SQL

PROJECT

KUMARSHUBHAM8689@GMAIL.COM

Pizza Sales Analysis Using SQL

Developed a comprehensive SQL-based

hello

HOW ARE YOU?

MY NAME IS SHUBHAM KUMAR, AND IN THIS PROJECT, I USED SQL TO ANALYZE PIZZA SALES DATA. THE GOAL IS TO SOLVE VARIOUS BUSINESS-RELATED QUESTIONS USING SQL QUERIES, FOCUSING ON ASPECTS LIKE REVENUE, CUSTOMER PREFERENCES, AND PIZZA POPULARITY.T

PIZZA
CORNER

SINCE 1999

Retrieve the total number of orders placed


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

Result Grid

total_orders
21350

Calculate the total revenue generated from pizza sales.


```
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sum
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid

total_sum
817860.05

dentify the lowest-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;
```

Result Grid | Filter Ro

name	price
The Greek Pizza	35.95

Identify the most common pizza size ordered


```
SELECT
    COUNT(orders_details.order_details_id), pizzas.size
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size;
```

Result Grid | Filter Rows:

COUNT(orders_details.order_details_id)	size
15385	M
18526	L
14137	S
544	XL
28	XXL

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


```sql
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 category
```

Result Grid

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Determine the distribution of orders by hour of the day

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

Result Grid | Filter Rows:

HOUR(order_time)	COUNT(order_id)
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.


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

Result Grid | Filter

	AVG(order_quantity)
▶	138.4749

Determine the top 3 most ordered pizza types based on revenue


```
SELECT
    pizza_types.name,
    SUM(pizzas.price * orders_details.quantity) AS revenue
FROM
    pizzas
        JOIN
    pizza_types 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 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, sum(orders_details.quantity*pizzas.price) AS REVNUE
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
```

Result Grid | Filter Rows:

category	REVNUE
Classic	220053.1000000001
Supreme	208196.99999999822
Chicken	195919.5
Veggie	193690.45000000298

Analyze the cumulative revenue generated over time

```sql
SELECT
    orders.order_date,
    SUM(pizzas.price * orders_details.quantity) AS daily_revenue,
    SUM(SUM(pizzas.price * orders_details.quantity)) OVER (ORDER BY orders.order_date) AS cumulative_revenue
FROM
    orders
JOIN
    orders_details ON orders.order_id = orders_details.order_id
JOIN
    pizzas ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY
    orders.order_date
ORDER BY
    orders.order_date;
```

PIZZA
CORNER

SINCE 1999

Result Grid | Filter Rows: | Export: | Wrap Ce

order_date	daily_revenue	cumulative_revenue
2015-01-01	2713.8500000000004	2713.8500000000004
2015-01-02	2731.8999999999996	5445.75
2015-01-03	2662	8.15
2015-01-04	1755.4500000000003	9863.6
2015-01-05	2065.95	11929.55
2015-01-06	2428.95	14358.5
2015-01-07	2202.2000000000003	16560.7
2015-01-08	2838.3499999999995	19399.05
2015-01-09	2127.3500000000004	21526.4
2015-01-10	2463.95	23990.350000000002
2015-01-11	1872.3000000000002	25862.65
2015-01-12	1919.0500000000002	27781.7
2015-01-13	2049.6000000000004	29831.300000000003
2015-01-14	2527.3999999999996	32358.700000000004

2731.8999999999996

THANK'S FOR WATCHING

This project by Shubham Kumar uses SQL queries to analyze pizza sales data. It focuses on revenue trends, top-selling pizzas, and customer behavior to provide insights for optimizing sales, marketing, and inventory strategies.