PIZZA SALES ANALYSIS

SQL PROJECT

Objective

The objective of the Pizza Sales Analysis project is to Utilize SQL queries to analyse sales data, identify key performance metrices, and derive actionable insights to optimize business operations. This includes Total Revenue by dates, Peak Sales Period, Popular Pizza types, understanding customer preferences etc. The analysis aims to support data-driven decision making to enhance sales strategies, improve inventory management and increase revenue.

Questions

1. Retrieve the total number of orders placed.

2. Calculate the total revenue generated from pizza sales.

3. Identify the highest-priced pizza.

4. Identify the most common pizza size ordered.

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

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

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

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

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

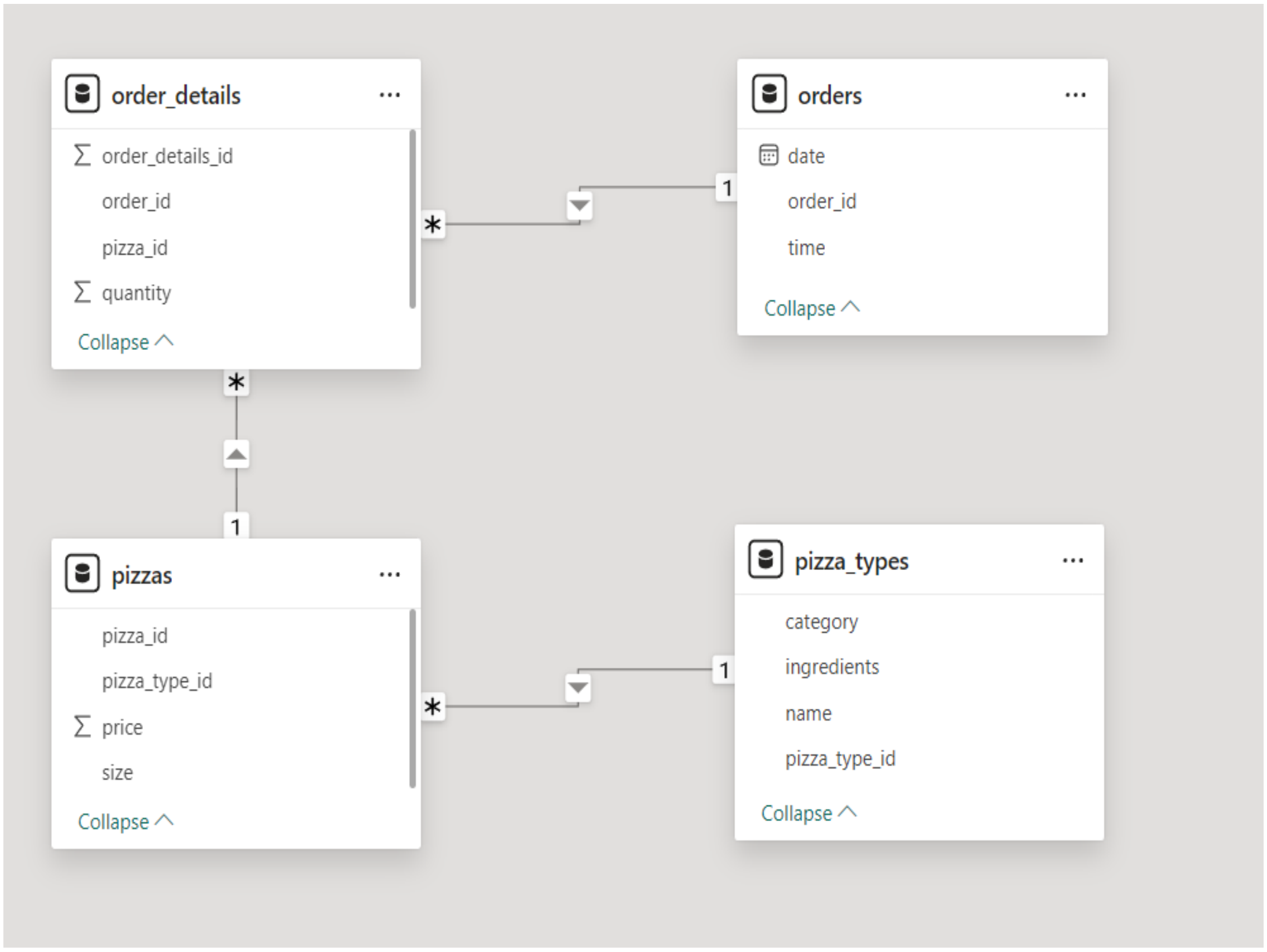
10. Determine the top 3 most ordered pizza types based on revenue.

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

12. Analyse the cumulative revenue generated over time.

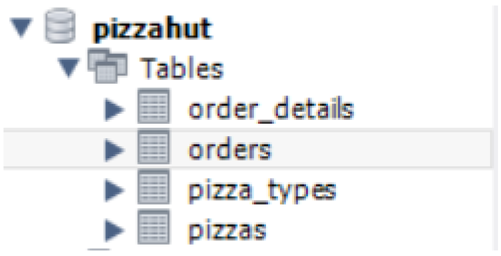
13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Data Model View



Database:

Pizzahut



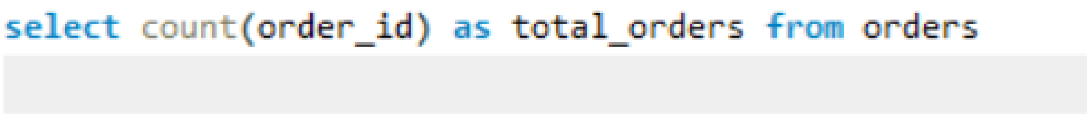
Tool:

MySQL Workbench

Question 1:

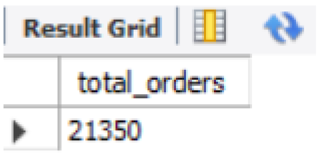
Retrieve the total number of orders placed.

Code:



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

Result:

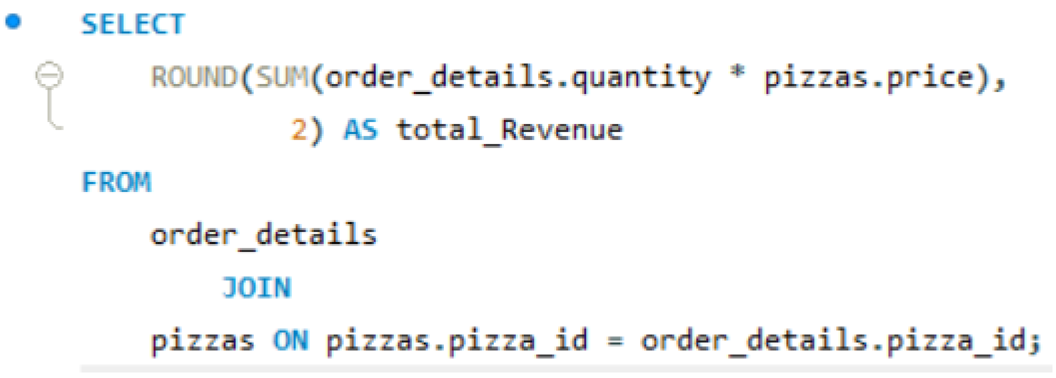


```
total_orders
21350
```

Question 2:

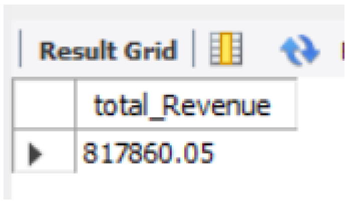
Calculate the total revenue generated from pizza sales.

Code:



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

Result:



```
total_Revenue
817860.05
```

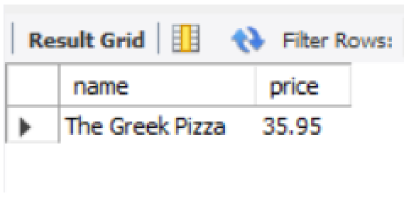
Question 3:

Identify the highest-priced pizza.

Code:

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

Result:


Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

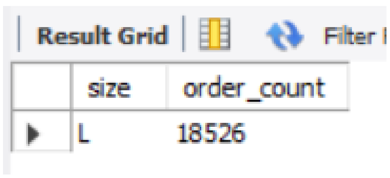
Question 4:

Identify the most common pizza size ordered.

Code:

```sql
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
LIMIT 1;
```

Result:


Result Grid | Filter

size	order_count
L	18526

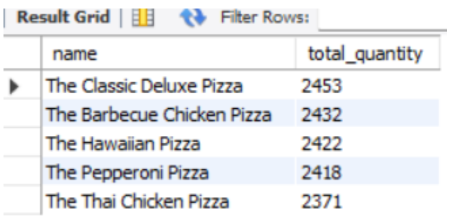
Question 5:

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

Code:

```
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS total_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 total_quantity DESC
LIMIT 5;
```

Result:



Result Grid | Filter Rows:

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

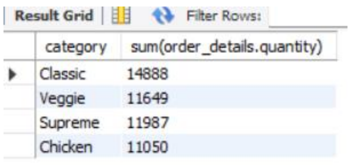
Question 6:

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

Code:

```
SELECT
    pizza_types.category, SUM(order_details.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;
```

Result:



Result Grid | Filter Rows:

category	sum(order_details.quantity)
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

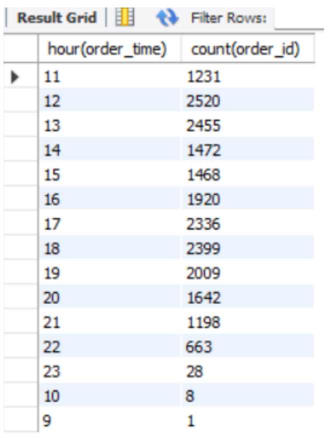
Question 7:

Determine the distribution of orders by hour of the day.

Code:

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

Result:



hour(order_time)	count(order_id)
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

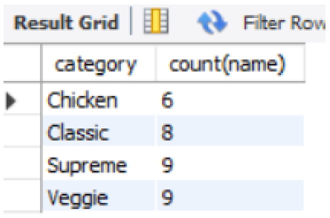
Question 8:

Find category-wise distribution of pizzas.

Code:

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

Result:

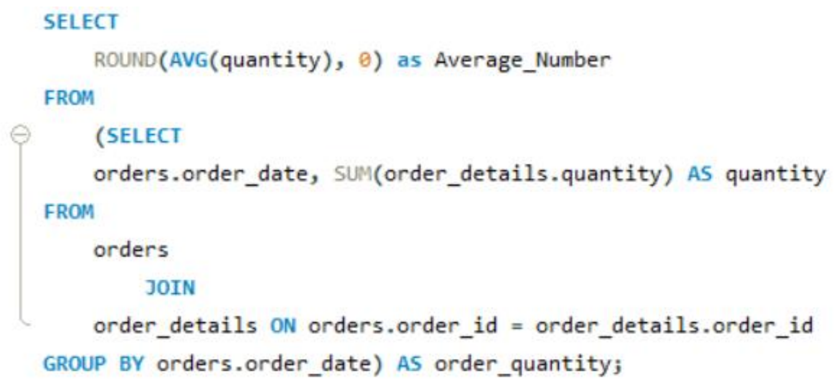


category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

Question 9:

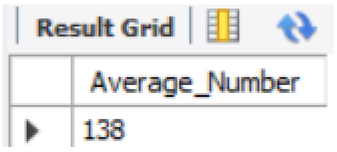
Group the orders by date and calculate the average number of pizzas ordered per day.

Code:



```
SELECT
    ROUND(AVG(quantity), 0) as Average_Number
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;
```

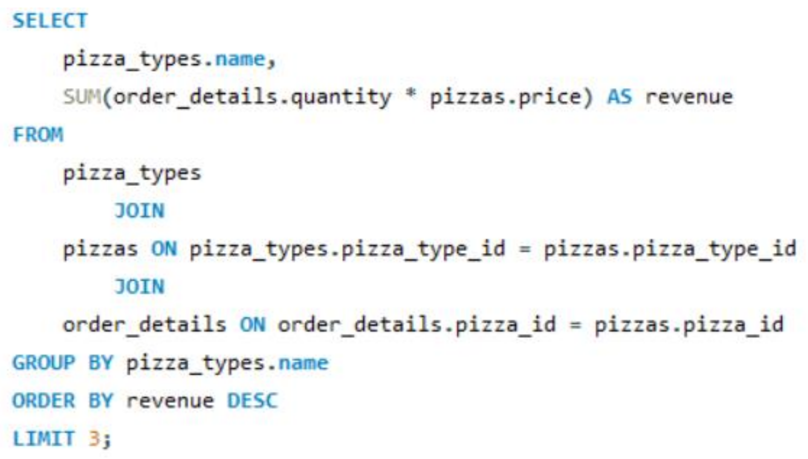
Result:



```
Average_Number
138
```

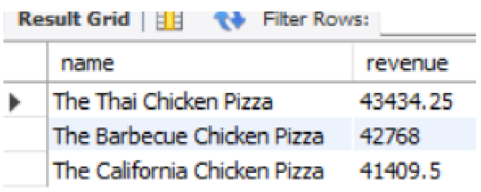
Question 10:

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 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;
```

Result:



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

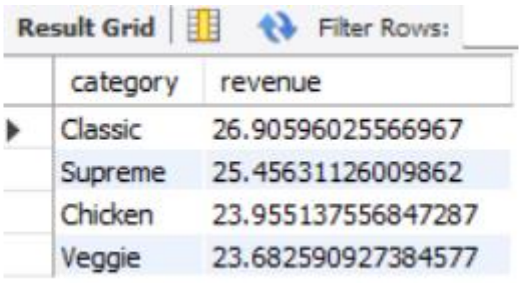
Question 11:

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

Code:

```
SELECT
    pizza_types.category,
    (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 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:



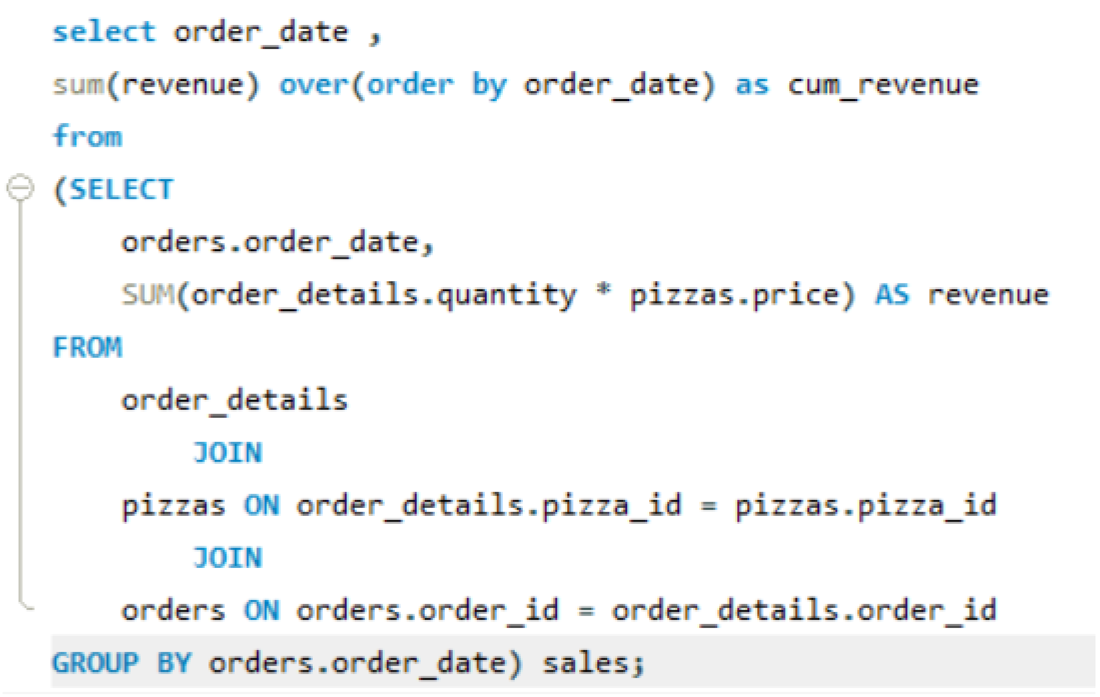
Result Grid | Filter Rows:

category	revenue
Classic	26.90596025566967
Supreme	25.45631126009862
Chicken	23.955137556847287
Veggie	23.682590927384577

Question 12:

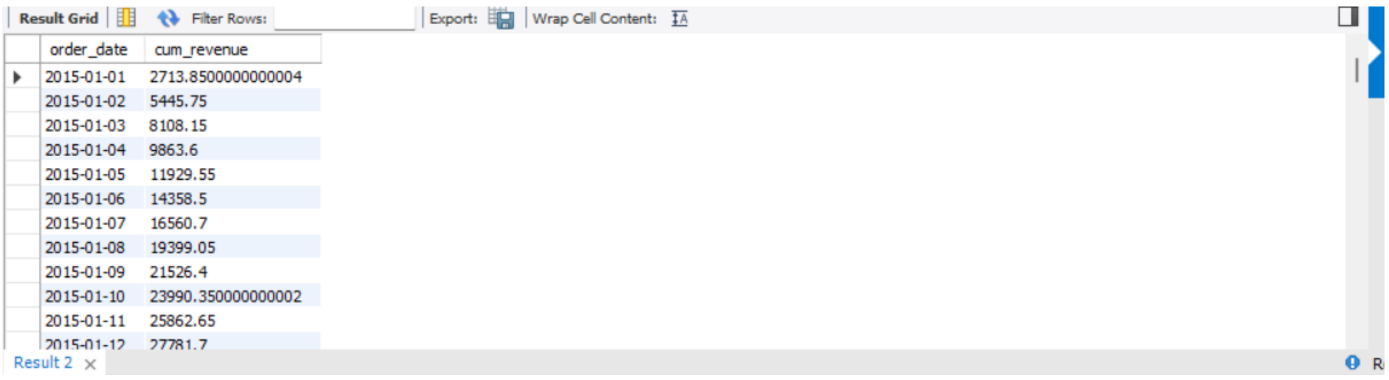
Analyse the cumulative revenue generated over time.

Code:



```
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) sales;
```

Result:

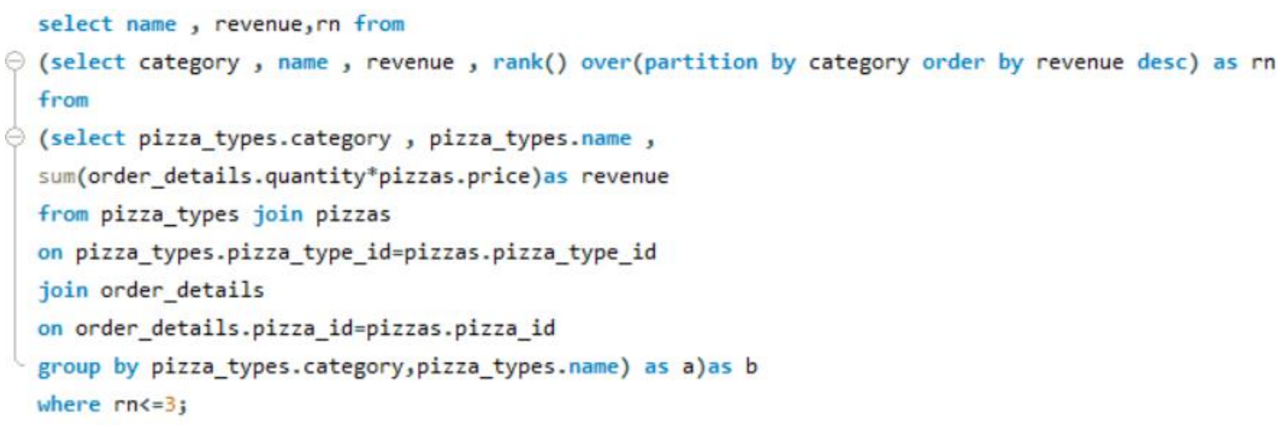


order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7

Question 13:

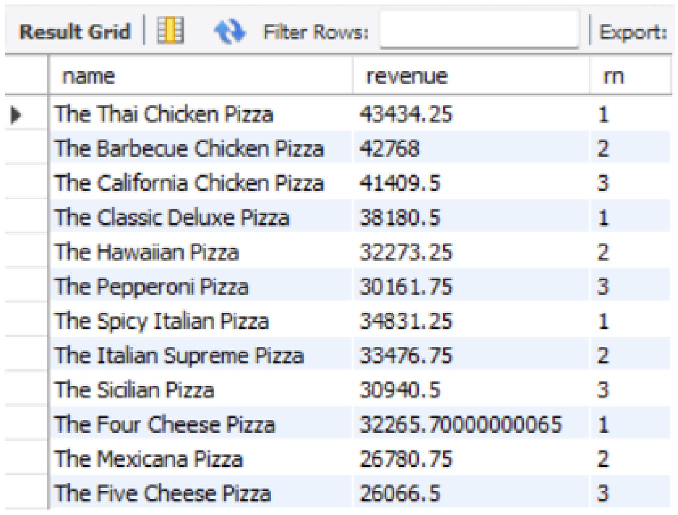
Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Code:



```
select name , revenue,rn 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;
```

Result:



name	revenue	rn
The Thai Chicken Pizza	43434.25	1
The Barbecue Chicken Pizza	42768	2
The California Chicken Pizza	41409.5	3
The Classic Deluxe Pizza	38180.5	1
The Hawaiian Pizza	32273.25	2
The Pepperoni Pizza	30161.75	3
The Spicy Italian Pizza	34831.25	1
The Italian Supreme Pizza	33476.75	2
The Sicilian Pizza	30940.5	3
The Four Cheese Pizza	32265.70000000065	1
The Mexicana Pizza	26780.75	2
The Five Cheese Pizza	26066.5	3