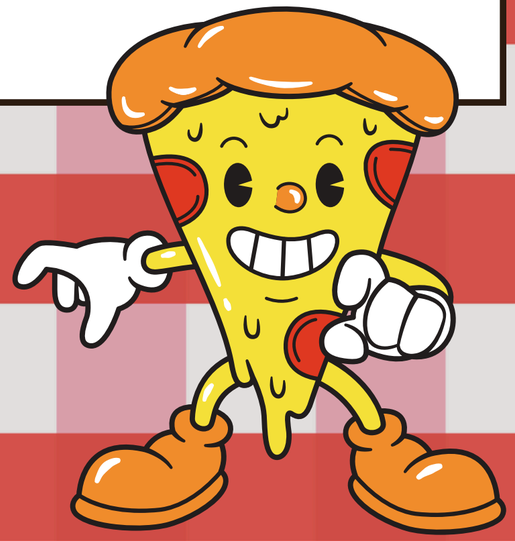
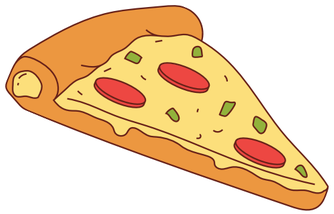
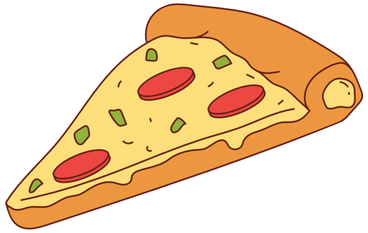
PIZZA SALES REPORT SYSTEM



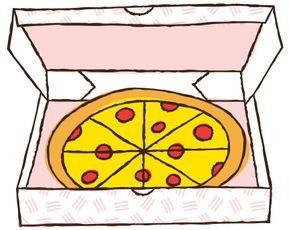


Welcome

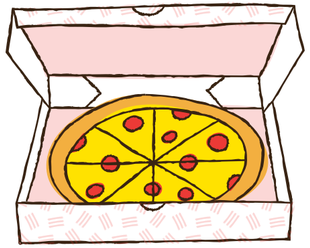


!

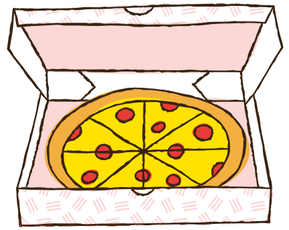
I am Santosh Tirmare. In this project utilize MySQL querys to solve questions to related to pizza sales.



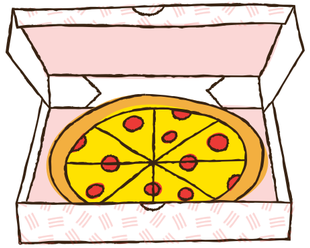
MySQL



MySQL is a popular open-source relational database management system (RDBMS) known for its reliability, scalability, and ease of use. It uses Structured Query Language (SQL) for accessing and managing database information. MySQL is widely used in web applications, especially with PHP, and supports various storage engines and data types. It offers robust security features, including user authentication and data encryption. MySQL's community and enterprise editions cater to different needs, from small projects to large-scale enterprise applications.

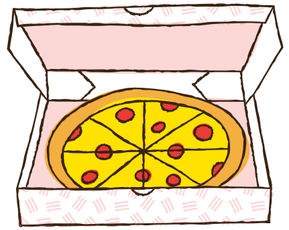


Pizza Sales Report System

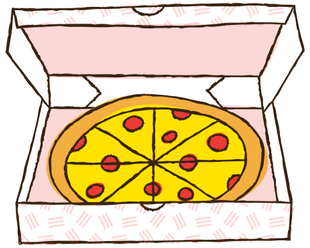


Overview:

Developed a Pizza Sales Report System to efficiently manage and analyze sales data using MySQL.



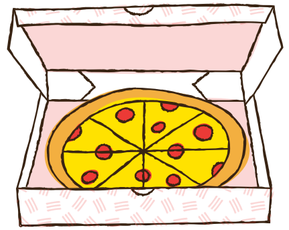
Pizza Sales Report System



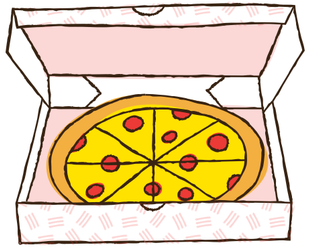
Key Components:

- **Database Design:**
  - Designed a normalized schema to store pizzas, orders, customers, and sales transactions.
  - Ensured data integrity and minimized redundancy.

- **Data Insertion:**
  - Implemented robust scripts to import and validate sales data.
  - Ensured consistency and accuracy in data entries.

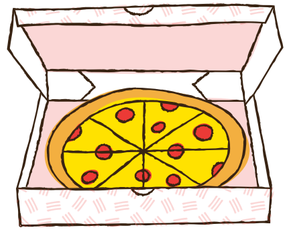


Pizza Sales Report System

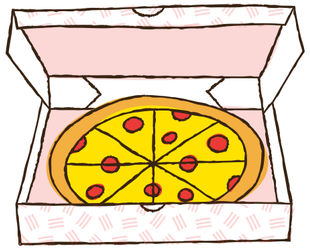


Key Components:

- **Queries and Reporting::**
  - **Created complex SQL queries for generating various sales reports.**
  - **Reports include daily, weekly, and monthly sales summaries, top-selling pizzas, and customer purchase patterns.**
  -
- **Data Analysis:**
  - **Utilized aggregate functions and joins to derive meaningful insights**
  - **Analyzed sales trends to inform business decisions and strategies..**



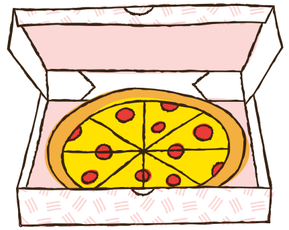
Pizza Sales Report System



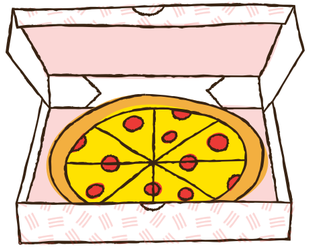
Key Components:

- **Security:**
  - Applied indexing and query optimization to enhance database performance.
  - Ensured quick retrieval and processing of large data sets.

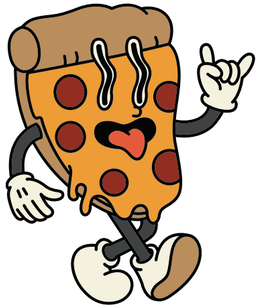
- **Performance Optimization:**
  - Applied indexing and query optimization to enhance database performance.
  - Ensured quick retrieval and processing of large data sets.



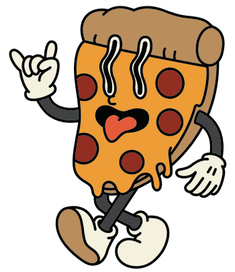
Questions



1. **Retrieve the total number of orders placed.**
2. **Calculate the total revenue generated from pizza sales.**
3. **Identify the highest-priced pizza.**
4. **Identify the most common pizza size ordered.**
5. **List the top 5 most ordered pizza types along with their quantities.**
6. **Join the necessary tables to find the total quantity of each pizza category ordered.**
7. **Determine the distribution of orders by hour of the day.**
8. **Join relevant tables to find the category-wise distribution of pizzas.**
9. **Group the orders by date and calculate the average number of pizzas ordered per day.**
10. **Determine the top 3 most ordered pizza types based on revenue.**



Retrieve the total number of orders placed.



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

Result Grid | Filter

	total_orders
▶	21350

Calculate the total revenue generated from pizza sales.

```
3 •  SELECT
4        ROUND(SUM(orders_details.quantity_id * pizzas.price),
5                2) AS total_sales
6    FROM
7        orders_details
8            JOIN
9        pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid

	total_sales
▶	3646.3

Identify the highest-priced pizza.

```
3 •  SELECT
4        pizza_types.name, pizzas.price
5    FROM
6        pizza_types
7            JOIN
8        pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9    ORDER BY pizzas.price DESC
0    LIMIT 1;
```

Result Grid | Filter Rows:

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.

```
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;
```

Result Grid | Filter

	size	order_count
▶	L	97

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

```
SELECT
    pizza_types.name,
    SUM(orders_details.quantity_id) 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 Italian Supreme Pizza	17
The Thai Chicken Pizza	16
The Barbecue Chicken Pizza	14
The Classic Deluxe Pizza	13
The Mexicana Pizza	11

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

```sql
SELECT
    pizza_types.category,
    SUM(orders_details.quantity_id) 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

category	quantity
Classic	63
Veggie	52
Supreme	52
Chicken	50

Determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

Result Grid | Filter Rov

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

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

```
select category, count(name) from pizza_types
group by category;
```

Result Grid | Filter Rov

	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.

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

Result Grid | Filter Ro

ROUND(AVG(quantity), 0)
109

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

```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity_id * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas 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 revenue DESC
LIMIT 3;
```

Result Grid | Filter Rows:

name	revenue
The Italian Supreme Pizza	302.25
The Thai Chicken Pizza	296
The Barbecue Chicken Pizza	262.5

Thank You...

PIZZA PARTY ?