PIZZA SALES

PIZZA SALES

WELCOME TO DATABASE

pizzasales.csv file
order_details
orders
pizzas
pizza_types

Containing order numbers,uantity,pice,categort.pizza_id,order_id etc

PIZZA SALES

QUERIES

Basic:

Retrieve the total number of orders placed.
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:

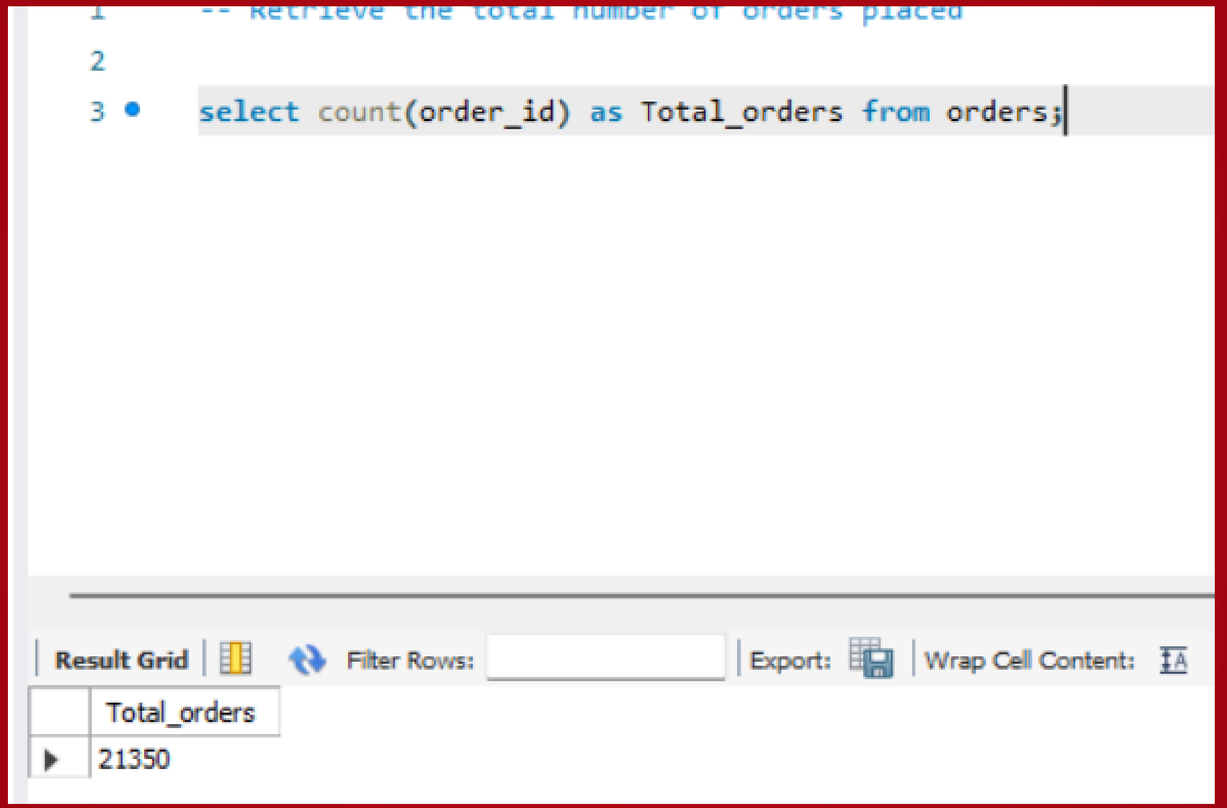
Join the necessary tables to find the total quantity of each pizza category ordered.
Determine the distribution of orders by hour of the day.
Join relevant tables to find the category-wise distribution of pizzas.
Group the orders by date and calculate the average number of pizzas ordered per day.
Determine the top 3 most ordered pizza types based on revenue.

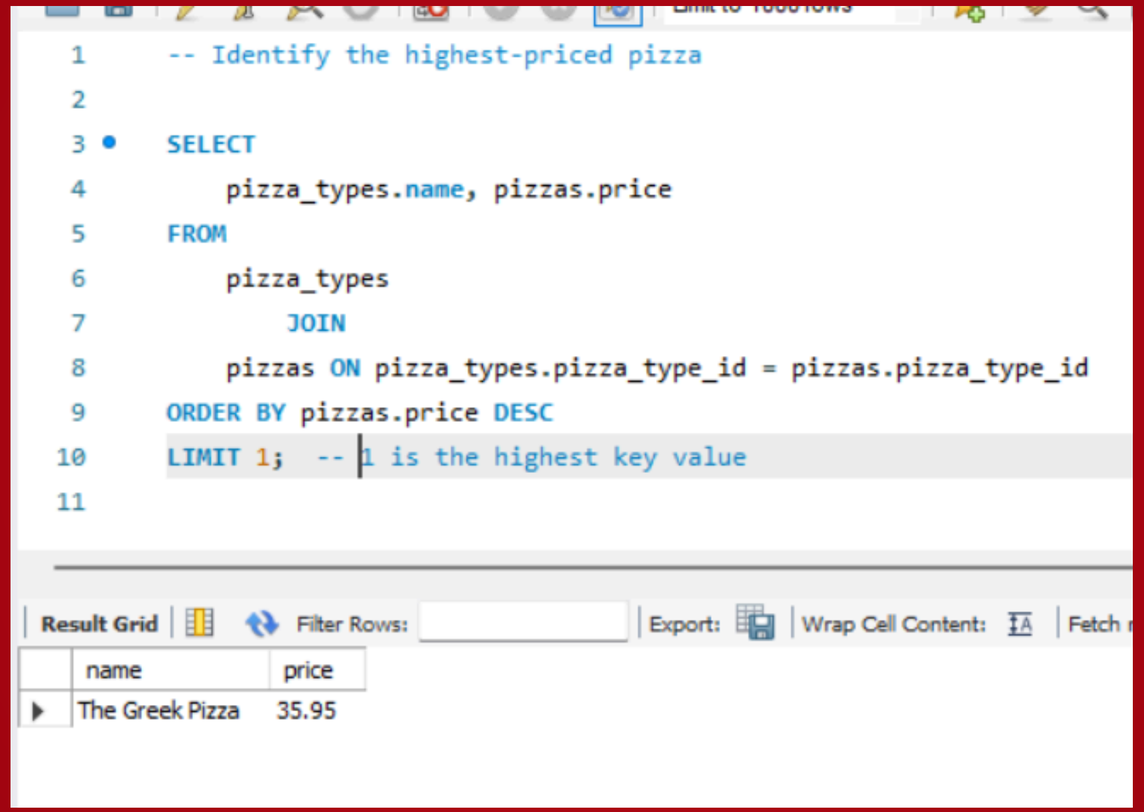
Advanced:

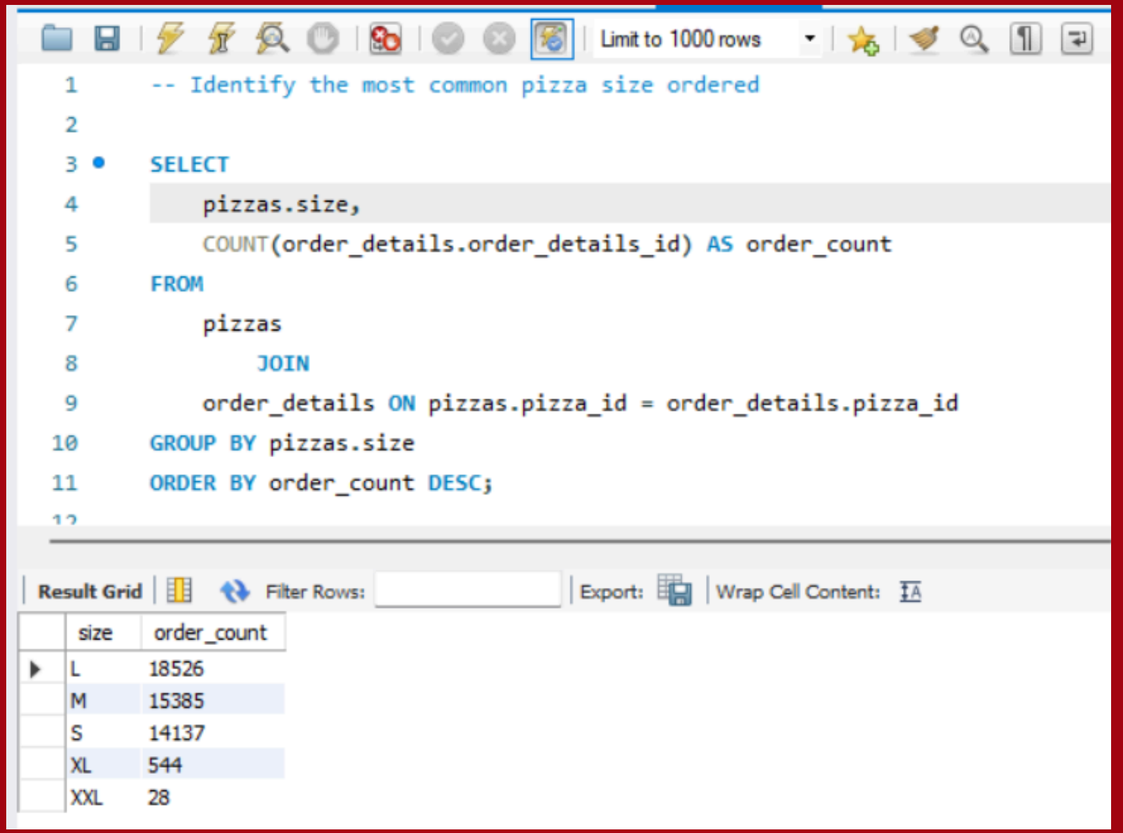
Analyze the cumulative revenue generated over time.
Determine the top 3 most ordered pizza types based on revenue for each pizza category.t

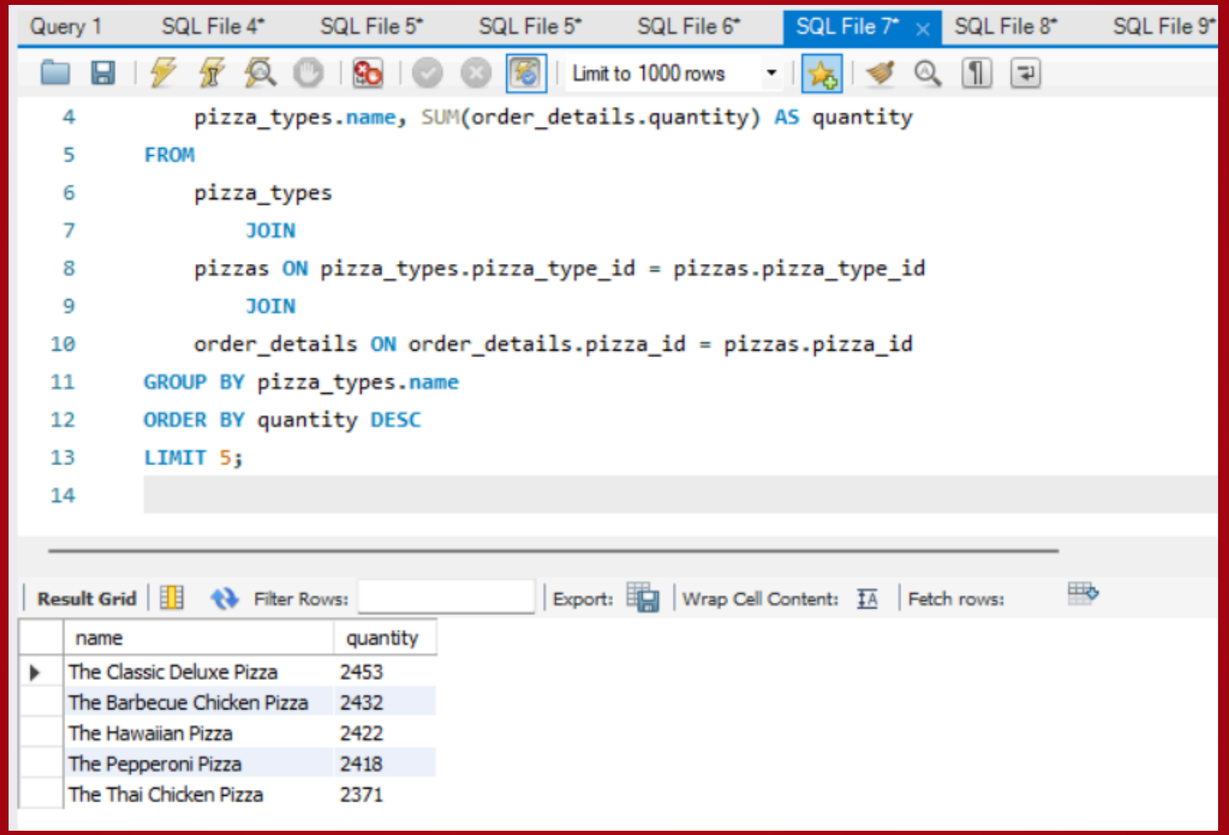
PIZZA SALES

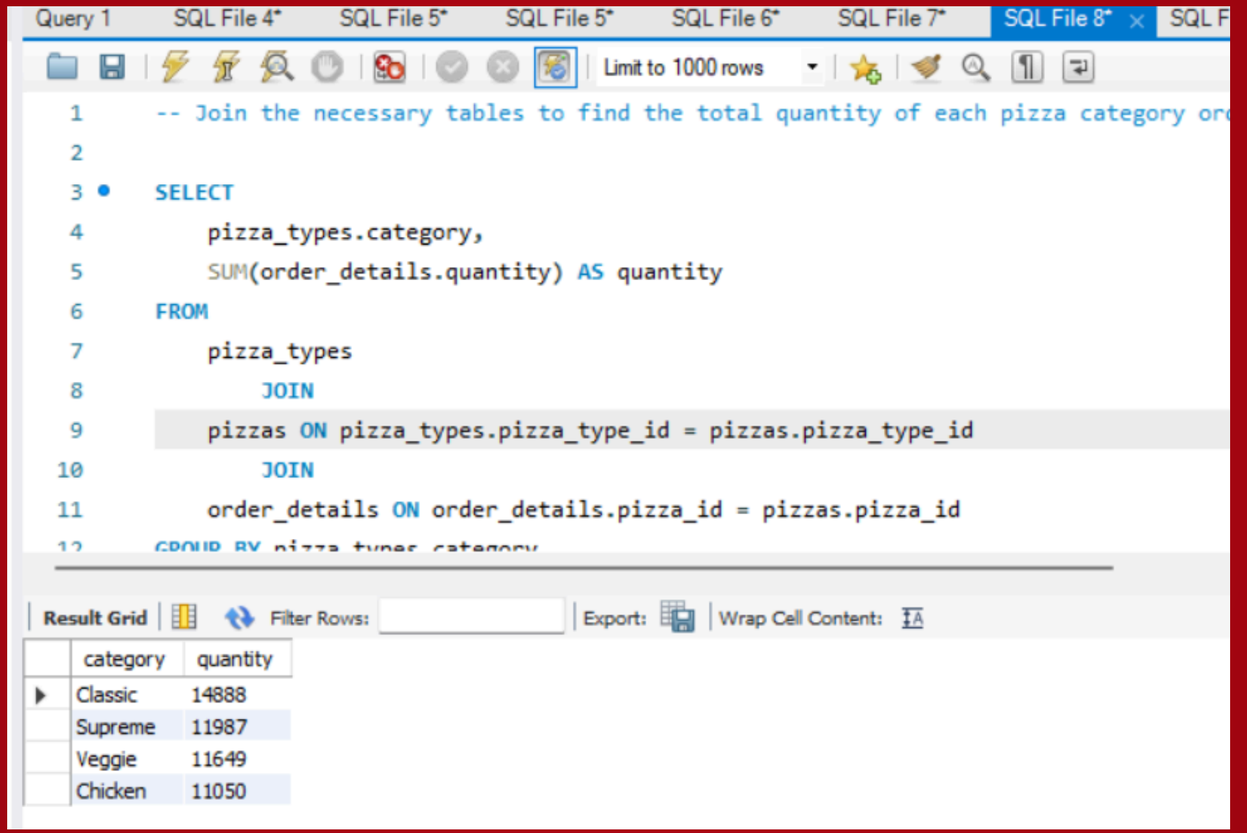
```sql
create table orders (
order_id int not null,
order_date date not null,
order_time time not null,
primary key(order_id) );

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) );
```

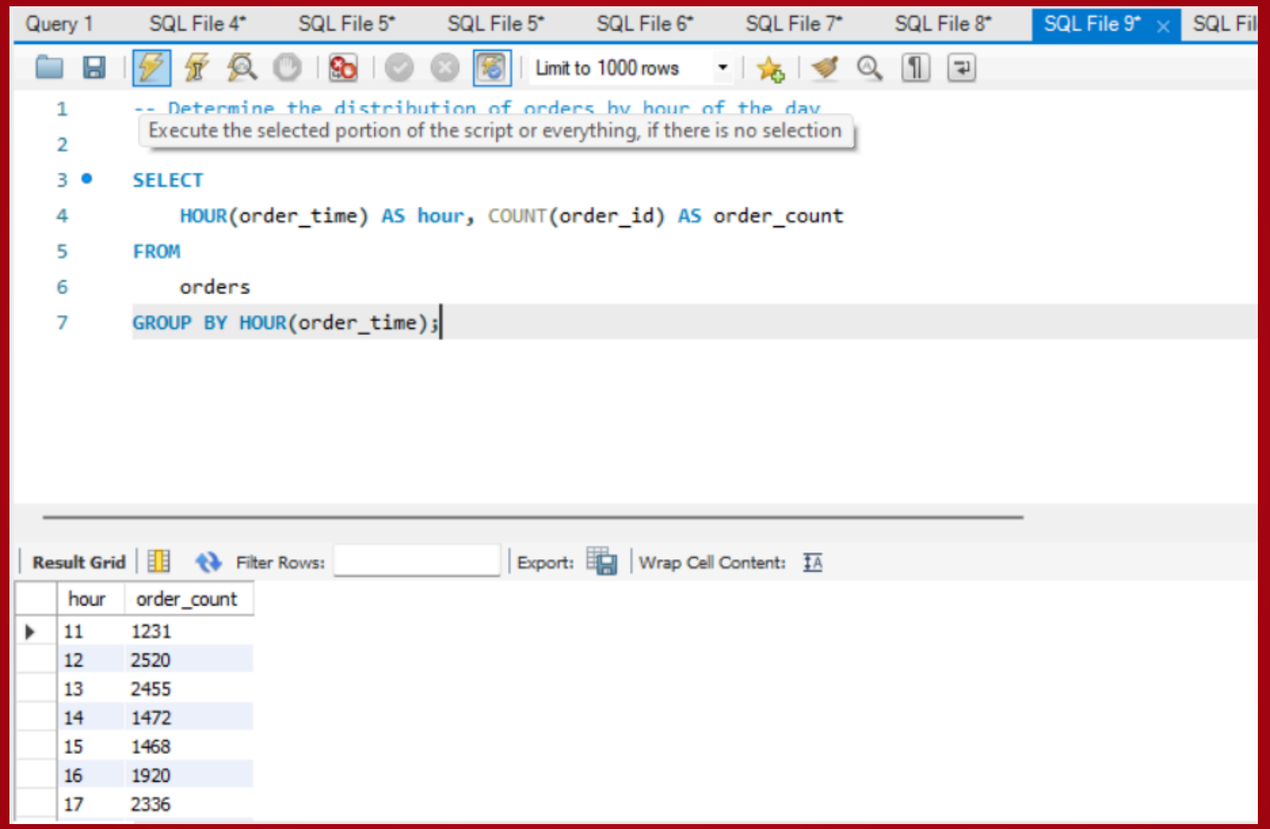

-- Retrieve the total number of orders placed
select count(order_id) as Total_orders from orders;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Total_orders
21350

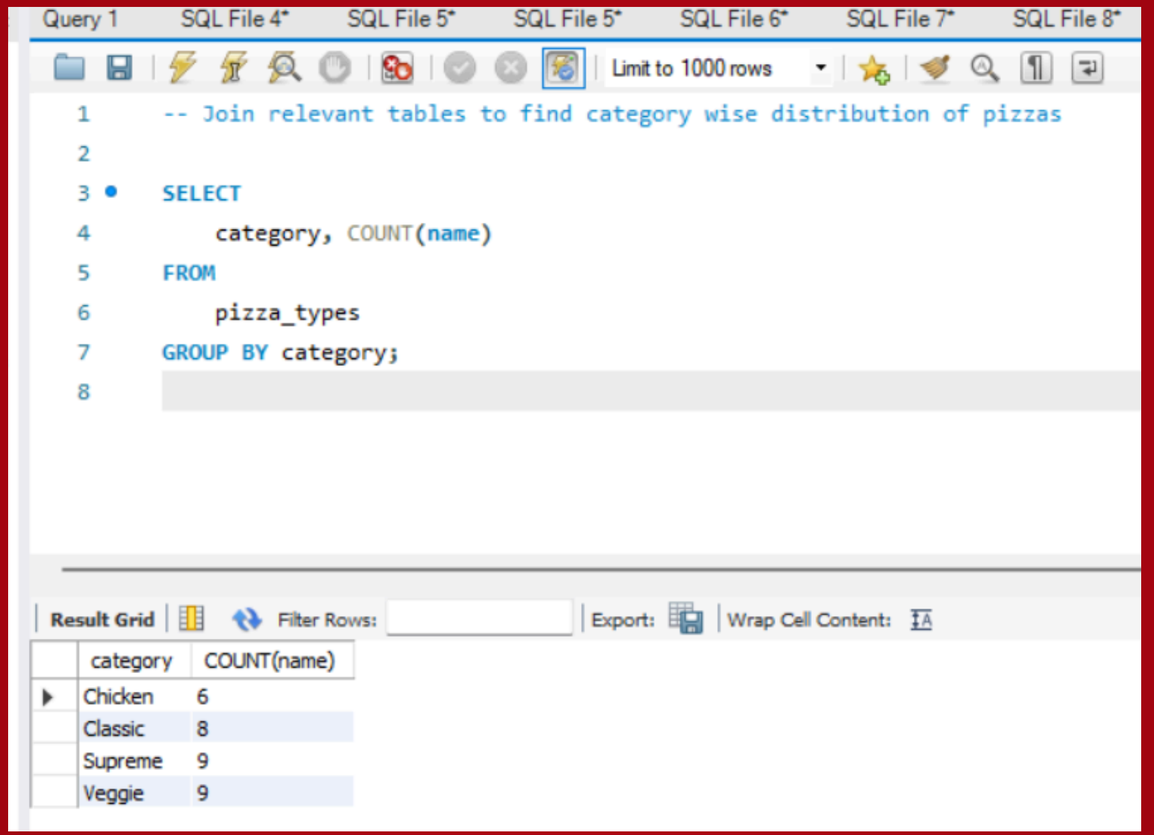

-- Identify the highest-priced pizza
SELECT
pizza_types.name, pizzas.price
FROM
pizza_types
JOIN
pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1; -- 1 is the highest key value
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
name
price
The Greek Pizza
35.95

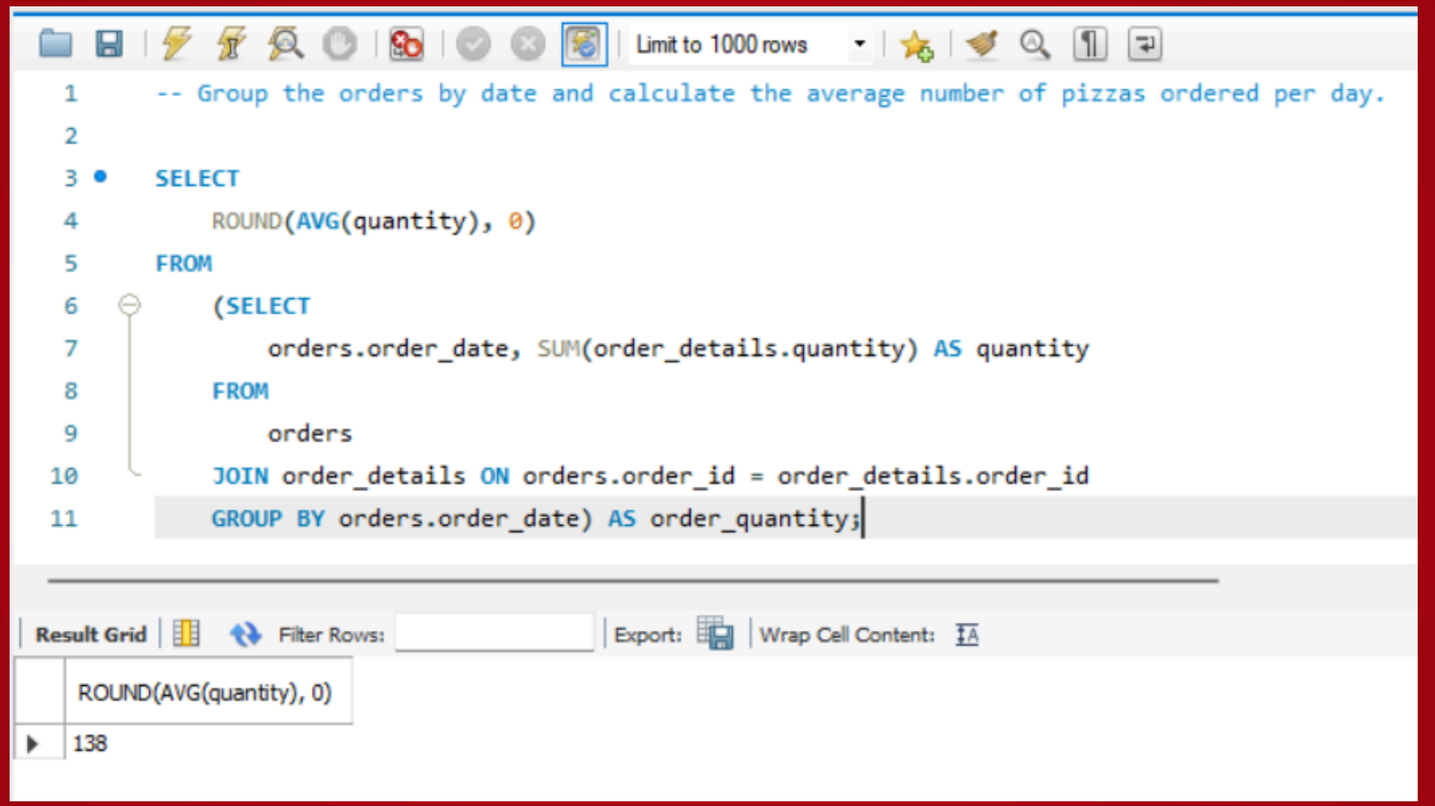

Limit to 1000 rows
-- Identify the most common pizza size ordered
SELECT
pizzas.size,
COUNT(order_details.order_details_id) AS order_count
FROM
pizzas
JOIN
order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
size
order_count
L
18526
M
15385
S
14137
XL
544
XXL
28

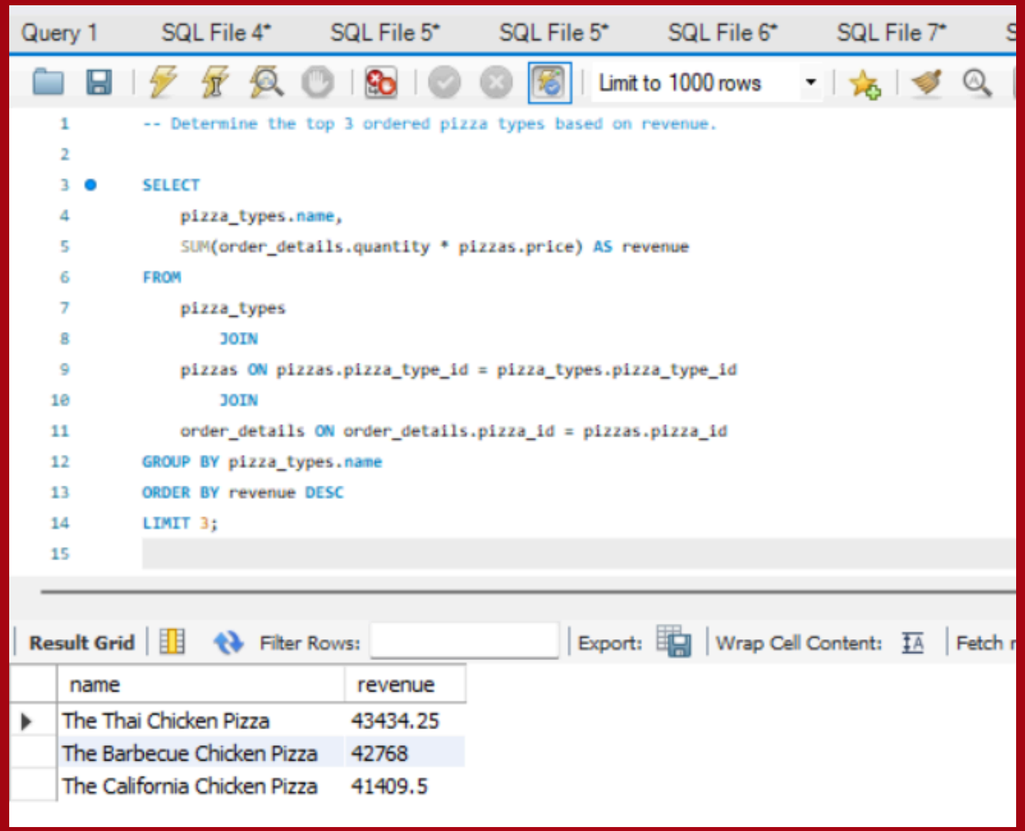

Query 1
SQL File 4*
SQL File 5*
SQL File 5*
SQL File 6*
SQL File 7*
SQL File 8*
SQL File 9*
Limit to 1000 rows
4 pizza_types.name, SUM(order_details.quantity) AS quantity
5 FROM
6 pizza_types
7 JOIN
8 pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9 JOIN
10 order_details ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.name
12 ORDER BY quantity DESC
13 LIMIT 5;
14
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:
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

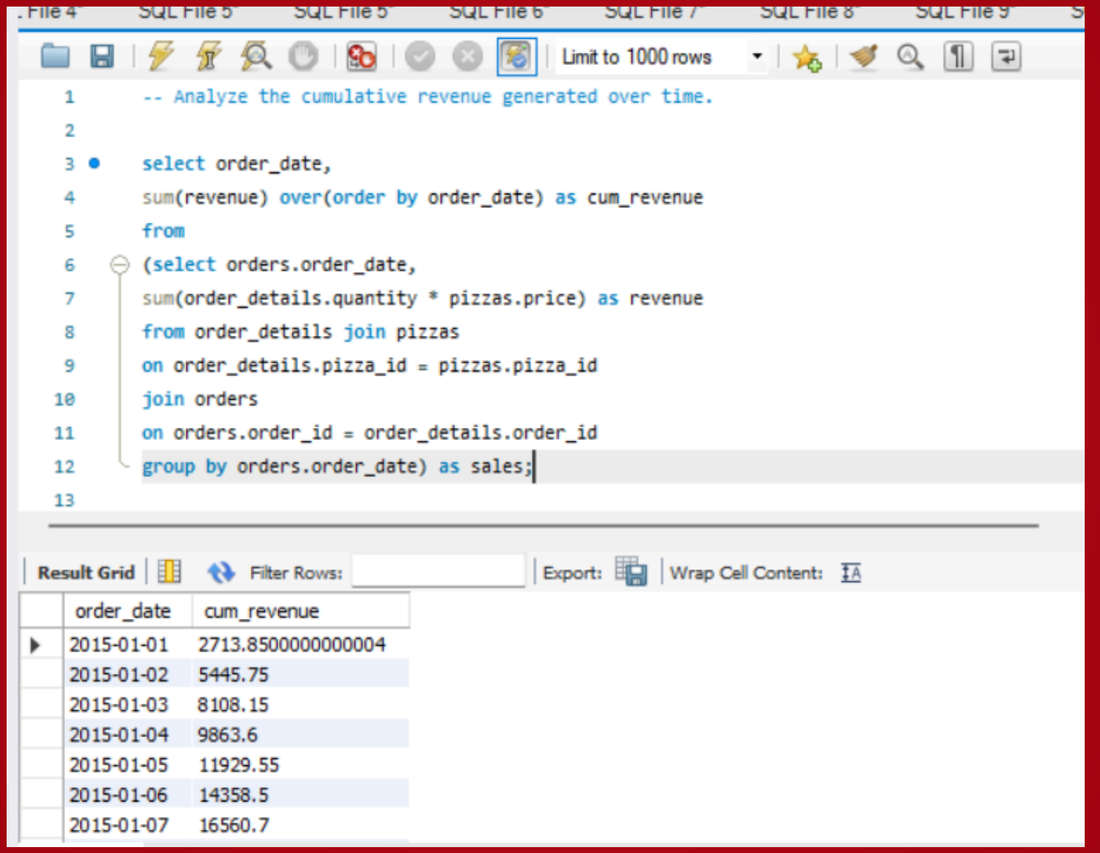

Query 1
SQL File 4*
SQL File 5*
SQL File 5*
SQL File 6*
SQL File 7*
SQL File 8*
Limit to 1000 rows
-- Join the necessary tables to find the total quantity of each pizza category or
SELECT
pizza_types.category,
SUM(order_details.quantity) AS quantity
FROM
pizza_types
JOIN
pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
order_details ON order_details.pizza_id = pizzas.pizza_id
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
category quantity
Classic 14888
Supreme 11987
Veggie 11649
Chicken 11050


Query 1
SQL File 4*
SQL File 5*
SQL File 5*
SQL File 6*
SQL File 7*
SQL File 8*
SQL File 9*
Limit to 1000 rows
-- Determine the distribution of orders by hour of the day
Execute the selected portion of the script or everything, if there is no selection
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 1231
12 2520
13 2455
14 1472
15 1468
16 1920
17 2336


Query 1
SQL File 4*
SQL File 5*
SQL File 5*
SQL File 6*
SQL File 7*
SQL File 8*
Limit to 1000 rows
-- Join relevant tables to find category wise distribution of pizzas
SELECT
category, COUNT(name)
FROM
pizza_types
GROUP BY category;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
category
COUNT(name)
Chicken 6
Classic 8
Supreme 9
Veggie 9


Limit to 1000 rows
-- 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(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:
ROUND(AVG(quantity), 0)
138


Query 1
SQL File 4*
SQL File 5*
SQL File 5*
SQL File 6*
SQL File 7*
Limit to 1000 rows
-- Determine the top 3 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;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch
name
revenue
The Thai Chicken Pizza
43434.25
The Barbecue Chicken Pizza
42768
The California Chicken Pizza
41409.5


SQL File 4
SQL File 5
SQL File 5
SQL File 6
SQL File 7
SQL File 8
SQL File 9
Limit to 1000 rows
-- 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;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
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
2015-01-07
16560.7