PIZZA

SALES ANALYSIS USING SQL

Brief

IN THIS PROJECT, I USED SQL TO ANALYZE PIZZA SALES DATA, EXTRACT MEANINGFUL INSIGHTS, AND ANSWER KEY BUSINESS QUESTIONS. THE GOAL WAS TO DEMONSTRATE DATA HANDLING, QUERYING, AND STORYTELLING USING REAL-WORLD SCENARIOS.

Why This Project?

THIS PROJECT SIMULATES A BUSINESS SCENARIO OF A PIZZA STORE CHAIN. USING RAW TRANSACTIONAL DATA, I APPLIED SQL TECHNIQUES TO PERFORM EDA (EXPLORATORY DATA ANALYSIS), UNCOVERING PATTERNS IN SALES, REVENUE, ORDER TRENDS, AND POPULAR MENU ITEMS.

TOOL USED: MYSQL

Tables Overview

pizzas
pizza_id
pizza_type_id
size
price

order_details
order_details_id
order_id
pizza_id
quantity

pizza_types
pizza_type_id
name
category
ingredients

orders
order_id
order_date
order_time

Basic Questions

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.

Intermediate Questions

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

Advanced Questions

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

Retrieve the total number of orders placed.

Query:

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

Calculate the total revenue generated from pizza sales.

Query:

```sql
SELECT
    ROUND(SUM(od.quantity * p.price), 3) AS total_revenue
FROM
    order_details od
        INNER JOIN
    pizzas p ON od.pizza_id = p.pizza_id;
```

Identify the highest-priced pizza.

Query:

```sql
SELECT
    pt.name, max(p.price) AS Highest_Price_Pizza
FROM
    pizza_types pt
        INNER JOIN
    pizzas p ON p.pizza_type_id = pt.pizza_type_id
group by pt.name order by max(p.price) desc limit 1;
```

Identify the most common pizza size ordered.

Query:

```sql
select p.size, count(od.quantity) as Total_Time_Ordered
from pizzas p
inner join order_details od
on p.pizza_id=od.pizza_id
group by p.size
order by count(od.quantity) desc
limit 1;
```

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

Query:

```sql
SELECT
    pizza_types.name, SUM(order_details.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 SUM(order_details.quantity) DESC
LIMIT 5;
```

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

Query:

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

Determine the distribution of orders by hour of the day.

Query:

```sql
select hour(order_time), count(order_id) from orders
group by hour(order_time)
order by count(order_id) desc;
```

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

Query:

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

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

Query:

```sql
select avg(quantity) from
(select o.order_date, sum(od.quantity) as quantity
from orders o
join order_details od
on o.order_id=od.order_id
group by o.order_date) as ordered_quantity;
```

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

Query:

```sql
select pt.name, sum(p.price*od.quantity) as Total_earning
from pizzas p
join pizza_types pt on p.pizza_type_id=pt.pizza_type_id
join order_details od on p.pizza_id=od.pizza_id
group by pt.name
order by total_earning desc
limit 3;
```

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

Query:

```sql
select pt.category, round((sum(od.quantity*p.price)/817860.050)*100,2) as revenue
from pizzas p
join pizza_types pt on p.pizza_type_id=pt.pizza_type_id
join order_details od on p.pizza_id=od.pizza_id
group by pt.category
order by revenue;
```

Analyze the cumulative revenue generated over time.

Query:

```sql
select order_date ,revenue, sum(revenue) over (order by order_date) as cum_revenue from
(select o.order_date, sum(od.quantity*p.price) as revenue
from order_details od
join orders o on o.order_id=od.order_id
join pizzas p on p.pizza_id=od.pizza_id
group by o.order_date
order by revenue) as sales;
```

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

Query:

```sql
select category, 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(od.quantity*p.price) as revenue
from pizza_types pt
join pizzas p on p.pizza_type_id=pt.pizza_type_id
join order_details od on od.pizza_id=p.pizza_id
group by pt.category, pt.name) as a) as b
where rn<=3;
```

Summary of Insights

- Total Revenue Generated : 817860 $
- Most common pizza size ordered: Large
- Most ordered pizza: The Classic Deluxe Pizza
- Peak hours are 12PM to 1PM and 5PM to 7PM
- Avg number of pizza ordered per day: 138
- Chicken pizzas are more popular.

What I learned?

- Gained hands-on SQL practice with group by, joins, CTEs, window functions.
- Improved analytical thinking through business use cases.

Find the full project with accessible SQL scripts on github

- Click here for GitHub link

Contact me:

Email: dhokareabhi@gmail.com