```
------------------------Complex Query ----------------------------------------
------------------------   Q1  -----------------------------------------------
--trip and user leet code
--write a query to find cancellation rate request with unbanned users , both cle
-- each day betwen 2013/10/01- 2013/10/03


Create table  Trips (id int, client_id int, driver_id int, city_id int, status v
Create table Users (users_id int, banned varchar(50), role varchar(50));
Truncate table Trips;
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
insert into Trips (id, client_id, driver_id, city_id, status, request_at) values
Truncate table Users;
insert into Users (users_id, banned, role) values ('1', 'No', 'client');
insert into Users (users_id, banned, role) values ('2', 'Yes', 'client');
insert into Users (users_id, banned, role) values ('3', 'No', 'client');
insert into Users (users_id, banned, role) values ('4', 'No', 'client');
insert into Users (users_id, banned, role) values ('10', 'No', 'driver');
insert into Users (users_id, banned, role) values ('11', 'No', 'driver');
insert into Users (users_id, banned, role) values ('12', 'No', 'driver');
insert into Users (users_id, banned, role) values ('13', 'No', 'driver');

select * from trips
SELECT * from Users
```

```
Total execution time: 00:00:00.025
```

```
select * from trips
SELECT * from Users
```

(10 rows affected)
(8 rows affected)
Total execution time: 00:00:00.049

id	client_id	driver_id	city_id	status	request_at
1	1	10	1	completed	2013-10-01
2	2	11	1	cancelled_by_driver	2013-10-01
3	3	12	6	completed	2013-10-01
4	4	13	6	cancelled_by_client	2013-10-01
5	1	10	1	completed	2013-10-02
6	2	11	6	completed	2013-10-02
7	3	12	6	completed	2013-10-02
8	2	12	12	completed	2013-10-03
9	3	10	12	completed	2013-10-03
10	4	13	12	cancelled_by_driver	2013-10-03

users_id	banned	role
1	No	client
2	Yes	client
3	No	client
4	No	client
10	No	driver
11	No	driver
12	No	driver
13	No	driver

```
select request_at,count(case when status in ('Cancelled_by_client','cancelled_b
count(1) as total_trip,
1.0*count(case when status in ('Cancelled_by_client','cancelled_by_driver') the
from trips as t
inner join Users c on c.users_id = t.client_id
inner join Users d on d.users_id = t.client_id
where c.banned='No' and d.banned='No'
group by request_at
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(3 rows affected)
Total execution time: 00:00:00.044

request_at	cancelled_trip_count	total_trip	cancelled_percentage
2013-10-01	1	3	33.333333333300
2013-10-02	0	2	0.000000000000
2013-10-03	1	2	50.000000000000

----------q2----------

----------TOurnamnet winner ----------

--- LEET CODE HARD PROBLEM ----------

--- Write aquery to find winner in each group , winner in each group is the player who scored the maximum points within the group

----in cas eof tie lower player wins

```
create table players
(player_id int,
group_id int)

insert into players values (15,1);
insert into players values (25,1);
insert into players values (30,1);
insert into players values (45,1);
insert into players values (10,2);
insert into players values (35,2);
insert into players values (50,2);
insert into players values (20,3);
insert into players values (40,3);

create table matches
(
match_id int,
first_player int,
second_player int,
first_score int,
second_score int)

insert into matches values (1,15,45,3,0);
insert into matches values (2,30,25,1,2);
insert into matches values (3,30,15,2,0);
insert into matches values (4,40,20,5,2);
insert into matches values (5,35,50,1,1);
```

Total execution time: 00:00:00.013

```
select * from players
select * from matches
```

(9 rows affected)
(5 rows affected)
Total execution time: 00:00:00.012

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

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

```
select first_player as player_id, first_score as score from matches
```

(5 rows affected)
Total execution time: 00:00:00.005

player_id	score
15	3
30	1
30	2
40	5
35	1

```
select second_player as player_id, second_score as score from matches
```

(5 rows affected)
Total execution time: 00:00:00.009

player_id	score
45	0
25	2
15	0
20	2
50	1

```
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
```

(10 rows affected)
Total execution time: 00:00:00.017

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

```
with player_score 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_score as(
select p.group_id,ps.player_id,sum(score) as score
from player_score as ps
inner join players as p on p.player_id=ps.player_id
group by p. group_id ,ps.player_id)

,final_ranking as(
select *
,rank()over(partition by group_id order by score desc, player_id asc)as rn
from final_score)

select * from final_ranking
```

(8 rows affected)
Total execution time: 00:00:00.016

group_id	player_id	score	rn
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

```
with player_score 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_score as(
select p.group_id,ps.player_id,sum(score) as score
from player_score as ps
inner join players as p on p.player_id=ps.player_id
group by p. group_id ,ps.player_id)

,final_ranking as(
select *
,rank()over(partition by group_id order by score desc, player_id asc)as rn
from final_score)

select *
from final_ranking
where rn=1
```

(3 rows affected)
Total execution time: 00:00:00.009

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

----------q3----------

----------mARKET ANALYSIS----------

--FIND FOR EACH SELLAR THERE FAV BRAND OF SECOND ITEM IF SELLAR SOLD LESS THAN 2 ITEMS, REPORT THE ANSWER FOR THAT SELLAR AS NO

```
create table user1 (
user_id         int     ,
 join_date       date    ,
 favorite_brand  varchar(50));

 create table orders1 (
 order_id       int     ,
 order_date     date    ,
 item_id        int     ,
 buyer_id       int     ,
 seller_id      int
 );

 create table items1
 (
 item_id        int     ,
 item_brand     varchar(50)
 );


 insert into user1 values (1,'2019-01-01','Lenovo'),(2,'2019-02-09','Samsung'),

 insert into items1 values (1,'Samsung'),(2,'Lenovo'),(3,'LG'),(4,'HP');

 insert into orders1 values (1,'2019-08-01',4,1,2),(2,'2019-08-02',2,1,3),(3,'2
 ,(5,'2019-08-04',1,3,4),(6,'2019-08-05',2,2,4);
```

Total execution time: 00:00:00.019

```
select * from orders1
select * from items1
```

(6 rows affected)
(4 rows affected)
Total execution time: 00:00:00.007

order_id	order_date	item_id	buyer_id	seller_id
1	2019-08-01	4	1	2
2	2019-08-02	2	1	3
3	2019-08-03	3	2	3
4	2019-08-04	1	4	2
5	2019-08-04	1	3	4
6	2019-08-05	2	2	4

item_id	item_brand
1	Samsung
2	Lenovo
3	LG
4	HP

```
select *,
rank() over(partition by seller_id order by order_date asc) as rn
from orders1
```

(6 rows affected)
Total execution time: 00:00:00.013

order_id	order_date	item_id	buyer_id	seller_id	rn
1	2019-08-01	4	1	2	1
4	2019-08-04	1	4	2	2
2	2019-08-02	2	1	3	1
3	2019-08-03	3	2	3	2
5	2019-08-04	1	3	4	1
6	2019-08-05	2	2	4	2

```
with rank_orders as(
select *,
rank() over(partition by seller_id order by order_date asc) as rn
from orders1)

select u.user_id,
case when i.item_brand=u.favorite_brand then 'yes' else 'no' end as item_fav
from user1 as u
left join rank_orders as ro on ro.seller_id=u.user_id and rn=2
left join items1 as i on i.item_id=ro.item_id
```

(4 rows affected)
Total execution time: 00:00:00.038

user_id	item_fav
1	no
2	yes
3	yes
4	no

----------Q4----------

----------find start and end date and sucess in continious term

```
create table tasks (
date_value date,
state varchar(10)
);

insert into tasks  values ('2019-01-01','success'),('2019-01-02','success'),('2
,('2019-01-05','fail'),('2019-01-06','success')
```

(6 rows affected)
Total execution time: 00:00:00.028

```
select * from tasks
```

(6 rows affected)
Total execution time: 00:00:00.003

date_value	state
2019-01-01	success
2019-01-02	success
2019-01-03	success
2019-01-04	fail
2019-01-05	fail
2019-01-06	success

```
select *,
row_number()over(PARTITION by state order by date_value)
,DATEADD(day,-1*row_number()over(PARTITION by state order by date_value),date_v
from tasks
```

(6 rows affected)
Total execution time: 00:00:00.007

date_value	state	(No column name)	group_data
2019-01-04	fail	1	2019-01-03
2019-01-05	fail	2	2019-01-03
2019-01-01	success	1	2018-12-31
2019-01-02	success	2	2018-12-31
2019-01-03	success	3	2018-12-31
2019-01-06	success	4	2019-01-02

```
with cte as
(select *,
row_number()over(PARTITION by state order by date_value)
,DATEADD(day,-1*row_number()over(PARTITION by state order by date_value),date_v
from tasks)

select group_date,state,min(date_value) as start_date,max(date_value) as end_da
from cte
group by group_date,state
order by start_date
```

Total execution time: 00:00:00.004

----------Q5----------

---USER PURCHASE PLATFORM

--table logs the spending history of users that makes purchases from online shoping website which has a desktop and a mobile

-- application , write aquery to find total number of users and total amount spent using mobile only , desktop

```
create table spending
(
user_id int,
spend_date date,
platform varchar(10),
amount int
);

insert into spending values(1,'2019-07-01','mobile',100),(1,'2019-07-01','deskt
,(2,'2019-07-02','mobile',100),(3,'2019-07-01','desktop',100),(3,'2019-07-02','
```

Total execution time: 00:00:00.021

```
select * from spending
```

(6 rows affected)
Total execution time: 00:00:00.009

user_id	spend_date	platform	amount
1	2019-07-01	mobile	100
1	2019-07-01	desktop	100
2	2019-07-01	mobile	100
2	2019-07-02	mobile	100
3	2019-07-01	desktop	100
3	2019-07-02	desktop	100

```
select spend_date,user_id,max(platform) as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=1
```

(4 rows affected)
Total execution time: 00:00:00.041

spend_date	user_id	platform	amt
2019-07-01	2	mobile	100
2019-07-02	2	mobile	100
2019-07-01	3	desktop	100
2019-07-02	3	desktop	100

```
select spend_date,user_id,'both' as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=2
```

(1 row affected)
Total execution time: 00:00:00.015

spend_date	user_id	platform	amt
2019-07-01	1	both	200

```
select spend_date,null as user_id,'both' as platform,0 as amt
from spending
```

(6 rows affected)
Total execution time: 00:00:00.005

spend_date	user_id	platform	amt
2019-07-01	NULL	both	0
2019-07-01	NULL	both	0
2019-07-01	NULL	both	0
2019-07-02	NULL	both	0
2019-07-01	NULL	both	0
2019-07-02	NULL	both	0

```
select spend_date,user_id,max(platform) as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=1
union ALL
select spend_date,user_id,'both' as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=2
union ALL
select spend_date,null as user_id,'both' as platform,0 as amt
from spending
```

(11 rows affected)
Total execution time: 00:00:00.016

spend_date	user_id	platform	amt
2019-07-01	2	mobile	100
2019-07-02	2	mobile	100
2019-07-01	3	desktop	100
2019-07-02	3	desktop	100
2019-07-01	1	both	200
2019-07-01	NULL	both	0
2019-07-01	NULL	both	0
2019-07-01	NULL	both	0
2019-07-02	NULL	both	0
2019-07-01	NULL	both	0
2019-07-02	NULL	both	0

```
with all_spend as
(select spend_date,user_id,max(platform) as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=1
union ALL
select spend_date,user_id,'both' as platform,sum(amount) as amt
from spending
group by spend_date,user_id
having count(distinct platform )=2
union ALL
select spend_date,null as user_id,'both' as platform,0 as amt
from spending
)

select spend_date,platform,sum(amt) as total_sales,count(distinct user_id) as t
from all_spend
group by spend_date,platform
order by spend_date,platform desc
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(6 rows affected)
Total execution time: 00:00:00.026

spend_date	platform	total_sales	total_users
2019-07-01	mobile	100	1
2019-07-01	desktop	100	1
2019-07-01	both	200	1
2019-07-02	mobile	100	1
2019-07-02	desktop	100	1
2019-07-02	both	0	0

----------Q6----------

---total sales by year

```
create table sales (
product_id int,
period_start date,
period_end date,
average_daily_sales int
);

insert into sales values(1,'2019-01-25','2019-02-28',100),(2,'2018-12-01','2020
```

Total execution time: 00:00:00.008

```
select * from sales
```

(0 rows affected)
Total execution time: 00:00:00.002

product_id	period_start	period_end	average_daily_sales

```
with rcte as(
select min(period_start) as dates,max(period_end) as max_date from sales
union ALL
select dateadd(day,1,dates) as dates, max_date from rcte
where dates <max_date)

select product_id,year(dates) as report_year,
sum(average_daily_sales) as total_amount from rcte
inner join sales on dates between period_start and period_end
group by product_id,year(dates)
order by product_id,year(dates)
option(maxrecursion 1000)
```

(0 rows affected)
Total execution time: 00:00:00.016

product_id	report_year	total_amount

----------Q7----------

-----Recommendation based product pair

```
create table orders
(
order_id int,
customer_id int,
product_id int,
);

insert into orders VALUES
(1, 1, 1),
(1, 1, 2),
(1, 1, 3),
(2, 2, 1),
(2, 2, 2),
(2, 2, 4),
(3, 1, 5);

create table products (
id int,
name varchar(10)
);
insert into products VALUES
(1, 'A'),
(2, 'B'),
(3, 'C'),
(4, 'D'),
(5, 'E');
```

Total execution time: 00:00:00.008

```
select * from orders
select * from products
```

(7 rows affected)
(5 rows affected)
Total execution time: 00:00:00.012

order_id	customer_id	product_id
1	1	1
1	1	2
1	1	3
2	2	1
2	2	2
2	2	4
3	1	5

id	name
1	A
2	B
3	C
4	D
5	E

```
select o1.order_id,o1.product_id as p1, o2.product_id as p2
from orders o1
inner join orders o2 on o1.order_id=o2.order_id
where o1.order_id=1 and o1.product_id!=o2.product_id and o1.product_id>o2.produ
----here we get all the combinations for order id in p1,p2
```

(3 rows affected)
Total execution time: 00:00:00.020

order_id	p1	p2
1	2	1
1	3	1
1	3	2

```
--now just group by and get p1,p2
--now we have to find purchase frequency


select pr1.name as p1,pr2.name as p2,count(1) as freq
from orders o1
inner join orders o2 on o1.order_id=o2.order_id
inner join products pr1 on pr1.id = o1.product_id
inner join products pr2 on pr2.id = o2.product_id
where o1.product_id<o2.product_id
group by pr1.name,pr2.name
```

(5 rows affected)
Total execution time: 00:00:00.027

p1	p2	freq
A	B	2
A	C	1
B	C	1
A	D	1
B	D	1

```
select * from products

select pr1.name+ ' ' +pr2.name as pair_name
,count(1) as freq
from orders o1
inner join orders o2 on o1.order_id=o2.order_id
inner join products pr1 on pr1.id = o1.product_id
inner join products pr2 on pr2.id = o2.product_id
where o1.product_id<o2.product_id
group by pr1.name,pr2.name
```

(5 rows affected)
(5 rows affected)
Total execution time: 00:00:00.019

id	name
1	A
2	B
3	C
4	D
5	E

pair_name	freq
A B	2
A C	1
B C	1
A D	1
B D	1

----------Q8 ----------

--Amazon prime subscription rate logic in SQL. here is the problem statement:

--Given the following two tables, return the fraction of users, rounded to two decimal places,

--who accessed Amazon music and upgraded to prime membership within the first 30 days of signing up.

```
create table users1
(
user_id integer,
name varchar(20),
join_date date
);
insert into users1
values (1, 'Jon', CAST('2-14-20' AS date)),
(2, 'Jane', CAST('2-14-20' AS date)),
(3, 'Jill', CAST('2-15-20' AS date)),
(4, 'Josh', CAST('2-15-20' AS date)),
(5, 'Jean', CAST('2-16-20' AS date)),
(6, 'Justin', CAST('2-17-20' AS date)),
(7, 'Jeremy', CAST('2-18-20' AS date));

create table events
(
user_id integer,
type varchar(10),
access_date date
);

insert into events values
(1, 'Pay', CAST('3-1-20' AS date)),
(2, 'Music', CAST('3-2-20' AS date)),
(2, 'P', CAST('3-12-20' AS date)),
(3, 'Music', CAST('3-15-20' AS date)),
(4, 'Music', CAST('3-15-20' AS date)),
(1, 'P', CAST('3-16-20' AS date)),
(3, 'P', CAST('3-22-20' AS date));
```

```
select * from users1
select * from events
```

(7 rows affected)
(7 rows affected)
Total execution time: 00:00:00.012

user_id	name	join_date
1	Jon	2020-02-14
2	Jane	2020-02-14
3	Jill	2020-02-15
4	Josh	2020-02-15
5	Jean	2020-02-16
6	Justin	2020-02-17
7	Jeremy	2020-02-18

user_id	type	access_date
1	Pay	2020-03-01
2	Music	2020-03-02
2	P	2020-03-12
3	Music	2020-03-15
4	Music	2020-03-15
1	P	2020-03-16
3	P	2020-03-22

```
select *
from users1 as u1
left join events e on u1.user_id = e.user_id and e.type = 'P'
where u1.user_id in (select user_id from events where type = 'Music')
```

(3 rows affected)
Total execution time: 00:00:00.033

user_id	name	join_date	user_id	type	access_date
2	Jane	2020-02-14	2	P	2020-03-12
3	Jill	2020-02-15	3	P	2020-03-22
4	Josh	2020-02-15	NULL	NULL	NULL

```
select u1.*,e.type,e.access_date,DATEDIFF(day,u1.join_date,e.access_date) as no
from users1 as u1
left join events e on u1.user_id = e.user_id and e.type = 'P'
where u1.user_id in (select user_id from events where type = 'Music')
```

(3 rows affected)
Total execution time: 00:00:00.017

user_id	name	join_date	type	access_date	no_of_days
2	Jane	2020-02-14	P	2020-03-12	27
3	Jill	2020-02-15	P	2020-03-22	36
4	Josh	2020-02-15	NULL	NULL	NULL

```
select u1.*,e.type,e.access_date,DATEDIFF(day,u1.join_date,e.access_date) as no
from users1 as u1
left join events e on u1.user_id = e.user_id and e.type = 'P'
where u1.user_id in (select user_id from events where type = 'Music')
```

(3 rows affected)
Total execution time: 00:00:00.014

user_id	name	join_date	type	access_date	no_of_days
2	Jane	2020-02-14	P	2020-03-12	27
3	Jill	2020-02-15	P	2020-03-22	36
4	Josh	2020-02-15	NULL	NULL	NULL

```
select count(1) as no_of_users, count(distinct case when DATEDIFF(day,u1.join_c
from users1 as u1
left join events e on u1.user_id = e.user_id and e.type = 'P'
where u1.user_id in (select user_id from events where type = 'Music')
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(1 row affected)
Total execution time: 00:00:00.026

no_of_users	no_of_prime_members
3	1

```
select count(1) as no_of_users, count(distinct case when DATEDIFF(day,u1.join_c
,1.0*count(distinct case when DATEDIFF(day,u1.join_date,e.access_date) <=30 the
from users1 as u1
left join events e on u1.user_id = e.user_id and e.type = 'P'
where u1.user_id in (select user_id from events where type = 'Music')
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(1 row affected)
Total execution time: 00:00:00.024

no_of_users	no_of_prime_members	percentage_of_ofmembership
3	1	33.333333333300

----------Q9 -
==========
==========

-----customer retention and sustomer churn

/* ustomer retention refers to the ability of a company or product to retain its customers over some specified period.

High customer retention means customers of the product or business tend to return to, continue to buy or in

some other way not defect to another product or business, or to non-use entirely.

Company programs to retain customers: Zomato Pro , Cashbacks, Reward Programs etc.

Once these programs in place we need to build metrics to check if programs are working or not.

That is where we will write SQL to drive customer retention count. */

```
create table transactions(
order_id int,
cust_id int,
order_date date,
amount int
);

insert into transactions values
(1,1,'2020-01-15',150)
,(2,1,'2020-02-10',150)
,(3,2,'2020-01-16',150)
,(4,2,'2020-02-25',150)
,(5,3,'2020-01-10',150)
,(6,3,'2020-02-20',150)
,(7,4,'2020-01-20',150)
,(8,5,'2020-02-20',150)
;
```

Total execution time: 00:00:00.022

```
select * from transactions
select * from transactions
```

(8 rows affected)
(8 rows affected)
Total execution time: 00:00:00.015

order_id	cust_id	order_date	amount
1	1	2020-01-15	150
2	1	2020-02-10	150
3	2	2020-01-16	150
4	2	2020-02-25	150
5	3	2020-01-10	150
6	3	2020-02-20	150
7	4	2020-01-20	150
8	5	2020-02-20	150

order_id	cust_id	order_date	amount
1	1	2020-01-15	150
2	1	2020-02-10	150
3	2	2020-01-16	150
4	2	2020-02-25	150
5	3	2020-01-10	150
6	3	2020-02-20	150
7	4	2020-01-20	150
8	5	2020-02-20	150

```
select *
from transactions as this_month
left join transactions as last_month on this_month.cust_id = last_month.cust_ic
DATEDIFF(month,this_month.order_date,last_month.order_date)=1
```

(8 rows affected)
Total execution time: 00:00:00.026

order_id	cust_id	order_date	amount	order_id	cust_id	order_date	amount
1	1	2020-01-15	150	2	1	2020-02-10	150
2	1	2020-02-10	150	NULL	NULL	NULL	NULL
3	2	2020-01-16	150	4	2	2020-02-25	150
4	2	2020-02-25	150	NULL	NULL	NULL	NULL
5	3	2020-01-10	150	6	3	2020-02-20	150
6	3	2020-02-20	150	NULL	NULL	NULL	NULL
7	4	2020-01-20	150	NULL	NULL	NULL	NULL
8	5	2020-02-20	150	NULL	NULL	NULL	NULL

```
select month(this_month.order_date) as month_date , count (distinct last_month.
from transactions as this_month
left join transactions as last_month on this_month.cust_id = last_month.cust_ic
DATEDIFF(month,last_month.order_date,this_month.order_date)=1
group by month(this_month.order_date)
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(2 rows affected)
Total execution time: 00:00:00.012

month_date	count_of_customer
1	0
2	3

----------Q 10 ----------

----------above is customer retention ----------

----------now for custome rchur ----------

--for retained custiome rthere is no churn if there is change in customer count then there is churn

--customer who ordered last month but not this month

```
select last_month.*,this_month.*
--month(this_month.order_date) as month_date , count (distinct last_month.cust_
from transactions as last_month
left join transactions as this_month on this_month.cust_id = last_month.cust_ic
DATEDIFF(month,last_month.order_date,this_month.order_date)=1
where month(last_month.order_date)=1
--group by month(this_month.order_date)
```

(4 rows affected)
Total execution time: 00:00:00.011

order_id	cust_id	order_date	amount	order_id	cust_id	order_date	amount
1	1	2020-01-15	150	2	1	2020-02-10	150
3	2	2020-01-16	150	4	2	2020-02-25	150
5	3	2020-01-10	150	6	3	2020-02-20	150
7	4	2020-01-20	150	NULL	NULL	NULL	NULL

```
--group by month(this_month.order_date)

select month(last_month.order_date) as month_date ,
count (distinct last_month.cust_id) as count_of_customer
from transactions as last_month
left join transactions as this_month on this_month.cust_id = last_month.cust_ic
DATEDIFF(month,last_month.order_date,this_month.order_date)=1
where this_month.cust_id is NULL
group by month(last_month.order_date)
```

(2 rows affected)
Total execution time: 00:00:00.014

month_date	count_of_customer
1	1
2	4

----------Q11----------

/* Where we need to find second most recent activity and if user has only 1 activoty then return that as it is.

We will use SQL window functions to solve this problem. */

```
create table UserActivity
(
username      varchar(20) ,
activity      varchar(20),
startDate     Date   ,
endDate      Date
);

insert into UserActivity values
('Alice','Travel','2020-02-12','2020-02-20')
,('Alice','Dancing','2020-02-21','2020-02-23')
,('Alice','Travel','2020-02-24','2020-02-28')
,('Bob','Travel','2020-02-11','2020-02-18');
```

Total execution time: 00:00:00.008

```
select *,
count(1) over(partition by username) as total,
rank() over(partition by username order by startDate desc) as rnk
from UserActivity
```

(4 rows affected)
Total execution time: 00:00:00.017

username	activity	startDate	endDate	total	rnk
Alice	Travel	2020-02-24	2020-02-28	3	1
Alice	Dancing	2020-02-21	2020-02-23	3	2
Alice	Travel	2020-02-12	2020-02-20	3	3
Bob	Travel	2020-02-11	2020-02-18	1	1

```
with cte_1 as(
select *,
count(1) over(partition by username) as total,
rank() over(partition by username order by startDate desc) as rnk
from UserActivity
)
select *
from cte_1
 where total=1 or rnk=2
```

(2 rows affected)
Total execution time: 00:00:00.008

username	activity	startDate	endDate	total	rnk
Alice	Dancing	2020-02-21	2020-02-23	3	2
Bob	Travel	2020-02-11	2020-02-18	1	1

----------Q12----------

/*I will be solving it using Analytical function. You will learn how to use Lead analytical function with partition by clause and how to deal with data ranges in SQL.

total charges as per billing rate */

```
create table billings
(
emp_name varchar(10),
bill_date date,
bill_rate int
);
delete from billings;
insert into billings values
('Sachin','01-JAN-1990',25)
,('Sehwag' ,'01-JAN-1989', 15)
,('Dhoni' ,'01-JAN-1989', 20)
,('Sachin' ,'05-Feb-1991', 30)
;

create table HoursWorked
(
emp_name varchar(20),
work_date date,
bill_hrs int
);
insert into HoursWorked values
('Sachin', '01-JUL-1990' ,3)
,('Sachin', '01-AUG-1990', 5)
,('Sehwag','01-JUL-1990', 2)
,('Sachin','01-JUL-1991', 4)
```

Total execution time: 00:00:00.010

```
select *from billings
select * from HoursWorked
```

(4 rows affected)
(4 rows affected)
Total execution time: 00:00:00.009

emp_name	bill_date	bill_rate
Sachin	1990-01-01	25
Sehwag	1989-01-01	15
Dhoni	1989-01-01	20
Sachin	1991-02-05	30

emp_name	work_date	bill_hrs
Sachin	1990-07-01	3
Sachin	1990-08-01	5
Sehwag	1990-07-01	2
Sachin	1991-07-01	4

```
select *
,lead(bill_date,1)over(partition by emp_name order by bill_date asc) as bill_er
from billings
```

(4 rows affected)
Total execution time: 00:00:00.018

emp_name	bill_date	bill_rate	bill_end
Dhoni	1989-01-01	20	NULL
Sachin	1990-01-01	25	1991-02-05
Sachin	1991-02-05	30	NULL
Sehwag	1989-01-01	15	NULL

```
select *
,lead(DATEADD(day,-1,bill_date),1,'9999-12-12')over(partition by emp_name order
from billings
```

(4 rows affected)
Total execution time: 00:00:00.008

emp_name	bill_date	bill_rate	bill_end
Dhoni	1989-01-01	20	9999-12-12
Sachin	1990-01-01	25	1991-02-04
Sachin	1991-02-05	30	9999-12-12
Sehwag	1989-01-01	15	9999-12-12

```
with daterange as(
select *
,lead(DATEADD(day,-1,bill_date),1,'9999-12-12')over(partition by emp_name order
from billings)

select hw.*,ct.bill_rate
from daterange as ct
inner join HoursWorked hw on ct.emp_name=hw.emp_name and
hw.work_date BETWEEN ct.bill_date and ct.bill_end
```

(4 rows affected)
Total execution time: 00:00:00.018

emp_name	work_date	bill_hrs	bill_rate
Sachin	1990-07-01	3	25
Sachin	1990-08-01	5	25
Sachin	1991-07-01	4	30
Sehwag	1990-07-01	2	15

```
with daterange as(
select *
,lead(DATEADD(day,-1,bill_date),1,'9999-12-12')over(partition by emp_name order
from billings)

select hw.emp_name,sum(ct.bill_rate*hw.bill_hrs) as final
from daterange as ct
inner join HoursWorked hw on ct.emp_name=hw.emp_name and
hw.work_date BETWEEN ct.bill_date and ct.bill_end
group by hw.emp_name
```

(2 rows affected)
Total execution time: 00:00:00.013

emp_name	final
Sachin	320
Sehwag	30

----------Q13----------

/Spotify case study. This case study has 5 questions and with each question difficulty level will go up/

```
CREATE table activity
(
user_id varchar(20),
event_name varchar(20),
event_date date,
country varchar(20)
);
delete from activity;
insert into activity values (1,'app-installed','2022-01-01','India')
,(1,'app-purchase','2022-01-02','India')
,(2,'app-installed','2022-01-01','USA')
,(3,'app-installed','2022-01-01','USA')
,(3,'app-purchase','2022-01-03','USA')
,(4,'app-installed','2022-01-03','India')
,(4,'app-purchase','2022-01-03','India')
,(5,'app-installed','2022-01-03','SL')
,(5,'app-purchase','2022-01-03','SL')
,(6,'app-installed','2022-01-04','Pakistan')
,(6,'app-purchase','2022-01-04','Pakistan');
```

Total execution time: 00:00:00.010

```
SELECT * FROM activity
```

(11 rows affected)
Total execution time: 00:00:00.003

user_id	event_name	event_date	country
1	app-installed	2022-01-01	India
1	app-purchase	2022-01-02	India
2	app-installed	2022-01-01	USA
3	app-installed	2022-01-01	USA
3	app-purchase	2022-01-03	USA
4	app-installed	2022-01-03	India
4	app-purchase	2022-01-03	India
5	app-installed	2022-01-03	SL
5	app-purchase	2022-01-03	SL
6	app-installed	2022-01-04	Pakistan
6	app-purchase	2022-01-04	Pakistan

```
-----1------
--find the total active users each day .o/p - event date , total active users
--activity table shows app installed and purchages activity for spotify along w

select * from activity

select event_date, count(distinct user_id)
from activity
group by event_date
```

(11 rows affected)
(4 rows affected)
Total execution time: 00:00:00.025

user_id	event_name	event_date	country
1	app-installed	2022-01-01	India
1	app-purchase	2022-01-02	India
2	app-installed	2022-01-01	USA
3	app-installed	2022-01-01	USA
3	app-purchase	2022-01-03	USA
4	app-installed	2022-01-03	India
4	app-purchase	2022-01-03	India
5	app-installed	2022-01-03	SL
5	app-purchase	2022-01-03	SL
6	app-installed	2022-01-04	Pakistan
6	app-purchase	2022-01-04	Pakistan

event_date	(No column name)
2022-01-01	3
2022-01-02	1
2022-01-03	3
2022-01-04	1

```
---2
-- find active users each week
--o/p - week no , total active uers

select *, DATEPART(week, event_date) as weeknumber
 from activity


select DATEPART(week, event_date) as weeknumber, count(distinct user_id) as act
from activity
group by  DATEPART(week, event_date)
```

(11 rows affected)
(2 rows affected)
Total execution time: 00:00:00.013

user_id	event_name	event_date	country	weeknumber
1	app-installed	2022-01-01	India	1
1	app-purchase	2022-01-02	India	2
2	app-installed	2022-01-01	USA	1
3	app-installed	2022-01-01	USA	1
3	app-purchase	2022-01-03	USA	2
4	app-installed	2022-01-03	India	2
4	app-purchase	2022-01-03	India	2
5	app-installed	2022-01-03	SL	2
5	app-purchase	2022-01-03	SL	2
6	app-installed	2022-01-04	Pakistan	2
6	app-purchase	2022-01-04	Pakistan	2

weeknumber	active_users
1	3
2	5

```
---3
--date wise total number of users who made the purchase same day they installec
--o/p - event date , no.of users same day purchage

--no of events on event date

select user_id, event_date,count(distinct event_name) as noofevents
from activity
group by user_id, event_date
```

(8 rows affected)
Total execution time: 00:00:00.010

user_id	event_date	noofevents
1	2022-01-01	1
2	2022-01-01	1
3	2022-01-01	1
1	2022-01-02	1
3	2022-01-03	1
4	2022-01-03	2
5	2022-01-03	2
6	2022-01-04	2

```
select event_date, count(user_id) as no_of_users from (
select user_id, event_date,count(distinct event_name) as noofevents
from activity
group by user_id, event_date
having count(distinct event_name)=2) a
group by event_date
```

(2 rows affected)
Total execution time: 00:00:00.009

event_date	no_of_users
2022-01-03	2
2022-01-04	1

```
---- here also we are not geteting all the results becaise of having clause
--
select event_date, count(new_user) as no_of_users from (
select user_id, event_date,
case when count(distinct event_name)=2 then user_id else null end AS new_user
from activity
group by user_id, event_date
--having count(distinct event_name)=2
)
a
group by event_date
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(4 rows affected)
Total execution time: 00:00:00.013

event_date	no_of_users
2022-01-01	0
2022-01-02	0
2022-01-03	2
2022-01-04	1

```
---4--
/* % of paid user in india, usa and any other country will be tagges as other
o/p : country , % user
*/

select *
from activity
where event_name='app-purchase'
```

(5 rows affected)
Total execution time: 00:00:00.005

user_id	event_name	event_date	country
1	app-purchase	2022-01-02	India
3	app-purchase	2022-01-03	USA
4	app-purchase	2022-01-03	India
5	app-purchase	2022-01-03	SL
6	app-purchase	2022-01-04	Pakistan

```
select country, count(distinct user_id)
from activity
where event_name='app-purchase'
group by country
```

(4 rows affected)
Total execution time: 00:00:00.014

country	(No column name)
India	2
Pakistan	1
SL	1
USA	1

```
select count(distinct user_id),
case when country in ('USA','India') then country else 'other' end as new_count
from activity
where event_name='app-purchase'
group by case when country in ('USA','India') then country else 'other' end
```

(3 rows affected)
Total execution time: 00:00:00.010

(No column name)	new_country
2	India
2	other
1	USA

```
with country_user as(
    select count(distinct user_id)as user_cnt,
case when country in ('USA','India') then country else 'other' end as new_count
from activity
where event_name='app-purchase'
group by case when country in ('USA','India') then country else 'other' end
)
,total as ( select sum(user_cnt) as total_user from country_user)
```

Total execution time: 00:00:00.004

```
select *,1.0*user_cnt/total_user *100 as percent_user
from country_user,total;
```

Total execution time: 00:00:00.005

```
with country_user as(
    select count(distinct user_id)as user_cnt,
case when country in ('USA','India') then country else 'other' end as new_count
from activity
where event_name='app-purchase'
group by case when country in ('USA','India') then country else 'other' end
)
,total as ( select sum(user_cnt) as total_user from country_user)
select *,1.0*user_cnt/total_user *100 as percent_user
from country_user,total;
```

(3 rows affected)
Total execution time: 00:00:00.017

user_cnt	new_country	total_user	percent_user
2	India	5	40.000000000000
2	other	5	40.000000000000
1	USA	5	20.000000000000

```
--5
--amoung all the users who installed the app on given day how many diid app pur
--daywise , op: event date , count user

select *
,lag(event_name,1) over(partition by user_id order by event_date) as lag_event_
,lag(event_date,1) over(partition by user_id order by event_date) as lag_event_
from activity
```

(11 rows affected)
Total execution time: 00:00:00.010

user_id	event_name	event_date	country	lag_event_name	lag_event_date
1	app-installed	2022-01-01	India	NULL	NULL
1	app-purchase	2022-01-02	India	app-installed	2022-01-01
2	app-installed	2022-01-01	USA	NULL	NULL
3	app-installed	2022-01-01	USA	NULL	NULL
3	app-purchase	2022-01-03	USA	app-installed	2022-01-01
4	app-installed	2022-01-03	India	NULL	NULL
4	app-purchase	2022-01-03	India	app-installed	2022-01-03
5	app-installed	2022-01-03	SL	NULL	NULL
5	app-purchase	2022-01-03	SL	app-installed	2022-01-03
6	app-installed	2022-01-04	Pakistan	NULL	NULL
6	app-purchase	2022-01-04	Pakistan	app-installed	2022-01-04

```
with prev_data as(
    select *
,lag(event_name,1) over(partition by user_id order by event_date) as lag_event_
,lag(event_date,1) over(partition by user_id order by event_date) as lag_event_
from activity
)
,final as(
select *
from prev_data
where event_name='app-purchase' and lag_event_name='app-installed' and
DATEDIFF(day,lag_event_date,event_date)=1)
```

Total execution time: 00:00:00.003

```
select event_date, count(user_id)
from final
group by event_date
```

Total execution time: 00:00:00.004

```
with prev_data as(
    select *
,lag(event_name,1) over(partition by user_id order by event_date) as lag_event_
,lag(event_date,1) over(partition by user_id order by event_date) as lag_event_
from activity
)
,final as(
select *
from prev_data
where event_name='app-purchase' and lag_event_name='app-installed' and
DATEDIFF(day,lag_event_date,event_date)=1)
select event_date, count(user_id)
from final
group by event_date
```

(1 row affected)
Total execution time: 00:00:00.011

event_date	(No column name)
2022-01-02	1

Q14

--3 or more consecutive empty seats

/* 1) lead lag

2) advanced aggregation

3) analytical row umber function */

```
  create table bms (seat_no int ,is_empty varchar(10));
insert into bms values
(1,'N')
,(2,'Y')
,(3,'N')
,(4,'Y')
,(5,'Y')
,(6,'Y')
,(7,'N')
,(8,'Y')
,(9,'Y')
,(10,'Y')
,(11,'Y')
,(12,'N')
,(13,'Y')
,(14,'Y');
```

Total execution time: 00:00:00.015

```
select * from bms
```

(14 rows affected)
Total execution time: 00:00:00.005

seat_no	is_empty
1	N
2	Y
3	N
4	Y
5	Y
6	Y
7	N
8	Y
9	Y
10	Y
11	Y
12	N
13	Y
14	Y

```
select *,
lag(is_empty,1) over(order by seat_no) as prev_1
,lag(is_empty,2) over(order by seat_no) as prev_2
,lead(is_empty,1) over(order by seat_no) as next_1
,lead(is_empty,2) over(order by seat_no) as next_2
from bms
```

(14 rows affected)
Total execution time: 00:00:00.012

seat_no	is_empty	prev_1	prev_2	next_1	next_2
1	N	NULL	NULL	Y	N
2	Y	N	NULL	N	Y
3	N	Y	N	Y	Y
4	Y	N	Y	Y	Y
5	Y	Y	N	Y	N
6	Y	Y	Y	N	Y
7	N	Y	Y	Y	Y
8	Y	N	Y	Y	Y
9	Y	Y	N	Y	Y
10	Y	Y	Y	Y	N
11	Y	Y	Y	N	Y
12	N	Y	Y	Y	Y
13	Y	N	Y	Y	NULL
14	Y	Y	N	NULL	NULL

```
select *
from (
    select *,
lag(is_empty,1) over(order by seat_no) as prev_1
,lag(is_empty,2) over(order by seat_no) as prev_2
,lead(is_empty,1) over(order by seat_no) as next_1
,lead(is_empty,2) over(order by seat_no) as next_2
from bms
) A
where is_empty='Y' and prev_1='Y' and prev_2 = 'Y'
or (is_empty='Y' and prev_1='Y' and next_1 = 'Y')
or (is_empty='Y' and next_1='Y' and next_2 = 'Y')
```

(7 rows affected)
Total execution time: 00:00:00.021

seat_no	is_empty	prev_1	prev_2	next_1	next_2
4	Y	N	Y	Y	Y
5	Y	Y	N	Y	N
6	Y	Y	Y	N	Y
8	Y	N	Y	Y	Y
9	Y	Y	N	Y	Y
10	Y	Y	Y	Y	N
11	Y	Y	Y	N	Y

```
-----method 2

select *
,sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwe
,sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwe
,sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwe
from bms
```

(14 rows affected)
Total execution time: 00:00:00.051

seat_no	is_empty	prev_2	prev_next_1	next_2
1	N	0	1	1
2	Y	1	1	2
3	N	1	2	2
4	Y	2	2	3
5	Y	2	3	2
6	Y	3	2	2
7	N	2	2	2
8	Y	2	2	3
9	Y	2	3	3
10	Y	3	3	2
11	Y	3	2	2
12	N	2	2	2
13	Y	2	2	2
14	Y	2	2	1

```
select *
from (
  select *,
sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwee
,sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwe
,sum(case when is_empty='Y' then 1 else 0 end) over(order by seat_no rows betwe
from bms
) A
where prev_2=3 or  prev_next_1=3 or next_2=3
```

(7 rows affected)
Total execution time: 00:00:00.012

seat_no	is_empty	prev_2	prev_next_1	next_2
4	Y	2	2	3
5	Y	2	3	2
6	Y	3	2	2
8	Y	2	2	3
9	Y	2	3	3
10	Y	3	3	2
11	Y	3	2	2

```
-----method 3
select * from bms
where is_empty='Y'
```

(10 rows affected)
Total execution time: 00:00:00.005

seat_no	is_empty
2	Y
4	Y
5	Y
6	Y
8	Y
9	Y
10	Y
11	Y
13	Y
14	Y

```
select *,
ROW_NUMBER() over(order by seat_no) as rn ,
seat_no-row_number() over(order by seat_no) as diff
from bms
where is_empty='Y'
```

(10 rows affected)
Total execution time: 00:00:00.006

seat_no	is_empty	rn	diff
2	Y	1	1
4	Y	2	2
5	Y	3	2
6	Y	4	2
8	Y	5	3
9	Y	6	3
10	Y	7	3
11	Y	8	3
13	Y	9	4
14	Y	10	4

```
with diff_num as(
   select *,
ROW_NUMBER() over(order by seat_no) as rn ,
seat_no-row_number() over(order by seat_no) as diff
from bms
where is_empty='Y'
)
,cnt as(
select diff , count(1) as c
from diff_num
group by diff
having count(1)>3)
```

Total execution time: 00:00:00.003

```
select * from diff_num where diff in (select diff from cnt)
```

Total execution time: 00:00:00.003

```
with diff_num as(
   select *,
ROW_NUMBER() over(order by seat_no) as rn ,
seat_no-row_number() over(order by seat_no) as diff
from bms
where is_empty='Y'
)
,cnt as(
select diff , count(1) as c
from diff_num
group by diff
having count(1)>3)
select * from diff_num where diff in (select diff from cnt)
```

(4 rows affected)
Total execution time: 00:00:00.021

seat_no	is_empty	rn	diff
8	Y	5	3
9	Y	6	3
10	Y	7	3
11	Y	8	3

q15

-- STORE TABLE , EACH STORE IS CLOSED FOR 1 QUARTER FOR MAINTENACE , WE HAVE TO FIND MISSING QUARTER FOR EACH STORE

--DDL and DML:

```
CREATE TABLE STORES (
Store varchar(10),
Quarter varchar(10),
Amount int);

INSERT INTO STORES (Store, Quarter, Amount)
VALUES ('S1', 'Q1', 200),
('S1', 'Q2', 300),
('S1', 'Q4', 400),
('S2', 'Q1', 500),
('S2', 'Q3', 600),
('S2', 'Q4', 700),
('S3', 'Q1', 800),
('S3', 'Q2', 750),
('S3', 'Q3', 900);
```

Total execution time: 00:00:00.015

```
Select * from STORES
```

(9 rows affected)
Total execution time: 00:00:00.007

Store	Quarter	Amount
S1	Q1	200
S1	Q2	300
S1	Q4	400
S2	Q1	500
S2	Q3	600
S2	Q4	700
S3	Q1	800
S3	Q2	750
S3	Q3	900

```
-----Method 1
-- q(1+2+3+4)=10 , 10- any value will give quarter

select store,sum(cast(right(quarter,1) as int)) as q_no_from_store
from stores
group by Store
```

(3 rows affected)
Total execution time: 00:00:00.009

store	q_no_from_store
S1	7
S2	8
S3	6

```
select store,10-sum(cast(right(quarter,1) as int)) as q_no_from_store
from stores
group by Store
```

(3 rows affected)
Total execution time: 00:00:00.011

store	q_no_from_store
S1	3
S2	2
S3	4

```
select store, 'Q'+ cast(10-sum(cast(right(quarter,1) as int)) as char(2)) as qu
from stores
group by Store
```

(3 rows affected)
Total execution time: 00:00:00.008

store	quarter_missing
S1	Q3
S2	Q2
S3	Q4

```
-----Method 2--------------------
--recursive cte
with cte as
(select distinct Store ,1 as q_no from Stores
union ALL
select store ,q_no+1 as q_no from cte
where q_no<4)

,with q as(
    select store,'Q' +cast(q_no as char(1)) as q_no from cte )
select *
from STORES
left join stores s on q.store = s.store
and q.q_no = s.Store
```

Total execution time: 00:00:00.004

q16

---find student with same mars in physics and chemistry

```
    create table exams (student_id int, subject varchar(20), marks int);
delete from exams;
insert into exams values (1,'Chemistry',91),(1,'Physics',91)
,(2,'Chemistry',80),(2,'Physics',90)
,(3,'Chemistry',80)
,(4,'Chemistry',71),(4,'Physics',54);
```

Total execution time: 00:00:00.019

```
select * from exams
```

(7 rows affected)
Total execution time: 00:00:00.007

student_id	subject	marks
1	Chemistry	91
1	Physics	91
2	Chemistry	80
2	Physics	90
3	Chemistry	80
4	Chemistry	71
4	Physics	54

```
select student_id
from exams
where subject in ('Chemistry','Physics')
group by student_id
having count(distinct subject)=2 and count(distinct marks)=1
```

(1 row affected)
Total execution time: 00:00:00.029

student_id
1

q17

--find city where covid cases are inscreading continuously

```
create table covid(city varchar(50),days date,cases int);
delete from covid;
insert into covid values('DELHI','2022-01-01',100);
insert into covid values('DELHI','2022-01-02',200);
insert into covid values('DELHI','2022-01-03',300);

insert into covid values('MUMBAI','2022-01-01',100);
insert into covid values('MUMBAI','2022-01-02',100);
insert into covid values('MUMBAI','2022-01-03',300);

insert into covid values('CHENNAI','2022-01-01',100);
insert into covid values('CHENNAI','2022-01-02',200);
insert into covid values('CHENNAI','2022-01-03',150);

insert into covid values('BANGALORE','2022-01-01',100);
insert into covid values('BANGALORE','2022-01-02',300);
insert into covid values('BANGALORE','2022-01-03',200);
insert into covid values('BANGALORE','2022-01-04',400);
```

Total execution time: 00:00:00.018

```
select * from covid
```

(13 rows affected)
Total execution time: 00:00:00.009

city	days	cases
DELHI	2022-01-01	100
DELHI	2022-01-02	200
DELHI	2022-01-03	300
MUMBAI	2022-01-01	100
MUMBAI	2022-01-02	100
MUMBAI	2022-01-03	300
CHENNAI	2022-01-01	100
CHENNAI	2022-01-02	200
CHENNAI	2022-01-03	150
BANGALORE	2022-01-01	100
BANGALORE	2022-01-02	300
BANGALORE	2022-01-03	200
BANGALORE	2022-01-04	400

```
select *,
rank() over (partition by city order by days ASC) as rnk1,
rank() over (partition by city order by cases asc) as rnk2,
rank() over (partition by city order by days ASC)  - rank() over (partition by
from covid
order by city,DAYS
```

(13 rows affected)
Total execution time: 00:00:00.020

city	days	cases	rnk1	rnk2	dif
BANGALORE	2022-01-01	100	1	1	0
BANGALORE	2022-01-02	300	2	3	-1
BANGALORE	2022-01-03	200	3	2	1
BANGALORE	2022-01-04	400	4	4	0
CHENNAI	2022-01-01	100	1	1	0
CHENNAI	2022-01-02	200	2	3	-1
CHENNAI	2022-01-03	150	3	2	1
DELHI	2022-01-01	100	1	1	0
DELHI	2022-01-02	200	2	2	0
DELHI	2022-01-03	300	3	3	0
MUMBAI	2022-01-01	100	1	1	0
MUMBAI	2022-01-02	100	2	1	1
MUMBAI	2022-01-03	300	3	3	0

```
with xx as(
select *,
rank() over (partition by city order by days ASC)  - rank() over (partition by
from covid
)

select city
from xx
group by city
having count(distinct dif) =1 and max(dif)=0
```

(1 row affected)
Total execution time: 00:00:00.046

city
DELHI

q 18

----------GOOOGLE SQL INTERVIEW QUESTION----------

--FIND COMPANIES WHO HAS ATLEAST 2 USERS WHO SPEAKS ENGLIAH AND GERMAN BOTH THE LANGUAGE

```
create table company_users
(
company_id int,
user_id int,
language varchar(20)
);

insert into company_users values (1,1,'English')
,(1,1,'German')
,(1,2,'English')
,(1,3,'German')
,(1,3,'English')
,(1,4,'English')
,(2,5,'English')
,(2,5,'German')
,(2,5,'Spanish')
,(2,6,'German')
,(2,6,'Spanish')
,(2,7,'English');
```

Total execution time: 00:00:00.019

```
select * from company_users
```

(12 rows affected)
Total execution time: 00:00:00.016

company_id	user_id	language
1	1	English
1	1	German
1	2	English
1	3	German
1	3	English
1	4	English
2	5	English
2	5	German
2	5	Spanish
2	6	German
2	6	Spanish
2	7	English

```
select company_id, user_id ,count(1) as language_spoke
from company_users
where language in ('English','German')
group by company_id, user_id
having count(distinct language)=2
```

(3 rows affected)
Total execution time: 00:00:00.046

company_id	user_id	language_spoke
1	1	2
1	3	2
2	5	2

q19

--Meeshoe hacker rank question

-- find how many products fall under customer budget along with list of products

--in case of clsh choose less costly product

```
create table products1
(
product_id varchar(20) ,
cost int
);
insert into products1 values ('P1',200),('P2',300),('P3',500),('P4',800);

create table customer_budget
(
customer_id int,
budget int
);

insert into customer_budget values ('100',400),('200',800),('300',1500)
```

Total execution time: 00:00:00.024

```
select * from products1
select * from customer_budget
```

(4 rows affected)
(7 rows affected)
Total execution time: 00:00:00.007

product_id	cost
P1	200
P2	300
P3	500
P4	800

customer_id	budget
1	1
2	3
3	5
4	6
100	400
200	800
300	1500

```
--- we will creating running cost first

select *,
sum(cost) over (order by cost asc) as r_cost
from products1
```

(4 rows affected)
Total execution time: 00:00:00.005

product_id	cost	r_cost
P1	200	200
P2	300	500
P3	500	1000
P4	800	1800

```
--- now we will join this with customer budget table

with running_cost as(
select *,
sum(cost) over (order by cost asc) as r_cost
from products1
)
select * from running_cost
```

(4 rows affected)
Total execution time: 00:00:00.019

product_id	cost	r_cost
P1	200	200
P2	300	500
P3	500	1000
P4	800	1800

```
with running_cost as(
select *,
sum(cost) over (order by cost asc) as r_cost
from products1
)

select customer_id,budget, count(1) as no_of_products, STRING_AGG(product_id,',
from customer_budget as cb
left join running_cost as rc on rc.r_cost <cb.budget
group by customer_id,budget
```

(7 rows affected)
Total execution time: 00:00:00.030

customer_id	budget	no_of_products	list_of_products
1	1	1	NULL
2	3	1	NULL
3	5	1	NULL
4	6	1	NULL
100	400	1	P1
200	800	2	P1,P2
300	1500	3	P1,P2,P3

q 20

--------AMAZON PROBLEM ----------

-- FIND TOTAL NUMBER OF MESSAGES EXCHANGED BETWEEN EACH PERSON PER DAY

--Horizontal sorting in sql

```
CREATE TABLE subscriber (
 sms_date date ,
 sender varchar(20) ,
 receiver varchar(20) ,
 sms_no int
);
-- insert some values
INSERT INTO subscriber VALUES ('2020-4-1', 'Avinash', 'Vibhor',10);
INSERT INTO subscriber VALUES ('2020-4-1', 'Vibhor', 'Avinash',20);
INSERT INTO subscriber VALUES ('2020-4-1', 'Avinash', 'Pawan',30);
INSERT INTO subscriber VALUES ('2020-4-1', 'Pawan', 'Avinash',20);
INSERT INTO subscriber VALUES ('2020-4-1', 'Vibhor', 'Pawan',5);
INSERT INTO subscriber VALUES ('2020-4-1', 'Pawan', 'Vibhor',8);
INSERT INTO subscriber VALUES ('2020-4-1', 'Vibhor', 'Deepak',50);
```

Total execution time: 00:00:00.016

```
select * from subscriber
```

(7 rows affected)
Total execution time: 00:00:00.008

sms_date	sender	receiver	sms_no
2020-04-01	Avinash	Vibhor	10
2020-04-01	Vibhor	Avinash	20
2020-04-01	Avinash	Pawan	30
2020-04-01	Pawan	Avinash	20
2020-04-01	Vibhor	Pawan	5
2020-04-01	Pawan	Vibhor	8
2020-04-01	Vibhor	Deepak	50

```
select *,
case when sender>receiver then sender else receiver end as p1,
case when receiver>sender then sender else receiver end as p2
from subscriber
```

(7 rows affected)
Total execution time: 00:00:00.006

sms_date	sender	receiver	sms_no	p1	p2
2020-04-01	Avinash	Vibhor	10	Vibhor	Avinash
2020-04-01	Vibhor	Avinash	20	Vibhor	Avinash
2020-04-01	Avinash	Pawan	30	Pawan	Avinash
2020-04-01	Pawan	Avinash	20	Pawan	Avinash
2020-04-01	Vibhor	Pawan	5	Vibhor	Pawan
2020-04-01	Pawan	Vibhor	8	Vibhor	Pawan
2020-04-01	Vibhor	Deepak	50	Vibhor	Deepak

```
select sms_date,p1,p2,sum(sms_no) as total_sms
from(
select sms_date,
case when sender>receiver then sender else receiver end as p1,
case when receiver>sender then sender else receiver end as p2,
sms_no
from subscriber) a
group by sms_date,p1,p2
```

(4 rows affected)
Total execution time: 00:00:00.113

sms_date	p1	p2	total_sms
2020-04-01	Pawan	Avinash	50
2020-04-01	Vibhor	Avinash	30
2020-04-01	Vibhor	Deepak	50
2020-04-01	Vibhor	Pawan	13

q 21

----tRICKY PROBLEM----------

```
CREATE TABLE [students](
 [studentid] [int] NULL,
 [studentname] [nvarchar](255) NULL,
 [subject] [nvarchar](255) NULL,
 [marks] [int] NULL,
 [testid] [int] NULL,
 [testdate] [date] NULL
)
data:
insert into students values (2,'Max Ruin','Subject1',63,1,'2022-01-02');
insert into students values (3,'Arnold','Subject1',95,1,'2022-01-02');
insert into students values (4,'Krish Star','Subject1',61,1,'2022-01-02');
insert into students values (5,'John Mike','Subject1',91,1,'2022-01-02');
insert into students values (4,'Krish Star','Subject2',71,1,'2022-01-02');
insert into students values (3,'Arnold','Subject2',32,1,'2022-01-02');
insert into students values (5,'John Mike','Subject2',61,2,'2022-11-02');
insert into students values (1,'John Deo','Subject2',60,1,'2022-01-02');
insert into students values (2,'Max Ruin','Subject2',84,1,'2022-01-02');
insert into students values (2,'Max Ruin','Subject3',29,3,'2022-01-03');
insert into students values (5,'John Mike','Subject3',98,2,'2022-11-02');
```

Total execution time: 00:00:00.019

```
SELECT * FROM students
```

(11 rows affected)
Total execution time: 00:00:00.010

studentid	studentname	subject	marks	testid	testdate
2	Max Ruin	Subject1	63	1	2022-01-02
3	Arnold	Subject1	95	1	2022-01-02
4	Krish Star	Subject1	61	1	2022-01-02
5	John Mike	Subject1	91	1	2022-01-02
4	Krish Star	Subject2	71	1	2022-01-02
3	Arnold	Subject2	32	1	2022-01-02
5	John Mike	Subject2	61	2	2022-11-02
1	John Deo	Subject2	60	1	2022-01-02
2	Max Ruin	Subject2	84	1	2022-01-02
2	Max Ruin	Subject3	29	3	2022-01-03
5	John Mike	Subject3	98	2	2022-11-02

```
---Q21 -1-- write a sql quert to find list of students who score above the aver
with avg_mark as
(select subject ,avg(marks) as avg_marks
from students
group by subject)

select * from avg_mark
```

(3 rows affected)
Total execution time: 00:00:00.019

subject	avg_marks
Subject1	77
Subject2	61
Subject3	63

```
with avg_mark as
(select subject ,avg(marks) as avg_marks
from students
group by subject)

select s.*,av.*
from avg_mark as av
inner join students as s on av.subject=s.subject
where s.marks> av.avg_marks
```

(5 rows affected)
Total execution time: 00:00:00.017

studentid	studentname	subject	marks	testid	testdate	subject	avg_marks
3	Arnold	Subject1	95	1	2022-01-02	Subject1	77
5	John Mike	Subject1	91	1	2022-01-02	Subject1	77
4	Krish Star	Subject2	71	1	2022-01-02	Subject2	61
2	Max Ruin	Subject2	84	1	2022-01-02	Subject2	61
5	John Mike	Subject3	98	2	2022-11-02	Subject3	63

```
---2) FIND SQL TO GET PERCENTAGE OF STUDENT WHO SCORE MORE THAN 90 IN EACH SUBJ

select
count(distinct case when marks>90 then studentid else null end),
count(distinct studentid),
1.0*count(distinct case when marks>90 then studentid else null end)/count(disti

from students
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(1 row affected)
Total execution time: 00:00:00.032

(No column name)	(No column name)	perc
2	5	40.000000000000

```
---3-- write query to get second highest and 2nd lowest marks in each subject
--o/p: subject . second highest , 2nd lowest

with cte as (select subject,marks,
rank() over(partition by subject order by marks asc) as rnk_asc,
rank() over(partition by subject order by marks desc) as rnk_desc
from students)
--select * from cte
--where rnk_asc=2 or rnk_desc =2
select subject,
sum(case when rnk_asc =2 then marks else null end) as second_highest_marks,
sum(case when rnk_desc=2 then marks else null end) as second_lowest_marks
from cte
group by subject
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(3 rows affected)
Total execution time: 00:00:00.010

subject	second_highest_marks	second_lowest_marks
Subject1	63	91
Subject2	60	71
Subject3	98	29

```
 --4 from each student and test , identify if there marks increae or decreased

 select *,
case when marks>prev_marks then 'inc' else 'desc' end as status
 from (
 select *,
lag(marks,1) over(partition by studentid order by testdate,subject) as prev_mar
 from students) A
```

(11 rows affected)
Total execution time: 00:00:00.008

studentid	studentname	subject	marks	testid	testdate	prev_marks	status
1	John Deo	Subject2	60	1	2022-01-02	NULL	desc
2	Max Ruin	Subject1	63	1	2022-01-02	NULL	desc
2	Max Ruin	Subject2	84	1	2022-01-02	63	inc
2	Max Ruin	Subject3	29	3	2022-01-03	84	desc
3	Arnold	Subject1	95	1	2022-01-02	NULL	desc
3	Arnold	Subject2	32	1	2022-01-02	95	desc
4	Krish Star	Subject1	61	1	2022-01-02	NULL	desc
4	Krish Star	Subject2	71	1	2022-01-02	61	inc
5	John Mike	Subject1	91	1	2022-01-02	NULL	desc
5	John Mike	Subject2	61	2	2022-11-02	91	desc
5	John Mike	Subject3	98	2	2022-11-02	61	inc

q22

--FIND THE LARGEST ORDER Y VALUE FOR EACH SALESPERSON AND DISPLAY ORDER DETAIL

-- GET RESULT WITHOUT USING SUBQUERY ,CTE,WINDOW FUNCTION ,TEMP/TABLE

```
CREATE TABLE [dbo].[int_orders](
 [order_number] [int] NOT NULL,
 [order_date] [date] NOT NULL,
 [cust_id] [int] NOT NULL,
 [salesperson_id] [int] NOT NULL,
 [amount] [float] NOT NULL
) ON [PRIMARY];

INSERT INTO [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT into [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT INTO [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT INTO [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT into [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT into [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales

INSERT into [dbo].[int_orders] ([order_number], [order_date], [cust_id], [sales
```

Total execution time: 00:00:00.014

```
select * from int_orders
```

(7 rows affected)
Total execution time: 00:00:00.033

order_number	order_date	cust_id	salesperson_id	amount
30	1995-07-14	9	1	460
10	1996-08-02	4	2	540
40	1998-01-29	7	2	2400
50	1998-02-03	6	7	600
60	1998-03-02	6	7	720
70	1998-05-06	9	7	150
20	1999-01-30	4	8	1800

```
select a.order_number,a.order_date,a.cust_id,a.salesperson_id,a.amount
from int_orders as b
LEFT join int_orders as a on a.salesperson_id=b.salesperson_id
group by a.order_number,a.order_date,a.cust_id,a.salesperson_id,a.amount
HAVING a.amount > max(b.amount)
```

(0 rows affected)
Total execution time: 00:00:00.035

order_number	order_date	cust_id	salesperson_id	amount

q 23

---on /off problem

---o/p: loggin , loggout, cnt

```
create table event_status
(
event_time varchar(10),
status varchar(10)
);
insert into event_status
values
('10:01','on'),('10:02','on'),('10:03','on'),('10:04','off'),('10:07','on'),('1
,('10:11','on'),('10:12','off');
```

Total execution time: 00:00:00.017

```
select * from event_status
```

(9 rows affected)
Total execution time: 00:00:00.010

event_time	status
10:01	on
10:02	on
10:03	on
10:04	off
10:07	on
10:08	on
10:09	off
10:11	on
10:12	off

```
select *
,lag(status,1,status) over(order by event_time asc) as prev_status
from event_status
```

(9 rows affected)
Total execution time: 00:00:00.019

event_time	status	prev_status
10:01	on	on
10:02	on	on
10:03	on	on
10:04	off	on
10:07	on	off
10:08	on	on
10:09	off	on
10:11	on	off
10:12	off	on

```
select *,
sum(case when status='on' and prev_status='off'then 0 end ) over(order by event
from
(select *
,lag(status,1,status) over(order by event_time asc) as prev_status
from event_status) A
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(9 rows affected)
Total execution time: 00:00:00.010

event_time	status	prev_status	group_key
10:01	on	on	NULL
10:02	on	on	NULL
10:03	on	on	NULL
10:04	off	on	NULL
10:07	on	off	0
10:08	on	on	0
10:09	off	on	0
10:11	on	off	0
10:12	off	on	0

```
with cte_a as(
select *,
sum(case when status='on' and prev_status='off'then 0 end ) over(order by event
from
(select *
,lag(status,1,status) over(order by event_time asc) as prev_status
from event_status) A)

select min(event_time) as loginn, max(event_time) as loggoff , count(1)-1 as or
from cte_a
group by group_key
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(2 rows affected)
Total execution time: 00:00:00.023

loginn	loggoff	on_off
10:01	10:04	3
10:07	10:12	4

q 24

--LeetCode problem where we need to pivot the data from row to column. The interesting part about this problem is

--we don't have a common key to pivot the data on.

--player location by geography

```
create table players_location
(
name varchar(20),
city varchar(20)
);
delete from players_location;
insert into players_location
values ('Sachin','Mumbai'),('Virat','Delhi') , ('Rahul','Bangalore'),('Rohit','
```

Total execution time: 00:00:00.020

```
select * from players_location
```

(5 rows affected)
Total execution time: 00:00:00.003

name	city
Sachin	Mumbai
Virat	Delhi
Rahul	Bangalore
Rohit	Mumbai
Mayank	Bangalore

```
select *,
row_number() over(partition by city order by name asc) as player_group
from players_location
```

(5 rows affected)
Total execution time: 00:00:00.013

name	city	player_group
Mayank	Bangalore	1
Rahul	Bangalore	2
Virat	Delhi	1
Rohit	Mumbai	1
Sachin	Mumbai	2

```
with cte_q as(select *,
row_number() over(partition by city order by name asc) as player_group
from players_location)

select * from cte_q
```

(5 rows affected)
Total execution time: 00:00:00.021

name	city	player_group
Mayank	Bangalore	1
Rahul	Bangalore	2
Virat	Delhi	1
Rohit	Mumbai	1
Sachin	Mumbai	2

```
with cte_q as(select *,
row_number() over(partition by city order by name asc) as player_group
from players_location)

SELECT player_group
,max(case when city = 'Banglore' then name else '' end) as Banglore
,max(case when city = 'Delhi' then name else '' end) as Delhi
,max(case when city = 'Mumbai' then name else '' end) as Mumbai
from cte_q
group by player_group
--order by player_group
```

(2 rows affected)
Total execution time: 00:00:00.011

player_group	Banglore	Delhi	Mumbai
1		Virat	Rohit
2			Sachin

q 25

--query to find median salary of each employee

```
create table employee
(
emp_id int,
company varchar(10),
salary int
);

insert into employee values (1,'A',2341)
insert into employee values (2,'A',341)
insert into employee values (3,'A',15)
insert into employee values (4,'A',15314)
insert into employee values (5,'A',451)
insert into employee values (6,'A',513)
insert into employee values (7,'B',15)
insert into employee values (8,'B',13)
insert into employee values (9,'B',1154)
insert into employee values (10,'B',1345)
insert into employee values (11,'B',1221)
insert into employee values (12,'B',234)
insert into employee values (13,'C',2345)
insert into employee values (14,'C',2645)
insert into employee values (15,'C',2645)
insert into employee values (16,'C',2652)
insert into employee values (17,'C',65);
```

Total execution time: 00:00:00.028

```
select * from employee
```

(17 rows affected)
Total execution time: 00:00:00.008

emp_id	company	salary
1	A	2341
2	A	341
3	A	15
4	A	15314
5	A	451
6	A	513
7	B	15
8	B	13
9	B	1154
10	B	1345
11	B	1221
12	B	234
13	C	2345
14	C	2645
15	C	2645
16	C	2652
17	C	65

```
-- median for - 2,5,4,8,9 - 4
--medain for - 2,5,6,7,8,9 - 6.5 (for even recods -6+7=13/2)

select *,
row_number() over(partition by company order by salary asc) as rn
,count(1) over(partition by company) as total_cnt
from employee
order by company,salary
```

(17 rows affected)
Total execution time: 00:00:00.023

emp_id	company	salary	rn	total_cnt
3	A	15	1	6
2	A	341	2	6
5	A	451	3	6
6	A	513	4	6
1	A	2341	5	6
4	A	15314	6	6
8	B	13	1	6
7	B	15	2	6
12	B	234	3	6
9	B	1154	4	6
11	B	1221	5	6
10	B	1345	6	6
17	C	65	1	5
13	C	2345	2	5
14	C	2645	3	5
15	C	2645	4	5
16	C	2652	5	5

```
select *
from (
select *,
row_number() over(partition by company order by salary asc) as rn
,count(1) over(partition by company) as total_cnt
from employee
) A
where rn between total_cnt*1.0/2 and total_cnt*1.0/2+1
```

(5 rows affected)
Total execution time: 00:00:00.034

emp_id	company	salary	rn	total_cnt
5	A	451	3	6
6	A	513	4	6
12	B	234	3	6
9	B	1154	4	6
14	C	2645	3	5

```
select company,avg(salary)
from (
select *,
row_number() over(partition by company order by salary asc) as rn
,count(1) over(partition by company) as total_cnt
from employee
) A
where rn between total_cnt*1.0/2 and total_cnt*1.0/2+1
group by company
```

(3 rows affected)
Total execution time: 00:00:00.028

company	(No column name)
A	482
B	694
C	2645

Q 26

/* WRITE A QUERY TO DISPALAY RECORDS WHICH HAS 3 OR MORE CONSECUTIVE ROWS WITH THE AMOUNT OF PEOPLE MORE THAN 100 (INCLUSIVE) EACH DAY

hard leet code problem called human traffic of stadium. We will be using SQL analytical functions to solve the problem */

```
create table stadium (
id int,
visit_date date,
no_of_people int
);

insert into stadium
values (1,'2017-07-01',10)
,(2,'2017-07-02',109)
,(3,'2017-07-03',150)
,(4,'2017-07-04',99)
,(5,'2017-07-05',145)
,(6,'2017-07-06',1455)
,(7,'2017-07-07',199)
,(8,'2017-07-08',188);
```

Total execution time: 00:00:00.017

```
SELECT * FROM STADIUM
```

(8 rows affected)
Total execution time: 00:00:00.010

id	visit_date	no_of_people
1	2017-07-01	10
2	2017-07-02	109
3	2017-07-03	150
4	2017-07-04	99
5	2017-07-05	145
6	2017-07-06	1455
7	2017-07-07	199
8	2017-07-08	188

```
select *
,row_number() over (order by visit_date) as rn
from stadium
where no_of_people >=100
```

(6 rows affected)
Total execution time: 00:00:00.018

id	visit_date	no_of_people	rn
2	2017-07-02	109	1
3	2017-07-03	150	2
5	2017-07-05	145	3
6	2017-07-06	1455	4
7	2017-07-07	199	5
8	2017-07-08	188	6

```
with group_number 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 group_number
where grp in (
select grp
from group_number
group by grp
having count(1)>=3)
```

(4 rows affected)
Total execution time: 00:00:00.024

id	visit_date	no_of_people
5	2017-07-05	145
6	2017-07-06	1455
7	2017-07-07	199
8	2017-07-08	188

q 27

/* BUSINESS CITY TABLE HAS DATA FROM DAY UDAAN HAS STARTED OPERATION

WRITE A QUERY FOR YEAR WISE COUNT OF NEW CITY WHERE UDAN STARTED THERE OPERATION

o/p : year, count_new city

*/

```
create table business_city (
business_date date,
city_id int
);
delete from business_city;
insert into business_city
values(cast('2020-01-02' as date),3),(cast('2020-07-01' as date),7),(cast('2021
,(cast('2022-12-01' as date),3),(cast('2022-12-15' as date),3),(cast('2022-02-2
```

Total execution time: 00:00:00.020

```
SELECT * FROM business_city
```

(7 rows affected)
Total execution time: 00:00:00.030

business_date	city_id
2020-01-02	3
2020-07-01	7
2021-01-01	3
2021-02-03	19
2022-12-01	3
2022-12-15	3
2022-02-28	12

```
select datepart(year,business_date) as year_ud, city_id
from business_city
```

(7 rows affected)
Total execution time: 00:00:00.058

year_ud	city_id
2020	3
2020	7
2021	3
2021	19
2022	3
2022	3
2022	12

```
--- taking self join from this table

with cte as (
    select datepart(year,business_date) as bus_yr, city_id
from business_city
)
 select c1.bus_yr, count(distinct case when c2.city_id is null then c1.city_id
 from cte as c1
 left join cte c2 on c1.bus_yr > c2.bus_yr and c1.city_id = c2.city_id
group by c1.bus_yr
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(3 rows affected)
Total execution time: 00:00:00.047

bus_yr	#_no_of_new_city
2020	2
2021	1
2022	1

q 28

/* there are 3 rows in movie hall each with 10 seats in each row , write sql to find 4 consecutive empty seats . */

```
create table movie(
seat varchar(50),occupancy int
);
insert into movie values('a1',1),('a2',1),('a3',0),('a4',0),('a5',0),('a6',0),(
('b1',0),('b2',0),('b3',0),('b4',1),('b5',1),('b6',1),('b7',1),('b8',0),('b9',0
('c1',0),('c2',1),('c3',0),('c4',1),('c5',1),('c6',0),('c7',1),('c8',0),('c9',0
```

Total execution time: 00:00:00.019

```
select * from movie
```

(30 rows affected)
Total execution time: 00:00:00.006

seat	occupancy
a1	1
a2	1
a3	0
a4	0
a5	0
a6	0
a7	1
a8	1
a9	0
a10	0
b1	0
b2	0
b3	0
b4	1
b5	1
b6	1
b7	1
b8	0
b9	0
b10	0
c1	0
c2	1
c3	0
c4	1
c5	1
c6	0
c7	1
c8	0
c9	0
c10	1

```
--we will cast in substring because we have use order by later
select *,
left(seat,1) as row_id , cast(SUBSTRING(seat,2,2) as int) as seat_id
from movie
```

(30 rows affected)
Total execution time: 00:00:00.006

seat	occupancy	row_id	seat_id
a1	1	a	1
a2	1	a	2
a3	0	a	3
a4	0	a	4
a5	0	a	5
a6	0	a	6
a7	1	a	7
a8	1	a	8
a9	0	a	9
a10	0	a	10
b1	0	b	1
b2	0	b	2
b3	0	b	3
b4	1	b	4
b5	1	b	5
b6	1	b	6
b7	1	b	7
b8	0	b	8
b9	0	b	9
b10	0	b	10
c1	0	c	1
c2	1	c	2
c3	0	c	3
c4	1	c	4
c5	1	c	5
c6	0	c	6
c7	1	c	7
c8	0	c	8
c9	0	c	9
c10	1	c	10

```
with cte1 as (
    select *,
left(seat,1) as row_id , cast(SUBSTRING(seat,2,2) as int) as seat_id
from movie
)

select * from cte1
```

(30 rows affected)
Total execution time: 00:00:00.005

seat	occupancy	row_id	seat_id
a1	1	a	1
a2	1	a	2
a3	0	a	3
a4	0	a	4
a5	0	a	5
a6	0	a	6
a7	1	a	7
a8	1	a	8
a9	0	a	9
a10	0	a	10
b1	0	b	1
b2	0	b	2
b3	0	b	3
b4	1	b	4
b5	1	b	5
b6	1	b	6
b7	1	b	7
b8	0	b	8
b9	0	b	9
b10	0	b	10
c1	0	c	1
c2	1	c	2
c3	0	c	3
c4	1	c	4
c5	1	c	5
c6	0	c	6
c7	1	c	7
c8	0	c	8
c9	0	c	9
c10	1	c	10

```
with cte1 as (
```

```
    select *,
left(seat,1) as row_id , cast(SUBSTRING(seat,2,2) as int) as seat_id
from movie
)

,cte2 as(
select *,
max(occupancy)over(partition by row_id order by seat_id rows between current ro
count(occupancy)over(partition by row_id order by seat_id rows between current
from cte1
)

select * from cte2
```

(30 rows affected)
Total execution time: 00:00:00.011

seat	occupancy	row_id	seat_id	is_4_empty	is_4_empty_cnt
a1	1	a	1	1	4
a2	1	a	2	1	4
a3	0	a	3	0	4
a4	0	a	4	1	4
a5	0	a	5	1	4
a6	0	a	6	1	4
a7	1	a	7	1	4
a8	1	a	8	1	3
a9	0	a	9	0	2
a10	0	a	10	0	1
b1	0	b	1	1	4
b2	0	b	2	1	4
b3	0	b	3	1	4
b4	1	b	4	1	4
b5	1	b	5	1	4
b6	1	b	6	1	4
b7	1	b	7	1	4
b8	0	b	8	0	3
b9	0	b	9	0	2
b10	0	b	10	0	1
c1	0	c	1	1	4
c2	1	c	2	1	4
c3	0	c	3	1	4
c4	1	c	4	1	4
c5	1	c	5	1	4
c6	0	c	6	1	4
c7	1	c	7	1	4
c8	0	c	8	1	3
c9	0	c	9	1	2
c10	1	c	10	1	1

```
with cte1 as (
    select *,
left(seat,1) as row_id , cast(SUBSTRING(seat,2,2) as int) as seat_id
from movie
)
,cte2 as(
select *,
max(occupancy)over(partition by row_id order by seat_id rows between current ro
count(occupancy)over(partition by row_id order by seat_id rows between current
from cte1
)

,cte3 as (
select * from cte2
where is_4_empty = 0 and is_4_empty_cnt=4
)

select * from cte3
```

(1 row affected)
Total execution time: 00:00:00.015

seat	occupancy	row_id	seat_id	is_4_empty	is_4_empty_cnt
a3	0	a	3	0	4

```
with cte1 as (
    select *,
left(seat,1) as row_id , cast(SUBSTRING(seat,2,2) as int) as seat_id
from movie
)
,cte2 as(
select *,
max(occupancy)over(partition by row_id order by seat_id rows between current ro
count(occupancy)over(partition by row_id order by seat_id rows between current
from cte1
)
,cte3 as (
select * from cte2
where is_4_empty = 0 and is_4_empty_cnt=4
)

select cte2.*
from cte2
inner join cte3 on cte2.row_id = cte3.row_id and cte2.seat_id between cte3.seat
```

(4 rows affected)
Total execution time: 00:00:00.023

seat	occupancy	row_id	seat_id	is_4_empty	is_4_empty_cnt
a3	0	a	3	0	4
a4	0	a	4	1	4
a5	0	a	5	1	4
a6	0	a	6	1	4

q 29

/* WRITE A SQL TO DETERMINE PHONE NO THAT SATISFY BELOW CONDITION :

1) NUMBER HAS BOTH INCOMING AND OUTGOING CALLS

2) SUM OF DURATION OF OUTGOING CALL SHOULD BE GREATE RTHEN SUM OF DURATION OF INCOMING CALLS */

```
create table call_details  (
call_type varchar(10),
call_number varchar(12),
call_duration int
);

insert into call_details
values ('OUT','181868',13),('OUT','2159010',8)
,('OUT','2159010',178),('SMS','4153810',1),('OUT','2159010',152),('OUT','914015
,('SMS','9168204',1),('OUT','9168204',576),('INC','2159010',5),('INC','2159010'
,('SMS','4535614',1),('OUT','181868',20),('INC','181868',54),('INC','218748',20
,('INC','197432',66),('SMS','2159010',1),('SMS','4535614',1);
```

Total execution time: 00:00:00.036

```
SELECT * FROM CALL_DETAILS
```

(20 rows affected)
Total execution time: 00:00:00.017

call_type	call_number	call_duration
OUT	181868	13
OUT	2159010	8
OUT	2159010	178
SMS	4153810	1
OUT	2159010	152
OUT	9140152	18
SMS	4162672	1
SMS	9168204	1
OUT	9168204	576
INC	2159010	5
INC	2159010	4
SMS	2159010	1
SMS	4535614	1
OUT	181868	20
INC	181868	54
INC	218748	20
INC	2159010	9
INC	197432	66
SMS	2159010	1
SMS	4535614	1

```
---- 1) method
----cte and filter clause

select call_number ,
sum(case when call_type ='OUT' then call_duration else null end ) as out_durat
 sum(case when call_type ='INC' then call_duration else null end ) as INCOMING
from call_details
group by call_number
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(9 rows affected)
Total execution time: 00:00:00.036

call_number	out_duration	INCOMING_duration
181868	33	54
197432	NULL	66
2159010	338	18
218748	NULL	20
4153810	NULL	NULL
4162672	NULL	NULL
4535614	NULL	NULL
9140152	18	NULL
9168204	576	NULL

```
with cte as(
     select call_number ,
 sum(case when call_type ='OUT' then call_duration else null end ) as out_durat
  sum(case when call_type ='INC' then call_duration else null end ) as INCOMING
 from call_details
 group by call_number
)

select call_number
from cte
where out_duration is not null and INCOMING_duration is not null and  out_durat
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(1 row affected)
Total execution time: 00:00:00.116

call_number
2159010

```
---- method 2 using having clause

select call_number
from call_details
 group by call_number
having sum(case when call_type ='OUT' then call_duration else null end ) >0 anc
sum(case when call_type ='INC' then call_duration else null end ) >0 and
 sum(case when call_type ='OUT' then call_duration else null end ) > sum(case w
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(1 row affected)
Total execution time: 00:00:00.031

call_number
2159010

```
----- method 3 ,using xte and joins

with cte_out as(
   select call_number,
   sum(call_duration ) as duration
   from call_details
   where call_type = 'OUT'
   group by call_number)
,cte_in as (
   select call_number,
   sum(call_duration ) as duration
   from call_details
   where call_type = 'INC'
   group by call_number
)

select cte_out.call_number
from cte_out
inner join cte_in on cte_out.call_number = cte_in.call_number
where cte_out.duration> cte_in.duration
```

(1 row affected)
Total execution time: 00:00:00.019

call_number
2159010

q 30

-- WRITE A SQL TO POPULATE CATEGORY VALUES TO THE LAST NOT NULL VA;UE

```
create table brands
(
category varchar(20),
brand_name varchar(20)
);
insert into brands values
('chocolates','5-star')
,(null,'dairy milk')
,(null,'perk')
,(null,'eclair')
,('Biscuits','britannia')
,(null,'good day')
,(null,'boost');
```

Total execution time: 00:00:00.025

```
select * from brands
```

(7 rows affected)
Total execution time: 00:00:00.007

category	brand_name
chocolates	5-star
NULL	dairy milk
NULL	perk
NULL	eclair
Biscuits	britannia
NULL	good day
NULL	boost

```
---lead lag will not work here
select *,
row_number()over(order by(select null)) as rn
from brands
```

(7 rows affected)
Total execution time: 00:00:00.008

category	brand_name	rn
chocolates	5-star	1
NULL	dairy milk	2
NULL	perk	3
NULL	eclair	4
Biscuits	britannia	5
NULL	good day	6
NULL	boost	7

```
--- to find rn for not null value
with cte as(select *,
row_number()over(order by(select null)) as rn
from brands)
select *
from cte where category is not NULL
```

(2 rows affected)
Total execution time: 00:00:00.017

category	brand_name	rn
chocolates	5-star	1
Biscuits	britannia	5

```
---now we can sy (1-4 is choclqte and use lead)
with cte1 as(select *,
row_number()over(order by(select null)) as rn
from brands)
,cte2 as(
    select *,
    lead(rn,1,9999)over(order by rn) as next_rn
     from cte1
     where category is not NULL)

select cte2.category, cte1.brand_name
from cte1
inner join cte2 on cte1.rn >= cte2.rn and cte1.rn <= cte2.next_rn-1
```

(7 rows affected)
Total execution time: 00:00:00.019

category	brand_name
chocolates	5-star
chocolates	dairy milk
chocolates	perk
chocolates	eclair
Biscuits	britannia
Biscuits	good day
Biscuits	boost

q 31

/* Find the Quiet Students in All Exams. We will discuss a step by step solution. scroll down for scripts.

8/

```
create table students1
(
student_id int,
student_name varchar(20)
);
insert into students1 values
(1,'Daniel'),(2,'Jade'),(3,'Stella'),(4,'Jonathan'),(5,'Will');

create table exams1
(
exam_id int,
student_id int,
score int);

insert into exams1 values
(10,1,70),(10,2,80),(10,3,90),(20,1,80),(30,1,70),(30,3,80),(30,4,90),(40,1,60)
,(40,2,70),(40,4,80);
```

Total execution time: 00:00:00.011

```
select * from students1
select * from exams1
```

(5 rows affected)
(10 rows affected)
Total execution time: 00:00:00.010

student_id	student_name
1	Daniel
2	Jade
3	Stella
4	Jonathan
5	Will

exam_id	student_id	score
10	1	70
10	2	80
10	3	90
20	1	80
30	1	70
30	3	80
30	4	90
40	1	60
40	2	70
40	4	80

```
select exam_id , min(score) as minimum , max(score) as maximum
from exams1
group by exam_id
```

(4 rows affected)
Total execution time: 00:00:00.024

exam_id	minimum	maximum
10	70	90
20	80	80
30	70	90
40	60	80

```
with all_scores as(
select exam_id , min(score) as minimum , max(score) as maximum
from exams1
group by exam_id
)
select ex.*,minimum, maximum
from exams1 ex
inner join all_scores on ex.exam_id = all_scores.exam_id
```

(10 rows affected)
Total execution time: 00:00:00.018

exam_id	student_id	score	minimum	maximum
10	1	70	70	90
10	2	80	70	90
10	3	90	70	90
20	1	80	80	80
30	1	70	70	90
30	3	80	70	90
30	4	90	70	90
40	1	60	60	80
40	2	70	60	80
40	4	80	60	80

```
with all_scores as(
select exam_id , min(score) as minimum , max(score) as maximum
from exams1
group by exam_id
)
select ex.student_id,
max(case when score = min(score) or score = max(score)then 1 else 0 end) AS rec
from exams1 ex
inner join all_scores on ex.exam_id = all_scores.exam_id
group by student_id
```

Total execution time: 00:00:00.010

q 32

/* where from a phone log history we need to find if the caller had done first and last call for the day to the same person.

FIND CALLER WHOSE 1ST AND LAST CALL WAS TO SAME PERSON IN A GIVEN DAY */

```
create table phonelog(
    Callerid int,
    Recipientid int,
    Datecalled datetime
);

insert into phonelog(Callerid, Recipientid, Datecalled)
values(1, 2, '2019-01-01 09:00:00.000'),
       (1, 3, '2019-01-01 17:00:00.000'),
       (1, 4, '2019-01-01 23:00:00.000'),
       (2, 5, '2019-07-05 09:00:00.000'),
       (2, 3, '2019-07-05 17:00:00.000'),
       (2, 3, '2019-07-05 17:20:00.000'),
       (2, 5, '2019-07-05 23:00:00.000'),
       (2, 3, '2019-08-01 09:00:00.000'),
       (2, 3, '2019-08-01 17:00:00.000'),
       (2, 5, '2019-08-01 19:30:00.000'),
       (2, 4, '2019-08-02 09:00:00.000'),
       (2, 5, '2019-08-02 10:00:00.000'),
       (2, 5, '2019-08-02 10:45:00.000'),
       (2, 4, '2019-08-02 11:00:00.000');
```

Total execution time: 00:00:00.019

```
SELECT * FROM PHONELOG
select callerid, cast(datecalled as date) as called_date,
min(datecalled) as first_call, max(datecalled) as lastcall
from phonelog
group by callerid, cast(datecalled as date)
```

(14 rows affected)
(4 rows affected)
Total execution time: 00:00:00.029

Callerid	Recipientid	Datecalled
1	2	2019-01-01 09:00:00.000
1	3	2019-01-01 17:00:00.000
1	4	2019-01-01 23:00:00.000
2	5	2019-07-05 09:00:00.000
2	3	2019-07-05 17:00:00.000
2	3	2019-07-05 17:20:00.000
2	5	2019-07-05 23:00:00.000
2	3	2019-08-01 09:00:00.000
2	3	2019-08-01 17:00:00.000
2	5	2019-08-01 19:30:00.000
2	4	2019-08-02 09:00:00.000
2	5	2019-08-02 10:00:00.000
2	5	2019-08-02 10:45:00.000
2	4	2019-08-02 11:00:00.000

callerid	called_date	first_call	lastcall
1	2019-01-01	2019-01-01 09:00:00.000	2019-01-01 23:00:00.000
2	2019-07-05	2019-07-05 09:00:00.000	2019-07-05 23:00:00.000
2	2019-08-01	2019-08-01 09:00:00.000	2019-08-01 19:30:00.000
2	2019-08-02	2019-08-02 09:00:00.000	2019-08-02 11:00:00.000

```
with calls as (
    select callerid, cast(datecalled as date) as called_date,
min(datecalled) as first_call, max(datecalled) as lastcall
from phonelog
group by callerid, cast(datecalled as date)
)
select c.*,
p1.Recipientid as first_rec, p2.Recipientid as last_rec
from calls as c
inner join phonelog p1 on c.callerid= P1.callerid and c.first_call = p1.datecal
inner join phonelog p2 on c.callerid= p2.callerid and c.first_call = p2.datecal
```

(4 rows affected)
Total execution time: 00:00:00.015

callerid	called_date	first_call	lastcall	first_rec	last_rec
1	2019-01-01	2019-01-01 09:00:00.000	2019-01-01 23:00:00.000	2	2
2	2019-07-05	2019-07-05 09:00:00.000	2019-07-05 23:00:00.000	5	5
2	2019-08-01	2019-08-01 09:00:00.000	2019-08-01 19:30:00.000	3	3
2	2019-08-02	2019-08-02 09:00:00.000	2019-08-02 11:00:00.000	4	4

Q 33

--this problem we have to write a SQL to build a team with a combination of seniors and juniors within a given salary budget

/*A COMPANY WANTS TO HIRE NEW EMPLOYEES THE BUDGET OF THE COMPANY FOR SALARIES IS $70000, THE COMPANIES CRITERIA FOR HIRING ARE :

1) KEEP HIRING SENIOR WITH SAMLLEST SALARY UNTIL YOU CANNOT HIRE ANY MORE SENIOR , USE THE REMAINING BUDET TO

HIRE THE JUINOR WITH SMALLEST SALARY KEEP HIRING THE JUNIOR UNTIL YOU CANNOT HIRE ANY MORE JUNIOR

WRITE A SQL QUERY TO FIND NO OF SENIOR AND JUNIOR HIRES UNDER MENTIONED CRITERIA.*/

```
create table candidates (
emp_id int,
experience varchar(20),
salary int
);
delete from candidates;
insert into candidates values
(1,'Junior',10000),(2,'Junior',15000),(3,'Junior',40000),(4,'Senior',16000),(5,
```

Total execution time: 00:00:00.017

```
SELECT * FROM candidates
```

(6 rows affected)
Total execution time: 00:00:00.031

emp_id	experience	salary
1	Junior	10000
2	Junior	15000
3	Junior	40000
4	Senior	16000
5	Senior	20000
6	Senior	50000

```
-- find the running sum on epereince basis
select *,
sum(salary)over(partition by experience order by salary) as running_sum
from candidates
-- here plus point is salaries are not dupliacte , if salary is dupliacte then
-- so we will use advanced aggregate function .
```

(6 rows affected)
Total execution time: 00:00:00.022

emp_id	experience	salary	running_sum
1	Junior	10000	10000
2	Junior	15000	25000
3	Junior	40000	65000
4	Senior	16000	16000
5	Senior	20000	36000
6	Senior	50000	86000

```
with tot_sal as(select *,
sum(salary) over(partition by experience order by salary asc rows between unbou
from candidates),
seniors as (
    select * from tot_sal
    where experience ='Senior' and running_sum < 70000)
select * from tot_sal
where experience = 'Junior' and running_sum < 70000-(select sum(salary) from se
union all
select * from seniors
```

(4 rows affected)
Total execution time: 00:00:00.060

emp_id	experience	salary	running_sum
1	Junior	10000	10000
2	Junior	15000	25000
4	Senior	16000	16000
5	Senior	20000	36000

Q 34

/* WRITE ALIST TO MENTION EMPLOYEE NAME ALONG WITH THERE MANAGER NAME ANSD SENIOR MANAGER NAME , SENIOR MANAGER IS MANAGERS MANAGER */

```
create table emp(
emp_id int,
emp_name varchar(20),
department_id int,
salary int,
manager_id int,
emp_age int);

insert into emp
values
(1, 'Ankit', 100,10000, 4, 39);
insert into emp
values (2, 'Mohit', 100, 15000, 5, 48);
insert into emp
values (3, 'Vikas', 100, 12000,4,37);
insert into emp
values (4, 'Rohit', 100, 14000, 2, 16);
insert into emp
values (5, 'Mudit', 200, 20000, 6,55);
insert into emp
values (6, 'Agam', 200, 12000,2, 14);
insert into emp
values (7, 'Sanjay', 200, 9000, 2,13);
insert into emp
values (8, 'Ashish', 200,5000,2,12);
insert into emp
values (9, 'Mukesh',300,6000,6,51);
insert into emp
values (10, 'Rakesh',500,7000,6,50);
```

Total execution time: 00:00:00.039

```
SELECT * FROM EMP
```

(19 rows affected)
Total execution time: 00:00:00.015

emp_id	emp_name	department_id	salary	manager_id	emp_age
1	Ankit	100	10000	4	39
2	Mohit	100	15000	5	48
3	Vikas	100	12000	4	37
4	Rohit	100	14000	2	16
5	Mudit	200	20000	6	55
6	Agam	200	12000	2	14
7	Sanjay	200	9000	2	13
8	Ashish	200	5000	2	12
9	Mukesh	300	6000	6	51
10	Rakesh	500	7000	6	50
2	Mohit	100	15000	5	48
3	Vikas	100	10000	4	37
4	Rohit	100	5000	2	16
5	Mudit	200	12000	6	55
6	Agam	200	12000	2	14
7	Sanjay	200	9000	2	13
8	Ashish	200	5000	2	12
9	Mukesh	300	6000	6	51
10	Rakesh	300	7000	6	50

```
select e.emp_id,e.emp_name as junior_name,m.emp_name as manager_name,sm.emp_nam
from emp e
left join emp m on e.manager_id=m.emp_id
left join emp sm on m.manager_id=sm.emp_id
```

(76 rows affected)
Total execution time: 00:00:00.028

emp_id	junior_name	manager_name	senior_manager
1	Ankit	Rohit	Mohit
1	Ankit	Rohit	Mohit
1	Ankit	Rohit	Mohit
1	Ankit	Rohit	Mohit
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
3	Vikas	Rohit	Mohit
3	Vikas	Rohit	Mohit
3	Vikas	Rohit	Mohit

3	Vikas	Rohit	Mohit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
5	Mudit	Agam	Mohit
5	Mudit	Agam	Mohit
5	Mudit	Agam	Mohit
5	Mudit	Agam	Mohit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
2	Mohit	Mudit	Agam
3	Vikas	Rohit	Mohit
3	Vikas	Rohit	Mohit
3	Vikas	Rohit	Mohit
3	Vikas	Rohit	Mohit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
4	Rohit	Mohit	Mudit
5	Mudit	Agam	Mohit
5	Mudit	Agam	Mohit
5	Mudit	Agam	Mohit

5	Mudit	Agam	Mohit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
6	Agam	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
7	Sanjay	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
8	Ashish	Mohit	Mudit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
9	Mukesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit
10	Rakesh	Agam	Mohit

Q 35

/* there is live production table orders that captures order indormation in in real time and a copy of order is taken at that time also

write a query to find new records after snapshort was taken

eg - order table : 5000 records

order_copy : 3000 records

```
create table tbl_orders (
order_id integer,
order_date date
);
insert into tbl_orders
values (1,'2022-10-21'),(2,'2022-10-22'),
(3,'2022-10-25'),(4,'2022-10-25');
```

Total execution time: 00:00:00.017

```
select * into tbl_orders_copy from  tbl_orders;
```

Total execution time: 00:00:00.007

```
select * from tbl_orders;
insert into tbl_orders
values (5,'2022-10-26'),(6,'2022-10-26');
delete from tbl_orders where order_id=1;
```

(7 rows affected)
(2 rows affected)
(0 rows affected)
Total execution time: 00:00:00.021

order_id	order_date
5	2022-10-26
2	2022-10-22
3	2022-10-25
4	2022-10-25
5	2022-10-26
6	2022-10-26
6	2022-10-26

```
select * FROM tbl_orders_copy;
select * from tbl_orders;
```

(4 rows affected)
(9 rows affected)
Total execution time: 00:00:00.009

order_id	order_date
1	2022-10-21
2	2022-10-22
3	2022-10-25
4	2022-10-25

order_id	order_date
5	2022-10-26
2	2022-10-22
3	2022-10-25
4	2022-10-25
5	2022-10-26
6	2022-10-26
6	2022-10-26
5	2022-10-26
6	2022-10-26

```
select o.order_id,c.order_id,
case when c.order_id is null then 'INSERT' else ' ' end,
case when o.order_id is null then 'deleted' else  ' ' end
 from tbl_orders o
full outer join tbl_orders_copy c on  o.order_id =c.order_id
```

(10 rows affected)
Total execution time: 00:00:00.022

order_id	order_id	(No column name)	(No column name)
5	NULL	INSERT	
2	2		
3	3		
4	4		
5	NULL	INSERT	
6	NULL	INSERT	
6	NULL	INSERT	
5	NULL	INSERT	
6	NULL	INSERT	
NULL	1		deleted

```
select coalesce(o.order_id,c.order_id),
case when c.order_id is null then 'INSERT' else ' ' end,
case when o.order_id is null then 'deleted' else  ' ' end
 from tbl_orders o
full outer join tbl_orders_copy c on  o.order_id =c.order_id
where c.order_id is null or o.order_id is NULL
```

(7 rows affected)
Total execution time: 00:00:00.013

(No column name)	(No column name)	(No column name)
5	INSERT	
5	INSERT	
6	INSERT	
6	INSERT	
5	INSERT	
6	INSERT	
1		deleted

Q 36

----------Q 46----------

-----uber interview question

/* write aquery to print total rides and profit rides for each driver

profit ride is when end location of current ride is same as start location on next ride

o/p: id, total_rides,profit_rides

2 methods - 1) window lead function

2) join function

*/

```
create table drivers(id varchar(10), start_time time, end_time time, start_loc
insert into drivers values('dri_1', '09:00', '09:30', 'a','b'),('dri_1', '09:30
insert into drivers values('dri_1', '12:00', '12:30', 'f','g'),('dri_1', '13:30
insert into drivers values('dri_2', '12:15', '12:30', 'f','g'),('dri_2', '13:30
```

Total execution time: 00:00:00.018

```
select * from drivers
```

(7 rows affected)
Total execution time: 00:00:00.011

id	start_time	end_time	start_loc	end_loc
dri_1	09:00:00	09:30:00	a	b
dri_1	09:30:00	10:30:00	b	c
dri_1	11:00:00	11:30:00	d	e
dri_1	12:00:00	12:30:00	f	g
dri_1	13:30:00	14:30:00	c	h
dri_2	12:15:00	12:30:00	f	g
dri_2	13:30:00	14:30:00	c	h

```
select *,
lead(start_loc,1) over(partition by id order by start_time asc) as next_start_l
from drivers
```

(7 rows affected)
Total execution time: 00:00:00.047

id	start_time	end_time	start_loc	end_loc	next_start_loc
dri_1	09:00:00	09:30:00	a	b	b
dri_1	09:30:00	10:30:00	b	c	d
dri_1	11:00:00	11:30:00	d	e	f
dri_1	12:00:00	12:30:00	f	g	c
dri_1	13:30:00	14:30:00	c	h	NULL
dri_2	12:15:00	12:30:00	f	g	c
dri_2	13:30:00	14:30:00	c	h	NULL

```
-----by lead window function

select id,count(1) as total_rides,
sum(case when end_loc = next_start_loc then 1 else 0 end) as profit_rides
from (select *,
lead(start_loc,1) over(partition by id order by start_time asc) as next_start_l
from drivers ) A
group by id
```

(2 rows affected)
Total execution time: 00:00:00.034

id	total_rides	profit_rides
dri_1	5	1
dri_2	2	0

```
--- by self join
with rides as(
    select *,
    row_number() over(partition by id order by start_time asc) as rn
    from drivers
)
select r1.rides,count(1) as total_rides, count(r2.id) as profit_rides
from rides as r1
left join rides r2 on r1.id = r2.id and r1.end_loc=r2.start_loc and r1.rn+1=r2.
group  by r1.id
```

Total execution time: 00:00:00.017

Q 37

/* WRITE A SQL QUERY TO FIND USERS WHO PURCHASED DIFFERENT PRODUCTS ON DIFFERENT DATES IE

PRODUCT PURCHASED ON ANY GIVEN DAY IS NOT REPEATED ON ANY OTHER DAY

O/P : USER _ID

I/P: USER_ID,PRODUCT_ID,PURCHASED_DATE

*/

```
create table purchase_history
(userid int
,productid int
,purchasedate date
);
SET DATEFORMAT dmy;
insert into purchase_history values
(1,1,'23-01-2012')
,(1,2,'23-01-2012')
,(1,3,'25-01-2012')
,(2,1,'23-01-2012')
,(2,2,'23-01-2012')
,(2,2,'25-01-2012')
,(2,4,'25-01-2012')
,(3,4,'23-01-2012')
,(3,1,'23-01-2012')
,(4,1,'23-01-2012')
,(4,2,'25-01-2012')
```

Total execution time: 00:00:00.018

```
SELECT * FROM purchase_history
```

(11 rows affected)
Total execution time: 00:00:00.008

userid	productid	purchasedate
1	1	2012-01-23
1	2	2012-01-23
1	3	2012-01-25
2	1	2012-01-23
2	2	2012-01-23
2	2	2012-01-25
2	4	2012-01-25
3	4	2012-01-23
3	1	2012-01-23
4	1	2012-01-23
4	2	2012-01-25

```
SELECT userid, count(distinct purchasedate) as no_of_dates,
count(productid) as product_cnt,
count(distinct productid) as dist_prod_cnt
from purchase_history
group by userid
```

(4 rows affected)
Total execution time: 00:00:00.024

userid	no_of_dates	product_cnt	dist_prod_cnt
1	2	3	3
2	2	4	3
3	1	2	2
4	2	2	2

```
with cte as(
SELECT userid, count(distinct purchasedate) as no_of_dates,
count(productid) as product_cnt,
count(distinct productid) as dist_prod_cnt
from purchase_history
group by userid
)
select userid
from cte
where no_of_dates>1 and product_cnt = dist_prod_cnt
```

(2 rows affected)
Total execution time: 00:00:00.013

userid
1
4

```
--2nd approach without cte
SELECT userid,
count(distinct purchasedate) as no_of_dates,
count(productid) as product_cnt,
count(distinct productid) as dist_prod_cnt
from purchase_history
group by userid
having count(distinct purchasedate) >1 and count(productid) = count(distinct pr
```

(2 rows affected)
Total execution time: 00:00:00.017

userid	no_of_dates	product_cnt	dist_prod_cnt
1	2	3	3
4	2	2	2

Q 38

/*

Success of a Marketing Campaign. WE HAVE A TABLE OF IN APP PURCHASES BY USERS USERS THAT MAKE THERE FIRST IN APP PURCHASE ARE PLACED IN

A MARKETING CAMPEIGN WHERE THEY CAN SEE TO ACTIONS FOR MORE IN APP PURCHASES

FIND THE NUMBER OF USERS THAT MADE ADDITIONAL IN APP PURCHAGES DUE TO SUCESS OF MARKETING CAMPEIGN

THE MARKETING CAMPEIGN DOEST START UNTIL ONE DAY AFTER THE INITIAL IN APP PURCHASE SO USER THAT ONLY MADE ONE OR MULTIPLE PURCHASES ON 1ST DAY ARE NOT COUNTED NOR DO WE COUNT USERS

THET OVER TIME PURCHASE ONLY THE PRODUCT THEY PURCHAED ON FIRST DAY

*/

```
CREATE TABLE [marketing_campaign](
 [user_id] [int] NULL,
 [created_at] [date] NULL,
 [product_id] [int] NULL,
 [quantity] [int] NULL,
 [price] [int] NULL
);
insert into marketing_campaign values (10,'2019-01-01',101,3,55),
(10,'2019-01-02',119,5,29),
```

```
(10,'2019-03-31',111,2,149),
(11,'2019-01-02',105,3,234),
(11,'2019-03-31',120,3,99),
(12,'2019-01-02',112,2,200),
(12,'2019-03-31',110,2,299),
(13,'2019-01-05',113,1,67),
(13,'2019-03-31',118,3,35),
(14,'2019-01-06',109,5,199),
(14,'2019-01-06',107,2,27),
(14,'2019-03-31',112,3,200),
(15,'2019-01-08',105,4,234),
(15,'2019-01-09',110,4,299),
(15,'2019-03-31',116,2,499),
(16,'2019-01-10',113,2,67),
(16,'2019-03-31',107,4,27),
(17,'2019-01-11',116,2,499),
(17,'2019-03-31',104,1,154),
(18,'2019-01-12',114,2,248),
(18,'2019-01-12',113,4,67),
(19,'2019-01-12',114,3,248),
(20,'2019-01-15',117,2,999),
(21,'2019-01-16',105,3,234),
(21,'2019-01-17',114,4,248),
(22,'2019-01-18',113,3,67),
(22,'2019-01-19',118,4,35),
(23,'2019-01-20',119,3,29),
(24,'2019-01-21',114,2,248),
(25,'2019-01-22',114,2,248),
(25,'2019-01-22',115,2,72),
(25,'2019-01-24',114,5,248),
(25,'2019-01-27',115,1,72),
(26,'2019-01-25',115,1,72),
(27,'2019-01-26',104,3,154),
(28,'2019-01-27',101,4,55),
(29,'2019-01-27',111,3,149),
(30,'2019-01-29',111,1,149),
(31,'2019-01-30',104,3,154),
(32,'2019-01-31',117,1,999),
(33,'2019-01-31',117,2,999),
(34,'2019-01-31',110,3,299),
(35,'2019-02-03',117,2,999),
(36,'2019-02-04',102,4,82),
(37,'2019-02-05',102,2,82),
(38,'2019-02-06',113,2,67),
(39,'2019-02-07',120,5,99),
(40,'2019-02-08',115,2,72),
(41,'2019-02-08',114,1,248),
(42,'2019-02-10',105,5,234),
```

```
(43,'2019-02-11',102,1,82),
(43,'2019-03-05',104,3,154),
(44,'2019-02-12',105,3,234),
(44,'2019-03-05',102,4,82),
(45,'2019-02-13',119,5,29),
(45,'2019-03-05',105,3,234),
(46,'2019-02-14',102,4,82),
(46,'2019-02-14',102,5,29),
(46,'2019-03-09',102,2,35),
(46,'2019-03-10',103,1,199),
(46,'2019-03-11',103,1,199),
(47,'2019-02-14',110,2,299),
(47,'2019-03-11',105,5,234),
(48,'2019-02-14',115,4,72),
(48,'2019-03-12',105,3,234),
(49,'2019-02-18',106,2,123),
(49,'2019-02-18',114,1,248),
(49,'2019-02-18',112,4,200),
(49,'2019-02-18',116,1,499),
(50,'2019-02-20',118,4,35),
(50,'2019-02-21',118,4,29),
(50,'2019-03-13',118,5,299),
(50,'2019-03-14',118,2,199),
(51,'2019-02-21',120,2,99),
(51,'2019-03-13',108,4,120),
(52,'2019-02-23',117,2,999),
(52,'2019-03-18',112,5,200),
(53,'2019-02-24',120,4,99),
(53,'2019-03-19',105,5,234),
(54,'2019-02-25',119,4,29),
(54,'2019-03-20',110,1,299),
(55,'2019-02-26',117,2,999),
(55,'2019-03-20',117,5,999),
(56,'2019-02-27',115,2,72),
(56,'2019-03-20',116,2,499),
(57,'2019-02-28',105,4,234),
(57,'2019-02-28',106,1,123),
(57,'2019-03-20',108,1,120),
(57,'2019-03-20',103,1,79),
(58,'2019-02-28',104,1,154),
(58,'2019-03-01',101,3,55),
(58,'2019-03-02',119,2,29),
(58,'2019-03-25',102,2,82),
(59,'2019-03-04',117,4,999),
(60,'2019-03-05',114,3,248),
(61,'2019-03-26',120,2,99),
(62,'2019-03-27',106,1,123),
(63,'2019-03-27',120,5,99),
```

```
(64,'2019-03-27',105,3,234),
(65,'2019-03-27',103,4,79),
(66,'2019-03-31',107,2,27),
(67,'2019-03-31',102,5,82)
```

Total execution time: 00:00:00.071

```
Select * from marketing_campaign
```

(102 rows affected)
Total execution time: 00:00:00.014

user_id	created_at	product_id	quantity	price
10	2019-01-01	101	3	55
10	2019-01-02	119	5	29
10	2019-03-31	111	2	149
11	2019-01-02	105	3	234
11	2019-03-31	120	3	99
12	2019-01-02	112	2	200
12	2019-03-31	110	2	299
13	2019-01-05	113	1	67
13	2019-03-31	118	3	35
14	2019-01-06	109	5	199
14	2019-01-06	107	2	27
14	2019-03-31	112	3	200
15	2019-01-08	105	4	234
15	2019-01-09	110	4	299
15	2019-03-31	116	2	499
16	2019-01-10	113	2	67
16	2019-03-31	107	4	27
17	2019-01-11	116	2	499
17	2019-03-31	104	1	154
18	2019-01-12	114	2	248
18	2019-01-12	113	4	67
19	2019-01-12	114	3	248
20	2019-01-15	117	2	999
21	2019-01-16	105	3	234
21	2019-01-17	114	4	248
22	2019-01-18	113	3	67
22	2019-01-19	118	4	35
23	2019-01-20	119	3	29
24	2019-01-21	114	2	248
25	2019-01-22	114	2	248
25	2019-01-22	115	2	72
25	2019-01-24	114	5	248

25	2019-01-24 114	5	248
25	2019-01-27 115	1	72
26	2019-01-25 115	1	72
27	2019-01-26 104	3	154
28	2019-01-27 101	4	55
29	2019-01-27 111	3	149
30	2019-01-29 111	1	149
31	2019-01-30 104	3	154
32	2019-01-31 117	1	999
33	2019-01-31 117	2	999
34	2019-01-31 110	3	299
35	2019-02-03 117	2	999
36	2019-02-04 102	4	82
37	2019-02-05 102	2	82
38	2019-02-06 113	2	67
39	2019-02-07 120	5	99
40	2019-02-08 115	2	72
41	2019-02-08 114	1	248
42	2019-02-10 105	5	234
43	2019-02-11 102	1	82
43	2019-03-05 104	3	154
44	2019-02-12 105	3	234
44	2019-03-05 102	4	82
45	2019-02-13 119	5	29
45	2019-03-05 105	3	234
46	2019-02-14 102	4	82
46	2019-02-14 102	5	29
46	2019-03-09 102	2	35
46	2019-03-10 103	1	199
46	2019-03-11 103	1	199
47	2019-02-14 110	2	299
47	2019-03-11 105	5	234
48	2019-02-14 115	4	72
48	2019-03-12 105	3	234
49	2019-02-18 106	2	123
49	2019-02-18 114	1	248
49	2019-02-18 112	4	200
49	2019-02-18 116	1	499
50	2019-02-20 118	4	35
50	2019-02-21 118	4	29
50	2019-03-13 118	5	299
50	2019-03-14 118	2	199
51	2019-02-21 120	2	99
51	2019-03-13 108	4	120

52	2019-02-23 117	2	999
52	2019-03-18 112	5	200
53	2019-02-24 120	4	99
53	2019-03-19 105	5	234
54	2019-02-25 119	4	29
54	2019-03-20 110	1	299
55	2019-02-26 117	2	999
55	2019-03-20 117	5	999
56	2019-02-27 115	2	72
56	2019-03-20 116	2	499
57	2019-02-28 105	4	234
57	2019-02-28 106	1	123
57	2019-03-20 108	1	120
57	2019-03-20 103	1	79
58	2019-02-28 104	1	154
58	2019-03-01 101	3	55
58	2019-03-02 119	2	29
58	2019-03-25 102	2	82
59	2019-03-04 117	4	999
60	2019-03-05 114	3	248
61	2019-03-26 120	2	99
62	2019-03-27 106	1	123
63	2019-03-27 120	5	99
64	2019-03-27 105	3	234
65	2019-03-27 103	4	79
66	2019-03-31 107	2	27
67	2019-03-31 102	5	82

```
with rnk_data as(
select *
,rank() over (partition by user_id order by created_at asc) as rnk
from marketing_campaign
where user_id in (11,14,25) )       -----11,14,25 is user id sample case just t

,first_app_purchases as(
    select * from rnk_data
    where rnk=1
)
,except_first_app_purchases as(
    select * from rnk_data
    where rnk>1
)
--SELECT * FROM first_app_purchases
SELECT * FROM except_first_app_purchases
```

(4 rows affected)
Total execution time: 00:00:00.008

user_id	created_at	product_id	quantity	price	rnk
11	2019-03-31	120	3	99	2
14	2019-03-31	112	3	200	3
25	2019-01-24	114	5	248	3
25	2019-01-27	115	1	72	4

```
with rnk_data as(
select *
,rank() over (partition by user_id order by created_at asc) as rnk
from marketing_campaign
where user_id in (11,14,25) )       -----11,14,25 is user id sample case just f

,first_app_purchases as(
    select * from rnk_data
    where rnk=1
)
,except_first_app_purchases as(
    select * from rnk_data
    where rnk>1
)

select
a.user_id,a.product_id,b.user_id,b.product_id,a.created_at,b.created_at
from except_first_app_purchases A
left join first_app_purchases B on a.user_id = b.user_id and a.product_id = b.p
where b.product_id is NULL
```

(2 rows affected)
Total execution time: 00:00:00.023

user_id	product_id	user_id	product_id	created_at	created_at
11	120	NULL	NULL	2019-03-31	NULL
14	112	NULL	NULL	2019-03-31	NULL

Q 39

/* write a sql to find all the couples of trade for same stock that happened in range of 10s and having price difference by more than 10%

output result also list the percentage of price difference between 2 trade */

```
Create Table Trade_tbl(
TRADE_ID varchar(20),
Trade_Timestamp time,
Trade_Stock varchar(20),
Quantity int,
Price Float
)

Insert into Trade_tbl Values('TRADE1','10:01:05','ITJunction4All',100,20)
Insert into Trade_tbl Values('TRADE2','10:01:06','ITJunction4All',20,15)
Insert into Trade_tbl Values('TRADE3','10:01:08','ITJunction4All',150,30)
Insert into Trade_tbl Values('TRADE4','10:01:09','ITJunction4All',300,32)
Insert into Trade_tbl Values('TRADE5','10:10:00','ITJunction4All',-100,19)
Insert into Trade_tbl Values('TRADE6','10:10:01','ITJunction4All',-300,19)
```

Total execution time: 00:00:00.016

```
select * from Trade_tbl
```

(6 rows affected)
Total execution time: 00:00:00.011

TRADE_ID	Trade_Timestamp	Trade_Stock	Quantity	Price
TRADE1	10:01:05	ITJunction4All	100	20
TRADE2	10:01:06	ITJunction4All	20	15
TRADE3	10:01:08	ITJunction4All	150	30
TRADE4	10:01:09	ITJunction4All	300	32
TRADE5	10:10:00	ITJunction4All	-100	19
TRADE6	10:10:01	ITJunction4All	-300	19

```
select t1.TRADE_ID, t2.TRADE_ID
from Trade_tbl t1
inner join Trade_tbl t2 on 1=1
where t1.TRADE_ID!= t2.TRADE_ID and t1.Trade_Timestamp<t2.Trade_Timestamp
order by t1.TRADE_ID
```

(15 rows affected)
Total execution time: 00:00:00.023

TRADE_ID	TRADE_ID
TRADE1	TRADE2
TRADE1	TRADE3
TRADE1	TRADE4
TRADE1	TRADE5
TRADE1	TRADE6
TRADE2	TRADE3
TRADE2	TRADE4
TRADE2	TRADE5
TRADE2	TRADE6
TRADE3	TRADE4
TRADE3	TRADE5
TRADE3	TRADE6
TRADE4	TRADE5
TRADE4	TRADE6
TRADE5	TRADE6

```
select t1.TRADE_ID, t2.TRADE_ID,(t1.Price-t2.Price)*1.0/t1.Price*100
from Trade_tbl t1
inner join Trade_tbl t2 on 1=1
where t1.Trade_Timestamp<t2.Trade_Timestamp and DATEDIFF(second,t1.Trade_Timest
order by t1.TRADE_ID
```

(7 rows affected)
Total execution time: 00:00:00.013

TRADE_ID	TRADE_ID	(No column name)
TRADE1	TRADE2	25
TRADE1	TRADE3	-50
TRADE1	TRADE4	-60
TRADE2	TRADE3	-100
TRADE2	TRADE4	-113.33333333333333
TRADE3	TRADE4	-6.666666666666667
TRADE5	TRADE6	0

Q 40

/* we have a table which stores data of multiple sections every section has 3 number

we have to find top 4 number from 2 sections (2 numbers each) whos addition should be maximum

so in this case we will chosse section b where we have 19 (10+9), then we need to choose either c or d becuse both has sum of 18

but in D we have 10 which is big then 9 so we will give priority to D

*/

```
create table section_data
(
section varchar(5),
number integer
)
insert into section_data
values ('A',5),('A',7),('A',10) ,('B',7),('B',9),('B',10) ,('C',9),('C',7),('C'
```

Total execution time: 00:00:00.020

```
select * from section_data
```

(12 rows affected)
Total execution time: 00:00:00.008

section	number
A	5
A	7
A	10
B	7
B	9
B	10
C	9
C	7
C	9
D	10
D	3
D	8

```
with cte as(
    select *,
    ROW_NUMBER() over(partition by section order by number desc) as rn
    from section_data
)

SELECT * FROM cte
```

(12 rows affected)
Total execution time: 00:00:00.020

section	number	rn
A	10	1
A	7	2
A	5	3
B	10	1
B	9	2
B	7	3
C	9	1
C	9	2
C	7	3
D	10	1
D	8	2
D	3	3

```
with cte as(
    select *,
    ROW_NUMBER() over(partition by section order by number desc) as rn
    from section_data
)
, cte2 as(select *
,sum(number) over(PARTITION by section) as sum_total
, max(number) over(partition by section) as maxim
from cte
where rn<=2)

SELECT * FROM cte2
```

(8 rows affected)
Total execution time: 00:00:00.020

section	number	rn	sum_total	maxim
A	10	1	17	10
A	7	2	17	10
B	10	1	19	10
B	9	2	19	10
C	9	1	18	9
C	9	2	18	9
D	10	1	18	10
D	8	2	18	10

```
with cte as(
    select *,
    ROW_NUMBER() over(partition by section order by number desc) as rn
    from section_data
)
, cte2 as(select *
,sum(number) over(PARTITION by section) as sum_total
, max(number) over(partition by section) as maxim
from cte
where rn<=2)

select * from (
    select *,
    DENSE_RANK() over(order by sum_total desc ,maxim desc ) as rnk
    from cte2
) A where rnk<=2
```

(4 rows affected)
Total execution time: 00:00:00.017

section	number	rn	sum_total	maxim	rnk
B	10	1	19	10	1
B	9	2	19	10	1
D	10	1	18	10	2
D	8	2	18	10	2

Q 41

/* T1= BOOKING TABLE , T2= USER_TABLE

WRITE A SQL TO GIVE SUMMARY AT SEGMENT LEVLE

O/P: SEGMENT, TOTAL_USER_COUNT,USER_WHO_BOOKED_FLIGHT_IN APRIL2022

*/

```
CREATE TABLE booking_table(
   Booking_id       VARCHAR(3) NOT NULL
  ,Booking_date     date NOT NULL
  ,User_id          VARCHAR(2) NOT NULL
  ,Line_of_business VARCHAR(6) NOT NULL
);
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
INSERT INTO booking_table(Booking_id,Booking_date,User_id,Line_of_business) VAL
;
```

```
Total execution time: 00:00:00.025
```

```
CREATE TABLE user_table(
   User_id VARCHAR(3) NOT NULL
  ,Segment VARCHAR(2) NOT NULL
);
INSERT INTO user_table(User_id,Segment) VALUES ('u1','s1');
INSERT INTO user_table(User_id,Segment) VALUES ('u2','s1');
INSERT INTO user_table(User_id,Segment) VALUES ('u3','s1');
INSERT INTO user_table(User_id,Segment) VALUES ('u4','s2');
INSERT INTO user_table(User_id,Segment) VALUES ('u5','s2');
INSERT INTO user_table(User_id,Segment) VALUES ('u6','s3');
INSERT INTO user_table(User_id,Segment) VALUES ('u7','s3');
INSERT INTO user_table(User_id,Segment) VALUES ('u8','s3');
INSERT INTO user_table(User_id,Segment) VALUES ('u9','s3');
INSERT INTO user_table(User_id,Segment) VALUES ('u10','s3');
```

Total execution time: 00:00:00.016

```
select * from booking_table
select * from user_table
```

(20 rows affected)
(10 rows affected)
Total execution time: 00:00:00.027

Booking_id	Booking_date	User_id	Line_of_business
b1	2022-03-23	u1	Flight
b2	2022-03-27	u2	Flight
b3	2022-03-28	u1	Hotel
b4	2022-03-31	u4	Flight
b5	2022-04-02	u1	Hotel
b6	2022-04-02	u2	Flight
b7	2022-04-06	u5	Flight
b8	2022-04-06	u6	Hotel
b9	2022-04-06	u2	Flight
b10	2022-04-10	u1	Flight
b11	2022-04-12	u4	Flight
b12	2022-04-16	u1	Flight
b13	2022-04-19	u2	Flight
b14	2022-04-20	u5	Hotel
b15	2022-04-22	u6	Flight
b16	2022-04-26	u4	Hotel
b17	2022-04-28	u2	Hotel
b18	2022-04-30	u1	Hotel
b19	2022-05-04	u4	Hotel
b20	2022-05-06	u1	Flight

User_id	Segment
u1	s1
u2	s1
u3	s1
u4	s2
u5	s2
u6	s3
u7	s3
u8	s3
u9	s3
u10	s3

```
select u.*,b.*
from user_table u
LEFT join booking_table b on u.User_id=b.user_id
```

(25 rows affected)
Total execution time: 00:00:00.020

User_id	Segment	Booking_id	Booking_date	User_id	Line_of_business
u1	s1	b1	2022-03-23	u1	Flight
u1	s1	b3	2022-03-28	u1	Hotel
u1	s1	b5	2022-04-02	u1	Hotel
u1	s1	b10	2022-04-10	u1	Flight
u1	s1	b12	2022-04-16	u1	Flight
u1	s1	b18	2022-04-30	u1	Hotel
u1	s1	b20	2022-05-06	u1	Flight
u2	s1	b2	2022-03-27	u2	Flight
u2	s1	b6	2022-04-02	u2	Flight
u2	s1	b9	2022-04-06	u2	Flight
u2	s1	b13	2022-04-19	u2	Flight
u2	s1	b17	2022-04-28	u2	Hotel
u3	s1	NULL	NULL	NULL	NULL
u4	s2	b4	2022-03-31	u4	Flight
u4	s2	b11	2022-04-12	u4	Flight
u4	s2	b16	2022-04-26	u4	Hotel
u4	s2	b19	2022-05-04	u4	Hotel
u5	s2	b7	2022-04-06	u5	Flight
u5	s2	b14	2022-04-20	u5	Hotel
u6	s3	b8	2022-04-06	u6	Hotel
u6	s3	b15	2022-04-22	u6	Flight
u7	s3	NULL	NULL	NULL	NULL
u8	s3	NULL	NULL	NULL	NULL
u9	s3	NULL	NULL	NULL	NULL
u10	s3	NULL	NULL	NULL	NULL

```
 select u.Segment,count(distinct u.User_id) as no_of_users
from user_table u
LEFT join booking_table b on u.User_id=b.user_id
group by u.Segment
```

(3 rows affected)
Total execution time: 00:00:00.029

Segment	no_of_users
s1	3
s2	2
s3	5

```
select u.Segment,count(distinct u.User_id) as no_of_users
 ,count(distinct case when b.Line_of_business='Flight' and b.Booking_date betw
from user_table u
LEFT join booking_table b on u.User_id=b.user_id
group by u.Segment
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(3 rows affected)
Total execution time: 00:00:00.029

Segment	no_of_users	user_who_booked_flight
s1	3	2
s2	2	2
s3	5	1

```
--------------------------2) write a quy to identift user whose whos first book

select * from (
    select *,
    rank() over(partition by user_id order by Booking_date) as rnk
    from booking_table
) A
where rnk=1 and Line_of_business='Hotel'
```

(1 row affected)
Total execution time: 00:00:00.029

Booking_id	Booking_date	User_id	Line_of_business	rnk
b8	2022-04-06	u6	Hotel	1

```
---- or second approach is
select *,
    FIRST_VALUE(Line_of_business) over(partition by user_id order by Booking_da
    from booking_table
```

(20 rows affected)
Total execution time: 00:00:00.022

Booking_id	Booking_date	User_id	Line_of_business	rnk
b1	2022-03-23	u1	Flight	Flight
b3	2022-03-28	u1	Hotel	Flight
b5	2022-04-02	u1	Hotel	Flight
b10	2022-04-10	u1	Flight	Flight
b12	2022-04-16	u1	Flight	Flight
b18	2022-04-30	u1	Hotel	Flight
b20	2022-05-06	u1	Flight	Flight
b2	2022-03-27	u2	Flight	Flight
b6	2022-04-02	u2	Flight	Flight
b9	2022-04-06	u2	Flight	Flight
b13	2022-04-19	u2	Flight	Flight
b17	2022-04-28	u2	Hotel	Flight
b4	2022-03-31	u4	Flight	Flight
b11	2022-04-12	u4	Flight	Flight
b16	2022-04-26	u4	Hotel	Flight
b19	2022-05-04	u4	Hotel	Flight
b7	2022-04-06	u5	Flight	Flight
b14	2022-04-20	u5	Hotel	Flight
b8	2022-04-06	u6	Hotel	Hotel
b15	2022-04-22	u6	Flight	Hotel

```
select distinct USER_ID
from (
    select *,
    FIRST_VALUE(Line_of_business) over(partition by user_id order by Booking_da
    from booking_table
) A
where first_booking='Hotel'
```

(1 row affected)
Total execution time: 00:00:00.025

USER_ID
u6

```
---------3) write aquery to calculate find no of days between first and last bo

select user_id,min(booking_date),max(booking_date),DATEDIFF(day,max(booking_dat
from booking_table
group by User_id
```

(5 rows affected)
Total execution time: 00:00:00.008

user_id	(No column name)	(No column name)	(No column name)
u1	2022-03-23	2022-05-06	-44
u2	2022-03-27	2022-04-28	-32
u4	2022-03-31	2022-05-04	-34
u5	2022-04-06	2022-04-20	-14
u6	2022-04-06	2022-04-22	-16

```
----------4) -find no of flights and hotel booking in each of user segments for
select *
from booking_table t
inner join user_table b on t.User_id=b.User_id
--- user wise flight booking and hotel booking
select *
,case when line_of_business ='Flight' then 1 else 0 end as flag_flight
,case when line_of_business ='Hotel' then 1 else 0 end as flag_hotel
from booking_table
```

(20 rows affected)
(20 rows affected)
Total execution time: 00:00:00.020

Booking_id	Booking_date	User_id	Line_of_business	User_id	Segment
b1	2022-03-23	u1	Flight	u1	s1
b2	2022-03-27	u2	Flight	u2	s1
b3	2022-03-28	u1	Hotel	u1	s1
b4	2022-03-31	u4	Flight	u4	s2
b5	2022-04-02	u1	Hotel	u1	s1
b6	2022-04-02	u2	Flight	u2	s1
b7	2022-04-06	u5	Flight	u5	s2
b8	2022-04-06	u6	Hotel	u6	s3
b9	2022-04-06	u2	Flight	u2	s1
b10	2022-04-10	u1	Flight	u1	s1
b11	2022-04-12	u4	Flight	u4	s2
b12	2022-04-16	u1	Flight	u1	s1
b13	2022-04-19	u2	Flight	u2	s1
b14	2022-04-20	u5	Hotel	u5	s2

SQL_Complex1.ipynb - Colab 23/08/25, 2:42 PM

b15	2022-04-22	u6	Flight	u6	s3
b16	2022-04-26	u4	Hotel	u4	s2
b17	2022-04-28	u2	Hotel	u2	s1
b18	2022-04-30	u1	Hotel	u1	s1
b19	2022-05-04	u4	Hotel	u4	s2
b20	2022-05-06	u1	Flight	u1	s1

Booking_id	Booking_date	User_id	Line_of_business	flag_flight	flag_hotel
b1	2022-03-23	u1	Flight	1	0
b2	2022-03-27	u2	Flight	1	0
b3	2022-03-28	u1	Hotel	0	1
b4	2022-03-31	u4	Flight	1	0
b5	2022-04-02	u1	Hotel	0	1
b6	2022-04-02	u2	Flight	1	0
b7	2022-04-06	u5	Flight	1	0
b8	2022-04-06	u6	Hotel	0	1
b9	2022-04-06	u2	Flight	1	0
b10	2022-04-10	u1	Flight	1	0
b11	2022-04-12	u4	Flight	1	0
b12	2022-04-16	u1	Flight	1	0
b13	2022-04-19	u2	Flight	1	0
b14	2022-04-20	u5	Hotel	0	1
b15	2022-04-22	u6	Flight	1	0
b16	2022-04-26	u4	Hotel	0	1
b17	2022-04-28	u2	Hotel	0	1
b18	2022-04-30	u1	Hotel	0	1
b19	2022-05-04	u4	Hotel	0	1
b20	2022-05-06	u1	Flight	1	0

```
---
select user_id
,sum(case when line_of_business ='Flight' then 1 else 0 end) as flight_booking
,sum(case when line_of_business ='Hotel' then 1 else 0 end) as hotel_booking
from booking_table
group by User_id
```

(5 rows affected)
Total execution time: 00:00:00.009

user_id	flight_booking	hotel_booking
u1	4	3
u2	4	1
u4	2	2
u5	1	1
u6	1	1

```
-- but we want this segment wise so we have to join this

select segment
,sum(case when line_of_business ='Flight' then 1 else 0 end) as flight_booking
,sum(case when line_of_business ='Hotel' then 1 else 0 end) as hotel_booking
from booking_table B
inner join user_table u on b.User_id=u.User_id
group by Segment
```

(3 rows affected)
Total execution time: 00:00:00.011

segment	flight_booking	hotel_booking
s1	8	4
s2	3	3
s3	1	1

```
-- year wise filters in segments then
select segment
,sum(case when line_of_business ='Flight' then 1 else 0 end) as flight_booking
,sum(case when line_of_business ='Hotel' then 1 else 0 end) as hotel_booking
from booking_table B
inner join user_table u on b.User_id=u.User_id
where DATEPART(year,Booking_date) = 2022
group by Segment
```

(3 rows affected)
Total execution time: 00:00:00.019

segment	flight_booking	hotel_booking
s1	8	4
s2	3	3
s3	1	1

Q 42

/* MERGE OVERLAPPING EVENTS IN SMALL HALL

```
create table hall_events
(
hall_id integer,
start_date date,
end_date date
);
delete from hall_events
insert into hall_events values
(1,'2023-01-13','2023-01-14')
,(1,'2023-01-14','2023-01-17')
,(1,'2023-01-15','2023-01-17')
,(1,'2023-01-18','2023-01-25')
,(2,'2022-12-09','2022-12-23')
,(2,'2022-12-13','2022-12-17')
,(3,'2022-12-01','2023-01-30');
```

Total execution time: 00:00:00.022

```
SELECT * FROM hall_events;
```

(7 rows affected)
Total execution time: 00:00:00.013

hall_id	start_date	end_date
1	2023-01-13	2023-01-14
1	2023-01-14	2023-01-17
1	2023-01-15	2023-01-17
1	2023-01-18	2023-01-25
2	2022-12-09	2022-12-23
2	2022-12-13	2022-12-17
3	2022-12-01	2023-01-30

```
with cte as(
select *
,row_number()  over(order by hall_id, start_date) as event_id
from hall_events)

SELECT * FROM CTE
```

(7 rows affected)
Total execution time: 00:00:00.010

hall_id	start_date	end_date	event_id
1	2023-01-13	2023-01-14	1
1	2023-01-14	2023-01-17	2
1	2023-01-15	2023-01-17	3
1	2023-01-18	2023-01-25	4
2	2022-12-09	2022-12-23	5
2	2022-12-13	2022-12-17	6
3	2022-12-01	2023-01-30	7

```
with cte as(
select *
,row_number()  over(order by hall_id, start_date) as event_id
from hall_events)

, r_cte as(
select hall_id, start_date, end_date, event_id from cte
where event_id =1 )

SELECT * FROM r_cte
```

(1 row affected)
Total execution time: 00:00:00.010

hall_id	start_date	end_date	event_id
1	2023-01-13	2023-01-14	1

```
with cte as(
select *
,row_number()  over(order by hall_id, start_date) as event_id
from hall_events)

, r_cte as(
select hall_id, start_date, end_date, event_id from cte
where event_id =1
union ALL
select cte.hall_id, cte.start_date, cte.end_date, cte.event_id from r_cte
inner join cte on r_cte.event_id +1 = cte.event_id)

select * from r_cte
```

(7 rows affected)
Total execution time: 00:00:00.020

hall_id	start_date	end_date	event_id
1	2023-01-13	2023-01-14	1
1	2023-01-14	2023-01-17	2
1	2023-01-15	2023-01-17	3
1	2023-01-18	2023-01-25	4
2	2022-12-09	2022-12-23	5
2	2022-12-13	2022-12-17	6
3	2022-12-01	2023-01-30	7

```
----- case when condition with falg

with cte as(
select *
,row_number()  over(order by hall_id, start_date) as event_id
from hall_events)

, r_cte as(
select hall_id, start_date, end_date, event_id , 1 as flag from cte
where event_id =1

union ALL

select cte.hall_id, cte.start_date, cte.end_date, cte.event_id
, case when cte.hall_id =r_cte.hall_id  and ( cte.start_date between r_cte.star
                                              or r_cte.start_date between cte.s
       then 0 else 1 end +flag  as flag
from r_cte
inner join cte on r_cte.event_id +1 = cte.event_id)

--select * from r_cte

select hall_id,flag,min(start_date) as start_date ,max(start_date) as end_date
from r_cte
group by hall_id,flag
```

(4 rows affected)
Total execution time: 00:00:00.016

hall_id	flag	start_date	end_date
1	1	2023-01-13	2023-01-15
1	2	2023-01-18	2023-01-18
2	3	2022-12-09	2022-12-13
3	4	2022-12-01	2022-12-01

Q 43

/* WE NEED TO OBTAIN LIST OF DEPARTMENTS WITH AVERAGE SALARY LOWER THAN THE OVERALL AVERAGE SALARY OF COMPANY

HOWEVER WHEN CALCULATING COMANY AVERAGE SALARY YOU MUST EXCLUDE THE SALARIES OF DEPARTMENT YOU ARE COMPARING IT WITH

EG : IF YOU ARE CALCULATING AVERAGE SALAY OF HR , THEN IN AVG SALARY OF COMPANY CALCULATION DEPARTMENT HR SHOULD BE EXCLUDED

LIKEWISE IF YYOU WANT TO COMPARE AVERAGE SALARY OF FINANCE DEPARTMENT WITH COMPANY AVERAGE THEN COMPANY AVERAGE SHOULD NOT INCLUDE

SALARIES OF FINANCE DEPARTMENT

WRITE A DYNAMIC QUERY IN CASE NEW DEPARTMENT IS ADDED

*/

```
create table emp1(
emp_id int,
emp_name varchar(20),
department_id int,
salary int,
manager_id int,
emp_age int);

insert into emp1
values
(1, 'Ankit', 100,10000, 4, 39);
insert into emp1
values (2, 'Mohit', 100, 15000, 5, 48);
insert into emp1
values (3, 'Vikas', 100, 10000,4,37);
insert into emp1
values (4, 'Rohit', 100, 5000, 2, 16);
insert into emp1
values (5, 'Mudit', 200, 12000, 6,55);
insert into emp1
values (6, 'Agam', 200, 12000,2, 14);
insert into emp1
values (7, 'Sanjay', 200, 9000, 2,13);
insert into emp1
values (8, 'Ashish', 200,5000,2,12);
insert into emp1
values (9, 'Mukesh',300,6000,6,51);
insert into emp1
values (10, 'Rakesh',300,7000,6,50);
```

Total execution time: 00:00:00.010

```
SELECT * FROM emp1
```

(13 rows affected)
Total execution time: 00:00:00.008

emp_id	emp_name	department_id	salary	manager_id	emp_age
1	Ankit	100	10000	4	39
2	Mohit	100	15000	5	48
3	Vikas	100	10000	4	37
1	Ankit	100	10000	4	39
2	Mohit	100	15000	5	48
3	Vikas	100	10000	4	37
4	Rohit	100	5000	2	16
5	Mudit	200	12000	6	55
6	Agam	200	12000	2	14
7	Sanjay	200	9000	2	13
8	Ashish	200	5000	2	12
9	Mukesh	300	6000	6	51
10	Rakesh	300	7000	6	50

```
with avg_sal_dep as(
select department_id ,avg(salary) as dep_avg,count(*) as emp_count, sum(salary)
from emp1
group by department_id)

select *
from avg_sal_dep as a
INNER JOIN AVG_SAL_DEP AS B ON A.DEPARTMENT_ID!=B.DEPARTMENT_ID
```

(6 rows affected)
Total execution time: 00:00:00.019

department_id	dep_avg	emp_count	total_dep_sal	department_id	dep_avg	emp_count	total_de
100	10714	7	75000	200	9500	4	38000
100	10714	7	75000	300	6500	2	13000
200	9500	4	38000	100	10714	7	75000
200	9500	4	38000	300	6500	2	13000
300	6500	2	13000	100	10714	7	75000
300	6500	2	13000	200	9500	4	38000

```
with avg_sal_dep as(
select department_id ,avg(salary) as dep_avg,count(*) as emp_count, sum(salary)
from emp1
group by department_id)

select a.department_id,a.dep_avg,sum(b.emp_count) as noofemp,sum(b.total_dep_sa
,sum(b.total_dep_sal)/sum(b.emp_count) as company_avg_sal
from avg_sal_dep as a
INNER JOIN AVG_SAL_DEP AS B ON A.DEPARTMENT_ID!=B.DEPARTMENT_ID
group by a.department_id,a.dep_avg
order by a.department_id
-----------------
```

(3 rows affected)
Total execution time: 00:00:00.012

department_id	dep_avg	noofemp	total_salary	company_avg_sal
100	10714	6	51000	8500
200	9500	9	88000	9777
300	6500	11	113000	10272

```
--we have to find department which has salry less than compay avg

with avg_sal_dep as(
select department_id ,avg(salary) as dep_avg,count(*) as emp_count, sum(salary)
from emp1
group by department_id)

select department_id from
(select a.department_id,a.dep_avg,sum(b.emp_count) as noofemp,sum(b.total_dep_s
,sum(b.total_dep_sal)/sum(b.emp_count) as company_avg_sal
from avg_sal_dep as a
INNER JOIN AVG_SAL_DEP AS B ON A.DEPARTMENT_ID!=B.DEPARTMENT_ID
group by a.department_id,a.dep_avg
order by a.department_id) a
where dep_avg<company_avg_sal
```

Total execution time: 00:00:00.004

----------Q 54 ----------

---(SET OF 4 PROBLEMS)

/* We will solve it with 2 methods and also going to solve it with a twist.

T1= EMP CHEQ IN DEATILS, T2= EMP DETAILS

T1=(EMP_ID, ENTRY_DETAILS,TIMESTAMP_DETAILS)

T2=(EMP_ID, PHONENO , ISDEFAULT)

find -o/p- emp_id, total entry , total login , total log out , latest login , latest logout e,emp_detail phone no .

*/

```
CREATE TABLE [employee_checkin_details] (
    [employeeid]    INT,
    [entry_details] VARCHAR(512),
    [timestamp_details] DATETIME
);

INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
INSERT INTO [employee_checkin_details] ([employeeid], [entry_details], [timesta
```

Total execution time: 00:00:00.026

```
select * from employee_checkin_details
```

(8 rows affected)
Total execution time: 00:00:00.032

employeeid	entry_details	timestamp_details
1000	login	2023-06-16 01:00:15.340
1000	login	2023-06-16 02:00:15.340
1000	login	2023-06-16 03:00:15.340
1000	logout	2023-06-16 12:00:15.340
1001	login	2023-06-16 01:00:15.340
1001	login	2023-06-16 02:00:15.340
1001	login	2023-06-16 03:00:15.340
1001	logout	2023-06-16 12:00:15.340

```
CREATE TABLE [emp_det] (
    [employeeid]    INT,
    [phone_number]  INT,
    [isdefault] VARCHAR(512)
);

INSERT INTO [emp_det] ([employeeid], [phone_number], [isdefault]) VALUES ('1001
INSERT INTO [emp_det] ([employeeid], [phone_number], [isdefault]) VALUES ('1001
INSERT INTO [emp_det] ([employeeid], [phone_number], [isdefault]) VALUES ('1001
INSERT INTO [emp_det] ([employeeid], [phone_number], [isdefault]) VALUES ('1003
```

Total execution time: 00:00:00.030

```
select * from emp_det
```

(4 rows affected)
Total execution time: 00:00:00.028

employeeid	phone_number	isdefault
1001	9999	false
1001	1111	false
1001	2222	true
1003	3333	false

```
with loggin as(
select employeeid, count(*) as total_loggins,max(timestamp_details) as latestlo
from employee_checkin_details
where entry_details='login'
group by employeeid)

,logout as(
  select employeeid, count(*) as total_loggout,min(timestamp_details) as loggou
from employee_checkin_details
where entry_details='logout'
group by employeeid
)

select a.employeeid,a.total_loggins,a.latestloggin,b.employeeid,b.total_loggout
a.total_loggins +b.total_loggout as total_loggins,
from loggin as a
inner join logout b on a.employeeid=b.employeeid
```

Total execution time: 00:00:00.023

```
-- emp phome number is pending

with loggin as(
select employeeid, count(*) as total_loggins,max(timestamp_details) as latestlo
from employee_checkin_details
where entry_details='login'
group by employeeid)

,logout as(
  select employeeid, count(*) as total_loggout,min(timestamp_details) as loggou
from employee_checkin_details
where entry_details='logout'
group by employeeid
)

select a.employeeid,a.total_loggins,a.latestloggin,b.employeeid,b.total_loggout
a.total_loggins +b.total_loggout as total_loggins,
c.phone_number,c.isdefault
from loggin as a
inner join logout b on a.employeeid=b.employeeid
left join emp_det c on a.employeeid=c.employeeid and c.isdefault='True'
```

(2 rows affected)
Total execution time: 00:00:00.046

employeeid	total_loggins	latestloggin	employeeid	total_loggout	loggout	total_loggins	ph
1000	3	2023-06-16 03:00:15.340	1000	1	2023-06-16 12:00:15.340	4	NU

```
-----2nd approach

select a.employeeid , c.phone_number, count(*) as total_entry,
count(case when entry_details='login' then timestamp_details else null end ) as
,count(case when entry_details='logout' then timestamp_details else null end )
,max(case when entry_details='login' then timestamp_details else null end ) as
,max(case when entry_details='logout' then timestamp_details else null end ) as
from employee_checkin_details a
left join emp_det c on a.employeeid=c.employeeid and c.isdefault='True'
group by a.employeeid, c.phone_number
```

Warning: Null value is eliminated by an aggregate or other SET operation.
(2 rows affected)
Total execution time: 00:00:00.041

employeeid	phone_number	total_entry	totalloggins	totallogout	lateslogout	logintime
1000	NULL	4	3	1	2023-06-16 03:00:15.340	2023-06-16 12:00:15.340
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2023-06-16	2023-06-16