**SQL Assignment**

**In the yesterday session , I was informed that , everyone know SQL**

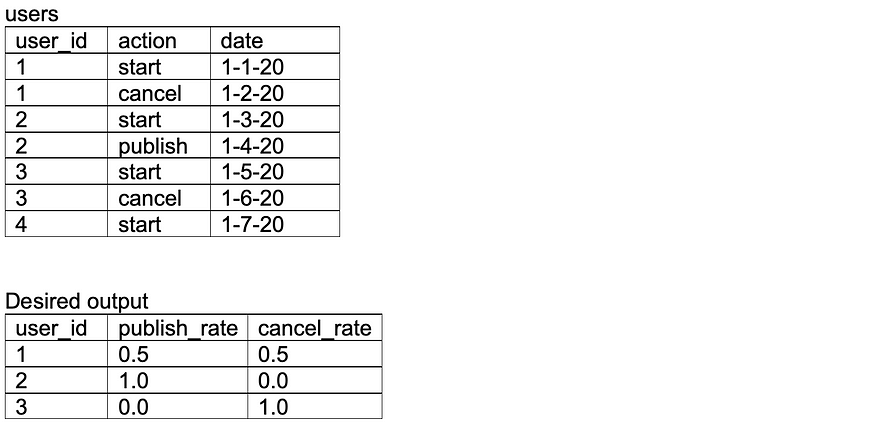
**This is the lab exercise. Send code by Friday morning Before 9 pm(30th Jun 2023)**

**Write SQL code and screen shot for the below**

**I don’t take class today . Just complete the code and screen shot**

**1. Cancellation rates**

From the following table of user IDs, actions, and dates, write a query to return the publication and cancellation rate for each user.



select user\_id, publish\_rate,cancel\_rate from (SELECT user\_id,

COUNT(CASE WHEN action = 'publish' THEN 1 END) AS publish\_rate,

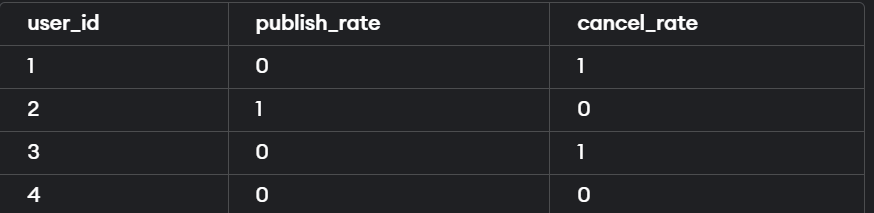
COUNT(CASE WHEN action = 'cancel' THEN 1 END) AS cancel\_rate,

COUNT(CASE WHEN action = 'publish' THEN 1 END) \* 100.0 / COUNT(\*) AS publish\_rate,

COUNT(CASE WHEN action = 'cancel' THEN 1 END) \* 100.0 / COUNT(\*) AS cancel\_rate

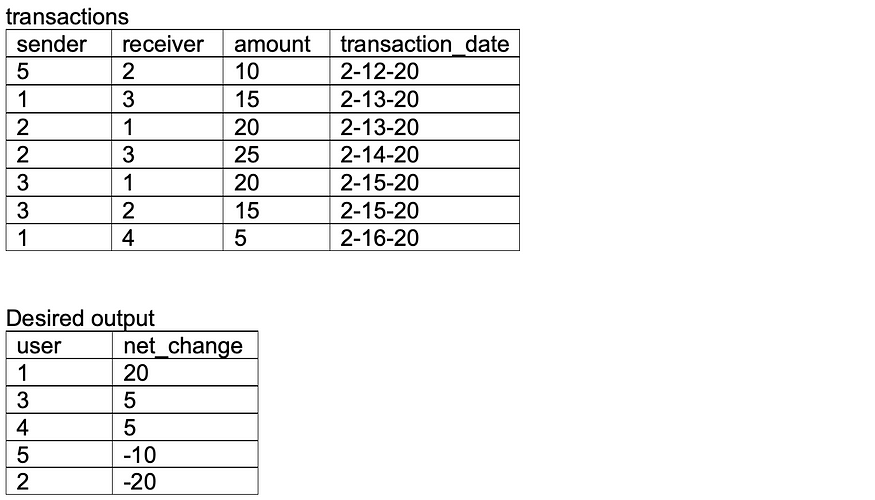
FROM users

GROUP BY user\_id);



## 2. Changes in net worth

From the following table of transactions between two users, write a query to return the change in net worth for each user, ordered by decreasing net change.



ANS:

CREATE TABLE transactions (

sender int ,

receive int,

amount int,

transaction\_date varchar(10)

);

INSERT into transactions VALUES( 5,2,10,'2-12-20');

INSERT into transactions VALUES( 1,3,15,'2-13-20');

INSERT into transactions VALUES( 2,1,20,'2-12-20');

INSERT into transactions VALUES( 2,3,25,'2-14-20');

INSERT into transactions VALUES( 3,1,20,'2-15-20');

INSERT into transactions VALUES( 3,2,15,'2-12-20');

INSERT into transactions VALUES( 1,4,5,'2-16-20');

SELECT \* from transactions;

SELECT t1.sender AS user1,

t1.receive AS user2,

SUM(t1.amount) - SUM(t2.amount) AS net\_amount

FROM transactions t1

JOIN transactions t2 ON t1.sender = t2.receive AND t1.receive = t2.sender

GROUP BY t1.sender, t1.receive;

SELECT user, SUM(b) - SUM(a) as net\_amount

FROM (

SELECT sender as user, SUM(amount) as a, 0 as b

FROM transactions

GROUP BY sender

UNION ALL

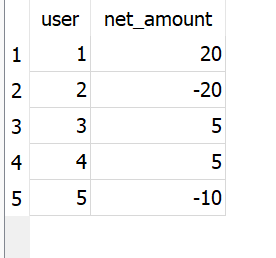
SELECT receive as user, 0 as a, SUM(amount) as b

FROM transactions

GROUP BY receive

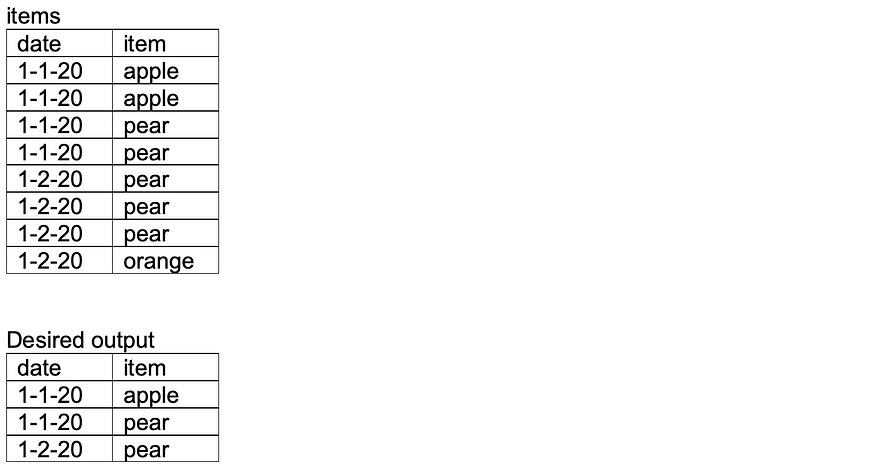
)

GROUP BY user;



## 3. Most frequent items

From the following table containing a list of dates and items ordered, write a query to return the most frequent item ordered on each date. Return multiple items in the case of a tie.



ANS:

CREATE TABLE items (

dates varchar(10), item varchar(10)

);

INSERT INTO items

VALUES ('1-1-20','apple');

INSERT INTO items

VALUES ('1-1-20','apple');

INSERT INTO items

VALUES ('1-1-20','pear');

INSERT INTO items

VALUES ('1-1-20','pear');

INSERT INTO items

VALUES ('1-2-20','pear');

INSERT INTO items

VALUES ('1-2-20','pear');

INSERT INTO items

VALUES ('1-2-20','orange');

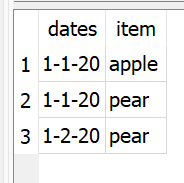
select \* from items;

select dates,item from (Select dates, item, RANK() over (partition by dates order by freq desc) as freq from

(SELECT dates, item, COUNT(\*) AS freq

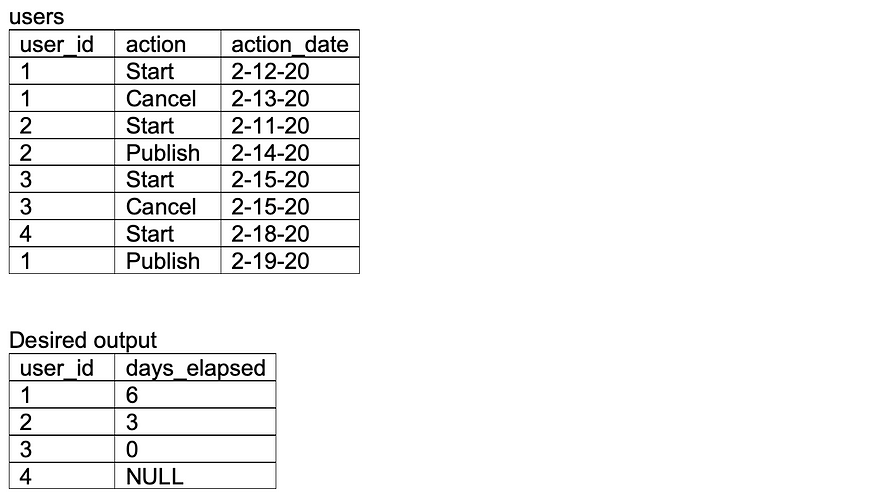
FROM items

GROUP BY dates, item)) where freq=1;



## 4. Time difference between latest actions

From the following table of user actions, write a query to return for each user the time elapsed between the last action and the second-to-last action, in ascending order by user ID.



ANS:

create table users (

user\_id int,

action varchar(10),

action\_date date);

insert into users VALUES(1,'start','2-12-20');

insert into users VALUES(1,'cancel','2-13-20');

insert into users VALUES(2,'start','2-11-20');

insert into users VALUES(2,'publish','2-14-20');

insert into users VALUES(3,'start','2-15-20');

insert into users VALUES(3,'cancel','2-15-20');

insert into users VALUES(4,'start','2-18-20');

insert into users VALUES(1,'publish','2-19-20');

SELECT \* from users;

SELECT DATEDIFF(last\_action\_date, second\_last\_action\_time) AS date\_elapsed from

(SELECT MAX(action\_date) AS second\_last\_action\_time

FROM users

WHERE action\_date < (

SELECT MAX(action\_date)

FROM users LIMIT 1)

))

GROUP BY user\_id) as a

join

(SELECT user\_id, MAX(action\_date) AS last\_action\_date

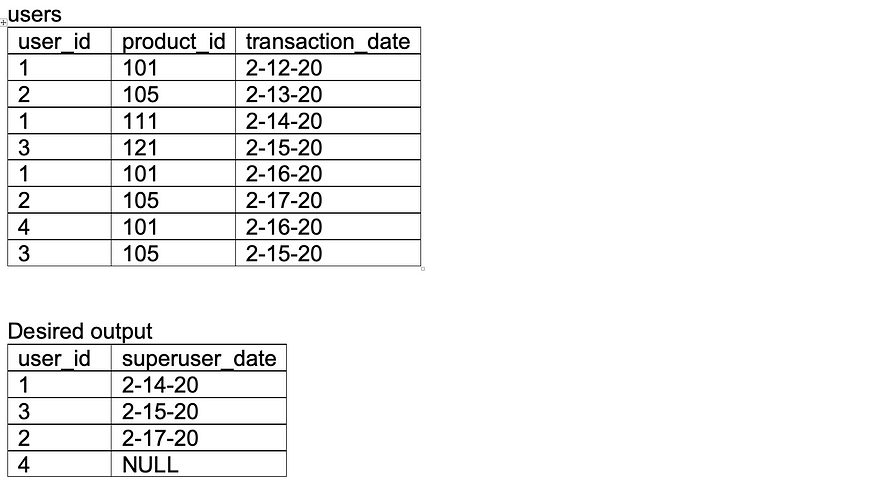
FROM users

GROUP BY user\_id) as b

on a.user\_id=b.user\_id;

## 5. Super users

A company defines its super users as those who have made at least two transactions. From the following table, write a query to return, for each user, the date when they become a super user, ordered by oldest super users first. Users who are not super users should also be present in the table.



create TABLE users( user\_id int, product\_id int, transaction\_date date);

insert into users values(1,101,'2-12-20');

insert into users values(2,105,'2-13-20');

insert into users values(1,111,'2-14-20');

insert into users values(3,121,'2-15-20');

insert into users values(1,101,'2-16-20');

insert into users values(2,105,'2-17-20');

insert into users values(4,101,'2-16-20');

insert into users values(3,105,'2-15-20');

select \* from users;

SELECT users.user\_id, MIN(temp.transaction\_date) AS SuperUserDate

FROM users

LEFT JOIN(SELECT user\_id,transaction\_date

FROM (SELECT user\_id, transaction\_date, ROW\_NUMBER() OVER (PARTITION BY user\_id ORDER BY transaction\_date) AS rownumber

FROM users) where rownumber=2) temp on users.user\_id=temp.user\_id group by users.user\_id order by superuserdate asc;

