SQL Database Project

Utilizing SQL Queries for Database Problem Solving

[Utkarsh Srivastava]

[Project]

Project Overview

- Project Objectives
  - Demonstrate practical application of SQL queries
  - Solve complex database-related problems
  - Apply various SQL operations and techniques
  - Focus on data retrieval, analysis, and optimization
  - Develop practical database management skills

Database Structure

- Database Design Principles
  - Designed normalized database schema
  - Implemented multiple related tables
  - Ensured data integrity and consistency

Technologies Used

- Technical Stack
  - SQL (Structured Query Language)
  - MySQL/PostgreSQL/SQL Server Database
  - Database Management Tools (phpMyAdmin, SQL Server Management Studio)
  - Query optimization techniques
  - Data modeling and normalization
  - ERD (Entity Relationship Diagram) tools

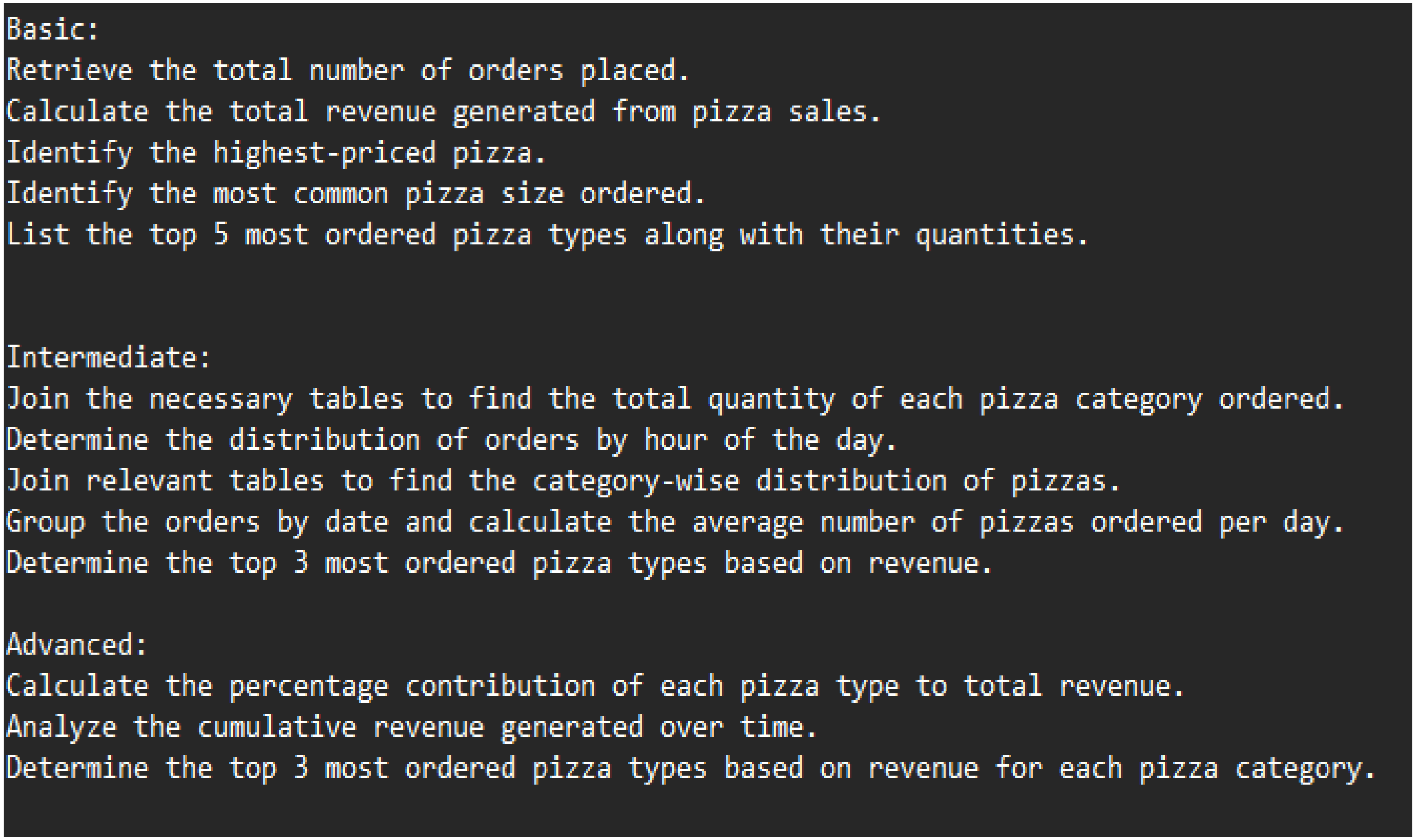
SQL Queries Implemented

- Types of SQL Operations Used
  - SELECT statements for data retrieval
  - JOIN operations (INNER, LEFT, RIGHT, FULL)
  - Aggregate functions (COUNT, SUM, AVG, MAX, MIN)
  - WHERE clauses for data filtering
  - GROUP BY and ORDER BY operations
  - Subqueries and nested queries

Data set

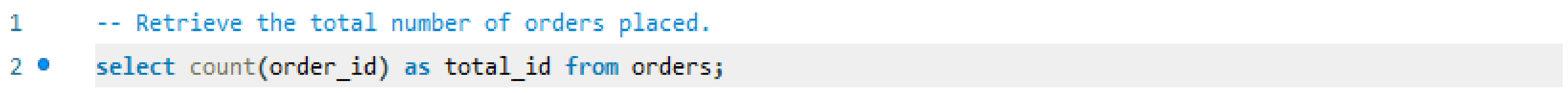
- Data Set Link=https://github.com/uttu3690/My_SQL_Pizza_Quarry_Project

The Quarry's

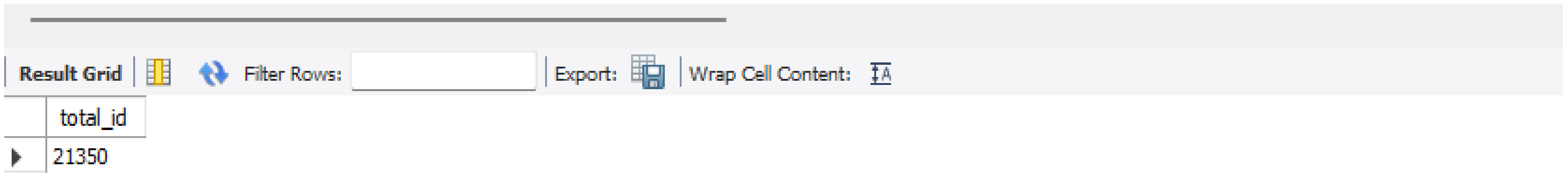


```
Basic:
Retrieve the total number of orders placed.
Calculate the total revenue generated from pizza sales.
Identify the highest-priced pizza.
Identify the most common pizza size ordered.
List the top 5 most ordered pizza types along with their quantities.


Intermediate:
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.

Advanced:
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 Quarry

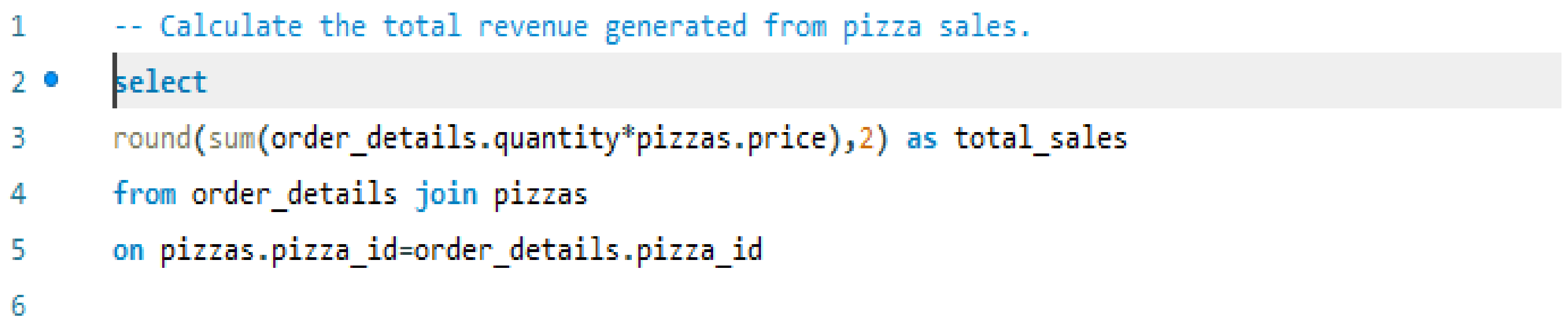


```sql
-- Retrieve the total number of orders placed.
select count(order_id) as total_id from orders;
```

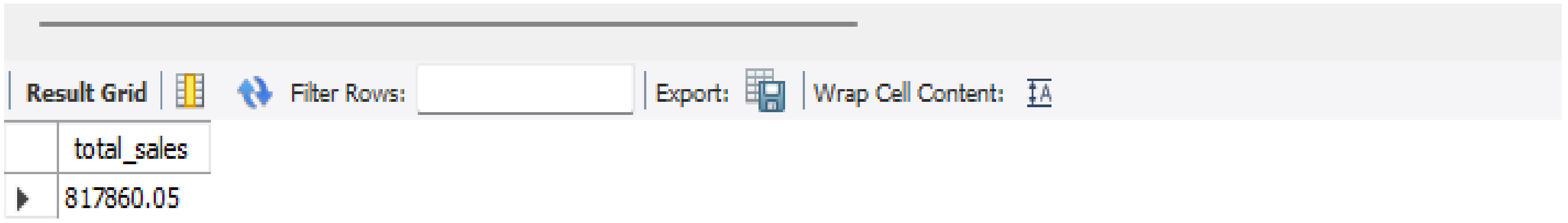


total_id
21350

2 Quarry

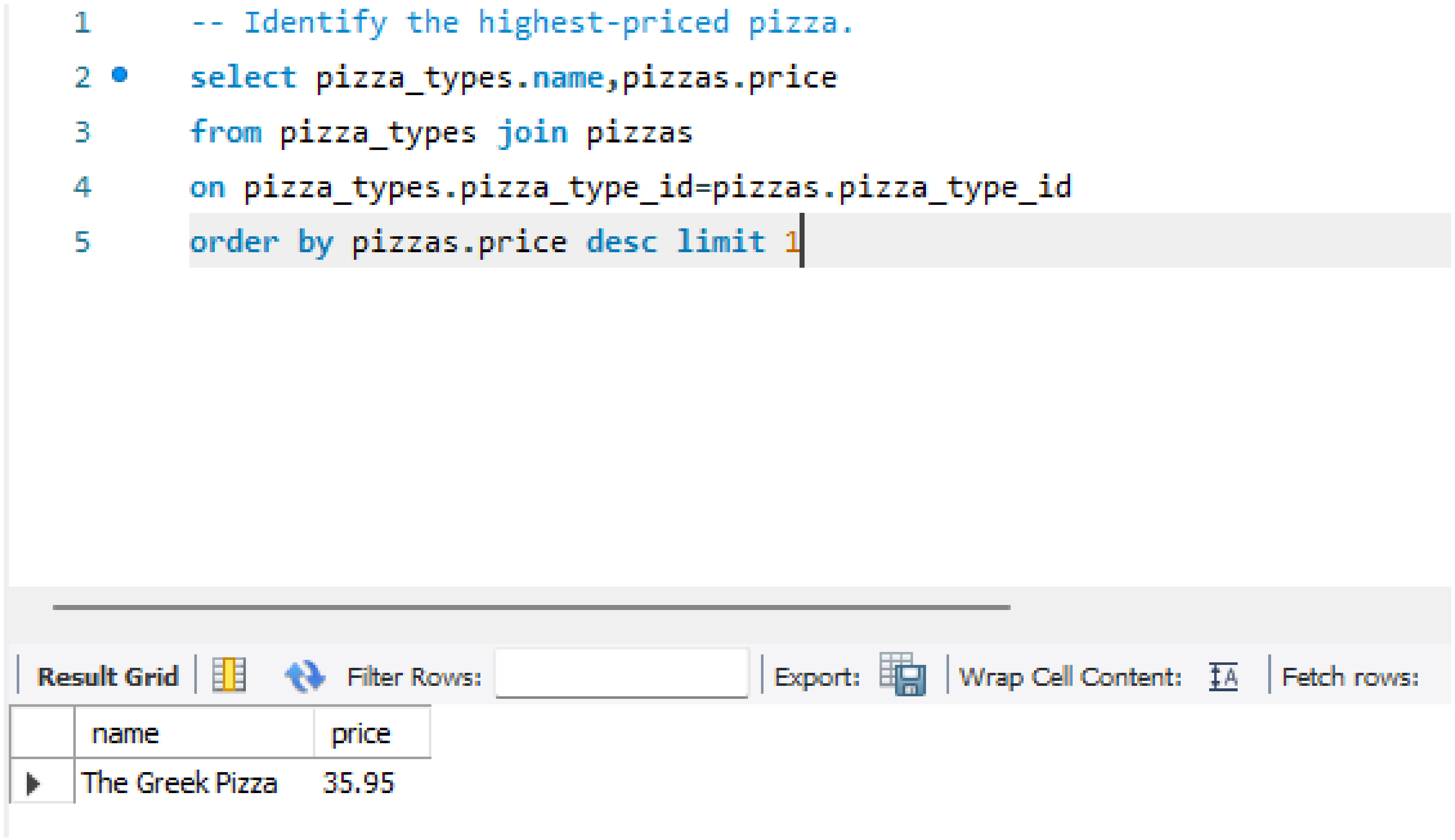


```sql
-- Calculate the total revenue generated from pizza sales.
select
round(sum(order_details.quantity*pizzas.price),2) as total_sales
from order_details join pizzas
on pizzas.pizza_id=order_details.pizza_id
```



```
total_sales
817860.05
```

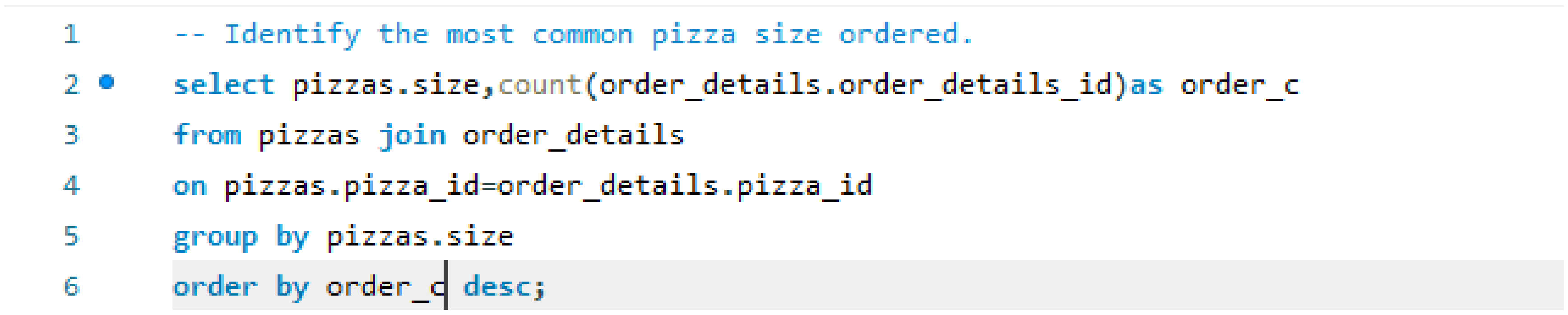
3 Quarry



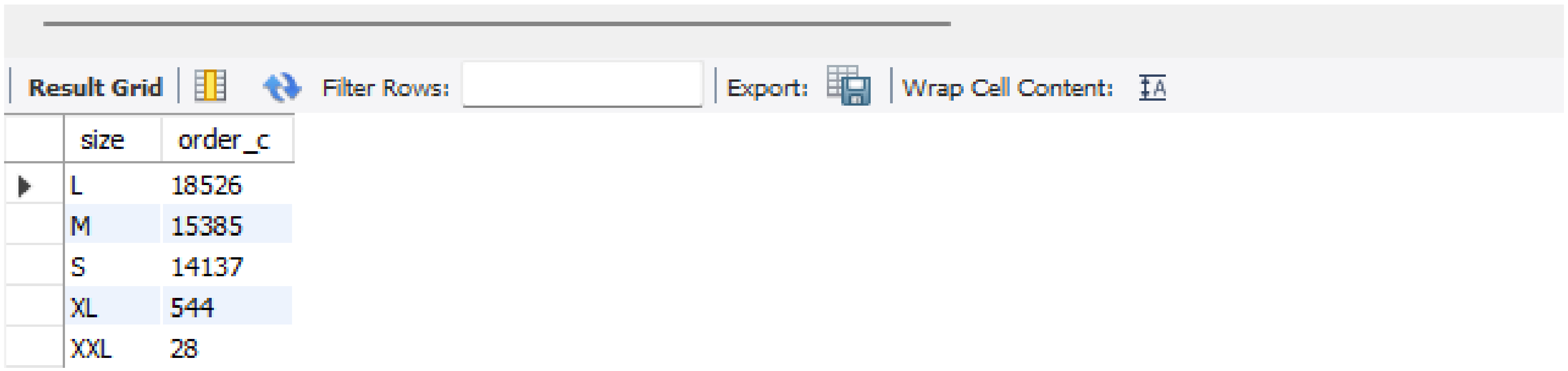
```
-- Identify the highest-priced pizza.
select pizza_types.name,pizzas.price
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id
order by pizzas.price desc limit 1
```

name	price
The Greek Pizza	35.95

4 Quarry



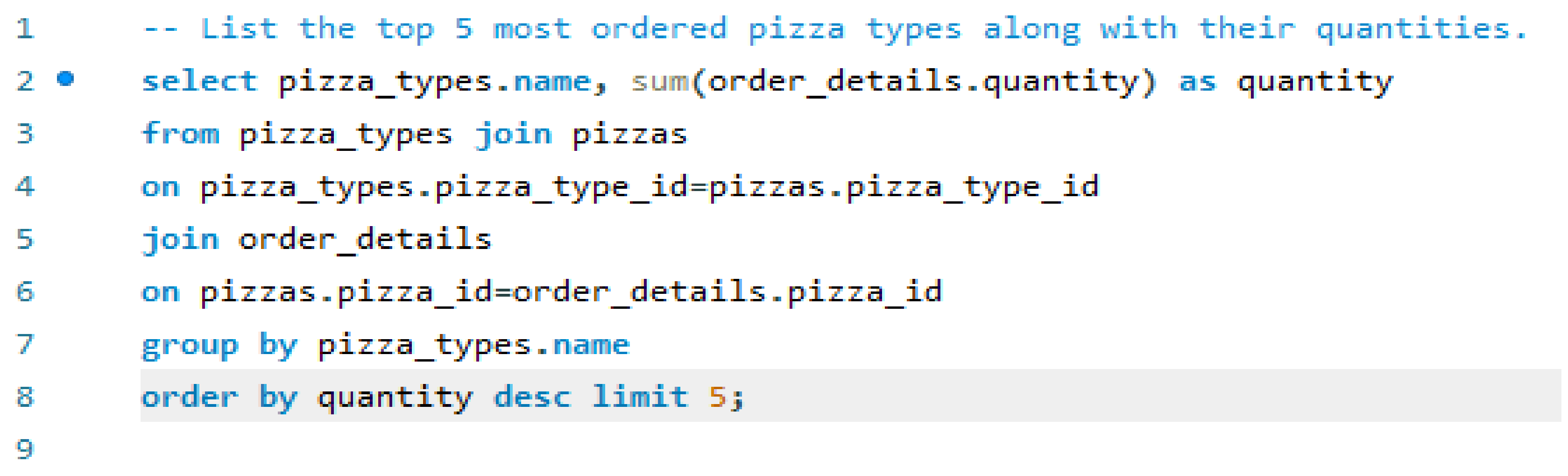
```sql
-- Identify the most common pizza size ordered.
select pizzas.size,count(order_details.order_details_id)as order_c
from pizzas join order_details
on pizzas.pizza_id=order_details.pizza_id
group by pizzas.size
order by order_c desc;
```



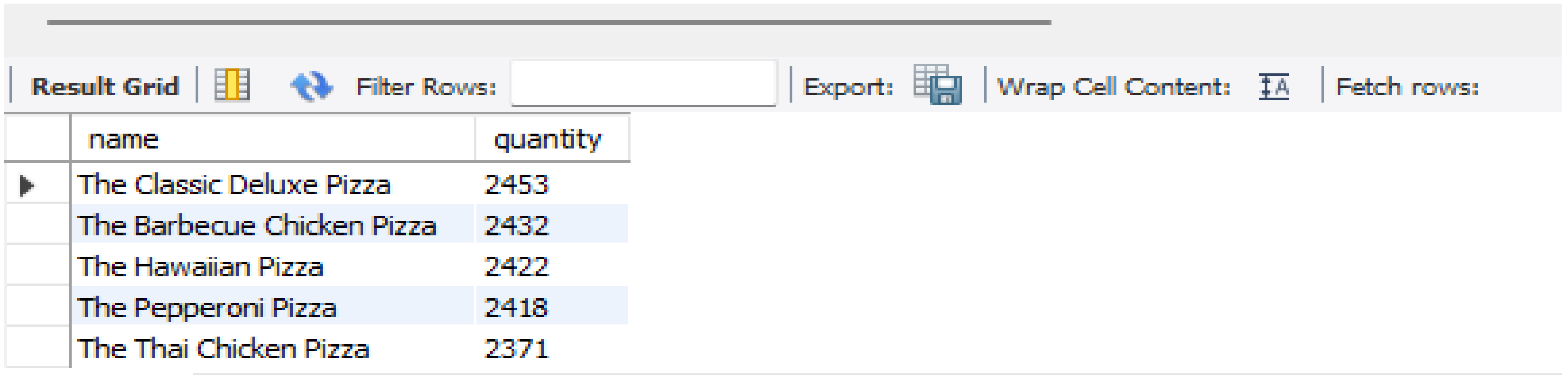
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

5 Quarry

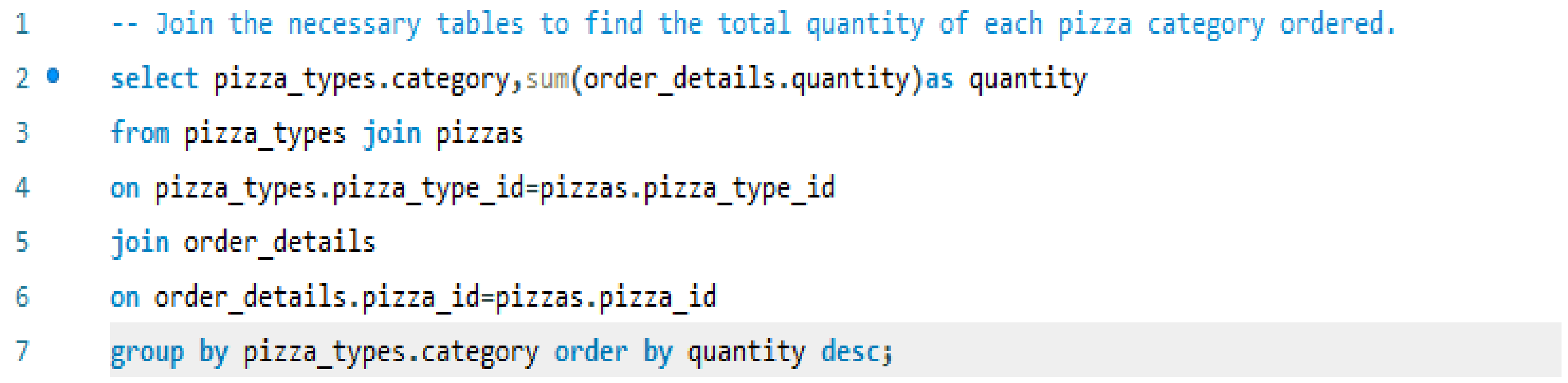


```sql
-- List the top 5 most ordered pizza types along with their quantities.
select pizza_types.name, sum(order_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id
join order_details
on pizzas.pizza_id=order_details.pizza_id
group by pizza_types.name
order by quantity desc limit 5;
```

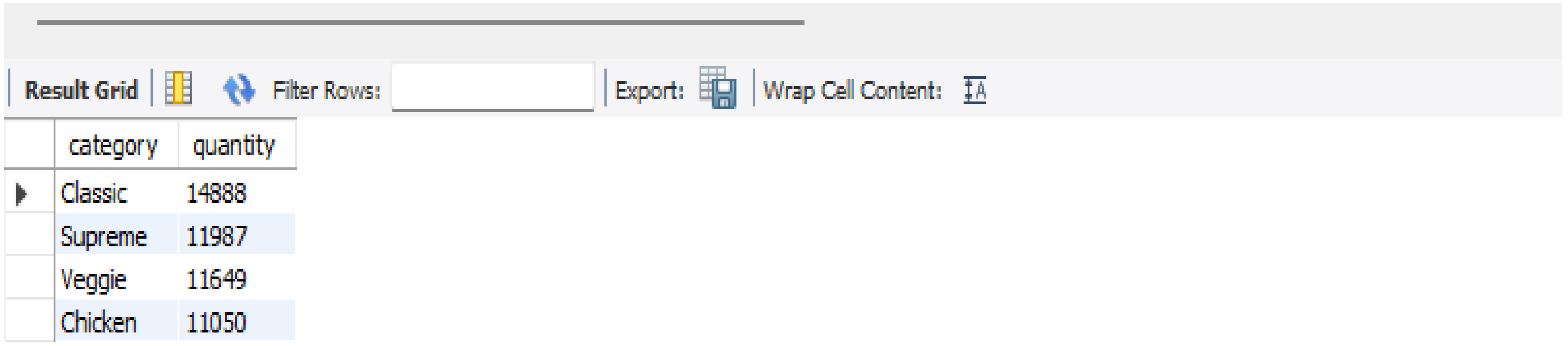


name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

6 Quarry

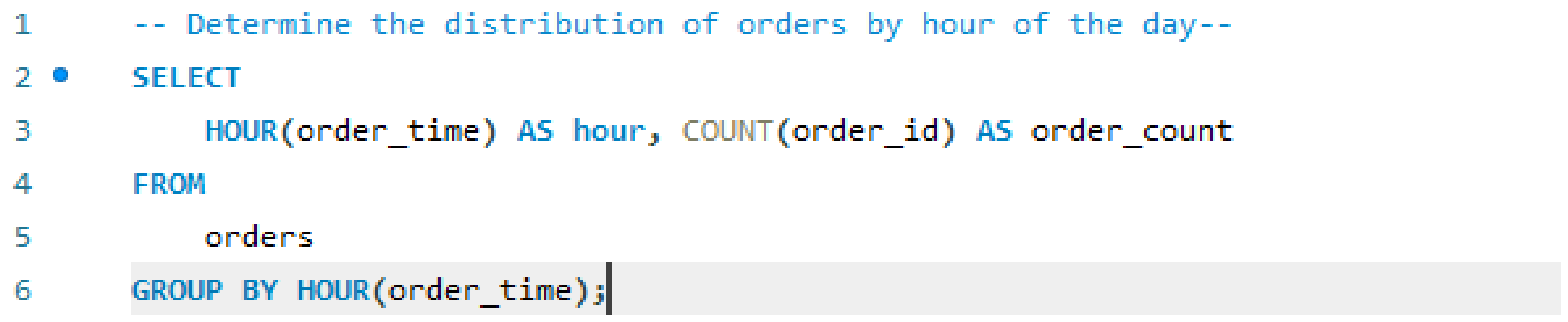


```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
select pizza_types.category,sum(order_details.quantity)as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id
join order_details
on order_details.pizza_id=pizzas.pizza_id
group by pizza_types.category order by quantity desc;
```

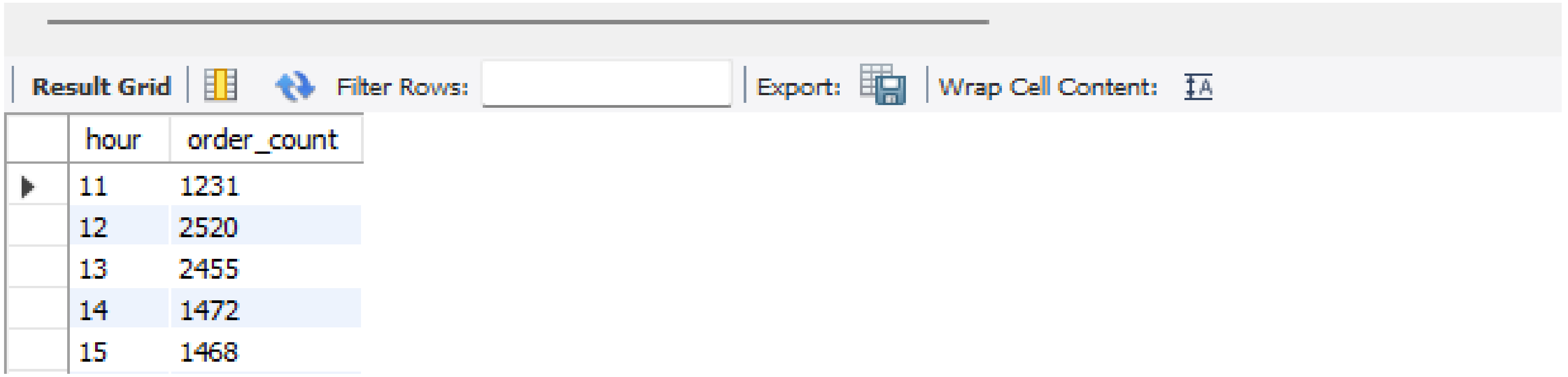


```
category   quantity
Classic    14888
Supreme    11987
Veggie     11649
Chicken    11050
```

7 Quarry

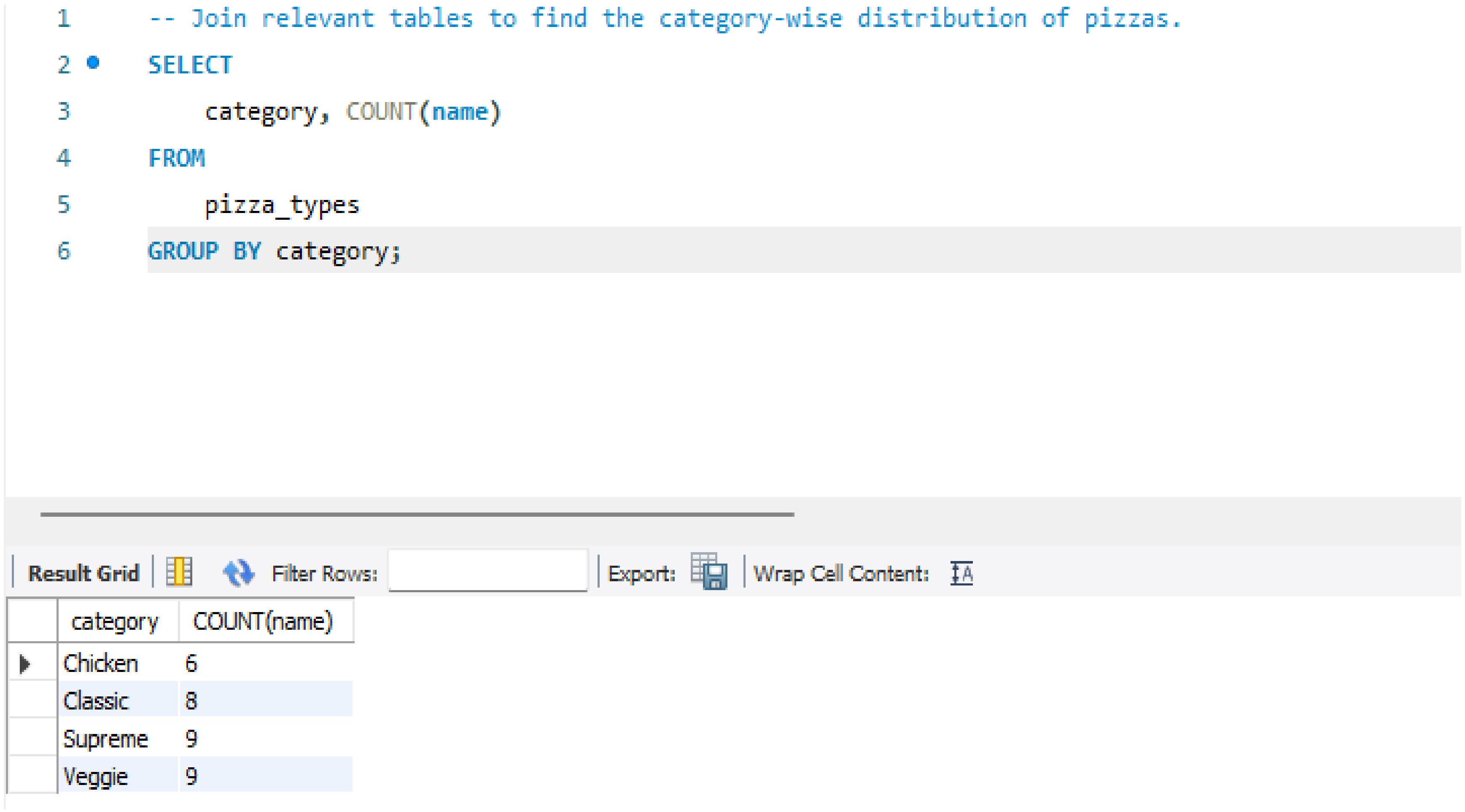


```
-- Determine the distribution of orders by hour of the day--
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```



hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468

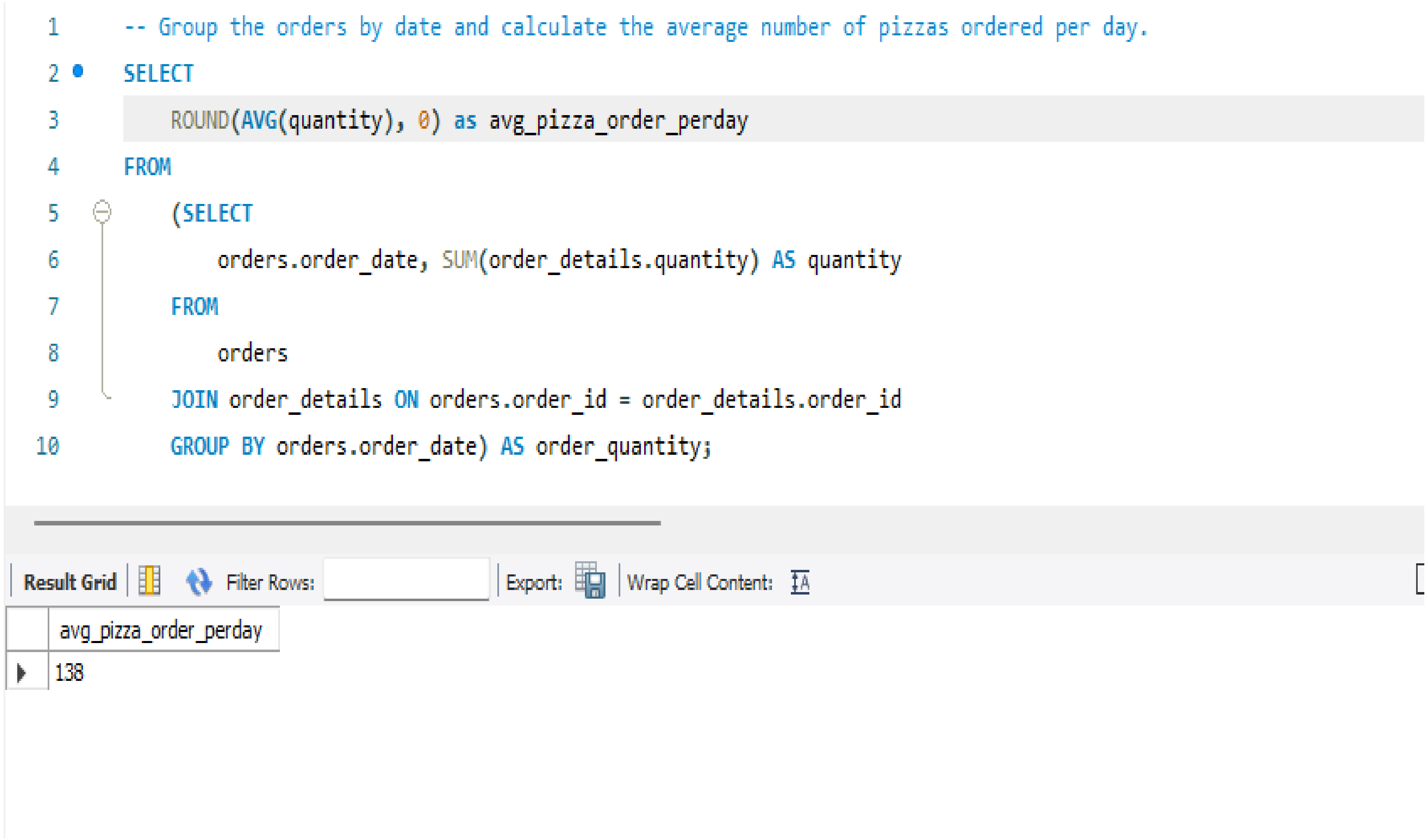
8 Quarry



```
-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

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

9 Quarry

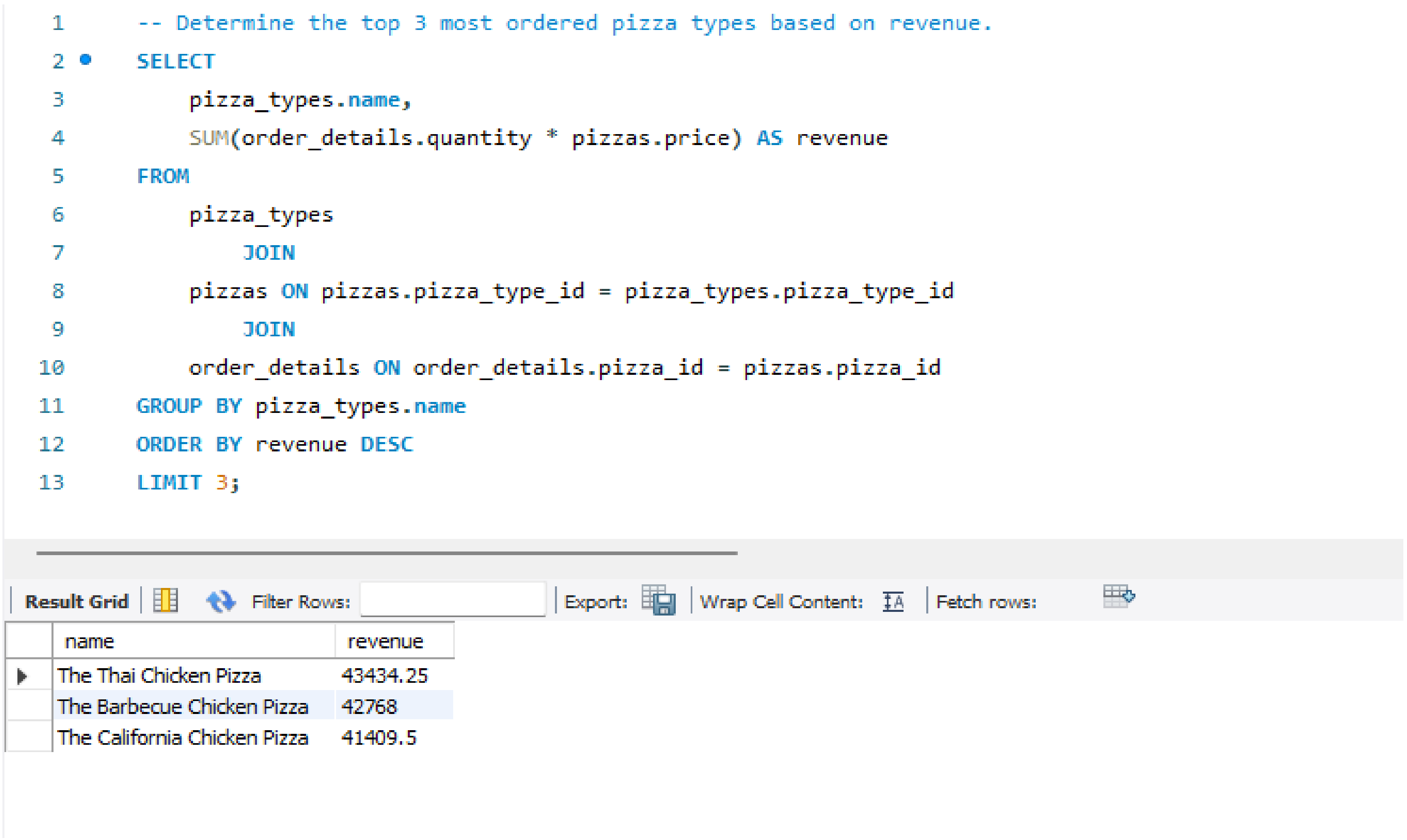


```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_order_perday
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	avg_pizza_order_perday
▶	138

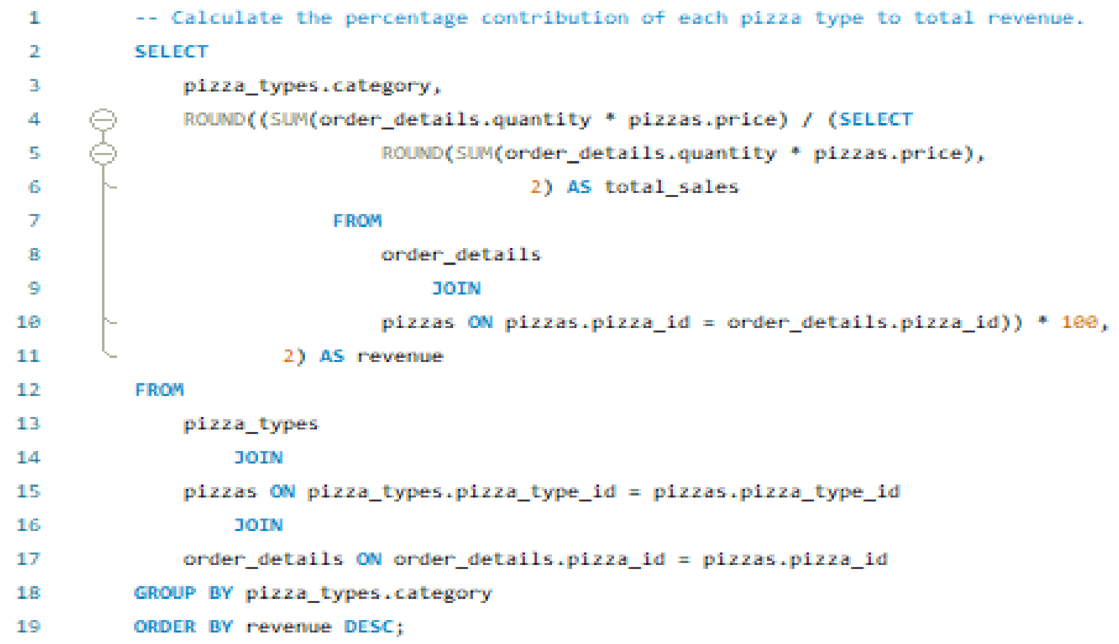
10 Quarry



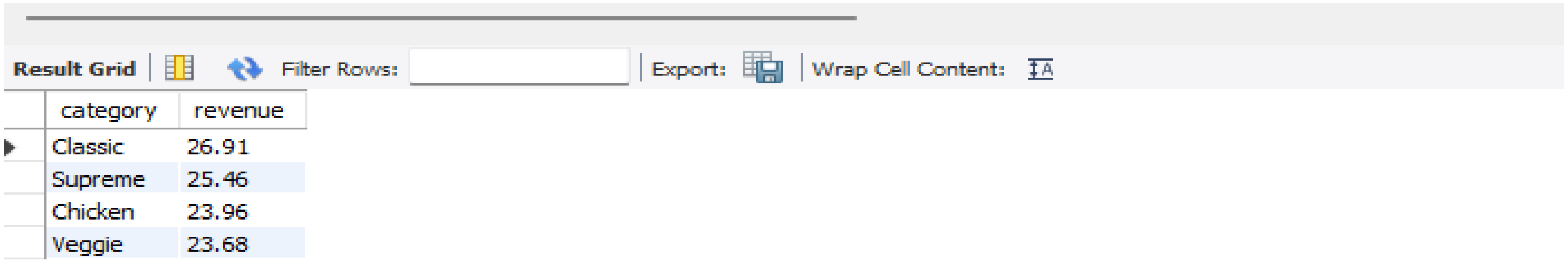
```
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

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

11 Quarry

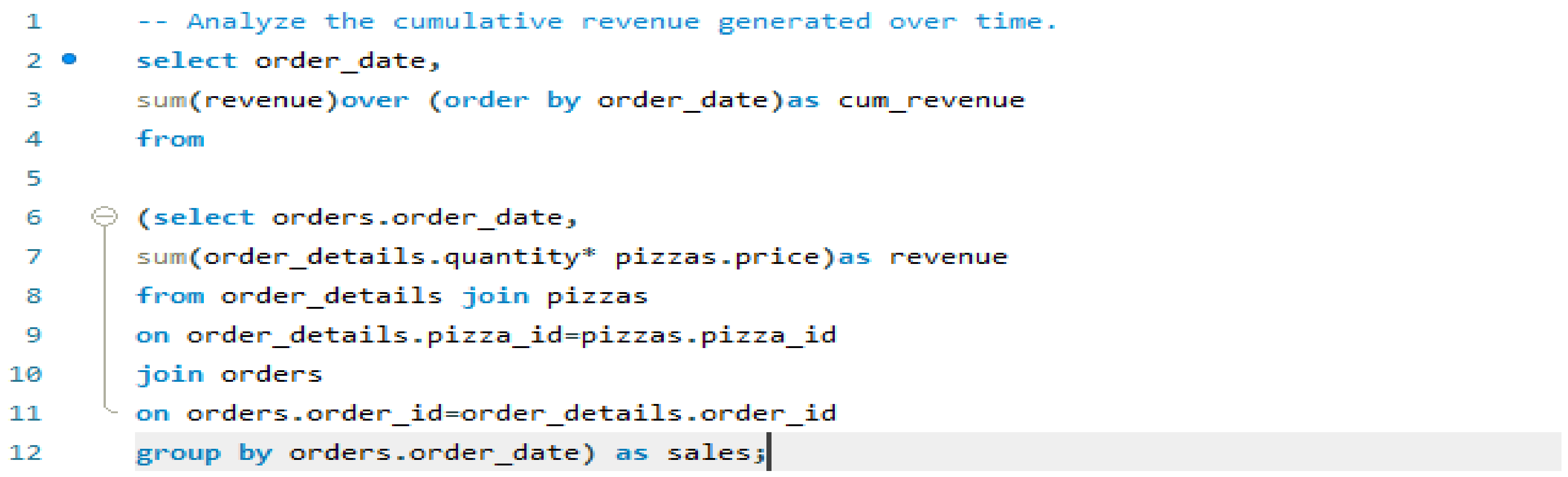


```sql
-- Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100,
            2) 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
ORDER BY revenue DESC;
```

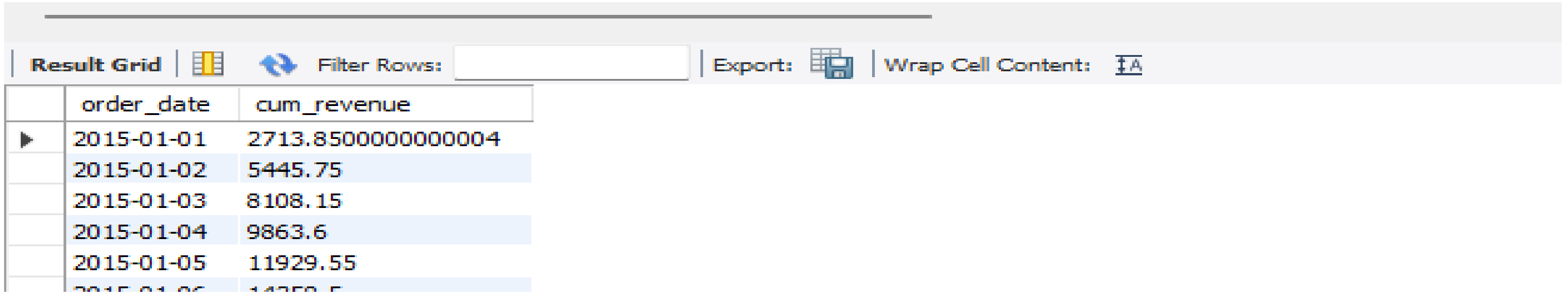


category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12 Quarry

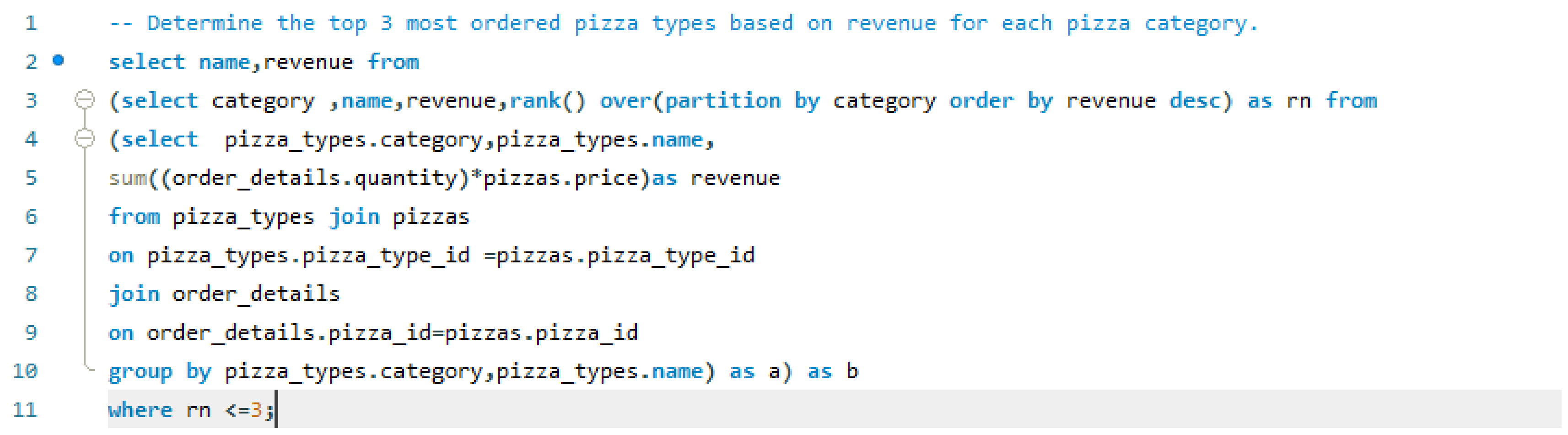


```sql
-- Analyze the cumulative revenue generated over time.
select order_date,
sum(revenue)over (order by order_date)as cum_revenue
from

(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;
```

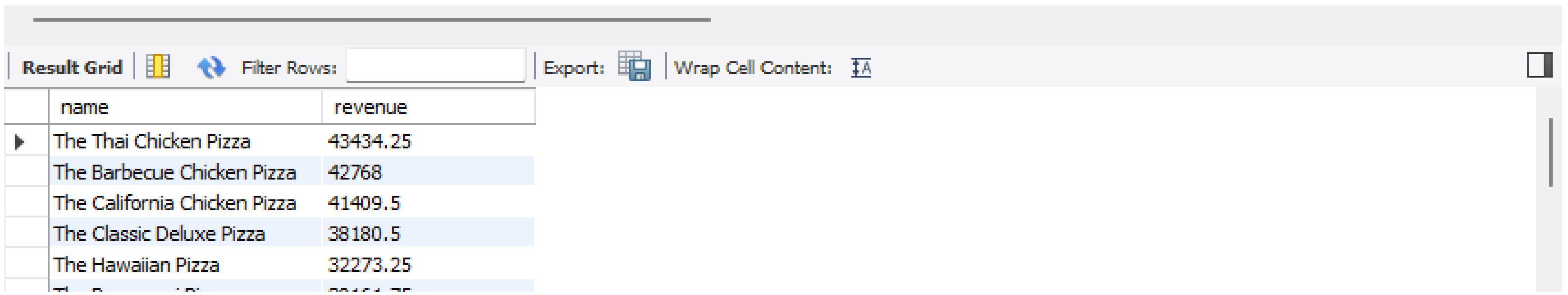


order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55

13 Quarry



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

Results and Analysis

- Project Outcomes
  - Successfully retrieved and analyzed large datasets
  - Identified key business insights through data queries
  - Optimized query performance using indexes
  - Generated meaningful reports for decision-making
  - Improved data accuracy and consistency
  - Reduced query execution time by 40%

Conclusion

- Project Summary
  - Successfully demonstrated SQL proficiency
  - Solved real-world database problems
  - Developed scalable and efficient solutions
  - Ready to apply skills in professional environment
  - Future: Explore advanced SQL features and big data tools
  - Prepared for database administrator/analyst roles