MATTHEW ODUDU-ABASI

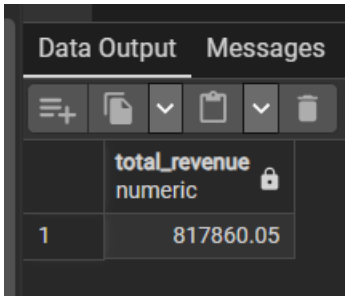
VEPH/20B/DA173

TASK 56A

PIZZA SALES SQL QUERIES

A. KPI's

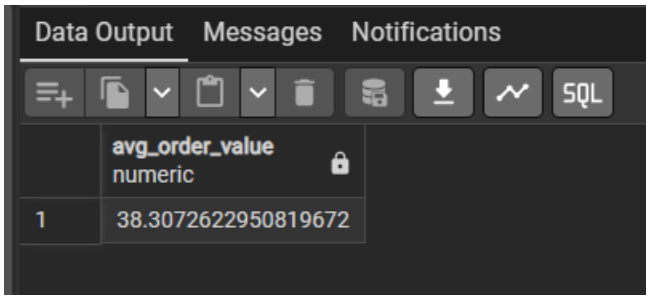
1. Total Revenue.
   SELECT SUM(total_price) AS Total_Revenue from pizza_sales



   Description: Calculates the total revenue by summing the total_price column across all orders.

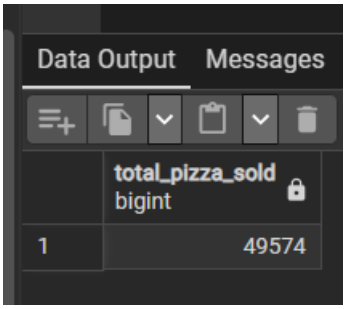
2. Average order value – the average amount spent per order calculated by dividing the total revenue by the total number of orders

   SELECT SUM(total_price) / COUNT(DISTINCT order_id) AS Avg_order_value from pizza_sales



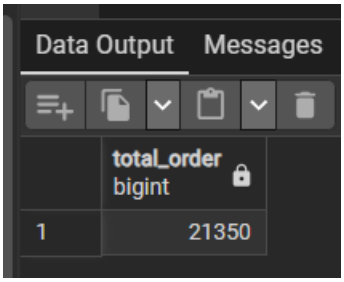
   Description: Computes the average amount spent per order by dividing total revenue by the total number of unique orders.

3. Total pizza sold – the sum of quantities of all pizza sold
   SELECT SUM(quantity) AS Total_pizza_sold from pizza_sales



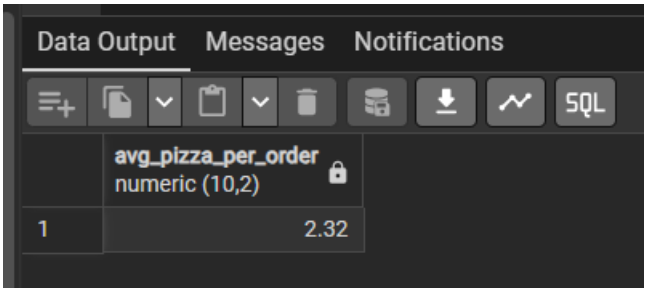
Description: Finds the total number of pizzas sold by summing up the quantity column.

4. Total orders – the total number of orders placed
   SELECT COUNT (DISTINCT order_id) Total_order from pizza_sales



Description: Counts the total number of unique orders placed.

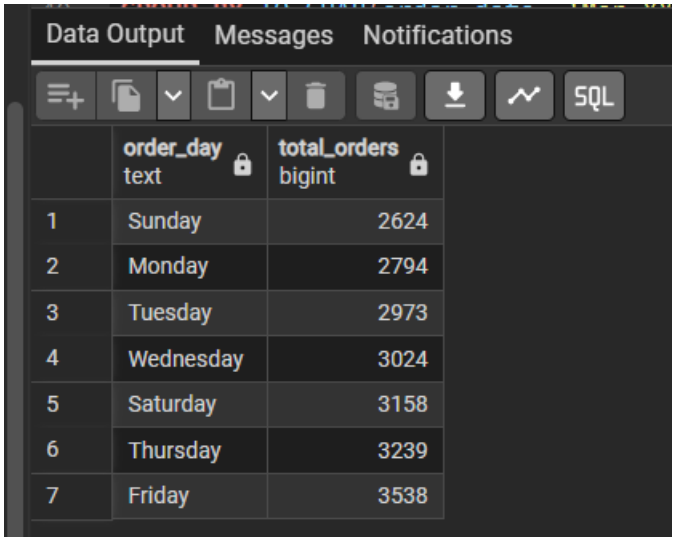
5. Average pizzas per order – the average number of pizzas sold per order calculated by dividing the total number of pizzas sold by the total number of orders
   SELECT CAST (CAST (SUM(quantity) AS DECIMAL (10,2)) /
   CAST (COUNT (DISTINCT order_id) AS DECIMAL (10,2)) AS DECIMAL (10,2)) AS Avg_pizza_per_order from pizza_sales



Description: Calculates the average number of pizzas per order by dividing total pizzas sold by total orders, rounded to 2 decimal places.

6. Total orders per day.
   SELECT
   TO_CHAR (order_date, 'FMDay') AS order_day,
   COUNT (DISTINCT order_id) AS total_orders
   FROM pizza_sales

GROUP BY TO_CHAR (order_date, 'FMDay')
ORDER BY total_orders



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

Description: Groups orders by day of the week and counts the number of unique orders each day.

7. Total orders per month

```
SELECT
  TO_CHAR (order_date, 'Mon YYYY') AS order_month,
  COUNT (DISTINCT order_id) AS total_orders
FROM pizza_sales
GROUP BY TO_CHAR (order_date, 'Mon YYYY')
ORDER BY MIN (order_date);
```

Data Output | Messages | Notifications

	order_month text	total_orders bigint
1	Jan 2015	1845
2	Feb 2015	1685
3	Mar 2015	1840
4	Apr 2015	1799
5	May 2015	1853
6	Jun 2015	1773
7	Jul 2015	1935
8	Aug 2015	1841
9	Sep 2015	1661
10	Oct 2015	1646
11	Nov 2015	1792
12	Dec 2015	1680

Description: Shows the percentage share of sales for each pizza category, restricted to January (month = 1).

8. Percentage of Sales by Pizza Category – the percentage contribution of each pizza category to total sales

   SELECT pizza_category, SUM (total_price) * 100 / (SELECT SUM(total_price) FROM pizza_sales) AS PCT_OF_SALES
   FROM pizza_sales
   GROUP BY pizza_category

Data Output | Messages | Notifications

	pizza_category character varying (100)	pct_of_sales numeric
1	Supreme	25.4563112600988396
2	Chicken	23.9551375568472870
3	Veggie	23.6825909273842145
4	Classic	26.9059602556696589

Description: Shows the percentage share of sales for each pizza category, restricted to January (month = 1).

9. --This query is telling us which pizza categories bring in the most money,

both in total sales and as a percentage of overall sales.

```
SELECT
  pizza_category,
  SUM(total_price) AS total_sales,
  ROUND(
    SUM(total_price) * 100.0 / (
      SELECT SUM(total_price)
      FROM pizza_sales      ),      2  ) AS pct_of_sales
FROM pizza_sales
GROUP BY pizza_category
ORDER BY total_sales DESC;
```

Data Output | Messages | Notifications

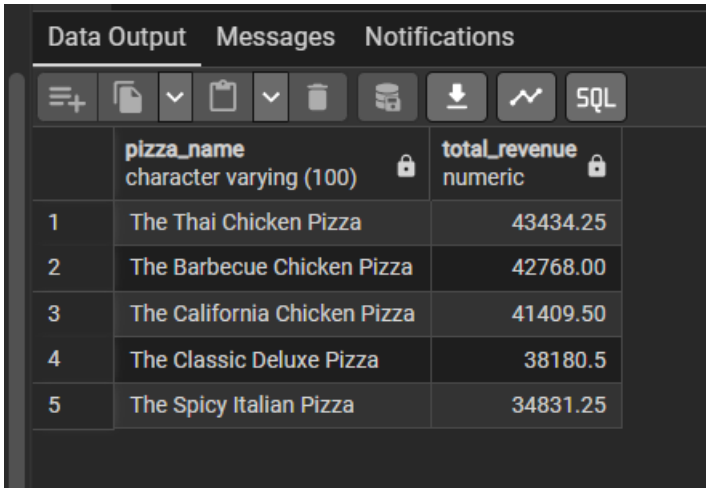
SQL

	pizza_category character varying (100)	total_sales numeric	pct_of_sales numeric
1	Classic	220053.10	26.91
2	Supreme	208197.00	25.46
3	Chicken	195919.50	23.96
4	Veggie	193690.45	23.68

Description: Shows each pizza category's sales contribution (percentage) to two decimal places, ordered by highest revenue.

10. Top 5 Pizzas by Revenue

```
SELECT pizza_name, SUM(total_price) AS Total_Revenue FROM pizza_sales
GROUP BY pizza_name
ORDER BY Total_revenue DESC
LIMIT 5
```

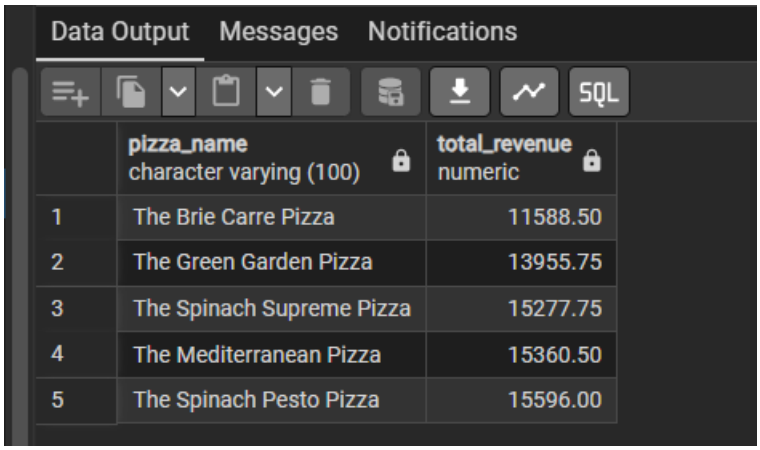


	pizza_name character varying (100)	total_revenue numeric
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768.00
3	The California Chicken Pizza	41409.50
4	The Classic Deluxe Pizza	38180.5
5	The Spicy Italian Pizza	34831.25

Description: Lists the top 5 pizzas that generated the highest revenue.

11. Bottom 5 Pizzas by Revenue

```
SELECT pizza_name, SUM(total_price) AS Total_Revenue FROM pizza_sales
GROUP BY pizza_name
ORDER BY Total_revenue ASC
LIMIT 5
```



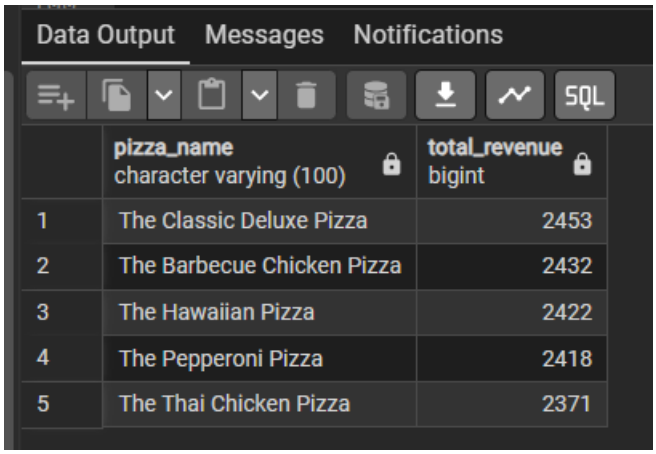
	pizza_name character varying (100)	total_revenue numeric
1	The Brie Carre Pizza	11588.50
2	The Green Garden Pizza	13955.75
3	The Spinach Supreme Pizza	15277.75
4	The Mediterranean Pizza	15360.50
5	The Spinach Pesto Pizza	15596.00

Description: Lists the bottom 5 pizzas that generated the least revenue.

12. Top 5 best selling pizza

```
SELECT pizza_name, SUM(quantity) AS Total_Revenue FROM pizza_sales
GROUP BY pizza_name
ORDER BY Total_revenue desc
```

LIMIT 5



	pizza_name character varying (100)	total_revenue 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

Description: Finds the top 5 pizzas with the highest sales volume (quantity sold).

Connecting SQL and Power BI

1. **Connecting PostgreSQL to Power BI**

   Installed PostgreSQL 17 and created the pizza_sales database.

   the dataset to .csv and imported it into PostgreSQL.

   Power BI: Get Data → More → Database → PostgreSQL database.

   Entered server name (localhost:5432) and database name.

   Logged in with PostgreSQL username and password.

   Power BI Reads from SQL

   Power BI queries PostgreSQL directly.

   SQL scripts aggregate, clean, and calculate metrics.

   Power BI visualizes the query results in dashboards.

   Learned how SQL provides structured calculations (totals, averages, percentages).

   Used PostgreSQL as the backbone for data preparation.

   Integrated SQL results into Power BI visuals for deeper business insights.