USE CASE STUDY REPORT

Study Group II

Student Names: Aditya Kondepudi & Vignesh Sivakumar

Executive Summary:

The central objective of this study was to design and implement a relational database that is ready for the Gymnasium/recreational industry for the use in gymnasiums across the world who have constantly complained about how inputting data about the members, instructors, payment transactions and workout plans required for running a gym business is a very hectic process.
Also, there is a chance of huge amounts of data duplication, and data can be reused by the properties using a relational database. This relational database reduces data input process time by more than 50% and results in cost saving benefits across the industry. The database also implements a central analytics platform that has immense potential for analytics on the gymnasium industry by studying members information, instructor information, membership and workout plans and maintain a tab on the transactions happening in the gym.

The database was modelled taking requirements of data fields required by the gymnasium industry. The EER and UML diagrams were modelled, followed by the mapping of the conceptual model to a relational model with the required primary and foreign keys. This database was then implemented fully using MySQL as well as using MongoDB on the NoSQL environment.

The created database is a huge success, and by connecting it to Python, the analytics capabilities are immense, some of which have been shown in the study. These queries can be very helpful in tracking details about the members, their membership and workout plans, payments and information about the instructors.

I. Introduction

The Squash Busters team in the Boston Metropolitan area is in plans to develop a Recreational facility for which it needs an administration system to monitor the activities and transactions happening in the facility.

This is where a relational database comes into picture. Having a relational database for recording all the data of users, members, instructors, membership and workout plans in the facility would solve the problem of Data duplication and can shorten the data entry process time by more than 50% as the whole form need not be filled each time thereby eliminating the data duplication as well as data repetition process.

The objective is to track all the details and to have a centralized system to manage all the activities across the facility.

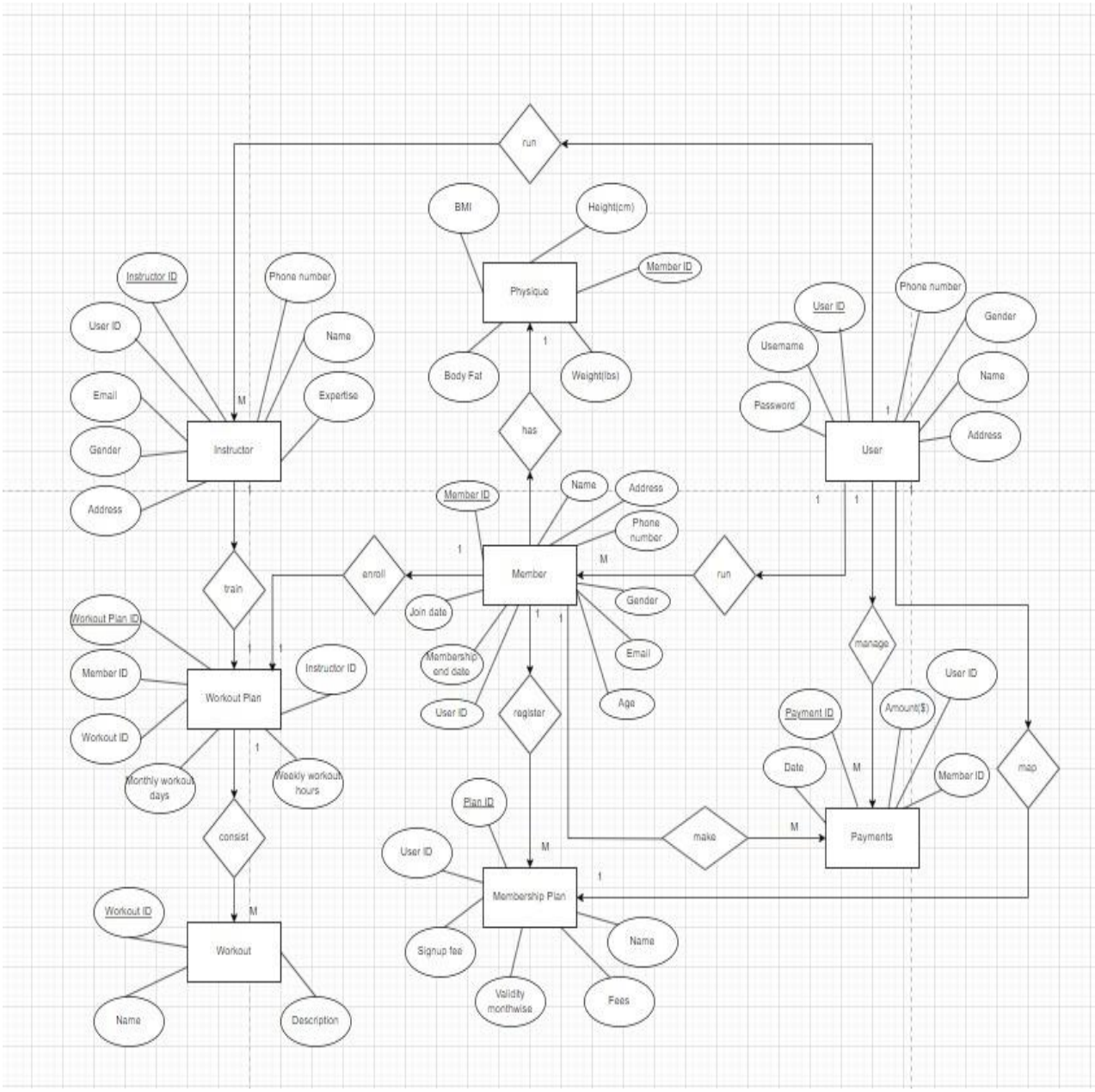
The recreational facility data contains the list of users in the facility which consist of members and the instructors. They are identified by unique IDs. There are different workouts and workout plans in the database wherein each instructor is assigned a specific workout plan. Each instructor trains one or members in the facility.

The members pay the membership fee based on the plan they have undertaken, and these transactions are maintained in the database.

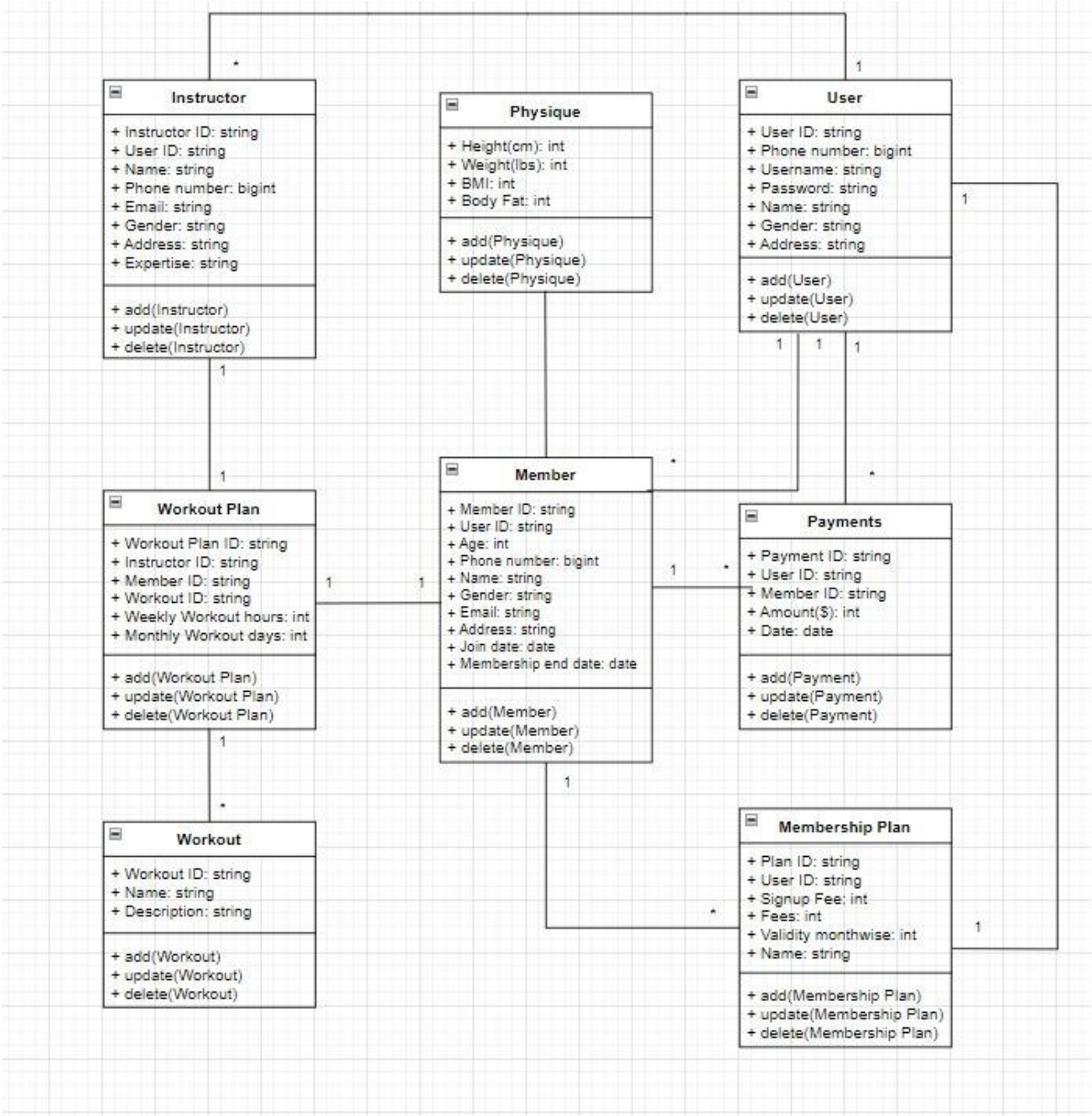
Each member has a specific body type. The system will be able to retrieve physical details of the members like BMI, height, weight, body fat, gender, etc. to maintain statistics and track the progress of the members.

II. Conceptual Data Modelling

EER Diagram



UML Diagram


Instructor
+ Instructor ID: string
+ User ID: string
+ Name: string
+ Phone number: bigint
+ Email: string
+ Gender: string
+ Address: string
+ Expertise: string
+ add(Instructor)
+ update(Instructor)
+ delete(Instructor)
Physique
+ Height(cm): int
+ Weight(lbs): int
+ BMI: int
+ Body Fat: int
+ add(Physique)
+ update(Physique)
+ delete(Physique)
User
+ User ID: string
+ Phone number: bigint
+ Username: string
+ Password: string
+ Name: string
+ Gender: string
+ Address: string
+ add(User)
+ update(User)
+ delete(User)
Workout Plan
+ Workout Plan ID: string
+ Instructor ID: string
+ Member ID: string
+ Workout ID: string
+ Weekly Workout hours: int
+ Monthly Workout days: int
+ add(Workout Plan)
+ update(Workout Plan)
+ delete(Workout Plan)
Member
+ Member ID: string
+ User ID: string
+ Age: int
+ Phone number: bigint
+ Name: string
+ Gender: string
+ Email: string
+ Address: string
+ Join date: date
+ Membership end date: date
+ add(Member)
+ update(Member)
+ delete(Member)
Payments
+ Payment ID: string
+ User ID: string
+ Member ID: string
+ Amount($): int
+ Date: date
+ add(Payment)
+ update(Payment)
+ delete(Payment)
Workout
+ Workout ID: string
+ Name: string
+ Description: string
+ add(Workout)
+ update(Workout)
+ delete(Workout)
Membership Plan
+ Plan ID: string
+ User ID: string
+ Signup Fee: int
+ Fees: int
+ Validity monthwise: int
+ Name: string
+ add(Membership Plan)
+ update(Membership Plan)
+ delete(Membership Plan)

III. mapping Conceptual Model to Relational Model

Primary Key Attributes are indicated by an underline. – _____
Foreign Key Attributes are indicated by a dashed underline – _ _ _ _ _ _ _ _

- **USER (User ID, Name, Gender, Contact, Address, Username, Password)**

- **MEMBER (Member ID, Name, Join date, Membership end date, User ID, Age, Email, Gender, Phone Number, Address,)**
  User ID foreign key refers to User ID from **USER**

- **MEMBERSHIP_PLAN (Plan ID, User ID, Signup fee, Validity, Fees, Name)**
  User ID foreign key refers to User ID from **USER**

- **INSTRUCTOR (Instructor ID, Name, Phone Number, Email, Gender, Address, User ID, Expertise)**
  User ID foreign key refers to User ID from **USER**

- **WORKOUT (Workout ID, Name, Description)**

- **WORKOUT_PLAN (Workout Plan ID, Member ID, Workout ID, Monthly workout days, Weekly workout hours, Instructor ID)**
  Workout ID foreign key refers to Workout ID from **WORKOUT**
  Member ID foreign key refers to Member ID from **MEMBER**
  Instructor ID foreign key refers to Instructor ID from **INSTRUCTOR**

- **PAYMENTS (Payment_ID, Date, Amount, User ID, Member ID)**
  User ID foreign key refers to User ID from **USER**
  Member ID foreign key refers to Member ID from **MEMBER**

- **PHYSIQUE (Member ID , BMI, Height, Weight, Body Fat)**
  Member ID foreign key refers to Member ID from **MEMBER**

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

1. Selecting members having age more than the average age

```
with temporarytable(averagevalue) as
        (select avg(m.age)
        from member m)
select m.`member id`,m.name,m.age
from member m, temporarytable
where m.age > temporarytable.averagevalue;
```

member id	name	age
M1	Aditya	26
M10	Samantha	30
M3	North	26
M9	Corey	32

2. Selecting members having BMI above or below the average

```
select m.name,m.age,ph.bmi,ph.`body fat`
from member m
inner join physique ph on
m.`member id` = ph.`member id`
where bmi > '24.9' or bmi < '18'
group by ph.bmi
order by ph.bmi desc;
```

name	age	bmi	body fat
Phil	23	35	30
Phoebe	19	30	28
North	26	28	26
Craig	24	25	24
Aditya	26	17	15

3. Selecting the instructors working on Flexibility and Endurance workout plans

```
        (select i.name,i.`instructor id`,wp.`workout plan id`,w.name
        from instructor i
        inner join workout_plan wp on
        i.`instructor id` = wp.`instructor id`
        inner join workout w ON
        wp.`workout id` = w.`workout id` where w.name = 'Flexibility
union
        (select i.name,i.`instructor id`,wp.`workout plan id`,w.name
        from instructor i
        inner join workout_plan wp on
        i.`instructor id` = wp.`instructor id`
        inner join workout w on
        wp.`workout id` = w.`workout id` where w.name = 'Endurance');
```

name	instructor...	workout plan...	name
Kelsey	I5	WP4	Flexibility
Doherty	I1	WP1	Endurance
Doherty	I1	WP2	Endurance
Natalie	I4	WP7	Endurance

4. Selecting instructor name with skillset

```
select i.name, i.expertise
from instructor as i
group by i.name, i.expertise
order by i.name asc
limit 5;
```

name	expertise
Doherty	Legs
Heather	Legs
Kelsey	Upper body
Natalie	Cardio
Roger	Upper body

5. Selecting male members in the gym

```
select u1.`user id`,u1.name,u1.gender
from user as u1
where not exists
        (select * from user as u2
        where u1.`user id` = u2.`user id`
                and u1.gender = 'Female')
```

user id	name	gender
A001	Aditya	Male
A002	Doherty	Male
A003	Adam	Male
A004	Roger	Male
A005	North	Male
A006	Craig	Male
A007	Phil	Male
A010	Smith	Male
A014	Corey	Male
NULL	NULL	NULL

6. Selecting members in the gym based on their workout plan

```
select m.name, wp.`workout id`,w.name
from workout_plan wp, member m, workout w
        where m.`member id` = wp.`member id`
        and w.`workout id` = wp.`workout id`;
```

name	workout id	name
Aditya	W1	Endurance
Adam	W1	Endurance
Foram	W1	Endurance
North	W2	Strength
Rachel	W2	Strength
Phoebe	W3	Balance
Corey	W3	Balance
Phil	W4	Flexibility

7. Selecting member id, name, age, gender, membership end date, payment info having a workout plan

```
select m.`member id`, m.name, m.gender, m.age,m.`membership end date`, p.`payment
id`,p.`amount($)`,wp.`workout plan id`
  from member m
  inner join payments p
  on m.`member id` = p.`member id`
  inner join workout_plan wp
  on m.`member id` = wp.`member id`;
```

member id	name	gender	age	membership end d...	payment id	amount($)	workout plan...
M1	Aditya	Male	26	2023-02-22	Payment1	6100	WP1
M2	Adam	Male	21	2022-03-22	Payment2	600	WP2
M3	North	Male	26	2022-05-22	Payment3	1600	WP3
M5	Phil	Male	23	2023-02-25	Payment5	6100	WP4
M6	Phoebe	Female	19	2022-08-28	Payment6	3100	WP5
M7	Rachel	Female	20	2023-02-28	Payment7	6100	WP6
M8	Foram	Female	21	2022-06-01	Payment8	1600	WP7
M9	Corey	Male	32	2022-04-01	Payment9	600	WP8

8. Total workout hours spent

```
select sum(`weekly workout hours`) from
        (select instructor.`name`, instructor.`expertise`,
         workout_plan.`weekly workout hours`
                from instructor
                inner join
                workout_plan on
                instructor.`instructor id`=workout_plan.`instructor id`) as c;
```

sum(`weekly workout hou...
203

9. Selecting physique age wise of each member

```
select m.`member id`,m.name,m.age,ph.bmi,ph.`height(cm)`,ph.`weight(lbs)`,ph.`body fat`
from member m
inner join physique ph
on m.`member id` = ph.`member id`
group by m.age
order by m.age asc;
```

member id	name	age	bmi	height(c...	weight(lbs)	body fat
M6	Phoebe	19	30	162	205	28
M7	Rachel	20	19	170	130	20
M2	Adam	21	23	167	145	20
M5	Phil	23	35	165	105	30
M4	Craig	24	25	152	130	24
M1	Aditya	26	17	165	105	15
M10	Samantha	30	30	160	170	29
M9	Corey	32	22	170	165	25

10. Retrieving the counts of each workout plan registered

```
select name, count(*)
from (Select m.`member id`, wp.`workout id`,w.name
from workout_plan wp, member m, workout w
        where m.`member id` = wp.`member id`
        and w.`workout id` = wp.`workout id`) as c
group by `name`;
```

name	count(*)
Endurance	3
Strength	2
Balance	2
Flexibility	1

NoSQL Implementation:

1. To find the number of male members registered for the gym

```
> db.user.countDocuments({'gender': 'Male'})
< 9
```

2. To find the total number of weekly workout hours in the gym

```
> db.workout_plan.aggregate([{$group: {_id: null, total:{$sum:"$weekly workout hours"}}}])
< { _id: null, total: 203 }
```

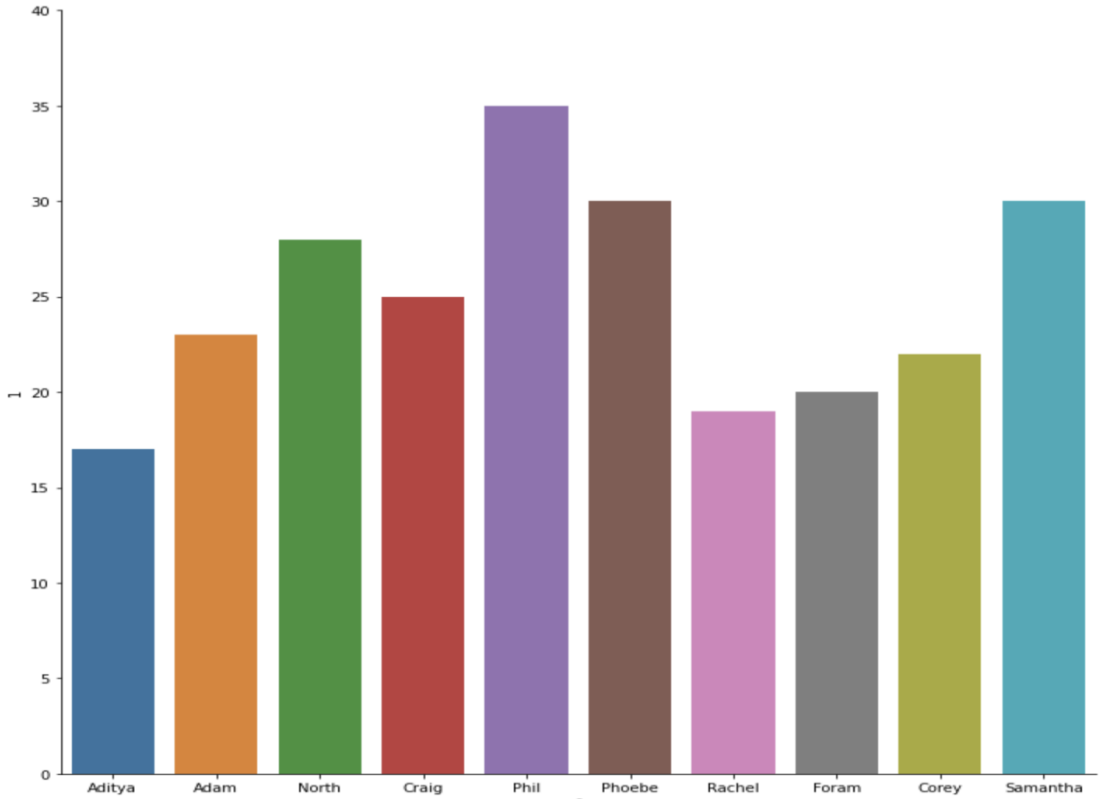
3. To find the average physique properties of the members in the gym

```
> db.physique.aggregate([{$group: {_id: "gender", AverageBMI:{$avg:"$bmi"}, AverageHeight:{$avg:"$height(cm)"}, AverageWeight:{$avg:"$weight(lbs)"}}}])
< { _id: 'gender',
    AverageBMI: 24.9,
    AverageHeight: 164.1,
    AverageWeight: 145.5 }
```

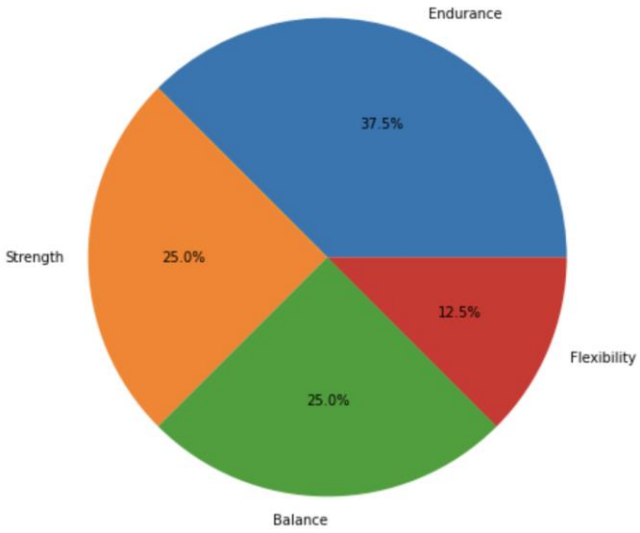
V. Database Access via Python

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetch all from query, followed by converting the list into a dataframe using pandas library and using matplotlib to plot the graphs for the analytics.

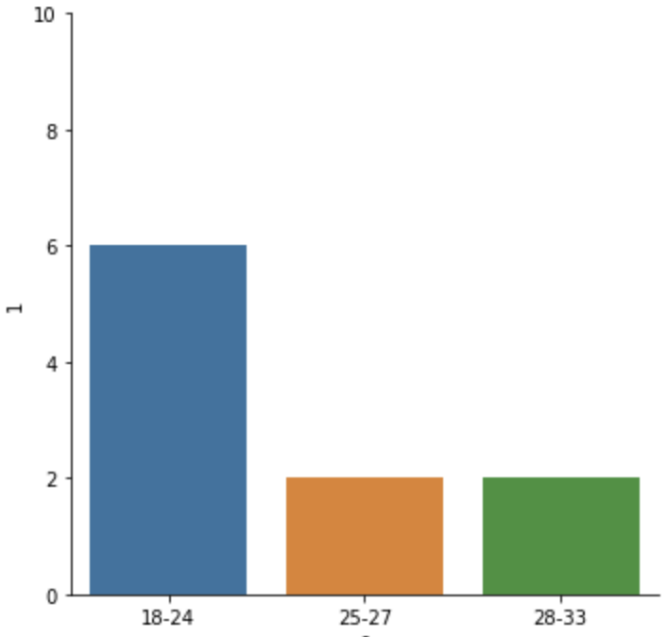
1. Plotting the bar graph each member in the gym and their corresponding BMI



2. Plotting the pie chart for each workout plan registered in the gym.



3. Plotting the bar graph for different age groups of members in the gym



VI. Summary and Recommendation

The Gymnasium management system designed on MySQL is an industry ready relational database that can be implemented in the recreational industry. It will result in great cost savings in the member and instructor data input process and provides great analytics capabilities, a small part of which is shown in this report utilizing Python.

There are a few shortcomings regarding the database system. We need to consider where the information needs to be stored in case the whole system goes down. We will also want to see how the system is stored, in a database in a cloud or how accessible it would be if the system were to go down. Any hacking in the system is not preventable unless security of the database is tightened. By adding security to the database system, frequent attacks by the hackers can be prevented and the data will be safe and secure.