DBMS - MINI PROJECT

“Leetcode like coding contest database managment”

Submitted By:

Manoj kumar H R

PES1UG20CS560

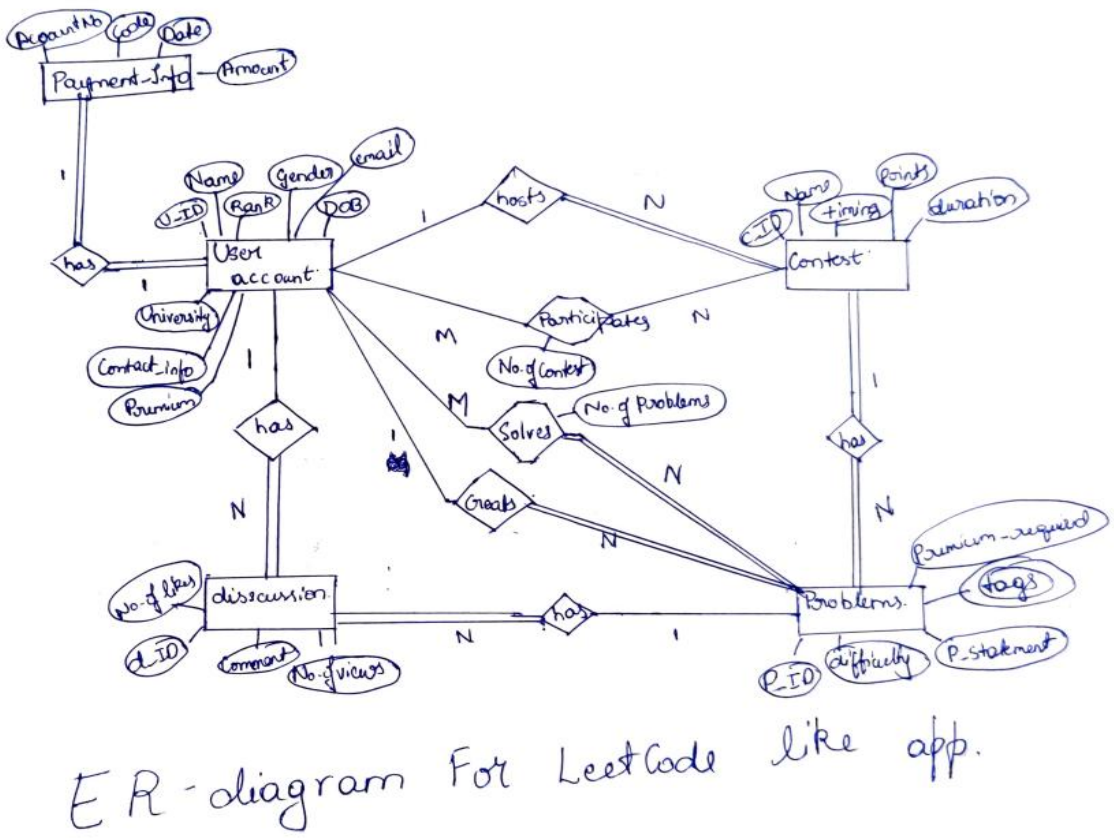
V Semester

Section J

ABSTRACT

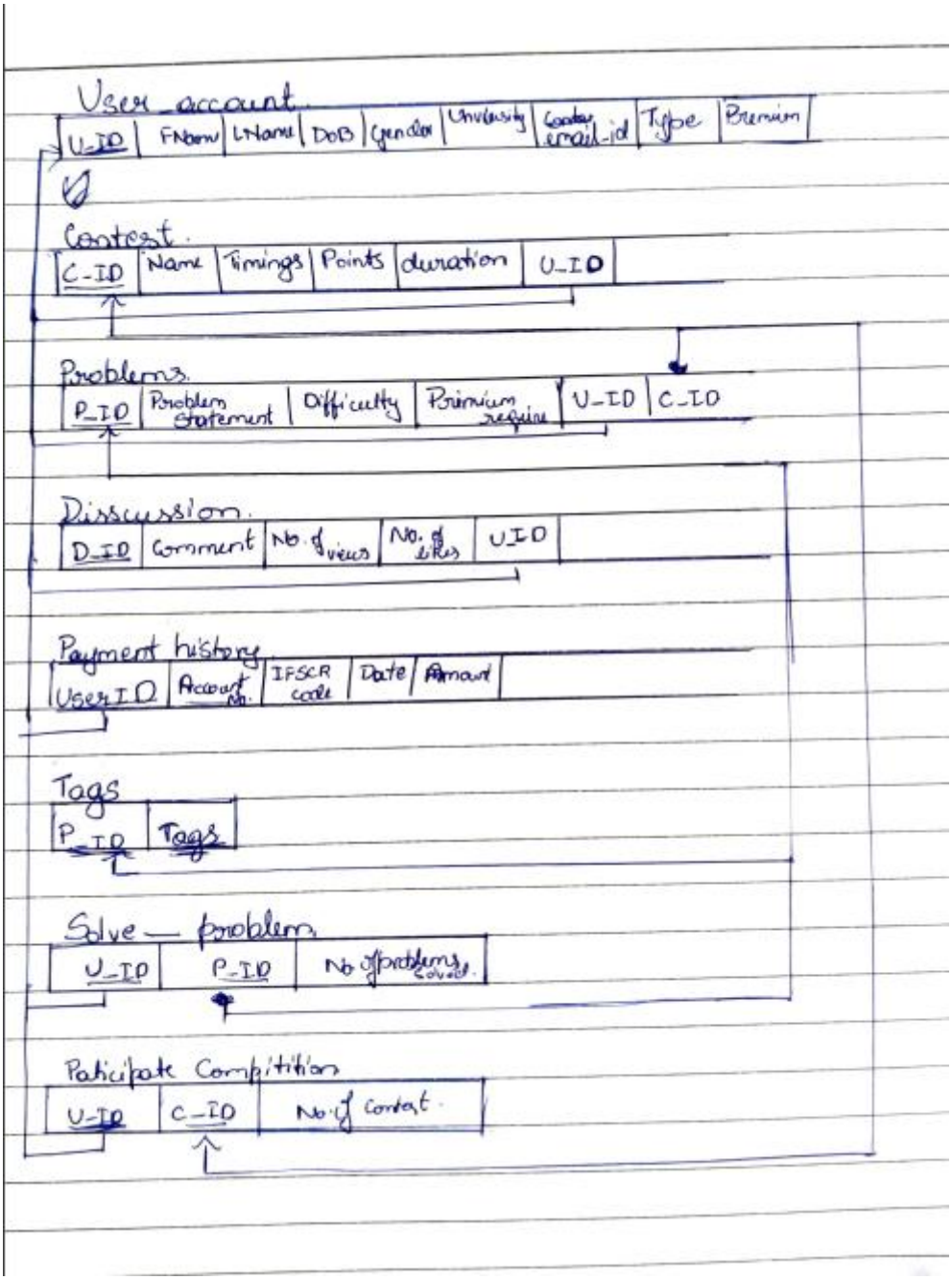
With the increase in number of engineering students every year, The challenge of getting better in coding is always there. Leetcode a very famous coding platform helps students to improve on their coding skills and with their regular contest, different levels of problems the data to manage is very hard, 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



ER-diagram For LeetCode like app.

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)

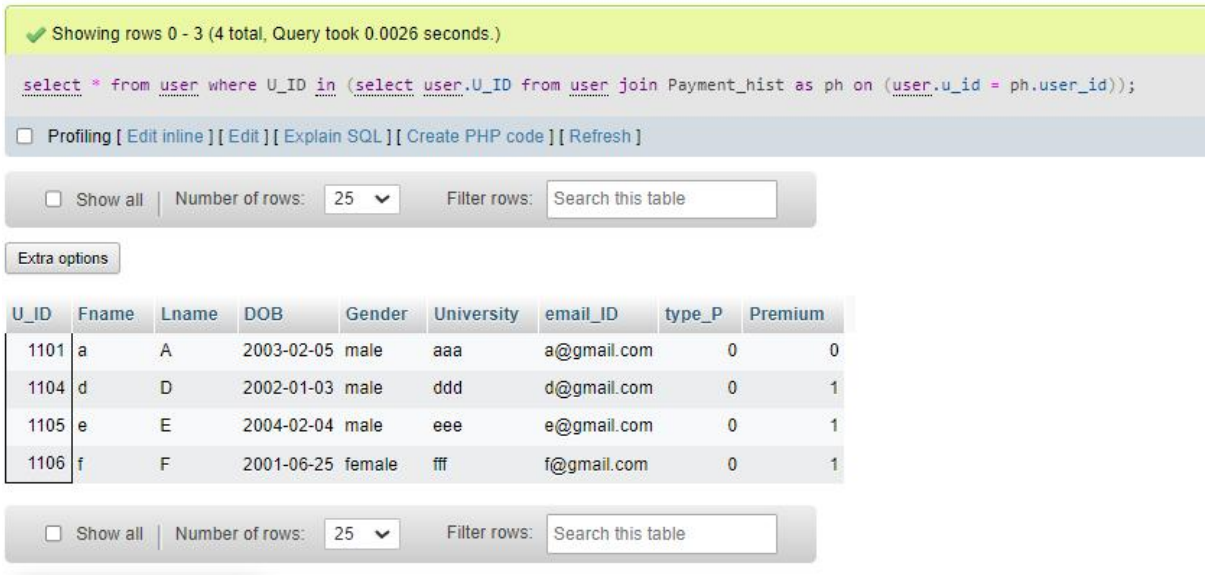
```
statement: display user detials solved atleast one problem
code: select fname,lname,dob,University,Gender from user join Solve_problem as
sp on (user.U_ID=sp.U_ID);
```

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:



Showing rows 0 - 3 (4 total, Query took 0.0026 seconds.)

```
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));
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

U_ID	Fname	Lname	DOB	Gender	University	email_ID	type_P	Premium
1101	a	A	2003-02-05	male	aaa	a@gmail.com	0	0
1104	d	D	2002-01-03	male	ddd	d@gmail.com	0	1
1105	e	E	2004-02-04	male	eee	e@gmail.com	0	1
1106	f	F	2001-06-25	female	fff	f@gmail.com	0	1

Show all | Number of rows: 25 Filter rows: Search this table

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:

Showing rows 0 - 1 (2 total, Query took 0.0027 seconds.)

```
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);
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	U_ID
Edit Copy Delete	1104
Edit Copy Delete	1106

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);
```

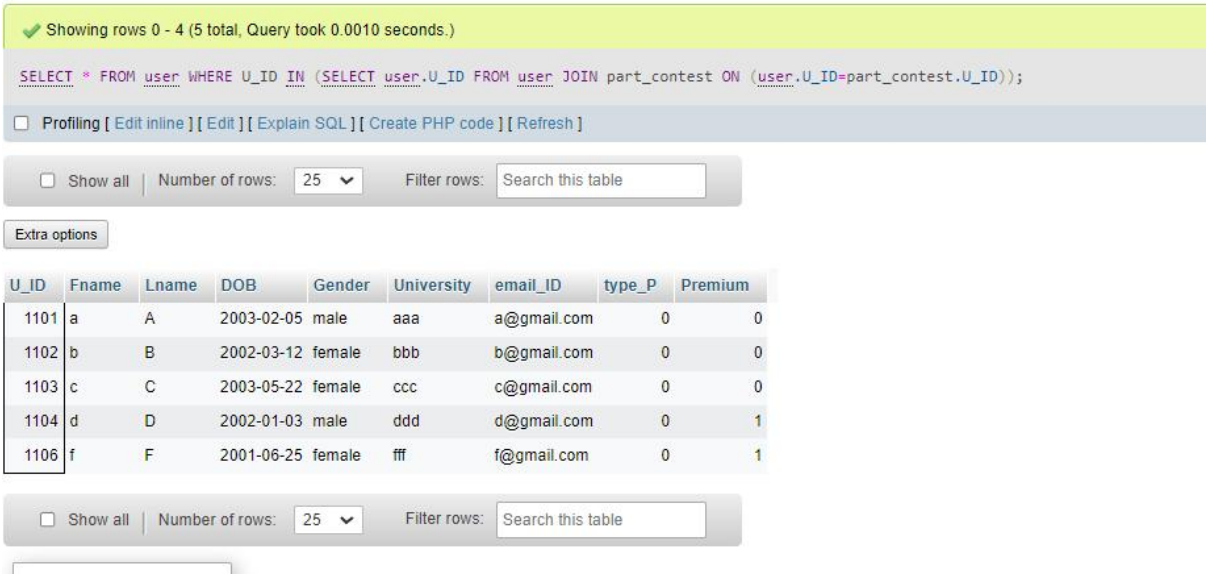
output:

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));
```

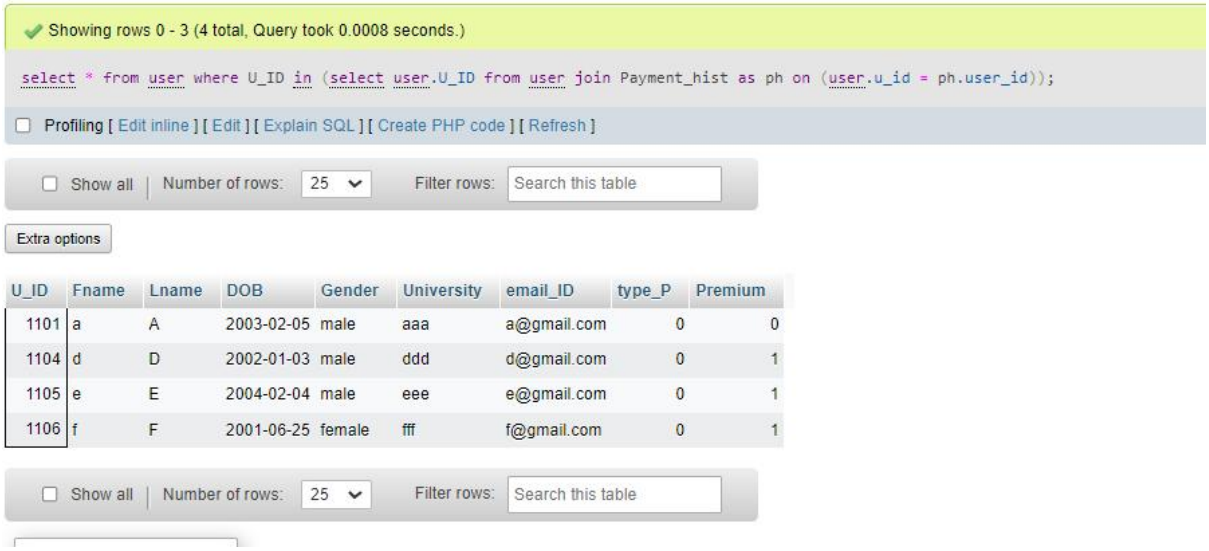
output:



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:



Showing rows 0 - 3 (4 total, Query took 0.0008 seconds.)

```
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));
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

U_ID	Fname	Lname	DOB	Gender	University	email_ID	type_P	Premium
1101	a	A	2003-02-05	male	aaa	a@gmail.com	0	0
1104	d	D	2002-01-03	male	ddd	d@gmail.com	0	1
1105	e	E	2004-02-04	male	eee	e@gmail.com	0	1
1106	f	F	2001-06-25	female	fff	f@gmail.com	0	1

Show all | Number of rows: 25 Filter rows: Search this table

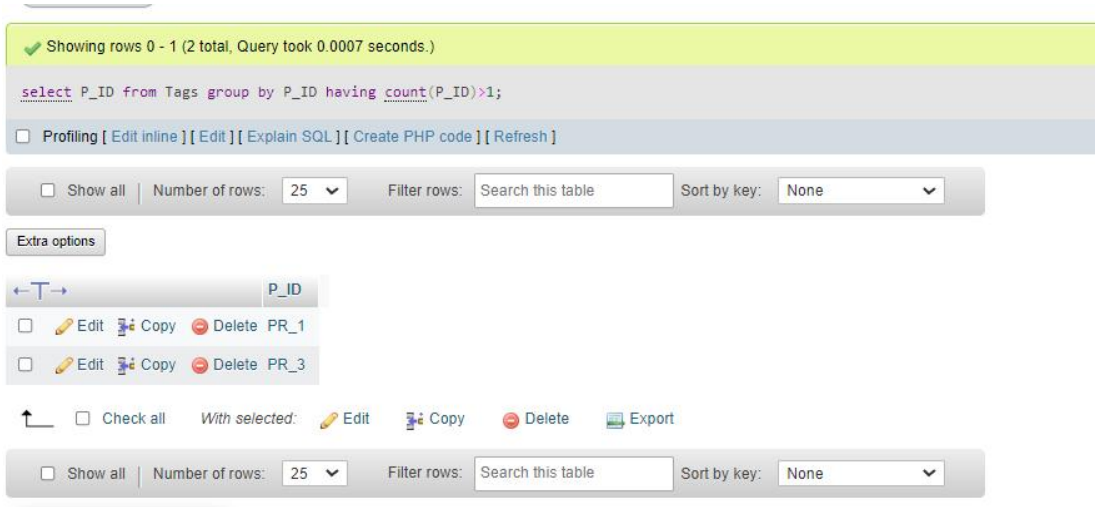
Aggregate Functions (at least 2)

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:



Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

```
select P_ID from Tags group by P_ID having count(P_ID)>1;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	P_ID
Edit Copy Delete	PR_1
Edit Copy Delete	PR_3

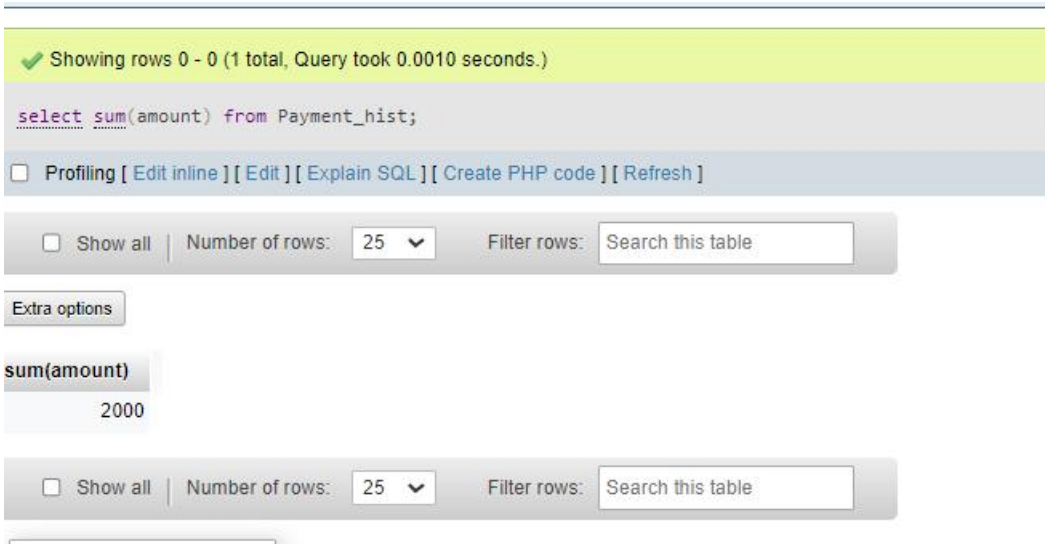
Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

2)

```
display the total amount collected through premium buys
select sum(amount) from Payment_hist;
```

Output:



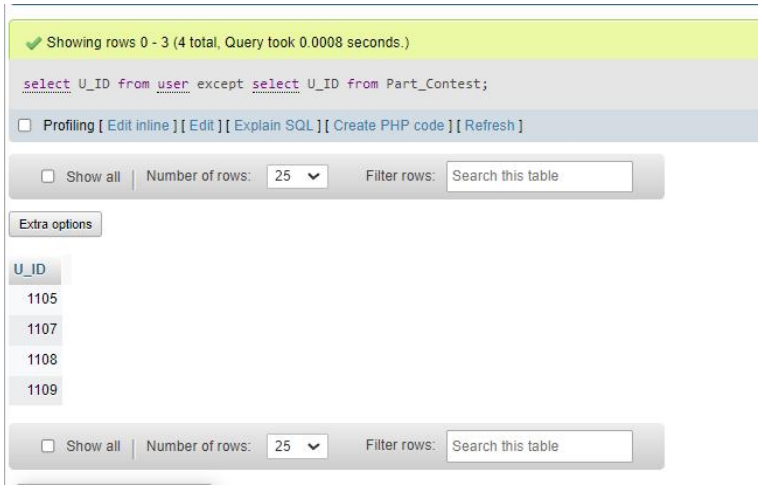
Set Operations (at least 2)

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.

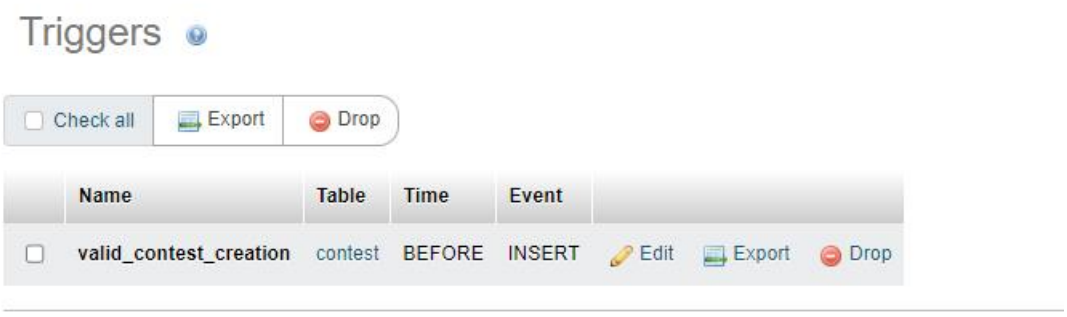
Triggers (Functions or Procedures)

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

display the results.

Trigger:

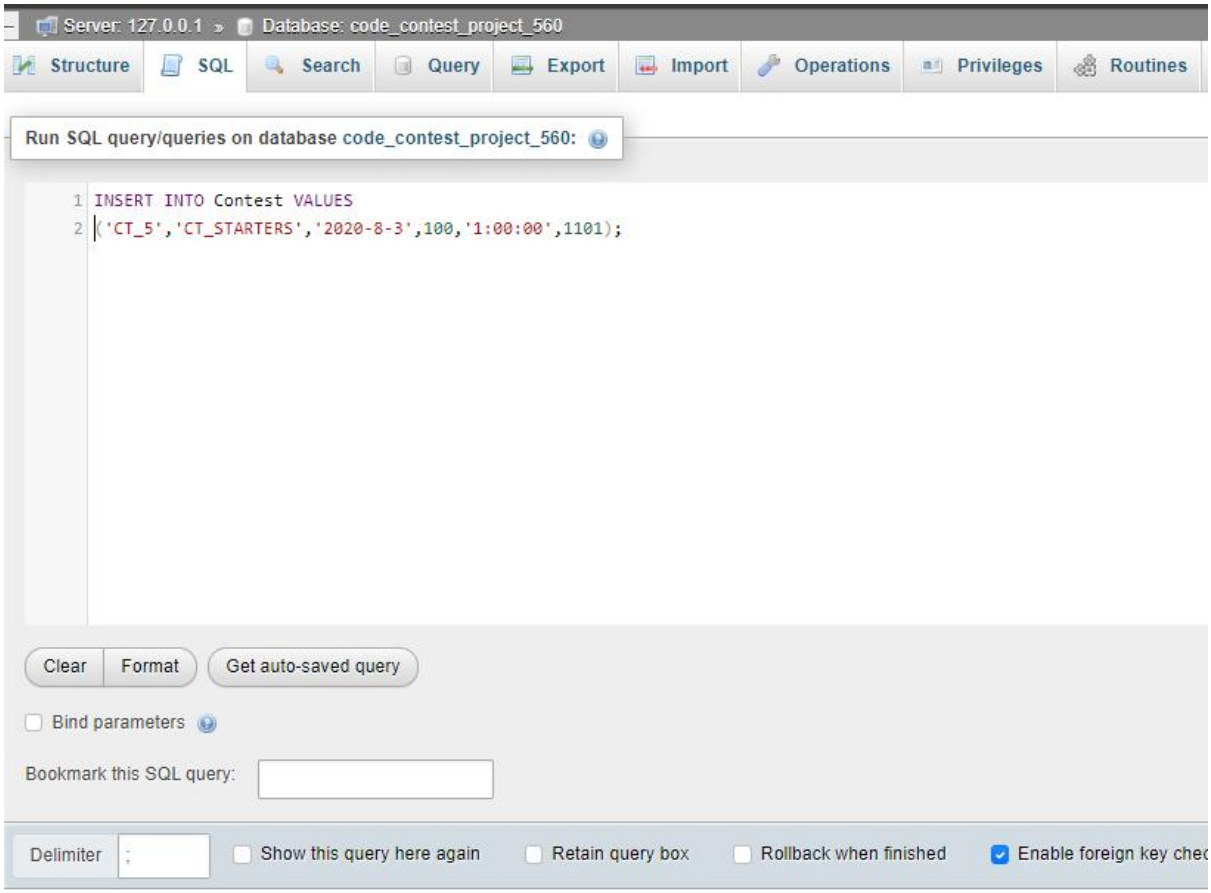
```
delimiter $$

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:



Error

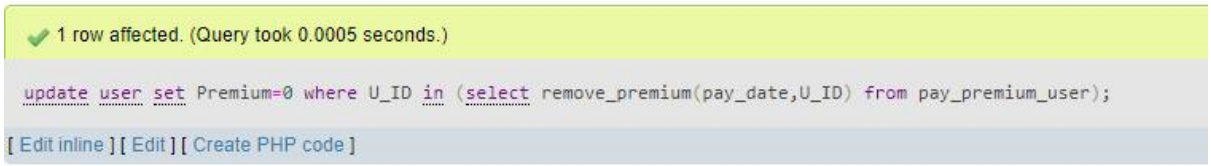
SQL query: Copy

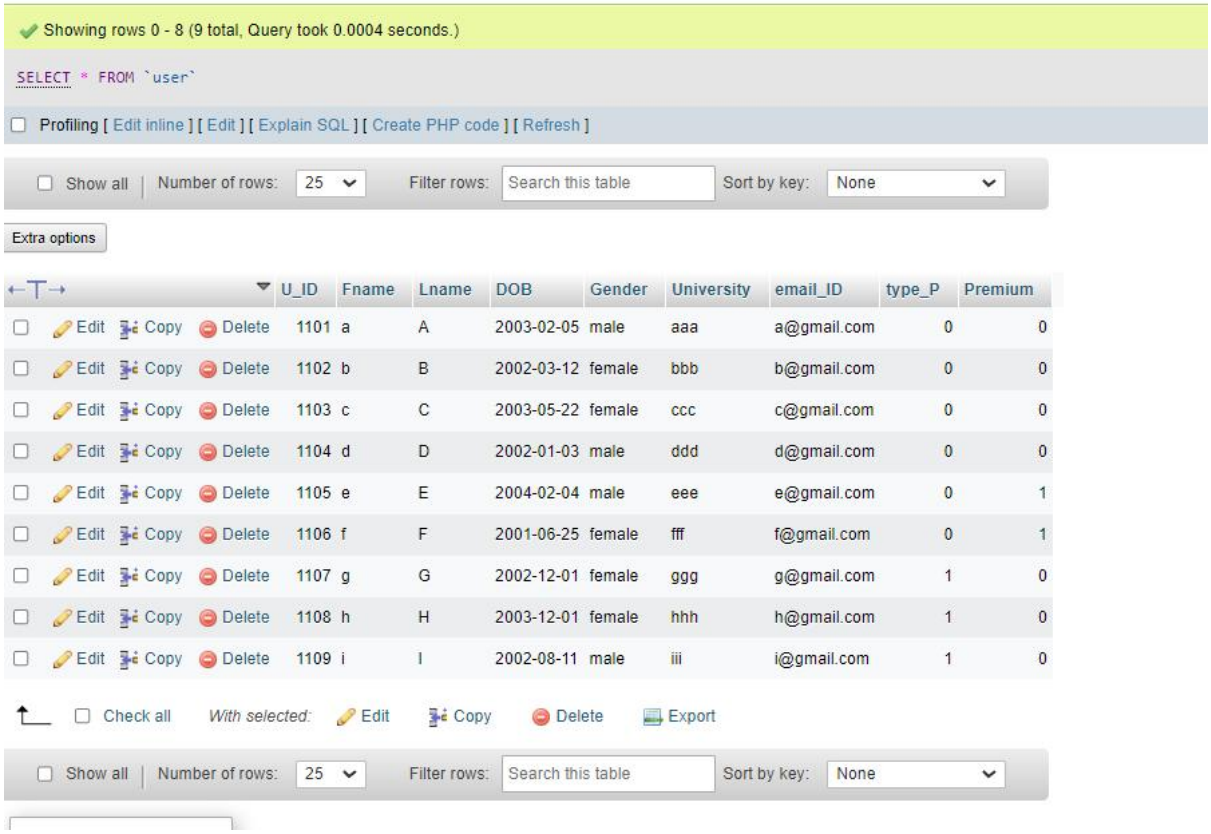
```
INSERT INTO Contest VALUES
('CT_5','CT_STARTERS','2020-8-3',100,'1:00:00',1101);
```

MySQL said:

```
#1644 - ERROR: You don't have admin privilages to add contest
```

Function and view results:





	U_ID	Fname	Lname	DOB	Gender	University	email_ID	type_P	Premium
Edit Copy Delete	1101	a	A	2003-02-05	male	aaa	a@gmail.com	0	0
Edit Copy Delete	1102	b	B	2002-03-12	female	bbb	b@gmail.com	0	0
Edit Copy Delete	1103	c	C	2003-05-22	female	ccc	c@gmail.com	0	0
Edit Copy Delete	1104	d	D	2002-01-03	male	ddd	d@gmail.com	0	0
Edit Copy Delete	1105	e	E	2004-02-04	male	eee	e@gmail.com	0	1
Edit Copy Delete	1106	f	F	2001-06-25	female	fff	f@gmail.com	0	1
Edit Copy Delete	1107	g	G	2002-12-01	female	ggg	g@gmail.com	1	0
Edit Copy Delete	1108	h	H	2003-12-01	female	hhh	h@gmail.com	1	0
Edit Copy Delete	1109	i	I	2002-08-11	male	iii	i@gmail.com	1	0

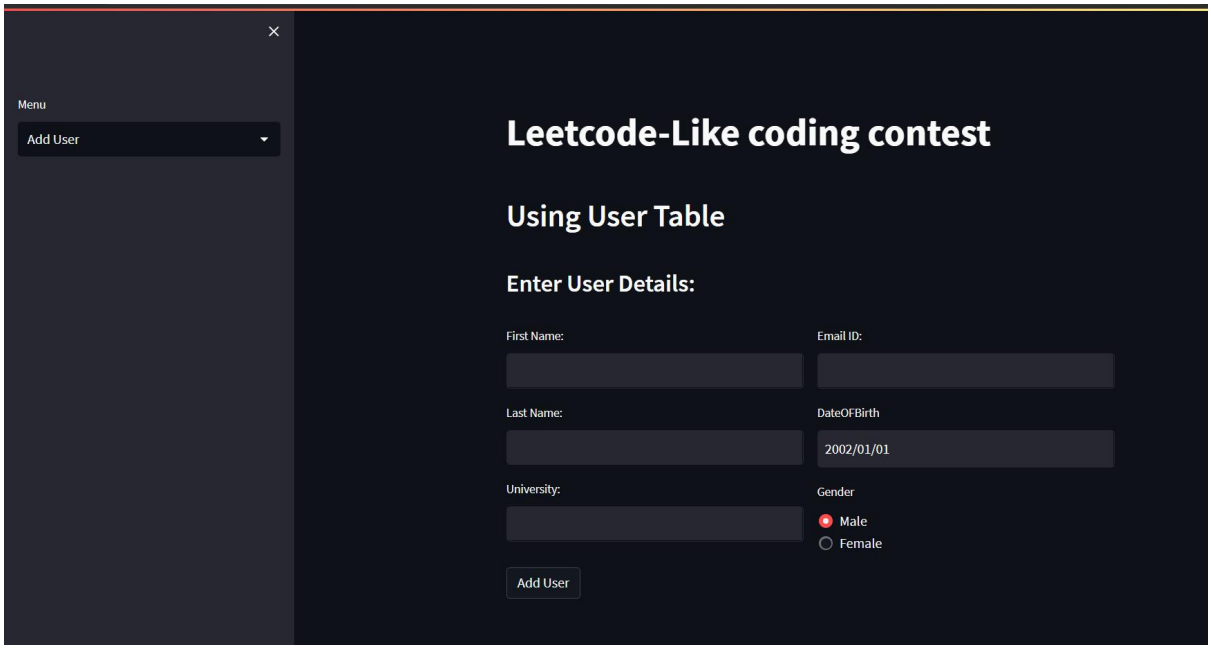
Developing a Frontend

The frontend should support

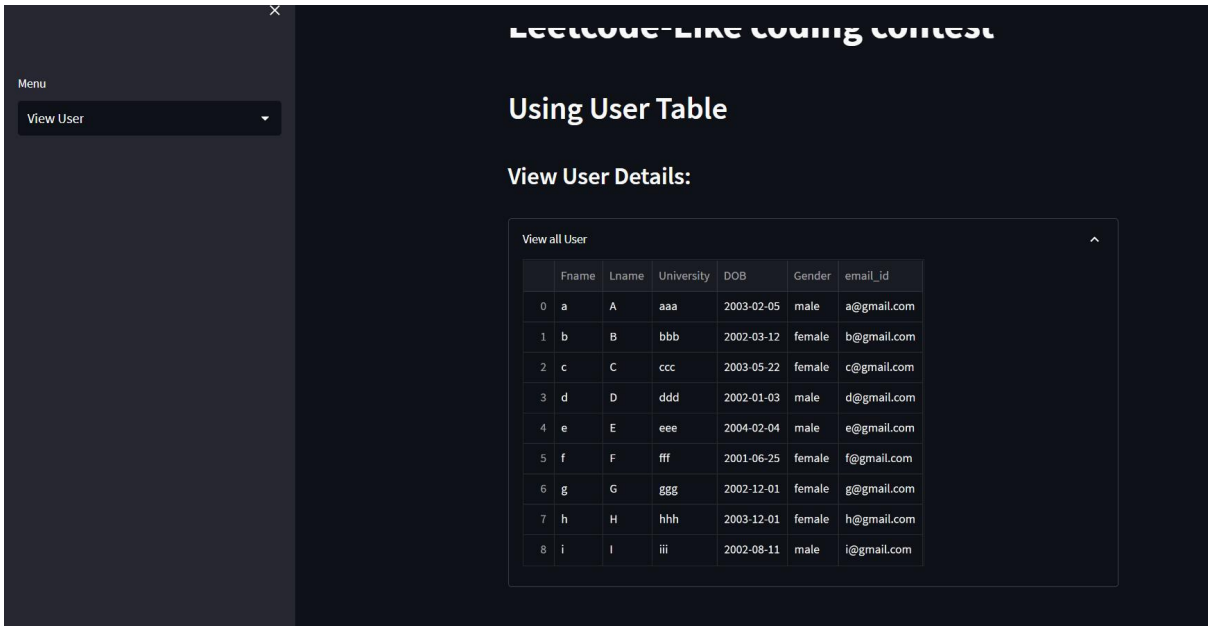
1. Addition, Modification and Deletion of records from any chosen table

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

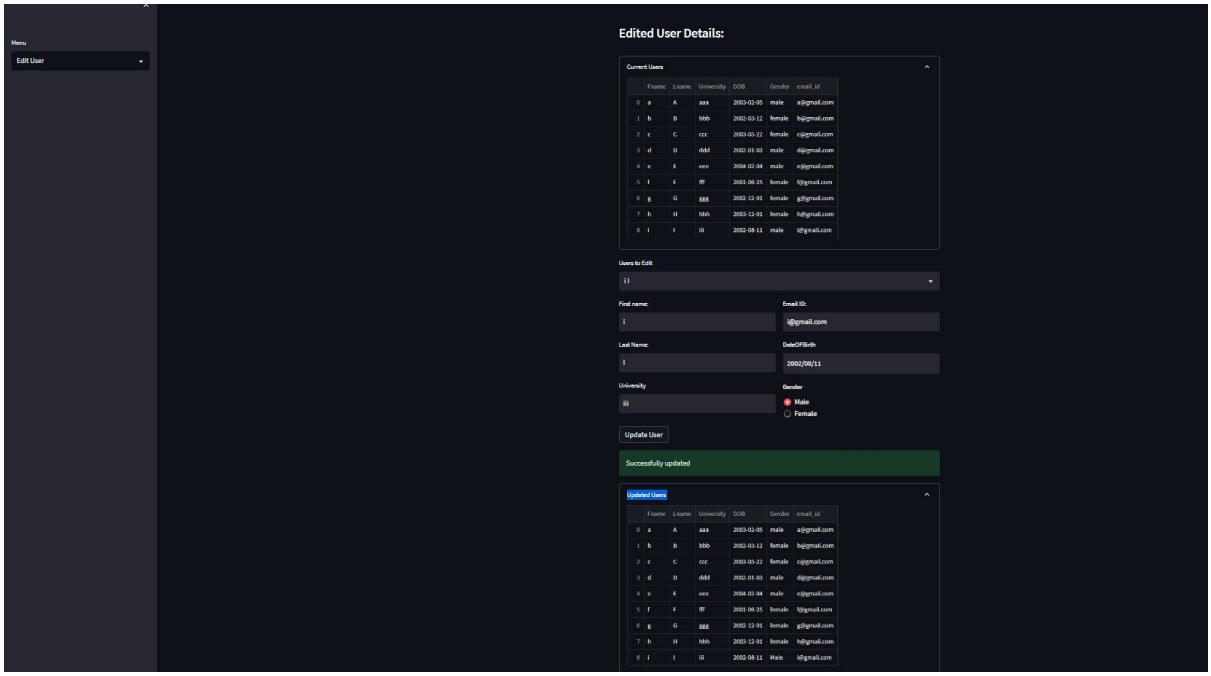
Add:



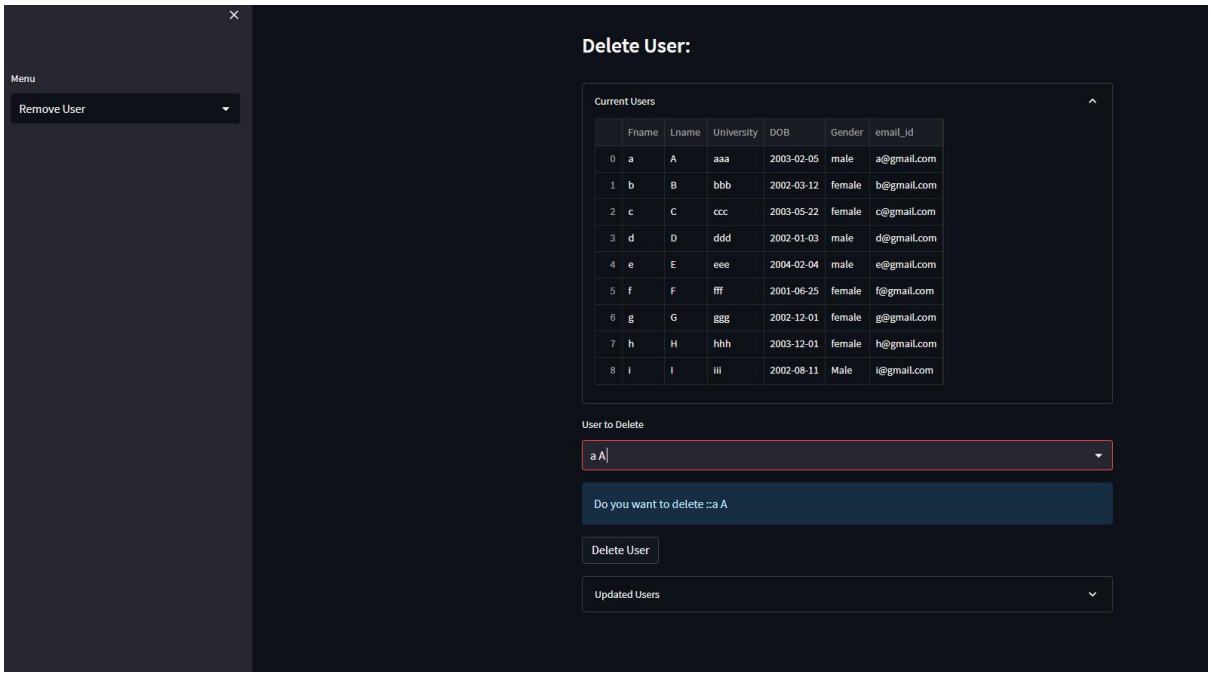
View:



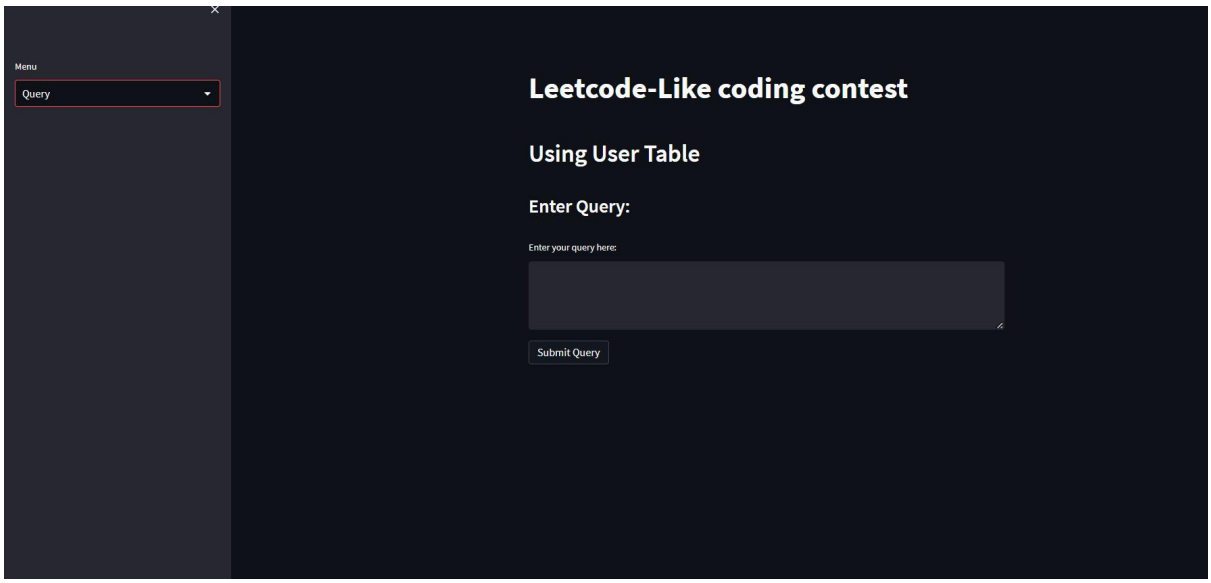
Update:



Delete:



Query Box:


Menu
Query
Leetcode-Like coding contest
Using User Table
Enter Query:
Enter your query here:
Submit Query