Pizza Sales Analysis Using SQL

Objective

The goal of this project is to analyse pizza sales data using SQL to uncover trends, identify high-performing products, and understand customer purchasing patterns over time.

Dataset Description

- Source: Kaggle

- Tables:

- order_details (order_details_id, order_id, pizza_id, quantity)

- orders (order_id, date, time)

- pizza_types (pizza_type_id, name, category, ingredients)

- pizzas ((pizza_id, pizza_type_id, size, price)

- Tool Used: Microsoft SQL Server

SQL QUERIES

```
--Retrieve the total number of orders placed.

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

Results | Messages

	total_orders
1	21350

```sql
--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;
```

Results Messages

	total_sales
1	817860.05

```sql
--Identify the highest-priced pizza.


SELECT TOP 1
    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;
```

Results Messages

	name	price
1	The Greek Pizza	35.9500007629395

```sql
--Identify the most common pizza size ordered.

SELECT TOP 1
    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;
```

Results Messages

	size	order_count
1	L	18526

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

    SELECT TOP 5
    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;
```

Results | Messages

	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

```sql
  --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 order_details.pizza_id = pizzas.pizza_id
GROUP BY
    pizza_types.category
ORDER BY
    quantity DESC;
```

Results | Messages

	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

```sql
--Determine the distribution of orders by hour of the day.

SELECT
    DATEPART(HOUR, time) AS hour,
    COUNT(order_id) AS order_count
FROM
    orders
GROUP BY
    DATEPART(HOUR, time)
ORDER BY
    hour;
```

Results | Messages

	hour	order_count
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399
11	19	2009
12	20	1642
13	21	1198
14	22	663
15	23	28

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

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

Results | Messages

	category	(No column name)
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

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


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

Results Messages

	average_quantity
1	138

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

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

Results Messages

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

```sql
--Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) * 100.0 /
        (SELECT SUM(order_details.quantity * pizzas.price)
         FROM order_details
         JOIN pizzas ON pizzas.pizza_id = order_details.pizza_id), 2) AS percentage_of_total_sales
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
    percentage_of_total_sales DESC;
```

Results | Messages

	category	percentage_of_total_sales
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

```sql
    --Analyze the cumulative revenue generated over time.

SELECT
    date,
    SUM(revenue) OVER (ORDER BY date) AS cum_revenue
FROM (
    SELECT
        orders.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.date
) AS Sales
ORDER BY
    date;
```

Results | Messages

	date	cum_revenue
1	2015-01-01	2713.85000228882
2	2015-01-02	5445.7500038147
3	2015-01-03	8108.15000724792
4	2015-01-04	9863.60000801086
5	2015-01-05	11929.5500087738
6	2015-01-06	14358.5000114441
7	2015-01-07	16560.700012207
8	2015-01-08	19399.0500183105
9	2015-01-09	21526.4000225067
10	2015-01-10	23990.350025177

```sql
    --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 pizza_types.category, pizza_types.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.category, pizza_types.name
) as a) as b
where rn <=3;
```

Results | Messages

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5
4	The Classic Deluxe Pizza	38180.5
5	The Hawaiian Pizza	32273.25
6	The Pepperoni Pizza	30161.75
7	The Spicy Italian Pizza	34831.25
8	The Italian Supreme Pizza	33476.75
9	The Sicilian Pizza	30940.5
10	The Four Cheese Pizza	32265.7010040283
11	The Mexicana Pizza	26780.75
12	The Five Cheese Pizza	26066.5

Insight :

- Total revenue generated is $817,860.05
- Classic and Chicken categories generate the most revenue.
- Steady revenue growth, with spikes during weekends and holidays.
- The top-selling pizzas are Classic Deluxe, BBQ Chicken, and Five Cheese.

Key Learnings :

Learned to use joins, group by, window functions, and aggregation in SQL.

- Understood how to derive real-world business insights from raw sales data.

- Practiced working with a relational schema and writing optimized queries.

Conclusion :

This analysis helped identify the best-performing pizzas and revenue patterns. The business can focus on high-performing categories and target weekend promotions based on trends.