Neeraj Dangwal

SQL Project on

PIZZA SALES

Introduction

This project analyzes the Kaggle Pizza Sales dataset to showcase SQL skills in deriving business insights. Using joins, subqueries, aggregates, and window functions, raw sales data was transformed into meaningful findings on customer behavior, product performance, and revenue trends

Key Insight

- Top Sellers: Classic & Supreme pizzas had the highest order volume.
- Revenue: Large pizzas drove the most revenue, followed by medium.
- Customer Behavior: Peak orders occurred between 6–9 PM.
- Popular Picks: Pepperoni & BBQ Chicken led in revenue.
- Trends: Revenue grew steadily over time

How it Works

- **Store Data – Pizza sales records are organized in relational SQL tables.**
- **Run Queries – SQL queries with joins, filters, and aggregations analyze the data.**
- **Get Insights – Extract reports on top-selling pizzas, sales trends, and revenue performance.**

Problem

Problem 1

Retrieve the total number of orders placed.

Problem 2

Calculate the total revenue generated from pizza sales.

Problem 3

Identify the highest-priced pizza.

Problem 4

Identify the most common pizza size ordered.

Problem 5

List the top 5 most ordered pizza types along with their quantities.

Problem 6

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

Problem 7

Determine the distribution of orders by hour of the day.

Problem 8

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

Problem 9

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

Problem 10

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

Problem 11

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

Problem 12

Analyze the cumulative revenue generated over time.

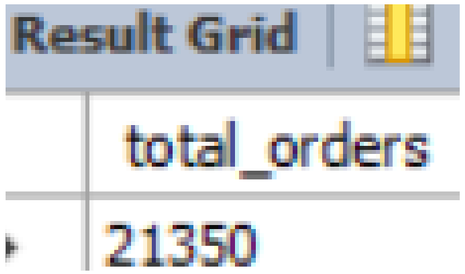
Problem 13

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

Solutions

Solution 1

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

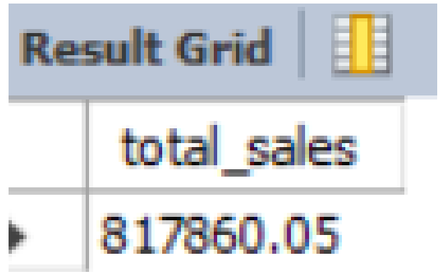


Result Grid

total_orders
21350

Solution 2

```sql
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

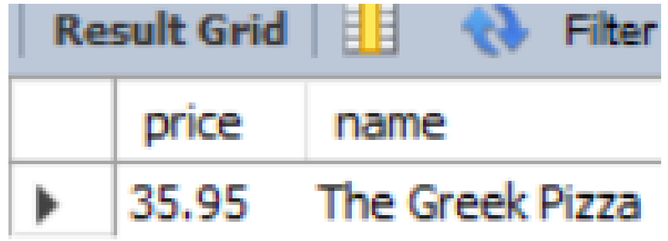


Result Grid

total_sales
817860.05

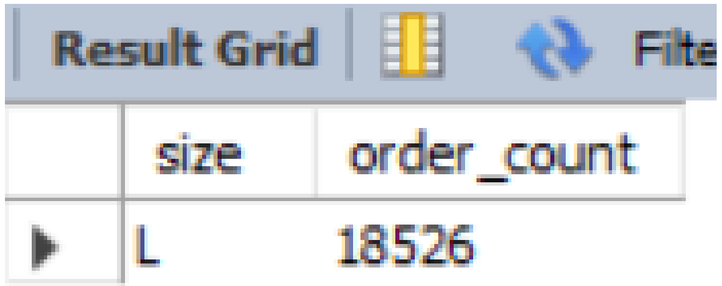
Solution 3

```sql
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;
```



Solution 4

```sql
SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC
LIMIT 1;
```



Solution 5

```sql
SELECT
    SUM(orders_details.quantity) AS quantities, pizza_types.name
FROM
    pizzas
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantities DESC
LIMIT 5;
```

Result Grid | Filter Rows:

quantities	name
2453	The Classic Deluxe Pizza
2432	The Barbecue Chicken Pizza
2422	The Hawaiian Pizza
2418	The Pepperoni Pizza
2371	The Thai Chicken Pizza

Solution 6

```sql
SELECT
    SUM(orders_details.quantity) AS quantities,
    pizza_types.category
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.category
ORDER BY quantities desc;
```

Result Grid | Filter

quantities	category
14888	Classic
11987	Supreme
11649	Veggie
11050	Chicken

Solution 7

```
SELECT
    HOUR(order_time) AS Hour, COUNT(order_id) AS Order_count
FROM
    orders
GROUP BY Hour;
```

Hour	Order_cour
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336

Solution 8

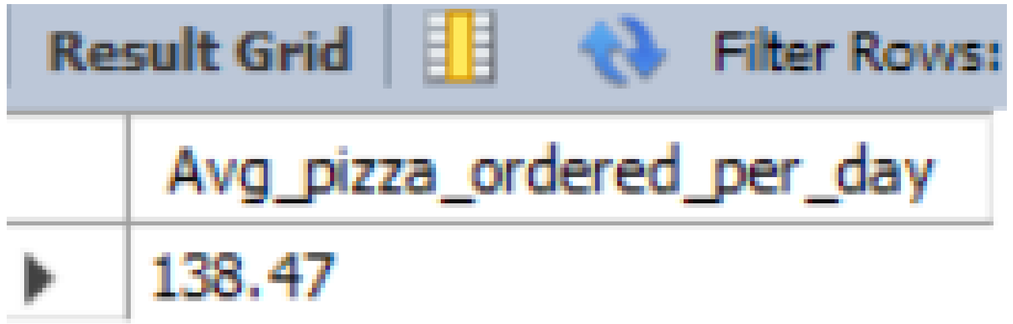
```
SELECT
    category,COUNT(name) AS Quantity
FROM
    pizza_types
GROUP BY category;
```

Result Grid Filt

category	Quantity
Chicken	6
Classic	8
Supreme	9
Veggie	9

Solution 9

```sql
SELECT
    ROUND(AVG(quantity), 2) AS Avg_pizza_ordered_per_day
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders_details.order_id = orders.order_id
    GROUP BY orders.order_date) AS order_quantity;
```



Solution 10

```sql
SELECT
    pizza_types.name,
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue
LIMIT 3;
```

Result Grid | Filter Rows:

name	revenue
The Brie Carre Pizza	11588.5
The Green Garden Pizza	13955.75
The Spinach Supreme Pizza	15277.75

Solution 11

```sql
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(orders_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    orders_details
                        JOIN
                    pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100,
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

Result Grid | Filt

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Solution 12

```sql
SELECT order_date,
       SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM (
    SELECT orders.order_date,
           SUM(orders_details.quantity * pizzas.price) AS revenue
    FROM orders
    JOIN orders_details ON orders_details.order_id = orders.order_id
    JOIN pizzas ON pizzas.pizza_id = orders_details.pizza_id
    GROUP BY orders.order_date
) AS sales;
```

order_date	cum_revenue
2015-01-01	2713.850000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6

Solution 13

```sql
SELECT name, category, 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(orders_details.quantity * pizzas.price) AS revenue
        FROM pizza_types
        JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN orders_details ON orders_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: | Export:

name	category	revenue
The Thai Chicken Pizza	Chicken	43434.25
The Barbecue Chicken Pizza	Chicken	42768
The California Chicken Pizza	Chicken	41409.5
The Classic Deluxe Pizza	Classic	38180.5

Conclusion

This project offered hands-on experience in analyzing structured sales data with SQL. It highlights skills in query optimization, data modeling, and analytical problem-solving, while uncovering key business insights such as sales trends, customer preferences, and revenue performance.

Thank You,

Prepared by Neeraj Dangwal