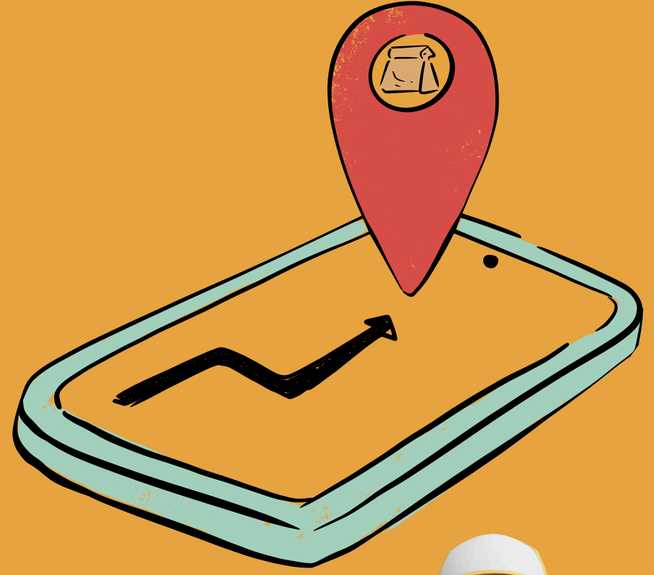
SQL Project On Delicious Pizza Sales

Delicious Pizzas For Everyone!



Hello!

My Name is Keshav Garg and
In This Project i have utilise SQL Queries to solve
a questions the were related to pizzas sales

This project focuses on analyzing pizza sales data to gain insights into sales performance, customer preferences, and operational efficiency. Using SQL, the project involves extracting, transforming, and analyzing data from a relational database that includes tables such as Orders, Customers, Pizzas, Toppings, and Sales. Key objectives include identifying top-selling pizzas, understanding peak sales periods, examining customer purchase patterns, and optimizing inventory management. The analysis helps in making data-driven decisions to enhance sales strategies and improve overall business performance.

Three Winning Strategies

Loyalty rewards boost customer retention

Implement a points system to reward customers for repeat orders and encourage loyalty.

Interactive online ordering enhances customer experience

Develop a user-friendly website and mobile app for seamless ordering and tracking.

Social media promotions drive online visibility

Leverage platforms like Facebook and Instagram to run promotions and reach a wider audience.

Incorporate SQL data to optimize pizza sales by analyzing customer preferences, ordering patterns efficiency to increase profitability and customer satisfaction.

Retrieve the total number of orders placed.

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

output

Results | Messages

	total_orders
1	21350

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

```
SELECT pizza_types.category,
SUM(order_details.quantity) AS quantity
FROM     pizza_types INNER JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
 INNER JOIN order_details
 ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC
```

output

Results | Messages

	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

Determine the distribution of orders by hour of the day.

```sql
SELECT { fn HOUR(time) } AS hours,
COUNT(order_id) AS order_count
FROM      orders
GROUP BY { fn HOUR(time) }
```

output

Results | Messages

	hours	order_count
1	23	28
2	15	1468
3	9	1
4	12	2520
5	21	1198
6	18	2399
7	10	8
8	19	2009

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

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

output

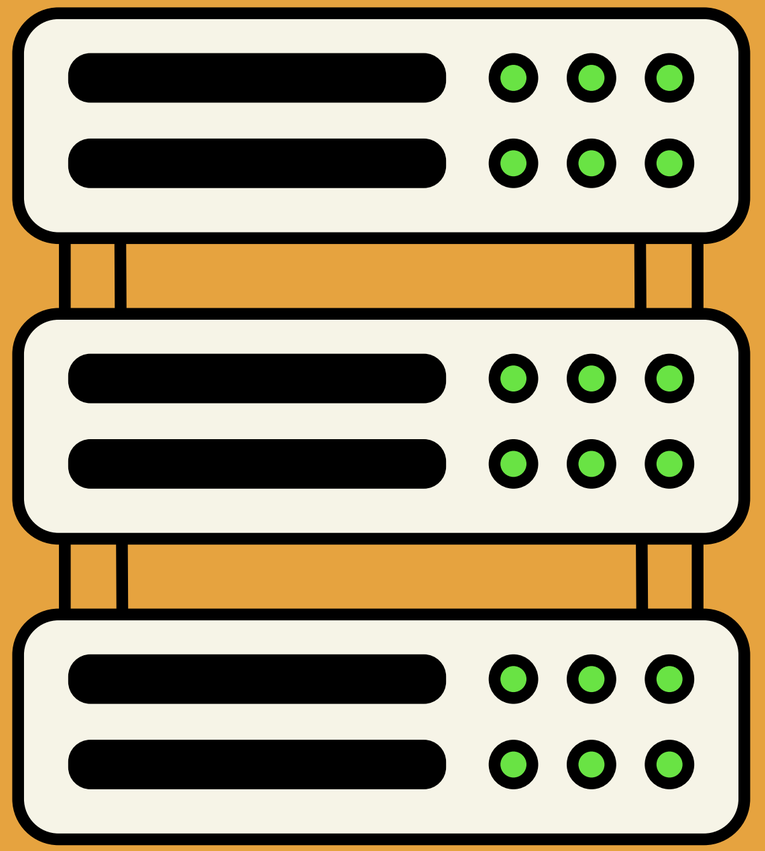
85 %

Results | Messages

	category	Distribution_of_pizza
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

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

```sql
SELECT ROUND(AVG(quantity), 0)
AS avg_no_of_pizza_ordered_per_day
FROM     (SELECT orders.date,
SUM(order_details.quantity) AS quantity
FROM  orders INNER JOIN order_details
ON orders.order_id = order_details.order_id
GROUP BY orders.date) AS order_quantity
```



output

Results | Messages

	avg_no_of_pizza_ordered_per_day
1	138

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

```sql
SELECT TOP (3) pizza_types.name,
ROUND(SUM(order_details.quantity * pizzas.price),
0) AS revenue
FROM  pizza_types
INNER JOIN pizzas
ON pizzas.pizza_type_id = pizza_types.pizza_type_id
INNER JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
```

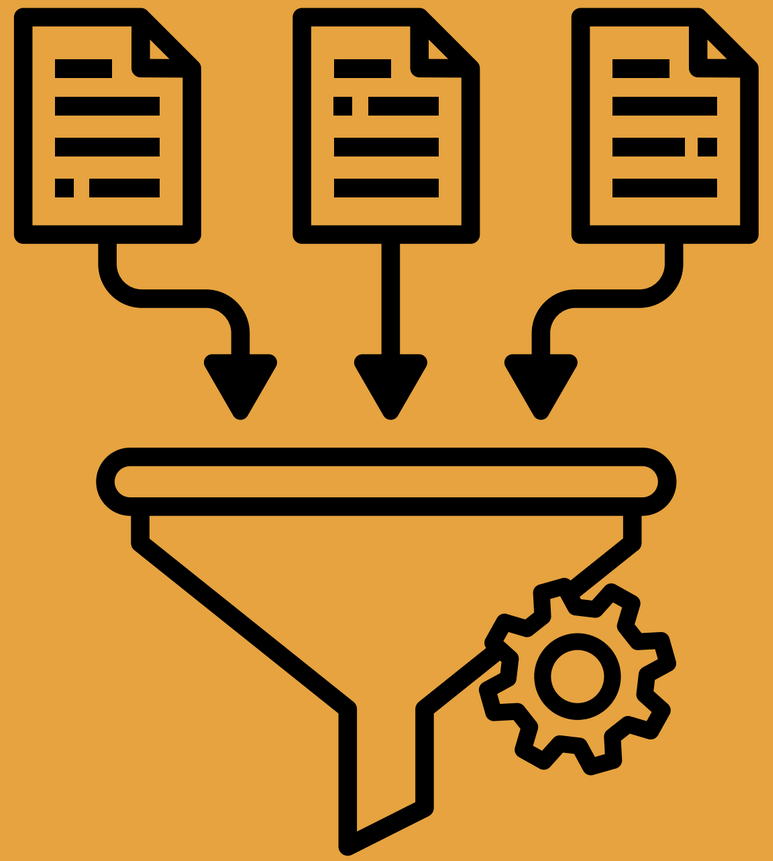
output

Results | Messages

	name	revenue
1	The Thai Chicken Pizza	43434
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41410

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

```sql
SELECT pizza_types.category,
SUM(order_details.quantity * pizzas.price) /
(SELECT ROUND(SUM(order_details.quantity * pizzas.price)
, 2) AS total_sales
FROM order_details INNER JOIN pizzas
ON pizzas.pizza_id = order_details.pizza_id) * 100 AS revenue
FROM     pizza_types INNER JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
INNER JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC
```



output

Results | Messages

	category	revenue
1	Classic	26.9059602582816
2	Supreme	25.4563112372441
3	Chicken	23.9551375568473
4	Veggie	23.6825910501471

Analyze the cumulative revenue generated over time.

```sql
select date,
sum(revenue)over(order by date) as cum_revenue
from
(select orders.date,
sum(order_details.quantity * pizzas.price)as revenue
from order_details join pizzas
on order_details.pizza_id=pizzas.pizza_id
join orders
on orders.order_id=order_details.order_id
group by orders.date) as sales;
```

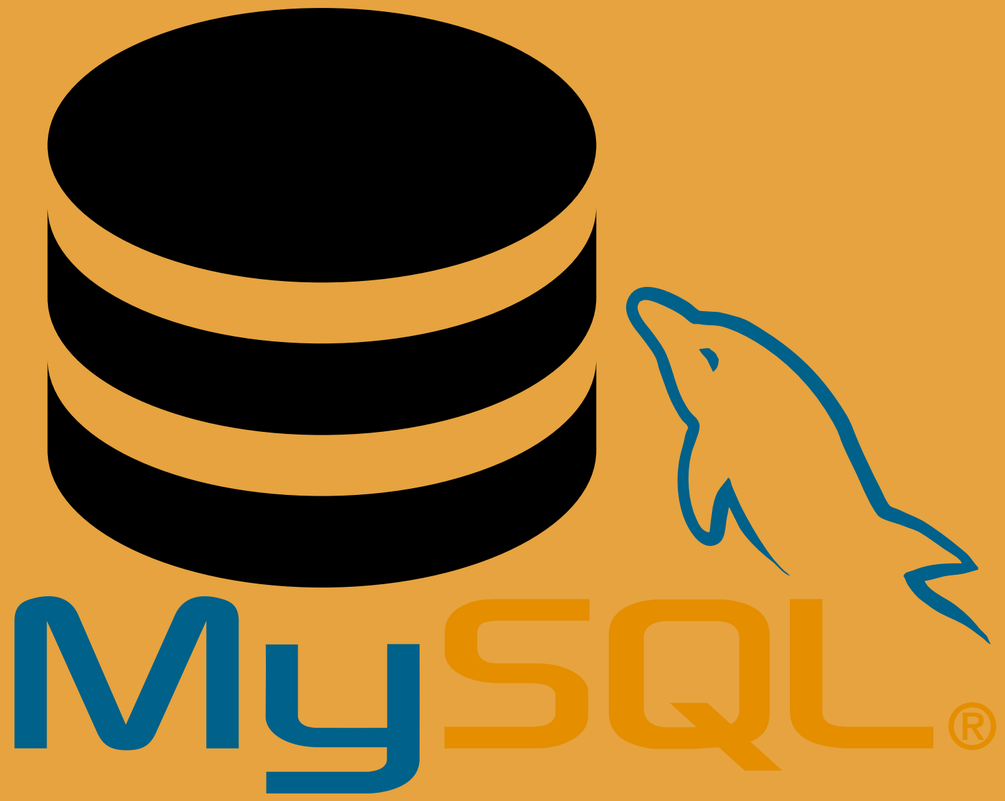
output

Results | Messages

	date	cum_revenue
1	2015-01-01	2713.85000228882
2	2015-01-02	5445.7500038147
3	2015-01-03	8108.15000724792
4	2015-01-04	9863.60000801086
5	2015-01-05	11929.5500087738
6	2015-01-06	14358.5000114441
7	2015-01-07	16560.700012207
8	2015-01-08	19399.0500183105
9	2015-01-09	21526.4000225067
10	2015-01-10	23990.350025177

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((order_details.quantity)*pizzas.price) as revenue
from pizza_types
join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id
join order_details
on order_details.pizza_id=pizzas.pizza_id
group by pizza_types.category,pizza_types.name)as a)as b
where rn<=3;
```



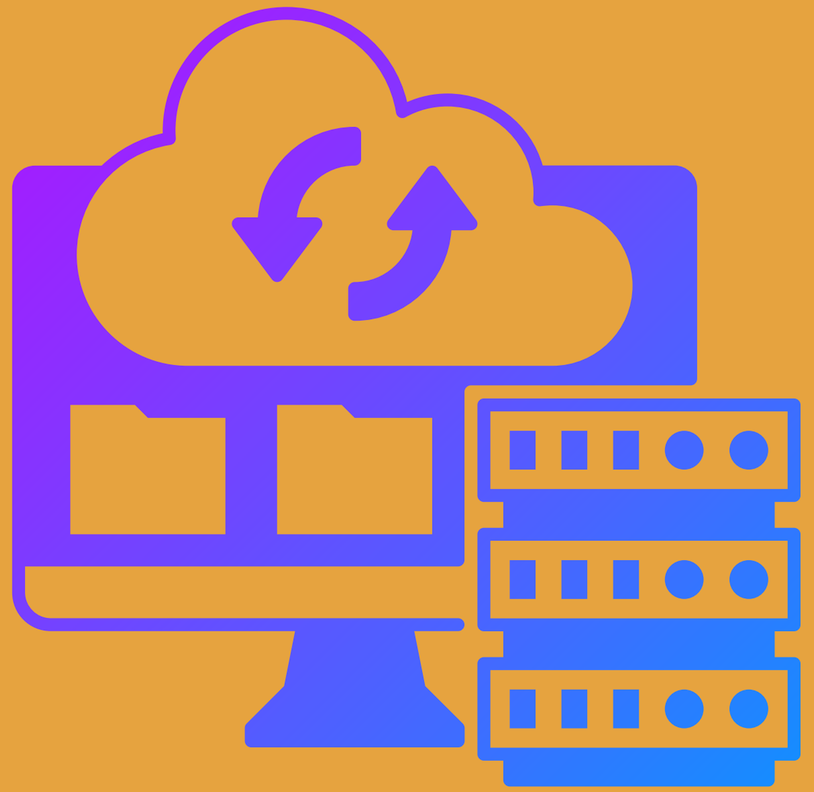
output

Results | Messages

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

Calculate the total revenue generated from pizza sales.

```sql
SELECT ROUND(SUM(o.quantity * p.price), 2)
AS Total_Revenue_Generated
FROM order_details AS o
INNER JOIN pizzas AS p
ON o.pizza_id = p.pizza_id
```



output

Results | Messages

	Total_Revenue_Generated
1	817860.05

Identify the highest-priced pizza.

```sql
SELECT p1.name, ROUND(p.price, 2) AS Expr1
FROM pizza_types AS p1
INNER JOIN pizzas As p
ON p1.pizza_type_id = p.pizza_type_id
ORDER BY p.price DESC
```

output

Results | Messages

	name	Expr1
1	The Greek Pizza	35.95
2	The Greek Pizza	25.5
3	The Brie Carre Pizza	23.65
4	The Italian Vegetables Pizza	21
5	The Spicy Italian Pizza	20.75
6	The Spinach Supreme Pizza	20.75
7	The Spinach Pesto Pizza	20.75
8	The Italian Supreme Pizza	20.75

Identify the most common pizza size ordered.

```sql
SELECT p.size,
COUNT(o1.order_details_id) AS Order_Count
FROM pizzas AS p
INNER JOIN order_details AS o1
ON p.pizza_id = o1.pizza_id
GROUP BY p.size
ORDER BY Order_Count DESC
```

output

Results | Messages

	size	Order_Count
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

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

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

output

85 %

Results | Messages

	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

Pizza SQL

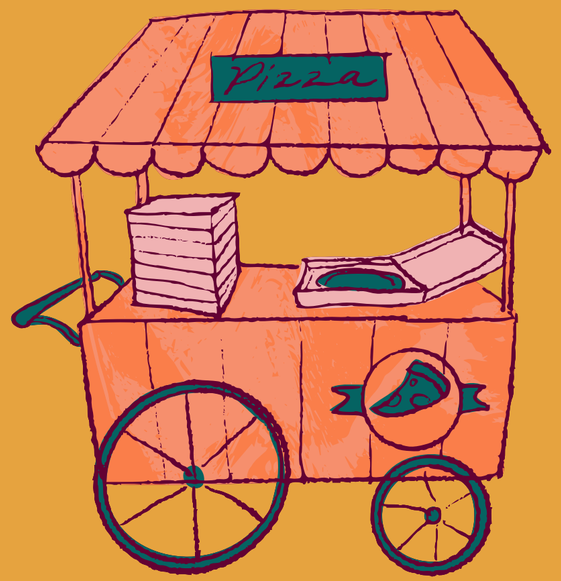
Incorporate SQL data to optimize pizza sales by analyzing customer preferences, ordering patterns efficiency to increase profitability and customer satisfaction.

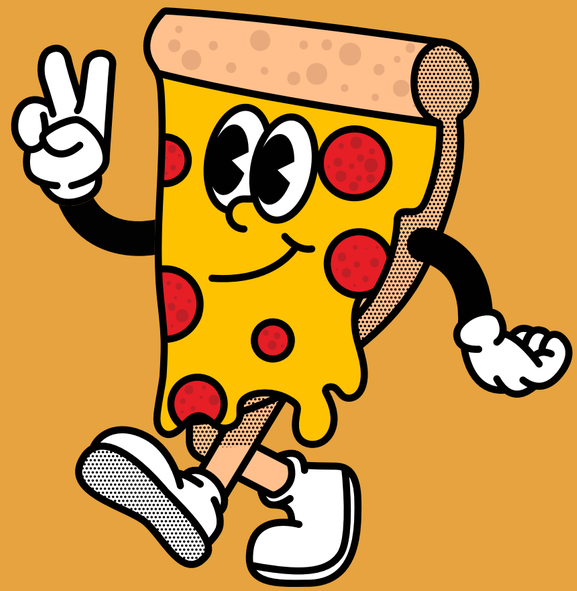
MySQL®

SQLicious

Satisfy your hunger for data and pizza with our irresistible freebies. Indulge in the perfect combination of flavor and function.

Get a free slice with every SQL query order! Our delicious pizzas are a perfect reward for your database endeavors. Order now!





Thank you!

We appreciate your time and attention. If you have any questions or need further information, please feel free to ask.