BY MIDHUN N S

TURNING SLICES INTO INSIGHTS – ONE QUERY AT A TIME

# WELCOME TO PIZZA QUERY

SOLVING 13 REAL WORLD BUSINESS QUESTIONS THROUGH DATA

# PIZZA SALES ANALYSIS USING SQL

Hello, my name is Midhun N S. In this project, I have leveraged Structured Query Language (SQL) to analyze and
extract valuable business insights from a Pizza Sales dataset. The objective was to solve real-world sales-related
queries that reflect the performance, trends, and customer preferences of a pizza business.
Through this project, I wrote and executed 13 SQL queries covering a range of analytical tasks including:

*Revenue and order trends
*Best-selling products
*Customer ordering behavior
*Time-based performance analysis

This hands-on project strengthened my skills in data exploration, aggregation, filtering, joins, and performance analysis, making it a strong addition to my data portfolio

# WHAT'S COOKING IN PIZZA QUERY?

## BASIC LEVEL ANALYSIS

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.

## INTERMEDIATE LEVEL ANALYSIS

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.84,Group the orders by date and

9,calculate the average number of pizzas ordered per day.

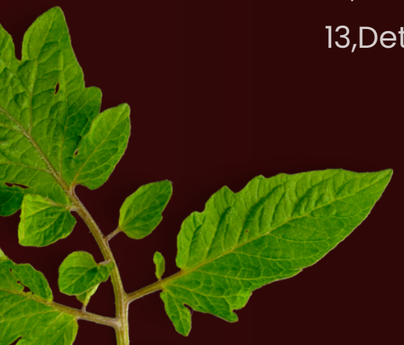
10,Determine the top 3 most ordered pizza types based on revenue.

## ADVANCED LEVEL ANALYSIS

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

12,Analyze the cumulative revenue generated over time.

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



# RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

Result Grid

| total_orders |
| --- |
| 21350 |

# CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
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
```

Result Grid

| | total_sales |
|---|---|
| ▶ | 817860.05 |

# IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
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 |

# 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;
```

Result Grid

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

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



```sql
SELECT
    pt.category, SUM(o.quantity) AS Quantity
FROM
    pizza_types AS pt
        JOIN
    pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
        JOIN
    order_details AS o ON o.pizza_id = p.pizza_id
GROUP BY pt.category
ORDER BY Quantity DESC;
```

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

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

```sql
SELECT
    pt.name, SUM(o.quantity) AS Quantity
FROM
    pizza_types AS pt
        JOIN
    pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
        JOIN
    order_details AS o ON o.pizza_id = p.pizza_id
GROUP BY pt.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 |

# DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.



```
2 •   SELECT
3         HOUR(order_time) AS Hour, COUNT(order_id) AS Order_count
4     FROM
5         orders
6     GROUP BY HOUR(order_time);
```

| Hour | Order_count |
|---|---|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |
| 16 | 1920 |

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

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

| 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.

```sql
SELECT
    ROUND(AVG(quantity)) AS avg_pizza_ordered_per_day
FROM
    (SELECT
        O1.order_date, SUM(o2.quantity) AS quantity
    FROM
        orders AS o1
    JOIN order_details AS o2 ON o1.order_id = o2.order_id
    GROUP BY o1.order_date) AS order_quantity;
```

| avg_pizza_ordered_per_day |
| --- |
| 138 |

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    pt.name, ROUND(SUM(o.quantity * p.price)) AS Revenue
FROM
    pizza_types AS pt
        JOIN
    pizzas AS P ON pt.pizza_type_id = p.pizza_type_id
        JOIN
    order_details AS o ON o.pizza_id = p.pizza_id
GROUP BY pt.name
ORDER BY Revenue DESC
LIMIT 3;
```

| name | Revenue |
|---|---|
| The Thai Chicken Pizza | 43434 |
| The Barbecue Chicken Pizza | 42768 |
| The California Chicken Pizza | 41410 |

# CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.



```sql
SELECT pt.category,
ROUND((ROUND(SUM(o.quantity * p.price)) /
  (SELECT ROUND(SUM(o.quantity * p.price), 2) AS total_revenue
FROM order_details AS o
JOIN pizzas AS p
  ON o.pizza_id = p.pizza_id)) * 100, 2) AS Revenue
FROM pizza_types AS pt
JOIN pizzas AS p
  ON pt.pizza_type_id = p.pizza_type_id
JOIN order_details AS o
  ON o.pizza_id = p.pizza_id
  GROUP BY pt.category
  ORDER BY Revenue DESC;
```

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

# ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.



```sql
SELECT order_date,
SUM(Revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM(
SELECT o.order_date,
SUM(o1.quantity*p.price) AS Revenue
FROM order_details AS o1
JOIN pizzas AS p
  ON o1.pizza_id = p.pizza_id
JOIN orders AS o
  ON o.order_id  = o1.order_id
  GROUP BY o.order_date
  ) AS Sales
```

Result Grid | Filter Rows:

| 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 |
| 2015-01-16 | 36937.65000000001 |

# DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.



```sql
SELECT name, revenue
FROM
(SELECT category, name, revenue,
RANK() OVER(PARTITION BY category ORDER BY revenue  DESC) AS rn
FROM
(SELECT pt.category, pt.name,
SUM(o.quantity*p.price) AS revenue
FROM pizza_types AS pt
JOIN pizzas AS p
  ON pt.pizza_type_id = p.pizza_type_id
JOIN order_details AS o
  ON o.pizza_id = p.pizza_id
  GROUP BY pt.category, pt.name) AS a) AS b
WHERE rn <= 3  ;
```

Result Grid | Filter Rows: | Export:

| 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 |

# THANK YOU

## FOR ATTENTION