LOGO HERE

PIZZA SALES PROJECT USING SQL

ORDER NOW

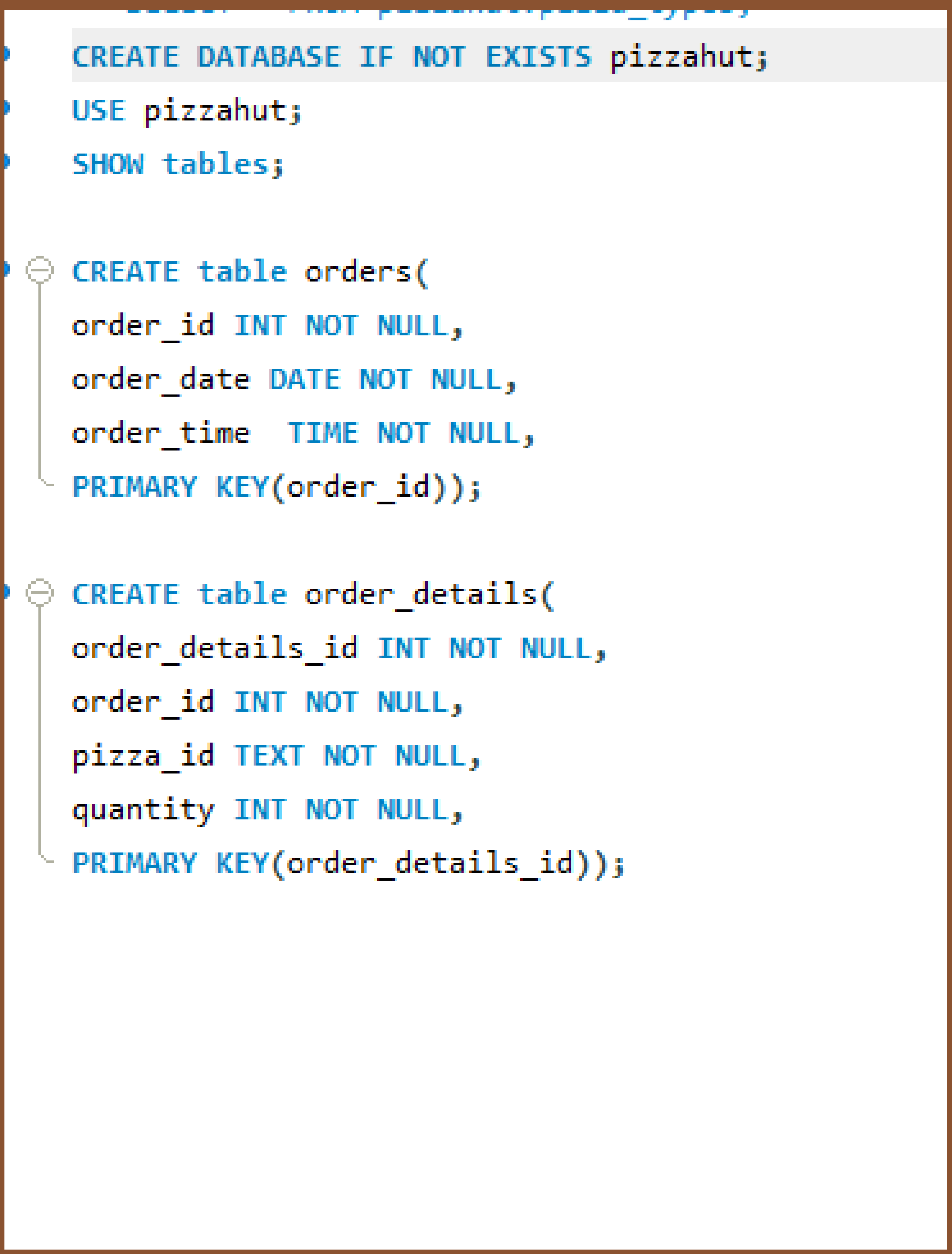
HELLO!

My name is Siddharth kumar and in this project I have utilized SQL Queries to solve the problems related to design this project.

Dummy Datasets required for this project

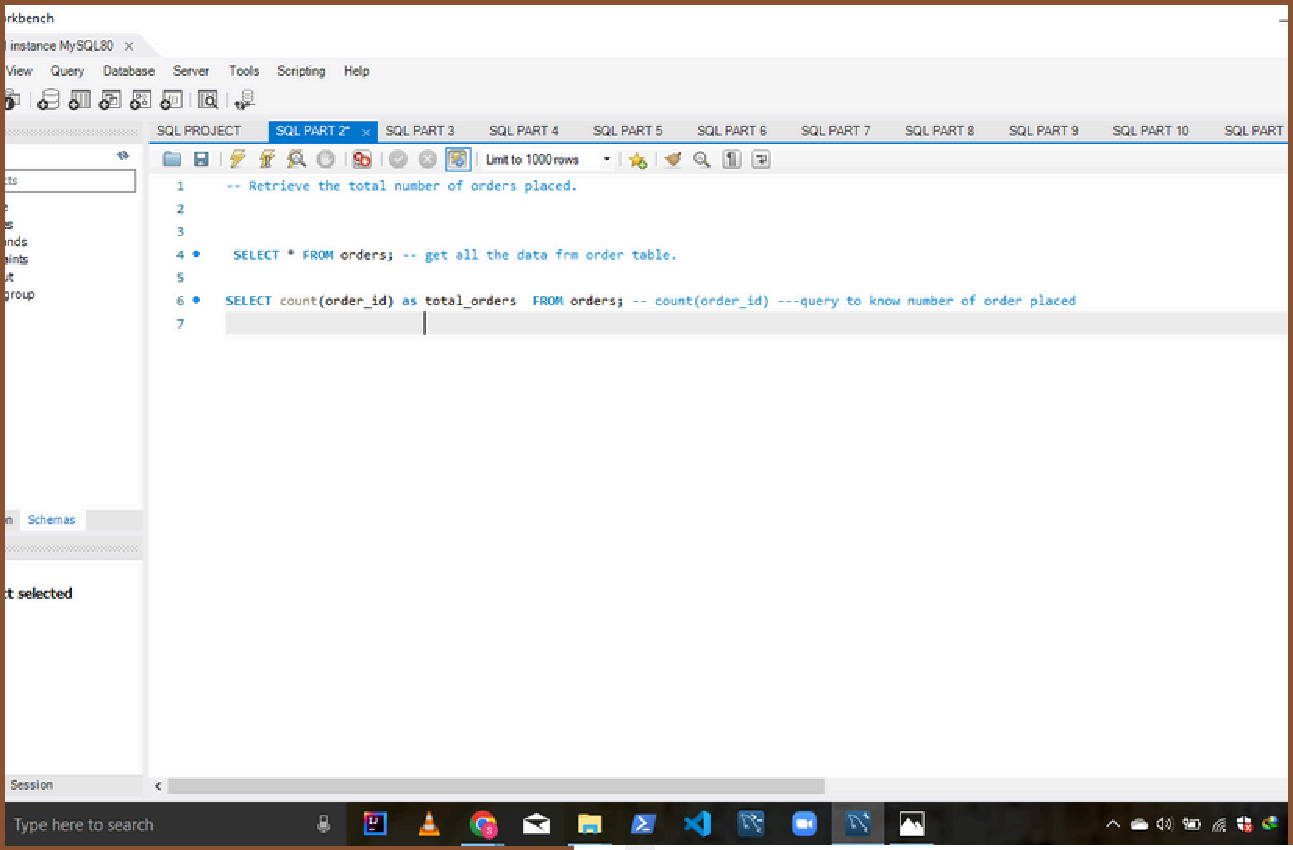
pizza_id	pizza_type_id	size	price
bbq_ckn_s	bbq_ckn	S	12.75
bbq_ckn_m	bbq_ckn	M	16.75
bbq_ckn_l	bbq_ckn	L	20.75
cali_ckn_s	cali_ckn	S	12.75
cali_ckn_m	cali_ckn	M	16.75
cali_ckn_l	cali_ckn	L	20.75
ckn_alfredo_s	ckn_alfredo	S	12.75
ckn_alfredo_m	ckn_alfredo	M	16.75
ckn_alfredo_l	ckn_alfredo	L	20.75
ckn_pesto_s	ckn_pesto	S	12.75
ckn_pesto_m	ckn_pesto	M	16.75
ckn_pesto_l	ckn_pesto	L	20.75
southw_ckn_s	southw_ckn	S	12.75
southw_ckn_m	southw_ckn	M	16.75
southw_ckn_l	southw_ckn	L	20.75
thai_ckn_s	thai_ckn	S	12.75
thai_ckn_m	thai_ckn	M	16.75
thai_ckn_l	thai_ckn	L	20.75
big_meat_s	big_meat	S	12
big_meat_m	big_meat	M	16
big_meat_l	big_meat	L	20.5
classic_dlx_s	classic_dlx	S	12
classic_dlx_m	classic_dlx	M	16
classic_dlx_l	classic_dlx	L	20.5
hawaiian_s	hawaiian	S	10.5
hawaiian_m	hawaiian	M	13.25
hawaiian_l	hawaiian	L	16.5
ital_cpcllo_s	ital_cpcllo	S	12
ital_cpcllo_m	ital_cpcllo	M	16
ital_cpcllo_l	ital_cpcllo	L	20.5
napolitana_s	napolitana	S	12
napolitana_m	napolitana	M	16
napolitana_l	napolitana	L	20.5
pep_msh_pep_s	pep_msh_pep	S	11
pep_msh_pep_m	pep_msh_pep	M	14.5
pep_msh_pep_l	pep_msh_pep	L	17.5
pepperoni_s	pepperoni	S	9.75
pepperoni_m	pepperoni	M	12.5
pepperoni_l	pepperoni	L	15.25
the_greek_s	the_greek	S	12
the_greek_m	the_greek	M	16
the_greek_l	the_greek	L	20.5
the_greek_xl	the_greek	XL	25.5
the_greek_xxl	the_greek	XXL	35.95
brie_carre_s	brie_carre	S	23.65
calabrese_s	calabrese	S	12.25
calabrese_m	calabrese	M	16.25
calabrese_l	calabrese	L	20.25
ital_supr_s	ital_supr	S	12.5
ital_supr_m	ital_supr	M	16.5
ital_supr_l	ital_supr	L	20.75
peppr_salami_s	peppr_salami	S	12.5

-- SELECT * FROM pizzahut.pizza_types;

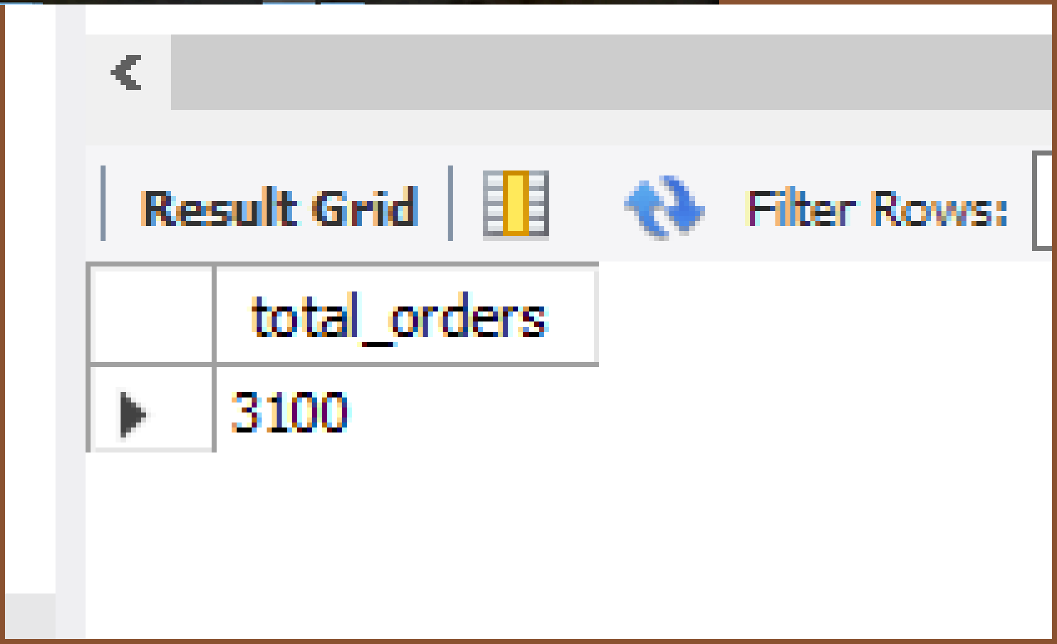


```
CREATE DATABASE IF NOT EXISTS pizzahut;
USE pizzahut;
SHOW tables;

CREATE table orders(
order_id INT NOT NULL,
order_date DATE NOT NULL,
order_time  TIME NOT NULL,
PRIMARY KEY(order_id));

CREATE table order_details(
order_details_id INT NOT NULL,
order_id INT NOT NULL,
pizza_id TEXT NOT NULL,
quantity INT NOT NULL,
PRIMARY KEY(order_details_id));
```

Retrieve the total number of orders placed.

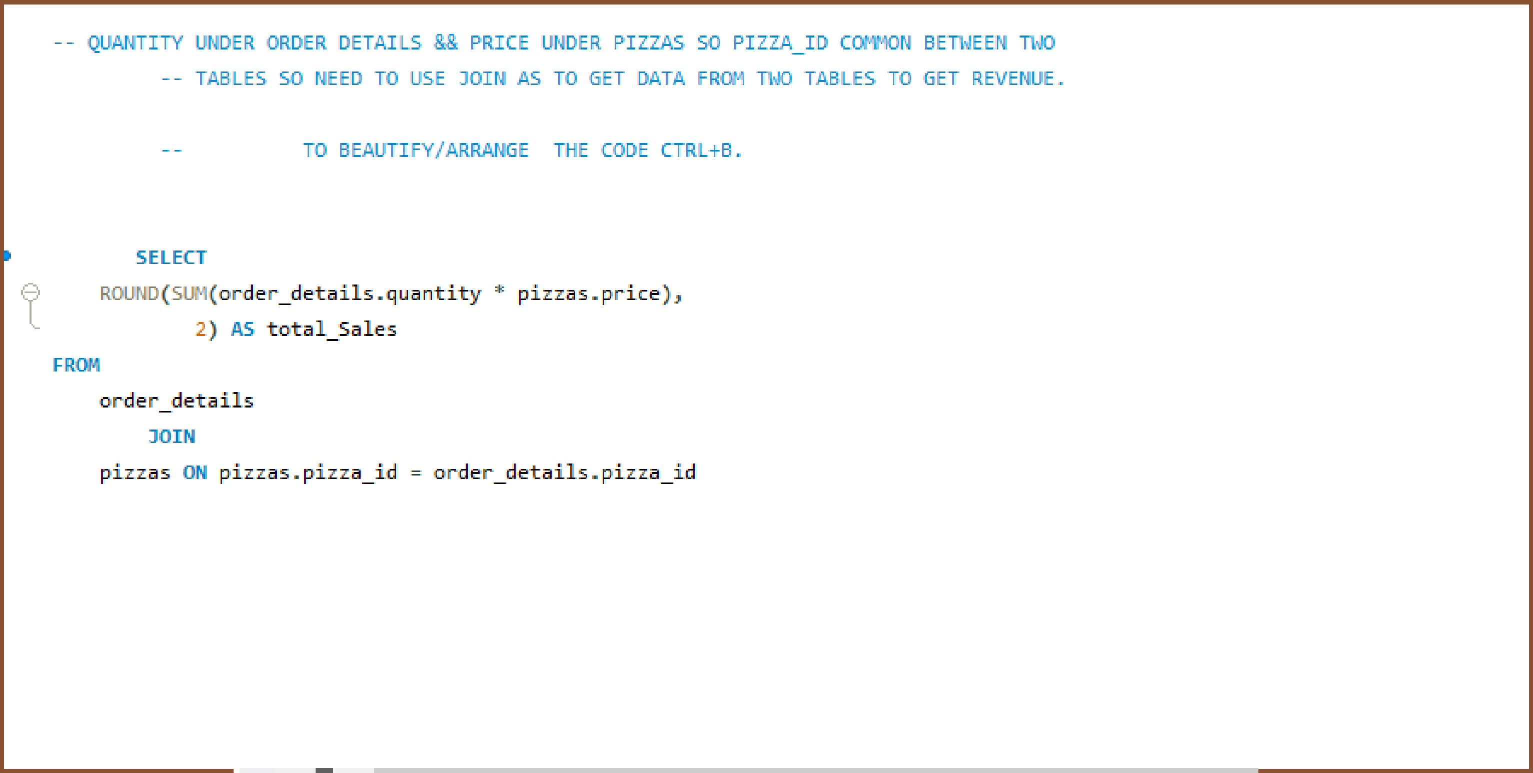


```
-- Retrieve the total number of orders placed.


 SELECT * FROM orders; -- get all the data frm order table.

SELECT count(order_id) as total_orders  FROM orders; -- count(order_id) ---query to know number of order placed

```

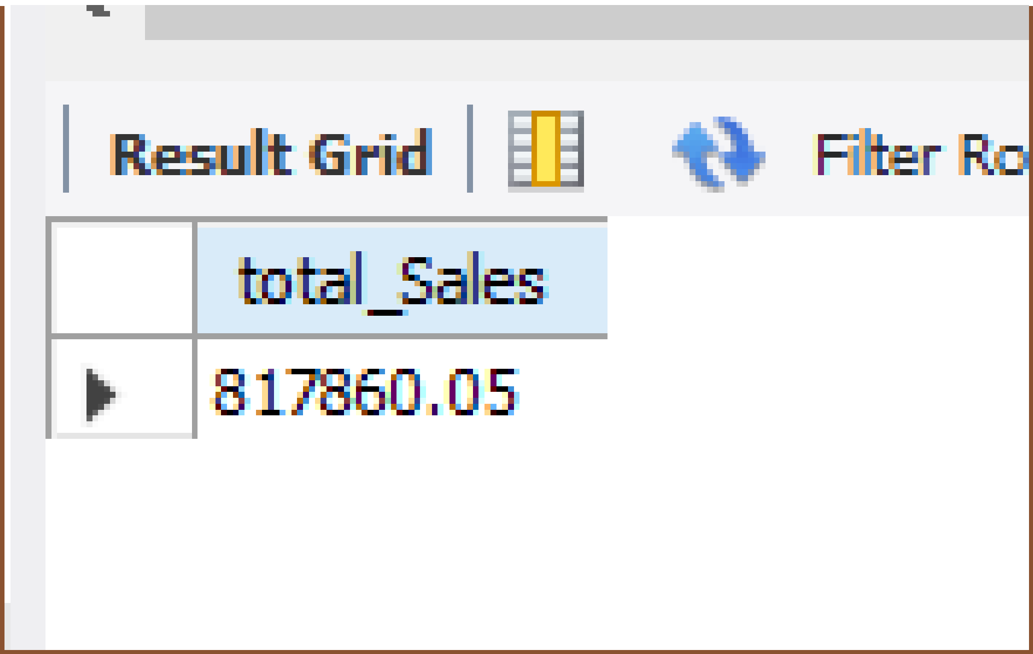


```
total_orders
3100
```

Calculate the total revenue generated from pizza sales.

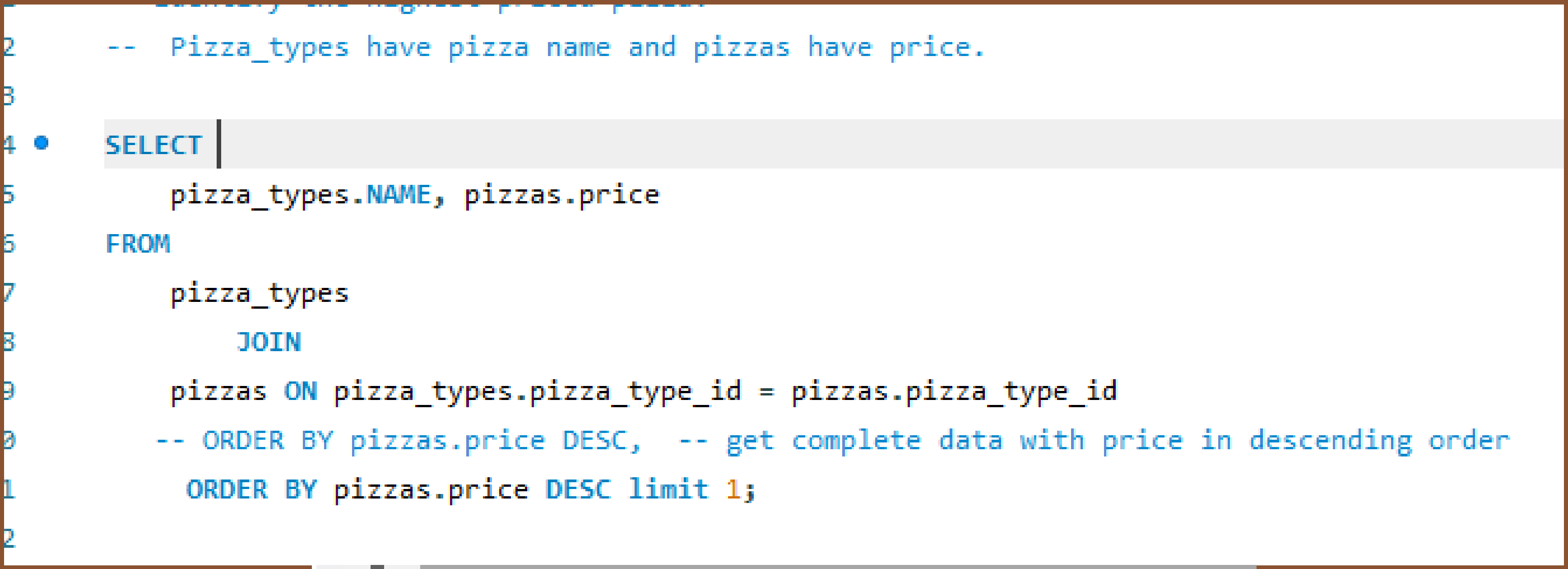


```
-- QUANTITY UNDER ORDER DETAILS && PRICE UNDER PIZZAS SO PIZZA_ID COMMON BETWEEN TWO
        -- TABLES SO NEED TO USE JOIN AS TO GET DATA FROM TWO TABLES TO GET REVENUE.

        --          TO BEAUTIFY/ARRANGE  THE CODE CTRL+B.


      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
```

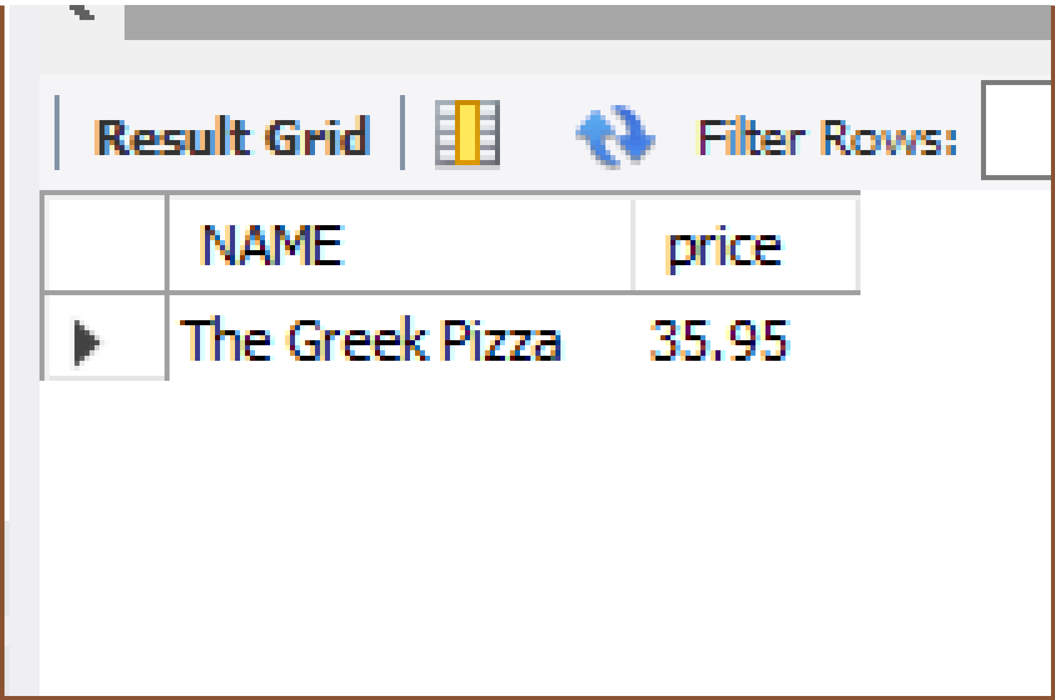


```
total_Sales
817860.05
```

Identify the highest-priced pizza.

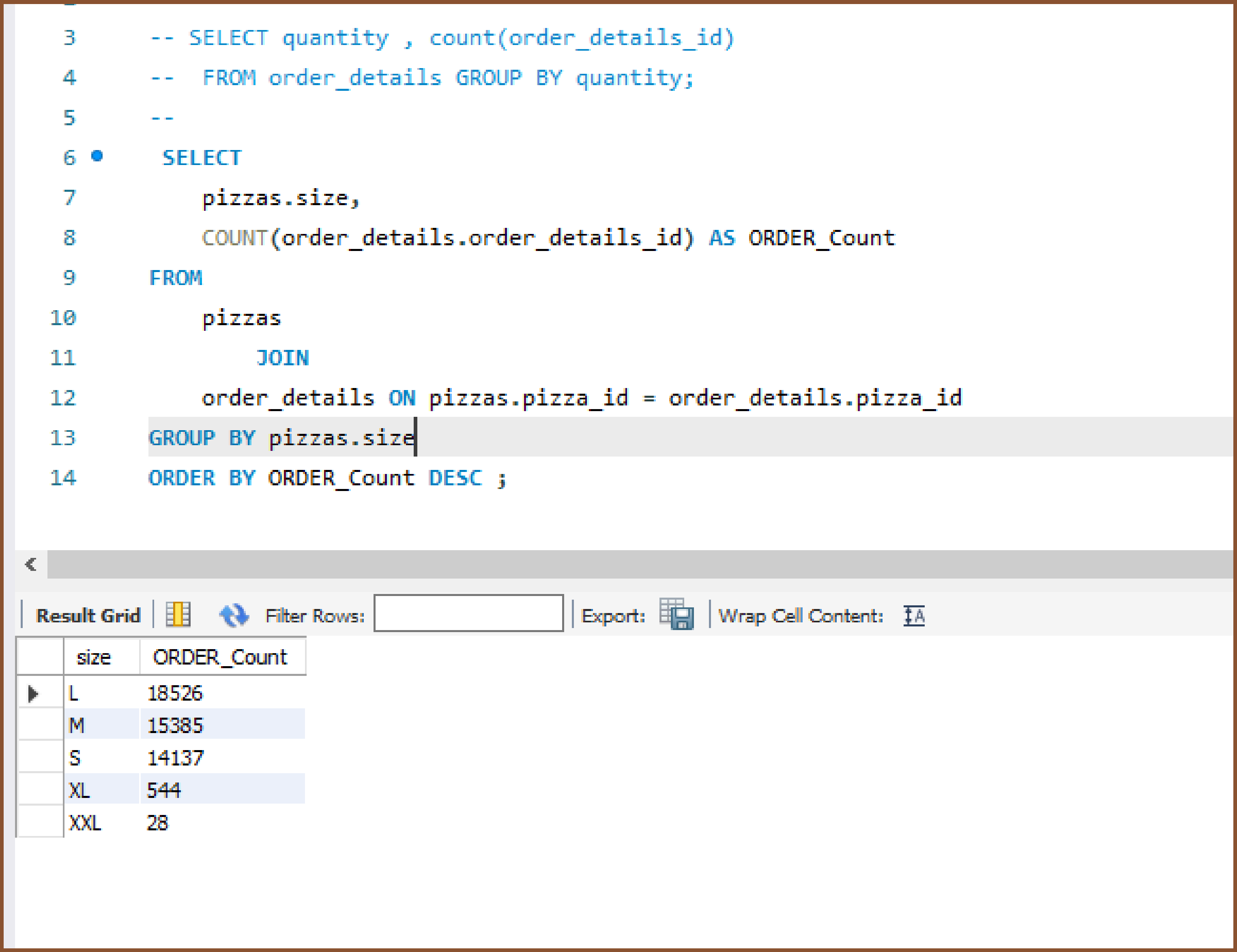


```
--  Pizza_types have pizza name and pizzas have price.

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,  -- get complete data with price in descending order
     ORDER BY pizzas.price DESC limit 1;
```



NAME	price
The Greek Pizza	35.95

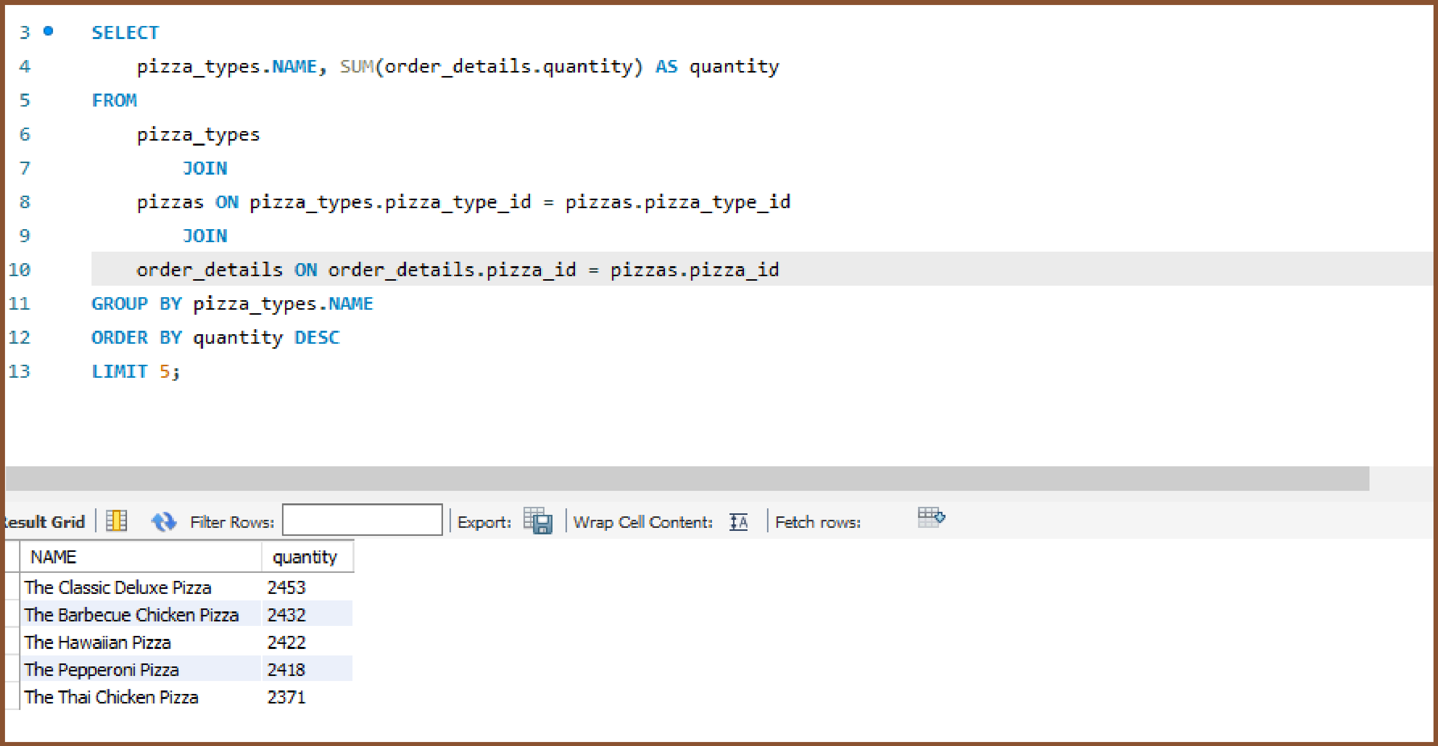
Identify the most common pizza size ordered.



```
-- SELECT quantity , count(order_details_id)
--  FROM order_details GROUP BY quantity;
--
 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
M	15385
S	14137
XL	544
XXL	28

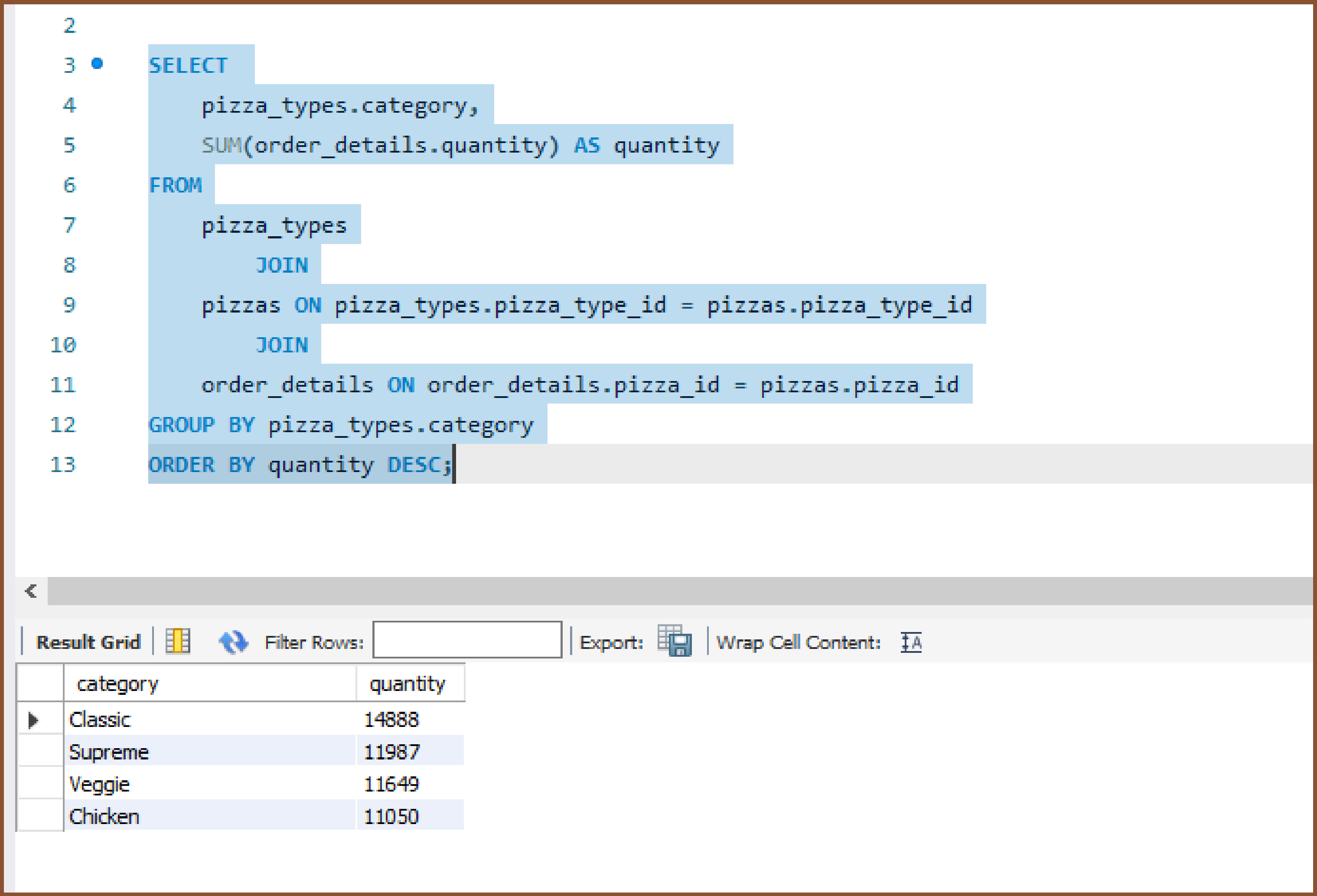
List the top 5 most ordered pizza types along with their quantities.



```
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

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

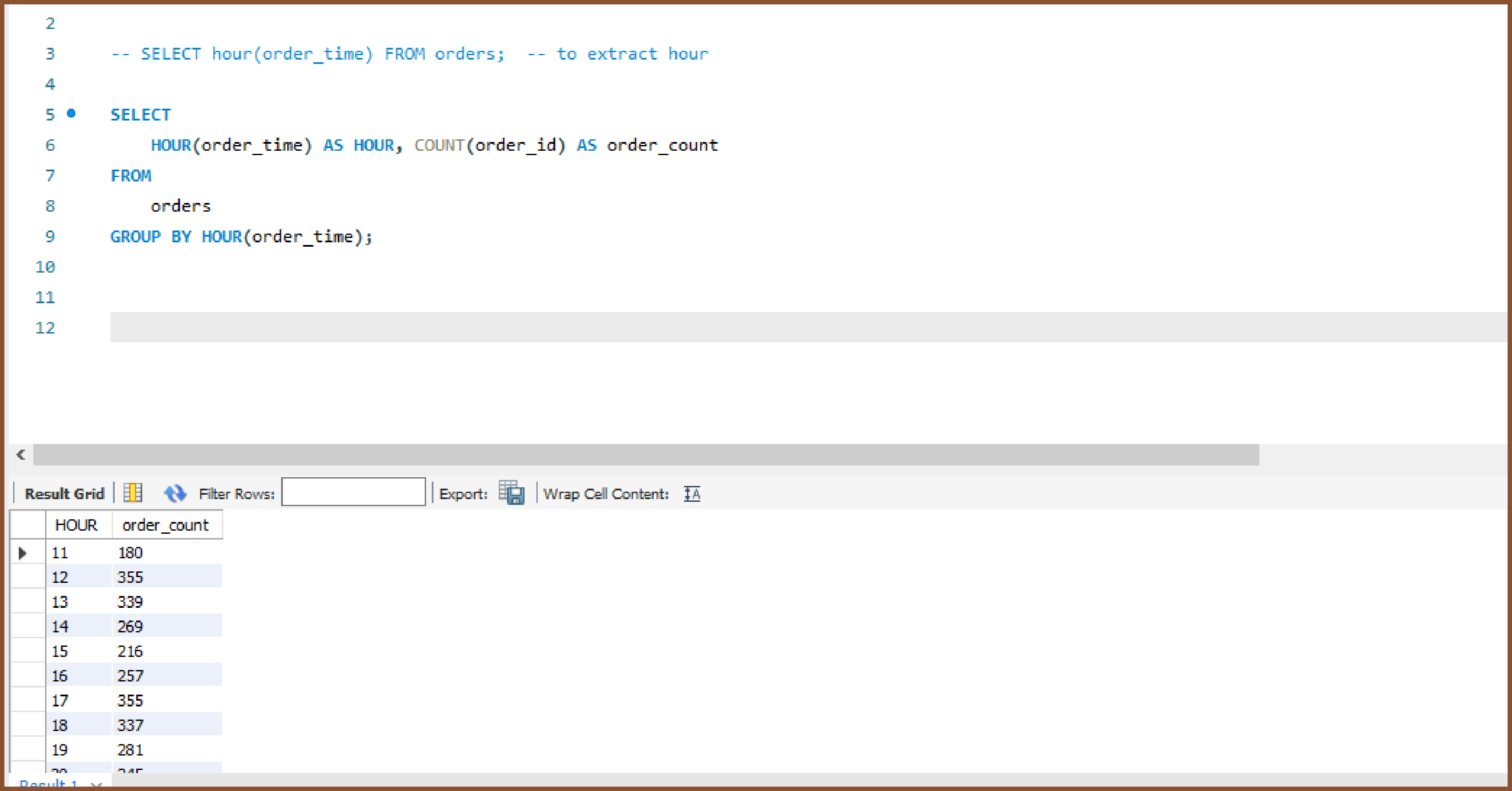


```sql

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;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

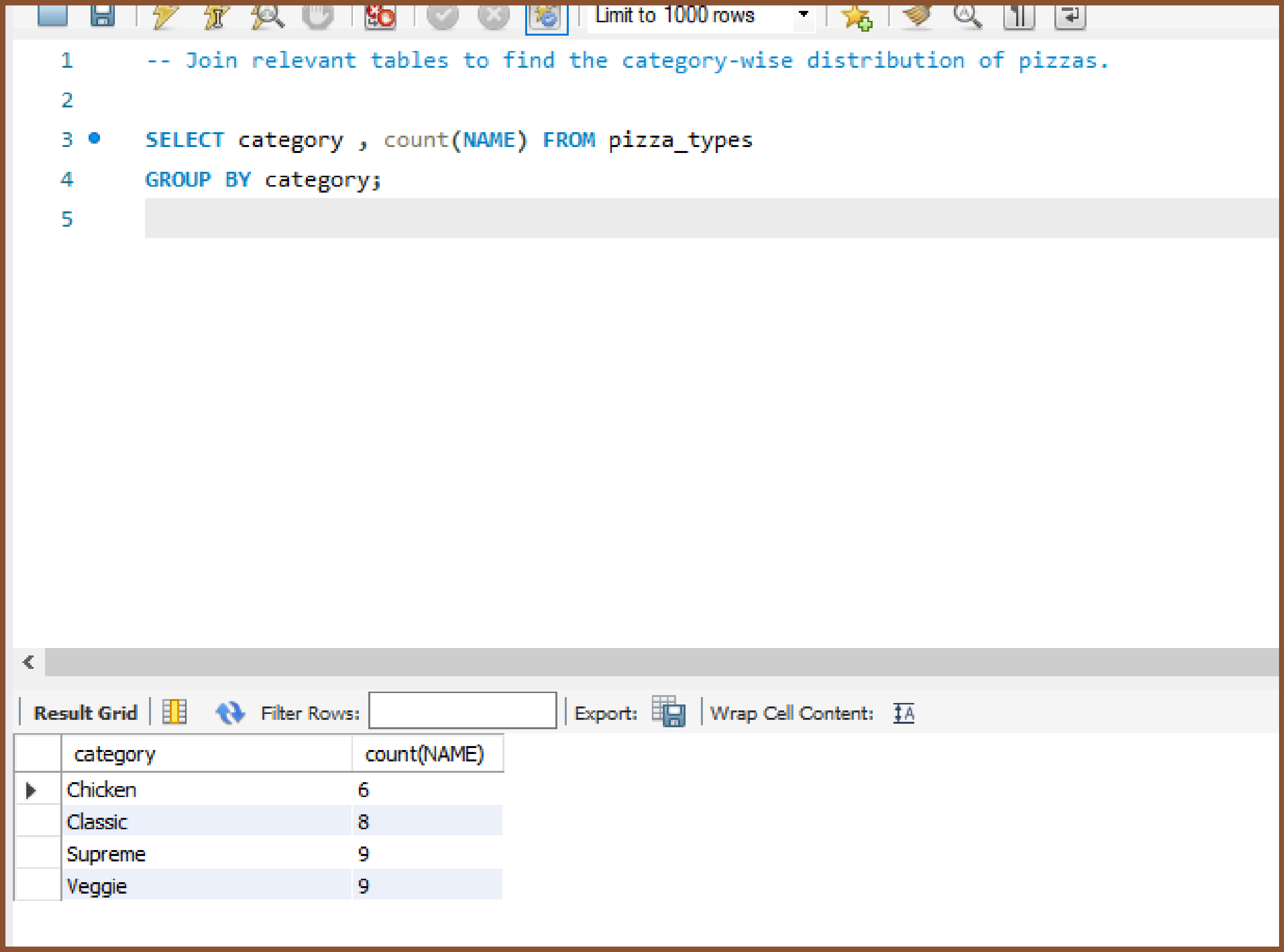


```
-- SELECT hour(order_time) FROM orders;  -- to extract hour

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

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

HOUR	order_count
11	180
12	355
13	339
14	269
15	216
16	257
17	355
18	337
19	281

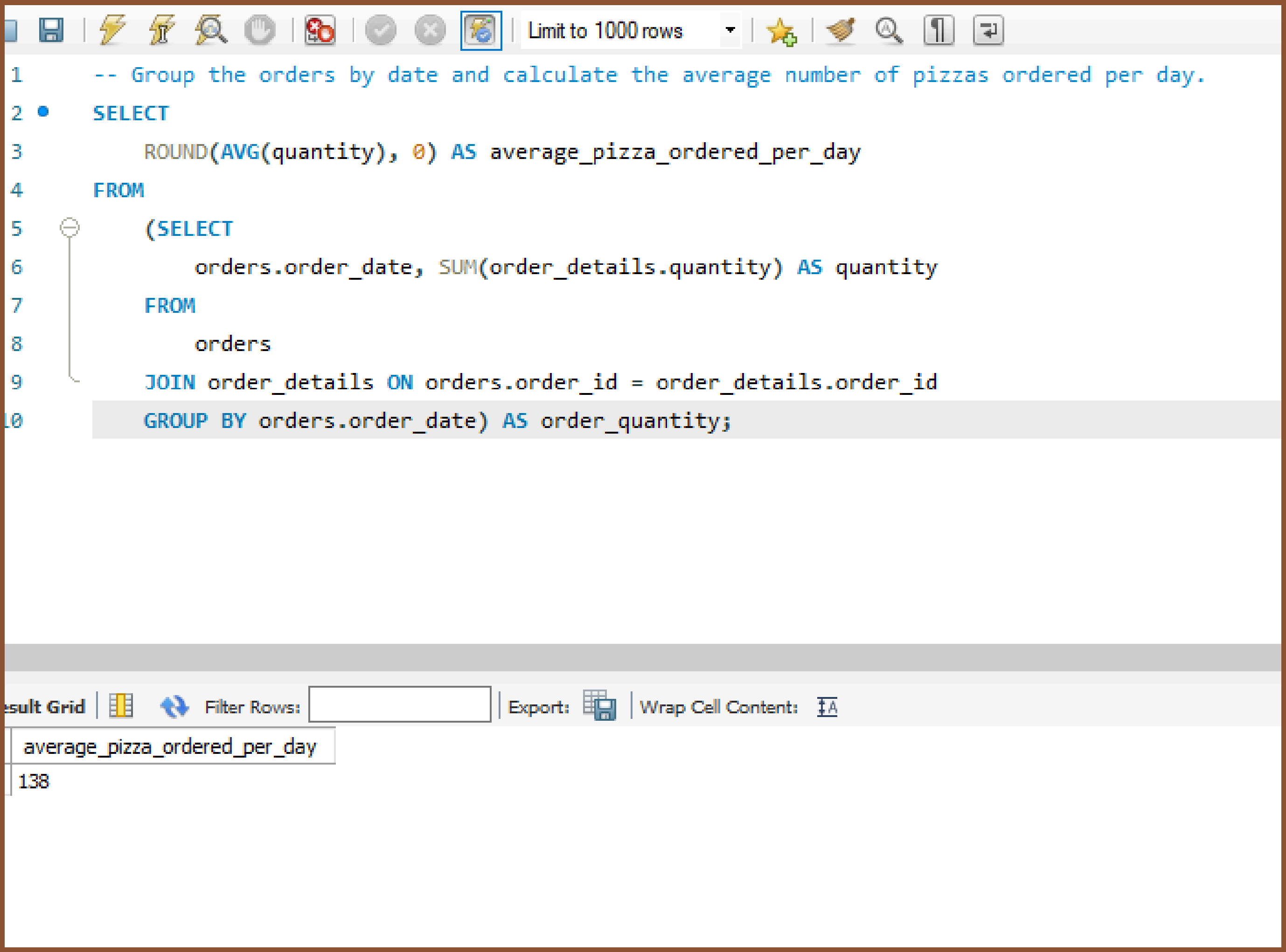
Join relevant tables to find the category-wise distribution of pizzas.



```sql
-- 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

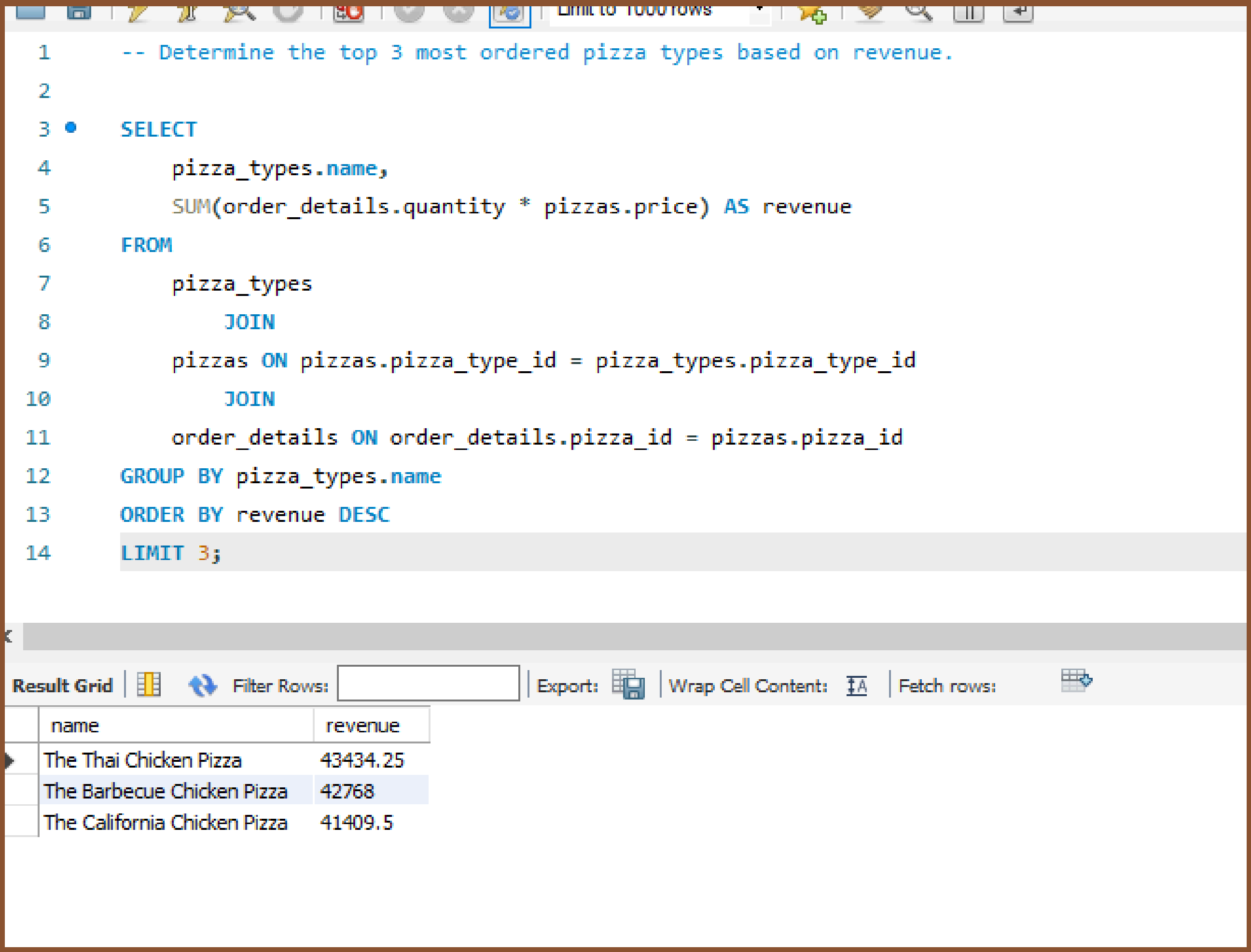
Group the orders by date and calculate the average number of pizzas ordered per day.



```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) AS average_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_quantity;
```

```
average_pizza_ordered_per_day
138
```

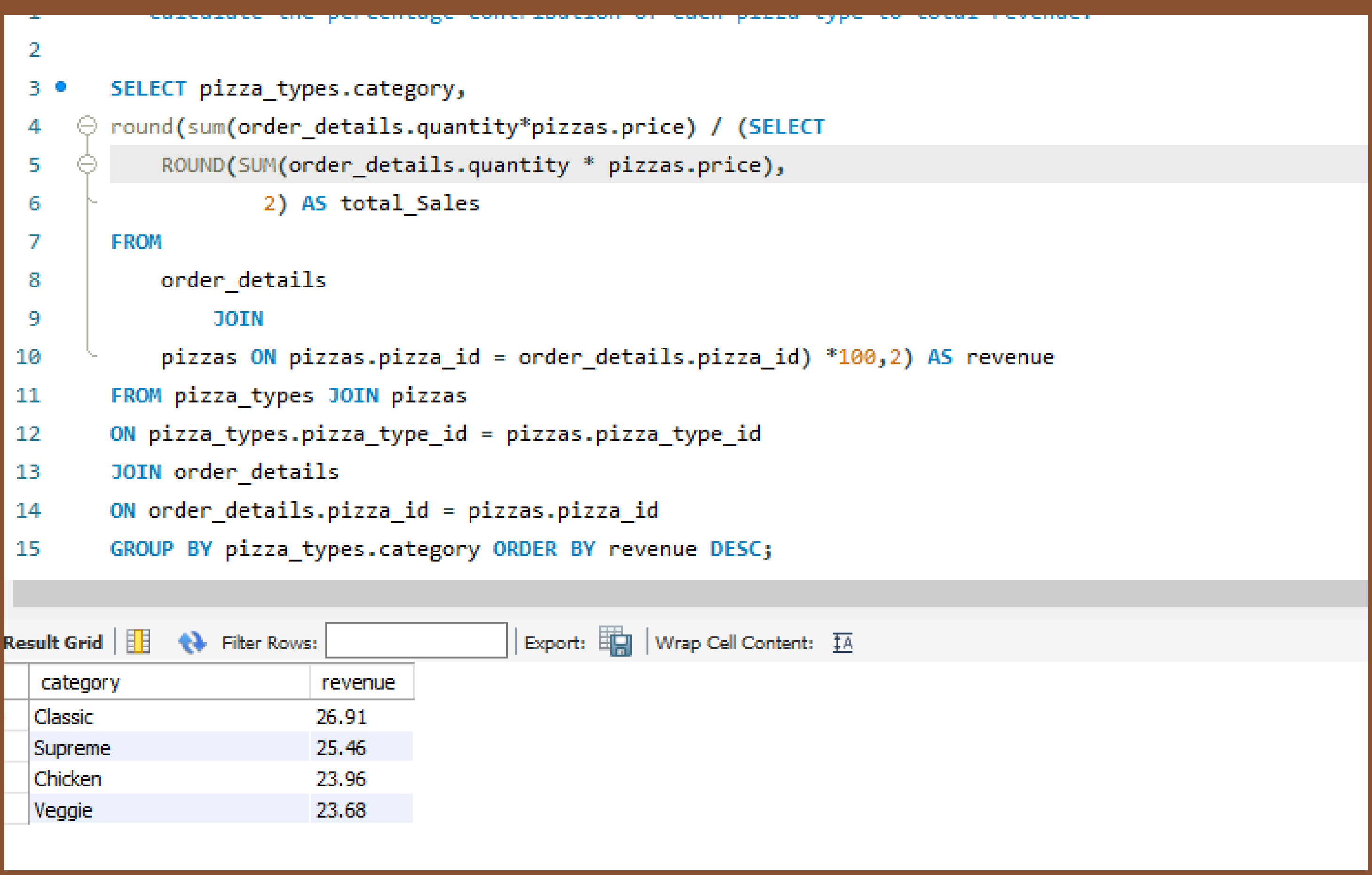
Determine the top 3 most ordered pizza types based on revenue.



```sql
-- 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

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

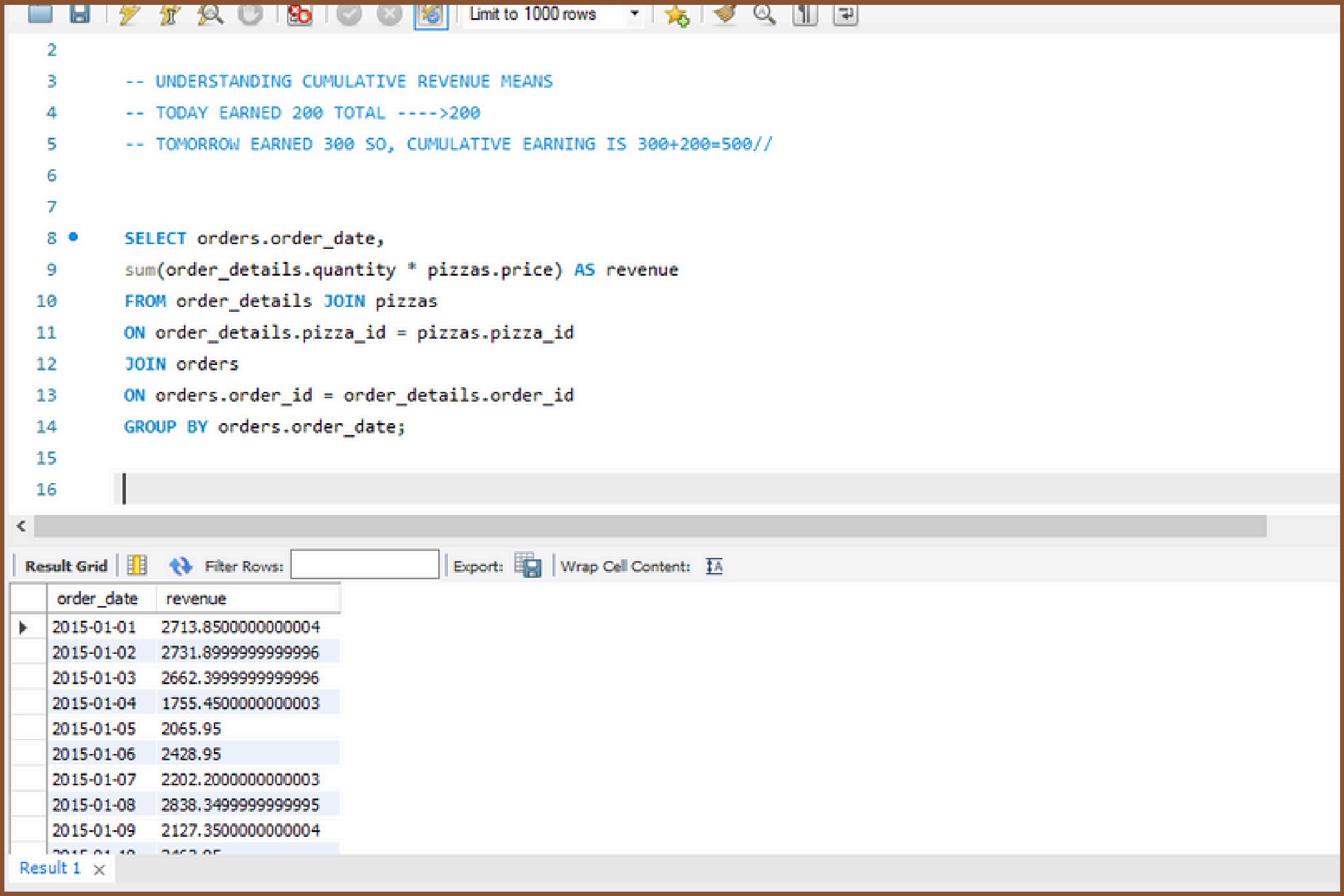


```
SELECT pizza_types.category,
round(sum(order_details.quantity*pizzas.price) / (SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_Sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2) AS revenue
FROM pizza_types 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;
```

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

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

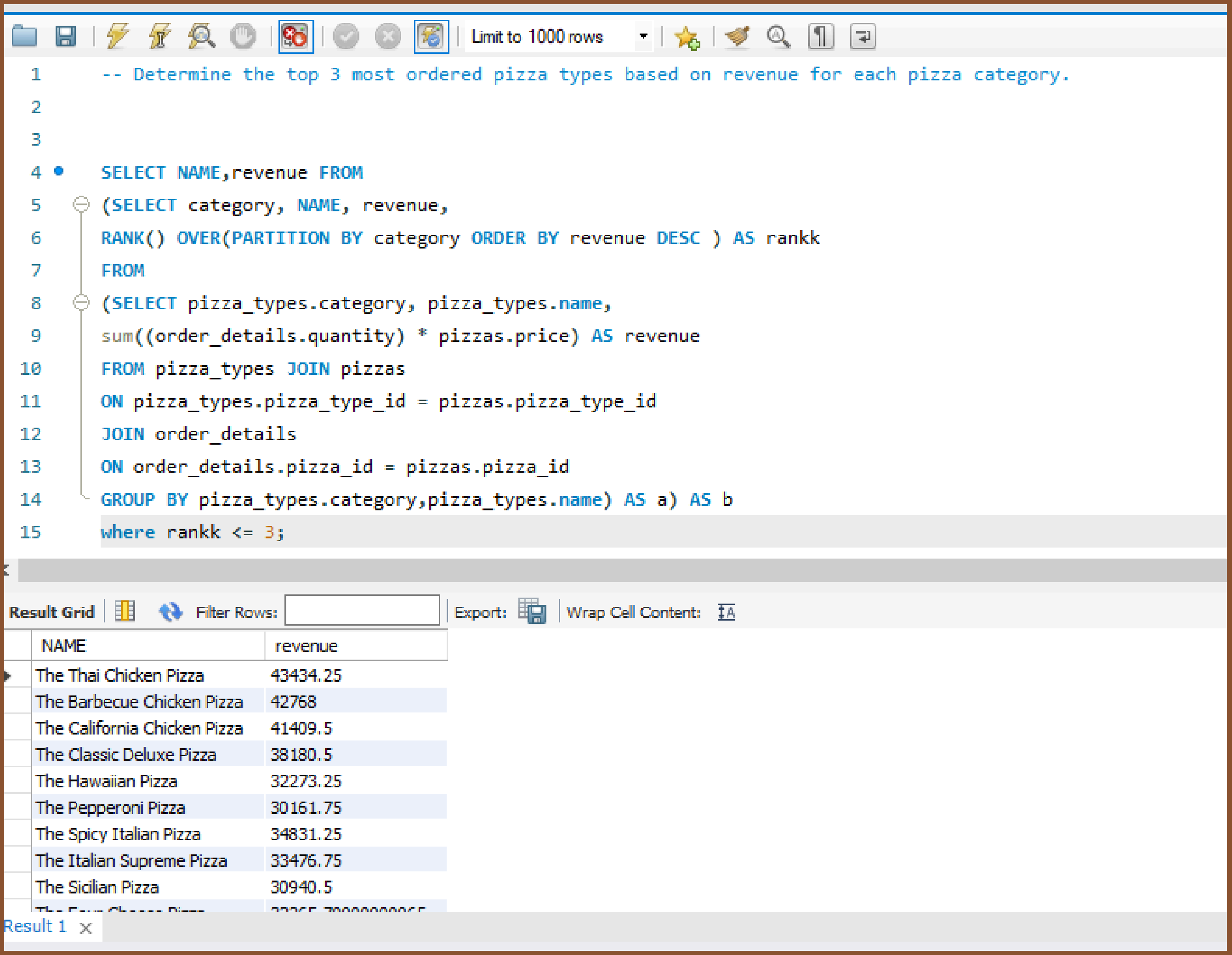
Analyze the cumulative revenue generated over time.



```
-- UNDERSTANDING CUMULATIVE REVENUE MEANS
-- TODAY EARNED 200 TOTAL ---->200
-- TOMORROW EARNED 300 SO, CUMULATIVE EARNING IS 300+200=500//


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;
```

order_date	revenue
2015-01-01	2713.8500000000004
2015-01-02	2731.8999999999996
2015-01-03	2662.3999999999996
2015-01-04	1755.4500000000003
2015-01-05	2065.95
2015-01-06	2428.95
2015-01-07	2202.2000000000003
2015-01-08	2838.3499999999995
2015-01-09	2127.3500000000004

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



```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 rankk
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 rankk <= 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
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5