

1. Find the Previous Purchase for Each Customer Using LAG()

Question: For each customer, display their current purchase and the previous purchase they made, ordered by market_date.

Solution:

```
SELECT customer_id, market_date, product_id,  
LAG(product_id, 1) OVER (PARTITION BY customer_id ORDER BY market_date) AS previous_product  
FROM customer_purchases;
```

2. Calculate the Difference in Purchase Amount Between Current and Previous Transaction Using LAG()

Question: For each customer, calculate the difference in purchase amount between the current transaction and their previous transaction.

Solution:

```
SELECT customer_id, market_date, quantity * cost_to_customer_per_qty AS current_amount,  
LAG(quantity * cost_to_customer_per_qty, 1) OVER (PARTITION BY customer_id ORDER BY  
market_date) AS previous_amount,  
(quantity * cost_to_customer_per_qty) - LAG(quantity * cost_to_customer_per_qty, 1) OVER  
(PARTITION BY customer_id ORDER BY market_date) AS amount_difference  
FROM customer_purchases;
```

3. Find the Next Product a Customer Purchased Using LEAD()

Question: For each customer, display their current product and the next product they purchased, ordered by market_date.

Solution:

```
SELECT customer_id, market_date, product_id,  
LEAD(product_id, 1) OVER (PARTITION BY customer_id ORDER BY market_date) AS next_product  
FROM customer_purchases;
```

4. Identify the First Purchase and Second Purchase of Each Customer Using LEAD()

Question: For each customer, display their first and second purchases, ordered by market_date.

Solution:

```
SELECT customer_id, market_date, product_id,  
LEAD(product_id, 1) OVER (PARTITION BY customer_id ORDER BY market_date) AS second_purchase  
FROM customer_purchases  
  
WHERE market_date = (SELECT MIN(market_date) FROM customer_purchases WHERE customer_id =  
customer_purchases.customer_id);
```

5. Find the Purchase Date Difference Between Consecutive Purchases for Each Customer

Question: Calculate the number of days between consecutive purchases for each customer.

Solution:

```
SELECT customer_id, market_date,  
DATEDIFF(market_date, LAG(market_date, 1) OVER (PARTITION BY customer_id ORDER BY  
market_date)) AS days_between_purchases  
FROM customer_purchases;
```

6. Compare the Current and Next Booth Price for Vendors Using LEAD()

Question: For each vendor, compare the current booth price with the price for the next market date.

Solution:

```
SELECT vendor_id, market_date, booth_number,  
booth_price_level,  
LEAD(booth_price_level, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS  
next_booth_price  
FROM vendor_booth_assignments
```

JOIN booth ON vendor_booth_assignments.booth_number = booth.booth_number;

7. Calculate the Running Total of Customer Purchases Using PARTITION BY

Question: For each customer, calculate the running total of their purchases over time.

Solution:

```
SELECT customer_id, market_date,  
SUM(quantity * cost_to_customer_per_qty) OVER (PARTITION BY customer_id ORDER BY  
market_date) AS running_total  
FROM customer_purchases;
```

8. Find the Previous Booth Assignment for Each Vendor

Question: For each vendor, display their current booth assignment and the previous booth assignment.

Solution:

```
SELECT vendor_id, market_date, booth_number,  
LAG(booth_number, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS previous_booth  
FROM vendor_booth_assignments;
```

9. Calculate the Average Cost Difference Between Consecutive Purchases

Question: Calculate the average difference in purchase cost between consecutive purchases for each customer.

Solution:

```
SELECT customer_id, AVG((quantity * cost_to_customer_per_qty) - LAG(quantity *  
cost_to_customer_per_qty, 1) OVER (PARTITION BY customer_id ORDER BY market_date)) AS  
avg_cost_difference  
FROM customer_purchases  
GROUP BY customer_id;
```

10. Identify the Next Vendor Assignment for Each Booth

Question: For each booth, display the current vendor assigned and the next vendor to be assigned to that booth on a future market date.

Solution:

```
SELECT booth_number, vendor_id, market_date,  
LEAD(vendor_id, 1) OVER (PARTITION BY booth_number ORDER BY market_date) AS next_vendor  
FROM vendor_booth_assignments;
```

11. Find the Product Sold Before and After the Most Popular Product

Question: For each vendor, identify the product sold before and after the most popular product in terms of quantity sold.

Solution:

```
WITH product_sales AS (  
    SELECT vendor_id, product_id, SUM(quantity) AS total_quantity,  
    ROW_NUMBER() OVER (PARTITION BY vendor_id ORDER BY SUM(quantity) DESC) AS rank  
    FROM customer_purchases  
    GROUP BY vendor_id, product_id  
)  
SELECT vendor_id, product_id, total_quantity,  
LAG(product_id, 1) OVER (PARTITION BY vendor_id ORDER BY total_quantity DESC) AS  
previous_product,  
LEAD(product_id, 1) OVER (PARTITION BY vendor_id ORDER BY total_quantity DESC) AS next_product  
FROM product_sales  
WHERE rank = 1;
```

12. Compare the Current and Next Market Date for Each Vendor

Question: For each vendor, compare the current market date with the next market date they are scheduled to attend.

Solution:

```
SELECT vendor_id, market_date,  
LEAD(market_date, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS next_market_date  
FROM vendor_booth_assignments;
```

13. Calculate the Lead Time Between Product Sales for Each Vendor

Question: For each vendor, calculate the time (in days) between each product sale and the next sale.

Solution:

```
SELECT vendor_id, product_id, market_date,  
DATEDIFF(LEAD(market_date, 1) OVER (PARTITION BY vendor_id ORDER BY market_date),  
market_date) AS lead_time  
FROM customer_purchases;
```

14. Compare the Current and Previous Inventory Update Date for Vendors

Question: For each vendor, compare the current inventory update date with the previous update date.

Solution:

```
SELECT vendor_id, market_date,  
LAG(market_date, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS  
previous_update_date  
FROM vendor_inventory;
```

15. Identify Vendors with the Shortest Lead Time Between Sales

Question: For each vendor, calculate the lead time between their consecutive sales and identify vendors with the shortest lead time.

Solution:

```
WITH lead_times AS (  
    SELECT vendor_id, DATEDIFF(LEAD(market_date, 1) OVER (PARTITION BY vendor_id ORDER BY  
market_date), market_date) AS lead_time  
    FROM customer_purchases  
)  
SELECT vendor_id, MIN(lead_time) AS shortest_lead_time  
FROM lead_times  
GROUP BY vendor_id;
```

These questions cover **LEAD()** and **LAG()** functions, as well as **PARTITION BY**, and they range from basic to advanced concepts. Let me know if you need further clarifications!

40

You said:

some more basic to medium level questions

ChatGPT said:

ChatGPT

Here are some additional **basic to medium level** questions focused on the **LEAD()**, **LAG()**, and **PARTITION BY** window functions, with solutions, based on the Farmers Market Database:

16. Find the Previous Purchase Date for Each Customer

Question: For each customer, display their current purchase date and the previous purchase date, ordered by market_date.

Solution:

```
SELECT customer_id, market_date,  
LAG(market_date, 1) OVER (PARTITION BY customer_id ORDER BY market_date) AS  
previous_purchase_date  
FROM customer_purchases;
```

17. Compare the Current and Next Product Price for Each Vendor

Question: For each vendor, display the current product price and the next product price, ordered by market_date.

Solution:

```
SELECT vendor_id, product_id, market_date, cost_to_customer_per_qty AS current_price,  
LEAD(cost_to_customer_per_qty, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS  
next_price  
FROM customer_purchases;
```

18. Calculate the Lead Time Between Vendor Booth Assignments

Question: For each vendor, calculate the time (in days) between consecutive booth assignments.

Solution:

```
SELECT vendor_id, market_date,  
DATEDIFF(LEAD(market_date, 1) OVER (PARTITION BY vendor_id ORDER BY market_date),  
market_date) AS days_between_booths  
FROM vendor_booth_assignments;
```

19. Find the Previous and Next Booth Assignment for Each Vendor

Question: For each vendor, display their current booth assignment, along with their previous and next booth assignment.

Solution:

```
SELECT vendor_id, booth_number, market_date,  
LAG(booth_number, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS previous_booth,  
LEAD(booth_number, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS next_booth  
FROM vendor_booth_assignments;
```

20. Identify Vendors Who Changed Booths Between Consecutive Market Dates

Question: Identify vendors who were assigned to a different booth on consecutive market dates.

Solution:

```
SELECT vendor_id, market_date, booth_number,  
LAG(booth_number, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS previous_booth  
FROM vendor_booth_assignments  
  
WHERE booth_number != LAG(booth_number, 1) OVER (PARTITION BY vendor_id ORDER BY  
market_date);
```

21. Calculate the Sales Difference Between Consecutive Transactions

Question: For each customer, calculate the difference in total cost between consecutive transactions.

Solution:

```
SELECT customer_id, market_date, quantity * cost_to_customer_per_qty AS current_sale,  
LAG(quantity * cost_to_customer_per_qty, 1) OVER (PARTITION BY customer_id ORDER BY  
market_date) AS previous_sale,  
  
(quantity * cost_to_customer_per_qty) - LAG(quantity * cost_to_customer_per_qty, 1) OVER  
(PARTITION BY customer_id ORDER BY market_date) AS sale_difference  
  
FROM customer_purchases;
```

22. Find the First and Second Most Sold Products for Each Vendor

Question: For each vendor, display the first and second most sold products, ordered by total quantity sold.

Solution:

```
WITH product_ranking AS (  
    SELECT vendor_id, product_id, SUM(quantity) AS total_quantity,  
    ROW_NUMBER() OVER (PARTITION BY vendor_id ORDER BY SUM(quantity) DESC) AS rank
```



```
FROM customer_purchases
GROUP BY vendor_id, product_id
)
SELECT vendor_id, product_id, total_quantity
FROM product_ranking
WHERE rank IN (1, 2);
```

23. Compare the Current and Previous Total Revenue for Each Vendor

Question: For each vendor, compare their total revenue for the current market_date and the previous market_date.

Solution:

```
SELECT vendor_id, market_date, SUM(quantity * cost_to_customer_per_qty) AS current_revenue,
LAG(SUM(quantity * cost_to_customer_per_qty), 1) OVER (PARTITION BY vendor_id ORDER BY
market_date) AS previous_revenue
FROM customer_purchases
GROUP BY vendor_id, market_date;
```

24. Display the Running Total of Products Sold by Each Vendor

Question: For each vendor, display the running total of products sold across different market dates.

Solution:

```
SELECT vendor_id, market_date, product_id, quantity,
SUM(quantity) OVER (PARTITION BY vendor_id ORDER BY market_date) AS running_total
FROM customer_purchases;
```

25. Identify Customers Who Increased Their Purchase Quantity Over Consecutive Transactions

Question: For each customer, identify those whose purchase quantity increased between consecutive transactions.

Solution:

```
SELECT customer_id, market_date, quantity,  
LAG(quantity, 1) OVER (PARTITION BY customer_id ORDER BY market_date) AS previous_quantity  
FROM customer_purchases  
WHERE quantity > LAG(quantity, 1) OVER (PARTITION BY customer_id ORDER BY market_date);
```

26. Calculate the Difference in Booth Prices Between Consecutive Assignments for Vendors

Question: For each vendor, calculate the difference in booth price between consecutive booth assignments.

Solution:

```
SELECT vendor_id, market_date, booth_price_level,  
LAG(booth_price_level, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS  
previous_booth_price,  
booth_price_level - LAG(booth_price_level, 1) OVER (PARTITION BY vendor_id ORDER BY  
market_date) AS price_difference  
FROM vendor_booth_assignments  
JOIN booth ON vendor_booth_assignments.booth_number = booth.booth_number;
```

27. Find Vendors with the Highest and Lowest Sales for Consecutive Market Dates

Question: Identify vendors who have the highest and lowest sales for consecutive market_dates.

Solution:

```
WITH vendor_sales AS (  
  SELECT vendor_id, market_date, SUM(quantity * cost_to_customer_per_qty) AS total_sales  
  FROM customer_purchases  
  GROUP BY vendor_id, market_date  
)  
SELECT vendor_id, market_date, total_sales,
```

```
LAG(total_sales, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS previous_sales,  
LEAD(total_sales, 1) OVER (PARTITION BY vendor_id ORDER BY market_date) AS next_sales  
FROM vendor_sales;
```