



HELLO!

Hello my name is ANUP KUMAR RAJ.
In this project i have solved SQL queries,
which were related to the Pizza sales.

QUESTIONS

- 1) 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) Analyze the cumulative revenue generated over time.







Pizza is a dish that originates from Italy and is one of the favorite foods of many people in various parts of the world.

Let's start our adventure in the world of pizza!

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

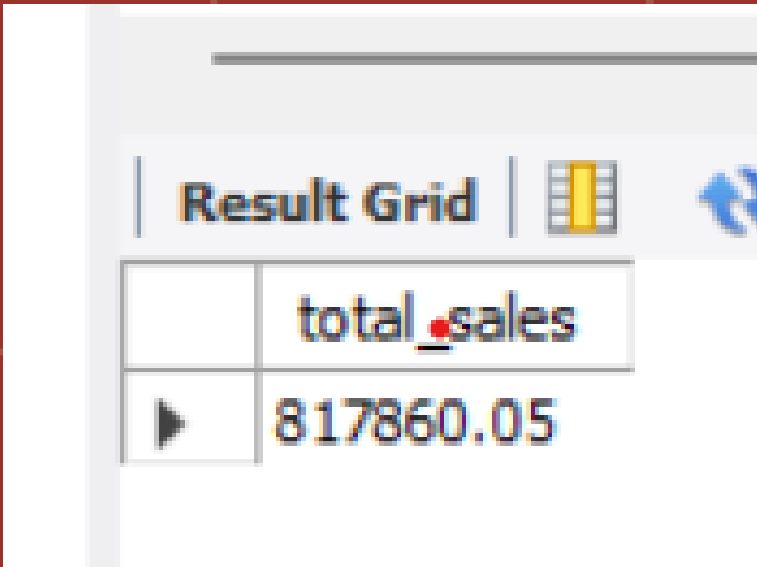
```
SELECT  
    COUNT(order_id) AS total_orders  
FROM  
    orders;
```



| Result Grid | |  |  | File |
|---------------------------------------------------------------------------------------|--------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|------|
| | total_orders | | | |
|  | 21350 | | | |

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES


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



A screenshot of a database query result grid. The grid has a header row with the column name 'total_sales' and a data row with the value '817860.05'. The grid is titled 'Result Grid' and has a blue arrow icon on the right.

| Result Grid | |
|-------------|-------------|
| | total_sales |
| ▶ | 817860.05 |

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;
```

| Result Grid | | | Filter Rows: |
|-------------|-----------------|-------|--------------|
| | name | price | |
| ▶ | The Greek Pizza | 35.95 | |

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

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

| Result Grid | | | Filter |
|-------------|------|-------------|--------|
| | size | order_count | |
| ▶ | L | 18526 | |
| | M | 15385 | |
| | S | 14137 | |
| | XL | 544 | |
| | XXL | 28 | |

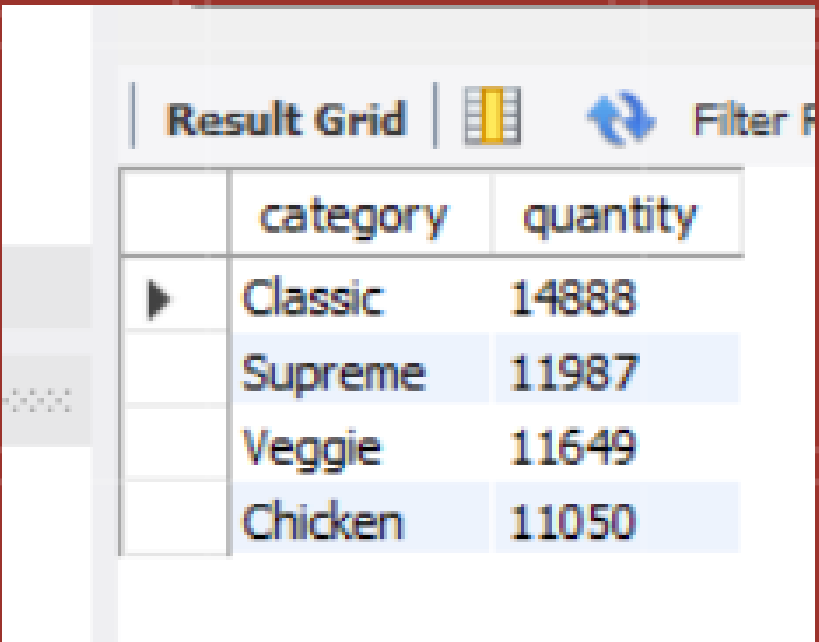
LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES

```
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 orders_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 |

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED

```
SELECT
    pizza_types.category,
    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 orders_details.pizza_id = pizzas.pizza_id
GROUP BY
    pizza_types.category
ORDER BY
    quantity DESC;
```

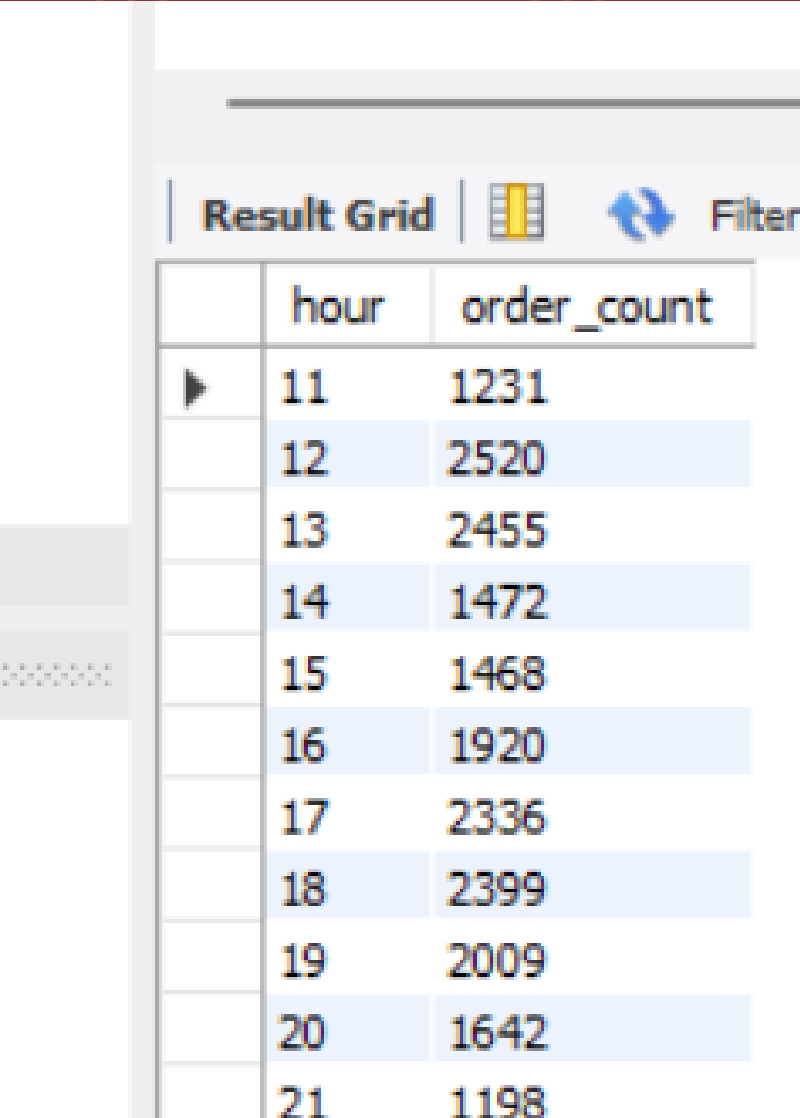


A screenshot of a database query result grid. The grid has a header row with 'category' and 'quantity' columns. Below the header, there are four rows of data: 'Classic' with quantity 14888, 'Supreme' with quantity 11987, 'Veggie' with quantity 11649, and 'Chicken' with quantity 11050. The rows are ordered by quantity in descending order. The interface includes a 'Result Grid' tab, a 'Filter' button, and a 'Filter F' label.

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

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

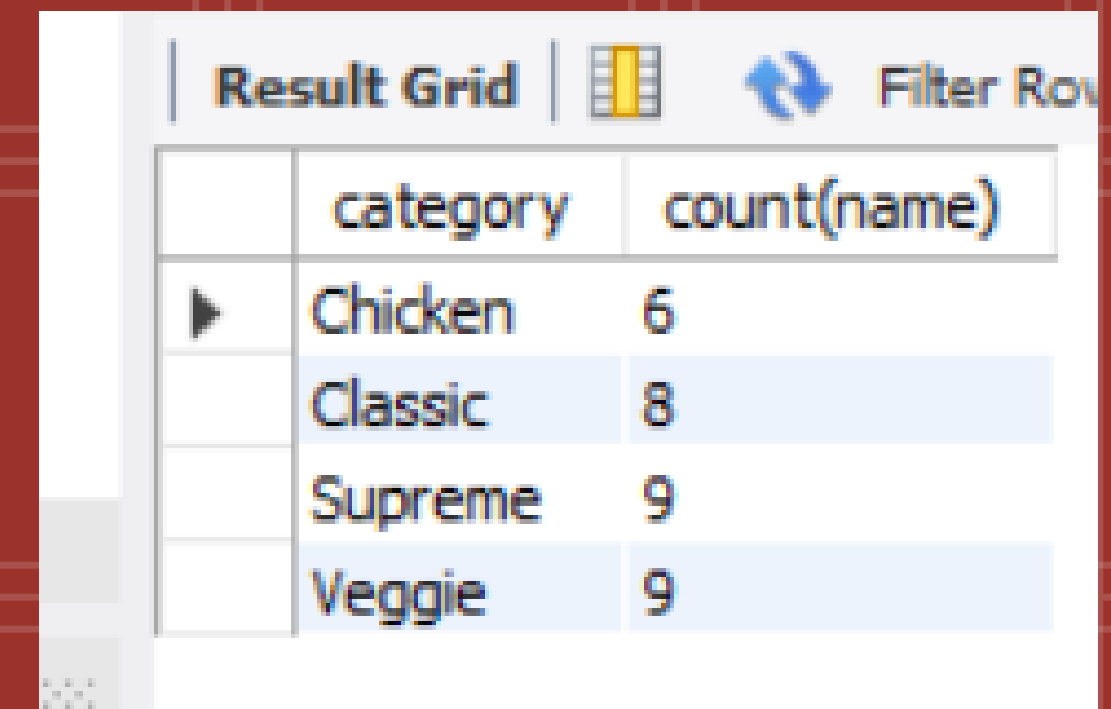


A screenshot of a database query result grid. The grid has two columns: 'hour' and 'order_count'. The data is as follows:

| hour | order_count |
|------|-------------|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |
| 16 | 1920 |
| 17 | 2336 |
| 18 | 2399 |
| 19 | 2009 |
| 20 | 1642 |
| 21 | 1198 |

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS

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



The screenshot shows a 'Result Grid' window with a table containing the results of the SQL query. The table has two columns: 'category' and 'count(name)'. The data is as follows:

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

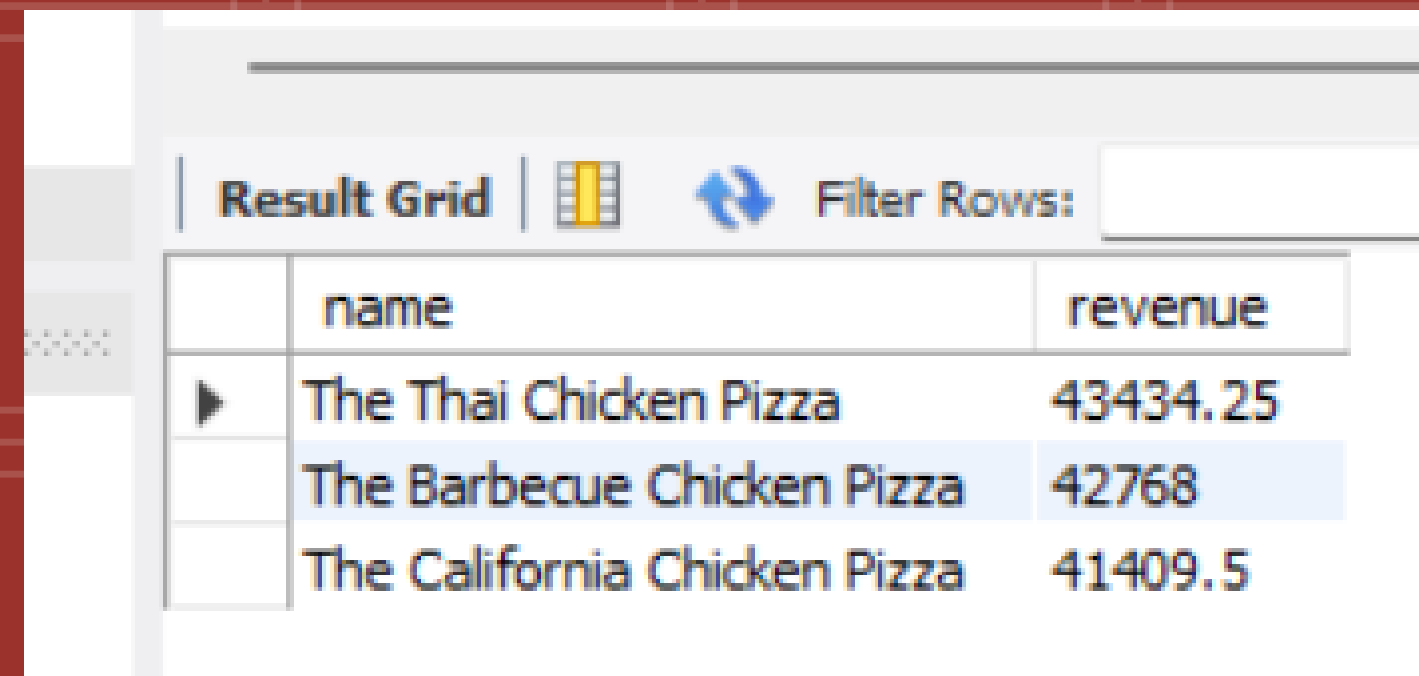
GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY

```
SELECT
    ROUND(AVG(total_quantity), 0) as avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date,
        SUM(orders_details.quantity) AS total_quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

| Result Grid | | Filter Rows: |
|-------------|---------------------------|--------------|
| | avg_pizza_ordered_per_day | |
| ▶ | 138 | |

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 pizzas.pizza_type_id = pizza_types.pizza_type_id
    JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

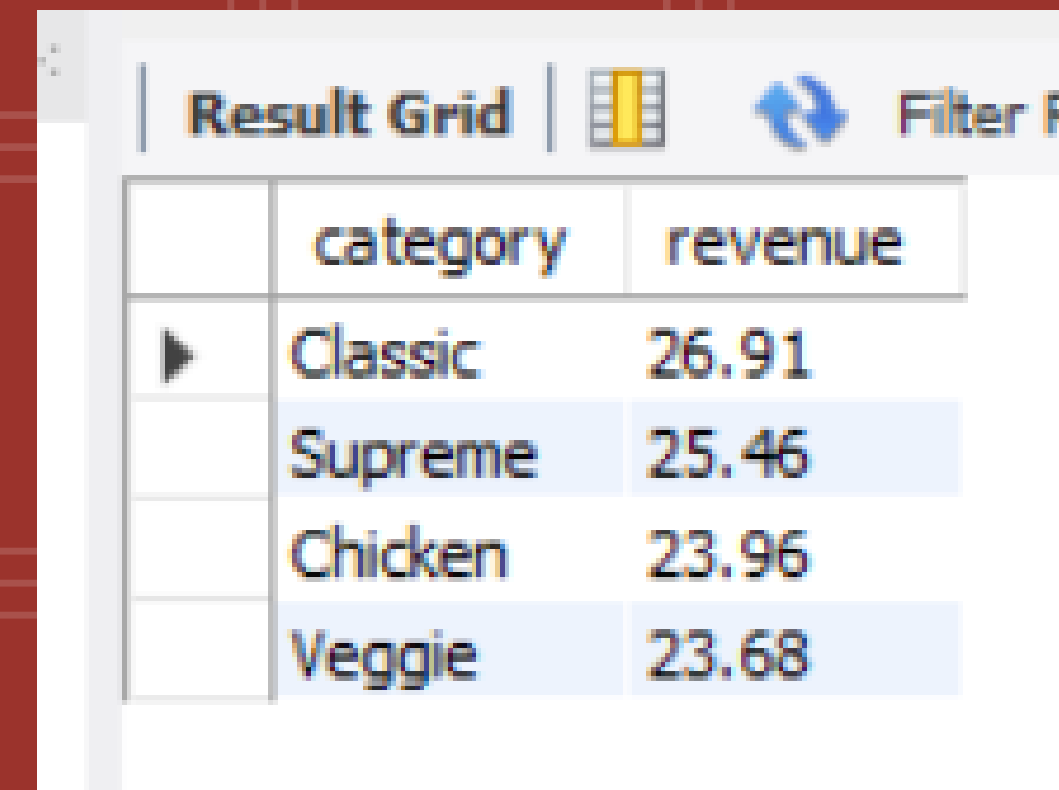


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of the SQL query, showing the top 3 most ordered pizza types based on revenue. The columns are 'name' and 'revenue'. The rows are sorted in descending order of revenue.

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

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

```
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / NULLIF((
        SELECT ROUND(SUM(orders_details.quantity * pizzas.price), 2) AS total_sales
        FROM orders_details
        JOIN pizzas ON pizzas.pizza_id = orders_details.pizza_id
    ), 0) * 100, 2) 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
ORDER BY
    revenue DESC;
```



The screenshot shows a 'Result Grid' window with a table containing the results of the SQL query. The table has two columns: 'category' and 'revenue'. The data is sorted in descending order of revenue. The categories are Classic, Supreme, Chicken, and Veggie. The revenue values are 26.91, 25.46, 23.96, and 23.68 respectively. The window also includes a 'Filter' button and a 'Result Grid' label.

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

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 daily_revenue;
```

| Result Grid | | | Filter Rows: |
|-------------|------------|----------------------|--------------|
| | order_date | cum_revenue | |
| ▶ | 2015-01-01 | 2713.850000000000004 | |
| | 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 | |



THANK YOU