**DBMS - MINI PROJECT**

**Fitness Management System**

Submitted By:

Name : Ghanashyam Mahesh Bhat

SRN : PES1UG20CS153

V Semester Section C

**ABSTRACT**

Fitness is an important part of the life, and the Fitness management system helps the user maintain the fitness and good health with the structured information on healthy diet and exercise routine.

The user can subscribe to membership which gives the access to personal trainer. The personal trainer can assign the routine and diet plan; when user follows the assigned routine, he will be awarded with streaks and level ups.

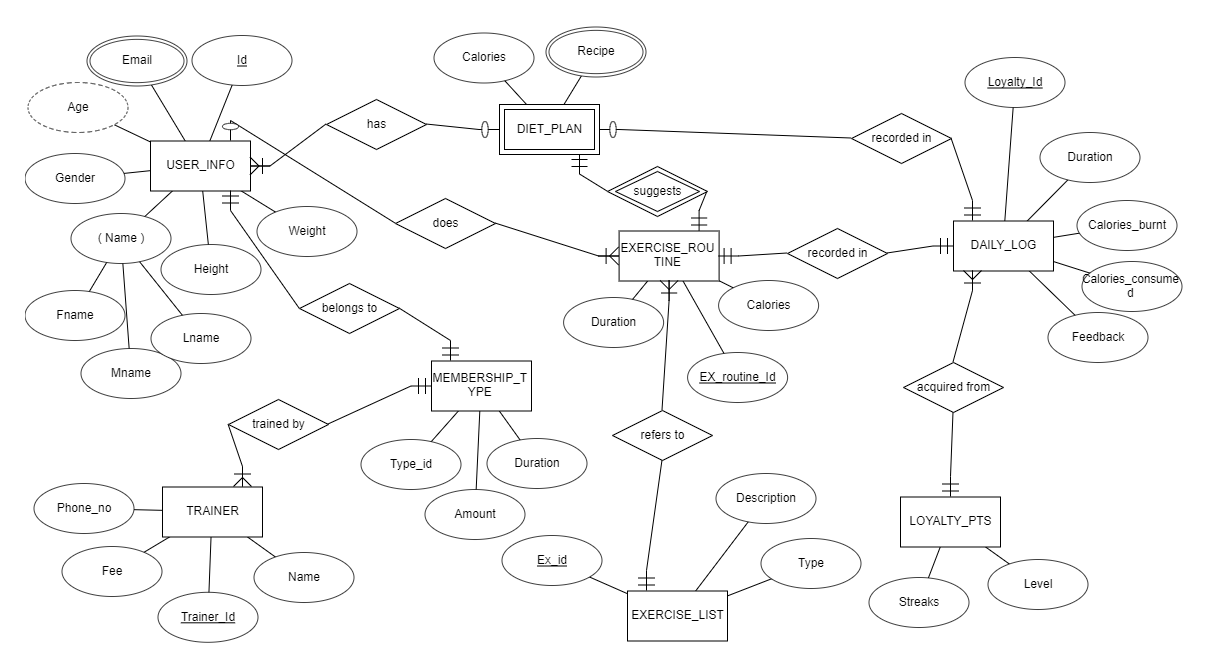
The streaks and levels helps the user to keep the motivation and work towards better health.

With the help of this tool user can get access to personalized diet plan and workout plan. Daily workout logs will be stored by the database which can further be used to do some analytics to obatin useful insights

The project servers as a platform for the user to get certified directions in the fitness journey and serve as tracker for maintaing the fitness and health.

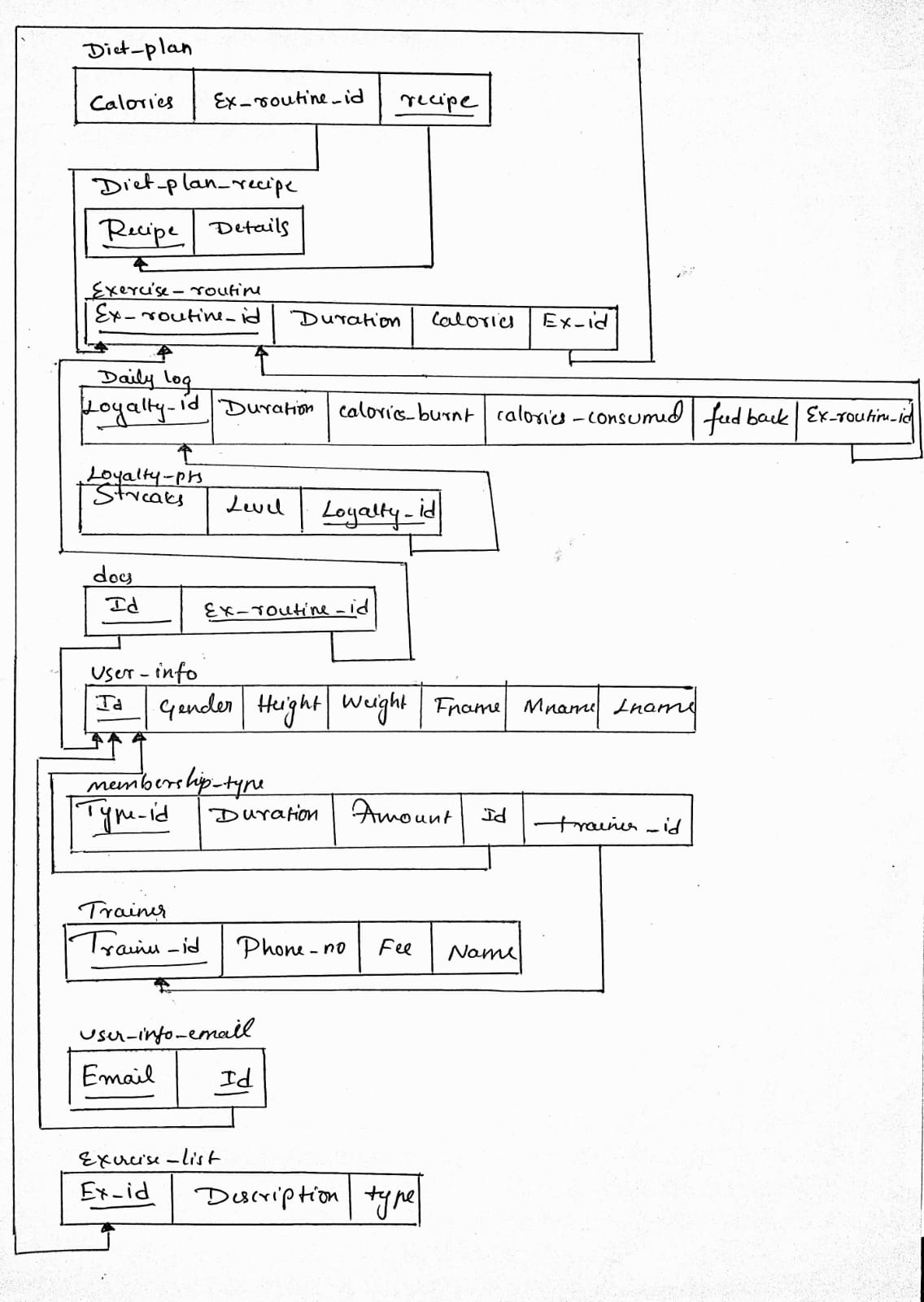
.

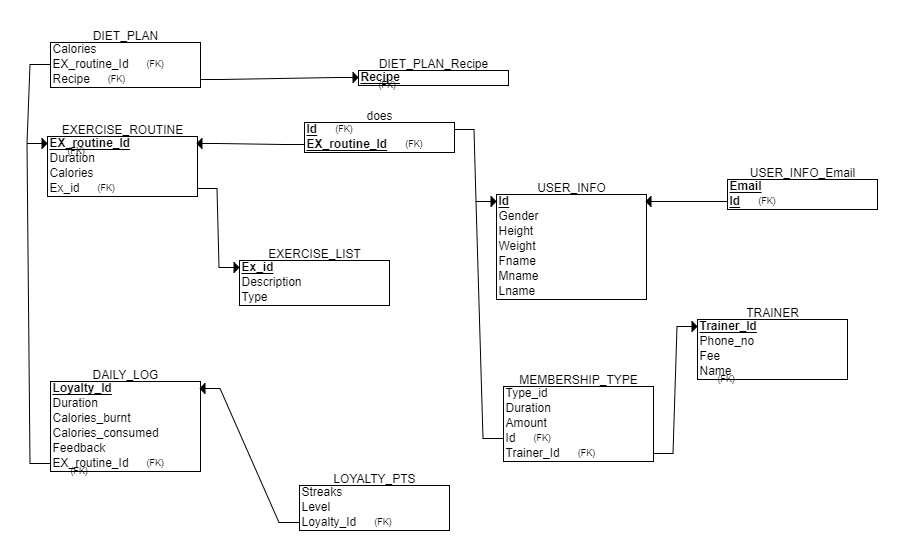
**ER Diagram**



The ER diagram shows the 8 entities and their interrelations. Each entity has different kind of attributes defining the charecteristics of the entity.

**Relational Schema**



**DDL statements - Building the database**

CREATE DATABASE fitnessmanagement;

USE fitnessmanagement;

CREATE TABLE diet\_plan (

  ex\_id varchar(20) NOT NULL ,

  recipe varchar(30) DEFAULT NULL,

  calories int(10) DEFAULT NULL

);

CREATE TABLE exercise\_routine (

  ex\_routine\_id varchar(20) NOT NULL,

  duration int NOT NULL,

  calories int(10) DEFAULT NULL,

  ex\_id varchar(20) NOT NULL

);

CREATE TABLE daily\_log (

  loyalty\_id varchar(20) NOT NULL,

  duration int NOT NULL,

  calories\_burnt int(10) DEFAULT NULL,

  calories\_consumed int(10) DEFAULT NULL,

  feedback varchar(20) DEFAULT NULL,

  ex\_routine\_id varchar(20) NOT NULL

);

CREATE TABLE diet\_plan\_recipe (

  recipe varchar(30) DEFAULT NULL

);

CREATE TABLE does (

  id varchar(20) NOT NULL,

  ex\_routine\_id varchar(20) NOT NULL

);

CREATE TABLE exercise\_list (

  ex\_id varchar(20) NOT NULL ,

  description varchar(30) DEFAULT NULL,

  type varchar(10) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE loyalty\_pts (

  loyalty\_id varchar(20) NOT NULL ,

  level int(3) NOT NULL,

  streaks int(10) NOT NULL DEFAULT 0

);

CREATE TABLE user\_info(

  id varchar(20) NOT NULL,

  PRIMARY KEY(id),

  gender int(2) NOT NULL,

  height int NOT NULL,

  weight int NOT NULL,

  fname varchar(20) NOT NULL,

  mname varchar(20) DEFAULT NULL,

  lname varchar(20) DEFAULT NULL

);

CREATE TABLE membership\_type (

  type\_id varchar(20) NOT NULL,

  duration int NOT NULL,

  amount decimal(10,2) NOT NULL,

  id varchar(20) NOT NULL ,

  trainer\_id varchar(20) NOT NULL

);

CREATE TABLE user\_info\_email (

  id varchar(20) NOT NULL ,

  email varchar(30) NOT NULL

);

CREATE TABLE trainer (

  trainer\_id varchar(20) NOT NULL,

  PRIMARY KEY (trainer\_id),

  phoneno int(10) NOT NULL,

  amount decimal(10,2) NOT NULL,

  name varchar(20) NOT NULL

);

ALTER TABLE diet\_plan

  ADD KEY ex\_id (ex\_id),

  ADD KEY recipe (recipe);

ALTER TABLE exercise\_routine

  ADD PRIMARY KEY (ex\_routine\_id),

  ADD KEY ex\_id (ex\_id);

ALTER TABLE daily\_log

  ADD PRIMARY KEY (loyalty\_id),

  ADD KEY ex\_routine\_id (ex\_routine\_id);

ALTER TABLE diet\_plan\_recipe

  ADD KEY recipe (recipe);

ALTER TABLE does

  ADD PRIMARY KEY (id,ex\_routine\_id),

  ADD KEY id (id),

  ADD KEY ex\_routine\_id (ex\_routine\_id);

ALTER TABLE exercise\_list

  ADD PRIMARY KEY (ex\_id);

ALTER TABLE loyalty\_pts

  ADD KEY (loyalty\_id);

ALTER TABLE membership\_type

  ADD PRIMARY KEY (type\_id),

  ADD KEY id (id),

  ADD KEY trainer\_id (trainer\_id);

ALTER TABLE user\_info\_email

  ADD PRIMARY KEY (id,email),

  ADD KEY id (id);

ALTER TABLE diet\_plan

  ADD CONSTRAINT diet\_plan\_ibfk\_1 FOREIGN KEY (ex\_id) REFERENCES exercise\_routine (ex\_routine\_id);

ALTER TABLE diet\_plan

  ADD CONSTRAINT diet\_plan\_ibfk\_2 FOREIGN KEY (recipe) REFERENCES diet\_plan\_recipe (recipe);

ALTER TABLE exercise\_routine

  ADD CONSTRAINT exercise\_routine\_ibfk\_1 FOREIGN KEY (ex\_id) REFERENCES exercise\_list (ex\_id);

ALTER TABLE does

  ADD CONSTRAINT does\_ibfk\_1 FOREIGN KEY (ex\_routine\_id) REFERENCES exercise\_routine (ex\_routine\_id);

ALTER TABLE daily\_log

  ADD CONSTRAINT daily\_log\_ibfk\_1 FOREIGN KEY (ex\_routine\_id) REFERENCES exercise\_routine (ex\_routine\_id);

ALTER TABLE loyalty\_pts

  ADD CONSTRAINT loyalty\_pts\_ibfk\_1 FOREIGN KEY (loyalty\_id) REFERENCES daily\_log (loyalty\_id);

ALTER TABLE does

  ADD CONSTRAINT does\_ibfk\_2 FOREIGN KEY (id) REFERENCES user\_info (id);

ALTER TABLE user\_info\_email

  ADD CONSTRAINT user\_info\_email\_ibfk\_1 FOREIGN KEY (id) REFERENCES user\_info (id);

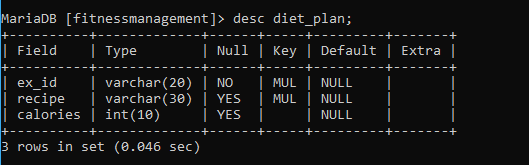
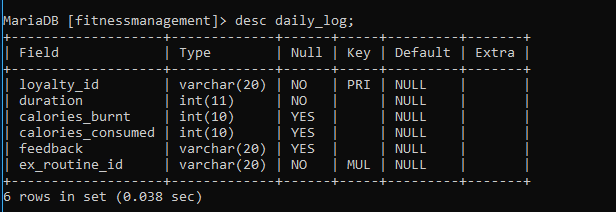
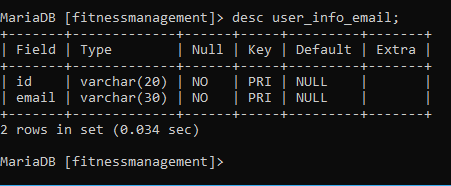
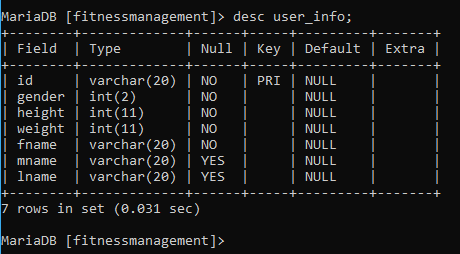
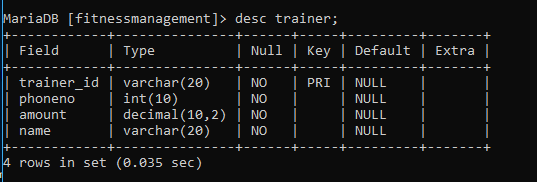
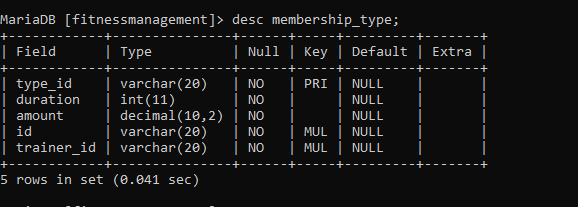
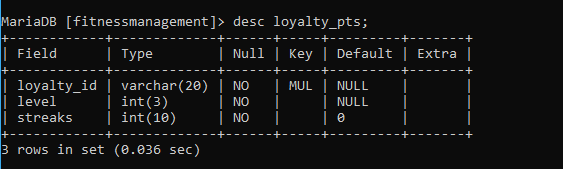
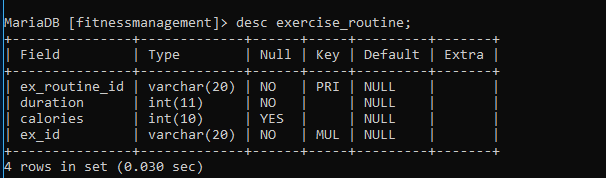
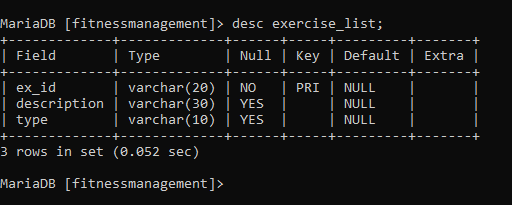
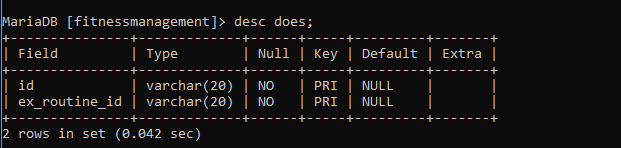
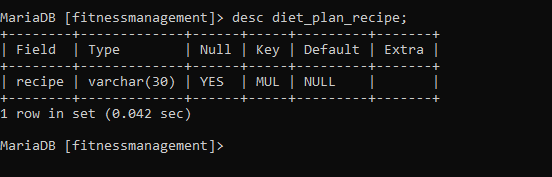
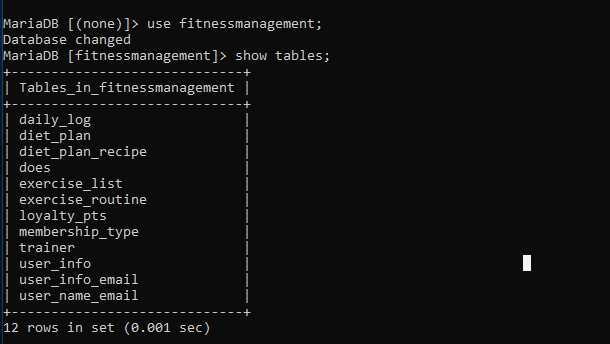
ALTER TABLE membership\_type

  ADD CONSTRAINT membership\_type\_ibfk\_1 FOREIGN KEY (id) REFERENCES user\_info (id);

ALTER TABLE membership\_type

  ADD CONSTRAINT membership\_type\_ibfk\_2 FOREIGN KEY (trainer\_id) REFERENCES trainer (trainer\_id);

The table list and their description after the execution of DDL commands.



**Populating the Database**

The population of data is done by the below python code which exports CSV files with data for all the tables.

These CSV files are imported into tables using phpmyadmin.

import csv

from random import randint as r

class dataGeneration:

    def \_\_init\_\_(self):

        self.l1 = ['Calories','EX\_Id',"Recipe"]

        self.l2 = ["Ex\_routine\_Id",'Duration',"Calories","Ex\_id"]

        self.l3 = ['Loyalty\_Id',"Duration","Calories\_burnt","Calories\_consumed","Feedback","Ex\_routine\_Id"]

        self.l4 = ["Recipe"]

        self.l5 = ["Id","Ex\_routine\_Id"]

        self.l6 = ["Ex\_id","Description","Type"]

        self.l7 = ["Streaks","Level" , "Loyalty\_Id"]

        self.l8 = ["Id","Gender","Height","Weight","Fname","Mname","Lname"]

        self.l9 = ["Type\_id","Duration","Amount","Id","Trainer\_Id"]

        self.l10 = ["Email","Id"]

        self.l11 = ["Trainer\_Id","Phoneno","amount","Name"]

    def diet\_plan(self,count,ex\_routine\_min,ex\_routine\_max,cal\_min,cal\_max,recipe\_min,recipe\_max):

        #f1

        result = []

        for i in range(count):

            result.append([r(cal\_min,cal\_max),'EX\_routine'+str(r(ex\_routine\_min,ex\_routine\_max)),"Recipe"+str(r(recipe\_min,recipe\_max))])

        return result

    def exercise\_routine(self,count,duration\_min,duration\_max,cal\_min,cal\_max,ex\_id\_min,ex\_id\_max):

        #f2

        result = []

        for i in range(count):

            result.append(['EX\_routine'+str(i),r(duration\_min,duration\_max),r(cal\_min,cal\_max),"EX\_ID"+str(r(ex\_id\_min,ex\_id\_max))])

        return result

    def daily\_log(self,count,duration\_min,duration\_max,cal\_min,cal\_max,cal\_consume\_min,cal\_consume\_max,feedback\_min,feedback\_max,ex\_routine\_min,ex\_routine\_max):

        #f3

        result = []

        for i in range(count):

            result.append(["LYL"+str(i),r(duration\_min,duration\_max),r(cal\_min,cal\_max),r(cal\_consume\_min,cal\_consume\_max),"Feedback"+str(r(feedback\_min,feedback\_max),),'EX\_routine'+str(r(ex\_routine\_min,ex\_routine\_max))])

        return result

    def diet\_plan\_recipe(self,count):

        #f4

        result = []

        for i in range(count):

            result.append(["Recipe"+str(i)])

        return result

    def does(self,count,user\_id\_min,user\_id\_max,ex\_routine\_min,ex\_routine\_max):

        #f5

        result = []

        check = []

        for i in range(count):

            id = "USR"+str(r(user\_id\_min,user\_id\_max))

            ex\_r\_id = 'EX\_routine'+str(r(ex\_routine\_min,ex\_routine\_max))

            if (id,ex\_r\_id) in check:

                pass

            else:

                result.append([id,ex\_r\_id])

                check.append((id,ex\_r\_id))

        return result

    def exercise\_list(self,count,description\_min,description\_max,type\_min,type\_max):

        #f6

        result = []

        for i in range(count):

            result.append(["EX\_ID"+str(i),"Description"+str(r(description\_min,description\_max)),"Type"+str(r(type\_min,type\_max))])

        return result

    def loyalty\_pts(self,count,streaks\_min,streaks\_max,Level\_min,Level\_max,loyalty\_min,loyalty\_max):

        #f7

        result = []

        for i in range(count):

            result.append([r(streaks\_min,streaks\_max),r(Level\_min,Level\_max),"LYL"+str(r(loyalty\_min,loyalty\_max))])

        return result

    def user\_info(self,count,height\_min,height\_max,weight\_min,weight\_max,Fname\_min,Fname\_max,Mname\_min,Mname\_max,Lname\_min,Lname\_max):

        #f8

        gender\_min = 0

        gender\_max = 2

        result = []

        for i in range(count):

            result.append(["USR"+str(i),r(gender\_min,gender\_max),r(height\_min,height\_max),r(weight\_min,weight\_max),"Fname"+str(r(Fname\_min,Fname\_max)),"Mname"+str(r(Mname\_min,Mname\_max)),"Lname"+str(r(Lname\_min,Lname\_max))])

        return result

    def membership\_type(self,count,period\_min,period\_max,amount\_min,amount\_max,user\_id\_min,user\_id\_max,trainer\_id\_min,trainer\_id\_max):

        #f9

        result = []

        for i in range(count):

            result.append(["MEM"+str(i),r(period\_min,period\_max),r(amount\_min,amount\_max),"USR"+str(r(user\_id\_min,user\_id\_max)),"TRAINER"+str(r(trainer\_id\_min,trainer\_id\_max))])

        return result

    def user\_info\_email(self,count,email\_min,email\_max,user\_id\_min,user\_id\_max):

        #f10

        result = []

        check = list()

        for i in range(count):

            email = "email.id."+str(r(email\_min,email\_max))+"@email.com"

            id = "USR"+str(r(user\_id\_min,user\_id\_max))

            if (email,id) in check:

                pass

            else:

                result.append([email,id])

                check.append((email,id))

        return result

    def trainer(self,count,fee\_min,fee\_max,name\_min,name\_max,ph\_min=1000000000,ph\_max=9999999999):

        #f11

        result = []

        for i in range(count):

            result.append(["TRAINER"+str(i),r(ph\_min,ph\_max),r(fee\_min,fee\_max),"TrainerName"+str(r(name\_min,name\_max))])

        return result

if \_\_name\_\_=='\_\_main\_\_':

    dataGen = dataGeneration()

    filename = "diet\_plan.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l1)

        csvwriter.writerows(dataGen.diet\_plan(count=100,ex\_routine\_min=20,ex\_routine\_max=60,cal\_min=100,cal\_max=350,recipe\_min=5,recipe\_max=25))

    filename = "exercise\_routine.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l2)

        csvwriter.writerows(dataGen.exercise\_routine(count=200,duration\_min=30,duration\_max=120,cal\_min=50,cal\_max=400,ex\_id\_min=0,ex\_id\_max=120))

    filename = "daily\_log.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l3)

        csvwriter.writerows(dataGen.daily\_log(count=400,duration\_min=40,duration\_max=90,cal\_min=90,cal\_max=320,cal\_consume\_min=1000,cal\_consume\_max=2500,feedback\_min=0,feedback\_max=100,ex\_routine\_min=20,ex\_routine\_max=80))

    filename = "diet\_plan\_recipe.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l4)

        csvwriter.writerows(dataGen.diet\_plan\_recipe(count=80))

    filename = "does.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l5)

        csvwriter.writerows(dataGen.does(count=70,user\_id\_min=15,user\_id\_max=100,ex\_routine\_min=11,ex\_routine\_max=95))

    filename = "exercise\_list.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l6)

        csvwriter.writerows(dataGen.exercise\_list(count=150,description\_min=20,description\_max=300,type\_min=5,type\_max=300))

    filename = "loyalty\_pts.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l7)

        csvwriter.writerows(dataGen.loyalty\_pts(count=50,streaks\_min=0,streaks\_max=20,Level\_min=1,Level\_max=5,loyalty\_min=30,loyalty\_max=300))

    filename = "user\_info.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l8)

        csvwriter.writerows(dataGen.user\_info(count=90,height\_min=150,height\_max=195,weight\_min=45,weight\_max=95,Fname\_min=1,Fname\_max=200,Mname\_min=5,Mname\_max=250,Lname\_min=5,Lname\_max=250))

    filename = "membership\_type.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l9)

        csvwriter.writerows(dataGen.membership\_type(count=5,period\_min=2,period\_max=12,amount\_min=1000,amount\_max=10000,user\_id\_min=1,user\_id\_max=75,trainer\_id\_min=1,trainer\_id\_max=20))

    filename = "user\_info\_email.csv"

    with open(filename,'w',newline='') as csvfile:

        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l10)

        csvwriter.writerows(dataGen.user\_info\_email(count=60,email\_min=1,email\_max=50,user\_id\_min=1,user\_id\_max=50))

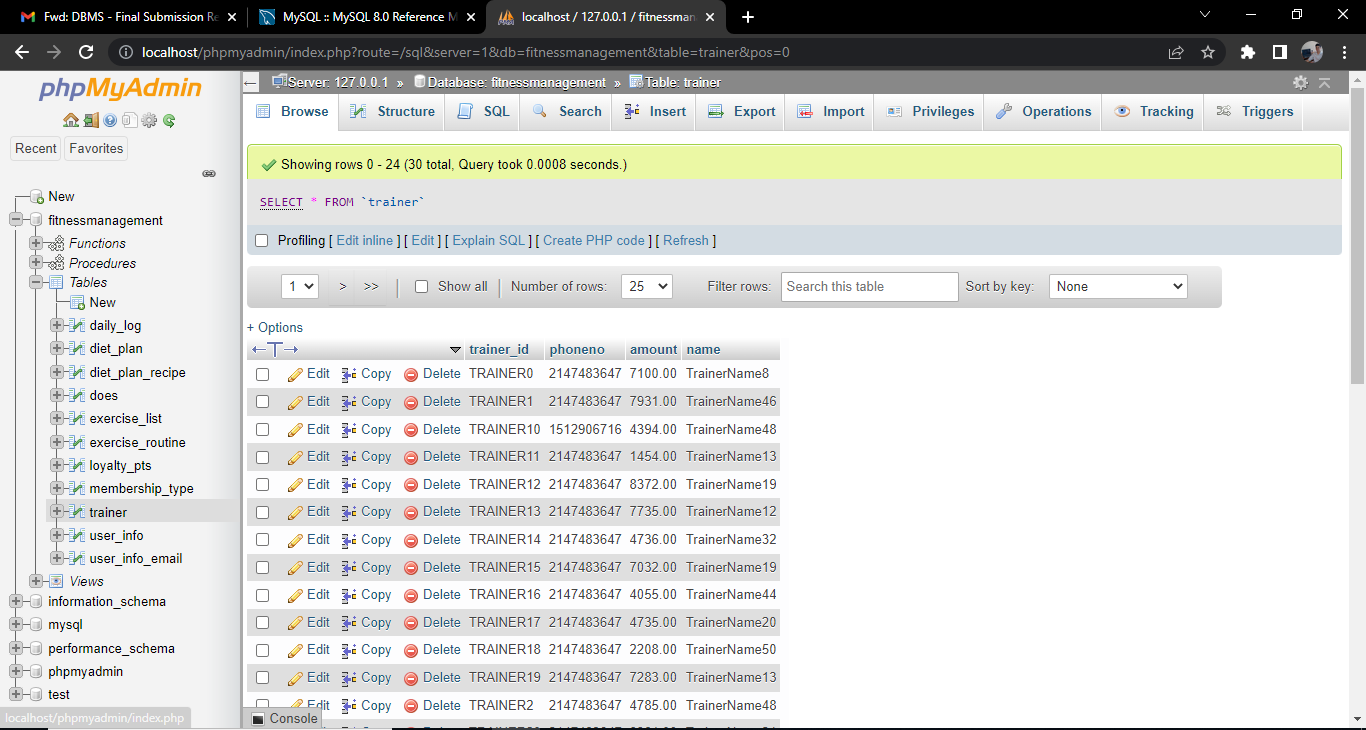
    filename = "trainer.csv"

    with open(filename,'w',newline='') as csvfile:

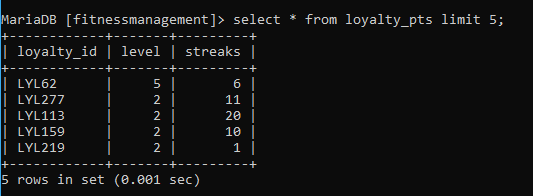
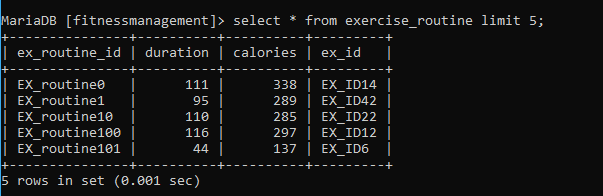
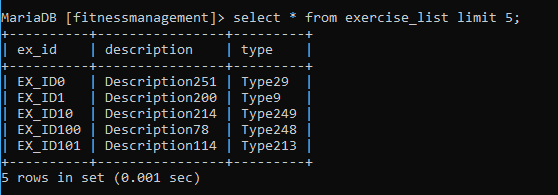
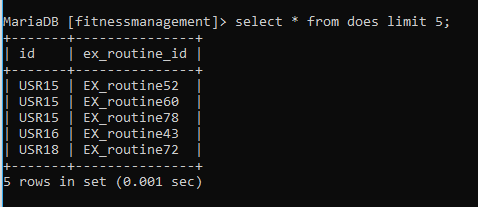
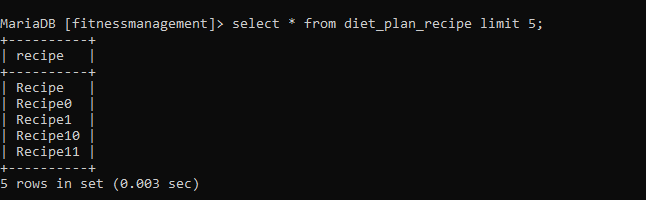
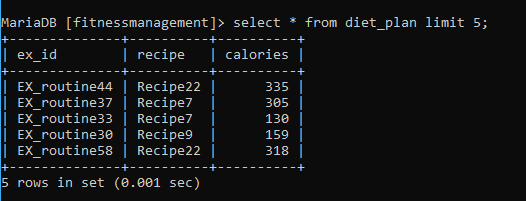
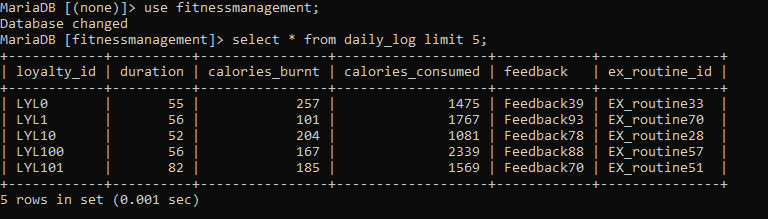
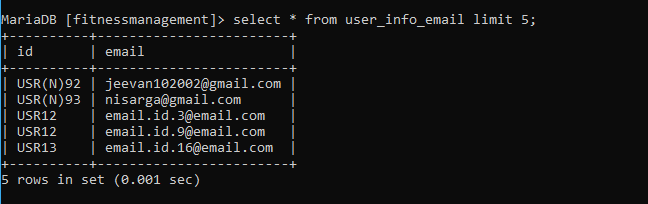
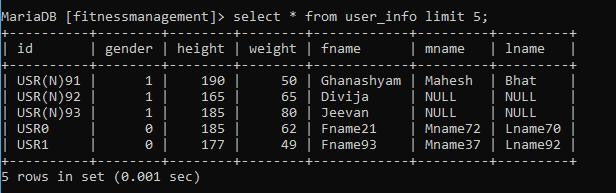
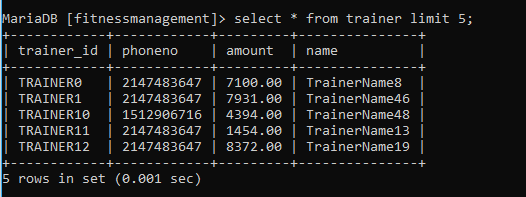
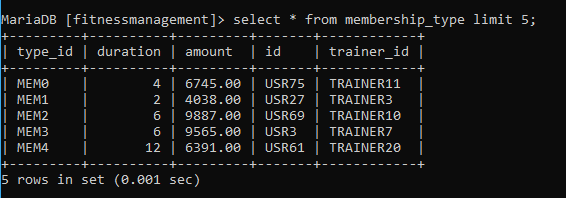
        csvwriter = csv.writer(csvfile)

        csvwriter.writerow(dataGen.l11)

        csvwriter.writerows(dataGen.trainer(count=30,fee\_min=1000,fee\_max=10000,name\_min=1,name\_max=50,ph\_min=1000000000,ph\_max=9999999999))



Sample data of every table after population of database



**Tool Used**

The below are the tools used for the project development

|  |  |
| --- | --- |
| Frontend | Backend |
| Flask | mysql |
| HTML | Sql sqlalchemy |
| Bootstrap | Python3.x |
| Jinja | javascript |

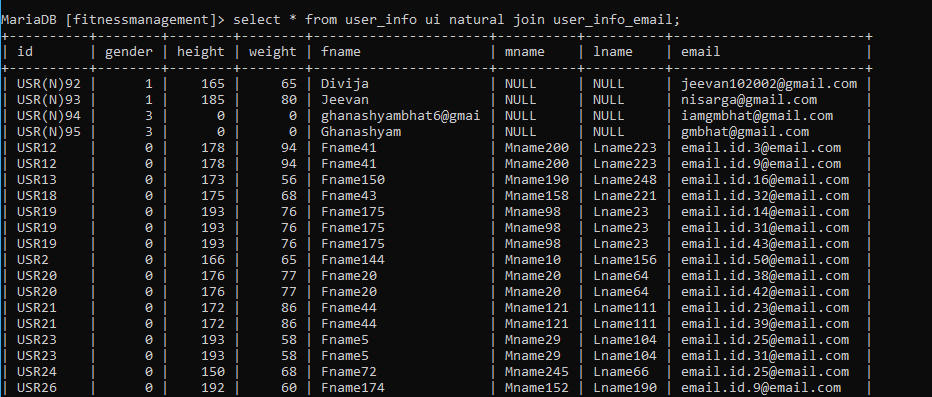
**Queries**

**Join queries (at least 6)**

**Regualar Join (2 queries):**

1. To display the user information along with the user email ID by combining user\_info and user\_info\_email table.

select \* from user\_info ui natural join user\_info\_email;

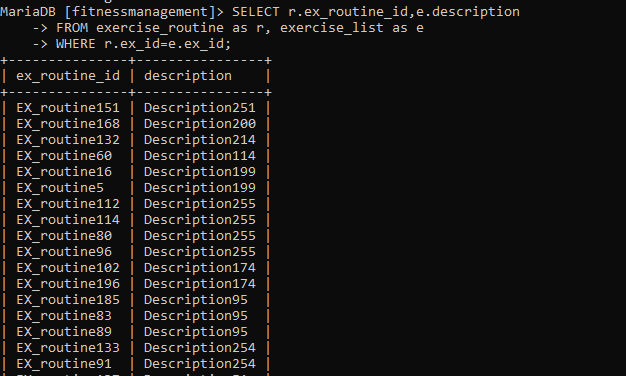


2. Display each exercise routine id along with the exercise description by combining excercise list and exercise routine table

SELECT r.ex\_routine\_id,e.description

FROM exercise\_routine as r, exercise\_list as e

WHERE r.ex\_id=e.ex\_id;



**Co-related Join (2 Queries)**

1. Display name of the trainer and the amount if the amount charged by the trainer is less than the average amount

SELECT t.name,m.amount

FROM membership\_type m,trainer t

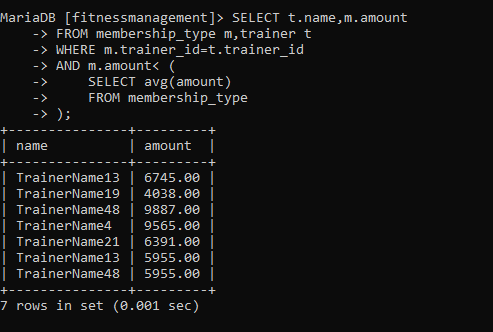
WHERE m.trainer\_id=t.trainer\_id

AND m.amount< (

    SELECT avg(amount)

    FROM membership\_type

);



2. Display the ID and name of the user if he burns more than 30% of the calories consumed.

SELECT u.id,u.fname

FROM user\_info u

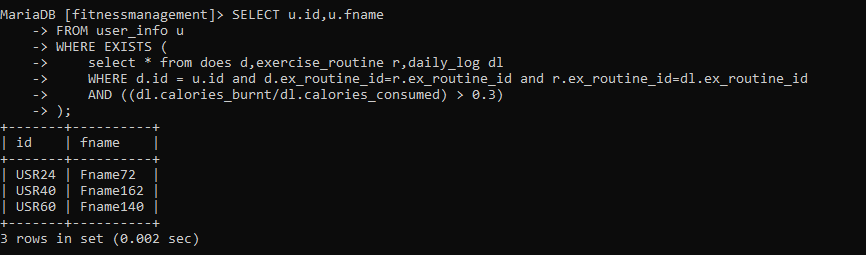
WHERE EXISTS (

    select \* from does d,exercise\_routine r,daily\_log dl

    WHERE d.id = u.id and d.ex\_routine\_id=r.ex\_routine\_id and r.ex\_routine\_id=dl.ex\_routine\_id

    AND ((dl.calories\_burnt/dl.calories\_consumed) > 0.3)

);



**Nested Join (2 Queries)**

1. Display exercise routine and recipe with calories consumed and calories burned.

SELECT d.calories,r.calories,d.recipe

FROM exercise\_routine as r, (

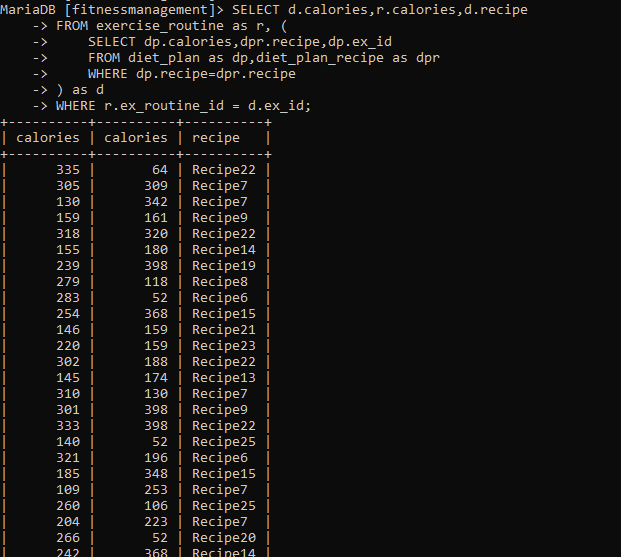
    SELECT dp.calories,dpr.recipe,dp.ex\_id

    FROM diet\_plan as dp,diet\_plan\_recipe as dpr

    WHERE dp.recipe=dpr.recipe

) as d

WHERE r.ex\_routine\_id = d.ex\_id;



2. List user info, subscription purchased and trainer info for the user who has subscribed to a membership

SELECT u.id,u.fname,t.trainer\_id,t.name,t.duration,t.type\_id

FROM user\_info as u

JOIN (

    SELECT m.id,m.duration,m.type\_id,t.trainer\_id,t.name

    FROM trainer as t

    JOIN

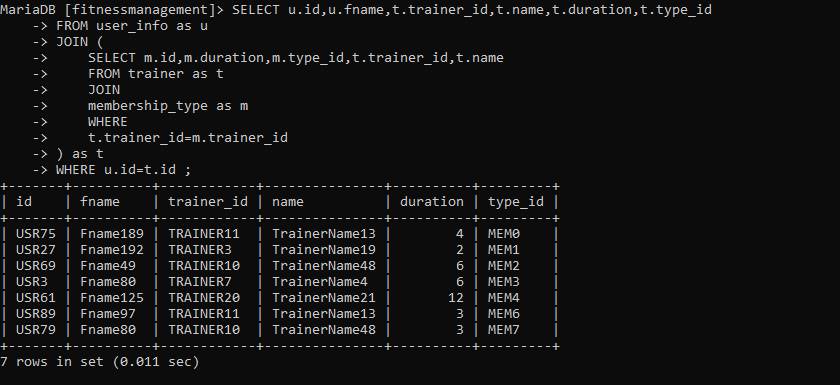
    membership\_type as m

    WHERE

    t.trainer\_id=m.trainer\_id

) as t

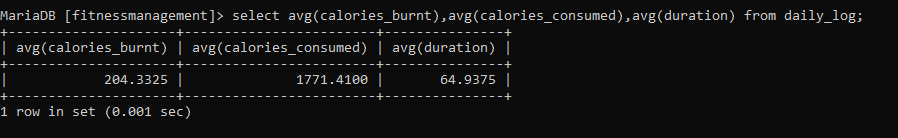
WHERE u.id=t.id ;



**Aggregate Functions (at least 2)**

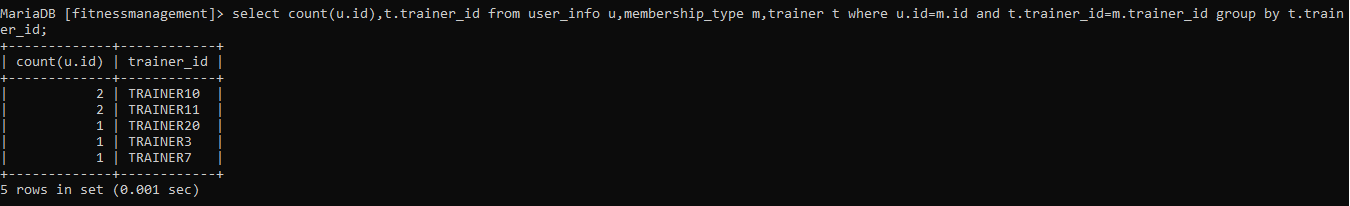
1.Display the average calories consumed, average calories burned and duration of exercise by all the users.

select avg(calories\_burnt),avg(calories\_consumed),avg(duration) from daily\_log;



2. Display number of users per trainer as per the membership purchased

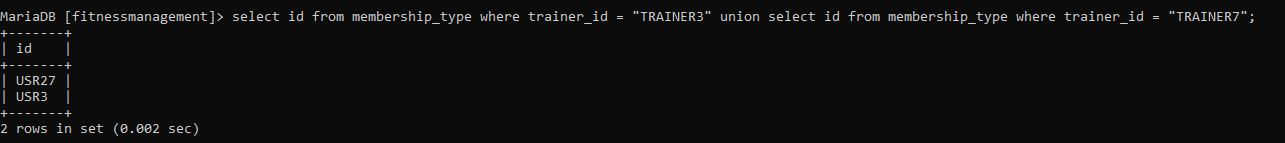
select count(u.id),t.trainer\_id from user\_info u,membership\_type m,trainer t where u.id=m.id and t.trainer\_id=m.trainer\_id group by t.trainer\_id;



**Set Operations (at least 2)**

**1.** Display users who have subscribed to Trainer3 or Trainer7

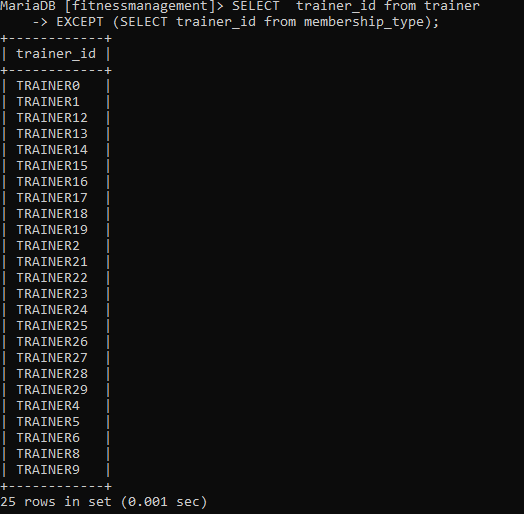
select id from membership\_type where trainer\_id = "TRAINER3" union select id from membership\_type where trainer\_id = "TRAINER7";



**2.** Display all the trainers who are not assigned to any user

SELECT  trainer\_id from trainer

EXCEPT (SELECT trainer\_id from membership\_type);



**View (atleast 1)**

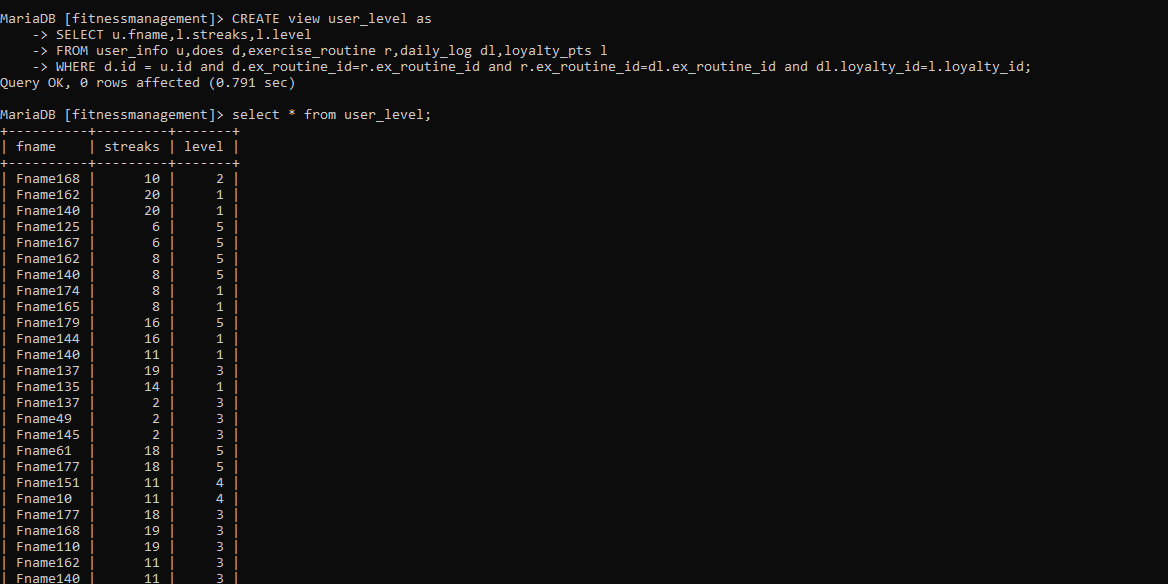
The view is created for easier access of Streak and Level information of every user based on their daily log.

CREATE view user\_level as

SELECT u.fname,l.streaks,l.level

FROM user\_info u,does d,exercise\_routine r,daily\_log dl,loyalty\_pts l

WHERE d.id = u.id and d.ex\_routine\_id=r.ex\_routine\_id and r.ex\_routine\_id=dl.ex\_routine\_id and dl.loyalty\_id=l.loyalty\_id;



**Function and Procedure**

The function takes gender value (INT) as input and returns 1, if the value is greater than 0, otherewise returns 0.

The procedure has been created with this function to make the gender as binary value in the database.

DELIMITER $$ ;

CREATE FUNCTION correcting\_gender(GENDER1 INT)

RETURNS INT

DETERMINISTIC

BEGIN

   DECLARE gender INT;

   SET gender = GENDER1 ;

   IF gender > 0 THEN

   SET gender = 1;

   ELSE

   SET gender = 0;

   END IF;

   RETURN gender;

END; $$

DELIMITER ;

DELIMITER $$

CREATE procedure gender\_updation()

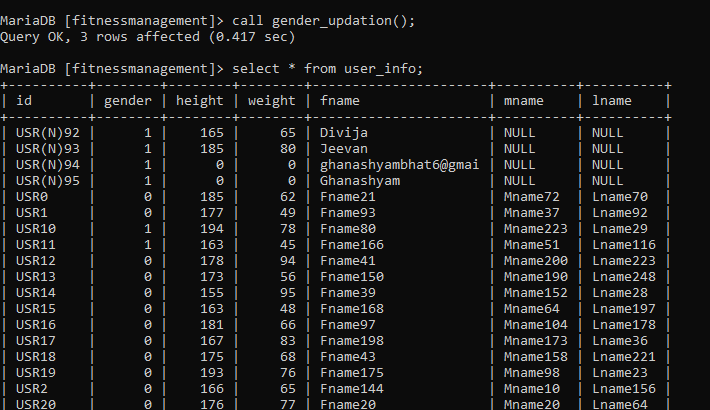
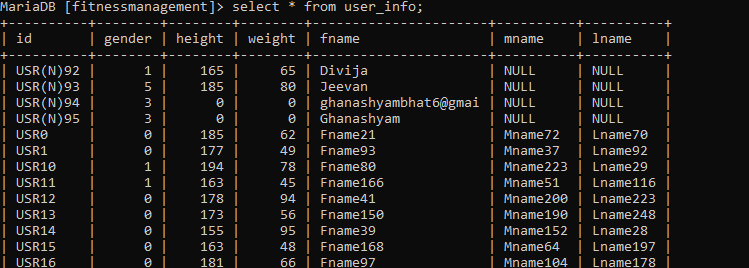
BEGIN

UPDATE user\_info

SET gender = correcting\_gender(gender);

END;$$

DELIMITER ;



**Trigger**

The trigger here prevent the Trainer from having more than 3 subscribers.

When a new user takes membership, he cannot choose the trainer with already 3 subscribers.

DELIMITER $$

CREATE TRIGGER total\_no\_of\_user\_per\_trainer\_exceed\_check

BEFORE INSERT

ON membership\_type FOR EACH ROW

BEGIN

    DECLARE error\_msg VARCHAR(255);

    SET error\_msg = (' total number of users per trainer exceeds more than 3');

    IF (select count(\*) from membership\_type where trainer\_id=new.trainer\_id group by trainer\_id) > 2 THEN

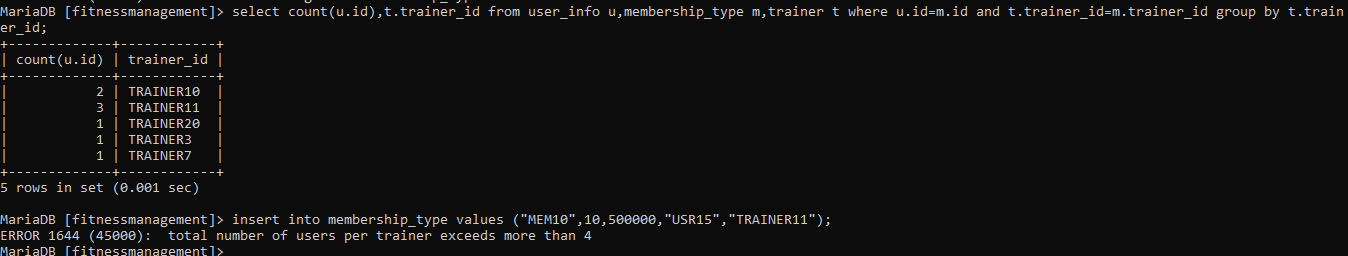
    SIGNAL SQLSTATE '45000'

    SET MESSAGE\_TEXT = error\_msg;

    END IF;

END $$

DELIMITER ;



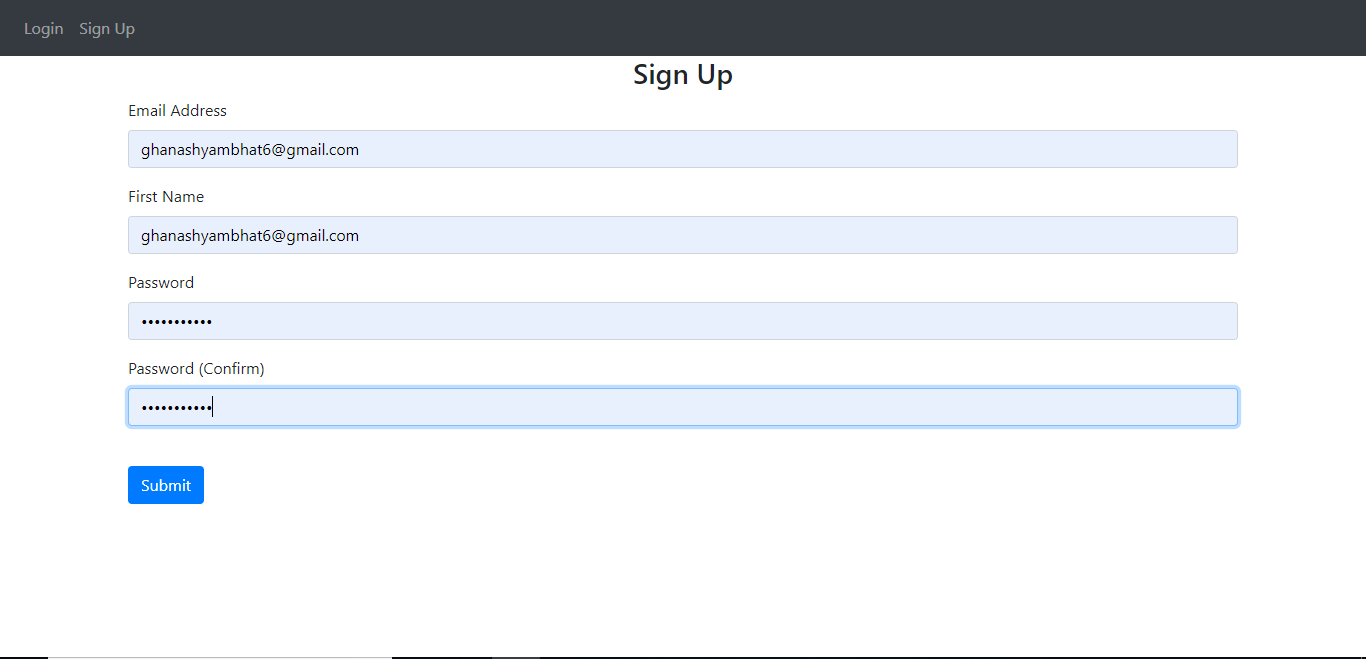
**Developing a Frontend**

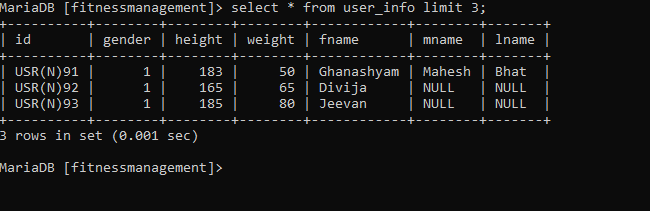
The frontend should support

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

Addition of data

When the user signs up, new user will be added to the database. The data like Name, password, email will be stored in the database. New user ID will be allocated to each user on creation of new account.

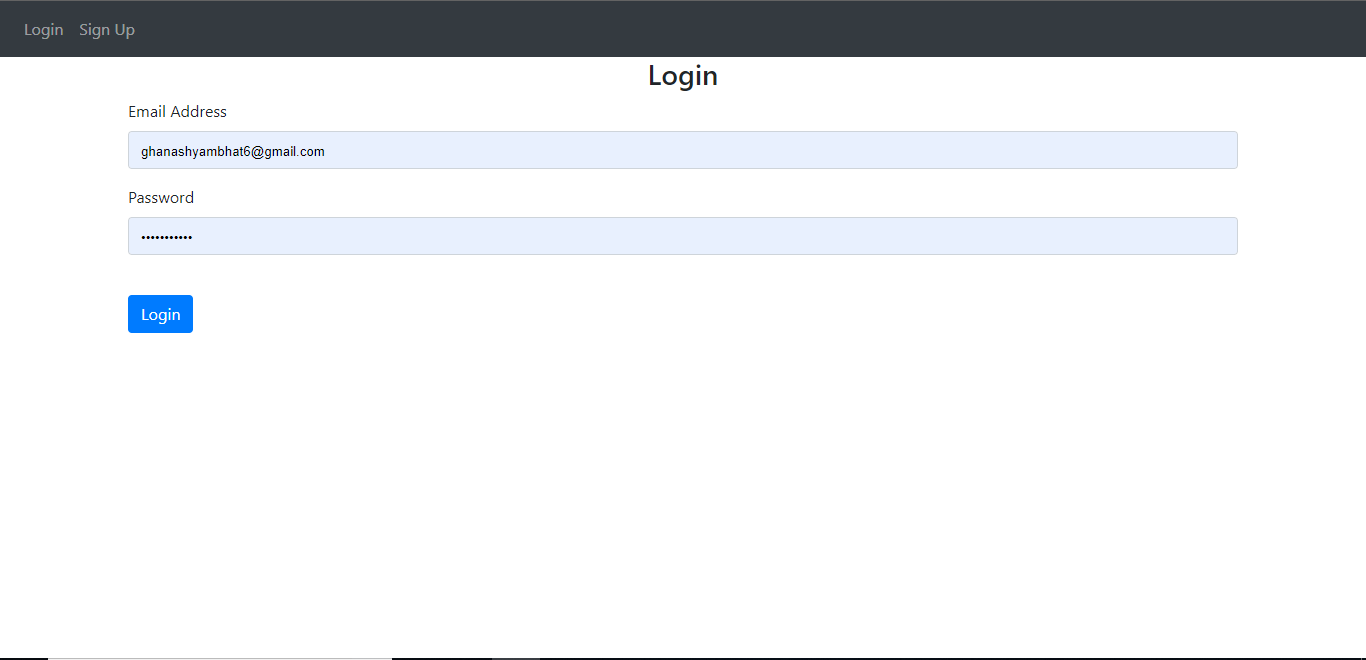




Fetch Data

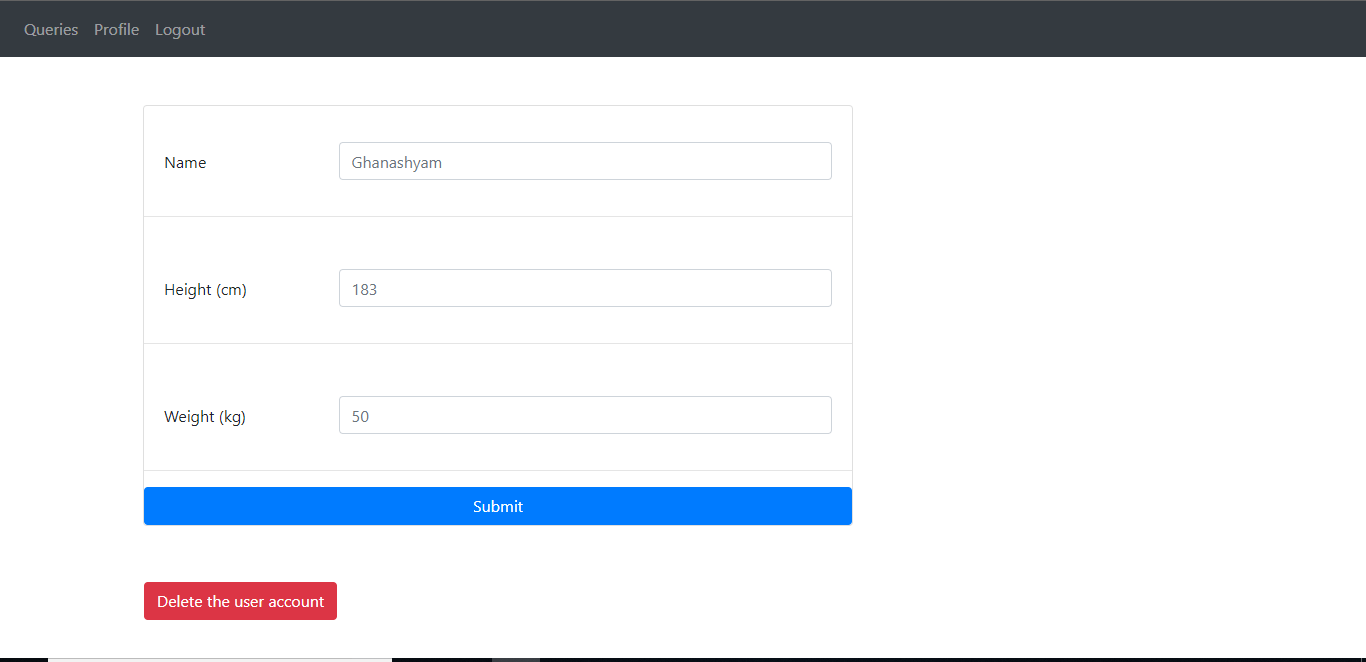
The user who already has an account can log in to the account and access all his data by providing the correct password.

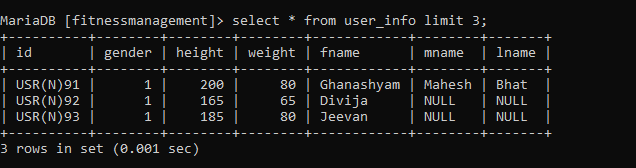
The password will be checked with the stored hash in the database and the user will be logged in if it matches.



Data Modification

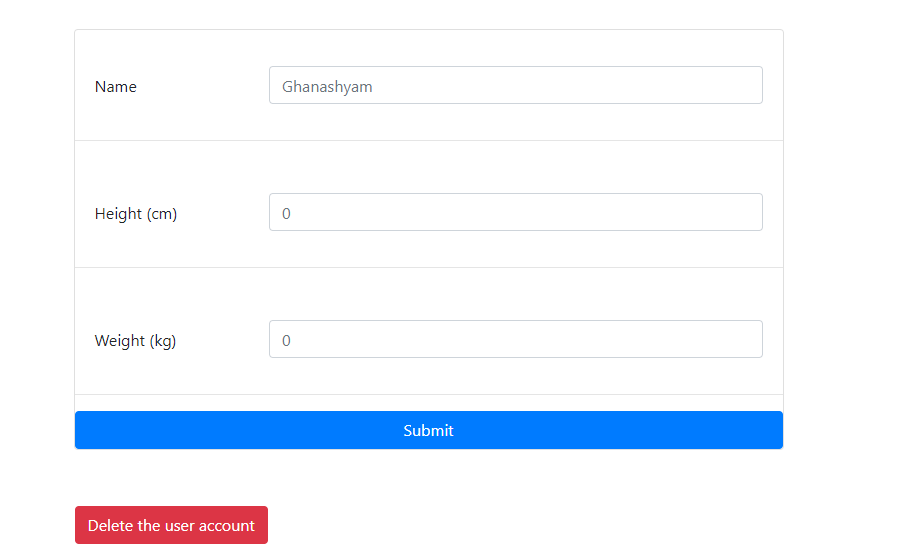
The user can modify his name,height and weight data. On clicking the submit button, the data gets updated.

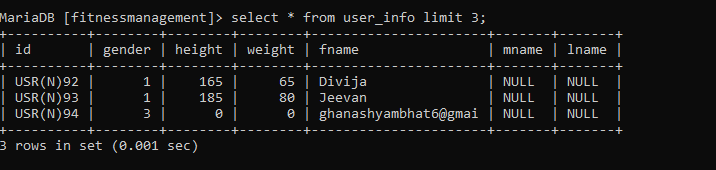




Delete Data

On pressing delete account button, the user account will be deleted from the database and the user cannot login using same account in future.





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

