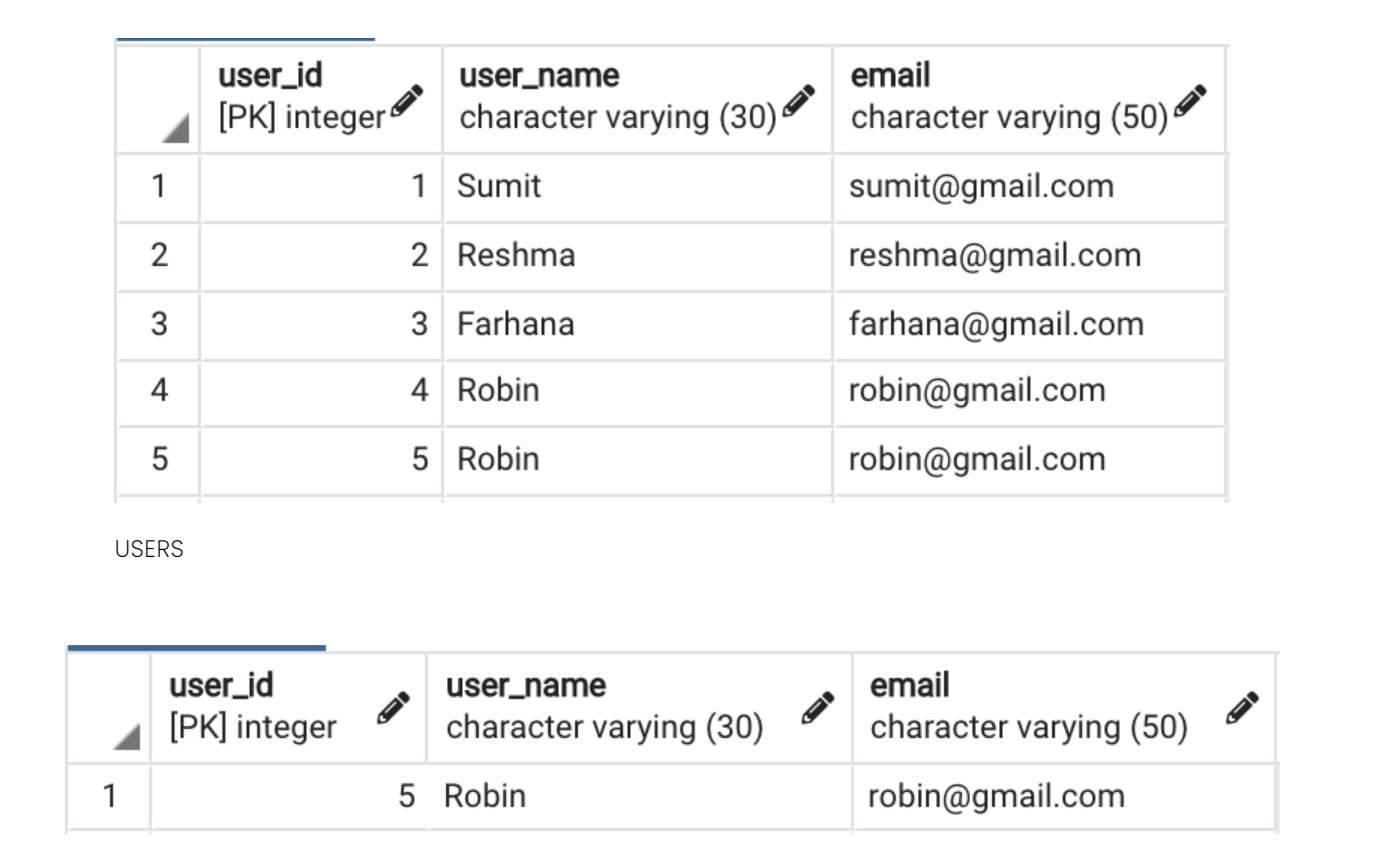
https://techtfq.com/blog/learn-how-to-write-sql-queries-practice-complex-sql-queries

1. **Write a SQL Query to fetch all the duplicate records in a table.**

**Table Name**: USERS

**Note**: Record is considered duplicate if a user name is present more than once.

**Approach**: Partition the data based on user name and then give a row number to each of the partitioned user name. If a user name exists more than once then it would have multiple row numbers. Using the row number which is other than 1, we can identify the duplicate records.

Select \* from (

Select \*,

row\_number() OVER(partition by user\_name order by user\_id) as rn

from users

order by userid) as x

where x.rn>1;

### 2. Write a SQL query to fetch the second last record from employee table.

**Table Name**: EMPLOYEE

**Approach**: Using window function sort the data in descending order based on employee id. Provide a row number to each of the record and fetch the record having row number as 2.

Select \* from (

Select \*, row\_number() over(order by emp\_id DESC) as rn

From employee e) as x

Where x.rn =2;

### 3. Write a SQL query to display only the details of employees who either earn the highest salary or the lowest salary in each department from the employee table.

**Table Name**: EMPLOYEE

**Approach**: Write a sub query which will partition the data based on each department and then identify the record with maximum and minimum salary for each of the partitioned department. Finally, from the main query fetch only the data which matches the maximum and minimum salary returned from the sub query.

~~Select \* from (~~

~~Select \*, row\_number() over(partition by dept\_name order by salary ASC) as rn~~

~~Lag() over(rn) as next~~

~~from employee) as x~~

~~where x.rn=1 or x.next=1;~~

select x.\*

from employee e

join (select \*,

max(salary) over (partition by dept\_name) as max\_salary,

min(salary) over (partition by dept\_name) as min\_salary

from employee) x

on e.emp\_id = x.emp\_id

and (e.salary = x.max\_salary or e.salary = x.min\_salary)

order by x.dept\_name, x.salary;

### 4. From the doctors table, fetch the details of doctors who work in the same hospital but in different specialty.

**Table Name**: DOCTORS

**Approach**: Use self join to solve this problem. Self join is when you join a table to itself.

**Additional Query**: Write SQL query to fetch the doctors who work in same hospital irrespective of their specialty.

Select \* from DOCTORS D1 JOIN DOCTORS D2 ON

D1.ID <> D2.ID AND D1.HOSPITAL = D2.HOSPITAL AND D1.SPECALITY <> D2.SPECALITY;

### 5. From the login\_details table, fetch the users who logged in consecutively 3 or more times.

**Table Name**: LOGIN\_DETAILS

**Approach**: We need to fetch users who have appeared 3 or more times consecutively in login details table. There is a window function which can be used to fetch data from the following record. Use that window function to compare the user name in current row with user name in the next row and in the row following the next row. If it matches then fetch those records.

~~SELECT USER\_NAME FROM (~~

~~SELECT \*, ROW\_NUMBER() OVER(PARTITION BY USER\_NAME ORDER BY LOGIN\_DATE ASC) AS RN FROM LOGIN\_DETAILS~~

~~) AS X WHERE X.RN>3;~~

SELECT DISTINCT USER\_NAME

FROM (

SELECT \*,

CASE WHEN USER\_NAME = LEAD(USER\_NAME) OVER(ORDER BY LOGIN\_ID)

AND USER\_NAME = LEAD(USER\_NAME, 2) OVER(ORDER BY LOGIN\_ID)

THEN USER\_NAME ELSE NULL

END AS REPEATED\_USERS

FROM LOGIN\_DETAILS) X

WHERE X.REPEATED\_USERS IS NOT NULL;

6. From the students table, write a SQL query to interchange the adjacent student names.

Note: If there are no adjacent student then the student name should stay the same.

--Table Structure:

drop table students;

create table students

(

id int primary key,

student\_name varchar(50) not null

);

insert into students values

(1, 'James'),

(2, 'Michael'),

(3, 'George'),

(4, 'Stewart'),

(5, 'Robin');

**SOLUTION:**

SELECT ID, STUDENT\_NAME

CASE WHEN ID%2 <> 0 THEN LEAD(STUDENT\_NAME, 1, STUDENT\_NAME) OVER(ORDER BY ID)

WHEN ID%2 == 0 THEN LAG(STUDENT\_NAME) OVER(ORDER BY ID) END AS NEW\_STUDENT\_NAME FROM STUDENTS;

### 7. From the weather table, fetch all the records when London had extremely cold temperature for 3 consecutive days or more.

**Note**: Weather is considered to be extremely cold when its temperature is less than zero.

**Table Name**: WEATHER

SELECT ID, CITY, TEMPERATURE, DAY

FROM ( SELECT \*,

CASE WHEN TEMPERATURE < 0

AND LEAD(TEMPERATURE) OVER(ORDER BY DAY)<0

AND LEAD(TEMPERATURE,2) OVER(ORDER BY DAY) < 0

THEN ‘Y’

WHEN TEMPERATURE < 0

AND LEAD(TEMPERATURE) OVER (ORDER BY DAY) < 0

AND LAG(TEMPERATURE) OVER(ORDER BY DAY) < 0

THEN ‘Y’

WHEN TEMPERATURE<0

AND LAG(TEMPERATURE) OVER(ORDER BY DAY) < 0

AND LAG(TEMPERATURE,2) OVER ( ORDER BY DAY) < 0

THEN ‘Y’

END AS FLAG

FROM WEATHER) X

WHERE X.FLAG = ‘Y’;