Window functions

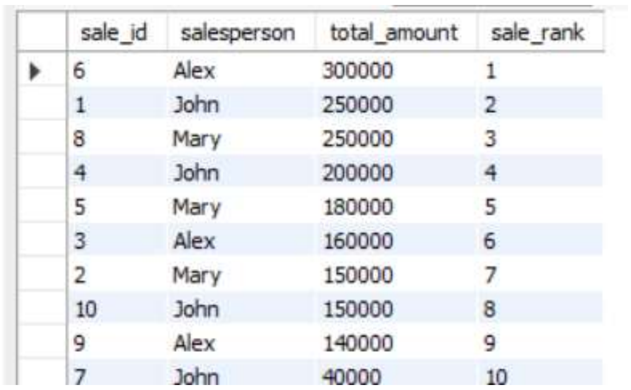
Sample Table: sales_data

sale_id	salesperson	region	sale_date	product	units_sold	total_amount
1	John	North	2024-01-02	Laptop	5	250000
2	Mary	South	2024-01-03	Laptop	3	150000
3	Alex	North	2024-01-05	Tablet	8	160000
4	John	North	2024-02-01	Laptop	4	200000
5	Mary	South	2024-02-04	Tablet	10	180000
6	Alex	East	2024-02-05	Laptop	6	300000
7	John	North	2024-03-02	Tablet	2	40000
8	Mary	South	2024-03-06	Laptop	5	250000
9	Alex	East	2024-03-08	Tablet	7	140000
10	John	North	2024-03-09	Laptop	3	150000

```
CREATE TABLE sales_data (
    sale_id INT PRIMARY KEY,
    salesperson VARCHAR(50),
    region VARCHAR(50),
    sale_date DATE,
    product VARCHAR(50),
    units_sold INT,
    total_amount INT
);
INSERT INTO sales_data
(sale_id, salesperson, region, sale_date, product, units_sold, total_amount)
VALUES
(1,  'John', 'North', '2024-01-02', 'Laptop', 5, 250000),
(2,  'Mary', 'South', '2024-01-03', 'Laptop', 3, 150000),
(3,  'Alex', 'North', '2024-01-05', 'Tablet', 8, 160000),
(4,  'John', 'North', '2024-02-01', 'Laptop', 4, 200000),
(5,  'Mary', 'South', '2024-02-04', 'Tablet', 10, 180000),
(6,  'Alex', 'East',  '2024-02-05', 'Laptop', 6, 300000),
(7,  'John', 'North', '2024-03-02', 'Tablet', 2, 40000),
(8,  'Mary', 'South', '2024-03-06', 'Laptop', 5, 250000),
(9,  'Alex', 'East',  '2024-03-08', 'Tablet', 7, 140000),
(10, 'John', 'North', '2024-03-09', 'Laptop', 3, 150000);
```

1. Write a query to assign a unique rank to each sale in order of total_amount (highest first).

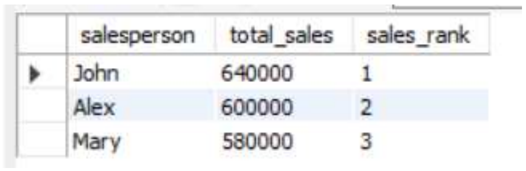
```
SELECT
    sale_id,
    salesperson,
    total_amount,
    ROW_NUMBER() OVER (ORDER BY total_amount DESC) AS sale_rank
FROM sales_data;
```



	sale_id	salesperson	total_amount	sale_rank
▶	6	Alex	300000	1
	1	John	250000	2
	8	Mary	250000	3
	4	John	200000	4
	5	Mary	180000	5
	3	Alex	160000	6
	2	Mary	150000	7
	10	John	150000	8
	9	Alex	140000	9
	7	John	40000	10

2. Find the top 3 salespersons based on total sales amount using the RANK() function.

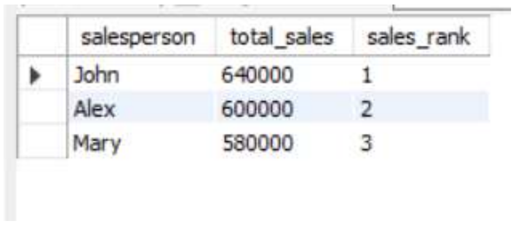
```
SELECT salesperson, total_sales, sales_rank
FROM (
SELECT
    salesperson,
    SUM(total_amount) AS total_sales,
    RANK() OVER (ORDER BY SUM(total_amount) DESC) AS sales_rank
  FROM sales_data
  GROUP BY salesperson
) ranked_sales
WHERE sales_rank <= 3;
```



	salesperson	total_sales	sales_rank
▶	John	640000	1
	Alex	600000	2
	Mary	580000	3

3. Display each salesperson's rank based on their total sales amount, ensuring that rank numbers are consecutive even for ties.

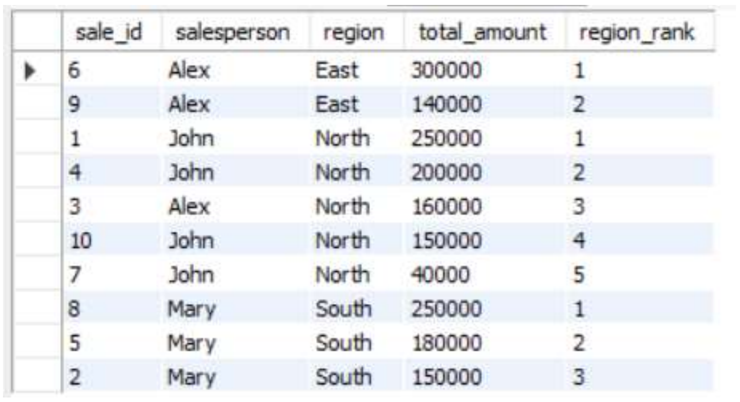
```
SELECT
  salesperson,
  total_sales,
  DENSE_RANK() OVER (ORDER BY total_sales DESC) AS sales_rank
FROM (
  SELECT
    salesperson,
    SUM(total_amount) AS total_sales
  FROM sales_data
  GROUP BY salesperson
) sales_summary;
```



	salesperson	total_sales	sales_rank
▶	John	640000	1
	Alex	600000	2
	Mary	580000	3

4.Show the rank of each salesperson within their region based on total_amount.

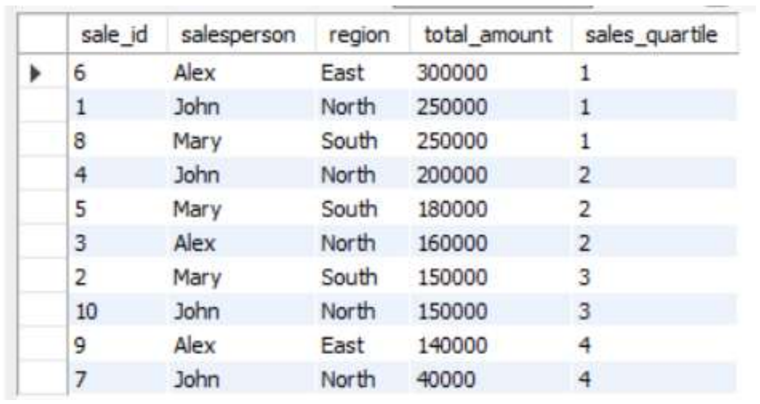
```
SELECT
    sale_id,
    salesperson,
    region,
    total_amount,
    RANK() OVER (
        PARTITION BY region
        ORDER BY total_amount DESC
    ) AS region_rank
FROM sales_data;
```



	sale_id	salesperson	region	total_amount	region_rank
▶	6	Alex	East	300000	1
	9	Alex	East	140000	2
	1	John	North	250000	1
	4	John	North	200000	2
	3	Alex	North	160000	3
	10	John	North	150000	4
	7	John	North	40000	5
	8	Mary	South	250000	1
	5	Mary	South	180000	2
	2	Mary	South	150000	3

5.Divide all sales into 4 quartiles (NTILE(4)) based on total sales amount and show which quartile each sale falls into.

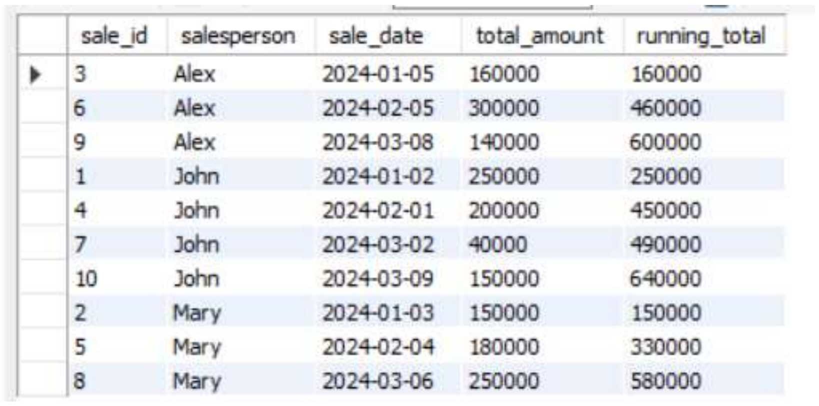
```
SELECT
    sale_id,
    salesperson,
    region,

    total_amount,

    NTILE(4) OVER (ORDER BY total_amount DESC) AS sales_quartile

FROM sales_data;
```



sale_id	salesperson	region	total_amount	sales_quartile
6	Alex	East	300000	1
1	John	North	250000	1
8	Mary	South	250000	1
4	John	North	200000	2
5	Mary	South	180000	2
3	Alex	North	160000	2
2	Mary	South	150000	3
10	John	North	150000	3
9	Alex	East	140000	4
7	John	North	40000	4

6. Calculate the **running total** of total_amount for each salesperson ordered by sale_date.

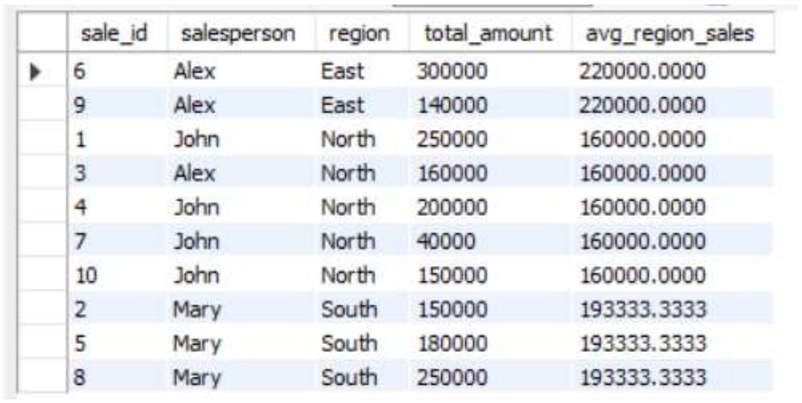
```
SELECT

    sale_id,

    salesperson,

    sale_date,

    total_amount,

    SUM(total_amount) OVER (

        PARTITION BY salesperson

        ORDER BY sale_date

        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

    ) AS running_total

FROM sales_data

ORDER BY salesperson, sale_date;
```



sale_id	salesperson	sale_date	total_amount	running_total
3	Alex	2024-01-05	160000	160000
6	Alex	2024-02-05	300000	460000
9	Alex	2024-03-08	140000	600000
1	John	2024-01-02	250000	250000
4	John	2024-02-01	200000	450000
7	John	2024-03-02	40000	490000
10	John	2024-03-09	150000	640000
2	Mary	2024-01-03	150000	150000
5	Mary	2024-02-04	180000	330000
8	Mary	2024-03-06	250000	580000

7.For each region, find the **average total_amount** using AVG() OVER (PARTITION BY region).

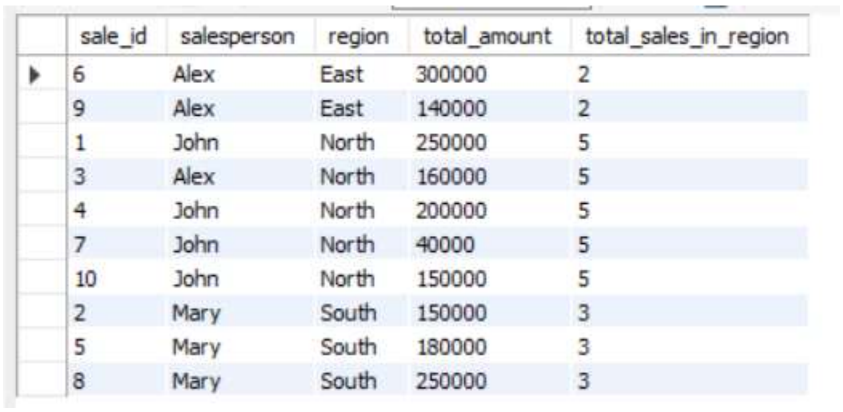
```
SELECT
   sale_id,
   salesperson,
   region,
   total_amount,
   AVG(total_amount) OVER (PARTITION BY region) AS avg_region_sales
FROM sales_data
ORDER BY region, sale_id;
```



sale_id	salesperson	region	total_amount	avg_region_sales
6	Alex	East	300000	220000.0000
9	Alex	East	140000	220000.0000
1	John	North	250000	160000.0000
3	Alex	North	160000	160000.0000
4	John	North	200000	160000.0000
7	John	North	40000	160000.0000
10	John	North	150000	160000.0000
2	Mary	South	150000	193333.3333
5	Mary	South	180000	193333.3333
8	Mary	South	250000	193333.3333

8. For each region, show the total number of sales made (without grouping).

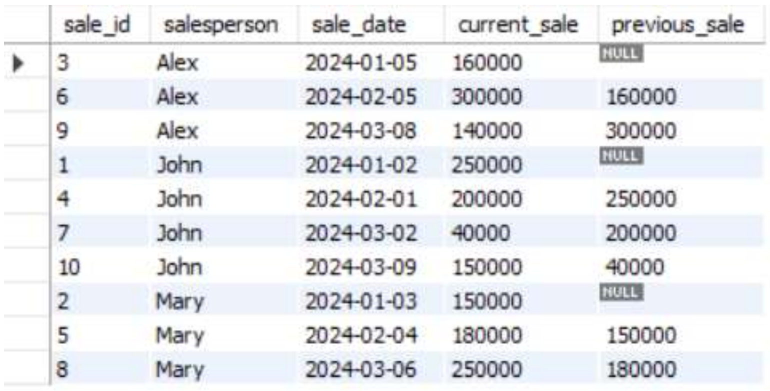
```
SELECT
   sale_id,
   salesperson,
   region,
   total_amount,
    COUNT(*) OVER (PARTITION BY region) AS total_sales_in_region
FROM sales_data
ORDER BY region, sale_id;
```



sale_id	salesperson	region	total_amount	total_sales_in_region
6	Alex	East	300000	2
9	Alex	East	140000	2
1	John	North	250000	5
3	Alex	North	160000	5
4	John	North	200000	5
7	John	North	40000	5
10	John	North	150000	5
2	Mary	South	150000	3
5	Mary	South	180000	3
8	Mary	South	250000	3

9.For each salesperson, display their current sale amount and the previous sale amount using LAG(total_amount).

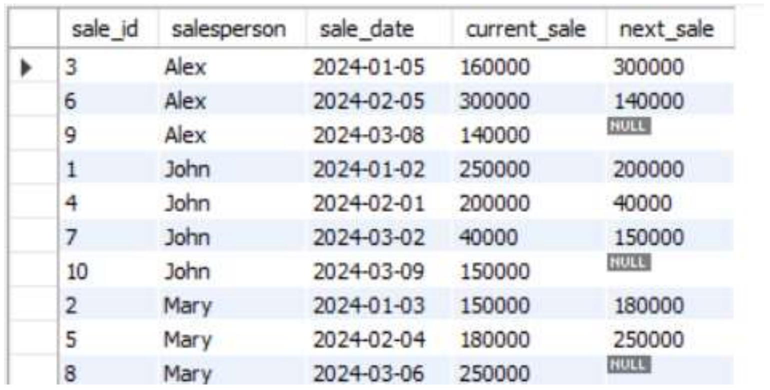
```
SELECT
   sale_id,
   salesperson,
   sale_date,
   total_amount AS current_sale,
   LAG(total_amount) OVER (
      PARTITION BY salesperson
      ORDER BY sale_date
   ) AS previous_sale
FROM sales_data
ORDER BY salesperson, sale_date;
```



sale_id	salesperson	sale_date	current_sale	previous_sale
3	Alex	2024-01-05	160000	NULL
6	Alex	2024-02-05	300000	160000
9	Alex	2024-03-08	140000	300000
1	John	2024-01-02	250000	NULL
4	John	2024-02-01	200000	250000
7	John	2024-03-02	40000	200000
10	John	2024-03-09	150000	40000
2	Mary	2024-01-03	150000	NULL
5	Mary	2024-02-04	180000	150000
8	Mary	2024-03-06	250000	180000

10. For each salesperson, display their sale and the **next sale amount** using LEAD(total_amount).

```
SELECT
    sale_id,
    salesperson,
    sale_date,
    total_amount AS current_sale,
    LEAD(total_amount) OVER (
        PARTITION BY salesperson
        ORDER BY sale_date
    ) AS next_sale
FROM sales_data
ORDER BY salesperson, sale_date;
```

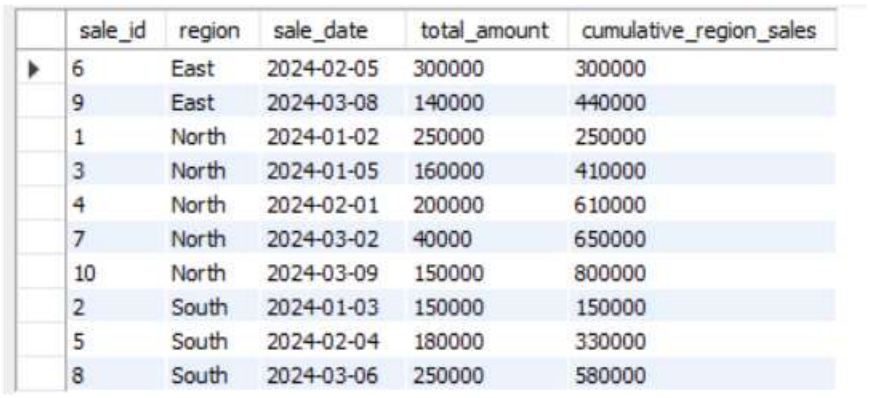


sale_id	salesperson	sale_date	current_sale	next_sale
3	Alex	2024-01-05	160000	300000
6	Alex	2024-02-05	300000	140000
9	Alex	2024-03-08	140000	NULL
1	John	2024-01-02	250000	200000
4	John	2024-02-01	200000	40000
7	John	2024-03-02	40000	150000
10	John	2024-03-09	150000	NULL
2	Mary	2024-01-03	150000	180000
5	Mary	2024-02-04	180000	250000
8	Mary	2024-03-06	250000	NULL

11. Calculate the **cumulative sales** of each region over time (ordered by sale_date).

```
SELECT
```

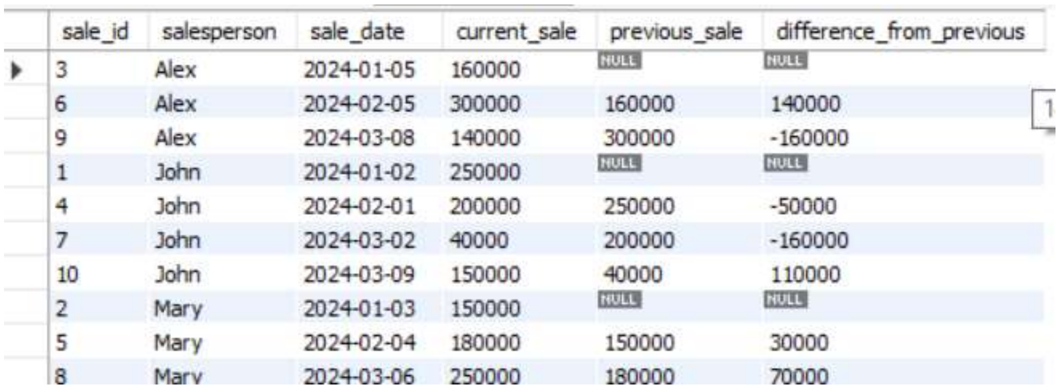
```
   sale_id,
   region,
   sale_date,
   total_amount,
   SUM(total_amount) OVER (
      PARTITION BY region
      ORDER BY sale_date
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
   ) AS cumulative_region_sales
FROM sales_data
ORDER BY region, sale_date;
```



sale_id	region	sale_date	total_amount	cumulative_region_sales
6	East	2024-02-05	300000	300000
9	East	2024-03-08	140000	440000
1	North	2024-01-02	250000	250000
3	North	2024-01-05	160000	410000
4	North	2024-02-01	200000	610000
7	North	2024-03-02	40000	650000
10	North	2024-03-09	150000	800000
2	South	2024-01-03	150000	150000
5	South	2024-02-04	180000	330000
8	South	2024-03-06	250000	580000

12.For each salesperson, find how much more (or less) they sold compared to their previous sale using
total_amount - LAG(total_amount) OVER (PARTITION BY salesperson ORDER BY sale_date).

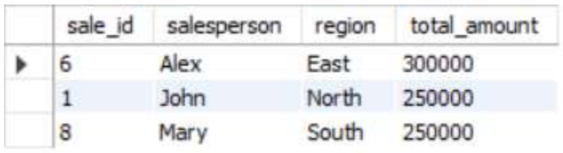
```
SELECT
   sale_id,
   salesperson,
   sale_date,
   total_amount AS current_sale,
   LAG(total_amount) OVER (
      PARTITION BY salesperson
      ORDER BY sale_date
   ) AS previous_sale,
    total_amount - LAG(total_amount) OVER (
        PARTITION BY salesperson
        ORDER BY sale_date
    ) AS difference_from_previous
FROM sales_data
ORDER BY salesperson, sale_date;
```



sale_id	salesperson	sale_date	current_sale	previous_sale	difference_from_previous
3	Alex	2024-01-05	160000	NULL	NULL
6	Alex	2024-02-05	300000	160000	140000
9	Alex	2024-03-08	140000	300000	-160000
1	John	2024-01-02	250000	NULL	NULL
4	John	2024-02-01	200000	250000	-50000
7	John	2024-03-02	40000	200000	-160000
10	John	2024-03-09	150000	40000	110000
2	Mary	2024-01-03	150000	NULL	NULL
5	Mary	2024-02-04	180000	150000	30000
8	Mary	2024-03-06	250000	180000	70000

13.Find the **top-performing salesperson in each region** using a CTE and RANK() OVER (PARTITION BY region ORDER BY total_amount DESC).

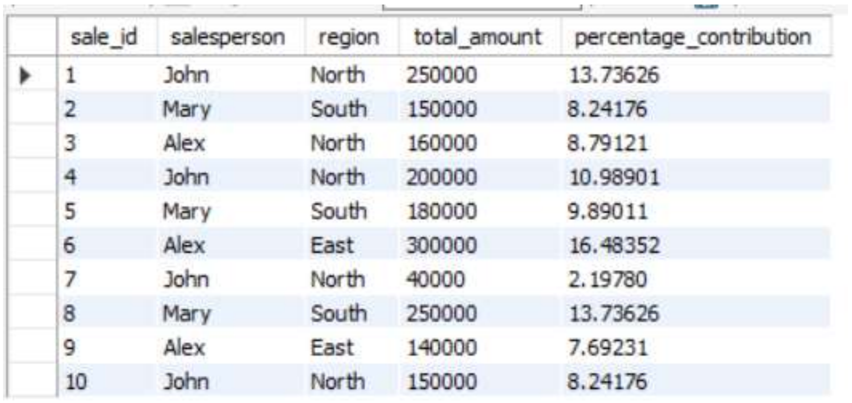
```
WITH ranked_sales AS (
    SELECT
        sale_id,
        salesperson,
        region,
        total_amount,
        RANK() OVER (
            PARTITION BY region
            ORDER BY total_amount DESC
        ) AS region_rank
    FROM sales_data
)
SELECT
    sale_id,
    salesperson,
    region,
    total_amount
FROM ranked_sales
WHERE region_rank = 1;
```



sale_id	salesperson	region	total_amount
6	Alex	East	300000
1	John	North	250000
8	Mary	South	250000

14.Calculate the **percentage contribution** of each sale to the total company sales using (total_amount * 100.0 / SUM(total_amount) OVER ()).

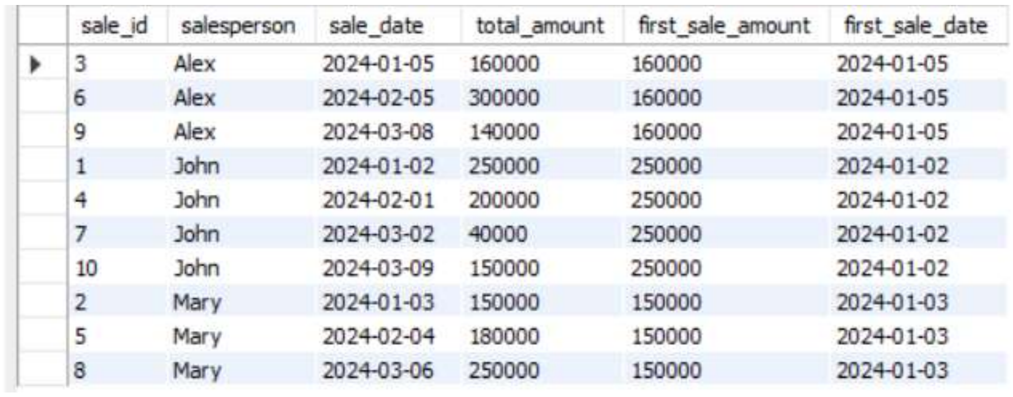
```
SELECT
    sale_id,
    salesperson,
    region,
    total_amount,
    (total_amount * 100.0 / SUM(total_amount) OVER ()) AS percentage_contribution
FROM sales_data
ORDER BY sale_id;
```



sale_id	salesperson	region	total_amount	percentage_contribution
1	John	North	250000	13.73626
2	Mary	South	150000	8.24176
3	Alex	North	160000	8.79121
4	John	North	200000	10.98901
5	Mary	South	180000	9.89011
6	Alex	East	300000	16.48352
7	John	North	40000	2.19780
8	Mary	South	250000	13.73626
9	Alex	East	140000	7.69231
10	John	North	150000	8.24176

15. Using FIRST_VALUE(), show each salesperson's **first sale amount** and sale_date.

```
SELECT
    sale_id,
    salesperson,
    sale_date,

    total_amount,

    FIRST_VALUE(total_amount) OVER (

        PARTITION BY salesperson

        ORDER BY sale_date

        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

    ) AS first_sale_amount,

    FIRST_VALUE(sale_date) OVER (

        PARTITION BY salesperson

        ORDER BY sale_date

        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

    ) AS first_sale_date

FROM sales_data

ORDER BY salesperson, sale_date;
```



sale_id	salesperson	sale_date	total_amount	first_sale_amount	first_sale_date
3	Alex	2024-01-05	160000	160000	2024-01-05
6	Alex	2024-02-05	300000	160000	2024-01-05
9	Alex	2024-03-08	140000	160000	2024-01-05
1	John	2024-01-02	250000	250000	2024-01-02
4	John	2024-02-01	200000	250000	2024-01-02
7	John	2024-03-02	40000	250000	2024-01-02
10	John	2024-03-09	150000	250000	2024-01-02
2	Mary	2024-01-03	150000	150000	2024-01-03
5	Mary	2024-02-04	180000	150000	2024-01-03
8	Mary	2024-03-06	250000	150000	2024-01-03