VEGGIE PIZZA

VEGGIE PIZZA

PIZZA SALES SQL PROJECT

SOLVING REAL-WORLD BUSINESS QUESTIONS USING SQL

SHIVANI PAWAR

VEGGIE PIZZA

DATASET: PIZZA SALES (ORDERS,
PIZZAS,
PIZZA_TYPES,
ORDER_DETAILS)

TOOLS USED: MYSQL

GOAL: DATA ANALYSIS USING SQL

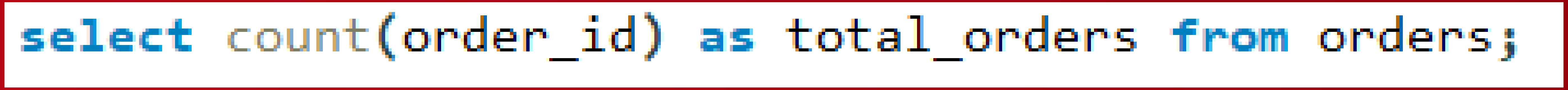
BASIC QUESTIONS

NO.	QUESTION
1	TOTAL NUMBER OF ORDERS PLACED
2	TOTAL REVENUE GENERATED
3	HIGHEST PRICED PIZZA
4	MOST COMMON PIZZA SIZE
5	TOP 5 PIZZAS BY QUANTITY

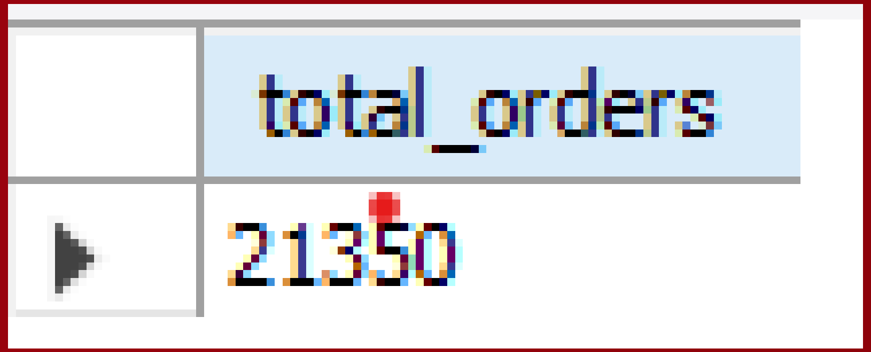
INTERMEDIATE QUESTONS

NO.	QUESTION
6	QUANTITY OF EACH CATEGORY
7	ORDER BY HOUR
8	CATEGORY-WISE PIZZA COUNT
9	AVG. PIZZAS PER DAY
10	TOP 3 PIZZAS BY REVENUE
11	% REVENUE BY PIZZA TYPE
12	CUMULATIVE REVENUE OVER TIME
13	TOP 3 PIZZAS BY REVENUE IN EACH CATEGORY

TOTAL NUMBER OF ORDERS PLACED.

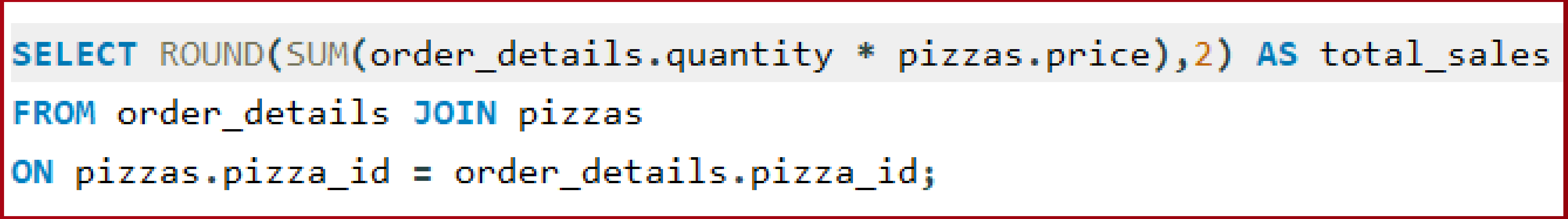


```
select count(order_id) as total_orders from orders;
```

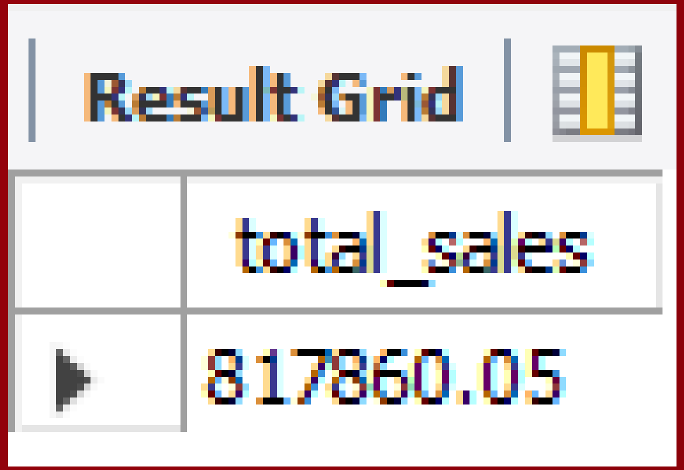


```
total_orders
21350
```

TOTAL REVENUE GENERATED.

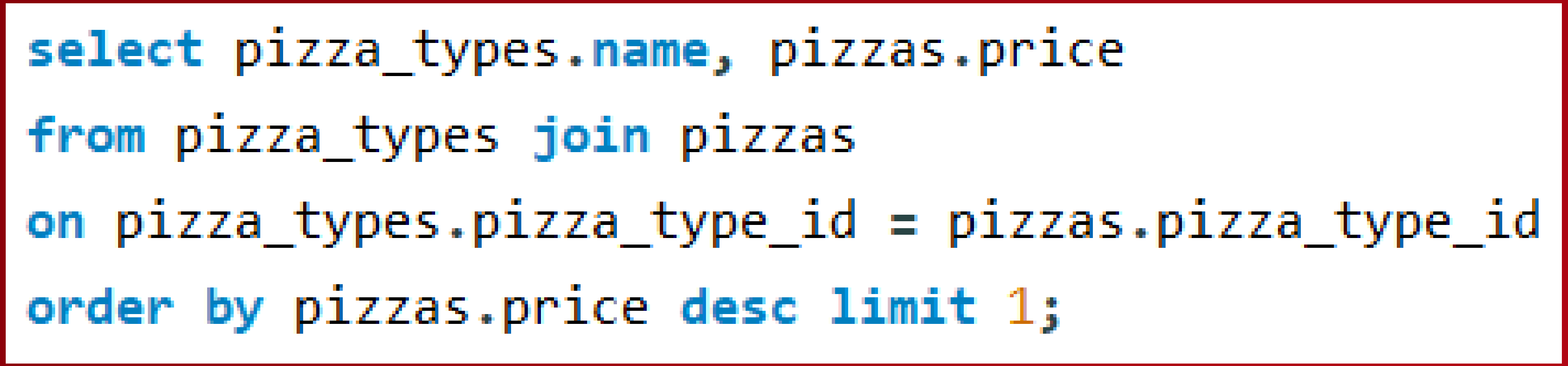


```
SELECT ROUND(SUM(order_details.quantity * pizzas.price),2) AS total_sales
FROM order_details JOIN pizzas
ON pizzas.pizza_id = order_details.pizza_id;
```

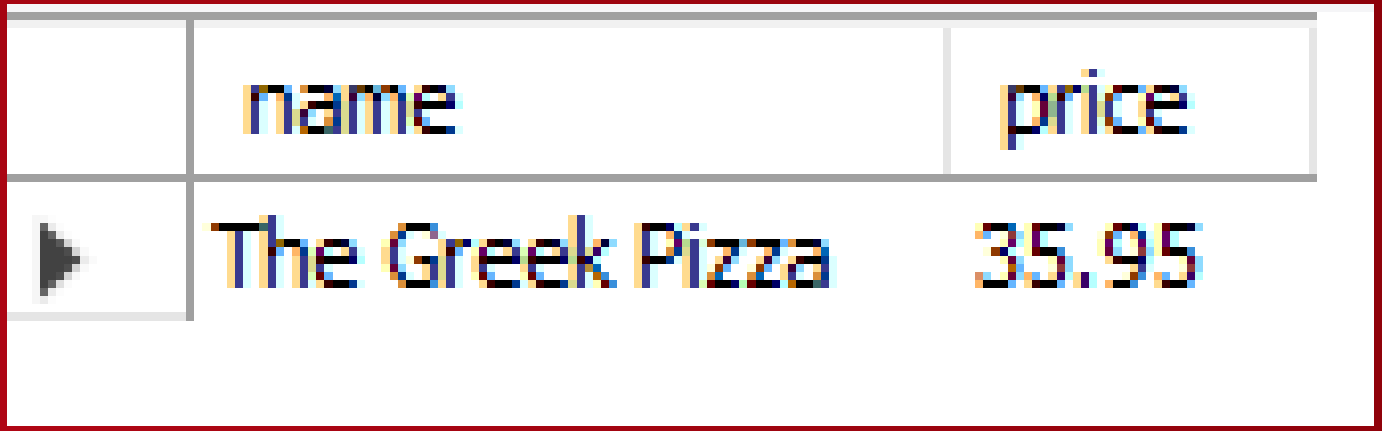


```
total_sales
817860.05
```

HIGHEST PRICED PIZZA

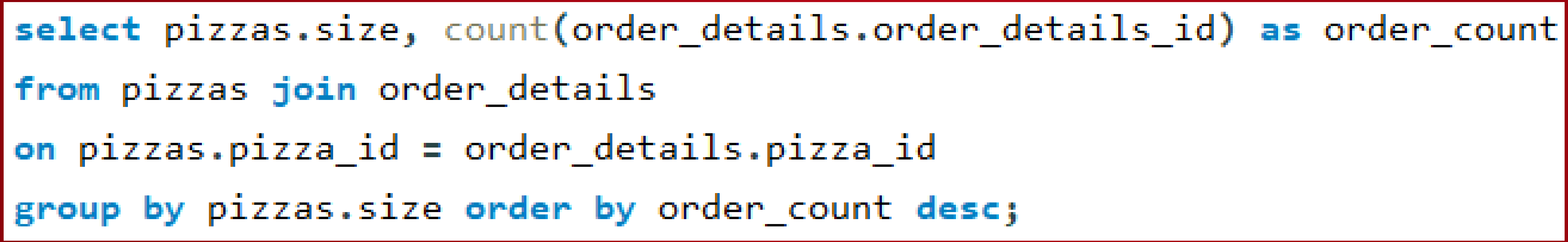


```
select pizza_types.name, pizzas.price
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
order by pizzas.price desc limit 1;
```

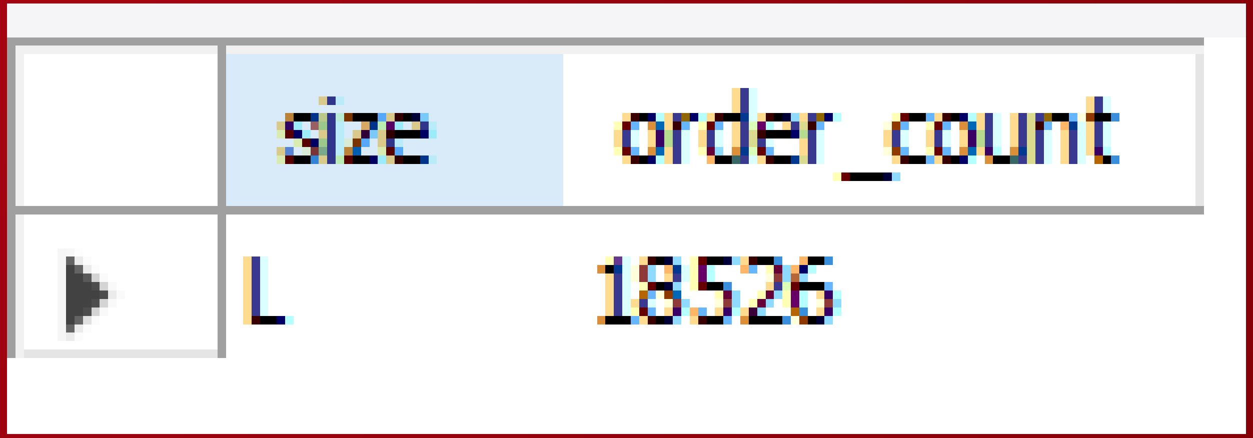


name	price
The Greek Pizza	35.95

MOST COMMON PIZZA SIZE

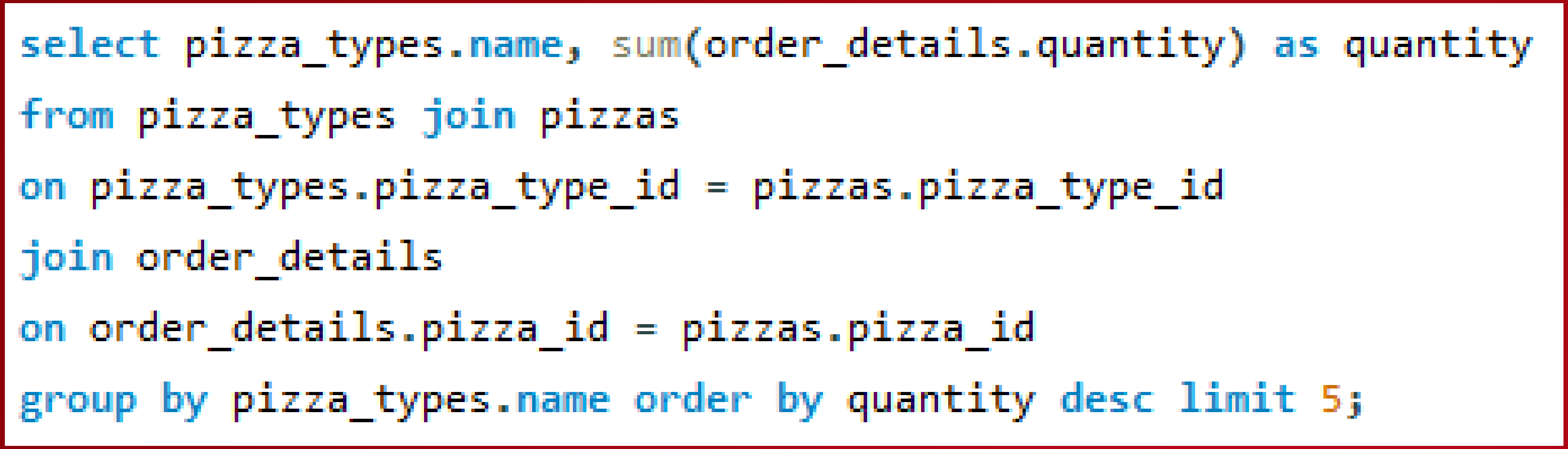


```
select pizzas.size, count(order_details.order_details_id) as order_count
from pizzas join order_details
on pizzas.pizza_id = order_details.pizza_id
group by pizzas.size order by order_count desc;
```

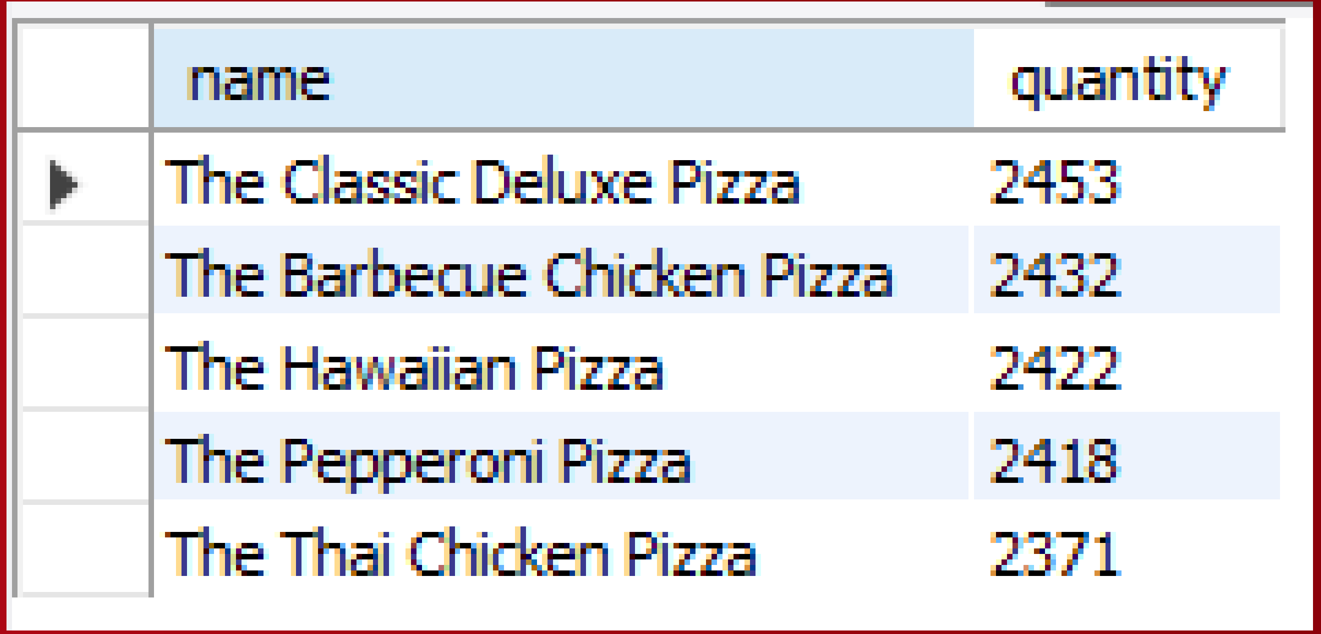


size	order_count
L	18526

TOP 5 PIZZAS BY QUANTITY

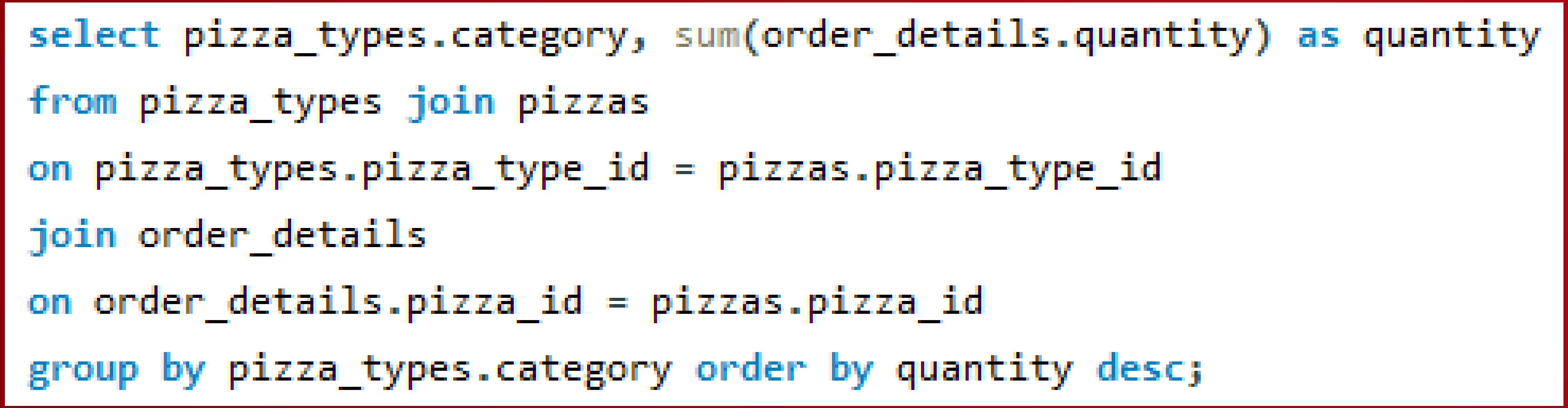


```
select pizza_types.name, sum(order_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name order by quantity desc limit 5;
```

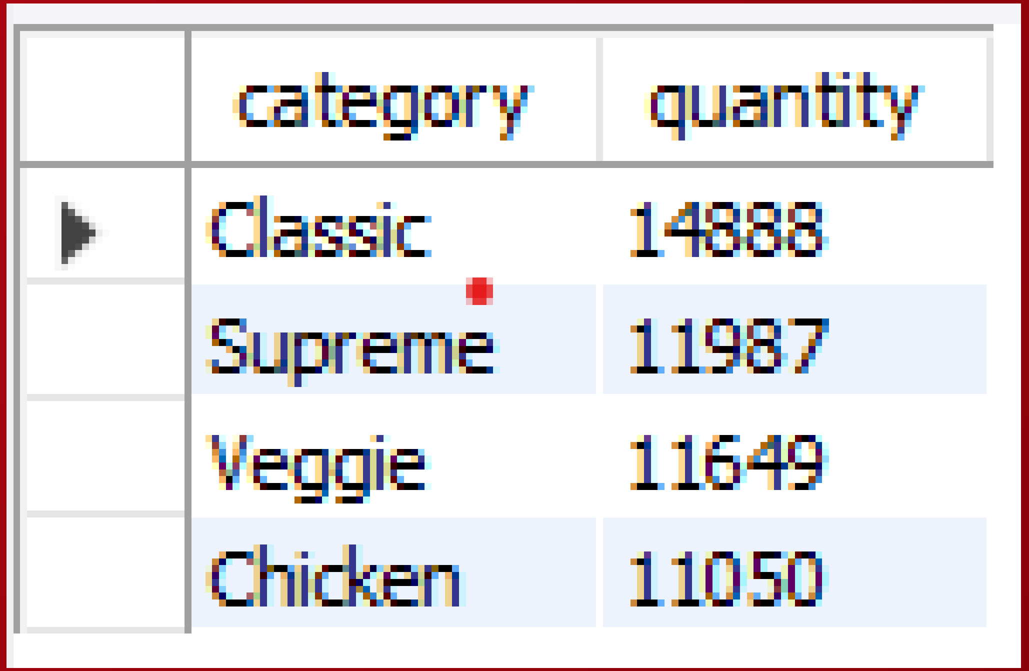


name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

QUANTITY OF EACH CATEGORY

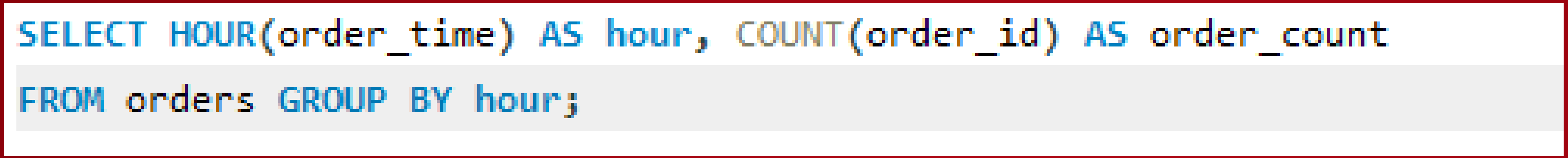


```
select pizza_types.category, sum(order_details.quantity) as quantity
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by quantity desc;
```

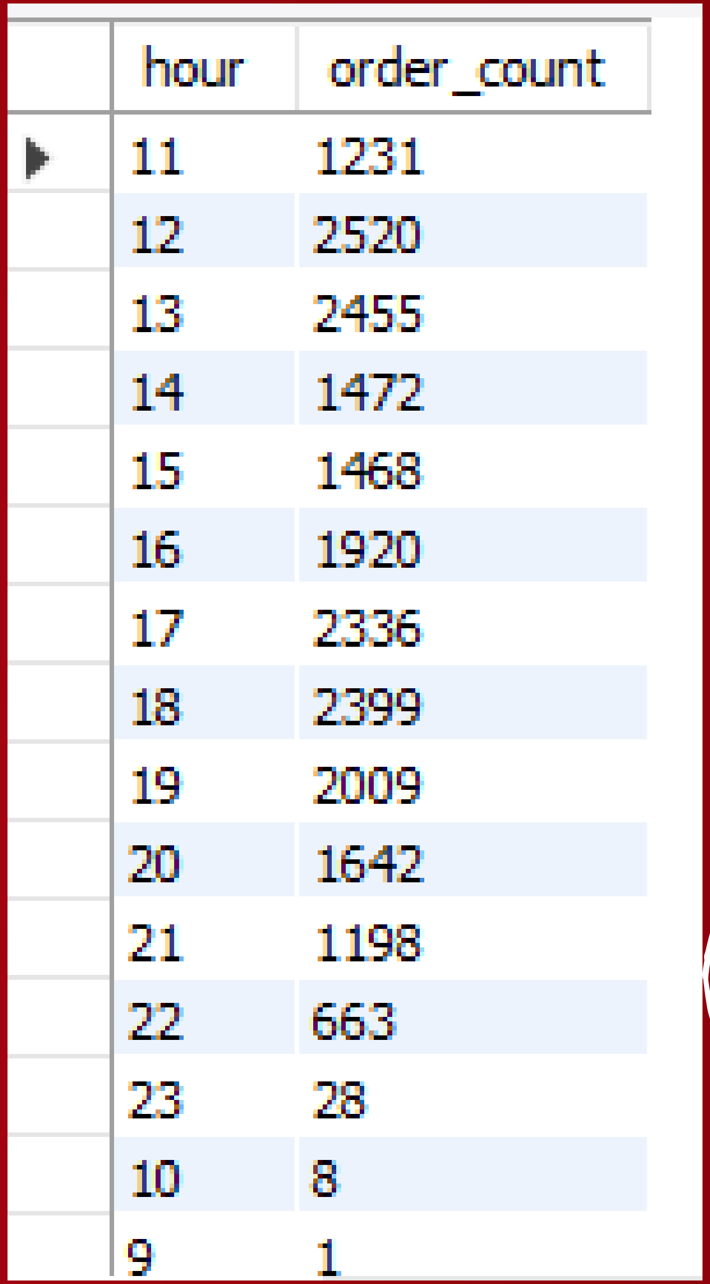


category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

ORDER BY HOUR

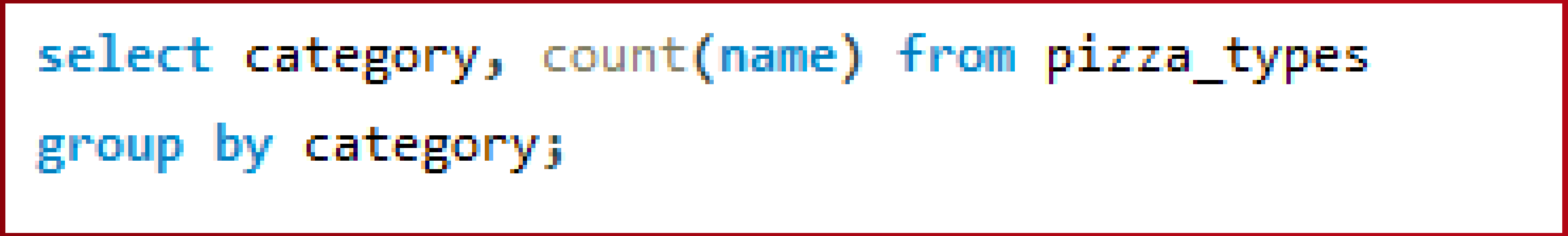


```
SELECT HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM orders GROUP BY hour;
```

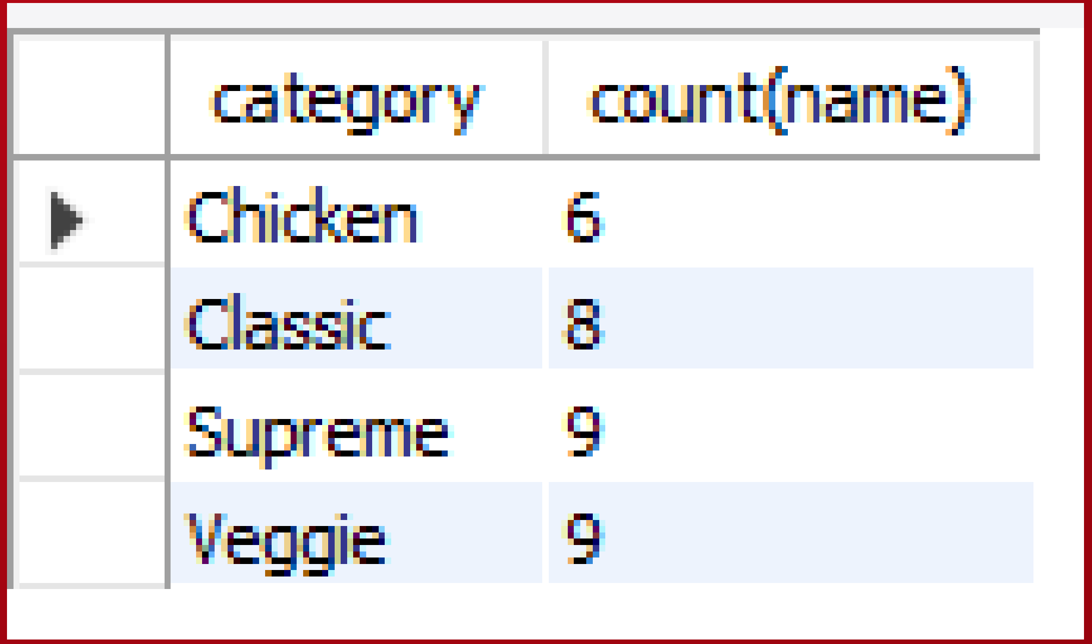


hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

CATEGORY-WISE PIZZA COUNT

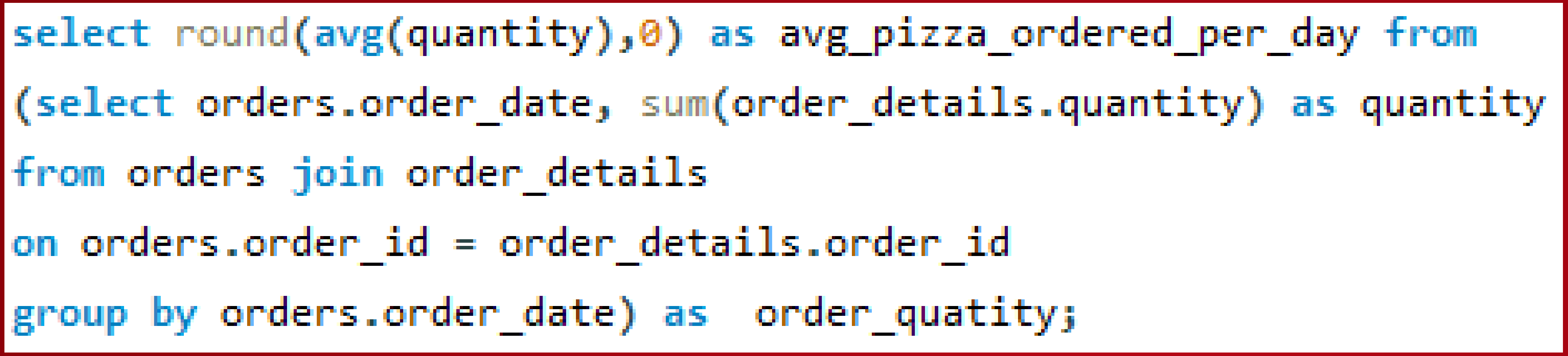


```
select category, count(name) from pizza_types
group by category;
```

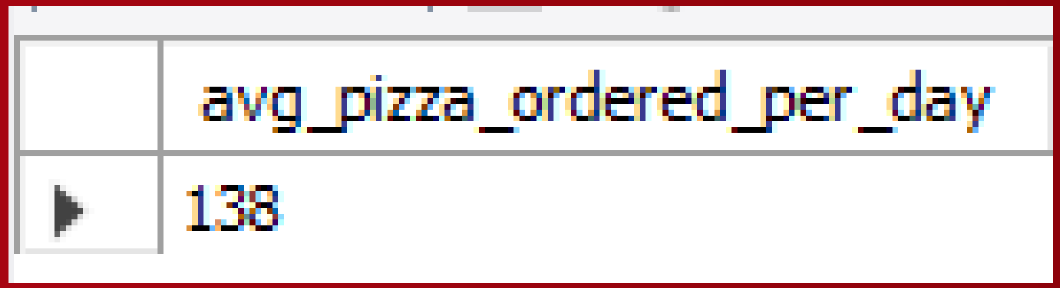


category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

AVG. PIZZAS PER DAY

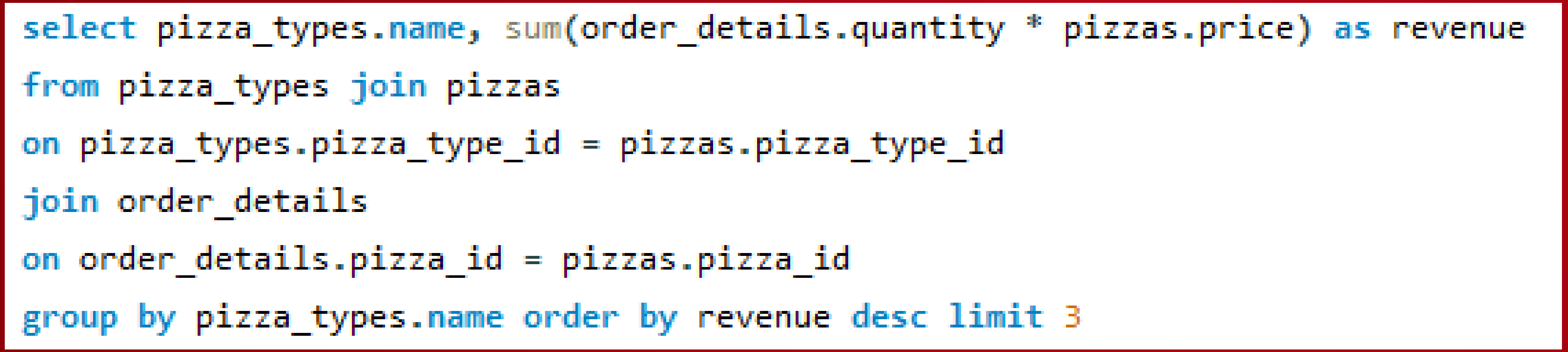


```
select round(avg(quantity),0) as avg_pizza_ordered_per_day from
(select orders.order_date, sum(order_details.quantity) as quantity
from orders join order_details
on orders.order_id = order_details.order_id
group by orders.order_date) as  order_quatity;
```

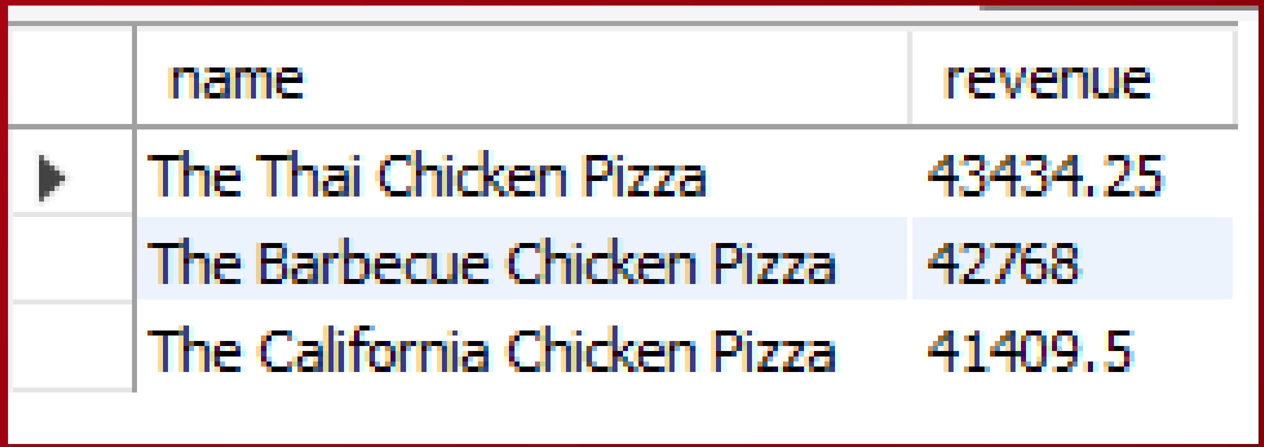


```
avg_pizza_ordered_per_day
138
```

TOP 3 PIZZAS BY REVENUE

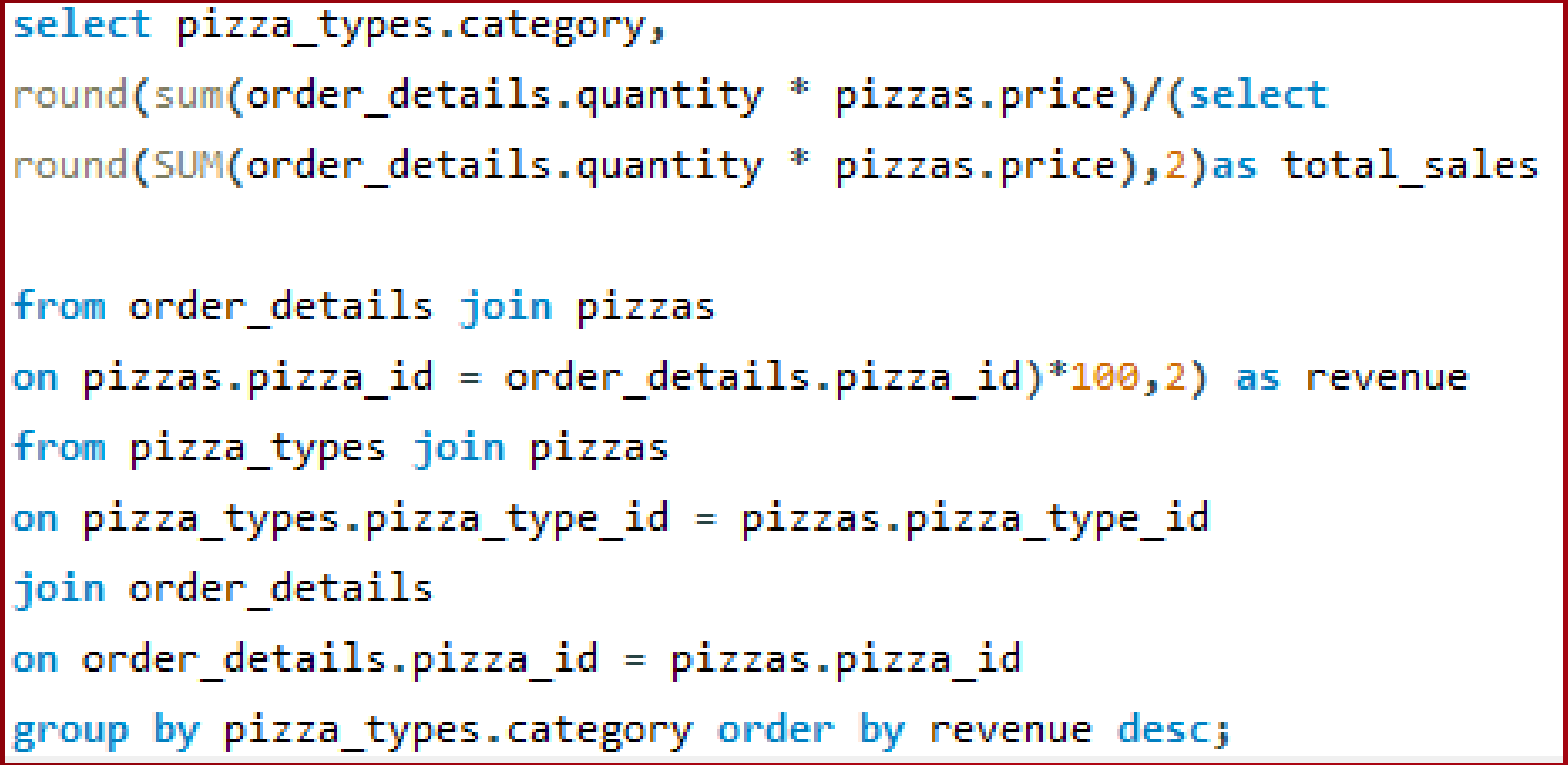


```
select pizza_types.name, sum(order_details.quantity * pizzas.price) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name order by revenue desc limit 3
```



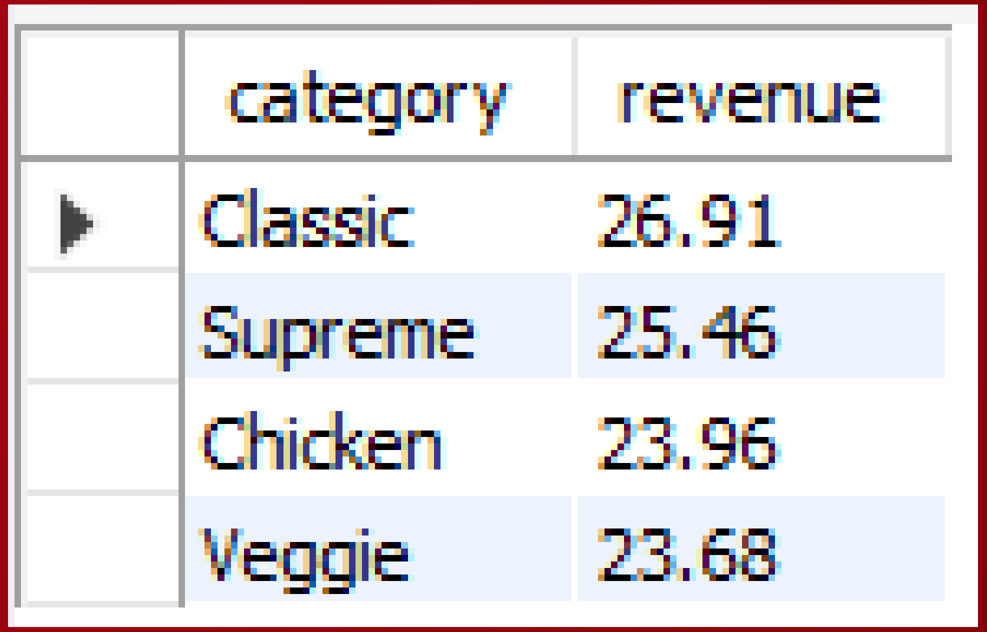
```
name                            revenue
The Thai Chicken Pizza          43434.25
The Barbecue Chicken Pizza      42768
The California Chicken Pizza    41409.5
```

% REVENUE BY PIZZA TYPE



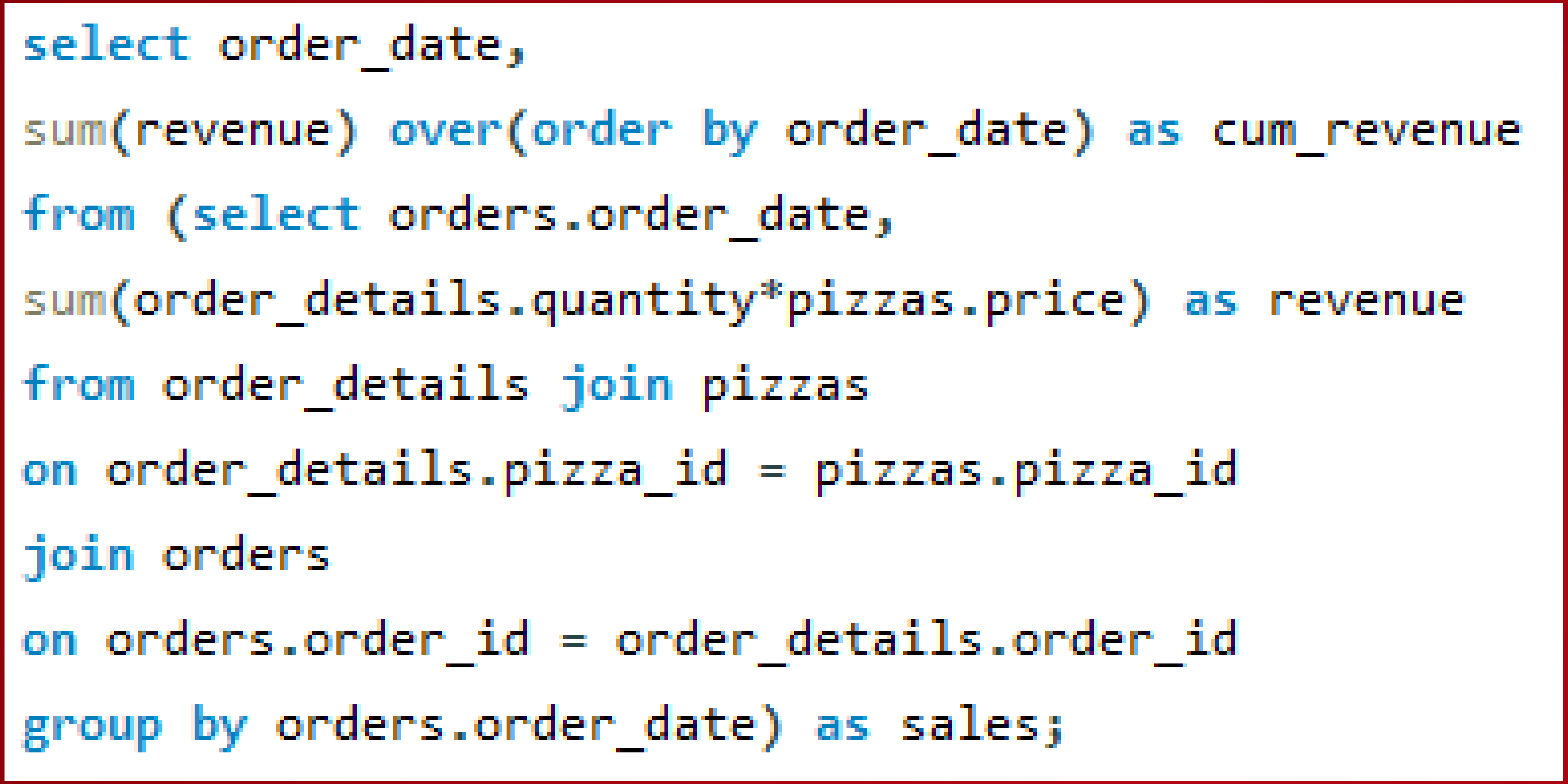
```
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price)/(select
round(SUM(order_details.quantity * pizzas.price),2)as total_sales

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;
```

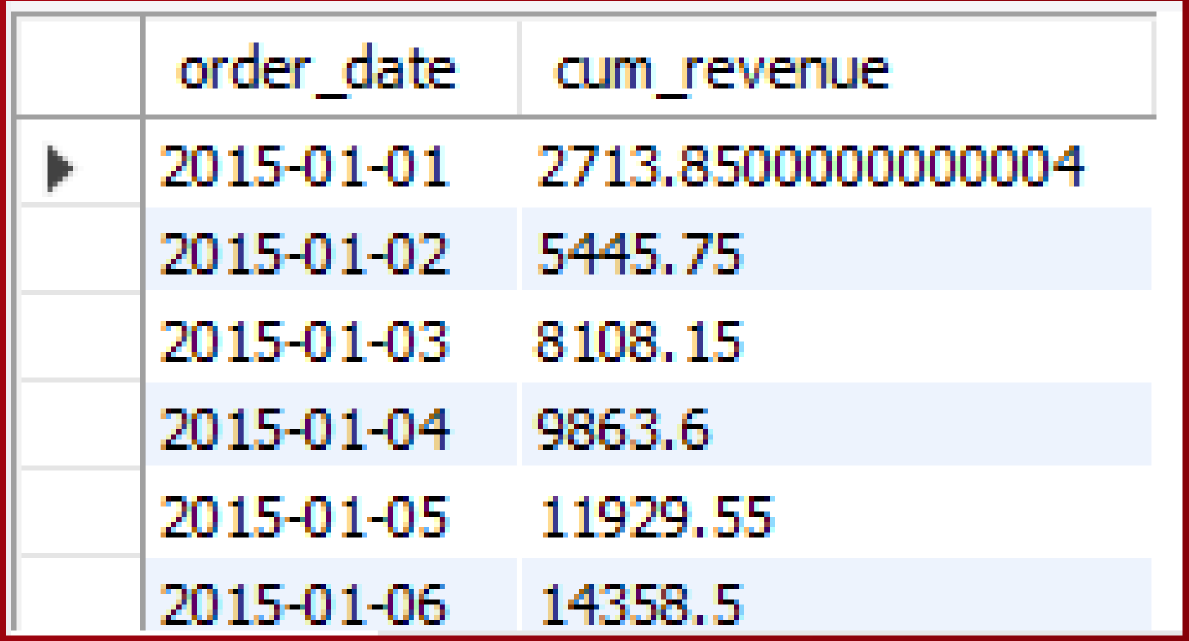


category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

CUMULATIVE REVENUE OVER TIME

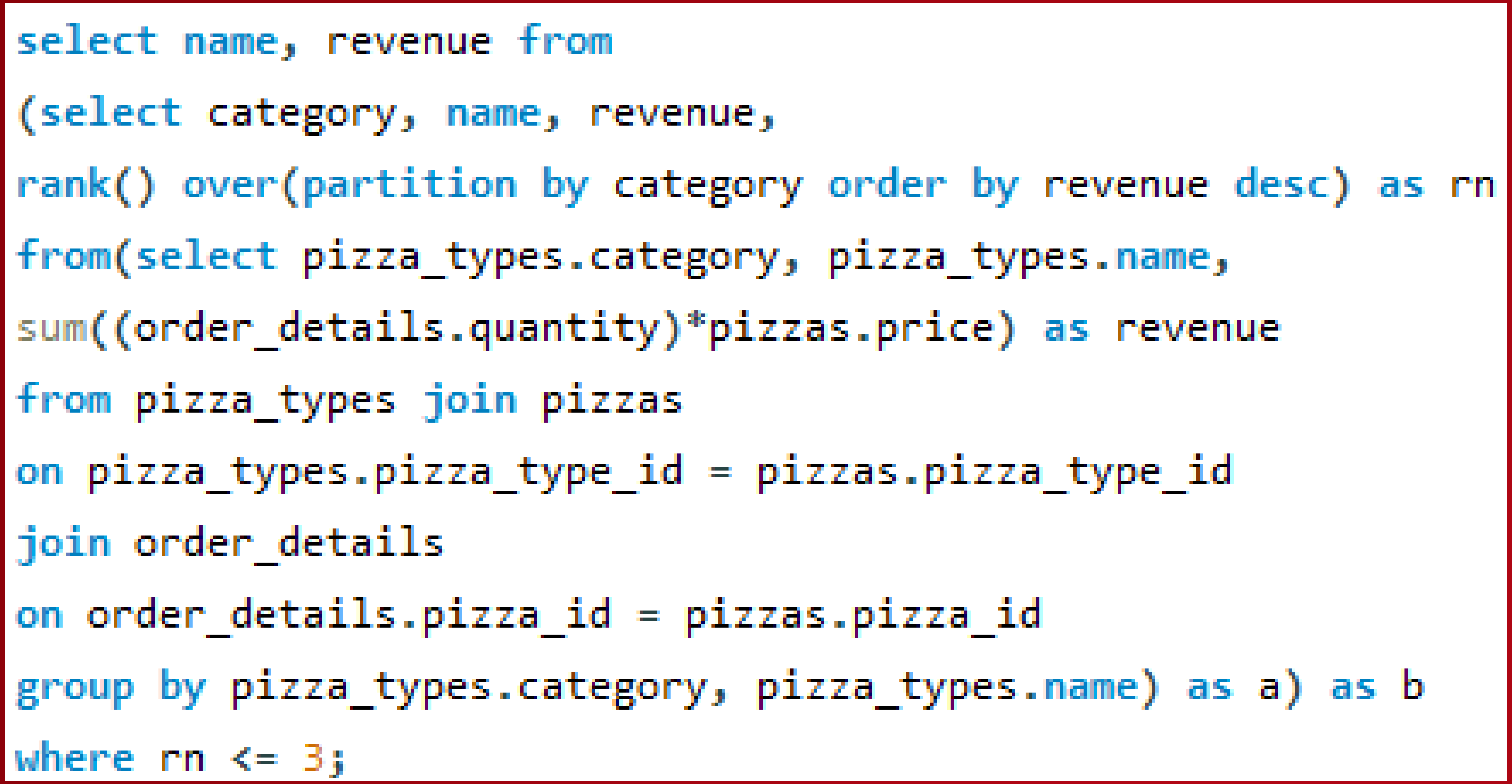


```
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from (select orders.order_date,
sum(order_details.quantity*pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = order_details.order_id
group by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

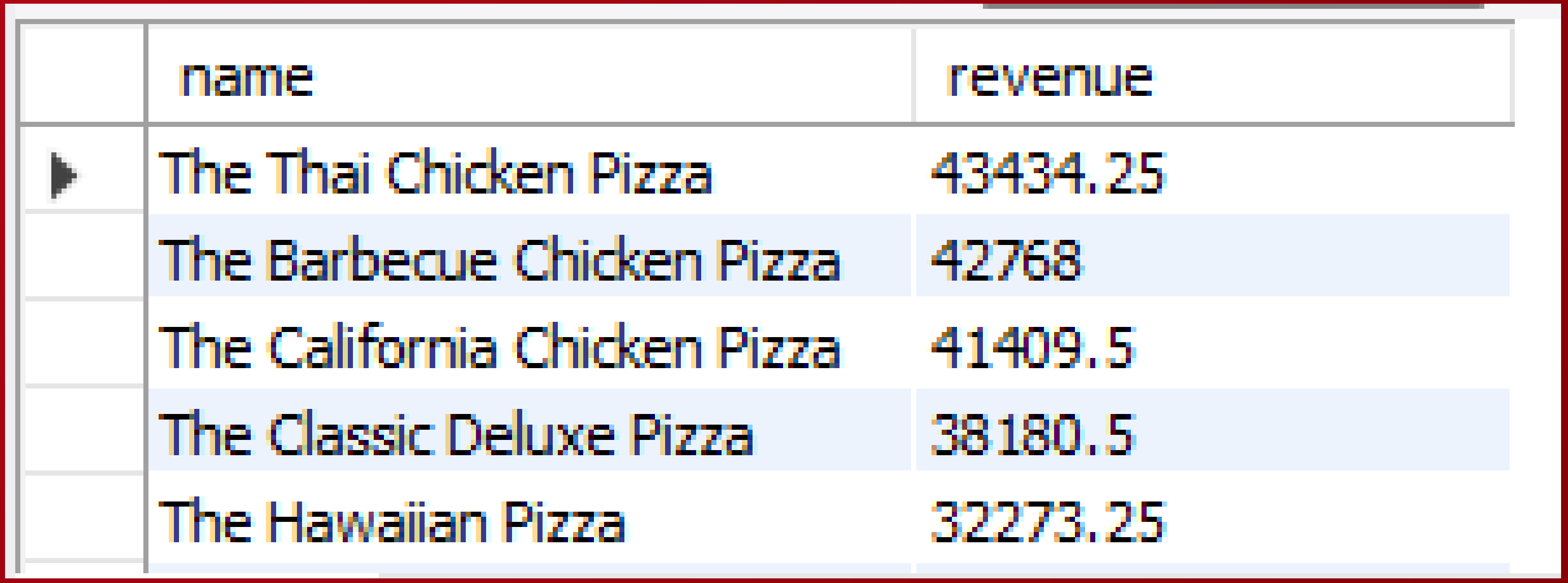


order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5

TOP 3 PIZZAS BY REVENUE IN EACH 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

- Joins: merged data from multiple tables like orders, pizzas, customers using join to get a comprehensive report
- Aggregates (sum, count): sum() used to calculate total revenue, count() used to count total orders and item sales
- Group by & Order by: group by used for category-wise and date-wise analysis, order by applied to rank top-selling pizzas or peak revenue dates
- Limit: retrieved top 5 / top 10 results (e.g., best-selling pizzas, customers)
- Time functions: extracted month, day using extract() or date_format(), helped in identifying high-performing days/months
- Subqueries: used for filtering data before aggregation, enabled comparison like “above average revenue days”
- Window functions: applied rank(), row_number(), sum() over(), helped in calculating cumulative metrics and rankings

VEGGIE PIZZA

THANK YOU