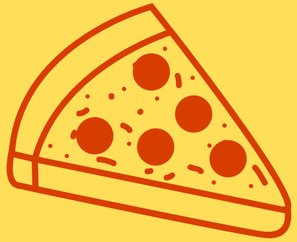


SQL Projects for Pizza Sales A Comprehensive Analysis of Pizza Sales Data Using SQL

Hemant Soni | Ujjain, India

ABOUT ME & PROJECT OVERVIEW

Introduction to Project:
Project Overview: The goal of this project is to analyze pizza sales data using SQL to uncover key insights into sales trends, revenue generation, and customer preferences. This analysis is broken down into basic, intermediate, and advanced SQL queries to progressively extract deeper insights.

About Me:

- Hemant Soni Ujjain, India
- Email: Hemantsoni12122003@gmail.com
- LinkedIn: linkedin.com/in/hemant-soni-5210b7226/
- GitHub: github.com/Hemant-5516 (Link to project repository)

PROJECT OVERVIEW

- **Objective**: To leverage SQL queries in analyzing pizza sales data to gain actionable insights.
- **Scope**: This project involves basic retrievals, intermediate data joins, and advanced analytical queries to explore sales patterns and customer behaviors.
- **Outcome**: The project provides a detailed understanding of pizza sales trends, helping in identifying top-selling pizzas, peak ordering times, and revenue distribution.

DATABASE STRUCTURE

Tables Involved:

- **Orders**: Contains details of each order placed.
- **Order_Details**: Holds specific details about pizzas in each order.
- **Pizzas**: Information about the pizzas, including size and price.
- **Pizza_Types**: Categorizes pizzas by type and ingredients.

Key Columns:

- **order_id**: Unique identifier for orders.
- **pizza_id**: Unique identifier for pizzas.
- **pizza_type**: Type/category of pizza.
- **price**: Cost of the pizza.
- **size**: Size of the pizza (e.g., Small, Medium, Large).
- **quantity**: Number of pizzas ordered.
- **date and time**: When the order was placed

Relationships:
Orders → Order_Details: One-to-many relationship (one order can have multiple pizza details).
Order_Details → Pizzas: Many-to-one relationship (each detail links to a specific pizza).
Pizzas → Pizza_Types: Many-to-one relationship (each pizza is categorized under one type).

LEVELS

01. Basic
- Simple retrievals like total orders, revenue, highest-priced pizza.
- Suitable for understanding the foundational data.

02. Intermediate
- Joining tables, analyzing orders by time, and calculating averages.
- Ideal for exploring data relationships and trends.

03. Advanced
- Complex queries involving cumulative revenue and percentage contributions.
- Focuses on deeper insights and strategic decisions.

BASIC:

1. Retrieve the total number of orders placed.

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

	total_orders
▶	21350

2. 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 pizzas.pizza_id = orders_details.pizza_id
```

	total_revenue
▶	817860.05

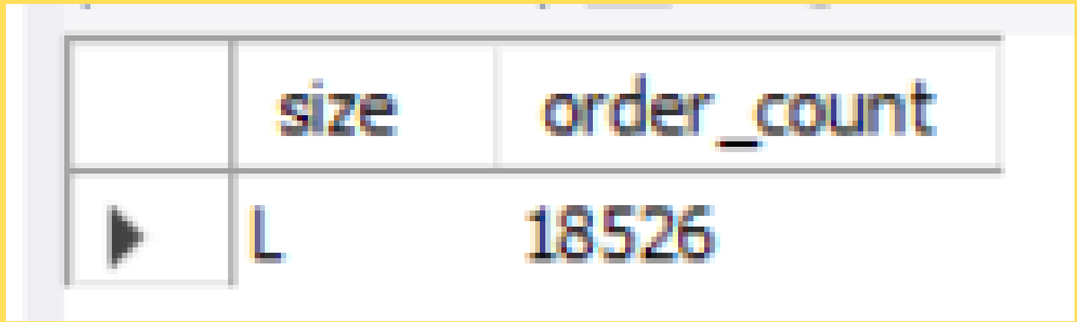
3. Identify the highest-priced pizza.

```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

4. Identify the most common pizza size ordered.

```sql
SELECT
    pizzas.size,
    COUNT(orders_details.order_detail_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;
```



size	order_count
L	18526

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

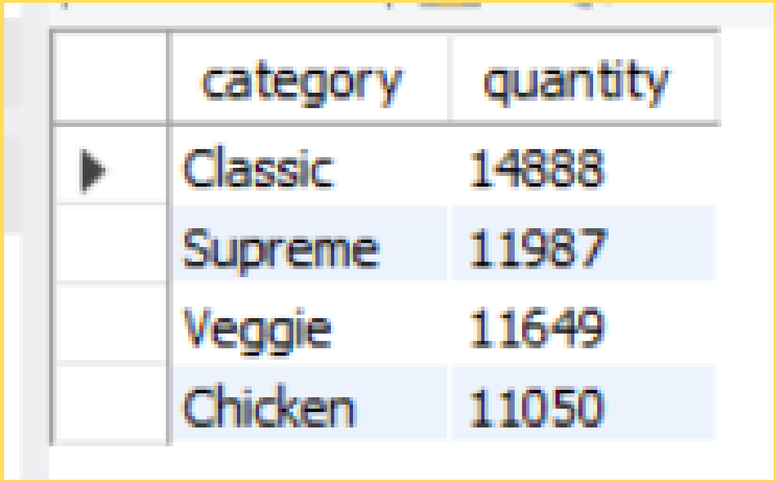
```sql
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

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

INTERMEDIATE:

1. 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;
```



	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

2. 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);
```

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

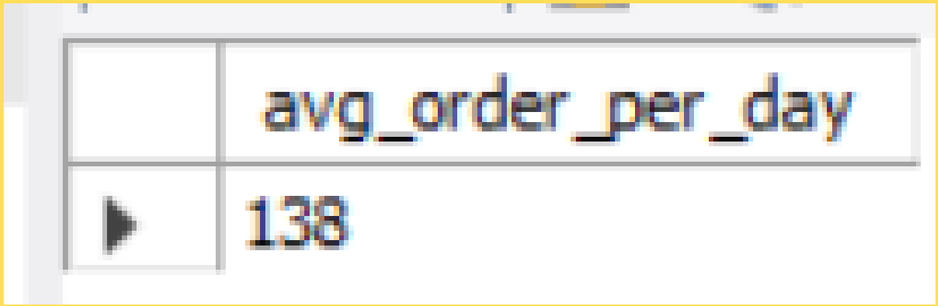
3. Join relevant tables to find the category-wise distribution of pizzas.

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

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

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

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_order_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;
```



avg_order_per_day
138

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

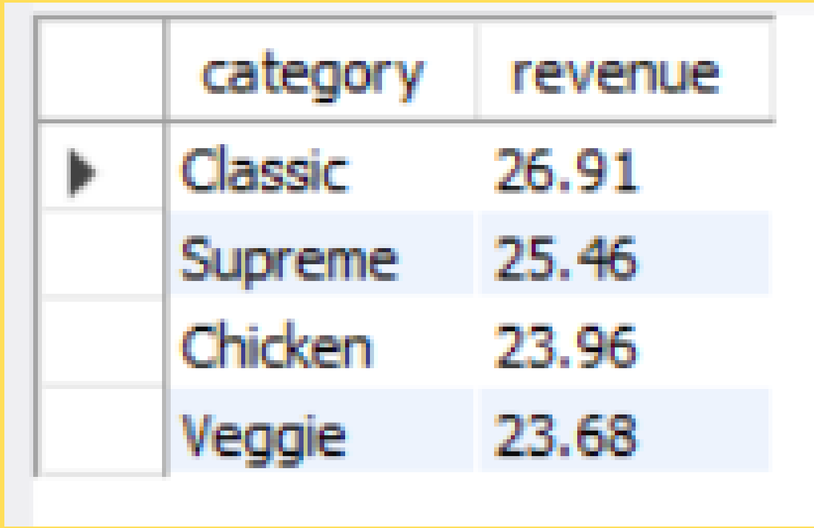
```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

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

ADVANCED:

1. 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 pizzas.pizza_id = orders_details.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;
```



	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

2. Analyze the cumulative revenue generated over time.

```sql
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;
```

order_date	cum_revenue
2015-01-01	2713.850000000004
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
2015-01-19	43365.75000000001
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001

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

```sql
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;
```

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
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

CONCLUSION & FURTHER RESOURCES

- **Summary**: This project has successfully analyzed pizza sales data to extract valuable insights into sales patterns, revenue generation, and customer preferences.
- **Future Work**: Future analysis could involve integrating more data points, such as customer feedback, and applying machine learning models for predictive analytics.

HEMANT SONI

PIZZA SALES SQL PROJECT

GET IN TOUCH WITH US

hemantsoni12122003@gmail.com

linkedin.com/in/hemant-soni-5210b7226/

github.com/Hemant-5516

THANK YOU!

THANK YOU FOR VIEWING MY PRESENTATION. I APPRECIATE YOUR TIME AND LOOK FORWARD TO CONNECTING WITH YOU