PORTFOLIO PROJECT:

DATA ANALYSIS OF PIZZA SALES USING SQL.

=- MOHAMMED MOIN AHMED

PROJECT - PIZZASALES

Hello, I am Mohammed Moin Ahmed. In this project, I applied SQL queries to analyze pizza sales data, focusing on trends, customer preferences, and inventory management.

Schema Overview:

The Pizza sales database consists of four interconnected tables:

- *pizza_types:* Stores information about different pizza types, identified by pizza_type_id.
- *pizzas*: Contains details about each pizza, identified by pizza_id, including its size and price. It references pizza_types via the pizza_type_id foreign key.
- *orders*: Tracks customer orders, uniquely identified by order_id.
- *orderdetails*: Links individual pizzas to specific orders, referencing both orders through order_id and pizzas through pizza_id.

This structure allows for efficient querying of pizza sales, types, and order details.

SOLVED QUESTIONS ON THREE CATEGORIES:

Basic:

- 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:

- 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:

- 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.

Q1. Retrieve the total number of orders placed.

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

Result Grid

	total_orders
▶	21350

Q2. Calculate the total revenue generated from pizza sales.

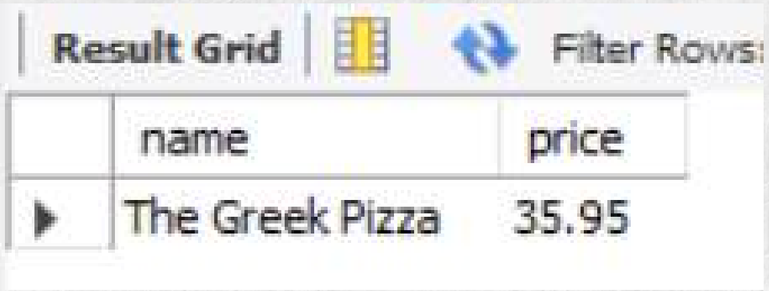
```
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid

	total_sales
▶	817860.05

Q3. 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;
```



	name	price
▶	The Greek Pizza	35.95

Q4. Identify the most common pizza size ordered.

```sql
SELECT
    pizzas.size, COUNT(orders_details.quantity) 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
LIMIT 1;
```

Result Grid | Filt

	size	order_count
▶	L	18526

Q5. 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 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.

```
SELECT
    pizza_types.category,
    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.category
ORDER BY Quantity DESC;
```

Result Grid | Filte

	category	Quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

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

```
SELECT hour(order_time), COUNT(order_id) AS order_count
FROM orders
GROUP BY hour(order_time)
ORDER BY order_count DESC;
```

Result Grid | Filter Rows:

hour(order_time)	order_count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472
15	1468
11	1231
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_per_day
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS Order_quantity;
```

Result Grid

Avg_pizzas_per_day
138

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

```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS Revenue
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 Revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

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.

Using SubQueires

```sql
SELECT
    pizza_types.category,
    CONCAT(ROUND((SUM(orders_details.quantity * pizzas.price) / (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 pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizza_types.category
ORDER BY Revenue DESC;
```

Result Grid | Filte

	category	Revenue
▶	Classic	26.91%
	Supreme	25.46%
	Chicken	23.96%
	Veggie	23.68%

Continue Next Page

= Using CTEs

```sql
WITH Cte1 AS
(SELECT pizza_types.category,
round(SUM(orders_details.quantity * pizzas.price), 2) AS Total_sales
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.category)
SELECT category,
concat(round((Total_sales/(SELECT SUM(Total_sales) FROM cte1))*100, 2), '%') AS Revenue_percent
FROM Cte1
ORDER BY Revenue_percent DESC;
```

Result Grid | Filter Rows:

category	Revenue_percent
Classic	26.91%
Supreme	25.46%
Chicken	23.96%
Veggie	23.68%

Q12. Analyze the cumulative revenue generated over time.

Using SubQueires

```
SELECT order_date,
SUM(Revenue) OVER(ORDER BY order_date) AS Cum_Revenue
FROM
(SELECT orders.order_date,
SUM(orders_details.quantity * pizzas.price) AS Revenue
FROM orders_details JOIN pizzas
ON orders_details.pizza_id = pizzas.pizza_id
JOIN orders
ON orders_details.order_id = orders.order_id
GROUP BY orders.order_date) AS Sales;
```

Result Grid | Filter Rows:

order_date	Cum_Revenue
2015-01-01	2713.85
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
2015-01-10	23990.35
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3

.....

Continue Next Page

= Using CTEs

```
WITH Per_day_Sales AS
(SELECT orders.order_date,
round(SUM(orders_details.quantity * pizzas.price), 2) AS Revenue
FROM orders_details JOIN pizzas
ON orders_details.pizza_id = pizzas.pizza_id
JOIN orders
ON orders_details.order_id = orders.order_id
GROUP BY orders.order_date)
SELECT order_date, Revenue,
round(SUM(Revenue) OVER(ORDER BY order_date), 2) AS Cum_Revenue
FROM Per_day_Sales;
```

Result Grid | Filter Rows:

order_date	Revenue	Cum_Revenue
2015-01-01	2713.85	2713.85
2015-01-02	2731.9	5445.75
2015-01-03	2662.4	8108.15
2015-01-04	1755.45	9863.6
2015-01-05	2065.95	11929.55
2015-01-06	2428.95	14358.5
2015-01-07	2202.2	16560.7
2015-01-08	2838.35	19399.05
2015-01-09	2127.35	21526.4
2015-01-10	2463.95	23990.35
2015-01-11	1872.3	25862.65
2015-01-12	1919.05	27781.7
2015-01-13	2049.6	29831.3

Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Using SubQueires

```
SELECT category, name, revenue
FROM
(SELECT category, name, revenue,
RANK() OVER(PARTITION BY category ORDER BY Revenue DESC) AS Rank_Num
FROM
(SELECT pizza_types.category, pizza_types.name,
SUM(orders_details.quantity * pizzas.price) 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.category, pizza_types.name) AS A) AS B
WHERE Rank_Num <= 3 ;
```

Result Grid | Filter Rows: | Export:

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	32265.70000000065
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

Continue Next Page

= Using CTEs

```
WITH cte1 (category, name, Revenue)  AS
(SELECT pizza_types.category, pizza_types.name,
SUM(orders_details.quantity * pizzas.price) 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.category, pizza_types.name), cte_rank AS
(SELECT *,
RANK() OVER(PARTITION BY category ORDER BY Revenue DESC) AS Rank_Num
FROM cte1)
SELECT *
FROM cte_rank
WHERE Rank_Num <= 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

category	name	Revenue	Rank_Num
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Spicy Italian Pizza	34831.25	1
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Sicilian Pizza	30940.5	3
Veggie	The Four Cheese Pizza	32265.70000000065	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

THANK YOU!

Email - mdmoin94@gmail.com
LinkedIn - www.linkedin.com/in/mohammedmoinahmed