ABOUT US

PIZZA STORE

MY SQL PROJECT

NEXT >

ABOUT US

I am an aspiring data analyst with a strong foundation in SQL, data modeling, and analytical problem-solving. I enjoy uncovering insights from data and presenting them in a meaningful way to drive informed decisions. My skills include data cleaning, aggregation, and visualization using tools like Power BI and Excel. I am currently expanding my technical stack with the MERN stack and MySQL to complement my analytical capabilities with backend and web development skills.

THIS PROJECT SIMULATES A REAL-WORLD PIZZA ORDERING SYSTEM USING SQL TO ANALYZE SALES AND CUSTOMER ORDERS. IT FEATURES A WELL-STRUCTURED RELATIONAL DATABASE WITH TABLES SUCH AS ORDERS, ORDER_DETAILS, PIZZAS, AND PIZZA_TYPES. USING ADVANCED SQL QUERIES, INCLUDING JOINS, AGGREGATIONS, AND WINDOW FUNCTIONS LIKE RANK() AND SUM() OVER, I EXTRACTED KEY BUSINESS INSIGHTS SUCH AS TOP-SELLING PIZZA TYPES BY CATEGORY AND DAILY CUMULATIVE REVENUE. THE PROJECT DEMONSTRATES STRONG SKILLS IN DATA MODELING, PROBLEM-SOLVING, AND TURNING RAW TRANSACTIONAL DATA INTO ACTIONABLE ANALYTICS.

AJAY SAHNAI

1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.


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

OUTPUT >

Result Grid

	total_order
▶	21350

< BACK

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.


```sql
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 >

Result Grid

	total_revenue
▶	817860.05

< BACK

3. IDENTIFY THE HIGHEST-PRICED PIZZA.


```sql
SELECT
    pizza_types.name,
    order_details.quantity
FROM
    order_details
    JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
    JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
ORDER BY
    pizzas.price DESC
LIMIT 1;
```

OUTPUT >

Result Grid | Filter Rows:

	name	quantity
▶	The Greek Pizza	1

< BACK

4. 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 pizzas.size
ORDER BY order_count DESC;
```

OUTPUT >

Result Grid | Filter

size	order_count
L	1127
M	934
S	869
XL	36
XXL	2

36

< BACK

5. 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;
```

OUTPUT >

Result Grid | Filter Rows:

name	quantity
The Hawaiian Pizza	168
The Thai Chicken Pizza	159
The California Chicken Pizza	139
The Classic Deluxe Pizza	137
The Pepperoni Pizza	136

< BACK

6. 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 >

Result Grid | Filter F

	category	quantity
▶	Classic	925
	Veggie	733
	Supreme	715
	Chicken	651

< BACK

7. 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(order_time);
```

OUTPUT >

Result Grid | Filte

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
10	8
9	1

< BACK

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


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

OUTPUT >

Result Grid | Filter Rows

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

< BACK

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


```
SELECT
    AVG(quantity)
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;
```

OUTPUT >

Result Grid

	AVG(quantity)
▶	138.4749

< BACK

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


```sql
SELECT
    pizza_types.name,
    SUM(pizzas.price * order_details.quantity) 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;
```

OUTPUT >

Result Grid | Filter Rows:

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

< BACK

11. 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)
        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

Result Grid | Filter Ro

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

< BACK

12. 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;
```

OUTPUT

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	19399.050000000003
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3
2015-01-14	32358.7
2015-01-15	34343.5
2015-01-16	36937.65
2015-01-17	39001.75
2015-01-18	40978.6
2015-01-19	43365.75

< BACK

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


```sql
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;
```

OUTPUT >

esult 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	32265.700000000554
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

< BACK

THANK YOU!

WWW.REALLYGREATSITE.COM