Project Title:
Sales and Performance Analysis Using SQL – Bike Store Database

By:
Muhammad Subhan
Aspiring Data Analyst

Tools Used:
PostgreSQL, SQL, CSV, pgAdmin

Date:
07-June-2025

📘Introduction

This project uses the Bike Store Sample Database from Kaggle to perform data analysis using PostgreSQL. The goal was to answer business-related questions about sales, customer behavior, staff performance, and store performance. The dataset includes tables such as `customers`, `orders`, `order_items`, `products`, `staffs`, and more. I created and populated all the necessary tables and then wrote 10 analytical SQL queries to generate insights that would help decision-makers in a retail bike store environment.

❓ Business Questions, SQL Queries, and Insights

1. **Which products generate the highest and lowest total sales revenue?**
   → Group by product or category, order by total sales.

```
SELECT
   OI.product_id,
   P.product_name,
   SUM(OI.quantity) AS total_quantity_sold,
   SUM(OI.quantity * OI.list_price) AS total_revenue
FROM order_items OI
JOIN orders O ON OI.order_id = O.order_id
JOIN products P ON OI.product_id = P.product_id
GROUP BY OI.product_id, P.product_name
ORDER BY total_revenue DESC; --use when show highest revenue
ORDER BY total_revenue ASC;  --use when show lowest revenue
```

Data Output | Messages | Notifications

Show

	product_id integer	product_name character varying (100)	total_quantity_sold bigint	total_revenue numeric
1	7	Trek Slash 8 27.5 - 2016	154	615998.46
2	9	Trek Conduit+ - 2016	145	434998.55
3	4	Trek Fuel EX 8 29 - 2016	143	414698.57
4	11	Surly Straggler 650b - 2016	151	253829.49
5	56	Trek Domane SLR 6 Disc - 2017	43	236499.57
6	10	Surly Straggler - 2016	147	227703
7	8	Trek Remedy 29 Carbon Frameset - 2016	125	224998.75
8	61	Trek Powerfly 8 FS Plus - 2017	41	204999.59
9	58	Trek Madone 9.2 - 2017	39	194999.61
10	51	Trek Silque SLR 8 Women's - 2017	29	188499.71

Total rows: 307 Query complete 00:00:00.130

Data Output | Messages | Notifications

Showing

	product_id integer	product_name character varying (100)	total_quantity_sold bigint	total_revenue numeric
1	270	Trek Precaliber 16 Boy's - 2018	1	209.99
2	262	Trek MT 201 - 2018	1	249.99
3	222	Electra Cruiser 1 Tall - 2016/2018	1	269.99
4	285	Electra Soft Serve 1 (16-inch) - Girl's - 2018	1	279.99
5	287	Electra Straight 8 1 (16-inch) - Boy's - 2018	1	279.99
6	273	Trek Precaliber 20 6-speed Girl's - 2018	1	289.99
7	218	Electra Cruiser 7D - 2016/2017/2018	1	319.99
8	279	Trek Precaliber 24 7-speed Girl's - 2018	1	319.99
9	290	Electra Superbolt 3i 20" - 2018	1	369.99
10	294	Electra Tiger Shark 3i (20-inch) - Boys' - 2018	1	369.99

Total rows: 307 Query complete 00:00:00.084

2. **What are the top 5 selling product categories in terms of quantity and revenue?**

   → Use aggregation + ranking (window functions or `LIMIT`).

Select C.category_id, C.category_name,

```
        SUM(OI.quantity) As Total_quantity_sold,
        Sum(OI.quantity * OI.list_price) As Total_revenue
From order_items OI
Join orders O on O.order_id = OI.order_id
Join products P on OI.product_id = P.product_id
Join categories C on P.category_id = C.category_id
Group by C.category_id, C.category_name
Order by Total_revenue Desc
Limit 5;
```

Data Output | Messages | Notifications

	category_id [PK] integer	category_name character varying (50)	total_quantity_sold bigint	total_revenue numeric
1	6	Mountain Bikes	1755	3030775.71
2	7	Road Bikes	559	1852555.60
3	3	Cruisers Bicycles	2063	1109151.04
4	5	Electric Bikes	315	1020236.85
5	4	Cyclocross Bicycles	394	799874.60

3. **Which customers are the top buyers in terms of total purchase value?**
   → Join customers and sales data, group by customer, use SUM and RANK.

```
Select C.customer_id, C.first_name,
        Sum(OI.list_price*OI.quantity) as Total_purchase
From order_items OI
Join orders O  on O.order_id = OI.order_id
Join customers C on O.customer_id = C.customer_id
Group by C.customer_id,C.first_name
```

Order by Total_purchase Desc;

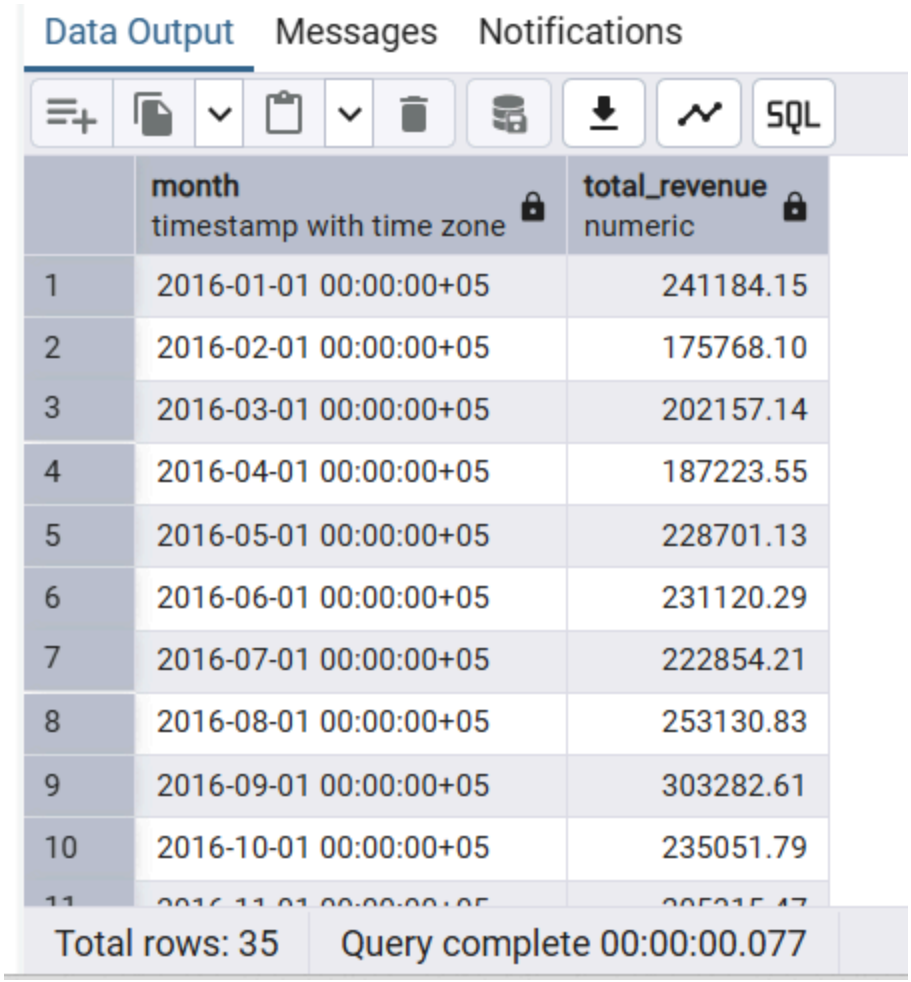
Data Output | Messages | Notifications

	customer_id [PK] integer	first_name character varying (50)	total_purchase numeric
1	10	Pamelia	37801.84
2	75	Abby	37500.89
3	94	Sharyn	37138.86
4	6	Lyndsey	35857.86
5	16	Emmitt	34503.82
6	73	Melanie	34390.88
7	1	Debra	30645.87
8	61	Elinore	29661.83
9	93	Corrina	29214.89
10	122	Shena	27618.95

Total rows: 1445 | Query complete 00:00:00.148

4. **What are the monthly sales trends over the past year?**
   → Use date functions and grouping by MONTH.

```
Select
        DATE_TRUNC('month',O.order_date) as month,
        Sum(OI.list_price * OI.quantity) as Total_Revenue
From order_items OI
Join Orders O on O.order_id = OI.order_id
group by month
order by month;
```



Data Output | Messages | Notifications

	month timestamp with time zone	total_revenue numeric
1	2016-01-01 00:00:00+05	241184.15
2	2016-02-01 00:00:00+05	175768.10
3	2016-03-01 00:00:00+05	202157.14
4	2016-04-01 00:00:00+05	187223.55
5	2016-05-01 00:00:00+05	228701.13
6	2016-06-01 00:00:00+05	231120.29
7	2016-07-01 00:00:00+05	222854.21
8	2016-08-01 00:00:00+05	253130.83
9	2016-09-01 00:00:00+05	303282.61
10	2016-10-01 00:00:00+05	235051.79

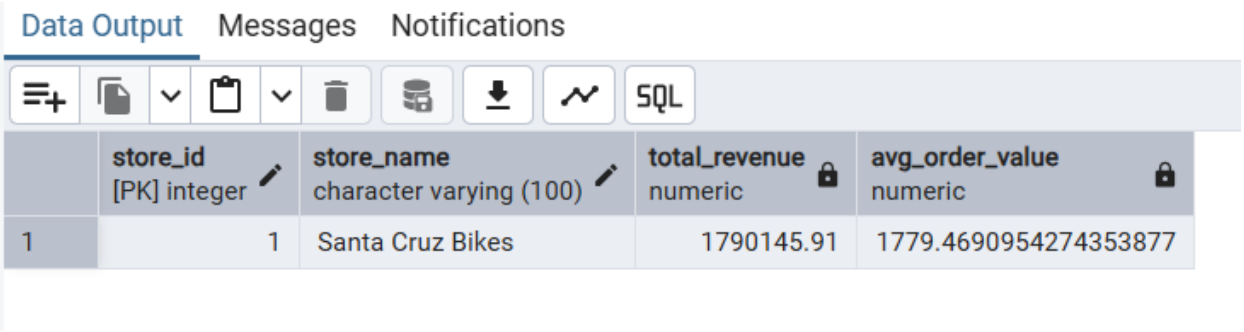
Total rows: 35 Query complete 00:00:00.077

5. **Which stores are underperforming based on total revenue or average order value?**

→ Group by store, calculate total and average order values.

```
Select S.store_id, S.store_name,
        Sum(OI.list_price * OI.quantity) as Total_Revenue,
        Avg(OI.list_price * OI.quantity) as Avg_Order_Value
From order_items OI
Join Orders O on O.order_id = OI.order_id
Join stores S on S.store_id = O.store_id
group by S.store_id, S.store_name
HAVING
   SUM(OI.list_price * OI.quantity) < 100000 -- threshold for total revenue (can be adjusted)
   OR AVG(OI.list_price * OI.quantity) < 1800  -- threshold for average order value (can be adjusted)
```

Order by Total_Revenue Desc;

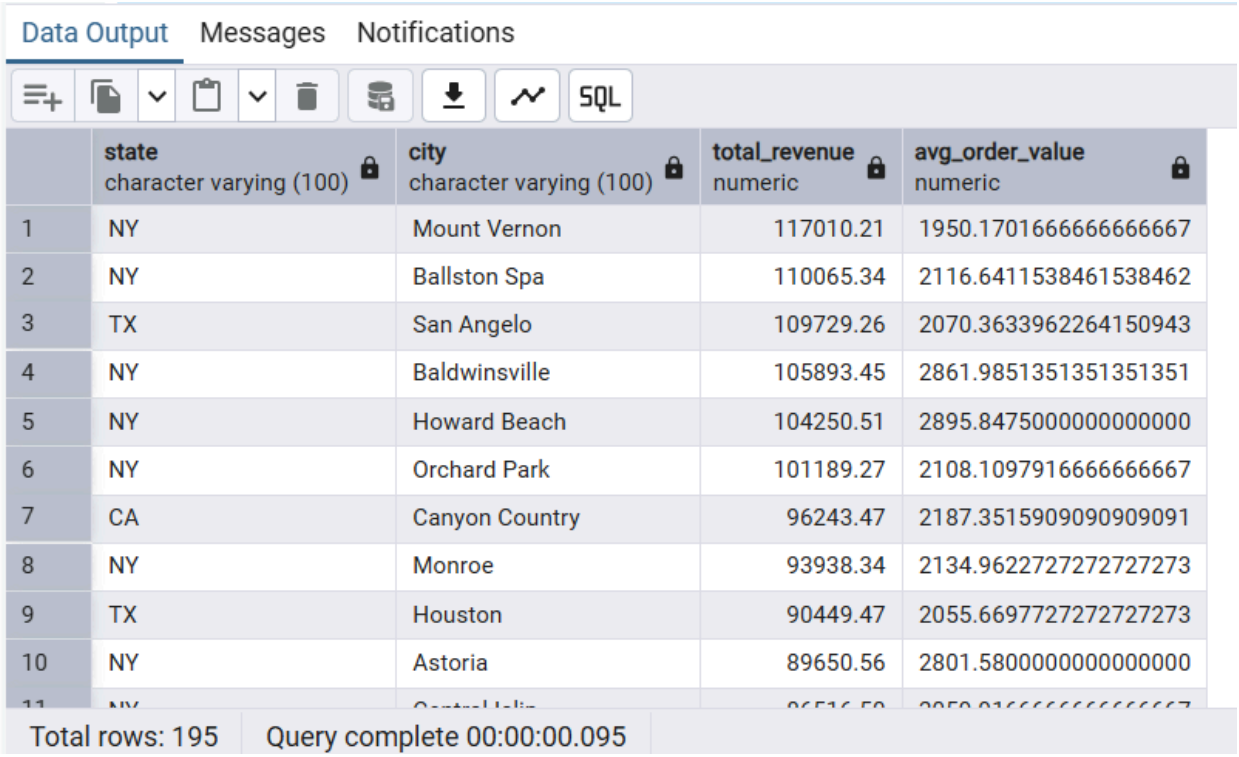


Data Output | Messages | Notifications

	store_id [PK] integer	store_name character varying (100)	total_revenue numeric	avg_order_value numeric
1	1	Santa Cruz Bikes	1790145.91	1779.4690954274353877

6. **How do different customer demographics (e.g., state, city, gender) affect purchase behavior?**
   → Group by customer location or gender, analyze average and total sales.

```
Select C.State, C.city,
        Sum(OI.list_price * OI.quantity) as Total_Revenue,
        Avg(OI.list_price * OI.quantity) as Avg_Order_Value
From order_items OI
Join Orders O on O.order_id = OI.order_id
Join customers C on C.customer_id = O.customer_id
group by C.State, C.city
order by Total_Revenue Desc;
```



Data Output | Messages | Notifications

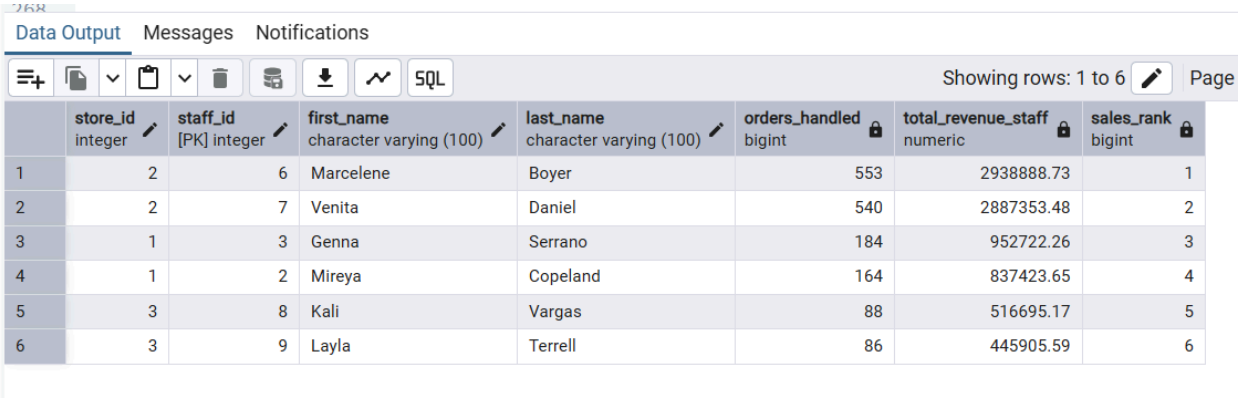
	state character varying (100)	city character varying (100)	total_revenue numeric	avg_order_value numeric
1	NY	Mount Vernon	117010.21	1950.1701666666666667
2	NY	Ballston Spa	110065.34	2116.6411538461538462
3	TX	San Angelo	109729.26	2070.3633962264150943
4	NY	Baldwinsville	105893.45	2861.9851351351351351
5	NY	Howard Beach	104250.51	2895.8475000000000000
6	NY	Orchard Park	101189.27	2108.1097916666666667
7	CA	Canyon Country	96243.47	2187.3515909090909091
8	NY	Monroe	93938.34	2134.9622727272727273
9	TX	Houston	90449.47	2055.6697727272727273
10	NY	Astoria	89650.56	2801.5800000000000000

Total rows: 195 | Query complete 00:00:00.095

7. **Which staff members are generating the most sales?**
   → Join staff and sales tables, group and rank.

```
Select * from order_items;
Select * from orders;
select * from staffs;

Select S.store_id, S.staff_id, S.first_name,S.last_name,
   COUNT(DISTINCT O.order_id) AS Orders_Handled,
   SUM(OI.list_price * OI.quantity) AS Total_Revenue_Staff,
   RANK() OVER (ORDER BY SUM(OI.list_price * OI.quantity) DESC) AS Sales_Rank
from order_items OI
Join Orders O on O.order_id = OI.order_id
Join Staffs S on O.staff_id = S.staff_id
group by S.store_id, S.staff_id, S.first_name,S.last_name
order by Total_Revenue_Staff Desc;
```



Data Output | Messages | Notifications

Showing rows: 1 to 6 | Page

	store_id integer	staff_id [PK] integer	first_name character varying (100)	last_name character varying (100)	orders_handled bigint	total_revenue_staff numeric	sales_rank bigint
1	2	6	Marcelene	Boyer	553	2938888.73	1
2	2	7	Venita	Daniel	540	2887353.48	2
3	1	3	Genna	Serrano	184	952722.26	3
4	1	2	Mireya	Copeland	164	837423.65	4
5	3	8	Kali	Vargas	88	516695.17	5
6	3	9	Layla	Terrell	86	445905.59	6

8. **What is the average order value by store and by product category?**
   → Join tables, group by store and category.

```
Select S.store_id,S.store_name,C.category_name,
        Avg(OI.quantity * OI.list_price) as Avg_order_value
from order_items OI
join orders O on O.order_id = OI.order_id
join products P on OI.product_id = P.product_id
join categories C on C.category_id = P.category_id
Join stores S on S.store_id = O.store_id
group by S.store_id,S.store_name,C.category_name
order by Avg_order_value Desc;
```

Data Output | Messages | Notifications

	store_id integer	store_name character varying (100)	category_name character varying (50)	avg_order_value numeric
1	2	Baldwin Bikes	Road Bikes	5082.2456400000000000
2	2	Baldwin Bikes	Electric Bikes	5056.1968181818181818
3	3	Rowlett Bikes	Road Bikes	5041.3600000000000000
4	1	Santa Cruz Bikes	Electric Bikes	4480.7127272727272727
5	1	Santa Cruz Bikes	Road Bikes	4473.8014473684210526
6	3	Rowlett Bikes	Electric Bikes	4255.1868000000000000
7	3	Rowlett Bikes	Cyclocross Bicycles	3248.1328571428571429
8	1	Santa Cruz Bikes	Cyclocross Bicycles	3156.8675438596491228
9	2	Baldwin Bikes	Cyclocross Bicycles	3099.5638202247191011
10	3	Rowlett Bikes	Mountain Bikes	2661.0079389312977099

Total rows: 21 | Query complete 00:00:00.093

9. **Are there any seasonal trends in product sales (e.g., certain categories doing better in certain months)?**
   → Use date filtering, group by month and category.

```
SELECT
        to_Char(date_trunc('month',O.order_date), 'Mon YYYY') as Month_year,
        C.category_id, C.category_name,
        SUM(OI.list_price * OI.quantity) AS Total_monthly_revenue
from order_items OI
Join orders O on O.order_id = OI.order_id
Join products P on OI.product_id = P.product_id
Join categories C on P.category_id = C.category_id
Group by Month_year, C.category_id, C.category_name
Order by Total_monthly_revenue Desc;
```

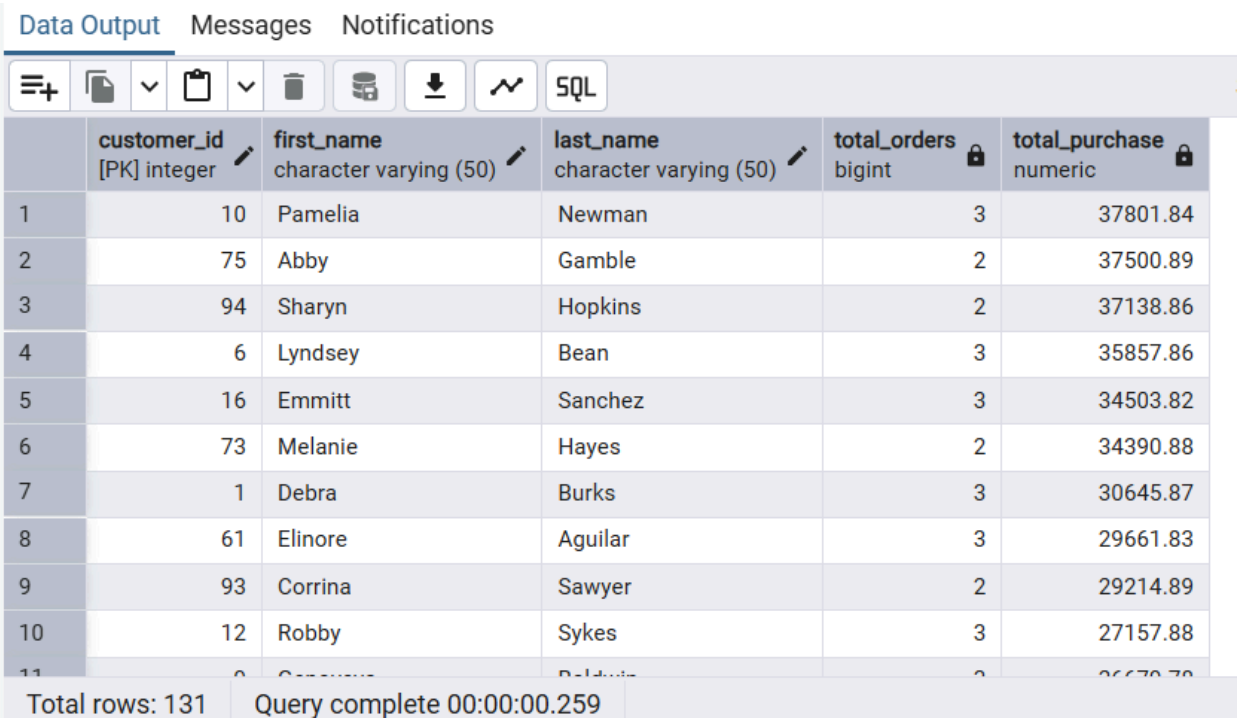
Data Output | Messages | Notifications

	month_year text	category_id [PK] integer	category_name character varying (50)	total_monthly_revenue numeric
1	Apr 2018	7	Road Bikes	307362.14
2	Apr 2018	6	Mountain Bikes	182914.04
3	Sep 2016	6	Mountain Bikes	178188.03
4	Jan 2018	7	Road Bikes	177505.50
5	Apr 2018	5	Electric Bikes	175139.46
6	Jun 2017	7	Road Bikes	146579.61
7	Mar 2017	6	Mountain Bikes	145484.13
8	Oct 2017	6	Mountain Bikes	130410.31
9	Aug 2016	6	Mountain Bikes	129138.30
10	May 2016	6	Mountain Bikes	127717.29

Total rows: 207 Query complete 00:00:00.130

10. **What is the repeat customer rate, and who are the most loyal customers?**
→ Count distinct invoices per customer, identify those with multiple purchases.

```
Select
        C.Customer_id, C.first_name, C.last_name,
        COUNT(DISTINCT O.order_id) AS total_orders,
        sum (OI.quantity*OI.list_price) as total_purchase
from order_items OI
join orders O on O.order_id = OI.order_id
Join customers C on C.customer_id = O.customer_id
group by C.Customer_id, C.first_name, C.last_name
HAVING COUNT(DISTINCT O.order_id) > 1
order by total_purchase Desc;
```



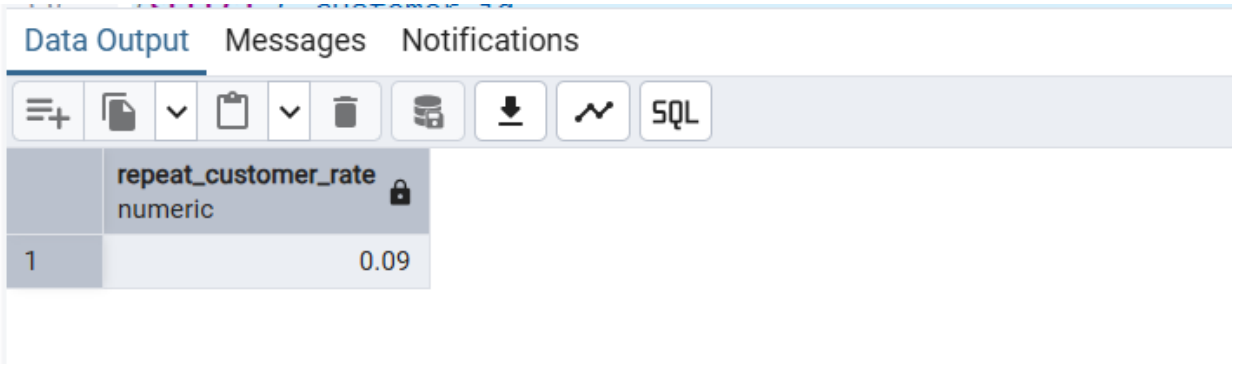
Data Output | Messages | Notifications

	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)	total_orders bigint	total_purchase numeric
1	10	Pamelia	Newman	3	37801.84
2	75	Abby	Gamble	2	37500.89
3	94	Sharyn	Hopkins	2	37138.86
4	6	Lyndsey	Bean	3	35857.86
5	16	Emmitt	Sanchez	3	34503.82
6	73	Melanie	Hayes	2	34390.88
7	1	Debra	Burks	3	30645.87
8	61	Elinore	Aguilar	3	29661.83
9	93	Corrina	Sawyer	2	29214.89
10	12	Robby	Sykes	3	27157.88

Total rows: 131 Query complete 00:00:00.259

```
SELECT
   ROUND(
      COUNT(DISTINCT CASE WHEN order_count > 1 THEN customer_id END)::DECIMAL
      / COUNT(DISTINCT customer_id), 2
   ) AS repeat_customer_rate
FROM

(SELECT C.customer_id,
   COUNT(DISTINCT O.order_id) AS order_count
   FROM customers C
   JOIN orders O ON O.customer_id = C.customer_id
   GROUP BY C.customer_id
) AS customer_orders;
```


Data Output
Messages
Notifications
SQL
repeat_customer_rate
numeric
1
0.09