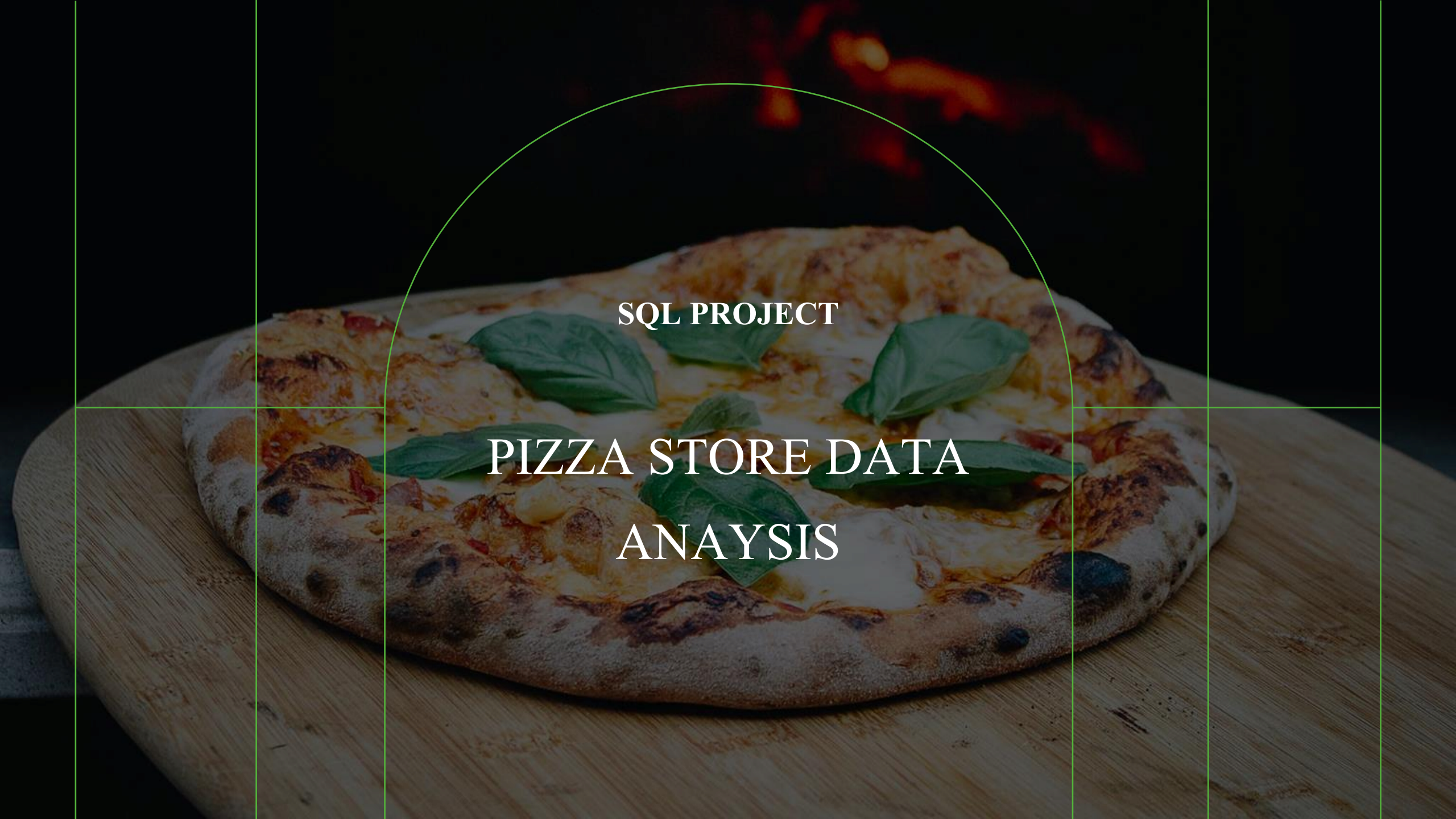


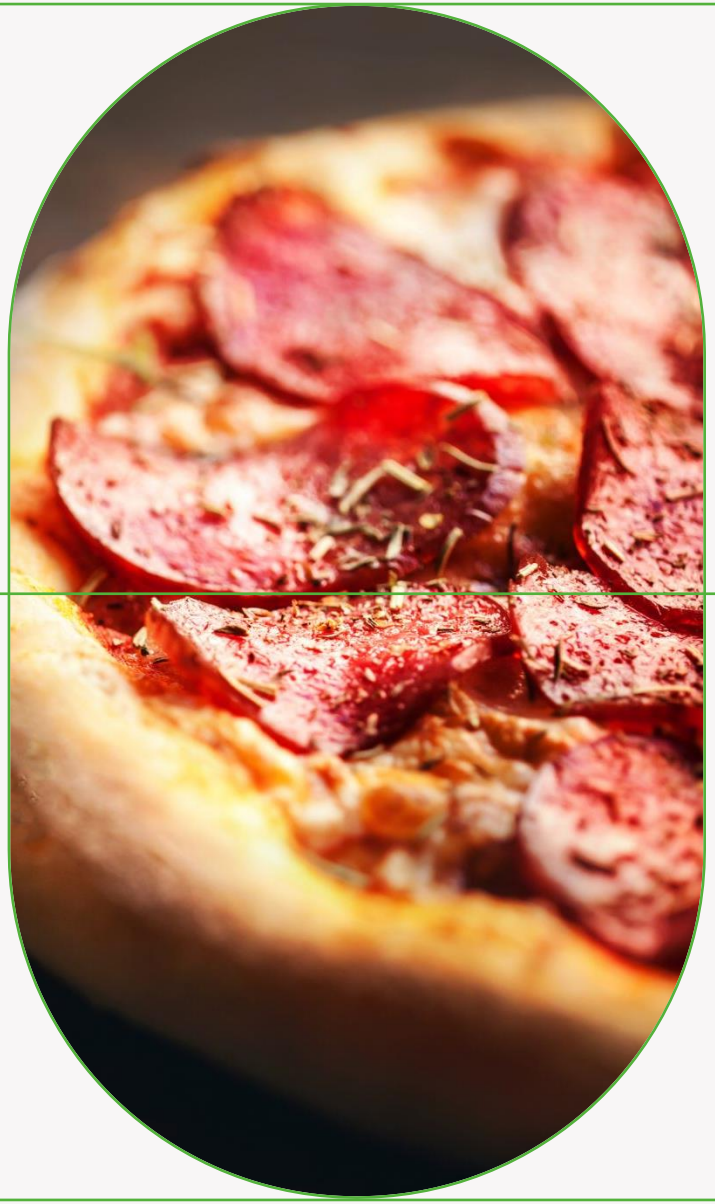
SQL PROJECT

PIZZA STORE DATA ANAYSIS

INTRODUCTION

- This project dives into a Pizza Store dataset from Kaggle, where I explored order patterns, revenue generation, and customer preferences through hands-on SQL queries. You will find data on total orders, revenue contributions, pizza size preferences, and much more.
- The data was divided in four datasets namely:
  a) pizzas- having 768 records,
- b) pizza_types- having 32 records,
- C) orders – having 21350 records, and
- d) order_details- having 48620 records.

Following slides, you will see how SQL was used for Data analysis.



SQL Queries – Basic

```sql
use pizza

select count(*) as Number_of_records from pizzas;
select count(*) as Number_of_records from order_details;
select count(*) as Number_of_records from pizza_types;
select count(*) as Number_of_records from orders;

-- Total number of pizzas sold--49574
select sum(quantity) as Total_pizzas_sold from order_details;

--Average order Value-- 17
Select round(sum(Revenue)/(count(Orders)),0) as Average_order_Value from
(Select  p.pizza_id, p.pizza_type_id, p.price, OD.order_details_id as Orders, OD.quantity, (p.price*OD.quantity) as Revenue from pizzas as p
join
order_details as OD on p.pizza_id= OD.pizza_id) as Table1;
```

Results | Messages

	Number_of_records
1	768

	Number_of_records
1	48620

	Number_of_records
1	32

	Number_of_records
1	21350

	Total_pizzas_sold
1	49574

	Average_order_Value
1	17

Basic queries continues

```sql
--Average order Value-- 17
Select round(sum(Revenue)/(count(Orders)),0) as Average_order_Value from
(Select  p.pizza_id, p.pizza_type_id, p.price, OD.order_details_id as Orders, OD.quantity, (p.price*OD.quantity) as Revenue from pizzas as p
join
order_details as OD on p.pizza_id= OD.pizza_id) as Table1;

--Average pizzas quantity sold--2
select round(sum(quantity)/count(distinct order_id),0) as Average_qty_sold from order_details;

--Retrieve the total number of orders placed.
select count( distinct order_id) as 'No of orders' from orders; --There are total of 21350 orders.

-- Week day wise pizzas quantity sold
select DATENAME(DW,o.date) as Days, sum(od.quantity) as Sale from orders as o
join
order_details as od on o.order_id=od.order_id
group by DATENAME(DW,date)
order by Sale DESC;
```

Results | Messages

	Average_order_Value
1	17

	Average_qty_sold
1	2

	No of orders
1	21350

	Days	Sale
1	Friday	8242
2	Saturday	7493
3	Thursday	7478
4	Wedne...	6946
5	Tuesday	6895
6	Monday	6485
7	Sunday	6035

SQL Queries- Intermediate

```sql
--Calculate the total revenue generated from pizza sales -- Used Inner Join on pizzas and order_details table and total revenue is 6542880.4
select ROUND(SUM(p.price*o.quantity),2) as 'Total Reneve' from
pizzas as p
Join
order_details as o
on p.pizza_id = o.pizza_id;

-- Identify the highest-priced pizza. -- The Greek Pizza is the highest priced pizza: 35.95
Select TOP 1 pt.name as Pizza_Name, p.price as Amount from
pizzas as p
Join
pizza_types as pt on p.pizza_type_id= pt.pizza_type_id
order by p.price DESC;

--Identify the most common pizza size ordered. -- L size pizza with count of 148208
select p.size, sum(o.quantity) as Quantity from
pizzas as p
Join
order_details as o on p.pizza_id= o.pizza_id
group by p.size
order by sum(o.quantity) DESC;
```

Results | Messages

	Total Reneve
1	6542880.4

	Pizza_Name	Amount
1	The Greek Pizza	35.95

	size	Quantity
1	L	151648
2	M	125080
3	S	115224
4	XL	4416
5	XXL	224

Intermediate queries continues

```sql
--List the top 5 most ordered pizza types along with their quantities. --The classic Deluxe pizza, barbecue chicken pizza, hawaiian pizza, peopperoni pizza, and thai pizza
select top 5 pt.name as Pizza_Name, sum(o.quantity) as Total_Quantity from
pizzas as p
inner join
order_details as o on p.pizza_id = o.pizza_id
inner join
pizza_types as pt on p.pizza_type_id = pt.pizza_type_id
group by pt.name
order by sum(o.quantity) DESC;


--Join the necessary tables to find the total quantity of each pizza category ordered.-- Classic, Supreme, Veggie,Chicken
select pt.category as Category, sum(o.quantity) as Total_Quantity from
pizzas as p
inner join
order_details as o on p.pizza_id = o.pizza_id
inner join
pizza_types as pt on p.pizza_type_id = pt.pizza_type_id
group by pt.category
order by sum(o.quantity) DESC;


--Determine the distribution of orders by hour of the day.-- 12,13 are extreme hours for orders received
select DATEPART(HOUR, time) as Time, count(order_id) as Total_orders from orders
group by DATEPART(HOUR, time)
order by count(order_id) DESC;
```

Results | Messages

	Pizza_Name	Total_Quantity
1	The Classic Deluxe Pizza	19624
2	The Barbecue Chicken Pizza	19456
3	The Hawaiian Pizza	19376
4	The Pepperoni Pizza	19344
5	The Thai Chicken Pizza	18968

	Category	Total_Quantity
1	Classic	119104
2	Supreme	95896
3	Veggie	93192
4	Chicken	88400

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

Intermediate queries continues

```sql
--Join relevant tables to find the category-wise distribution of pizzas.--Chicken(6), classic(8), Supreme(9), Veggie(9)
Select category, count(*) as Quantity from pizza_types
group by category
order by count(*);


--Group the orders by date and calculate the average number of orders per day.--59
Select AVG(No_of_orders) as AverageCount from
(Select CONVERT(DATE, date) as Date, count(order_id) as No_of_orders
from orders
Group by Date) as OrderCount;


--Group the orders by date and calculate the average number of pizzas ordered per day.--138
Select Round(AVG(quantity),1) as Average_Pizzas_Count from (Select convert(Date, O.date) as Date,Sum(OD.quantity) as Quantity from
order_details as OD
Inner join
orders as O on OD.order_id= O.order_id
group by convert(Date, O.date)) as Pizza_per_day;
```

Results | Messages

	category	Quantity
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

	AverageCount
1	59

	Average_Pizzas_Count
1	138.5

SQL Queries- Advanced

```sql
--Determine the top 3 most ordered pizza types based on revenue.--The Thai chicken, Barbecue chicken, California
Select top 3 pt.name as Pizza_name, sum(p.price*OD.quantity) as Amount
from pizzas as p
Inner join
order_details as OD
on p.pizza_id=  OD.pizza_id
join
pizza_types as pt
on p.pizza_type_id=pt.pizza_type_id
group by pt.name
order by Amount DESC;

--Calculate the percentage contribution of each pizza type to total revenue.-- Classic(27),Supreme(25),Veggie(24),Chicken(24),
WITH RenvueTotalCat as(
select pt.category as Name, Round(SUM(p.price*od.quantity),0) as Total_Revenue
from pizzas as p
join order_details as od
on p.pizza_id=od.pizza_id
join pizza_types as pt on p.pizza_type_id=pt.pizza_type_id
group by pt.category
),

Revenue as(
Select sum(Total_Revenue) as TotalR
from RenvueTotalCat)

select Name, Round(Total_Revenue/TotalR*100,0) as PercentageContribution  from RenvueTotalCat, Revenue
Order by PercentageContribution DESC ;
```

Results | Messages

	Pizza_name	Amount
1	The Thai Chicken Pizza	347474
2	The Barbecue Chicken Pizza	342144
3	The California Chicken Pizza	331276

	Name	PercentageContribution
1	Classic	27
2	Supreme	25
3	Veggie	24
4	Chicken	24

Advanced queries continues

```sql
--Analyze the cumulative revenue generated over time.
Select Date, sum(amount) over (order by Date) as cumulative_rev
from
(select Convert(Date,o.date) as Date, SUM(p.price*quantity) as amount from pizzas as p
join
order_details as od
on p.pizza_id = od.pizza_id
join
orders as o on od.order_id=o.order_id
Group by Convert(Date,o.date)) as RevenueByDate;


--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 rank
from
(Select pt.category as Category, pt.name as Name, Round(SUM(price*od.quantity),0) as Revenue
from pizzas as p
join
order_details as od
on p.pizza_id=od.pizza_id
join
pizza_types as pt on p.pizza_type_id= pt.pizza_type_id
Group by pt.category, name) as a) as b where rank <=3;
```

Results | Messages

	Date	cumulative_rev
1	2015-01-01	21710.8
2	2015-01-02	43566
3	2015-01-03	64865.2
4	2015-01-04	78908.8
5	2015-01-05	95436.4
6	2015-01-06	114868
7	2015-01-07	132485.6
8	2015-01-08	155192.4

	Name	Revenue
1	The Thai Chicken Pizza	347474
2	The Barbecue Chicken Pizza	342144
3	The California Chicken Pizza	331276
4	The Classic Deluxe Pizza	305444
5	The Hawaiian Pizza	258186
6	The Pepperoni Pizza	241294
7	The Spicy Italian Pizza	278650
8	The Italian Supreme Pizza	267814
9	The Sicilian Pizza	247524
10	The Four Cheese Pizza	258126
11	The Mexicana Pizza	214246
12	The Five Cheese Pizza	208532

Project Highlights

Data Integration and ETL: Designed and implemented a smooth ETL process to bring together data from multiple sources. This involved extracting a multi-table dataset from Kaggle, transforming it by normalizing and validating the data, and loading it into MySQL using SSMS. The result was a well-organized database ready for analysis.

SQL Query Development: Developed and ran SQL queries to uncover key metrics such as the total number of orders and revenue. This hands-on experience allowed for effective data retrieval and manipulation, answering important business questions with precision.

Advanced Data Analysis Techniques: Leveraged advanced SQL techniques like table joins, subqueries, and grouping to explore patterns in order trends, revenue distribution, and pizza sales across various categories. This deep dive into the data provided valuable insights into complex relationships.

Complex Data Aggregation: Tackled complex aggregation tasks to compute cumulative revenue over time and assess each pizza type's contribution to total sales. These analyses sharpened analytical skills and provided a clear picture of overall performance.

Practical Application of SQL: Utilized SQL to address a range of analytical questions, including identifying the most frequently ordered pizzas and calculating average daily orders. This demonstrated the ability to translate business needs into actionable data insights.

Data Cleaning and Transformation: Focused on cleaning and transforming raw data, resolving inconsistencies, and optimizing datasets for efficient querying. This step was vital for ensuring the accuracy and reliability of the analysis outcomes.

Thank you

By Shama Ravani