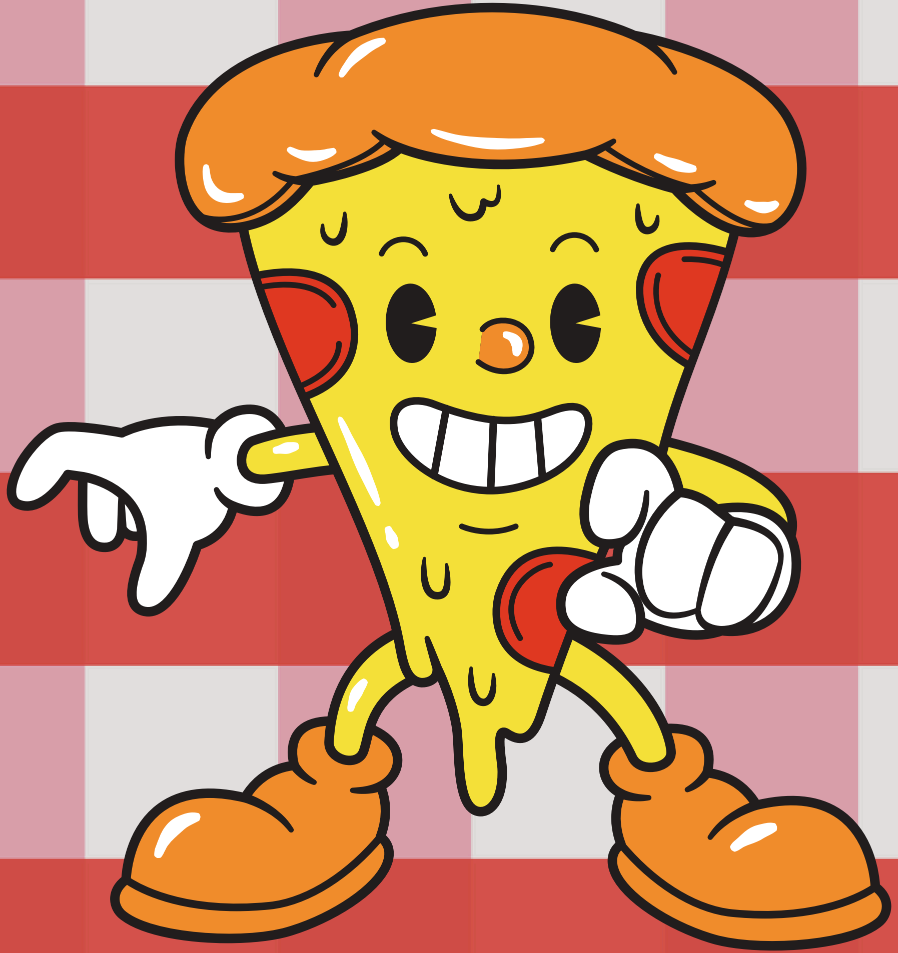
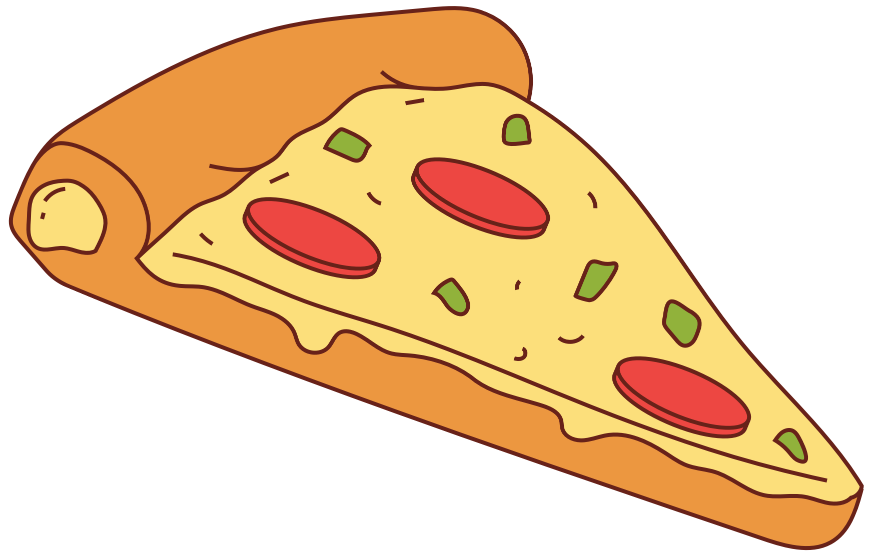
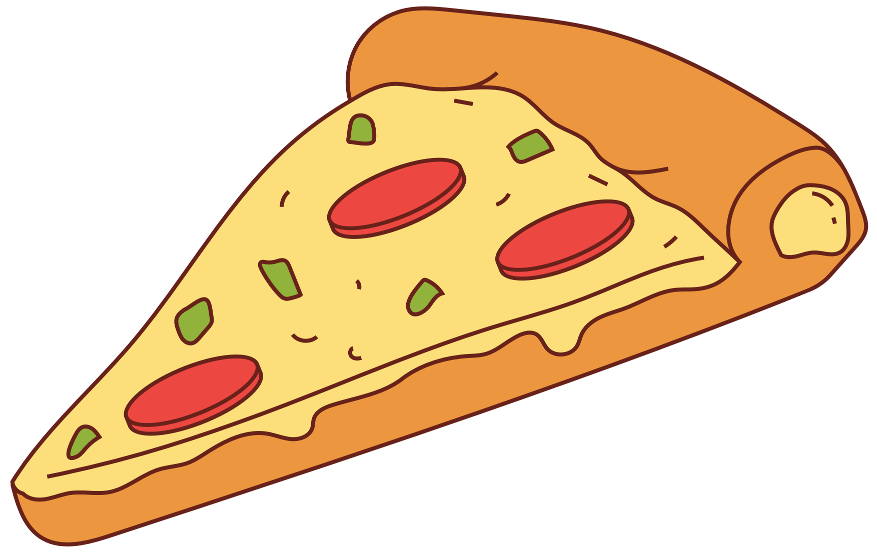
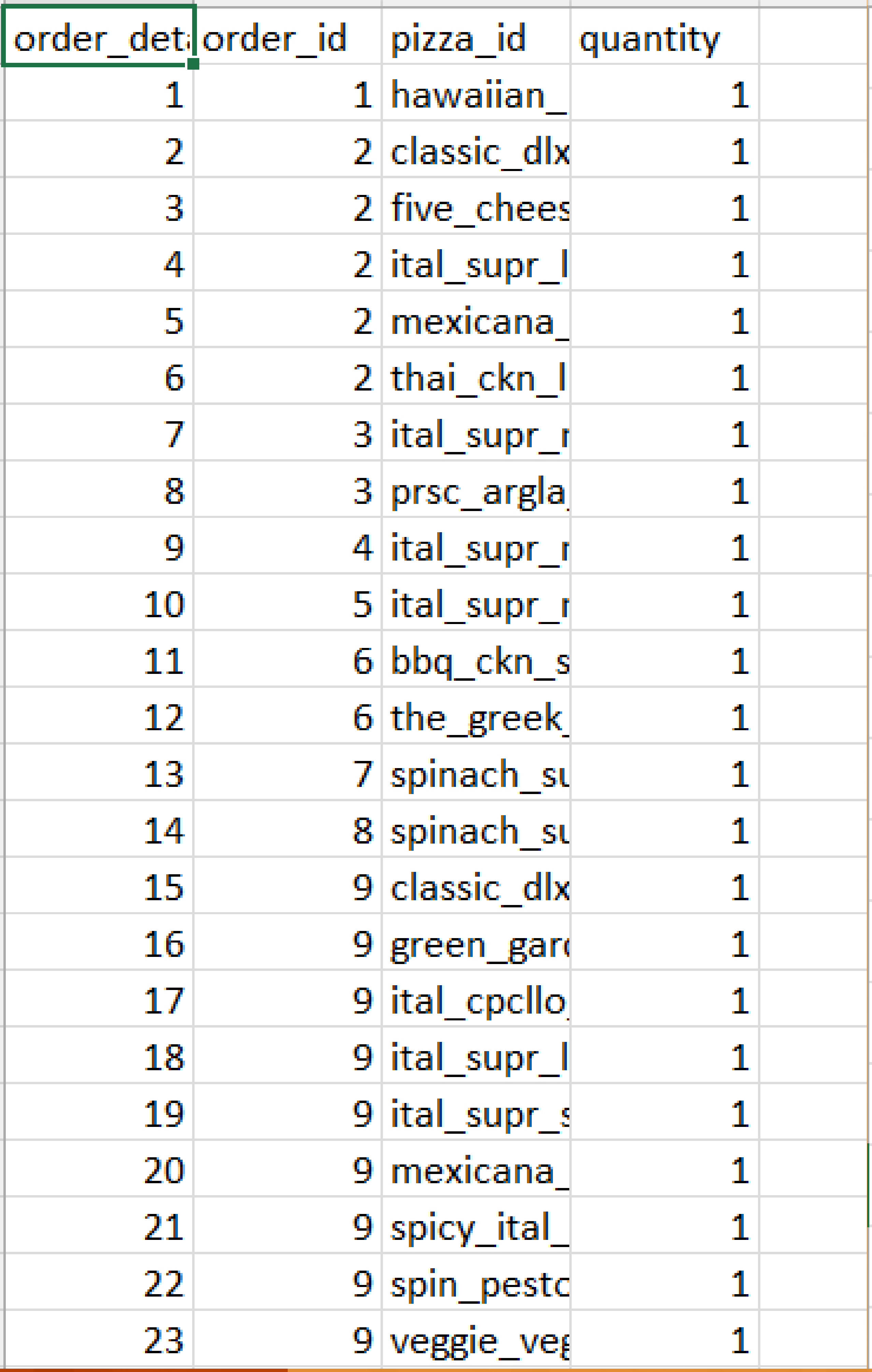
SQL PROJECT ON PIZZA_SALESE



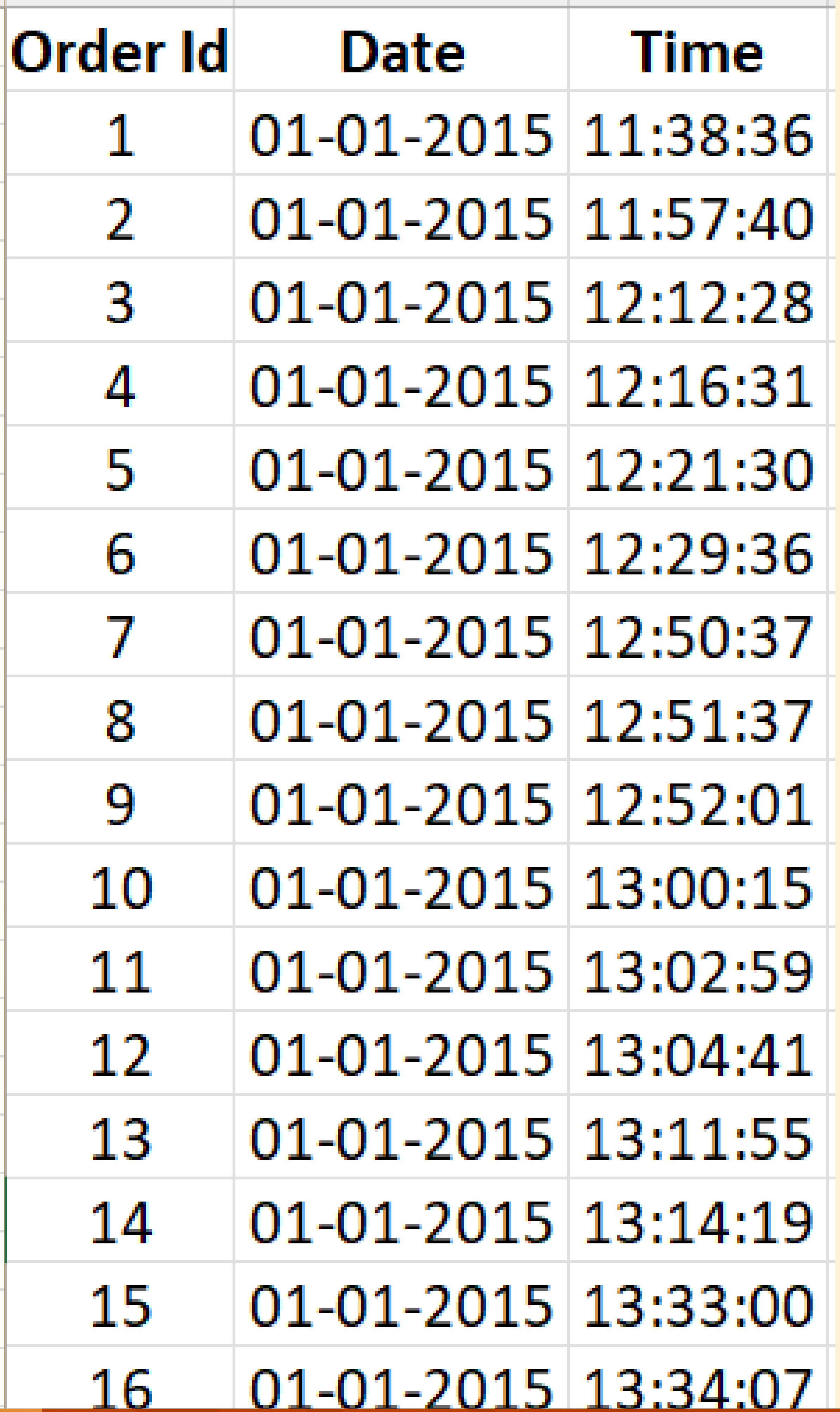
HELLO

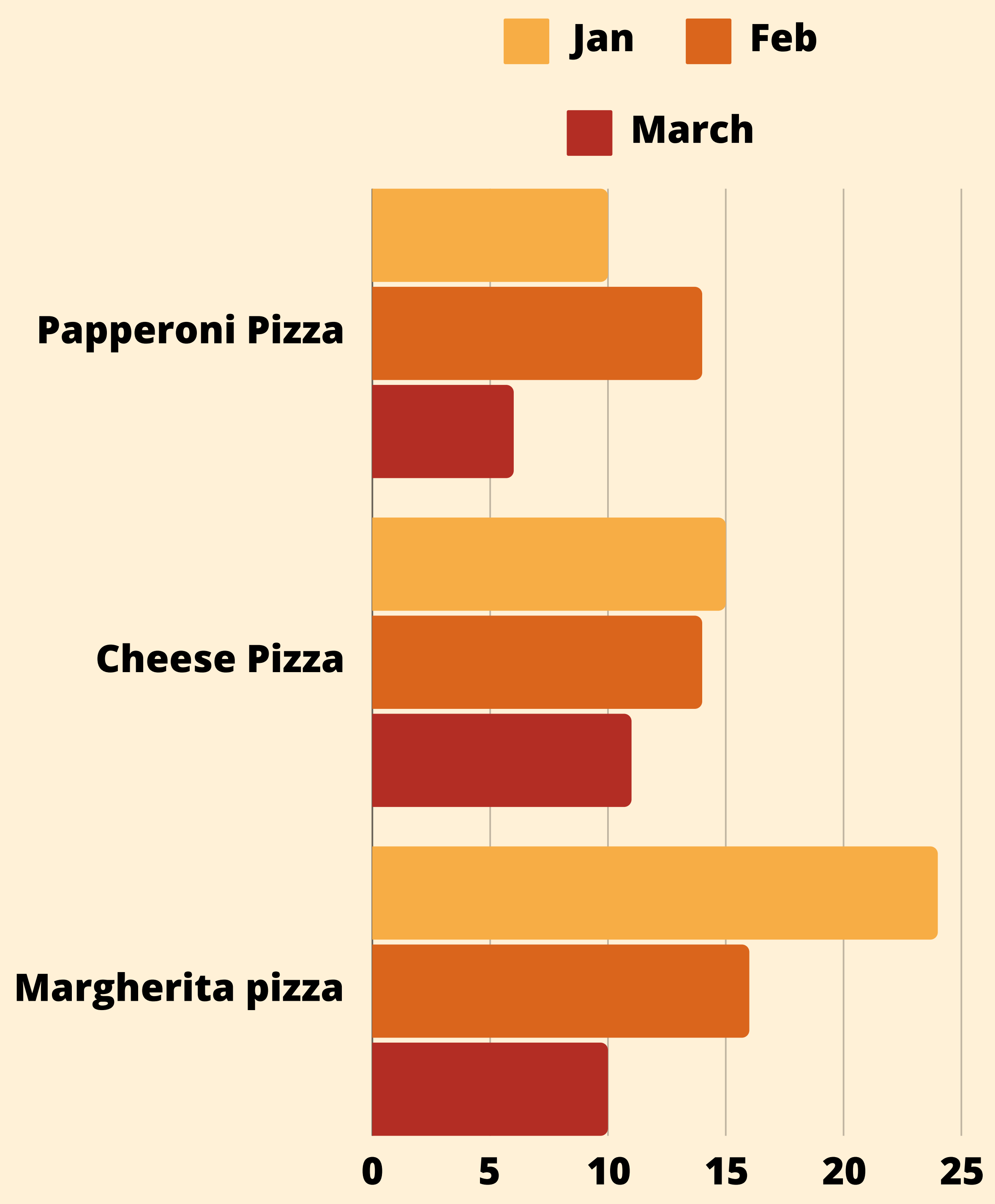
"My name is Raul Chakraborty, and this SQL project focuses on pizza sales. I designed a database to manage and analyze sales data, using complex queries to track orders, customer preferences, and inventory. The project optimized data retrieval to provide actionable insights for improving business performance."



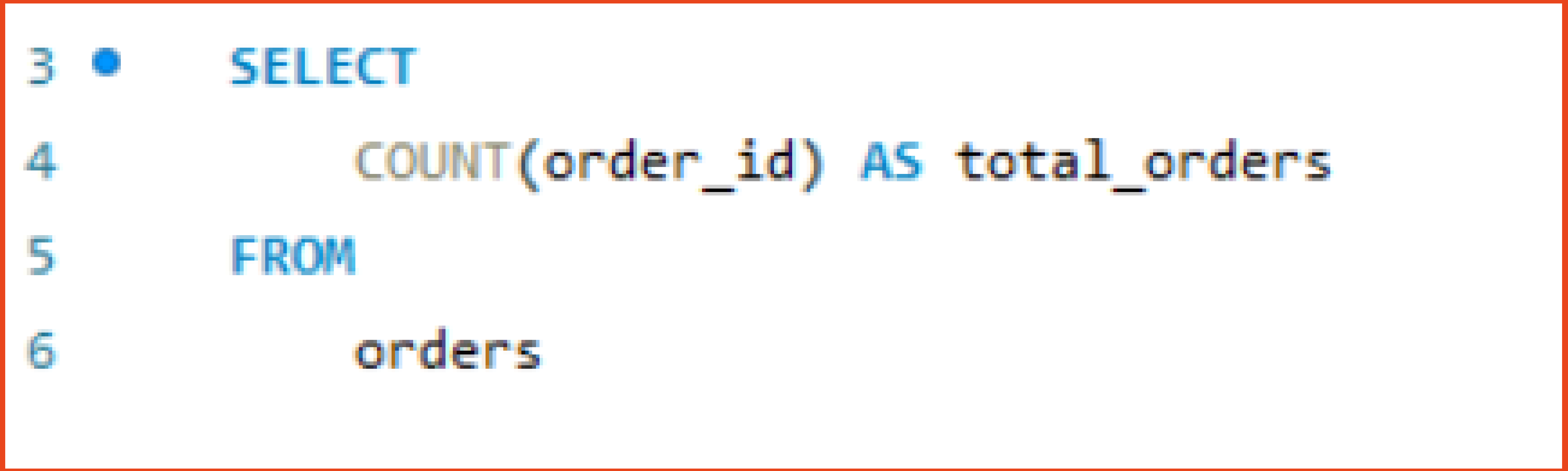
order_det	order_id	pizza_id	quantity
1	1	hawaiian_	1
2	2	classic_dlx	1
3	2	five_chees	1
4	2	ital_supr_l	1
5	2	mexicana_	1
6	2	thai_ckn_l	1
7	3	ital_supr_r	1
8	3	prsc_argla	1
9	4	ital_supr_r	1
10	5	ital_supr_r	1
11	6	bbq_ckn_s	1
12	6	the_greek_	1
13	7	spinach_su	1
14	8	spinach_su	1
15	9	classic_dlx	1
16	9	green_gar	1
17	9	ital_cpcllo	1
18	9	ital_supr_l	1
19	9	ital_supr_s	1
20	9	mexicana_	1
21	9	spicy_ital_	1
22	9	spin_pestc	1
23	9	veggie_veg	1



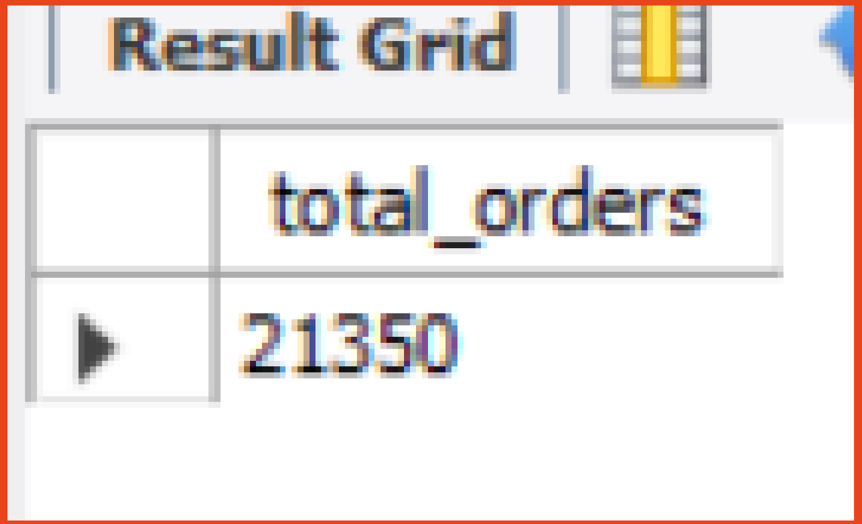
Order Id	Date	Time
1	01-01-2015	11:38:36
2	01-01-2015	11:57:40
3	01-01-2015	12:12:28
4	01-01-2015	12:16:31
5	01-01-2015	12:21:30
6	01-01-2015	12:29:36
7	01-01-2015	12:50:37
8	01-01-2015	12:51:37
9	01-01-2015	12:52:01
10	01-01-2015	13:00:15
11	01-01-2015	13:02:59
12	01-01-2015	13:04:41
13	01-01-2015	13:11:55
14	01-01-2015	13:14:19
15	01-01-2015	13:33:00
16	01-01-2015	13:34:07



Retrieve the total number of orders placed.

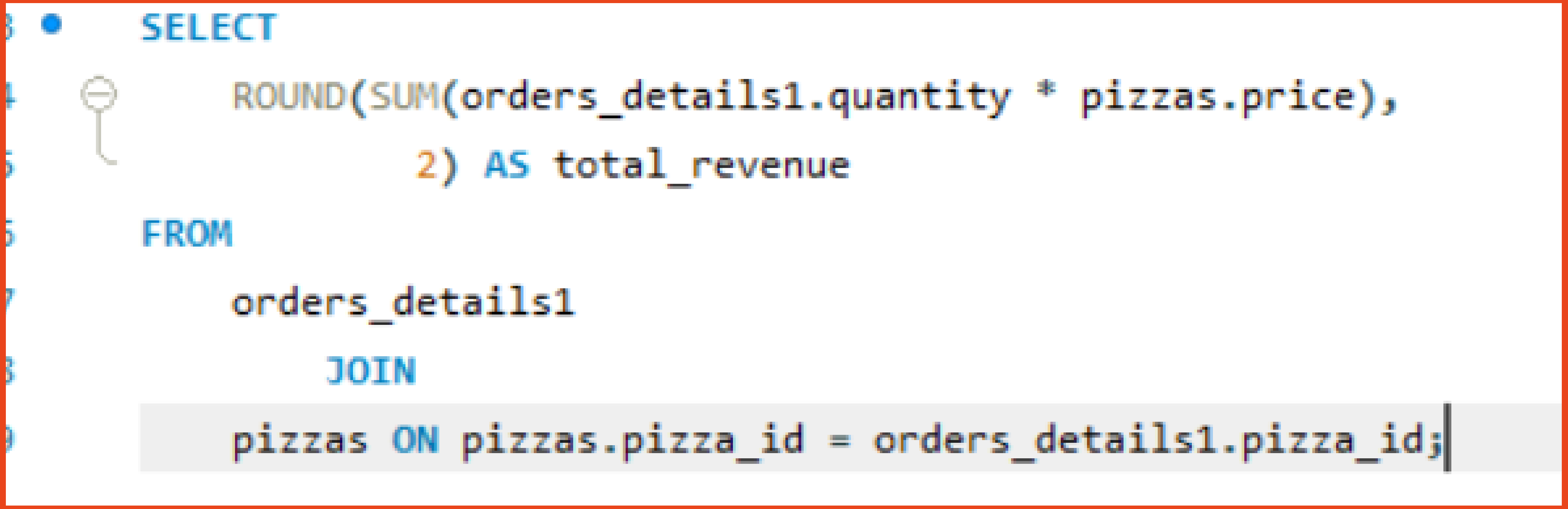


```
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders
```

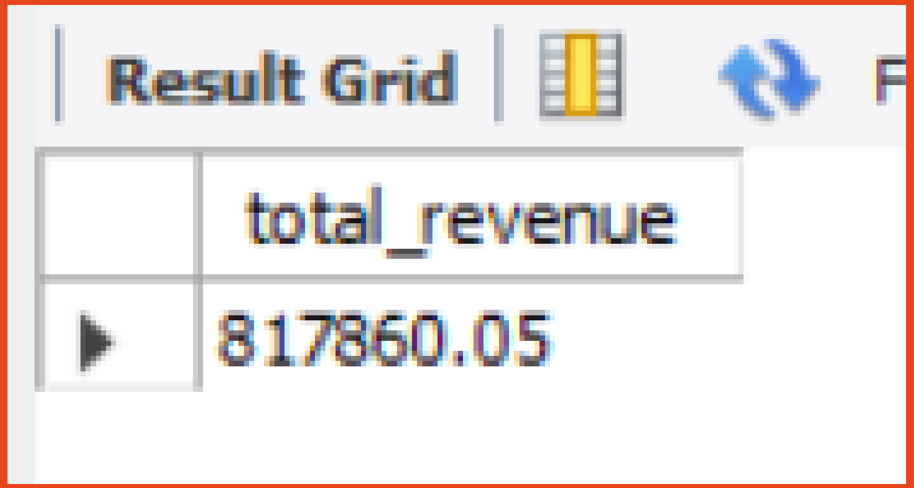


```
Result Grid
total_orders
21350
```

Calculate the total revenue generated from.

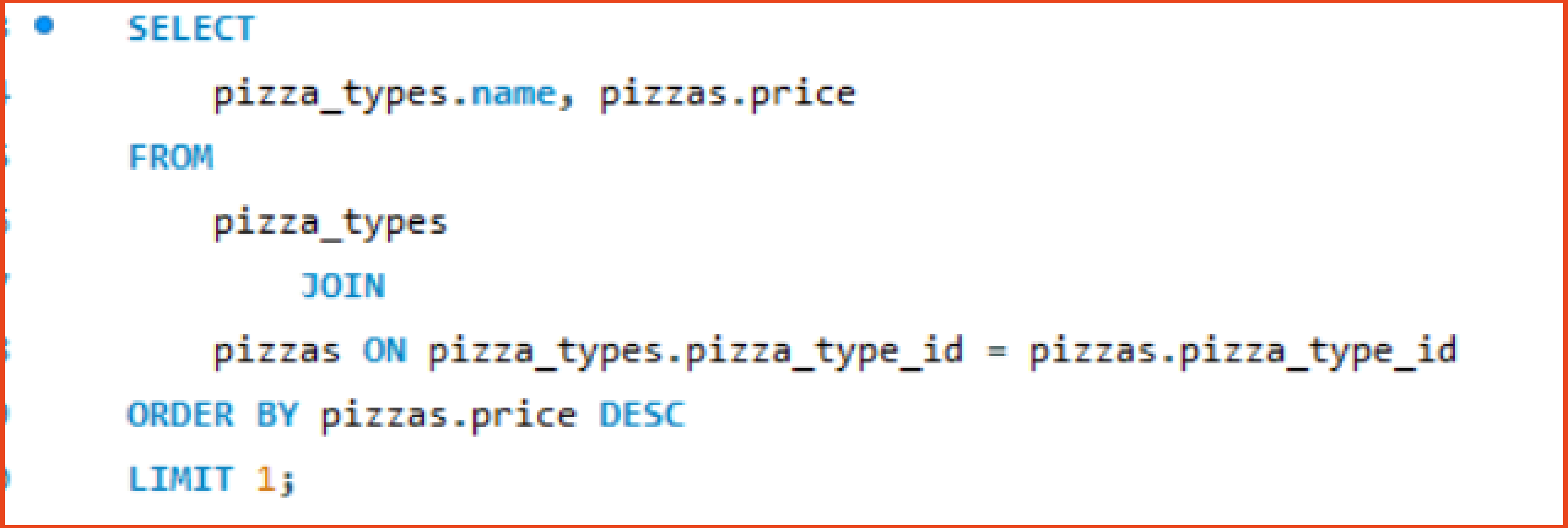


```
SELECT
    ROUND(SUM(orders_details1.quantity * pizzas.price),
            2) AS total_revenue
FROM
    orders_details1
        JOIN
    pizzas ON pizzas.pizza_id = orders_details1.pizza_id;
```

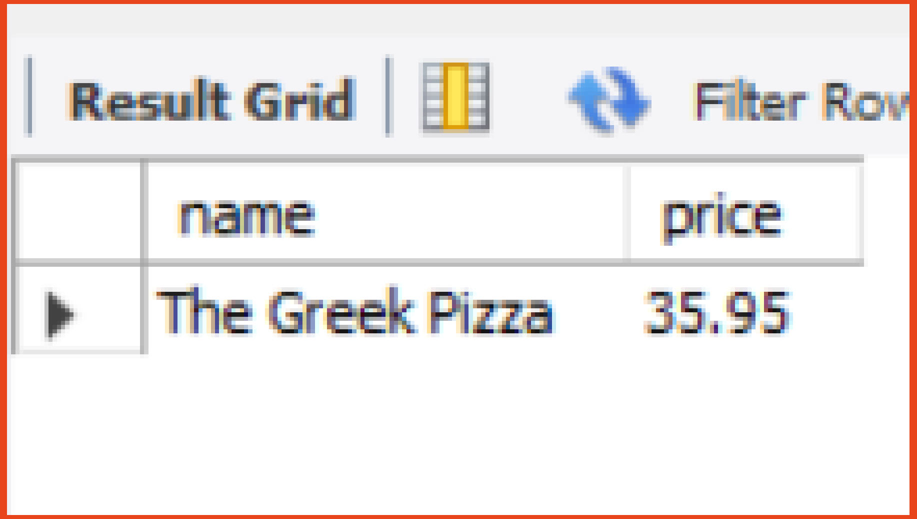


```
total_revenue
817860.05
```

Identify the highest-priced pizza.

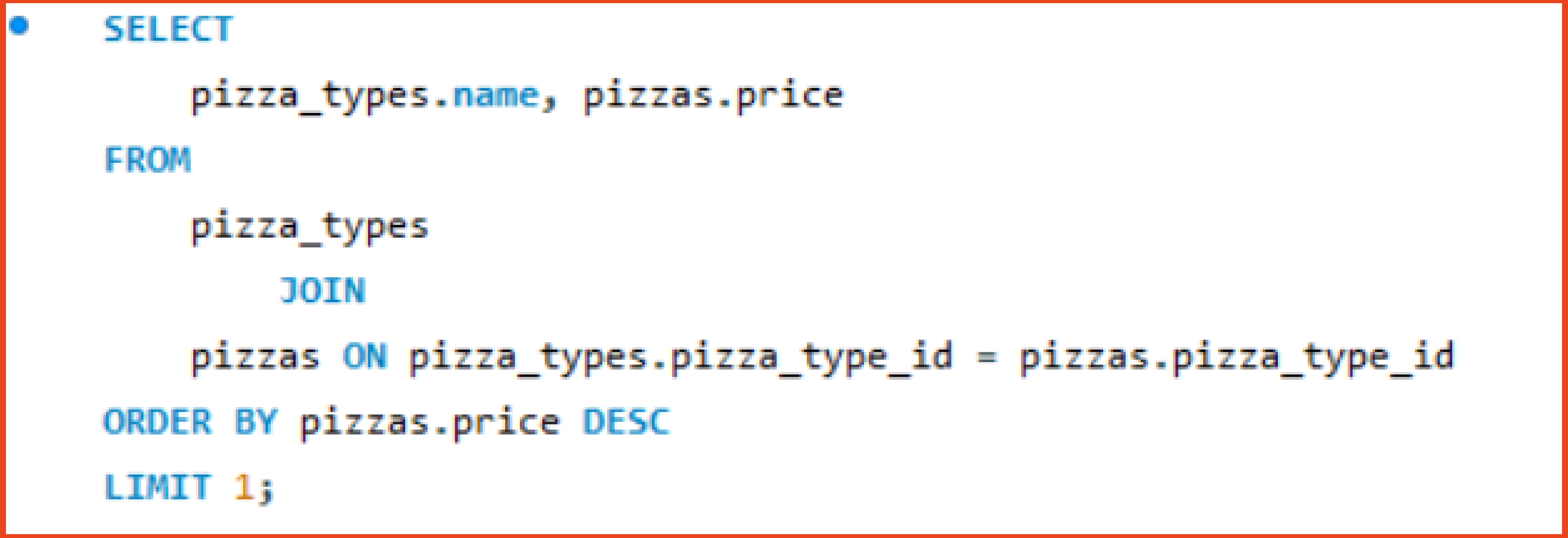


```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

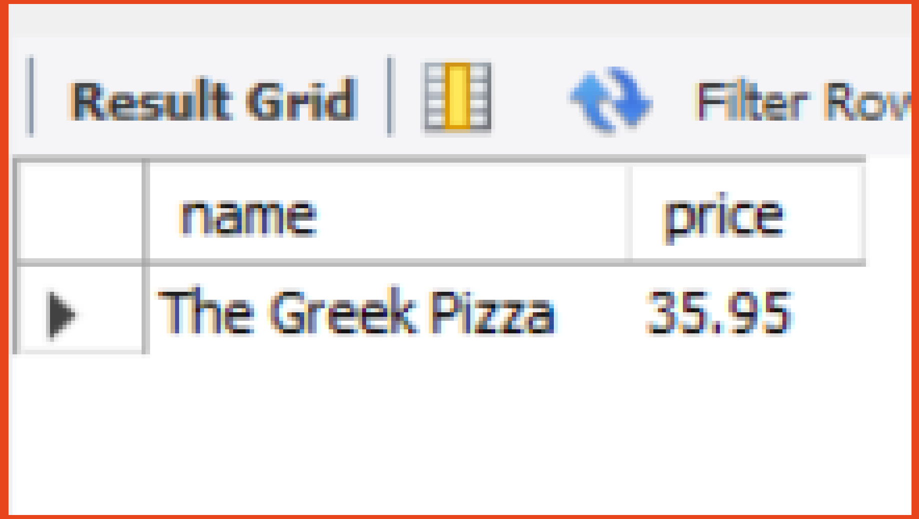


```
name             price
The Greek Pizza  35.95
```

Identify the most common pizza size ordered.

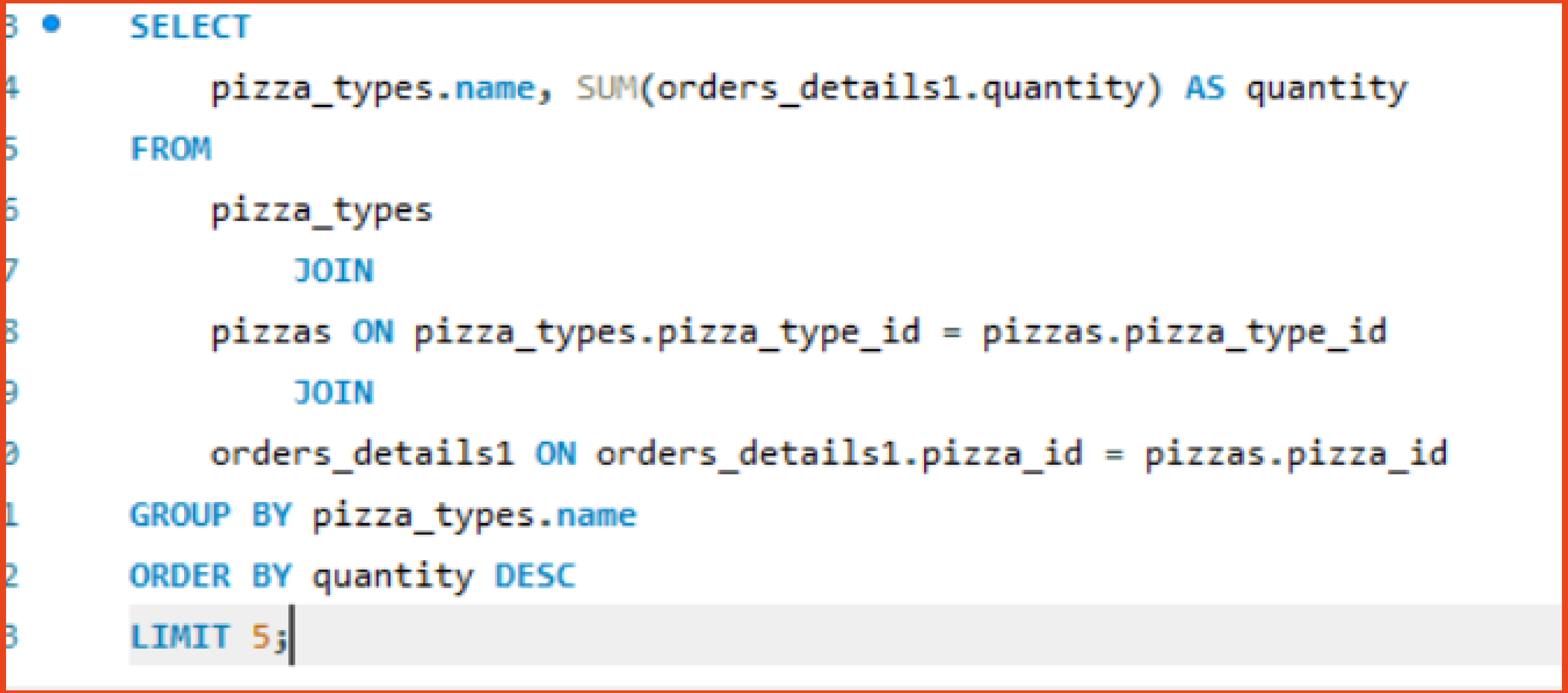


```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

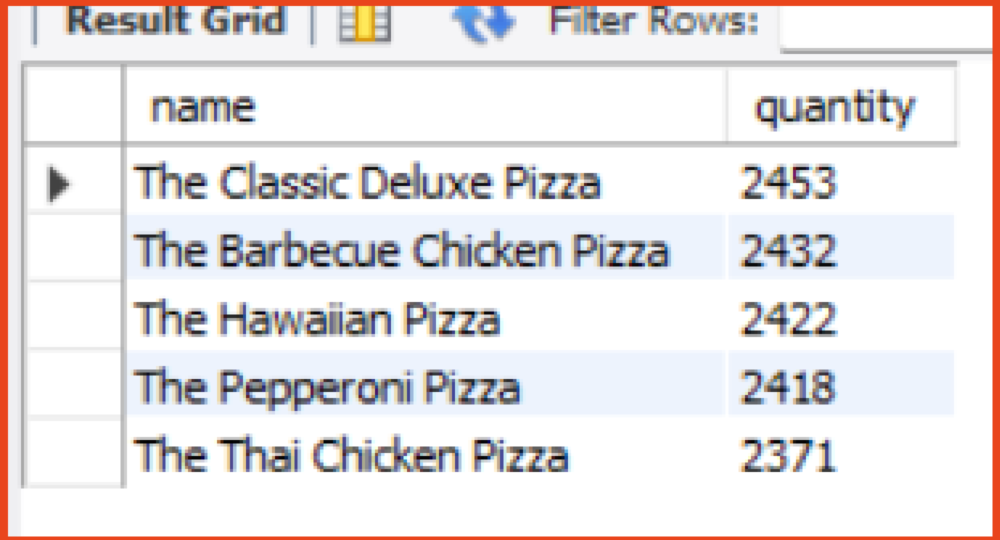


```
name             price
The Greek Pizza  35.95
```

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

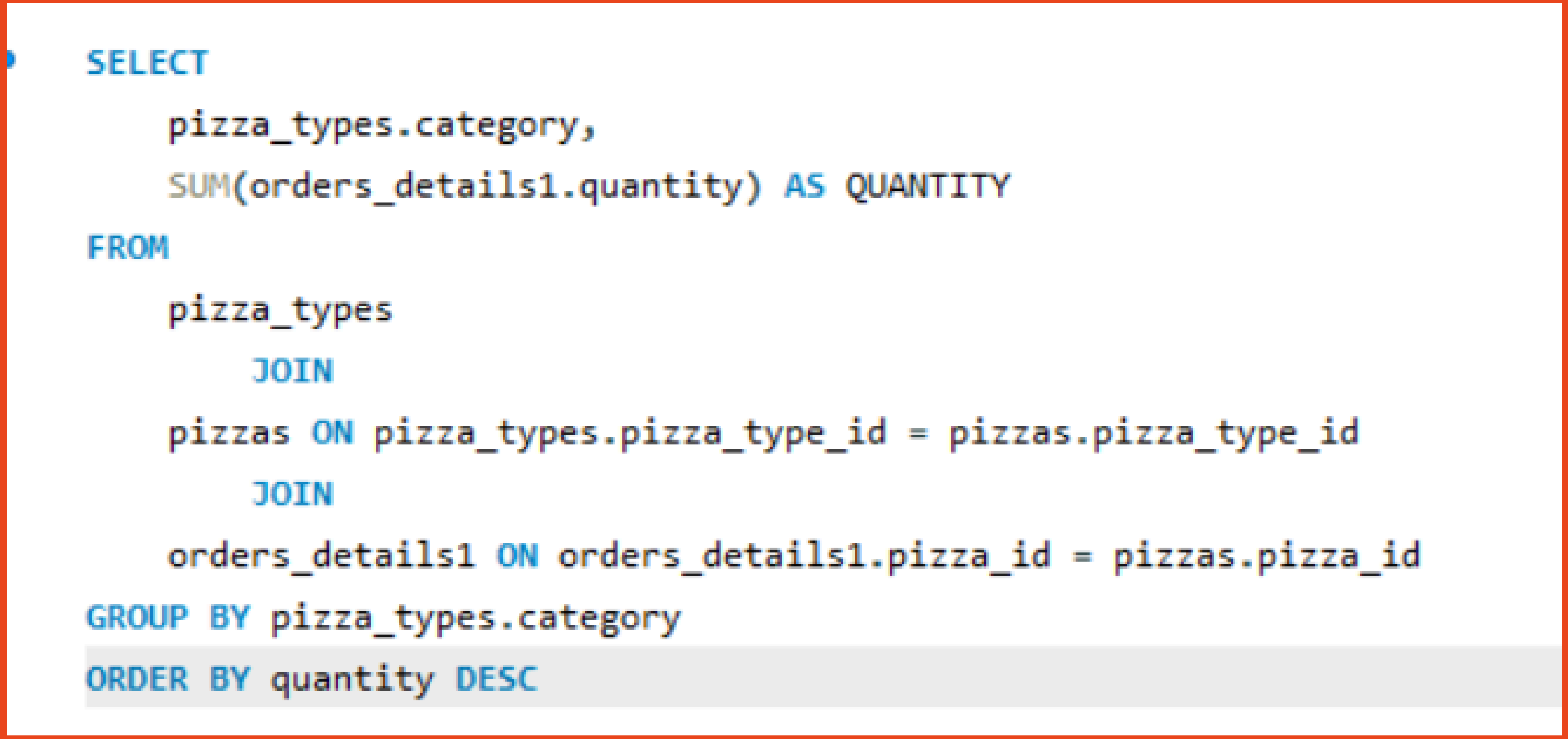


```sql
SELECT
    pizza_types.name, SUM(orders_details1.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details1 ON orders_details1.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

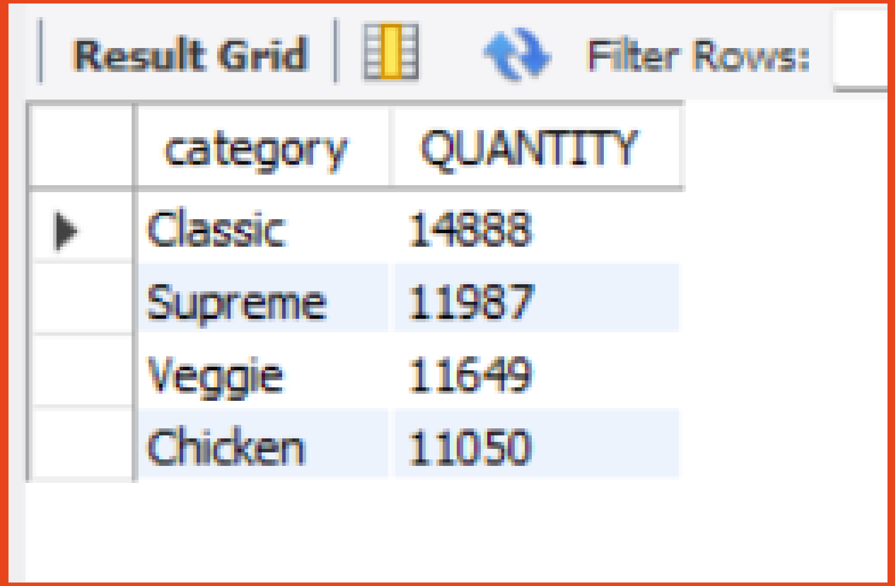


name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

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



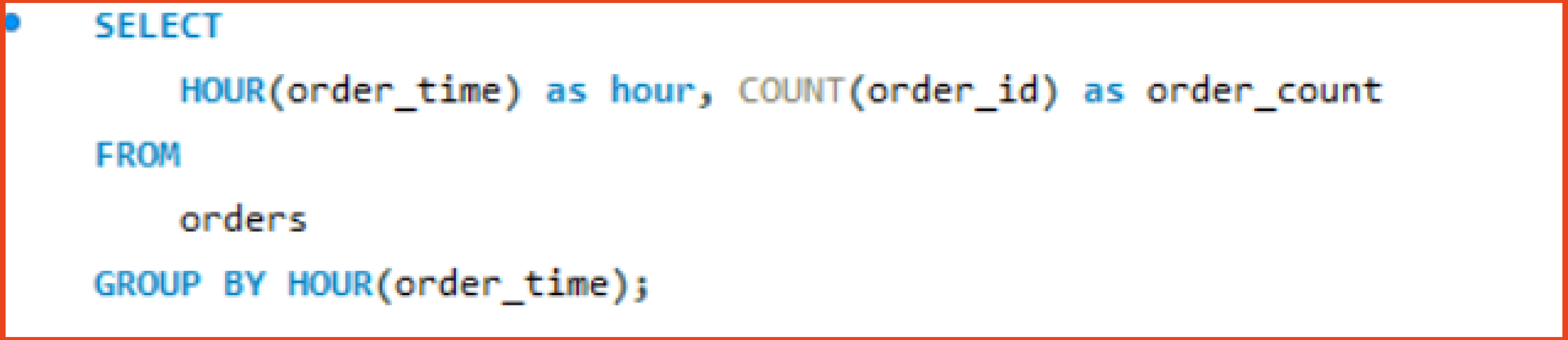
```
SELECT
    pizza_types.category,
    SUM(orders_details1.quantity) AS QUANTITY
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details1 ON orders_details1.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC
```



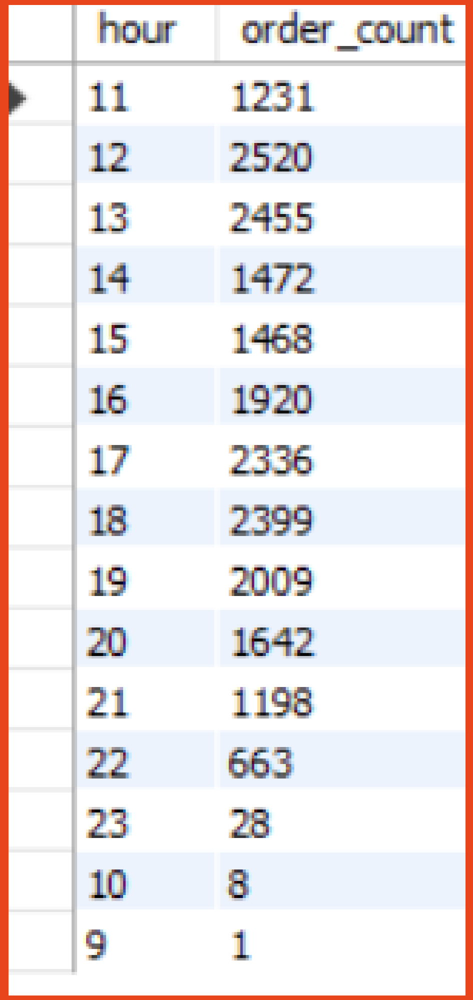
Result Grid | Filter Rows:

category	QUANTITY
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

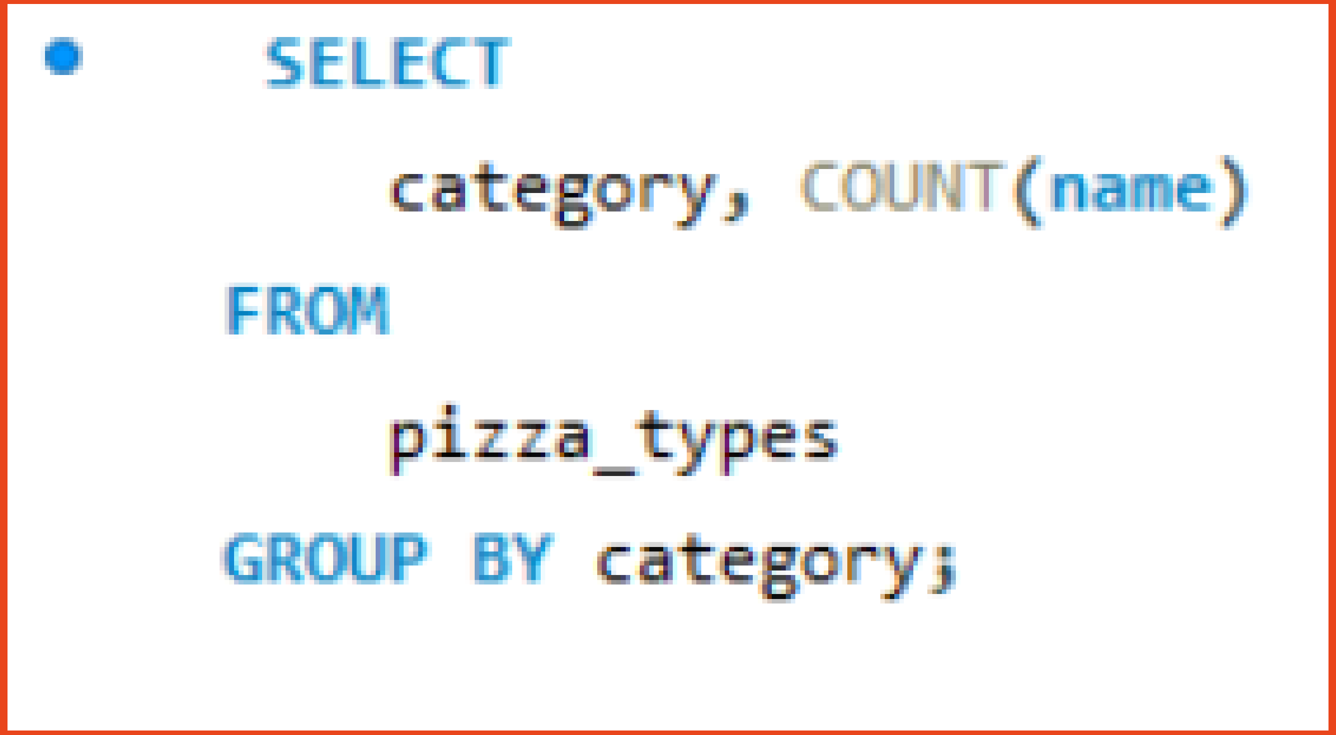


```
SELECT
    HOUR(order_time) as hour, COUNT(order_id) as order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

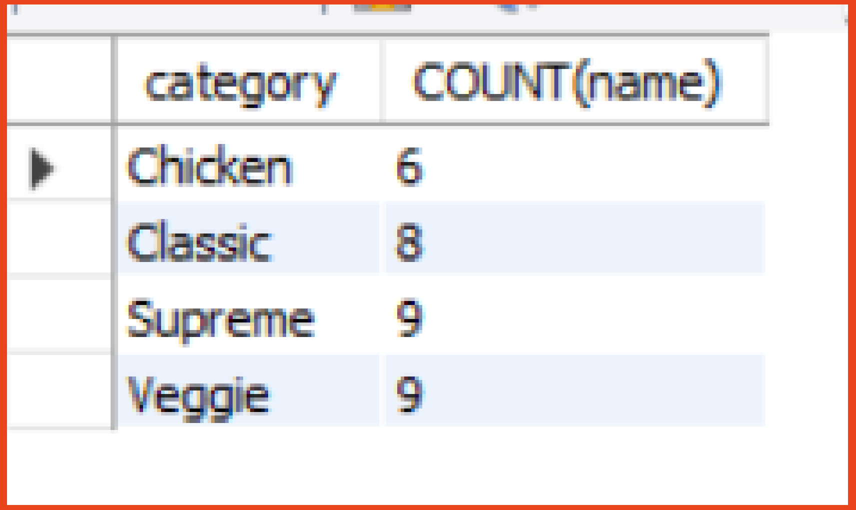


hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

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

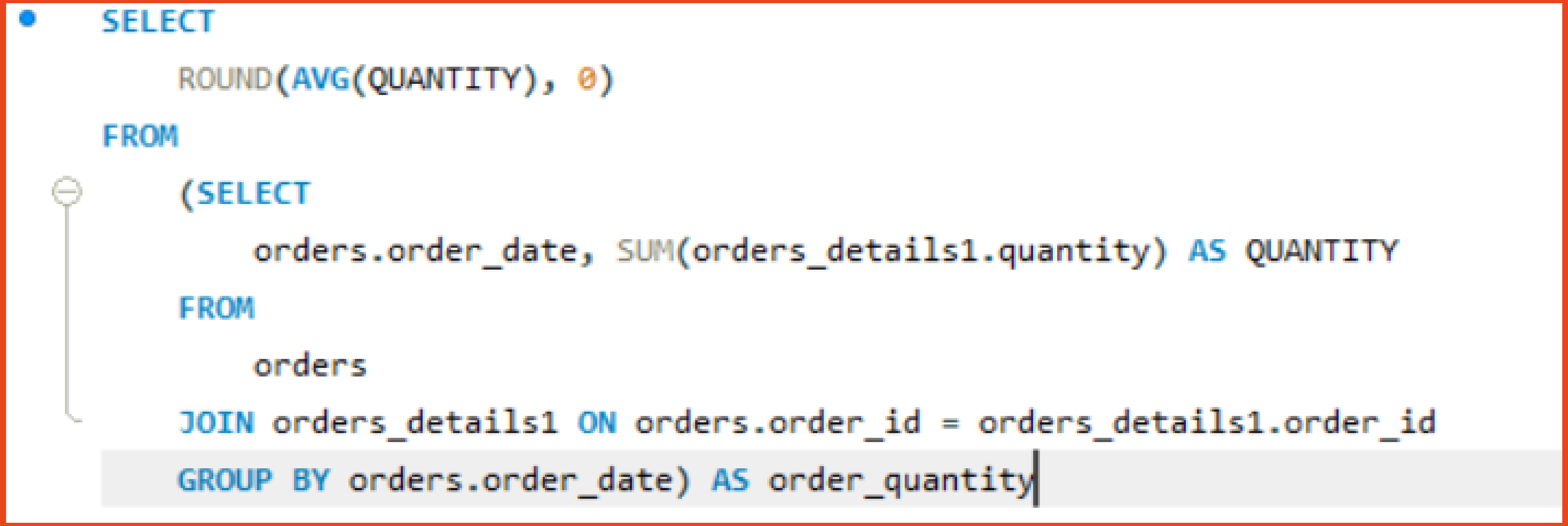


```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

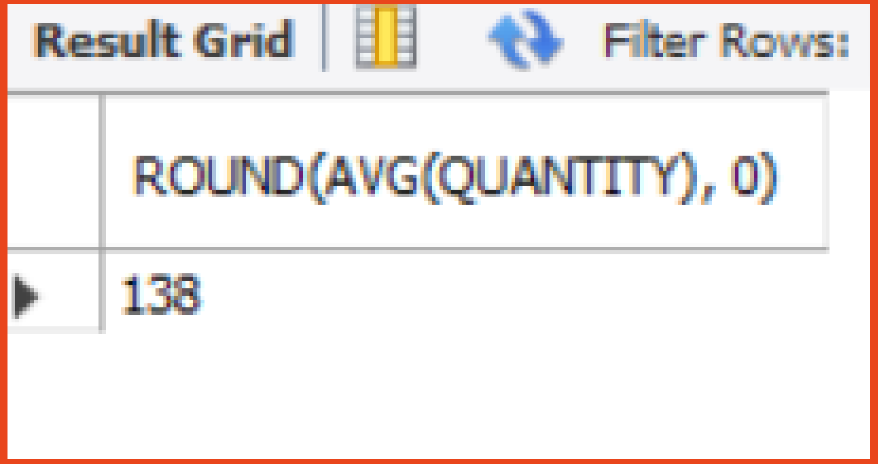


category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

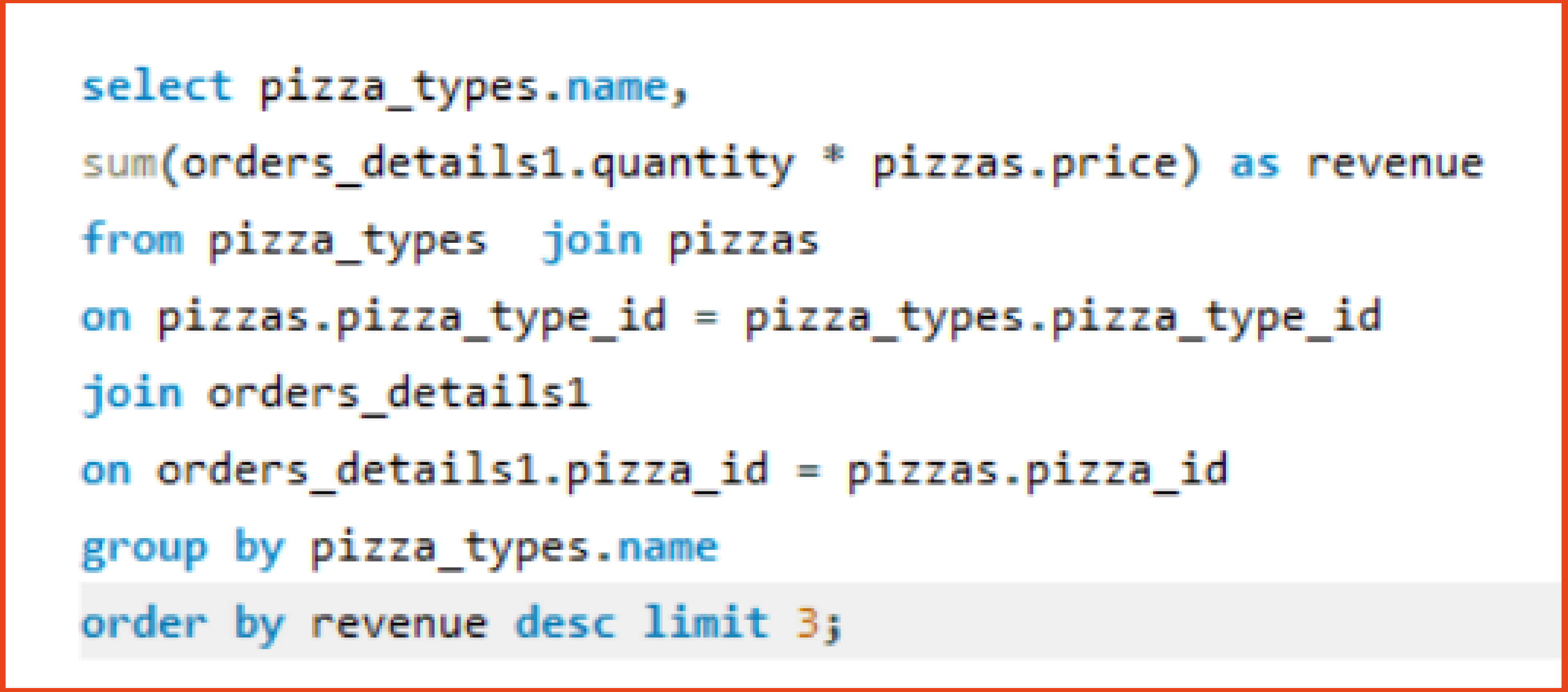


```
SELECT
    ROUND(AVG(QUANTITY), 0)
FROM
    (SELECT
        orders.order_date, SUM(orders_details1.quantity) AS QUANTITY
    FROM
        orders
    JOIN orders_details1 ON orders.order_id = orders_details1.order_id
    GROUP BY orders.order_date) AS order_quantity
```

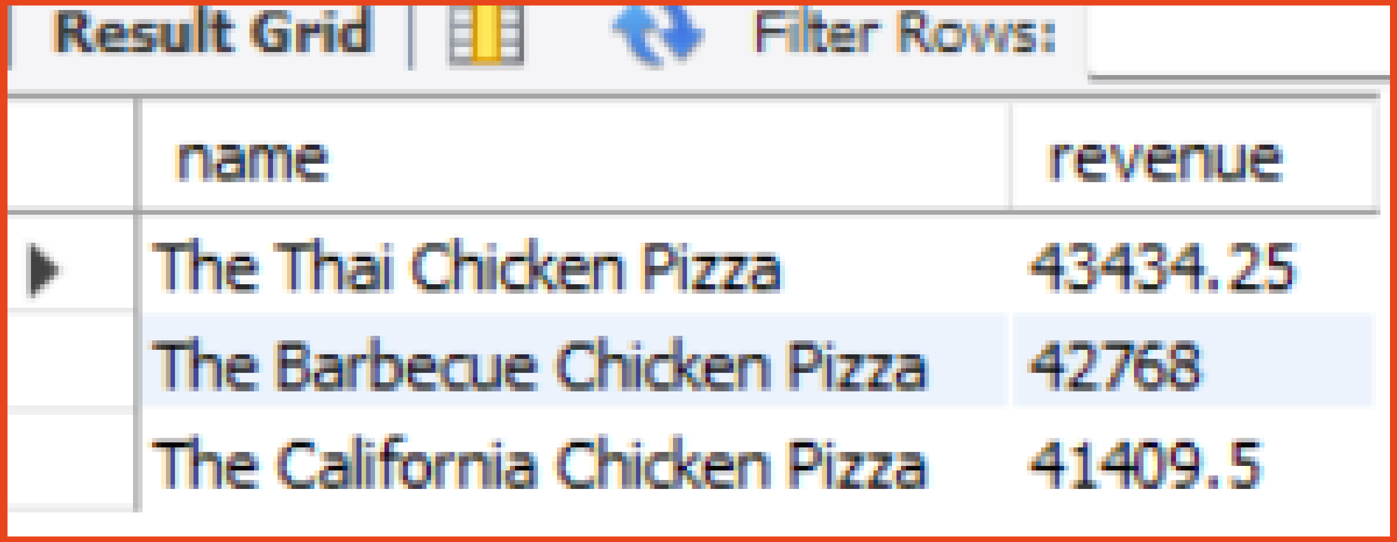


```
ROUND(AVG(QUANTITY), 0)
138
```

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

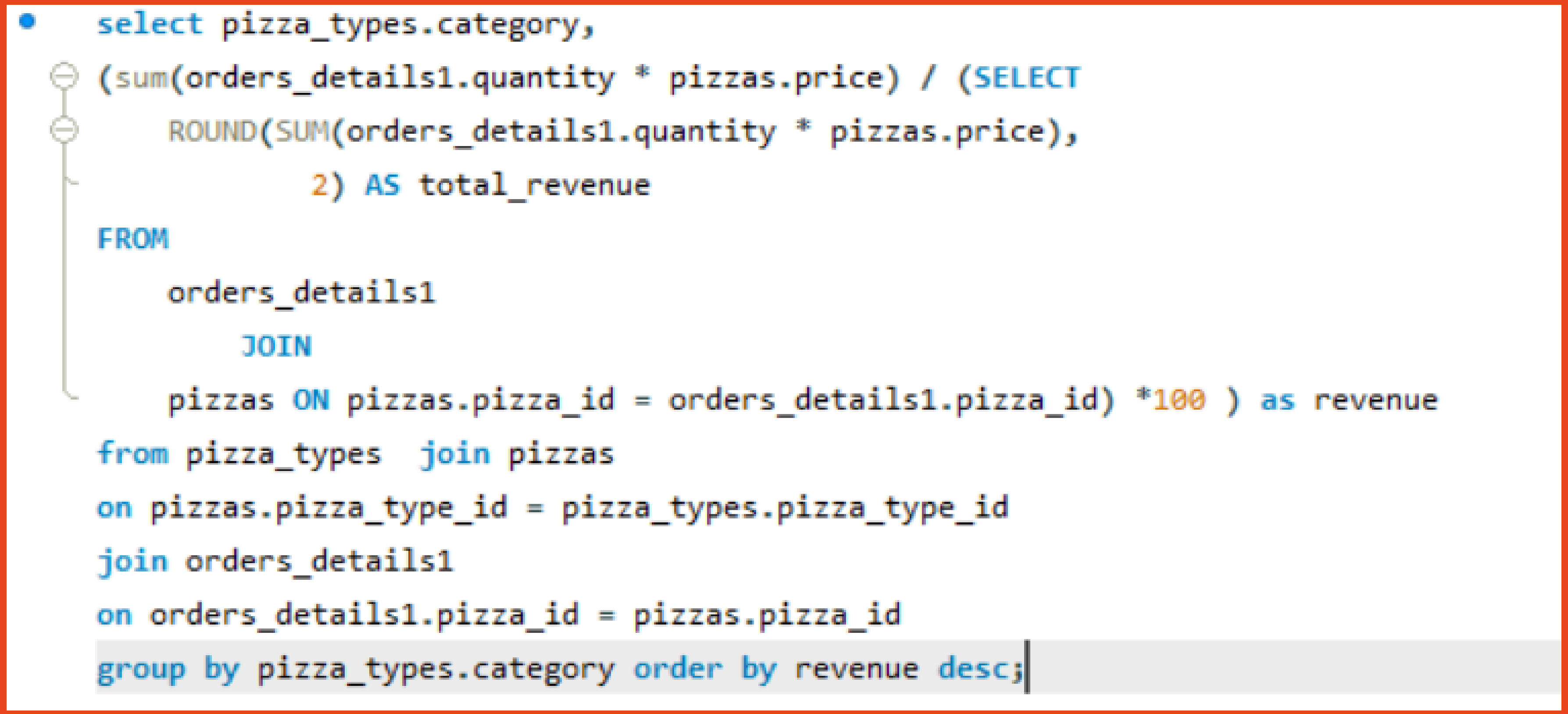


```
select pizza_types.name,
sum(orders_details1.quantity * pizzas.price) as revenue
from pizza_types  join pizzas
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join orders_details1
on orders_details1.pizza_id = pizzas.pizza_id
group by pizza_types.name
order by revenue desc limit 3;
```

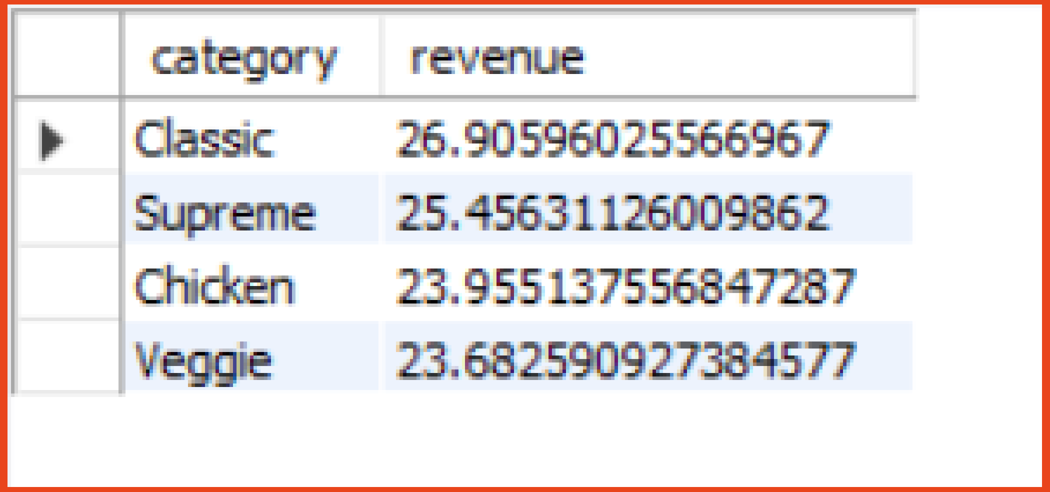


name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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

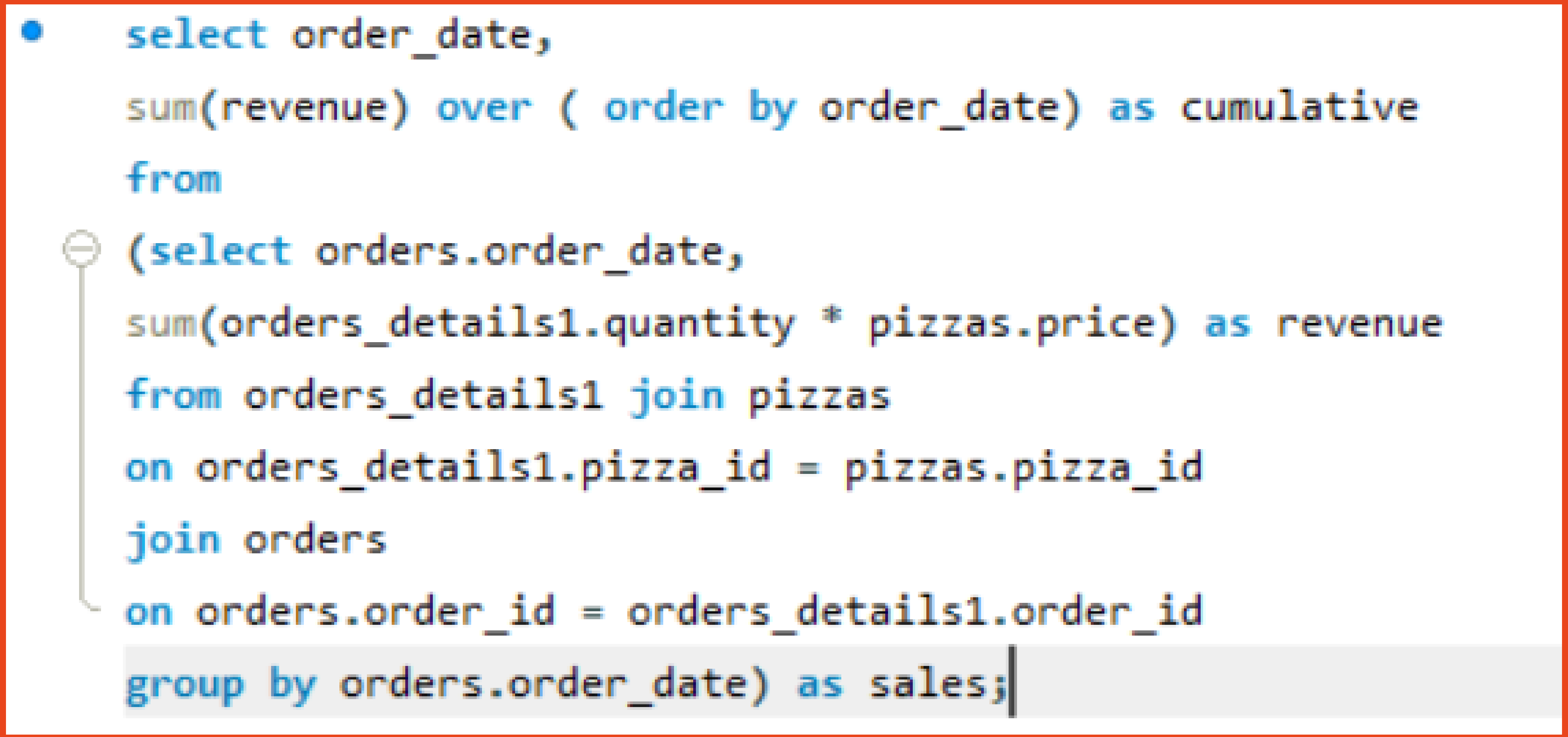


```
select pizza_types.category,
(sum(orders_details1.quantity * pizzas.price) / (SELECT
    ROUND(SUM(orders_details1.quantity * pizzas.price),
            2) AS total_revenue
FROM
    orders_details1
        JOIN
    pizzas ON pizzas.pizza_id = orders_details1.pizza_id) *100 ) as revenue
from pizza_types  join pizzas
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join orders_details1
on orders_details1.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

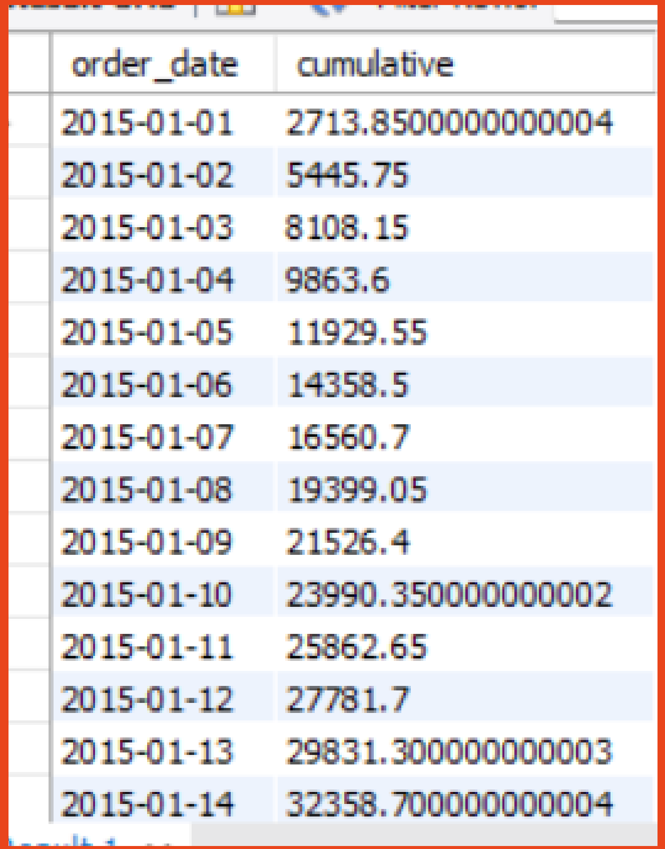


category	revenue
Classic	26.90596025566967
Supreme	25.45631126009862
Chicken	23.955137556847287
Veggie	23.682590927384577

Analyze the cumulative revenue generated over time.



```
select order_date,
sum(revenue) over ( order by order_date) as cumulative
from
(select orders.order_date,
sum(orders_details1.quantity * pizzas.price) as revenue
from orders_details1 join pizzas
on orders_details1.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = orders_details1.order_id
group by orders.order_date) as sales;
```



order_date	cumulative
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
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004

THANK YOU