

LAB-7

QUERIES

Query-1: Top 10 places having most expensive dishes

Relational Algebra:

$$\pi_{r.Name, m.Dish_name, m.Price} (\tau_{m.Price \text{ DESC}} (\sigma_{r.Restaurant_id = m.Restaurant_id}(Restaurant \times Menu)))_{Top\ 10}$$

SQL: `SELECT r.Name AS Restaurant, m.Dish_name, m.Price
FROM Restaurant r
JOIN Menu m ON r.Restaurant_id = m.Restaurant_id
ORDER BY m.Price DESC
LIMIT 10;`

Output:

The screenshot shows a database query results window. At the top, it displays the SQL query: `498 LIMIT 5;`. Below the query, there are tabs for Data Output, Messages, and Notifications. The Data Output tab is selected and shows a table with three columns: `restaurant`, `dish_name`, and `price`. The table contains 10 rows of data, each representing a dish from a restaurant, ordered by price in descending order. The data is as follows:

	restaurant	dish_name	price
1	Leato	Pizza (Pepperoni)	350
2	Rajkot Rasoi	Thali Combo	350
3	Leato	Lasagna	350
4	Tandoori Treats	Butter Chicken	350
5	Leato	Pizza (Vegetable)	320
6	Leato	Pizza (Margherita)	300
7	Rangoli Rasoi	Dal Makhani	300
8	Limbdhi Delight	Dal Makhani	300
9	Rajkot Rasoi	Dal Baati Churma	300
10	Tandoori Treats	Chicken Tikka Masala	300

At the bottom of the window, it says `Total rows: 10 of 10 Query complete 00:00:00.070`.

Query-2: Top 5 Restaurants with Highest Average Rating

Relational Algebra:

2) $\Pi_{\text{top 5}} (\tau_{\text{AVG(rr.Rating)}} \text{DESC} (r_1.\text{Name}, \text{AVG}(r_1.\text{Rating}))$
 $r_1.\text{Restaurant_id} = rr.\text{Restaurant_id}$ (Restaurant X Ratings_and_Review))

```
SQL: SELECT r.Name AS Restaurant, AVG(rr.Rating) AS Average_Rating
      FROM Restaurant r
      JOIN Ratings_and_Review rr ON r.Restaurant_id = rr.Restaurant_id
      GROUP BY r.Name
      ORDER BY Average_Rating DESC
      LIMIT 5;
```

Output:

	restaurant	average_rating
1	Imperial Palace	4.6
2	Tandoori Treats	4.533333333333334
3	Leato	4.5
4	The Sweet Junction	4.5
5	Limbdi Delight	4.4

Query-3: Customers Who Have Used Coupons

Relational Algebra:

$$3) \Pi_{c.Name, cu.CouponCode, cu.DiscountAmount} ((c \times cu) \text{ such that } c.Customer_id = cu.Customer_id)$$

SQL: `SELECT c.Name AS Customer, cu.CouponCode, cu.DiscountAmount
FROM Customer c
JOIN Coupons cu ON c.Customer_id = cu.Customer_id;`

Output:

The screenshot shows a SQL query being run in a database environment. The query is:

```
513 ✓ SELECT dp.Name AS Delivery_Personnel, AVG(fdp.Delivery_Person_Rating) AS Average_R
```

The results are displayed in a table:

	customer	couponcode	discountamount
1	Parth Patel	WELCOME10	10.00
2	Riya Shah	SAVE20	20.00
3	Dhruv Mehta	FESTIVAL50	50.00
4	Kavya Trivedi	NEWUSER25	25.00
5	Harshil Joshi	WEEKENDDEAL15	15.00

Query-4: Orders by Customers in Specific Area

Relational Algebra:

$$\begin{aligned} & \pi_{o.Order_id, c.Name, o.Item_Ordered, o.Total_Amount, o.Order_date_and_time} \\ & (\sigma_{ca.DeliveryAddress \text{ LIKE } \% \text{Kalawad Road}\%}) \\ & (\sigma_{c.Customer_id=ca.CustomerID} (\sigma_{o.Customer_id=c.Customer_id} (Orders \times \\ & Customer) \times Customer Addresses))) \end{aligned}$$

SQL: `SELECT o.Order_id, c.Name AS Customer, o.Item_Ordered, o.Total_Amount, o.Order_date_and_time
FROM Orders o
JOIN Customer c ON o.Customer_id = c.Customer_id
JOIN CustomerAddresses ca ON c.Customer_id = ca.CustomerID
WHERE ca.DeliveryAddress LIKE '%Kalawad Road%';`

Output:

The screenshot shows a database query results window. At the top, there is a SQL code block:

```
512 --Delivery Personnel with the Highest Average Rating
513 v SELECT dp.Name AS Delivery_Personnel, AVG(fdp.Delivery_Person_Rating) AS Average_Rating
```

Below the SQL code is a table with the following data:

	order_id	customer	item_ordered	total_amount	order_date_and_time
1	20230303	Dhruv Mehta	Vada Pav	30	2024-10-17 20:45:00
2	20230309	Jay Patel	Paneer Tikka	200	2024-10-18 18:45:00
3	20230312	Dhruv Mehta	Chicken Tikka Masala	300	2024-10-16 11:30:00
4	20230318	Jay Patel	Rasmalai	80	2024-10-17 20:00:00
5	20230321	Dhruv Mehta	Khichdi	150	2024-10-15 15:15:00
6	20230327	Jay Patel	Pizza (Vegetable)	320	2024-10-16 09:45:00
7	20230330	Dhruv Mehta	Ice Cream	80	2024-10-19 13:15:00

Query-5: Delivery Personnel with the Highest Average Rating

Relational Algebra:

5) $\pi_{\text{top 5 } (\tau_{\text{AVG}(\text{fdp.Delivery_Person_Rating}) \text{ DESC}})}$

$\gamma_{dp.\text{Name}, \text{AVG}(\text{fdp.Delivery_Person_Rating})}$

$\sigma_{dp.\text{Delivery_Person_id} = fdp.\text{Delivery_Person_id}}$

$\text{Delivery_Personnel} \times \text{Feedback_for_Delivery_Personnel}}$

SQL: `SELECT dp.Name AS Delivery_Personnel, AVG(fdp.Delivery_Person_Rating) AS Average_Rating
FROM Delivery_Personnel dp
JOIN Feedback_for_Delivery_Personnel fdp ON dp.Delivery_Person_id =
fdp.Delivery_Person_id
GROUP BY dp.Name
ORDER BY Average_Rating DESC
LIMIT 5;`

--Customer Feedback with Below Average Ratings

526 > `SELECT c.Name AS Customer, r.Name AS Restaurant, rr.Rating, rr.Review_text...`

Data Output Messages Notifications

	delivery_personnel	average_rating
1	Jignesh Desai	4.9
2	Ramesh Bhatt	4.75
3	Dinesh Mehta	4.75
4	Priyanka Solanki	4.7
5	Tejas Gandhi	4.6

Output:

Query-6: List of Orders with Pending Payment Status

Relational Algebra:

$$\begin{array}{l} \text{6) } \pi_{\text{Order_id, Name} \rightarrow \text{Customer}, \text{Total_Amount}, \text{Payment_status}} \\ \text{Orders} \bowtie \text{Orders} \cdot \text{customer_id} = \text{Customer} \cdot \text{customer_id} \text{ Customer} \\ \sigma_{\text{Payment_status} = 'COD'} \end{array}$$

SQL: `SELECT o.Order_id, c.Name AS Customer, o.Total_Amount, o.Payment_Status
FROM Orders o
JOIN Customer c ON o.Customer_id = c.Customer_id
WHERE o.Payment_Status = 'COD';`

Output:

The screenshot shows a database query results window with the following details:

- Header: Data Output, Messages, Notifications.
- Toolbar: Includes icons for new table, open table, save, copy, delete, refresh, and search.
- Table Structure:

	order_id	customer	total_amount	payment_status
1	20230305	Harshil Joshi	250	COD
2	20230310	Hetal Thakker	100	COD
3	20230315	Sonal Solanki	200	COD
4	20230320	Riya Shah	150	COD
5	20230325	Vivek Desai	100	COD
6	20230330	Dhruv Mehta	80	COD
- Footer: Total rows: 6 of 6, Query complete 00:00:00.108.

Query-7: Customer Feedback with Below Average Ratings

Relational Algebra: .

7) $\pi_{\text{customer.Name} \rightarrow (\text{customer}, \text{Restaurant}), \text{Name} \rightarrow \text{Restaurant}, \text{Rating}, \text{Review_text}}$

$\sigma_{\text{Rating} < 3} ((\text{Ratings_and_Review} \bowtie \text{Ratings_and_Review}. \text{customer.id} =$

$\text{Customer}.(\text{customer.id}) \bowtie \text{Ratings_and_Review}. \text{Restaurant.id} = \text{Restaurant}.$

$\text{Restaurant_id Restaurant}))$

```
SQL: SELECT c.Name AS Customer, r.Name AS Restaurant, rr.Rating, rr.Review_text  
FROM Ratings_and_Review rr  
JOIN Customer c ON rr.Customer_id = c.Customer_id  
JOIN Restaurant r ON rr.Restaurant_id = r.Restaurant_id  
WHERE rr.Rating < 3;
```

```
532
533 --Top 5 Most Ordered Dishes
534 > SELECT o.Item_Ordered, COUNT(o.Order_id) AS Times_Ordered...
539
```

Data Output Messages Notifications

customer character varying (100) restaurant character varying (100) rating double precision review_text text

	customer	restaurant	rating	review_text
1	Kavya Trivedi	Swad Samosa Junction	2.5	Dahi Vada lacked flavor.

Query-8: Top 5 Most Ordered Dishes

Relational Algebra:

$$\begin{array}{l} 8) \pi_{Item_Ordered, Times_Ordered} (\sigma_{Times_Ordered \text{ DESC}} (\sigma_{Top\ 5} (\\ \gamma_{Item_Ordered, COUNT(Order_id)} \rightarrow Times_Ordered (Orders)))) \end{array}$$

```
SQL: SELECT o.Item_Ordered, COUNT(o.Order_id) AS Times_Ordered
FROM Orders o
GROUP BY o.Item_Ordered
ORDER BY Times_Ordered DESC
LIMIT 5;
```

Output:

The screenshot shows a database query results window. The query is:

```
547 > SELECT o.Item_Ordered, o.Total_Amount, o.Order_date_and_time...
```

The results are displayed in a table with three columns: item_ordered, times_ordered, and total_amount. The table has 5 rows, corresponding to the top 5 most ordered items.

	item_ordered	times_ordered	total_amount
1	Thali Combo	1	100.00
2	Manchurian	1	100.00
3	Undhiyu	1	100.00
4	Butter Chicken	1	100.00
5	Pizza (Margherita)	1	100.00

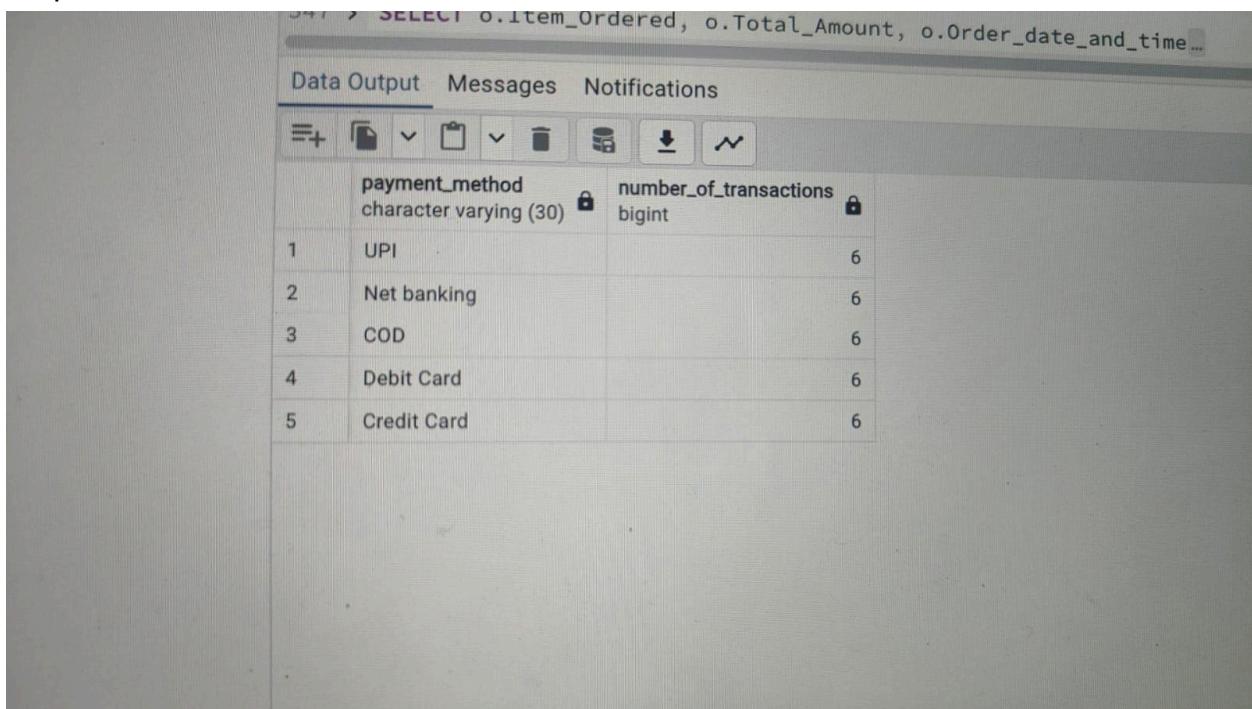
Query-9: Most Popular Payment Method

9) $\pi_{\text{Payment_Status} \rightarrow \text{Payment_Method}, \text{Number_of_Transactions}}$
 $\tau_{\text{Number_of_Transactions DESC}} (\gamma_{\text{Payment_Status}, (\text{COUNT}(\text{Order_id}) \rightarrow \text{Number_of_Transactions}))}$

Relational Algebra:

```
SQL: SELECT o.Payment_Status AS Payment_Method, COUNT(o.Order_id) AS  
Number_of_Transactions  
FROM Orders o  
GROUP BY o.Payment_Status  
ORDER BY Number_of_Transactions DESC;
```

Output:



The screenshot shows a database interface with a query editor at the top containing the following SQL code:

```
SELECT o.item_Ordered, o.Total_Amount, o.order_date_and_time...
```

Below the query editor is a toolbar with icons for Data Output, Messages, Notifications, and various file operations.

The main area displays a table with two columns: `payment_method` and `number_of_transactions`. The data is as follows:

	payment_method	number_of_transactions
1	UPI	6
2	Net banking	6
3	COD	6
4	Debit Card	6
5	Credit Card	6

Query-10: Orders Delivered by a Specific Delivery Personnel

Relational Algebra:

10) $\pi_{\text{Order_id}, \text{Item_Ordered}, \text{Order_date_and_time}, \text{Total_Amount}} (\sigma_{\text{Delivery_Person_id} = 3}(\text{Orders}))$

SQL: `SELECT o.Order_id, o.Item_Ordered, o.Order_date_and_time, o.Total_Amount
FROM Orders o
WHERE o.Delivery_Person_id = 3;`

Output:

	order_id [PK] bigint	item_ordered character varying (100)	order_date_and_time timestamp without time zone	total_amount integer
1	20230303	Vada Pav	2024-10-17 20:45:00	30
2	202303013	Dahi Vada	2024-10-17 19:45:00	50
3	202303023	Pani Puri	2024-10-17 18:15:00	30

Query-11: Average Order Value by Restaurant

Relational Algebra:

$$\begin{aligned} \text{11) } & \pi_{\text{Name} \rightarrow \text{Restaurant}, \text{Average_Order_Value}} (\sigma_{\text{Average_Order_Value DESC}} (\gamma_{\text{Name}, \text{AVG}(\text{Total_Amount}) \rightarrow \text{Average_Order_Value}} (\text{Restaurant} \bowtie \\ & \text{Restaurant}, \text{Restaurant_id} = \text{Orders}. \text{Restaurant_id} \text{ Orders})))) \end{aligned}$$

SQL: `SELECT r.Name AS Restaurant, AVG(o.Total_Amount) AS Average_Order_Value
FROM Restaurant r
JOIN Orders o ON r.Restaurant_id = o.Restaurant_id
GROUP BY r.Name
ORDER BY Average_Order_Value DESC;`

Output:

The screenshot shows a database query results window with the following details:

- Header: Data Output, Messages, Notifications.
- Toolbar: Includes icons for new table, file operations (open, save, etc.), and download.
- Table Structure:

	restaurant	average_order_value
character varying (100)		numeric
- Data:

	restaurant	average_order_value
1	Rangoli Rasoi	306.6666666666666667
2	Tandoori Treats	300.0000000000000000
3	Limbdi Delight	266.6666666666666667
4	Imperial Palace	233.3333333333333333
5	Kathiawadi Kitchen	183.3333333333333333
6	Rajkot Rasoi	183.3333333333333333
7	The Sweet Junction	110.0000000000000000
8	The Fusion Factory	90.0000000000000000
9	Leato	63.3333333333333333
10	Swad Samosa Junction	36.6666666666666667
- Message Bar: Total rows: 10 of 10 Query complete 00:00:00.078
- Bottom: MacBook Pro

Query-12: Total Number of Orders for Each Day in the Last Week

Relational Algebra:

12) $\pi_{\text{Order_Date}, \text{Total_Orders}} (\sigma_{\text{Order_date_and_time} \geq \text{NOW}() - \text{INTERVAL}'30days'} (\rho_{\text{CAST}(\text{Order_date_and_time}) \text{ AS DATE}} (\text{COUNT}(\text{Order_id}) \rightarrow \text{Total_Orders})))$

SQL: `SELECT CAST(o.Order_date_and_time AS DATE) AS Order_Date,
COUNT(o.Order_id) AS Total_Orders
FROM Orders o
WHERE o.Order_date_and_time >= NOW() - INTERVAL '7 days'
GROUP BY Order_Date
ORDER BY Order_Date;`

Output:

	order_date	total_orders
1	2024-10-15	6
2	2024-10-16	6
3	2024-10-17	6
4	2024-10-18	6
5	2024-10-19	6

Query-13: Restaurants That Serve the Same Dish

Relational Algebra:

3) $\Pi_{\text{Name} \rightarrow \text{Restaurant}, \text{Dish_name}} (\sigma_{\text{Dish_name} = \text{'Gulab Jamun'}} (\text{Restaurant} \bowtie \text{Menu}))$
 $\text{Restaurant} \bowtie \text{Menu}$ $\text{Restaurant}.\text{Restaurant_id} = \text{Menu}.\text{Restaurant_id}$

SQL: `SELECT r.Name AS Restaurant, m.Dish_name
FROM Restaurant r
JOIN Menu m ON r.Restaurant_id = m.Restaurant_id
WHERE m.Dish_name = 'Gulab Jamun';`

Output:

	restaurant	dish_name
1	Rajkot Rasoi	Gulab Jamun
2	Tandoori Treats	Gulab Jamun
3	Rangoli Rasoi	Gulab Jamun

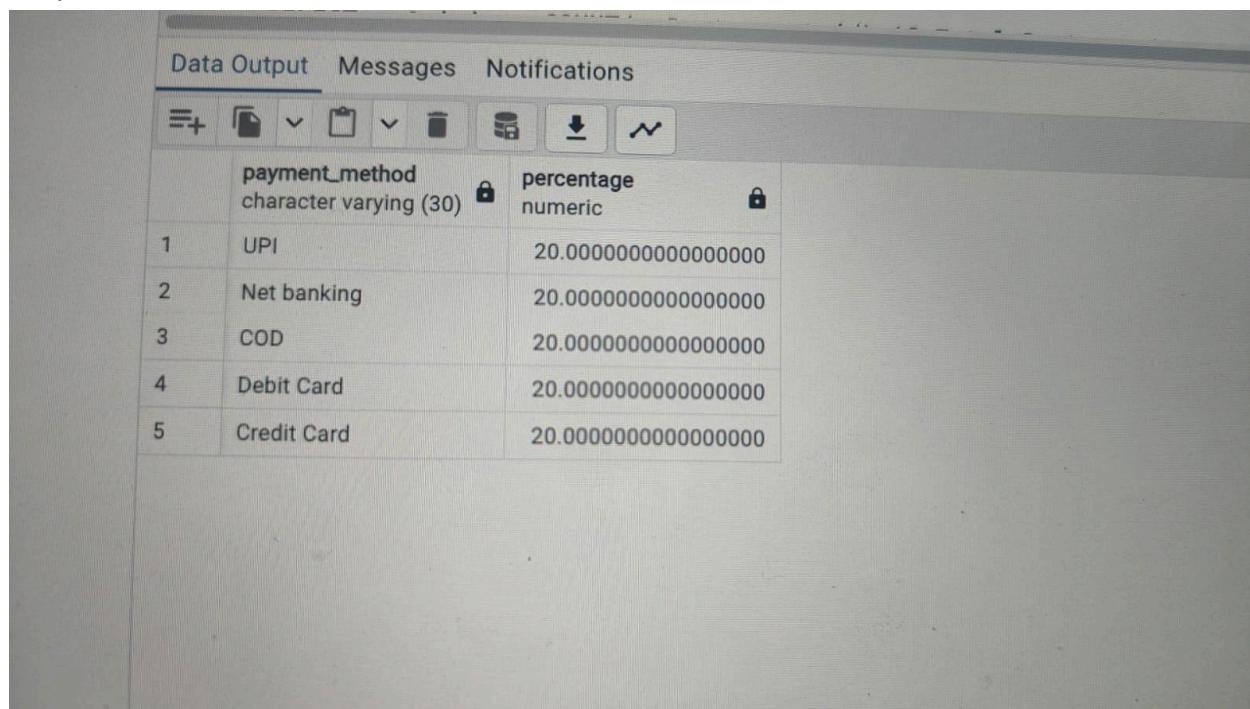
Query-14: Percentage of Orders with Each Payment Method

Relational Algebra:

14) $\pi_{\text{Payment_Status} \rightarrow \text{Payment_Method}, \left(\frac{\text{COUNT(Order_id)} \times 100.0}{\text{COUNT(Oders)}} \right) \rightarrow \text{Percentage}}$
 $(\gamma_{\text{Payment_Status}, \text{COUNT(Order_id)}} (\text{Orders}))$

SQL: `SELECT o.Payment_Status AS Payment_Method,
 (COUNT(o.Order_id) * 100.0 / (SELECT COUNT(*) FROM Orders)) AS Percentage
 FROM Orders o
 GROUP BY o.Payment_Status;`

Output:



The screenshot shows a database interface with a toolbar at the top labeled "Data Output", "Messages", and "Notifications". Below the toolbar is a grid of data. The first column is labeled "payment_method" and "character varying (30)". The second column is labeled "percentage" and "numeric". The data consists of five rows, each representing a payment method and its corresponding percentage:

	payment_method	percentage
1	UPI	20.0000000000000000
2	Net banking	20.0000000000000000
3	COD	20.0000000000000000
4	Debit Card	20.0000000000000000
5	Credit Card	20.0000000000000000

Query-15: Most Popular Cuisine Type

Relational Algebra:

$$\begin{aligned} 15) \quad & \pi_{(\text{cuisine}, \text{Total_Restaurants})} \left(\tau_{\text{Total_Restaurants}} \downarrow \gamma_{(\text{cuisine}, \text{COUNT}(\text{Restaurant_id}))} \right. \\ & \quad \left. (\text{Restaurant}) \right) \\ & \rightarrow \text{Total_Restaurants} \end{aligned}$$

SQL: `SELECT r.Cuisine, COUNT(r.Restaurant_id) AS Total_Restaurants
FROM Restaurant r
GROUP BY r.Cuisine
ORDER BY Total_Restaurants DESC;`

Output:

The screenshot shows a database interface with the following details:

- SQL code at the top:

```
591 --Customers with No Active Coupons
592 > SELECT c.Name AS Customer...
596
597 --Restaurants with the Highest Number of Menu Items
598 > SELECT r.Name AS Restaurant, COUNT(m.Dish_name) AS Number of Dishes
```
- Toolbar below the code:

Data Output Messages Notifications
- Table results:

	cuisine	total_restaurants
1	Gujarati	2
2	North Indian	1
3	Kathiawadi	1
4	Italian	1
5	Snacks	1
6	Multi-Cuisine	1
7	Mughlai	1
8	Indian	1
9	Desserts	1
- Status bar at the bottom:

Total Rows: 9 of 9 Query completed: 00:00:00.000

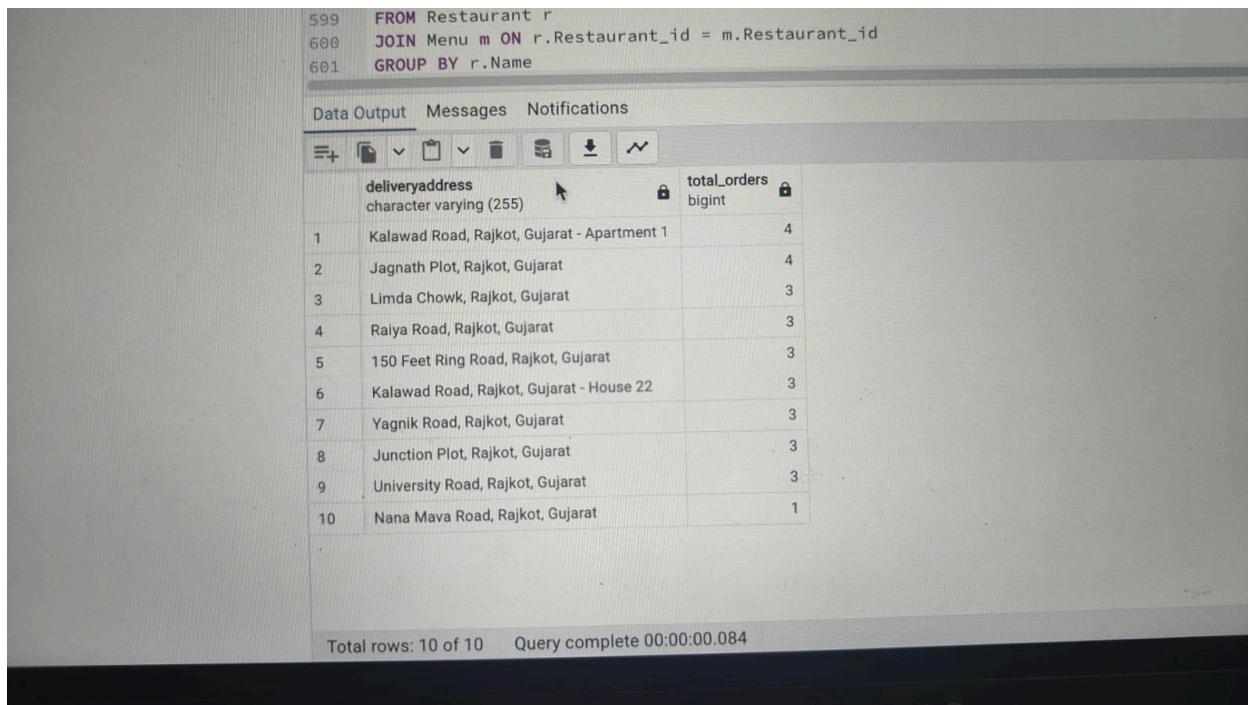
Query-16: Most Common Delivery Locations

Relational Algebra:

16) $\pi_{ca.\text{DeliveryAddress}, \text{Total_Orders}} (T \text{ Total_Orders DESC } \gamma_{ca.\text{DeliveryAddress},})$
 $\text{COUNT}(o.\text{order_id}) \rightarrow \text{Total_Orders} (o \bowtie_{o.\text{Customer_id} = ca.\text{Customer_id}} ca))$

SQL: `SELECT ca.DeliveryAddress, COUNT(o.Order_id) AS Total_Orders
FROM Orders o
JOIN CustomerAddresses ca ON o.Customer_id = ca.CustomerID
GROUP BY ca.DeliveryAddress
ORDER BY Total_Orders DESC;`

Output:



The screenshot shows a database query results window. At the top, there is a code editor with the following SQL query:

```
599 FROM Restaurant r
600 JOIN Menu m ON r.Restaurant_id = m.Restaurant_id
601 GROUP BY r.Name
```

Below the code editor is a data grid with the following columns:

	deliveryaddress	total_orders
1	Kalawad Road, Rajkot, Gujarat - Apartment 1	4
2	Jagnath Plot, Rajkot, Gujarat	4
3	Limda Chowk, Rajkot, Gujarat	3
4	Raiya Road, Rajkot, Gujarat	3
5	150 Feet Ring Road, Rajkot, Gujarat	3
6	Kalawad Road, Rajkot, Gujarat - House 22	3
7	Yagnik Road, Rajkot, Gujarat	3
8	Junction Plot, Rajkot, Gujarat	3
9	University Road, Rajkot, Gujarat	3
10	Nana Mava Road, Rajkot, Gujarat	1

At the bottom of the window, it says "Total rows: 10 of 10" and "Query complete 00:00:00.084".

Query-17: Customers with No Active Coupons

Relational Algebra:

$$\begin{aligned} 17) \pi_{c.Name \rightarrow customer} (\sigma_{cu.Coupon_id \text{ IS NULL}} (c \bowtie_{c.Customer_id} cu)) \\ = (c.Customer_id \text{ AND } cu.ExpirationDate \geq \text{Current_Date}) \end{aligned}$$

SQL: `SELECT c.Name AS Customer`

`FROM Customer c`

`LEFT JOIN Coupons cu ON c.Customer_id = cu.Customer_id AND cu.ExpirationDate`
`>= CURRENT_DATE`

`WHERE cu.Coupon_id IS NULL;`

Output:

customer	
1	Niharika Joshi
2	Manav Vyas
3	Raghav Panchal
4	Payal Kothari
5	Hetal Thakker
6	Sweta Zaveri
7	Hiren Chauhan
8	Aman Bhandari
9	Isahn Pandya
10	Krupa Gohil
11	Sonal Solanki
12	Ritika Desai
13	Megha Jani
14	Jay Patel
15	Vivek Desai

Query-18: Restaurants with the Highest Number of Menu Items

Relational Algebra:

18) $\Pi_{r.Name \rightarrow \text{Restaurant}, \text{Number_of_Dishes}} (T \text{ Number_of_Dishes DESC})$

$\gamma_{r.Name, \text{COUNT}(m.Dish_name) \rightarrow \text{Number_of_Dishes}}$

$r \bowtie_{r.RestaurantId = m.Restaurant_id} m))) \text{ Top 5}$

SQL: `SELECT r.Name AS Restaurant, COUNT(m.Dish_name) AS Number_of_Dishes
FROM Restaurant r
JOIN Menu m ON r.Restaurant_id = m.Restaurant_id
GROUP BY r.Name
ORDER BY Number_of_Dishes DESC
LIMIT 5;`

Data Output Messages Notifications

	restaurant	number_of_dishes
1	Rangoli Rasoi	13
2	Leato	13
3	The Fusion Factory	11
4	Imperial Palace	11
5	Rajkot Rasoi	10

Output:

Query-19: Average Rating Per Cuisine Type

Relational Algebra:

$$\begin{aligned} & \text{19) } \pi_{r.\text{cuisine}, \text{Average_Rating}} \text{ (T} \\ & \quad \text{Average_Rating DESC)} \\ & \gamma_{r.\text{cuisine}, \text{AVG}(rr.\text{Rating}) \rightarrow \text{Average_Rating}} \\ & \quad r \bowtie_{r.\text{Restaurant_id} = rr.\text{Restaurant_id}} r \end{aligned}$$

SQL: `SELECT r.Cuisine, AVG(rr.Rating) AS Average_Rating
FROM Restaurant r
JOIN Ratings_and_Review rr ON r.Restaurant_id = rr.Restaurant_id
GROUP BY r.Cuisine
ORDER BY Average_Rating DESC;`

Output:

	cuisine	average_rating
1	Indian	4.6
2	Mughlai	4.533333333333334
3	Desserts	4.5
4	Italian	4.5
5	North Indian	4.4
6	Kathiawadi	4.4
7	Gujarati	4.05
8	Snacks	3.933333333333336
9	Multi-Cuisine	3.7

Query-20: Identify Customers with Active Coupons

Relational Algebra:

$$\begin{array}{l} \text{20)} \\ \text{TT}_{\text{Customer.Name}} \left(\sigma_{\text{ValidCoupons.Coupon_id IS NULL}} \right) \\ \text{Customer} \bowtie_{\text{customer.Customer_id} = \text{ValidCoupons.Customer_id}} \\ \sigma_{\text{ExpirationDate} \geq \text{Current_Date}} (\text{Coupons})) \end{array}$$

SQL: SELECT c.Name AS Customer
FROM Customer c
LEFT JOIN Coupons cu ON c.Customer_id = cu.Customer_id AND cu.ExpirationDate
>= CURRENT_DATE
WHERE cu.Coupon_id IS NULL;

Output:

The screenshot shows a database interface with a sidebar on the left containing various navigation items such as Data Wrappers, Tables, Triggers, Dictionaries, Parsers, Templates, Design Tables, Functions, Materialized Views, Operators, Procedures, References, and Views (11). The main area displays a table with a single column labeled 'customer' and a data type of 'character varying (100)'. The table contains 18 rows, each showing a customer name. The names listed are: Niharika Joshi, Manav Vyas, Raghav Panchal, Payal Kothari, Hetal Thakker, Sweta Zaveri, Hiren Chauhan, Aman Bhandari, Harshil Joshi, Ishan Pandya, Krupa Gohil, Sonal Solanki, Ritika Desai, Kavya Trivedi, Dhruv Mehta, Megha Jani, Jay Patel, and Vivek Desai.

	customer
1	Niharika Joshi
2	Manav Vyas
3	Raghav Panchal
4	Payal Kothari
5	Hetal Thakker
6	Sweta Zaveri
7	Hiren Chauhan
8	Aman Bhandari
9	Harshil Joshi
10	Ishan Pandya
11	Krupa Gohil
12	Sonal Solanki
13	Ritika Desai
14	Kavya Trivedi
15	Dhruv Mehta
16	Megha Jani
17	Jay Patel
18	Vivek Desai

