Gym Buddy

Project Report

<u>Group 1</u>

Naga Sumanth Reddy Bareddy
Ojasvi Pravin Dere

857-693-9351 (Tel of Student 1)
857-313-5134 (Tel of Student 2)

bareddy.n@northeastern.edu
dere.o@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Naga Sumanth Reddy Bareddy

Signature of Student 2: Ojasvi Pravin Dere

Submission Date: 10th December 2023

USE CASE STUDY REPORT

Group No.: Group 01

Student Names: Ojasvi Pravin Dere and Naga Sumanth Reddy Bareddy

Executive Summary:

The goal of this project is to develop a relational database for efficient and effective functioning of gyms. As more individuals place a higher priority on living an active lifestyle, the fitness industry is currently undergoing a significant transition. However, important information like membership records, trainer sessions and equipment availability and booking are still handled by outdated manual methods in the gym management industry. This old method urgently needs to be modernized and made more efficient, especially considering the rising number of fitness facilities. By developing a comprehensive system that serves both gym administration and members, our project aims to address these issues. We seek to improve the efficacy and efficiency of gym administration by modeling gym operations from both sides, resulting in a seamless and fulfilling experience for everyone. Additionally, our platform will feature real-time information on equipment availability during specific time slots, and usage tracking and personalized workout routines tailored to individual members' goals and needs by professional trainers. These advanced functionalities will empower gyms to provide a superior level of service while simplifying the day-today management of their facilities.

The EER and UML diagrams were made on the basis of the database which was followed by mapping of the conceptual model to a relational model by using primary keys and foreign keys.The database was then further implemented in MySQL with tables for user, gym, booking, receptionist, payments, location, transactions, trainer and warranty info. The database was also implemented in NoSQL. Further connecting the database to Python visual analytics was obtained about the data which helped immensely in the project.

I. Introduction:

The Gym Buddy platform tackles a complex business issue that includes problems encountered by gym operators, customers, and the fitness sector. These issues are caused by the outdated and laborious methods used to manage gym data, which lead to operational inefficiencies, member unhappiness, and restrictions on resource optimization.

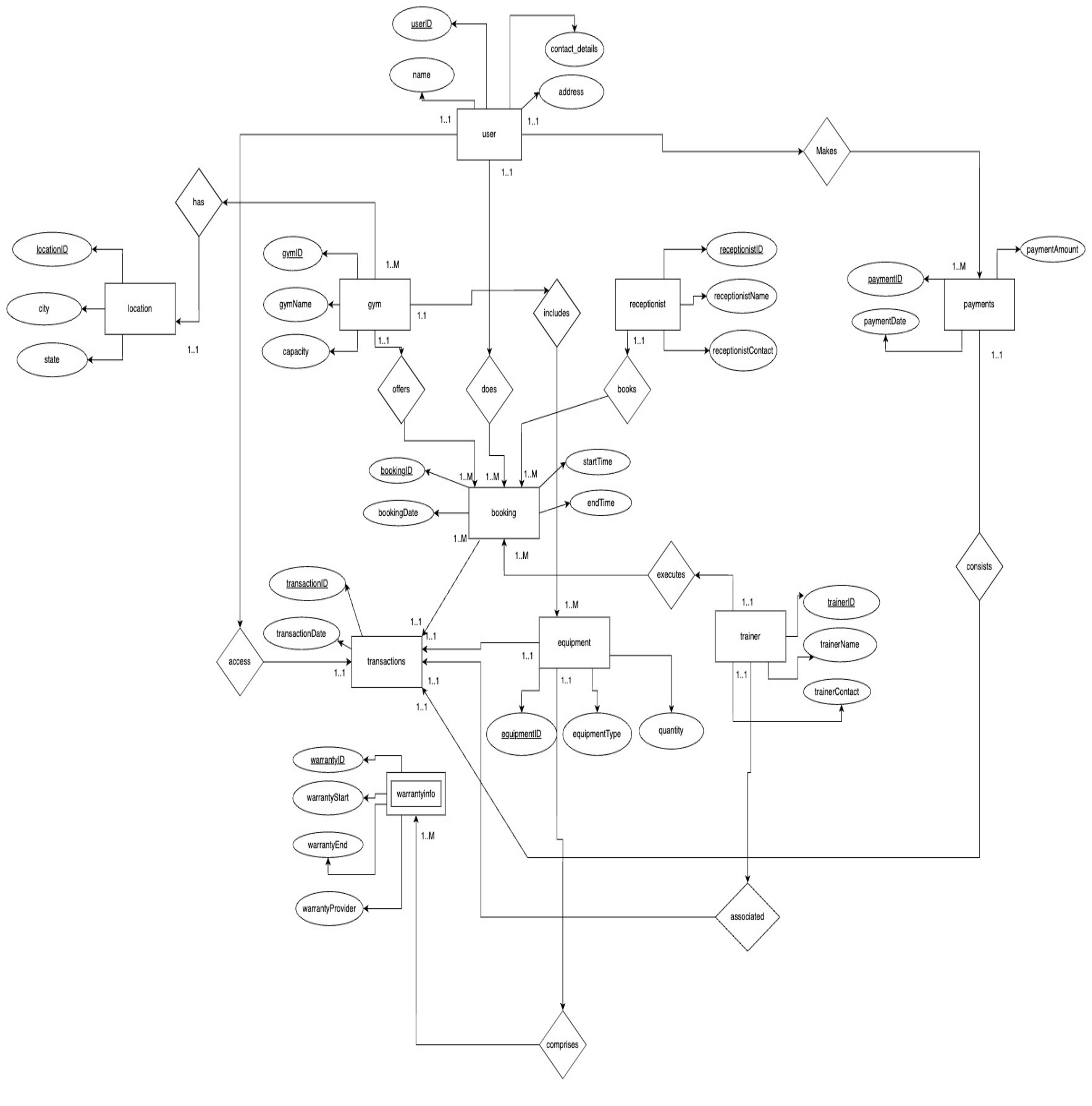
Problem Statement: To create a gym management that is user centric where users pay only according to the facilities utilized in the gym.

Goal: GymBuddy aims to enhance the overall fitness journey for individuals while ensuring optimal usability and security, with a focus on personalized service and efficient financial management.

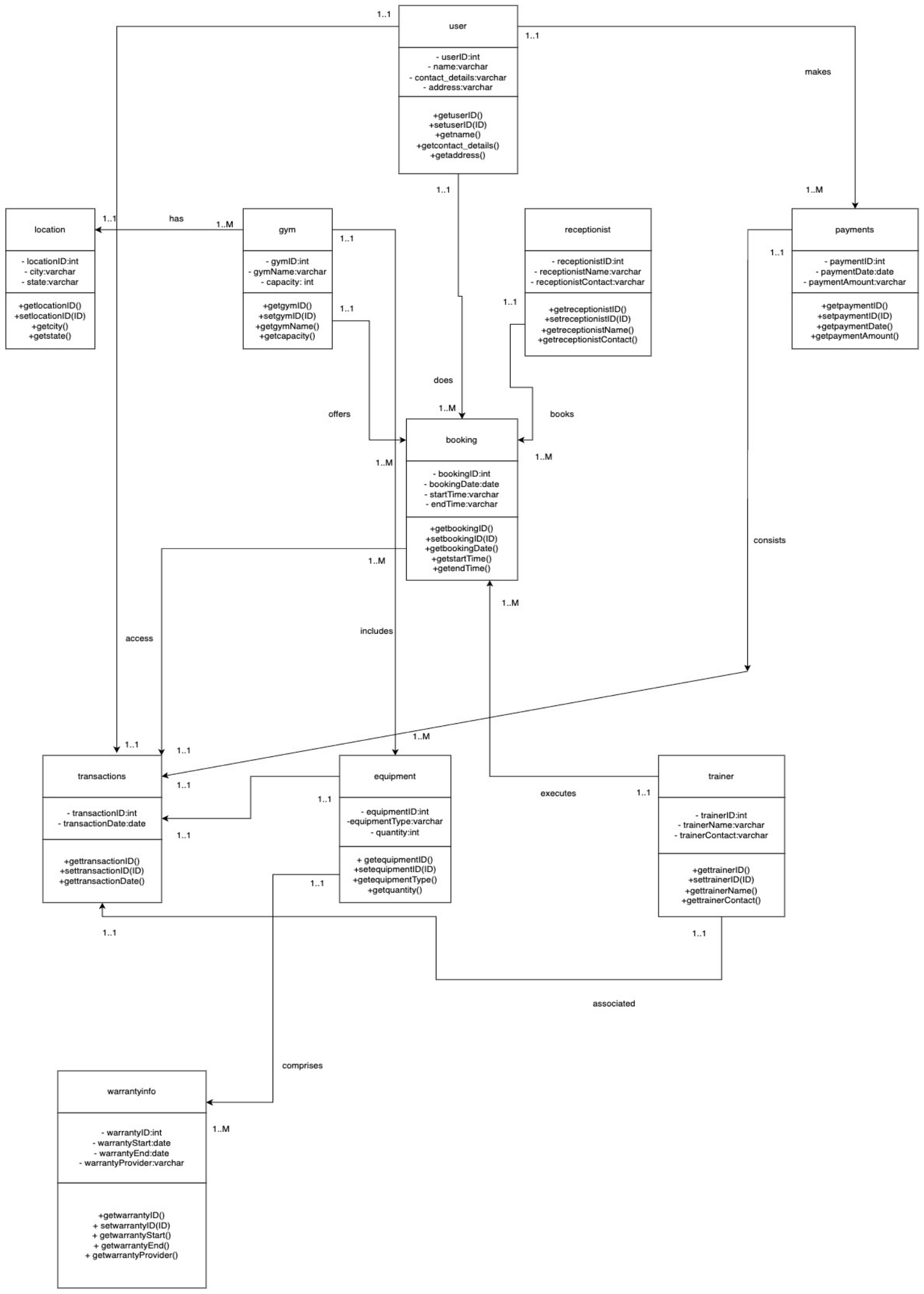
Requirements: User can do multiple bookings and is associated with multiple payments.
A gym can have multiple equipment and receptionists to handle the operations.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key: Underlined
Foreign Key: *Italicized*

user (userID, name, contact_details, address)

gym (gymID, gymName, capacity, *locationID*)
FOREIGN KEY locationID refers to locationID in location; NULL NOT ALLOWED

receptionist (receptionistID, receptionistName, receptionistContact)

location (locationID, city, state)

equipment (equipmentID, equipmentType, quantity, *gymID*)
FOREIGN KEY gymID refers to gymID in gym; NULL NOT ALLOWED

booking (bookingID , bookingDate, startTime, endTime, *userID, gymID, receptionistID, transactionID, trainerID*)
FOREIGN KEY userID refers to userID in user; NULL NOT ALLOWED
FOREIGN KEY gymID refers to gymID in gym; NULL NOT ALLOWED
FOREIGN KEY receptionistID refers to receptionistID in receptionist; NULL NOT ALLOWED
FOREIGN KEY transactionID refers to transactionID in transactions; NULL NOT ALLOWED
FOREIGN KEY trainerID refers to trainerID in trainer; NULL NOT ALLOWED

trainer (trainerID , trainerName, trainerContact)

payments (paymentID, paymentAmount, paymentDate, *userID*)
FOREIGN KEY userID refers to userID in user; NULL NOT ALLOWED

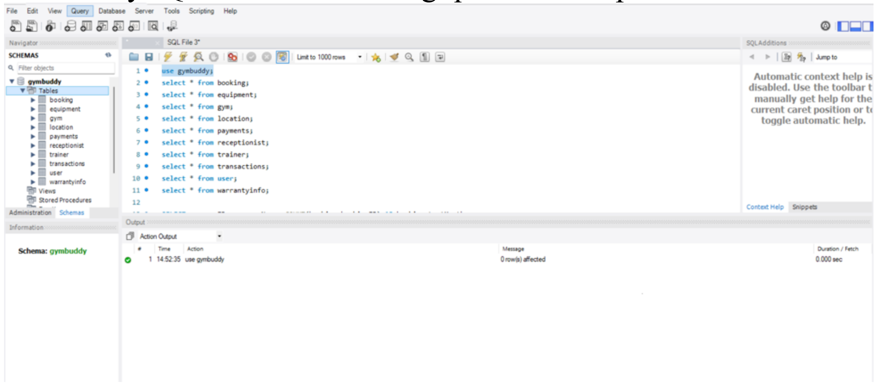
transactions (transactionID, paymentAmount, transactionDate, *userID, trainerID, equipmentID, paymentID*)
FOREIGN KEY userID refers to userID in user; NULL NOT ALLOWED
FOREIGN KEY trainerID refers to trainerID in trainer; NULL NOT ALLOWED
FOREIGN KEY equipmentID refers to equipmentID in equipment; NULL NOT ALLOWED
FOREIGN KEY paymentID refers to paymentID in payments; NULL NOT ALLOWED

warrantyinfo (warrantyID, warrantyStart, warrantyEnd, warrantyProvider, *equipmentID*)
FOREIGN KEY equipmentID refers to equipmentID in equipment; NULL NOT ALLOWED

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

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



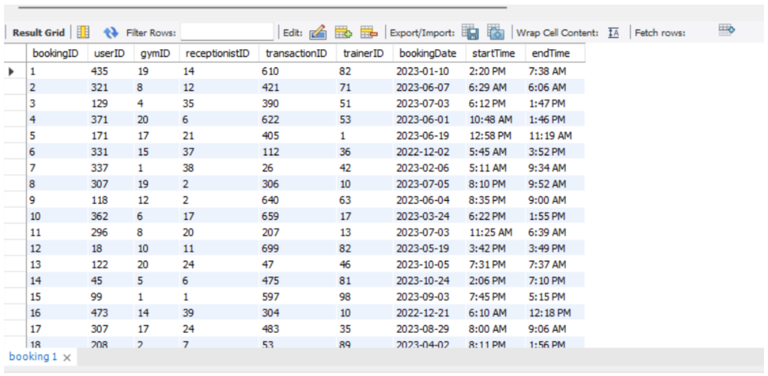
```
use gymbuddy;
select * from booking;
select * from equipment;
select * from gym;
select * from location;
select * from payments;
select * from receptionist;
select * from trainer;
select * from transactions;
select * from user;
select * from warrantyinfo;
```

Schema: gymbuddy

```
#  Time      Action         Message             Duration / Fetch
1  14:52:35  use gymbuddy   0 row(s) affected   0.000 sec
```

1. Simple Query: Retrieve the first 20 rows from the booking table

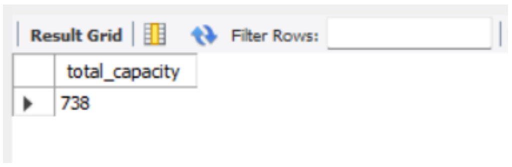
   SELECT *
   FROM booking
   LIMIT 20;



bookingID	userID	gymID	receptionistID	transactionID	trainerID	bookingDate	startTime	endTime
1	435	19	14	610	82	2023-01-10	2:20 PM	7:38 AM
2	321	8	12	421	71	2023-06-07	6:29 AM	6:06 AM
3	129	4	35	390	51	2023-07-03	6:12 PM	1:47 PM
4	371	20	6	622	53	2023-06-01	10:48 AM	1:46 PM
5	171	17	21	405	1	2023-06-19	12:58 PM	11:19 AM
6	331	15	37	112	36	2022-12-02	5:45 AM	3:52 PM
7	337	1	38	26	42	2023-02-06	5:11 AM	9:34 AM
8	307	19	2	306	10	2023-07-05	8:10 PM	9:52 AM
9	118	12	2	640	63	2023-06-04	8:35 PM	9:00 AM
10	362	6	17	659	17	2023-03-24	6:22 PM	1:55 PM
11	296	8	20	207	13	2023-07-03	11:25 AM	6:39 AM
12	18	10	11	699	82	2023-05-19	3:42 PM	3:49 PM
13	122	20	24	47	46	2023-10-05	7:31 PM	7:37 AM
14	45	5	6	475	81	2023-10-24	2:06 PM	7:10 PM
15	99	1	1	597	98	2023-09-03	7:45 PM	5:15 PM
16	473	14	39	304	10	2022-12-21	6:10 AM	12:18 PM
17	307	17	24	483	35	2023-08-29	8:00 AM	9:06 AM
18	208	2	7	53	89	2023-04-02	8:11 PM	1:56 PM

2. Aggregate: To calculate total capacity of all gyms

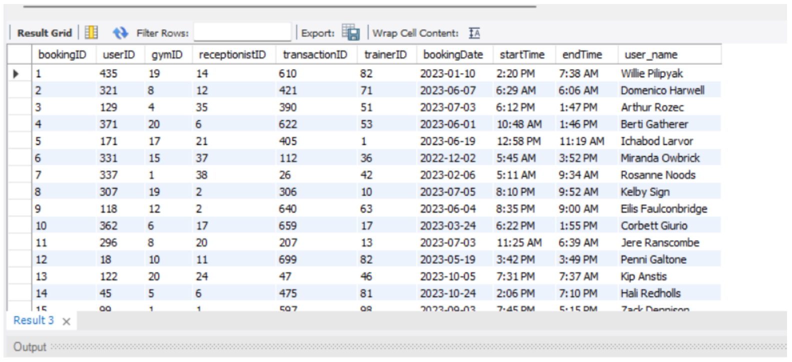
   SELECT SUM(capacity) AS total_capacity FROM gym;



total_capacity
738

3. Inner Join: Retrieve booking infor along with usernames for each booking

   SELECT booking.*, user.name AS user_name
   FROM booking
   INNER JOIN user ON booking.userID
   user.userID;

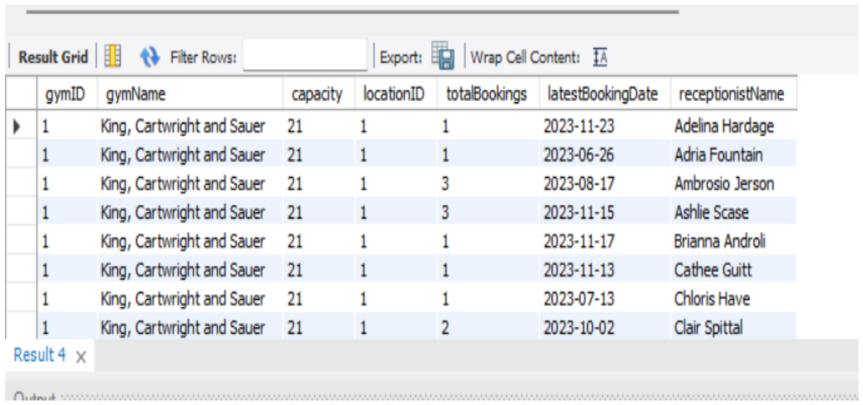


bookingID	userID	gymID	receptionistID	transactionID	trainerID	bookingDate	startTime	endTime	user_name
1	435	19	14	610	82	2023-01-10	2:20 PM	7:38 AM	Willie Pilipyak
2	321	8	12	421	71	2023-06-07	6:29 AM	6:06 AM	Domenico Harwell
3	129	4	35	390	51	2023-07-03	6:12 PM	1:47 PM	Arthur Rozec
4	371	20	6	622	53	2023-06-01	10:48 AM	1:46 PM	Berti Gatherer
5	171	17	21	405	1	2023-06-19	12:58 PM	11:19 AM	Ichabod Larvor
6	331	15	37	112	36	2022-12-02	5:45 AM	3:52 PM	Miranda Owbrick
7	337	1	38	26	42	2023-02-06	5:11 AM	9:34 AM	Rosanne Noods
8	307	19	2	306	10	2023-07-05	8:10 PM	9:52 AM	Kelby Sign
9	118	12	2	640	63	2023-06-04	8:35 PM	9:00 AM	Eilis Faulconbridge
10	362	6	17	659	17	2023-03-24	6:22 PM	1:55 PM	Corbett Giurio
11	296	8	20	207	13	2023-07-03	11:25 AM	6:39 AM	Jere Ranscombe
12	18	10	11	699	82	2023-05-19	3:42 PM	3:49 PM	Penni Galtone
13	122	20	24	47	46	2023-10-05	7:31 PM	7:37 AM	Kip Anstis
14	45	5	6	475	81	2023-10-24	2:06 PM	7:10 PM	Hali Redholls
15	99	1	1	597	98	2023-09-03	7:45 PM	5:15 PM	Zack Dennison

4. Outer: To analyze gym performance, including booking counts, latest dates, and receptionist details, covering all gyms

   SELECT gym.gymID, gym.gymName, gym.capacity, gym.locationID,

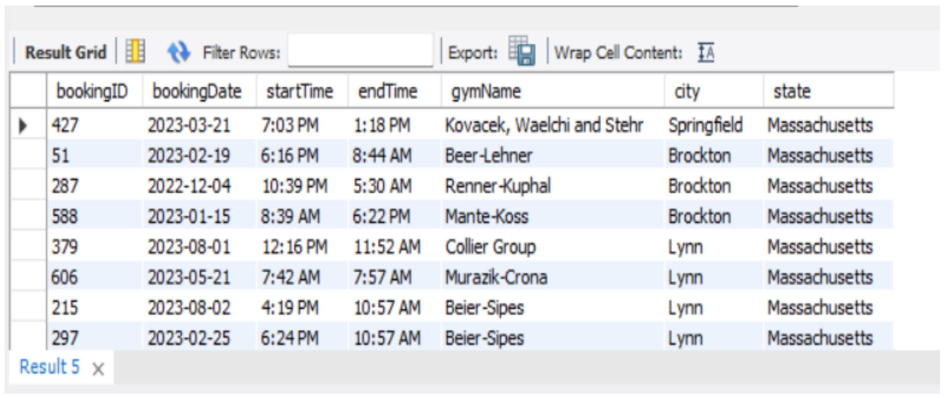
```
            COUNT(booking.bookingID) AS totalBookings,
            MAX(booking.bookingDate) AS latestBookingDate,
            Receptionist.receptionistName
  FROM gym
  LEFT JOIN booking ON gym.gymID =
  booking.gymID
  LEFT JOIN receptionist ON
  booking.receptionistID =
  receptionist.receptionistID
  GROUP BY gym.gymID, gym.gym.Name,
  gym.capacity, gym.locationID,
  receptionist.receptionistName
  ORDER BY gym.gymID;
```



gymID	gymName	capacity	locationID	totalBookings	latestBookingDate	receptionistName
1	King, Cartwright and Sauer	21	1	1	2023-11-23	Adelina Hardage
1	King, Cartwright and Sauer	21	1	1	2023-06-26	Adria Fountain
1	King, Cartwright and Sauer	21	1	3	2023-08-17	Ambrosio Jerson
1	King, Cartwright and Sauer	21	1	3	2023-11-15	Ashlie Scase
1	King, Cartwright and Sauer	21	1	1	2023-11-17	Brianna Androli
1	King, Cartwright and Sauer	21	1	1	2023-11-13	Cathee Guitt
1	King, Cartwright and Sauer	21	1	1	2023-07-13	Chloris Have
1	King, Cartwright and Sauer	21	1	2	2023-10-02	Clair Spittal

5. Nested Query: To retrieve details of bookings made by a specific trainer (Nari Kann) for gyms located in a MA

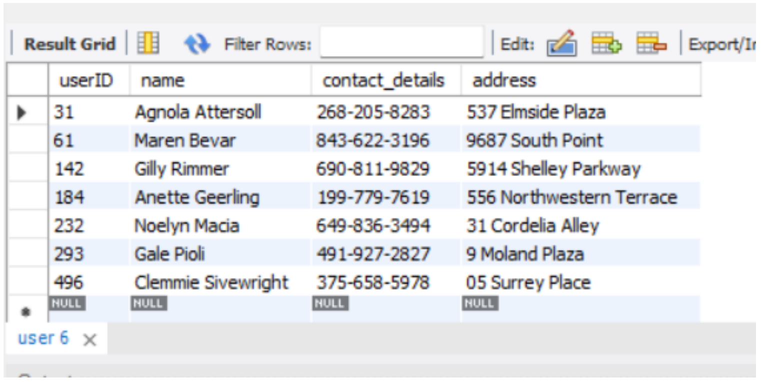
```
SELECT booking.bookingID,  booking.bookingDate, booking.startTime, booking.endTime,
gym-gymName, location.city,
  location.state
FROM booking
JOIN gym ON booking.gymID = gym-
 gymID
JOIN location ON gym.locationID =
  location.locationID
WHERE booking.trainerID = (SELECT
 trainerID FROM trainer WHERE
 trainerName = 'Nari Kann')
AND location.state = 'Massachusetts';
```



bookingID	bookingDate	startTime	endTime	gymName	city	state
427	2023-03-21	7:03 PM	1:18 PM	Kovacek, Waelchi and Stehr	Springfield	Massachusetts
51	2023-02-19	6:16 PM	8:44 AM	Beer-Lehner	Brockton	Massachusetts
287	2022-12-04	10:39 PM	5:30 AM	Renner-Kuphal	Brockton	Massachusetts
588	2023-01-15	8:39 AM	6:22 PM	Mante-Koss	Brockton	Massachusetts
379	2023-08-01	12:16 PM	11:52 AM	Collier Group	Lynn	Massachusetts
606	2023-05-21	7:42 AM	7:57 AM	Murazik-Crona	Lynn	Massachusetts
215	2023-08-02	4:19 PM	10:57 AM	Beier-Sipes	Lynn	Massachusetts
297	2023-02-25	6:24 PM	10:57 AM	Beier-Sipes	Lynn	Massachusetts

6. Correlated Query: To find all users who made transactions after a certain date

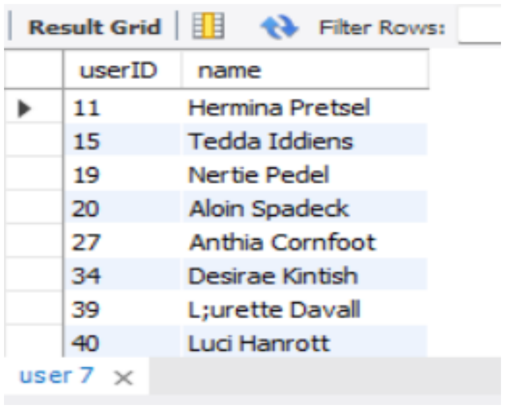
```
SELECT *
FROM user u
WHERE EXISTS (
SELECT 1
FROM transactions t
WHERE t.userID = u.userID
AND t. transactionDate > '2023-11-25'
);
```



userID	name	contact_details	address
31	Agnola Attersoll	268-205-8283	537 Elmside Plaza
61	Maren Bevar	843-622-3196	9687 South Point
142	Gilly Rimmer	690-811-9829	5914 Shelley Parkway
184	Anette Geerling	199-779-7619	556 Northwestern Terrace
232	Noelyn Macia	649-836-3494	31 Cordelia Alley
293	Gale Pioli	491-927-2827	9 Moland Plaza
496	Clemmie Sivewright	375-658-5978	05 Surrey Place
NULL	NULL	NULL	NULL

7. >=ALL/>ANY/EXISTS/Not Exists: To retrieve
Users who have made transactions with payment amounts greater than the maximum payment amount

```
SELECT u.userID, u.name
FROM user u
WHERE u.userID < > ALL (
            SELECT t1.userID
            FROM transactions t1
            WHERE t1.paymentAmount >= ANY (
            SELECT t2.paymentAmount
            FROM transactions t2
            WHERE t2.userID < > t1.userID
)
);
```



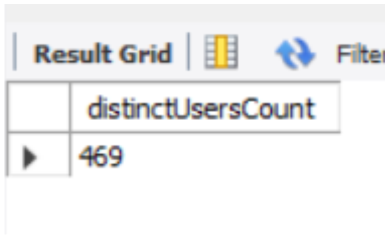
Result Grid | Filter Rows:

userID	name
11	Hermina Pretsel
15	Tedda Iddiens
19	Nertie Pedel
20	Aloin Spadeck
27	Anthia Cornfoot
34	Desirae Kintish
39	L;urette Davall
40	Luci Hanrott

user 7

8. Set Operations (Union): To retrieve count of distinct users who made bookings or transactions

```
SELECT COUNT (DISTINCT userID) AS distinctUsersCount
FROM  (
            SELECT userID FROM booking
            UNION
            SELECT userID FROM transactions
) AS combinedResults;
```

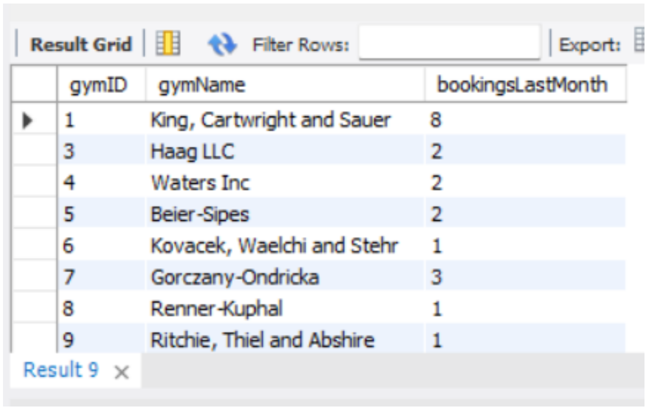


Result Grid

distinctUsersCount
469

9.Subqueries: To retrieve a list of gyms with the count of bookings made in the last month, including gyms with no bookings.

```
SELECT gym.gymID, gym.gymName, COUNT
(booking.bookingID) AS bookingsLastMonth
FROM gym
LEFT JOIN booking ON gym-gymID = booking.gymID
WHERE booking.bookingDate >=
DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
OR booking.bookingID IS NULL
GROUP BY gym.gymID, gym.gymName;
```



Result Grid | Filter Rows: | Export:

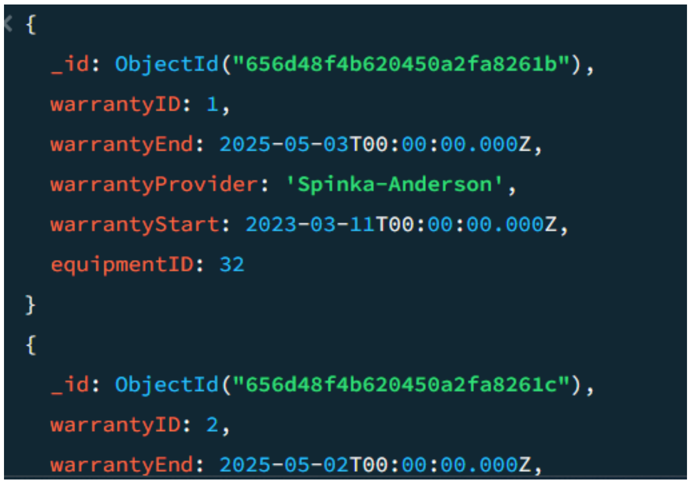
gymID	gymName	bookingsLastMonth
1	King, Cartwright and Sauer	8
3	Haag LLC	2
4	Waters Inc	2
5	Beier-Sipes	2
6	Kovacek, Waelchi and Stehr	1
7	Gorczany-Ondricka	3
8	Renner-Kuphal	1
9	Ritchie, Thiel and Abshire	1

Result 9

NoSQL Implementation:

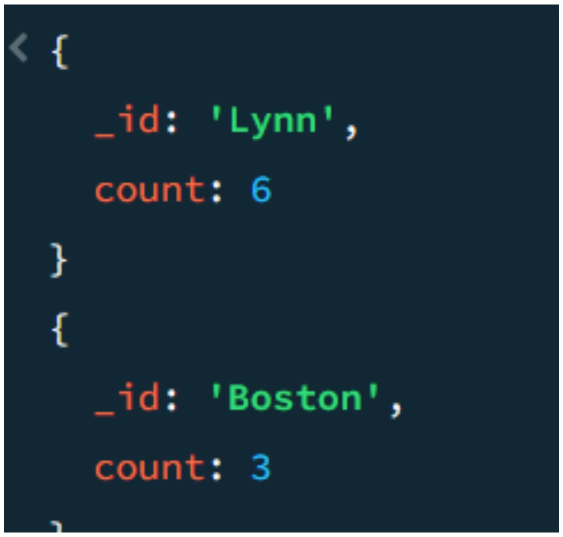
use gymbuddy
switched to db gymbuddy
gymbuddy>

1. › db.warrantyInfo. find ({
"warrantyEnd": {
$gt: new Date ("2024-12-31")
}
}) ;



```
{
  _id: ObjectId("656d48f4b620450a2fa8261b"),
  warrantyID: 1,
  warrantyEnd: 2025-05-03T00:00:00.000Z,
  warrantyProvider: 'Spinka-Anderson',
  warrantyStart: 2023-03-11T00:00:00.000Z,
  equipmentID: 32
}
{
  _id: ObjectId("656d48f4b620450a2fa8261c"),
  warrantyID: 2,
  warrantyEnd: 2025-05-02T00:00:00.000Z,
```

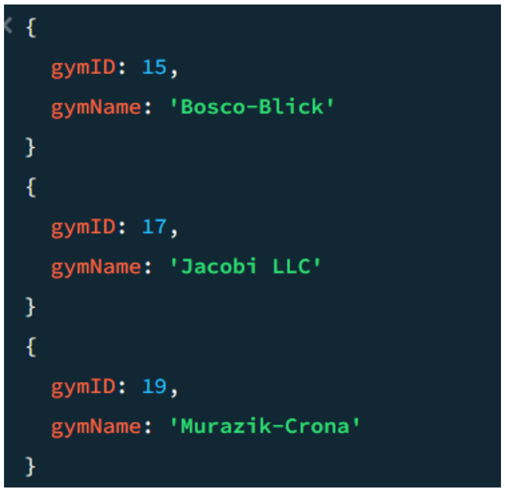
2. db-gym.aggregate ([
{
$lookup: {
from: "location",
localField: "locationID",
foreignField: "locationID",
as: "locationInfo"
} },
{
$unwind: "$locationInfo"
},
{$group: {
_id: "$locationInfo. city",
count: {$sum: 1}
} }
]) ;



```
{
  _id: 'Lynn',
  count: 6
}
{
  _id: 'Boston',
  count: 3
```

3. db.gym. aggregate ([
$lookup: {
from: "equipment",
localField: "gymID",
foreignField: "gymID",
as: "equipmentInfo"
} } ,
{
$unwind: "$equipnentInfo"},
{$match: {
"equipment Info equipmentType": {$in: ["Rowing", "Cycle"]}

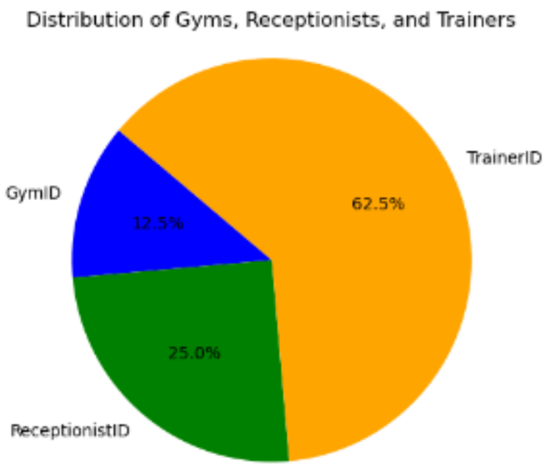
```
}
} ,
$group: {
_id: "$gymID",
gymName: ($first: "$gymName"},
equipmentTypes: {$addtoSet: "$equipmentInfo.equipmentType"}
$match: {equipmentTypes: {$all: ["Rowing", "Cycle"]}
}  } ,
{
$project: {
_id: 0, gymID: "$_id", gymName: {$ifNull: ["$gymName",
"Unknown"}
} } }
] )
```

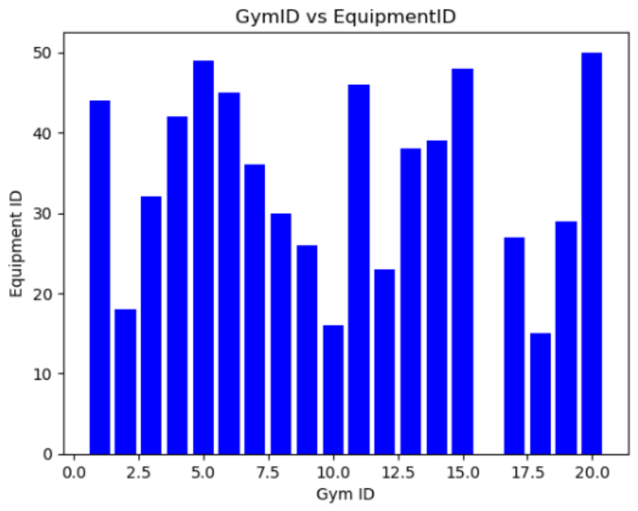


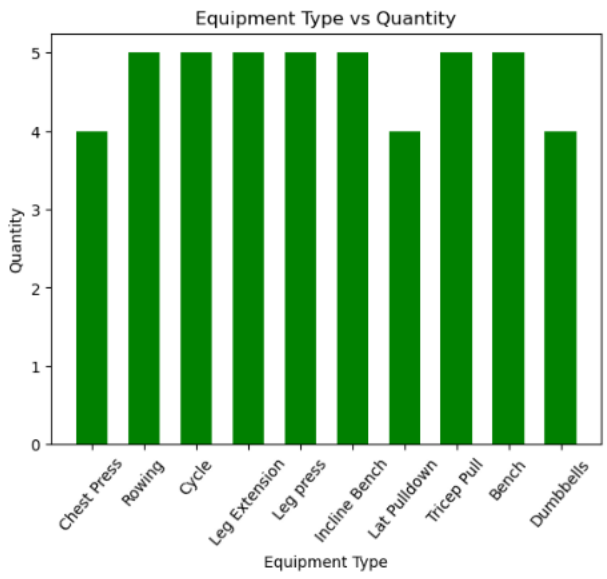
```
{
  gymID: 15,
  gymName: 'Bosco-Blick'
}
{
  gymID: 17,
  gymName: 'Jacobi LLC'
}
{
  gymID: 19,
  gymName: 'Murazik-Crona'
}
```

V. Database Access via Python

The database has been accessed using Python and the connection of MySQL to python is done using mysql.connector and cursor.execute to get the query, further using pandas and matplotlib to make visualizations.







VI. Summary and recommendation

The Gym Buddy platform tackles a complex business issue that includes problems encountered by gym operators, customers, and the fitness sector. These issues are caused by the outdated and laborious methods used to manage gym data, which lead to operational inefficiencies, member unhappiness, and restrictions on resource optimization. While users struggle to get real-time information on equipment availability and receive individualized fitness advice, gym owners deal with the laborious processing of membership records, attendance tracking, and customized training regimens. Fitness enthusiasts, gym management, and trainers would be the users of the platform. Using MySQL and NoSQL queries and python visualizations the project is created, and analytics are obtained by connecting it to database and creating tables for all entities.

Recommendations: Connecting data to Tableau and creating wireframes would give a clear idea about the database and visual part of the project.