SQL PROJECT

ON PIZZA SALES

INTRODUCTION

In this project, we leveraged SQL queries to perform an in-depth analysis of a pizza sales dataset, extracting valuable business intelligence and actionable insights. By executing complex queries, we identified key sales trends, peak revenue-generating time slots, and the highest-performing pizza varieties. We also conducted customer preference analysis, order frequency evaluation, and revenue segmentation to assess the impact of pricing strategies on overall profitability. Additionally, we explored sales distribution across different categories, enabling data-driven decision-making for operational efficiency and business growth. This analytical approach provided a comprehensive understanding of consumer behavior, helping optimize sales strategies for maximum profitability.

ABOUT DATASET

The pizza sales dataset is a rich repository of transactional data that provides deep insights into customer purchasing behavior, sales performance, and revenue trends. It enables data-driven analysis of key business metrics such as order frequency, product popularity, peak sales periods, and revenue segmentation. By leveraging this dataset, businesses can optimize pricing strategies, enhance inventory management, and implement targeted marketing campaigns to boost profitability. Additionally, it facilitates predictive analytics, helping businesses anticipate demand fluctuations and improve operational efficiency, ultimately leading to enhanced customer satisfaction and strategic decision-making.

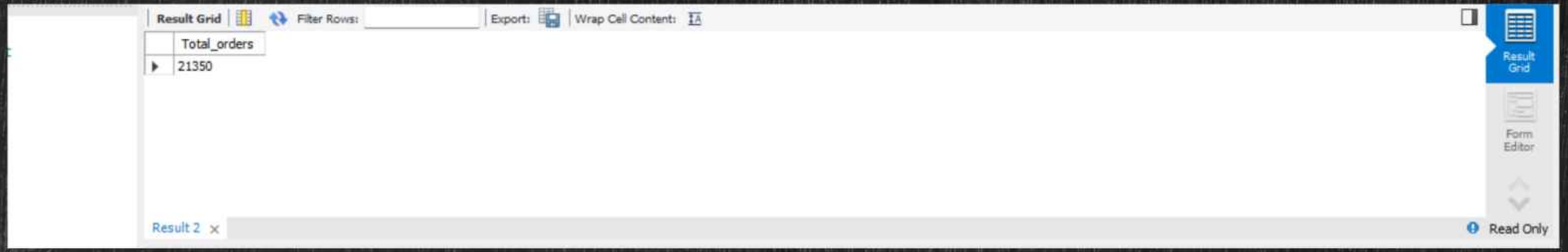
INDEX

1. RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
-- retrieve the total no of orders placed

SELECT
    COUNT(order_id) AS Total_orders
FROM
    pizzahut.orders;
```

CODE



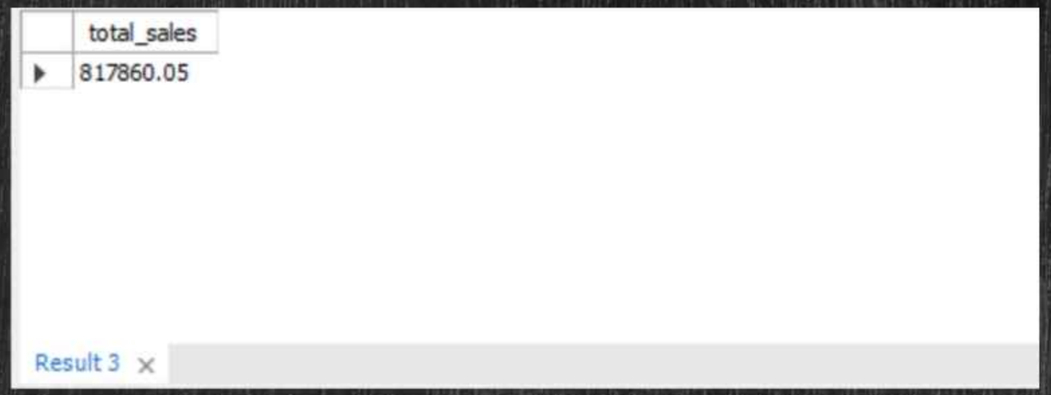
RESULT

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
-- calculate the total revenue generated from pizza sales
SELECT
    ROUND(SUM(pizzas.price * order_details.quantity),
            2) AS total_sales
FROM
    pizzahut.order_details
        JOIN
    pizzahut.pizzas ON pizzas.pizza_id = order_details.pizza_id;

```

CODE



RESULT

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
-- Identify the highest priced pizza

SELECT
    pizza_types.name, pizzas.price
FROM
    pizzahut.pizzas
        JOIN
    pizzahut.pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

CODE

Result Grid | Filter Rows: | Export:

name	price
The Greek Pizza	35.95

RESULT

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
-- Identify the most common pizza size ordered.
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS no_of_pizzas_ordered
FROM
    pizzahut.pizzas
        JOIN
    pizzahut.order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size;
```

CODE

Result Grid | Filter Rows:

size	no_of_pizzas_ordered
M	15385
L	18526
S	14137
XL	544
XXL	28

RESULT

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
    pizzahut.pizza_types
        JOIN
    pizzahut.pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    pizzahut.order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
```

CODE

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

RESULT

6. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(order_time) AS order_hour,
    COUNT(order_id) AS order_count
FROM
    pizzahut.orders
GROUP BY HOUR(order_time);
```

CODE

order_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

RESULT

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

```sql
-- Join relevant tables to find the category-wise distribution of pizzas
SELECT
    category, COUNT(name)
FROM
    pizzahut.pizza_types
GROUP BY category
```

CODE

Result Grid | Filter Rows:

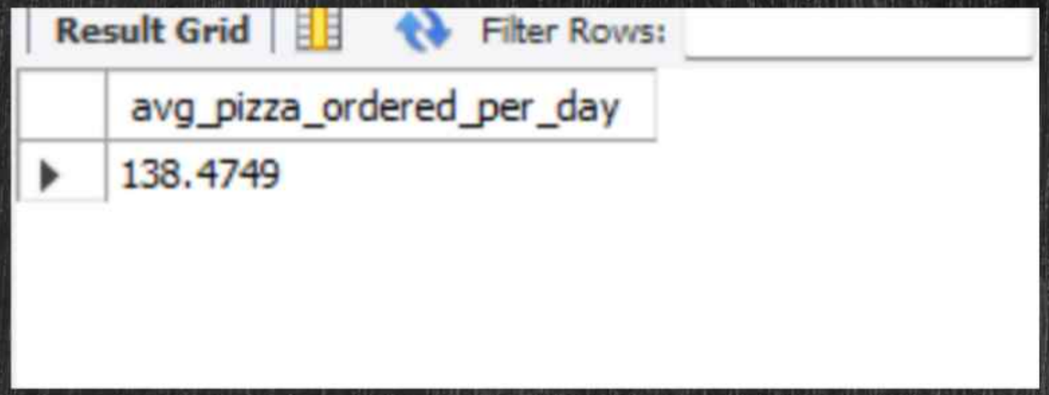
category	count(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

RESULT

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

```sql
-- Group the orders by date and calculate the average number of pizzas ordered
 -- per day.
SELECT
    AVG(quantity) AS avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        pizzahut.orders
    JOIN pizzahut.order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

CODE



RESULT

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

```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizzahut.pizza_types
        JOIN
    pizzahut.pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    pizzahut.order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

CODE

Result Grid | Filter Rows:

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

RESULT

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

```sql
-- 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(pizzas.price * order_details.quantity),
            2) AS total_sales
FROM
    pizzahut.order_details
        JOIN
    pizzahut.pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2)
    as revenue
FROM
    pizzahut.pizza_types
        JOIN
    pizzahut.pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    pizzahut.order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

CODE

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

RESULT

11. ANALYSE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
-- Analyze the cumulative revenue generated over time.
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 pizzahut.order_details join pizzahut.pizzas
on order_details.pizza_id = pizzas.pizza_id
join pizzahut.orders
on orders.order_id = order_details.order_id
group by orders.order_date) as sales
```

CODE

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

Result 2

RESULT

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

```sql
-- 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 pizzahut.pizza_types join pizzahut.pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join pizzahut.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;
```

CODE

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

Result 3

RESULT


THANK YOU