Project Title: Pizza House Sales Analysis

Name: Shraddha Ramesh Mule

Description:

This project involves utilizing PostgreSQL to address real-world business challenges by analyzing sample data. Through the application of various PostgreSQL techniques and queries, the project aims to derive meaningful insights and solutions to enhance business decision-making processes. The analysis includes, but is not limited to, revenue calculations, performance ranking, and trend identification.

Introduction to Pizza house Sales Analysis

This project focuses on analyzing sales data from a pizza shop using PostgreSQL. The goal is to answer various business-related questions and uncover insights that can help improve decision-making. By querying the data, I aim to explore patterns in sales performance, customer preferences, and operational efficiency. The dataset includes details about pizza orders, sales transactions, pizza types etc. Through a series of 13 SQL queries, I will address key questions such as:

- What are the peak sales times?
- Which is the highest selling category of Pizza?
- How does sales performance vary by month or day of the week?

The insights derived from these queries can help the pizza shop optimize its operations, improve customer satisfaction, and increase profitability.

Retrieve the total number of orders placed.

Input:

Query | Query History

```sql
-- Retrieve the total number of orders placed

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

Output:

	total_orders bigint
1	21350

Calculate the revenue generated from pizza sales

Input:

Query | Query History

```sql
-- calculate the total revenue generated from pizza sales

SELECT
    ROUND(SUM(order_details.quantity * pizzas.prize)::numeric, 2) AS total_sales
FROM
    order_details
JOIN
    pizzas
ON
    pizzas.pizza_id = order_details.pizza_id;
```

Output:

	total_sales numeric
1	817860.05

Identify the highest-priced pizza.

Input:

Query | Query History

```sql
-- identify the highest-priced pizza.

SELECT pizzas.prize, pizza_types.name
FROM pizzas
JOIN pizza_types
ON pizzas.pizza_types_id = pizza_types.pizza_type_id
ORDER BY pizzas.prize DESC
LIMIT 1;
```

Output:

	prize double precision	name text
1	35.95	The Greek Pizza

Identify the most common pizza size ordered.

Input:

Query | Query History

```sql
-- identify the most common pizzas size ordered.

SELECT pizzas.size, COUNT(order_details.order_details_id) AS order_count
FROM pizzas
JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC
limit 1;
```

Output:

	size text	order_count bigint
1	L	18526

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

Input:

Query | Query History

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

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

Output:

	name text	quantity bigint
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

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

Input:

Query | Query History

```sql
-- 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
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_types_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

Output:

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

Determine the distribution of orders by hour of day

Input:

Query | Query History

```sql
-- Determine the distribution of orders by hour of the day.


SELECT EXTRACT(HOUR FROM time) AS hour,
    COUNT(order_id) AS order_count
FROM orders GROUP BY EXTRACT(HOUR FROM time)
ORDER BY hour;

```

Output:

https://shorturl.at/bJauC

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

Input:

Query | Query History

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

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

Output:

	category text	name_count bigint
1	Supreme	9
2	Chicken	6
3	Classic	8
4	Veggie	9

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

Input:

Query | Query History

```sql
-- 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 orders.order_date,
           SUM(order_details.quantity) AS quantity
    FROM orders
    JOIN order_details
    ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date
) AS daily_totals;
```

Output:

	avg_pizza_ordered_per_day numeric
1	138

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

Input:

Query | Query History

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

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

Output:

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

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

Input:

Query | Query History

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

select pizza_types.category,
round(sum(order_details.quantity * pizzas.prize) / (select
(sum(order_details.quantity * pizzas.prize)
) as total_sales
from order_details
join pizzas on pizzas.pizza_id = order_details.pizza_id) * 100) as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_types_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

Output:

	category text	revenue double precision
1	Classic	27
2	Supreme	25
3	Chicken	24
4	Veggie	24

Analyze the cumulative revenue generated over time.

Input:

Query | Query History

```sql
-- Analyze the cumulative revenue generated over time.

SELECT order_date,
       SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM (
    SELECT orders.order_date,
           SUM(order_details.quantity * pizzas.prize) 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.order_date
) AS sales;
```

Output:

https://shorturl.at/7AS7i

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

Input:

Query | Query History

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

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.prize) AS revenue
        FROM pizza_types
        JOIN pizzas
        ON pizza_types.pizza_type_id = pizzas.pizza_types_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:

https://shorturl.at/oBhqU

THANK YOU