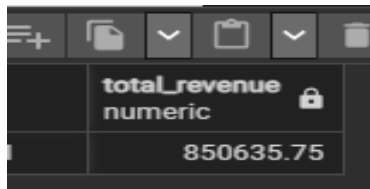


Pizza Sales SQL Queries

A. KPI

1. Total Revenue:

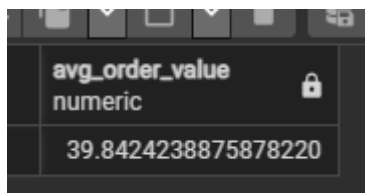
```
SELECT SUM(quantity * total_price) AS total_revenue FROM pizza
```



total_revenue
850635.75

2. Average Order Value

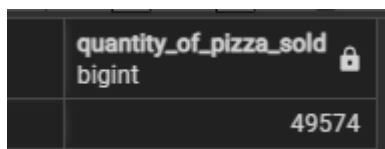
```
SELECT SUM(quantity * total_price) / COUNT(DISTINCT order_id) AS Avg_Order_value  
FROM pizza
```



avg_order_value
39.8424238875878220

3. Quantity of Pizza Sold

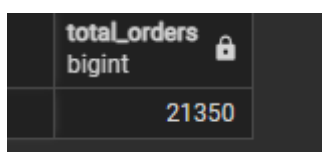
```
SELECT SUM(quantity) AS quantity_of_pizza_sold FROM pizza
```



quantity_of_pizza_sold
49574

4. Total orders placed

```
SELECT COUNT(DISTINCT order_id) AS Total_Orders FROM pizza
```



total_orders
21350

5. Average Pizzas Per Order

```
SELECT CAST(SUM(quantity) AS DECIMAL(10,2)) / CAST(COUNT(DISTINCT order_id) AS DECIMAL(10,2)) AS Average_Pizzas_Per_Order FROM pizza
```

average_pizzas_per_order
numeric
2.3219672131147541

CHART REQUIREMENT

1. Daily Trend for total orders

```
SELECT TO_CHAR(order_date, 'Day') AS order_day, COUNT(DISTINCT order_id) AS total_orders FROM pizza GROUP BY TO_CHAR(order_date, 'Day') ORDER BY COUNT(DISTINCT order_id) DESC
```

	order_day text	total_orders bigint
1	Friday	3538
2	Thursday	3239
3	Saturday	3158
4	Wednesday	3024
5	Tuesday	2973
6	Monday	2794
7	Sunday	2624

2. Hourly trend for total orders

```
SELECT TO_CHAR(order_time, 'HH24') AS peak_hour, COUNT(DISTINCT order_id) FROM  
pizza GROUP BY TO_CHAR(order_time, 'HH24') ORDER BY COUNT(DISTINCT order_id)  
DESC
```

1	peak_hour	count
2	12	2520
3	13	2455
4	18	2399
5	17	2336
6	19	2009
7	16	1920
8	20	1642
9	14	1472
10	15	1468
11	11	1231
12	21	1198
13	22	663
14	23	28
15	10	8
16	09	1

3. Monthly trend for total orders

```
SELECT TO_CHAR(order_date, 'Month') AS peak_month, COUNT(DISTINCT order_id)  
FROM pizza GROUP BY TO_CHAR(order_date, 'Month') ORDER BY COUNT(DISTINCT  
order_id) DESC
```

peak_month	count
July	1935
May	1853
January	1845
August	1841
March	1840
April	1799
November	1792
June	1773
February	1685
December	1680
September	1661
October	1646

4. Percentage of sales and total sales by Pizza Category

```
SELECT pizza_category, SUM(total_price) AS total_sales, ROUND(SUM(total_price) * 100  
/ (SELECT SUM(total_price) FROM pizza), 2) AS percentage_of_total_sales FROM pizza  
GROUP BY pizza_category
```

	pizza_category character varying 🔒	total_sales numeric 🔒	percentage_of_total_sales numeric 🔒
1	Supreme	208197.00	25.46
2	Chicken	195919.50	23.96
3	Veggie	193690.45	23.68
4	Classic	220053.10	26.91

NOTE

If you want to apply the Month, Quater, Week filters to the above query you can use WHERE clause, follow some of the examples

```
SELECT    pizza_category,    SUM(total_price)    AS    total_sales,
ROUND(SUM(total_price) * 100 / (SELECT SUM(total_price) FROM pizza), 2)
AS percentage_of_total_sales FROM pizza WHERE EXTRACT(Month FROM
order_date) = 1 GROUP BY pizza_category
```

This returns values where the Month represented in the WHERE clause by equating it to 1 is January. Equating it to 2 will give the values for February and so on.

If you want to find values for the QUARTER of the year(remember there are only 4 quarters in a year) use this

```
SELECT    pizza_category,    SUM(total_price)    AS    total_sales,
ROUND(SUM(total_price) * 100 / (SELECT SUM(total_price) FROM pizza), 2)
AS percentage_of_total_sales FROM pizza WHERE EXTRACT(QUARTER
FROM order_date) = 1 GROUP BY pizza_category
```

The number equated to the WHERE clause determines the values that will be displayed in accordance to the selected quarter.

5. Percentage of sales by pizza size.

```
SELECT      pizza_size,      SUM(total_price)      AS      total_sales,
ROUND(SUM(total_price) * 100 / (SELECT SUM(total_price) FROM pizza), 2)
AS percentage_of_total_sales FROM pizza GROUP BY pizza_size ORDER BY
percentage_of_total_sales DESC
```

	pizza_size character varying 🔒	total_sales numeric 🔒	percentage_of_total_sales numeric 🔒
1	L	375318.70	45.89
2	M	249382.25	30.49
3	S	178076.50	21.77
4	XL	14076.0	1.72
5	XXL	1006.60	0.12

6. Total Pizza sold by pizza category

```
SELECT      pizza_category,      MIN(unit_price),      MAX(unit_price),
ROUND(AVG(unit_price),3) AS average_unit_price, SUM(quantity) AS
quantit_sold, SUM(total_price) AS total_price FROM pizza GROUP BY
pizza_category ORDER BY SUM(total_price) DESC
```

	pizza_category character varying 🔒	min numeric 🔒	max numeric 🔒	average_unit_price numeric 🔒	quantit_sold bigint 🔒	total_price numeric 🔒
1	Classic	9.75	35.95	14.797	14888	220053.10
2	Supreme	12.25	23.65	17.363	11987	208197.00
3	Chicken	12.75	20.75	17.709	11050	195919.50
4	Veggie	12	21	16.613	11649	193690.45

7. Top 5 best Sellers by Revenue and Total Quantity

SELECT




pizza_name, SUM(unit_price * quantity) AS revenue, SUM(quantity) AS total_quantity

FROM pizza GROUP BY pizza_name ORDER BY revenue DESC, total_quantity DESC LIMIT 5

	pizza_name character varying 	revenue numeric 	total_quantity bigint 
1	The Thai Chicken Pizza	43434.25	2371
2	The Barbecue Chicken Pizza	42768.00	2432
3	The California Chicken Pizza	41409.50	2370
4	The Classic Deluxe Pizza	38180.5	2453
5	The Spicy Italian Pizza	34831.25	1924

8. Bottom 5 Sellers by Revenue and Total Quantity

SELECT pizza_name, SUM(unit_price * quantity) AS revenue, SUM(quantity) AS total_quantity
FROM pizza GROUP BY pizza_name ORDER BY revenue ASC, total_quantity ASC LIMIT 5

	pizza_name character varying 	revenue numeric 	total_quantity bigint 
1	The Brie Carre Pizza	11588.50	490
2	The Green Garden Pizza	13955.75	997
3	The Spinach Supreme Pizza	15277.75	950
4	The Mediterranean Pizza	15360.50	934
5	The Spinach Pesto Pizza	15596.00	970