# Analysis of Pizza Sales Using SQL

Comprehensive Analysis from Data Extraction to Insights

# INTRODUCTION

**This project aims to provide valuable insights from our pizza sales data, enhancing business decision-making and improving sales performance.**

**I’veanalyzed data from:**

- **Orders: Details of each order, including date and time.**
- **Order Details: Specifics about the pizzas ordered, including type, size, and quantity.**
- **Pizzas: Information about different pizza offerings, including prices.**
- **Pizza Types: Categories and characteristics of pizzas.**

**Also, using Power BI, I created interactive dashboards to visualize these insights. These dashboards include various charts and slicers, enabling easy filtering and exploration of the sales data.**

**https://github.com/HarshadKamble77/My-Projects/tree/main/Dashboard/Pizza%20Sales%20Dashboard**

# KEY QUESTIONS

**Analysis addressed the following questions:**

**Basic:**

- **Total number of orders.**
- **Total revenue generated.**
- **Highest-priced pizza.**
- **Most common pizza size.**
- **Top 5 most ordered pizza types.**

**Intermediate:**

- **Total quantity of each pizza category ordered.**
- **Order distribution by hour.**
- **Category-wise distribution of pizzas.**
- **Average number of pizzas ordered per day.**
- **Top 3 pizza types based on revenue.**

**Advanced:**

- **Percentage contribution of each pizza type to total revenue.**
- **Cumulative revenue over time.**
- **Top 3 pizza types based on revenue within each category.**

## DATA DESCRIPTION

**In data there are four tables  as follows:-**

**1. Orders:**

- **order_id: Unique identifier for each order.**
- **date: Date when the order was placed.**
- **time: Time when the order was placed.**

**2. Order Details:**

- **order_details_id: Unique identifier for each order detail.**
- **order_id: Reference to the order.**
- **pizza_id: Identifier for the specific pizza.**
- **quantity: Number of pizzas ordered.**

**3. Pizzas:**

- **pizza_id: Unique identifier for each pizza.**
- **pizza_type_id: Identifier for the type of pizza.**
- **size: Size of the pizza (S, M, L, XL, XXL).**
- **price: Price of the pizza.**

**4. Pizza Types:**

- **pizza_type_id: Unique identifier for each pizza type.**
- **name: Name of the pizza type.**
- **category: Category of the pizza (e.g., Vegetarian, Non-Vegetarian).**
- **ingredients: Ingredients used in the pizza**

## DATA PREPARATION

**1. Data Collection:**

Collected data from four tables: Orders, Order Details, Pizzas, and Pizza Types**.**

**2. Data Cleaning:**

Checked for missing values and inconsistencies,

Standardized date and time formats,

Ensured data types were correctly assigned.

**3. Data Integration: Merged Tables:**

- Orders and Order Details: Combined on order_id to get detailed order information.
- Order Details and Pizzas: Joined on pizza_id to get pizza size and price.
- Pizzas and Pizza Types: Joined on pizza_type_id to get category and ingredients.

**4. Data Transformation: Created new columns for analysis, such as:**

- **Revenue**: Calculated as quantity * price.
- **Order Date and Time**: Combined date and time for detailed time analysis.

**Grouped data for various aggregation purposes, such as total orders, total revenue, and quantity of each pizza type.**

**5. Data Validation:**

Ensured the integrity and accuracy of the merged data.

Verified key metrics with raw data to check for discrepancies.

**6. Data Import into Power BI:**

Loaded the cleaned and integrated data into Power BI.

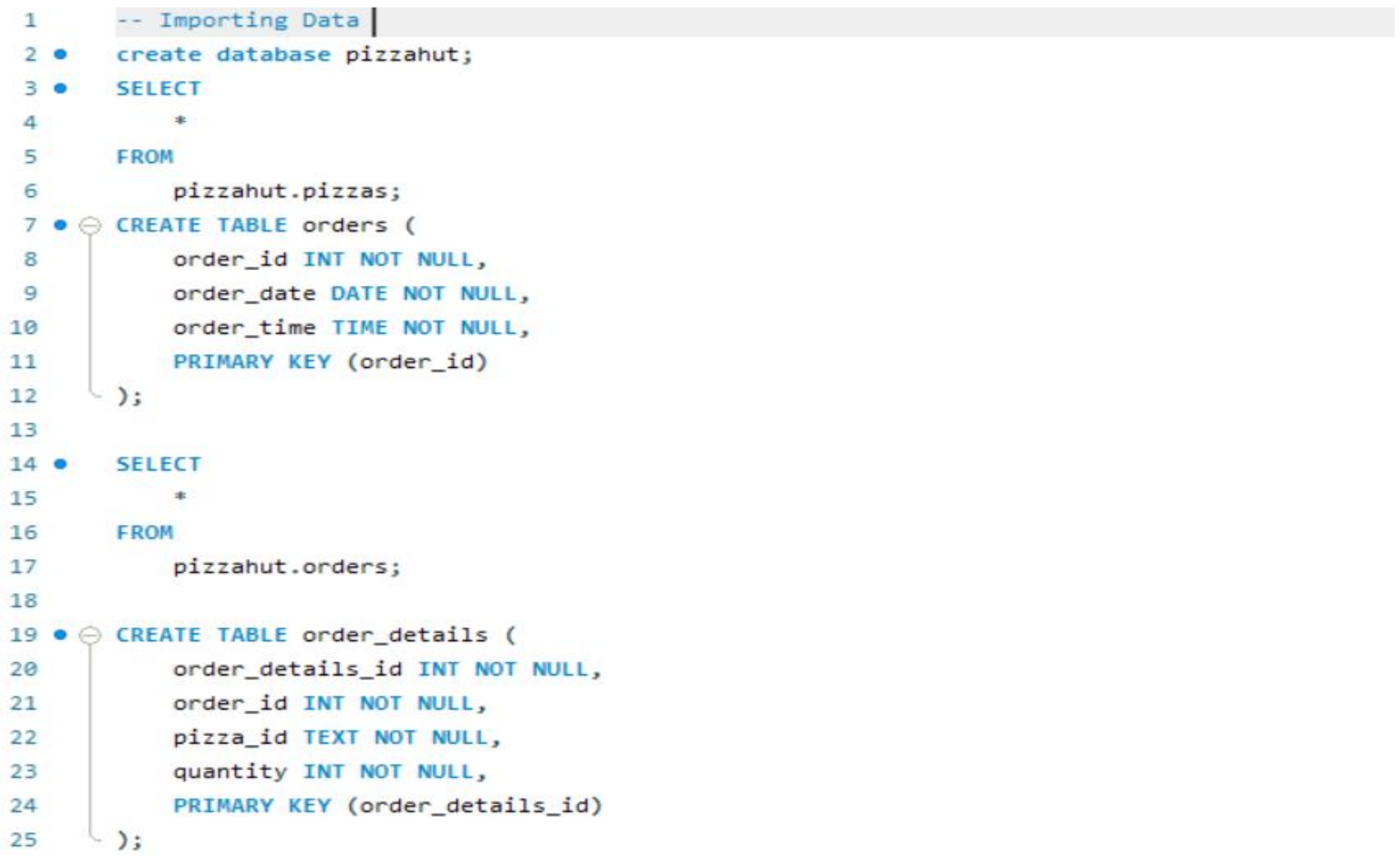
Created relationships between tables within Power BI for seamless data analysis.

**7. Creation of Calculated Columns and Measures:**

Used DAX functions to create necessary measures.

**This thorough data preparation ensured that the dataset was ready for effective analysis and visualization in Power BI, leading to the insights presented in this project.**

## BASIC ANALYSIS



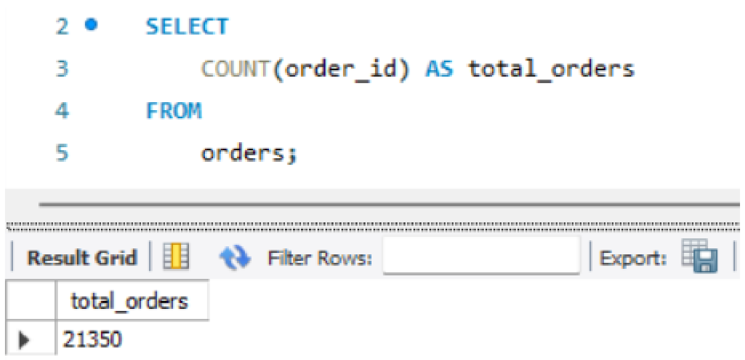
```
-- Importing Data
create database pizzahut;
SELECT
    *
FROM
    pizzahut.pizzas;
CREATE TABLE orders (
    order_id INT NOT NULL,
    order_date DATE NOT NULL,
    order_time TIME NOT NULL,
    PRIMARY KEY (order_id)
);

SELECT
    *
FROM
    pizzahut.orders;

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)
);
```

Pizzas & Pizz_types data imported by adding table to created tables.

## #Retrieve the total number of orders placed.

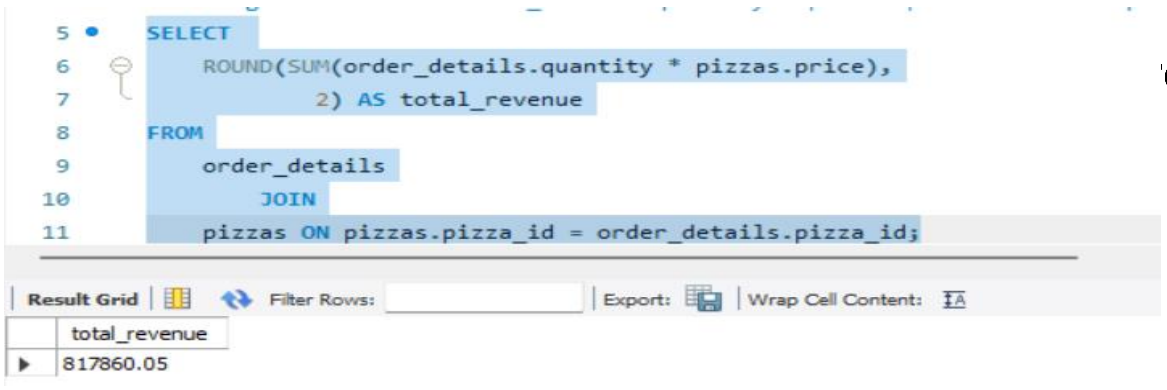


```
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

Result Grid | Filter Rows: | Export:

```
total_orders
21350
```

**om pizza sales.**

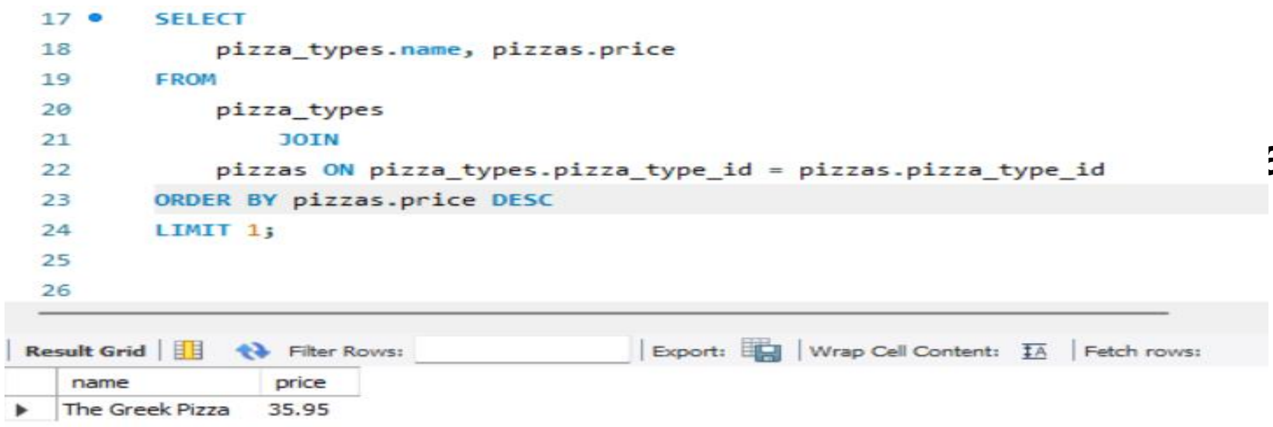


```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

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

```
total_revenue
817860.05
```

**a.**

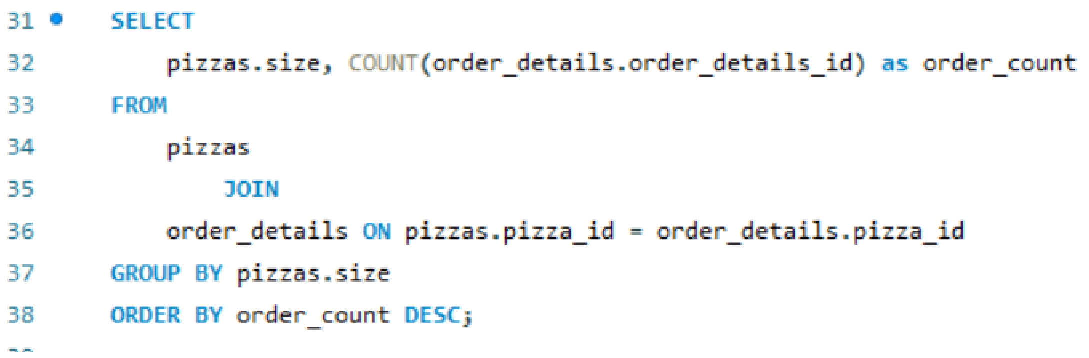


```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

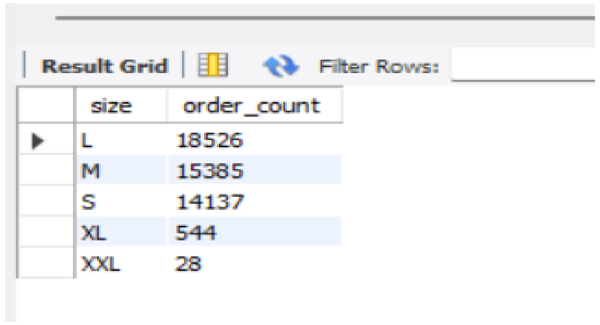
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

```
name              price
The Greek Pizza   35.95
```

# Identify the most common pizza size ordered.

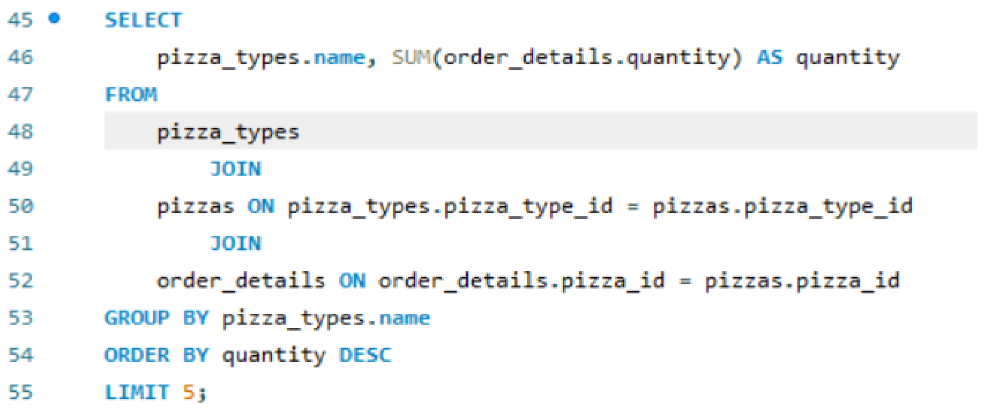


```sql
SELECT
    pizzas.size, COUNT(order_details.order_details_id) as order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

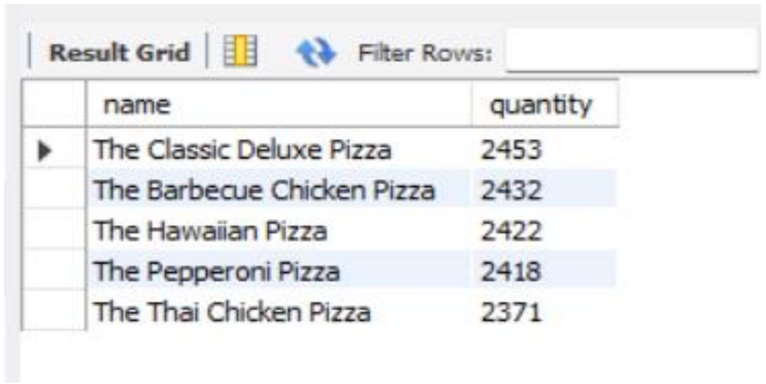


| size | order_count |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

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



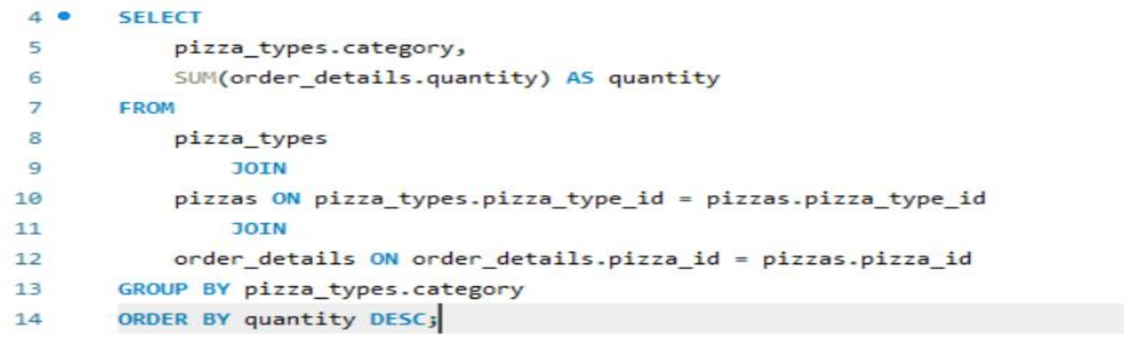
```sql
SELECT
    pizza_types.name, SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```



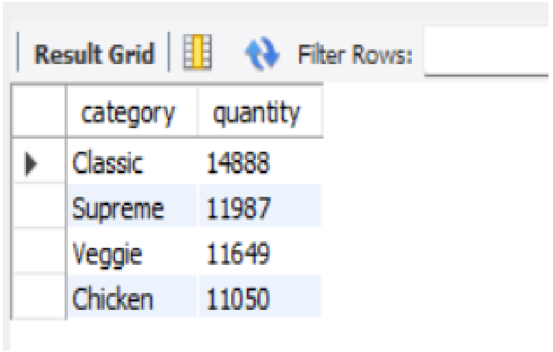
| name | quantity |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

## INTERMEDIATE ANALYSIS

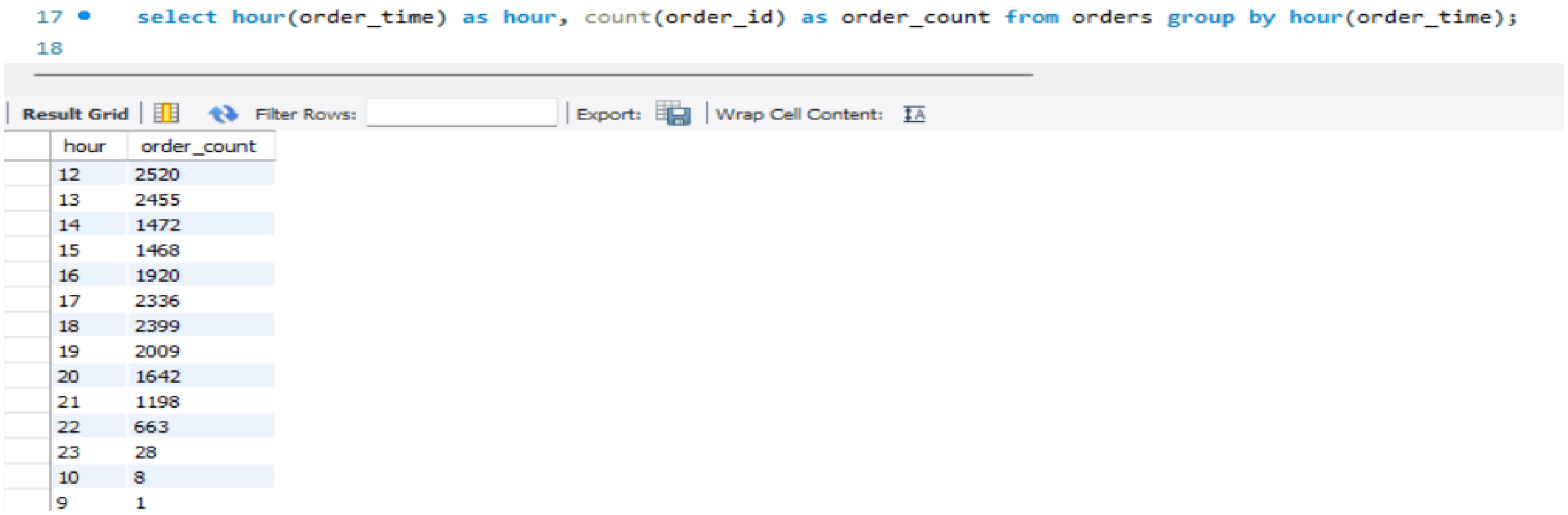
**#Join the necessary tables to find the total quantity of each pizza category ordered.**



```
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;
```



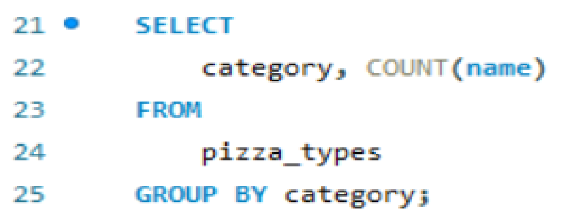
| category | quantity |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |



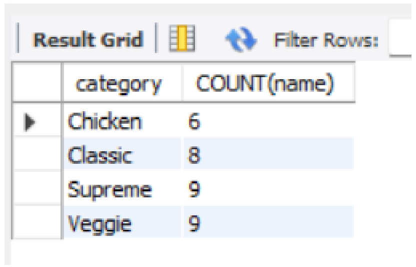
```
select hour(order_time) as hour, count(order_id) as order_count from orders group by hour(order_time);
```

| hour | order_count |
|---|---|
| 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 |

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



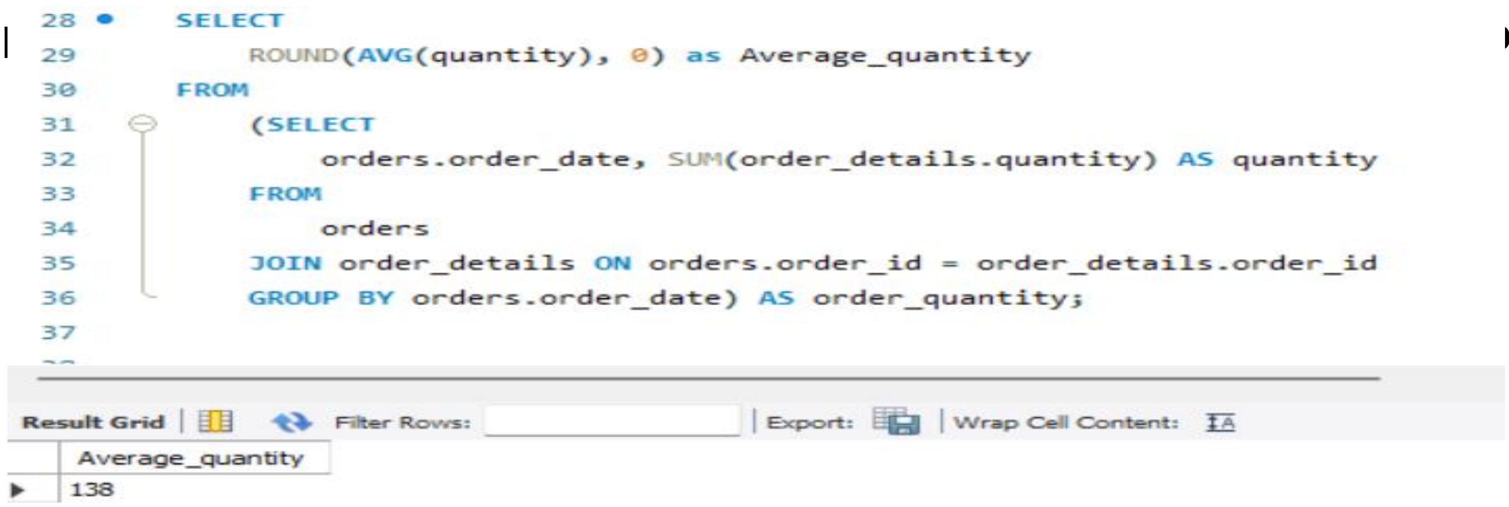
```sql
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```



Result Grid | Filter Rows:

| category | COUNT(name) |
|---|---|
| Chicken | 6 |
| Classic | 8 |
| Supreme | 9 |
| Veggie | 9 |

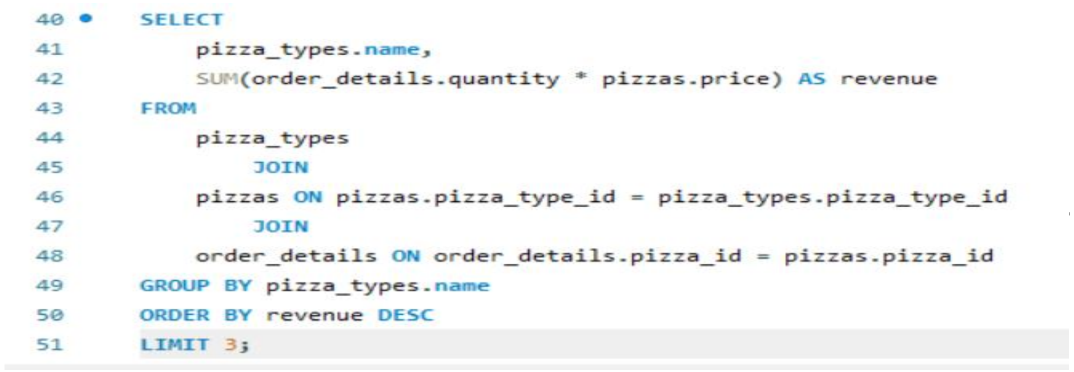
## #Grou er day.



```sql
SELECT
    ROUND(AVG(quantity), 0) as Average_quantity
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;
```

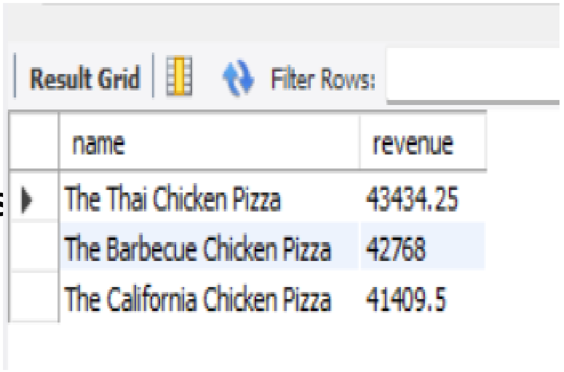
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Average_quantity |
|---|
| 138 |



```sql
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;
```

types

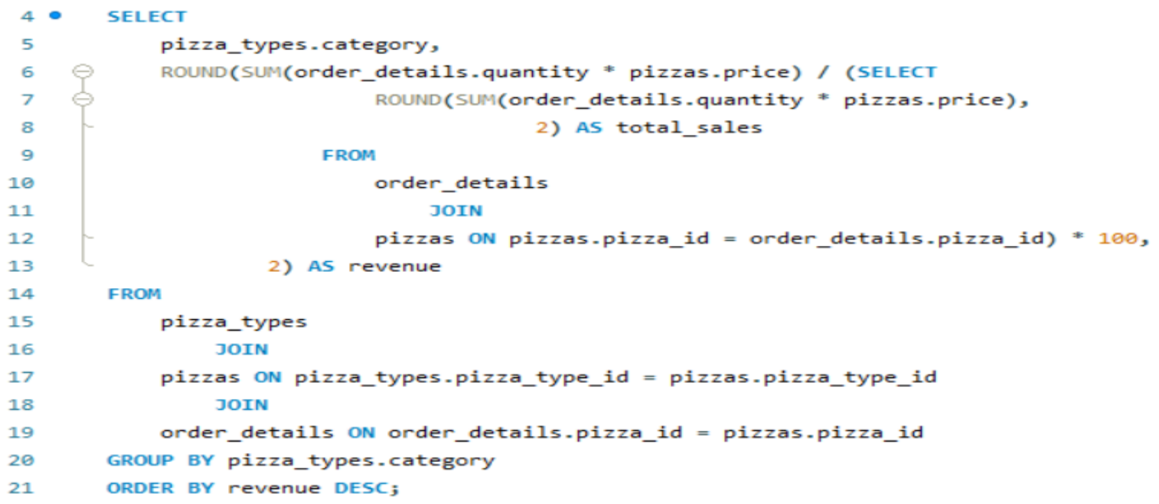


Result Grid | Filter Rows:

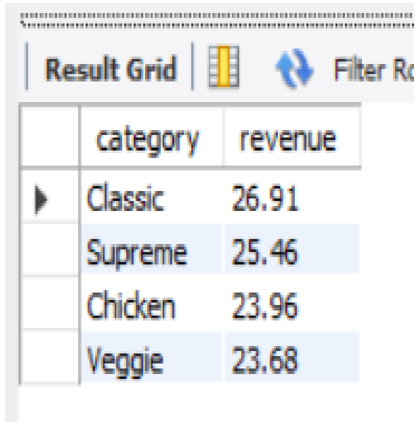
| name | revenue |
|---|---|
| The Thai Chicken Pizza | 43434.25 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41409.5 |

# ADVANCED ANALYSIS

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

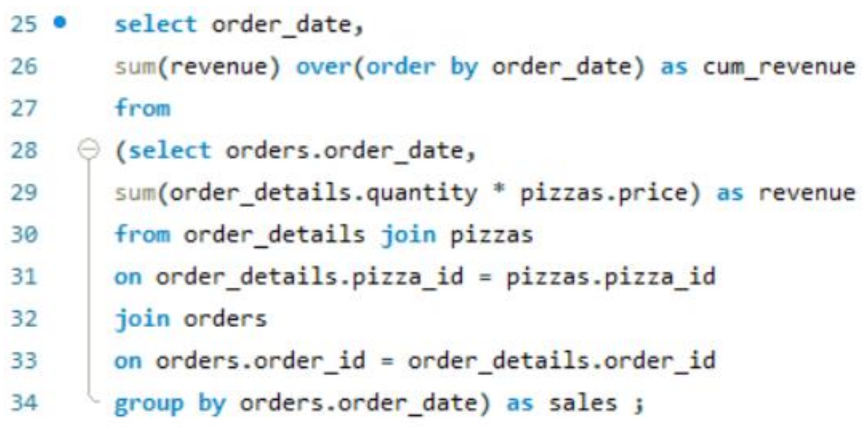


```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                ROUND(SUM(order_details.quantity * pizzas.price),
                            2) AS total_sales
            FROM
                order_details
                    JOIN
                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;
```

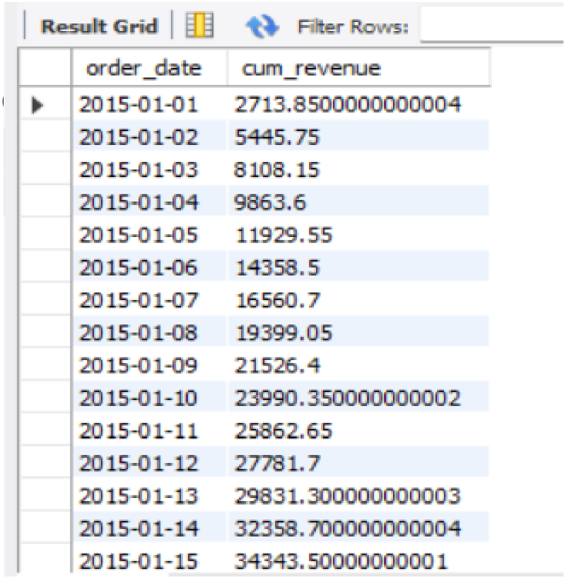


| category | revenue |
|---|---|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

enue g



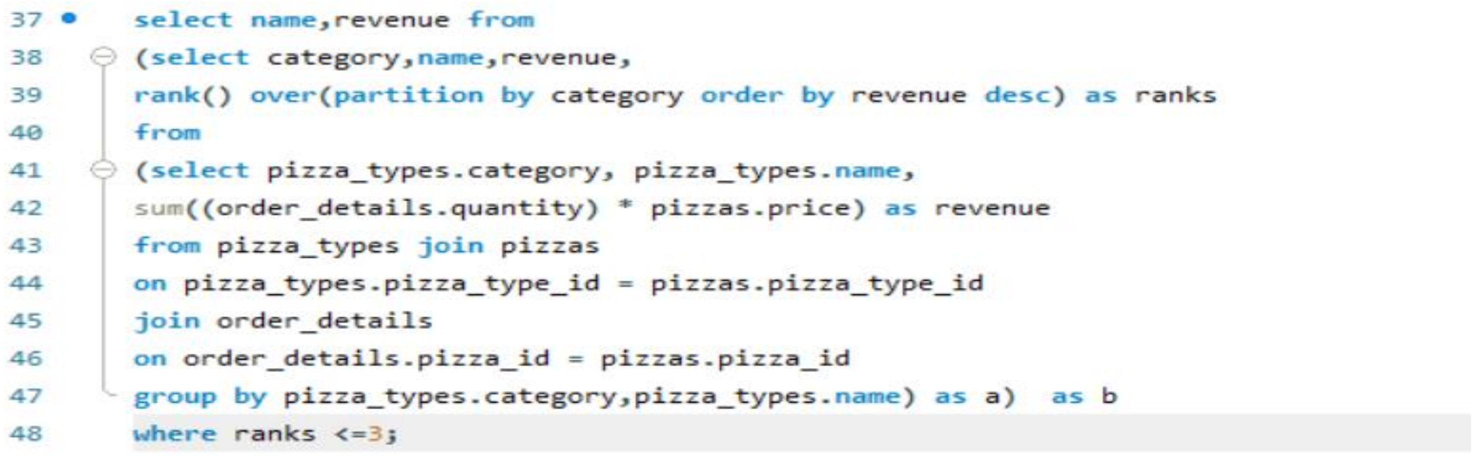
```sql
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from
(select orders.order_date,
sum(order_details.quantity * pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = order_details.order_id
group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



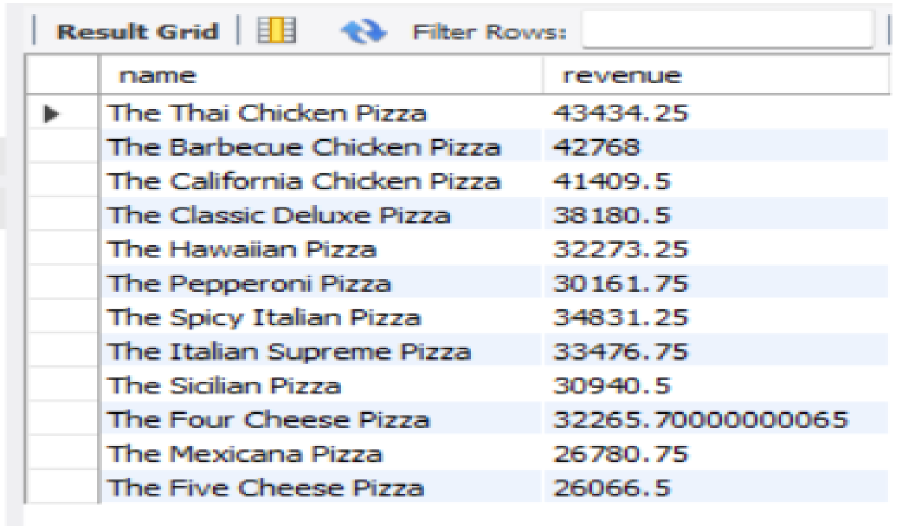
| order_date | cum_revenue |
|---|---|
| 2015-01-01 | 2713.8500000000004 |
| 2015-01-02 | 5445.75 |
| 2015-01-03 | 8108.15 |
| 2015-01-04 | 9863.6 |
| 2015-01-05 | 11929.55 |
| 2015-01-06 | 14358.5 |
| 2015-01-07 | 16560.7 |
| 2015-01-08 | 19399.05 |
| 2015-01-09 | 21526.4 |
| 2015-01-10 | 23990.350000000002 |
| 2015-01-11 | 25862.65 |
| 2015-01-12 | 27781.7 |
| 2015-01-13 | 29831.300000000003 |
| 2015-01-14 | 32358.700000000004 |
| 2015-01-15 | 34343.50000000001 |

## ADVANCE ANALYSIS

**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 ranks
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 ranks <=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 |
| The Mexicana Pizza | 26780.75 |
| The Five Cheese Pizza | 26066.5 |

## CONCLUSION

SQL-based analysis of the pizza sales data provided significant insights that can drive strategic decisions for improving sales and operational efficiency.

The SQL queries used in this project efficiently extracted and transformed data to provide these insights. These results form a strong foundation for making data-driven decisions that enhance business performance, customer satisfaction, and profitability.