# PIZZAHUT SALES REPORT

YUMMY CHESSY PIZZAS

50% OFF

BUY 2 GET 1 FREE

VARIATES OF PIZZAS

# LIST OF PIZZA CATEGORY

- CHICKEN
- CLASSIC
- SUPREME
- VEGGIS

# VARIATIONS

CLASSIC

SUPREME

CHICKEN

VIGGIES

# INTRODUCTION

Overall, pizza sales data analysis is crucial for Pizza Hut to optimize operations, enhance customer satisfaction, and maintain a competitive edge in the food service industry.

# SQL DATA ANALYSIS

**THE PIZZA SALES ANALYSIS AMIS IS TO EXPLORE AND ANALYSE THE SALES PERFORMANCE OF PIZZA USING EXPLORATORY DATA ANALYSIS.**

**PIZZA SALES ANALYSIS PROVIDES INFORMATION ABOUT MAXIMUM ORDER QUANTITY,REVENUE,MOST FAVOURATE PIZZA TYPE OF CUSTOMERS AND MAXIMUM ORDERED PIZZA BASED ON CATEGOURIES.**

**Their are total four table in this projects**

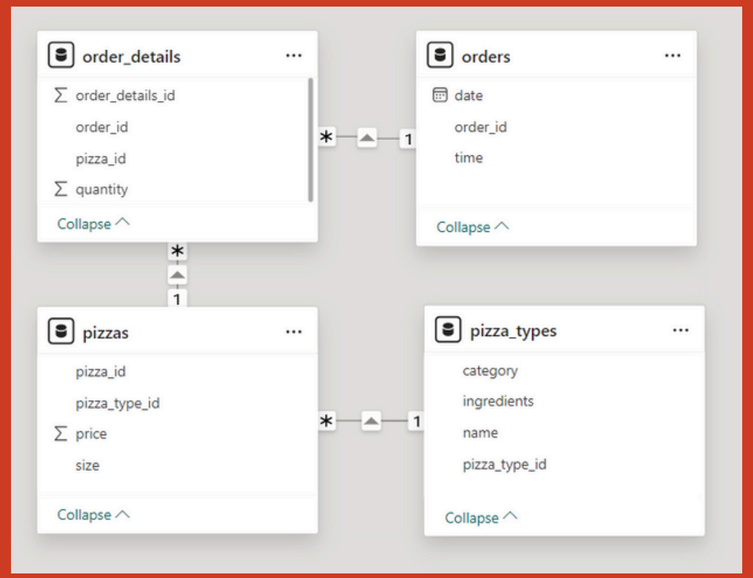
**1 ) order_details**

**2 ) orders**

**3 ) pizzas**

**4) pizza_types**

## DATA MODELING



- Here we can see the connection within multiple tables, which will help to perform joins to extract common elements.
- The order_details table is connected with the pizzas table using pizza_id and with the orders table using order_id.

# OVERVIEW OF TABLES

## PIZZAS TABLE

| pizza_id | pizza_type_id | size | price |
|---|---|---|---|
| bbq_ckn_s | bbq_ckn | S | 12.75 |
| bbq_ckn_m | bbq_ckn | M | 16.75 |
| bbq_ckn_l | bbq_ckn | L | 20.75 |
| cali_ckn_s | cali_ckn | S | 12.75 |
| cali_ckn_m | cali_ckn | M | 16.75 |
| cali_ckn_l | cali_ckn | L | 20.75 |

## ORDERS TABLE

| order_id | order_date | order_time |
|---|---|---|
| 1 | 2015-01-01 | 11:38:36 |
| 2 | 2015-01-01 | 11:57:40 |
| 3 | 2015-01-01 | 12:12:28 |
| 4 | 2015-01-01 | 12:16:31 |
| 5 | 2015-01-01 | 12:21:30 |
| 6 | 2015-01-01 | 12:29:36 |

| pizza_type_id | name | category | ingredients |
|---|---|---|---|
| peppr_salami | The Pepper Salami Pizza | Supreme | Genoa Salami, Capocollo, Pepperoni, Tomatoes,... |
| prsc_argla | The Prosciutto and Arugula ... | Supreme | Prosciutto di San Daniele, Arugula, Mozzarella C... |
| sicilian | The Sicilian Pizza | Supreme | Coarse Sicilian Salami, Tomatoes, Green Olives, ... |
| soppressata | The Soppressata Pizza | Supreme | Soppressata Salami, Fontina Cheese, Mozzarell... |
| spicy_ital | The Spicy Italian Pizza | Supreme | Capocollo, Tomatoes, Goat Cheese, Artichokes,... |
| spinach_supr | The Spinach Supreme Pizza | Supreme | Spinach, Red Onions, Pepperoni, Tomatoes, Art... |

PIZZA_TYPE_DETAILS

| order_detail_id | order_id | pizza_id | quantity |
|---|---|---|---|
| 1 | 1 | hawaiian_m | 1 |
| 2 | 2 | classic_dlx_m | 1 |
| 3 | 2 | five_cheese_l | 1 |
| 4 | 2 | ital_supr_l | 1 |
| 5 | 2 | mexicana_m | 1 |
| 6 | 2 | thai_ckn_l | 1 |

ORDER_DETAILS

# Q1 Retrieve the total number of orders placed.

```
SELECT COUNT(order_id) AS Total_Orders FROM order_details;
```

- To obtain the total number of orders placed, we used the order_details table because order_id contains distinct values in the entire table.
- COUNT is an aggregate function used to determine the total number of entities present in a particular column.

This is our output in MYSQL :

| | Total_Orders |
|---|---|
| ▶ | 48620 |

**THE TOTAL NUMBER OF ORDER PLACED IS 48620**

# Q2 calculate the total revenue generated from 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;
```

| | Total_Revenue |
|---|---|
| ▶ | 817860.05 |

**THE TOTAL REVENUE GENERATED IS 817860.05;**

- The total revenue has been calculated by multiplying the quantity and price.
- Quantities are in the order_details table, and the prices of pizzas are in the pizzas table.
- To join both tables, we used the JOIN command, which is a useful command in SQL and is usually used to join multiple tables according to their requirements.
- Here, order_details and pizzas are joined using the JOIN command, and the common column in both tables is pizza_id.

# Q3 Identify the highest-priced pizza.

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

| | name | price |
|---|---|---|
| | The Greek Pizza | 35.95 |

Master Blaster

**THE HIGHEST-PRICED PIZZA IS THE GREECK PIZZA**

- The highest-price pizza can be obtained from the pizza_types and pizzas tables because these tables contain the name and price of the pizzas.
- We used JOIN to connect the two tables, with the common column being pizza_type_id. After connecting, we used the ORDER BY clause to arrange the rows in sequential order.
- To find the highest price, the LIMIT clause is used, which restricts the result to display only one row.

# Q4 Identify the most common pizza size ordered.

```
SELECT
   pizzas.size,
   COUNT(order_details.order_detail_id) AS id_count
FROM
   order_details
      JOIN
   pizzas ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY id_count DESC
LIMIT 1;
```

| | size | id_count |
|---|---|---|
| ▶ | L | 18526 |

**THE MOST COMMON PIZZA SIZE ORDERED IS LARGE.**

- Here we want to find most ordered pizza size.
- Which can be obtained from the table pizzas and order_details because size and order_detail_id are present in this table.
- This two tables are joined with pizza_id which is common in both the columns.
- Then we have used group by clause to group the data with respect to their size to find count of order_detail id.
- Later order by and limit clause is used to find most common pizza size.

# Q5 list the top 5 most ordered pizza types along with their quantities.

- To obtain the top 5 most ordered pizza types along with their quantities, we need to connect three tables because we want the name of the pizza and their respective quantities.
- There is no direct connection between pizza_types and order_details, so we have to connect them via the pizzas table.
- The top 5 most ordered pizza types are obtained using the GROUP BY clause on the name and the ORDER BY clause on the quantity.

```
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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

**THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES ARE :-**

| 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.

```
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 pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC
LIMIT 5;
```

**THE TOTAL QUANTITY OF EACH PIZZA ORDERED IS :-**

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

- Overall, there are four categories (classic, supreme, chicken, veggies). To find the total quantity of each pizza category ordered, we need to join the pizza_types and order_details tables.
- Since there is no direct connection between these two tables, we use the pizzas table to perform the join.
- Then, we use the GROUP BY clause on the category to obtain the total quantity of each pizza category.

# Q7 Determine the distribution of orders by hour of the day.

```
SELECT
   HOUR(order_time) AS
hours, COUNT(order_id)
FROM
   orders
GROUP BY hours
ORDER BY hours;
```

**THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY IS :-**

| hours | COUNT(order_id) |
|---|---|
| 9 | 1 |
| 10 | 8 |
| 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 |

- The distribution of orders by hour of the day is obtained from the orders table. In the orders table, the column order_time consists of the time at which orders have been placed.
- Using the GROUP BY and ORDER BY clauses, the desired outcome is obtained.

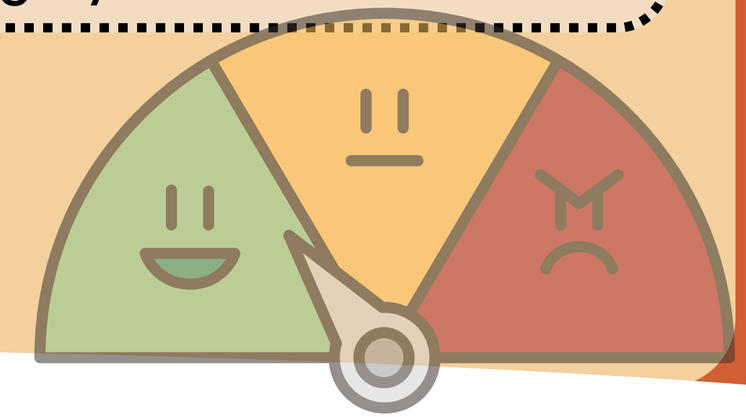
## Q8 join relevant tables to find the category-wise distribution of pizzas.

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

- To obtain the category-wise distribution of pizzas, we need the pizza_types table because this table contains the name and category of pizzas.
- The GROUP BY clause is then used to group the pizzas by category.

**THE CATEGORY-WISE DISTRIBUTION OF PIZZAS ARE :-**

| pizzas | category |
|---|---|
| 6 | Chicken |
| 8 | Classic |
| 9 | Supreme |
| 9 | Veggie |



# Q9 Group the orders by date and calculate the average number of pizzas ordered per day.

- First, we will find the order_date from the orders table and the sum of quantity from the order_details table.
- First, we will find the order_date from the orders table and the sum of quantity from the order_details table.

```
SELECT
   ROUND(AVG(quantity), 0)
FROM
   (SELECT
      orders.order_date, SUM(order_details.quantity)
AS quantity
   FROM
      order_details
   JOIN orders ON order_details.order_id =
orders.order_id
   GROUP BY orders.order_date) AS Order_Quantity;
```

**THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY IS 138**

| | Average_Orders |
|---|---|
| ▶ | 138 |

# Q10 Determine the top 3 most ordered pizza types based on revenue.

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

**THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE IS :-**

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

- Revenue is calculated by multiplying the price and quantity, which are in the pizzas and order_details tables, respectively.
- Since there is no direct connection between these two tables, we first connect pizzas with pizza_types and then connect pizzas to order_details.
- Finally, to obtain the top 3 most ordered pizza types based on revenue, we use the GROUP BY clause to group by name and the ORDER BY clause to find the top 3 values.

# Q11 Calculate the percentage contribution of each pizza type to total revenue.

```
 select pizza_types.category,
round(sum(pizzas.price*order_details.quantity)*0.0001222
70308,1) as Percentage_contribution FROM
   pizzas
      JOIN
   pizza_types ON pizzas.pizza_type_id =
pizza_types.pizza_type_id
      JOIN
   order_details ON pizzas.pizza_id =
order_details.pizza_id group by pizza_types.category;
   use pizzahut;
```

**THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPRE TO TOTAL REVENUE IS :-**

| category | Percentage_contribution |
|---|---|
| Classic | 26.9 |
| Veggie | 23.7 |
| Supreme | 25.5 |
| Chicken | 24 |

- Since we have four pizza types, to find the percentage contribution of each pizza type to the total revenue, the multiplying factor is obtained by 100 / Total_Revenue because the formula for calculating individual percentage is (individual_value / Total_value) * 100
- To find the category and its contribution to the total quantity sold, we need to perform three joins: one from pizzas to pizza_types and then from pizzas to order_details. The GROUP BY and ORDER BY clauses are not required.

# Q12 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(pizzas.price*order_d
etails.quantity)AS revenue FROM pizzas JOIN
order_details
ON pizzas.pizza_id=order_details.pizza_id
JOIN orders ON
orders.order_id=order_details.order_id
 group by orders.order_date) AS sales;
```

**THE CUMULATIVE REVENUE GENERATED OVER TIME IS :-**

| order_date | cum_revenue |
|---|---|
| 2015-01-01 | 2714 |
| 2015-01-02 | 5446 |
| 2015-01-03 | 8108 |
| 2015-01-04 | 9863 |
| 2015-01-05 | 11929 |
| 2015-01-06 | 14358 |
| 2015-01-07 | 16560 |
| 2015-01-08 | 19398 |
| 2015-01-09 | 21525 |
| 2015-01-10 | 23989 |
| 2015-01-11 | 25861 |
| 2015-01-12 | 27780 |
| 2015-01-13 | 29830 |
| 2015-01-14 | 32357 |
| 2015-01-15 | 34342 |
| 2015-01-16 | 36936 |
| 2015-01-17 | 39000 |
| 2015-01-18 | 40977 |
| 2015-01-19 | 43364 |

- To calculate the cumulative revenue generated over time on a daily basis, the revenue of each day is added to the total of the previous days' revenue.
- The operational tables are pizzas, orders, and order_details.
- The order date and the sum of revenue are extracted from these tables, and relevant common columns are used to perform the joins.
- The output of the select query is then classified as a subquery within another query.
- The OVER function is used to create the cumulative addition of revenue obtained on a daily basis in the subquery.

# Q13 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 rn from
(select
P_T.category,P_T.name,sum(P.price*O_D.quantity) as
revenue from
pizza_types P_T join pizzas P on
P.pizza_type_id=P_T.pizza_type_id join order_details
O_D on
O_D.pizza_id=P.pizza_id group by
P_T.category,P_T.name) as a) as b
where rn<=3);
```

**THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY :-**

| 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 |

- To rank pizza categories based on their revenue, there are a total of 4 categories and within these categories, there are 32 different sub-categories of pizzas.
- To find the top 3 most ordered pizzas based on sub-categories, we need to perform subqueries.
- First, we select the category, name, and revenue, and use an alias to name that table as a. Then, we select the revenue, name, and category, and form a partition on categories, arranging it in rank order using the RANK function and aliasing it as b.
- Lastly, we extract the name and revenue from table b.

# Conclusion

**In pizzas order most people prefers to order large size pizza.**

1. Value for Money: Customers might perceive large pizzas as providing better value for the price compared to smaller sizes.
2. Group Consumption: Large pizzas are more suitable for sharing among groups, making them a popular choice for gatherings and parties.

**Customers frequently visit between 12-2 PM in the afternoon and 5-7 PM in the evening.**

1. Lunch Hour: Many people take their lunch break between 12-2 PM, making it a convenient time to grab a quick and satisfying meal like pizza.
2. End of Workday: Many people finish work around 5 PM and prefer to pick up or order dinner on their way home.

# Business strategies

- Businesses might use this conclusion to adjust their inventory, marketing strategies, and pricing to cater to this preference, ensuring they have enough large pizzas available to meet demand and potentially offering promotions that highlight the value of larger sizes.

- Increase Staff During Peak Hours: Schedule more staff during early afternoon and early evening to ensure prompt service and reduce wait times.

- Train Staff for Efficiency: Make sure your staff is well-trained to handle the rush efficiently, maintaining quality and speed.

# Thankyou

"Thank you all for your time and attention. If you have any questions or need further information, feel free to ask now or reach out to me later. I appreciate your interest and participation."