PIZZA SALES ANALYSIS

By Using SQL Queries

Presented By : Shital Yadav

Introduction and Project Overview

- **Objective :** Analyze and manage pizza sales data to gain insights about customer behavior, popular pizzas, order trends, and more.
- **Tools Used :** MySQL for database management and query execution

- **Project Overview :**
  - This SQL project revolves around a database schema designed to manage and analyse data for a pizza store. The database consists of four primary tables given as following :
    - Order_details, pizzas, orders and pizza_types
  - Each table plays a crucial role in storing different facts of the business operations, from individual orders to the types of pizzas offered. Below is a detailed description of each table and its columns:

Data Source and Tables

Here's a concise description of each table in the dataset:

1. **order_details**
   - Stores details about each item (pizza) in an order, such as the specific pizza (pizza_id), the order it belongs to (order_id), and the quantity ordered.
   - Links to the "orders" table via order_id and to the pizza_types table via pizza_id.

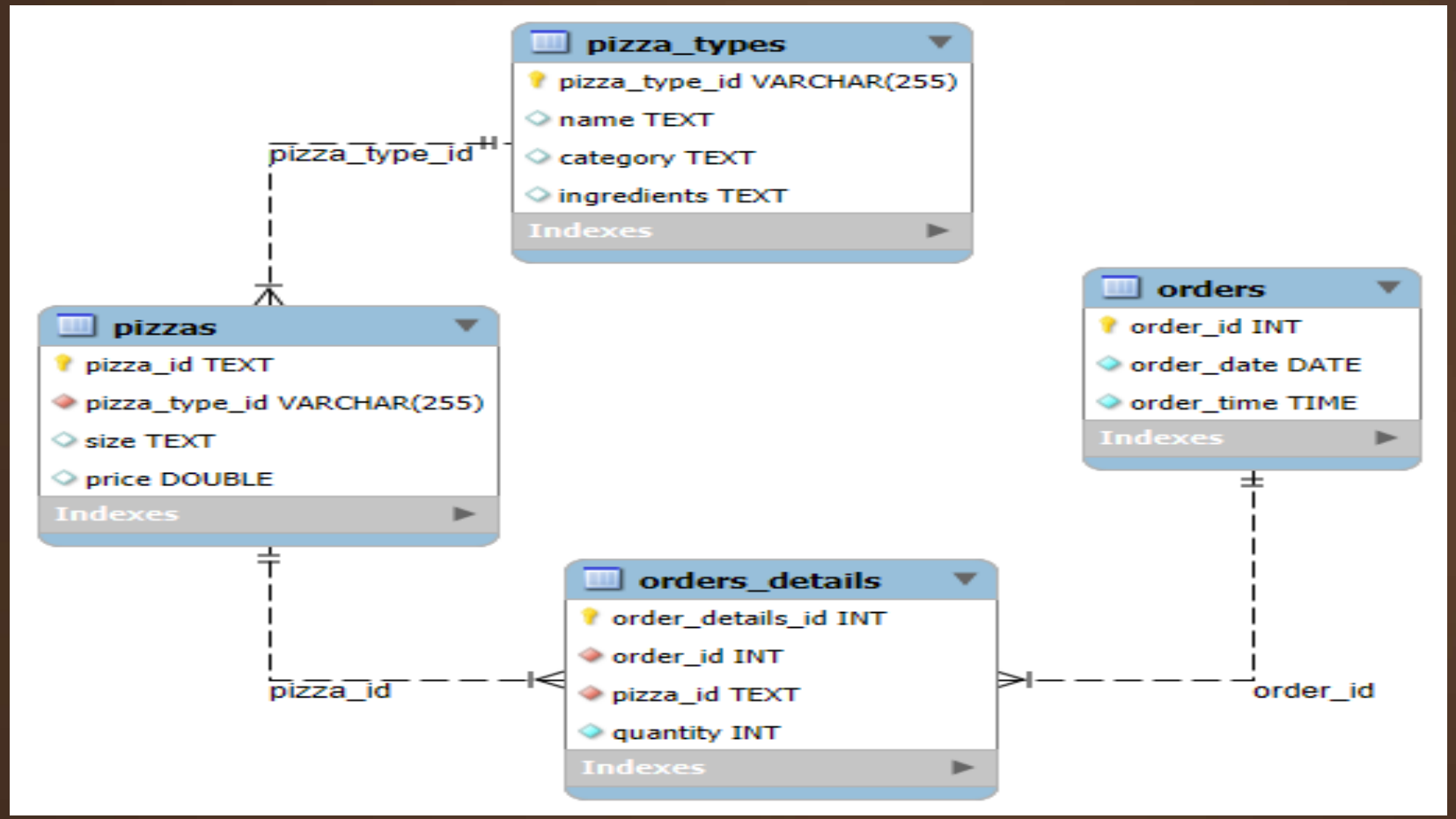
2. **orders**
   - Contains information about customer orders, including the unique order_id, the date (date), and time (time) the order was placed.
   - Links to the "order_details" table via order_id.

3. **pizza_types**
    - Describes the types of pizzas available, with a unique pizza_type_id, name, category (e.g., Chicken, Veg), and ingredients.
    - Links to the “order_details” table via pizza_id (though there may be differences in ID format that need mapping).

4. **Pizzas**
    - This table has information about pizza size and price of each pizza types
    - Links to the “ pizza_types” via pizza_type_id and with “order_details” via pizza_id

Entity-Relationship Diagram (ERD)

Created following diagram using MYSQL :



Here are some key points for discussing the **Entity-Relationship Diagram (ERD)**

Entities and Their Attributes:

1. **Orders :**
    - Attributes
        - order_id (primary key)
        - date
        - Time
    - Description : Represents an order placed by a customer, including when the order was made and the order's unique identifier.

2. **Order_details :**
    - Attributes
        - Order_details_id (primary key)
        - Order_id (foreign key to orders)
        - Pizza_id (foreign key to pizza_types)
        - Quantity
    - Description : Contains details of individual items in each order, including the type of pizza and the quantity ordered.
3. **Pizza_types :**
    - **Attributes**
        - **Pizza_type_id (primary key)**
        - **Name**

- Categories
- Ingradients

- Description : Describes the available pizza types with relevant information like pizza name, category (e.g., Chicken, Veg), and ingredients.

4. **Pizzas :**
   - Attributes
     - Pizza_id (primary key)
     - Pizza_type_id (foreign key)
     - Size
     - Price
   - Description : This table has information about pizza size and price of each pizza types

Key Insights and Analysis

- **Popular Pizza Types:** Query results showing the most ordered pizza types.
- Order Trends: Number of orders per day or time of day (e.g., peak order times).
- **Quantity Trends:** Most commonly ordered quantities for different pizza types.
- **Sales by Category:** Analysis of which pizza categories (e.g., Chicken, Veg) are most popular.

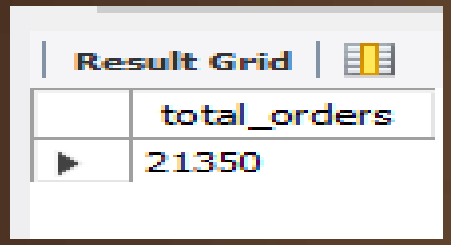
SQL Queries and Results

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

- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

1. **Retrieve the total number of orders placed.**

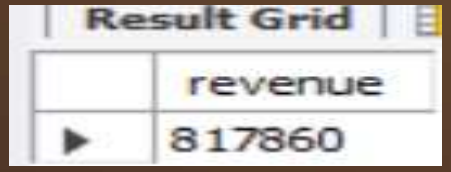
- select count(order_id) from orders;
  Result :



2. **Calculate the total revenue generated from pizza sales.**

- select round(sum(pizzas.price*orders_details.quantity),0) revenue from orders_details join pizzas on orders_details.pizza_id = pizzas.pizza_id;

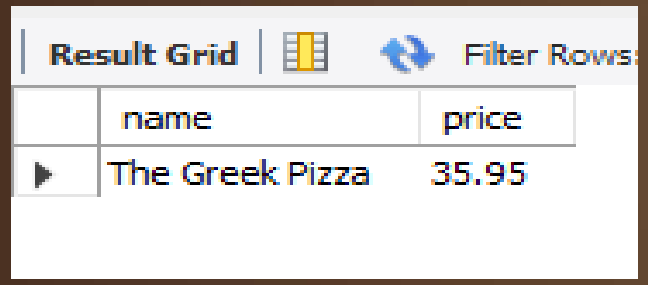
  Result :



3. Identify the highest-priced pizza by using limit also without using limit function.

```
select pizza_types.name, pizzas.price from pizzas
join pizza_types on pizzas.pizza_type_id = pizza_types.pizza_type_id
order by pizzas.price desc limit 1;
```

Result :

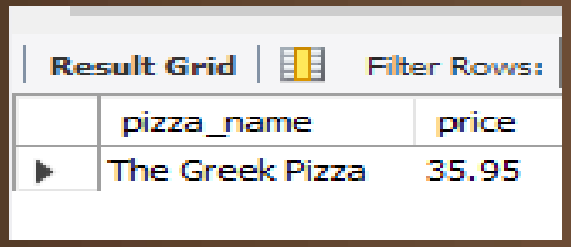


	name	price
▶	The Greek Pizza	35.95

By Using Window Function :

```
with cte as (select x2.name as pizza_name, x1.price as price,
        rank() over(order by price desc) as rn
        from pizzas x1
        join pizza_types x2 on x1.pizza_type_id = x2.pizza_type_id)
    select pizza_name, price from cte
    where rn = 1
```

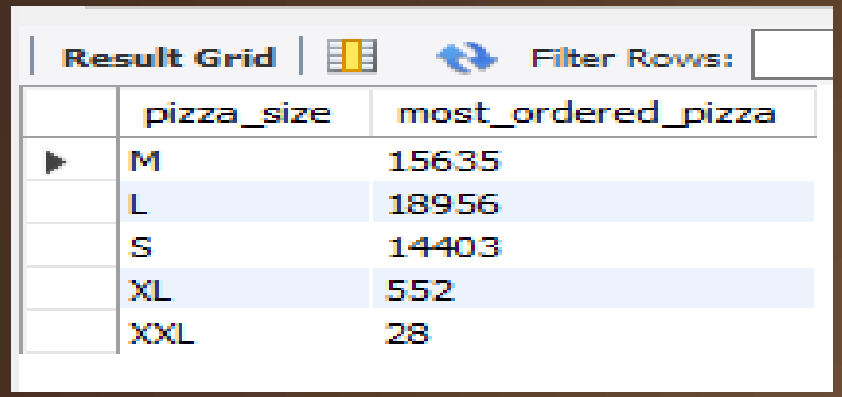
Result :



4. Identify the most common pizza size ordered.

```
select x1.size pizza_size, sum(x2.quantity) most_ordered_pizza
    from pizzas x1
    join orders_details x2 on x1.pizza_id = x2.pizza_id
    group by x1.size;
```

Results :



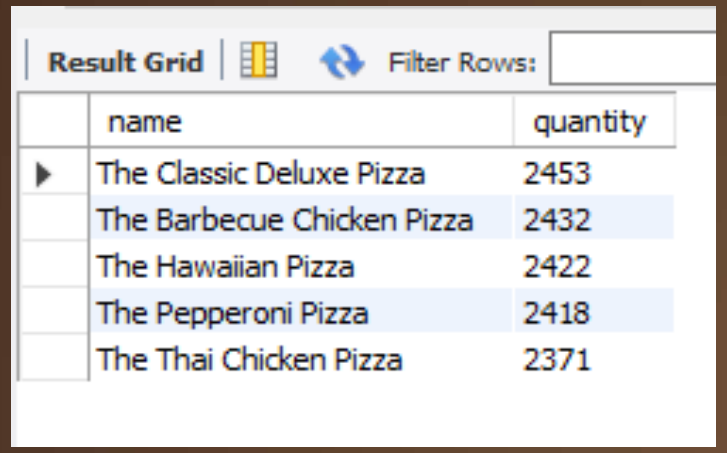
Result Grid | Filter Rows:

pizza_size	most_ordered_pizza
M	15635
L	18956
S	14403
XL	552
XXL	28

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

```
select x1.name, sum(x3.quantity) quantity
from pizza_types x1
join pizzas x2 on x1.pizza_type_id = x2.pizza_type_id
join orders_details x3 on x3.pizza_id = x2.pizza_id
group by x1.name order by quantity desc limit 5;
```

Result :

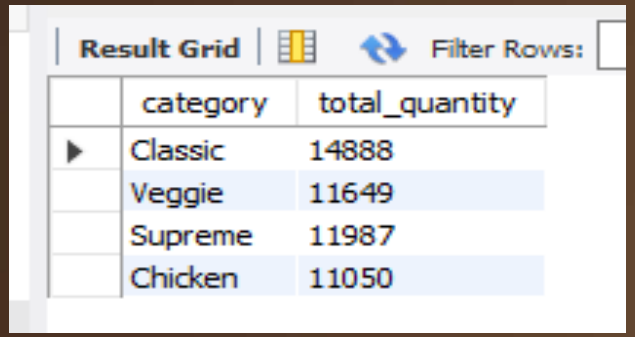

Result Grid | Filter Rows:

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

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

```
SELECT x1.category, SUM(x3.quantity) total_quantity
FROM pizza_types x1
JOIN pizzas x2 ON x1.pizza_type_id = x2.pizza_type_id
JOIN orders_details x3 ON x2.pizza_id = x3.pizza_id
GROUP BY x1.category;
```

Result :


Result Grid | Filter Rows:

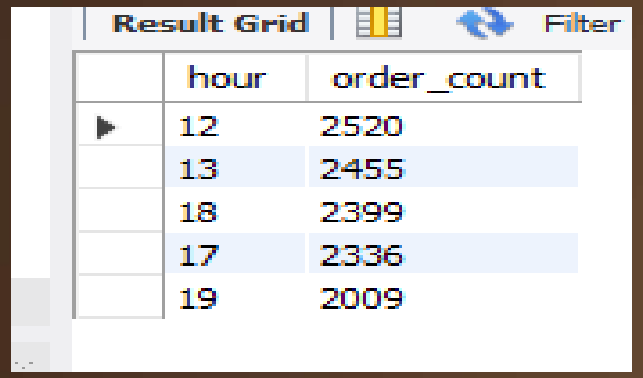
	category	total_quantity
▶	Classic	14888
	Veggie	11649
	Supreme	11987
	Chicken	11050

7. Top 5 most ordered day timings out of 24 hours

- select hour(order_time) hour, count(order_id) order_count
  from orders
  group by hour(order_time) order by order_count desc limit 5;

  Result :

Result : as we can see from result people mostly orders at (12, 1) pm in afternoon and at evening (5,6 and 7) pm

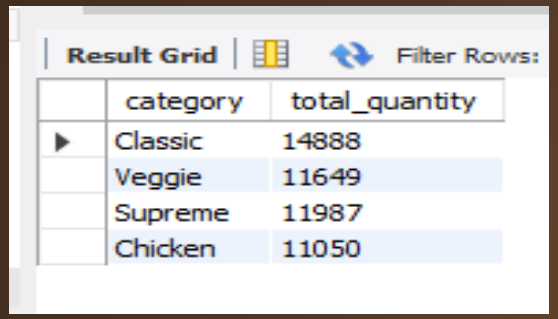


hour	order_count
12	2520
13	2455
18	2399
17	2336
19	2009

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

```
select x1.category, sum(x3.quantity)  total_quantity
from pizza_types x1
join pizzas x2 on x1.pizza_type_id = x2.pizza_type_id
join orders_details x3 on x2.pizza_id = x3.pizza_id
group by x1.category
```

Result :

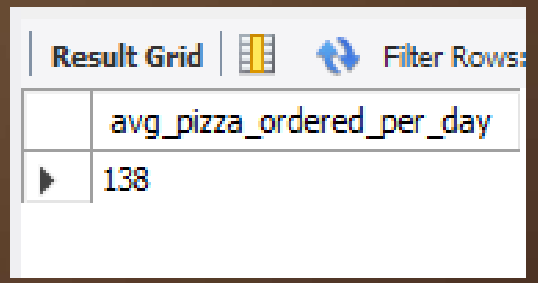


category	total_quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

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

- select round(avg(quantity),0) as avg_pizza_ordered_per_day
  from(select x1.order_date as date, sum(x2.quantity) as quantity from orders x1
  join orders_details x2 on x1.order_id = x2.order_id
  group by date) as order_quantity

Result :

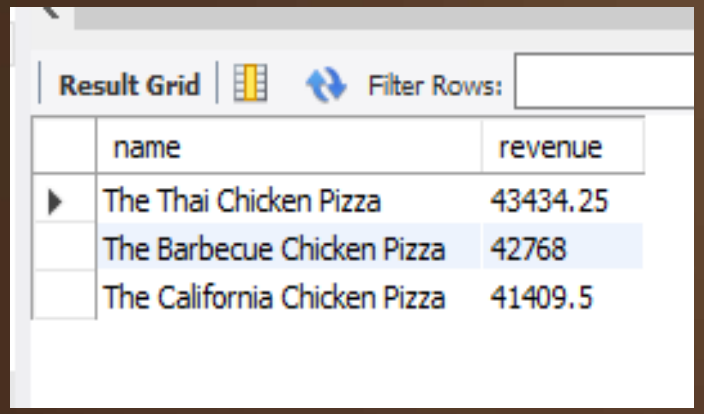


avg_pizza_ordered_per_day
138

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

```
select x1.name, sum(x2.price*x3.quantity) as revenue
from pizza_types x1
join pizzas x2 on x1.pizza_type_id = x2.pizza_type_id
join orders_details x3 on x3.pizza_id = x2.pizza_id
group by x1.name order by revenue desc limit 3;
```

Result :

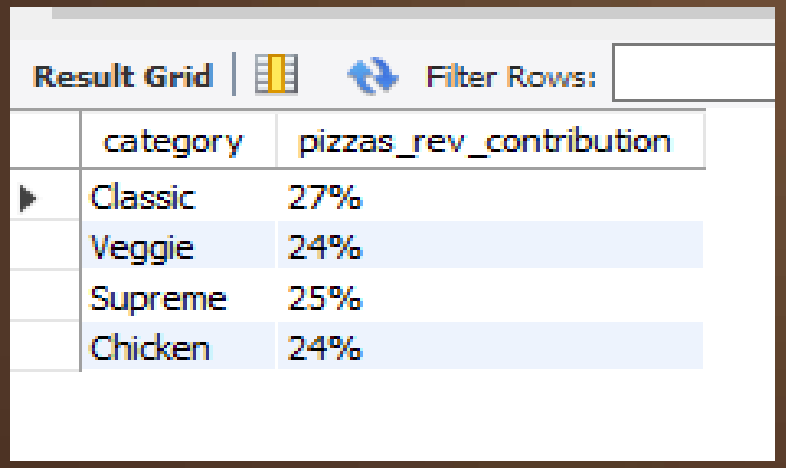


name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

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

```
Select x1.category, concat(round(sum(x2.price*x3.quantity) * 100 / (select
sum(pizzas.price*orders_details.quantity)
from pizzas join orders_details on pizzas.pizza_id = orders_details.pizza_id),0),'%')
as pizzas_rev_contribution
from pizza_types x1
join pizzas x2 on x1.pizza_type_id = x2.pizza_type_id
join orders_details x3 on x3.pizza_id = x2.pizza_id
group by x1.category;
```

Result :



category	pizzas_rev_contribution
Classic	27%
Veggie	24%
Supreme	25%
Chicken	24%

12. Analyze the cumulative revenue generated over time.

- Used aggragate of window function (to get the cumulative sum)

```
with cte as (
select x1.order_date as date, sum(x3.price*x2.quantity) as revenue
from orders x1
join orders_details x2 on x1.order_id = x2.order_id
join pizzas x3 on x2.pizza_id = x3.pizza_id
group by date
)
select time, revenue, sum(revenue) over(order by date) cum_sum
from cte
```

Result : query output is given in file format at right bottom corner

Microsoft Excel
ma Separated Valı

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

```
select category, name, revenue from
(select category, name, revenue,
 rank() over(partition by category order by revenue desc) as ranking
from(
select x1.category, x1.name, sum(x2.price*x3.quantity) as revenue
from pizza_types x1
join pizzas x2 on x1.pizza_type_id = x2.pizza_type_id
join orders_details x3 on x2.pizza_id = x3.pizza_id
group by x1.category, x1.name) as temp_table) as a
where ranking <= 3;
```

Result : query output is given in file format at right bottom corner

Microsoft Excel
ma Separated Valı

THE END