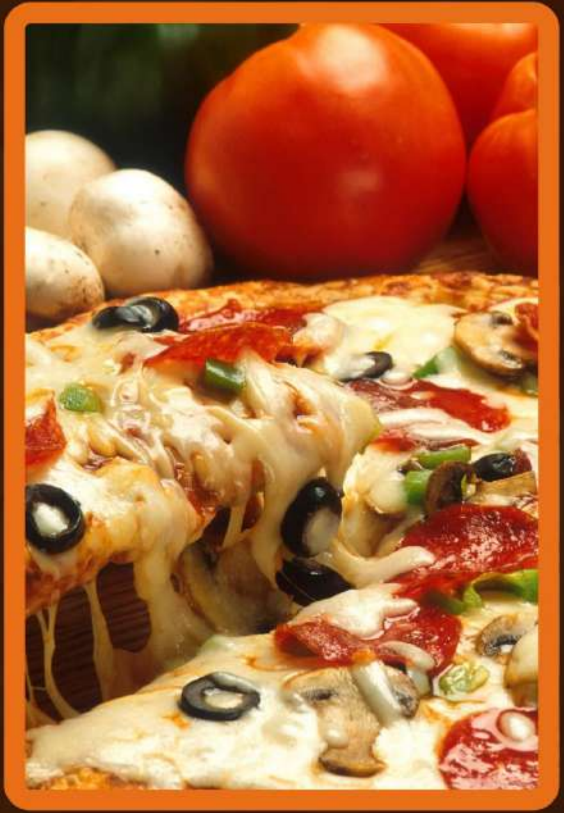
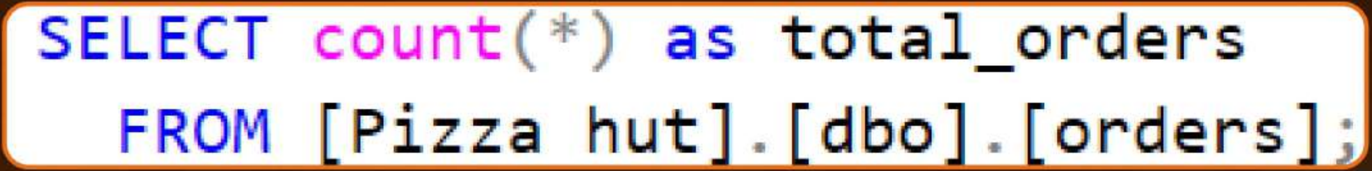
PIZZA SALES ANALYSIS



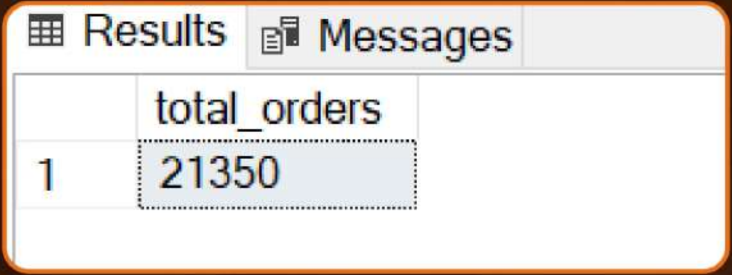
OVERVIEW

For this project, I played the role of a Data Analyst for a pizza restaurant chain. The objective was to develop an SQL-based solution to analyze key business metrics related to pizza sales. Using SQL Server, I queried the database to extract insights such as total orders, revenue generation, pizza popularity, and sales trends. The project involved joining multiple tables, performing aggregations, and deriving meaningful business insights to support decision-making.

Retrieve the Total Number of Order Placed.

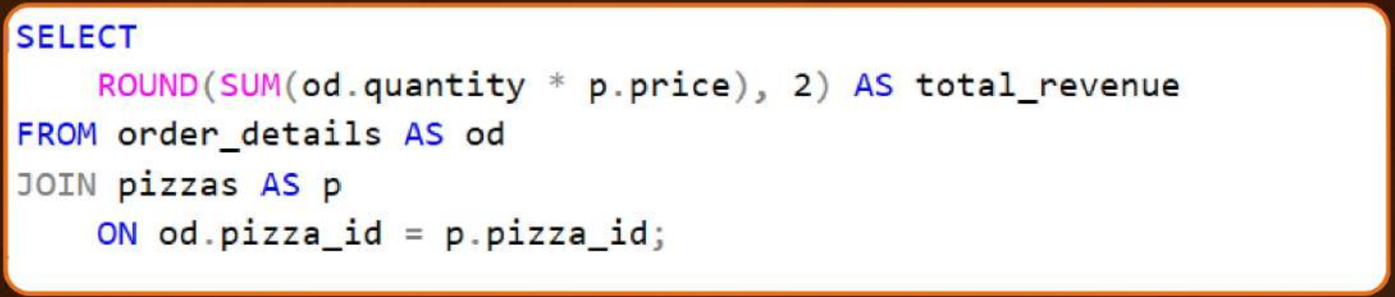


```
SELECT count(*) as total_orders
  FROM [Pizza hut].[dbo].[orders];
```

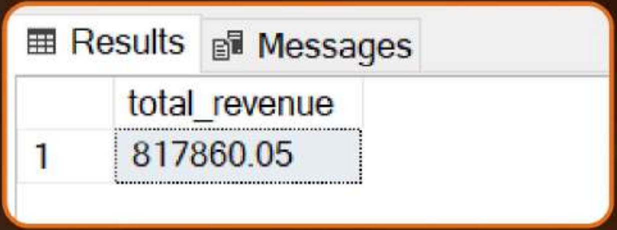


```
   total_orders
1  21350
```

Calculate the total revenue generated from pizza sales.

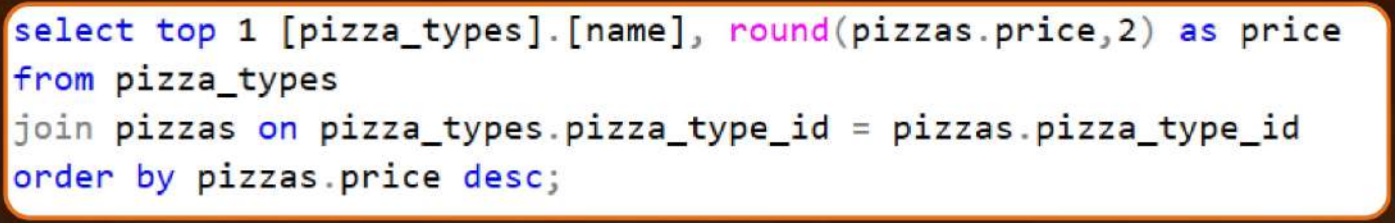


```
SELECT
    ROUND(SUM(od.quantity * p.price), 2) AS total_revenue
FROM order_details AS od
JOIN pizzas AS p
    ON od.pizza_id = p.pizza_id;
```

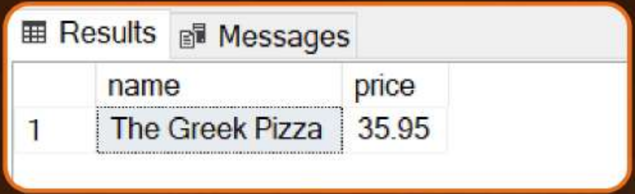


```
     total_revenue
1    817860.05
```

Identify the Highest Price Pizza.

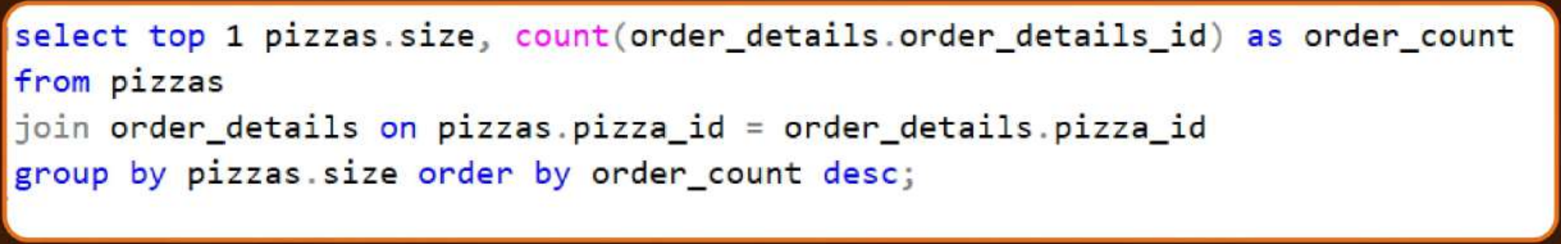


```
select top 1 [pizza_types].[name], round(pizzas.price,2) as price
from pizza_types
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
order by pizzas.price desc;
```

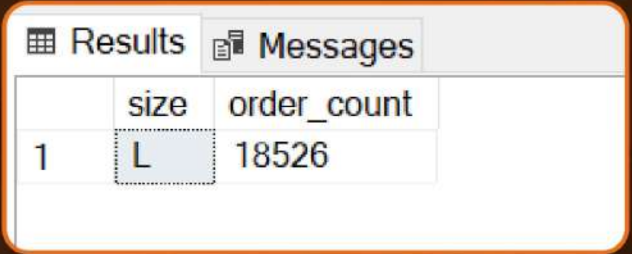


	name	price
1	The Greek Pizza	35.95

Identify the Most Common Pizza Size Ordered.

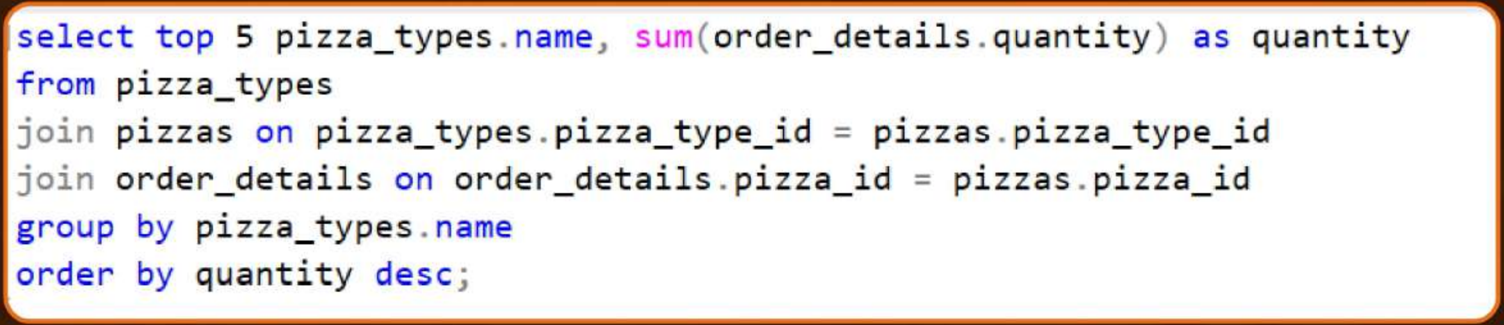


```
select top 1 pizzas.size, count(order_details.order_details_id) as order_count
from pizzas
join order_details on pizzas.pizza_id = order_details.pizza_id
group by pizzas.size order by order_count desc;
```

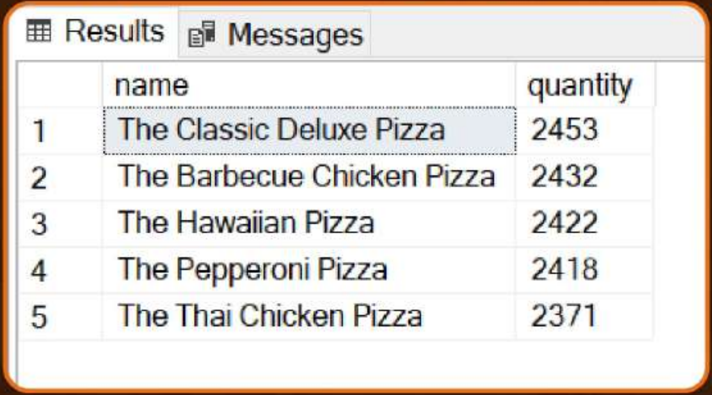


```
   size  order_count
1  L     18526
```

List the Top 5 Most Ordered Pizza Types Along with Their Types.

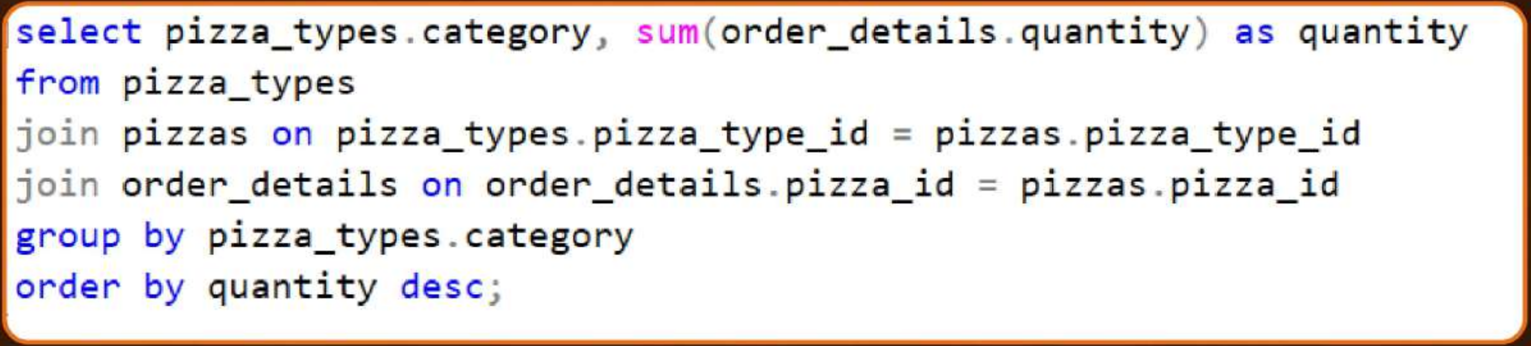


```
select top 5 pizza_types.name, sum(order_details.quantity) as quantity
from pizza_types
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name
order by quantity desc;
```

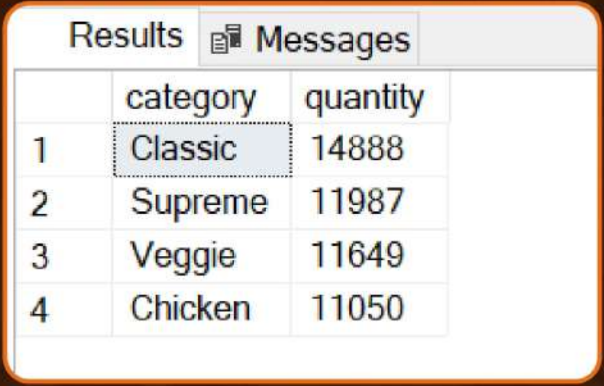


	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

Join the Necessary Tables to Find the Total Quantity of Each Pizza Category Ordered.



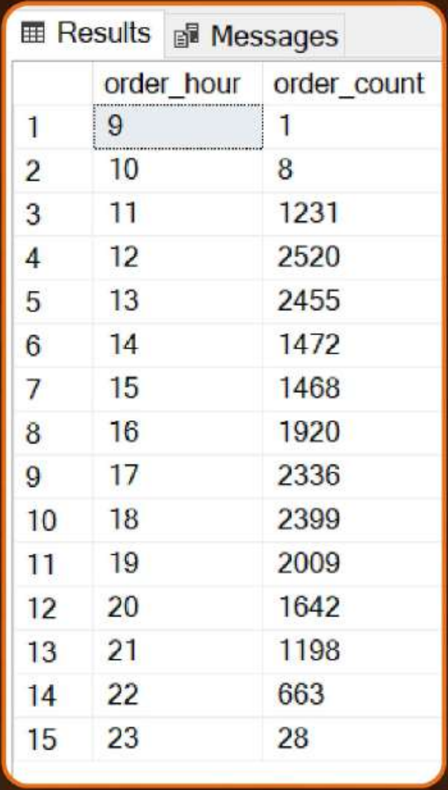
```
select pizza_types.category, sum(order_details.quantity) as quantity
from pizza_types
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category
order by quantity desc;
```



	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

Determine the Distribution of Orders by Hour of the Day.

```
SELECT DATEPART(HOUR, [orders].[time]) AS order_hour,
       COUNT(order_id) AS order_count
FROM [orders]
GROUP BY DATEPART(HOUR, [orders].[time])
ORDER BY order_hour;
```

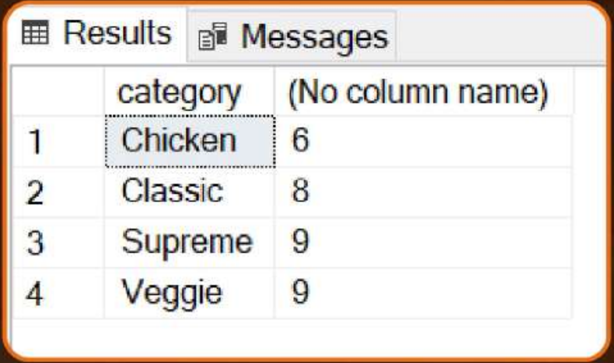


Results | Messages

	order_hour	order_count
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399
11	19	2009
12	20	1642
13	21	1198
14	22	663
15	23	28

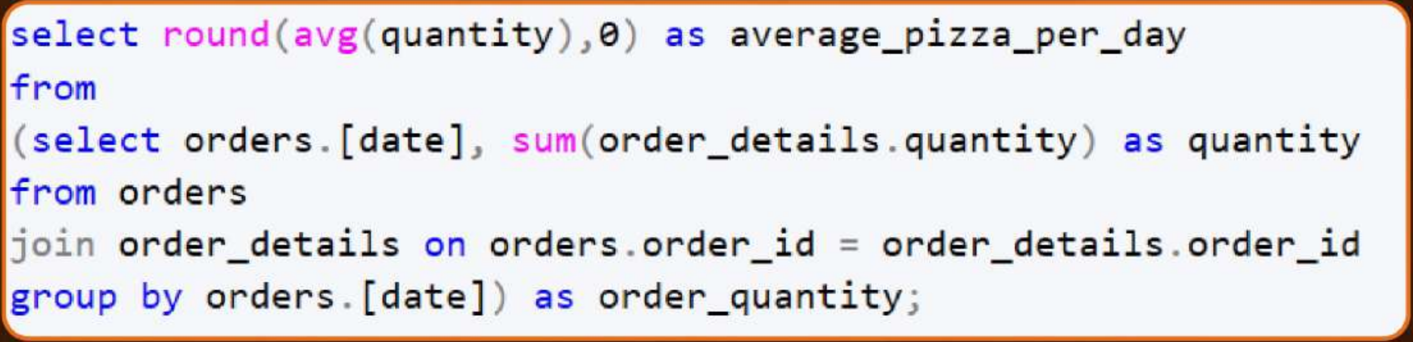
Join the Relevant Tables to Find the Category-Wise Distribution of Pizzas.

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

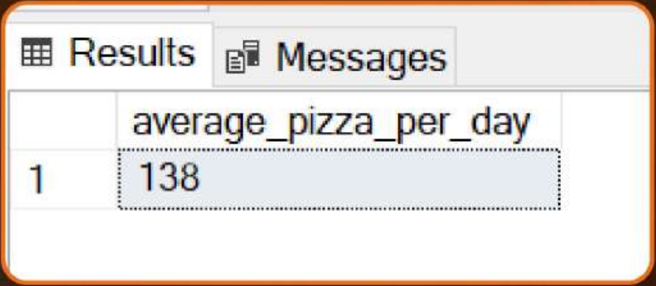


```
   category   (No column name)
1  Chicken    6
2  Classic    8
3  Supreme    9
4  Veggie     9
```

Group the Orders by Date and Calculate the Average Number of Pizzas Ordered per Day.

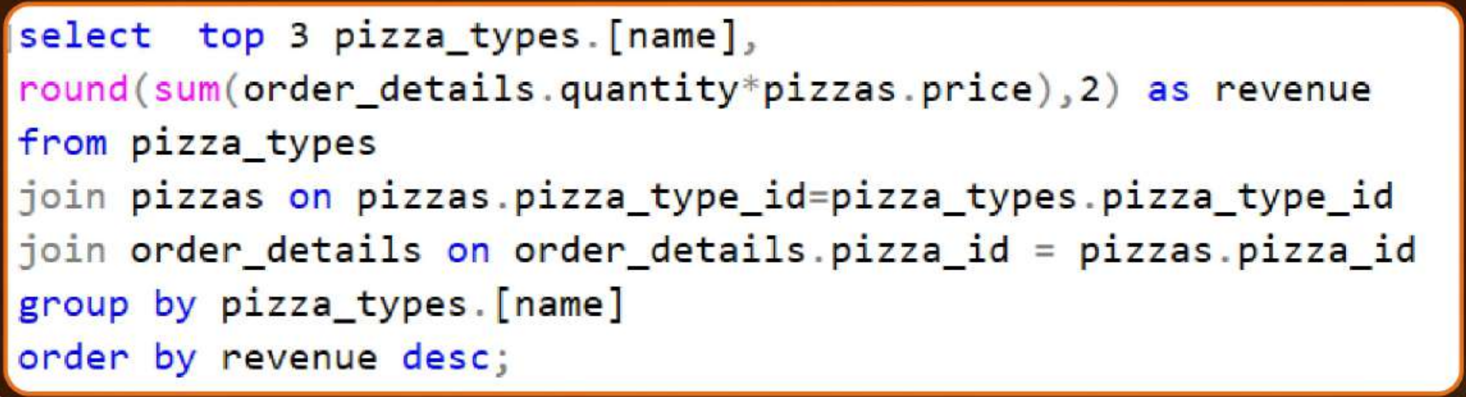


```sql
select round(avg(quantity),0) as average_pizza_per_day
from
(select orders.[date], sum(order_details.quantity) as quantity
from orders
join order_details on orders.order_id = order_details.order_id
group by orders.[date]) as order_quantity;
```

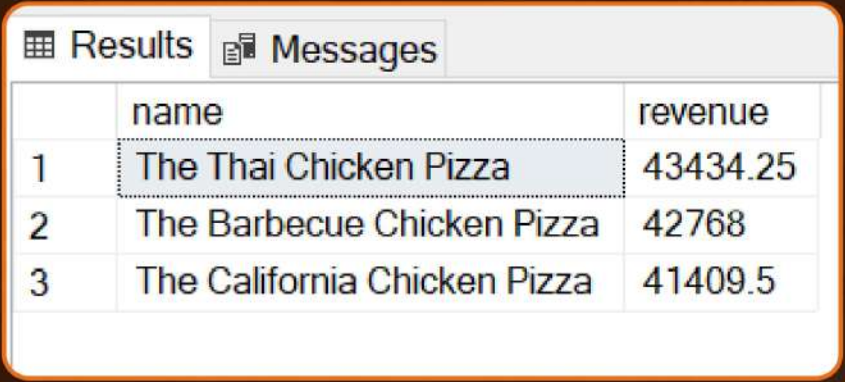


```
   average_pizza_per_day
1  138
```

Determine the Top 3 Most Ordered Pizza Types Based on Revenue.

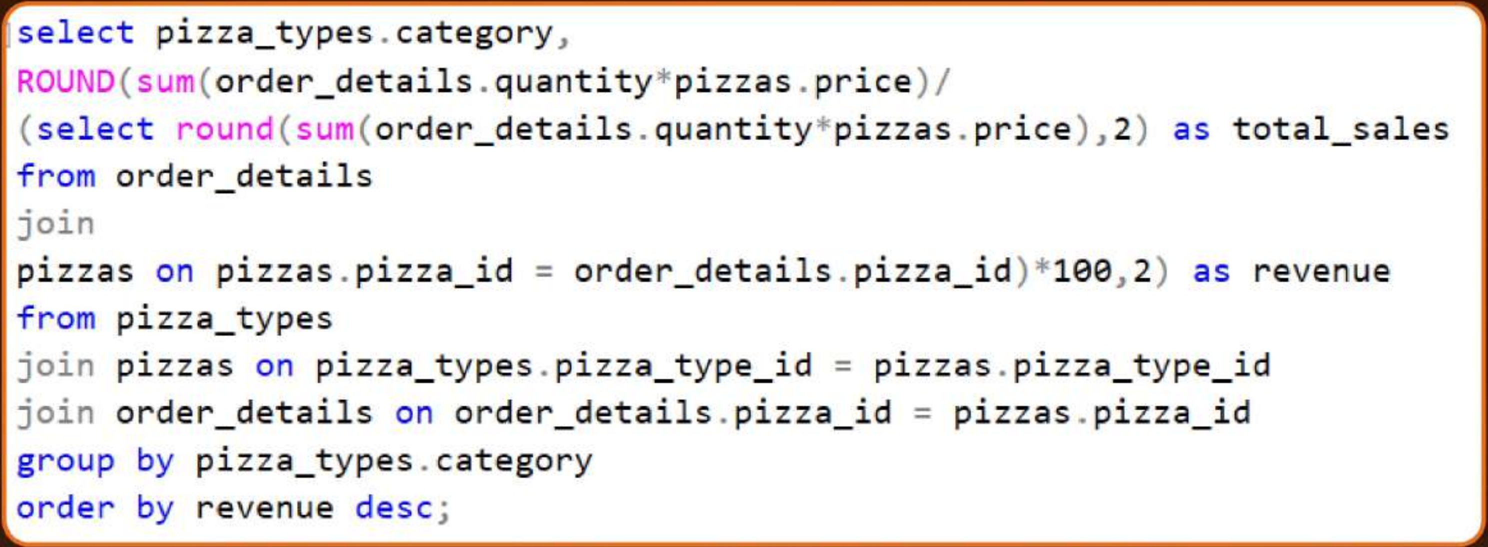


```
select  top 3 pizza_types.[name],
round(sum(order_details.quantity*pizzas.price),2) as revenue
from pizza_types
join pizzas on pizzas.pizza_type_id=pizza_types.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.[name]
order by revenue desc;
```

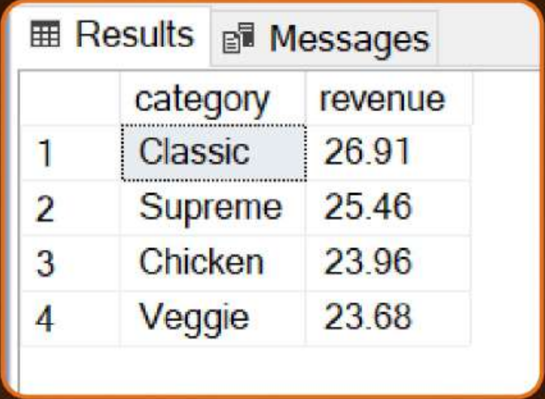


	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

Calculate the Percentage Contribution of Each Pizza Type to Total Revenue.

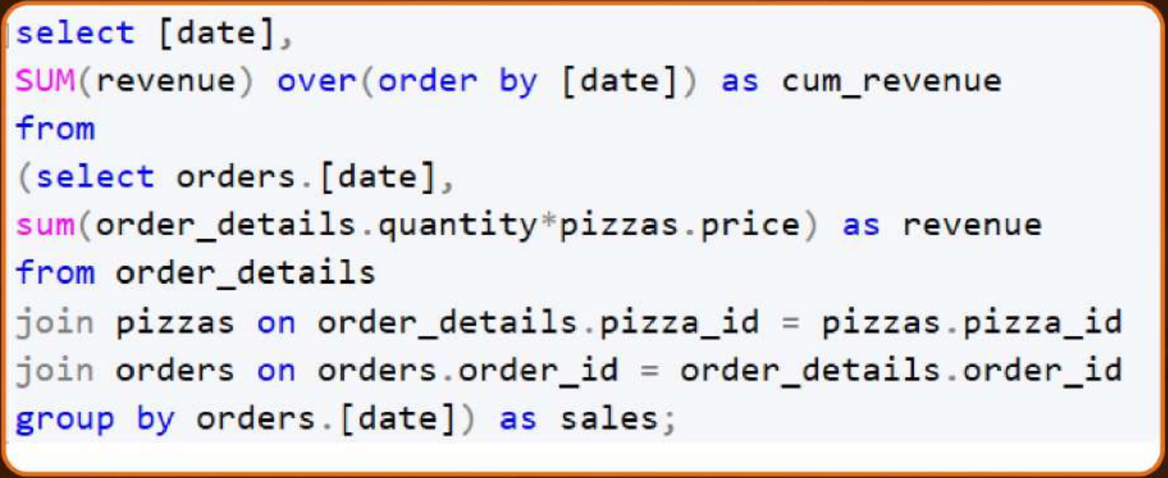


```
select pizza_types.category,
ROUND(sum(order_details.quantity*pizzas.price)/
(select round(sum(order_details.quantity*pizzas.price),2) as total_sales
from order_details
join
pizzas on pizzas.pizza_id = order_details.pizza_id)*100,2) as revenue
from pizza_types
join pizzas on pizza_types.pizza_type_id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

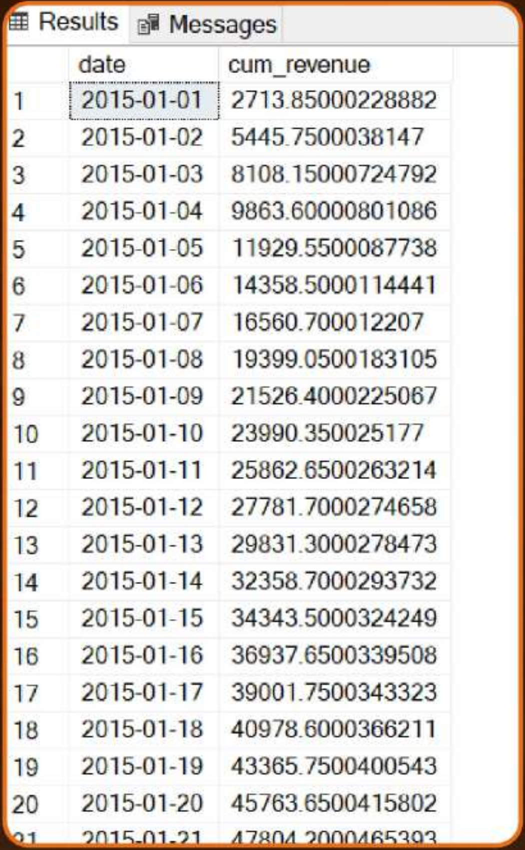


	category	revenue
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

Analyze the Cumulative Revenue Generated Over Time.

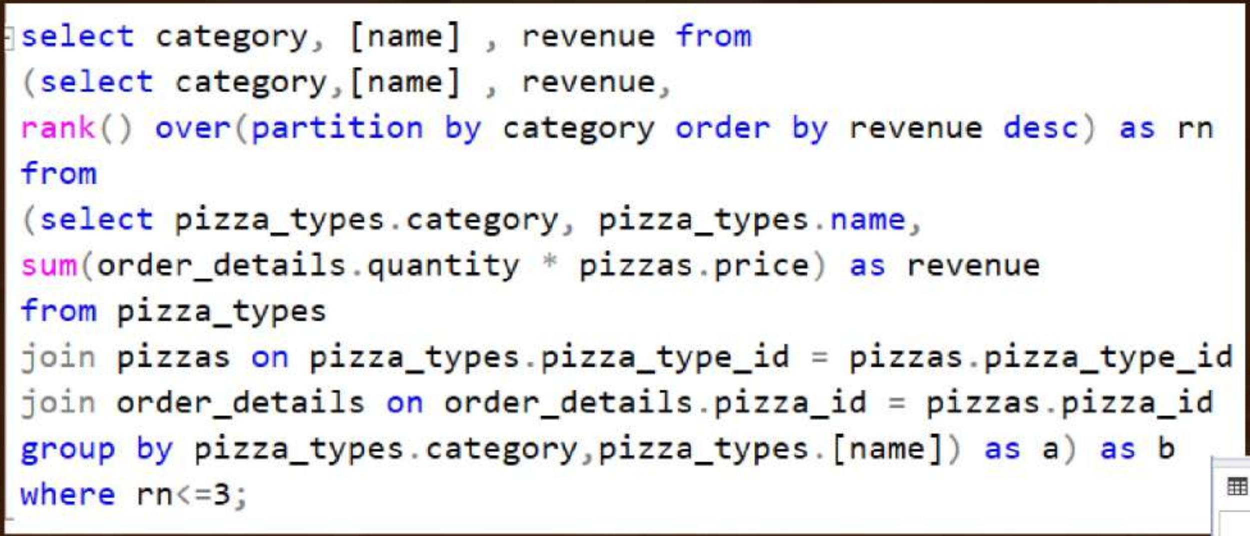


```
select [date],
SUM(revenue) over(order by [date]) as cum_revenue
from
(select orders.[date],
sum(order_details.quantity*pizzas.price) as revenue
from order_details
join pizzas on order_details.pizza_id = pizzas.pizza_id
join orders on orders.order_id = order_details.order_id
group by orders.[date])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

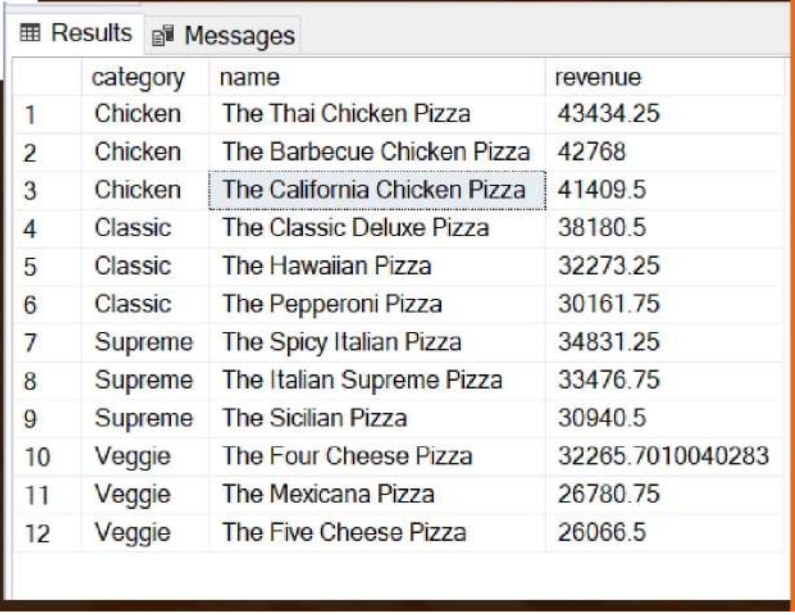


	date	cum_revenue
1	2015-01-01	2713.85000228882
2	2015-01-02	5445.7500038147
3	2015-01-03	8108.15000724792
4	2015-01-04	9863.60000801086
5	2015-01-05	11929.5500087738
6	2015-01-06	14358.5000114441
7	2015-01-07	16560.700012207
8	2015-01-08	19399.0500183105
9	2015-01-09	21526.4000225067
10	2015-01-10	23990.350025177
11	2015-01-11	25862.6500263214
12	2015-01-12	27781.7000274658
13	2015-01-13	29831.3000278473
14	2015-01-14	32358.7000293732
15	2015-01-15	34343.5000324249
16	2015-01-16	36937.6500339508
17	2015-01-17	39001.7500343323
18	2015-01-18	40978.6000366211
19	2015-01-19	43365.7500400543
20	2015-01-20	45763.6500415802
21	2015-01-21	47804.2000465393

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



```
select category, [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;
```



	category	name	revenue
1	Chicken	The Thai Chicken Pizza	43434.25
2	Chicken	The Barbecue Chicken Pizza	42768
3	Chicken	The California Chicken Pizza	41409.5
4	Classic	The Classic Deluxe Pizza	38180.5
5	Classic	The Hawaiian Pizza	32273.25
6	Classic	The Pepperoni Pizza	30161.75
7	Supreme	The Spicy Italian Pizza	34831.25
8	Supreme	The Italian Supreme Pizza	33476.75
9	Supreme	The Sicilian Pizza	30940.5
10	Veggie	The Four Cheese Pizza	32265.7010040283
11	Veggie	The Mexicana Pizza	26780.75
12	Veggie	The Five Cheese Pizza	26066.5

KEY INSIGHTS

- **Sales Performance:** Peak order periods indicate the need for optimized staffing and inventory management to handle demand efficiently.
- **Product Trends:** The most common pizza size and highest-priced pizza should be strategically promoted to maximize revenue.
- **Top-Selling Pizzas:** The top 5 pizzas should be highlighted in marketing campaigns and bundled deals to boost sales further.
- **Order Trends:** Peak hours require improved resource allocation, such as faster prep times and better order fulfillment strategies.
- **Category Sales:** High-performing pizza categories should be prioritized in menu updates and pricing adjustments.
- **Revenue Contribution:** High-revenue pizzas should be positioned as premium offerings with upselling opportunities.
- **Seasonal Demand**: Sales trends suggest launching seasonal discounts and special menus during peak months.
- **High-Revenue Items:** The top 3 revenue-generating pizzas per category should be strategically promoted to drive profitability.

THANK YOU

View project on github