I. Introduction

The dynamic nature of the fast-paced life today, has made it imperative for the people to be at their best fitness levels. But with limited amount of knowledge and understanding about how to maintain one's fitness levels, people tend to depend upon experts. Also, it is practically impossible for everyone to have all the equipment and the machines required for maintaining a good physique at their homes. So, people depend upon gyms for their needs.

But maintaining a gym is not that easy and involves various elements such as registration of members, tracking their billing details and diet plans, the equipment that the members need to use to achieve their fitness goals, maintaining payment details for various staff employed in the gym, assigning lockers to the members and staff, etc. It becomes a tedious job to track and maintain large amount of these records on paper. Hence, in this modern era of technology and internet, there needs to be a lateral shift from paper records to a database management system. This lateral shift is essential as databases have incredibly amazing features and advantages such as faster data retrieval, data backup and recovery, increased productivity, and reduced data inconsistencies.

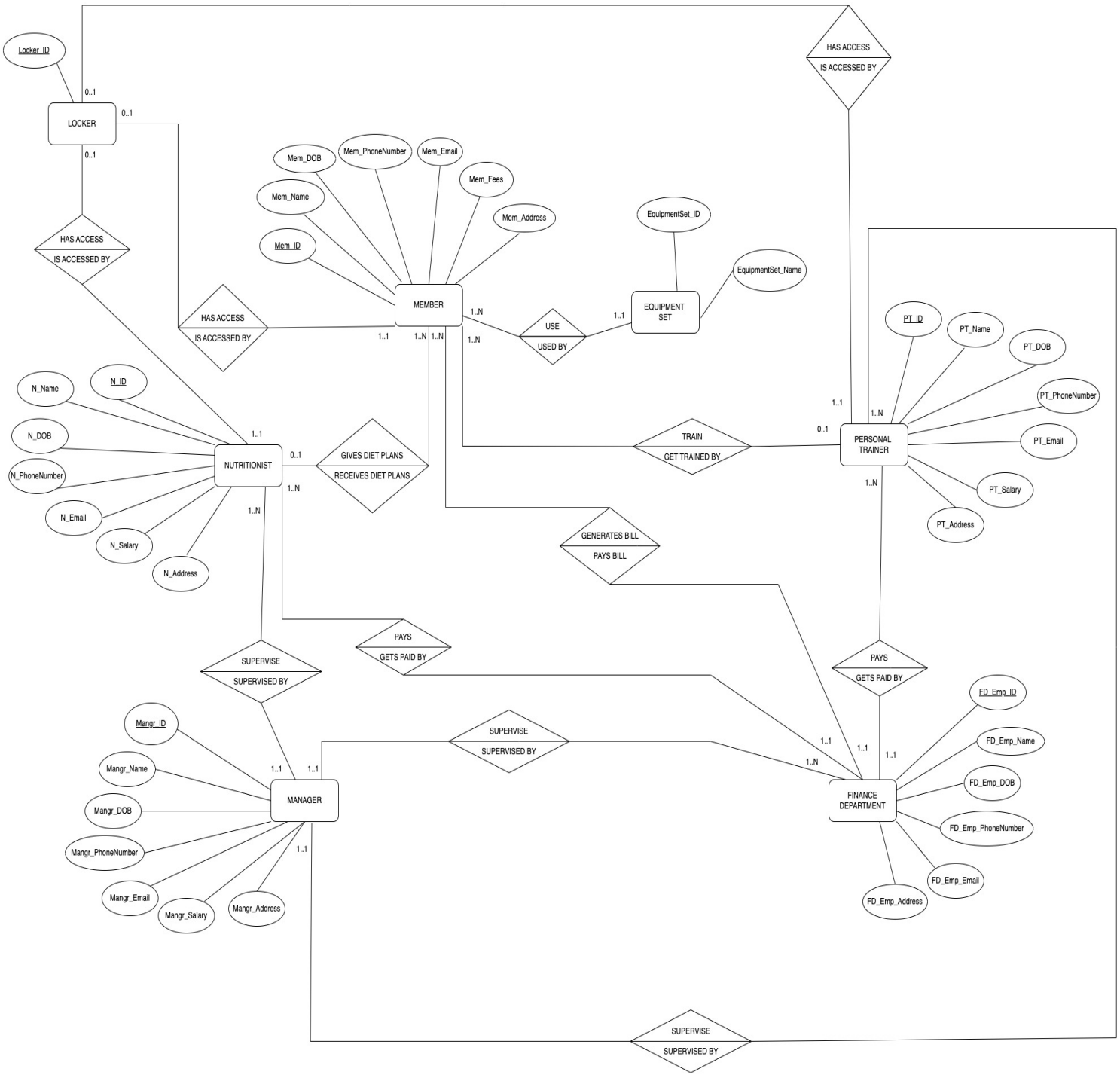
Hence, we will create a gym management system and will achieve this goal by creating 7 entities: Member, Equipment Set, Personal Trainer, Nutritionist, Locker, Finance Department, and Manager. We will establish relationships between all the entities, define the attributes for all of them and then create a database system around it.

Following are the relationships that will be implemented:

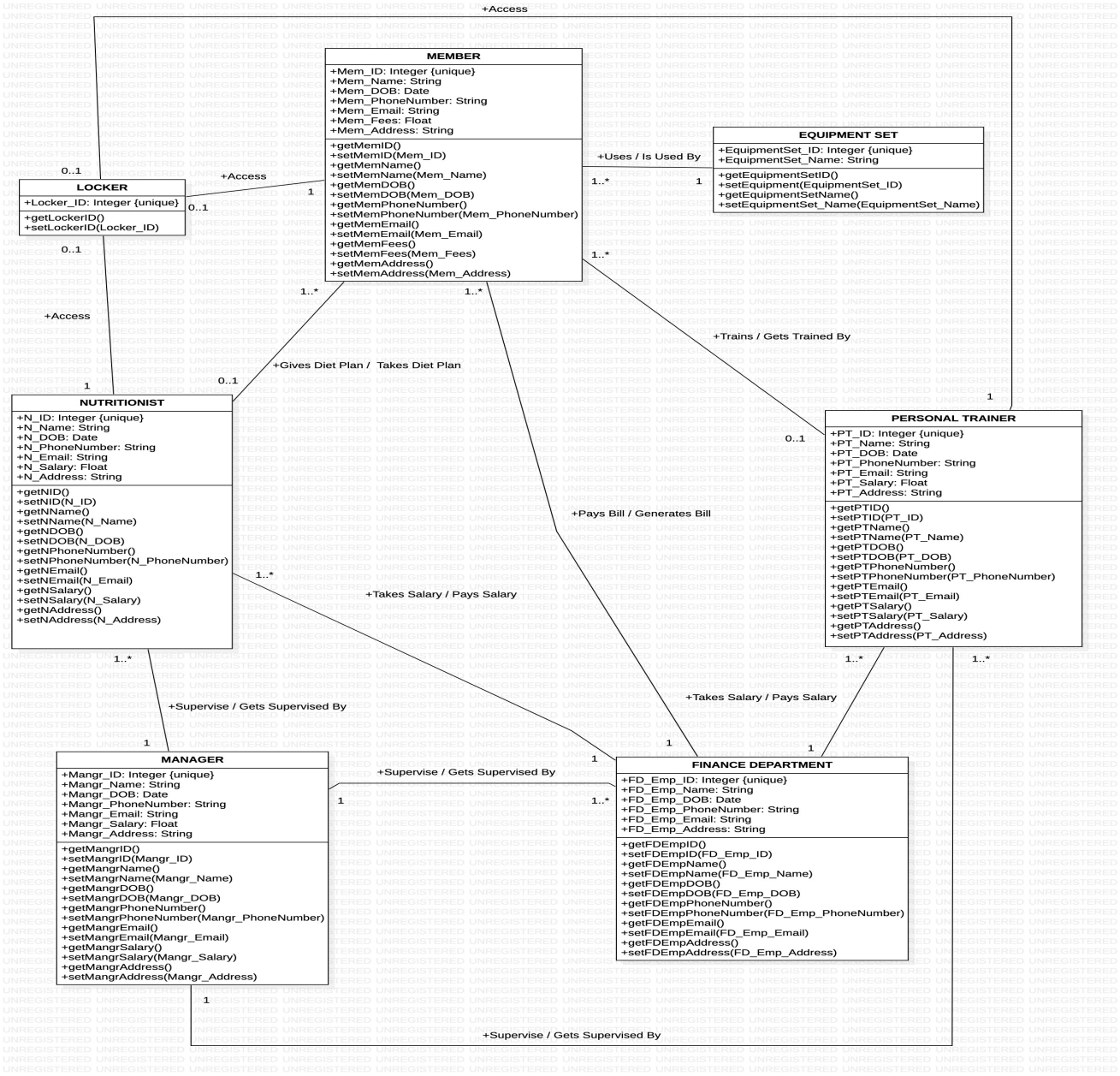
- Each member can be associated with a maximum of one nutritionist whereas a nutritionist can prepare diet plans for multiple members.
- Each member can be associated with a maximum of one personal trainer whereas a personal trainer can work with multiple members.
- Each member would a single equipment set, and each equipment set would be used by multiple members.
- Each member can have access to a maximum of one locker at a time.
- Each personal trainer would have access to a maximum of one locker at a time.
- Each nutritionist would have access to a maximum of one locker at a time.
- The Finance department would handle the billing procedure of all the members.
- The Finance department would handle the payments for each nutritionist, and personal trainer.
- Each manager would supervise multiple nutritionists, personal trainers, and the finance department.
- A nutritionist would report to a single manager.
- A personal trainer would report to a single manager.
- Each finance department member would report to a single manager.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Class Diagram



III. Mapping Conceptual Model to Relational Model

Note – The primary keys are represented by bold font and the foreign keys are represented by italics font

- **EQUIPMENT_SET** (**EquipmentSet_ID**, EquipmentSet_Name)

- **LOCKER** (**Locker_ID**)

- **MANAGER** (**Mangr_ID**, Mangr_Name, Mangr_DOB, Mangr_PhoneNumber, Mangr_Email, Mangr_Salary, Mangr_Address)

- **FINANCE_DEPARTMENT** (**FD_Emp_ID**, FD_Emp_Name, FD_Emp_DOB, FD_Emp_PhoneNumber, FD_Emp_Email, FD_Emp_Address, *Mangr_ID*)

  where Mangr_ID refers to Mangr_ID taken from MANAGER table, NOT NULL

- **NUTRITIONIST** (**N_ID**, N_Name, N_DOB, N_PhoneNumber, N_Email, N_Salary, N_Address, *Locker_ID*, *FD_Emp_ID*, *Mangr_ID*)

  where Locker_ID refers to Locker_ID taken from LOCKER table, NULL ALLOWED,
  FD_Emp_ID refers to FD_Emp_ID from FINANCE_DEPARTMENT table, NOT NULL,
  Mangr_ID refers to Mangr_ID taken from MANAGER table, NOT NULL

- **PERSONAL_TRAINER** (**PT_ID**, PT_Name, PT_DOB, PT_PhoneNumber, PT_Email, PT_Salary, PT_Address, *Locker_ID*, *FD_Emp_ID*, *Mangr_ID*)

  where Locker_ID refers to Locker_ID taken from LOCKER table, NULL ALLOWED,
  FD_Emp_ID refers to FD_Emp_ID taken from FINANCE_DEPARTMENT table, NOT NULL
  Mangr_ID refers to Mangr_ID taken from MANAGER table, NOT NULL

- **MEMBER** (**Mem_ID**, Mem_Name, Mem_DOB, Mem_PhoneNumber, Mem_Email, Mem_Fees, Mem_Address, *Locker_ID*, *PT_ID*, *N_ID*, *EquipmentSet_ID*, *FD_Emp_ID*)

  where Locker_ID refers to Locker_ID taken from LOCKER table, NULL ALLOWED,
  PT_ID refers to PT_ID taken from PERSONAL_TRAINER table, NULL ALLOWED,
  N_ID refers to N_ID taken from NUTRITIONIST table, NULL ALLOWED,
  EquipmentSet_ID refers to EquipmentSet_ID taken from EQUIPMENT_SET table, NOT NULL,
  FD_Emp_ID refers to FD_Emp_ID taken from FINANCE_DEPARTMENT table, NOT NULL

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL Workbench and the following queries were performed:

Query 1 - Display member ID, member name, member phone number, member email and their corresponding personal trainer details such as their name, phone number and email

select m.Mem_ID, m.Mem_Name, m.Mem_PhoneNumber, m.Mem_Email, pt.PT_Name, pt.PT_PhoneNumber, pt.PT_Email from member m INNER JOIN personal_trainer pt on m.PT_ID = pt.PT_ID order by Mem_ID asc;

Mem_ID	Mem_Name	Mem_PhoneNumber	Mem_Email	PT_Name	PT_PhoneNumber	PT_Email
701	Averil Madonna	214-193-2961	amadonna10@webeden.com	Kelley Towson	116-183-7518	ktowsonr@globo.com
702	Merci Smaile	237-521-8026	msmaile11@google.fr	Xenos Loughlan	323-784-8680	xloughlanl@google.co.jp
703	Lexis Canero	578-568-4892	lcanero12@com.com	Goldina Rainford	128-336-2863	grainfordq@mayoclinic.com
704	Carolina Frondt	690-613-8933	cfrondt13@cargocollective.com	Garland Ryce	688-631-5689	gryceo@bigcartel.com
706	Marinna Errigo	888-369-4487	merrigo15@plala.or.jp	Garland Ryce	688-631-5689	gryceo@bigcartel.com
707	Morey Lillecrop	809-628-5336	mlillecrop16@answers.com	Marten Mariette	939-934-1709	mmariettem@buzzfeed.com
710	Quinn Andrasch	381-691-1916	qandrasch19@seesaa.net	Charissa Smee	687-194-9115	csmeen@oracle.com
711	Fionnula Villaret	851-133-9439	fvillareta@sogou.com	Kristen Henriques	983-990-0027	khenriques2@answers.com
712	Johannah Connold	868-462-1349	jconnoldb@twitpic.com	Cynthia Tabourier	299-654-7252	ctabourier8@mysql.com
713	Erena McCurrie	335-558-3188	emccurriec@gov.uk	Sandro Cogdon	385-570-8687	scogdon5@canalblog.com
714	Clifford Troctor	448-403-7140	ctroctord@intel.com	Kelley Towson	116-183-7518	ktowsonr@globo.com
715	Malvina Ahren	318-306-0425	mahrene@wufoo.com	Othilie Dodsley	151-737-0636	ododsley4@mail.ru
716	Inger Hulburt	327-216-5759	ihulburtf@nytimes.com	Xenos Loughlan	323-784-8680	xloughlanl@google.co.jp
717	Jareb Rummer	116-150-6836	jrummerg@sitemeter.com	Brook Caldron	315-185-7097	bcaldron7@netvibes.com
718	Annie Blacksell	121-435-9649	ablacksellh@wikia.com	Meredithe Kyrke…	888-702-4561	mkyrkeman1@exblog.jp
719	Dorthy Gilchriest	587-835-6824	dgilchriesti@storify.com	Kelley Towson	116-183-7518	ktowsonr@globo.com
720	Mollie Sealey	764-802-3078	msealeyj@weebly.com	Uta Phelipeaux	195-256-4879	uphelipeaux6@pagespers…
721	Petey Pottle	356-123-0944	ppottlek@about.com	Xenos Loughlan	323-784-8680	xloughlanl@google.co.jp
722	Timmy Sellick	863-511-0940	tsellickl@clickbank.net	Garland Ryce	688-631-5689	gryceo@bigcartel.com
723	Carey Dangerfield	893-694-2594	cdangerfieldm@theguardian.c…	Vick Waddie	454-272-4416	vwaddie0@ameblo.jp
724	Jammie Celli	601-361-0776	jcellin@is.gd	Othilie Dodsley	151-737-0636	ododsley4@mail.ru

Query 2 - Display nutritionist ID, name, and salary where salary is greater than average salary of all the nutritionists

select n1.N_ID, n1.N_Name, n1.N_salary from nutritionist n1 where n1.N_Salary > (select avg(n2.N_Salary) from nutritionist n2);

N_ID	N_Name	N_salary
202	Nolan Chafer	45500.5
203	Oralla Southwell	50000
208	Rosemary Gubbins	49002.9
211	Anthea Speke	49275.1
212	Rupert Bloodworthe	47359
213	Helli Fildery	46072.7

Query 3 - Retrieve Name and ID of those managers who manage those nutritionists that give diet plans to members who pay membership fees more than 7500

select distinct(m1.Mangr_ID), m1.Mangr_Name from Manager m1 INNER JOIN Nutritionist n on (m1.Mangr_ID=n.Mangr_ID) INNER JOIN MEMBER mem on n.N_ID=mem.N_ID where mem.Mem_Fees>'7500';

Mangr_ID	Mangr_Name
1	Kunal Shastri
2	Byrom Alfwy
3	Zeke Halfacre
4	Richard Perott
5	Frank Van Merwe
6	Fred Klaasen
7	Jordan Pickford
8	Bhaichung Bhutia
9	Gareth Bale

Query 4 - Display the locker numbers used by members using EXISTS operator

select l.locker_id from locker l where EXISTS (select m.Mem_ID, m.Mem_Name from member m where m.Locker_ID = l.locker_id);

locker_id
21
22
23
25
26
27
29
30
31
32
33
36
37
38
39
41
43
44
45
46
47

Query 5 - Display member details like ID, Name, Phone Number, Email who use equipment set ID > 902 using ANY operator

select m.Mem_Id, m.Mem_Name, m.Mem_PhoneNumber, m.Mem_Email from member m where m.EquipmentSet_ID = ANY (select e.EquipmentSet_ID from Equipment_Set e where e.EquipmentSet_ID > 902);

Mem_Id	Mem_Name	Mem_PhoneNumber	Mem_Email
701	Averil Madonna	214-193-2961	amadonna10@webeden.com
704	Carolina Frondt	690-613-8933	cfrondt13@cargocollective.com
707	Morey Lillecrop	809-628-5336	mlillecrop16@answers.com
724	Jammie Celli	601-361-0776	jcellin@is.gd
725	Loleta Barthelmes	300-901-3070	lbarthelmeso@umich.edu
752	Warner Woolward	783-885-7459	wwoolward1f@google.com.hk
706	Marinna Errigo	888-369-4487	merrigo15@plala.or.jp
713	Erena McCurrie	335-558-3188	emccurriec@gov.uk
718	Annie Blacksell	121-435-9649	ablacksellh@wikia.com
721	Petey Pottle	356-123-0944	ppottlek@about.com
742	L;urette Johnsson	755-110-3527	ljohnsson15@cyberchimps.com
746	Lovell Tappington	352-365-2354	ltappington19@tiny.cc
748	Livvy Mowsdill	737-792-1577	lmowsdill1b@seattletimes.com
749	Padriac Svred	506-335-2706	psvred1c@163.com

Query 6 - Display minimum, maximum, and average salary of managers

select min(Mangr_salary) as Minimum_Salary_of_Manager, max(Mangr_salary) as Maximum_Salary_of_Manager, avg(Mangr_salary) as Average_Salary_of_Manager from manager;

Minimum_Salary_of_Mana...	Maximum_Salary_of_Mana...	Average_Salary_of_Manag...
42453.2	91234	69871.631640625

Query 7 - Retrieve the finance department employee details like ID, name, phone number and email who collects fees from more than 5 members

select fd.FD_Emp_ID, fd.FD_Emp_Name, fd.FD_Emp_PhoneNumber, fd.FD_Emp_Email from FINANCE_DEPARTMENT fd where 5 < (select count(*) from member m where m.FD_Emp_ID = fd.FD_Emp_ID);

FD_Emp_ID	FD_Emp_Name	FD_Emp_PhoneNumber	FD_Emp_Email
105	Madan Lal	333-555-8881	mlal143@lords.com
106	Vicky Donnovan	367-543-8121	donnovanvick@mysticfalls.com
107	Erlich Bachmann	412-754-3243	erlich@piedpier.com
108	Albie Morkel	332-545-5459	albiem34@csa.com
109	Tom Cruise	384-723-3448	cruisetommy56@hollywood.com

Query 8 - Display finance department employee details whose name begins with D or E

select * from finance_department where FD_Emp_Name like 'D%' or FD_Emp_Name like 'E%';

FD_Emp_ID	FD_Emp_Name	FD_Emp_DOB	FD_Emp_PhoneNumber	FD_Emp_Email	FD_Emp_Address	Mangr_ID
101	Derek Abbotson	2001-05-20 00:00:00	947-218-2617	derekabbotson@gmail.com	170 Parker Hill	2
103	Doralynn Coatsworth	1990-11-12 00:00:00	901-367-2617	dcoatsworth6@hotmail.com	881 Hungtington Ave	3
107	Erlich Bachmann	1985-03-12 00:00:00	412-754-3243	erlich@piedpier.com	857 Opera Pl	3

NoSQL Implementation:

All the tables and relations have been implemented using MongoDB Compass. The following NoSQL queries were performed.

Query 1: Display maximum, minimum and average salary of Managers

db.manager.aggregate ([{$group: {_id: null, Max_Salary: {$max: '$mangr_salary' }, Min_Salary: {$min: '$mangr_salary' }, Avg_Salary: {$avg: '$mangr_salary'}}}])

```
{ _id: null,
  Max_Salary: 91234,
  Min_Salary: 42453.19921875,
  Avg_Salary: 69871.6203125 }
```

Query 2: Display the count of Members that pay fees less than 10000

db.member.aggregate([{ $match: { mem_fees: { $lt: 10000 } } }, { $count: "count_of_members" }])

```
{ count_of_members: 27 }
```

Query 3: Display Personal Trainer details who either gets paid by Finance Department Employee number 105 or reports to Manager number 2

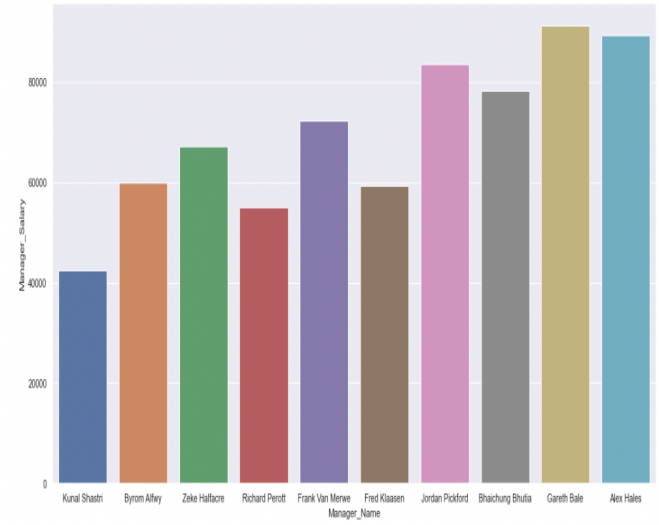
db.personal_trainer.find({$or: [{ FD_Emp_ID: 105},{ Mangr_ID: 2}]})


```
{ _id: ObjectId("638a702615d97d269fcc063a"),
  PT_ID: 301,
  PT_Name: 'Xenos Loughlan',
  PT_DOB: 1985-10-04T00:00:00.000Z,
  PT_PhoneNumber: '323-784-8680',
  PT_Email: 'xloughlanl@google.co.jp',
  pt_salary: 34635.8984375,
  PT_Address: '24 Farragut Junction',
  Locker_ID: null,
  FD_Emp_ID: 105,
  Mangr_ID: 2 }
{ _id: ObjectId("638a702615d97d269fcc063b"),
  PT_ID: 302,
  PT_Name: 'Marten Mariette',
  PT_DOB: 1989-11-17T00:00:00.000Z,
  PT_PhoneNumber: '939-934-1709',
  PT_Email: 'mmariettem@buzzfeed.com',
  pt_salary: 25000.69921875,
  PT_Address: '28214 Dakota Junction',
  Locker_ID: 81,
  FD_Emp_ID: 105,
  Mangr_ID: 3 }
```

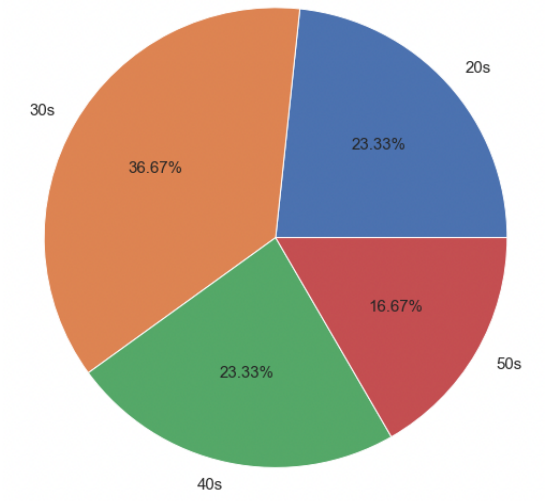

V. Database Access via Python

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 pd.read_sql_query to run the query and fetch all the data and store it in a dataframe. The visualizations and analytics are performed using various python libraries such as Matplotlib, Seaborn and Plotly.

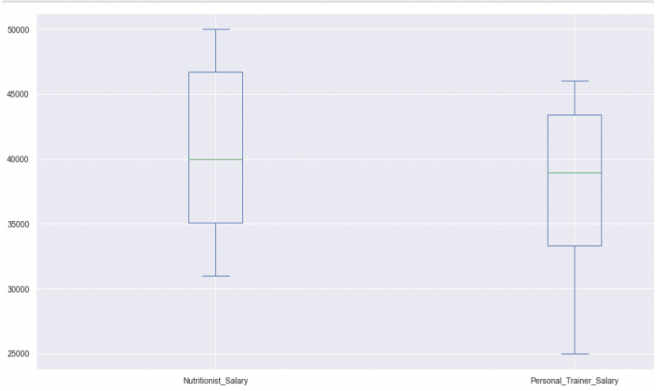
Bar Plot of Manager Name vs Manager Salary



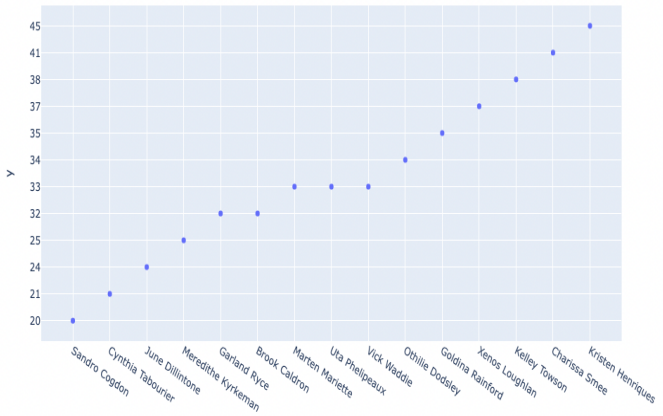
Pie Chart of Members Age Groups



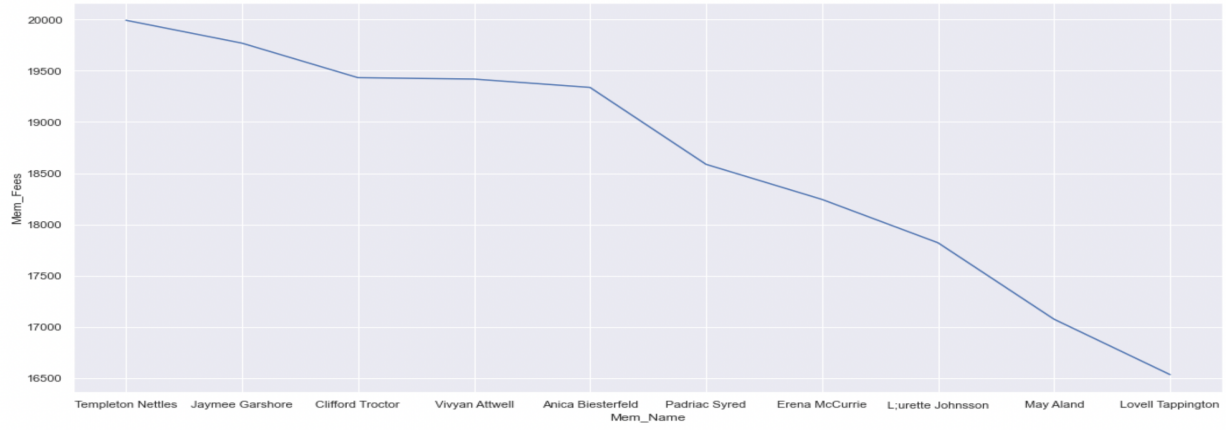
Boxplots of Nutritionist & Personal Trainer Salary



Scatterplot of Ages of Personal Trainers



Line Plot of Top 10 Highest Fees Paying Members



VI. Summary and Recommendation

Gym management system designed using various database models in this project can be implemented in real-world gyms. It will result in saving time and efforts for various tasks involved in the gym such as registration of members, tracking their billing details and diet plans, the equipment that the members need to use to achieve their fitness goals, maintaining payment details for various staff employed in the gym, assigning lockers to the members and staff, etc.

Also, by implementing this database management system in real-world gyms we have strong armed them with various features and advantages such as faster data retrieval, data backup and recovery, increased productivity, and reduced data inconsistencies. This database system provides great analytics capabilities, a small part of which is shown in this report utilizing Python.

The shortcomings of this project are that by utilizing a cloud-based database setup we could have focused more on MongoDB that would have helped the real-world gyms in gaining proper control of its whereabouts. Also, for key insights, the real-world gyms could shift more towards BI tools such as Tableau and Power BI.