

GP1: Project Definition & Scope

Introduction of Group: We are group of five members named Ta-Da, it is just reverse of Data. We will surely be able to broaden our understanding, refine the ideas we have, and produce excellent results with the support of your mentoring and direction.

Individual Introductions:

1. Maneesha Pusuluri: mpusulur@buffalo.edu



My name is Maneesha. I am an international student from India at the University at Buffalo. For the previous 2.5 years, I worked as a data analyst at Deloitte USI. I came here to advance my skills in the field of analyzing data and management. Apart from this, I also enjoy photography.

2. Sai Kiran Reddy Ponnnavolu: sponnavo@buffalo.edu



I am Sai Kiran Reddy Ponnnavolu, from India. I completed my Bachelors in 2022, and worked as a Civil Contractor for Govt of Andhra Pradesh. I'm here at UB to enhance my skills, to gain real-time experience. Beyond this I enjoy watching movies, riding motorcycles and trekking.

3. Saalem Mirza: saalemmi@buffalo.edu



I am Saalem Mirza, an international student from India at the University at Buffalo. I have worked as a Tableau developer in Accenture for 2 years. My major roles and responsibilities were creating executive dashboards and reports for Sales and Marketing teams on AWS Architecture.

4. Abhilash Kolapareddy: akolapar@buffalo.edu



Abhilash Kolapareddy, from India. I believe in the creation of products and brands that make our lives better, easier, and more fun. Over the Last 8 years I've worked at wonderful places such as Uber, WorkLLama, Supervek, Disside Design Studio, List Up, etc.. and co-founded edutech startups: My Preschool App, Bulbul Apps.

5. Vikram Srinivas Krishnamoorthy: vk48@buffalo.edu



My name is Vikram Srinivas. I am an international student from India at the University at Buffalo. For the previous 4 years, I worked as Senior Software Developer in Health care domain. I have choose UB for masters to enhance my skills in the field of product management.

Project description and scope:**Project:** Centralized Gym Management System**Description:**

A gym's or fitness center's operations and services are intended to be improved and streamlined through the Gym Management System initiative. This program is intended to make it easier to handle daily gym operations, member information, invoicing, scheduling, preferred location, administration, and workouts that one can perform at the gym to stay fit. For all people, exercising and staying fit are becoming increasingly vital and virtually a daily ritual. There are so many exercises to do, beginners or even professionals can wonder which exercise will target a specific muscle the best, and that is where this analysis can be useful. We also thought it would be amazing to visualize the details. This allows for enhanced management of member registration, attendance, and invoicing processes, an enhanced member experience through faster scheduling, class registration as per the specific location, and individualized services, and an increase in revenue through precise billing, increased member retention, and optimum resource allocation. Comprehensive data analysis and reporting for business expansion and strategic decision-making lead to the automation of repetitive administrative activities, enabling the staff and trainers to concentrate more on member engagement and satisfaction.

Scope:

The Centralized Gym Management System project is to completely transform the way gyms operate by giving customers a smooth and simple experience while equipping gym owners and staff with powerful management capabilities. It contributes significantly to the expansion and success of the gym business. It addresses the following aspects, all with the goal of establishing a smooth and effective gym operation:

- Member administration
 - Name
 - Gender
 - Age
 - Contact
 - Address
 - Type of Fitness
 - Session / Slots
 - Branch
 - Fitness Goals
 - BMI

GP2: Business Rules and Logic Diagram

Business rules:

1. Customers must provide their name, phone number, and email ID during the registration process.
2. The join date cannot be greater than the end date.
3. Join Date cannot be NULL. If the End Date is NULL, the member is a life-time member.
4. A Customer can attend one session at a time.
5. Gym members should be at least 18 years old.
6. Membership fees must be paid on time to maintain an active membership status.
7. A customer weight report must be taken every first day of the month.
8. If Payment Type = Cash then 1.5% discount on Amount.

Repository Data:

Entities:

CUSTOMER

MEMBERSHIP

PAYMENT

CUSTOMER REPORTS

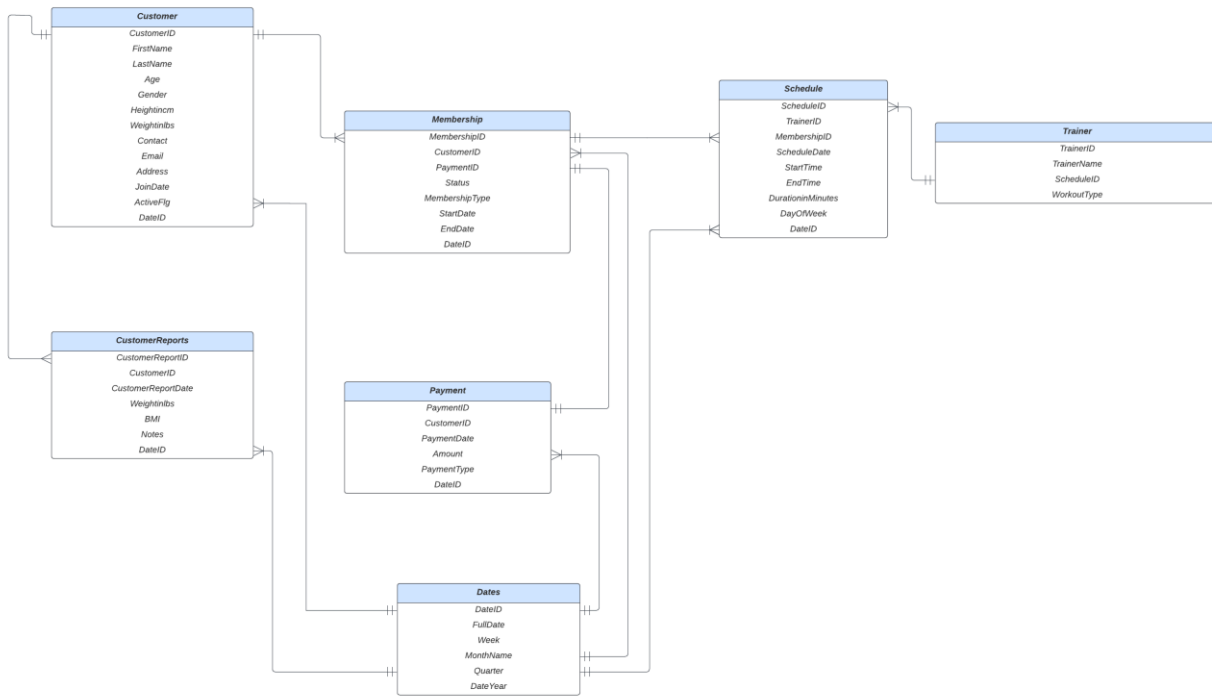
SCHEDULE

TRAINER

DATES

Data Name and Definition:

1. Customer: It includes the customer's information, like name, height, weight, and address
2. Payment: It depicts the customer's transactional activities
3. Membership: Tracks the registered customer's membership in the gym
4. Trainer: Details of the Trainer
5. Customer Reports: Refers to the individuals weight and BMI.
6. Schedule: Describes the length of time and time of the workout.
7. Dates: Gives the date-related information

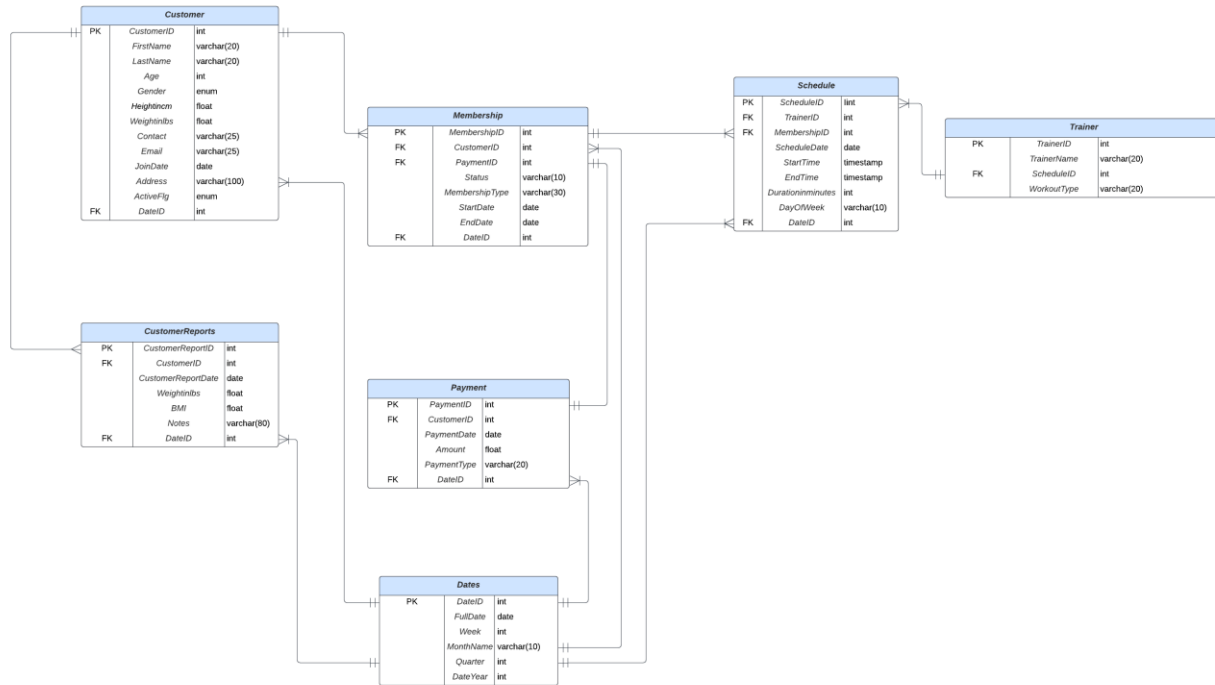
Entity Relationship Diagram:

Below are the primary keys and foreign keys for all relations.

Tables	Primary Keys	Foreign Keys
CUSTOMER	CustomerID	DateID
MEMBERSHIP	MembershipID	CustomerID, PaymentID, DateID
PAYMENT	PaymentID	CustomerID, DateID
CUSTOMER REPORTS	CustomerReportID	CustomerID, DateID
SCHEDULE	ScheduleID	TrainerID, MembershipID, DateID
TRAINER	TrainerID	ScheduleID
DATES	DateID	

GP3: Physical Diagram/DDL

Physical Diagram:



DDL:

1. Table: **Customer**

Creating enumerated datatypes in Gender and ActiveFlg columns to populate static values.

```
CREATE TYPE GenderType AS ENUM ('M', 'F', 'Prefer not to respond');
```

```
CREATE TYPE Flg AS ENUM ('Y', 'N');
```

```
CREATE TABLE Customer (
    CustomerID int PRIMARY KEY NOT NULL,
    DateID int REFERENCES Dates(DateID) NOT NULL,
    FirstName Varchar(20) NOT NULL,
    LastName Varchar(20) NOT NULL,
    Age int NOT NULL,
    Gender GenderType,
    Heightincm float,
    Weightinlbs float,
    Contact varchar(25),
    Email varchar(25),
    Address varchar(100),
```

```

JoinDate date,
ActiveFlg Flg NOT NULL
);

```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
customerid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
dateid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
firstname	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>
lastname	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>
age	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
gender	gendertype			<input type="checkbox"/>	<input type="checkbox"/>
heightincm	double precision			<input type="checkbox"/>	<input type="checkbox"/>
weightinlbs	double precision			<input type="checkbox"/>	<input type="checkbox"/>
contact	character varying	25		<input type="checkbox"/>	<input type="checkbox"/>
email	character varying	25		<input type="checkbox"/>	<input type="checkbox"/>
address	character varying	100		<input type="checkbox"/>	<input type="checkbox"/>
joindate	date			<input type="checkbox"/>	<input type="checkbox"/>
activeflg	flg			<input checked="" type="checkbox"/>	<input type="checkbox"/>

2. Table: Membership

```

CREATE TABLE Membership (
    MembershipID int PRIMARY KEY NOT NULL,
    CustomerID int REFERENCES Customer(CustomerID) NOT NULL,
    PaymentID int REFERENCES Payment(PaymentID) NOT NULL,
    DateID int REFERENCES Dates(DateID) NOT NULL,
    Status varchar(10),
    MembershipType varchar(30),
    StartDate date,
    EndDate date
);

```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
membershipid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
customerid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
paymentid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
dateid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
status	character varying	10		<input type="checkbox"/>	<input type="checkbox"/>
membershiptype	character varying	30		<input type="checkbox"/>	<input type="checkbox"/>
startdate	date			<input type="checkbox"/>	<input type="checkbox"/>
enddate	date			<input type="checkbox"/>	<input type="checkbox"/>

3. Table: Customer Reports

```
CREATE TABLE CustomerReports (
    CustomerReportID int PRIMARY KEY NOT NULL,
    CustomerID int REFERENCES Customer(CustomerID) NOT NULL,
    DateID int REFERENCES Dates(DateID) NOT NULL,
    CustomerReportDate date,
    Weightinlbs float,
    BMI float,
    Notes varchar(80)
);
```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
customerreportid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
customerid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
dateid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
customerreportdate	date			<input type="checkbox"/>	<input type="checkbox"/>
weightinlbs	double precision			<input type="checkbox"/>	<input type="checkbox"/>
bmi	double precision			<input type="checkbox"/>	<input type="checkbox"/>
notes	character varying	80		<input type="checkbox"/>	<input type="checkbox"/>

4. Table: **Payment**

```
CREATE TABLE Payment (
    PaymentID int PRIMARY KEY NOT NULL,
    CustomerID int REFERENCES Customer(CustomerID) NOT NULL,
    DateID int REFERENCES Dates(DateID) NOT NULL,
    PaymentDate date,
    Amount float,
    PaymentType varchar(20)
);
```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
paymentid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
customerid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
dateid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
paymentdate	date			<input type="checkbox"/>	<input type="checkbox"/>
amount	double precision			<input type="checkbox"/>	<input type="checkbox"/>
paymenttype	character varying	20		<input type="checkbox"/>	<input type="checkbox"/>

5. Table: **Dates**

```
CREATE TABLE Dates (
    DateID int PRIMARY KEY NOT NULL,
    FullDate date NOT NULL,
    Week int,
    MonthName varchar(10),
```

```

Quarter int,
DateYear int
);

```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
dateid	integer v			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
fulldate	date v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
week	integer v			<input type="checkbox"/>	<input type="checkbox"/>
monthname	character varying v	10		<input type="checkbox"/>	<input type="checkbox"/>
quarter	integer v			<input type="checkbox"/>	<input type="checkbox"/>
dateyear	integer v			<input type="checkbox"/>	<input type="checkbox"/>

6. Table: **Schedule**

```

CREATE TABLE Schedule (
    ScheduleID int PRIMARY KEY NOT NULL,
    TrainerID int REFERENCES Trainer(TrainerID) NOT NULL,
    MembershipID int REFERENCES Membership(MembershipID) NOT NULL,
    DateID int REFERENCES Dates(DateID) NOT NULL,
    ScheduleDate date,
    StartTime timestamp,
    EndTime timestamp,
    Durationinminutes int,
    DayOfWeek varchar(10)
);

```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
scheduleid	integer v			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
trainerid	integer v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
membershipid	integer v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
dateid	integer v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
scheduledate	date v			<input type="checkbox"/>	<input type="checkbox"/>
starttime	timestamp without time zone v			<input type="checkbox"/>	<input type="checkbox"/>
endtime	timestamp without time zone v			<input type="checkbox"/>	<input type="checkbox"/>
durationinminutes	integer v			<input type="checkbox"/>	<input type="checkbox"/>
dayofweek	character varying v	10		<input type="checkbox"/>	<input type="checkbox"/>

7. Table: **Trainer**

```
CREATE TABLE Trainer (  
    TrainerID int PRIMARY KEY NOT NULL,  
    ScheduleID int REFERENCES Schedule(ScheduleID) NOT NULL,  
    TrainerName varchar(20),  
    WorkoutType varchar(20)  
);
```

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
trainerid	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
scheduleid	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>
trainername	character varying	20		<input type="checkbox"/>	<input type="checkbox"/>
workouttype	character varying	20		<input type="checkbox"/>	<input type="checkbox"/>

GP4 - Queries & Output

All the data ingested in the database application (Postgres) is alias data which was randomly generated by ChatGPT. The data was downloaded in csv format and for creation of INSERT statements, we have used Formula bar in Excel using all the columns.

1. Assign BMI to the customer who has submitted their reports and assign it to different obesity categories.

Query Query History

```

1  SELECT
2      --Case statement to check the BMI range and assign category to it.
3      CASE
4      WHEN r.BMI <= 18 THEN 'Underweight'
5      WHEN r.BMI > 18 AND r.BMI <= 24 THEN 'Normal'
6      WHEN r.BMI > 24 AND r.BMI <=30 THEN 'Obese'
7      WHEN r.BMI > 30 THEN 'Extremely Obese'
8      END bmi_category
9      ,COUNT(DISTINCT c.CustomerID) AS no_of_members
10 FROM Customer c INNER JOIN CustomerReports r
11 ON c.CustomerID = r.CustomerID
12 GROUP BY bmi_category
13 ORDER BY no_of_members DESC;
14

```

Output:

Data Output		Messages	Notifications
	bmi_category text		no_of_members bigint
1	Normal		27
2	Obese		22
3	Extremely Obese		1

- To get information about average revenue and payment done in Cash or PayPal for Premium membership by Male and Female.

Query	Query History
1	SELECT
2	c.Gender
3	--Truncating all the below averages to 1 precision.
4	,TRUNC(AVG(c.Age)::NUMERIC,1) AS average_age
5	,TRUNC(AVG(p.Amount)::NUMERIC,1) AS average_revenue
6	,TRUNC(AVG(r.BMI)::NUMERIC,1) AS average_bmi
7	FROM Customer c
8	INNER JOIN CustomerReports r ON c.CustomerID = r.CustomerID
9	INNER JOIN Membership m ON c.CustomerID = m.CustomerID
10	INNER JOIN Payment p ON m.PaymentID = p.PaymentID
11	--Checking for Premium members with payment type in specific tenders.
12	WHERE MembershipType = 'Premium' AND PaymentType IN ('Cash','PayPal')
13	GROUP BY c.Gender;
14	

Output:

Data Output

Messages

Notifications

≡+

▼

▼

	<div>gender</div> <div>gendertype</div> <div></div>	<div>average_age</div> <div>numeric</div> <div></div>	<div>average_revenue</div> <div>numeric</div> <div></div>	<div>average_bmi</div> <div>numeric</div> <div></div>
1	M	32.0	198.7	23.0
2	F	27.1	198.2	24.0

3. Rank weekdays based on the revenue generated.

Query Query History

```

1  SELECT
2      /*Using Dense Rank function to assign consecutive ranks.
3       Ranks are assigned based on total revenue in descending order*/
4      DENSE_RANK() OVER (ORDER BY SUM(p.Amount) DESC) AS day_rank,
5      DayofWeek AS day_of_week,
6      SUM(Amount) AS revenue
7  FROM Schedule s LEFT JOIN Membership m
8  ON s.MembershipID = m.MembershipID
9  INNER JOIN Payment p
10 ON m.PaymentID = p.PaymentID
11 GROUP BY day_of_week;
12

```

Output:

Data Output

Messages

Notifications

≡ +

📄 ▼

📋 ▼

🗑️

🗄️

⬇️

📈

	<div>day_rank</div> <div>bigint</div> <div>🔒</div>	<div>day_of_week</div> <div>character varying (10)</div> <div>🔒</div>	<div>revenue</div> <div>double precision</div> <div>🔒</div>
1	1	Sunday	2091.3
2	2	Tuesday	2046.6
3	3	Monday	712
4	3	Wednesday	712
5	4	Thursday	667
6	5	Friday	664.3

4. Fetch five different ranks for customers as per their weight from the report summary, even though customers have similar weights.

Query Query History

```

1  --Creating a temporary table for assigning ranks
2  WITH Customer_ranking AS (
3      SELECT
4          CRS.customerid
5          ,CRS.customerreportid
6          ,CRS.weightinlbs
7          --Ranking customers as per their weights.
8          ,ROW_NUMBER() OVER (ORDER BY CRS.weightinlbs DESC) as Rank
9          FROM Customerreports AS CRS
10 )
11 --Selecting fields from the above temporary table
12 SELECT
13     CR.customerid as Customer_ID
14     ,CR.customerreportid as Customer_Reportid
15     ,CONCAT(CONCAT (c. FirstName,' '), c. LastName) as Customer_Name
16     ,CR.weightinlbs as Weightin_lbs
17     ,Rank
18 FROM Customer_ranking CR
19 JOIN CUSTOMER C
20 ON CR.CUSTOMERID=C.CUSTOMERID
21 WHERE Rank <= 5
22 ORDER BY Rank;
23

```

Output:

Data Output

Messages

Notifications

	customer_id integer	customer_reportid integer	customer_name text	weightin_lbs double precision	rank bigint
1	9	200009	William Anderson	220	1
2	3	200003	Michael Williams	210	2
3	5	200005	David Martinez	200	3
4	11	200442	Benjamin Jackson	190	4
5	45	200045	Benjamin Hayes	190	5

5. Checking number of customers assigned to a trainer and their workout type.

Query Query History

```

1  SELECT
2      --Using COALESCE statements for adding text to non-existing trainers
3      COALESCE(CAST(t.TrainerID AS VARCHAR),'No Trainer Assigned') AS trainer
4      ,COALESCE(t.workouttype,'No workout plan for now') AS workout_type
5      ,COUNT(DISTINCT c.CustomerID) AS no_of_customers
6  FROM
7  Trainer t RIGHT JOIN Schedule s ON t.ScheduleID = s.ScheduleID
8  LEFT JOIN Membership m ON s.MembershipID = m.MembershipID
9  LEFT JOIN Customer c ON c.CustomerID = m.CustomerID
10 GROUP BY t.TrainerID, workout_type
11 ORDER BY no_of_customers DESC;
12

```

Output:

Data Output Messages Notifications

	trainer character varying	workout_type character varying	no_of_customers bigint
1	No Trainer Assigned	No workout plan for now	38
2	600002	Strength Training	1
3	600003	Yoga	1
4	600004	HIIT	1
5	600006	CrossFit	1
6	600007	Functional Training	1
7	600001	Cardio	1
8	600009	Spin	1
9	600010	Kickboxing	1
10	600012	Dance Fitness	1
11	600013	Stretching	1
12	600076	Pilates	1
13	600008	Zumba	1

6. Comparing the current year revenue against the previous year revenue and computing its growth.

```

Query  Query History
1  SELECT
2      DISTINCT d.week,
3      SUM(p.Amount) AS revenue,
4      /*Using LAG function to get the previous week and its revenue
5      Assigning 0 to previous week columns of first week using COALESCE function*/
6      COALESCE(LAG (d.week,1) OVER (ORDER BY d.week),0) AS previous_week,
7      COALESCE(LAG (SUM(p.Amount),1) OVER (),0) AS previous_week_revenue,
8      /* Change in percentage = (Current Week - Previous Week)/Previous Week
9      Truncating the Double Precision output to 1 precision*/
10     COALESCE(
11         TRUNC(((SUM(p.Amount) - LAG (SUM(p.Amount),1) OVER ())/LAG (SUM(p.Amount),1) OVER ())*100)::NUMERIC,1),0)
12         AS growth_in_percentage
13 FROM dates d
14 INNER JOIN Payment p ON d.DateID = p.DateID
15 GROUP BY d.week
16 ORDER BY d.week;

```

Output:

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	week integer	revenue double precision	previous_week integer	previous_week_revenue double precision	growth_in_percentage numeric
1	6	667	0	0	0
2	10	713.4	6	667	6.9
3	11	667	10	713.4	-6.5
4	19	712	11	667	6.7
5	26	664.3	19	712	-6.6
6	29	664.3	26	664.3	0.0
7	33	712	29	664.3	7.1
8	43	709.3	33	712	-0.3
9	45	668.9	43	709.3	-5.6
10	48	715	45	668.9	6.8

7. Mode of payment displaying total amount year-wise

Query Query History

```

1  SELECT
2  --Extracting the year from the paymentdate
3      Extract (YEAR FROM paymentdate) as Year
4      ,paymenttype as Mode_of_Payment
5      ,SUM (amount) as Total_Amount
6  FROM payment
7  GROUP BY Mode_of_Payment, Year
8  ORDER BY Year asc;
9

```

Output:

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

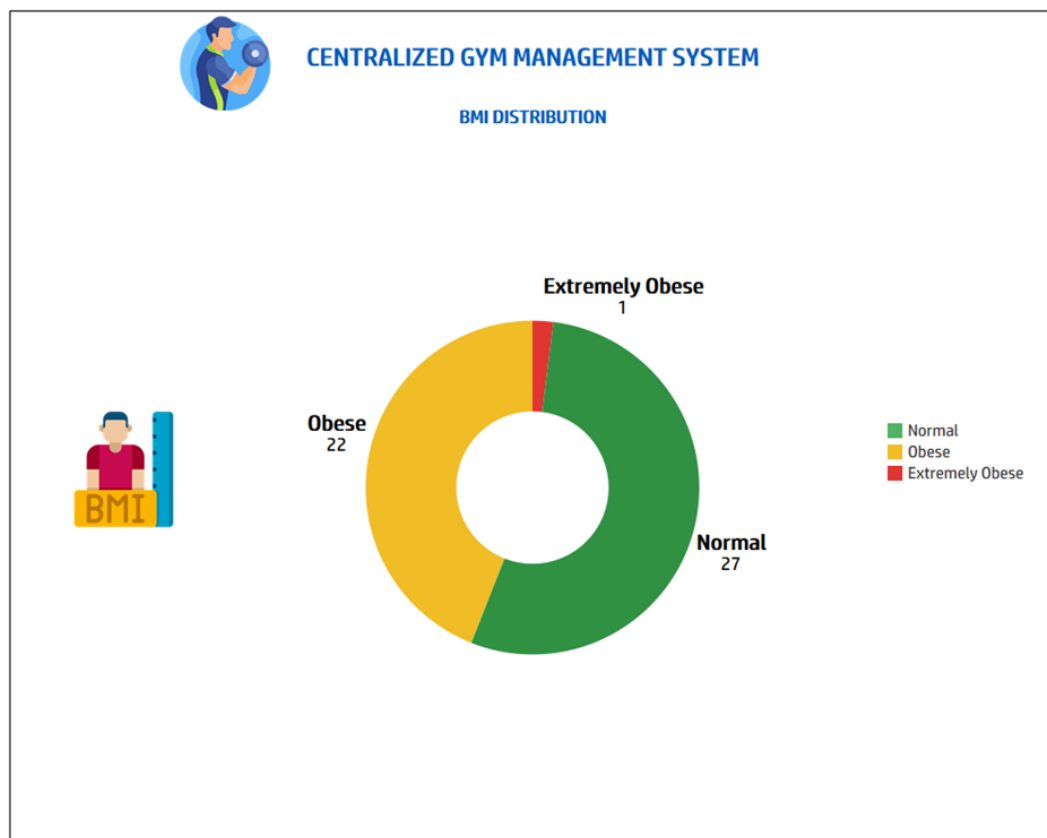
📈

	<div>year</div> <div>numeric</div> <div>🔒</div>	<div>mode_of_payment</div> <div>character varying (20)</div> <div>🔒</div>	<div>total_amount</div> <div>double precision</div> <div>🔒</div>
1	2016	Cash	315.4
2	2016	PayPal	595
3	2016	Credit Card	515
4	2017	Credit Card	320
5	2017	PayPal	515
6	2017	Cash	586.3
7	2018	PayPal	670
8	2018	Credit Card	790
9	2018	Cash	586.3
10	2019	PayPal	475
11	2019	Credit Card	865
12	2019	Cash	660.2

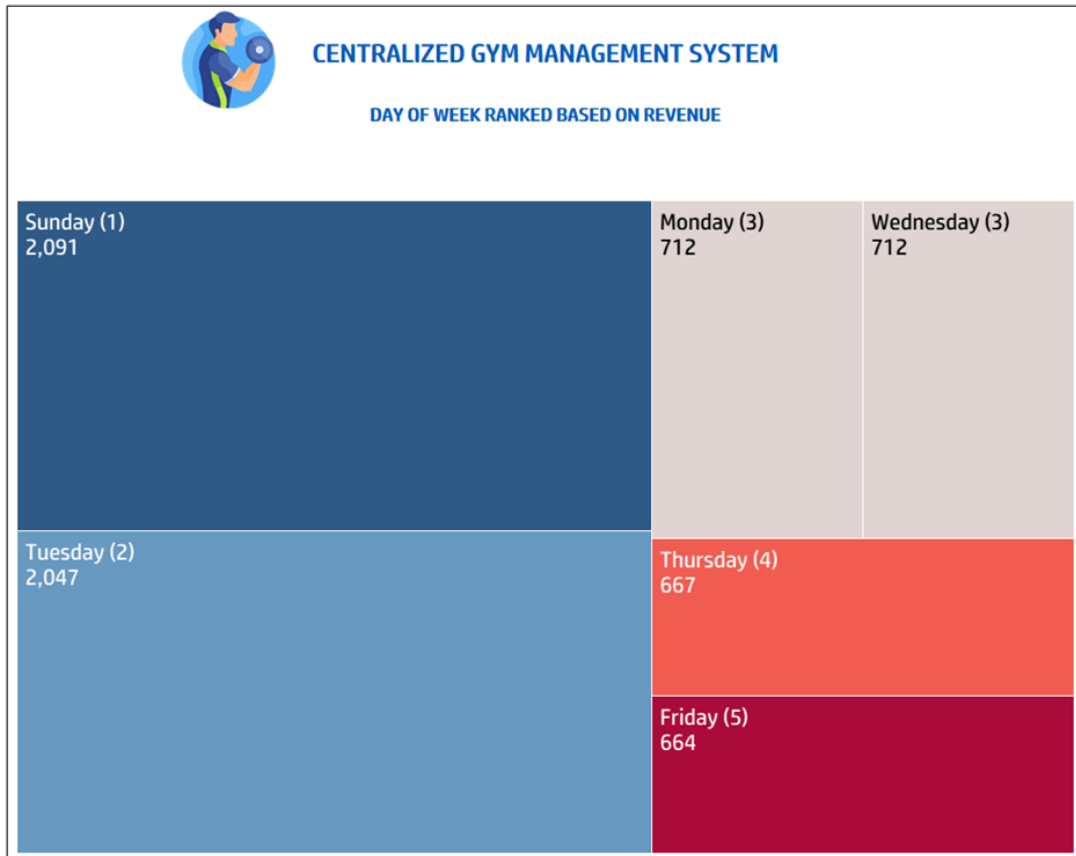
GP5 – Visualizations

In order to create visually appealing and comprehensive dashboards, Tableau Desktop was used to create below dashboards as Tableau seamlessly integrates with Postgres database.

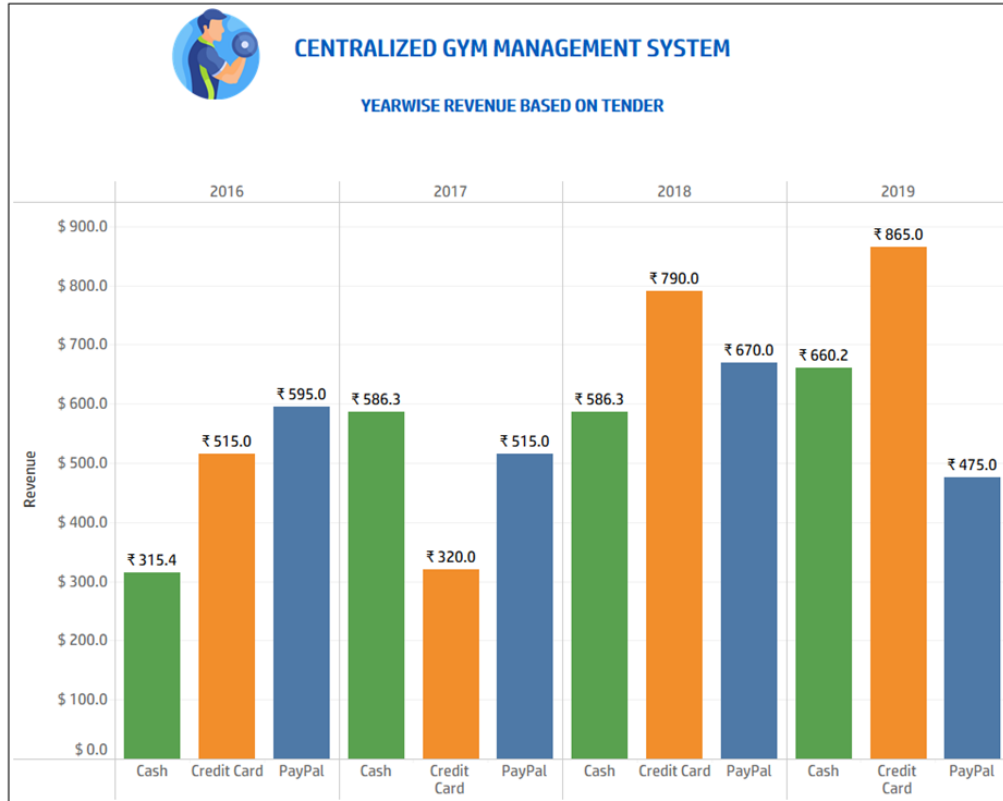
1. BMI (Body Mass Index) is a widely used metric to assess whether an individual's weight falls within a healthy range or if they are classified as normal, obese, or extremely obese. This dashboard aims to visually represent the distribution of individuals across these categories and offer valuable insights for gym management to create tailored workout programs or procure specific equipment for its members.



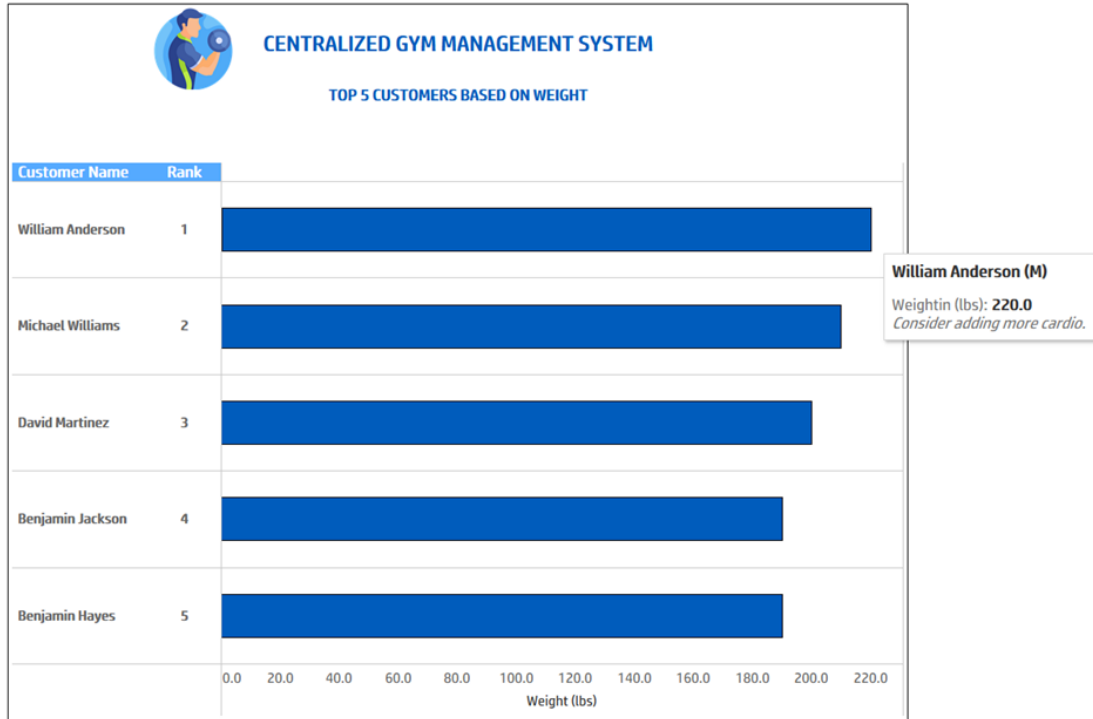
2. This Heat Map provides an immediate overview of high and low revenue days to optimize operations, marketing strategies, and staffing based on revenue fluctuations throughout the week.




3. The bar chart illustrates the gym's total revenue for each year, offering a clear view of revenue growth or fluctuations over time. This chart provides a foundation for understanding the financial performance of the gym. It also assists the gym management in understanding payment trends, optimizing payment processing, and making informed financial decisions.



4. Along with identifying and acknowledging the achievements of customers who have made significant progress in their fitness journeys, it is important as well to focus on customers who are struggling with their fitness journey. Below dashboard aims to focus on this pain point by ranking customers by weight and extracting insights from their reports for trainer's or instructor's reference.




5. Availability of trainer is the most important aspect of any gym and ensuring minimal idle time for effective utilization of trainers. So, the management should be able to track how many customers are assigned to each trainer, efficiently managing client assignments, monitor workload distribution, and optimize workouts (4 pictures has been attached)

 CENTRALIZED GYM MANAGEMENT SYSTEM NUMBER OF TRAINERS ASSIGNED TO CUSTOMERS	
Trainers	No of Customers
No Trainer Assigned	38
Assigned Trainers	12

(Trainers grouped together to give overview of assigned and unassigned trainers, then expandable Group field created in Tableau to show workout type breakdown).

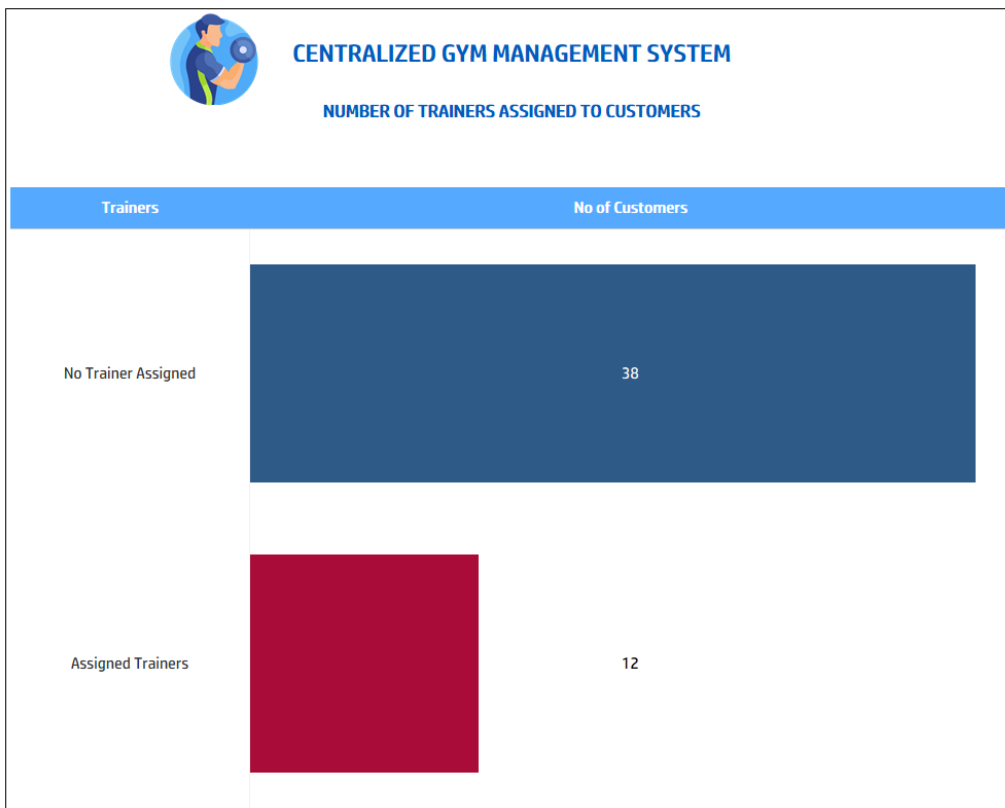
 CENTRALIZED GYM MANAGEMENT SYSTEM NUMBER OF TRAINERS ASSIGNED TO CUSTOMERS		
Trainers	Workout Type	No of Customers
No Trainer Assigned	No workout plan for now	38
Assigned Trainers	Cardio	1
	CrossFit	1
	Dance Fitness	1
	Functional Training	1
	HIIT	1
	Kickboxing	1
	Pilates	1
	Spin	1
	Strength Training	1
	Stretching	1
	Yoga	1
	Zumba	1



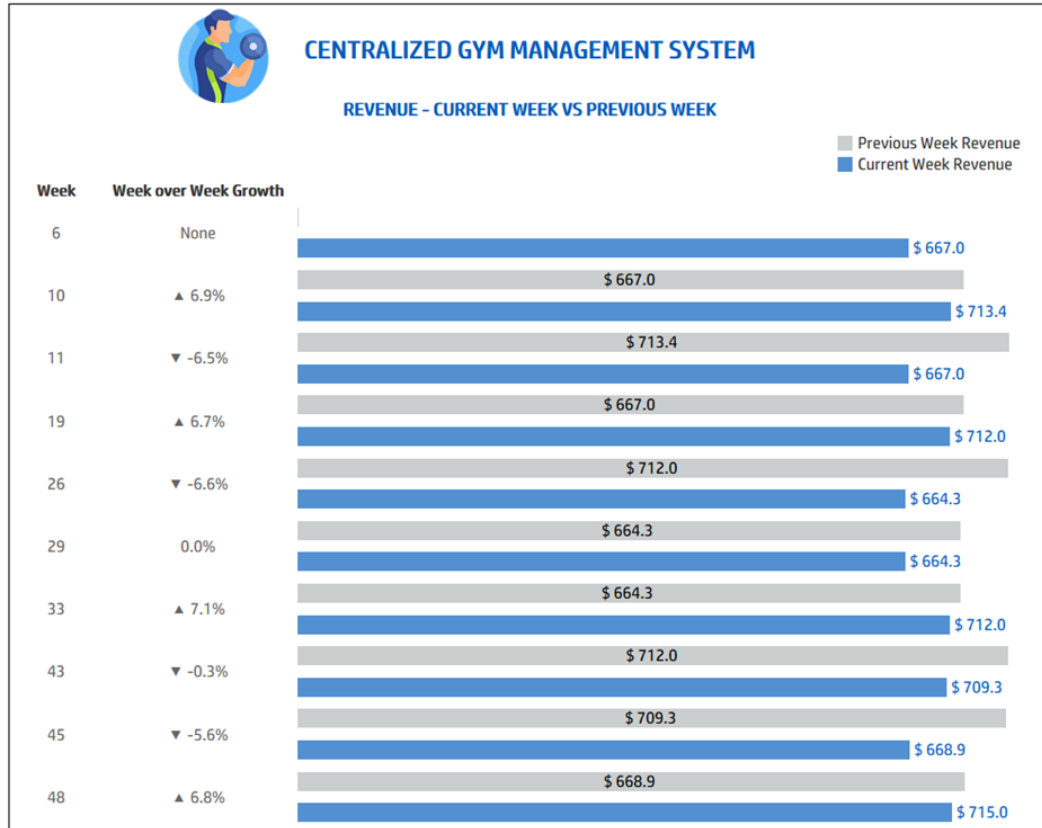
CENTRALIZED GYM MANAGEMENT SYSTEM

NUMBER OF TRAINERS ASSIGNED TO CUSTOMERS




Trainers	Workout Type	No of Customers
No Trainer Assigned	No workout plan for now	38
Assigned Trainers	Cardio	1
	CrossFit	1
	Dance Fitness	1
	Functional Training	1
	HIIT	1
	Kickboxing	1
	Pilates	1
	Spin	1
	Strength Training	1
	Stretching	1
	Yoga	1
	Zumba	1



6. Following dashboard provides gym management and stakeholders with a clear overview of the financial performance, highlighting trends and fluctuations week by week providing comparison between current week and previous week along with tickers.



7. By focusing on average age, average BMI, and average revenue, this dashboard enables gym management and stakeholders to better understand their clientele, identify trends, and make informed decisions by gender.

<div>CENTRALIZED GYM MANAGEMENT SYSTEM</div> <div>AVERAGE VALUES BASED ON GENDER</div>			
Gender	Average Age	Average BMI	Average Revenue
	32.0	23.0	198.7
	27.1	24.0	198.2