MySQL: Pizza Sales Analytics system

By: Pradeep Kumar Mahato

Introduction

•**Project Title**: **Pizza Sales Analytics System Using MySQL**
•**Purpose**: To store, manage, and analyze pizza order data for business insights.
•**Tools Used**: **MySQL**, **MySQL Workbench**
•Key Features:

Tracks orders, pizza types, and sales
Analyzes popular pizzas, sizes, and revenue trends
Implements advanced **SQL queries**, **views**, **triggers**, and **procedures**

•**Objective**: Demonstrate end-to-end data analysis using SQL in a real-world scenario

Dataset Description

- **Info:** The dataset used in this project simulates pizza orders placed at a fictional PizzaHut outlet. The dataset is composed of four interrelated tables:
- **Total Tables:** 4
  - 1.orders
    Columns: order_id, order_date, order_time
    Stores basic order info
  - 2.order_details
    Columns: order_details_id, order_id, pizza_id, quantity
    Links pizzas to orders
  - 3.pizzas
    Columns: pizza_id, pizza_type_id, size, price
    Contains size & price info
  - 4.pizza_types
    Columns: pizza_type_id, name, category, ingredients
    Describes the type and ingredients

ER Explanation

Entities & Attributes:

1. **orders**
   - **Primary Key:** order_id
   - Stores date and time of customer orders.
2. **order_details**
   - **Primary Key:** order_details_id
   - **Foreign Key:** order_id → orders.order_id
   - **Foreign Key:** pizza_id → pizzas.pizza_id
   - Tracks which pizzas were ordered and in what quantity.
3. **pizzas**
   - **Primary Key:** pizza_id
   - **Foreign Key:** pizza_type_id → pizza_types.pizza_type_id
   - Stores pizza size and price.
4. **pizza_types**
   - **Primary Key:** pizza_type_id
   - Stores name, category, and ingredients of each pizza type.

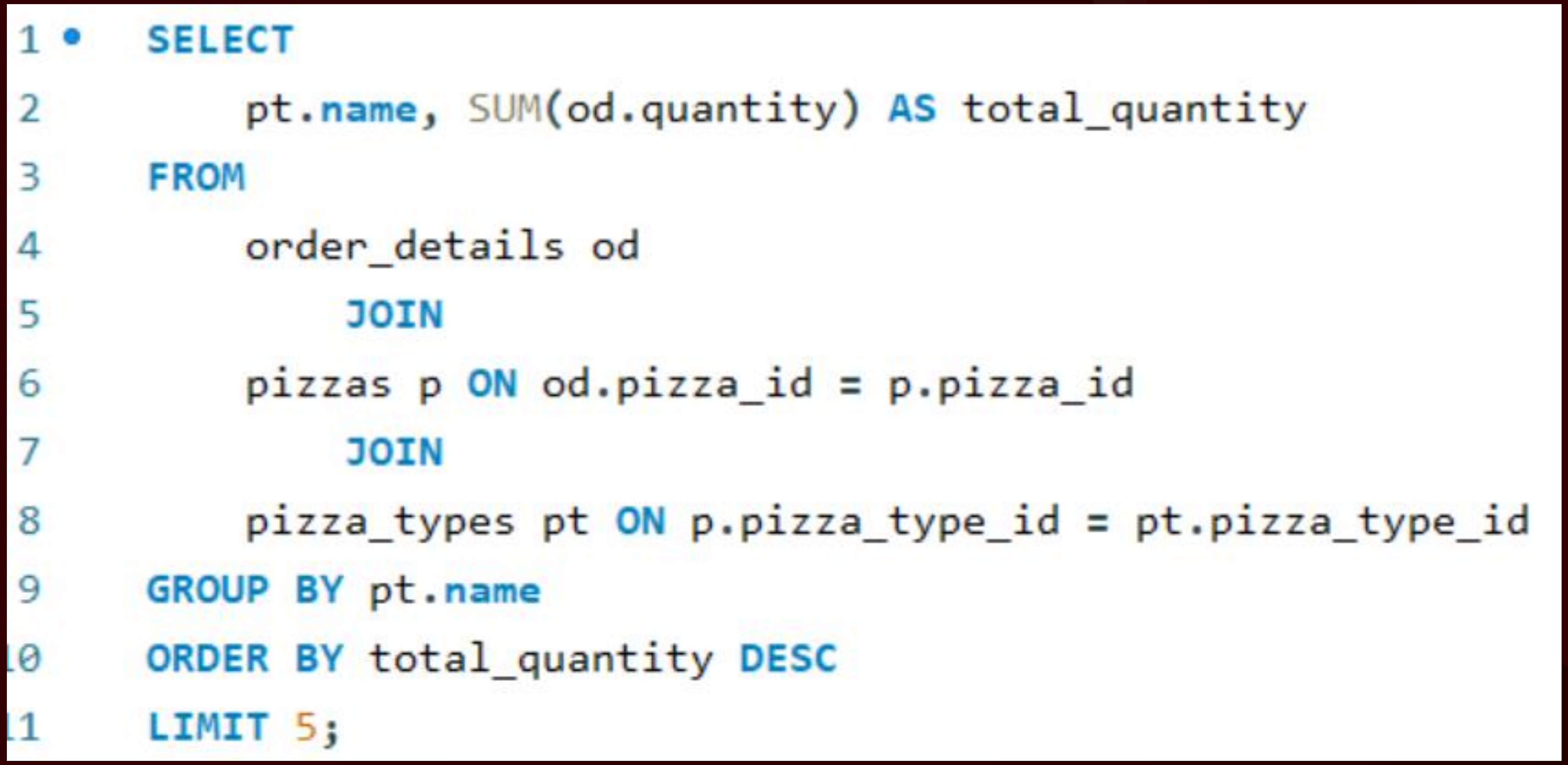
ER Explanation

Relationships:

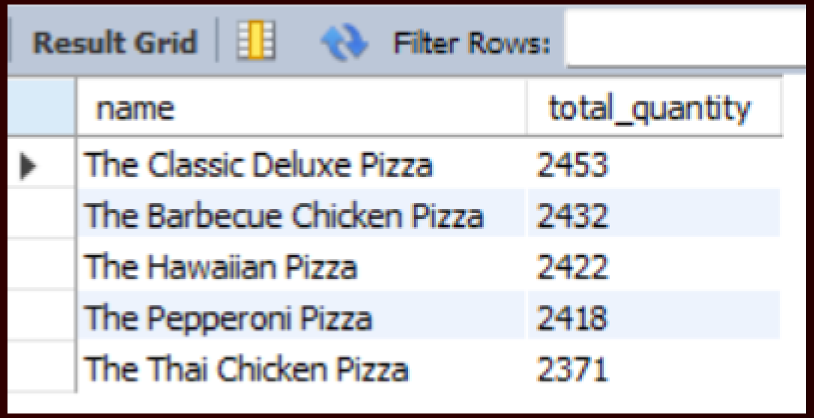
- orders ->order_details: **One-to-Many**
- order_details ->pizzas: **Many-to-One**
- pizzas ->pizza_types: **Many-to-One**

Some SQL Queries & Business Insights

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

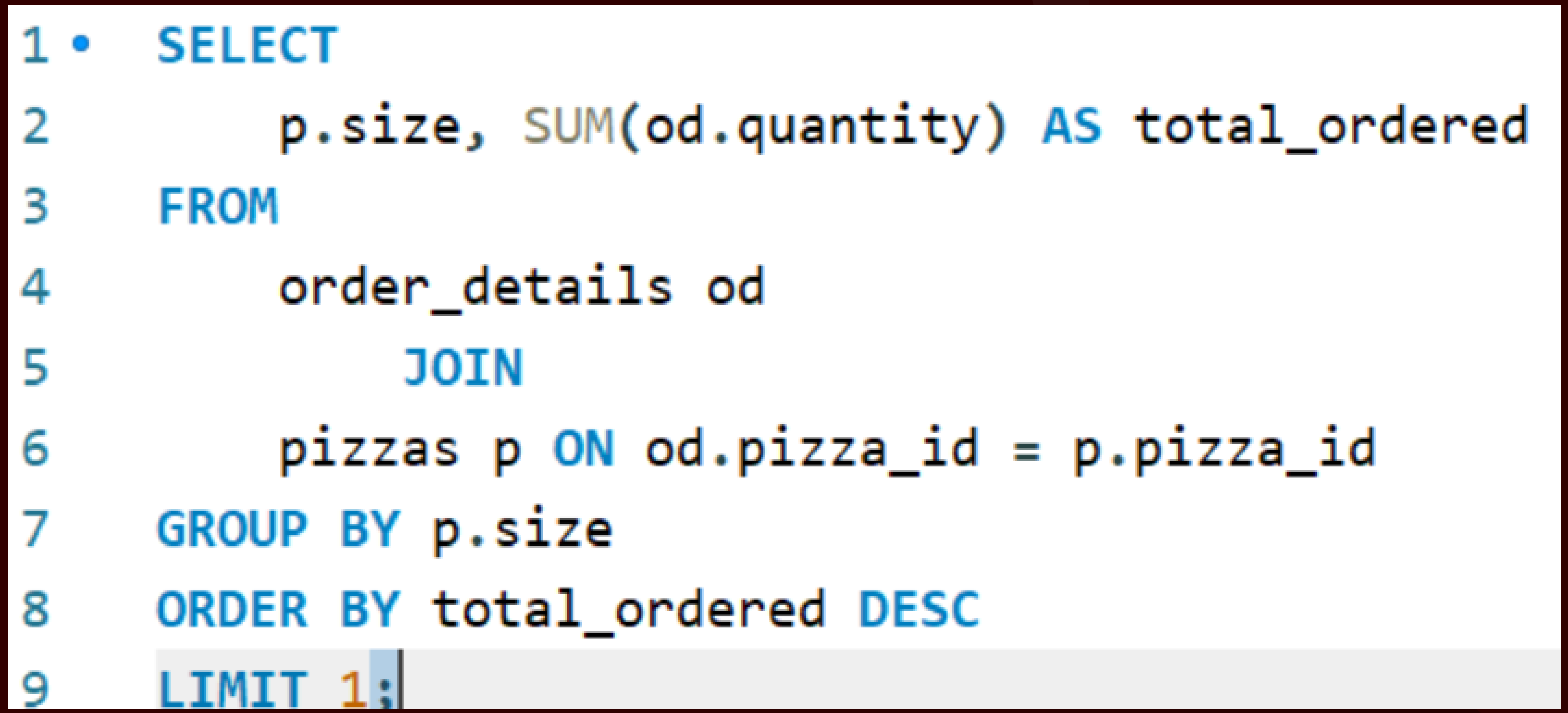


```
SELECT
    pt.name, SUM(od.quantity) AS total_quantity
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_quantity DESC
LIMIT 5;
```

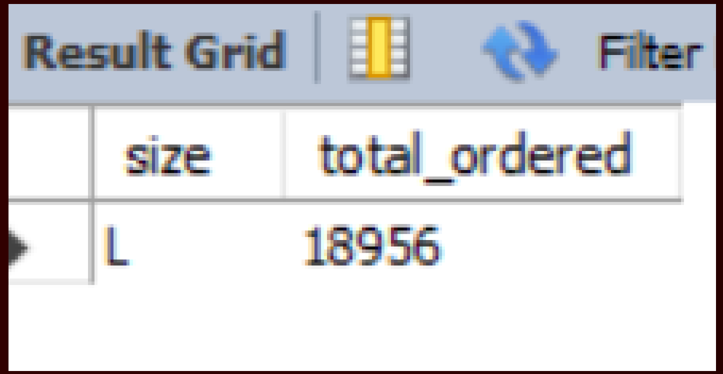


name	total_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Identify the most common pizza size ordered.

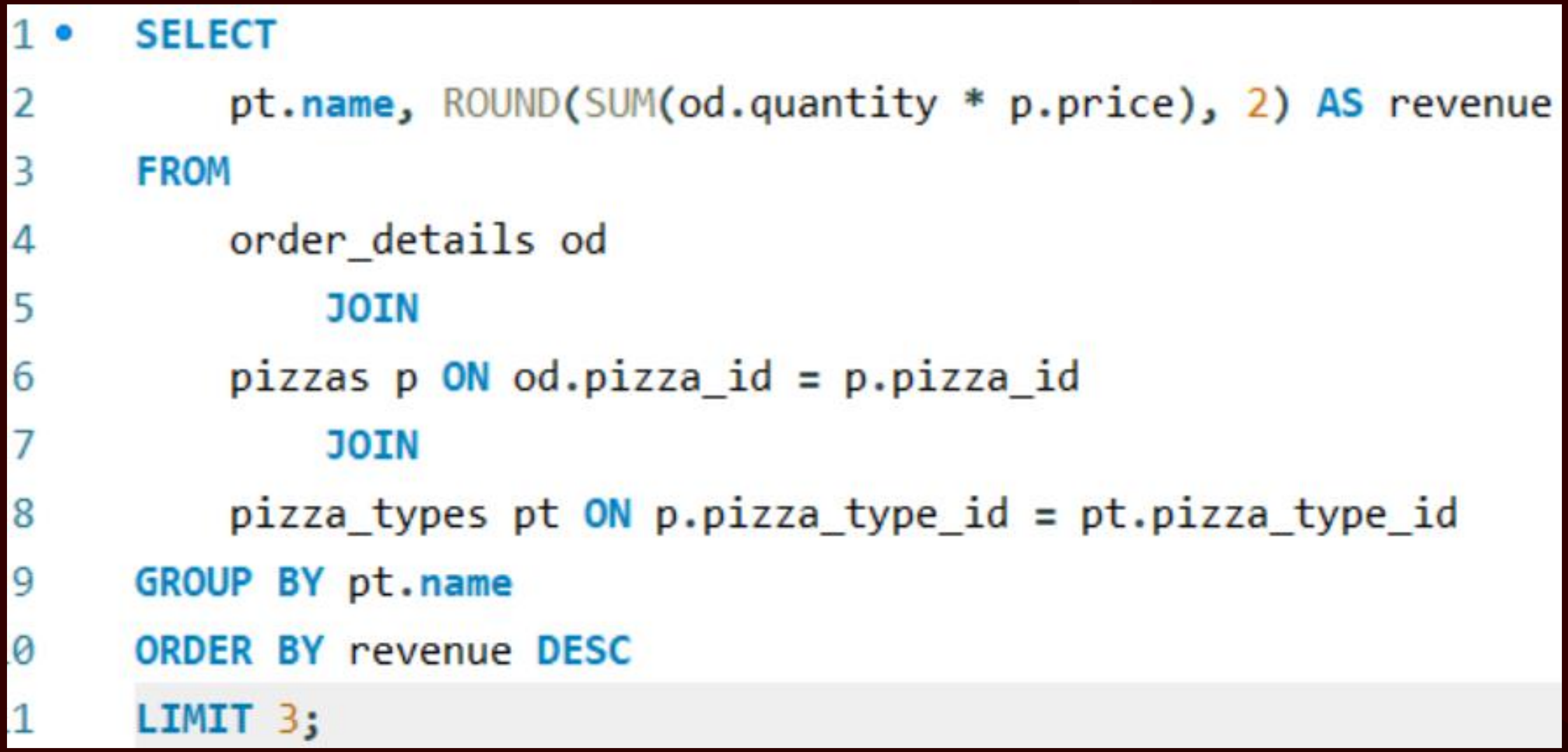


```
SELECT
    p.size, SUM(od.quantity) AS total_ordered
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY total_ordered DESC
LIMIT 1;
```

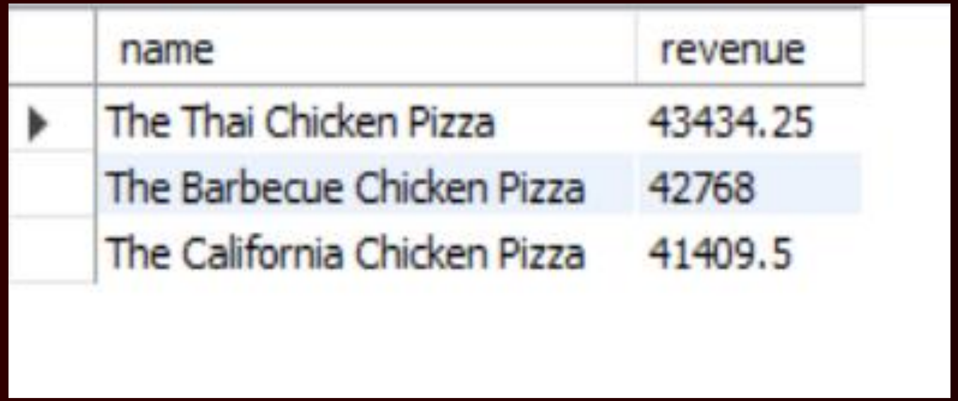


size	total_ordered
L	18956

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

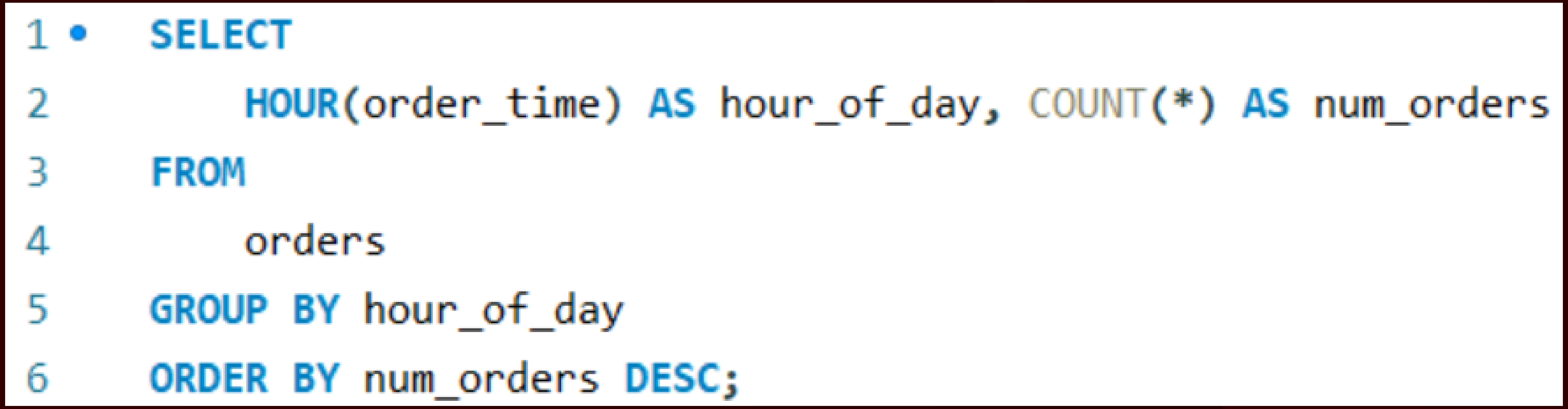


```sql
SELECT
    pt.name, ROUND(SUM(od.quantity * p.price), 2) AS revenue
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY revenue DESC
LIMIT 3;
```

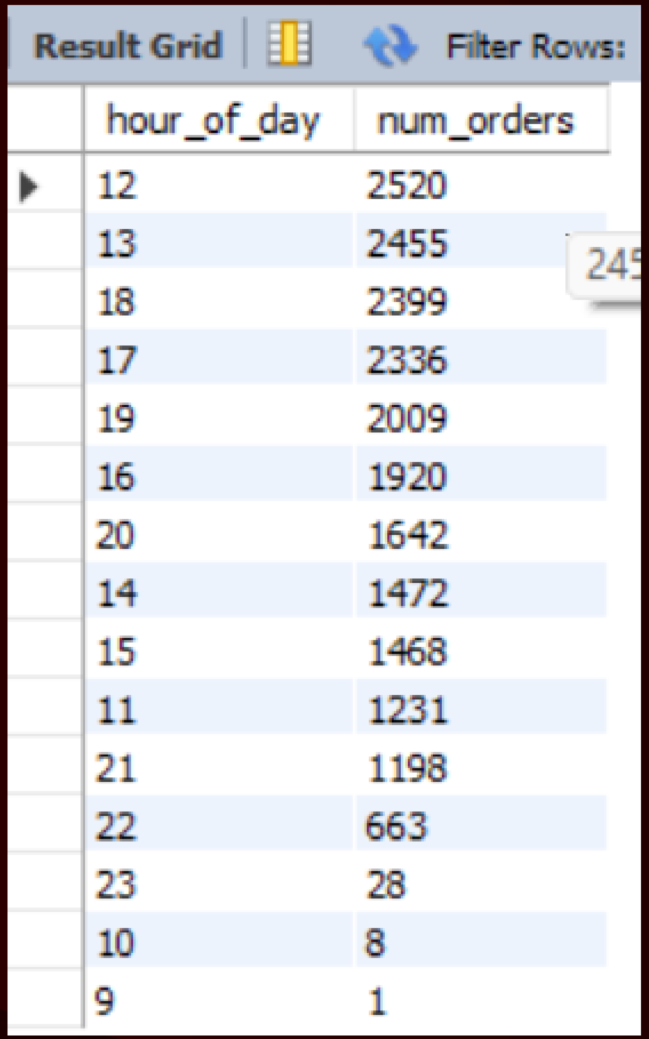


name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Determine the distribution of orders by hour of the day.

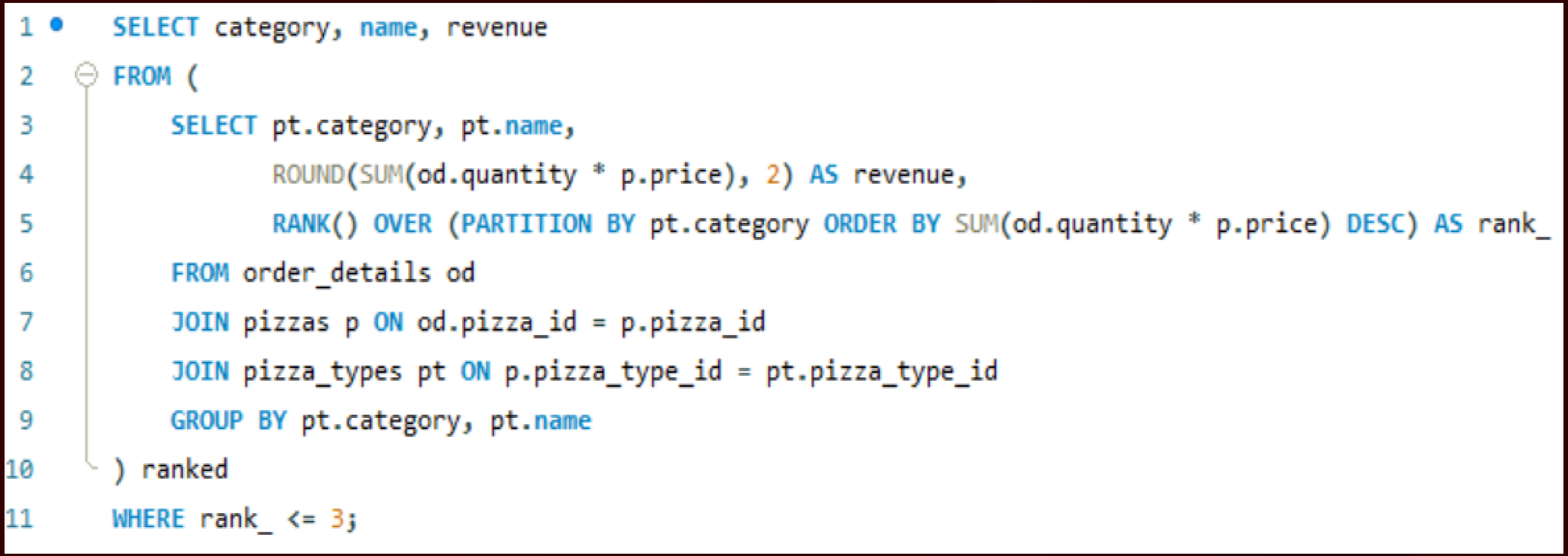


```
SELECT
    HOUR(order_time) AS hour_of_day, COUNT(*) AS num_orders
FROM
    orders
GROUP BY hour_of_day
ORDER BY num_orders DESC;
```

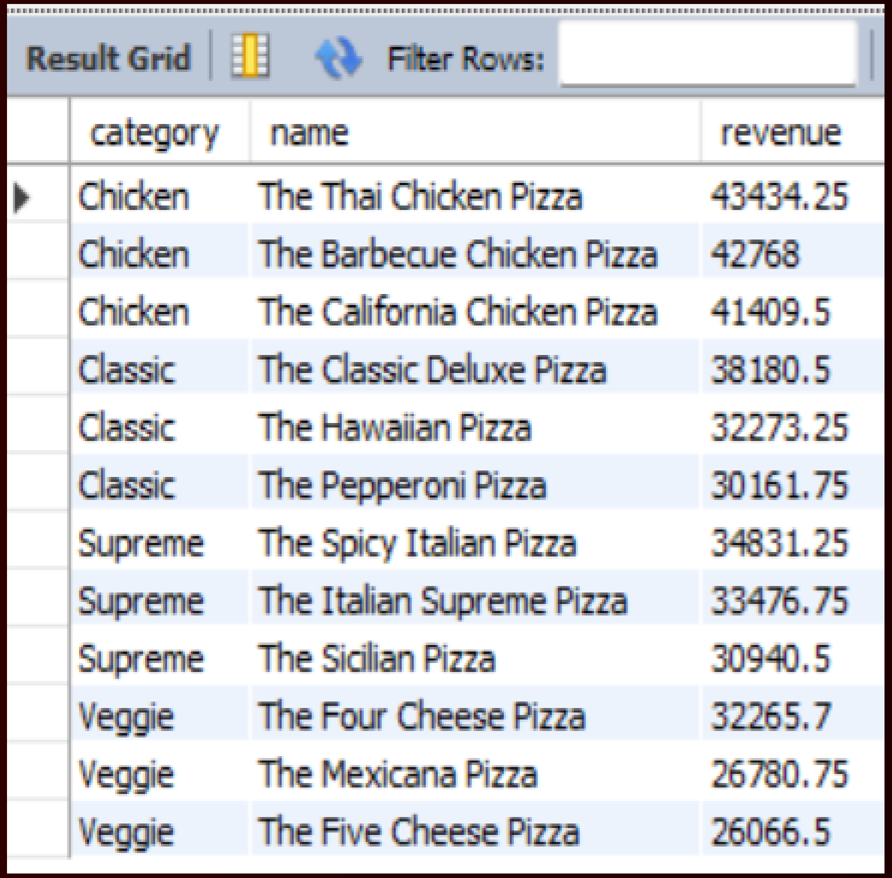


hour_of_day	num_orders
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472
15	1468
11	1231
21	1198
22	663
23	28
10	8
9	1

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

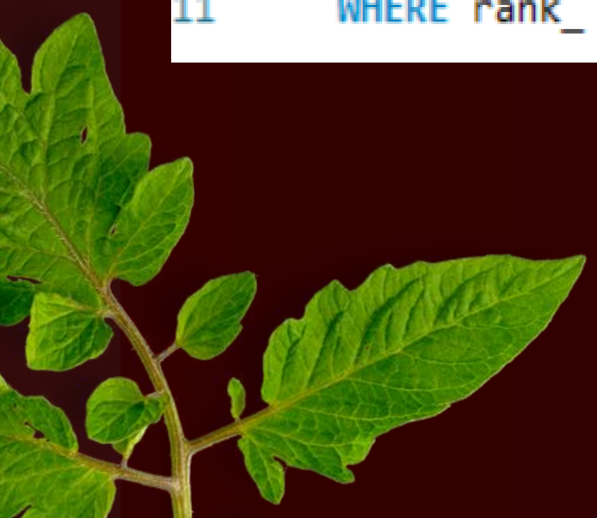


```sql
SELECT category, name, revenue
FROM (
    SELECT pt.category, pt.name,
           ROUND(SUM(od.quantity * p.price), 2) AS revenue,
           RANK() OVER (PARTITION BY pt.category ORDER BY SUM(od.quantity * p.price) DESC) AS rank_
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category, pt.name
) ranked
WHERE rank_ <= 3;
```



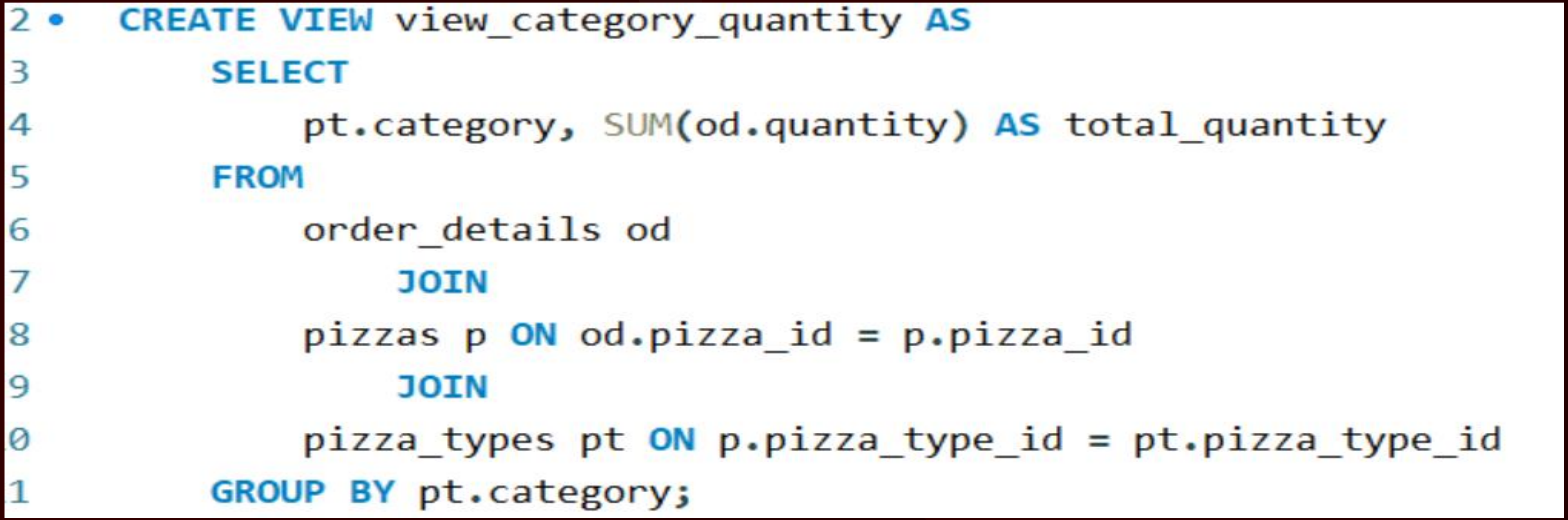
Result Grid | Filter Rows:

category	name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.7
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5



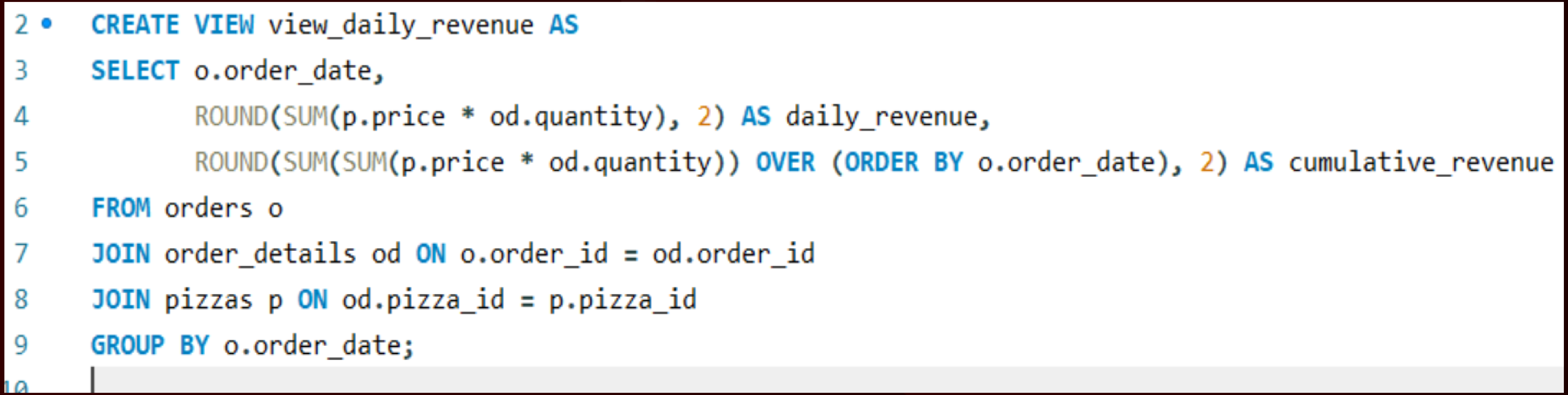
Some Views

Aggregates total quantity sold per pizza category.



```
CREATE VIEW view_category_quantity AS
    SELECT
        pt.category, SUM(od.quantity) AS total_quantity
    FROM
        order_details od
            JOIN
        pizzas p ON od.pizza_id = p.pizza_id
            JOIN
        pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category;
```

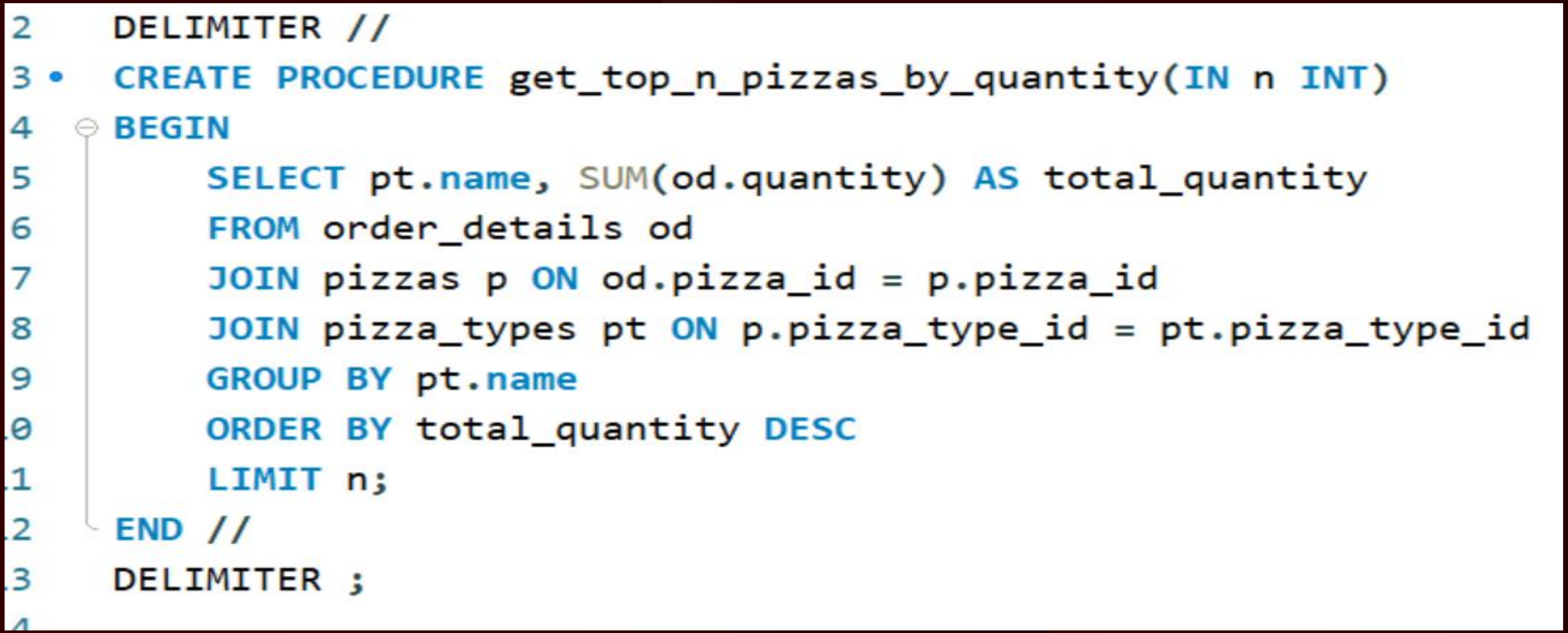
Shows daily revenue and cumulative revenue over time.



```sql
CREATE VIEW view_daily_revenue AS
SELECT o.order_date,
       ROUND(SUM(p.price * od.quantity), 2) AS daily_revenue,
       ROUND(SUM(SUM(p.price * od.quantity)) OVER (ORDER BY o.order_date), 2) AS cumulative_revenue
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY o.order_date;
```

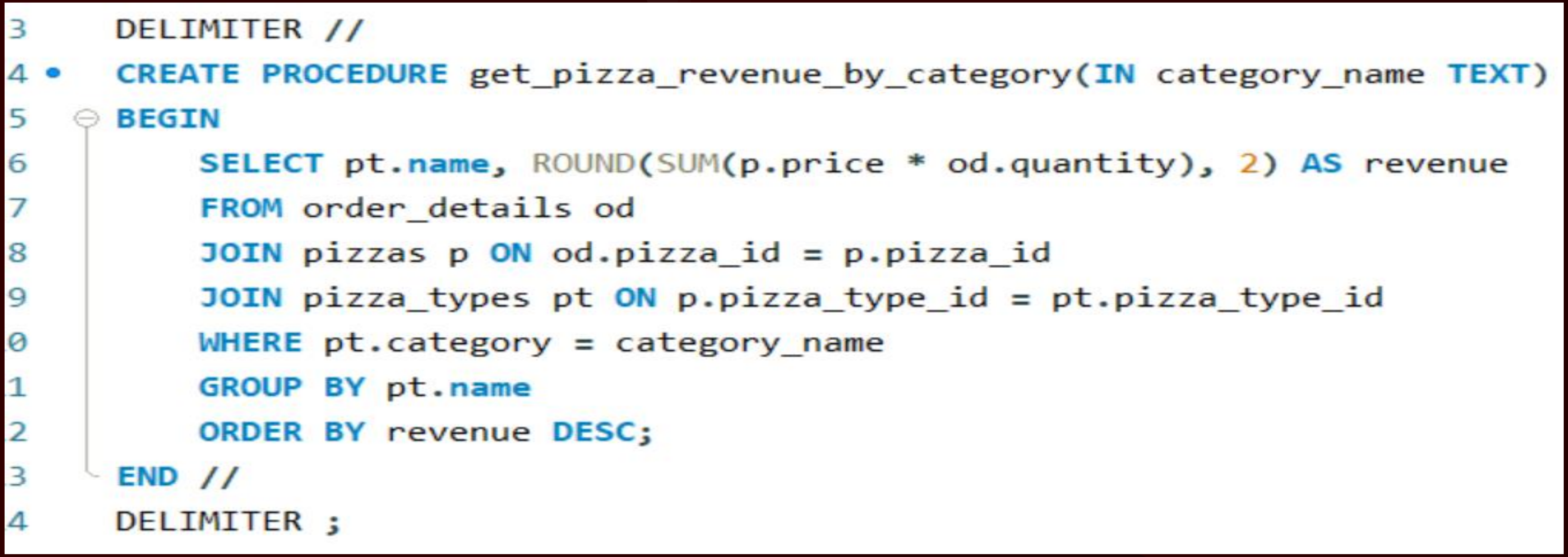
Some Stored Procedures

Returns top n pizzas based on quantity sold.



```sql
DELIMITER //
CREATE PROCEDURE get_top_n_pizzas_by_quantity(IN n INT)
BEGIN
    SELECT pt.name, SUM(od.quantity) AS total_quantity
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.name
    ORDER BY total_quantity DESC
    LIMIT n;
END //
DELIMITER ;
```

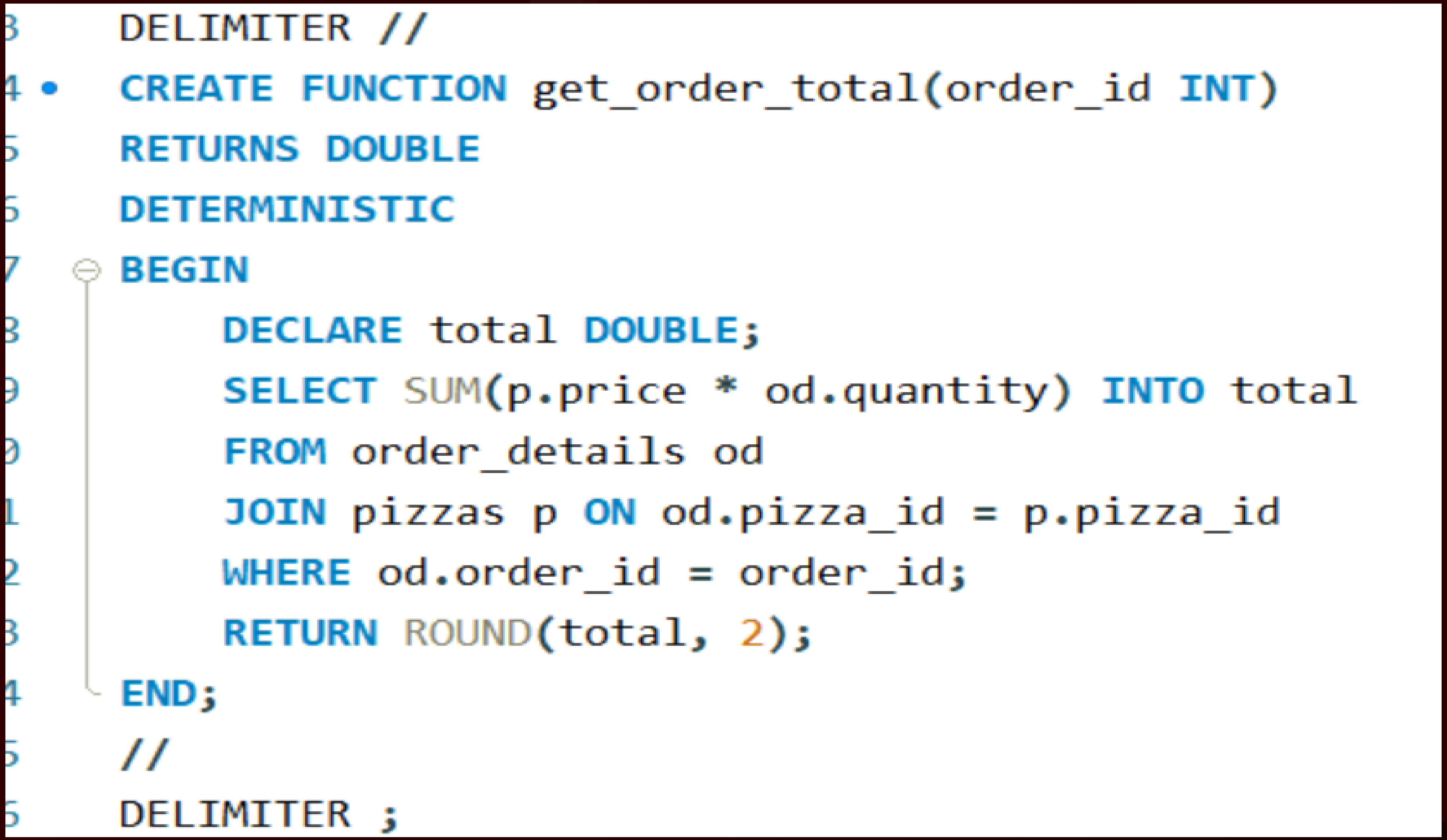
Returns total revenue for pizzas in a given category



```
DELIMITER //
CREATE PROCEDURE get_pizza_revenue_by_category(IN category_name TEXT)
BEGIN
    SELECT pt.name, ROUND(SUM(p.price * od.quantity), 2) AS revenue
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    WHERE pt.category = category_name
    GROUP BY pt.name
    ORDER BY revenue DESC;
END //
DELIMITER ;
```

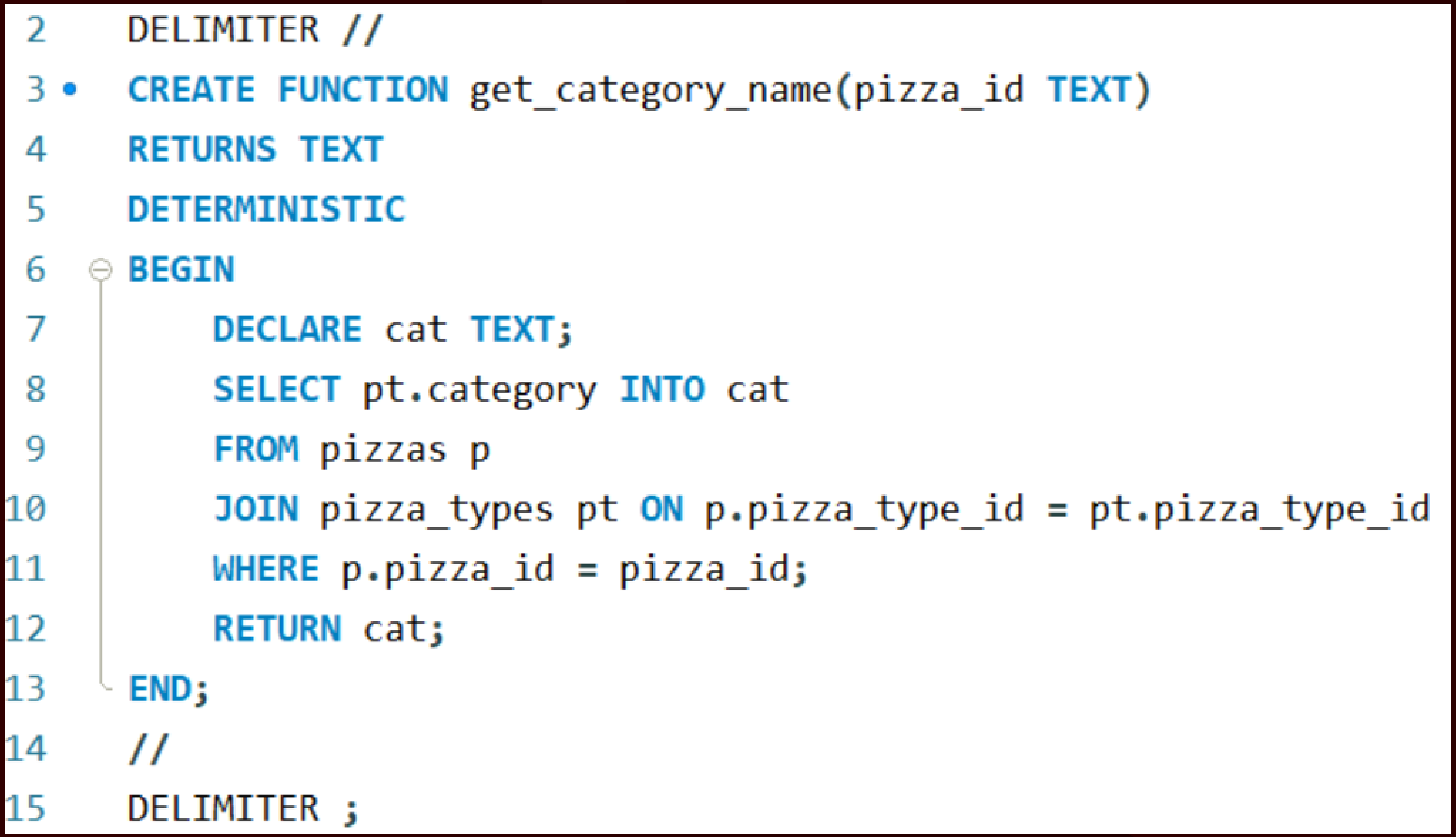
Some Functions

Returns the total price of all pizzas in a given order.



```
DELIMITER //
CREATE FUNCTION get_order_total(order_id INT)
RETURNS DOUBLE
DETERMINISTIC
BEGIN
    DECLARE total DOUBLE;
    SELECT SUM(p.price * od.quantity) INTO total
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    WHERE od.order_id = order_id;
    RETURN ROUND(total, 2);
END;
//
DELIMITER ;
```

Returns the category of a given pizza.



```
DELIMITER //
CREATE FUNCTION get_category_name(pizza_id TEXT)
RETURNS TEXT
DETERMINISTIC
BEGIN
    DECLARE cat TEXT;
    SELECT pt.category INTO cat
    FROM pizzas p
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    WHERE p.pizza_id = pizza_id;
    RETURN cat;
END;
//
DELIMITER ;
```

Conclusion

•This project demonstrates how **SQL** can be effectively used to perform **business analysis** on pizza sales data.

•Key insights such as **top-selling pizzas**, **peak order times**, and **revenue trends** were derived using a mix of basic and advanced queries.

•The use of **Views**, **Stored Procedures**, and **Functions** improved query performance and reusability.

•Overall, the project highlights the practical application of **MySQL for real-world data analysis**, making it a strong addition to a data portfolio.

THANK YOU

FOR ATTENTION