A. Srivarshan

Zomato Sales Analytics Project Using SQL – by [Srivarshan]

Zomato's Restaurant Order Analysis

Project Overview:

Objective: To analyze Zomato's restaurant order data using SQL to uncover customer preferences, popular items, peak order times, and regional performance for actionable business insights.

Tools Used: POSTGRESQL, EXCEL, CSV Files

Focus Areas:

- **Order Volume and Timing**
- **Most Ordered Cuisines and items**
- **Revenue and Sales Patterns**
- **Cuisine and restaurant performance by region**
- **Top customers by order frequency**

A. Srivarshan

Dataset Summary:

orders.csv	order_id, order_date, order_time, customer_id,
city_idorder_details.csv	order_id, menu_item_id, quantity,
pricemenu_items.csv	menu_item_id, item_name, cuisine_type,
pricecustomers.csv	customer_id, name, age, gender,
city_idrestaurants.csv	restaurant_id, restaurant_name, city_id,
ratingcities.csv	city_id, city_name, state, region

A. Srivarshan

✅ Basic SQL Analysis

- Total Orders Placed -> SELECT COUNT(*) FROM orders;
- Total Revenue Generated -> SELECT SUM(quantity * price) AS total_revenue FROM order_details;
- Highest Priced Menu Item -> Join order_details and calculate: -> SELECT * FROM menu_items ORDER BY price DESC LIMIT 1;
- Most Common Cuisine Type -> Join order_details → menu_items, then: -> GROUP BY cuisine_type ORDER BY SUM(quantity) DESC
- Top 5 Most Ordered Menu Items -> Join order_details → menu_items -> `GROUP BY item_name ORDER BY SUM(quantity) DESC LIMIT 5

A. Srivarshan

Intermediate SQL Analysis

- Average Order Value per Customer
- Join orders + order_details: -> SUM(price * quantity) / COUNT(DISTINCT customer_id)
- Orders by City and Restaurant -> Join orders → cities and restaurants: -> GROUP BY city_name, restaurant_name
- Orders by Day of Week and Month -> Extract day/month from order_date: -> EXTRACT(DAYOFWEEK FROM order_date), EXTRACT(MONTH FROM order_date)
- Repeat vs New Customer Orders -> Count number of orders per customer: -> HAVING COUNT(order_id) > 1 → Repeat ->
- Top 5 Customers by Orders frequency
- Top 3 Cuisines by Revenue -> Join order_details → menu_items -> GROUP BY cuisine_type ORDER BY SUM(quantity * price) DESC LIMIT 3

📈 Advanced SQL Analysis

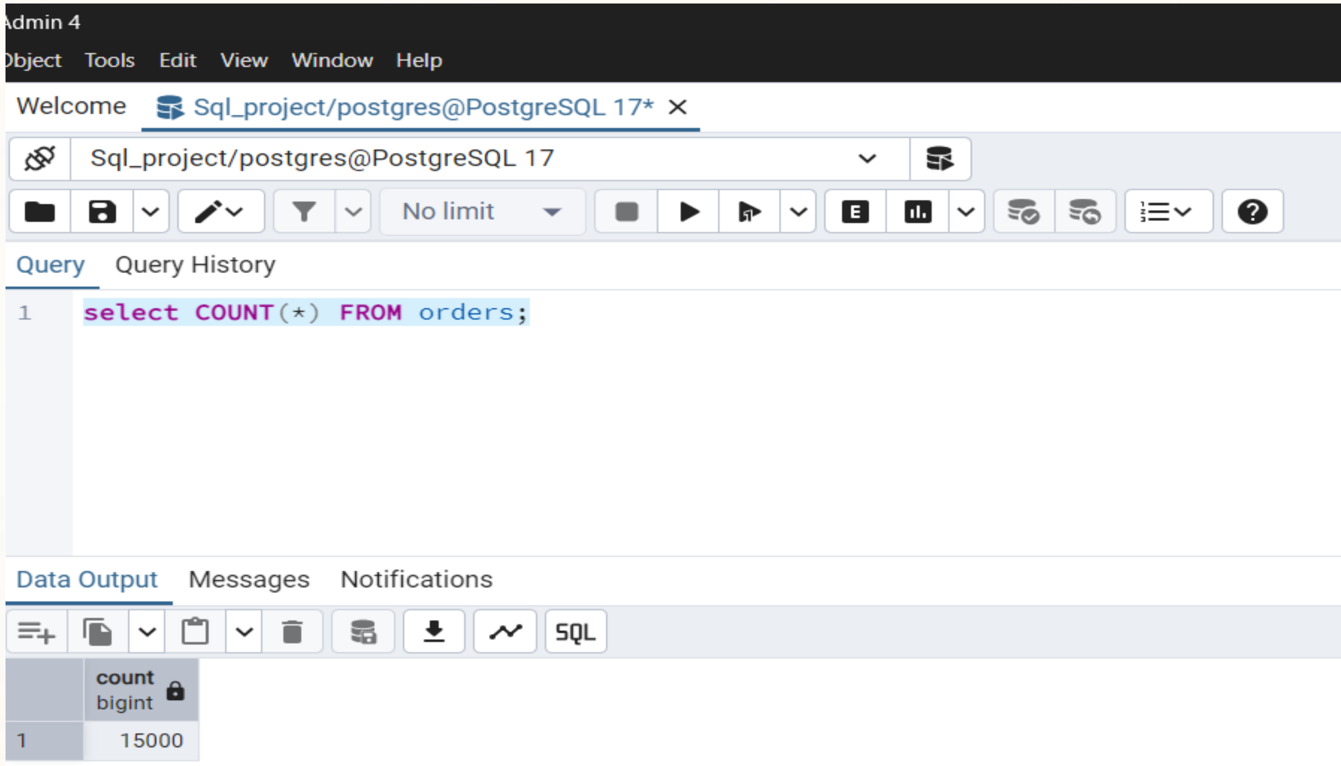
- % Revenue Contribution by Cuisine - > (SUM(price * quantity) for cuisine) / (Total Revenue) * 100
- Cumulative Revenue Over Time -> Use window function: -> SUM(quantity * price) OVER (ORDER BY order_date)
- Top Menu Items in Each Cuisine Category -> Use window function RANK() over PARTITION BY cuisine_type ORDER BY revenue DESC
- Customer Cohort Analysis (by first order month) -> MIN(order_date) per customer → Group by first month → Count
- City/Region-Wise Peak Order Patterns -> join orders → cities → extract hour from order_time -> GROUP BY region, HOUR(order_time)

A. Srivarshan

Total Orders Placed

QUERY:

SELECT COUNT(*) FROM orders;


Admin 4
Object Tools Edit View Window Help
Welcome
Sql_project/postgres@PostgreSQL 17*
Sql_project/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications
SQL

```
select COUNT(*) FROM orders;
```

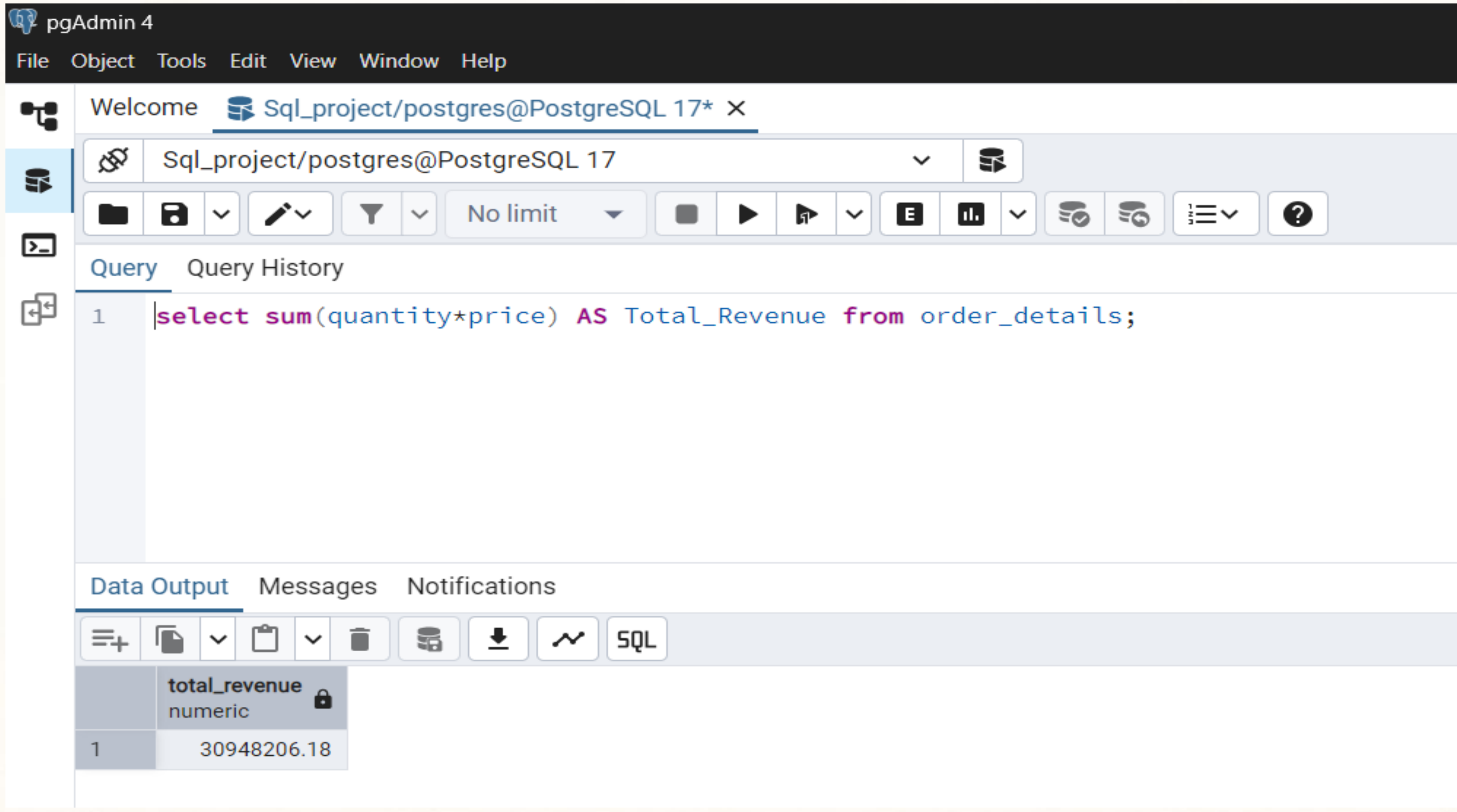
```
  count
  bigint
1 15000
```

A. Srivarshan

Total Revenue Generated

QUERY:

SELECT SUM(quantity*price) AS Total_Revenue FROM order_details;


pgAdmin 4
File Object Tools Edit View Window Help
Welcome
Sql_project/postgres@PostgreSQL 17*
Sql_project/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications
SQL

```
select sum(quantity*price) AS Total_Revenue from order_details;
```

```
   total_revenue
   numeric
1  30948206.18
```

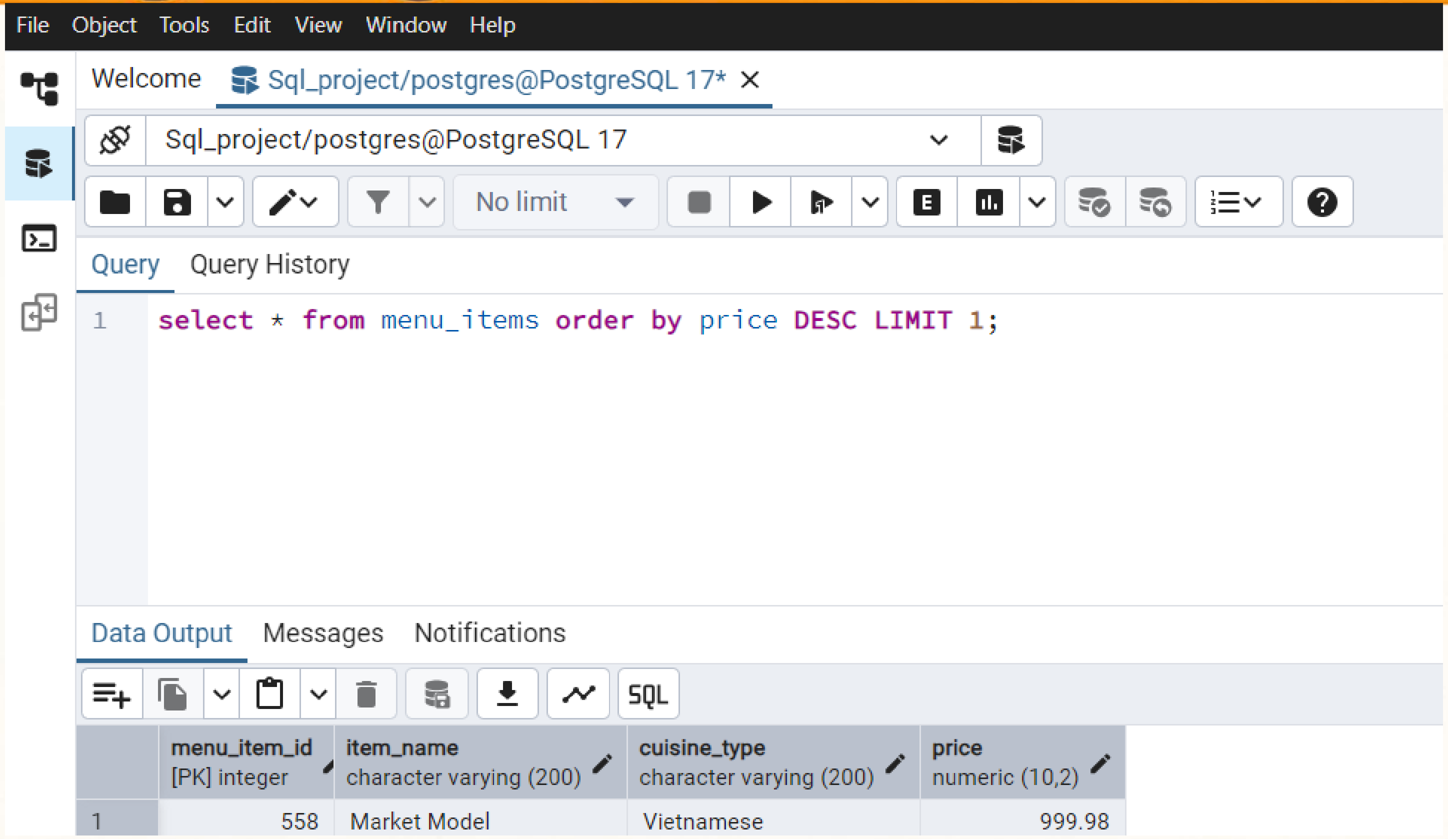
A. Srivarshan

Highest Priced Menu Item

QUERY:

SELECT * FROM menu_items

ORDER BY price DESC LIMIT 1;


File Object Tools Edit View Window Help
Welcome
Sql_project/postgres@PostgreSQL 17*
Sql_project/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications

```
select * from menu_items order by price DESC LIMIT 1;
```

	menu_item_id [PK] integer	item_name character varying (200)	cuisine_type character varying (200)	price numeric (10,2)
1	558	Market Model	Vietnamese	999.98

A. Srivarshan

Most Common Cuisine Type

QUERY:

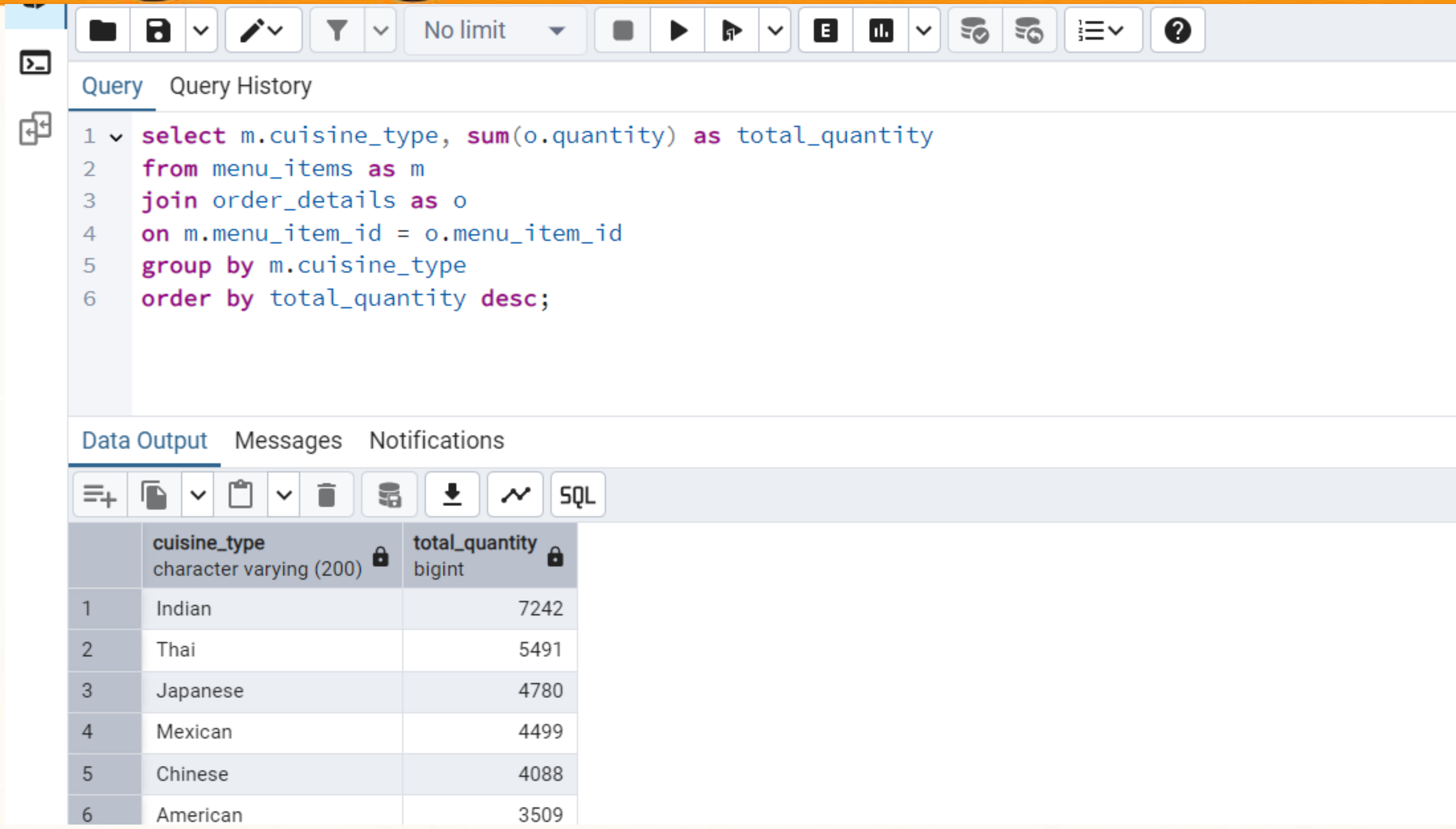
SELECT m.cuisine_type,
SUM(o.quantity) AS total_quantity
FROM menu_items AS m

JOIN order_detail AS o

ON m.menu_item_id = o.menu_item_id

GROUP BY m.cuisine_type

ORDER BY total_quantity DESC;



```
select m.cuisine_type, sum(o.quantity) as total_quantity
from menu_items as m
join order_details as o
on m.menu_item_id = o.menu_item_id
group by m.cuisine_type
order by total_quantity desc;
```

	cuisine_type character varying (200)	total_quantity bigint
1	Indian	7242
2	Thai	5491
3	Japanese	4780
4	Mexican	4499
5	Chinese	4088
6	American	3509

A. Srivarshan

Top 5 Most Ordered Menu Items

QUERY:

SELECT m.item_name, SUM(o.quantity) AS total_orders FROM menu_items as m

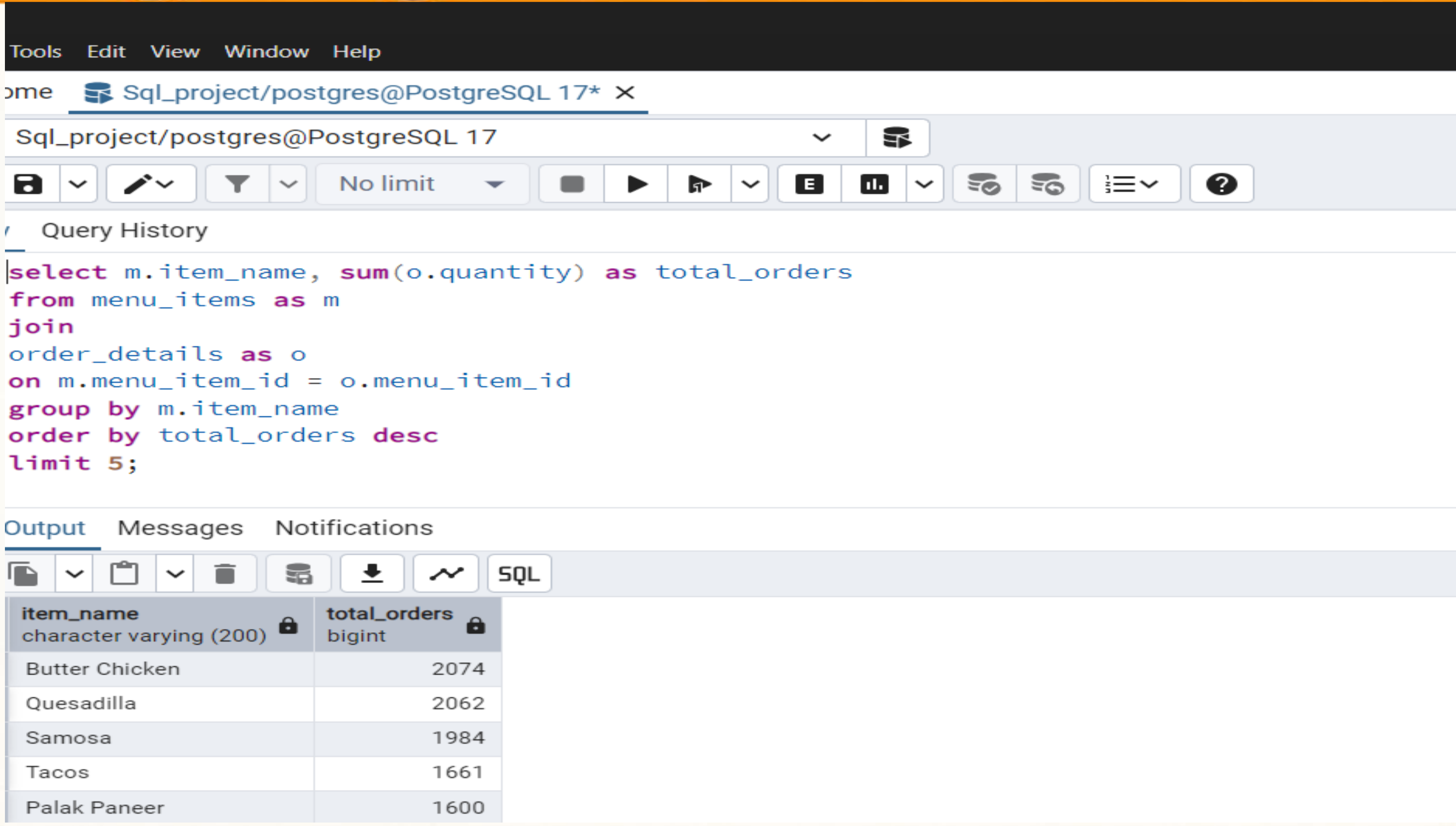
JOIN orders_details as o ON

m.menu_item_id = o.menu_item_id

GROUP BY m.item_name

ORDER BY total_orders DESC

LIMIT 5;



Tools Edit View Window Help
Sql_project/postgres@PostgreSQL 17*
Sql_project/postgres@PostgreSQL 17
No limit
Query History
Output Messages Notifications
SQL

```
select m.item_name, sum(o.quantity) as total_orders
from menu_items as m
join
order_details as o
on m.menu_item_id = o.menu_item_id
group by m.item_name
order by total_orders desc
limit 5;
```

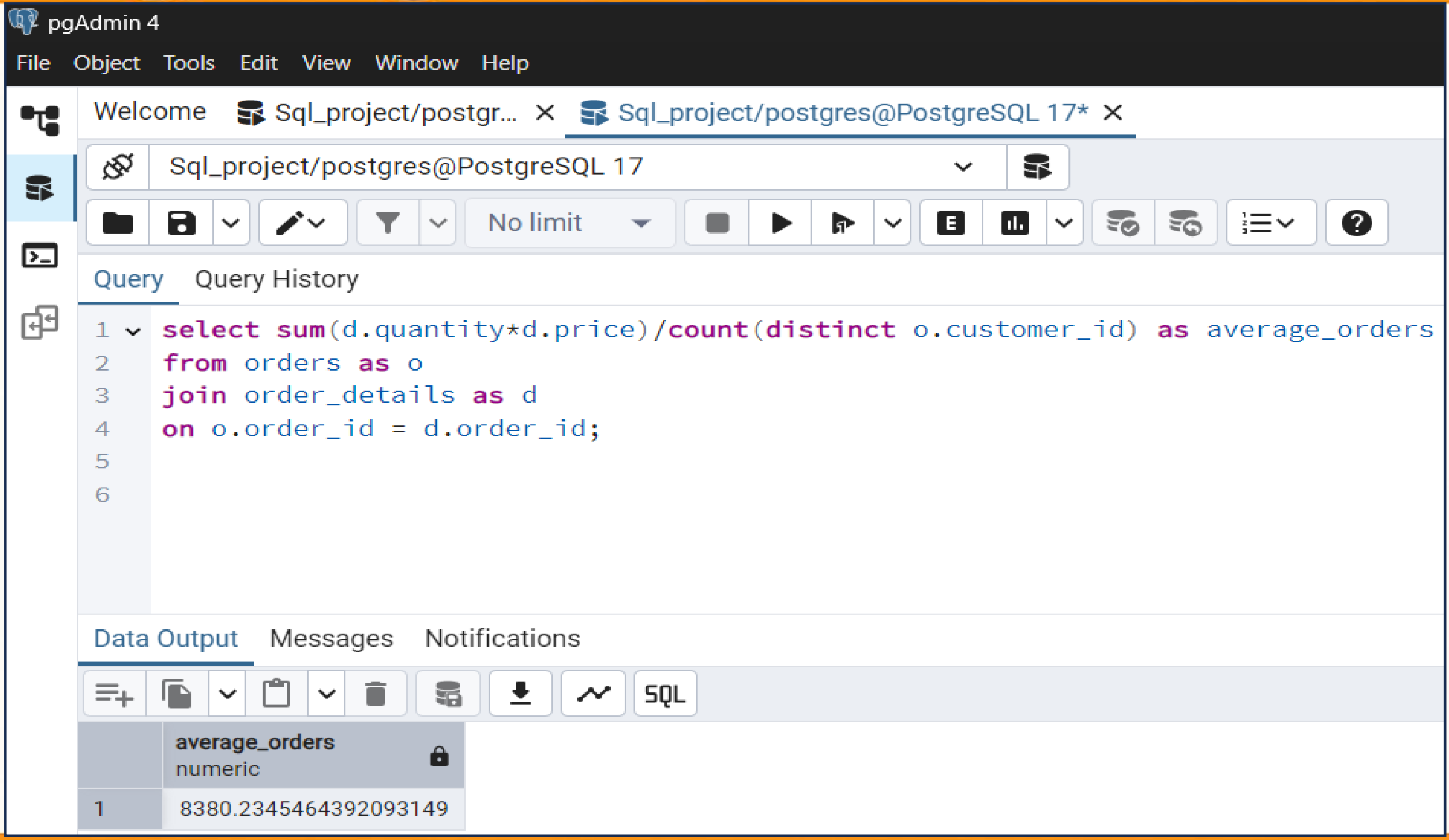
item_name character varying (200)	total_orders bigint
Butter Chicken	2074
Quesadilla	2062
Samosa	1984
Tacos	1661
Palak Paneer	1600

A. Srivarshan

Average Order Value per Customer

QUERY:

```
SELECT SUM(d.quantity*d.price) /
COUNT(DISTINCT o.customer_id) AS
average_orders

FROM orders AS o

JOIN order_details AS d

ON o.order_id = d.order_id;
```


pgAdmin 4
File Object Tools Edit View Window Help
Welcome
Sql_project/postgr...
Sql_project/postgres@PostgreSQL 17*
Sql_project/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications
SQL

```
select sum(d.quantity*d.price)/count(distinct o.customer_id) as average_orders
from orders as o
join order_details as d
on o.order_id = d.order_id;
```

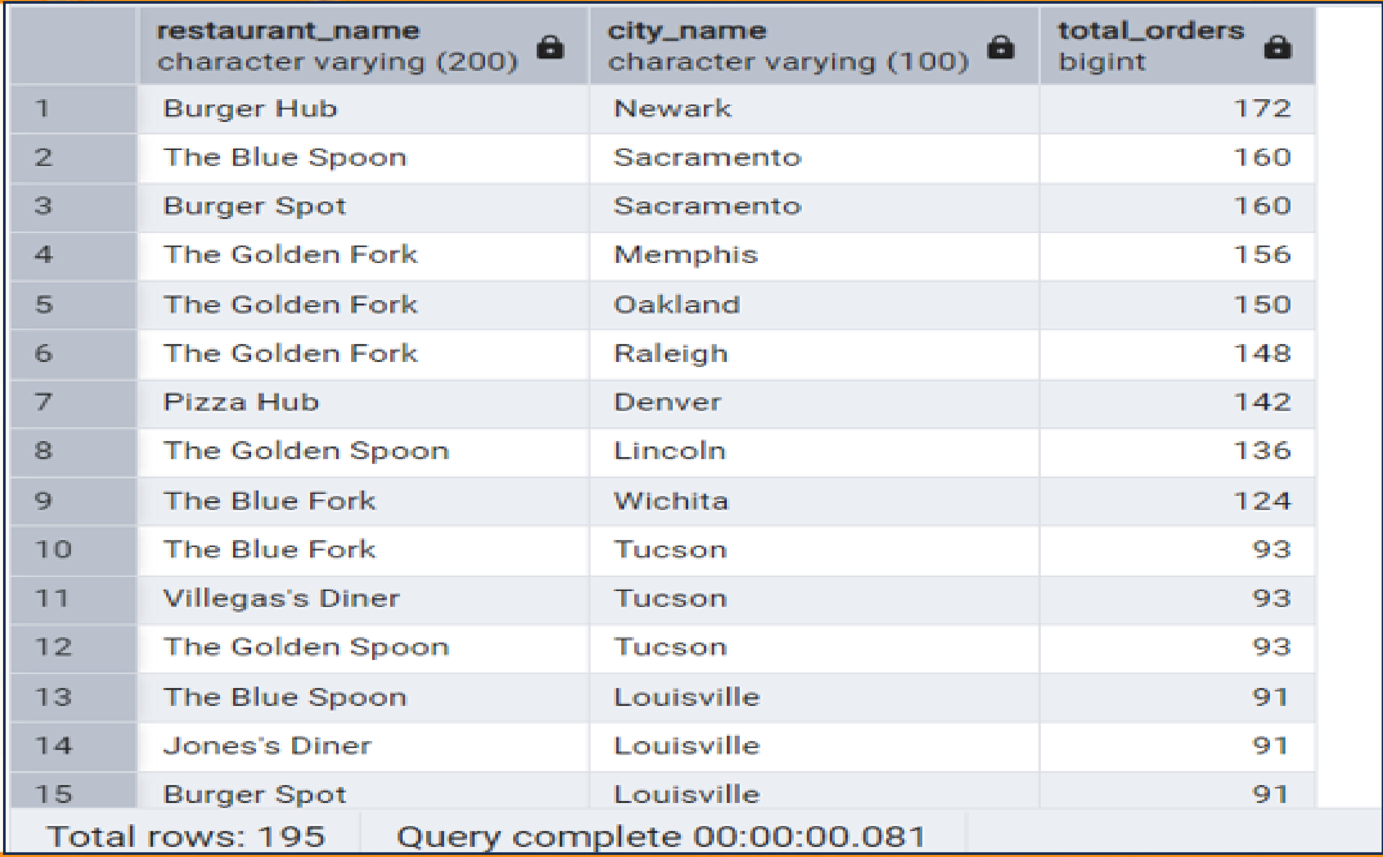
```
  average_orders
  numeric
1 8380.2345464392093149
```

A. Srivarshan

Orders by City and Restaurant

QUERY:

```
SELECT r.restaurant_name, c.city_name,
COUNT(o.order_id) AS total_orders

FROM orders AS o

JOIN restaurant as r

ON o.city_id = r.city_id

GROUP BY c.city_name, r.restaurant_name

ORDER BY total_orders DESC;
```



	restaurant_name character varying (200)	city_name character varying (100)	total_orders bigint
1	Burger Hub	Newark	172
2	The Blue Spoon	Sacramento	160
3	Burger Spot	Sacramento	160
4	The Golden Fork	Memphis	156
5	The Golden Fork	Oakland	150
6	The Golden Fork	Raleigh	148
7	Pizza Hub	Denver	142
8	The Golden Spoon	Lincoln	136
9	The Blue Fork	Wichita	124
10	The Blue Fork	Tucson	93
11	Villegas's Diner	Tucson	93
12	The Golden Spoon	Tucson	93
13	The Blue Spoon	Louisville	91
14	Jones's Diner	Louisville	91
15	Burger Spot	Louisville	91

Total rows: 195 Query complete 00:00:00.081

A. Srivarshan

Orders by Day of Week and Month

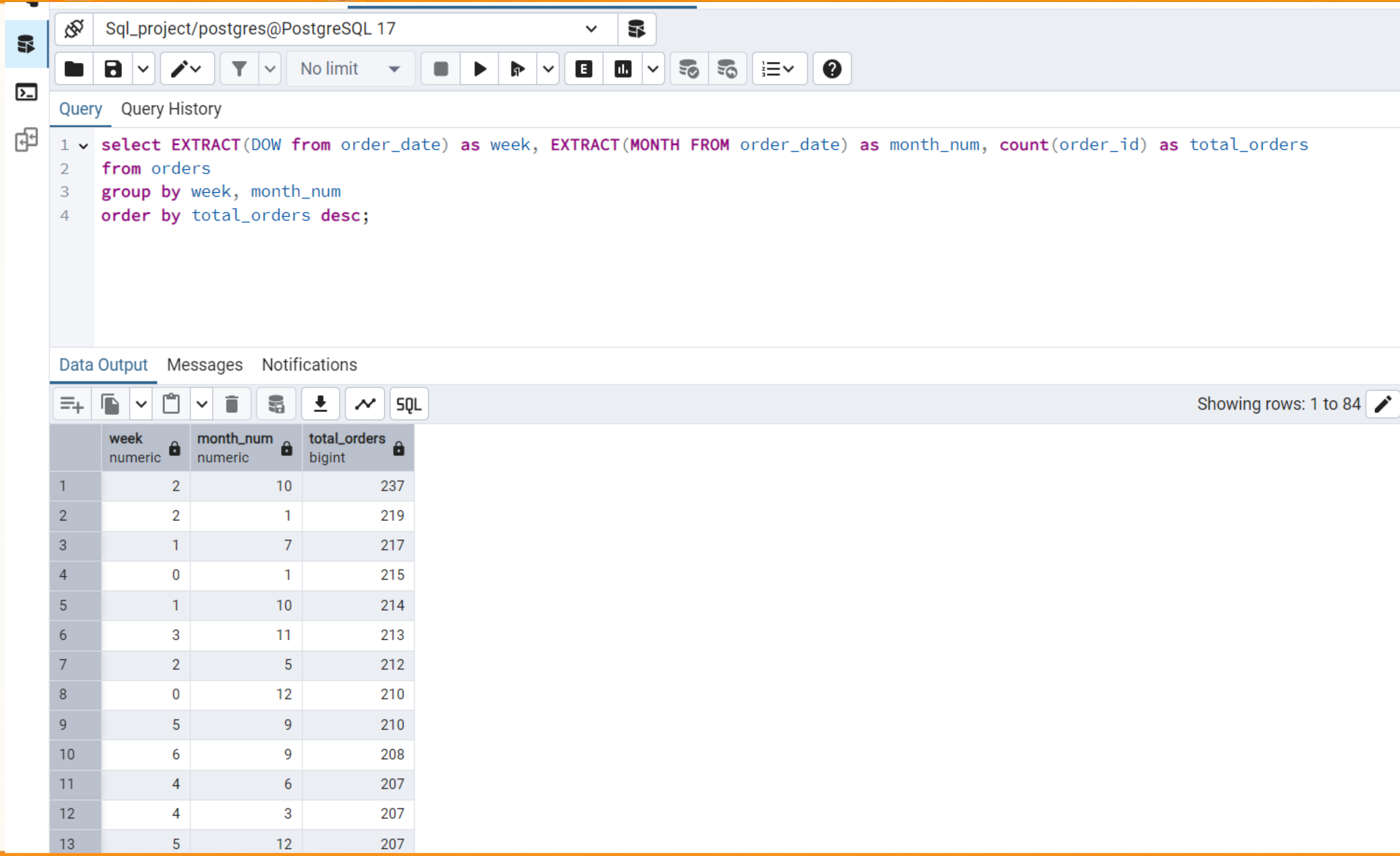
QUERY:

SELECT EXTRACT(DOW FROM order_date) AS week, EXTRACT(MONTH FROM order_date) AS month_num, COUNT(order_id) AS total_orders

FROM orders

GROUP BY week, month_num

ORDER BY total_orders DESC;



Sql_project/postgres@PostgreSQL 17

No limit

Query Query History

```
select EXTRACT(DOW from order_date) as week, EXTRACT(MONTH FROM order_date) as month_num, count(order_id) as total_orders
from orders
group by week, month_num
order by total_orders desc;
```

Data Output Messages Notifications

Showing rows: 1 to 84

	week numeric	month_num numeric	total_orders bigint
1	2	10	237
2	2	1	219
3	1	7	217
4	0	1	215
5	1	10	214
6	3	11	213
7	2	5	212
8	0	12	210
9	5	9	210
10	6	9	208
11	4	6	207
12	4	3	207
13	5	12	207

A. Srivarshan

Repeat vs New Customer Orders

QUERY:

SELECT customer_id, COUNT(order_id) AS total_orders,

CASE

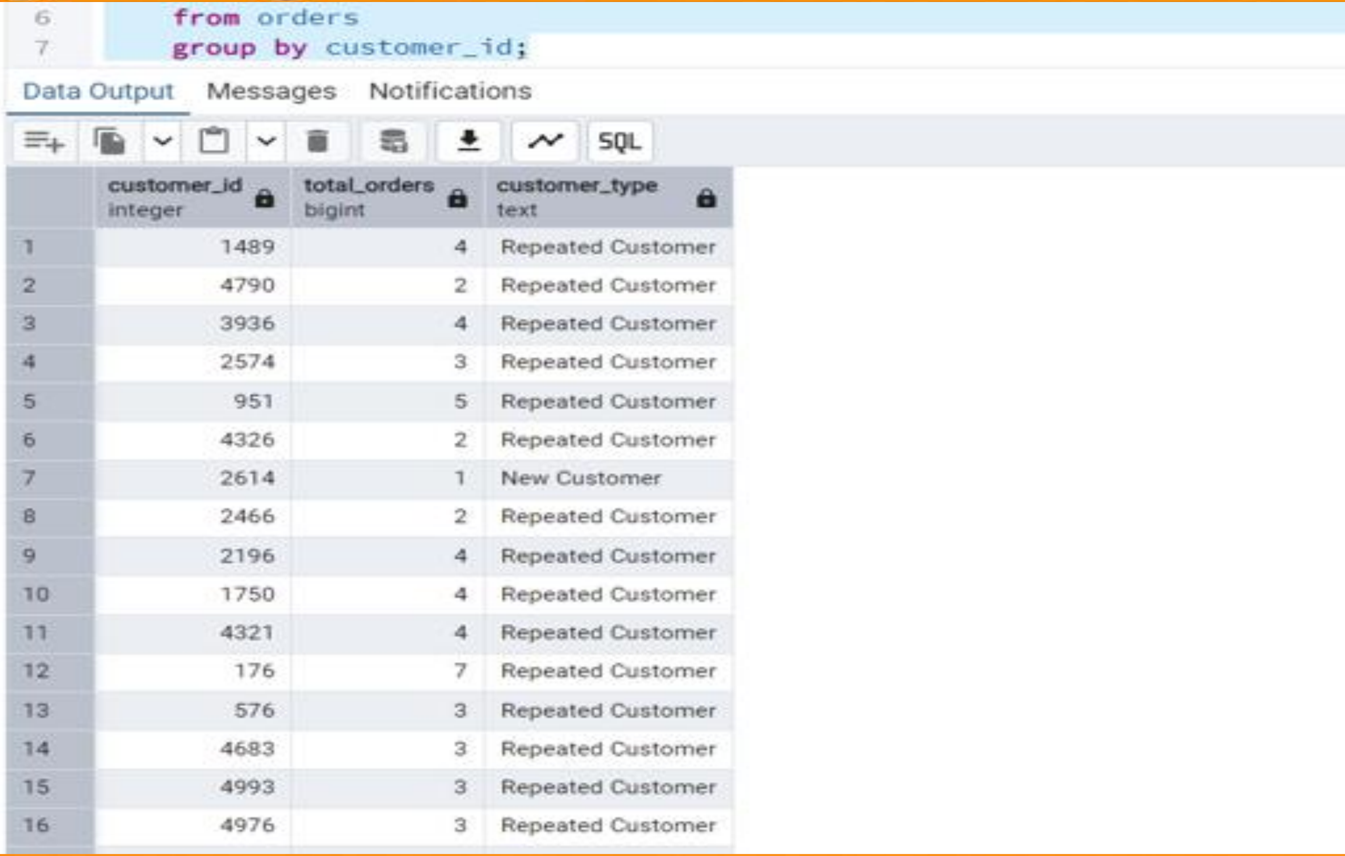
WHEN COUNT(order_id) > 1 THEN ‘Repeated Customer'

ELSE 'New Customer'

END AS customer_type

FROM orders

GROUP BY customer_id;



```
    from orders
    group by customer_id;
```

Data Output | Messages | Notifications

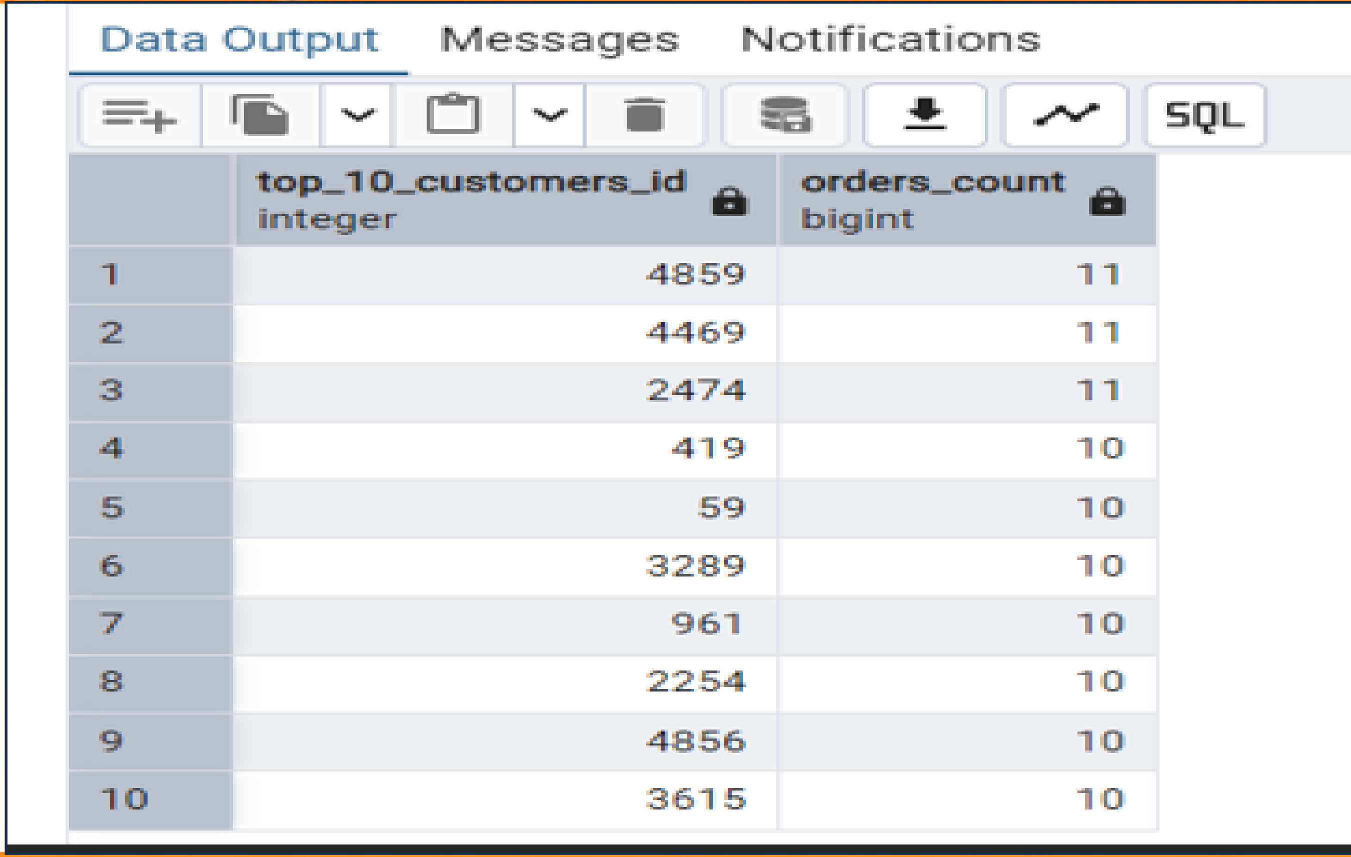
	customer_id integer	total_orders bigint	customer_type text
1	1489	4	Repeated Customer
2	4790	2	Repeated Customer
3	3936	4	Repeated Customer
4	2574	3	Repeated Customer
5	951	5	Repeated Customer
6	4326	2	Repeated Customer
7	2614	1	New Customer
8	2466	2	Repeated Customer
9	2196	4	Repeated Customer
10	1750	4	Repeated Customer
11	4321	4	Repeated Customer
12	176	7	Repeated Customer
13	576	3	Repeated Customer
14	4683	3	Repeated Customer
15	4993	3	Repeated Customer
16	4976	3	Repeated Customer

A. Srivarshan

Top 10 Customers With High Order Count

QUERY:

```
SELECT customer_id AS top_10_customers_id,
count(customer_id) AS orders_count

FROM orders

GROUP BY customer_id

ORDER BY orders_count DESC

LIMIT 10;
```



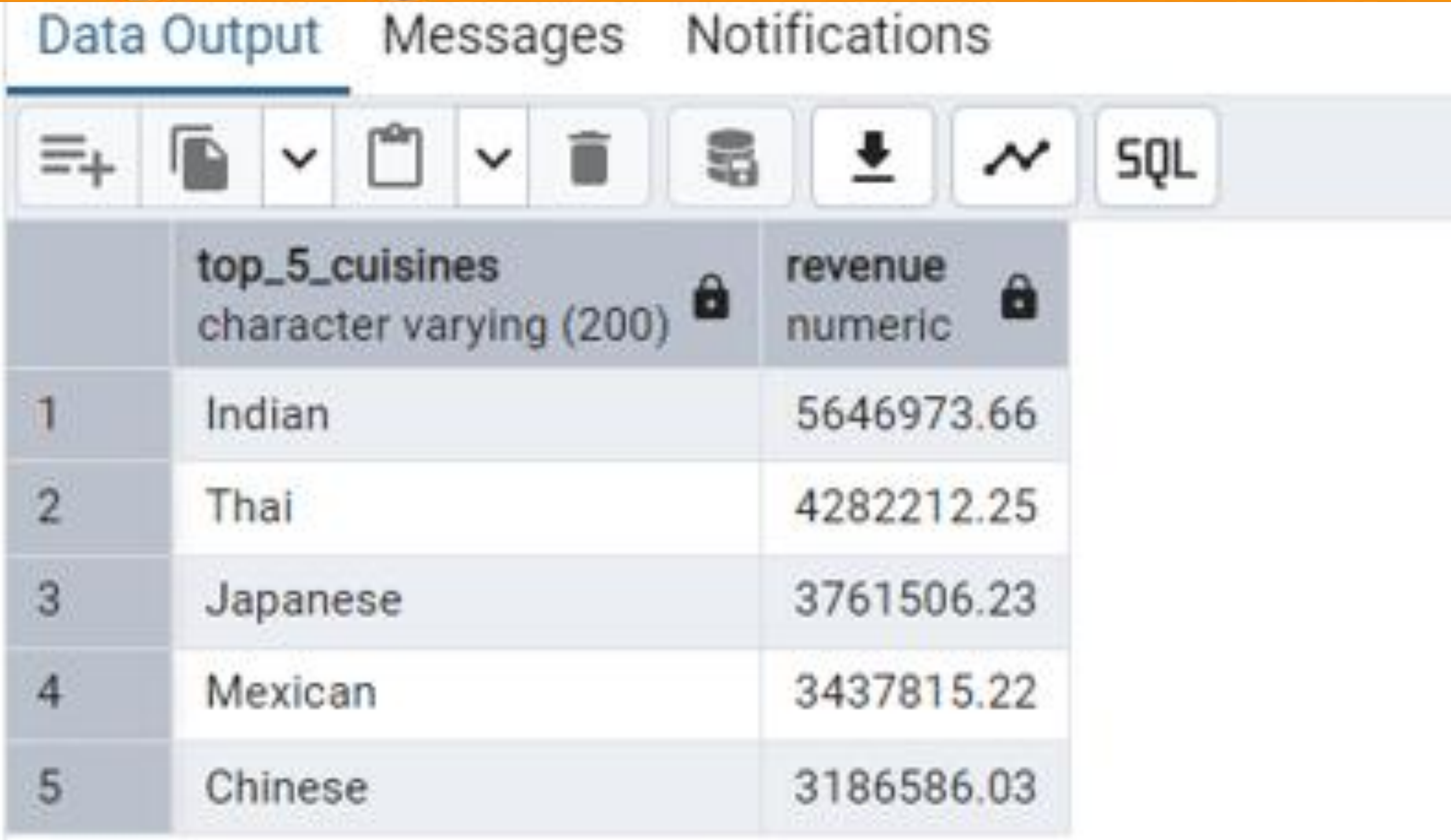
	top_10_customers_id integer	orders_count bigint
1	4859	11
2	4469	11
3	2474	11
4	419	10
5	59	10
6	3289	10
7	961	10
8	2254	10
9	4856	10
10	3615	10

A. Srivarshan

Top 5 Cuisines by Revenue

QUERY:

```
SELECT m.cuisine_type AS top_5_cuisines,
SUM(o.quantity*o.price) AS Revenue
FROM menu_items AS m
JOIN order_details AS o
ON m.menu_item_id = o.menu_item_id
GROUP BY m.cuisine_type
ORDER BY Revenue desc
limit 5;
```

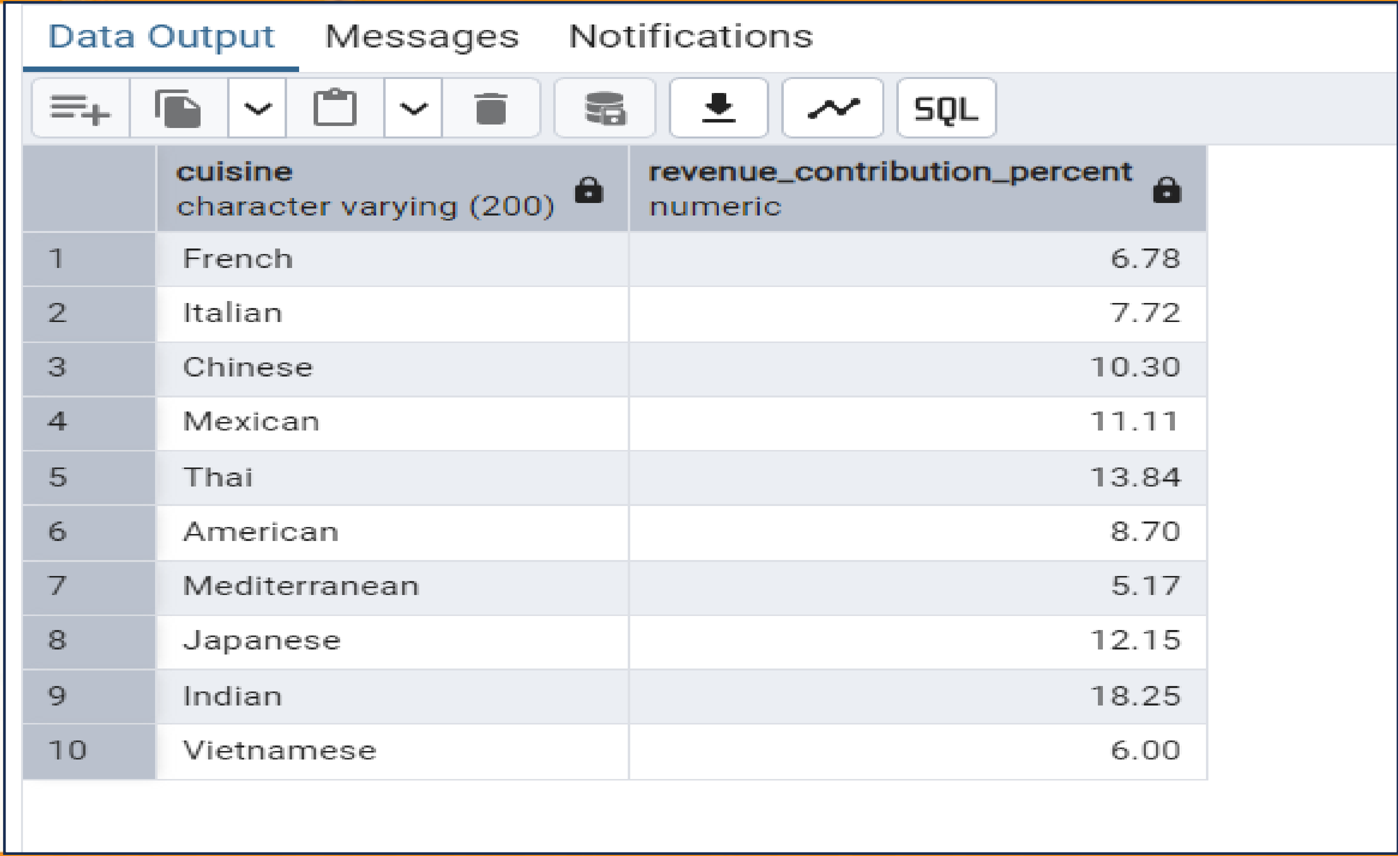


	top_5_cuisines character varying (200)	revenue numeric
1	Indian	5646973.66
2	Thai	4282212.25
3	Japanese	3761506.23
4	Mexican	3437815.22
5	Chinese	3186586.03

A. Srivarshan

% Revenue Contribution by Cuisine

```
SELECT m.cuisine_type AS Cuisine,ROUND(SUM(o.quantity * o.price) * 100.0 / SUM(SUM(o.quantity * o.price)) OVER (), 2) AS Revenue_Contribution_Percent

FROM order_details AS o

JOIN menu_items AS m ON o.menu_item_id = m.menu_item_id

GROUP BY m.cuisine_type;
```



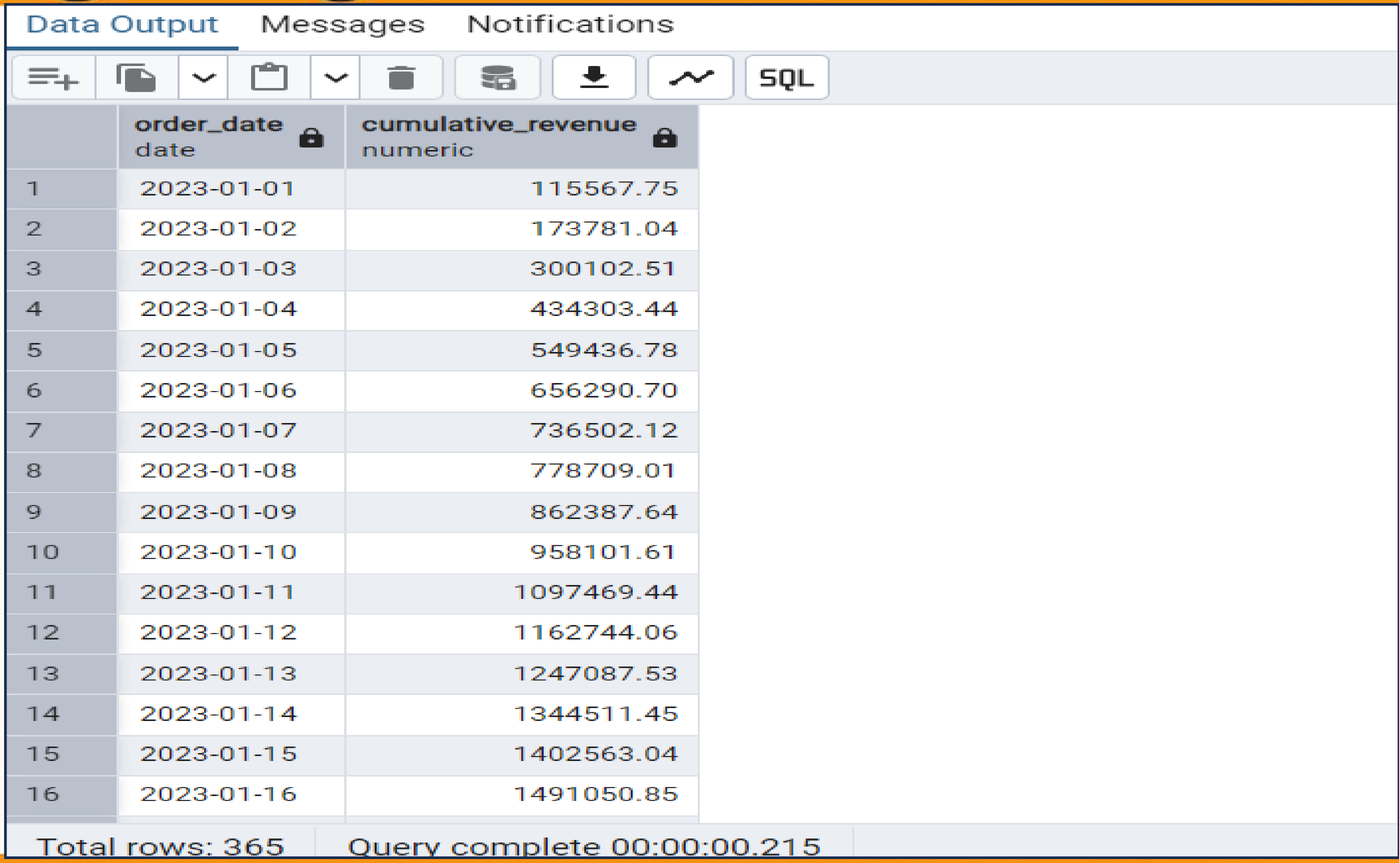
	cuisine character varying (200)	revenue_contribution_percent numeric
1	French	6.78
2	Italian	7.72
3	Chinese	10.30
4	Mexican	11.11
5	Thai	13.84
6	American	8.70
7	Mediterranean	5.17
8	Japanese	12.15
9	Indian	18.25
10	Vietnamese	6.00

A. Srivarshan

Cumulative Revenue Over Time

QUERY:

```
SELECT distinct o.order_date,
            SUM(d.quantity * d.price)  OVER
(order by o.order_date) AS
Cumulative_Revenue
FROM order_details AS d
JOIN
orders AS o
ON d.order_id = o.order_id
ORDER BY o.order_date;
```



Data Output | Messages | Notifications

	order_date date	cumulative_revenue numeric
1	2023-01-01	115567.75
2	2023-01-02	173781.04
3	2023-01-03	300102.51
4	2023-01-04	434303.44
5	2023-01-05	549436.78
6	2023-01-06	656290.70
7	2023-01-07	736502.12
8	2023-01-08	778709.01
9	2023-01-09	862387.64
10	2023-01-10	958101.61
11	2023-01-11	1097469.44
12	2023-01-12	1162744.06
13	2023-01-13	1247087.53
14	2023-01-14	1344511.45
15	2023-01-15	1402563.04
16	2023-01-16	1491050.85

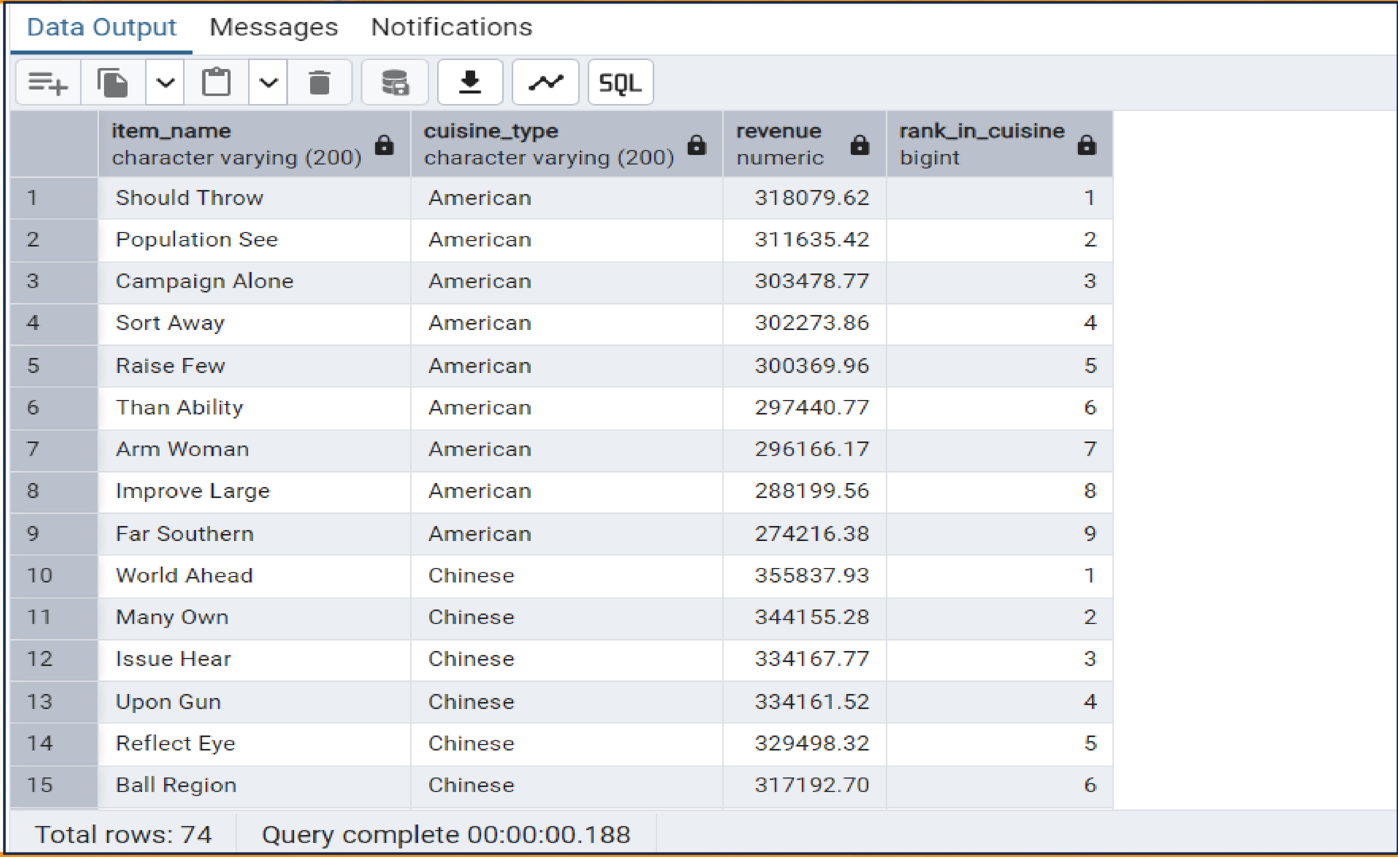
Total rows: 365   Query complete 00:00:00.215

A. Srivarshan

Top Menu Items in Each Cuisine Category

QUERY:

```
SELECT
    m.item_name,
    m.cuisine_type,
    SUM(o.quantity * o.price) AS revenue,
    RANK() OVER (
        PARTITION BY m.cuisine_type
        ORDER BY SUM(o.quantity * o.price) DESC
    ) AS rank_in_cuisine
FROM order_details AS o
JOIN menu_items AS m ON o.menu_item_id = m.menu_item_id
GROUP BY m.item_name, m.cuisine_type
ORDER BY m.cuisine_type, rank_in_cuisine;
```



Data Output | Messages | Notifications

	item_name character varying (200)	cuisine_type character varying (200)	revenue numeric	rank_in_cuisine bigint
1	Should Throw	American	318079.62	1
2	Population See	American	311635.42	2
3	Campaign Alone	American	303478.77	3
4	Sort Away	American	302273.86	4
5	Raise Few	American	300369.96	5
6	Than Ability	American	297440.77	6
7	Arm Woman	American	296166.17	7
8	Improve Large	American	288199.56	8
9	Far Southern	American	274216.38	9
10	World Ahead	Chinese	355837.93	1
11	Many Own	Chinese	344155.28	2
12	Issue Hear	Chinese	334167.77	3
13	Upon Gun	Chinese	334161.52	4
14	Reflect Eye	Chinese	329498.32	5
15	Ball Region	Chinese	317192.70	6

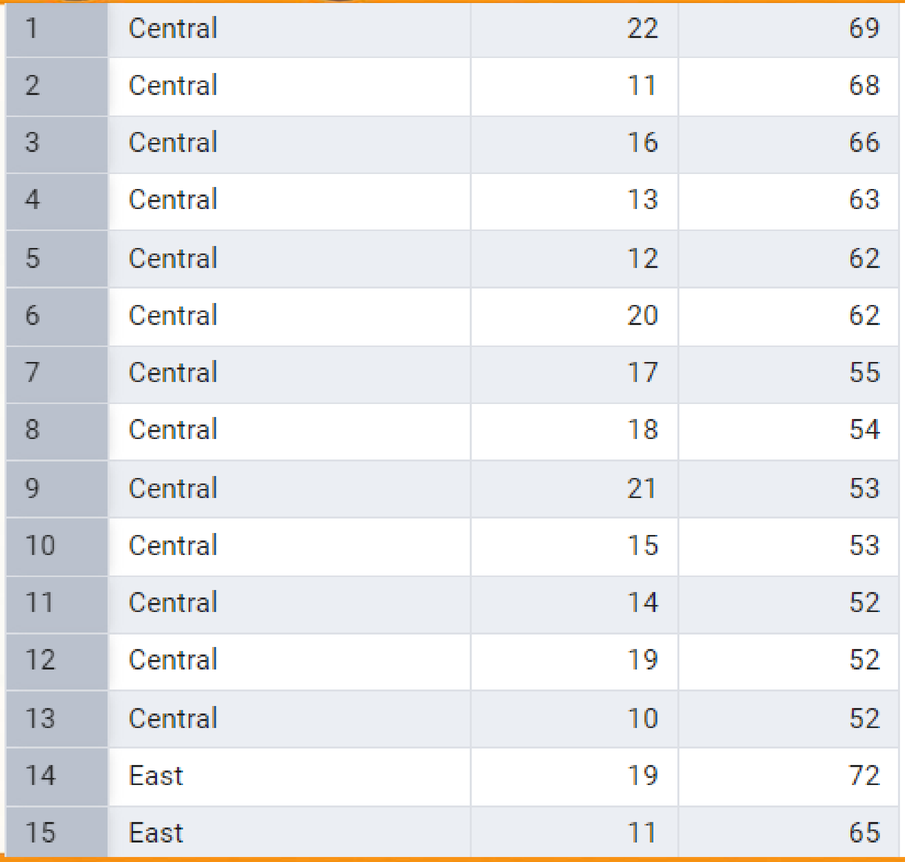
Total rows: 74 Query complete 00:00:00.188

A. Srivarshan

City/Region-Wise Peak Order Patterns

QUERY:

```
SELECT
    c.region,
    EXTRACT(HOUR FROM o.order_time) AS
order_hour,
    COUNT(o.order_id) AS total_orders
FROM orders AS o
JOIN cities AS c ON o.city_id = c.city_id
GROUP BY c.region, order_hour
ORDER BY c.region, total_orders DESC;
```



1	Central	22	69
2	Central	11	68
3	Central	16	66
4	Central	13	63
5	Central	12	62
6	Central	20	62
7	Central	17	55
8	Central	18	54
9	Central	21	53
10	Central	15	53
11	Central	14	52
12	Central	19	52
13	Central	10	52
14	East	19	72
15	East	11	65

A. Srivarshan

Presented by: Allapu Srivarshan Aspiring Data Analyst

Email: allapuramesh68@gmail.com

“SQL projects highlight a data analyst’s proficiency in retrieving, transforming, and analyzing data from real-world databases, demonstrating their ability to generate actionable insights through structured queries and analytical thinking.”

A. Srivarshan

THANK YOU

Thank You