# DBMS - Mini Project

# Title of the Project – CharterGhar(House Rental Management System)

Submitted By: Vrushank G

SRN : PES1UG20CS516

V Semester

Section : I

# Short Description and Scope of the Project

* House management has become important factor in modern society hence the need to have a house rental management system
* Charter-Ghar is a House rental management website where house owners, agents and tenants can exchange information effectively and inexpensively.
* Provides user-friendly interface, satisfying the needs of the consumers.
* Employs a new strategy that facilitates easy management of rental houses.

**Scope:**

The project scope defines the description of the work that is required in delivering the rental house management system.

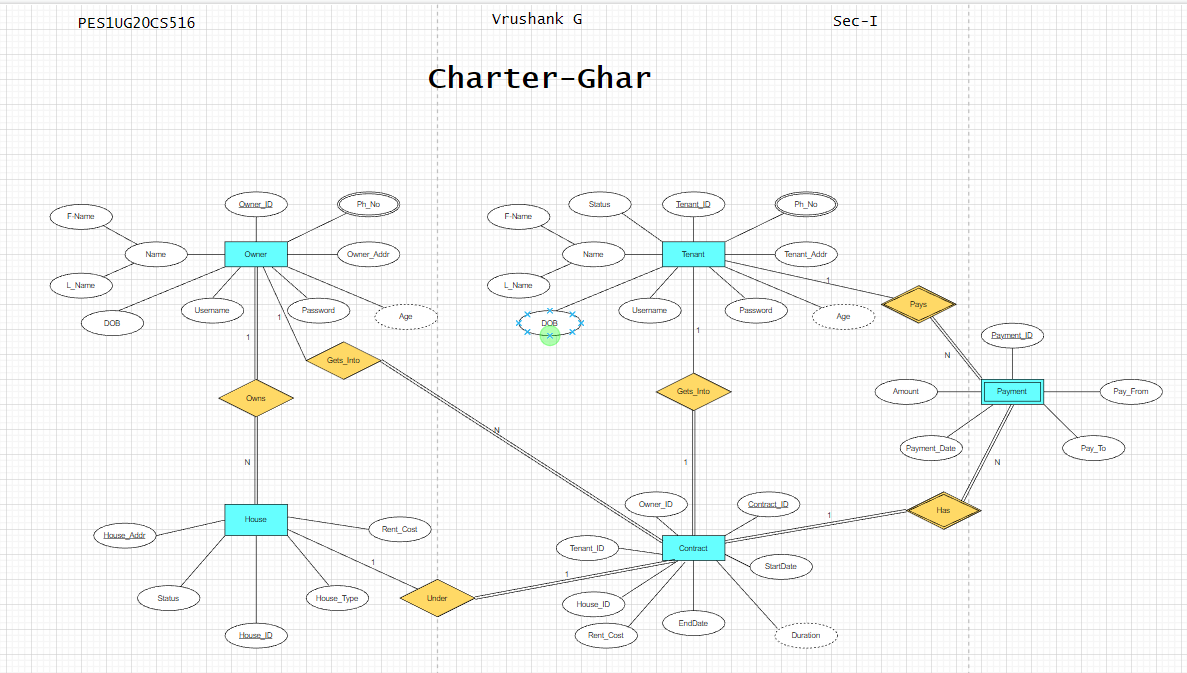
The following are the scopes of work during the course of the project:

* Study and understand the requirement of this project
* ER diagram and relational schema
* A minimalistic UI/front-end
* Creating and populating the database
* Using the SQL queries and show the reflections in the front-end developed

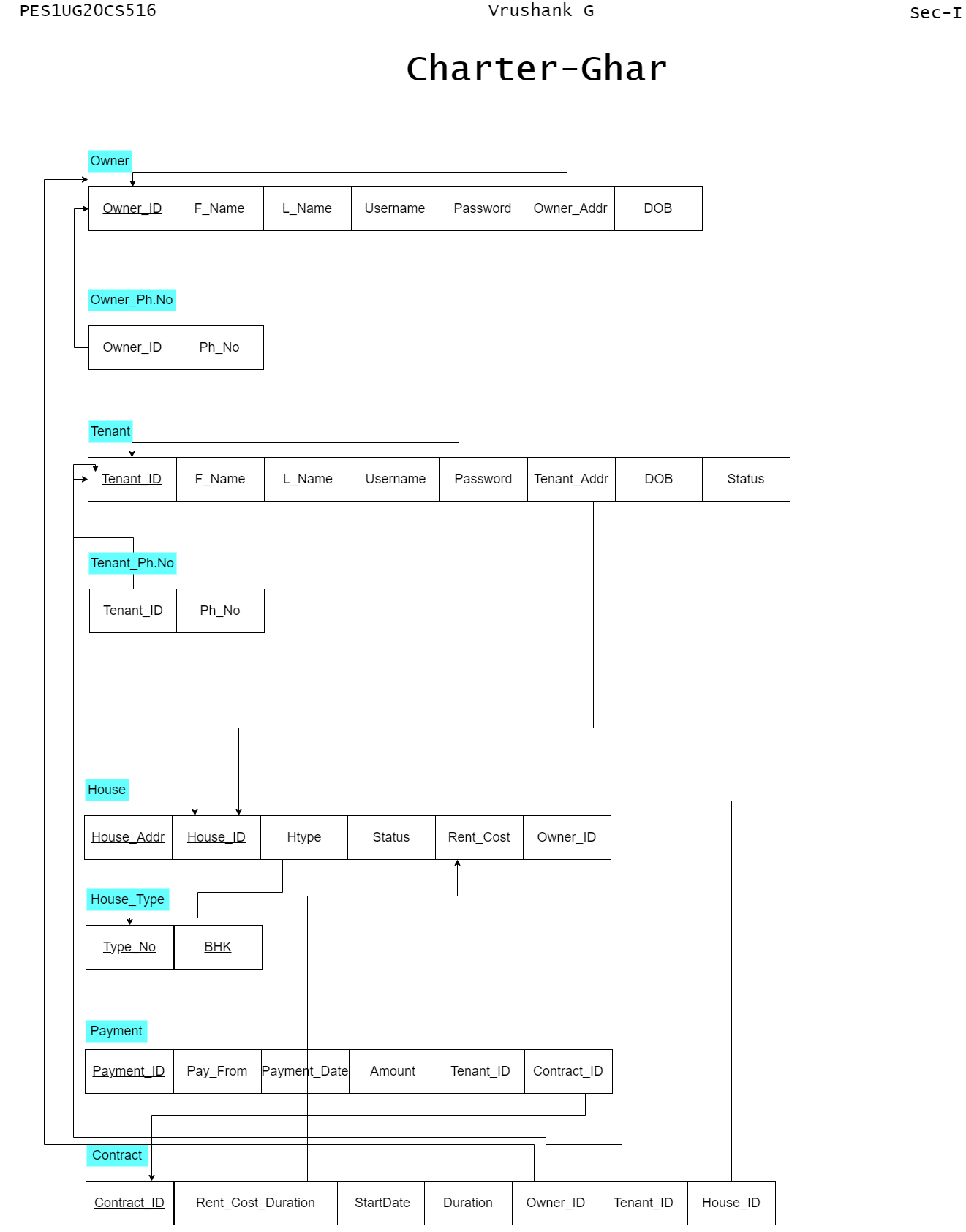
**Modules Used:**

* Owner: Displays the details of the Houses he/she owns for rent.
* Tenant: Searched the best suited house for the living and makes contract with the owner once he/she buys/rents the house.
* Contract: Stores the details of Tenant renting the house belonging to respective Owner.
* Payment: Payment window for making multiple payments for a contract

**ER Diagram**



# Relational Schema



**DDL statements - Building the database**

--Owner Table

CREATE TABLE IF NOT EXISTS Owner(

    Owner\_ID int(6) auto\_increment,

    primary key(Owner\_ID),

    F\_Name varchar(10) not null,

    L\_Name varchar(10) not null,

    Username varchar(10) not null unique,

    Password varchar(10) not null,

    Owner\_Addr varchar(50) unique,

    DOB date

);

alter table Owner auto\_increment=100000;

--Owner\_Ph.No

CREATE TABLE IF NOT EXISTS Owner\_Ph(

    Owner\_ID int(6),

    constraint fk\_owner\_id foreign key(Owner\_ID) REFERENCES Owner(Owner\_ID),

    Ph\_No char(10) not null

);

alter table Owner\_Ph add constraint check\_ph check(char\_length(Ph\_No)=10);

--Tenant

CREATE TABLE IF NOT EXISTS Tenant(

    Tenant\_ID int(6) auto\_increment,

    primary key(Tenant\_ID),

    F\_Name varchar(10) not null,

    L\_Name varchar(10) not null,

    Username varchar(10) not null unique,

    Password varchar(10) not null,

    Tenant\_Addr varchar(50) default NULL,

    -- constraint fk\_tenant\_addr foreign key(Tenant\_Addr) REFERENCES House(House\_Addr),

    DOB date,

    Status boolean default false

);

alter table Tenant auto\_increment=300000;

--Tenant\_Ph.No

CREATE TABLE IF NOT EXISTS Tenant\_Ph(

    Tenant\_ID int(6),

    constraint fk\_tenant\_id foreign key(Tenant\_ID) REFERENCES Tenant(Tenant\_ID),

    Ph\_No char(10) not null

);

alter table Tenant\_Ph add constraint check\_ph check(char\_length(Ph\_No)=10);

--House\_type

CREATE TABLE IF NOT EXISTS House\_Type(

    Type\_No int(2),

    BHK varchar(6),

    primary key(Type\_No,BHK)

);

--House

CREATE TABLE IF NOT EXISTS House(

    House\_ID int auto\_increment,

    House\_Addr varchar(50),

    primary key(House\_ID,House\_Addr),

    Htype int(2),

    constraint fk\_house\_type foreign key(Htype) REFERENCES House\_type(Type\_No),

    Rent\_Cost int(10) not null,

    Owner\_ID int(6),

    constraint fk\_owner\_id2 foreign key(Owner\_ID) REFERENCES Owner(Owner\_ID)

);

alter table House auto\_increment=200000;

--Contract

CREATE TABLE IF NOT EXISTS Contract(

    Contract\_ID int(6) auto\_increment,

    primary key(Contract\_ID),

    Start\_Date date,

    Duration int(2) not null default 1,

    Rent\_Cost\_Duration int(10),

    -- constraint fk\_rent\_cost foreign key(Rent\_Cost\_Duration) references House(Rent\_Cost),

    Owner\_ID int(6),

    constraint fk\_owner\_id3 foreign key(Owner\_ID) REFERENCES Owner(Owner\_ID),

    Tenant\_ID int(6),

    constraint fk\_tenant\_id2 foreign key(Tenant\_ID) REFERENCES Tenant(Tenant\_ID),

    House\_ID int(6),

    constraint fk\_house\_id foreign key(House\_ID) REFERENCES House(House\_ID)

);

alter table Contract auto\_increment=400000;

--Payment

CREATE TABLE IF NOT EXISTS Payment(

    Payment\_ID int auto\_increment,

    primary key(Payment\_ID),

    Pay\_From varchar(20) not null,

    Payment\_Date date,

    Amount int(10) not null,

    Tenant\_ID int(6),

    constraint fk\_tenant\_id3 foreign key(Tenant\_ID) REFERENCES Tenant(Tenant\_ID),

    Contract\_ID int(6),

    constraint fk\_contract\_id foreign key(Contract\_ID) REFERENCES Contract(Contract\_ID)

);

alter table Payment auto\_increment=500000;

# Populating the Database

--Owner

insert INTO Owner(F\_Name,L\_Name,username,password,Owner\_Addr,DOB) VALUES("Vrushank","G","vrush41","12345","2nd cross,millerpet,bellary",'2002-08-12');

insert INTO Owner(F\_Name,L\_Name,username,password,Owner\_Addr,DOB) VALUES("Hemanth","N","hemanth28","23456","4th cross,gauribidanur,chikkaballapur",'2002-01-28');

insert INTO Owner(F\_Name,L\_Name,username,password,Owner\_Addr,DOB) VALUES("Dhanush","M D","mdebro","34567","2nd cross,puttur,mangaluru",'2002-06-01');

insert INTO Owner(F\_Name,L\_Name,username,password,Owner\_Addr,DOB) VALUES("Srinivas","Y","vasu03","45678","3rd cross,rr nagar,bengaluru",'2002-03-27');

insert INTO Owner(F\_Name,L\_Name,username,password,Owner\_Addr,DOB) VALUES("Om","Prasad","om123","56789","2nd cross,jaynagar,bengaluru",'2002-05-10');

--Owner\_Ph

insert INTO Owner\_Ph(Owner\_ID,Ph\_No) VALUES(100000,(9876929479));

insert INTO Owner\_Ph(Owner\_ID,Ph\_No) VALUES(100001,(6968456239));

insert INTO Owner\_Ph(Owner\_ID,Ph\_No) VALUES(100002,(9087565642));

insert INTO Owner\_Ph(Owner\_ID,Ph\_No) VALUES(100003,(9980674554));

insert INTO Owner\_Ph(Owner\_ID,Ph\_No) VALUES(100004,(9780678543));

--Tenant

insert INTO Tenant(F\_Name,L\_Name,Username,Password,Tenant\_Addr,DOB) VALUES("Sathvik","A","stvk64","09875","guntur",'2002-04-03');

insert INTO Tenant(F\_Name,L\_Name,Username,Password,Tenant\_Addr,DOB) VALUES("Teja","Kanala","tsreddy43","98735","kurnool",'2002-07-21');

insert INTO Tenant(F\_Name,L\_Name,Username,Password,Tenant\_Addr,DOB) VALUES("Soumith","B","soumpi23","56780","bellary",'2002-01-23');

insert INTO Tenant(F\_Name,L\_Name,Username,Password,Tenant\_Addr,DOB) VALUES("Prathap","P","ptp45","12988","hindupur",'2002-07-09');

insert INTO Tenant(F\_Name,L\_Name,Username,Password,Tenant\_Addr,DOB) VALUES("Nayan","K","nyn987","09876","rajaji nagar,bengaluru",'2002-04-03');

--Tenant\_Ph

insert INTO Tenant\_Ph(Tenant\_ID,Ph\_No) VALUES(300000,(9067825372));

insert INTO Tenant\_Ph(Tenant\_ID,Ph\_No) VALUES(300001,(6273682936));

insert INTO Tenant\_Ph(Tenant\_ID,Ph\_No) VALUES(300002,(9808577578));

insert INTO Tenant\_Ph(Tenant\_ID,Ph\_No) VALUES(300003,(9768457902));

insert INTO Tenant\_Ph(Tenant\_ID,Ph\_No) VALUES(300020,(6363787893));

--House\_Type

insert INTO House\_Type(Type\_No,BHK) VALUES(1,"1 BHK");

insert INTO House\_Type(Type\_No,BHK) VALUES(2,"2 BHK");

insert INTO House\_Type(Type\_No,BHK) VALUES(3,"3 BHK");

insert INTO House\_Type(Type\_No,BHK) VALUES(4,"4 BHK");

insert INTO House\_Type(Type\_No,BHK) VALUES(5,"5 BHK");

--House

INSERT INTO House(House\_Addr,Htype,Rent\_Cost,Owner\_ID) VALUES("Bellary",3,10000,100000);

INSERT INTO House(House\_Addr,Htype,Rent\_Cost,Owner\_ID) VALUES("gauribidanur",2,8000,100001);

INSERT INTO House(House\_Addr,Htype,Rent\_Cost,Owner\_ID) VALUES("puttur",1,7000,100002);

INSERT INTO House(House\_Addr,Htype,Rent\_Cost,Owner\_ID) VALUES("rr nagar",5,15000,100003);

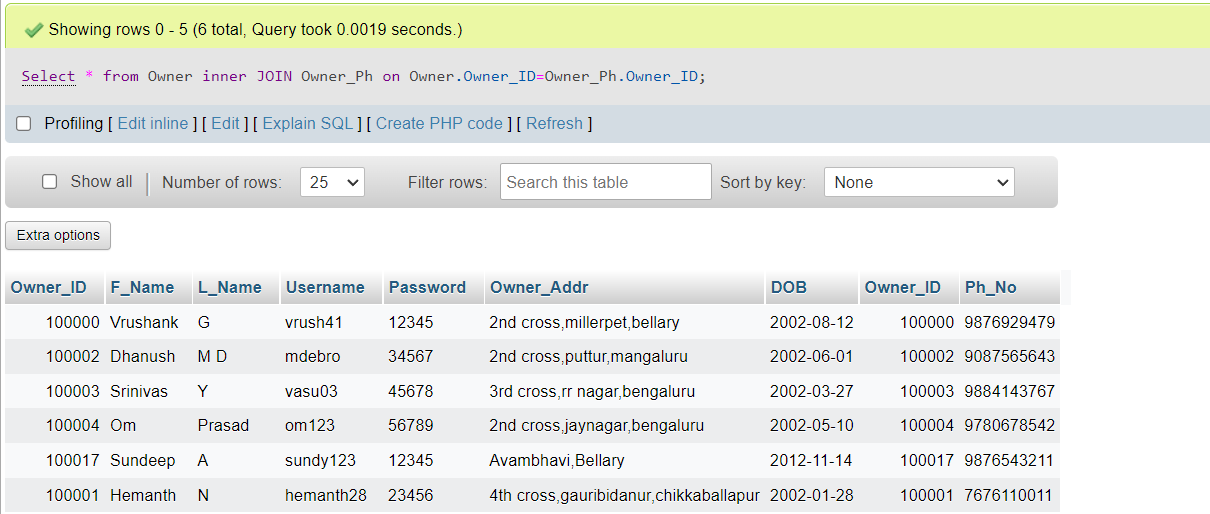
INSERT INTO House(House\_Addr,Htype,Rent\_Cost,Owner\_ID) VALUES("jaynagar",4,20000,100004);

**Join Queries**

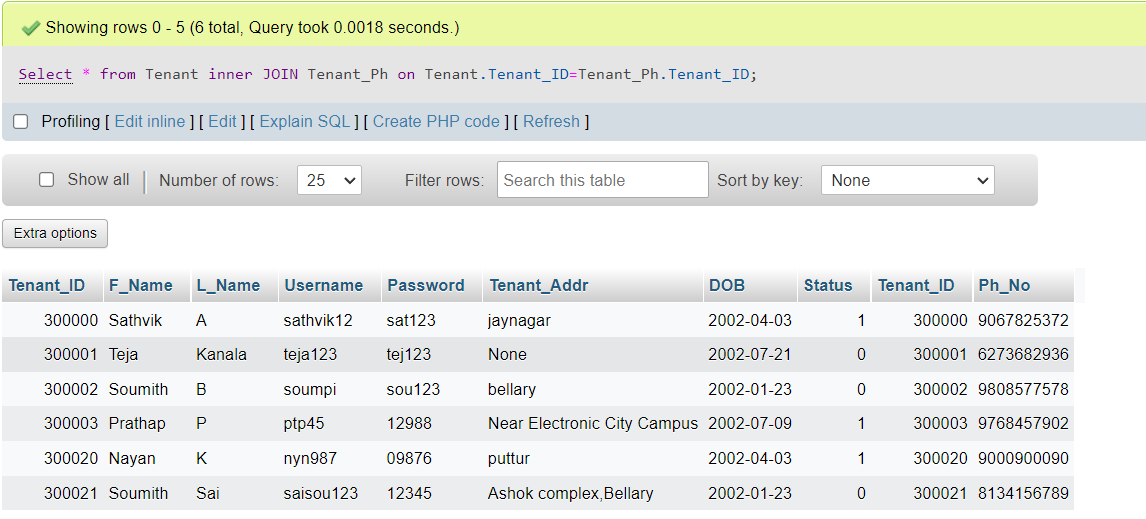
Showcase at least 4 join queries

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

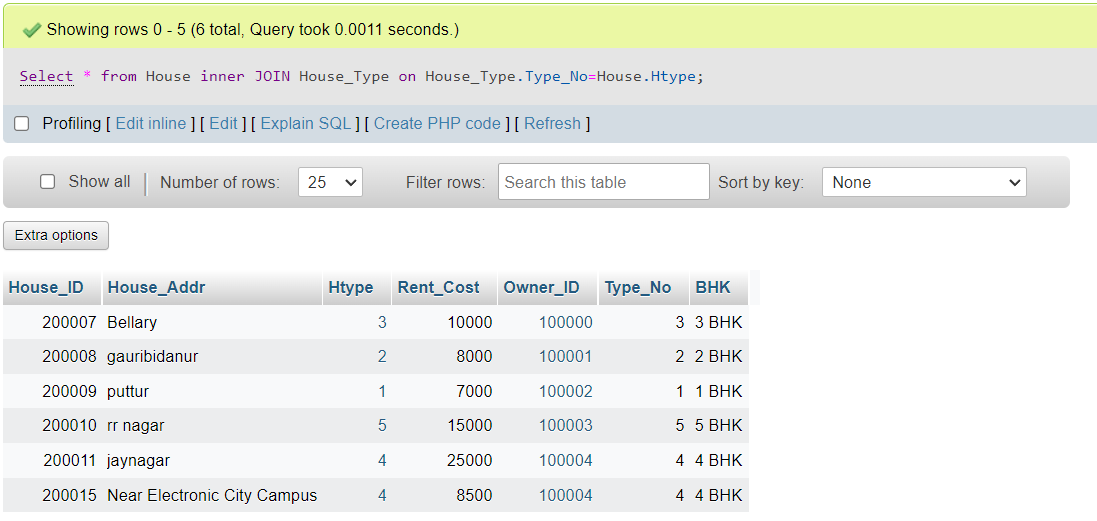
Select \* from Owner inner JOIN Owner\_Ph on Owner.Owner\_ID=Owner\_Ph.Owner\_ID;



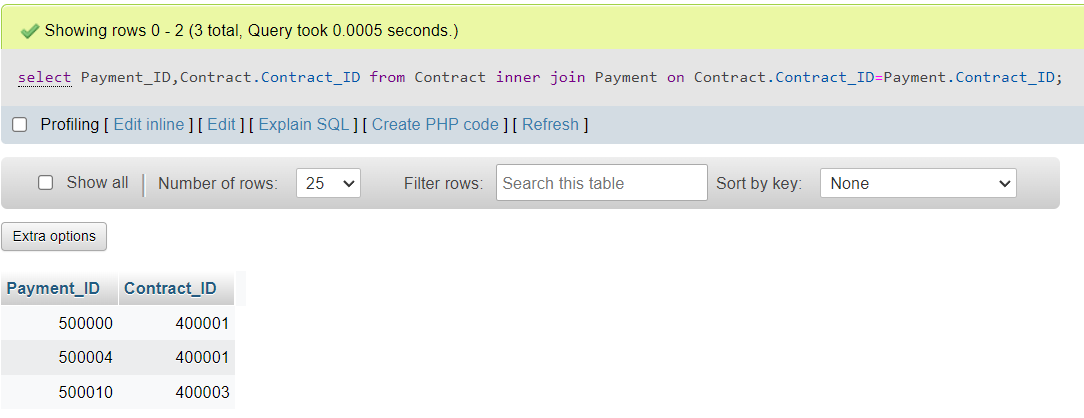
Select \* from Tenant inner JOIN Tenant\_Ph on Tenant.Tenant\_ID=Tenant\_Ph.Tenant\_ID;



Select \* from House inner JOIN House\_Type on House\_Type.Type\_No=House.Htype;



select Payment\_ID,Contract.Contract\_ID from Contract inner join Payment on Contract.Contract\_ID=Payment.Contract\_ID;



# Aggregate Functions

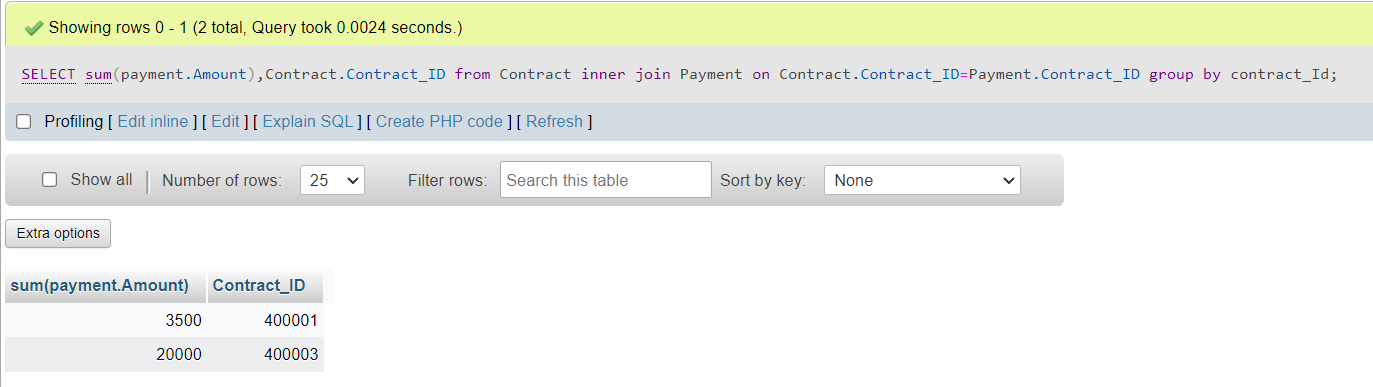
Showcase at least 4 Aggregate function queries

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

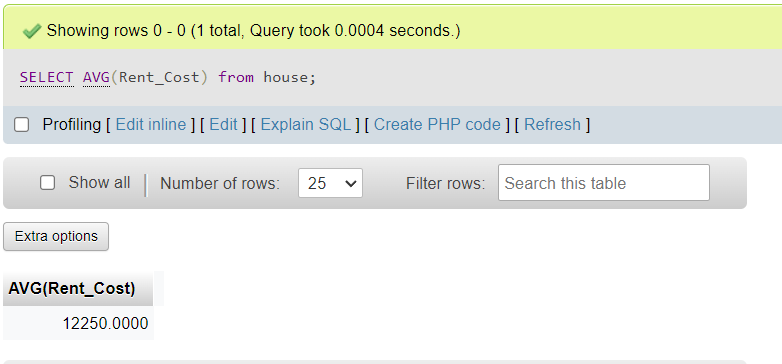
select count(Payment\_ID),Contract.Contract\_ID from Contract inner join Payment on Contract.Contract\_ID=Payment.Contract\_ID group by Contract\_ID;



SELECT sum(payment.Amount),Contract.Contract\_ID from Contract inner join Payment on Contract.Contract\_ID=Payment.Contract\_ID group by contract\_Id;



SELECT AVG(Rent\_Cost) from house;



Select Contract.Contract\_ID,Contract.Start\_Date,Contract.Duration,Contract.Rent\_Cost\_Duration,Contract.Owner\_ID,Contract.Tenant\_ID,Contract.House\_ID,count(Payment.Payment\_ID),sum(Payment.Amount),rent\_remain(sum(Payment.Amount),Rent\_Cost\_Duration) from Contract inner join Payment on Contract.Contract\_ID=Payment.Contract\_ID group by Contract\_ID;



# Set Operations

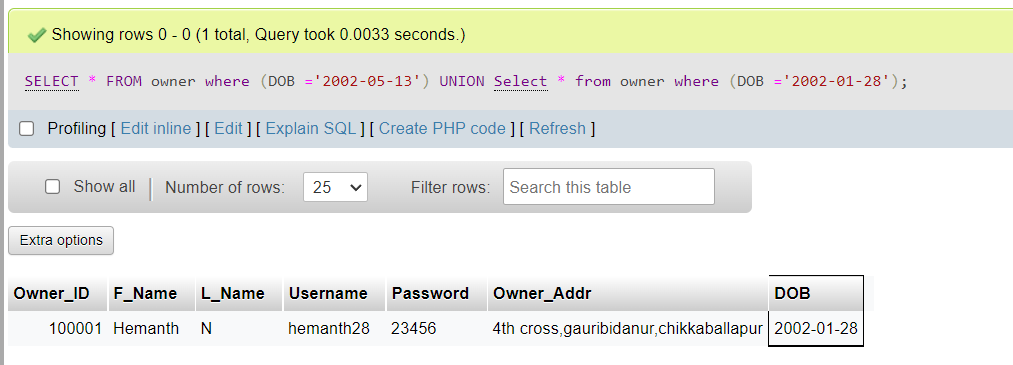
Showcase at least 4 Set Operations queries

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

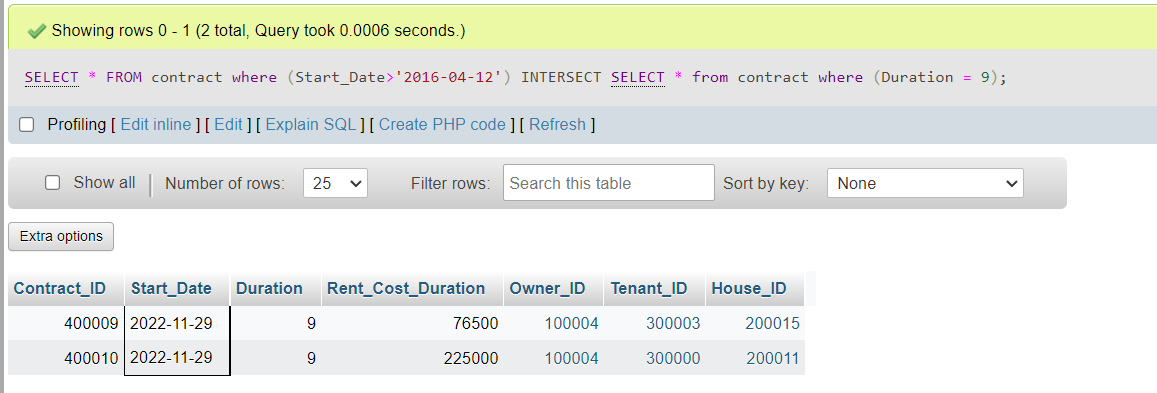
SELECT \* FROM house where (Rent\_Cost > 10000) UNION Select \* from house where (Rent\_Cost<8000);



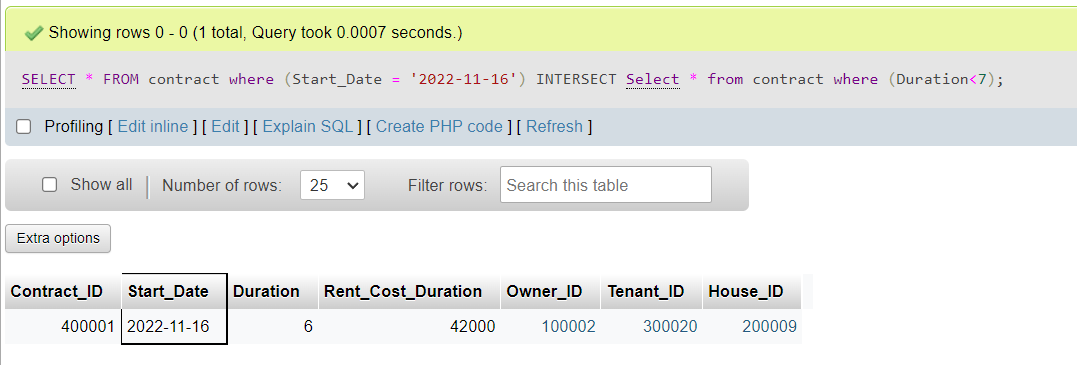
SELECT \* FROM owner where (DOB ='2002-05-13') UNION Select \* from owner where (DOB ='2002-01-28');



SELECT \* FROM contract where (Start\_Date>'2016-04-12') INTERSECT SELECT \* from contract where (Duration = 9);



SELECT \* FROM contract where (Start\_Date = '2022-11-16') INTERSECT Select \* from contract where (Duration<7);



# Functions and Procedures

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

Functions:

DELIMITER $

CREATE FUNCTION rent\_remain(Amount double,Total double)

  RETURNS double

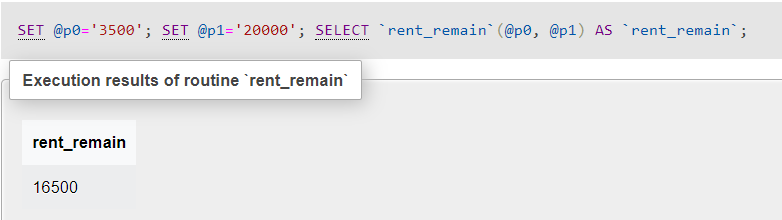
   DETERMINISTIC

    BEGIN

        return Total-Amount;

    END $

delimiter ;



DELIMITER $

CREATE FUNCTION totalrent(duration int,Total double)

  RETURNS double

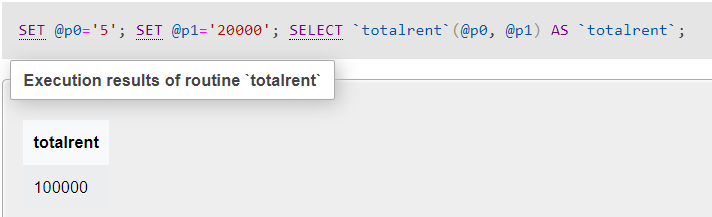
   DETERMINISTIC

    BEGIN

        return duration\*Total;

    END $

delimiter ;



Procedures:

delimiter $

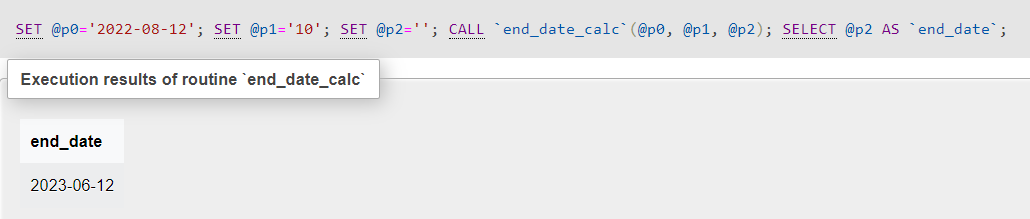
create Procedure end\_date\_calc(IN start\_date date,IN duration int, OUT end\_date date)

begin

    select date\_add(`start\_date`, INTERVAL `duration` MONTH) into end\_date;

end $

delimiter ;



create Procedure age\_dob(IN dob date, OUT age int)

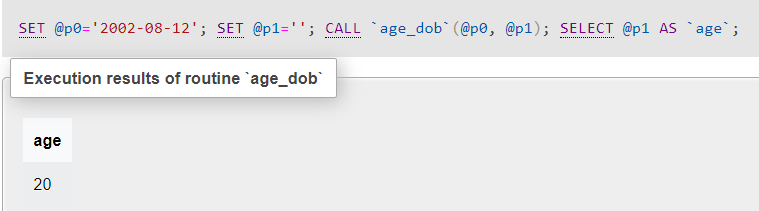
begin

    SELECT DATE\_FORMAT(FROM\_DAYS(DATEDIFF(NOW(), dob)), '%Y') + 0 AS age

into age;

end $

delimiter ;



# Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results.

**Cursor**:

For creating a backup:

DELIMITER $$

CREATE PROCEDURE backup\_member()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE Owner\_ID INTEGER;

DECLARE Start\_Date DATE;

DECLARE cur CURSOR FOR SELECT Owner\_ID,start\_date FROM contract ;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

label: LOOP

FETCH cur INTO Owner\_ID,Start\_Date;

INSERT INTO backup VALUES(Owner\_ID,Start\_Date);

IF done = 1 THEN LEAVE label;

END IF;

END LOOP;

CLOSE cur;

END$$

DELIMITER ;

--Creating backup table:

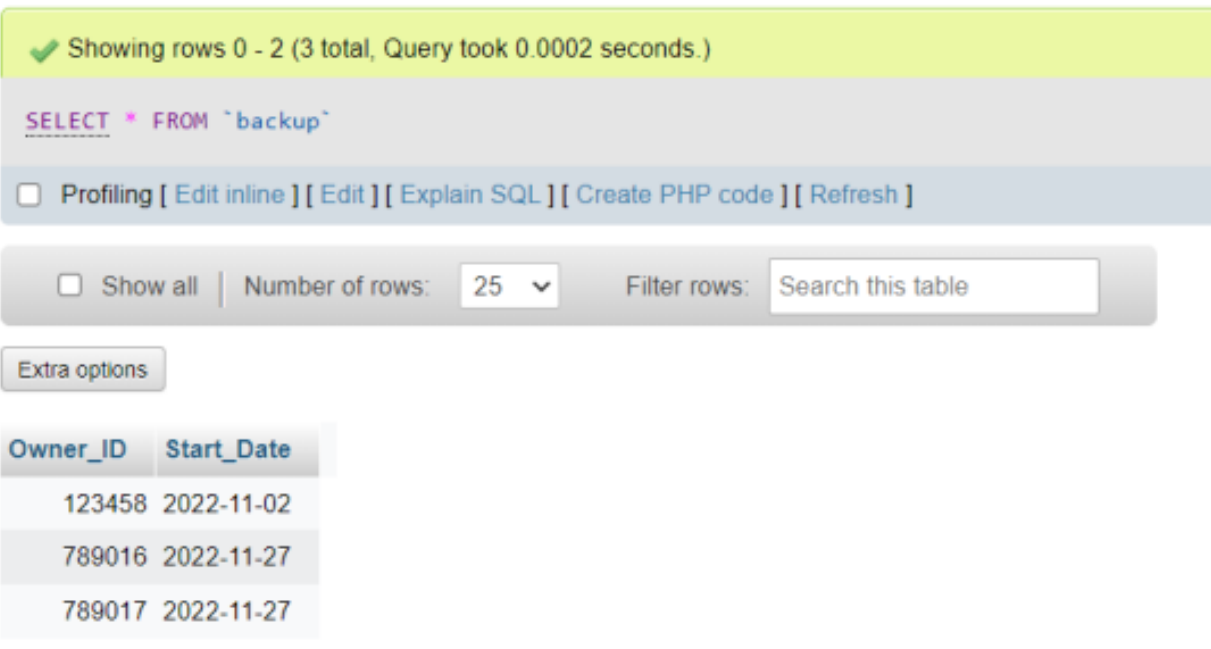
CREATE TABLE backup(Owner\_ID int,Start\_Date DATE);

-- CALL THE PROCEDURE:

CALL backup\_member();

--SHOW THE Backup TABLE created due to this cursor :

SELECT \* FROM backup;



**Trigger:**

c.execute(" drop trigger if exists amount\_exceed")

        qrystr = ("""  delimiter // \n"

                "  CREATE TRIGGER mytrigger BEFORE INSERT ON  Payment FOR EACH ROW \n"

                "  BEGIN \n"

                "   IF (select sum(Amount) from Payment group by Contract\_ID={}) > {} THEN \n"

                "      SIGNAL SQLSTATE '45000' SET message\_text="Amount can't exceed total Rent Cost";\n"

                "   END IF;\n"

                "  END;//\n"

                "  delimiter ; \n""".format(Contract\_ID,totalrent))

        c.execute(qrystr)



# Developing a Frontend

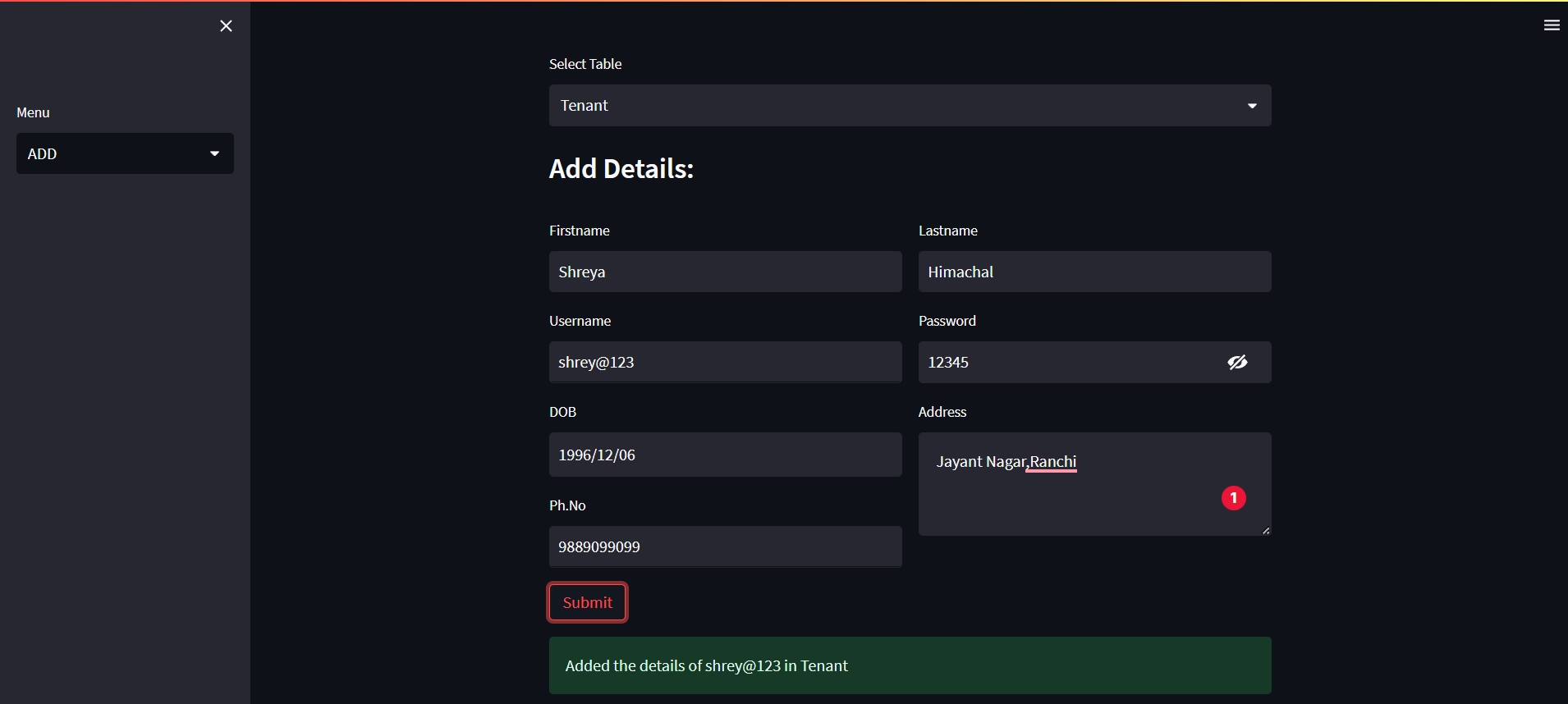
The frontend should support

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

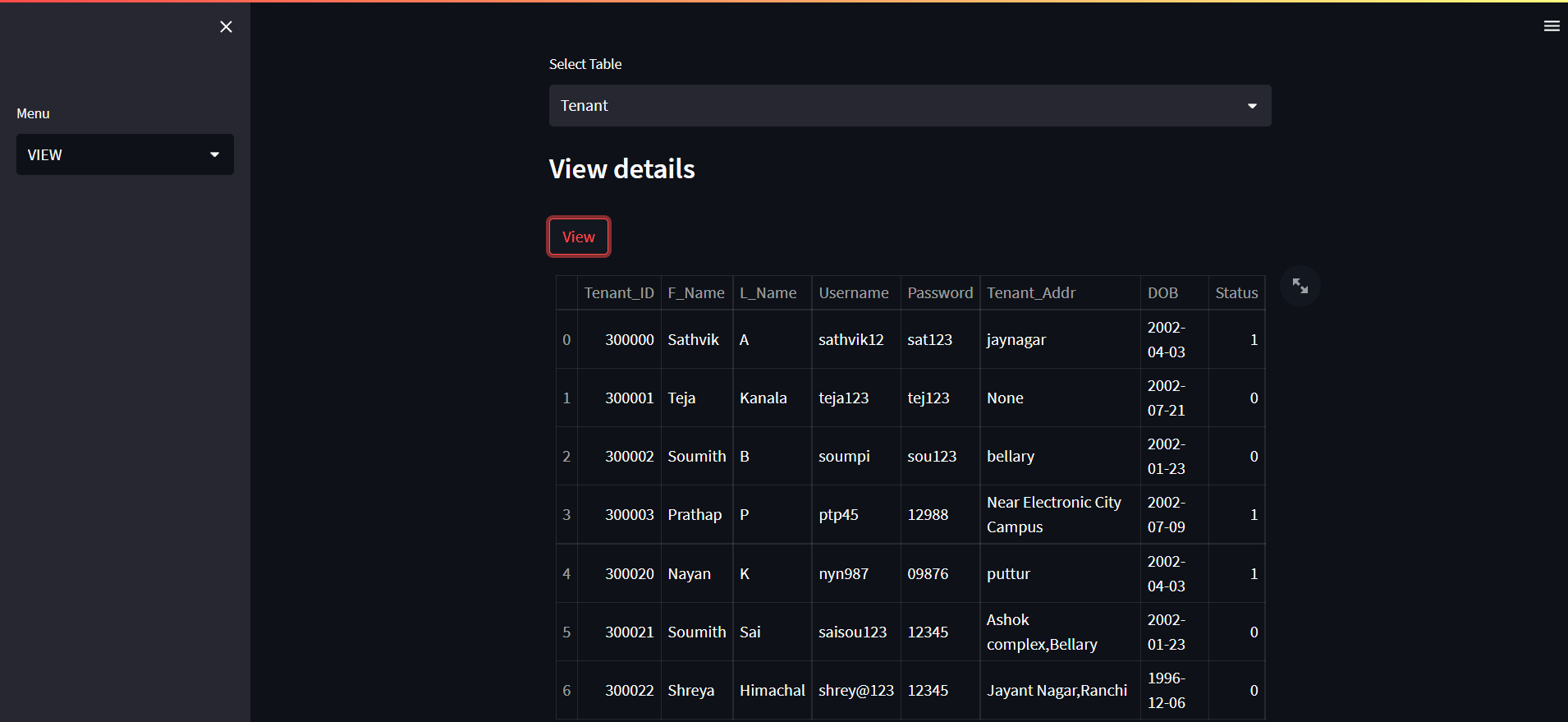
**FrontEnd**

****

**Addition**

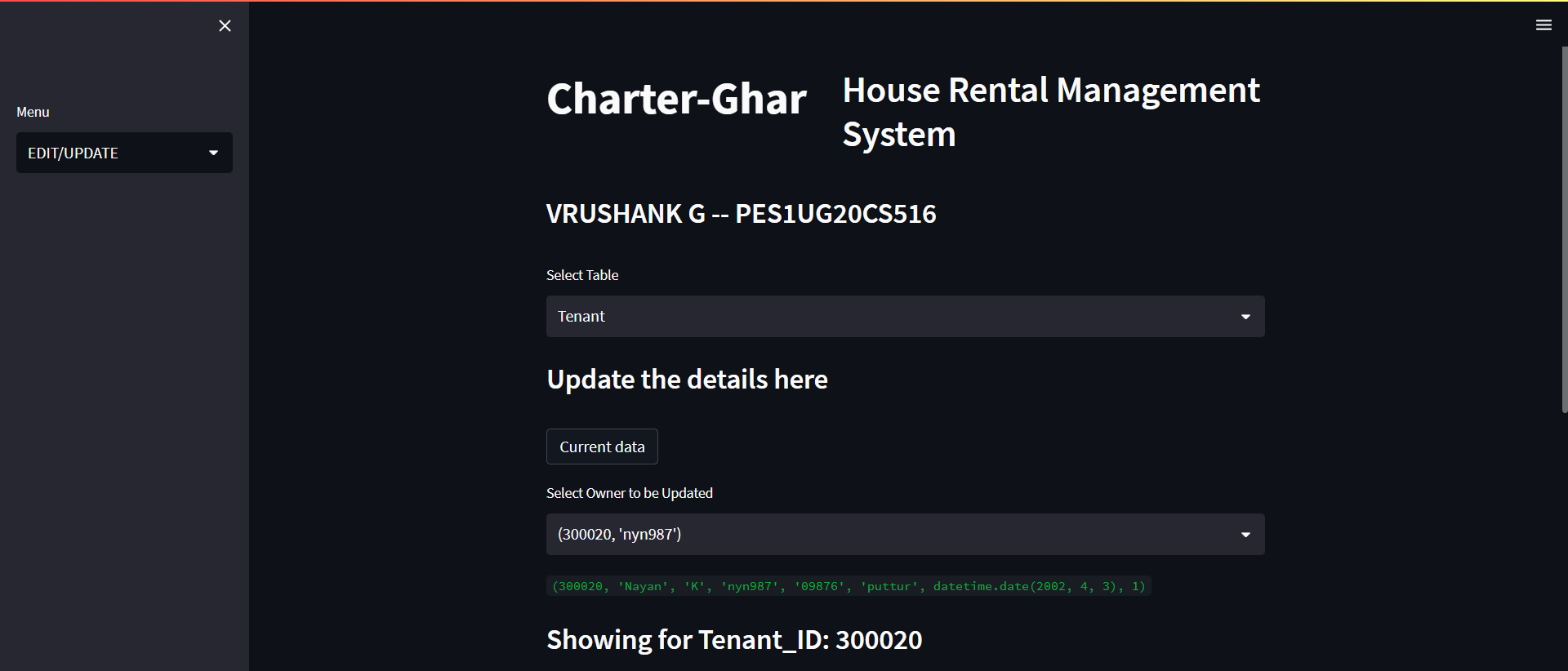
****

**Reading**

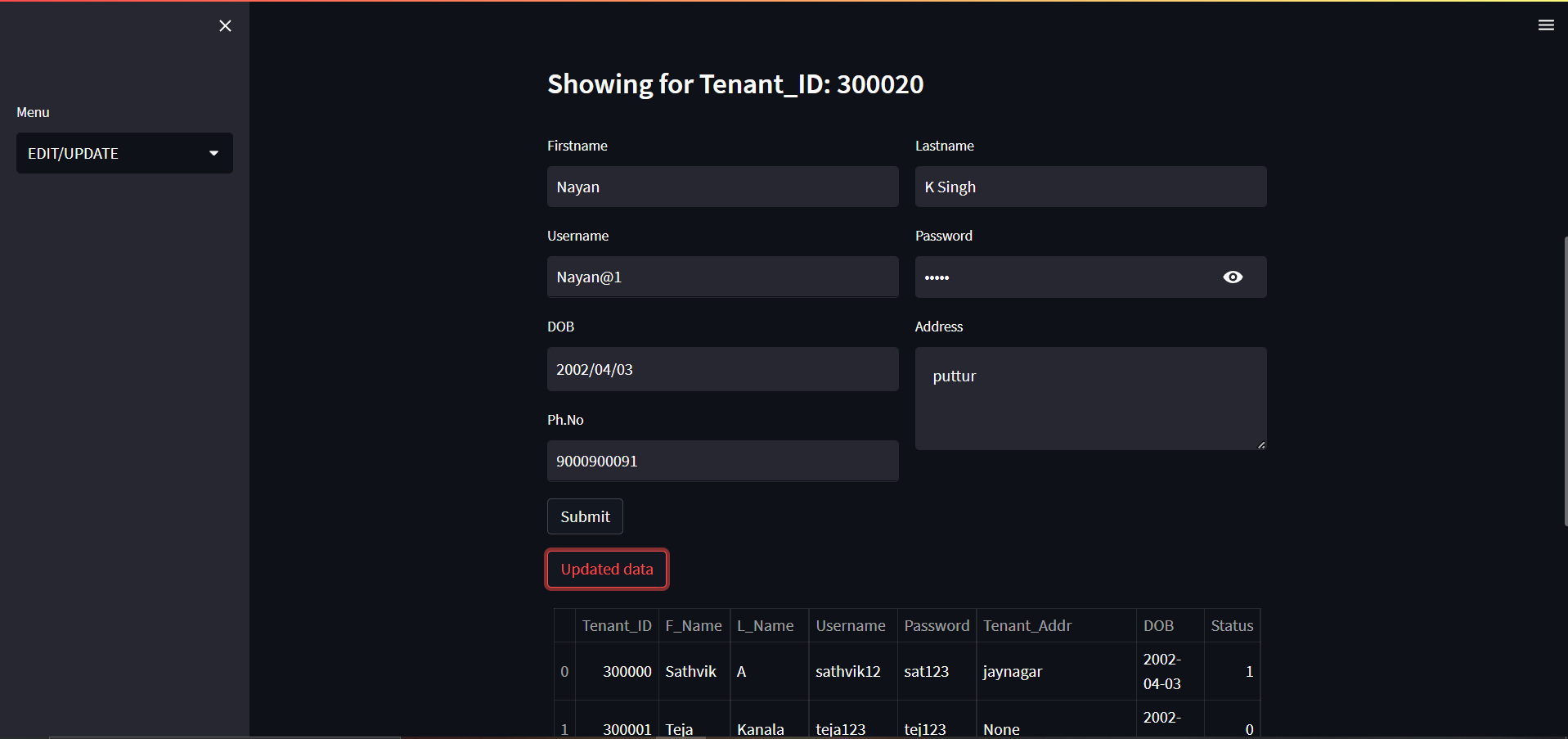
****

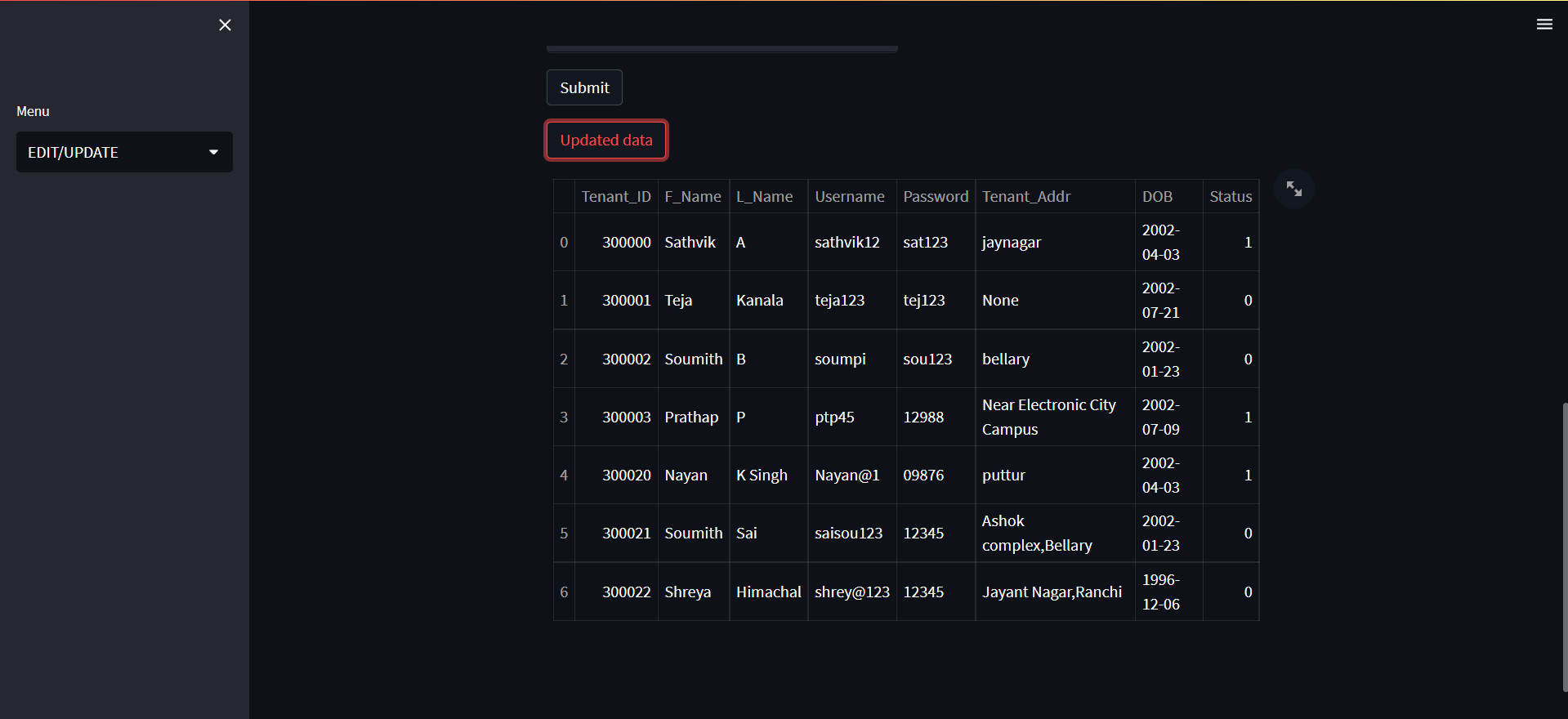
**Updation**

Before Update:



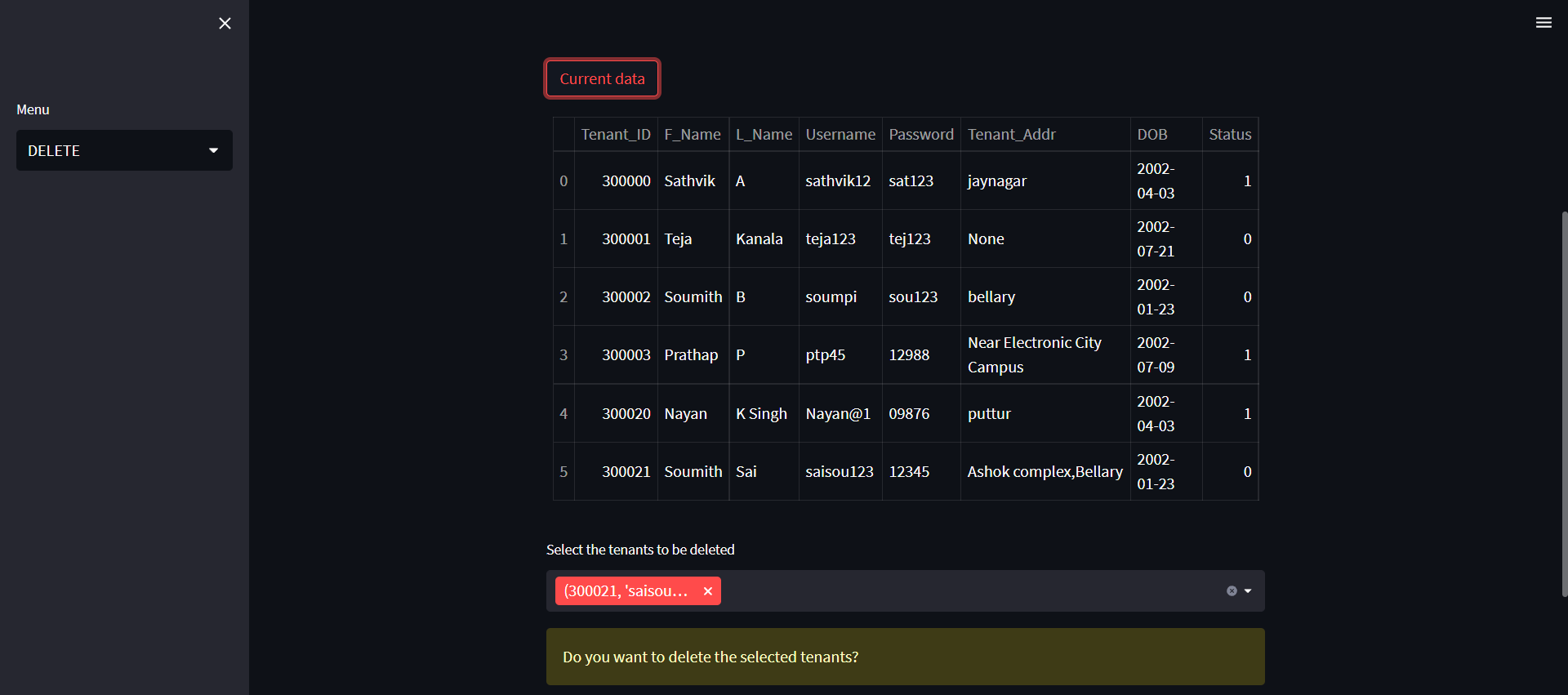
After Update:



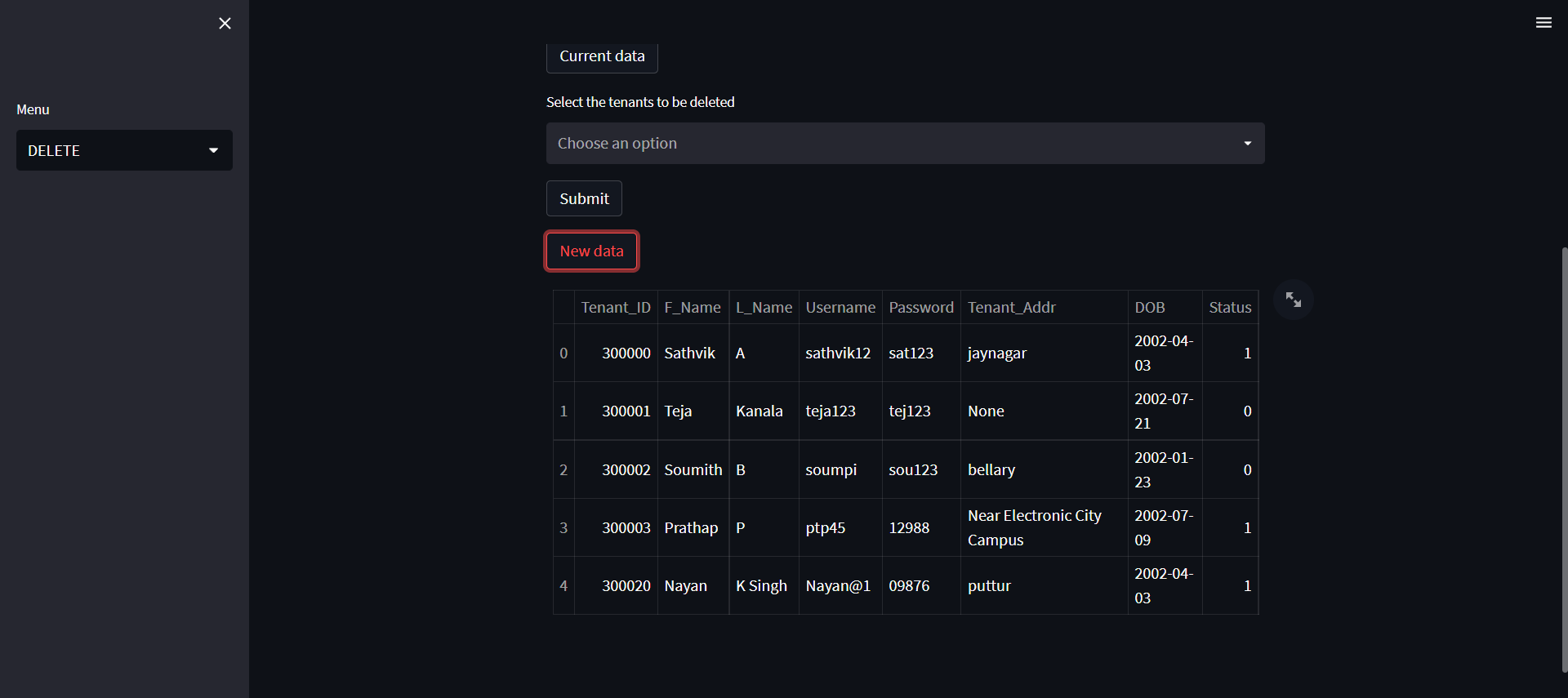


**Deletion:**

Before deletion:

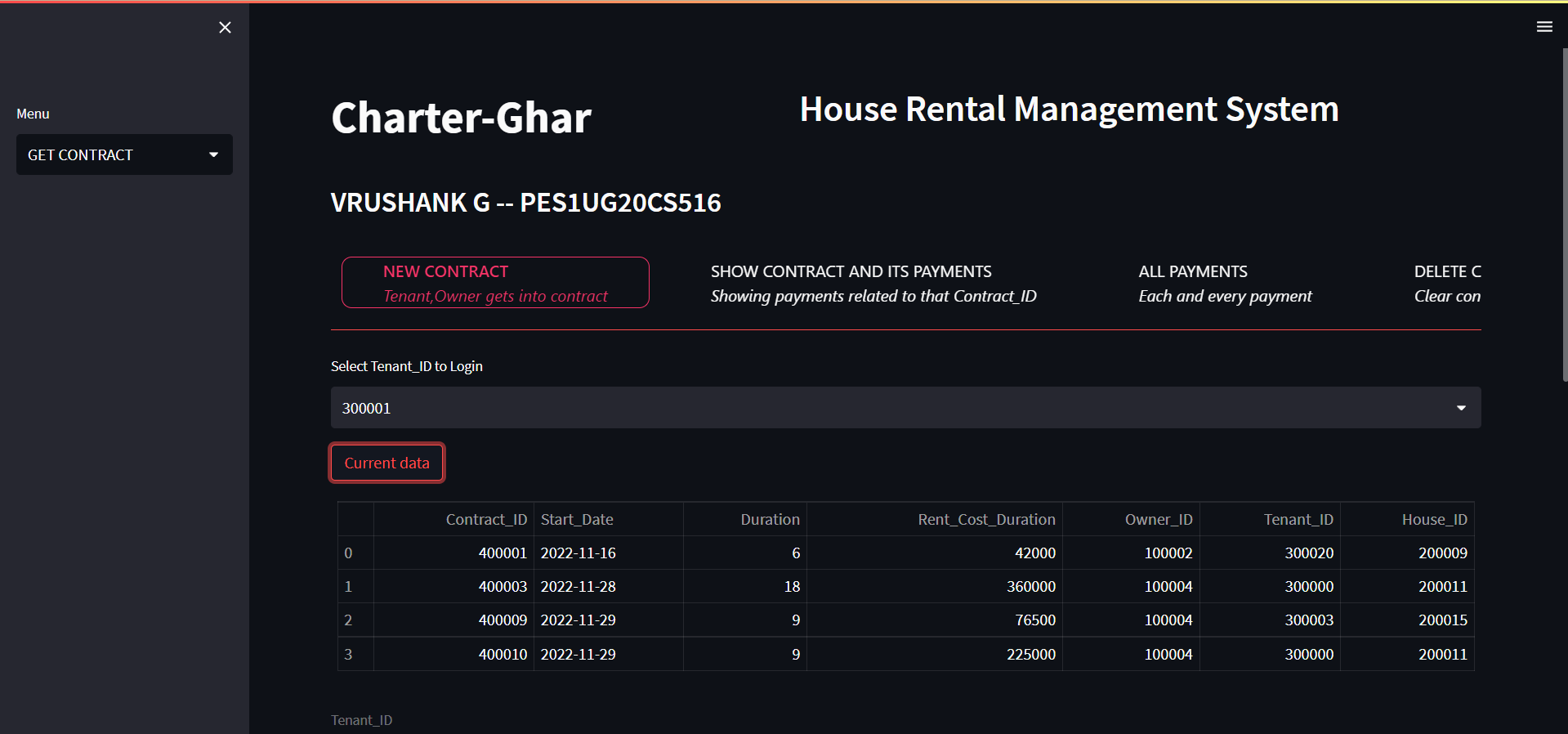


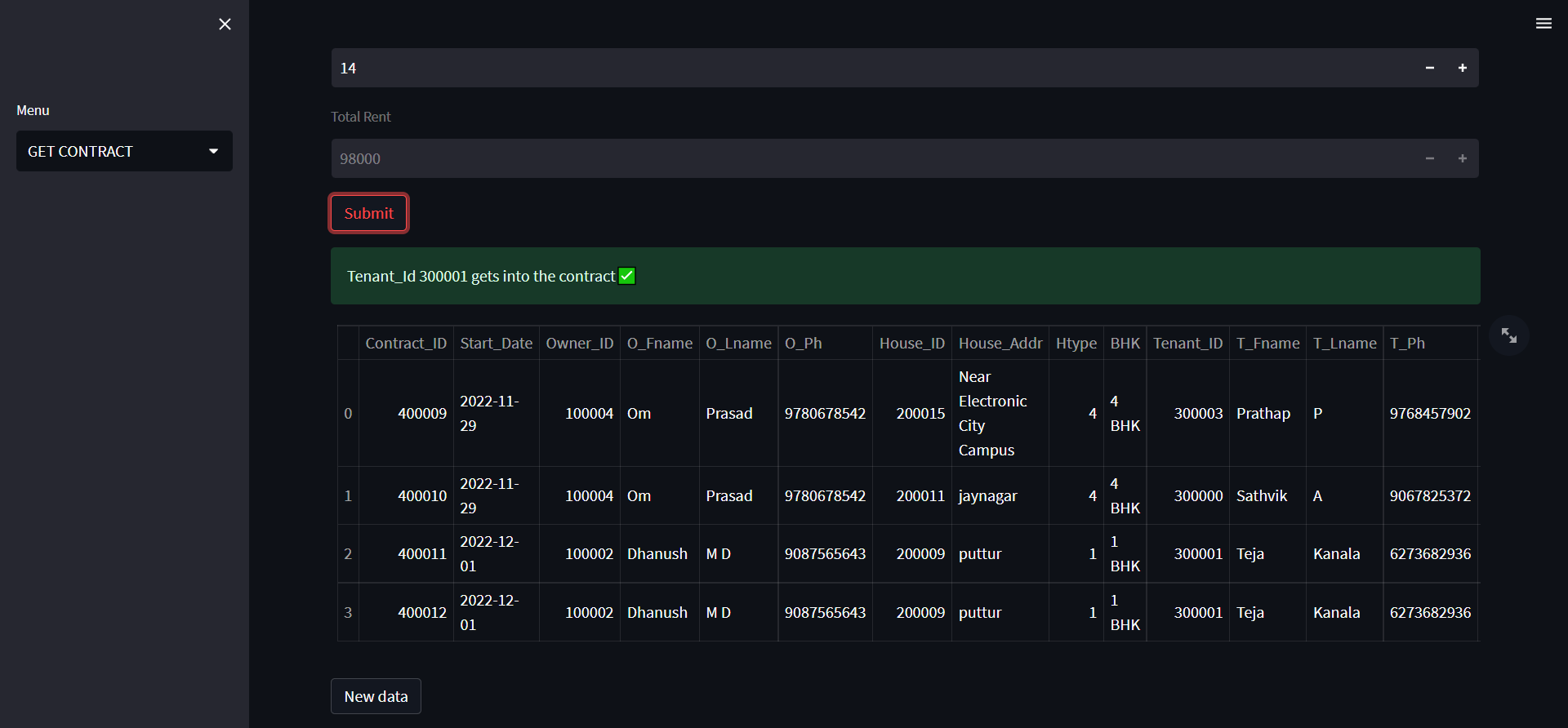
After deletion:

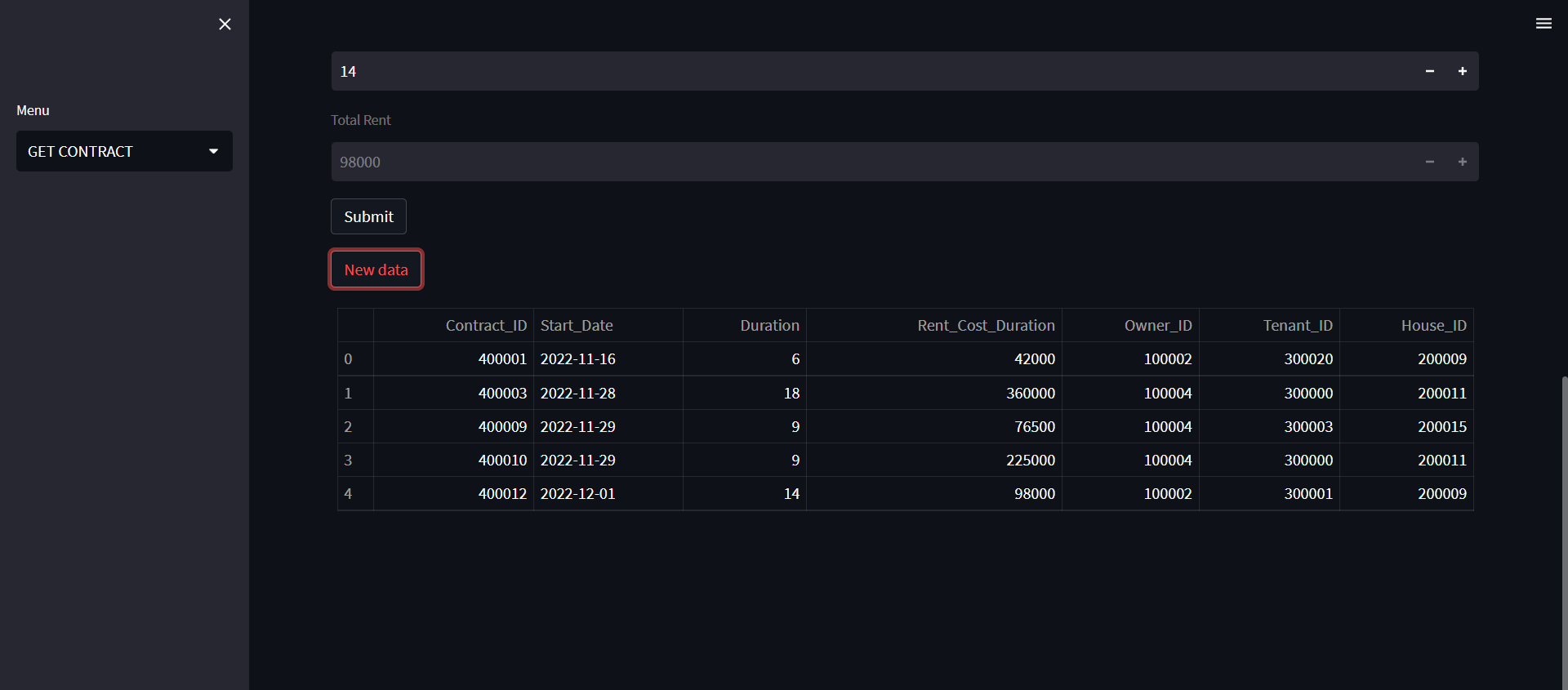


**Other Functional Features**

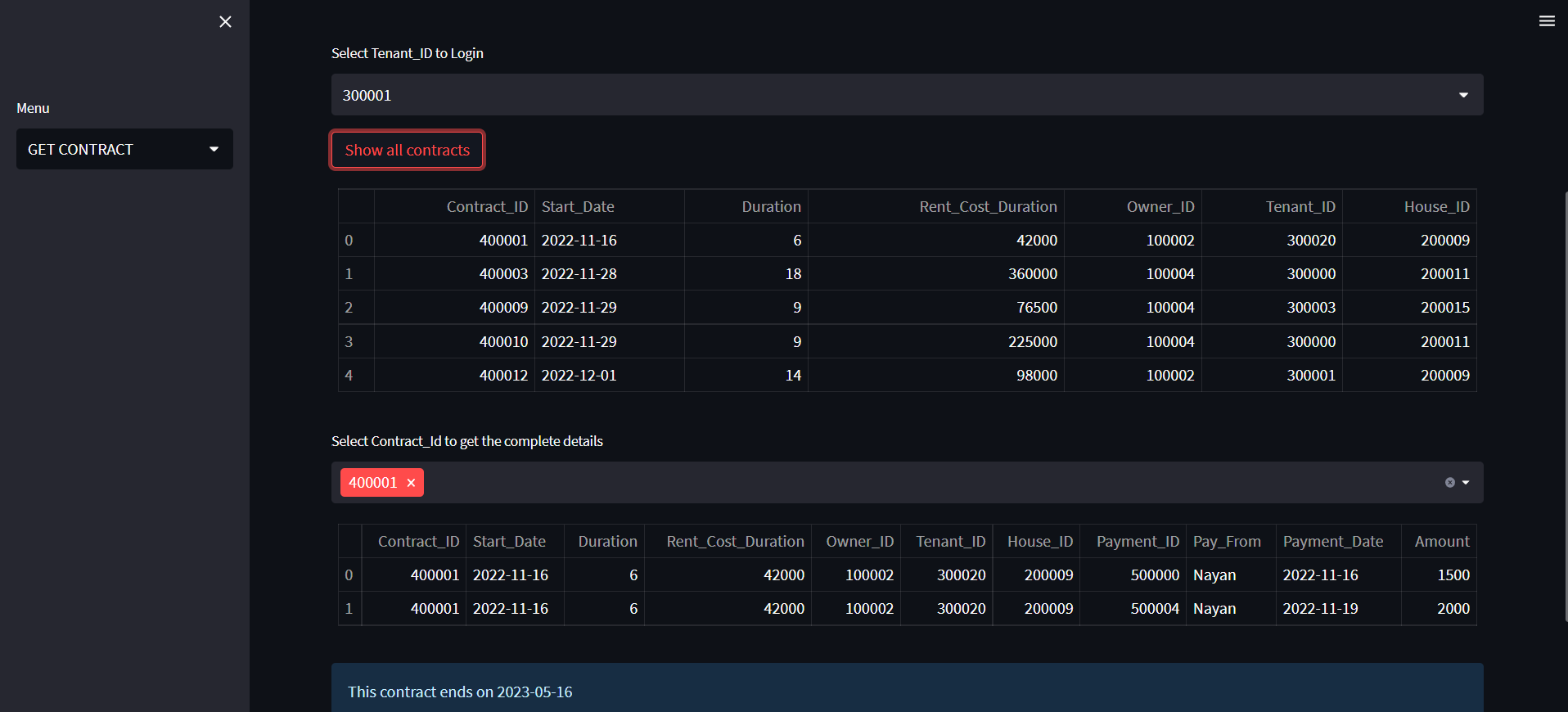
Get a new Contract:



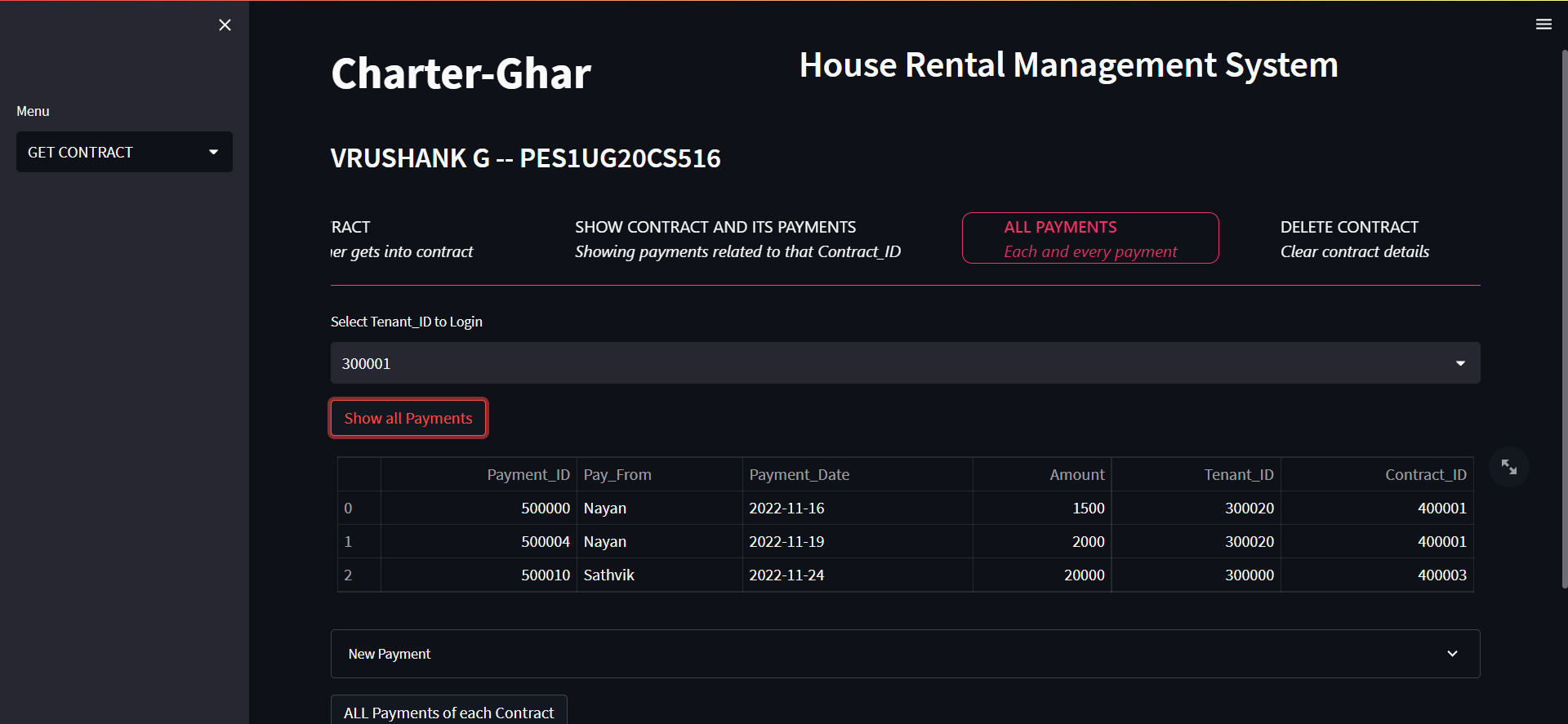




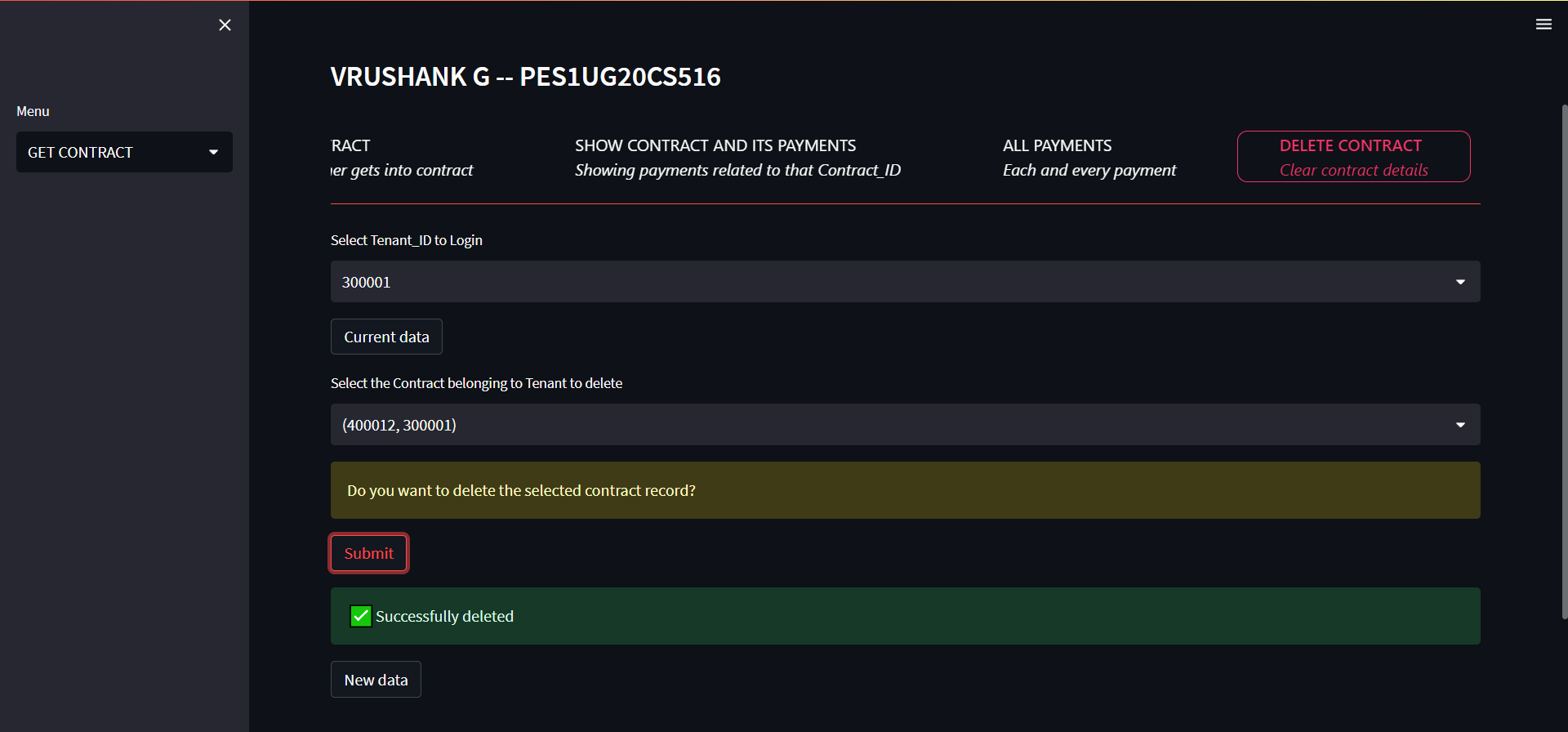
Show Contract and Payment details of that contract



All payments



Delete Contract and related Payments



1. There should be an window to accept and run any SQL statement and display the result



3.Modification suggested:

Trigger to display all the Contract details including Tenant details,House details,Owner details

Clearly after inserting into contract table

