HELLO!

My name is Shrey Srivastava and in this project I have utilized SQL queries to solve questions that are related to pizza sales.

INTRODUCTION

- Structured Query Language (SQL) is widely used for managing and analyzing data in relational databases.
- This project focuses on applying SQL queries to extract meaningful insights from a dataset.
- Through this project, I explored different SQL commands such as SELECT, WHERE, GROUP BY, ORDER BY, JOINS and aggregate functions.
- The aim was to practice real-world data operations and strengthen database handling skills.

OBJECTIVES

- To understand and apply fundamental SQL concepts in solving queries.
- To perform data retrieval, filtering, sorting, and aggregation effectively.
- To implement JOIN operations for combining data from multiple tables.
- To gain hands-on experience in query writing and optimization.
- To showcase problem-solving skills through SQL queries.

QUESTIONS

Basic:

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.

QUESTIONS

Intermediate:

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

2- Determine the distribution of orders by hour of the day.

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

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

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

QUESTIONS

Advanced:

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

2- Analyze the cumulative revenue generated over time.

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

RETRIEVE THE TOTAL NO. OF ORDERS PLACED.

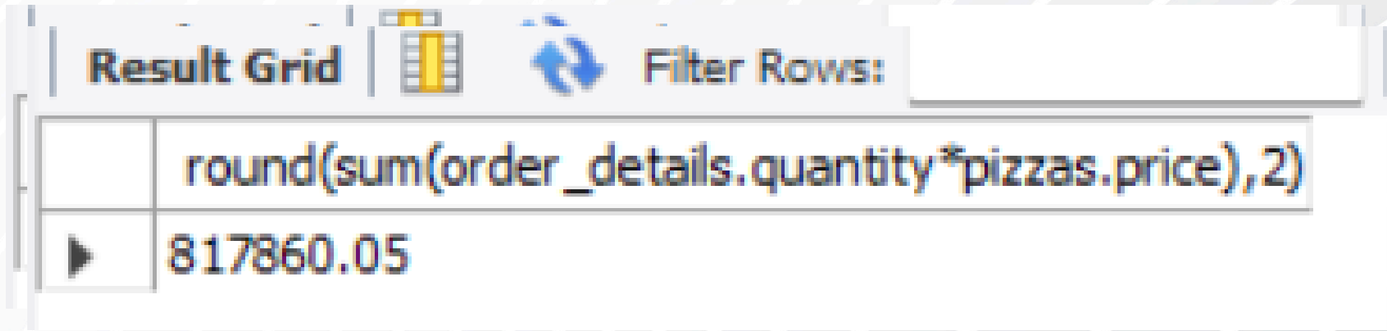
```
select count(order_id) as total_orders from orders;
```

Result Grid

	total_orders
▶	21350

RETRIEVE THE TOTAL NO. OF ORDERS PLACEDCALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
select round(sum(order_details.quantity*pizzas.price),2) from
order_details natural join pizzas;
```



IDENTIFY THE HIGHEST-PRICED PIZZA

```sql
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 Grid | Filter Rows:

	name	price
▶	The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA ORDERED.

```sql
select 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;
```

Result Grid | Filter R

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 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 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

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;
```

Result Grid | Filte

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF DAY.

```
use pizzahut;
select hour(order_time),count(order_id) from orders
group by hour(order_time);
```

Result Grid | Filter Rows:

hour(order_time)	count(order_id)
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
10	8
9	1

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

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

```
select  round(avg(quantity),0) as avg_pizza_ordered_per_day from
(select orders.order_date, sum(order_details.quantity) as quantity
from orders join order_details
on orders.order_id = order_details.order_id
group by orders.order_date) as order_quantity;
```

Result Grid | Filter Rows:

	avg_pizza_ordered_per_day
▶	138

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

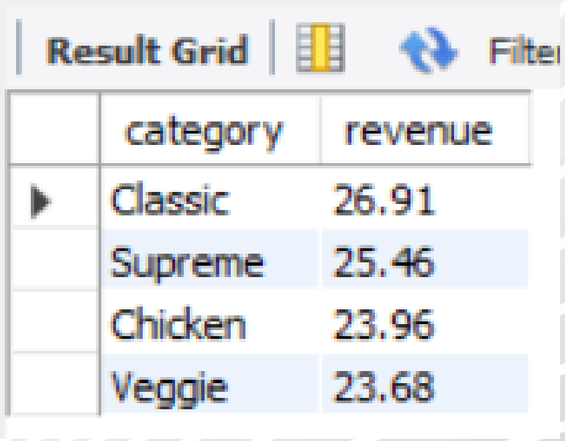
```sql
select pizza_types.name,
sum(order_details.quantity * pizzas.price) as revenue
from pizza_types join pizzas
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.name order by revenue desc limit 3;
```

Result Grid | Filter Rows:

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

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

```
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price) / (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) *100,2) 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 order by revenue desc;
```



Result Grid | Filter

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

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,
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.order_date) as sales;
```

Result Grid | Filter Rows:

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
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006

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;
```

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

CONCLUSION

This project strengthened my SQL skills and demonstrated how structured queries can transform raw data into meaningful insights.

THANK YOU