Important SQL Interview Questions

Find the employees having salary more than thier manager salary

```
Select employee.emp_id, employee.emp_name, M.emp_name as manager_name,
employee.salary, M.salary as Manager_salary
from employee
Inner Join employee M On employee.manager_id = M.emp_id
Where employee.salary > M.salary;
```

	emp_id [PK] bigint	emp_name character varying (100)	manager_name character varying (100)	salary bigint	manager_salary bigint
1	1	ankit	rohit	10000	5000
2	2	mohit	mudit	15000	12000
3	3	vikas	rohit	12000	5000
4	5	mudit	agam	12000	9000

Find the employee with second highest salary from each department

```
Select * from (select emp.*, dense_rank() Over(Partition By department_id Order By salary Desc ) as rn from emp) a
where rn= 2;
```

	emp_id integer	emp_name character varying (20)	department_id integer	salary integer	manager_id integer	emp_age integer	rn bigint
1	1	Ankit	100	10000	4	39	2
2	3	Vikas	100	10000	4	37	2
3	7	Sanjay	200	9000	2	13	2

find employees who are not present in the depratment table

```
Select emp.*, dept.dept_id, dept.dept_name
from emp
Left Join dept On emp.department_id = dept.dept_id
where dept.dept_id is Null;
```

	emp_id integer	emp_name character varying (20)	department_id integer	salary integer	manager_id integer	emp_age integer	dept_id integer	dept_name character varying (20)
1	5	Mudit	200	12000	6	55	[null]	[null]
2	6	Agam	200	12000	2	14	[null]	[null]
3	7	Sanjay	200	9000	2	13	[null]	[null]
4	8	Ashish	200	5000	2	12	[null]	[null]

How to find the mode

```
--Method 1
With Freq_cte as(
Select id, count(*) as freq from mode group By id)
Select * from freq_cte
where freq = (select Max(freq) from Freq_cte);
```

```
--Method 2
With Freq_cte as(
Select id, count(*) as freq from mode group By id),
rnk_cte as(
Select *, rank() over(order By freq desc) as rn
from Freq_cte)
Select * from rnk_cte where rn = 1
```

	id integer	freq bigint	rn bigint
1	3	4	1

find the change in employee status from 2020 to 2021 -> Example of full outer join

```
SELECT e20.*, e21.*,
    CASE
        WHEN e20.designation IS DISTINCT FROM e21.designation THEN 'Promoted'
        WHEN e21.designation IS NULL THEN 'Resigned'
        ELSE 'New'
    END AS comment
FROM emp_2020 e20
FULL OUTER JOIN emp_2021 e21 ON e20.emp_id = e21.emp_id
WHERE COALESCE(e20.designation, 'XXX') != COALESCE(e21.designation, 'YYY');
```

	emp_id integer	designation character varying (20)	emp_id integer	designation character varying (20)	comment text
1	1	Trainee	1	Developer	Promoted
2	3	Senior Developer	3	Manager	Promoted
3	4	Manager	[null]	[null]	Promoted
4	[null]	[null]	5	Trainee	Promoted

Rank the duplicate records

```
With cte_dup as(
Select id from list group by id Having count(1)>1),
cte_rank as(
select id, rank() Over(order by id asc) as rn from cte_dup)
select l.id, 'DUP' || cast(cr.rn as varchar(2)) as output
from list l
Left Join cte_rank cr On l.id = cr.id
```

	id character varying (5)	output text
1	a	DUP1
2	a	DUP1
3	b	[null]
4	c	DUP2
5	c	DUP2
6	c	DUP2
7	d	DUP3
8	d	DUP3
9	e	[null]

find the running sum

```
Select *,
sum(cost) Over(order by cost asc, product_id) as running_cost
from product;
```

	product_id character varying (20)	cost integer	running_cost bigint
1	P1	200	200
2	P2	300	500
3	P3	300	800
4	P4	500	1300
5	P5	800	2100

Identify and delete excat duplicate record

```
--method1
create table empl_backUp as Select * from empl;
Delete from empl;
Insert Into empl Select distinct * from empl_backUP;
Select * from empl
```

```
--method2
Insert Into empl
select id,name, salary, joining_date from
(Select  *, row_number() Over(partition by id order by
salary) as rn  from empl_backUP ) A
where rn =1;
```

	id integer	name character varying (100)	salary numeric	joining_date integer
1	1	Ankit	20000	2002
2	2	rohit	30000	1999
3	3	vinit	40000	1992

NTILE function

```
with cte as(
Select  Customer_Name,region, sum(sales) as total_Sales
from Order_A
group by Customer_Name, region
Order By region, total_sales desc
Limit 18)
Select * from (Select * ,
NTILE(4) Over (Partition By region Order by total_Sales) as cust_groups
from cte) a
where cust_groups IN(1)
```

	customer_name character varying (100)	region character varying (100)	total_sales numeric	cust_groups integer
1	Bill Eplett	Central	3501.90	1
2	Dianna Wilson	Central	3760.41	1
3	Zuschuss Carroll	Central	3869.80	1
4	Philisse Overcash	Central	3910.59	1
5	Ken Lonsdale	Central	4517.30	1

Find no of goldmedal per swimmer for swimmers who only won gold medals.

Select gold as player_name, count(1) as no_of_medals
from events
where gold not in (Select silver from events Union All Select bronze from events)
group by gold;

	player_name character varying (255)	no_of_medals bigint
1	Ronald	1
2	Amthhew Mcgarray	1
3	Thomas	3
4	Charles	3
5	jessica	1

Find no of goldmedal per swimmer for swimmers who only won gold medals.

```
Select gold as player_name, count(1) as no_of_medals
from events
where gold not in (Select silver from events Union All Select bronze from events)
group by gold;
```

```
--using CTE
With cte as (Select gold as player_name, 'gold' as medal_type from events
Union All Select silver, 'silver' as medal_type from events
Union All Select bronze, 'bronze' as medal_type from events)
Select player_name, count(1) as no_of_gold_medals
from cte
group by player_name
having count (distinct medal_type) = 1 and max(medal_type) = 'gold';
```

	player_name character varying (255)	no_of_medals bigint
1	Ronald	1
2	Amthhew Mcgarray	1
3	Thomas	3
4	Charles	3
5	jessica	1

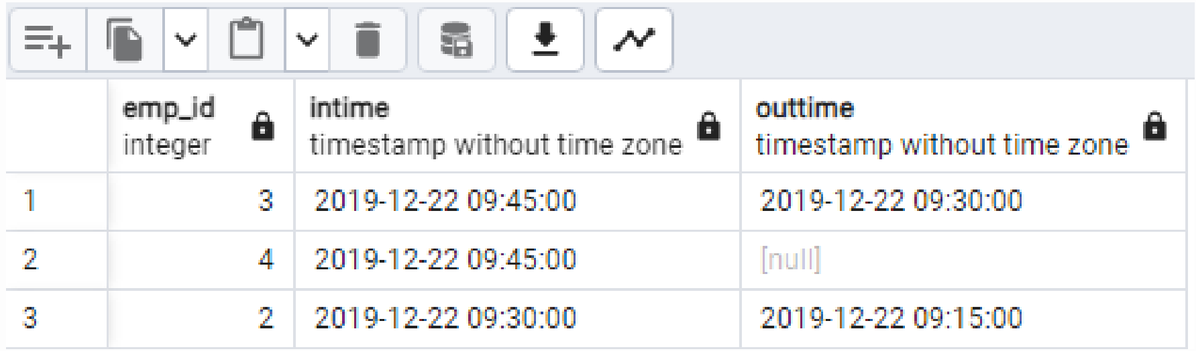
find business days between create days and resolved days excluding weekends and public holidays

```
SELECT
    t.ticket_id,
    t.create_date,
    t.resolved_date,
    COUNT(*) AS business_days
FROM
    tickets t
JOIN
    generate_series(t.create_date, t.resolved_date, interval '1 day') AS dates
    ON EXTRACT(ISODOW FROM dates) < 6 -- Exclude Saturday (6) and Sunday (7)
LEFT JOIN
    holidays h
    ON dates = h.holiday_date
WHERE
    h.holiday_date IS NULL
GROUP BY
    t.ticket_id, t.create_date, t.resolved_date;
```

	ticket_id character varying (10)	create_date date	resolved_date date	business_days bigint
1	1	2022-08-01	2022-08-03	3
2	2	2022-08-01	2022-08-12	9
3	3	2022-08-01	2022-08-16	10

find the total no of people present inside the hospital

```
--method1
with cte as(
select emp_id,
max(case when action='in' then time end) as intime,
max(case when action = 'out' then time end) as outtime
from hospital
group by emp_id)
Select * from cte
where intime > outtime or outtime is null
```



	emp_id integer	intime timestamp without time zone	outtime timestamp without time zone
1	3	2019-12-22 09:45:00	2019-12-22 09:30:00
2	4	2019-12-22 09:45:00	[null]
3	2	2019-12-22 09:30:00	2019-12-22 09:15:00

```
--method 2
with latest_time as(
select emp_id, max(time) as max_latest_time from hospital group by emp_id),
latest_in_time as (select emp_id, max(time) as max_in_time from hospital
    where action = 'in' group by emp_id)
select * from latest_time lt
inner Join latest_in_time lit on
lt.emp_id=lit.emp_id and max_latest_time = max_in_time;
```

return all employees whose salary same in same department

```
with sal_dep as (
select dept_id, salary
from emp_salary
group by dept_id, salary
having count(1)>1)
Select es.*
from emp_salary es
inner join sal_dep sd ON es.dept_id= sd.dept_id and es.salary = sd.salary
```

```
-- left join method 2
with sal_dep as (
select dept_id, salary
from emp_salary
group by dept_id, salary
having count(1)=1)
Select es.*
from emp_salary es
left join sal_dep sd ON es.dept_id= sd.dept_id and es.salary = sd.salary
where sd.dept_id is null
```

	emp_id integer	name character varying (20)	salary character varying (30)	dept_id integer
1	101	sohan	3000	11
2	102	rohan	4000	12
3	104	cat	3000	11
4	105	suresh	4000	12

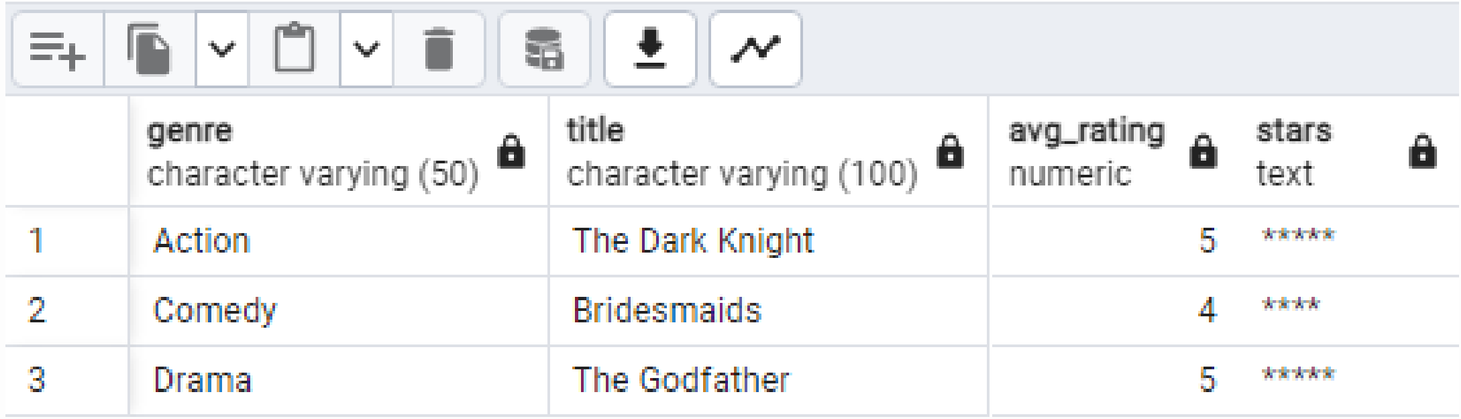
find the words whicch are repeating more than once considering all the rows of content column

```
SELECT word, COUNT(*) AS cnt_of_words
FROM (
    SELECT regexp_split_to_table(content, ' ') AS word
    FROM namaste_python
) AS split_words
GROUP BY word
HAVING COUNT(*) > 1
ORDER BY cnt_of_words desc;
```

	word text	cnt_of_words bigint
1	to	3
2	from	2
3	for	2
4	on	2

find, within each genre, the movie which have highest average rating and also print the star *****

```
with cte as(
 select m.genre, m.title,  round(avg(r.rating),0)  as avg_rating,
row_number() over(partition by m.genre order by avg(r.rating) desc) as rn
from movies m
inner join reviews r On m.id = r.movie_id
group by m.genre, m.title
order by m.genre, avg_rating desc)
select genre, title, avg_rating
, Repeat('*',avg_rating::int) as stars
from cte
where rn = 1
```



	genre character varying (50)	title character varying (100)	avg_rating numeric	stars text
1	Action	The Dark Knight	5	*****
2	Comedy	Bridesmaids	4	****
3	Drama	The Godfather	5	*****

write the query to find the highest and lowest salary employee in each department

```
--approach1
with cte as(
select dep_id, max(salary) as max_salary, min(salary) as min_salary
from employee1
group by dep_id)
select e.dep_id,
max(case when salary = max_salary then emp_name else null end )as max_sal_emp,
min(case when salary = min_salary then emp_name else null end )as min_sal_emp
from employee1 e
inner join cte On e.dep_id = cte.dep_id
group by e.dep_id;
```

	dep_id integer	max_sal_emp text	min_sal_emp text
1	2	raj	Sai
2	1	Sunny	Siva

```
--approach2
with cte as(
select *,
rank() over(partition by dep_id order by salary desc) as rank_desc,
rank() over(partition by dep_id order by salary asc) as rank_asc
from employee1)
select dep_id,
max(case when rank_desc = 1 then emp_name end) as max_sal_emp
,min(case when rank_asc = 1 then emp_name end )as min_sal_emp
from cte
group by dep_id
```

find the names which are not common in both source and target table.

```
--method1
Select coalesce(s.id,t.id) as id, -- s.name,t.name
case when t.name is null then 'new in source' when s.name is null then 'new in target'
else 'mismatched' end as comment
from source s
full outer join target t On s.id = t.id
where s.name != t.name or s.name is null or t.name is null
```

	id integer	comment text
1	3	new in source
2	4	mismatched
3	5	new in target

```
--method2
with cte as(
select *, 'source' as table_name from source
union all
select *, 'target' as table_name from target
)
select id,
case when min(name) != max(name) then 'mismatched'
when min(table_name)= 'source' then 'new in source'
when max(table_name) = 'target' then 'new in target' end as comment
from cte
group by id
having count(*) =1 or( count(*) = 2 and min(name) != max(name))
```

find the original and final destination for each flight

```
select o.cid, o.origin, o.destination as final_destination
from flights o
inner join flights d On o.destination = d.origin
```

	cid character varying (512)	origin character varying (512)	final_destination character varying (512)
1	1	Del	Hyd
2	2	Mum	Agra

find the count of new customer in each month

```
Select order_date, count(distinct customer) as count_new_cust from(
select *,
rank() over(partition by customer order by order_date) as rn
from sales) a
where rn = 1
group by order_date
```

	order_date date	count_new_cust bigint
1	2021-01-01	2
2	2021-02-01	1
3	2021-03-01	2
4	2021-04-01	1

find the father's name and mother's name of every child

```
-- approach 1
select r.c_id, p.name as child_name, max(m.name) as mother_name, max(f.name) as father_name
from relations r
left join people m on r.p_id = m.id and m.gender = 'F'
left join people f on r.p_id = f.id and f.gender = 'M'
inner join people p On p.id = r.c_id
group by r.c_id, p.name
```

	c_id integer	child_name character varying (20)	mother_name text	father_name text
1	145	Hawbaker	Days	Blackston
2	534	Waugh	Chatmon	Tong
3	618	Dimartino	Hansard	Beane
4	329	Mozingo	Criss	Nolf
5	278	Keffer	Hansel	Canty

```
--approach 2
select r.c_id as child_id, p.name as child_name,
 max(case when p.gender = 'F' then name end) as mother_name,
max(case when p.gender = 'M' then name end) as father_name
from relations r
inner join people p on r.p_id=p.id
group by 1, p.name
```

find the compnies which revenue is icreasing (no dip)

```
--approach1
with cte as(
select *,
 revenue - lag(revenue,1,0) over(partition by company order by year) as rev_diff,
 count(1) over(partition by company) as cnt
  from company_revenue
)
select company, cnt, count(1) as sales_inc_year
from cte
where rev_diff > 0
group by company, cnt
having cnt = count(1)
```

	company character varying (100)
1	ABC1

```
--appraoch2
with cte as(
select *,
 revenue - lag(revenue,1,0) over(partition by company order by year) as rev_diff
  from company_revenue
)
select company
from cte
where company not in (select company from cte where rev_diff < 0)
group by company
```

all the children will go on ride with adults only. adults can go alone if no more children is left for pairing. and eldest person will pair with youngest child.

```
with cte_adult as(
select *, row_number() over(order by age desc) as rn from family
 where type = 'Adult'
),
cte_child as(
select *, row_number() over(order by age asc) as rn from family
 where type = 'Child')
select a.person, c.person, a.age as adult_age, c.age as child_age
from cte_adult a
left join cte_child c On c.rn = a.rn
```

	person character varying (5)	person character varying (5)	adult_age integer	child_age integer
1	A4	C4	58	15
2	A1	C2	54	19
3	A5	C1	54	20
4	A2	C3	53	22
5	A3	[null]	52	[null]

determine the team which qualifies the below criteria
1.Both Criteria1 and Criteria2 are marked as 'Y' for a particular team.
2.The count of such occurrences of both 'Y' criteria within each team is at least 2.

```
--approach1
with qualified_team as(
select teamID, count(1) as no_of_eligible_members
from Ameriprise_LLC
where Criteria1 = 'Y' and Criteria2 = 'Y'
group by teamID
having count(1) >= 2)
select al.*,qt.*
, case when Criteria1 = 'Y' and Criteria2 = 'Y' and qt.no_of_eligible_members is not null then 'Y' else 'N' end as qualified_flag
from Ameriprise_LLC al
left join qualified_team qt On al.teamID = qt.teamID



--approach 2
select al.*,
 case when Criteria1 = 'Y' and Criteria2 = 'Y' and
sum(case when Criteria1 = 'Y' and Criteria2 = 'Y' then 1 else 0 end) over (partition by teamID) >=2 then 'Y' else 'N' end as qualified_flag
from Ameriprise_LLC al
```

determine the team which qualifies the below criteria

1.Both Criteria1 and Criteria2 are marked as 'Y' for a particular team.

2.The count of such occurrences of both 'Y' criteria within each team is at least 2.

	teamid character varying (2)	memberid character varying (10)	criteria1 character varying (1)	criteria2 character varying (1)	qualified_flag text
1	T1	T1_mbr1	Y	Y	Y
2	T1	T1_mbr2	Y	Y	Y
3	T1	T1_mbr3	Y	Y	Y
4	T1	T1_mbr4	Y	Y	Y
5	T1	T1_mbr5	Y	N	N
6	T2	T2_mbr1	Y	Y	N
7	T2	T2_mbr2	Y	N	N
8	T2	T2_mbr3	N	Y	N
9	T2	T2_mbr4	N	N	N
10	T2	T2_mbr5	N	N	N
11	T3	T3_mbr1	Y	Y	Y
12	T3	T3_mbr2	Y	Y	Y
13	T3	T3_mbr3	N	Y	N
14	T3	T3_mbr4	N	Y	N
15	T3	T3_mbr5	Y	N	N

Top 2 highest salried employees in each dept

```
select * from(
Select *,
row_number() over(partition by department_id order by salary desc) as rn,
dense_rank() over(partition by department_id order by salary desc) as dense_rn
from emp)a
--where rn <= 2
where dense_rn <= 2
```

	emp_id integer	emp_name character varying (20)	department_id integer	salary integer	manager_id integer	emp_age integer	rn bigint	dense_rn bigint
1	2	Mohit	100	15000	5	48	1	1
2	1	Ankit	100	10000	4	39	2	2
3	3	Vikas	100	10000	4	37	3	2
4	5	Mudit	200	12000	6	55	1	1
5	6	Agam	200	12000	2	14	2	1
6	7	Sanjay	200	9000	2	13	3	2
7	9	Mukesh	300	6000	6	51	1	1

Top 5 products by category and sales

```
with cte as (
select category, product_id, sum(sales) as sales
 from order_a
 group by  category, product_id
)
select * from(
select *,
row_number() over(partition by category order by sales desc) as rn
from cte)
where rn <= 5
```

	category character varying (50)	product_id character varying (100)	sales numeric	rn bigint
1	Furniture	FUR-CH-10002024	21870.57	1
2	Furniture	FUR-BO-10004834	15610.97	2
3	Furniture	FUR-TA-10003473	12995.28	3
4	Furniture	FUR-CH-10001215	12975.39	4
5	Furniture	FUR-BO-10002213	12921.64	5
6	Office Supplies	OFF-BI-10003527	27453.38	1
7	Office Supplies	OFF-BI-10001359	19823.48	2
8	Office Supplies	OFF-BI-10000545	19024.50	3
9	Office Supplies	OFF-BI-10004995	17965.07	4
10	Office Supplies	OFF-SU-10000151	17030.31	5
11	Technology	TEC-CO-10004722	61599.83	1
12	Technology	TEC-MA-10002412	22638.48	2
13	Technology	TEC-CO-10001449	18839.68	3
14	Technology	TEC-MA-10001127	18374.90	4
15	Technology	TEC-MA-10000822	16829.90	5

YOY grwoth/category wise products with sales more than pervious yesr sales(lead / lag function use case)

```
with cte as(
SELECT category, EXTRACT(YEAR FROM order_date) AS year_order, SUM(sales) AS sales
FROM order_a
GROUP BY category, EXTRACT(YEAR FROM order_date)
order BY category, EXTRACT(YEAR FROM order_date)
),
cte2 as(
select *,
 lag(sales,1,sales) over(partition by category order by year_order) as prev_year_sales
 from cte
)
select *,
Round((sales - prev_year_sales)*100/ prev_year_sales,0) as YOY_sales
from cte2
```

YOY grwoth/category wise products with sales more than pervious yesr sales(lead / lag function use case)

	category character varying (50)	year_order numeric	sales numeric	prev_year_sales numeric	yoy_sales numeric
1	Furniture	2018	157192.89	157192.89	0
2	Furniture	2019	170518.26	157192.89	8
3	Furniture	2020	198901.55	170518.26	17
4	Furniture	2021	215387.28	198901.55	8
5	Office Supplies	2018	151776.41	151776.41	0
6	Office Supplies	2019	137233.42	151776.41	-10
7	Office Supplies	2020	183940.07	137233.42	34
8	Office Supplies	2021	246097.09	183940.07	34
9	Technology	2018	175278.26	175278.26	0
10	Technology	2019	162780.78	175278.26	-7
11	Technology	2020	226364.24	162780.78	39
12	Technology	2021	271730.82	226364.24	20

runnuing/cummulative sales year wise/ rolling n months sales

```
with cte as(
SELECT category, EXTRACT(YEAR FROM order_date) AS year_order,
SUM(sales) AS sales
FROM order_a
GROUP BY category, EXTRACT(YEAR FROM order_date)
order by category, EXTRACT(YEAR FROM order_date)
)
select *,
sum(sales) over(partition by category order by year_order )
as cummulative_sales
from cte
```

	category character varying (50)	year_order numeric	sales numeric	cummulative_sales numeric
1	Furniture	2018	157192.89	157192.89
2	Furniture	2019	170518.26	327711.15
3	Furniture	2020	198901.55	526612.70
4	Furniture	2021	215387.28	741999.98
5	Office Supplies	2018	151776.41	151776.41
6	Office Supplies	2019	137233.42	289009.83
7	Office Supplies	2020	183940.07	472949.90
8	Office Supplies	2021	246097.09	719046.99
9	Technology	2018	175278.26	175278.26
10	Technology	2019	162780.78	338059.04
11	Technology	2020	226364.24	564423.28
12	Technology	2021	271730.82	836154.10

pivoting -> converting rows into column ->year wise sales for each category

```
Select Extract(Year from order_date) as year_order,
 Sum(case when category = 'Furniture' then sales else 0 end ) AS furniture_sales,
 Sum(case when category = 'Office Supplies' then sales else 0 end ) AS OS_sales,
 Sum(case when category = 'Technology' then sales else 0 end ) AS technology_sales
 Into order_a_b
FROM order_a
Group by Extract(year from order_date)
```

	year_order numeric	furniture_sales numeric	os_sales numeric	technology_sales numeric
1	2021	215387.28	246097.09	271730.82
2	2020	198901.55	183940.07	226364.24
3	2018	157192.89	151776.41	175278.26
4	2019	170518.26	137233.42	162780.78

find the category of games

1. No social interaction

2.one sided interaction

3.Both sided intraction without custome_typed messages

4.both side interaction with custome_typed message from at least one player

```
SELECT game_id,
    CASE
        WHEN count(interaction_type) = 0 THEN 'No social interaction'
        WHEN count(DISTINCT CASE WHEN interaction_type IS NOT NULL THEN user_id END) = 1
     THEN 'One sided interaction'
        WHEN count(DISTINCT CASE WHEN interaction_type IS NOT NULL THEN user_id END) = 2 AND
             count(DISTINCT CASE WHEN interaction_type = 'custom_typed' THEN user_id END) = 0
     THEN 'Both sided interaction without custom_typed messages'
        WHEN count(DISTINCT CASE WHEN interaction_type IS NOT NULL THEN user_id END) = 2 AND
             count(DISTINCT CASE WHEN interaction_type = 'custom_typed' THEN user_id END) >= 1
     THEN 'Both sided interaction with custom_typed message from at least one player'
    END AS game_type
FROM user_interactions
GROUP BY game_id;
```

find the category of games

1. No social interaction

2.one sided interaction

3.Both sided intraction without custome_typed messages

4.both side interaction with custome_typed message from at least one player

	game_id character varying (10)	game_type text
1	ab0000	Both sided interaction without custom_typed messages
2	ab1111	No social interaction
3	ab1234	Both sided interaction with custom_typed message from at least one pla...
4	ab9999	One sided interaction

Extract First name, middle name, last name

```
With cte as (
select *,
 length(customer_name) - length(replace(customer_name,' ','')) as no_of_space,
 Position(' 'IN customer_name) as first_space_position,
 Position(' ' IN Substring(customer_name From Position(' ' IN customer_name) + 1)) + Position(' ' IN customer_name)
AS second_space_position
 from customers
)
select *,
 case when no_of_space = 0 then customer_name
 else left(customer_name, first_space_position-1) end as First_name,
 case when no_of_space <= 1 then null
 else substring(customer_name, first_space_position +1, second_space_position - (first_space_position )) end as middle_name,
 case when no_of_space =0 then null
 when no_of_space = 1 then substring(customer_name, first_space_position +1, length(customer_name)- first_space_position)
 when no_of_space = 2 then substring(customer_name, second_space_position+1, length(customer_name)- second_space_position)
 end as last_name
from cte
```

Extract First name, middle name, last name

	customer_name character varying (30)	no_of_space integer	first_space_position integer	second_space_position integer	first_name text	middle_name text	last_name text
1	Manoj Vajpayee	1	6	6	Manoj	[null]	Vajpayee
2	Vishal Pratap Singh	2	7	14	Vishal	Pratap	Singh
3	Michael	0	0	0	Michael	[null]	[null]

Return the record which have 3 or more consecutive rows where no. of people is more than 100(inclusive)

```
with cte as (
select *, row_number() over(order by visit_date) as rn,
id -  row_number() over(order by visit_date) as grp
from stadium
where no_of_people >= 100)
select id,visit_date,no_of_people from cte
where grp IN(
select grp
from cte
group by grp
having count(1) >=3)
```

	id integer	visit_date date	no_of_people integer
1	5	2017-07-05	145
2	6	2017-07-06	1455
3	7	2017-07-07	199
4	8	2017-07-08	188

Find the employees whose salary is median of salary of all employees company wise

```
select company,avg(salary) from(
select *,
row_number() over(partition by company) as rn,
count(1) over(partition by company) as total_count
from median_salary) a
where rn between (total_count*1.0)/2 and (total_count*1.0 )/2 +1
group by company
```

	company character varying (10)	avg numeric
1	A	7664.5000000000000000
2	B	1345.0000000000000000
3	C	2645.0000000000000000

list of players according to city -> pivoting

```
select
max(case when city = 'Bangalore' then name end) as Bangalore,
max(case when city = 'Mumbai' then name end) as Mumbai,
max(case when city = 'Delhi' then name end) as Delhi
from
(select *,
row_number() over(partition by city) as players_location
from players_location) a
group by players_location
order by players_location
```

	bangalore text	mumbai text	delhi text
1	Rahul	Sachin	Virat
2	Mayank	Rohit	[null]

return the second most recent activity, if there is only one then return first

```
with cte as (
select *,
 count(1) over(partition by username) as total_activities,
 row_number() over(partition by username order by startDate) as rn
from UserActivity
) select * from cte
where total_activities = 1 or rn = 2
```

	username character varying (20)	activity character varying (20)	startdate date	enddate date	total_activities bigint	rn bigint
1	Alice	Dancing	2020-02-21	2020-02-23	3	2
2	Bob	Travel	2020-02-11	2020-02-18	1	1

Total sales by year

```
with recursive r_cte as(
select  min(period_start)::date as dates,
max(period_end)::date as max_date from recursive_sales
Union All
select (dates + INTERVAL '1 day')::date, max_date::date from r_cte
where dates < max_date)
SELECT product_id, Extract(year from dates) as report_year,
sum(average_daily_sales) as total_amount from r_cte
inner join recursive_sales On dates between period_start and period_end
group by product_id, report_year
order by product_id, report_year
```

	product_id integer	report_year numeric	total_amount bigint
1	1	2019	3500
2	2	2018	310
3	2	2019	3650
4	2	2020	10
5	3	2019	31
6	3	2020	31

Find the total number of users and total amount spent using mobile only, desktop only and mobile and desktop both for each date

```
with cte as (
select spend_date, user_id, max(platform) as platform, sum(amount) as amount
from spending
group by user_id, spend_date having count(distinct platform) =1
Union All
select spend_date, user_id, 'Both' as platform, sum(amount) as amount
from spending
group by user_id, spend_date having count(distinct platform) =2)
select spend_date, platform, sum(amount) as total_amount, count(distinct user_id) as total_users
from cte
group by spend_date, platform
order by spend_date, platform
```

	spend_date date	platform text	total_amount numeric	total_users bigint
1	2019-07-01	Both	200	1
2	2019-07-01	desktop	100	1
3	2019-07-01	mobile	100	1
4	2019-07-02	desktop	100	1
5	2019-07-02	mobile	100	1

Write a SQL query that reports the device that is first logged in for each player

```
select * from(
select *,
rank() over(partition by player_id order by event_date) as rn
from activity)a
where rn = 1
```

	player_id integer	device_id integer	event_date date	games_played integer	rn bigint
1	1	2	2016-03-01	5	1
2	2	3	2017-06-25	1	1
3	3	1	2016-03-02	0	1

Write an SQL query that reports for each player and date, how many games played so far by the player. That is, the total number of games played by the player until that date.

```
select *,
sum(games_played) over(partition by player_id order by event_date) as total_palyed
from activity
```

	player_id integer	device_id integer	event_date date	games_played integer	total_palyed bigint
1	1	2	2016-03-01	5	5
2	1	2	2016-03-02	6	11
3	2	3	2017-06-25	1	1
4	3	1	2016-03-02	0	0
5	3	4	2018-07-03	5	5

Write an SQL query that reports the fraction of playFind for each seller, whether the brand of the second item (by date) is his favourite brand. if a seller has sold leass than 2 item then report as No for him in the answerers that logged in again on the day after the day they first logged in, rounded to 2 decimal places

```
with rnk_orders as(
select *,
rank() over(partition by seller_id order by order_date asc) as rn
from orderss)
select u.user_id as seller_id,
case when i.item_brand = u.favorite_brand then 'Yes' else 'No' end as item_fav_brand
from users U
Left join rnk_orders ro on ro.seller_id = u.user_id and rn = 2
Left join items i on i.item_id = ro.item_id
```

	seller_id integer	item_fav_brand text
1	1	No
2	2	Yes
3	3	Yes
4	4	No

Find the winner in each group. the winner is one who scored maximum total points in that group .in case of a tie the lowest player_id wins

```
with player_scores as(
select first_player as player_id, first_score as score from matches
union all
select second_player as player_id, second_score as score from matches)
, final_scores as(
select p.group_id, ps.player_id, sum(score) as score
from player_scores ps
inner join players p on p.player_id = ps.player_id
group by ps.player_id, p.group_id
),
final_ranking as(
select *,
rank() over(partition by group_id order by score desc, player_id asc) as rn
from final_scores)
select * from final_ranking where rn = 1
```

	group_id integer	player_id integer	score bigint	rn bigint
1	1	15	3	1
2	2	35	1	1
3	3	40	5	1

THANK YOU!