Project of Data Analysis of Pizza Orders

PIZZA-SALES SQL QUERIES

EXPLORE NOW

By- UDIT SINGH

INTRODUCTION

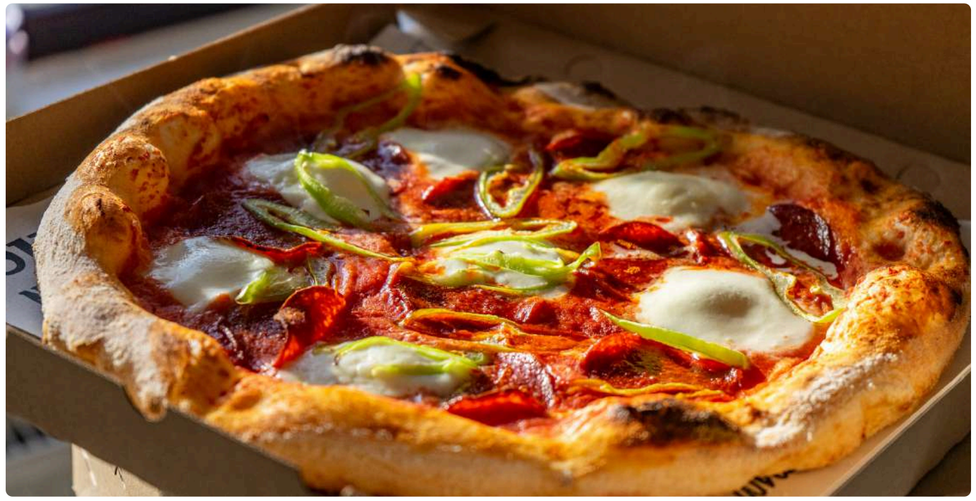
Briefly describe what the project is about.

The project is a data analysis initiative focused on a pizza-ordering dataset. It aims to uncover insights about customer preferences, ordering patterns, and revenue trends by leveraging SQL queries. The dataset comprises four tables: orders, order_details, pizzas, and pizza_types, each capturing different aspects of the pizza business.

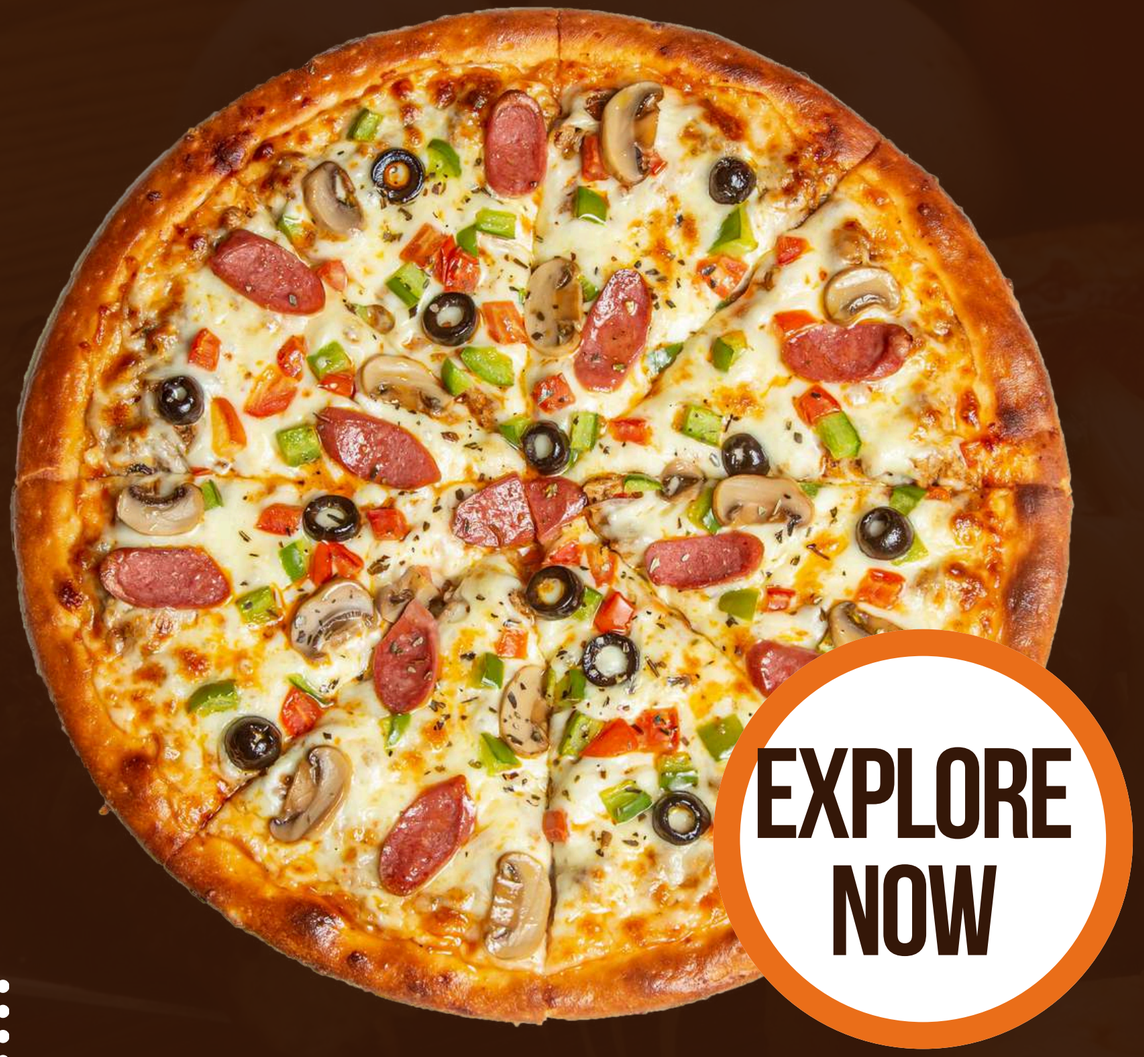
DATASET OVERVIEW

Tables and Columns

- Provide a simple diagram or bullet points of the dataset schema:
  - Orders Table: order_id, date, time.
  - Order Details Table: order_details_id, order_id, pizza_id, quantity.
  - Pizzas Table: pizza_id, pizza_type_id, size, price.
  - Pizza Types Table: pizza_type_id, name, category, ingredients.



Note: I have kept the limit of this dataset up to row 1000 only.



TOOLS AND TECHNOLOGIES

- **SQL for querying data.**
- **Database software: My SQL Workbench**

Why SQL?

- Highlight SQL's strengths in handling relational data and performing complex queries.

BASIC LEVEL QUESTION'S

1) Retrieve the total number of orders placed.

```
select count(order_id) as total_orders from orders;
```

Output :

esult Grid

total_orders
1005

BASIC LEVEL QUESTION'S

2) Calculate the total revenue generated from pizza sales.

```
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
```

Output

Result Grid

	total_sales
▶	93811.9

BASIC LEVEL QUESTION'S

3) Identify the highest-priced pizza.

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

Output

Result Grid | Filter Rows

	name	price
▶	The Greek Pizza	35.95

BASIC LEVEL QUESTION'S

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
LIMIT 1;
```

Output

Result Grid | Filte

size	order_count
L	2139

BASIC LEVEL QUESTION'S

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

name	quantity
The Pepperoni Pizza	313
The Barbecue Chicken Pizza	286
The California Chicken Pizza	277
The Classic Deluxe Pizza	254
The Hawaiian Pizza	253

INTERMEDIATE LEVEL QUESTION'S

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

```sql
SELECT
    pt.category AS pizza_category,
    SUM(od.quantity) AS total_quantity
FROM
    order_details od
        JOIN
    pizzas p ON od.pizza_id = p.pizza_id
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY total_quantity DESC;
```

Output

pizza_category	total_quantity
Classic	1689
Supreme	1417
Veggie	1362
Chicken	1229

INTERMEDIATE LEVEL QUESTION'S

2) Determine the distribution of orders by hour of the day.

```sql
SELECT
    HOUR(TIME(o.time)) AS order_hour,
    COUNT(o.order_id) AS total_orders
FROM
    orders o
GROUP BY order_hour
ORDER BY order_hour;
```

Output

order_hour	total_orders
11	50
12	121
13	111
14	90
15	77
16	88
17	117
18	107
19	96
20	73
21	45

INTERMEDIATE LEVEL QUESTION'S

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

```sql
SELECT
    pt.category AS pizza_category,
    COUNT(p.pizza_id) AS total_pizzas
FROM
    pizzas p
        JOIN
    pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY total_pizzas DESC;
```

Output

Result Grid | Filter Rows:

pizza_category	total_pizzas
Veggie	27
Classic	26
Supreme	25
Chicken	18

INTERMEDIATE LEVEL QUESTION'S

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

```sql
SELECT
    o.date AS order_date,
    AVG(od.quantity) AS avg_pizzas_per_day
FROM
    orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY
    o.date
ORDER BY
    order_date;
```

Output

order_date	avg_pizzas_per_day
2015-01-01 00:00:00	1.0062
2015-01-02 00:00:00	1.0313
2015-01-03 00:00:00	1.0260
2015-01-04 00:00:00	1.0000
2015-01-05 00:00:00	1.0331
2015-01-06 00:00:00	1.0208
2015-01-07 00:00:00	1.0376
2015-01-08 00:00:00	1.0117
2015-01-09 00:00:00	1.0325
2015-01-10 00:00:00	1.0069
2015-01-11 00:00:00	1.0175
2015-01-12 00:00:00	1.0085
2015-01-13 00:00:00	1.0256
2015-01-14 00:00:00	1.0417
2015-01-15 00:00:00	1.0000
2015-01-16 00:00:00	1.0194
2015-01-17 00:00:00	1.0299

INTERMEDIATE LEVEL QUESTION'S

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

```sql
SELECT
    pt.name AS pizza_type,
    SUM(od.quantity * p.price) AS total_revenue
FROM
    order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY
    pt.name
ORDER BY
    total_revenue DESC
LIMIT 3;
```

Output

Result Grid | Filter Rows:

pizza_type	total_revenue
The Barbecue Chicken Pizza	5090.5
The California Chicken Pizza	4767.75
The Thai Chicken Pizza	4597

ADVANCED LEVEL QUESTION'S

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

```sql
WITH revenue_per_pizza AS (
    SELECT
        pt.name AS pizza_type,
        SUM(od.quantity * p.price) AS total_revenue
    FROM
        order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY
        pt.name
), total_revenue AS (
    SELECT
        SUM(total_revenue) AS grand_total
    FROM
        revenue_per_pizza
)
SELECT
    rpp.pizza_type,
    rpp.total_revenue,
    (rpp.total_revenue / tr.grand_total) * 100 AS percentage_contribution
FROM
    revenue_per_pizza rpp
CROSS JOIN
    total_revenue tr
ORDER BY
    percentage_contribution DESC;
```

Output

pizza_type	total_revenue	percentage_contribution
The Barbecue Chicken Pizza	5090.5	5.426283872301916
The California Chicken Pizza	4767.75	5.082244363454958
The Thai Chicken Pizza	4597	4.9002312073415
The Italian Supreme Pizza	4039.75	4.306223410889238
The Spicy Italian Pizza	3944.25	4.204423959007333
The Pepperoni Pizza	3929	4.1881680255916365
The Classic Deluxe Pizza	3915	4.173244545734604
The Sicilian Pizza	3894.25	4.1511258166607865
The Four Cheese Pizza	3658.3499999999926	3.899665181069772
The Southwest Chicken Pizza	3565.25	3.80042404002051
The Hawaiian Pizza	3318.5	3.5373977075403027
The Five Cheese Pizza	3311.5	3.529935967611786
The Greek Pizza	3227.3999999999996	3.440288492184894
The Vegetables + Vegetable...	3154.5	3.3625798006436294
The Italian Capocollo Pizza	3121.5	3.3274030266949084
The Mexicana Pizza	3018.75	3.2178753441727546
The Napolitana Pizza	3012	3.2106800949559715
The Pepper Salami Pizza	3009.75	3.2082816785503767
The Prosciutto and Arugula ...	2984.5	3.1813661166653704
The Spinach and Feta Pizza	2848.25	3.0361286787710307
The Big Meat Pizza	2472	2.635060157613267
The Italian Vegetables Pizza	2175	2.31846919207478
The Chicken Alfredo Pizza	1983	2.1138043254640406
The Pepperoni, Mushroom, ...	1877	2.0008122636893617
The Spinach Supreme Pizza	1813.75	1.9333901136209803
The Soppressata Pizza	1763	1.8792924991392352
The Chicken Pesto Pizza	1734.25	1.8486460672899712

ADVANCED LEVEL QUESTION'S

2) Analyze the cumulative revenue generated over time.

```sql
SELECT
    o.date AS order_date,
    SUM(od.quantity * p.price) AS daily_revenue,
    SUM(SUM(od.quantity * p.price)) OVER (ORDER BY o.date) AS cumulative_revenue
FROM
    orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY
    o.date
ORDER BY
    order_date;
```

Output

order_date	daily_revenue	cumulative_revenue
2015-01-01 00:00:00	2713.8500000000004	2713.8500000000004
2015-01-02 00:00:00	2731.8999999999996	5445.75
2015-01-03 00:00:00	2662.4	8108.15
2015-01-04 00:00:00	1755.4500000000003	9863.6
2015-01-05 00:00:00	2065.95	11929.55
2015-01-06 00:00:00	2428.95	14358.5
2015-01-07 00:00:00	2202.2000000000003	16560.7
2015-01-08 00:00:00	2838.3499999999995	19399.05
2015-01-09 00:00:00	2127.3500000000004	21526.4
2015-01-10 00:00:00	2463.95	23990.350000000002
2015-01-11 00:00:00	1872.3000000000002	25862.65
2015-01-12 00:00:00	1919.0500000000002	27781.7
2015-01-13 00:00:00	2049.6000000000004	29831.300000000003
2015-01-14 00:00:00	2527.3999999999996	32358.700000000004
2015-01-15 00:00:00	1984.8000000000002	34343.50000000001
2015-01-16 00:00:00	2594.15	36937.65000000001
2015-01-17 00:00:00	1141.65	38079.30000000001

ADVANCED LEVEL QUESTION'S

3) 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 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

Result Grid | Filter Rows:

name	revenue
The Barbecue Chicken Pizza	5090.5
The California Chicken Pizza	4767.75
The Thai Chicken Pizza	4597
The Pepperoni Pizza	3929
The Classic Deluxe Pizza	3915
The Hawaiian Pizza	3318.5
The Italian Supreme Pizza	4039.75
The Spicy Italian Pizza	3944.25
The Sicilian Pizza	3894.25
The Four Cheese Pizza	3658.3499999999926
The Five Cheese Pizza	3311.5
The Vegetables + Vegetabl...	3154.5

IMPORTANT POINT

The project focuses on analyzing a subset of 1,000 rows from a pizza-ordering dataset to extract actionable insights. By working on this smaller dataset, the analysis efficiently addresses key business questions, such as identifying popular pizza categories, peak ordering times, and revenue trends. This approach demonstrates scalable data analysis techniques applicable to larger datasets.

Project of Data Analysis of Pizza Orders

THANK YOU FOR ATTENTION

Github repo link: https://github.com/Uditsingh1234/Data-Analysis-of-Pizza-Orders-.git

By - Udit Singh