**DBMS - MINI PROJECT**

**“Leetcode like coding contest database managment”**

Submitted By:

**Manoj kumar H R**

**PES1UG20CS560**

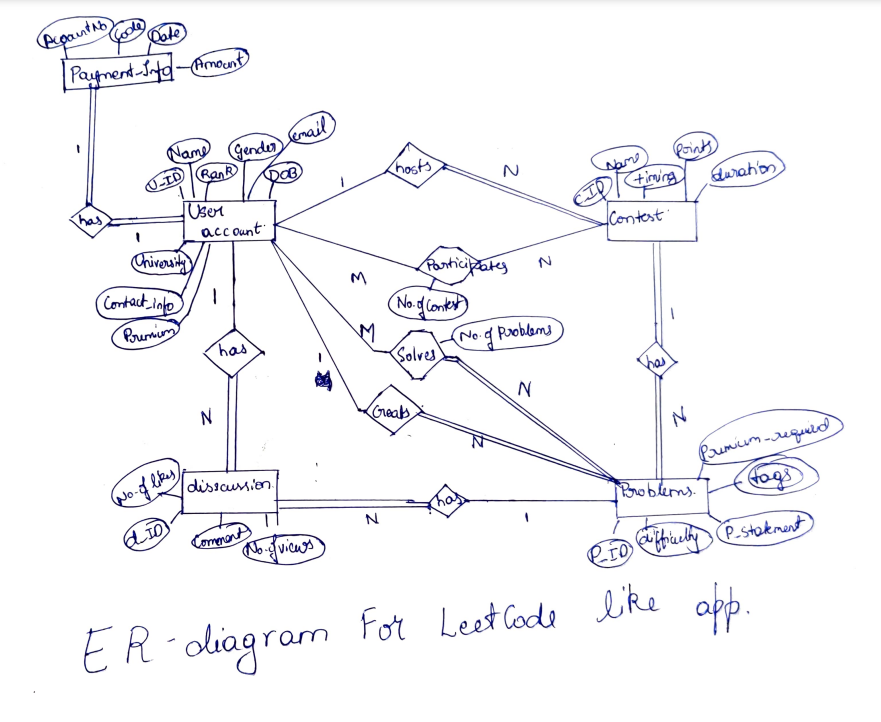
**V Semester**

**Section J**

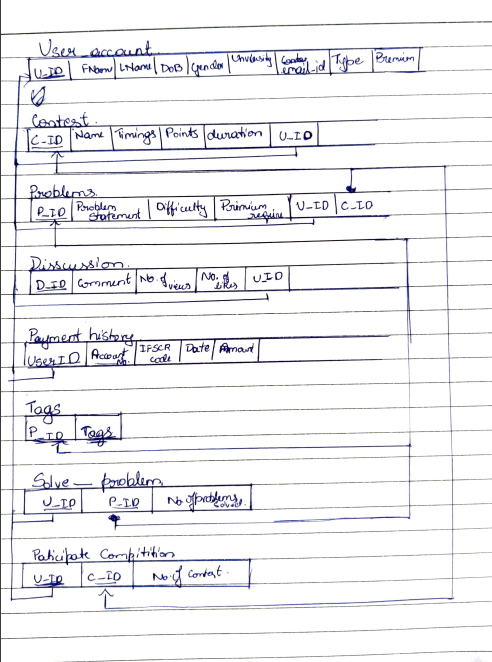
**ABSTRACT**

With the increase in number of engineering students every year, The challenge of getting better in coding is always there. Leetcode a very famous coding platform helps students to improve on their coding skills and with their regular contest, different levels of problems the data to manage is very hard, this is where relational database comes into picture. This project replicates how relational database system helps to store and retrieve the data for these kind of websites/platform.

**ER Diagram**



**Relational Schema**

****

**DDL statements - Building the database**

**Populating the Database**

create database code\_contest\_project\_560;

use code\_contest\_project\_560;

create table User(

    U\_ID int(16) not null,

    Fname varchar(25) not null,

    Lname varchar(25) ,

    DOB date,

    Gender varchar(10) ,

    University varchar(100),

    email\_ID varchar(50)  ,

    type\_P int default 0,

    Premium int default 0,

    constraint F\_M check (Gender='FEMALE' or Gender='MALE'),

    constraint E\_Check check(email\_ID like '%@%.com'),

    constraint T\_check check (type\_P=0 or type\_P=1),

    constraint P\_check check (Premium=0 or Premium=1),

    constraint Prim\_key\_User primary key (U\_ID)

);

create table Contest(

    C\_ID varchar(16) not null,

    C\_name varchar(100) not null,

    Timings date,

    points int(10),

    duration time,

    U\_ID int(16) not null,

    constraint Prim\_key\_Contest primary key (C\_ID),

    constraint For\_key\_Contest foreign key (U\_ID) references User(U\_ID)

);

create table Problems(

    P\_ID varchar(16) not null,

    P\_Statement MEDIUMTEXT not null,

    Difficulty varchar(10)  ,

    P\_require int  ,

    U\_ID int(16) not null,

    C\_ID varchar(16) not null,

    constraint D\_check check (Difficulty in ("easy","medium","hard","very hard")),

    constraint Req\_check check (P\_require=0 or P\_require=1),

    constraint Prim\_key\_Prob primary key (P\_ID),

    constraint For\_key\_Prob1 foreign key (U\_ID) references User(U\_ID),

    constraint For\_key\_Prob2 foreign key (C\_ID) references Contest(C\_ID)

);

create table Disscussion(

    D\_ID varchar(16) not null,

    comment MEDIUMTEXT not null,

    No\_of\_likes int(20),

    No\_of\_views int(100),

    U\_ID int(16) not null,

    P\_ID varchar(16) not null,

    constraint Prim\_key\_Diss primary key (D\_ID),

    constraint For\_key\_Diss1 foreign key (U\_ID) references User(U\_ID),

    constraint For\_key\_Diss2 foreign key (P\_ID) references Problems(P\_ID)

);

create table Payment\_hist(

    User\_ID int(16) not null,

    Account\_no varchar(50) not null,

     \_code varchar(25) not null,

    pay\_date date,

    amount int(20),

    constraint Prim\_key\_Pay primary key (User\_ID,Account\_no),

    constraint For\_key\_Pay foreign key (User\_ID) references User(U\_ID)

);

create table Tags (

    P\_ID varchar(16) not null,

    Tags varchar(50) not null,

    constraint Prim\_key\_Tags primary key (P\_ID,Tags),

    constraint For\_key\_Tags foreign key (P\_ID) references Problems(P\_ID)

);

create table Solve\_problem(

    U\_ID int(16) not null,

    P\_ID varchar(16) not null,

    No\_problems int(50),

    constraint Prim\_key\_Solve primary key (U\_ID,P\_ID,No\_problems),

    constraint For\_key\_Solve1 foreign key(U\_ID) references User(U\_ID),

    constraint For\_key\_Solve2 foreign key(P\_ID) references Problems(P\_ID)

);

create table Part\_Contest(

    U\_ID int(16) not null,

    C\_ID varchar(16) not null,

    No\_contest int(25),

    constraint Prim\_key\_Solve primary key (U\_ID,C\_ID,No\_contest),

    constraint For\_key\_ParCon1 foreign key(U\_ID) references User(U\_ID),

    constraint For\_key\_ParCon2 foreign key(C\_ID) references Contest(C\_ID)

);

INSERT INTO User VALUES

(1101,'a','A','2003-2-5','male','aaa','a@gmail.com',0,0),

(1102,'b','B','2002-3-12','female','bbb','b@gmail.com',0,0),

(1103,'c','C','2003-5-22','female','ccc','c@gmail.com',0,0),

(1104,'d','D','2002-1-3','male','ddd','d@gmail.com',0,1),

(1105,'e','E','2004-2-4','male','eee','e@gmail.com',0,1),

(1106,'f','F','2001-6-25','female','fff','f@gmail.com',0,1),

(1107,'g','G','2002-12-1','female','ggg','g@gmail.com',1,0),

(1108,'h','H','2003-12-1','female','hhh','h@gmail.com',1,0),

(1109,'i','I','2002-8-11','male','iii','i@gmail.com',1,0);

INSERT INTO Contest VALUES

('CT\_1','CT\_CODE','2020-8-3',100,'1:00:00',1107),

('CT\_2','CT\_KICKOFF','2020-8-10',100,'1:00:00',1107),

('CT\_3','CT\_DEV','2023-1-17',400,'3:00:00',1109),

('CT\_4','CT\_PROGRAM','2023-1-2',400,'3:00:00',1108);

INSERT INTO Problems VALUES

('PR\_1',

'Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.',

'Easy',0,1108,'CT\_1'),

('PR\_2',

'You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.',

'medium',0,1108,'CT\_1'),

('PR\_3',

'Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).',

'Hard',1,1109,'CT\_3');

INSERT INTO Disscussion VALUES

('DI\_1',

'It was asked in my interview',

5,60,1101,'PR\_1'),

('DI\_2',

'I am wondering why .sort() is used in most of the solutions when the time complexity requirement is O(log(m+n))?',

113,800,1104,'PR\_3'),

('DI\_3',

'The problem statement in the description states that we can assume there are no empty arrays

 but the test case inputs includes few examples with empty arrays.',

50,200,1106,'PR\_3'),

('DI\_4',

'Ihe problem is easy',

3,40,1104,'PR\_1'),

('DI\_5',

'There is a similar problem in easy mode',

150,600,1101,'PR\_2'),

('DI\_6',

'Noice Problem',

100,200,1103,'PR\_1');

INSERT INTO Payment\_hist VALUES

(1101,'1721 6454 2222','CNRB0000783','2020-3-14',500),

(1104,'1125 6251 6232','HDFC0000123','2021-1-2',500),

(1105,'5665 8522 5605','SBIF5214856','2022-11-17',500),

(1106,'7865 3641 9565','SBIF7961245','2022-12-21',500);

INSERT INTO Tags VALUES

('PR\_1','Array'),

('PR\_1','Dynamic programming'),

('PR\_1','Greedy'),

('PR\_2','Linked list'),

('PR\_3','Array'),

('PR\_3','Sorting'),

('PR\_3','Map');

INSERT INTO Solve\_problem VALUES

(1101,'PR\_1',2),

(1101,'PR\_3',2),

(1102,'PR\_3',1),

(1103,'PR\_1',1);

INSERT INTO Part\_Contest VALUES

(1101,'CT\_1',2),

(1103,'CT\_1',1),

(1101,'CT\_3',2),

(1102,'CT\_3',1),

(1104,'CT\_3',1),

(1106,'CT\_3',1);

**Tool Used**

Main tool used was the sql dump operator through which an entire database creation can be done with just the sql statements.  
example: mysql –u root < “file\_path”

**Queries**

**Join queries (at least 6)**

Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results.

Include 2 regular join, 2 co-related and 2 nested queries

Regular join:

1)

statement: display user detials solved atleast one problem

code: select fname,lname,dob,University,Gender from user join Solve\_problem as sp on (user.U\_ID=sp.U\_ID);

output:

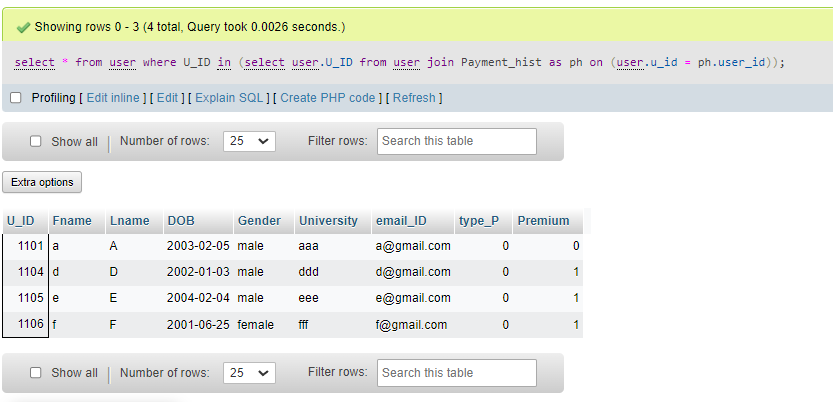


2)

display user detials who have purchased Premium atleast once

select \* from user where U\_ID in (select user.U\_ID from user join Payment\_hist as ph on (user.u\_id = ph.user\_id));

output:



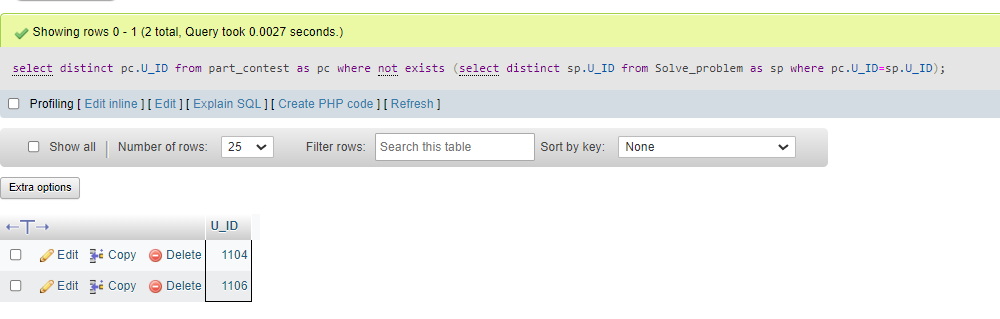
Correlated join.

1)

display user details who have joined contest but have not solved any question

select distinct pc.U\_ID from part\_contest as pc where not exists (select distinct sp.U\_ID from Solve\_problem as sp where pc.U\_ID=sp.U\_ID);

output:



2)

display user ID who have not solved any question but have a comment in disscussion

select distinct d.U\_ID from disscussion as d where not exists (select sp.u\_id from Solve\_problem as sp where d.U\_ID=sp.U\_ID and d.P\_ID = sp.P\_ID);

output:



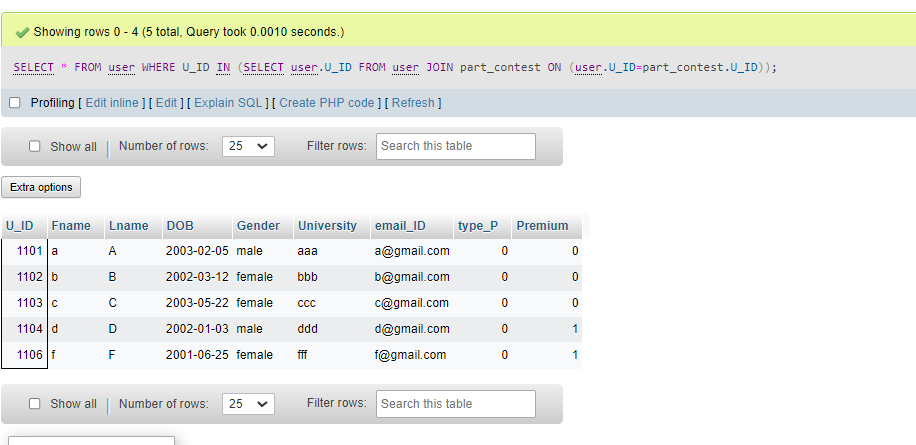
Nested Join:

1)

display user detials who have participated in any contest (except is a synonym for minus)

SELECT \* FROM user WHERE U\_ID IN (SELECT user.U\_ID FROM user JOIN part\_contest ON (user.U\_ID=part\_contest.U\_ID));

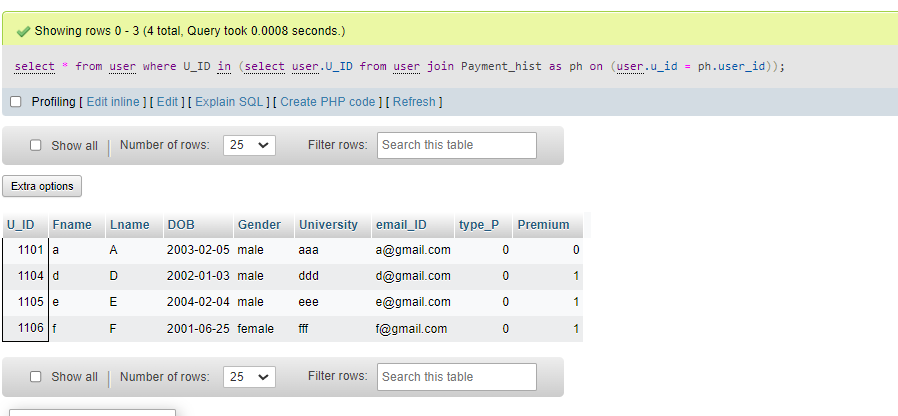
output:



2)

display user detials who have purchased Permium atleast once

select \* from user where U\_ID in (select user.U\_ID from user join Payment\_hist as ph on (user.u\_id = ph.user\_id));

output:

**Aggregate Functions (at least 2)**

Showcase at least 2 Aggregate function queries. Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results

1)

display problem ID of problems having number of tags more than 1

select P\_ID from Tags group by P\_ID having count(P\_ID)>1;

output:

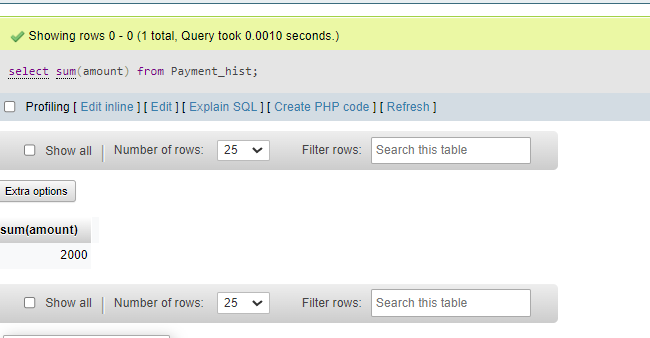


2)

display the total amount collected through premium buys

select sum(amount) from Payment\_hist;

Output:



**Set Operations (at least 2)**

Showcase at least 2 Set Operations queries . Write the query in English Language, Show the equivalent SQL statement and also screenshot of the query and the results

**1)**

display U\_ID of users not participating in any contest

select U\_ID from user except select U\_ID from Part\_Contest;

**output:**



**2)**

display U\_ID who have solved problem and have disscussion

select U\_ID from disscussion intersect select U\_ID from Solve\_problem;

**output:**



**View (atleast 1)**

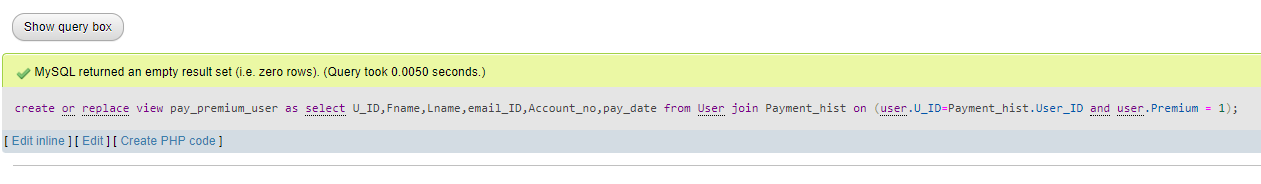
Demonstrate creation and querying one view

Creation

create or replace view pay\_premium\_user as

select U\_ID,Fname,Lname,email\_ID,Account\_no,pay\_date from User

join Payment\_hist on (user.U\_ID=Payment\_hist.User\_ID and user.Premium = 1);



Query:

update user set Premium=0 where U\_ID in (select remove\_premium(pay\_date,U\_ID) from pay\_premium\_user);

The results are shown in triggers and function.

**Triggers (Functions or Procedures)**

Create a Function or a Procedure. State the objective of the function / Procedure. Run and

display the results.

Trigger:

delimiter $$

create trigger valid\_contest\_creation

before insert

on contest for each row

begin

    declare ad\_ int;

    declare err\_message varchar(100);

    set err\_message = "ERROR: You don't have admin privilages to add contest";

    select type\_P into ad\_ from User where U\_ID=new.U\_ID;

    if ad\_=0 then

        signal sqlstate '45000'

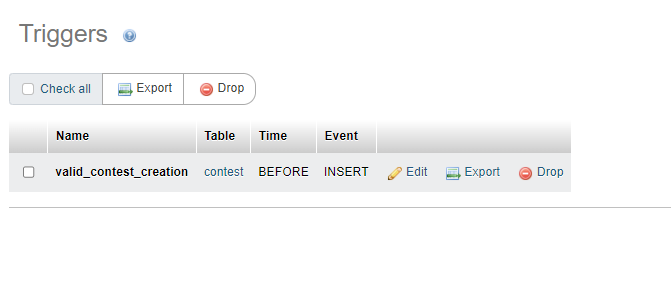
        set message\_text = err\_message;

    end if;

end;

$$

delimiter ;



Function:

delimiter ;

create or replace view pay\_premium\_user as

select U\_ID,Fname,Lname,email\_ID,Account\_no,pay\_date from User

join Payment\_hist on (user.U\_ID=Payment\_hist.User\_ID and user.Premium = 1);

delimiter $$

create or replace function remove\_premium(apply\_date date,id int)

    returns varchar(255)

    deterministic

    begin

        declare x int;

        declare age int;

        set age = DATEDIFF(now(),apply\_date);

        if (age > 365) then

            set x = id;

        end if;

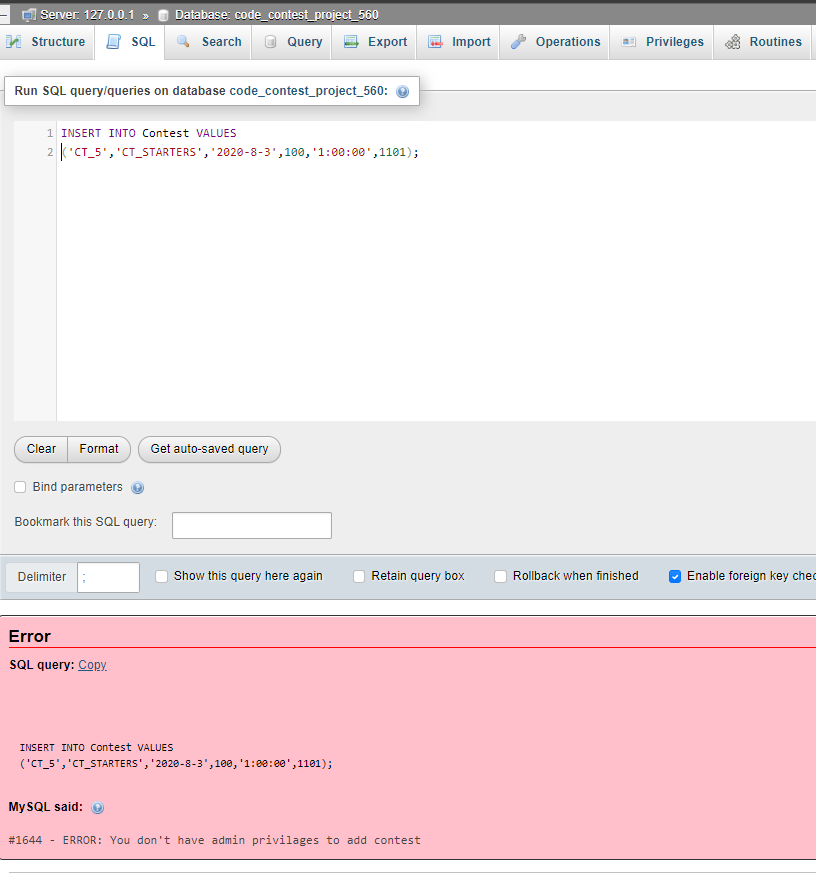
        return x;

    end;

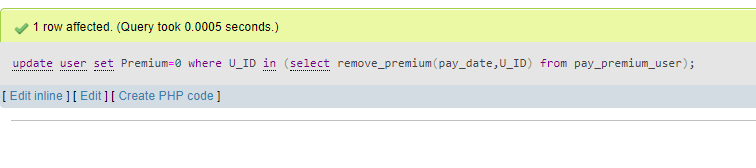
$$

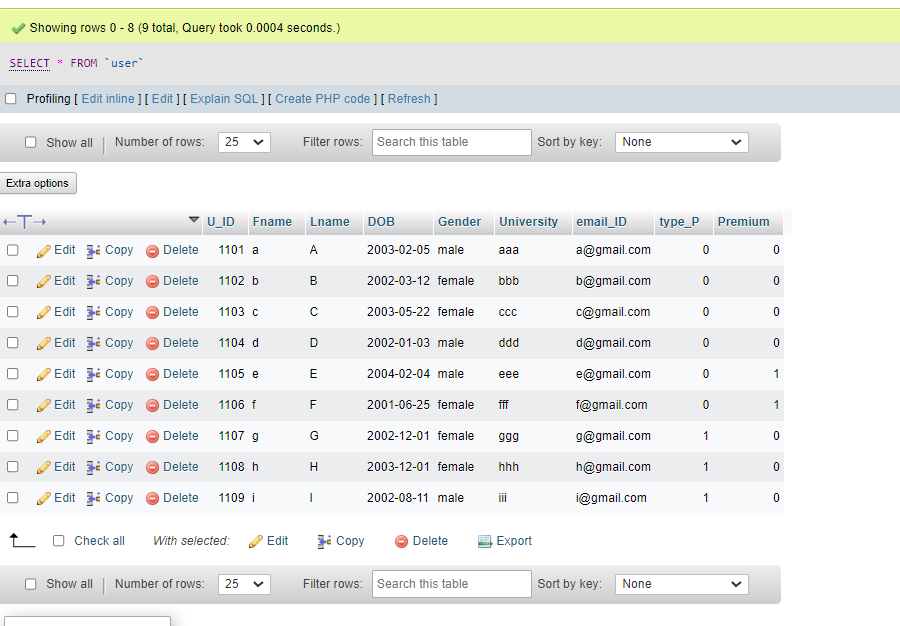
delimiter ;

Trigger results:



Function and view results:





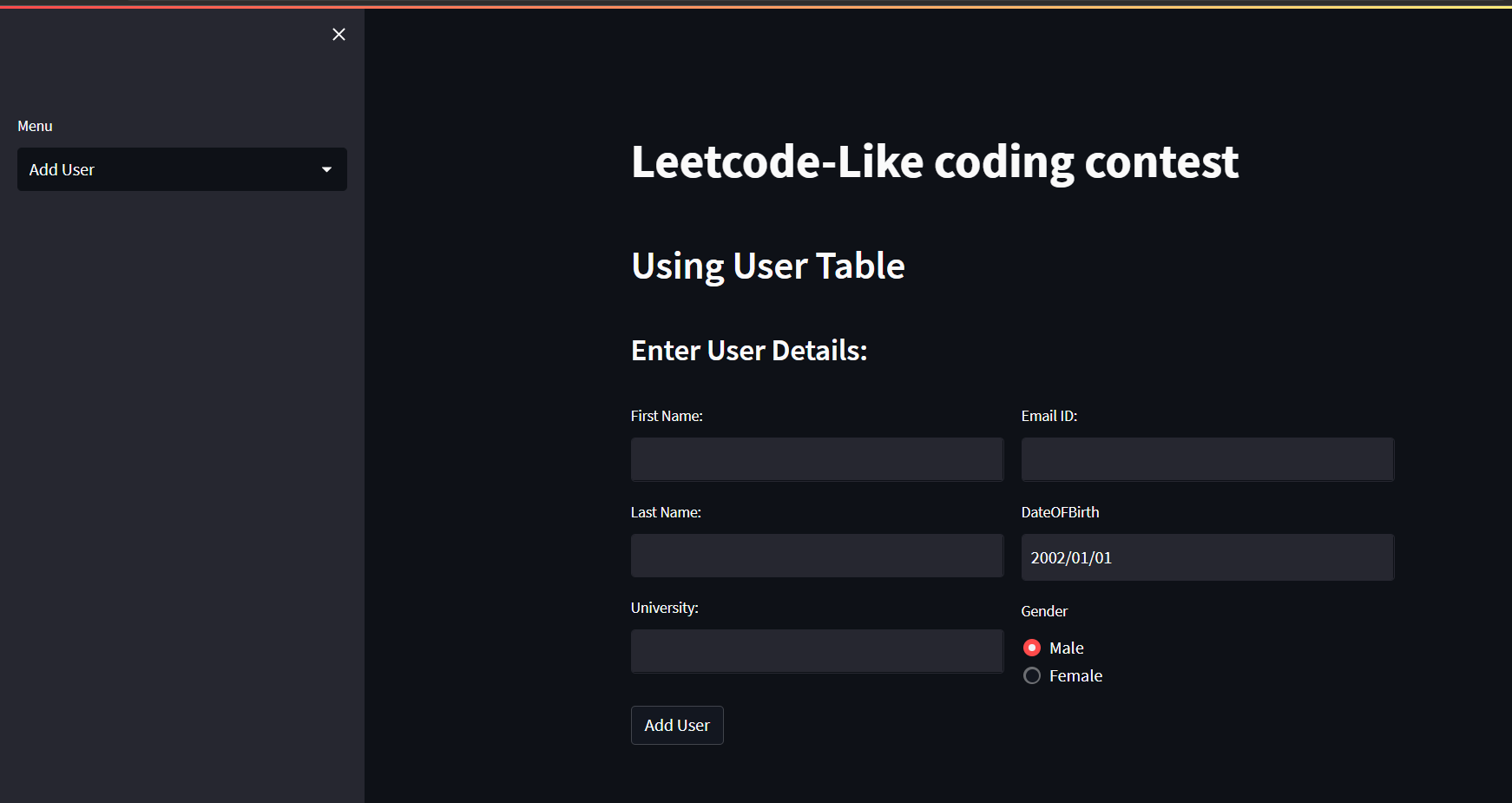
**Developing a Frontend**

The frontend should support

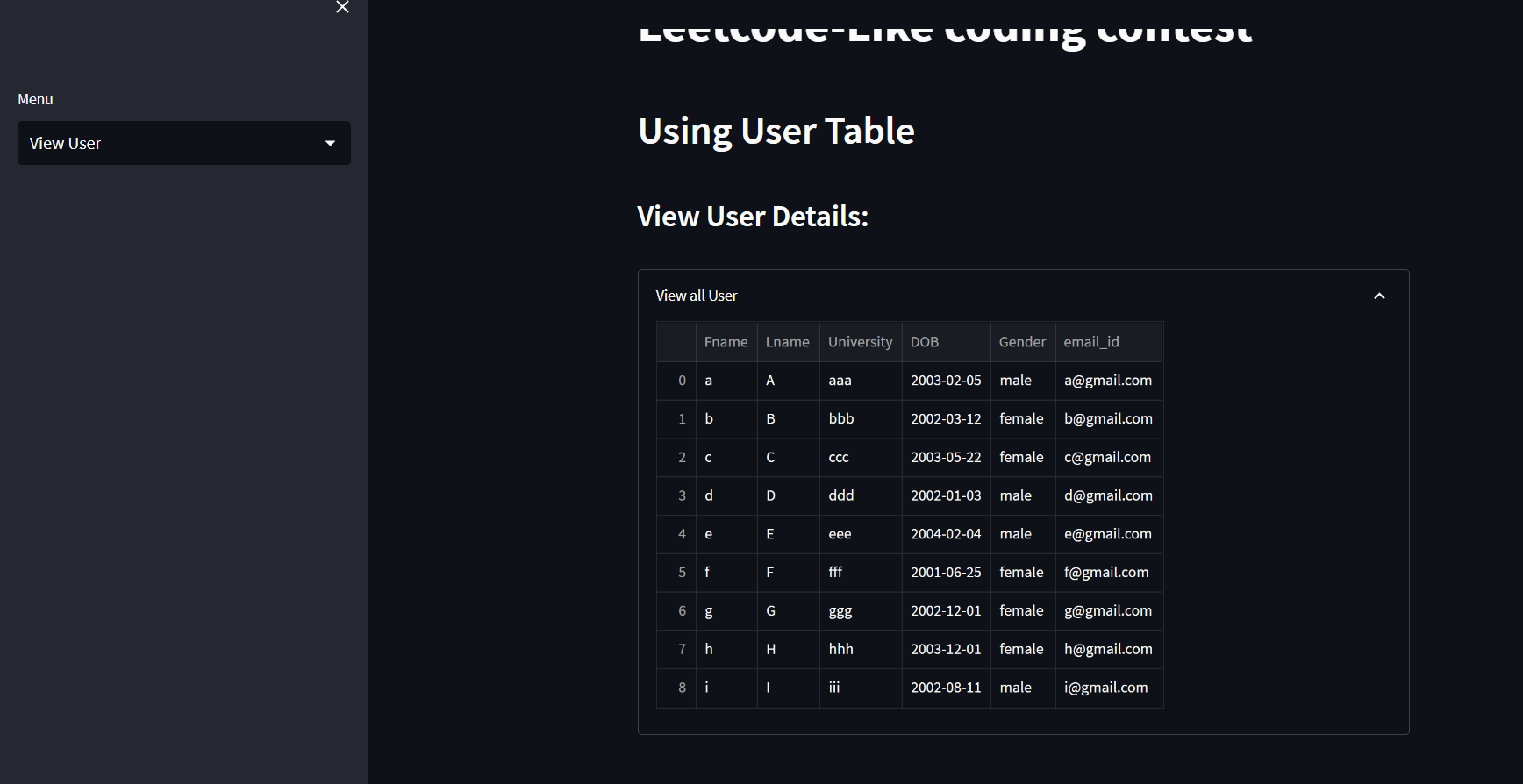
1. Addition, Modification and Deletion of records from any chosen table

2. There should be a window to accept and run any SQL statement and display the result

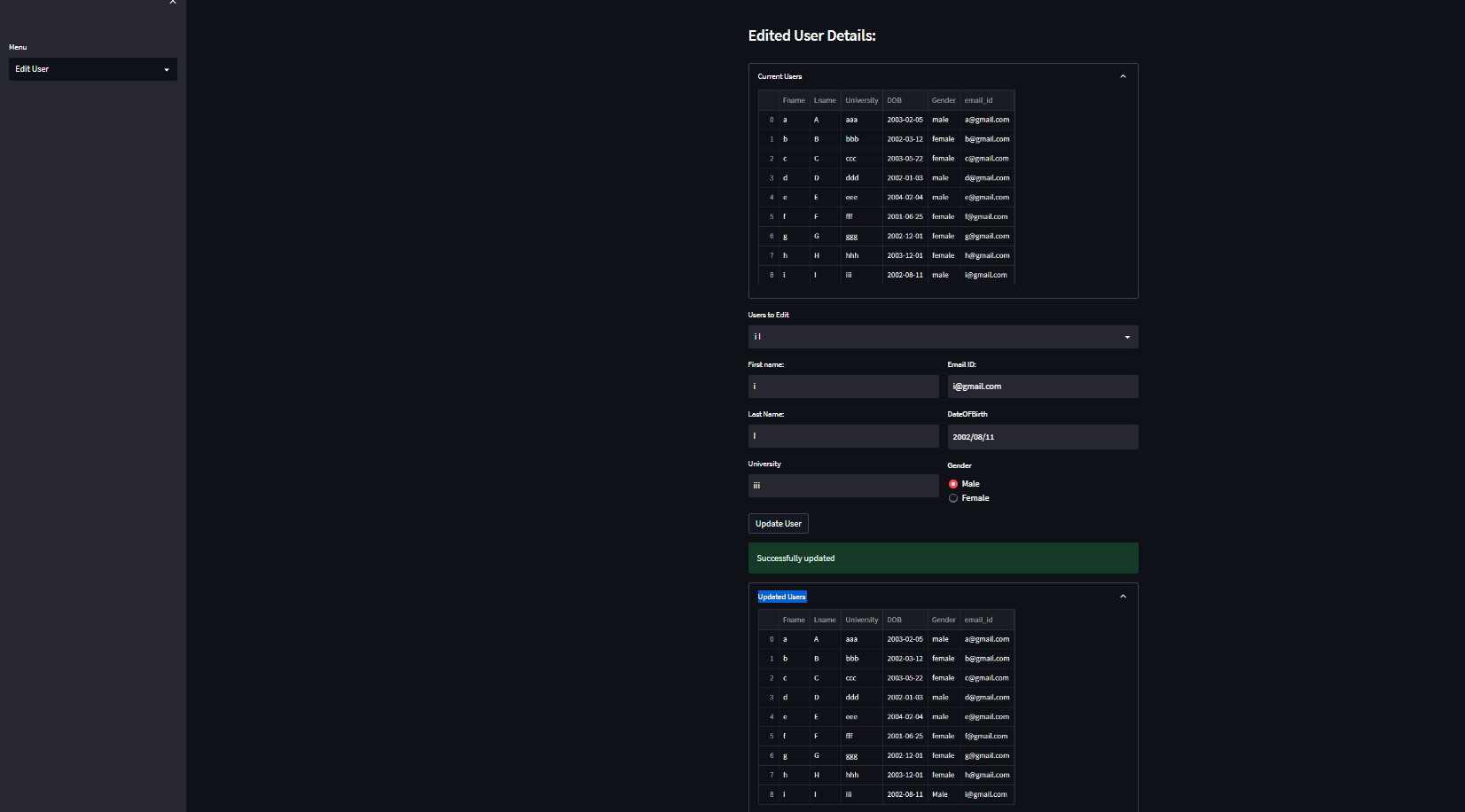
Add:



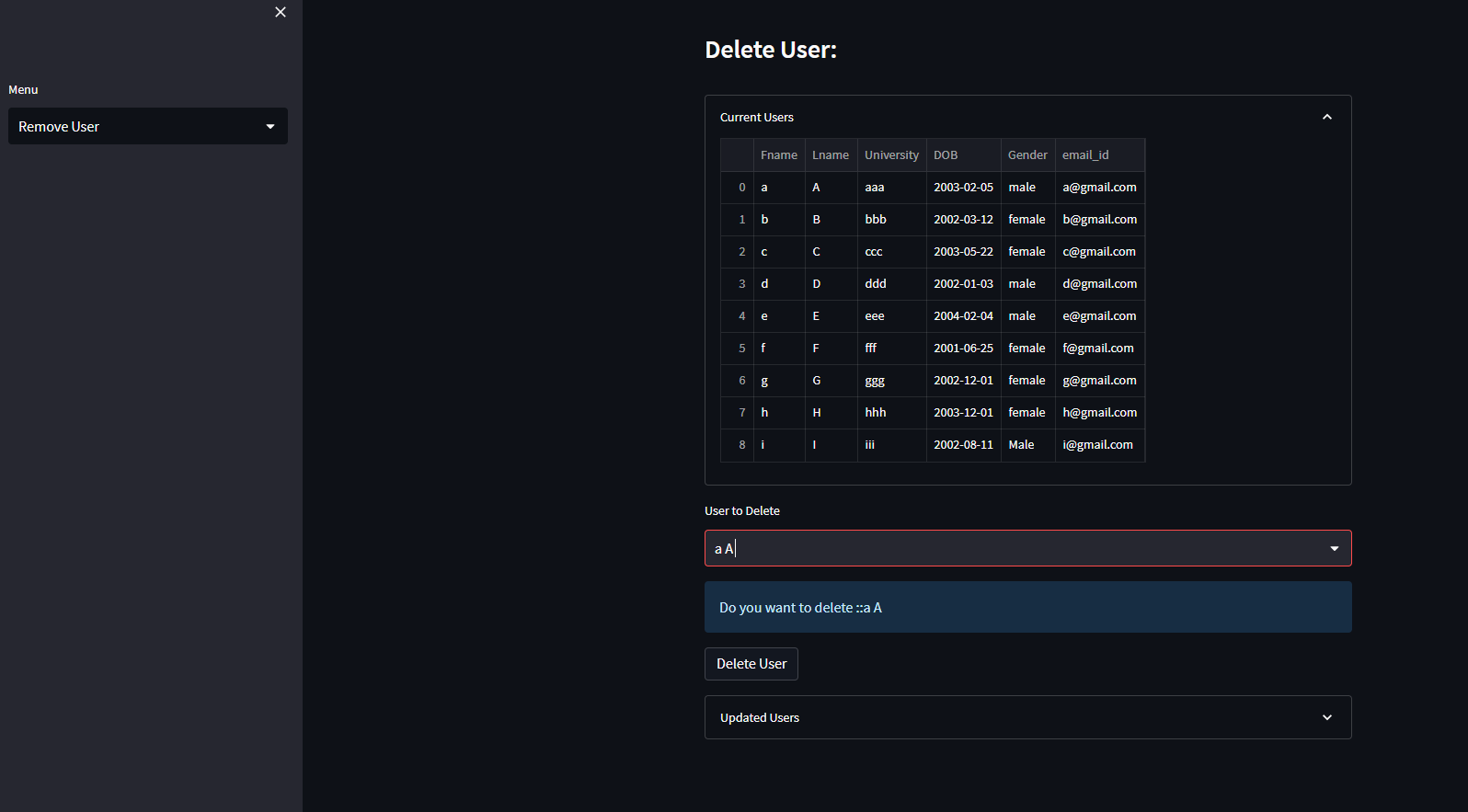
View:



Update:



Delete:



Query Box:

