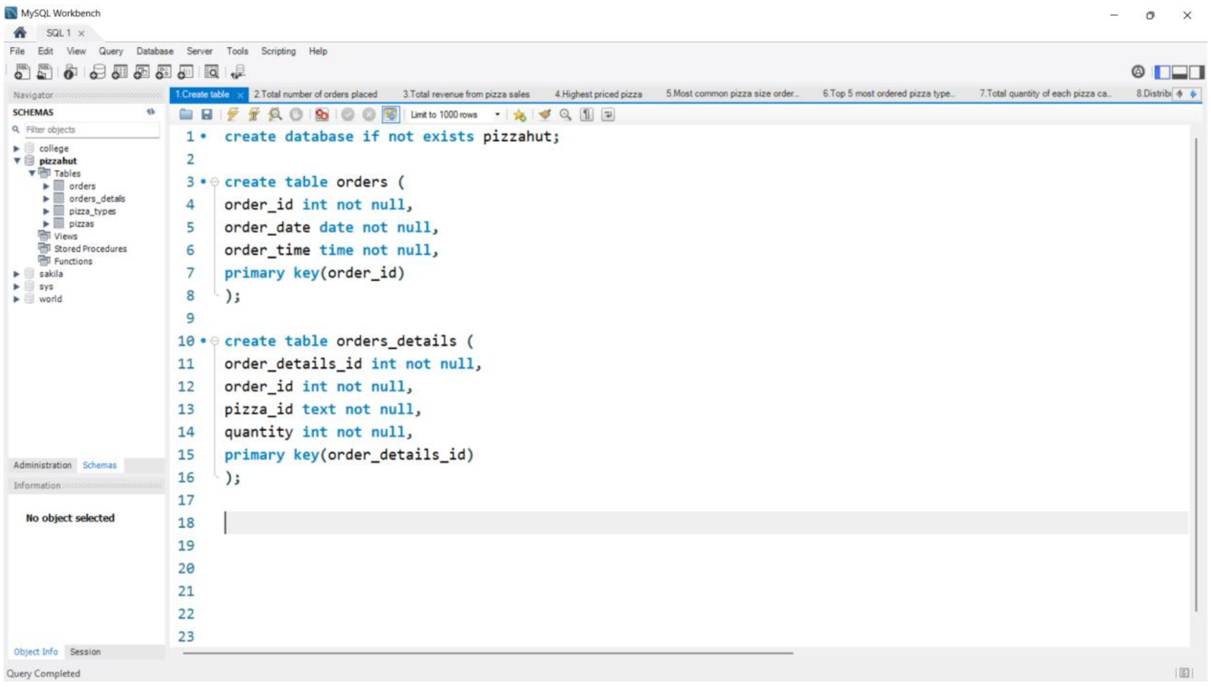
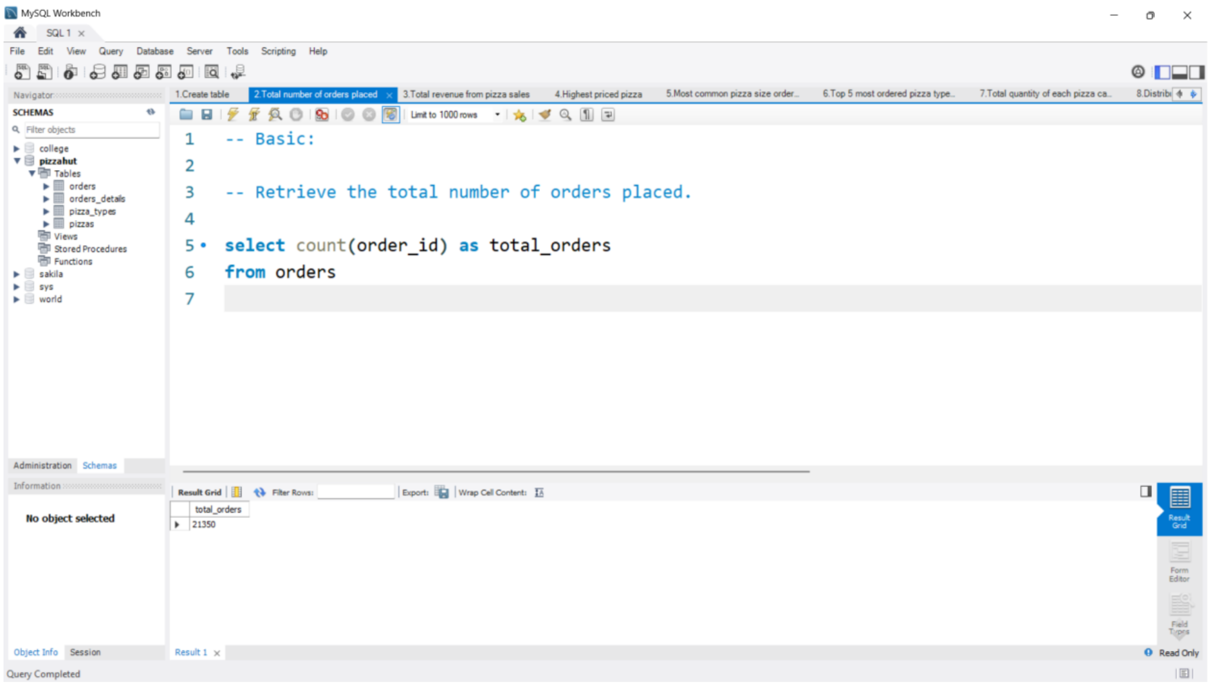
1.Create table



```sql
create database if not exists pizzahut;

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

create table orders_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)
);
```

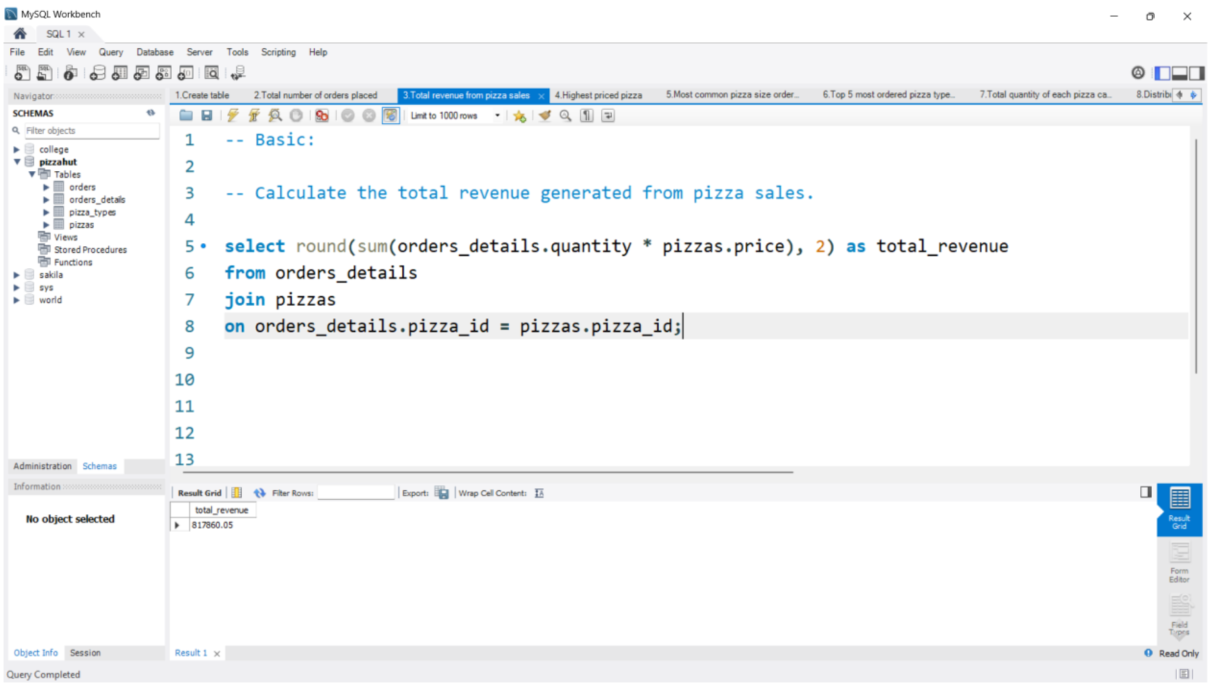
2.Total number of orders placed



```sql
-- Basic:

-- Retrieve the total number of orders placed.

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

total_orders
21350

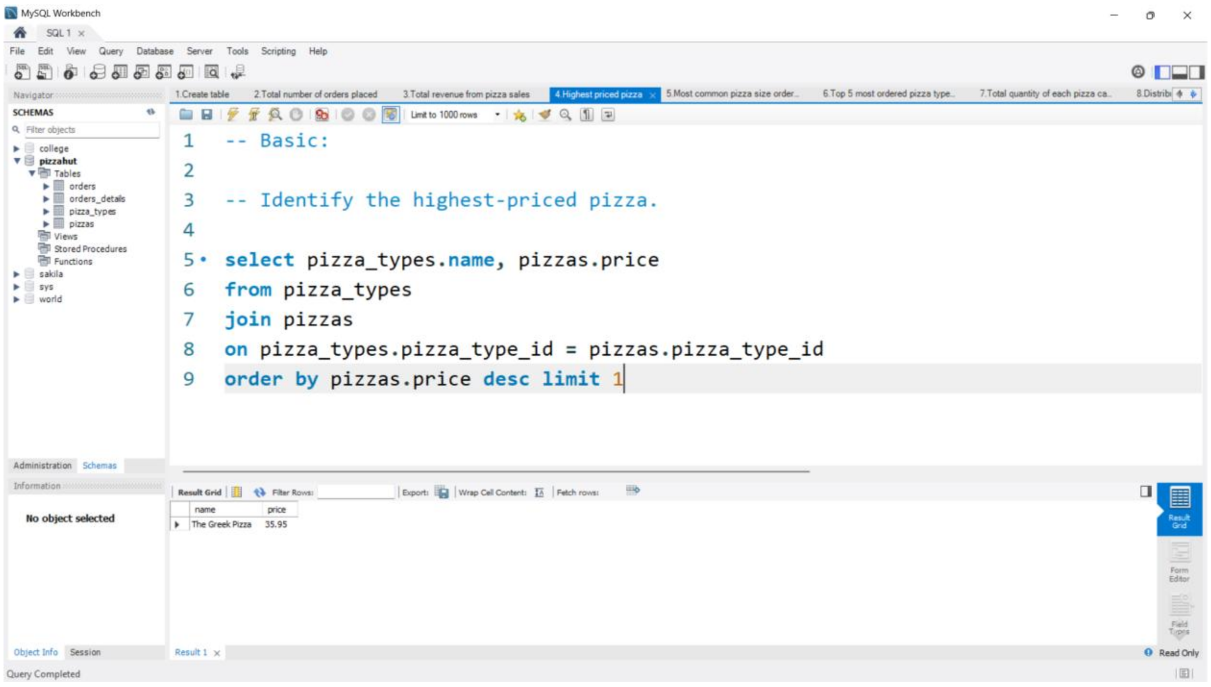
3.Total revenue from pizza sales



```sql
-- Basic:

-- Calculate the total revenue generated from pizza sales.

select round(sum(orders_details.quantity * pizzas.price), 2) as total_revenue
from orders_details
join pizzas
on orders_details.pizza_id = pizzas.pizza_id;
```

```
total_revenue
817860.05
```

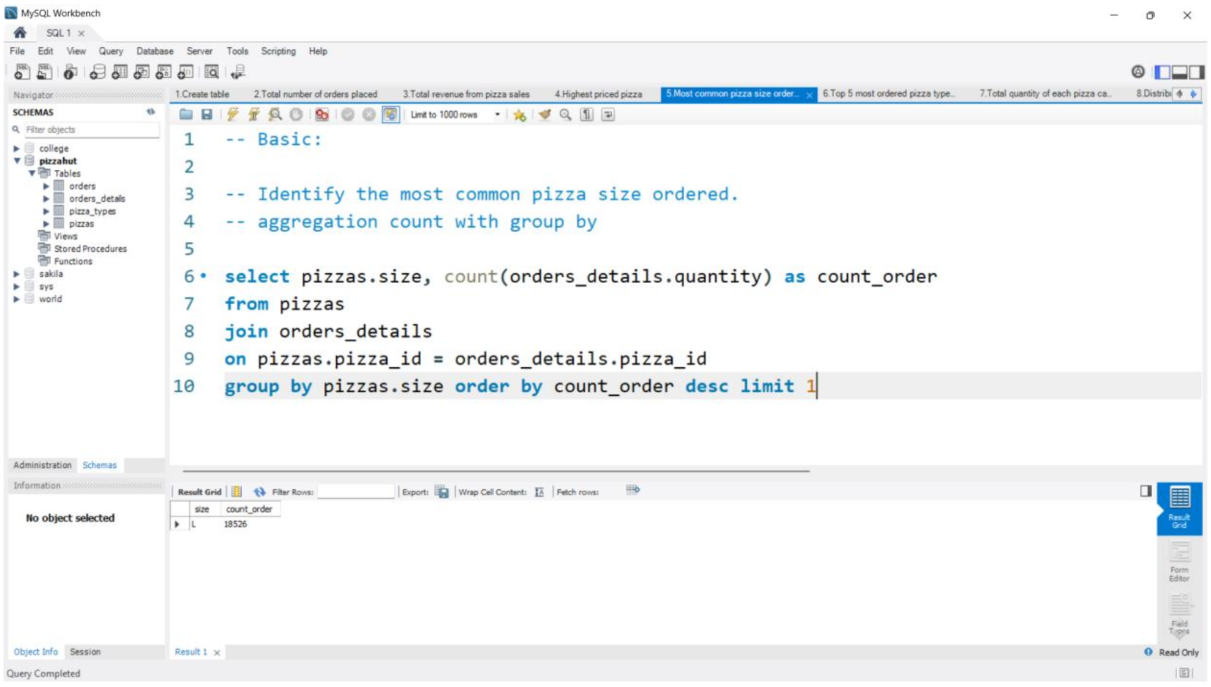
4.Highest priced pizza



```sql
-- Basic:

-- 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
```

5.Most common pizza size ordered

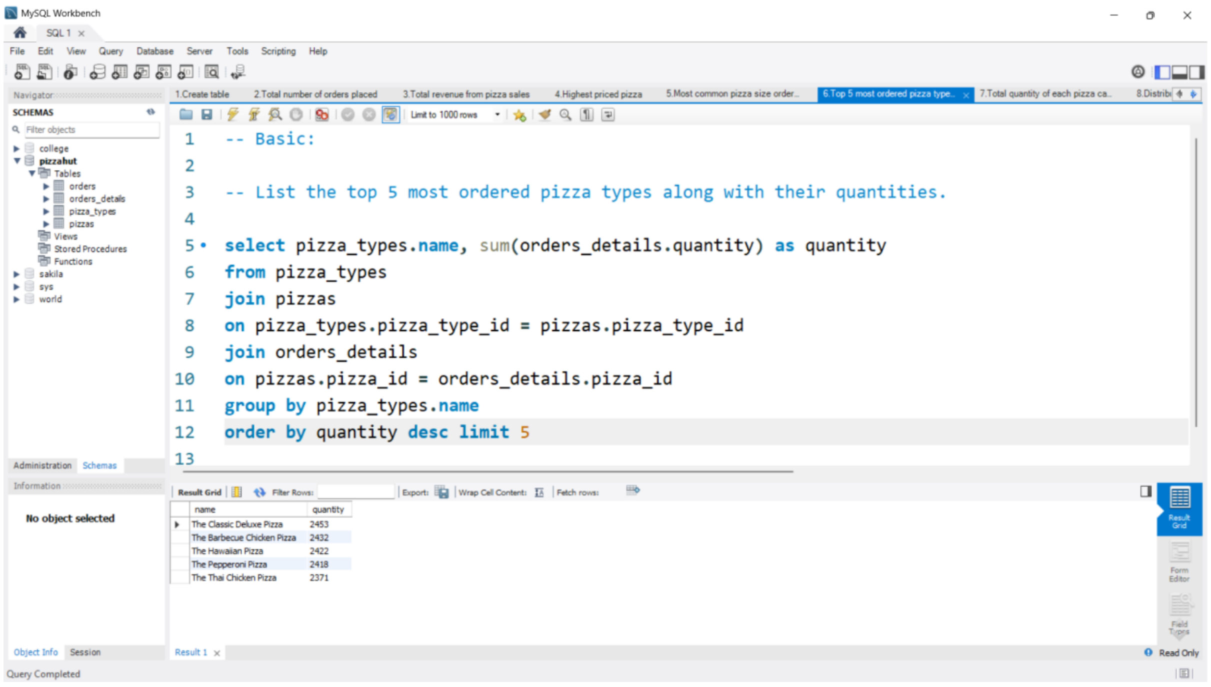


```sql
-- Basic:

-- Identify the most common pizza size ordered.
-- aggregation count with group by

select pizzas.size, count(orders_details.quantity) as count_order
from pizzas
join orders_details
on pizzas.pizza_id = orders_details.pizza_id
group by pizzas.size order by count_order desc limit 1
```

size	count_order
L	18526

Query Completed

6.Top 5 most ordered pizza types with quantities

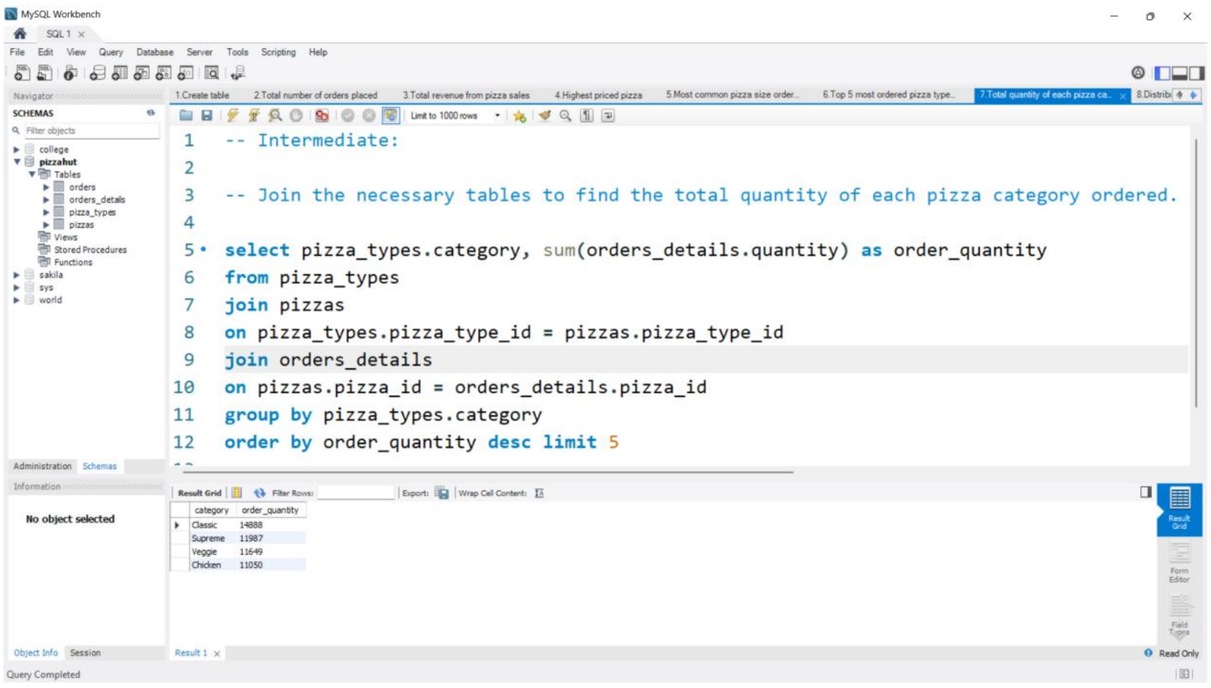


```sql
-- Basic:

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

select pizza_types.name, sum(orders_details.quantity) as quantity
from pizza_types
join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on pizzas.pizza_id = orders_details.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

Query Completed

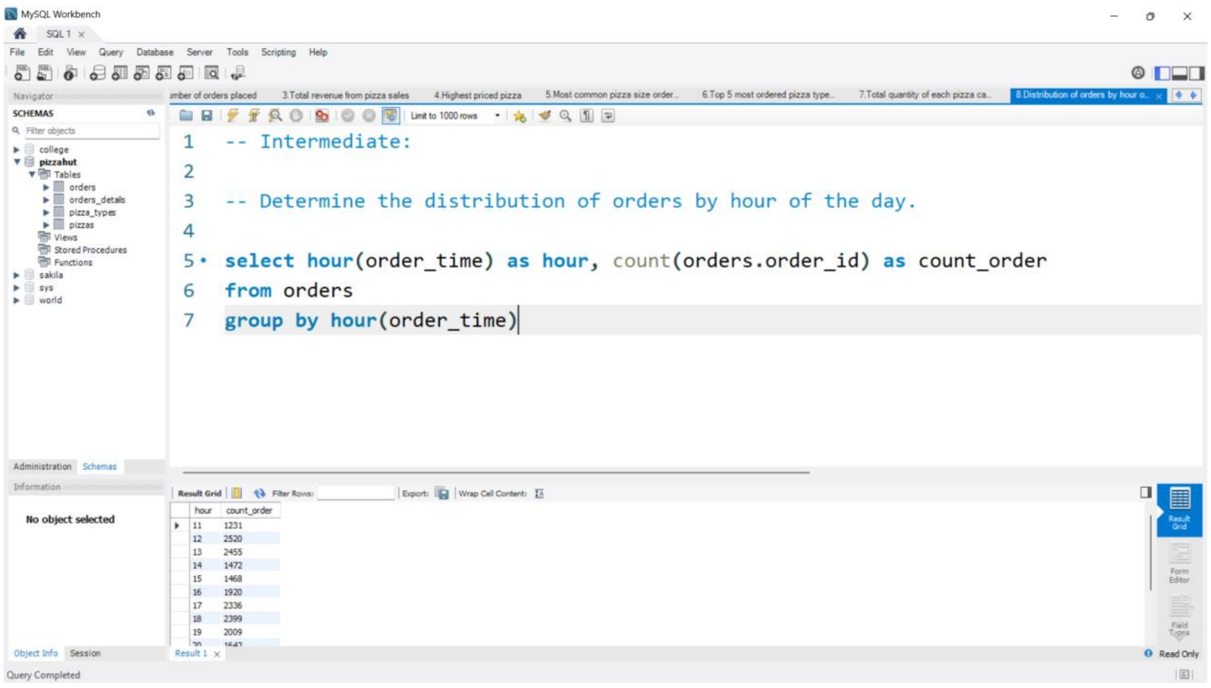
7.Total quantity of each pizza category ordered



```sql
-- Intermediate:

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

select pizza_types.category, sum(orders_details.quantity) as order_quantity
from pizza_types
join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on pizzas.pizza_id = orders_details.pizza_id
group by pizza_types.category
order by order_quantity desc limit 5
```

```
category  order_quantity
Classic   14888
Supreme   11987
Veggie    11649
Chicken   11050
```

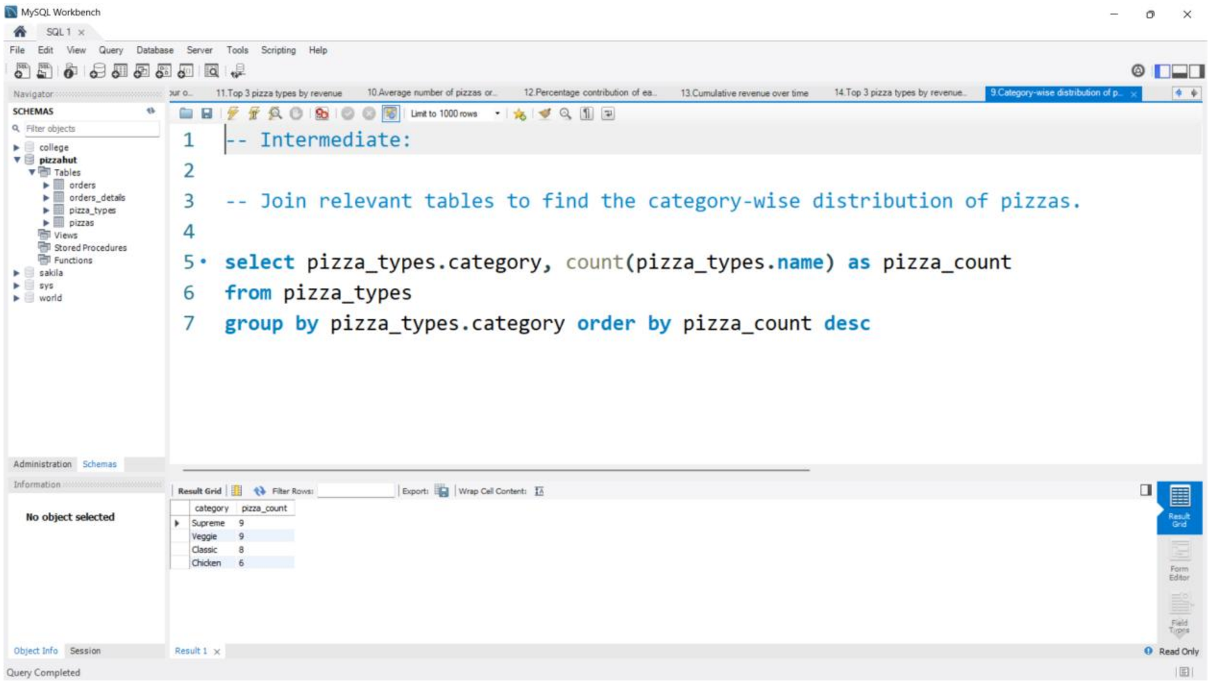
8.Distribution of orders by hour of the day



```sql
-- Intermediate:

-- Determine the distribution of orders by hour of the day.

select hour(order_time) as hour, count(orders.order_id) as count_order
from orders
group by hour(order_time)
```

```
hour  count_order
11    1231
12    2520
13    2455
14    1472
15    1468
16    1920
17    2336
18    2399
19    2009
20    1642
```

9.Category-wise distribution of pizzas

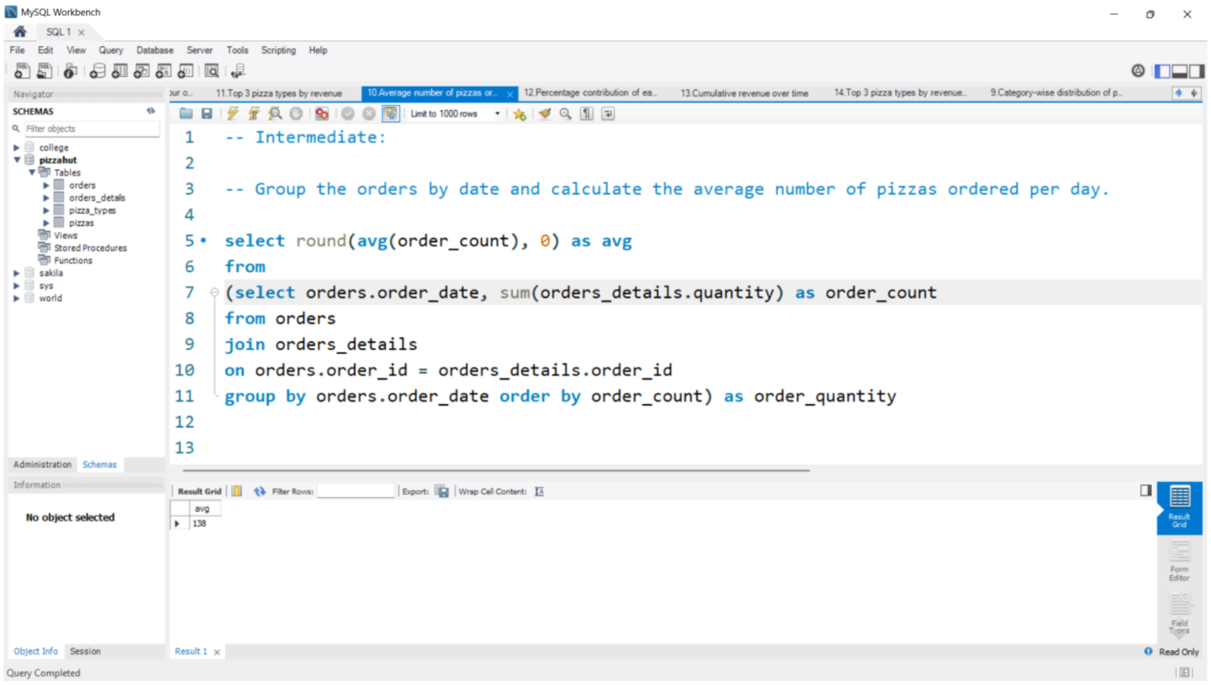


```sql
-- Intermediate:

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

select pizza_types.category, count(pizza_types.name) as pizza_count
from pizza_types
group by pizza_types.category order by pizza_count desc
```

category	pizza_count
Supreme	9
Veggie	9
Classic	8
Chicken	6

Query Completed

10.Average number of pizzas ordered per day

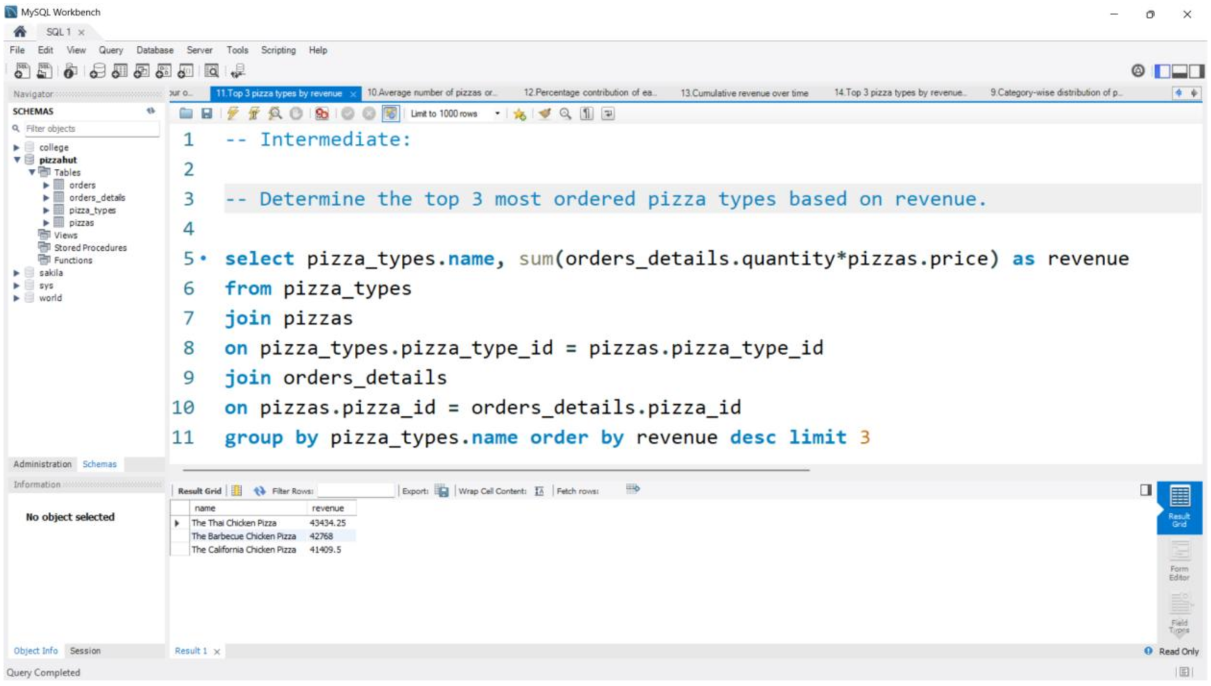


```sql
-- Intermediate:

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

select round(avg(order_count), 0) as avg
from
(select orders.order_date, sum(orders_details.quantity) as order_count
from orders
join orders_details
on orders.order_id = orders_details.order_id
group by orders.order_date order by order_count) as order_quantity
```

avg
138

Query Completed

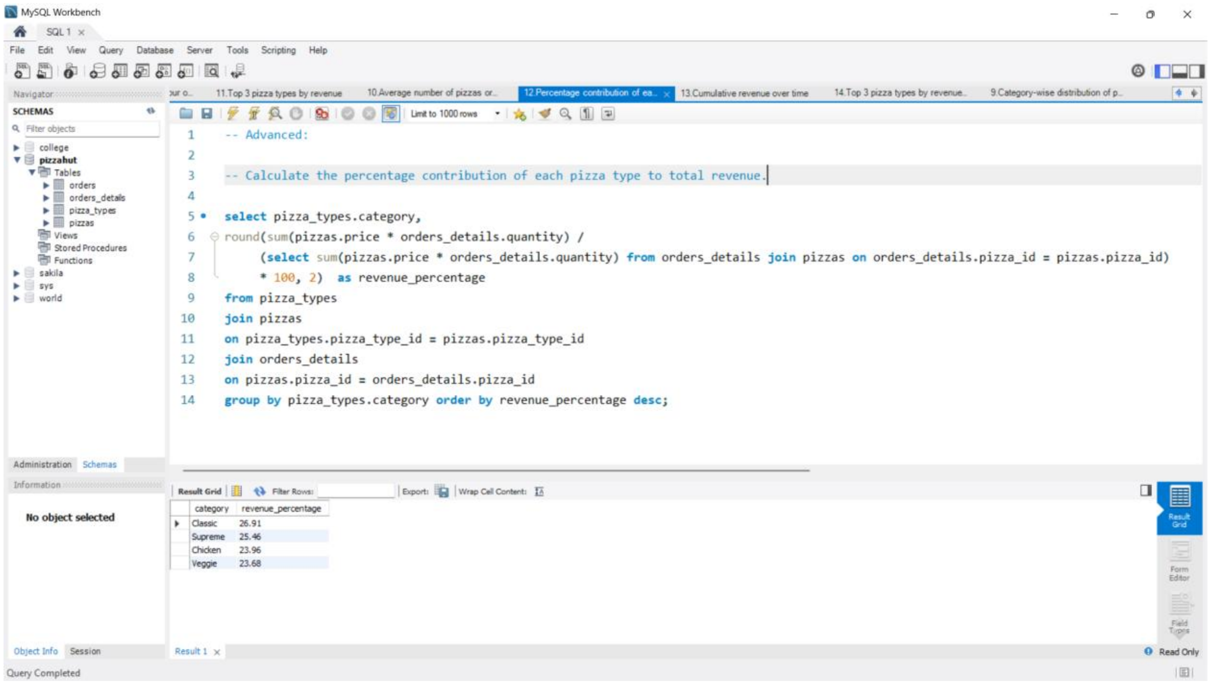
11.Top 3 pizza types by revenue



```sql
-- Intermediate:

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

select pizza_types.name, sum(orders_details.quantity*pizzas.price) as revenue
from pizza_types
join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on pizzas.pizza_id = orders_details.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

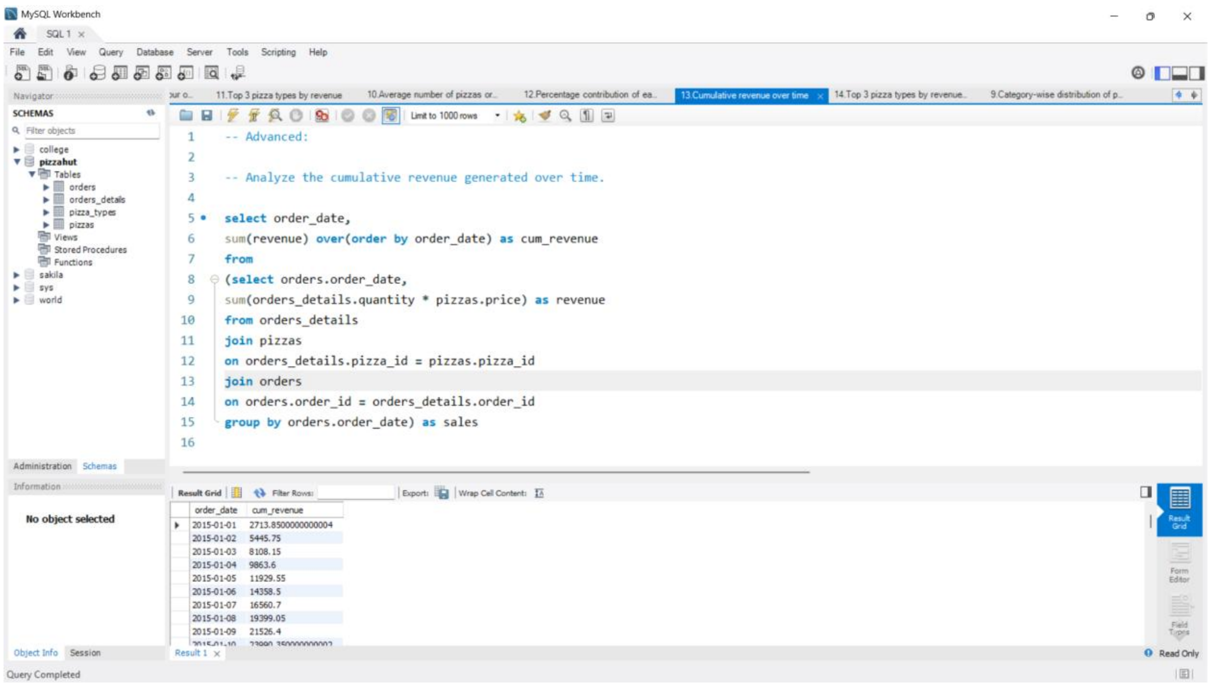
12.Percentage contribution of each pizza type to total revenue



```sql
-- Advanced:

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

select pizza_types.category,
round(sum(pizzas.price * orders_details.quantity) /
    (select sum(pizzas.price * orders_details.quantity) from orders_details join pizzas on orders_details.pizza_id = pizzas.pizza_id)
    * 100, 2)  as revenue_percentage
from pizza_types
join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on pizzas.pizza_id = orders_details.pizza_id
group by pizza_types.category order by revenue_percentage desc;
```

category	revenue_percentage
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

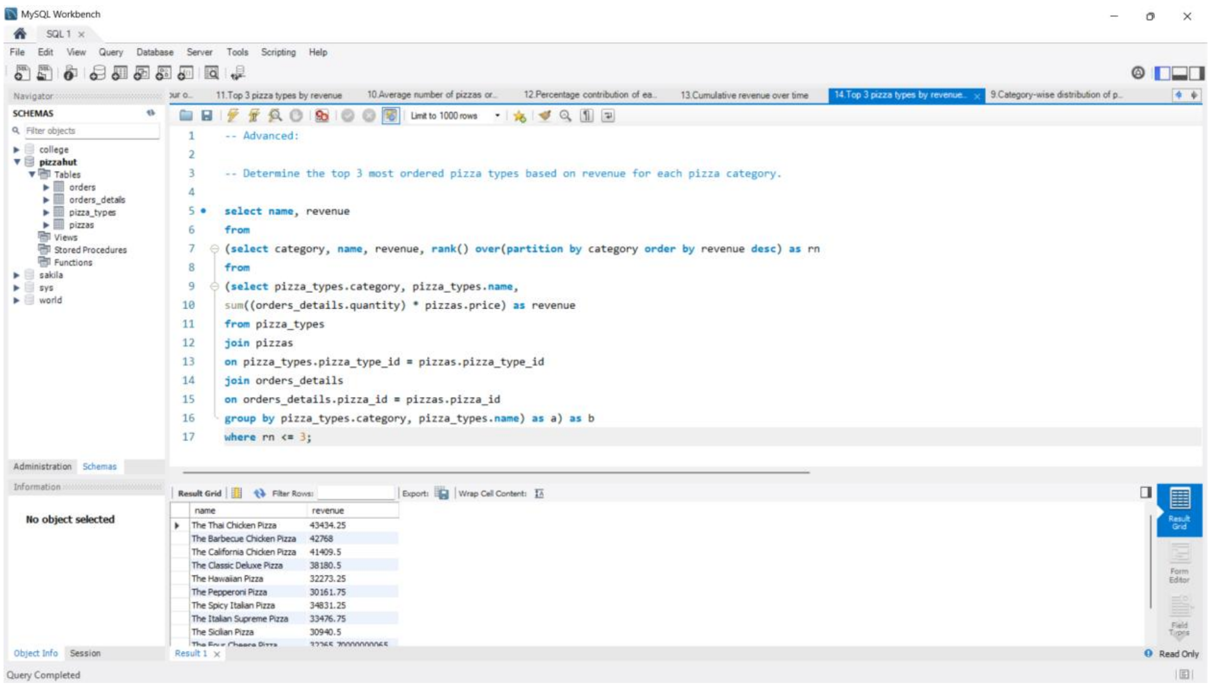
13.Cumulative revenue over time



```sql
-- Advanced:

-- 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(orders_details.quantity * pizzas.price) as revenue
from orders_details
join pizzas
on orders_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = orders_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
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002

14.Top 3 pizza types by revenue for each category



```sql
-- Advanced:

-- 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 rn
from
(select pizza_types.category, pizza_types.name,
sum((orders_details.quantity) * pizzas.price) as revenue
from pizza_types
join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_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
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065