# PIZZA SALES ANALYSIS USING SQL

PROJECT BY PRIYANSHU KUMAR

# PROJECT OVERVIEW

The "Pizza Sales Analysis Using SQL" project examines pizza sales data to uncover key insights into ordering patterns and revenue generation. It retrieves total orders, calculates revenue, identifies the highest-priced pizza and most common size, and lists the top 5 most ordered types. By joining tables, it finds the total quantity of each pizza category, determines order distribution by hour, analyzes category-wise distribution, and calculates daily average orders. It also identifies the top 3 most ordered pizzas by revenue, calculates revenue contribution percentages, analyzes cumulative revenue over time, and highlights the top 3 pizzas by revenue for each category. This analysis offers recommendations for optimizing inventory, marketing strategies, and overall business performance.

# DATASETS

**Orders Table:**

- order_id: Unique identifier for each order
- order_date: Date and time when the order was placed
- customer_id: Unique identifier for each customer

**Order Details Table:**

- **order_id: Unique identifier for each order (foreign key)**
- **pizza_id: Unique identifier for each pizza (foreign key)**
- **quantity: Number of pizzas ordered**

**Pizzas Table:**

- pizza_id: Unique identifier for each pizza
- pizza_name: Name of the pizza
- category_id: Unique identifier for each pizza category (foreign key)
- size: Size of the pizza (e.g., small, medium, large)
- price: Price of the pizza

**Categories Table:**

- **category_id: Unique identifier for each category**
- **category_name: Name of the category (e.g., Veg, Non-Veg)**

# KEY OBJECTIVES :

**Basic 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 Analysis:**

1. Join the necessary tables to find the total quantity of each pizza category ordered.
2. Determine the distribution of orders by hour of the day.
3. Join relevant tables to find the category-wise distribution of pizzas.
4. Group the orders by date and calculate the average number of pizzas ordered per day.
5. Determine the top 3 most ordered pizza types based on revenue.

**Advanced Analysis:**

1. Calculate the percentage contribution of each pizza type to total revenue.
2. Analyze the cumulative revenue generated over time.
3. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

# Q1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

Result Grid

| | total_orders |
|---|---|
| ▶ | 21350 |

# Q2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

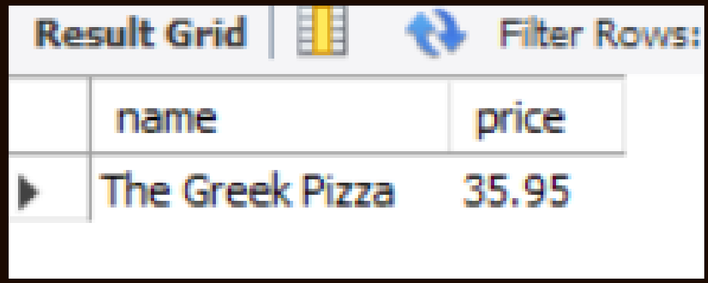
```sql
SELECT
    ROUND(SUM((order_details.quantity * pizzas.price)),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

Result Grid

| total_revenue |
| --- |
| 817860.05 |

# Q3. 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;
```



# Q4. 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 | Filte

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

# Q5. 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;
```

Result Grid | Filter Rows:

| 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 |

# Q6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
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;
```

Result Grid

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

# Q7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT hour(order_time) as hours ,
count(order_id) as orderCounts FROM orders
group by hour(order_time);
```

Result Grid | Filte

| hours | orderCounts |
|---|---|
| 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 |

# Q8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

Result Grid | Filter Row

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

# Q9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

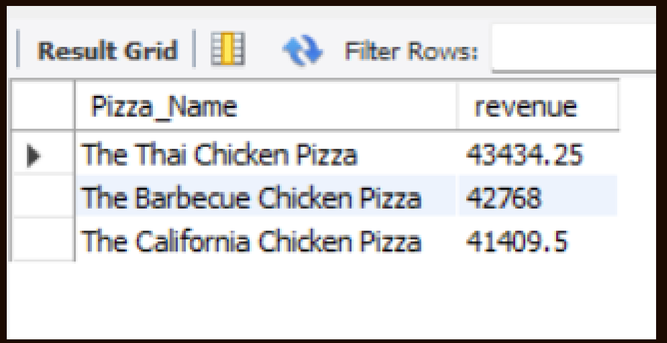
```sql
select round(avg(quantity),0) as Avg_pizzas_ordered_per_day
from
 (select orders.order_date , sum( order_details.quantity) as quantity
from orders join order_details
on orders.order_id = order_details.order_id
group by orders.order_date) as order_quantity;
```

Result Grid | Filter Rows:

| Avg_pizzas_ordered_per_day |
|---|
| 138 |

# Q10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
select pizza_types.name as Pizza_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 Grid | Filter Rows:

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

# Q11. 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;
```

Result Grid | Filter R

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

# Q12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```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;
```

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 |

# Q13. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
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,
round(sum((order_details.quantity) * pizzas.price),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, pizza_types.name) as a) as b
where rn <=3;
```

Result Grid | Filter Rows:

| category | name | revenue |
|---|---|---|
| Chicken | The Thai Chicken Pizza | 43434.25 |
| Chicken | The Barbecue Chicken Pizza | 42768 |
| Chicken | The California Chicken Pizza | 41409.5 |
| Classic | The Classic Deluxe Pizza | 38180.5 |
| Classic | The Hawaiian Pizza | 32273.25 |
| Classic | The Pepperoni Pizza | 30161.75 |
| Supreme | The Spicy Italian Pizza | 34831.25 |
| Supreme | The Italian Supreme Pizza | 33476.75 |
| Supreme | The Sicilian Pizza | 30940.5 |
| Veggie | The Four Cheese Pizza | 32265.7 |
| Veggie | The Mexicana Pizza | 26780.75 |
| Veggie | The Five Cheese Pizza | 26066.5 |

# RECOMMENDATIONS:

- Focus on Popular Items: Stock and promote common sizes and top 5 pizza types.
- Optimize Pricing: Adjust the pricing of high-value pizzas.
- Align Staffing: Staff according to peak order times.
- Targeted Marketing: Promote the top 3 revenue-generating pizzas.
- Improve Inventory: Manage inventory based on order trends.
- Track Revenue: Monitor revenue contributions for menu optimization.

These steps will enhance operations, customer satisfaction, and growth.

# CONCLUSION:

The "Pizza Sales Analysis Using SQL" project delivers crucial insights into customer preferences, order patterns, and revenue drivers. By examining total orders, revenue, popular pizza types, and distribution trends, the analysis provides actionable recommendations for optimizing inventory, staffing, pricing, and marketing strategies. These insights enable data-driven decision-making, improve operational efficiency, enhance customer satisfaction, and drive business growth. This project highlights the power of SQL in transforming raw data into valuable business intelligence.