Pizza Sales Data Analysis – MySQL Queries

Summary:

This document contains a tested MySQL query for calculating total revenue from the Pizza_Sales table in the pizzahut schema.

Query (1) – Calculate Rounded Total Revenue:

SQL Query: SELECT ROUND(SUM(total_price), 2) AS Total_Revenue FROM Pizza_Sales;

Purpose:

Calculates the sum of all sales (total_price) and rounds the result to two decimal places for easier reading.

Result Image:

	Total_Revenue
▶	336440.6

Query (2) – Calculate Average Order Value (AOV):

SQL Query:

```
SELECT ROUND(SUM(total_price) / COUNT(DISTINCT order_id), 2) AS Avg_Order_Value
FROM Pizza_Sales;
```

Purpose:

Calculates the Average Order Value by dividing the total revenue (SUM(total_price)) by the number of unique orders (COUNT(DISTINCT order_id)), rounded to two decimal places.

Result Image:

	Avg_Order_Value
▶	38.25

Query (3) – Total Pizzas Sold:

SQL Query: SELECT SUM(quantity) AS Total_pizza_sold FROM Pizza_Sales;

Purpose:
Calculates the total number of pizzas sold by summing all values in the quantity column.

Result Image:

	Total_pizza_sold
▶	20383

Query (4) – Total Orders:

SQL Query: SELECT COUNT(DISTINCT order_id) AS Total_Orders FROM Pizza_Sales;

Purpose:
Counts the total number of unique orders by identifying distinct order_id values in the sales table.

Result Image:

	Total_Orders
▶	8796

Query (5) – Average Pizzas per Order:

SQL Query: simplified version using ROUND():

SQL Query:
SELECT ROUND(SUM(quantity) / COUNT(DISTINCT order_id), 2) AS Avg_Pizzas_per_order
FROM Pizza_Sales;

Purpose:
Calculates the average number of pizzas sold per order by dividing the total quantity of pizzas by the number of unique orders, with the result rounded to two decimal places for clarity.

Result Image: (I used alternative)

	Avg_Pizzas_per_order
▶	2.32

Query (6) – Daily Trend for Total Orders:

SQL Query:

```
SELECT
   DAYNAME(STR_TO_DATE(order_date, '%d-%m-%Y')) AS order_day,
   COUNT(DISTINCT order_id) AS total_orders
FROM Pizza_Sales
WHERE order_date IS NOT NULL
  AND order_date <> ''
  AND STR_TO_DATE(order_date, '%d-%m-%Y') IS NOT NULL
GROUP BY order_day
ORDER BY FIELD(order_day,
            'Monday', 'Tuesday', 'Wednesday',
            'Thursday', 'Friday', 'Saturday', 'Sunday');
```

Purpose:

Analyzes the daily pattern of order volume, showing how many unique orders are placed on each day of the week.

Expected Result:

A table with two columns:

- order_day: The weekday name (e.g., "Monday")
- total_orders: The count of unique orders made on that day

This lets you compare order counts across weekdays to identify your busiest days.

Result Image:

	order_day	total_orders
▶	Monday	1234
	Tuesday	1213
	Wednesday	1254
	Thursday	1275
	Friday	1454
	Saturday	1263
	Sunday	1103

Query (7) – Hourly Trend for Orders:

SQL Query:

```
SELECT
   HOUR(order_time) AS order_hours,
   COUNT(DISTINCT order_id) AS total_orders
FROM Pizza_Sales
GROUP BY HOUR(order_time)
ORDER BY HOUR(order_time);
```

Purpose:

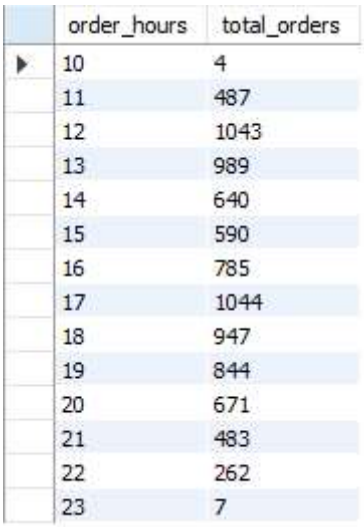
Shows how many unique orders are placed in each hour of the day, helping identify peak order times.

Expected Result:

A table with:

- order_hours → hour of the day in 24-hour format (e.g., 0 = midnight, 13 = 1 PM)
- total_orders → number of unique orders placed during that hour.

Result Image:



order_hours	total_orders
10	4
11	487
12	1043
13	989
14	640
15	590
16	785
17	1044
18	947
19	844
20	671
21	483
22	262
23	7

Query (8) – Percentage of Sales by Pizza Category:

SQL Query:

```
SELECT
  pizza_category,
  CAST(SUM(total_price) AS DECIMAL(10,2)) AS total_revenue,
  CAST(SUM(total_price) * 100 /
     (SELECT SUM(total_price) FROM Pizza_Sales)
  AS DECIMAL(10,2)) AS PCT
FROM Pizza_Sales
GROUP BY pizza_category;
```

Purpose:

Calculates total sales revenue for each pizza category and the percentage share of the total sales that category represents.

Expected Result:

A table with:

- pizza_category → name of the category (e.g., "Classic", "Veggie", "Chicken")
- total_revenue → total revenue for that category, rounded to 2 decimals
- PCT → percentage of overall sales from that category, rounded to 2 decimals

Result Image:

pizza_category	total_revenue	PCT
Classic	89853.05	26.71
Veggie	80751.80	24.00
Supreme	85459.75	25.40
Chicken	80376.00	23.89

Query (9) – Percentage of Sales by Pizza Size:

SQL Query:

```
SELECT
   pizza_size,
   CAST(SUM(total_price) AS DECIMAL(10,2)) AS total_revenue,
   CAST(SUM(total_price) * 100 /
      (SELECT SUM(total_price) FROM Pizza_Sales)
   AS DECIMAL(10,2)) AS PCT
FROM Pizza_Sales
GROUP BY pizza_size
ORDER BY pizza_size;
```

Purpose:

Breaks down sales revenue by pizza size and shows the percentage each size contributes to total sales.

Result Image:

	pizza_size	total_revenue	PCT
▶	L	154413.55	45.90
	M	102126.00	30.35
	S	73124.75	21.73
	XL	6273.00	1.86
	XXL	503.30	0.15

Query (10) – Total Pizzas Sold by Category (February Only):

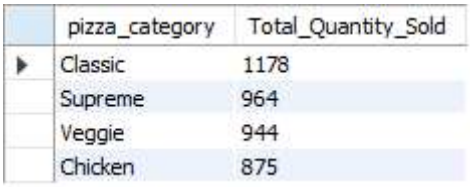
SQL Query:

```
SELECT
   pizza_category,
   SUM(quantity) AS Total_Quantity_Sold
FROM Pizza_Sales
WHERE
   order_date IS NOT NULL
   AND order_date <> ''
   AND STR_TO_DATE(order_date, '%d-%m-%Y') IS NOT NULL
   AND MONTH(STR_TO_DATE(order_date, '%d-%m-%Y')) = 2
GROUP BY pizza_category
ORDER BY Total_Quantity_Sold DESC;
```

Purpose:

Displays, for each pizza category, the total number of pizzas sold in February by summing the quantity column for orders in that month, grouped and ranked by category.

Result Image:



pizza_category	Total_Quantity_Sold
Classic	1178
Supreme	964
Veggie	944
Chicken	875

Query (11) – Top 5 Best Sellers by Total Pizzas Sold:

SQL Query:

```
SELECT
   pizza_name,
   SUM(quantity) AS Total_Pizza_Sold
FROM Pizza_Sales
GROUP BY pizza_name
ORDER BY Total_Pizza_Sold DESC
LIMIT 5;
```

Purpose:

Finds the five most popular pizzas based on total quantity sold, ranking them from highest to lowest.

Result Image:

pizza_name	Total_Pizza_Sold
The Barbecue Chicken Pizza	1046
The Pepperoni Pizza	1009
The Hawaiian Pizza	997
The Classic Deluxe Pizza	980
The California Chicken Pizza	958

Query (12) – Bottom 5 Best Sellers by Total Pizzas Sold:

SQL Query:

```
SELECT
  pizza_name,
  SUM(quantity) AS Total_Pizza_Sold
FROM Pizza_Sales
GROUP BY pizza_name
ORDER BY Total_Pizza_Sold ASC
LIMIT 5;
```

Purpose:

Lists the five lowest-selling pizzas based on total quantity sold, ranking from the smallest to largest number sold.

Result Image:

pizza_name	Total_Pizza_Sold
The Brie Carre Pizza	200
The Mediterranean Pizza	370
The Calabrese Pizza	375
The Spinach Pesto Pizza	390
The Chicken Pesto Pizza	395