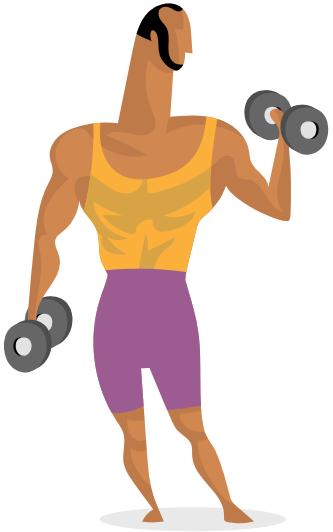
PL/SQL PROJECT

# GYM PROGRESS TRACKER DATABASE

## ORACLE APEX

JOASH DALIGCON
LANCE MIRANO

# Overview



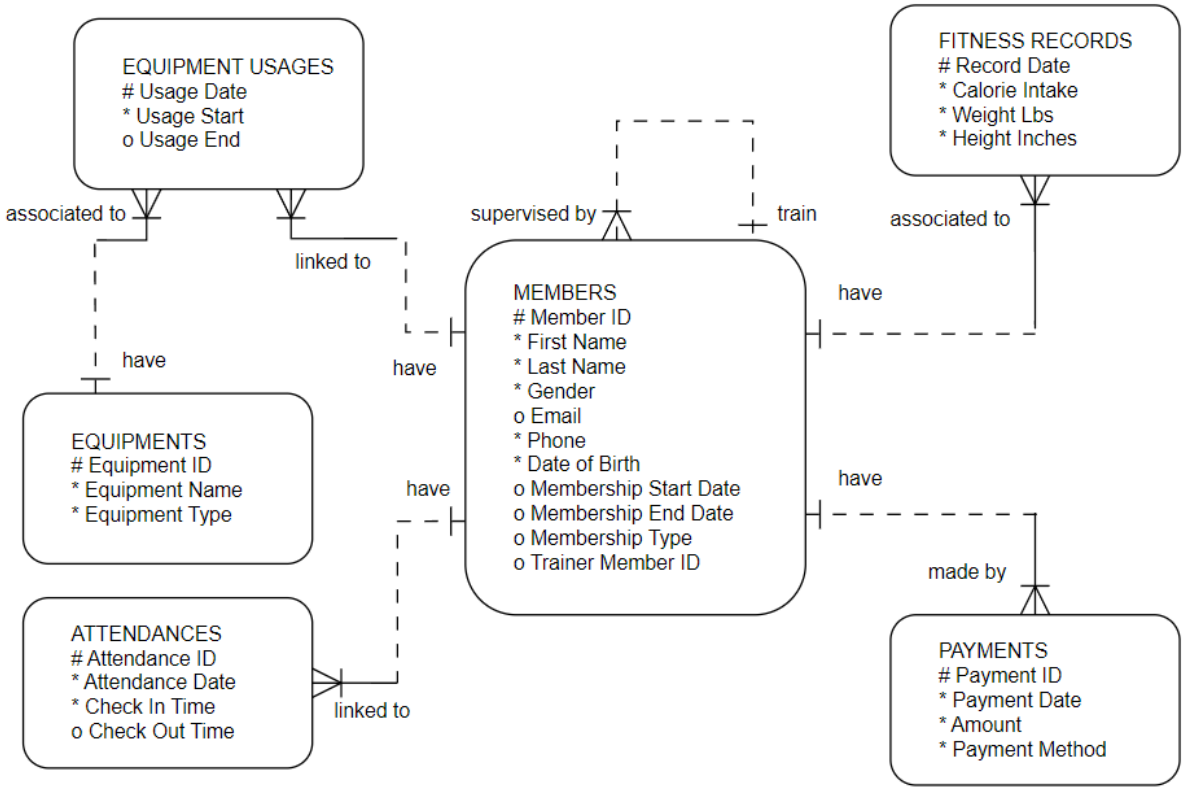
This database manages aspects of gym operations such as member registration, attendance tracking, and payment transactions. It also monitors members' progress through attributes such as calorie intake, height and weight for effective fitness management. Furthermore, it keep tracks of trainer assignments and equipment usage, that can aid in determining training insights and optimizing the overall gym experience.

# 1

# DATABASE DESIGN

xx

# Project ERD


EQUIPMENT USAGES
# Usage Date
* Usage Start
o Usage End
associated to
linked to
have
EQUIPMENTS
# Equipment ID
* Equipment Name
* Equipment Type
ATTENDANCES
# Attendance ID
* Attendance Date
* Check In Time
o Check Out Time
linked to
supervised by
train
have
have
have
MEMBERS
# Member ID
* First Name
* Last Name
* Gender
o Email
* Phone
* Date of Birth
o Membership Start Date
o Membership End Date
o Membership Type
o Trainer Member ID
FITNESS RECORDS
# Record Date
* Calorie Intake
* Weight Lbs
* Height Inches
associated to
have
made by
PAYMENTS
# Payment ID
* Payment Date
* Amount
* Payment Method

# Business Rules

1. Each member may be supervised by a member (trainer)
2. Each member (trainer) may train one or many members
3. Each attendance record must be linked to one and only one member
4. Each member may have one or many attendance records
5. Each equipment usage must be linked to one and only one member
6. Each member may have one or many equipment usages
7. Each equipment usage must be associated to one and only one equipment
8. Each equipment may have one or many equipment usages
9. Each fitness record must be associated to one and only one member
10. Each member may have one or many fitness records
11. Each payment record must be made by one and only one member
12. Each member may have one or many payment records

# Creating the Tables

```sql
-- Create the Members table
CREATE TABLE Members (
    MemberID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    FirstName VARCHAR2(50)  NOT NULL,
    LastName VARCHAR2(50)   NOT NULL,
    Gender CHAR(1)          NOT NULL,
    Email VARCHAR2(50),
    Phone VARCHAR2(20)      NOT NULL,
    DateOfBirth DATE        NOT NULL,
    MembershipStart DATE,
    MembershipEnd DATE,
    MembershipType VARCHAR2(10),
    TrainerMbrID NUMBER
);
-- Create the Payments table
CREATE TABLE Payments (
    PaymentID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    PaymentDate DATE            NOT NULL,
    Amount NUMBER(10, 2)        NOT NULL,
    PaymentMethod VARCHAR2(50)  NOT NULL,
    MemberID NUMBER             NOT NULL
);
```

```sql
-- Create the Attendances table
CREATE TABLE Attendances (
    AttendanceID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    AttendanceDate DATE     NOT NULL,
    CheckInTime TIMESTAMP   NOT NULL,
    CheckOutTime TIMESTAMP,
    MemberID NUMBER         NOT NULL
);
-- Create the Equipments table
CREATE TABLE Equipments (
    EquipmentID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    EquipmentName VARCHAR2(50) NOT NULL,
    EquipmentType VARCHAR2(50) NOT NULL
);
-- Create the EquipmentUsages table
CREATE TABLE EquipmentUsages (
    UsageDate DATE          NOT NULL,
    MemberID NUMBER         NOT NULL,
    EquipmentID NUMBER      NOT NULL,
    UsageStart TIMESTAMP    NOT NULL,
    UsageEnd TIMESTAMP,
    -- Member can use each equipment once per day:
    --   MemberID + UsageDate + EquipmentID must be unique
    PRIMARY KEY (UsageDate, MemberID, EquipmentID)
);
```

```sql
-- Create the FitnessRecords table
CREATE TABLE FitnessRecords (
    RecordDate DATE             NOT NULL,
    MemberID NUMBER             NOT NULL,
    CalorieIntake NUMBER(10, 2) NOT NULL,
    WeightLbs NUMBER(10, 2)     NOT NULL,
    HeightInches NUMBER(10, 2)  NOT NULL,
    PRIMARY KEY (RecordDate, MemberID)
);
```

# Adding Foreign Keys & Check Constraints

```sql
-- For the Members table
ALTER TABLE Members
ADD CONSTRAINT fkMembersMbrID FOREIGN KEY (TrainerMbrID) REFERENCES Members(MemberID);
ALTER TABLE Members
ADD CONSTRAINT chkMembersGender CHECK (Gender IN ('M', 'F'));
ALTER TABLE Members
ADD CONSTRAINT chkMembersMembershipType CHECK (MembershipType IN ('Standard', 'Premium'));

-- For the Payments table
ALTER TABLE Payments
ADD CONSTRAINT fkPaymentsMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);
ALTER TABLE payments
ADD CONSTRAINT chkPaymentsAmount CHECK (Amount IN (50, 100));
ALTER TABLE payments
ADD CONSTRAINT chkPaymentsPaymentMethod CHECK (PaymentMethod IN ('Cash', 'Debit Card', 'Credit Card'));

-- For the Attendances table
ALTER TABLE Attendances
ADD CONSTRAINT fkAttendancesMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);

-- For the EquipmentUsages table
ALTER TABLE EquipmentUsages
ADD CONSTRAINT fkEquipUsagesMembID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);
ALTER TABLE EquipmentUsages
ADD CONSTRAINT fkEquipUsagesEquipID FOREIGN KEY (EquipmentID) REFERENCES Equipments(EquipmentID);

-- For the FitnessRecords table
ALTER TABLE FitnessRecords
ADD CONSTRAINT fkFitnessRecordsMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);
```

# Inserting Values from CSV

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | MemberID | FirstName | LastName | Gender | Email | Phone | DateOfBirth | MembershipStart | MembershipEnd | MembershipType | TrainerMbrID |
| 2 | 1 | John | Doe | M | john.doe@example.com | 1.23E+09 | 1990-05-15 | 2023-08-04 | 2024-05-09 | Standard | |
| 3 | 2 | Jane | Smith | F | jane.smith@example.com | 9.88E+09 | 1985-08-20 | 2024-03-29 | 2024-05-11 | Standard | |
| 4 | 3 | Rahul | Kumar | M | rahul.kumar@example.com | 5.55E+09 | 1995-11-10 | 2024-01-05 | 2024-05-02 | Premium | 1 |
| 5 | 4 | Emily | Davis | F | emily.davis@example.com | 4.45E+09 | 1988-04-25 | 2024-04-04 | 2024-05-03 | Premium | 2 |
| 6 | 5 | Juan | Dela Cruz | M | juan.delacruz@example.com | 6.67E+09 | 1992-09-30 | 2024-04-05 | 2024-05-05 | Premium | 1 |
| 7 | 6 | Yuma | Lawerence | M | ylawerence0@miibeian.gov.cn | 1.59E+09 | 1998-08-17 | 2024-04-09 | 2024-05-08 | Premium | 1 |
| 8 | 10 | Demetra | Commander | F | dcommander1@answers.com | 2.38E+09 | 1999-02-14 | 2024-04-09 | 2024-05-08 | Premium | 1 |
| 9 | 12 | Kelly | Panner | M | kpanner2@dell.com | 9.16E+09 | 2004-09-30 | 2024-04-10 | 2024-05-09 | Premium | 1 |
| 10 | 13 | Filmer | Cromly | M | fcromly3@symantec.com | 2.17E+09 | 1995-08-24 | 2024-04-12 | 2024-05-11 | Premium | 2 |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | PaymentID | PaymentDate | Amount | PaymentMethod | MemberID |
| 2 | 1 | 2023-08-04 | 100 | Credit Card | 1 |
| 3 | 3 | 2024-01-05 | 50 | Debit Card | 3 |
| 4 | 12 | 2024-04-03 | 100 | Credit Card | 2 |
| 5 | 13 | 2024-04-04 | 50 | Cash | 4 |
| 6 | 14 | 2024-04-05 | 50 | Debit Card | 5 |
| 7 | 15 | 2024-04-07 | 50 | Credit Card | 6 |
| 8 | 16 | 2024-04-09 | 100 | Cash | 10 |
| 9 | 17 | 2024-04-10 | 100 | Cash | 12 |
| 10 | 18 | 2024-04-10 | 50 | Cash | 1 |
| 11 | 21 | 2024-04-12 | 100 | Credit Card | 13 |
| 12 | 22 | 2024-04-12 | 50 | Credit Card | 2 |

| | A | B | C |
|---|---|---|---|
| 1 | EquipmentID | EquipmentName | EquipmentType |
| 2 | 1 | Dumbbells | Strength |
| 3 | 2 | Barbell | Strength |
| 4 | 3 | Treadmill | Cardio |
| 5 | 4 | Exercise Bike | Cardio |
| 6 | 5 | Elliptical Machine | Cardio |

# Inserting Values from CSV

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Attendanc | AttendanceDate | CheckInTime | CheckOutTime | MemberID |
| 2 | 19 | 2023-08-04 | 2023-08-04 7:30 | 2023-08-04 9:30 | 1 |
| 3 | 20 | 2023-08-05 | 2023-08-05 7:30 | 2023-08-05 9:30 | 1 |
| 4 | 21 | 2023-08-06 | 2023-08-06 7:30 | 2023-08-06 9:30 | 1 |
| 5 | 22 | 2023-08-07 | 2023-08-07 7:30 | 2023-08-07 9:30 | 1 |
| 6 | 23 | 2023-08-08 | 2023-08-08 7:30 | 2023-08-08 9:30 | 1 |
| 7 | 24 | 2023-08-09 | 2023-08-09 7:30 | 2023-08-09 9:30 | 1 |
| 8 | 25 | 2023-08-10 | 2023-08-10 7:30 | 2023-08-10 9:30 | 1 |
| 9 | 26 | 2023-08-11 | 2023-08-11 7:30 | 2023-08-11 9:30 | 1 |
| 10 | 27 | 2023-08-12 | 2023-08-12 7:30 | 2023-08-12 9:30 | 1 |
| 11 | 28 | 2023-08-13 | 2023-08-13 7:30 | 2023-08-13 9:30 | 1 |
| 12 | 29 | 2023-08-14 | 2023-08-14 7:30 | 2023-08-14 9:30 | 1 |
| 13 | 30 | 2023-08-15 | 2023-08-15 7:30 | 2023-08-15 9:30 | 1 |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | UsageDate | MemberID | Equipmen | UsageStart | UsageEnd |
| 2 | 2023-08-04 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 3 | 2023-08-05 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 4 | 2023-08-06 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 5 | 2023-08-07 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 6 | 2023-08-08 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 7 | 2023-08-09 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 8 | 2023-08-10 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 9 | 2023-08-11 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 10 | 2023-08-12 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 11 | 2023-08-13 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 12 | 2023-08-14 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |
| 13 | 2023-08-15 | 1 | 3 | 2023-08-04 7:45 | 2023-08-04 8:30 |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | RecordDate | MemberID | CalorieIntake | WeightLbs | HeightInches |
| 2 | 2023-08-04 | 1 | 2600 | 151 | 68 |
| 3 | 2023-08-05 | 1 | 2600 | 151 | 68 |
| 4 | 2023-08-06 | 1 | 2600 | 150.5 | 68 |
| 5 | 2023-08-07 | 1 | 2600 | 151 | 68 |
| 6 | 2023-08-08 | 1 | 2600 | 150.5 | 68 |
| 7 | 2023-08-09 | 1 | 2600 | 151 | 68 |
| 8 | 2023-08-10 | 1 | 2600 | 151 | 68 |
| 9 | 2023-08-11 | 1 | 2600 | 151.5 | 68 |
| 10 | 2023-08-12 | 1 | 2600 | 151 | 68 |
| 11 | 2023-08-13 | 1 | 2600 | 151.5 | 68 |

# 2

# DATABASE PROGRAMMING

xx

# Definition of Views

```
-- View #1: vwMemberInitialWeight (Member View)
-- This view allows members to check their initial weight.
CREATE OR REPLACE VIEW vwMemberInitialWeight
AS
    SELECT  fr.MemberID,
            fr.RecordDate,
            fr.WeightLbs
    FROM FitnessRecords fr
    INNER JOIN (
        SELECT MemberID, MIN(RecordDate) AS InitialRecordDate
        FROM FitnessRecords
        GROUP BY MemberID) initialfr
    ON fr.MemberID = initialfr.MemberID AND fr.RecordDate = initialfr.InitialRecordDate;
/

-- View #2: vwMemberLatestWeight (Member View)
-- This view allows members to check their latest weight.
CREATE OR REPLACE VIEW vwMemberLatestWeight
AS
    SELECT  fr.MemberID,
            fr.RecordDate,
            fr.WeightLbs
    FROM FitnessRecords fr
    INNER JOIN (
        SELECT MemberID, MAX(RecordDate) AS InitialRecordDate
        FROM FitnessRecords
        GROUP BY MemberID) latestfr
    ON fr.MemberID = latestfr.MemberID AND fr.RecordDate = latestfr.InitialRecordDate;
/
```

# Definition of Views

```
-- View #3: vwMemberInitialVsLatest (Member View) - (View #1+2)
-- This view allows members to see both their initial and latest weights.
CREATE OR REPLACE VIEW vwMemberInitialVsLatest
AS
    SELECT i.MemberID,
           i.RecordDate AS InitialRecordDate,
           i.WeightLbs AS InitialWeightLbs,
           l.RecordDate AS LatestRecordDate,
           l.WeightLbs AS LatestWeightLbs
    FROM vwMemberInitialWeight i
    INNER JOIN vwMemberLatestWeight l
    ON i.MemberID = l.MemberID;
/

-- View #4: vwMemberLatestStatus (Member View)
-- This view allows members to review their membership status and track their latest progress.
CREATE OR REPLACE VIEW vwMemberLatestStatus
AS
    SELECT m.MemberID, m.FirstName, m.LastName, m.Gender, m.DateOfBirth,
           m.MembershipStart, m.MembershipEnd,
           vs.InitialRecordDate, vs.InitialWeightLbs,
           vs.LatestRecordDate, vs.LatestWeightLbs
    FROM Members m
    LEFT JOIN vwMemberInitialVsLatest vs
    ON m.MemberID = vs.MemberID;
/
```

# Definition of Views

```
-- View #5: vwTrainerNutritionTraining (Trainer View)
-- This view allows trainers to tailor workouts, monitor nutrition intake, and optimize training plans.
CREATE OR REPLACE VIEW vwTrainerNutritionTraining
AS
    SELECT  m.MemberID, fr.RecordDate,
            m.FirstName, m.LastName, m.DateOfBirth, m.Gender,
            GetCalorieMaintenance(fr.RecordDate, m.MemberID) AS CalorieMaintenance,
            fr.CalorieIntake, fr.WeightLbs, fr.HeightInches,
            GetBMIValue(fr.HeightInches, fr.WeightLbs) AS BMIValue,
            GetBMIClass(GetBMIValue(fr.HeightInches, fr.WeightLbs)) AS BMIClass,
            e.EquipmentName, e.EquipmentType, u.UsageStart, u.UsageEnd
    FROM Members m
    JOIN FitnessRecords fr
        ON m.MemberID = fr.MemberID
    JOIN EquipmentUsages u
        ON m.MemberID = u.MemberID AND u.UsageDate = fr.RecordDate
    JOIN Equipments e
        ON u.EquipmentID = e.EquipmentID;
/

-- View #6: vwFinanceMembershipPayment (Finance View)
-- This view allows finance staff to monitor payment statuses, track membership durations.
CREATE OR REPLACE VIEW vwFinanceMembershipPayment
AS
    SELECT m.MemberID, m.FirstName, m.LastName,
        m.MembershipStart, m.MembershipEnd, m.MembershipType,
        p.PaymentID, p.PaymentDate, p.Amount, p.PaymentMethod
    FROM Members m
    LEFT JOIN Payments p
    ON m.MemberID = p.MemberID;
/
```

# Definition of Functions

```sql
-- Function to calculate the BMI Value
CREATE OR REPLACE FUNCTION GetBMIValue (
    p_HeightInches IN FitnessRecords.HeightInches%TYPE,
    p_WeightLbs IN FitnessRecords.WeightLbs%TYPE
) RETURN DECIMAL AS
    v_BMI DECIMAL(10, 2);
BEGIN
    -- Calculate BMI using the formula (Weight / Height^2) * 703
    v_BMI := (p_WeightLbs / POWER(p_HeightInches, 2)) * 703;

    -- Return the calculated BMI value
    RETURN v_BMI;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any potential errors
        RETURN NULL;
END GetBMIValue;
/
```

```sql
-- Function to determine the BMI classification
CREATE OR REPLACE FUNCTION GetBMIClass (
    p_BMI IN DECIMAL
) RETURN VARCHAR AS
    v_BMICategory VARCHAR(50);
BEGIN
    -- Initialize the category variable
    v_BMICategory := NULL;

    -- Determine BMI category
    IF p_BMI < 18.5 THEN
        v_BMICategory := 'Underweight';
    ELSIF p_BMI >= 18.5 AND p_BMI < 25 THEN
        v_BMICategory := 'Normal';
    ELSIF p_BMI >= 25 AND p_BMI < 30 THEN
        v_BMICategory := 'Overweight';
    ELSIF p_BMI >= 30 THEN
        v_BMICategory := 'Obese';
    END IF;

    -- Return the determined category
    RETURN v_BMICategory;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any potential errors
        RETURN NULL;
END GetBMIClass;
/
```

# Definition of Stored Procedures

```
-- This stored procedure is responsible for adding a new member to the database.
CREATE OR REPLACE PROCEDURE spAddMember (
    p_FirstName       Members.FirstName%TYPE,
    p_LastName        Members.LastName%TYPE,
    p_Gender          Members.Gender%TYPE,
    p_Email           Members.Email%TYPE DEFAULT NULL,
    p_Phone           Members.Phone%TYPE,
    p_DateOfBirth     Members.DateOfBirth%TYPE,
    p_TrainerMbrID    Members.TrainerMbrID%TYPE DEFAULT NULL
) IS
    v_tmpMemberID     Members.MemberID%TYPE;
BEGIN
    -- Check if mandatory fields are provided
    IF p_FirstName IS NULL OR p_LastName IS NULL OR p_Gender IS NULL
       OR p_Phone IS NULL OR p_DateOfBirth IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mandatory fields cannot be null.');
    END IF;

    -- Insert member into Members table
    INSERT INTO Members (
        FirstName, LastName, Gender, Email, Phone, DateOfBirth, TrainerMbrID
    ) VALUES (
        p_FirstName, p_LastName, p_Gender, p_Email, p_Phone, p_DateOfBirth, p_TrainerMbrID
    );

    -- Print success message
    SELECT MAX(MemberID) INTO v_tmpMemberID FROM Members;
    DBMS_OUTPUT.PUT_LINE('Member successfully added. Your Member ID is: ' || v_tmpMemberID);

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END spAddMember;
/
```

# Definition of Stored Procedures

```sql
-- This stored procedure is responsible for member check-ins while checking active membership period.
CREATE OR REPLACE PROCEDURE spMemberCheckIn (
    p_MemberID IN Members.MemberID%TYPE
)
AS
    v_AttendanceDate Attendances.AttendanceDate%TYPE := SYSDATE;
    v_CheckInTime    Attendances.CheckInTime%TYPE := SYSTIMESTAMP;
    v_MembershipStart Members.MembershipStart%TYPE;
    v_MembershipEnd   Members.MembershipEnd%TYPE;
BEGIN
    -- Check if MemberID is provided
    IF p_MemberID IS NULL THEN
        RAISE_APPLICATION_ERROR(-20001, 'MemberID cannot be null.');
    END IF;

    -- Check if the member exists
    BEGIN
        SELECT MembershipStart, MembershipEnd
        INTO v_MembershipStart, v_MembershipEnd
        FROM Members
        WHERE MemberID = p_MemberID;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20002, 'Member ID does not exist.');
    END;

    -- Check if the attendance date falls within the member's active membership period
    IF v_AttendanceDate < v_MembershipStart OR v_AttendanceDate > v_MembershipEnd THEN
        RAISE_APPLICATION_ERROR(-20003, 'Attendance date is not within the member''s active membership period.');
    END IF;
    -- Insert attendance record into the Attendances table
    INSERT INTO Attendances (AttendanceDate, CheckInTime, MemberID)
    VALUES (v_AttendanceDate, v_CheckInTime, p_MemberID);

    -- Success message
    DBMS_OUTPUT.PUT_LINE('Member successfully checked in.');
EXCEPTION
    WHEN OTHERS THEN
      -- Handle unexpected errors
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/
```

# Definition of Triggers

```
-- Trigger #1: tgDeleteAttendanceOutsideGymHours
-- This trigger ensures records outside of gym hours are deleted from the Attendances table.
CREATE OR REPLACE TRIGGER tgDeleteAttendanceOutsideGymHours
BEFORE INSERT ON Attendances
FOR EACH ROW
DECLARE
    v_GymOpenTime    CONSTANT Attendances.CheckInTime%TYPE :=
        TO_TIMESTAMP(TO_CHAR(SYSDATE, 'DD-MON-YYYY') || ' ' || '04:30:00', 'DD-MON-YYYY HH24:MI:SS');
    v_GymCloseTime   CONSTANT Attendances.CheckInTime%TYPE :=
        TO_TIMESTAMP(TO_CHAR(SYSDATE, 'DD-MON-YYYY') || ' ' || '23:30:00', 'DD-MON-YYYY HH24:MI:SS');
    v_GymLastCheckIn CONSTANT Attendances.CheckInTime%TYPE := v_GymCloseTime - INTERVAL '1' HOUR;
BEGIN
    -- Check if the CheckInTime falls outside gym hours
    IF :NEW.CheckInTime < v_GymOpenTime OR :NEW.CheckInTime > v_GymLastCheckIn THEN
        RAISE_APPLICATION_ERROR(-20113, 'TG1: Check-in is outside gym hours.');
    END IF;
END;
/
```

# Definition of Triggers

```
-- Trigger #2: tgPreventCheckinsWithoutCheckout
-- This trigger ensures that members cannot check in multiple times without clocking out first.
CREATE OR REPLACE TRIGGER tgPreventCheckinsWithoutCheckout
BEFORE INSERT ON Attendances
FOR EACH ROW
DECLARE
    v_count INTEGER;
BEGIN
    -- Check if the member has already checked in but not checked out
    SELECT COUNT(*)
    INTO v_count
    FROM Attendances
    WHERE MemberID = :NEW.MemberID
    AND TO_TIMESTAMP(AttendanceDate) = TO_TIMESTAMP(:NEW.AttendanceDate)
    AND CheckOutTime IS NULL;

    -- If such a record exists, raise an error
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'TG2: Please check-out first.');
    END IF;

    -- If no violations, the new row will be inserted
END;
/
```

# Definition of Triggers

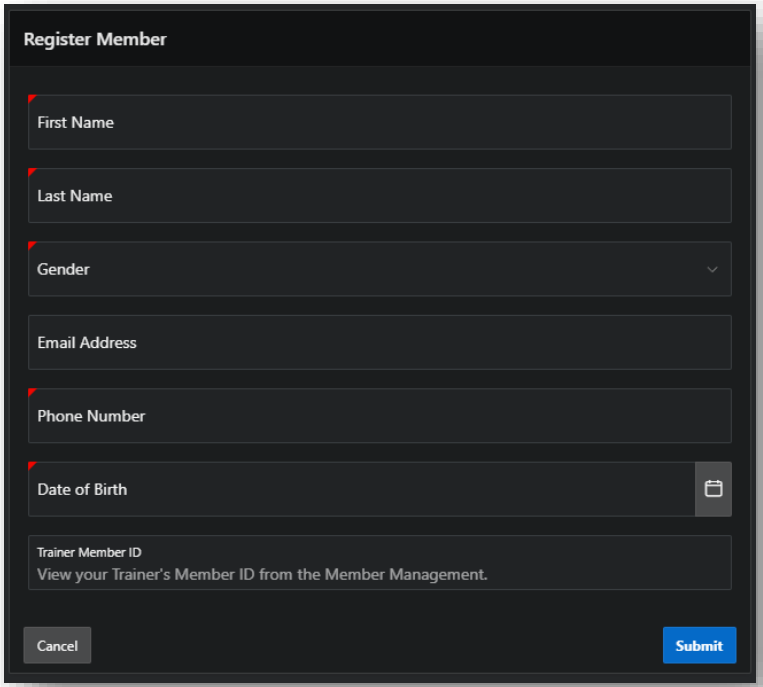
```
-- Trigger #3: tgPreventIncorrectPayment
-- Ensures correct payment amounts based on trainer membership status.
CREATE OR REPLACE TRIGGER tgPreventIncorrectPayment
BEFORE INSERT ON Payments
FOR EACH ROW
DECLARE
    v_TrainerMbrID Members.TrainerMbrID%TYPE;
BEGIN
    -- Retrieve TrainerMbrID for the MemberID being inserted
    SELECT TrainerMbrID
    INTO v_TrainerMbrID
    FROM Members
    WHERE MemberID = :NEW.MemberID;

    -- Validate the Amount based on TrainerMbrID
    IF :NEW.Amount NOT IN (50, 100) THEN
        RAISE_APPLICATION_ERROR(-20002, 'TG3: Must be $50 or $100.');
    ELSIF v_TrainerMbrID IS NULL AND :NEW.Amount <> 50 THEN
        RAISE_APPLICATION_ERROR(-20003, 'TG3: Amount should be 50.');
    ELSIF v_TrainerMbrID IS NOT NULL AND :NEW.Amount <> 100 THEN
        RAISE_APPLICATION_ERROR(-20004, 'TG3: Amount should be 100.');
    END IF;

    -- If validations pass, the row is inserted automatically.
END;
/
```

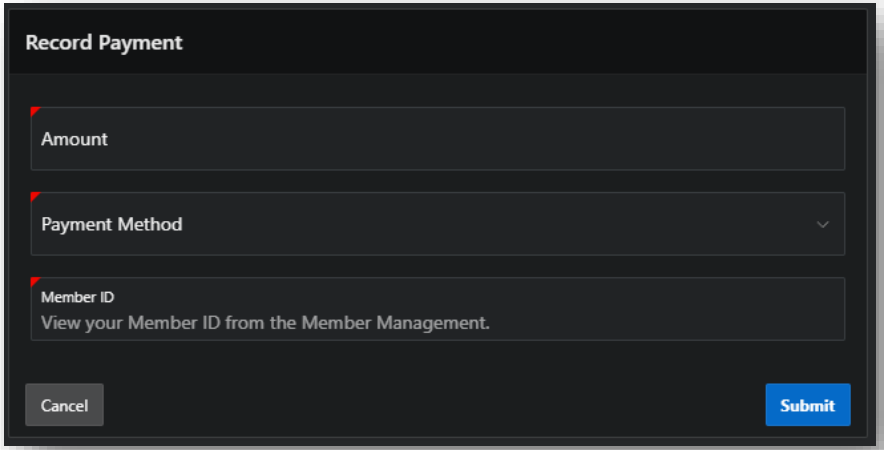
# 3

# APPLICATION DEVELOPMENT

xx

# Input Forms


Register Member
First Name
Last Name
Gender
Email Address
Phone Number
Date of Birth
Trainer Member ID
View your Trainer's Member ID from the Member Management.
Cancel
Submit


Record Payment
Amount
Payment Method
Member ID
View your Member ID from the Member Management.
Cancel
Submit

# Classic Reports

## Member status

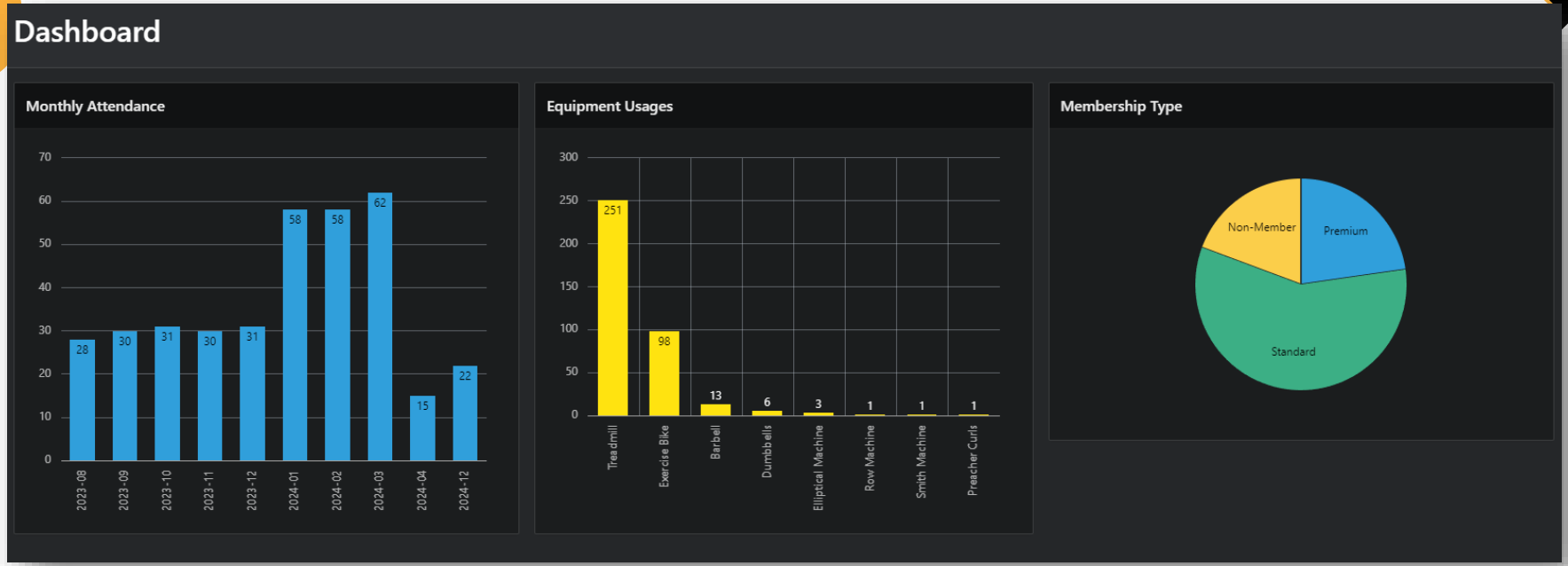
Go Actions

| Memberid | Firstname | Lastname | Gender | Dateofbirth | Membershipstart | Membershipend | Initialrecorddate | Initialweightlbs | Latestrecorddate | Latestweightlbs |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | M | 5/15/1990 | 8/4/2023 | 9/3/2023 | 8/4/2023 | 151 | 4/6/2024 | 151.5 |
| 2 | Jane | Smith | F | 8/20/1985 | 3/29/2024 | 5/11/2024 | 3/29/2024 | 170.5 | 4/6/2024 | 169.5 |
| 3 | Rahul | Kumar | M | 11/10/1995 | 1/5/2024 | 5/2/2024 | 1/5/2024 | 150.3 | 4/6/2024 | 120.2 |
| 4 | Emily | Davis | F | 4/25/1988 | 4/4/2024 | 5/3/2024 | 4/5/2024 | 150.3 | 4/6/2024 | 150.3 |
| 5 | Juan | Dela Cruz | | | | | | 0.2 | 4/6/2024 | 120.2 |
| 6 | Yuma | Lawerence | | | | | | | | |
| 7 | Lance | Mirano | | | | | | 50 | 12/7/2024 | 400 |
| 10 | Demetra | Commander | | | | | | | | |
| 12 | Kelly | Panner | | | | | | 0.5 | 4/10/2024 | 160.5 |
| 13 | Filmer | Cromly | | | | | | 50 | 4/12/2024 | 150 |

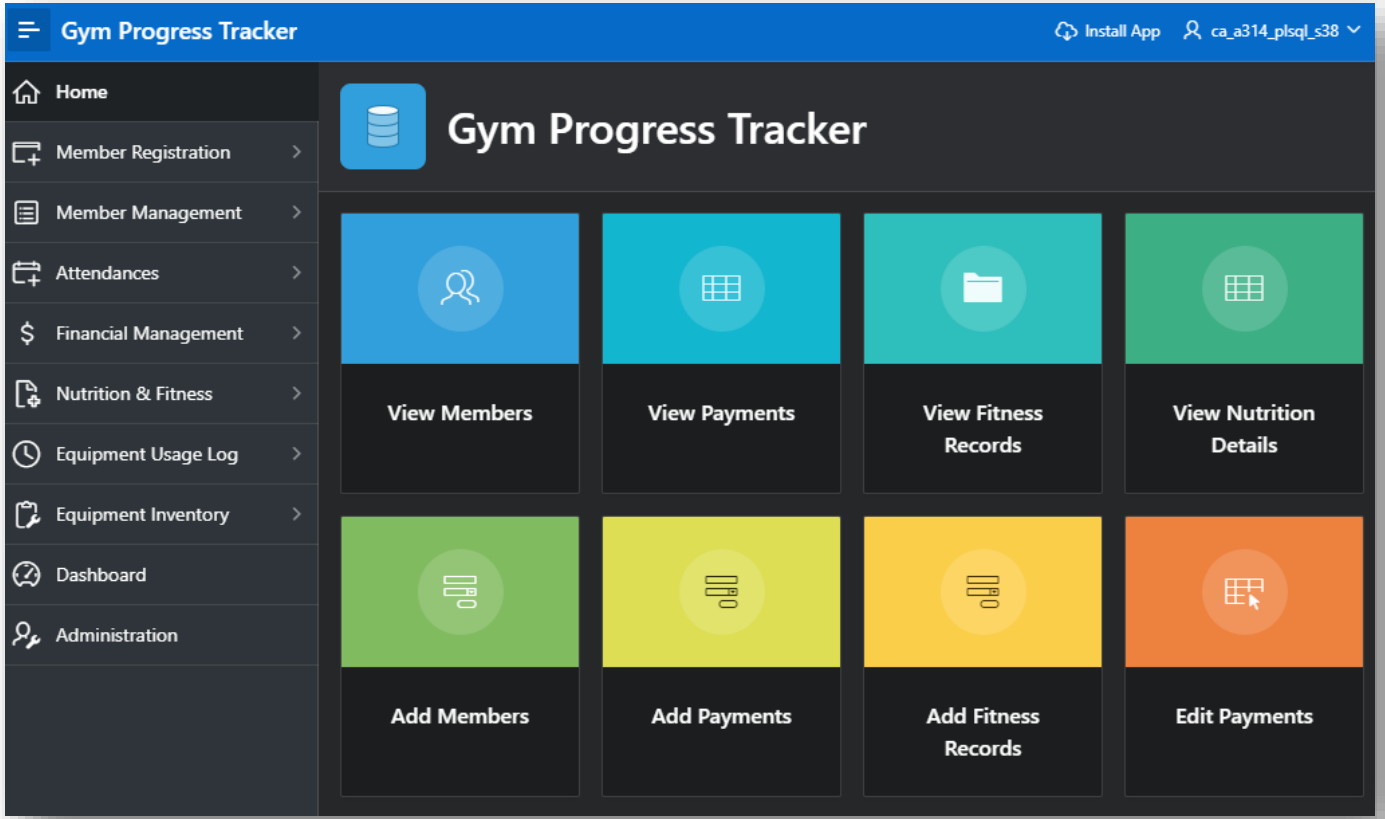
## PAYMENT (VIEW)

| Payment ID | Payment Date | Payment Amount | Payment Method | Member ID |
|---|---|---|---|---|
| 1 | 8/4/2023 | 50 | Credit Card | 1 |
| 2 | 12/3/2024 | 50 | Cash | 7 |
| 3 | 1/5/2024 | 50 | Debit Card | 3 |
| 12 | 4/3/2024 | 50 | Credit Card | 2 |
| 13 | 4/4/2024 | 50 | Cash | 4 |
| 14 | 4/5/2024 | 50 | Debit Card | 5 |
| 15 | 4/7/2024 | 50 | Credit Card | 6 |
| 16 | 4/9/2024 | 100 | Cash | 10 |
| 17 | 4/10/2024 | 100 | Cash | 12 |

# Charts


Dashboard
Monthly Attendance
70
60
50
40
30
20
10
0
28
30
31
30
31
58
58
62
15
22
2023-08
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-12
Equipment Usages
300
250
200
150
100
50
0
251
98
13
6
3
1
1
1
Treadmill
Exercise Bike
Barbell
Dumbbells
Elliptical Machine
Row Machine
Smith Machine
Preacher Curls
Membership Type
Non-Member
Premium
Standard

# Navigation Structure



# THANKS!

Does anyone have any questions?