A slice of pizza with red dots and green leaves

Description automatically generatedPizza Sales Revenue Analysis Report

**Unlocking Insights into Pizza Sales Performance**

1. **Total Revenue**

--Calculating the total revenue from the 'orders' table.

SELECT ROUND(SUM(total\_price),2) AS Revenue

FROM orders

Output

1. **Average Order Value**

--Calculating the average value of each order in the 'orders' table.

--Step 1: Adding the total prices of all orders using SUM Function.

--Step 2: Counting the distinct order\_ids.

SELECT ROUND(SUM(total\_price) / COUNT(DISTINCT order\_id),2) AS Average\_Order\_Value

A screenshot of a computer

Description automatically generatedFROM orders

Output

1. **Total Pizzas Sold**

--Calculating the total pizzas sold from the 'orders' table.

SELECT SUM(quantity) AS Total\_Pizzas\_Sold

FROM orders

A screenshot of a computer

Description automatically generated Output

1. **Total Orders**

--Calculating the total orders from the 'orders' table.

SELECT COUNT(DISTINCT order\_id) AS Total\_Orders

FROM orders

****

Output

1. **Average Pizzas Per Order**

--Calculating the Average Pizzas Per order from the 'orders' table.

--Step 1: Calculating the total quantity of orders placed.

--Step 2: Calculating the count of distinct order\_id's.

SELECT SUM(quantity)/COUNT(DISTINCT order\_ID) AS Average\_Pizzas\_Per\_Order

FROM orders

****

Output

1. **Daily Trend for Total Orders**

--We will use DATENAME function to extract the day of the week from the order\_date column.

--We will count the distinct order\_id to identify the total orders placed.

SELECT DATENAME(DW, order\_date) AS Order\_Day, COUNT(DISTINCT order\_id) AS Total\_Orders

FROM orders

--As we have used an aggregate function and another column in our query, we use group by clause to obtain the results.

**A screenshot of a table

Description automatically generated**GROUP BY DATENAME(DW, order\_date)

Output

1. **Month trend for Total Orders**

--We will use DATENAME function to extract the month from the orders\_date column.

--We will count the distinct order\_id to identify the total orders placed.

SELECT DATENAME(Month, order\_date) AS Month, COUNT(DISTINCT order\_id) AS Total\_Orders

FROM orders

--As we have used an aggregate function and another column in our query, we used group by clause to obtain the results.

GROUP BY DATENAME(Month, order\_date)

--Ordering the results of the Total\_orders placed in descending order to know which month has the highest order.

ORDER BY Total\_Orders DESC

**A screenshot of a calendar

Description automatically generated**

Output

1. **Percentage of Sales by Pizza Category**

--Calculating the percentage of total price for each pizza category relative to the overall total price of all orders.

--Selecting the pizza category and calculating the percentage.

SELECT pizza\_category,

--Calculating the percentage as the sum of total prices for the pizza category divided by the total price of all orders.

ROUND(SUM(total\_price) \* 100 / (SELECT SUM(total\_price) FROM orders), 2) AS Percentage\_of\_pizza\_category

--Retrieving data from the 'orders' and 'pizza' tables.

FROM orders

--Using the Left join to connect the pizza table and the orders table using the order\_details\_id.

LEFT JOIN pizza

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping the results by pizza category.

GROUP BY pizza\_category;

A screenshot of a data

Description automatically generated

Output

1. **Percentage of Sales by Pizza Size**

--Selecting the pizza\_size and calculating the percentage of it

SELECT pizza\_size,

--Calculating the percentage as the sum of the total\_price as per pizza size divided by the total price of all orders

ROUND(SUM(total\_price) \* 100 / (SELECT SUM(total\_price) FROM orders),2) AS Percentage\_of\_pizza\_size

FROM orders

--Using Left join to connect pizza and orders table using order\_details\_id column

LEFT JOIN pizza

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping by pizza\_size

GROUP BY pizza\_size

--Ordering the results as per the highest order

A screenshot of a data

Description automatically generatedORDER BY Percentage\_of\_pizza\_size DESC;

Output

1. **Top 5 Pizzas by Revenue**

--Selecting pizza\_name and calculating the total\_price from the pizza table and the orders table.

-- Selecting the top 5 pizza names.

SELECT TOP 5 pizza\_name, SUM(total\_price) AS Total\_Revenue

FROM pizza

--Using Left Join to connect the orders table and pizza table using order\_details\_id column.

LEFT JOIN orders

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping the results by pizza\_name.

GROUP BY pizza\_name

--Ordering the results of total\_revenue in descending order.

**A list of food items

Description automatically generated**ORDER BY Total\_Revenue DESC;

Output

1. **Top 5 Pizzas by Revenue**

--Selecting pizza\_name and calculating the total\_price from the pizza table and the orders table.

-- Selecting the Bottom 5 pizza names # Here we won't be ordering with DESC.

SELECT TOP 5 pizza\_name, ROUND(SUM(total\_price),2) AS Total\_Revenue

FROM pizza

--Using Left Join to connect the orders table and pizza table using order\_details\_id column.

LEFT JOIN orders

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping the results by pizza\_name.

GROUP BY pizza\_name

--Ordering the results of total\_revenue in ascending order.

ORDER BY Total\_Revenue ASC;

A list of food items

Description automatically generated

Output

1. **Top 5 Pizzas by Quantity**

--Selecting pizza\_name and calculating the total\_quantity from the pizza table and the orders table.

-- Selecting the Top 5 pizza names.

SELECT TOP 5 pizza\_name, SUM(quantity) AS Total\_Quantity

FROM pizza

--Using Left Join to connect the orders table and pizza table using order\_details\_id column.

LEFT JOIN orders

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping the results by pizza\_name.

GROUP BY pizza\_name

--Ordering the results of total\_quantity in descending order.

ORDER BY Total\_Quantity DESC;

A menu with a box

Description automatically generated with medium confidence

Output

1. **Bottom 5 Pizzas by Quantity**

--Selecting pizza\_name and calculating the total\_quantity from the pizza table and the orders table.

-- Selecting the Bottom 5 pizza names.

SELECT TOP 5 pizza\_name, SUM(quantity) AS Total\_Quantity

FROM pizza

--Using Left Join to connect the orders table and pizza table using order\_details\_id column.

LEFT JOIN orders

ON orders.order\_details\_id = pizza.order\_details\_id

--Grouping the results by pizza\_name.

GROUP BY pizza\_name

--Ordering the results of total\_quantity in ascending order.

ORDER BY Total\_Quantity ASC;

A screenshot of a menu

Description automatically generated

Output

1. **Top 5 Pizzas by Total\_Orders**

-- Selecting pizza\_name and calculating the total quantity from the pizza table and the orders table.

-- Selecting the top 5 pizza names.

SELECT TOP 5 pizza\_name, COUNT(DISTINCT order\_id) AS Total\_orders

-- Retrieve data from the 'pizza' and 'orders' tables.

FROM pizza

Group the results by pizza\_name.

GROUP BY pizza\_name

-- Order the results by the count of distinct orders in descending order.

-- Ordering in descending order to get the top 5

A menu of a pizza

Description automatically generatedORDER BY Total\_orders DESC

Output

1. **Bottom 5 Pizzas by Total\_Orders**

-- Selecting pizza\_name and calculating the total quantity from the pizza table and the orders table.

-- Selecting the bottom 5 pizza names.

SELECT TOP 5 pizza\_name, COUNT(DISTINCT order\_id) AS Total\_orders

-- Retrieve data from the 'pizza' and 'orders' tables.

FROM pizza -- The 'pizza' table

-- Group the results by pizza\_name.

GROUP BY pizza\_name

-- Order the results by the count of distinct orders in descending order.

-- Ordering in ascending order to get the bottom 5

ORDER BY Total\_orders ASC;

A screenshot of a menu

Description automatically generated

Output