PIZZA SALES ANALYSIS USING MYSQL

ABOUT THE PROJECT

This project is a data analysis of pizza sales using MySQL, where I explored sales data sourced from Kaggle. The objective was to gain insights into customer preferences, revenue trends, and ordering patterns by writing SQL queries to answer business-driven questions.

OBJECTIVES

The main goals of this analysis were to:

- Understand order volume and revenue trends
- Identify the most popular pizzas and sizes
- Explore category-wise ordering patterns
- Analyze time-based ordering behavior
- Determine the top revenue-generating pizzas

STEPS TAKEN

DATA COLLECTION

- DATASET OBTAINED FROM KAGGLE PIZZA SALES DATASET
- IMPORTED DATA INTO MYSQL FOR QUERYING AND ANALYSIS.

BASIC ANALYSIS

- RETRIEVED TOTAL NUMBER OF ORDERS PLACED.
- CALCULATED TOTAL REVENUE GENERATED.
- IDENTIFIED THE HIGHEST-PRICED PIZZA.
- FOUND THE MOST COMMON PIZZA SIZE ORDERED.
- LISTED TOP 5 MOST ORDERED PIZZAS BY QUANTITY

INTERMEDIATE ANALYSIS

- JOINED TABLES TO FIND TOTAL QUANTITY ORDERED PER CATEGORY.
- ANALYZED ORDER DISTRIBUTION BY HOUR OF THE DAY
- EXPLORED CATEGORY-WISE PIZZA DISTRIBUTION.
- GROUPED ORDERS BY DATE TO CALCULATE AVERAGE PIZZAS SOLD PER DAY.
- DETERMINED TOP 3 PIZZAS BASED ON REVENUE.

ADVANCED ANALYSIS

- CALCULATED EACH PIZZA TYPE'S PERCENTAGE CONTRIBUTION TO REVENUE.
- ANALYZED CUMULATIVE REVENUE TRENDS OVER TIME.
- FOUND THE TOP 3 REVENUE-GENERATING PIZZAS FOR EACH CATEGORY.

Q-1) RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED

QUERY-:

```
Limit to 1000 rows
1   -- Retrieve the total number of orders placed
2
3•  select * from orders;
4•  SELECT
5       COUNT(order_id) AS total_orders
6   FROM
7       orders;
```

OUTPUT-:

Result Grid | Filter Rows:

total_orders
21350

Q-2) CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

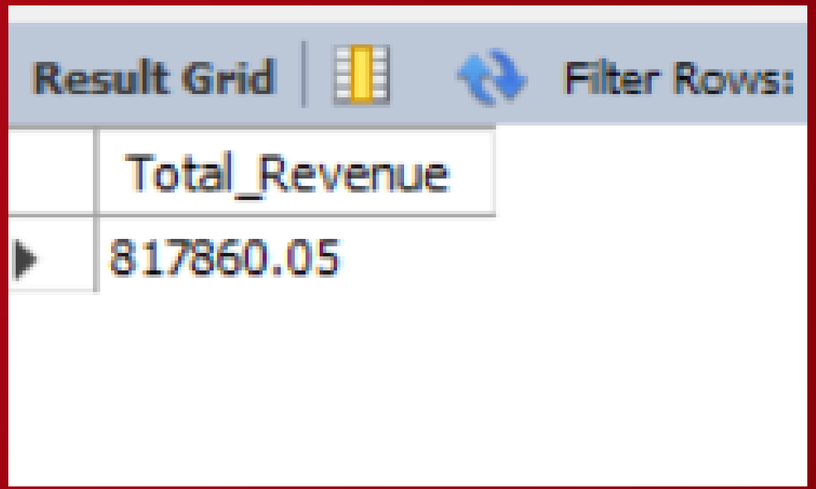
QUERY-:

```sql
-- Q2) Calculate the total revenue generated from pizza sales.

use pizza_sales;
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS Total_Revenue
FROM
    orders_details
        JOIN
    pizzas ON orders_details.pizza_id = pizzas.pizza_id;

-- Round lgaya taaki 2 decimal places tk aajaye values
```

OUTPUT-:



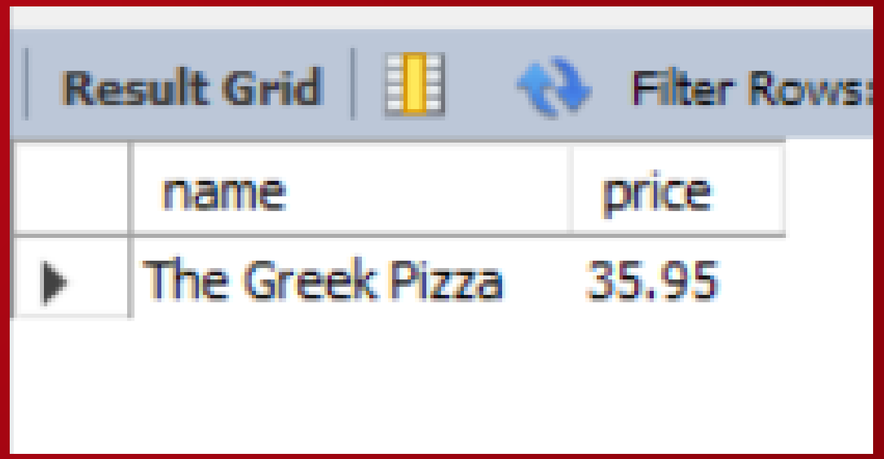
Q-3) IDENTIFY THE HIGHEST-PRICED PIZZA.

QUERY-:

```sql
-- Q3) Identify the highest-priced pizza.

use pizza_sales;
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

Q-4) IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

QUERY-:

```
Limit to 1000 rows
1   -- Q4) Identify the most common pizza size ordered.
2
3 • SELECT
4       pizzas.size, COUNT(orders_details.order_detais_id) AS count
5   FROM
6       pizzas
7           JOIN
8       orders_details ON pizzas.pizza_id = orders_details.pizza_id
9   GROUP BY pizzas.size
10  ORDER BY count DESC
11  LIMIT 1;
```

OUTPUT-:

Result Grid

size	count
L	18526

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

QUERY-:

```sql
-- Q5) List the top 5 most ordered pizza types along with their quantities.

SELECT
    pizza_types.name, SUM(orders_details.quantity) AS Quantity
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 Quantity DESC
LIMIT 5
```

OUTPUT-:

Result Grid | Filter Rows:

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

Q-6) JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

QUERY-:

```sql
-- Q6) Join the necessary tables to find the total
--     quantity of each pizza category ordered.

SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS Quantity
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
ORDER BY Quantity DESC;
```

OUTPUT-:

Result Grid | Filter Rows:

category	Quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Q-7) DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

QUERY-:

```
Limit to 1000 rows
1   -- Q7) Determine the distribution of orders by hour of the day.
2 • SELECT
3       HOUR(order_time) AS Hours, COUNT(order_id)
4   FROM
5       orders
6   GROUP BY Hours;
7
```

OUTPUT-:

Result Grid | Filter Rows:

Hours	COUNT(order_id)
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642

Q-8) Join relevant tables to find the category-wise distribution of pizzas.

QUERY-:

```sql
-- Q8) Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name) AS TotalCount
FROM
    pizza_types
GROUP BY category;
```

OUTPUT-:

Result Grid | Filter Ro

category	TotalCount
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

QUERY-:

```sql
-- Q9) Group the orders by date and calculate the
--     average number of pizzas ordered per day.

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

OUTPUT-:

Result Grid | Filter

avg_pizzas_per_day
138

Q-10) Determine the top 3 most ordered pizza types based on revenue.

QUERY-:

```sql
-- Q10) Determine the top 3 most ordered pizza types based on revenue.

SELECT
    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.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

Q-11) CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

QUERY-:

```sql
-- Q11) Calculate the percentage contribution
--       of each pizza type to total revenue

select pizza_types.category,
round(sum(orders_details.quantity * pizzas.price) / (SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),2) AS Total_Revenue
FROM orders_details
JOIN pizzas ON orders_details.pizza_id = pizzas.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 orders_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

Q-12) ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME

QUERY-:

```sql
-- Q12) 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(orders_details.quantity * pizzas.price) as Revenue
from orders_details join pizzas
on orders_details.pizza_id = pizzas.pizza_id
join orders
on orders.order_id = orders_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	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002

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

QUERY-:

```sql
-- Q13) 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((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;
```

OUTPUT-:

Result 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

OUTCOME

THROUGH THIS ANALYSIS, I UNCOVERED VALUABLE INSIGHTS SUCH AS THE MOST PROFITABLE PIZZAS, ORDERING TIME PATTERNS, AND REVENUE CONTRIBUTION BY CATEGORY, WHICH CAN HELP BUSINESSES IN INVENTORY PLANNING, MENU OPTIMIZATION, AND SALES STRATEGIES.

7042798486

https://github.com/xpiyush14/Pizza-Sales-Analysis-using-MySQL