

GYM DATABASE MANAGEMENT AND CUSTOMER WEBSITE.

REVIEW REPORT

Submitted by

DEEP HIREN KOTCHA (19BCE2301)

BHAVYA RUKHAIYAR (19BCE2365)

V.KAVIYARASU (19BCB0134)

Prepared For

DATABASE MANAGEMENT SYSTEM (CSE2004)

PROJECT COMPONENT

Submitted To

Dr. Priya M

Associate Professor

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CONTENTS:

| | |
|--|-------|
| Abstract | 3 |
| Chapter | 4 |
| 1. Introduction..... | 5-6 |
| 1.1 Background | |
| 1.2 Objective | |
| 1.3 Motivation | |
| 1.4 Contributions of the Project | |
| 1.5 Organization of the Project | |
| 2. Project Resource Requirements..... | 7 |
| 2.1 Software Requirements | |
| 2.2 Hardware Requirements | |
| 3. Design of the Project..... | 8-28 |
| 3.1 ER Diagram | |
| 3.3 ER to Relational Mapping (Schema Diagram) | |
| 3.4 Normalization | |
| 3.5 Tables and Constraints | |
| 4. Implementation..... | 29-43 |
| 4.1 Introduction | |
| 4.2 DDL & DML Queries | |
| 4.3 SQL Queries | |
| 4.4 PL/SQL | |
| 5. Screenshot (front end-with explanation)..... | 44-51 |
| 6. Conclusion | 52 |
| References(IEEE Style, Do not give websites in references..... | 53 |

ABSTRACT

The main objective of the project is to design and develop a user-friendly Gym Database Management and Customer Website system. This project also provides security to data by using login & password. This system is proposed to be an automate database management & transactions. This stores staff, customers, membership, class, equipments, exercise, and products information. The implementation of the system in the organization will considerably reduce data entry, time and also provide readily calculated reports.

CHAPTER:

As the Existing system was manual, Time consuming as data entry which include calculations took lot of time, Searching was very complex as there could be 100's of entry every year , The proposed system is expected to be faster than the existing system.

Searching a particular data specific to particular requirements is also very tedious in such system. In order to retrieve records, the responsible person needs to manually locate the appropriate register and locate the appropriate placement of that particular record which may be very time consuming.

Data Redundancy is also a great issue in such kind of system. 'Redundancy' means repetition. Thus, data modified Corrupted at a particular place may not be data modified corrupted at the other related place which may create in consistencies in data handling Destroys Data Integrity and creates confusion for the owner.

1.1 INTRODUCTION

BACKGROUND:

Generally, in order to structure these tasks Separate Registers are maintained. This whole process thus becomes Quite cumbersome for them to control manually. Moreover, any wrong data entered mistakenly can brings serious results.

OBJECTIVE:

The Project was made in order to effectively and efficiently cater to requirements of the fitness centre. Very frequently the person who generally holds the tasks to manage the Gym Centre needs to keep records of all the transactions as well as data manually.

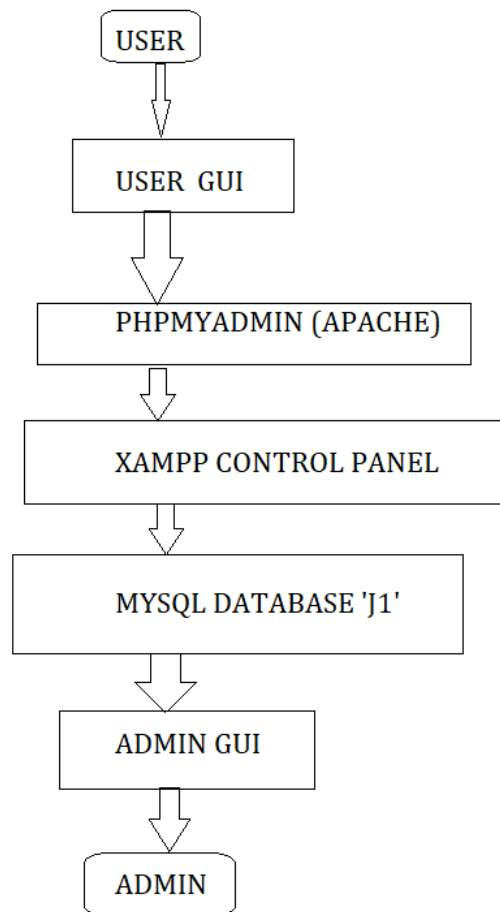
MOTIVATION:

The Manually Managed system of the store was also heavily prone to data loss due to certain causes Displacement of Registers Destruction of Registers Unauthorised access to registers etc. which can bring in disastrous situations.

CONTRIBUTIONS:

| REG NO. | NAME | WORK ASSIGNED |
|------------------|---------------------------|--|
| 19BCB0134 | V.KAVIYARASU | DATABASE DESIGN AND NORMALISATION OF TABLES |
| 19BCE2365 | BHAVYA RUKHAIYAR | HTML/CSS OF ALL PAGES |
| 19BCE2301 | DEEP HIREN KOTecha | PHP CODE, JavaScript code, Queries/Programs for the Database. |

ORGANISATION OF PROJECT:



PROJECT RESOURCE REQUIREMENTS:

2.2 HARDWARE REQUIREMENTS:

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

Laptop with 4Gb RAM ,500Gb Hard disk, i5 Processor

2.3 SOFTWARE REQUIREMENTS:

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

MYSQL DATABASE VER 8.0 OR HIGHER

XAMPP 7.4.10-0 FOR CONNECTING MYSQL

DATABASE to front end with APACHE

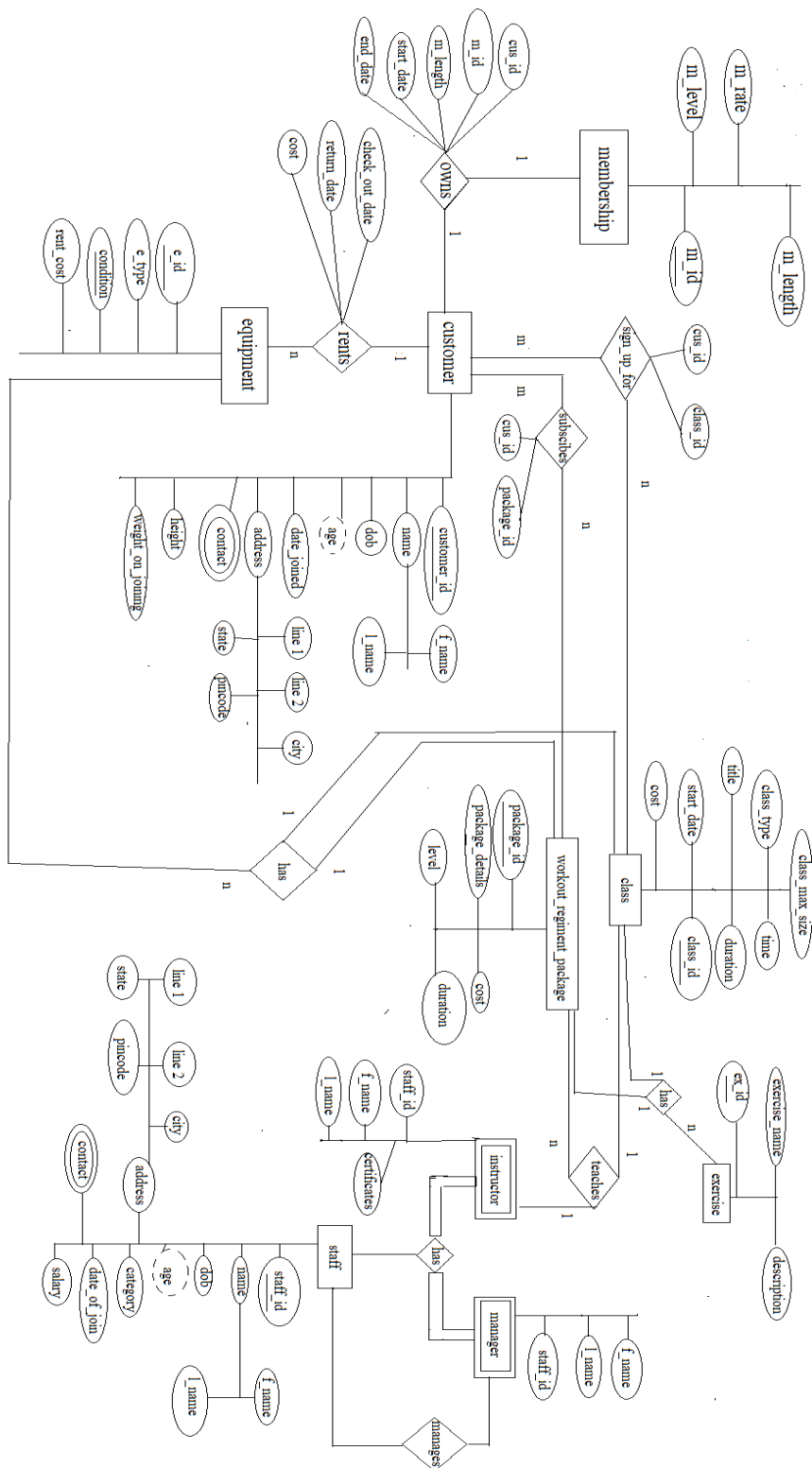
XAMPP CONTROL PANEL VER 3.2.4

FRONT-END : HTML/CSS AND JAVASCRIPT,

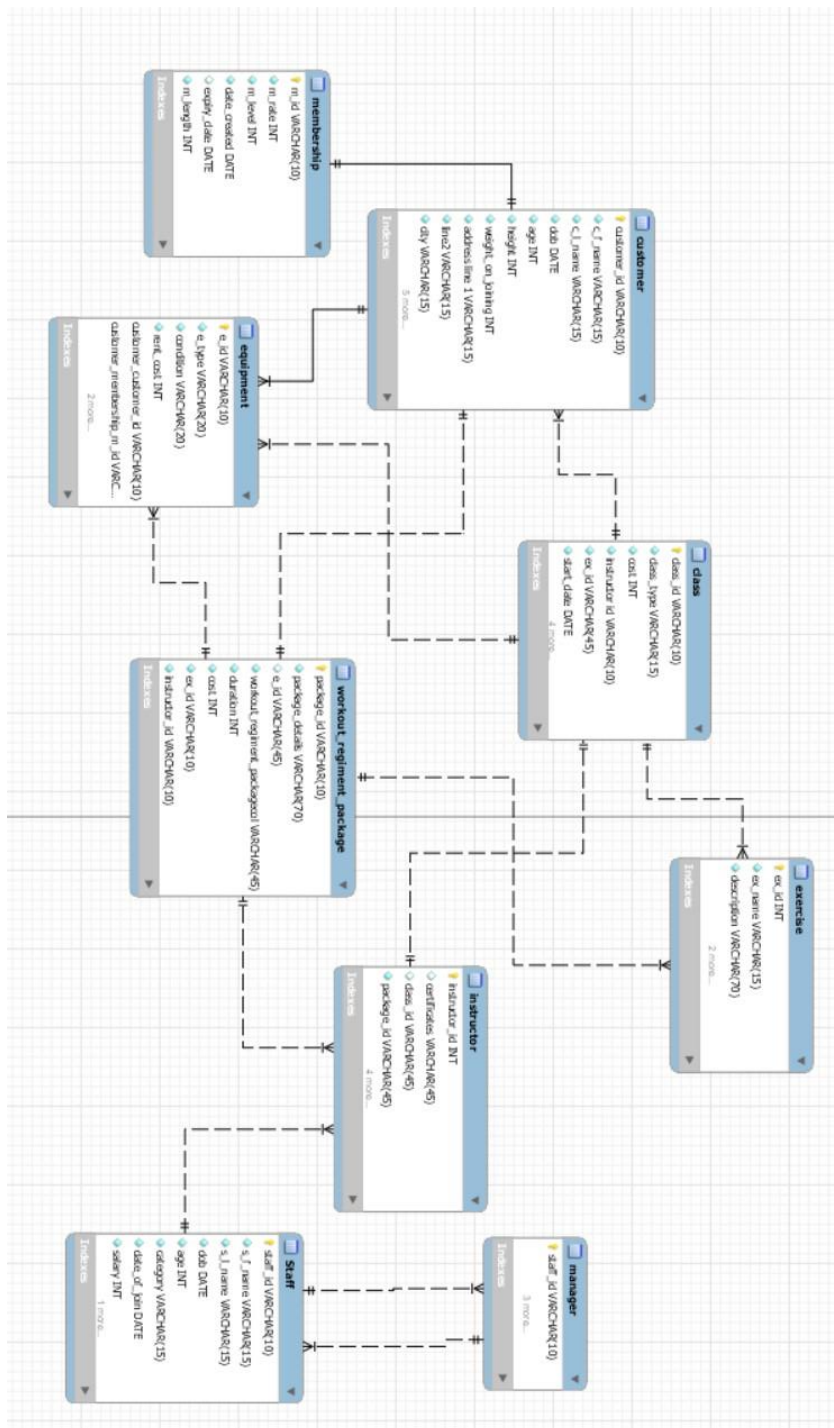
BACK END: PHP

3) DESIGN OF THE PROJECT:

1). ER DIAGRAM:



2) CONCEPTUAL SCHEMA:



NORMALISATION:

1NF:

All the attribute of every tuple is having atomic values. Not have any multivalued.

2NF:

1. Relation already exists in 1NF.
2. No partial dependency exists in the relation.

Partial dependencies = few attributes of the candidate key determine non-prime attribute.

3NF:

1. Relation already exists in 2NF.
2. No transitive dependency exists for non-prime attributes.

$A \rightarrow B$ is called a transitive dependency if and only if-

1. A is not a super key.
2. B is a non-prime attribute.

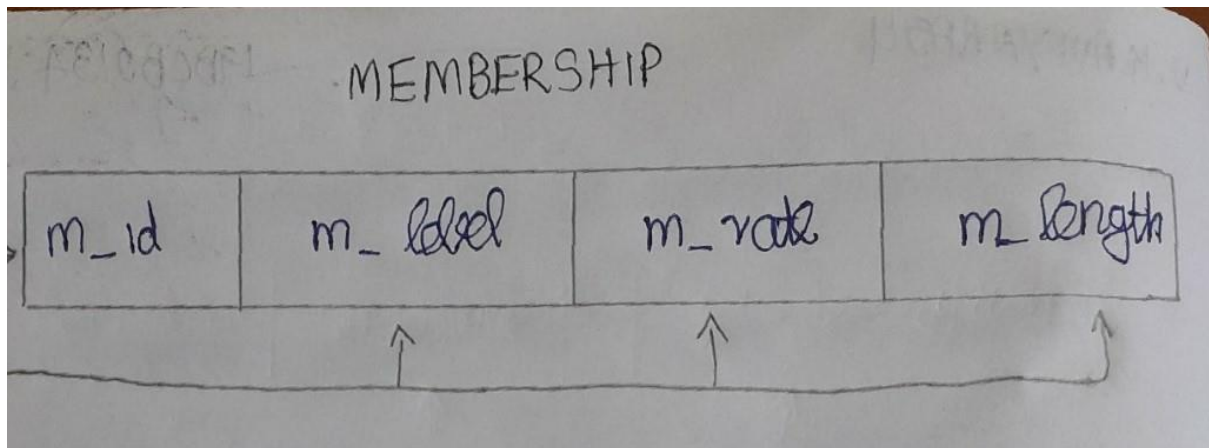
If any one condition fails, then it is not a transitive dependency.

BCNF:

1. Relation already exists in 3NF.
2. For each non-trivial functional dependency,

$A \rightarrow B$, A is a super key of the relation.

MEMBERSHIP



MEMBERSHIP (m_id, m_level, m_rate, m_length)

1NF:

The primary key is: {m_id}

In the table all column have the atomic values and no one have composite or multivalued attributes.

2NF:

The candidate key is {m_id}

The set of key attributes is {m_id}

Check that the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

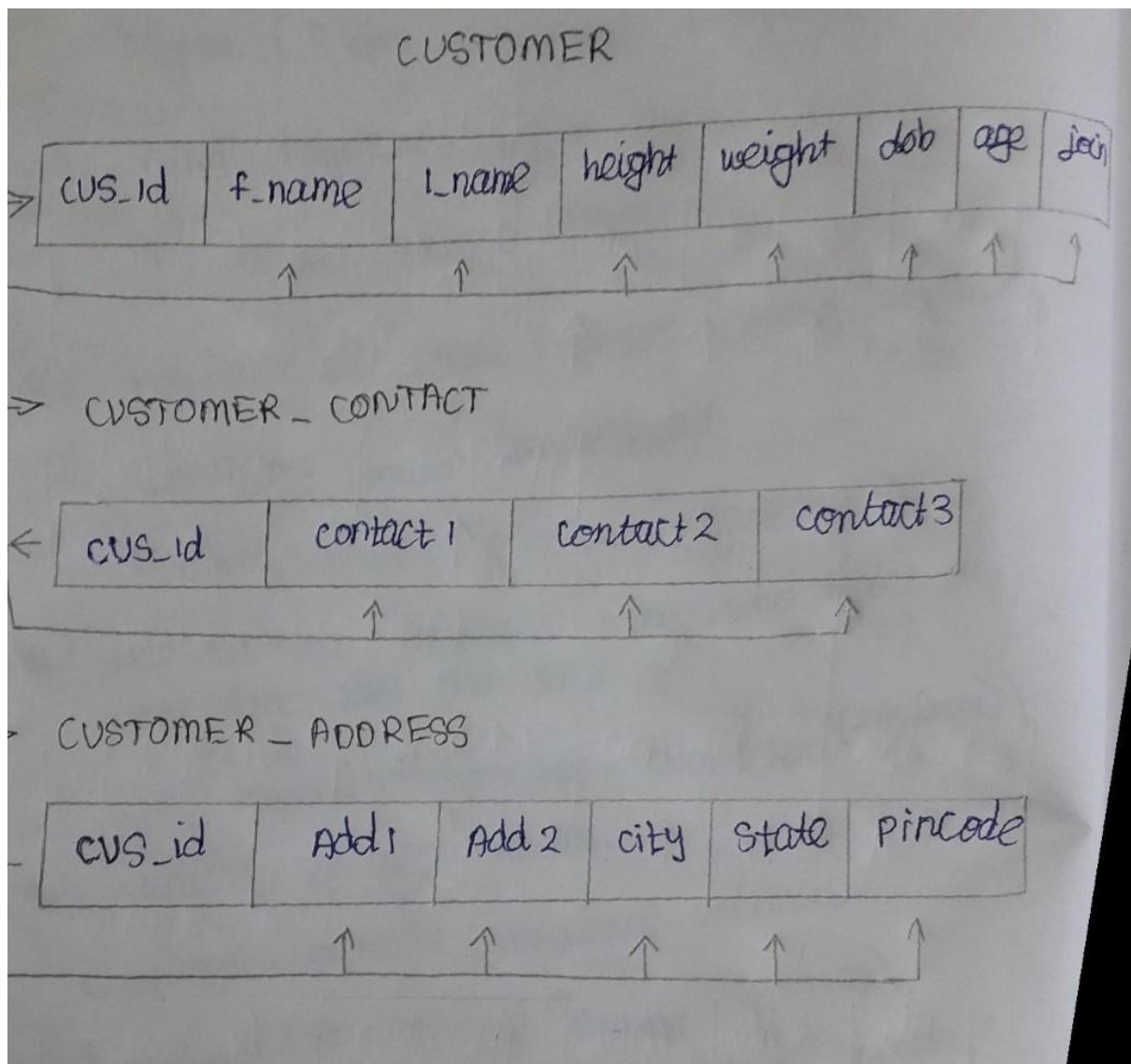
3NF:

For each function dependency check that the LHS is superkey or the RHS are all key attribute.

BCNF:

A table is in BCNF if and only if every non-trivial FD, the LHS is a superkey.

CUSTOMER



CUSTOMER (cus_id, f_name, l_name, height, weight_on_joining, dob, age, date_joined)

1NF:

The primary key is: {cus_id}

In the table all column have the atomic values and no one have composite or multivalued attributes.

2NF:

The candidate key is {cus_id}

The set of key attributes is {cus_id}

Check that the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

3NF:

For each function dependency check that the LHS is superkey or the RHS are all key attribute.

BCNF:

A table is in BCNF if and only if every non-trivial FD, the LHS is a superkey.

From the customer table these are normalized.

CUSTOMER_CONTACT (cus_id, contact1, contact2, contact3)

The table is obtaining all the rules of normalization.

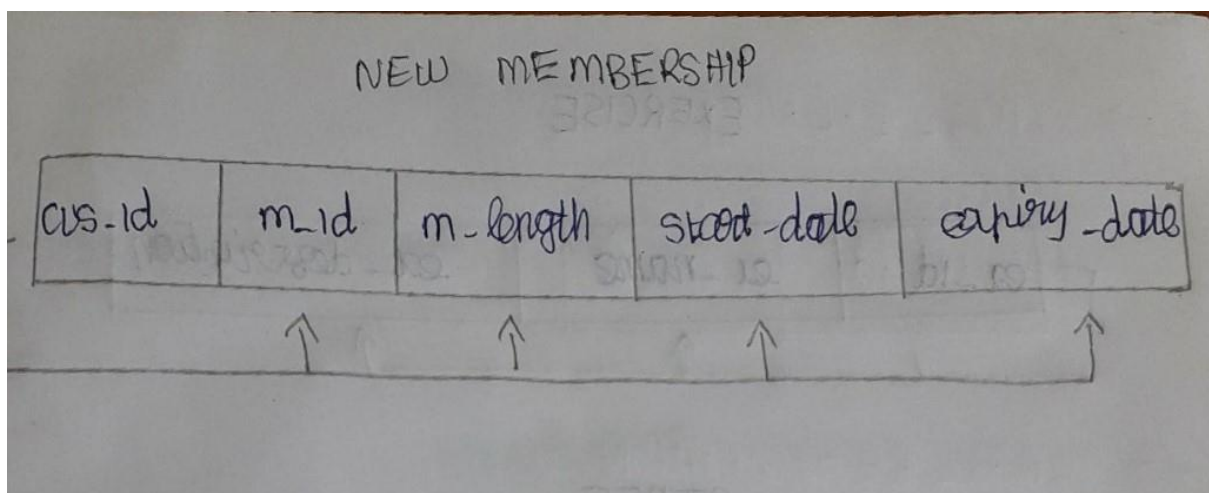
Table in BCNF.

CUSTOMER_ADDRESS (cus_id, add1, add2, city, state, pincode)

The table is obtaining all the rules of normalization.

Table in BCNF.

NEW MEMBERSHIP



NEW_MEMBERSHIP (cus_id, m_id, m_length, strat_date, expiry_date)

1NF:

In the table all column have the atomic values and no one have composite or multivalued attributes.

2NF:

The FD $\text{cus_id} \rightarrow \text{start_date}$ is a partial dependency.

To avoid this, we make another table,

tab1 = new_membership(cus_id, m_id, m_length)

tab2 = mem_date(m_id, start date, expiry date)

3NF:

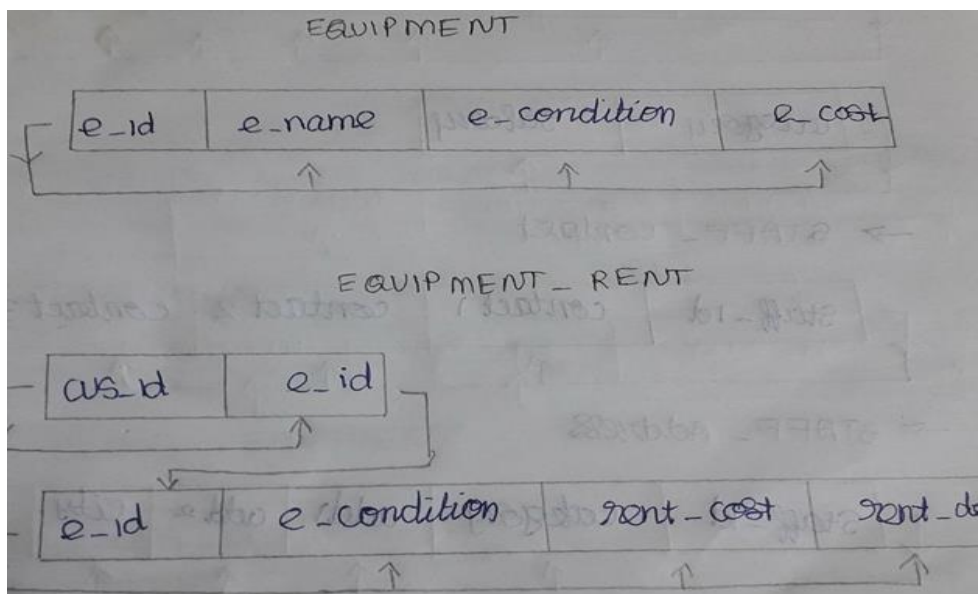
There is not have transitive dependency where the RHS includes non-key attribute

So table already in 3NF.

BCNF:

Table already in BCNF.

EQUIPMENT



EQUIPMENT (e_id, e_name, e_condition, e_cost)

1NF:

All the column in the table have atomic values.

So the table already in 1NF.

2NF:

No partial dependency are there so the table already in 2NF

3NF:

No transitive dependency are there so table already in 3NF

BCNF:

For every non-trivial FD, LHS is a superkey.

So already table in BCNF.

EQUIPMENT_RENT (cus_id, e_id, e_condition, rent_date, rent_cost)

1NF:

Cus_id is a primary key and all the values are atomic so the table already in 1NF.

2NF:

For e_id based the e_condition and rent_cost are same.

So partial dependency are there.

Tab1 = (cus_id,e_id)

Tab2 = (e_id, e_condition, rent_cost, rent_date)

Now the table in 2NF.

3NF:

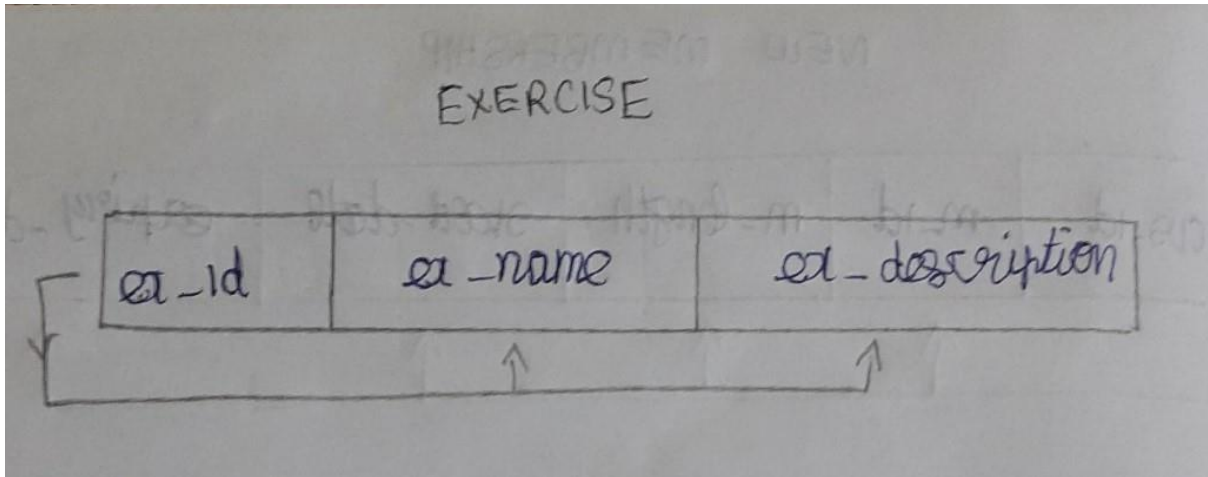
No transitive dependency so the table already in 3NF

BCNF:

For every non-trivial FD, LHS is a superkey.

So already table in BCNF.

EXERCISE



EXERCISE (`ex_id`, `ex_name`, `ex_description`)

1NF:

The table all the column have the atomic values. So the table already in 1NF

2NF:

No partial dependency are exist

Already in 2NF

3NF:

No transitive dependency are there

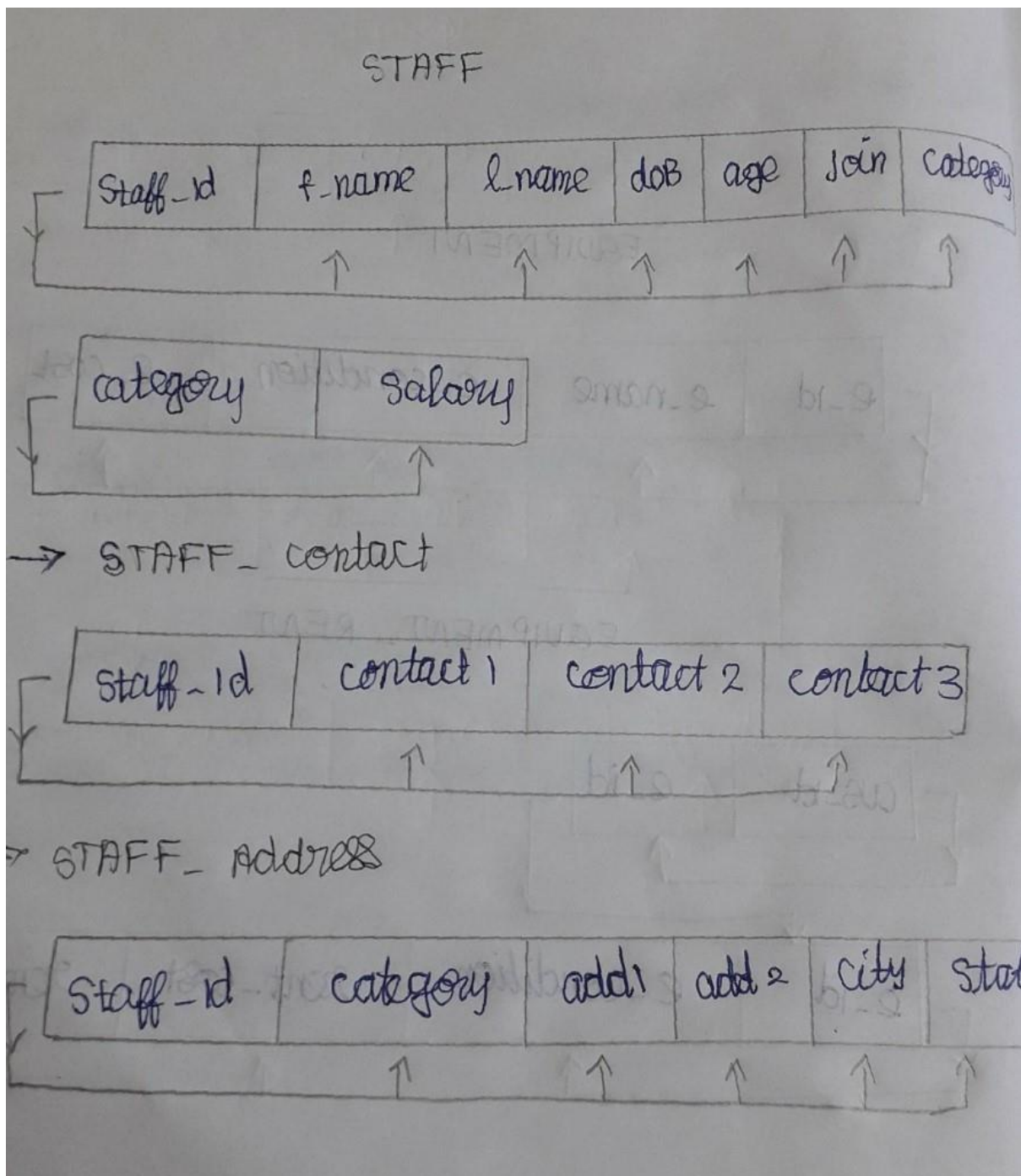
So already in 3NF

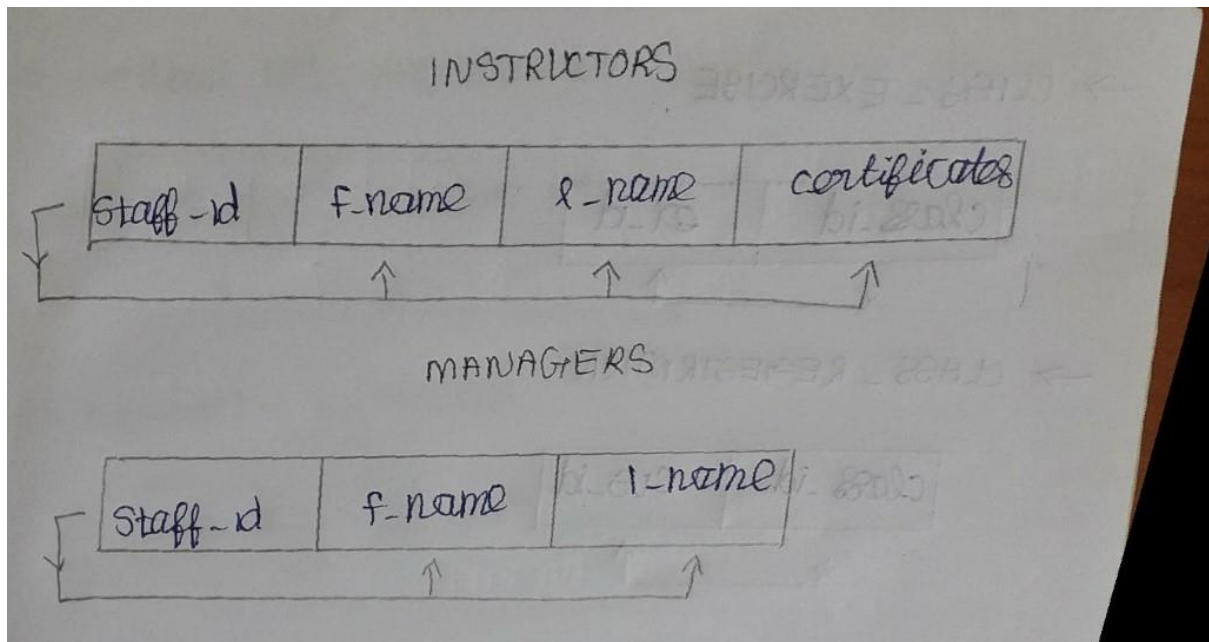
BCNF:

For every non-trivial FD, LHS is a superkey.

So already table in BCNF.

STAFF





STAFF (staff_id, f_name, l_name, dob, age, category, date_of_join, salary)

1NF:

For particular category salary is same only.

So it will violated.

Tab1 = (staff_id, f_name, l_name, dob, age, date_of_join, category)

Tab2 = (category, salary)

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

STAFF_CONTACT (staff_id, category, contact1, contact2, contact3)

These table obtain all the rules of normalization

STAFF_ADD (staff_id, category, add1, add2, city, state)

These table also obey the all normalization rules.

INSTRUCTORS (staff_id, f_name, l_name, certificates)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

MANAGERS (staff_id, f_name, l_name)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

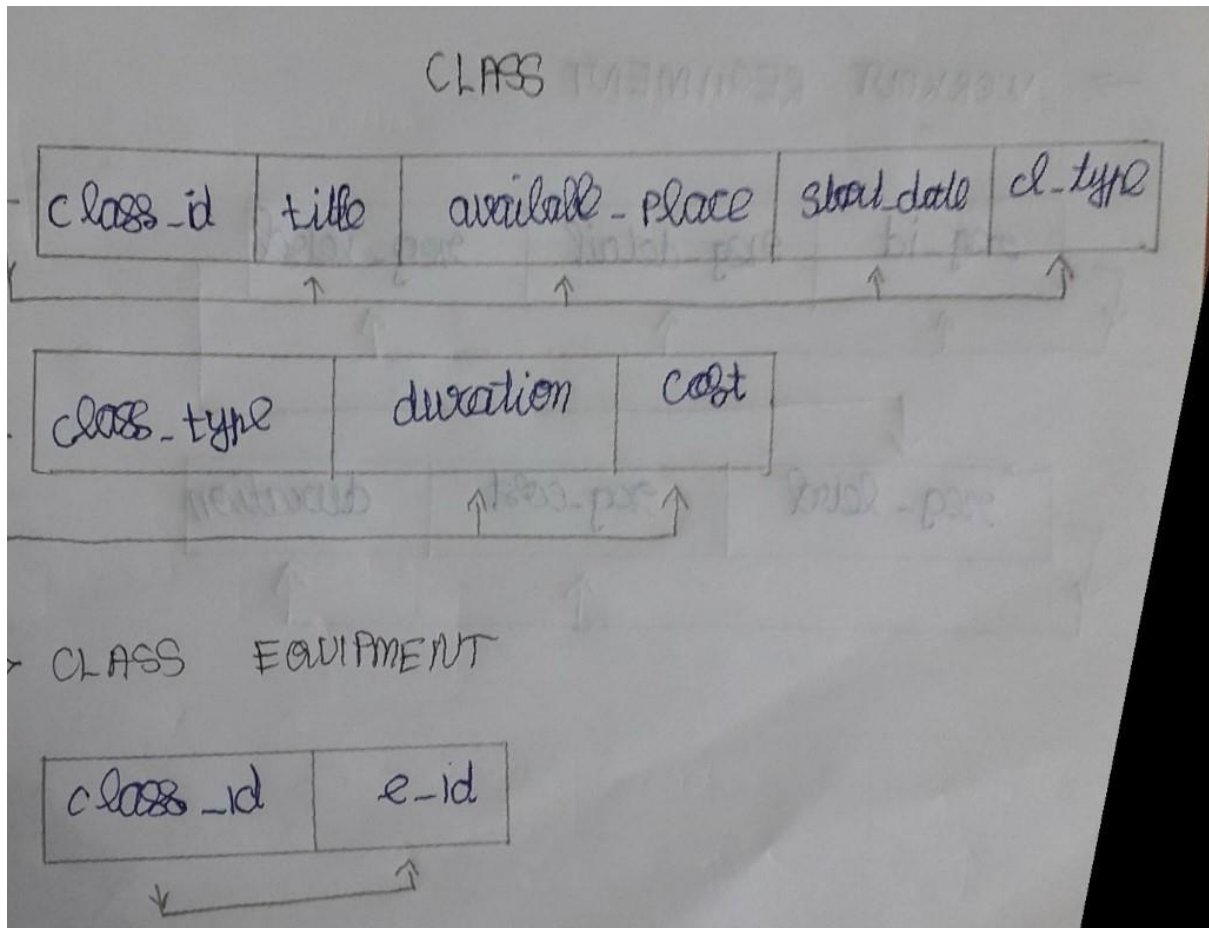
3NF:

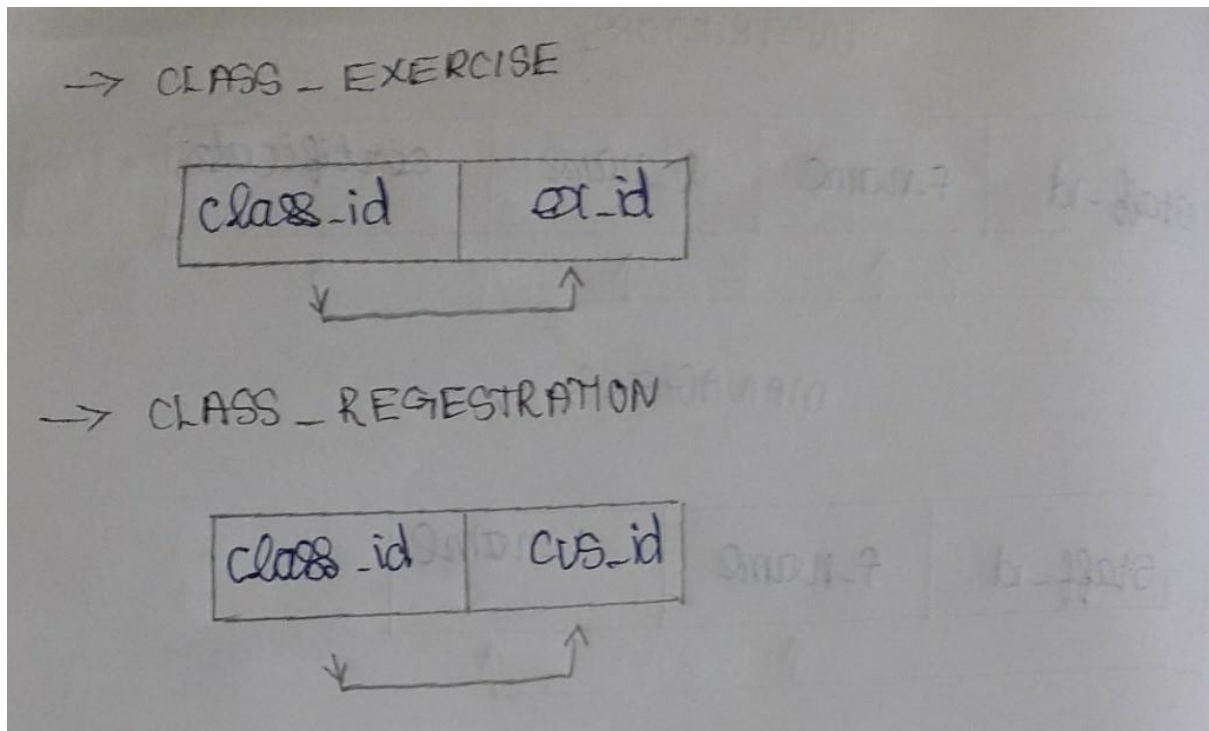
No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

CLASS





CLASS (class_id, title, class_type, available_space, start_date, duration, c_time, cost)

1NF:

Class_type have same duration, cost.

So it will violated.

Tab1 = (class_id, title, class_type, available_space, start_date)

Tab2 = (class_type, duration, cost)

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey

CLASS_EQUIPMENT (class_id, e_id)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

CLASS_EXERCISE (class_id, ex_id)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

CLASS_REGISTRATION (class_id, cus_id)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

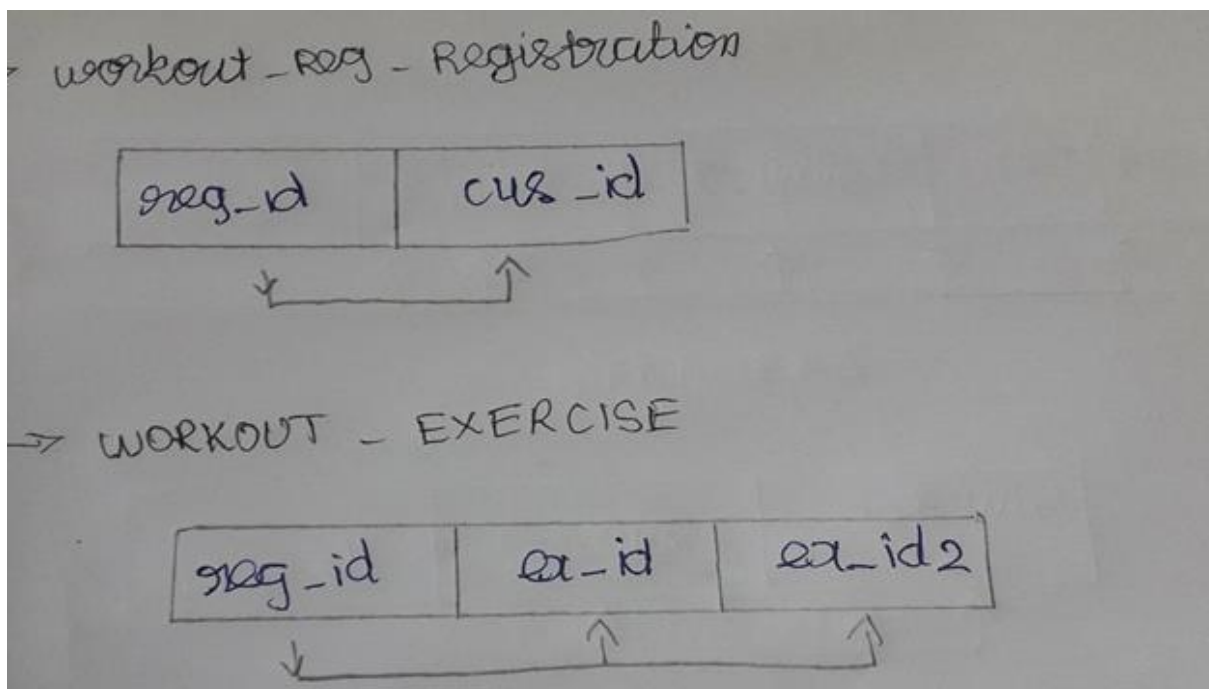
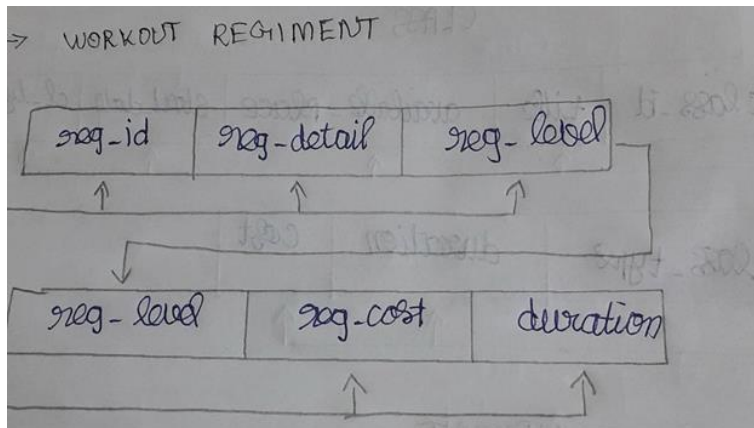
3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

WORKOUT REGIMENT



WORKOUT_REGIMENT (reg_id, reg_details, reg_level, cost, duration)

1NF:

For regiment particular level have same cost and the same duration. So this is violated.

Tab1 = (reg_id,reg_detail,reg_level)

Tab2 = (reg_level,cost,duration)

2NF:

No partial dependency are there

3NF:

No transitive dependency are there

BCNF:

Non trivial FD, LHS is a superkey

WORKOUR_REG_REGISTRATION (reg_id, cus_id)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey

WORKOUT_EXERCISE (reg_id, ex_id, ex_id2)

1NF:

All the column have atomic values

2NF:

No partial dependency exist

3NF:

No transitive dependence

BCNF:

For non-trivial FD LHS is a superkey.

TABLES AND CONSTRAINTS:

CUSTOMER:

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|---|---|--|
| Customer_id | Varchar(10) | Primary key |
| C_name C_f_name C_l_name | Varchar(15) Varchar(15) | Not null Not null |
| dob | date | Not null |
| age | Number(3) | Derived from dob timestampdiff(year,dob,sysdate()) |
| Date_joined | date | Not null |
| Address Line 1 Line 2 City State pincode | Varchar(30) Varchar(30) Varchar(20) Varchar(20) Number(8) | Not null Not null Not null Not null Not null |
| height | Number(4) | Not null |
| Weight_on_joining | Number(4) | Not null |
| contact | Number(12) | Not null, multivalued |

Membership:

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|-----------|----------------|---|
| M_id | Varchar(10) | Primary key Check(substring(M_id,1,1)='M') |
| M_level | Varchar(10) | Not null |
| M_rate | Number(5) | Not null |

| | | |
|--------------|-----------|---|
| M_length | Number(5) | Not null |
| Date_created | Date | Not null |
| Expiry_date | Date | Not null derived from m_length and date_created |

EXERCISE

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|-------------|----------------|---|
| Ex_id | Varchar(10) | Primary key Check(substring(EX_id,1,2) ='EX') |
| Ex_name | Varchar(20) | Not null |
| description | Varchar(50) | |

Class

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|----------------|----------------|---|
| Class_id | Varchar(10) | Primary key Check(substring(cL_id,1,2) ='CL') |
| Class_type | Varchar(20) | Not null |
| Gym_room_no | Number(4) | Not null |
| Start_date | Date | Not null |
| Time | Time | Not null |
| Duration | Number(4) | Not null |
| Class_status | Varchar(10) | Not null Check(c_s in ('yet to begin', 'in progress', 'completed')) |
| Cost | Number(5) | |
| Prereqs | Varchar(20) | |
| Class_max_size | Number(3) | Not null Check(c_m_s<50) |
| Instructor_id | Varchar(10) | Foreign key (staff.staff_id) |
| E_id | Varchar(10) | Foreign key (equipment.e_id) |
| Ex_id | Varchar(50) | Foreign key(exercise.ex_id) Multivalued |

Workout_regiment_package

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|-----------------|----------------|---|
| Package_id | Varchar(10) | Primary key Check(substring(P_id,1,1) ='P') |
| Package_details | Varchar(50) | Not null |
| Cost | Number(5) | Not null |
| duration | Number(3) | Not null |
| E_id | Varchar(10) | Foreign key (equipment.e_id) |
| Ex_id | Varchar(50) | Foreign key (exercise.ex_id) Multivalued |
| Instructor_id | Varchar(10) | Foreign key (staff.staff_id) |

Staff :

| ATTRIBUTE | DATATYPE(SIZE) | CONSTRAINTS |
|------------------------------|----------------------------|--|
| Staff_id | Varchar(10) | Primary key Check(substring(s_id,1,2) in('SI','SM')) |
| category | Varchar(10) | Not null Check(category in('Manager','Instructor')) |
| Name S_f_name S_l_name | Varchar(10) Varchar(10) | Not null Not null |
| dob | date | Not null |
| Age | Number(3) | Derived from dob |
| Date_of_join | date | Not null Check((doj-dob)>21) |
| salary | Number(7) | Not null |

IMPLEMENTATION:

INTRODUCTION:

The implementation of this project has been carried out in Mysql Database and hence all specified functions deliver output in the similar manner.

DDL AND DML QUERIES:

TABLE CREATION

```
create database j1;

use j1;

create table membership(
m_id varchar(10) primary key constraint m_id_ck check(substring(m_id,1,1)='M'),
m_level numeric(3),
m_rate numeric(6),
m_length numeric(5));

insert into membership values('m_id',m_level,m_rate,m_length);

create table customer(
cus_id varchar(10) primary key constraint c_id_ck check(substring(c_id,1,1)='C'),
f_name varchar(15) not null,
l_name varchar(20) not null,
height numeric(4) not null,
weight_on_joining numeric(4) not null,
dob date not null constraint dob_ck check(timestampdiff(year,dob,sysdate())>0),
age numeric(3) generated always as (timestampdiff(year,dob,sysdate())),
date_joined date default (date(sysdate())));

create table new_membership(
cus_id varchar(10) not null,
m_id varchar(10) not null ,
m_length numeric(4),
start_date date default (date(sysdate())),
expiry_date date,
constraint mem_cus_fk foreign key(cus_id) references customer(cus_id),
```

```

constraint mem_m_id_fk foreign key(m_id) references membership(m_id));
#insert into new_membership(cus_id,m_id,start_date) values(cus_id,m_id,start_date);
#update new_membership set m_length=(select m_length from membership where
new_membership.m_id=membership.m_id);
#update new_membership set expiry_date=date_add(start_date, interval m_length day);
#delete from new_membership where timestampdiff(day,expiry_date,date(sysdate()))>0;
# delete from new_membership where cus_id=<cus_id>;
create table cus_contact(
cus_id varchar(10) unique not null,
contact1 numeric(12) not null,
contact2 numeric(12) not null,
contact3 numeric(12),
constraint c_con_fk foreign key(cus_id) references customer(cus_id) on delete cascade );
#insert into cus_contact values(cus_id,contact1,contact2,contact3);
# update cus_contact set contact(1,2,3)=contact_no where cus_contact.cus_id=<cus_id>;
create table cus_address(
cus_id varchar(10) unique not null,
add1 varchar(25) not null,
add2 varchar(25) ,
city varchar(15) not null,
state varchar(15) not null,
pincode numeric(10) not null,
constraint cus_add_fk foreign key(cus_id) references customer(cus_id) on delete cascade);
create table equipment(
e_id varchar(10) not null unique constraint e_id_ck check(substring(e_id,1,1)='E'),
e_name varchar(20),
e_condition varchar(15) not null constraint e_con_ck check(e_condition in
('functional','damaged')),
rent_cost numeric(5) not null,
primary key(e_id,e_condition));
create table equipment_rent(

```

```

cus_id varchar(10) not null,
e_id varchar(10) not null,
e_condition varchar(15) not null constraint con_ck check(e_condition='functional'),
rent_date date not null,
rent_cost numeric(5) not null,
constraint rent_c_id_fk foreign key(cus_id) references customer(cus_id),
constraint rent_e_id_fk foreign key(e_id) references equipment(e_id) on delete cascade,
constraint rent_e_con_fk foreign key(e_condition) references equipment(e_condition) on
delete
cascade);
create table exercise(
ex_id varchar(10) primary key constraint ex_id_ck check(substring(ex_id,1,2)='EX'),
ex_name varchar(20) not null,
ex_description varchar(50));
create table staff(
staff_id varchar(10) constraint s_id_ck check(substring(staff_id,1,1)='S'),
f_name varchar(20) not null,
l_name varchar(20) not null,
dob date not null,
age numeric(3),
category varchar(15) constraint cate_ck check(lower(category) in('manager','instructor')),
date_of_join date,
salary numeric(8) not null,
primary key(staff_id,category));
create table staff_contact(
staff_id varchar(10) not null unique,
category varchar(10) not null,
contact1 numeric(12) not null,
contact2 numeric(12),
contact3 numeric(12),
constraint st_con_fk foreign key(staff_id,category) references staff(staff_id ,category) );

```

```

create table staff_add(
staff_id varchar(10) not null unique,
category varchar(10) not null,
add1 varchar(50) not null,
add2 varchar(50) ,
city varchar (20) not null,
state varchar(20) not null,
constraint staff_add_fk foreign key(staff_id,category) references staff(staff_id,category));
create table instructors(
staff_id varchar(10) not null unique,
f_name varchar(10) not null,
l_name varchar(10) not null,
certificates varchar(100));
# insert into instructors(staff_id,f_name,l_name) select staff_id,f_name,l_name from staff
where
lower(staff.category)='instructor';
# update instructors set certificates='<certificates name 1> ..... ' where staff_id='<given
staff_id>';
create table managers(
staff_id varchar(10) not null unique,
f_name varchar(10) not null,
l_name varchar(10) not null);
# insert into managers(staff_id,f_name,l_name) select staff_id,f_name,l_name from staff
where
lower(staff.category)='manager';
create table class(
class_id varchar(10) primary key constraint class_ck check(substring(class_id,1,2)='CL'),
title varchar(30) not null,
class_type varchar(30) not null,
available_space numeric(10) not null constraint space_ck check(available_space>=0),
start_date date not null,

```



```

duration numeric(3) not null,
c_time numeric(3) not null,
cost numeric(5));
create table class_equipment(
class_id varchar(10) not null,
e_id varchar(10) not null,
constraint class_eq_cl_fk foreign key(class_id) references class(class_id) ,
constraint class_eq_e_fk foreign key(e_id) references equipment(e_id));
create table class_exercise(
class_id varchar(10) not null,
ex_id varchar(10) not null,
constraint class_ex_cl_fk foreign key(class_id) references class(class_id) ,
constraint class_ex_ex_fk foreign key(ex_id) references exercise(ex_id));
create table class_registration(
class_id varchar(10) not null,
cus_id varchar(10) not null,
constraint class_reg_cl_fk foreign key(class_id) references class(class_id),
constraint class_reg_cus_fk foreign key(cus_id) references customer(cus_id));
# insert into class_registration(class_id,cus_id) select class_id,cus_id from class,customer
where
class.available_space>0;
# update class set available_space=available_space-1 where
class_registration.class_id=class_class_id;
create table workout_regiment(
reg_id varchar(10) primary key constraint reg_ck check(substring(reg_id,1,1)='R'),
reg_details varchar(100) not null,
reg_level numeric(4) not null,
cost numeric(5) not null,
duration numeric(4) not null);
create table workout_reg_registration(
reg_id varchar(10) not null,

```

```

cus_id varchar(10) not null unique,
constraint reg_r_id_fk foreign key(reg_id) references workout_regiment(reg_id),
constraint reg_cus_id_fk foreign key(cus_id) references customer(cus_id) on delete cascade);
create table workout_exercise(
reg_id varchar(10) not null,
ex_id varchar(10) not null,
ex_id2 varchar(10) not null,
constraint we_reg_fk foreign key(reg_id) references workout_regiment(reg_id) on delete
cascade,
constraint we_ex1_fk foreign key(ex_id) references exercise(ex_id),
constraint we_ex2_fk foreign key(ex2_id) references exercise(ex2_id));

```

SQL QUERIES:

1) QUERY TO ESTABLISH JOIN BETWEEN TABLES CUSTOMER AND CUS_CONTACT TO SHOW CUSTOMER CONTACT

QUERY:

```

SELECT CUSTOMER.F_NAME,CUSTOMER.L_NAME,CUS_CONTACT.* from customer join cus_contact
on customer.cus_id=cus_contact.cus_id where customer.cus_id <> "C0000";

```

| F_NAME | L_NAME | cus_id | contact1 | contact2 | contact3 |
|---------|------------|--------|------------|------------|------------|
| WESSLEY | GIBBIONS | C0001 | 9464532198 | 9976243563 | NULL |
| MICHEAL | PRATT | C0002 | 9765432198 | 7564565434 | 7927655340 |
| JAKE | BURRY | C0003 | 9429113103 | 9420109876 | NULL |
| DOROTHY | MICHEALSON | C0004 | 9464532198 | 9977663451 | 7926611098 |

4 rows in set (0.00 sec)

2) A VIEW QUERY FOR CUSTOMER NAME AND ADDRESS

QUERY:

```

CREATE OR REPLACE VIEW CUS_ADDRESS_DETAILS AS SELECT
CUSTOMER.F_NAME,CUSTOMER.L_NAME, CUS_ADDRESS.* FROM CUSTOMER JOIN
CUS_ADDRESS ON CUSTOMER.CUS_ID=CUS_ADDRESS.CUS_ID WHERE CUSTOMER.CUS_ID
<> "C0000";

```

| F_NAME | L_NAME | cus_id | HOUSE_NO | APP_COLONY_NAME | city | state | pincode |
|---------|------------|--------|----------|--------------------|-----------|---------|---------|
| WESSLEY | GIBBIONS | C0001 | 32 | SHIVALIK PLAZA | AHMEDABAD | GUJARAT | 380006 |
| MICHEAL | PRATT | C0002 | G-401 | GOTA APPARTMENTS | AHMEDABAD | GUJARAT | 380018 |
| JAKE | BURRY | C0003 | B-502 | PALAN SOCIETY | AHMEDABAD | GUJARAT | 380025 |
| DOROTHY | MICHEALSON | C0004 | 904 | SHARDA APPARTMENTS | AHMEDABAD | GUJARAT | 380011 |

4 rows in set (0.07 sec)

- 3) CUSTOMER LINKED TO VIEW OF ADDRESS VIEW TO GET BODY-MASS INDEX OF EACH CUSTOMER:

QUERY:

```
SELECT CUSTOMER.F_NAME , CUSTOMER.L_NAME, CUSTOMER_PROPORTIONS.* FROM
CUSTOMER JOIN CUSTOMER_PROPORTIONS ON
CUSTOMER.CUS_ID=CUSTOMER_PROPORTIONS.CUS_ID WHERE CUSTOMER.CUS_ID
IN(SELECT CUS_ID FROM CUS_ADDRESS_DETAILS);
```

| F_NAME | L_NAME | cus_id | height | weight | BMI |
|---------|------------|--------|--------|--------|---------|
| WESSLEY | GIBBIONS | C0001 | 177 | 69 | 22.0243 |
| MICHEAL | PRATT | C0002 | 169 | 58 | 20.3074 |
| JAKE | BURRY | C0003 | 154 | 85 | 35.8408 |
| DOROTHY | MICHEALSON | C0004 | 182 | 78 | 23.5479 |

- 4) CUSTOMER TO BE LINKED TO SPECIFIC MEMBERSHIP AND START DATE:

QUERY:

```
SELECT CUSTOMER.F_NAME ,CUSTOMER.L_NAME,CURRENT_MEMBERSHIP.* FROM
CUSTOMER JOIN CURRENT_MEMBERSHIP ON
CUSTOMER.CUS_ID=CURRENT_MEMBERSHIP.CUS_ID WHERE CUSTOMER.CUS_ID
<>"C0000";
```

| F_NAME | L_NAME | cus_id | m_id | start_date |
|---------|------------|--------|-------|------------|
| WESSLEY | GIBBIONS | C0001 | M0001 | 2020-10-12 |
| MICHEAL | PRATT | C0002 | M0002 | 2020-10-12 |
| JAKE | BURRY | C0003 | M0003 | 2020-10-12 |
| DOROTHY | MICHEALSON | C0004 | M0004 | 2020-10-12 |

4 rows in set (0.13 sec)

- 5) TO DETERMINE MEMBERSHIP EXPIRY DATE FOR EACH CUSTOMER WE FIND:

QUERY:

```
SELECT CUS_MEMBERSHIP.F_NAME,CUS_MEMBERSHIP.L_NAME,
CUS_MEMBERSHIP.CUS_ID, DATE_ADD(CUS_MEMBERSHIP.START_DATE, INTERVAL
MEMBERSHIP.M_LENGTH DAY) AS MEM_EXPIRY_DATE FROM MEMBERSHIP JOIN
CUS_MEMBERSHIP ON MEMBERSHIP.M_ID=CUS_MEMBERSHIP.M_ID ;
```

| F_NAME | L_NAME | cus_id | MEM_EXPIRY_DATE |
|---------|------------|--------|-----------------|
| WESSLEY | GIBBIONS | C0001 | 2020-11-11 |
| MICHEAL | PRATT | C0002 | 2021-01-10 |
| JAKE | BURRY | C0003 | 2021-04-10 |
| DOROTHY | MICHEALSON | C0004 | 2021-10-07 |

4 rows in set (0.09 sec)

- 6) MAKE VIEW FOR THE ABOVE RESULT:

QUERY:

```
CREATE OR REPLACE VIEW CUS_MEMBERSHIP_EXPIRY AS SELECT
CUS_MEMBERSHIP.F_NAME,CUS_MEMBERSHIP.L_NAME, CUS_MEMBERSHIP.CUS_ID,
DATE_ADD(CUS_MEMBERSHIP.START_DATE, INTERVAL MEMBERSHIP.M_LENGTH DAY) AS
MEM_EXPIRY_DATE FROM MEMBERSHIP JOIN CUS_MEMBERSHIP ON
MEMBERSHIP.M_ID=CUS_MEMBERSHIP.M_ID ;
```

| F_NAME | L_NAME | CUS_ID | MEM_EXPIRY_DATE |
|---------|------------|--------|-----------------|
| WESSLEY | GIBBIONS | C0001 | 2020-11-11 |
| MICHEAL | PRATT | C0002 | 2021-01-10 |
| JAKE | BURRY | C0003 | 2021-04-10 |
| DOROTHY | MICHEALSON | C0004 | 2021-10-07 |

4 rows in set (0.08 sec)

7) QUERY TO LINK STAFF WITH CORRESPONDING ADDRESS AND FORMULATING A FINAL VIEW:

QUERY:

```
CREATE OR REPLACE VIEW STAFF_ADDRESS_DETAILS AS SELECT
STAFF.F_NAME,STAFF.L_NAME, STAFF_ADD.* FROM STAFF JOIN STAFF_ADD ON
STAFF.STAFF_ID=STAFF_ADD.STAFF_ID;
```

| F_NAME | L_NAME | staff_id | category | house_no | app_colony | city | state |
|---------|---------|----------|----------|--------------------|------------|---------|--------|
| JAMES | HALPERT | S0001 | 32 | SHIVALIK PLAZA | AHMEDABAD | GUJARAT | 380006 |
| PAMELA | BEASLY | S0002 | G-401 | GOTA APPARTMENTS | AHMEDABAD | GUJARAT | 380018 |
| MICHEAL | SCOTT | S0003 | B-502 | PALAN SOCIETY | AHMEDABAD | GUJARAT | 380025 |
| JACOB | WILLAMS | S0004 | 904 | SHARDA APPARTMENTS | AHMEDABAD | GUJARAT | 380011 |

8) VIEW FOR STAFF WITH CONTACT:

QUERY:

```
CREATE OR REPLACE VIEW STAFF_CONTACT_DETAILS AS SELECT
STAFF.F_NAME,STAFF.L_NAME, STAFF_CONTACT.* FROM STAFF JOIN STAFF_CONTACT ON
STAFF.STAFF_ID=STAFF_CONTACT.STAFF_ID;
```

| F_NAME | L_NAME | staff_id | category | contact1 | contact2 | contact3 |
|---------|---------|----------|------------|------------|------------|------------|
| JAMES | HALPERT | S0001 | INSTRUCTOR | 9464532198 | 9976243563 | NULL |
| PAMELA | BEASLY | S0002 | MANAGER | 9765432198 | 7564565434 | 7927655340 |
| MICHEAL | SCOTT | S0003 | INSTRUCTOR | 9429113103 | 9420109876 | NULL |
| JACOB | WILLAMS | S0004 | MANAGER | 9464532198 | 9977663451 | 7926611098 |

4 rows in set (0.03 sec)

9) VIEW FOR CUSTOMER TABLE:

QUERY:

CREATE OR REPLACE VIEW CUS AS SELECT * FROM CUSTOMER WHERE CUS_ID<> "C0000";

| cus_id | cus_password | f_name | l_name | dob | date_joined |
|--------|--------------|---------|------------|------------|-------------|
| C0001 | WESGIB233 | WESSLEY | GIBBIONS | 1993-12-17 | 2020-10-06 |
| C0002 | MICPRATT22 | MICHEAL | PRATT | 1986-10-04 | 2020-10-06 |
| C0003 | THEJAKEBOY00 | JAKE | BURRY | 1995-11-10 | 2020-10-06 |
| C0004 | DOROTHYMM | DOROTHY | MICHEALSON | 2003-06-18 | 2020-10-06 |

4 rows in set (0.06 sec)

10) STAFF VIEW :

QUERY:

CREATE OR REPLACE VIEW STAFF_DETAILS AS SELECT * FROM STAFF;

| staff_id | f_name | l_name | dob | age | category | date_of_join | salary |
|----------|---------|---------|------------|-----|------------|--------------|--------|
| S0001 | JAMES | HALPERT | 2001-01-01 | 19 | INSTRUCTOR | 2020-10-12 | 10000 |
| S0002 | PAMELA | BEASLY | 1999-06-03 | 21 | MANAGER | 2020-10-12 | 10000 |
| S0003 | MICHEAL | SCOTT | 1998-09-04 | 22 | INSTRUCTOR | 2020-10-12 | 10000 |
| S0004 | JACOB | WILLAMS | 1997-10-11 | 23 | MANAGER | 2020-10-12 | 10000 |

4 rows in set (0.04 sec)

PL/SQL PROGRAMS:

1) Change length of Gym membership

```
declare

m_len membership.m_length%TYPE;
Cursor m_length_cur is
select m_length from membership where m_id!='M0000';
begin
open m_length_cur;
loop
fetch m_length_cur into m_len;
exit when m_length_cur%NOTFOUND;
update membership set m_length=m_length+5 where m_length=m_len;
end loop;
Close m_length_cur;
end;
```

Original length:

| | m_length |
|---|----------|
| ▶ | 30 |
| | 90 |
| | 180 |
| | 360 |

New length:

| | new_length |
|---|------------|
| ▶ | 35 |
| | 95 |
| | 185 |
| | 365 |

2) Calculating BMI of Customers:

```
declare

c_id Customer_proportions.cus_id%TYPE;
height Customer_proportions.height%TYPE;
weight Customer_proportions.weight%Type;
```

```

BMI_c float;
Cursor cus_id_cur is
select cus_id from Customer_proportions;
function BMI(height Customer_proportions.height%TYPE, weight Customer_proporti
ons.weight%Type)
return float
is
BMI float;
begin
BMI:= weight/((height/100)*(height/100));
end;
begin
open cus_id_cur;
loop
fetch cus_id_cur into c_id;
exit when cus_id_cur%NOTFOUND;
height:= select height from Customer_proportions where cus_id=c_id;
weight:= select weight from Customer_proportions where cus_id=c_id;
BMI_c:= BMI(height,weight);
update Customer_proportions set BMI='BMI_c' where cus_id=c_id;
end loop;
close cus_id_cur;
end;

```

Output:

| | BMI |
|---|---------|
| ► | 22.0243 |
| | 20.3074 |
| | 35.8408 |
| | 23.5479 |

3) Calculating number of customers of a particular membership_id:

```

declare
c_id new_membership.cus_id%TYPE;
mem_id new_membership.m_id%TYPE;

```



```

Res number(5);
Function cus_mem_count(mem_id new_membership.m_id%TYPE)
return number is
f number;
begin
f:= select count(cus_id) from new_membership where m_id=mem_id group by m_id;
end;
begin
mem_id:= :membership_id;
Res:= cus_mem_count(mem_id);
Dbms_output.put_line(Res);
end;

```

Output:

| | cus_count |
|---|-----------|
| ▶ | 22 |

4) Trigger for Customer age:

```

Trigger cus_age
after
insert
on customer
for each row
begin
update customer set age=floor(months_between(Sysdate,Customer.dob)/12);
end;

```

on inserting one record:

where DOB='03-03-1993'

age column reads:

| | age |
|---|-----|
| ▶ | 27 |

5) Staff age:

```

Procedure staff_age() is
begin
update customer set age=floor(months_between(Sysdate,patient.dob)/12);
end;
begin
staff_age();

```

end;

on inserting one record:

where DOB='03-03-1988'

age column reads:

| | age |
|---|-----|
| ▶ | 32 |

6) Calculate membership expiry date:

```
declare
mem_id Membership.m_id%TYPE;
m_len Membership.m_length%TYPE;
start_date date;
expiry_date date;
Function expiry(mem_id,start_date) return date is
f date;
begin
m_len:= select m_length from Membership where m_id=mem_id;
f:= DATEADD(start_date, INTERVAL m_len);
return f;
end;
begin
mem_id:= :Membership_id;
start_date:= :start_date;
expiry_date:= expiry(mem_id,start_date);
dbms_output.put_line(expiry_date);
end;
```

Post input

Output :

| | expiry_date |
|---|-------------|
| ▶ | 2021-02-10 |

7)Count number of Instructors:

```
declare
num number;
s_id Staff.staff_id%TYPE;
CURSOR Instructor_cur is
```

```

select staff_id from Staff where staff_category='Instructor';
begin
num:=0;
open Instructor_cur;
loop
fetch Instructor_cur into staff_id;
exit when Instructor_cur%NOTFOUND;
num:=num+1;
end loop;
dbms_output.put_line('The gym has '||num||' Instructors');
end;

```

The gym has 12 Instructors

Statement processed.

0.01 seconds

8) Count number of Managers:

```

declare
num number;
s_id Staff.staff_id%TYPE;
CURSOR Manager_cur is
select staff_id from Staff where staff_category='Manager';
begin
num:=0;
open Manager_cur;
loop
fetch Manager_cur into staff_id;
exit when Manager_cur%NOTFOUND;
num:=num+1;
end loop;
dbms_output.put_line('The gym has '||num||' Managers');
end;

```

The gym has 5 Managers

Statement processed.

0.00 seconds

9) Calculate Male customers:

```

declare
num number;
c_id Customer.Cus_id%TYPE;

```

```

CURSOR Customer_cur is
select cus_id from Customer where Gender='M';
begin
num:=0;
open Customer_cur;
loop
fetch Customer_cur into c_id;
exit when Customer_cur%NOTFOUND;
num:=num+1;
end loop;
dbms_output.put_line('The gym has '||num||' Male Customers');
end;

```

```
This gym has 24 Male Customers
```

```
Statement processed.
```

```
0.00 seconds
```

10) Female Instructors:

```

declare
num number;
s_id Staff.staff_id%TYPE;
CURSOR Instructor_cur is
select staff_id from Staff where staff_category='Instructor' and gender='F';
begin
num:=0;
open Instructor_cur;
loop
fetch Instructor_cur into staff_id;
exit when Instructor_cur%NOTFOUND;
num:=num+1;
end loop;
dbms_output.put_line('The gym has '||num||' Instructors');
end;

```

```
This gym has 4 Female Instructors
```

```
Statement processed.
```

```
0.01 seconds
```

SCREEN SHOTS:

Index.php:

GETFIT GYMS

Log in Register SERVICES SOCIAL MEDIA CONTACT US

Workout Regiments

Learn More

Upcoming Classes

Learn More

About us

Welcome to GETFIT Gyms
We present to you the Pinnacle of Gymnasium Experience where we offer Various Target oriented and Individual specific Workout Initiative
We also offer Various Classes to Allow all members to Stay at the peak of their Physique. We hope You Join Us and Enjoy The Experience

BMI Calculator

| BMI RANGE | OBESEITY LEVEL |
|----------------|------------------|
| Below 18.5 | UNDERWEIGHT |
| 18.5-24.0 | NORMAL |
| 24.0-29.9 | OVERWEIGHT |
| 30.0-34.9 | OBESEITY LEVEL 1 |
| 35.0 And Above | OBESEITY LEVEL 2 |

height: (in cm)

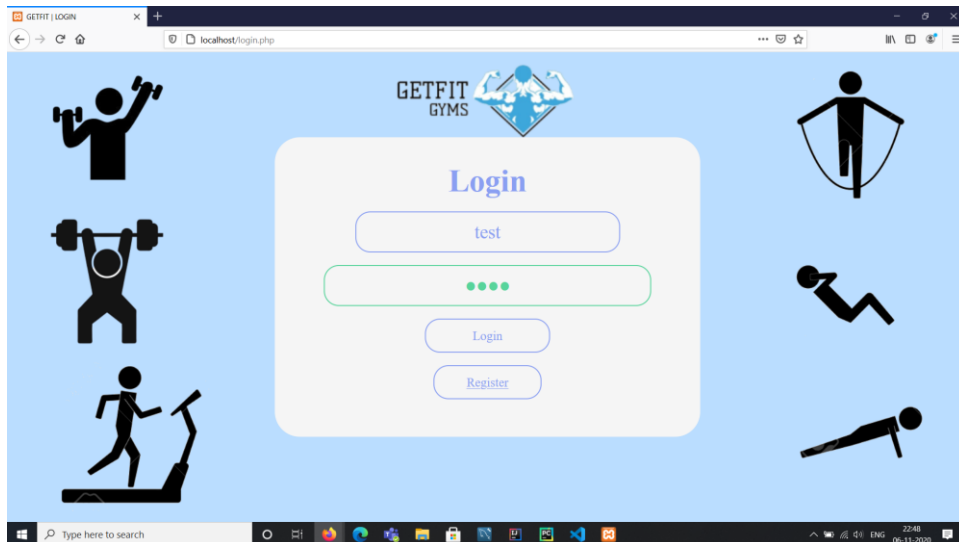
weight:

BMI:

Calculate

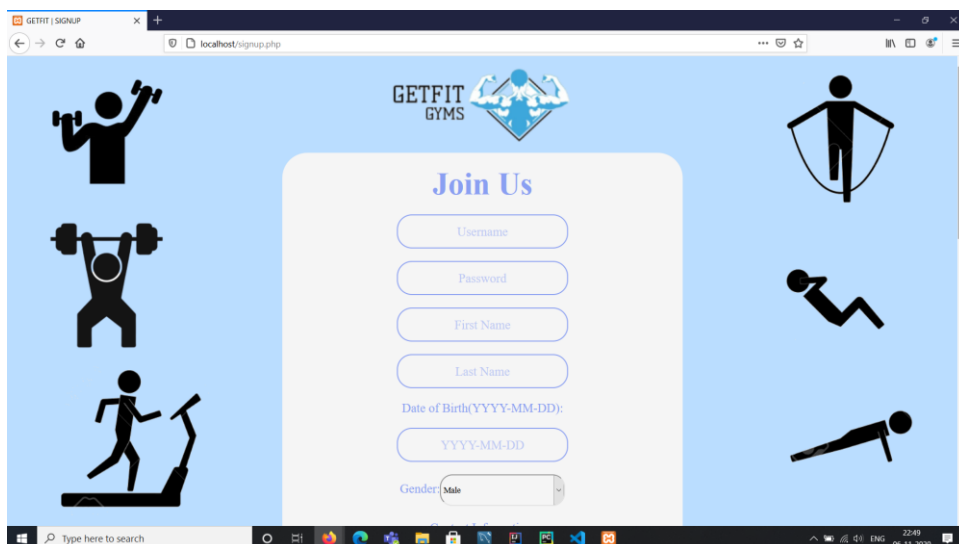
Explanation: This is the Home Page of the Website and Database management system. It Outlines Various Services like Links to Pages of Services, A functioning BMI Calculator and a reach to all Output Pages.

Login.php:



This Page allows The user / admin to log into the website /database management system
Eg: test here is a user .

Signup.php:



The image displays two sequential screenshots of a web browser window showing a sign-up form. The browser's address bar indicates the URL is `localhost/signup.php`.

First Screenshot: The form is titled "GETFIT | SIGNUP". It features two main sections: "Contact Information:" and "Address details:". Under "Contact Information:", there are input fields for "Mobile number", "Emergency Contact 1", and "Emergency Contact 2". Under "Address details:", there are input fields for "House Number and Society", "Street Name/Number", "City", "State", and "Pincode". Below these sections, the text "Please Pick your Membership" is visible.

Second Screenshot: This screenshot shows the form further down. It includes a dropdown menu for "Please Pick your Membership" (currently showing "1"), a dropdown menu for "Please Pick Number of Days" (currently showing "60"), and a dropdown menu for "Please Select Your Preferred Workout Regiment" (currently showing "Weight Loss"). A "Submit" button is located at the bottom of the form.

This Page facilitates the user to Join the Customer Tables in the website and have Their own Login.

UserPage.php:

GETFIT GYMS

test
Logout

Membership Details
Membership level: 1
Cost of Membership: Rs 2999
Membership expires in : 59 days

Workout Regiment: Weight Loss

Change

BMI Calculator:
height:
(in cm)

weight:

BMI:

Calculate

Upcoming Class (Registered): Pilates
Starts in: 45 days

Change

| BMI RANGE | OBEITY LEVEL |
|----------------|----------------|
| Below 18.5 | UNDERWEIGHT |
| 18.5-24.0 | NORMAL |
| 24.0-29.9 | OVERWEIGHT |
| 30.0-34.9 | OBEITY LEVEL 1 |
| 35.0 And Above | OBEITY LEVEL 2 |

User test own login which displays all necessary Information that they need it also gives them their own personal BMI calculator to Protect Their Information.

The Change Buttons are reference to The information above them is subject to Change.

Admin Pages:

GETFIT GYMS

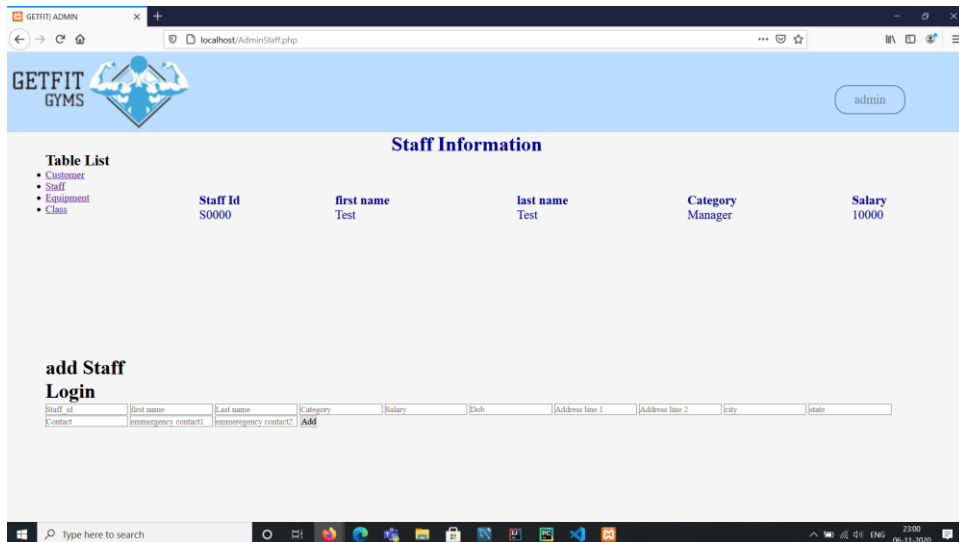
admin

Table List

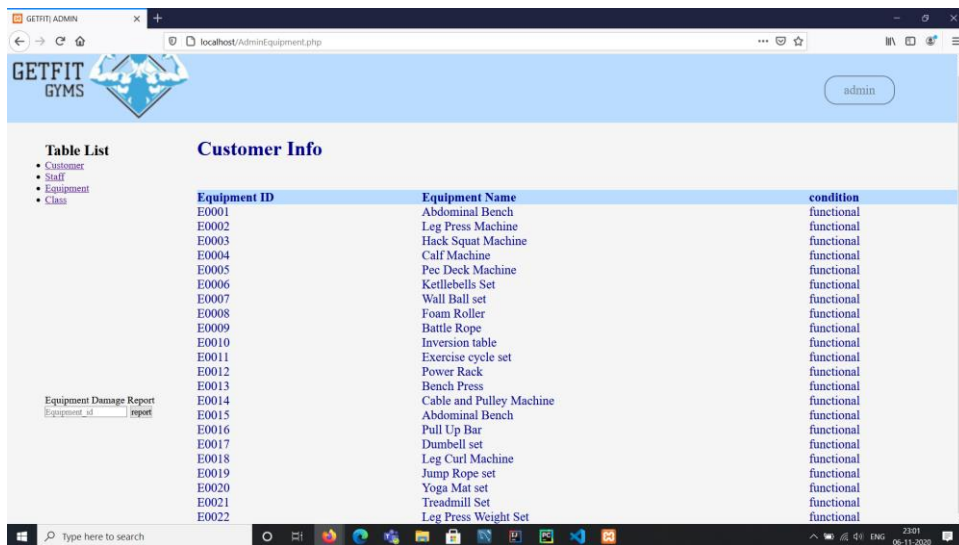
- Customer
- Staff
- Equipment
- Class

Customer Info

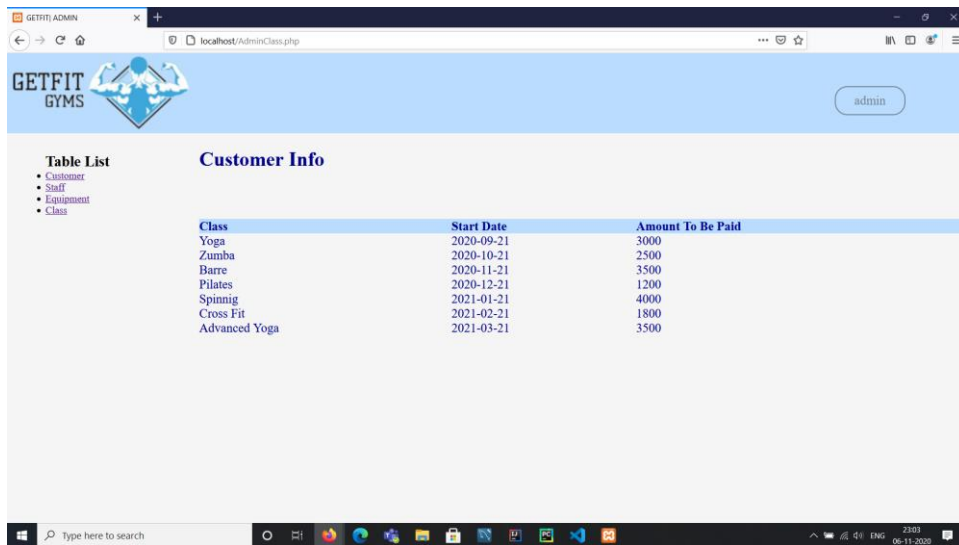
| first name | last name | Password | Membership_Level | Amount To Be Paid |
|------------|-----------|----------|------------------|-------------------|
| test | test | test | 1 | 2999 |
| test2 | test2 | test2 | 3 | 6499 |



This page also allows The admin To add or remove Staff from Database.



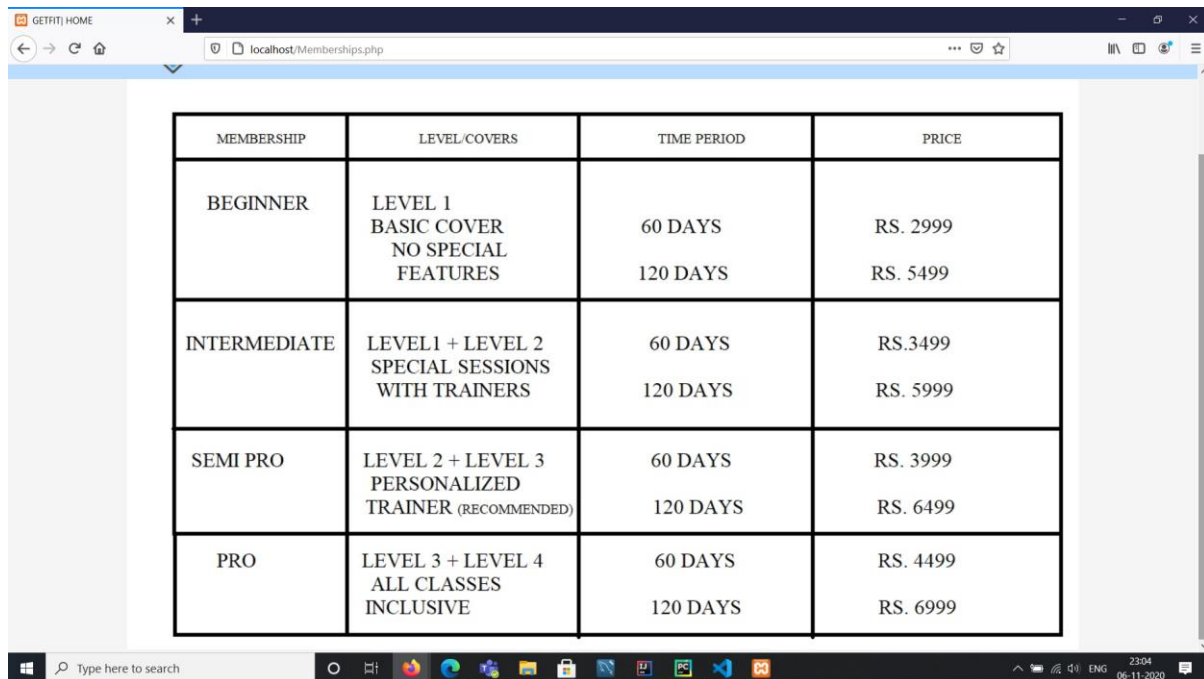
The Admin Can Also Notify any specific gym equipment damage



Brochure for workout regimnets:

| WORKOUT | REGIMENT | DESCRIPTION |
|-------------------------------------|---------------------------------|--|
| WEIGHT LOSS | | Specific focus on Regulated and Healthy Weight Loss by Cardio Exercises and Weight Training Focusing on Belly Fat , Limb Fat and Visceral fat. |
| MUSCLE GAIN | | Specialized Workout Structure to Gain effective Muscle mass and Build Muscle accordingly. Major Stress on Diet Control and Weight Training. |
| Constructive Building (recommended) | | Absolute overhaul of the body and the Understanding of varius exercises for the Upper and Lower Body for an effective Groundwork to Body Building. |
| Endurance Training | | Training For Specialized Stamina and Endurance Improvement Necessary for Advanced Workout Regiments |
| Blood Flow Training | (recomended for heart patients) | Special Exercises to enlist better Blood Circulations by Exercising the Right Muscles. |

Brochure for Membership :



The image is a screenshot of a web browser window displaying a membership brochure. The browser's address bar shows 'localhost/Memberships.php'. The table is centered on the page and contains four rows of membership options. The Windows taskbar is visible at the bottom of the screen.

| MEMBERSHIP | LEVEL/COVERS | TIME PERIOD | PRICE |
|--------------|--|-------------|----------|
| BEGINNER | LEVEL 1 BASIC COVER NO SPECIAL FEATURES | 60 DAYS | RS. 2999 |
| | | 120 DAYS | RS. 5499 |
| INTERMEDIATE | LEVEL1 + LEVEL 2 SPECIAL SESSIONS WITH TRAINERS | 60 DAYS | RS.3499 |
| | | 120 DAYS | RS. 5999 |
| SEMI PRO | LEVEL 2 + LEVEL 3 PERSONALIZED TRAINER (RECOMMENDED) | 60 DAYS | RS. 3999 |
| | | 120 DAYS | RS. 6499 |
| PRO | LEVEL 3 + LEVEL 4 ALL CLASSES INCLUSIVE | 60 DAYS | RS. 4499 |
| | | 120 DAYS | RS. 6999 |

CONCLUSION:

Since we are entering details of the customers electronically in the "GYM Management System", data will be secured. Using this application, we can retrieve customer's history with a single click. Thus, processing information will be faster. It guarantees accurate maintenance of all gym related details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

BIBLIOGRAPHY

- [1] Jerzy Letkowski, "Doing database design with MySQL" in Journal of Technology Research, Western New England University, vol. 6, December 2014.
- [2] Ž. Knok and M. Marčec, "Universtiy search engine," *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2016, pp. 897-900, doi: 10.1109/MIPRO.2016.7522267.
- [3] D. Cohen, "Database systems: Implementation of a distributed database management system to support logical subnetworks," in *The Bell System Technical Journal*, vol. 61, no. 9, pp. 2459-2474, Nov. 1982, doi: 10.1002/j.1538-7305.1982.tb03435.x.