DELICIOUS PIZZA

WELCOME TO MY SQL PIZZA SALES ANALYSIS! 🍕💡

MY NAME IS MOHAMMAD SHAHID,
In this project, I have solved many SQL questions to analyze pizza sales data. From identifying top-selling pizzas to understanding customer preferences and revenue trends, this analysis showcases how SQL can turn raw data into valuable business insights.
Join me as I break down key findings and demonstrate the power of SQL in real-world data analysis! 🚀📊

DELICIOUS PIZZA

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.

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.

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.

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

Result Grid

total_orders
21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND((o.quantity * p.price), 2) AS total_revenue
FROM
    order_details o
        INNER JOIN
    pizzas p ON o.pizza_id = p.pizza_id;
```

Result Grid

	total_revenue
▶	13.25
	16
	18.5

IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
SELECT
    t.name AS pizza_name, p.price AS highest_priced_pizza
FROM
    pizza_types t
        INNER JOIN
    pizzas p ON t.pizza_type_id = p.pizza_type_id
ORDER BY highest_priced_pizza DESC
LIMIT 1;
```

Result Grid | Filter Rows:

pizza_name	highest_priced_pizza
The Greek Pizza	35.95

IIDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
SELECT
    p.size AS pizza_size,
    COUNT(d.order_details_id) AS order_count
FROM
    pizzas p
        INNER JOIN
    order_details d ON p.pizza_id = d.pizza_id
GROUP BY pizza_size
ORDER BY order_count DESC;
```

	pizza_size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT
    t.name AS pizza_name, SUM(d.quantity) AS most_ordered
FROM
    pizza_types t
        INNER JOIN
    pizzas p ON t.pizza_type_id = p.pizza_type_id
        INNER JOIN
    order_details d ON d.pizza_id = p.pizza_id
GROUP BY pizza_name
ORDER BY most_ordered DESC
LIMIT 5;
```

pizza_name	most_ordered
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
SELECT
    t.category AS pizza_category,
    SUM(d.quantity) AS total_quantity
FROM
    pizza_types t
        INNER JOIN
    pizzas p ON t.pizza_type_id = p.pizza_type_id
        INNER JOIN
    order_details d ON d.pizza_id = p.pizza_id
GROUP BY pizza_category
ORDER BY total_quantity DESC;
```

pizza_category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

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

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    ROUND(AVG(per_day_order), 0)
FROM
    (SELECT
        o.order_date AS dates, SUM(d.quantity) AS per_day_order
    FROM
        orders o
    INNER JOIN order_details d ON o.order_id = d.order_id
    GROUP BY dates) AS order_table;
```

Result Grid | Filter Rows:

round(avg(per_day_order),0)
138

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    t.name AS pizza_name, SUM(p.price * d.quantity) AS revenue
FROM
    pizza_types t
        INNER JOIN
    pizzas p ON t.pizza_type_id = p.pizza_type_id
        INNER JOIN
    order_details d ON d.pizza_id = p.pizza_id
GROUP BY pizza_name
ORDER BY revenue DESC
LIMIT 3;
```

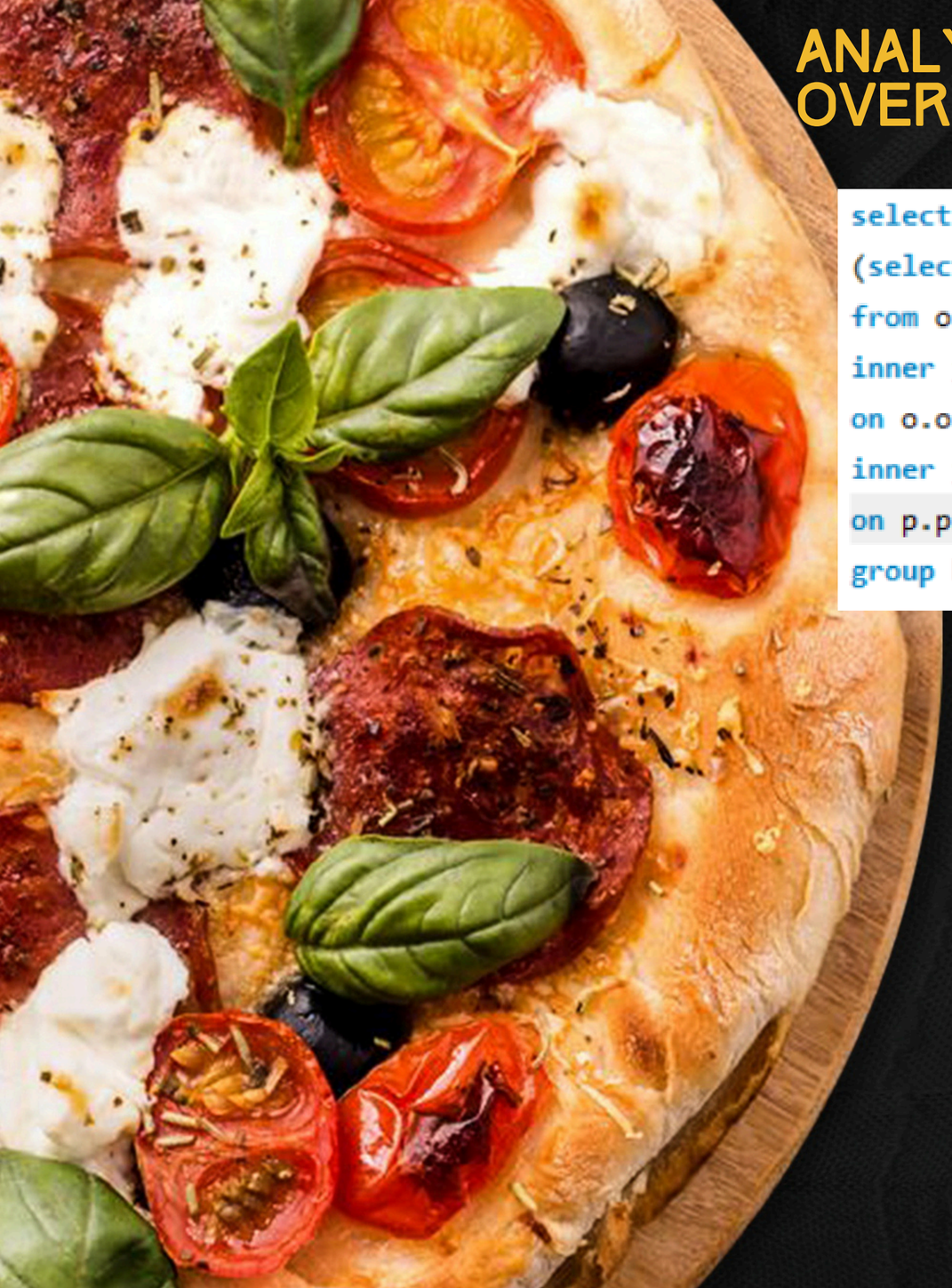
pizza_name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
select t.category as pizza_type, sum(p.price*d.quantity) /
(select sum(p.price*d.quantity) as total_sales
from order_details d
inner join pizzas p
on p.pizza_id = d.pizza_id)*100 as revenue

from pizza_types t
inner join pizzas p
on t.pizza_type_id = p.pizza_type_id
inner join order_details d
on d.pizza_id = p.pizza_id
group by pizza_type
order by revenue desc;
```

pizza_type	revenue
Classic	26.905960255669903
Supreme	25.45631126009884
Chicken	23.955137556847493
Veggie	23.682590927384783



ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

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

order_date	cum_revenue
2015-01-01	2713.8500000000004
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	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003

Result 5

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY

```sql
select category, name, revenue from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as ranks from
(select t.category, t.name, sum(d.quantity * p.price) as revenue
from pizza_types t
inner join pizzas p
on t.pizza_type_id = p.pizza_type_id
inner join order_details d
on d.pizza_id = p.pizza_id
group by t.category, t.name) as pa) as pas
where ranks <=3;
```

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

DELICIOUS PIZZA

THANK YOU!