Window Functions
==========

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);

INSERT INTO employees (employee_id, employee_name, department, salary,
hire_date)
VALUES
(1, 'Amit', 'HR', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 62000, '2022-09-01'),
(7, 'Kiran', 'Finance', 58000, '2021-02-14'),
(8, 'Sunita', 'Finance', 61000, '2023-01-11'),
(9, 'Priya', 'IT', 70000, '2020-12-02'),
(10, 'Anjali', 'IT', 67000, '2021-11-19'),
(11, 'Vikas', 'IT', 69000, '2022-05-20'),
(12, 'Sanjay', 'IT', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 68000, '2021-03-15');


-- find max salary for each department

select department, max(salary) as max_sal
from employees
group by department;

-- employee_id, employee_name

select department, max(salary) as max_sal, employee_id, employee_name
from employees
group by department;


select * from employees e
join (select department, max(salary) as max_sal
from employees
```

```
group by department) d
on e.department = d.department;

select e.employee_id,
e.employee_name,
e.department,
e.salary,
e.hire_date
from employees e
join (select department, max(salary) as max_sal
from employees
group by department) d
on e.department = d.department
and e.salary = d.max_sal;


-- I want top 2 employees from each department

-- the below will not work
select e.employee_id,
e.employee_name,
e.department,
e.salary,
e.hire_date
from employees e
join (select department, max(salary) as max_sal
from employees
group by department) d
on e.department = d.department
and e.salary = d.max_sal
limit 2;


==============


select employee_id,
employee_name,
department,
salary,
hire_date,
ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary) as
row_num
from employees;
```

ROW_NUMBER
RANK
DENSE_RANK

<WINDOW FUNCTION> OVER (PARTITION BY <partition column> ORDER BY <order by col>)

ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary desc)

WHAT IS THE ROLE OF GROUP BY?

to aggregate and for each group we get a single row

but using window functions the input and output
has the same number of rows for each partition

-- we can remove the partition by clause also

```
select employee_id,
employee_name,
department,
salary,
hire_date,
ROW_NUMBER() OVER (ORDER BY salary desc) as row_num
from employees;
```

-- can we remove the order by also?

```
select employee_id,
employee_name,
department,
salary,
hire_date,
ROW_NUMBER() OVER () as row_num
from employees;
```

-- partition by is optional but order by is mandatory

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
```

```
    department VARCHAR(50),
    city VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);

INSERT INTO employees (employee_id, employee_name, department, city,
salary, hire_date)
VALUES
(1, 'Amit', 'HR', 'Mumbai', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 'Mumbai', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 'Delhi', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 'Delhi', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 'Mumbai', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 'Delhi', 62000, '2022-09-01'),
(7, 'Kiran', 'Finance', 'Mumbai', 58000, '2021-02-14'),
(8, 'Sunita', 'Finance', 'Delhi', 61000, '2023-01-11'),
(9, 'Priya', 'IT', 'Mumbai', 70000, '2020-12-02'),
(10, 'Anjali', 'IT', 'Delhi', 67000, '2021-11-19'),
(11, 'Vikas', 'IT', 'Mumbai', 69000, '2022-05-20'),
(12, 'Sanjay', 'IT', 'Delhi', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 'Delhi', 68000, '2021-03-15');

select employee_id,
employee_name,
department,
city,
salary,
hire_date,
ROW_NUMBER() OVER (PARTITION BY department,city ORDER BY salary
desc) as row_num
from employees;

-- inserts with few duplicate salaries
INSERT INTO employees (employee_id, employee_name, department, city,
salary, hire_date)
VALUES
(1, 'Amit', 'HR', 'Mumbai', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 'Mumbai', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 'Delhi', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 'Delhi', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 'Mumbai', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 'Delhi', 60000, '2022-09-01'),
(7, 'Kiran', 'Finance', 'Mumbai', 58000, '2021-02-14'),
(8, 'Sunita', 'Finance', 'Delhi', 60000, '2023-01-11'),
(9, 'Priya', 'IT', 'Mumbai', 70000, '2020-12-02'),
```

```
(10, 'Anjali', 'IT', 'Delhi', 68000, '2021-11-19'),
(11, 'Vikas', 'IT', 'Mumbai', 68000, '2022-05-20'),
(12, 'Sanjay', 'IT', 'Delhi', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 'Delhi', 68000, '2021-03-15');


select employee_id,
employee_name,
department,
city,
salary,
hire_date,
ROW_NUMBER() OVER (PARTITION BY department,city ORDER BY salary
desc, hire_date) as row_num
from employees;

select employee_id,
employee_name,
department,
city,
salary,
hire_date,
ROW_NUMBER() OVER (ORDER BY salary desc, hire_date) as row_num
from employees;


-- the below is not going to work

select employee_id,
employee_name,
department,
city,
salary,
hire_date,
ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary
desc, hire_date) as row_num
from employees
where row_num <=2;

-- using subquery we can achieve it

select * from (select employee_id,
employee_name,
department,
city,
salary,
```

```
hire_date,
ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary
desc, hire_date) as row_num
from employees) e where row_num <=2;


-- convert the above to CTE

with emp_cte as (select employee_id,
employee_name,
department,
city,
salary,
hire_date,
ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary
desc, hire_date) as row_num
from employees)
select * from emp_cte where row_num <=2;
```

group by vs window function

==========

consider we have retail db

6 tables

-- we need to find top selling product in each category

-- products and order_items

-- category_id, product_id, product_name,
-- total_quantity, total_amount

let us join the 2 tables

```
select o.order_item_order_id as order_id,
o.order_item_id ,
o.order_item_product_id as product_id,
o.order_item_quantity,
o.order_item_subtotal,
p.product_category_id,
p.product_name
from order_items o
join products p
```

```
on o.order_item_product_id = p.product_id

-- we need to aggregate the total sale and total quantity
-- sold for each product

with products_agg as (select product_category_id,
product_id,product_name,
sum(order_item_quantity) as total_quantity,
sum(order_item_subtotal) as total_amount
from
(select o.order_item_order_id as order_id,
o.order_item_id ,
o.order_item_product_id as product_id,
o.order_item_quantity,
o.order_item_subtotal,
p.product_category_id,
p.product_name
from order_items o
join products p
on o.order_item_product_id = p.product_id) temp
group by product_category_id,product_id,product_name),
windowed_cte as (select *,
row_number() over (partition by product_category_id order by total_amount
desc) as row_num
from products_agg)
select * from windowed_cte where row_num <=1

============

ROW_NUMBER
RANK
DENSE_RANK

SELECT *,
ROW_NUMBER() OVER (ORDER BY hire_date) as row_num,
RANK() OVER (ORDER BY hire_date) as rnk,
DENSE_RANK() OVER (ORDER BY hire_date) as dense_rnk
FROM EMPLOYEES;

-- when there are no ties/duplicates then all of them will give
-- the same result.

the difference comes when there are ties

SELECT *,
ROW_NUMBER() OVER (ORDER BY salary desc) as row_num,
```

```
RANK() OVER (ORDER BY salary desc) as rnk,
DENSE_RANK() OVER (ORDER BY salary desc) as dense_rnk
FROM EMPLOYEES;

-- 2 people with same value have the same rank and same dense rank

-- 2 people with same value have the same rank and will skip the next rank

RANK()

-- dense rank is similar to rank but this will not skip anything

DENSE_RANK()

-- rank leaves gaps in ranking for tied data.
-- while row_number assigns a unique number to each row without
-- considering the ties.
-- dense_Rank assigns identical ranks to tied data, without any gaps.


-- top 10 customers with max purchase from each city
-- top 3 products with max sales from each city.
-- 3 newest employees from each department
-- top selling product from each category
-- top 3 students in each subject

lead, lag

select * from (SELECT *,
ROW_NUMBER() OVER (ORDER BY salary desc) as row_num,
RANK() OVER (ORDER BY salary desc) as rnk,
DENSE_RANK() OVER (ORDER BY salary desc) as dense_rnk
FROM EMPLOYEES) temp where dense_rnk <=5;

=======


LEAD
LAG


SELECT *,
LEAD(SALARY) OVER (ORDER BY SALARY DESC) as next_sal
LAG(SALARY) OVER (ORDER BY SALARY DESC) as prev_sal
FROM EMPLOYEES;
```

```
SELECT *,
LEAD(SALARY) OVER (ORDER BY SALARY DESC) as next_sal,
LAG(SALARY) OVER (ORDER BY SALARY DESC) as prev_sal
FROM EMPLOYEES;

CREATE TABLE sales (
    sale_date DATE,
    product_id INT,
    product_name VARCHAR(100),
    units_sold INT
);

INSERT INTO sales (sale_date, product_id, product_name, units_sold)
VALUES
('2024-08-15', 1, 'Laptop', 4),
('2024-08-16', 1, 'Laptop', 6),
('2024-08-17', 1, 'Laptop', 5),
('2024-08-18', 1, 'Laptop', 7),
('2024-08-19', 1, 'Laptop', 4),
('2024-08-20', 1, 'Laptop', 5),
('2024-08-21', 1, 'Laptop', 8),
('2024-08-22', 1, 'Laptop', 3),
('2024-08-23', 1, 'Laptop', 7),
('2024-08-24', 1, 'Laptop', 6),
('2024-08-25', 1, 'Laptop', 9),
('2024-08-26', 1, 'Laptop', 7),

('2024-08-15', 2, 'Smartphone', 15),
('2024-08-16', 2, 'Smartphone', 18),
('2024-08-17', 2, 'Smartphone', 12),
('2024-08-18', 2, 'Smartphone', 20),
('2024-08-19', 2, 'Smartphone', 17),
('2024-08-20', 2, 'Smartphone', 14),
('2024-08-21', 2, 'Smartphone', 19),
('2024-08-22', 2, 'Smartphone', 13),
('2024-08-23', 2, 'Smartphone', 21),
('2024-08-24', 2, 'Smartphone', 16),
('2024-08-25', 2, 'Smartphone', 18),
('2024-08-26', 2, 'Smartphone', 20),

('2024-08-15', 3, 'Tablet', 8),
('2024-08-16', 3, 'Tablet', 10),
('2024-08-17', 3, 'Tablet', 7),
('2024-08-18', 3, 'Tablet', 12),
('2024-08-19', 3, 'Tablet', 9),
('2024-08-20', 3, 'Tablet', 8),
```

```
('2024-08-21', 3, 'Tablet', 11),
('2024-08-22', 3, 'Tablet', 6),
('2024-08-23', 3, 'Tablet', 9),
('2024-08-24', 3, 'Tablet', 10),
('2024-08-25', 3, 'Tablet', 13),
('2024-08-26', 3, 'Tablet', 12);


select sale_date,
product_id,
product_name,
units_sold,
lag(units_sold,1) over (partition by product_id order by sale_date)
as previous_day_sales,
lead(units_sold,1) over (partition by product_id order by sale_date)
as next_day_sales
from sales;

-- lead with order by asc

-- lag with order by desc

select sale_date,
product_id,
product_name,
units_sold,
lag(units_sold,1) over (partition by product_id order by sale_date)
as lag_result,
lead(units_sold,1) over (partition by product_id order by sale_date desc)
as lead_result
from sales;

==========


CREATE TABLE social_media_followers (
    user_id INT,
    user_name VARCHAR(100),
    month DATE,
    linkedin_followers INT,
    twitter_followers INT,
    instagram_followers INT,
    youtube_followers INT
);
```

```
INSERT INTO social_media_followers (user_id, user_name, month,
linkedin_followers, twitter_followers, instagram_followers, youtube_followers)
VALUES
(1, 'Rajesh', '2023-08-01', 500, 1200, 800, 900),
(1, 'Rajesh', '2023-09-01', 550, 1300, 850, 950),
(1, 'Rajesh', '2023-10-01', 600, 1400, 900, 1000),
(1, 'Rajesh', '2023-11-01', 660, 1500, 950, 1050),
(1, 'Rajesh', '2023-12-01', 720, 1600, 1000, 1100),
(1, 'Rajesh', '2024-01-01', 790, 1700, 1060, 1160),
(1, 'Rajesh', '2024-02-01', 860, 1800, 1120, 1220),
(1, 'Rajesh', '2024-03-01', 940, 1900, 1190, 1290),
(1, 'Rajesh', '2024-04-01', 1020, 2000, 1260, 1360),
(1, 'Rajesh', '2024-05-01', 1100, 2100, 1330, 1430),
(1, 'Rajesh', '2024-06-01', 1190, 2200, 1410, 1510),
(1, 'Rajesh', '2024-07-01', 1280, 2300, 1490, 1590),
(1, 'Rajesh', '2024-08-01', 1380, 2400, 1580, 1680),

(2, 'Anjali', '2023-08-01', 800, 500, 600, 700),
(2, 'Anjali', '2023-09-01', 820, 510, 620, 710),
(2, 'Anjali', '2023-10-01', 840, 520, 640, 720),
(2, 'Anjali', '2023-11-01', 860, 530, 660, 730),
(2, 'Anjali', '2023-12-01', 880, 540, 680, 740),
(2, 'Anjali', '2024-01-01', 900, 550, 700, 750),
(2, 'Anjali', '2024-02-01', 920, 560, 720, 760),
(2, 'Anjali', '2024-03-01', 940, 570, 740, 770),
(2, 'Anjali', '2024-04-01', 960, 580, 760, 780),
(2, 'Anjali', '2024-05-01', 980, 590, 780, 790),
(2, 'Anjali', '2024-06-01', 1000, 600, 800, 800),
(2, 'Anjali', '2024-07-01', 1020, 610, 820, 810),
(2, 'Anjali', '2024-08-01', 1040, 620, 840, 820);


-- calculate % of total follower gains for each user month on month

select *,
lead(total_followers,1) over (partition by user_id order by month) as
next_mnth,
lag(total_followers,1) over (partition by user_id order by month) as prev_mnth
from (select user_id,
user_name,
month,
linkedin_followers + twitter_followers + instagram_followers +
youtube_followers
as total_followers
from social_media_followers) temp
```

```
with cte1 as (select user_id,
user_name,
month,
linkedin_followers + twitter_followers + instagram_followers +
youtube_followers
as total_followers
from social_media_followers),
cte2 as (select *,
lead(total_followers,1) over (partition by user_id order by month) as
next_mnth,
lag(total_followers,1) over (partition by user_id order by month) as prev_mnth
from cte1)
select *,
round((total_followers - prev_mnth)*100 / prev_mnth ,2 ) as gain_percent
from cte2;
```

==========

ROW_NUMBER
RANK
DENSE_RANK
lead
lag

```
drop table employees;

CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);

INSERT INTO employees (employee_id, employee_name, department, salary,
hire_date)
VALUES
(1, 'Amit', 'HR', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 62000, '2022-09-01'),
(7, 'Kiran', 'Finance', 58000, '2021-02-14'),
```

(8, 'Sunita', 'Finance', 61000, '2023-01-11'),
(9, 'Priya', 'IT', 70000, '2020-12-02'),
(10, 'Anjali', 'IT', 67000, '2021-11-19'),
(11, 'Vikas', 'IT', 69000, '2022-05-20'),
(12, 'Sanjay', 'IT', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 68000, '2021-03-15');

```
select employee_id,
employee_name,
department,
salary,
hire_date,
row_number() over (partition by department order by salary desc) as row_num
from employees;
```

but what if we have a requirement to find total salary paid for each department

ideally how you approach this based on whatever you have learnt till now?

dept_agg
HR , 241000
Finanace, 205000
IT, 250000

employees
(1, 'Amit', 'HR', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 62000, '2022-09-01'),
(7, 'Kiran', 'Finance', 58000, '2021-02-14'),
(8, 'Sunita', 'Finance', 61000, '2023-01-11'),
(9, 'Priya', 'IT', 70000, '2020-12-02'),
(10, 'Anjali', 'IT', 67000, '2021-11-19'),
(11, 'Vikas', 'IT', 69000, '2022-05-20'),
(12, 'Sanjay', 'IT', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 68000, '2021-03-15');

```
select e.*,d.total_sal
from employees e
join
(
select department, sum(salary) as total_sal
```

```
from employees
group by department
) d
on e.department = d.department;
```

can we achieve the above without using joins?

row_number
rank
dense_rank
lead
lag

we can do it using window functions

we can use aggregations as part of window functions

```
select *,
sum(salary) over (partition by department) as total_sal
from employees

-- row_number() over (partition by department order by salary desc)
-- sum(salary) over (partition by department)
```

here partition by is optional
now order by is also optional

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department VARCHAR(50),
    city VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);

INSERT INTO employees (employee_id, employee_name, department, city,
salary, hire_date)
VALUES
(1, 'Amit', 'HR', 'Mumbai', 50000, '2022-01-15'),
(2, 'Neha', 'HR', 'Mumbai', 55000, '2023-03-10'),
(3, 'Suresh', 'HR', 'Delhi', 48000, '2021-11-20'),
(4, 'Rohit', 'HR', 'Delhi', 52000, '2022-09-05'),
(5, 'Raj', 'Finance', 'Mumbai', 60000, '2021-07-23'),
(6, 'Ravi', 'Finance', 'Delhi', 62000, '2022-09-01'),
(7, 'Kiran', 'Finance', 'Mumbai', 58000, '2021-02-14'),
```

```
(8, 'Sunita', 'Finance', 'Delhi', 61000, '2023-01-11'),
(9, 'Priya', 'IT', 'Mumbai', 70000, '2020-12-02'),
(10, 'Anjali', 'IT', 'Delhi', 67000, '2021-11-19'),
(11, 'Vikas', 'IT', 'Mumbai', 69000, '2022-05-20'),
(12, 'Sanjay', 'IT', 'Delhi', 72000, '2023-04-30'),
(13, 'Meena', 'IT', 'Delhi', 68000, '2021-03-15');

-- total salary for each city
select *,
sum(salary) over (partition by city) as total_sal
from employees

select city, sum(salary)
from employees
group by city

-- total salary for each department in each city
select *,
sum(salary) over (partition by city, department) as total_sal
from employees

select *,
sum(salary) over (partition by department) as total_sal,
min(salary) over (partition by department) as min_sal,
max(salary) over (partition by department) as max_sal,
avg(salary) over (partition by department) as avg_sal,
count(salary) over (partition by department) as department_count
from employees


select *,
sum(salary) over () as total_sal,
min(salary) over () as min_sal,
max(salary) over () as max_sal,
avg(salary) over () as avg_sal,
count(salary) over () as department_count
from employees



<agg-func>(<column_name>) over ()


-- running total
select *,
sum(salary) over (order by hire_date) as total_sal
```

```
from employees;


CREATE TABLE sales (
    sale_date DATE,
    product_id INT,
    product_name VARCHAR(100),
    units_sold INT
);

INSERT INTO sales (sale_date, product_id, product_name, units_sold)
VALUES
('2024-08-15', 1, 'Laptop', 4),
('2024-08-16', 1, 'Laptop', 6),
('2024-08-17', 1, 'Laptop', 5),
('2024-08-18', 1, 'Laptop', 7),
('2024-08-19', 1, 'Laptop', 4),
('2024-08-20', 1, 'Laptop', 5),
('2024-08-21', 1, 'Laptop', 8),
('2024-08-22', 1, 'Laptop', 3),
('2024-08-23', 1, 'Laptop', 7),
('2024-08-24', 1, 'Laptop', 6),
('2024-08-25', 1, 'Laptop', 9),
('2024-08-26', 1, 'Laptop', 7),

('2024-08-15', 2, 'Smartphone', 15),
('2024-08-16', 2, 'Smartphone', 18),
('2024-08-17', 2, 'Smartphone', 12),
('2024-08-18', 2, 'Smartphone', 20),
('2024-08-19', 2, 'Smartphone', 17),
('2024-08-20', 2, 'Smartphone', 14),
('2024-08-21', 2, 'Smartphone', 19),
('2024-08-22', 2, 'Smartphone', 13),
('2024-08-23', 2, 'Smartphone', 21),
('2024-08-24', 2, 'Smartphone', 16),
('2024-08-25', 2, 'Smartphone', 18),
('2024-08-26', 2, 'Smartphone', 20),

('2024-08-15', 3, 'Tablet', 8),
('2024-08-16', 3, 'Tablet', 10),
('2024-08-17', 3, 'Tablet', 7),
('2024-08-18', 3, 'Tablet', 12),
('2024-08-19', 3, 'Tablet', 9),
('2024-08-20', 3, 'Tablet', 8),
('2024-08-21', 3, 'Tablet', 11),
('2024-08-22', 3, 'Tablet', 6),
```

```
('2024-08-23', 3, 'Tablet', 9),
('2024-08-24', 3, 'Tablet', 10),
('2024-08-25', 3, 'Tablet', 13),
('2024-08-26', 3, 'Tablet', 12);


-- running total for each product

select *,
sum(units_sold) over (partition by product_id order by sale_date)
as running_total
from sales;

-- what if I remove partition by
-- in case of duplicates it consider it as one group
select *,
sum(units_sold) over (order by sale_date)
as running_total
from sales;

-- you can always give a second column in order to remove the ties.
select *,
sum(units_sold) over (order by sale_date,product_id)
as running_total
from sales;


select *,
sum(units_sold) over (partition by product_id order by sale_date)
as running_total
from sales;

-- what if I need to have a moving window of 3 days

select *,
sum(units_sold) over (partition by product_id order by sale_date
rows between 2 preceding and current row)
from sales;


select *,
sum(units_sold) over (partition by product_id order by sale_date desc
rows between current row and 2 following) as 3_day_sales
from sales;
```

20
19
18
17
16
15

what if I say

```
select *,
sum(units_sold) over (partition by product_id order by sale_date
rows between 2 preceding and 2 following) as 5_day_sales
from sales;

select *,
sum(units_sold) over (partition by product_id order by sale_date
rows between unbounded preceding and current row) as 5_day_sales
from sales;

select *,
sum(units_sold) over (partition by product_id order by sale_date
rows between unbounded preceding and current row) as running_total1,
sum(units_sold) over (partition by product_id order by sale_date)
as running_total2
from sales;

-- this is the default behaviour also

select *,
sum(units_sold) over (partition by product_id order by sale_date desc
rows between current row and unbounded following) as running_total
from sales;


select *,
sum(units_sold) over (partition by product_id order by sale_date desc
rows between unbounded preceding and unbounded following) as
running_total
from sales;

select *,
sum(units_sold) over (partition by product_id order by sale_date desc
rows between unbounded preceding and unbounded following) as
running_total1,
sum(units_sold) over (partition by product_id) as running_total2
from sales;
```

sum(salary) over (<partition by> <order by>)

rows between

current row
2 preceding
2 following
unbounded preceding
unbounded following

==========

https://leetcode.com/problems/biggest-window-between-visits/

UserVisits table:

user_id	visit_date
1	2020-11-28
1	2020-10-20
1	2020-12-3
2	2020-10-5
2	2020-12-9
3	2020-11-11

user_id	visit_date	next_visit_date
1	2020-10-20	2020-11-28
1	2020-11-28	2020-12-03
1	2020-12-03	2021-01-1
2	2020-10-05	2020-12-09
2	2020-12-09	2021-01-1
3	2020-11-11	2021-01-1

```
select user_id,
visit_date,
lead(visit_date,1,'2021-01-1') over (partition by user_id order by visit_date)
as next_visit_date
from uservisits;
```

user_id	date_diff
1	39
1	5
1	29
2	65
2	23
3	51

```
with cte1 as (select user_id,
visit_date,
lead(visit_date,1,'2021-01-1') over (partition by user_id order by visit_date)
as next_visit_date
from uservisits),
cte2 as (select user_id,
    datediff(next_visit_date,visit_date) as date_diff from cte1)
select user_id,max(date_diff) as biggest_window
from cte2
group by user_id;
```

-- using subquery

user_id	visit_date	next_visit_date
1	2020-10-20	2020-11-28
1	2020-11-28	2020-12-03
1	2020-12-03	2021-01-1
2	2020-10-05	2020-12-09
2	2020-12-09	2021-01-1
3	2020-11-11	2021-01-1

```
select user_id,
visit_date,
lead(visit_date,1,'2021-01-1') over (partition by user_id order by visit_date)
as next_visit_date
from uservisits
```

user_id	date_diff
1	39
1	5
1	29

2	65
2	23
3	51

```
select user_id, max(datediff(next_visit_date, visit_date)) as biggest_window
from (select user_id,
visit_date,
lead(visit_date,1,'2021-01-1') over (partition by user_id order by visit_date)
as next_visit_date
from uservisits) c
group by user_id;
```

user_id	biggest_window
1	39
2	65
3	51

==========

https://leetcode.com/problems/tournament-winners/

approach 1

Players table:

player_id	group_id
15	1
25	1
30	1
45	1
10	2
35	2
50	2
20	3
40	3

Matches table:

match_id	first_player	second_player	first_score	second_score

1	15	45	3	0
2	30	25	1	2
3	30	15	2	0
4	40	20	5	2
5	35	50	1	1

match_id	player	score
1	15	3
2	30	1
3	30	2
4	40	5
5	35	1
1	45	0
2	25	2
3	15	0
4	20	2
5	50	1

```
with cte as (select match_id,
first_player as player,
first_score as score
from matches
union all
select match_id,
second_player as player,
second_score as score
from matches)
```

match_id	player	score	group_id
1	15	3	1
3	15	0	1
4	20	2	3
2	25	2	1
2	30	1	1
3	30	2	1
5	35	1	2
4	40	5	3
1	45	0	1
5	50	1	2

```
with cte as (select match_id,
first_player as player,
first_score as score
from matches
union all
select match_id,
second_player as player,
second_score as score
from matches),
cte_joined as (select c.*, p.group_id
from cte c
join players p
on c.player = p.player_id)
```

group_id	player	total_score
1	15	3
3	20	2
1	25	2
1	30	3
2	35	1
3	40	5
1	45	0
2	50	1

```
with cte as (select match_id,
first_player as player,
first_score as score
from matches
union all
select match_id,
second_player as player,
second_score as score
from matches),
cte_joined as (select c.*, p.group_id
from cte c
join players p
on c.player = p.player_id),
cte_grouped as (select group_id, player, sum(score) as total_score
from cte_joined
group by group_id, player)
```

group_id	player	rnk
1	15	1
1	30	2
1	25	3
1	45	4
2	35	1
2	50	2
3	40	1
3	20	2

```
with cte as (select match_id,
first_player as player,
first_score as score
from matches
union all
select match_id,
second_player as player,
second_score as score
from matches),
cte_joined as (select c.*, p.group_id
from cte c
join players p
on c.player = p.player_id),
cte_grouped as (select group_id, player, sum(score) as total_score
from cte_joined
group by group_id, player),
cte_windowed as (select group_id, player,
rank() over (partition by group_id order by total_score desc, player)
as rnk from cte_grouped)
select group_id, player as player_id
from cte_windowed where rnk = 1
```

approach 2

player_id	group_id	match_id	first_player	second_player	first_score	second_score
15	1	3	30	15	2	0
15	1	1	15	45	3	0
20	3	4	40	20	5	2

25	1	2	30	25	1	2
30	1	3	30	15	2	0
30	1	2	30	25	1	2
35	2	5	35	50	1	1
40	3	4	40	20	5	2

```
select p.*, m.*
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
```

group_id	player_id	total_score
1	15	0
1	15	3
3	20	2
1	25	2
1	30	2
1	30	1
2	35	1
3	40	5
1	45	0
2	50	1

```
select p.group_id, p.player_id,
case when p.player_id = m.first_player then m.first_score
else m.second_score
end as total_score
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
```

group_id	player_id	total_score
1	15	3
3	20	2
1	25	2
1	30	3

2	35	1
3	40	5
1	45	0
2	50	1

```
select p.group_id, p.player_id,
sum(case when p.player_id = m.first_player then m.first_score
else m.second_score
end) as total_score
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
group by p.group_id, p.player_id
```

group_id	player_id	total_score	rnk
1	15	3	1
1	30	3	2
1	25	2	3
1	45	0	4
2	35	1	1
2	50	1	2
3	40	5	1
3	20	2	2

```
select p.group_id, p.player_id,
sum(case when p.player_id = m.first_player then m.first_score
else m.second_score
end) as total_score,
rank() over (partition by p.group_id order by
    sum(case when p.player_id = m.first_player then m.first_score
else m.second_score
end) desc, player_id) as rnk
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
group by p.group_id, p.player_id
```

or

```
select group_id, player_id, total_score,
rank() over (partition by group_id order by total_score desc, player_id) as rnk
```

```
from
(select p.group_id, p.player_id,
sum(case when p.player_id = m.first_player then m.first_score
else m.second_score
end) as total_score
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
group by p.group_id, p.player_id) c
```

group_id	player_id
1	15
2	35
3	40

```
select group_id, player_id from
( select group_id, player_id, total_score,
rank() over (partition by group_id order by total_score desc, player_id) as rnk
from
(select p.group_id, p.player_id,
sum(case when p.player_id = m.first_player then m.first_score
else m.second_score
end) as total_score
from players p
join matches m
on p.player_id in (m.first_player, m.second_player)
group by p.group_id, p.player_id) c ) final
where rnk = 1;
```

==========

https://leetcode.com/problems/longest-winning-streak/

Matches table:

player_id	match_day	result
1	2022-01-17	Win
1	2022-01-18	Win
1	2022-01-25	Win
1	2022-01-31	Draw
1	2022-02-08	Win

2	2022-02-06	Lose
2	2022-02-08	Lose
3	2022-03-30	Win
+----------+------------+--------+

step 1
========

player_id	match_day	result	grp_num
1	2022-01-17	Win	0
1	2022-01-18	Win	0
1	2022-01-25	Win	0
1	2022-01-31	Draw	1
1	2022-02-08	Win	1
2	2022-02-06	Lose	1
2	2022-02-08	Lose	2
3	2022-03-30	Win	0

```
select *,
sum(result!='Win') over (partition by player_id order by match_day) as
grp_num
from matches
```

step 2
=========

player_id	grp_num	streak
1	0	3
1	1	1
3	0	1

```
with grp_cte as (select *,
sum(result!='Win') over (partition by player_id order by match_day) as
grp_num
from matches)
select player_id, grp_num,
count(1) as streak from grp_cte where result = 'Win'
group by player_id, grp_num;
```

step 3
========

| player_id | streak | rnum |

| --- | --- | --- |
1	3	1
1	1	2
3	1	1

```
with grp_cte as (select *,
sum(result!='Win') over (partition by player_id order by match_day) as
grp_num
from matches),
window_cte as (select player_id,
count(1) as streak,
row_number() over (partition by player_id order by count(1) desc) as rnum
from grp_cte where result = 'Win'
group by player_id, grp_num)
```

step 4
==========

player_id	longest_streak
1	3
2	0
3	1

```
with grp_cte as (select *,
sum(result!='Win') over (partition by player_id order by match_day) as
grp_num
from matches),
window_cte as (select player_id,
count(1) as streak,
row_number() over (partition by player_id order by count(1) desc) as rnum
from grp_cte where result = 'Win'
group by player_id, grp_num)
select m.player_id,
case when w.player_id is null
then 0
else w.streak
end as longest_streak
from (select distinct player_id from matches) m
left join (select * from window_cte where rnum = 1) w
on m.player_id = w.player_id
```

==========

https://leetcode.com/problems/top-three-wineries/description/

Wineries table:

id	country	points	winery
103	Australia	84	WhisperingPines
737	Australia	85	GrapesGalore
848	Australia	100	HarmonyHill
222	Hungary	60	MoonlitCellars
116	USA	47	RoyalVines
124	USA	45	EaglesNest
648	India	69	SunsetVines
894	USA	39	RoyalVines
677	USA	9	PacificCrest

step 1
========

country	winery	total_points
Australia	WhisperingPines	84
Australia	GrapesGalore	85
Australia	HarmonyHill	100
Hungary	MoonlitCellars	60
USA	RoyalVines	86
USA	EaglesNest	45
India	SunsetVines	69
USA	PacificCrest	9

```
select country, winery, sum(points) as total_points
from wineries
group by country, winery
```

step 2
=======

country	winery	rnk
Australia	HarmonyHill (100)	1
Australia	GrapesGalore (85)	2
Australia	WhisperingPines (84)	3

Hungary	MoonlitCellars (60)	1
India	SunsetVines (69)	1
USA	RoyalVines (86)	1
USA	EaglesNest (45)	2
USA	PacificCrest (9)	3

```
with points_agg as (select country, winery, sum(points) as total_points
from wineries
group by country, winery)
select country,
concat(winery, " (", total_points, ")") as winery,
rank() over (partition by country order by total_points desc, winery) as rnk
from points_agg
```

step 3
========

country	top_winery	second_winery	third_winery
Australia	HarmonyHill (100)	GrapesGalore (85)	WhisperingPines (84)
Hungary	MoonlitCellars (60)	No second winery	No third winery
India	SunsetVines (69)	No second winery	No third winery
USA	RoyalVines (86)	EaglesNest (45)	PacificCrest (9)

```
with points_agg as (select country, winery, sum(points) as total_points
from wineries
group by country, winery),
cte_ranked as (select country,
concat(winery, " (", total_points, ")") as winery,
rank() over (partition by country order by total_points desc, winery) as rnk
from points_agg)
select country,
max(case when rnk = 1 then winery end) as top_winery,
coalesce(max(case when rnk = 2 then winery else null end),'No second
winery')
   as second_winery,
coalesce(max(case when rnk = 3 then winery else null end),'No third winery')
   as third_winery,
from cte_ranked
group by country;
```