

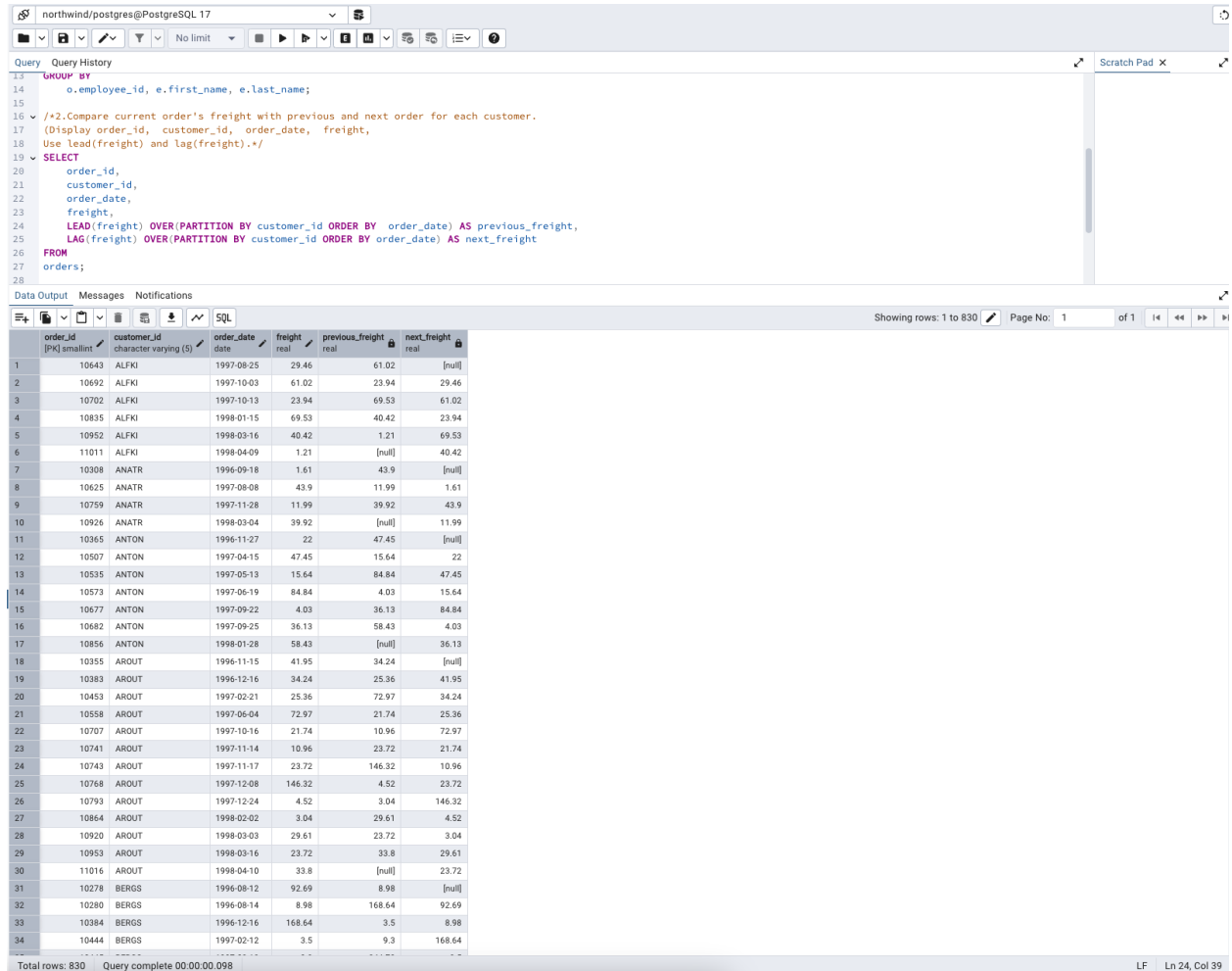
(Total sales = Total no of orders handled, JOIN employees and orders table)

Total rows: 9	Query complete 00:00:00.084	LF	Ln 5, Col
---------------	-----------------------------	----	-----------

2. Compare current order's freight with previous and next order for each customer.

(Display order\_id, customer\_id, order\_date, freight,

Use lead(freight) and lag(freight).



```
Query History
13  Query
14  o.employee_id, e.first_name, e.last_name;
15
16  /*2.Compare current order's freight with previous and next order for each customer.
17  (Display order_id, customer_id, order_date, freight,
18  Use lead(freight) and lag(freight).*/
19  SELECT
20  order_id,
21  customer_id,
22  order_date,
23  freight,
24  LEAD(freight) OVER(PARTITION BY customer_id ORDER BY order_date) AS previous_freight,
25  LAG(freight) OVER(PARTITION BY customer_id ORDER BY order_date) AS next_freight
26  FROM
27  orders;
28
```

order_id	customer_id	order_date	freight	previous_freight	next_freight
10643	ALFKI	1997-08-25	29.46	61.02	[null]
10692	ALFKI	1997-10-03	61.02	23.94	29.46
10702	ALFKI	1997-10-13	23.94	69.53	61.02
10835	ALFKI	1998-01-15	69.53	40.42	23.94
10952	ALFKI	1998-03-16	40.42	1.21	69.53
11011	ALFKI	1998-04-09	1.21	[null]	40.42
10308	ANATR	1996-09-18	1.61	43.9	[null]
10625	ANATR	1997-08-08	43.9	11.99	1.61
10759	ANATR	1997-11-28	11.99	39.92	43.9
10926	ANATR	1998-03-04	39.92	[null]	11.99
10365	ANTON	1996-11-27	22	47.45	[null]
10507	ANTON	1997-04-15	47.45	15.64	22
10535	ANTON	1997-05-13	15.64	84.84	47.45
10573	ANTON	1997-06-19	84.84	4.03	15.64
10677	ANTON	1997-09-22	4.03	36.13	84.84
10682	ANTON	1997-09-25	36.13	58.43	4.03
10856	ANTON	1998-01-28	58.43	[null]	36.13
10355	AROUT	1996-11-15	41.95	34.24	[null]
10383	AROUT	1996-12-16	34.24	25.36	41.95
10453	AROUT	1997-02-21	25.36	72.97	34.24
10558	AROUT	1997-06-04	72.97	21.74	25.36
10707	AROUT	1997-10-16	21.74	10.96	72.97
10741	AROUT	1997-11-14	10.96	23.72	21.74
10743	AROUT	1997-11-17	23.72	146.32	10.96
10768	AROUT	1997-12-08	146.32	4.52	23.72
10793	AROUT	1997-12-24	4.52	3.04	146.32
10864	AROUT	1998-02-02	3.04	29.61	4.52
10920	AROUT	1998-03-03	29.61	23.72	3.04
10953	AROUT	1998-03-16	23.72	33.8	29.61
11016	AROUT	1998-04-10	33.8	[null]	23.72
10278	BERGS	1996-08-12	92.69	8.98	[null]
10280	BERGS	1996-08-14	8.98	168.64	92.69
10384	BERGS	1996-12-16	168.64	3.5	8.98
10444	BERGS	1997-02-12	3.5	9.3	168.64

Total rows: 830 Query complete 00:00:00.098

3. Show products and their price categories, product count in each category, avg price:

(HINT:

- Create a CTE which should have price\_category definition:

WHEN unit\_price < 20 THEN 'Low Price'

WHEN unit\_price < 50 THEN 'Medium Price'

ELSE 'High Price'

- In the main query display: price\_category, product\_count in each price\_category, ROUND(AVG(unit\_price)::numeric, 2) as avg\_price)

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
28 /*3.Show products and their price categories, product count in each category, avg price:
29 (HINT:
30 • Create a CTE which should have price_category definition:
31 WHEN unit_price < 20 THEN 'Low Price'
32 WHEN unit_price < 50 THEN 'Medium Price'
33 ELSE 'High Price'
34 • In the main query display: price_category, product_count in each price_category, ROUND(AVG(unit_price)::numeric, 2) as avg_price)*/
35
36 WITH price_cte AS (
37 SELECT product_name, unit_price,
38 CASE WHEN unit_price < 20 THEN 'Low Price'
39 WHEN unit_price < 50 THEN 'Medium Price'
40 ELSE 'High Price'
41 END AS price_category
42 FROM products)
43 SELECT price_category,
44 COUNT(*) AS product_count, ROUND(AVG(unit_price)::numeric, 2) AS avg_price
45 FROM price_cte
46 GROUP BY price_category
47 ORDER BY price_category;
48
```

The results of the query are displayed in a table with 3 rows:

	price_category text	product_count bigint	avg_price numeric
1	High Price	7	105.11
2	Low Price	39	12.95
3	Medium Price	31	31.59

At the bottom of the interface, it says "Total rows: 3" and "Query complete 00:00:00.080".