PIZZA SALES ANALYSIS BY USING MYSQL

Pizza Sales Analysis By Using MySQL

Project Description

The Pizza Sales Analysis project is a comprehensive data-driven study that leverages MySQL to explore, analyze, and derive insights from a pizza sales dataset. The primary goal of this project is to identify key business metrics and trends, enabling data-driven decision-making for operational efficiency and customer satisfaction. The dataset comprises multiple tables, including orders, pizza details, and pizza categories. By addressing key business questions and performing advanced data analysis, this project provides actionable insights that can aid decision-making for improving operations and maximizing profitability.

Objectives

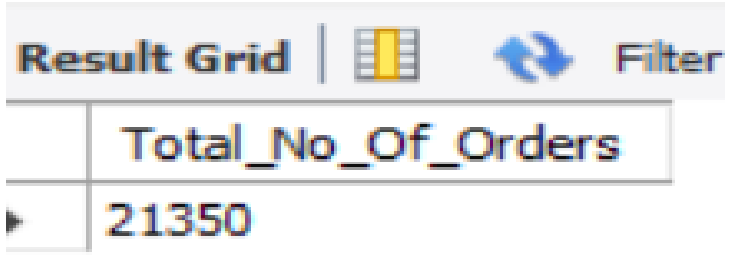
- **Basic Analysis**:
  - Retrieve the total number of orders placed.
  - Calculate the total revenue generated from pizza sales.
  - Identify the highest-priced pizza on the menu.
  - Find the most commonly ordered pizza size.
  - List the top 5 most ordered pizza types along with their quantities.
- **Intermediate Analysis**:
  - Join relevant tables to find the total quantity of pizzas ordered in each category.
  - Determine the distribution of orders by hour of the day to understand peak demand times.
  - Analyze the category-wise distribution of pizzas sold.
  - Group orders by date and calculate the average number of pizzas ordered per day.
  - Identify the top 3 most ordered pizza types based on revenue.
- **Advanced Analysis**:
  - Calculate the percentage contribution of each pizza type to total revenue.
  - Analyze cumulative revenue trends over time.
  - Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Tools and Techniques

- **Database Management**: MySQL for data storage and manipulation.
- **SQL Queries**: Complex joins, aggregations, and window functions.
- **Data Analysis**: Time-series and category-based grouping to uncover trends.

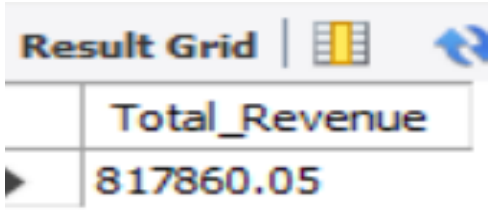
Retrieve the total number of orders placed

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



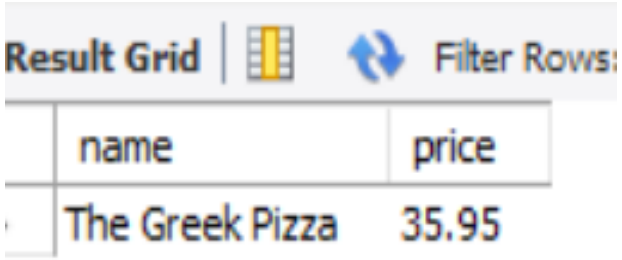
Calculate the total revenue generated from pizza sales

```
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;
```



Identify the highest-priced pizza

```sql
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;
```



name	price
The Greek Pizza	35.95

Identify the most common pizza size ordered

```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;
```

Result Grid | Filt

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

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;
```

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

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;
```

Result Grid | Filt

category	Quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

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

Result Grid | Filter

Hour	Noof_Orders
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

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

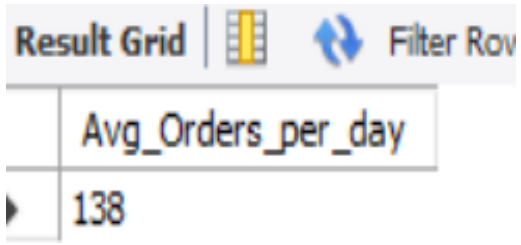
```
SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

Result Grid | Filter Rows

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

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

```
SELECT
    ROUND(AVG(sales), 0) as Avg_Orders_per_day
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS sales
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS daily_orders;
```



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

```sql
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;
```

Result Grid | Filter Rows:

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

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

```sql
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;
```

Result Grid | Filter R

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Analyze the cumulative revenue generated over time

```sql
SELECT
    order_date,
    SUM(revenue) OVER (ORDER BY order_date) AS cumulative_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 daily_revenue;
```

Result Grid | Filter Rows:

order_date	cumulative_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
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006

Result Grid | Filter Rows:

order_date	cumulative_revenue
2015-12-08	771820.5
2015-12-09	774392.05
2015-12-10	776377.65
2015-12-11	779011.65
2015-12-12	780971.8
2015-12-13	783216.9500000001
2015-12-14	785389.55
2015-12-15	787777
2015-12-16	790011.8
2015-12-17	791892.55
2015-12-18	794778.8500000001
2015-12-19	797083.05
2015-12-20	799187.9500000001
2015-12-21	801288.65
2015-12-22	803171.6
2015-12-23	805415.9
2015-12-24	807553.75
2015-12-26	809196.8
2015-12-27	810615.8
2015-12-28	812253
2015-12-29	813606.25
2015-12-30	814944.05
2015-12-31	817860.05

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

```sql
SELECT
    category,
    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 cate_revenue
) AS rank_wise
WHERE rn <= 3;
```

Result Grid | Filter Rows: | Export:

category	name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.70000000065
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

Key Insights Derived

- The total number of orders and revenue provided an understanding of business performance.
- Peak order timings were identified, allowing optimization of staffing and inventory management.
- The most commonly ordered pizza sizes and types indicated customer preferences, helping tailor promotions and deals
- Category-wise and revenue-based analysis revealed top-performing pizzas and their contribution to overall sales.
- Trends in cumulative revenue helped assess the growth trajectory of the business over time.

Business Recommendations

- Focus marketing efforts on the top-performing pizza types and sizes.
- Introduce deals or bundles for popular categories to increase sales further.
- Adjust staffing and inventory based on peak demand hours and high-performing days.
- Leverage insights into underperforming pizza types or categories to improve or replace them.
- Develop targeted campaigns for increasing the revenue contribution of less popular categories.