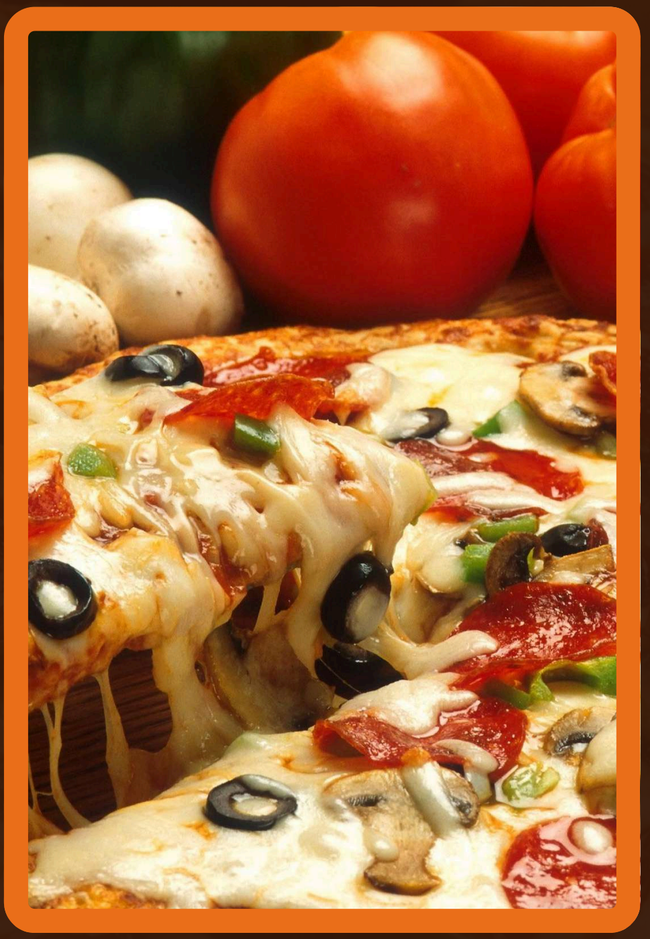
Where Every Slice is a Taste of Perfection

WELCOME TO RESTO PIZZA -SQL PORTFOLIO PROJECT

By- Riddhi Garudkar

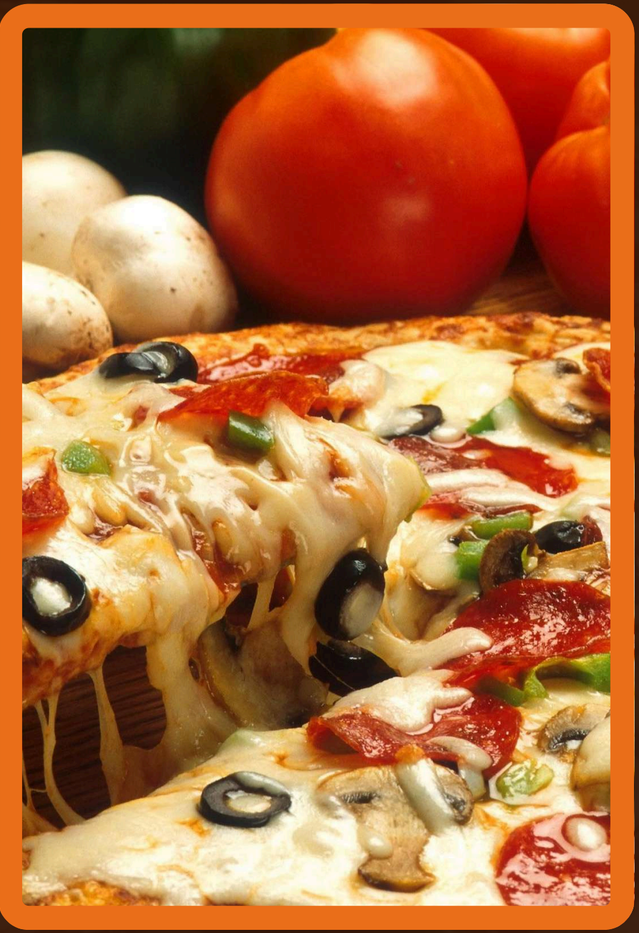
INTRODUCTION

This project demonstrates comprehensive data analysis on pizza sales data. The project is designed to extract actionable insights from a pizza store's database. By addressing questions ranging from basic metrics to advanced revenue contributions, the analysis provides a full spectrum of business intelligence to inform decision-making.



BASIC METRICS

- The foundational analysis focuses on essential business KPIs to offer a quick overview of sales performance.
- This includes calculating the total number of orders placed, the revenue generated, identifying popular items and sizes, and the most profitable offerings. These metrics provide key indicators for understanding the store's sales dynamics.
- Objective: To summarize the store's overall sales performance.



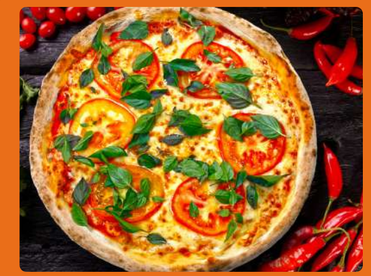
Q1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

output

Result Grid

	total_orders
▶	21350



Q2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

```
SELECT
    round(sum(order_details.quantity * pizzas.price),2) as total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

output

	total_revenue
▶	817860.05

Q3. IDENTIFY THE HIGHEST-PRICED PIZZA.

```
SELECT
    pizzas.price, pizza_types.name
FROM
    pizzas
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
order by pizzas.price desc limit 1;
```

output

	price	name
▶	35.95	The Greek Pizza

Q4. IDENTIFY THE MOST COMMON PIZZA SIZE 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 size
ORDER BY order_count DESC
LIMIT 1;
```

output

	size	order_count
▶	L	18526

Q5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```
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;
```

output

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

INTERMEDIATE ANALYSIS

- This section involves a deeper exploration of patterns in the data. By leveraging table joins and aggregations.
- This analysis provides valuable information for demand planning, resource allocation, and optimizing sales strategies based on customer behavior.
- Objective: To explore patterns and distributions in pizza orders by category and time.

Q6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
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;
```

output

	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) AS hours, COUNT(order_id) AS orders_count
FROM
    orders
GROUP BY hours;
```

output

	hours	orders_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
	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;
```

output

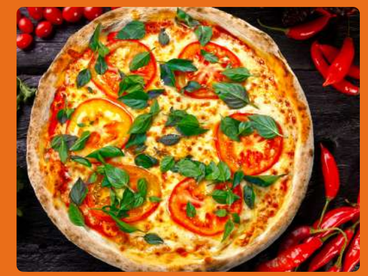
	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

Q9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

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

output

	average
▶	138

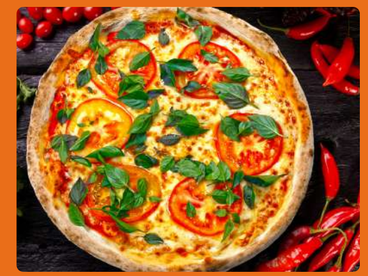


Q10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
SELECT
    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.name
ORDER BY revenue DESC
LIMIT 3;
```

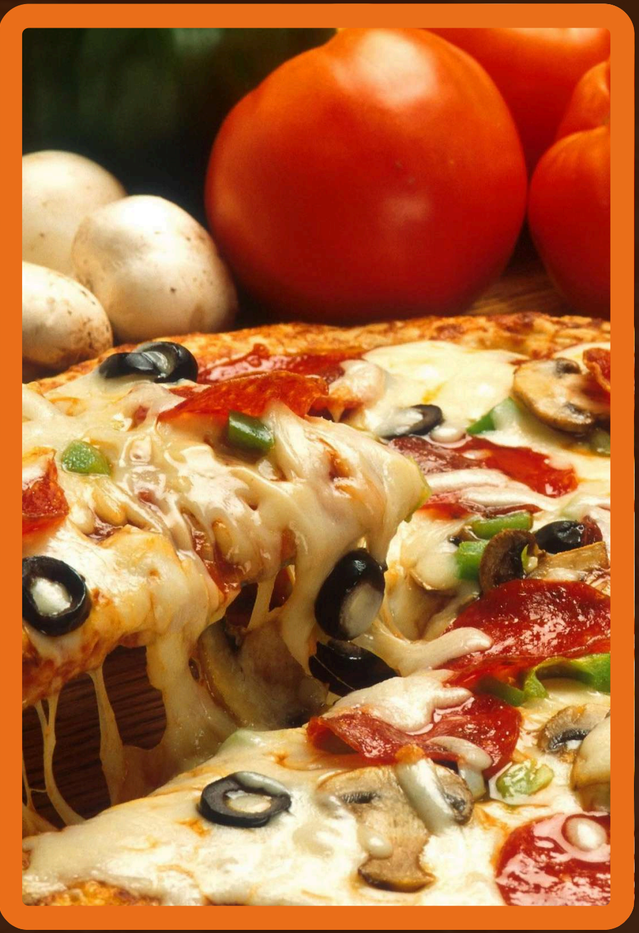
output

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



ADVANCED ANALYSIS

- This advanced section applies revenue-based analysis to determine the contribution of each pizza type to the store's total earnings.
- It further explores cumulative revenue growth over time and analyzes top-performing pizzas by category. These insights are critical for revenue optimization and strategic decision-making.
- Objective: To drive strategic decision-making by identifying high-revenue contributors and assessing revenue trends.

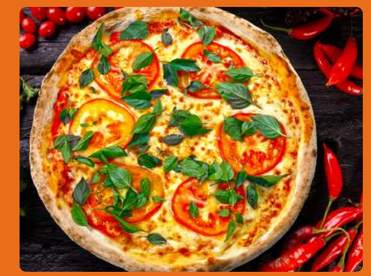


Q11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```sql
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;
```

output

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68



Q12. 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,
round(sum(order_details.quantity*pizzas.price),2) 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;
```

output

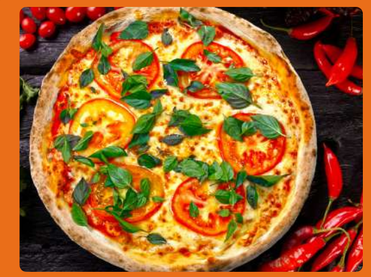
order_date	cum_revenue
2015-01-01	2713.85
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-12-24	807553.75
2015-12-26	809196.8
2015-12-27	810615.8
2015-12-28	812253
2015-12-29	813606.25
2015-12-30	814944.05
2015-12-31	817860.05

Q13. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPE 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 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;
```

output

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	32265.70
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5



CONCLUSION

This project showcases a structured approach to analyzing business data, moving from foundational metrics to advanced revenue-based insights. The insights gained here could be used to optimize menu offerings, streamline operations, and maximize profitability, making it an invaluable tool for data-driven decision-making in a restaurant business.

THANK YOU

available on github

https://github.com/garudkarriddhi?tab=repositories