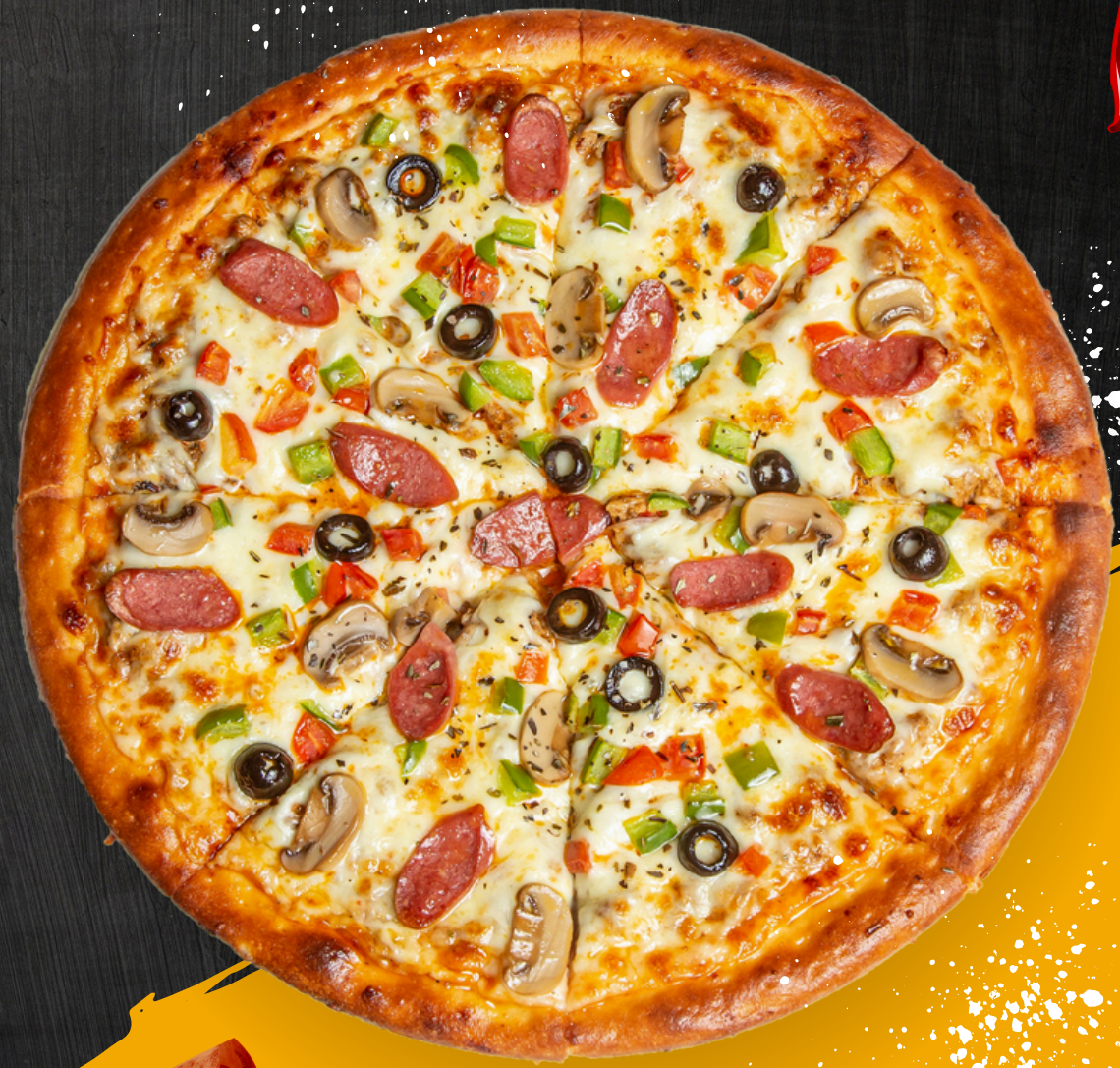
Insight BI

MySQL Project On

PIZZA HUT



KULDEEP SOSE

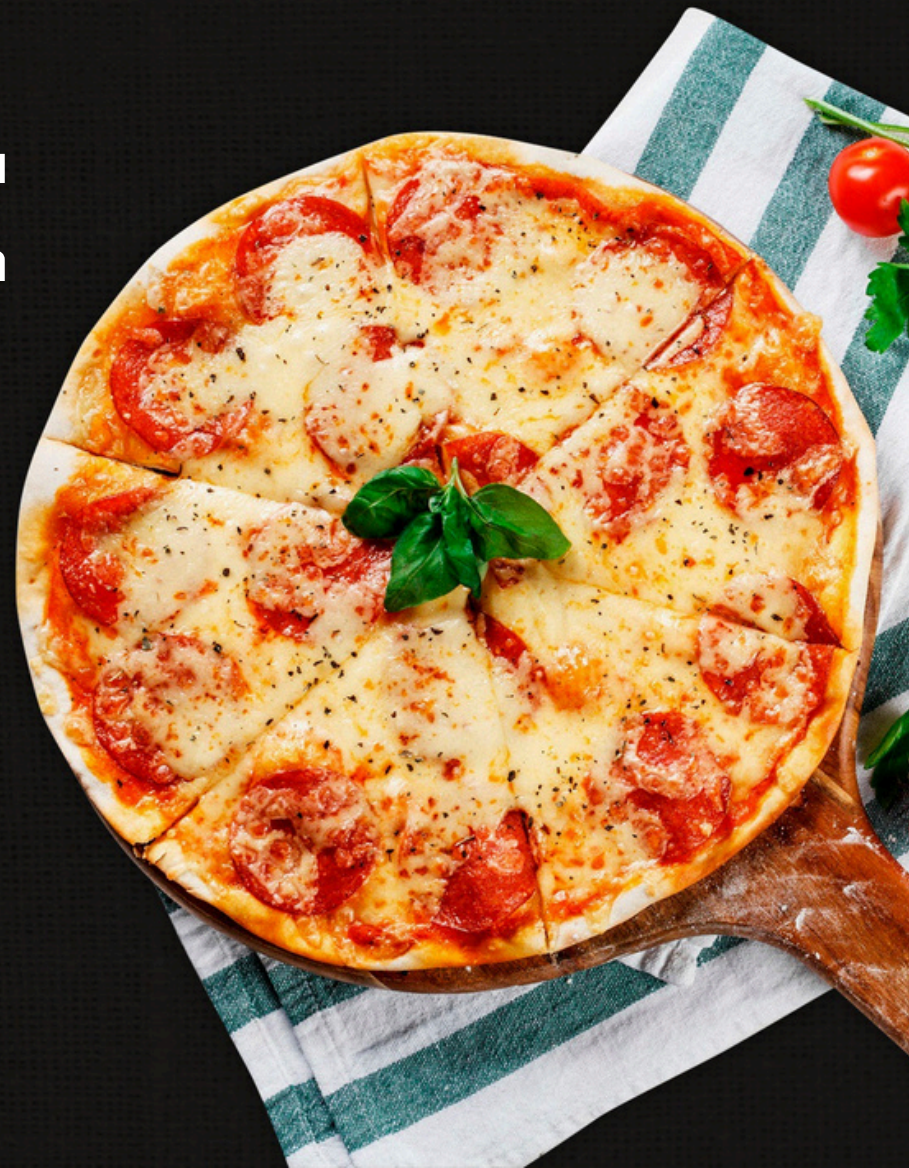
MBA | Power BI | My SQL

kuldeepsose20022@gmail.com

http://linkedin.com/in/kuldeep-sose

Creating My first project using MySQL for a pizza delivery application like "Pizza Hut" can be an exciting journey!

Starting with a project like Pizza Hut is a great way to learn MySQL and database-driven application development.



PIZZA HUT

Pizza Hut, LLC is an American multinational pizza restaurant chain and international franchise founded in 1958 in Wichita, Kansas by Dan and Frank Carney. The chain, headquartered in Plano, Texas, operates 19,866 restaurants worldwide as of 2023.

At Pizza Hut, we don't just make pizza. We make people happy. Pizza Hut was built on the belief that pizza night should be special, and we carry that belief into everything we do. With more than 60 years of experience under our belts, we understand how to best serve our customers through tried.

QUESTIONERY

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

CREATE SCHEMA

Here's a basic schema for a Pizza Hut database:

```sql
create database pizzahunt;
use pizzahunt;

CREATE TABLE orders (
    order_ID INT NOT NULL,
    order_date DATE NOT NULL,
    order_time TIME NOT NULL,
    PRIMARY KEY (order_Id)
);

CREATE TABLE orders_details (
    orders_details_ID INT PRIMARY KEY NOT NULL,
    order_ID INT NOT NULL,
    pizza_ID TEXT NOT NULL,
    quantity INT NOT NULL
);
```

1. Retrieve the total number of orders placed.

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

Result:

Result Grid | Filter Rows:

	total_orders
▶	21350

2. Calculate the total revenue generated from pizza sales.

```sql
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;
```

Result:

Result Grid | Filter Rows:

	total_sales
▶	817860.05

3. 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:

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

4. Identify the most common pizza size ordered.

```
SELECT
    pizzas.size,
    COUNT(orders_details.orders_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:

Result Grid | Filter Rows:

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

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

```sql
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;
```

Result:

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

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

```
3 •  SELECT
4        pizza_types.category,
5        SUM(orders_details.quantity) AS quantity
6    FROM
7        pizza_types
8            JOIN
9        pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10           JOIN
11       orders_details ON orders_details.pizza_ID = pizzas.pizza_id
12   GROUP BY pizza_types.category
13   ORDER BY quantity DESC;
14
```

Result:

Result Grid | Filter Rows:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

7. 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;
```

Result:

Result Grid | Filter Rows:

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
22	663
23	28

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

```
SELECT
    (pizza_types.category) AS Category,
    COUNT(pizza_types.name) AS Pizzas_Types
FROM
    pizza_types
GROUP BY Category;
```

Result:

Result Grid | Filter Rows:

Category	Pizzas_Types
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

```
SELECT
    ROUND(AVG(Avg_order), 0)
FROM
    (SELECT
        DATE(orders.order_date) AS Date,
            SUM(orders_details.quantity) AS Avg_order
    FROM
        orders
    JOIN orders_details ON orders.order_ID = orders_details.order_ID
    GROUP BY Date) AS order_quantity;
```

Result:

Result Grid | Filter Rows:

ROUND(AVG(Avg_order), 0)
138

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

```
SELECT
    pizza_types.name,
    SUM(pizzas.price * orders_details.quantity) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_ID
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

Result:

Result Grid | Filter Rows:

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

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

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

Result:

Result Grid | Filter Rows:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

12. 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,
round(sum(orders_details.quantity * pizzas.price),0) as revenue
from orders join orders_details
on orders_details.order_ID = orders.order_ID
join pizzas
on pizzas.pizza_id = orders_details.pizza_ID
group by orders.order_date) as sales;
```

Result:

Result Grid | Filter Rows:

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
2015-01-20	45762
2015-01-21	47803

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

```
5  (select category, name , revenue,
6  rank() over(partition by category order by revenue desc) As RN
7  From
8  (select pizza_types.category, pizza_types.name,
9  sum(orders_details.quantity * pizzas.price) as revenue
10 from pizza_types join pizzass
11 on pizza_types.pizza_type_id = pizzas.pizza_type_id
12 join orders_details
13 on orders_details.pizza_id = pizzas.pizza_id
14 group by pizza_types.category, pizza_types.name) as A) as b
15 where rn <=3;
```

Result:

Result Grid | Filter Rows:

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

Each task involves SQL queries that retrieve specific information from the dataset, such as order details, pizza types, sizes, prices, and order timestamps. By executing these queries, I gain insights into various aspects of the pizza delivery business, including sales performance, customer preferences, and revenue trends.

Determining the distribution of orders by hour of the day, category-wise distribution of pizzas, average number of pizzas ordered per day, percentage contribution of each pizza type to total revenue, cumulative revenue generated over time, and top 3 most ordered pizza types based on revenue for each pizza category, I would need more information about the database schema and table structures to provide accurate findings.

THANK YOU

WsCube Tech