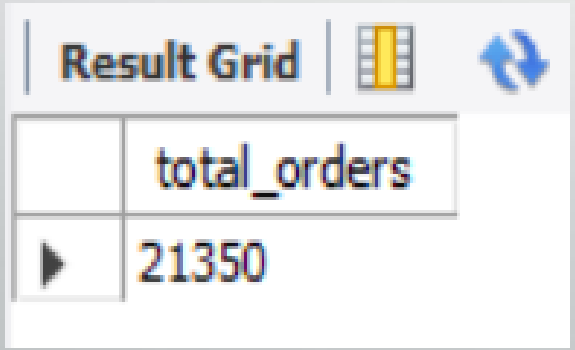
1. Retrieve the total number of orders placed.

2. Calculate the total revenue generated from pizza sales.

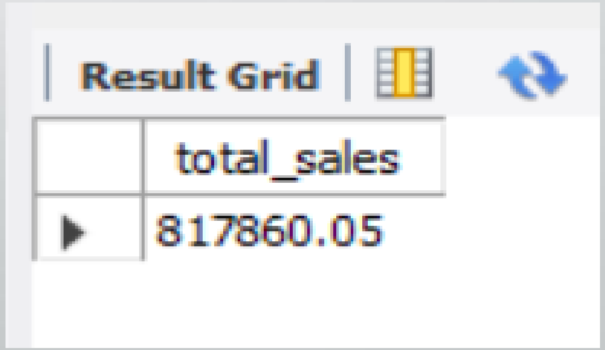


```
-- 1) Retrieve the total number of orders placed.
select count(order_id) as total_orders from orders;

-- 2) calculate total revenue from pizza sales.
select
round(sum(order_details.quantity * pizzas.price) , 2) as total_sales
from order_details join pizzas
on pizzas.pizza_id = order_details.pizza_id
```



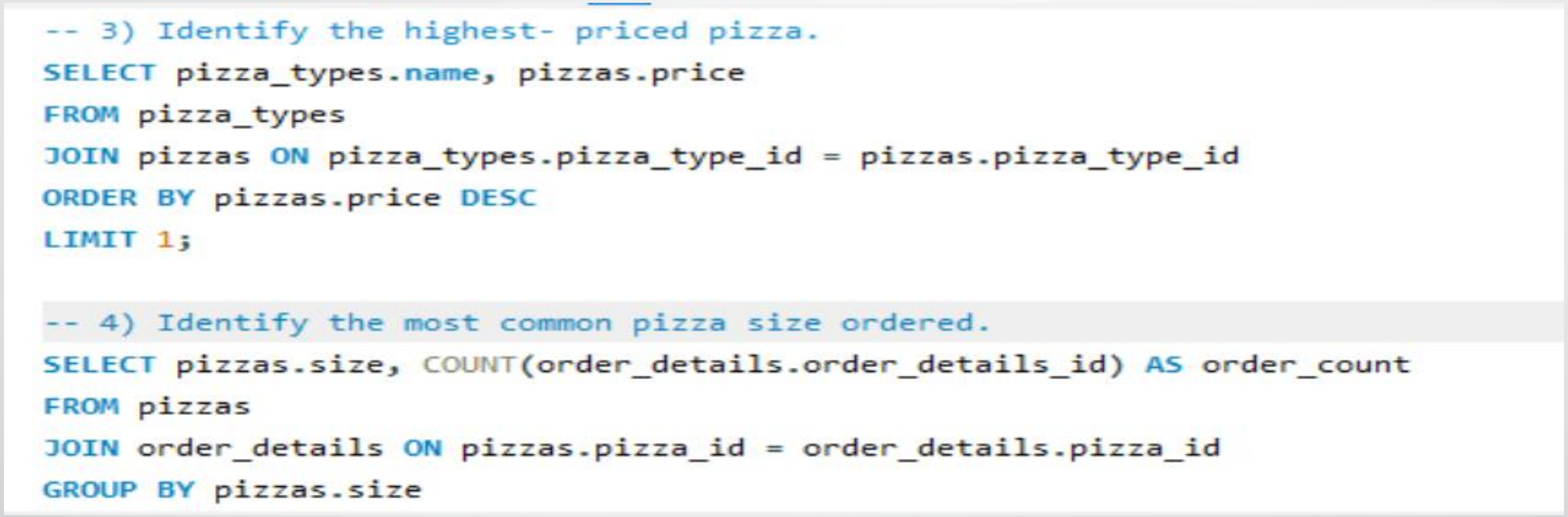
```
total_orders
21350
```



```
total_sales
817860.05
```

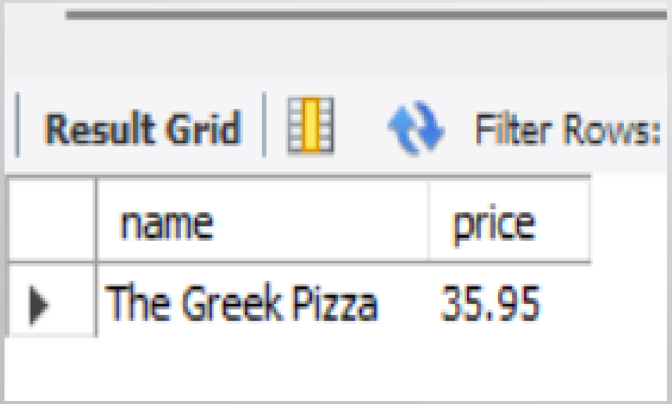
3. Identify the highest-priced pizza.

4. Identify the most common pizza size ordered.

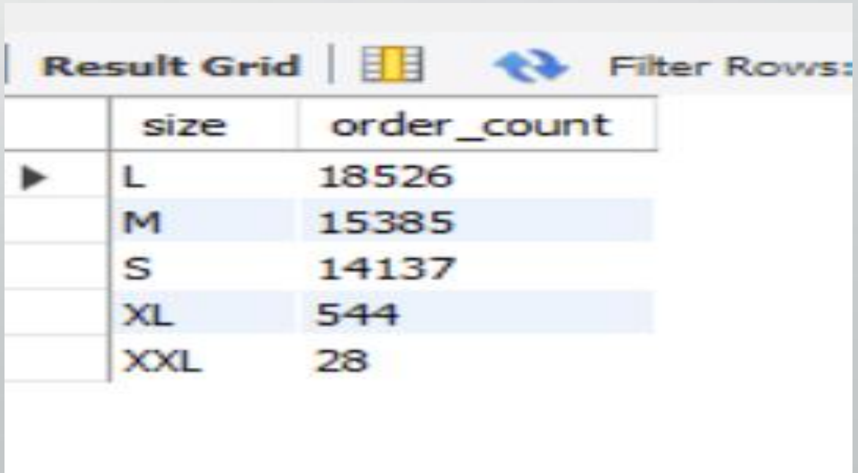


```
-- 3) Identify the highest- priced pizza.
SELECT pizza_types.name, pizzas.price
FROM pizza_types
JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;

-- 4) Identify the most common pizza size ordered.
SELECT pizzas.size, COUNT(order_details.order_details_id) AS order_count
FROM pizzas
JOIN order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
```

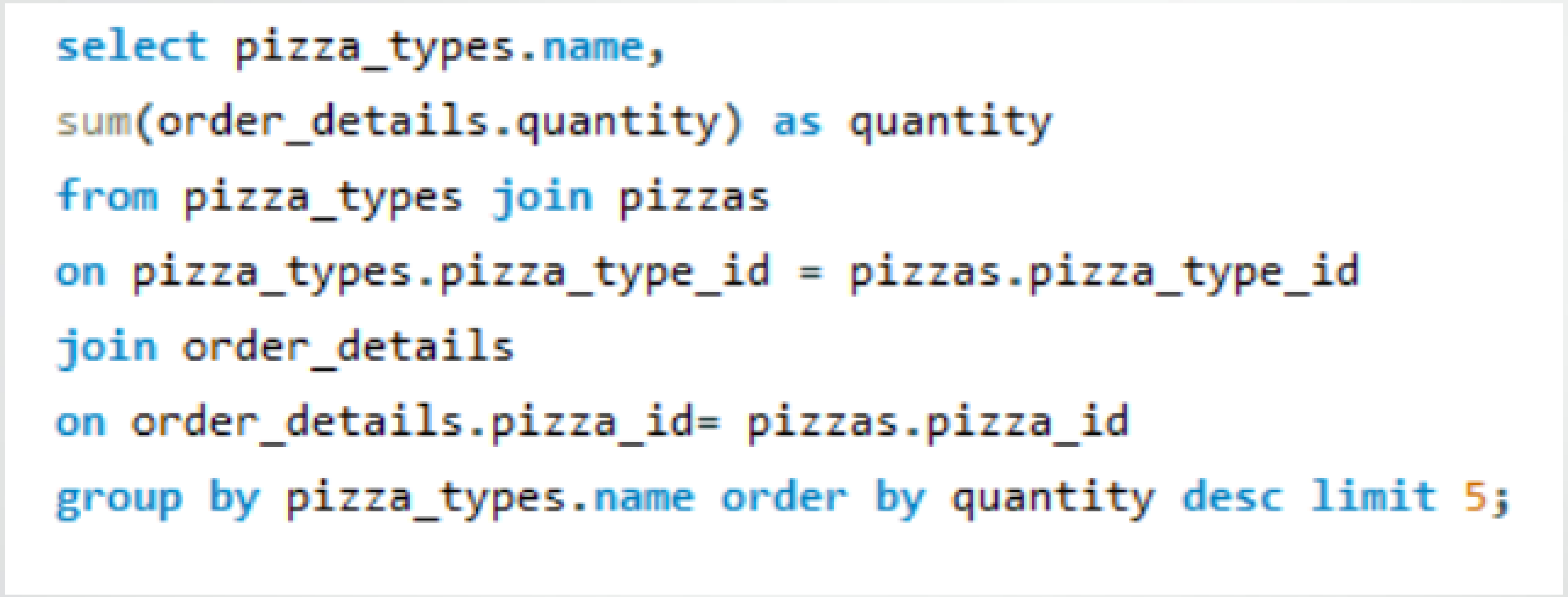


name	price
The Greek Pizza	35.95

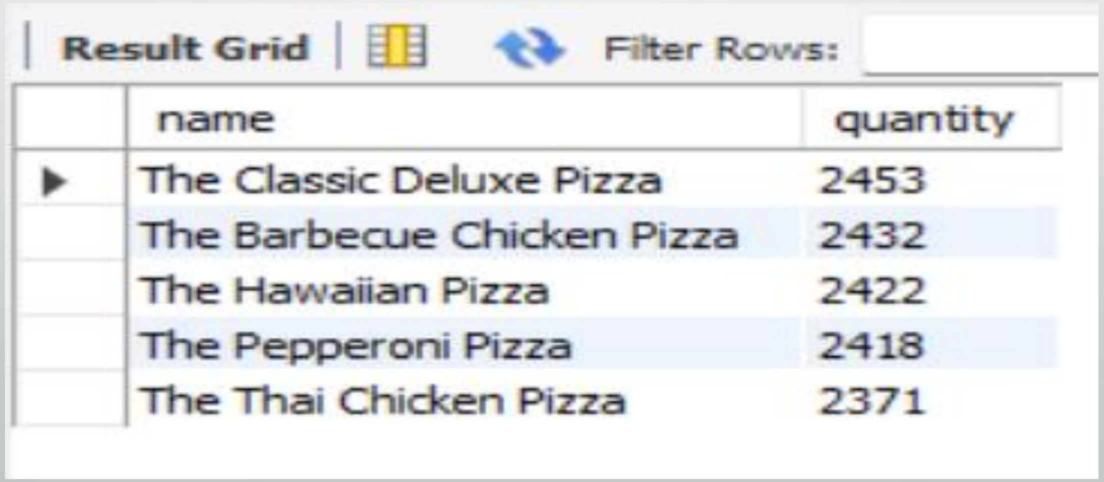


size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

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

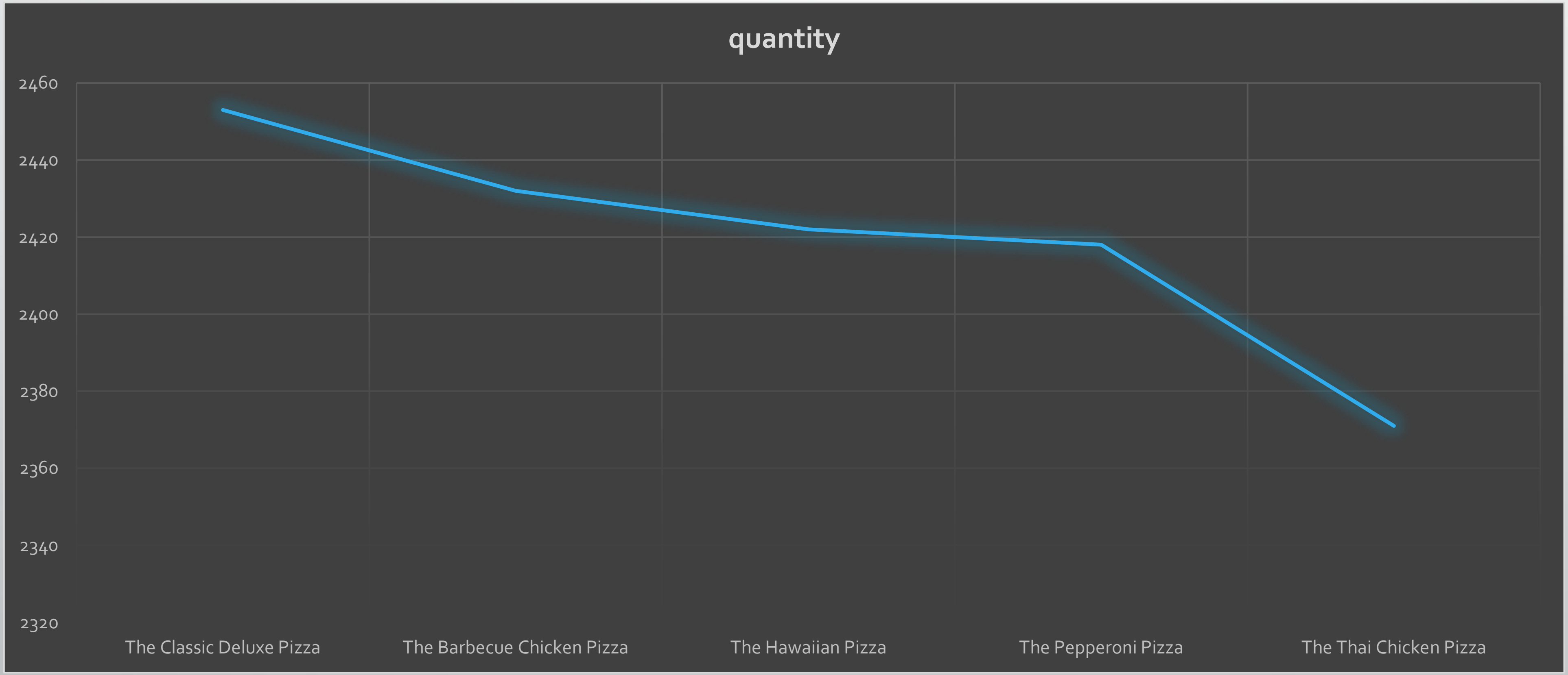


```
select pizza_types.name,
sum(order_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id= pizzas.pizza_id
group by pizza_types.name order by quantity desc limit 5;
```

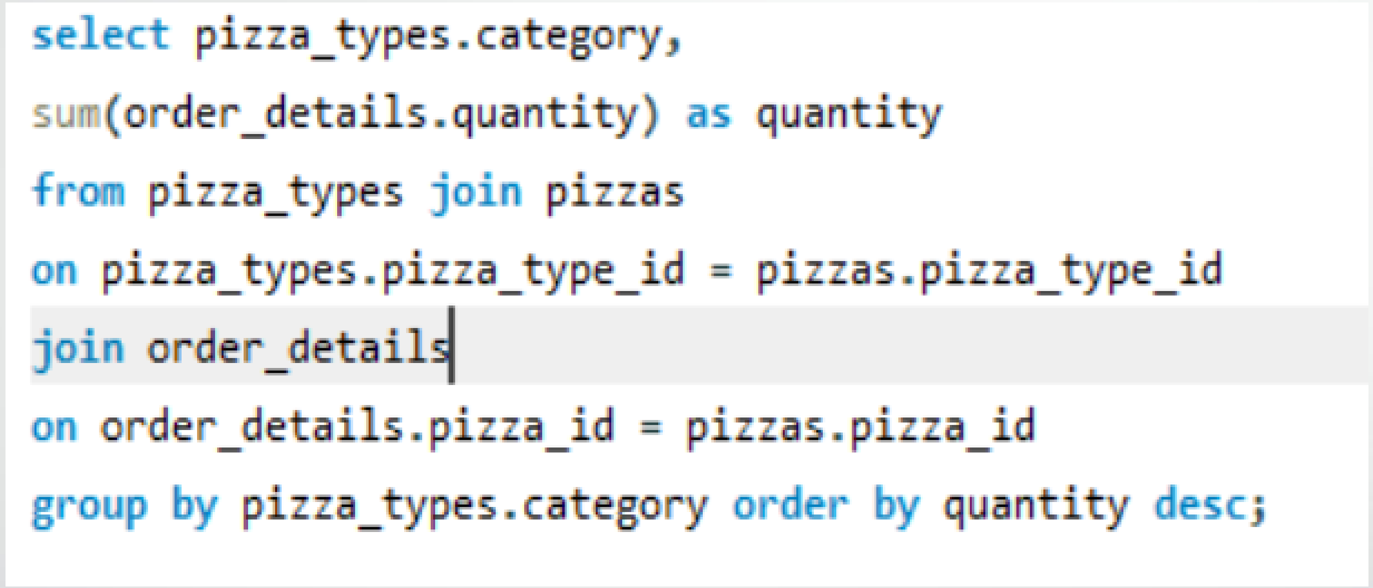


name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

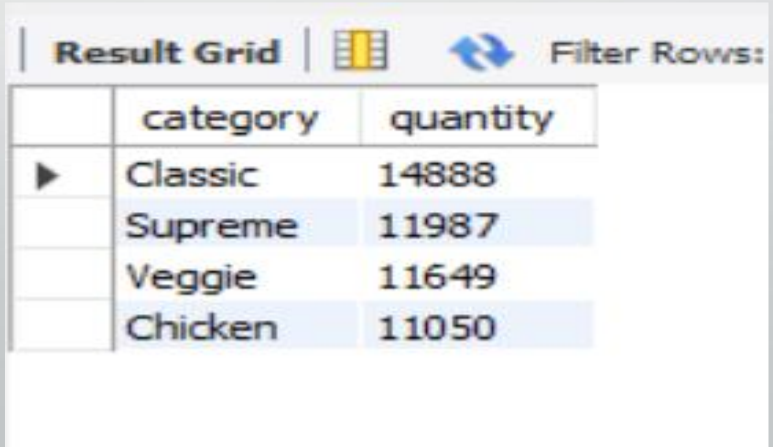
Line graph of the top 5 most ordered pizza types along with their quantities.



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

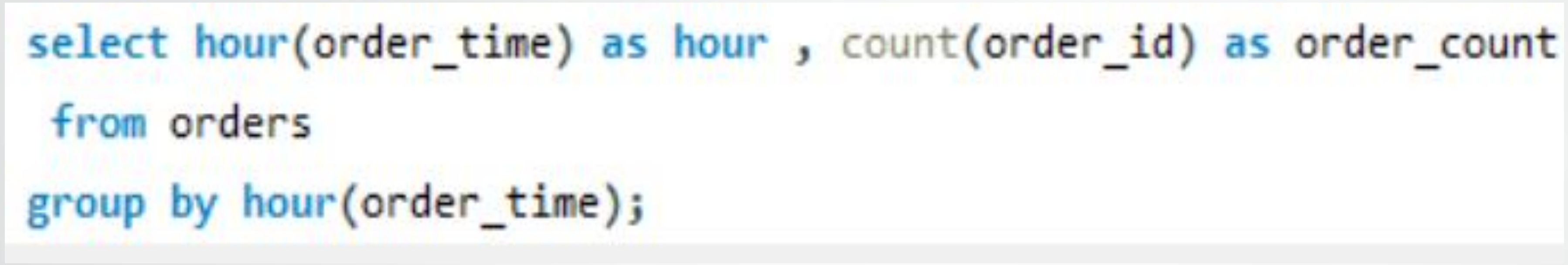


```
select pizza_types.category,
sum(order_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by quantity desc;
```

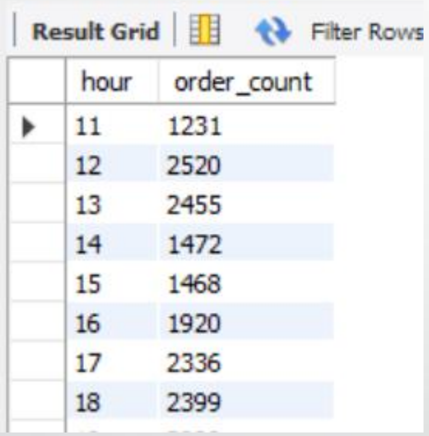


category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

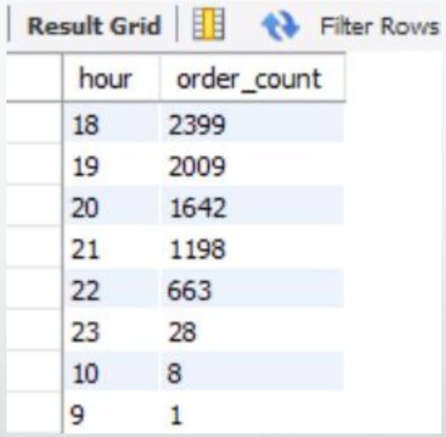
7. Determine the distribution of orders by hour of the day.
8. Join relevant tables to find the category-wise distribution of pizzas



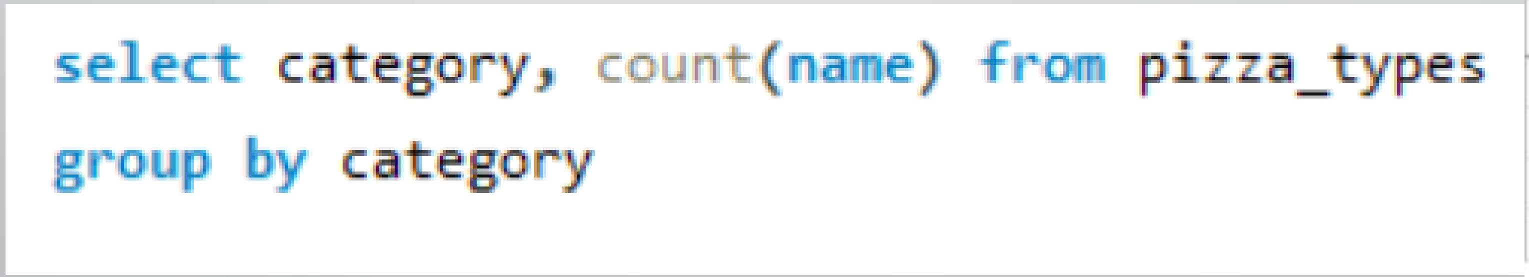
```sql
select hour(order_time) as hour , count(order_id) as order_count
 from orders
group by hour(order_time);
```



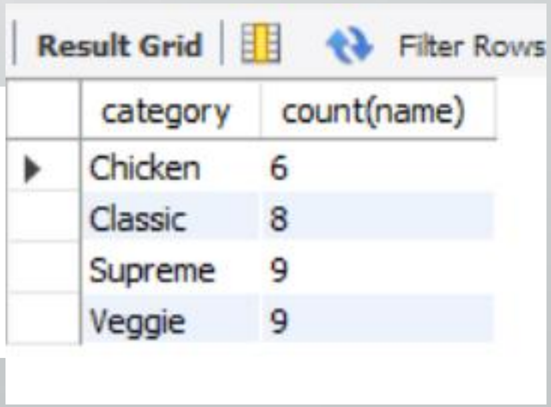
hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399



hour	order_count
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

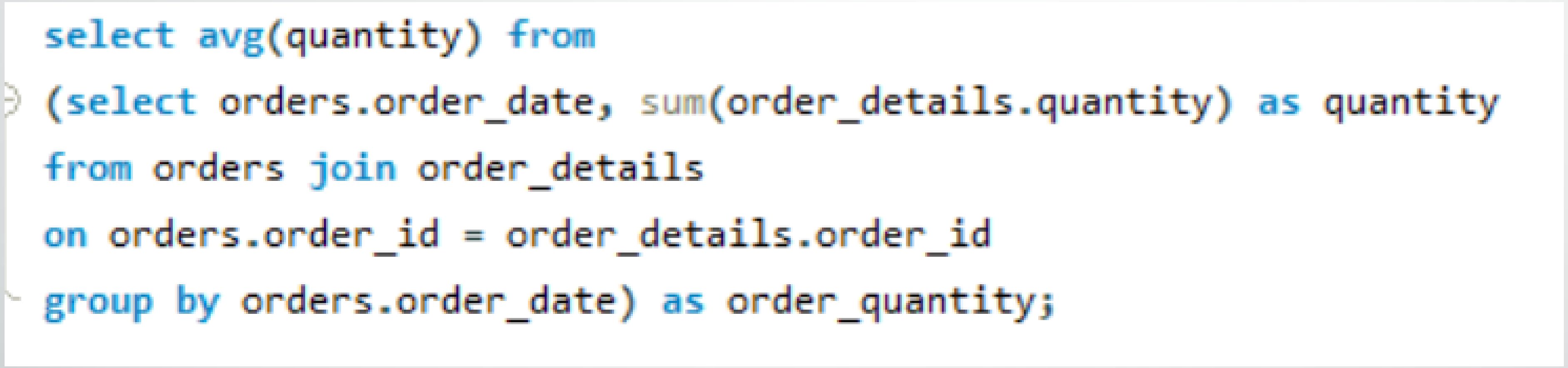


```sql
select category, count(name) from pizza_types
group by category
```

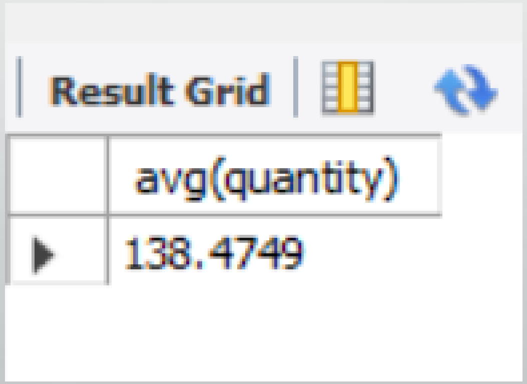


category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

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



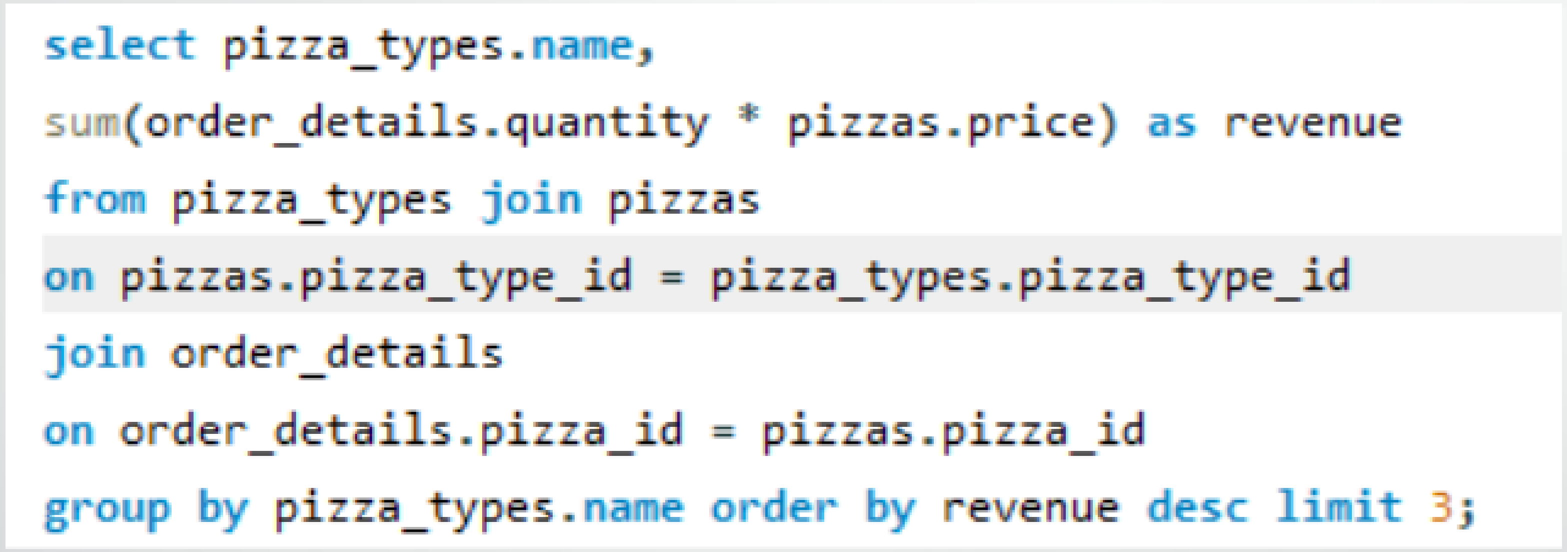
```
select avg(quantity) from
(select orders.order_date, sum(order_details.quantity) as quantity
from orders join order_details
on orders.order_id = order_details.order_id
group by orders.order_date) as order_quantity;
```



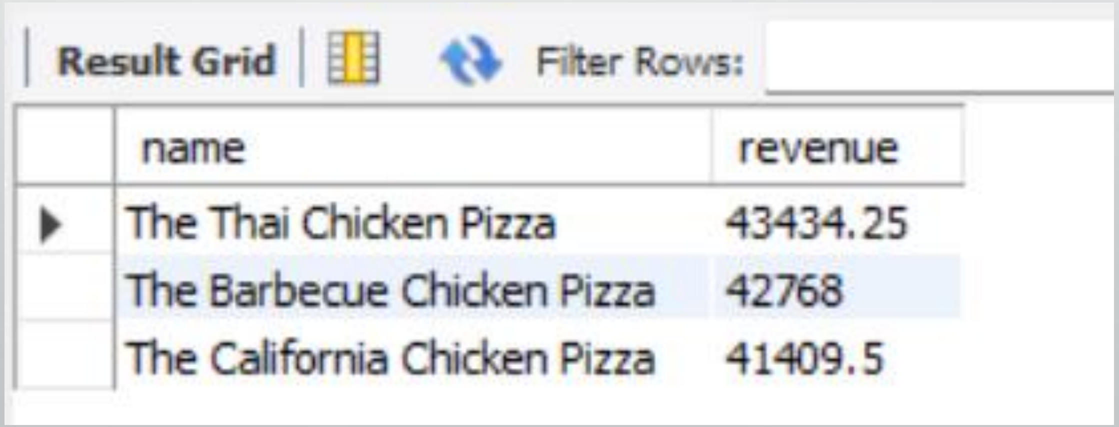
Result Grid

```
avg(quantity)
138.4749
```

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



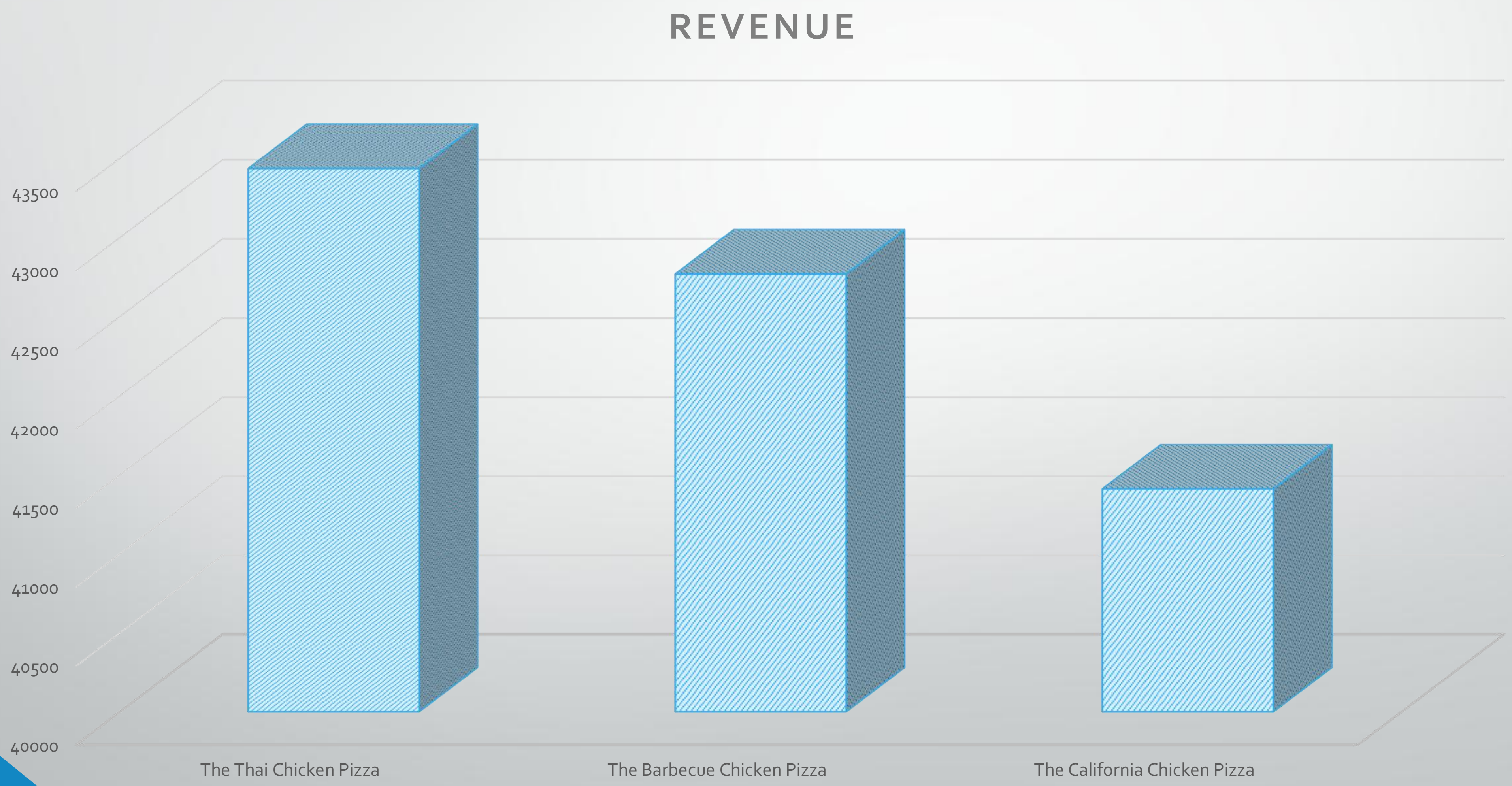
```
select pizza_types.name,
sum(order_details.quantity * pizzas.price) as revenue
from pizza_types join pizzas
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name order by revenue desc limit 3;
```



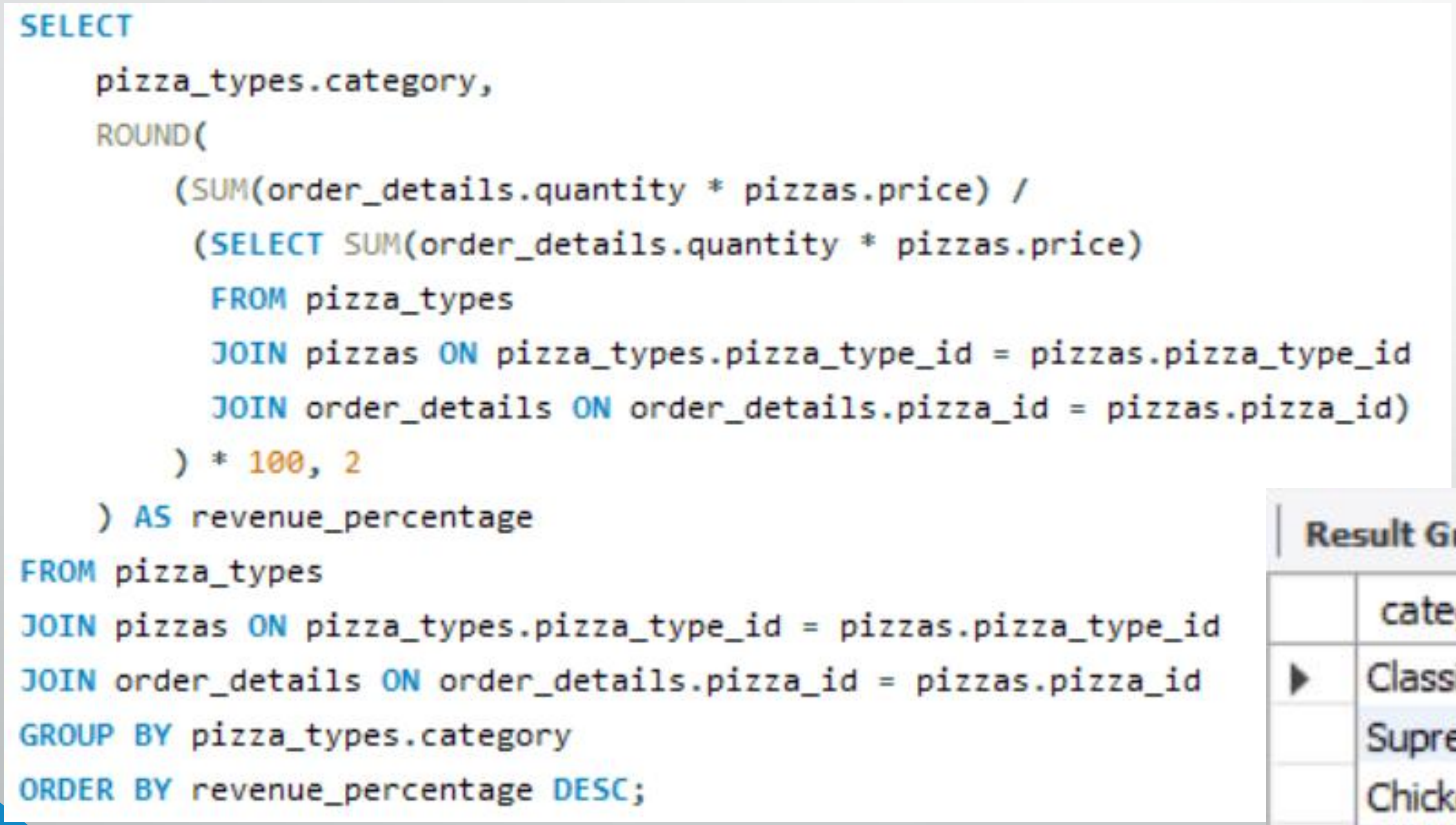
Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

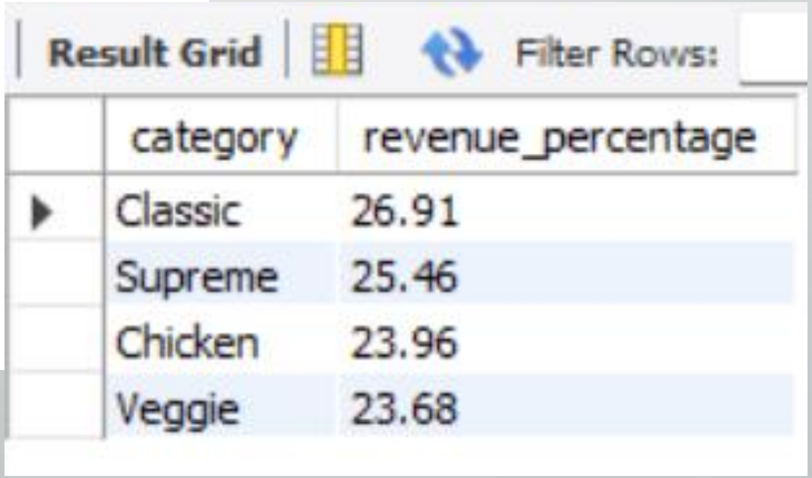
Top 3 Pizza Types by Revenue



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

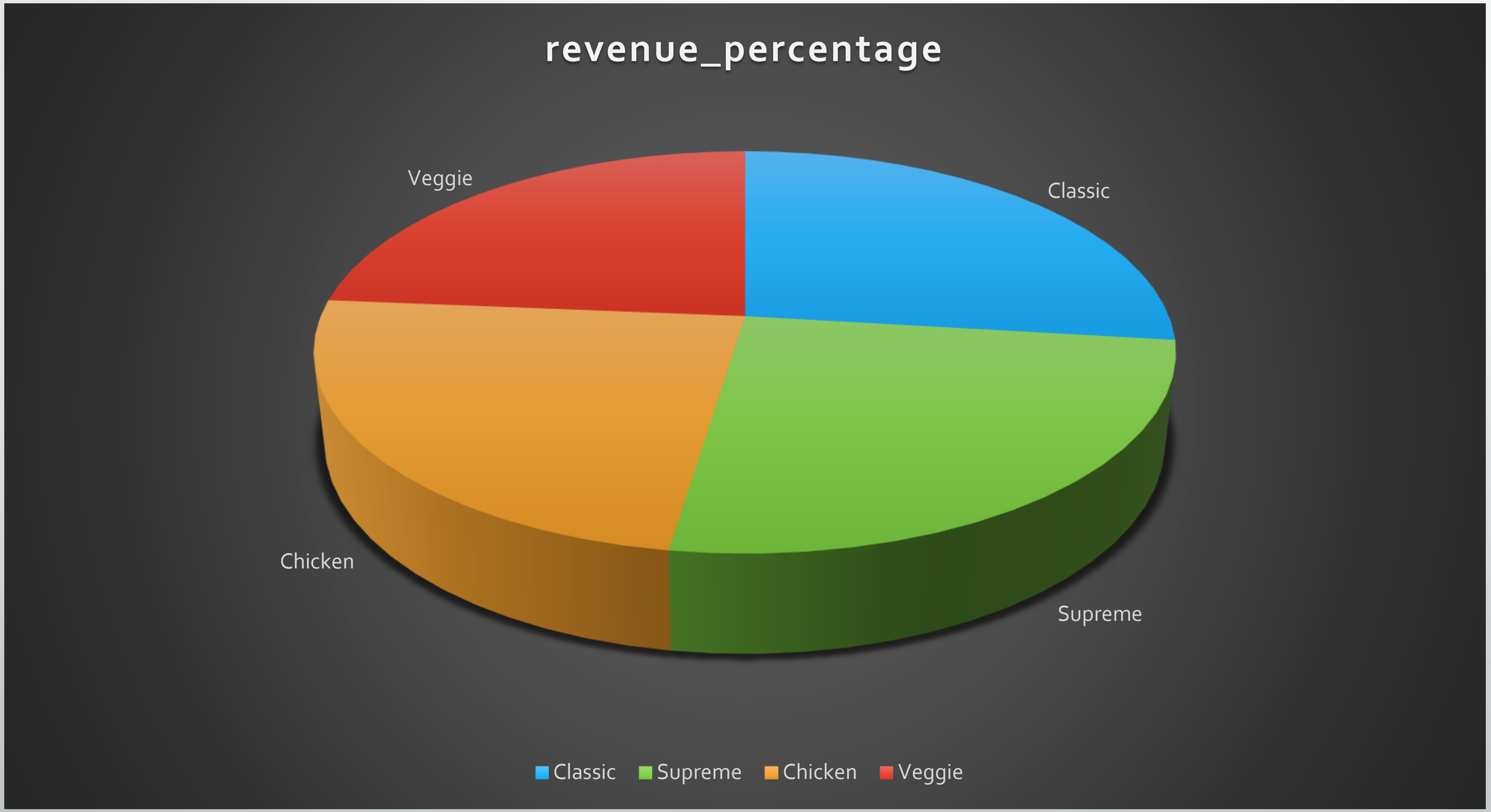


```
SELECT
    pizza_types.category,
    ROUND(
        (SUM(order_details.quantity * pizzas.price) /
         (SELECT SUM(order_details.quantity * pizzas.price)
          FROM pizza_types
          JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
          JOIN order_details ON order_details.pizza_id = pizzas.pizza_id)
        ) * 100, 2
    ) AS revenue_percentage
FROM pizza_types
JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue_percentage DESC;
```

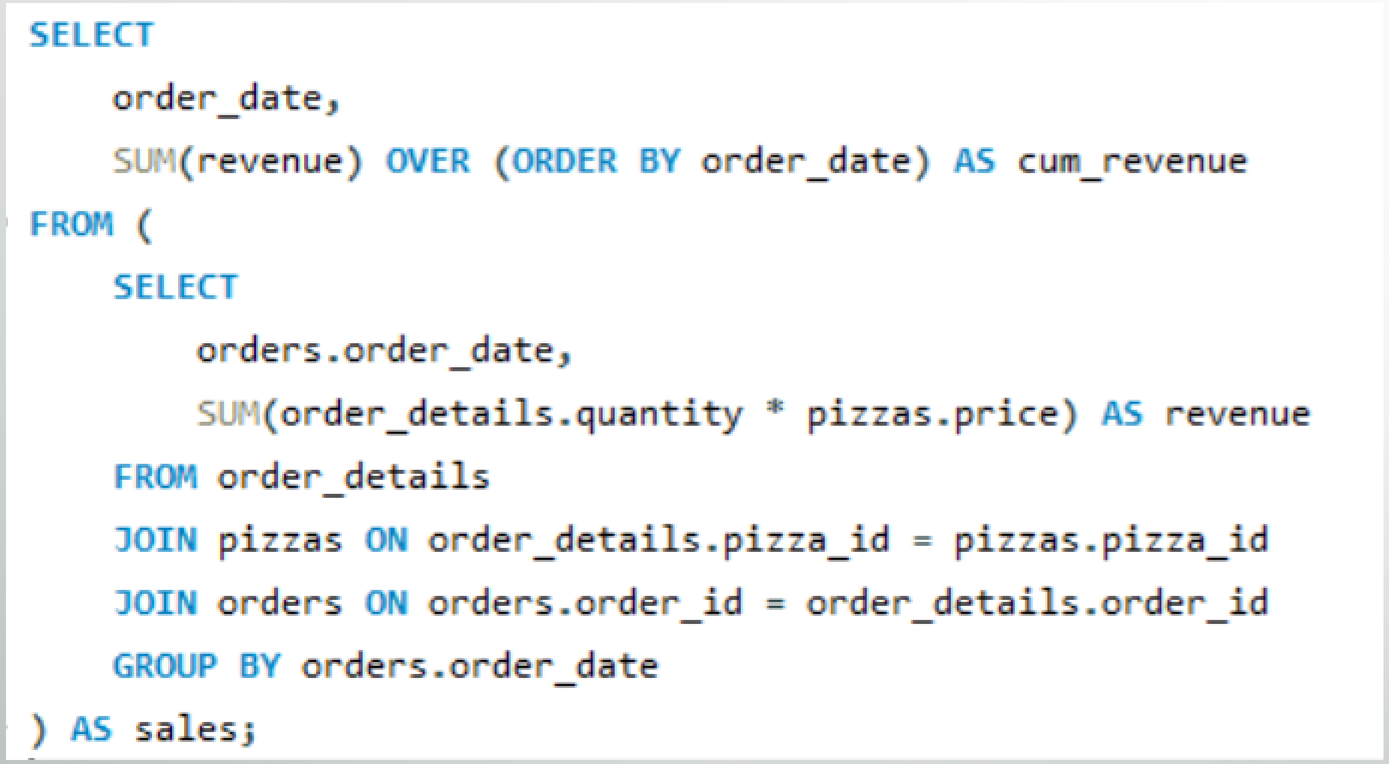


category	revenue_percentage
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

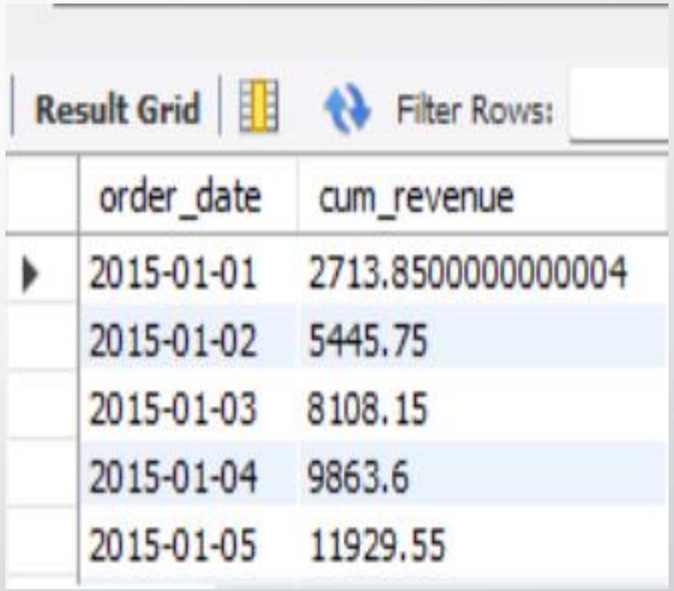
- **Pizza Types as categories**
- **Their share in total revenue expressed in percentage**



12. Analyze the cumulative revenue generated over time.



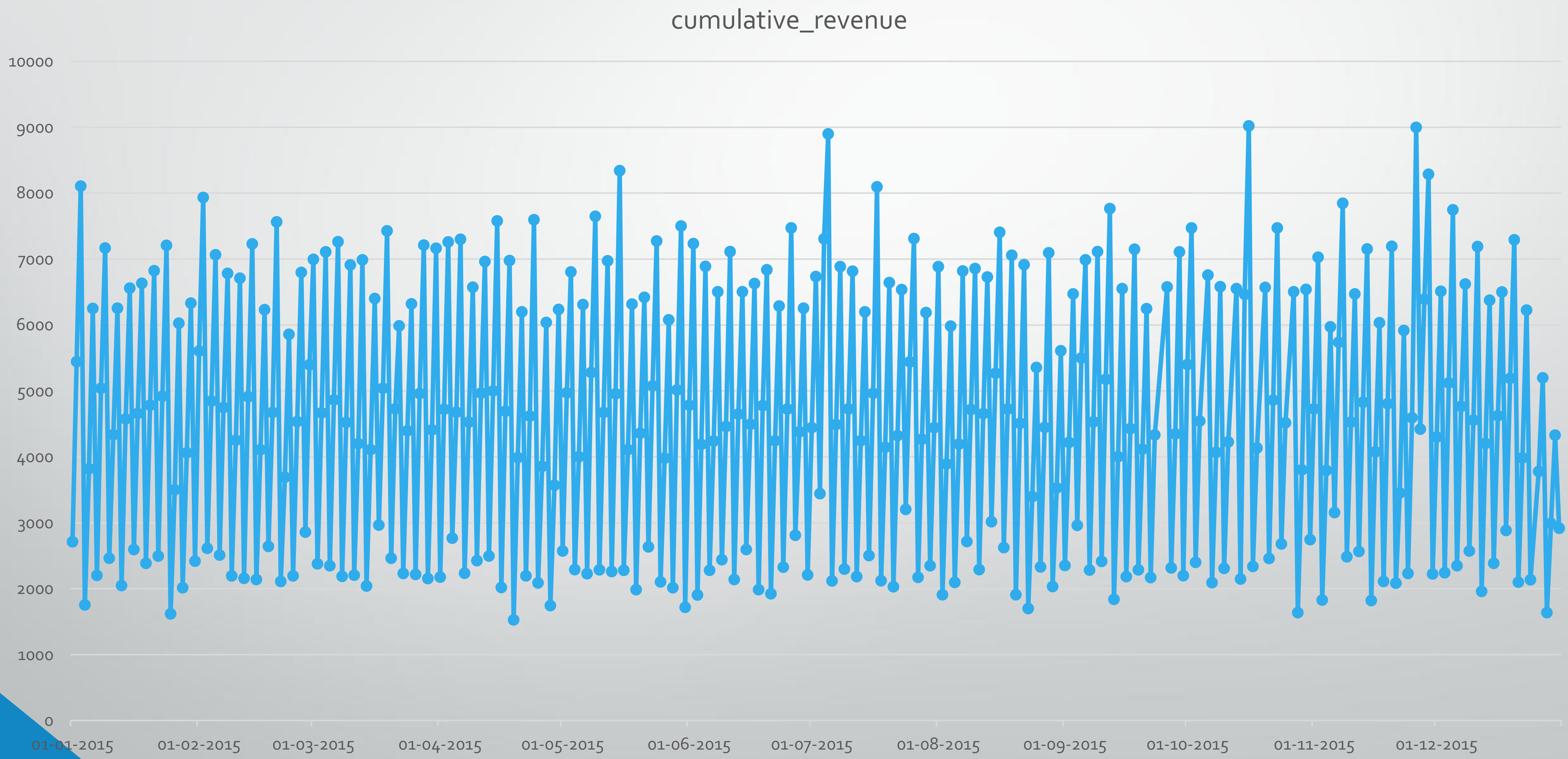
```
SELECT
    order_date,
    SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM (
    SELECT
        orders.order_date,
        SUM(order_details.quantity * pizzas.price) AS revenue
    FROM order_details
    JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN orders ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date
) AS sales;
```



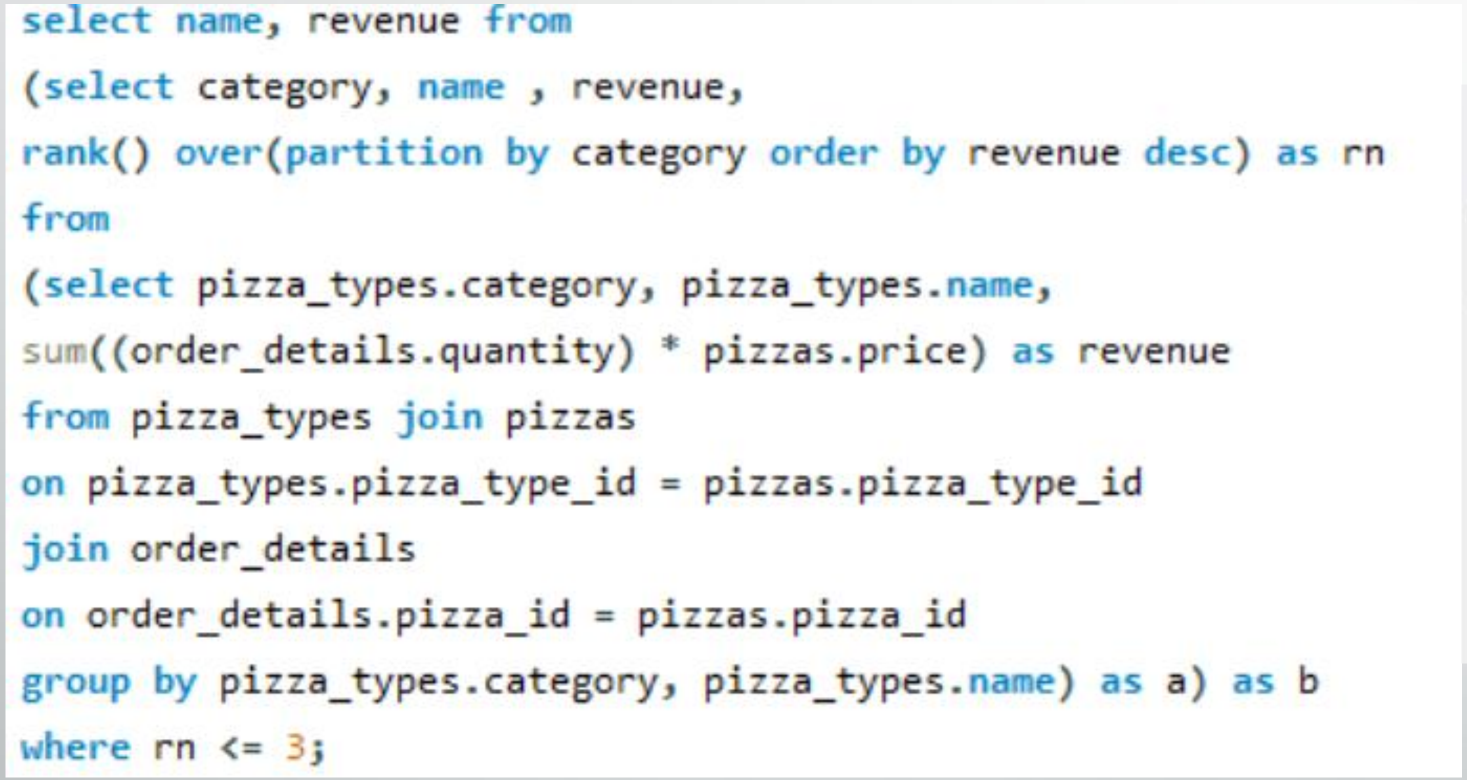
order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

Note: To avoid congestion, only 5 rows of result grid is shown here.

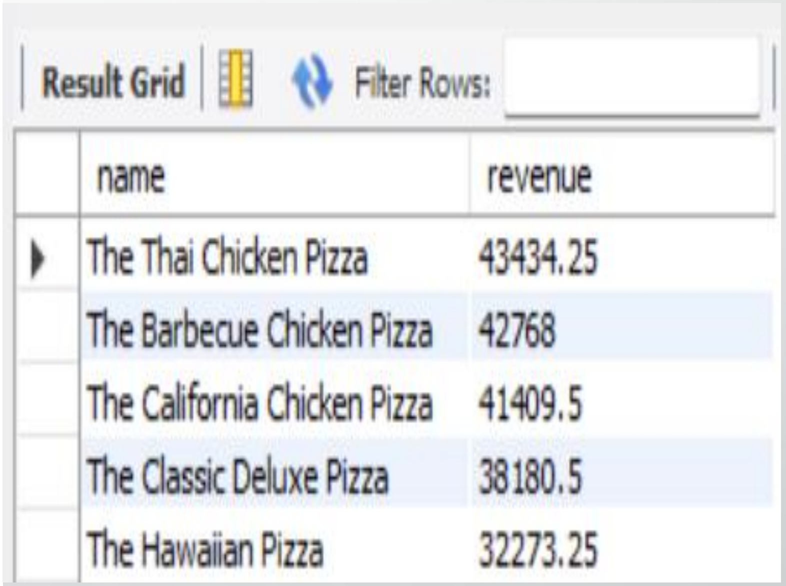
Cumulative Revenue Over Time



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



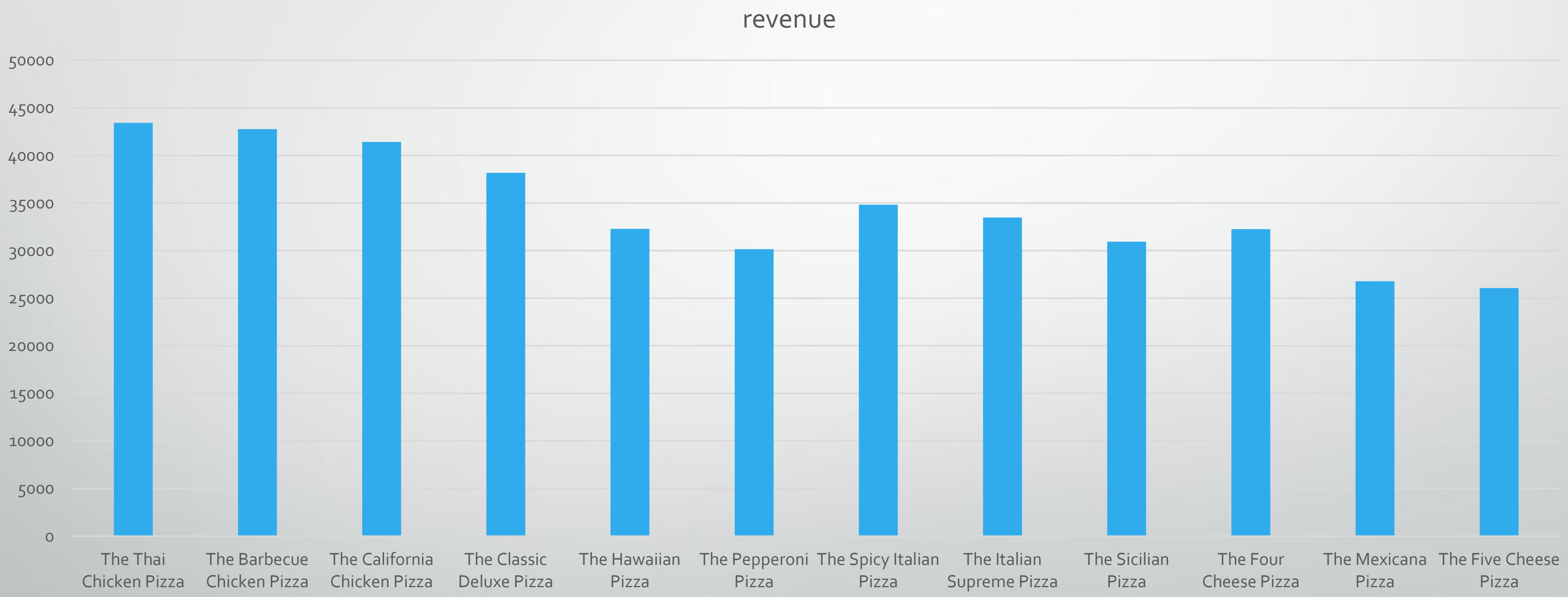
```sql
select name, revenue from
(select category, name , revenue,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category, pizza_types.name,
sum((order_details.quantity) * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn <= 3;
```



name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25

Note: To avoid congestion, only 5 rows of result grid is shown here.

Pizza Revenue Leaders Within Each Category



CONCLUSION:

- The project showcased end-to-end data exploration and insight generation using SQL. It strengthened my understanding of relational joins, grouping, aggregation, and subqueries. This analysis highlights how structured query language can turn raw business data into actionable intelligence.

Thank you