CPRO 1301: FINAL PROJECT

GYM PROGRESS TRACKER DATABASE

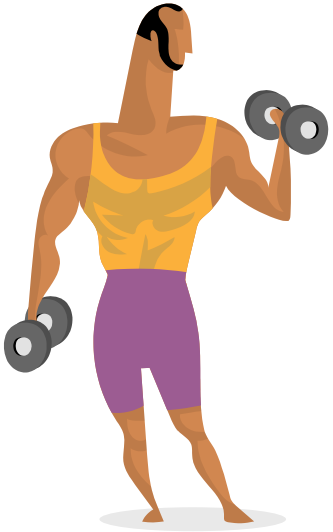
JOASH DALIGCON (000358654)
LANCE MIRANO (000368826)

1

DATABASE DESIGN

xx

Overview

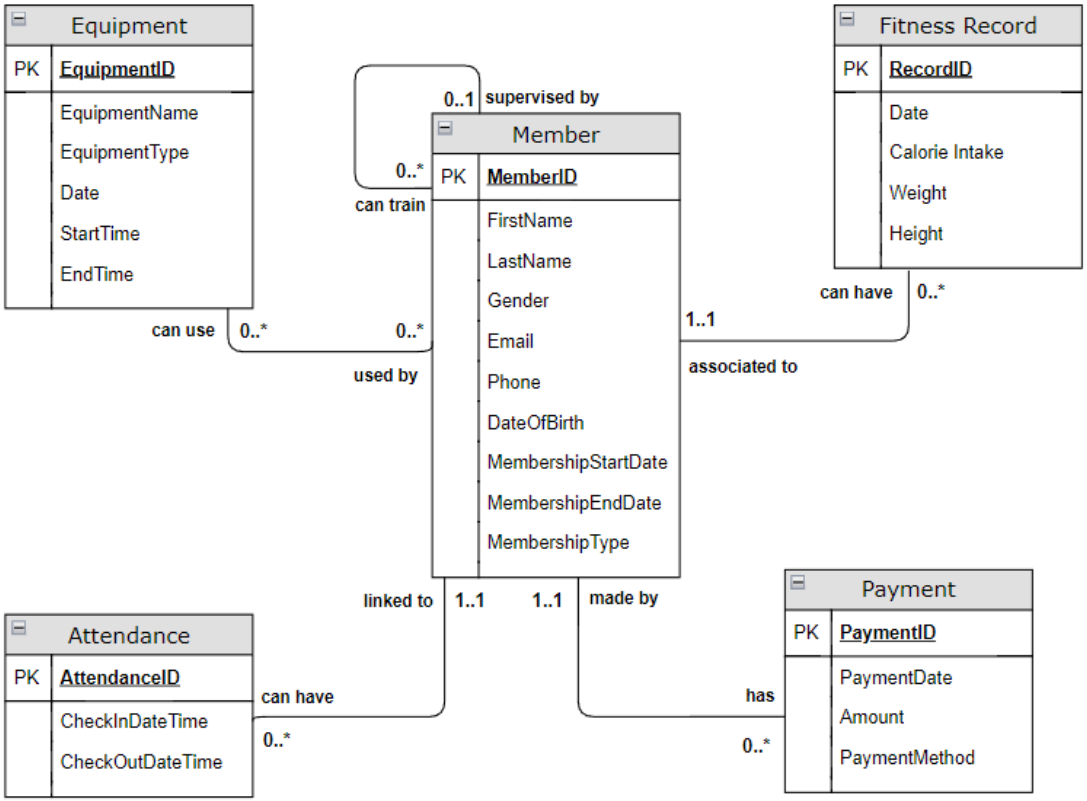


This database manages aspects of gym operations such as member registration, attendance tracking, and payment transactions. It also monitors members' progress through attributes such as calorie intake, height and weight for effective fitness management. Furthermore, it keep tracks of trainer assignments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

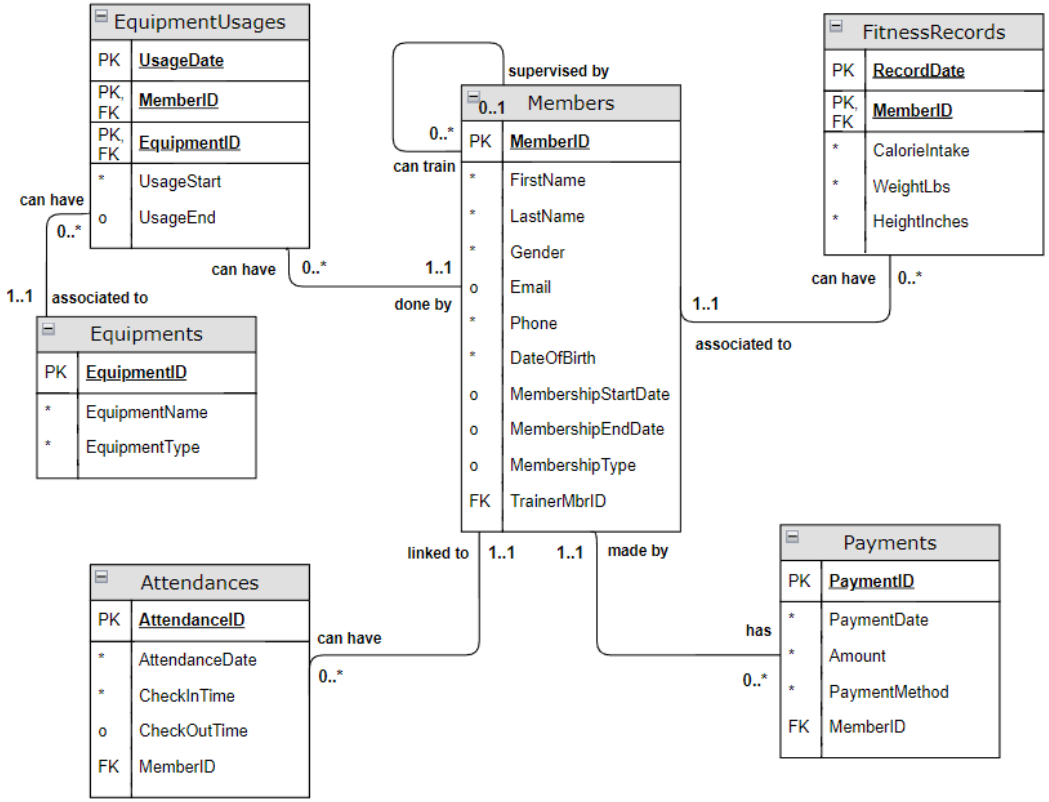
Business Rules

1. Each member may be supervised by a trainer
2. Each member can train zero or many members
3. Each member has zero or many payment records
4. Each payment record is made by one and only one member
5. Each attendance record is linked to one and only one member
6. Each member can have zero or many attendance records
7. Each fitness record is associated to one and only one member
8. Each member can have zero or many fitness records
9. Each gym equipment can be used by zero or many members
10. Each member can use zero or many gym equipment once per day

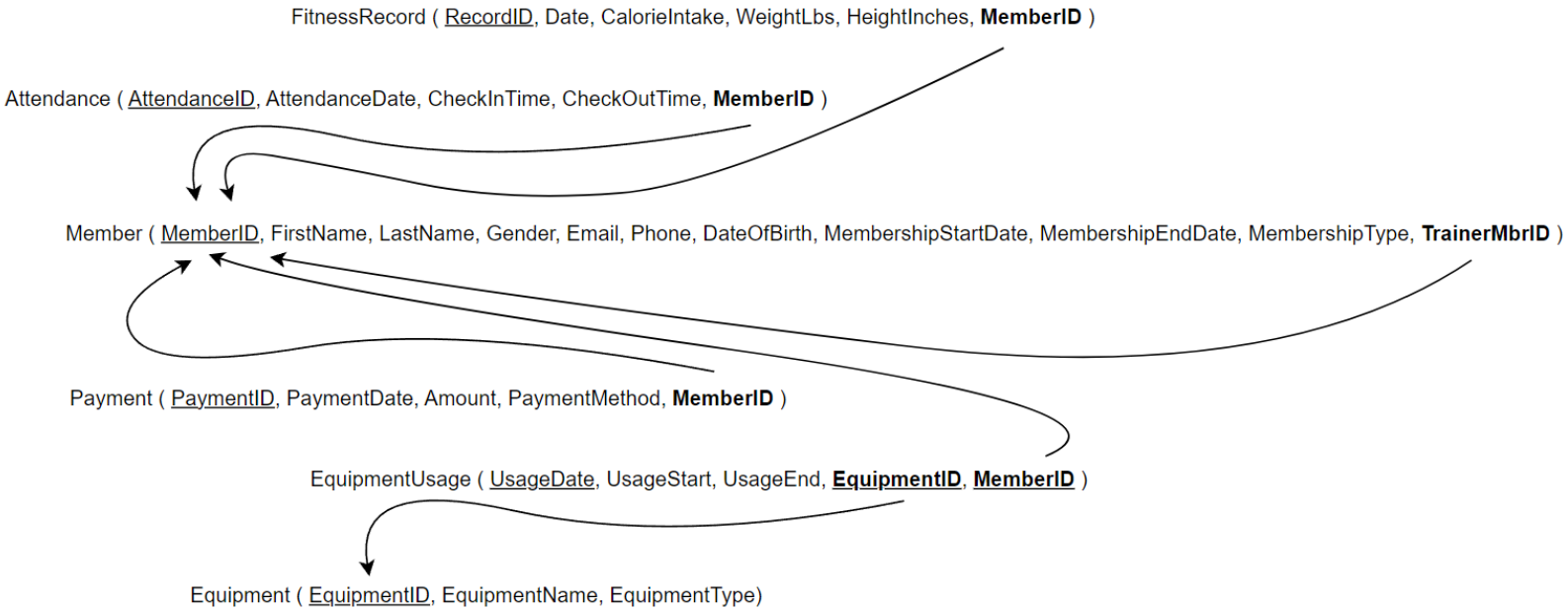
Conceptual Model



Relational Model



Relational Schema



2

DATABASE PROGRAMMING

xx

Creating the tables

```
CREATE TABLE Members (
    MemberID        INT IDENTITY(1,1)   NOT NULL,
    FirstName       NVARCHAR(50)        NOT NULL,
    LastName        NVARCHAR(50)        NOT NULL,
    Gender          CHAR(1)             NOT NULL,
    Email           NVARCHAR(50)        NULL,
    Phone           NVARCHAR(20)        NOT NULL,
    DateOfBirth     DATE                NOT NULL,
    MembershipStart DATE                NULL,
    MembershipEnd   DATE                NULL,
    MembershipType  NVARCHAR(10)        NULL,
    TrainerMbrID    INT                 NULL,
    PRIMARY KEY (MemberID),
    CHECK (Gender = 'M' OR Gender = 'F'),
    CHECK (MembershipType = 'Standard' OR MembershipType = 'Premium')
)
GO
```

```
CREATE TABLE Payments (
    PaymentID       INT IDENTITY(1,1)   NOT NULL,
    PaymentDate     DATE                NOT NULL,
    Amount          MONEY               NOT NULL,
    PaymentMethod   NVARCHAR(50)        NOT NULL,
    MemberID        INT                 NOT NULL,
    PRIMARY KEY (PaymentID),
    CHECK (Amount = 50 OR Amount = 100),
    CHECK (PaymentMethod IN ('Cash', 'Debit Card', 'Credit Card'))
)
GO
```

```
CREATE TABLE FitnessRecords (
    RecordDate          DATE                NOT NULL,
    MemberID            INT                 NOT NULL,
    CalorieIntake       DECIMAL(10,2)       NOT NULL,
    WeightLbs           DECIMAL(10,2)       NOT NULL,
    HeightInches        DECIMAL(10,2)       NOT NULL,
    PRIMARY KEY (RecordDate, MemberID)
)
GO
```

```
CREATE TABLE Attendances (
    AttendanceID    INT IDENTITY(1,1)   NOT NULL,
    AttendanceDate  DATE                NOT NULL,
    CheckInTime     TIME                NOT NULL,
    CheckOutTime    TIME                NULL,
    MemberID        INT                 NOT NULL,
    PRIMARY KEY (AttendanceID)
)
GO
```

```
CREATE TABLE Equipments (
    EquipmentID     INT IDENTITY(1,1)   NOT NULL,
    EquipmentName   NVARCHAR(50)        NOT NULL,
    EquipmentType   NVARCHAR(50)        NOT NULL,
    PRIMARY KEY (EquipmentID)
)
GO
```

```
CREATE TABLE EquipmentUsages (
    UsageDate   DATE    NOT NULL,
    MemberID    INT     NOT NULL,
    EquipmentID INT     NOT NULL,
    UsageStart  TIME    NOT NULL,
    UsageEnd    TIME    NULL,
    -- Member can use each equipment once per day:
    --   MemberID + UsageDate + EquipmentID must be unique
    PRIMARY KEY (UsageDate, MemberID, EquipmentID)
)
GO
```

Adding Foreign Keys

```
-- For the Members table
ALTER TABLE Members
ADD CONSTRAINT fkMembersMbrID FOREIGN KEY (TrainerMbrID) REFERENCES Members(MemberID);
-- For the Payments table
ALTER TABLE Payments
ADD CONSTRAINT fkPaymentsMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);
-- 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 fkEquipUsagesMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    CONSTRAINT fkEquipUsagesEqpID FOREIGN KEY (EquipmentID) REFERENCES Equipments(EquipmentID);
-- For the FitnessRecords table
ALTER TABLE FitnessRecords
ADD CONSTRAINT fkFitnessRecordsMbrID FOREIGN KEY (MemberID) REFERENCES Members(MemberID);
```

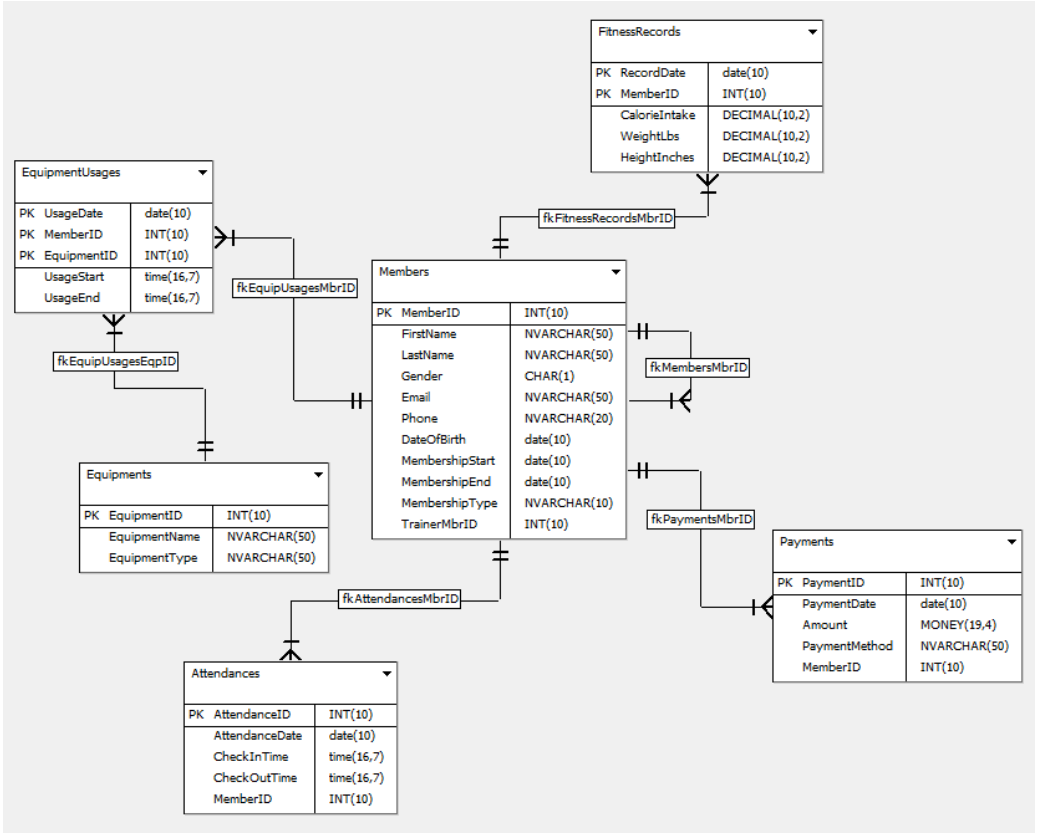
Inserting Values (1)

```sql
-- Insert value to Members table
INSERT INTO Members (FirstName, LastName, Gender, Email, Phone, DateOfBirth, MembershipStart, MembershipEnd, MembershipType, TrainerMbrID)
VALUES
    ('John', 'Doe', 'M', 'john.doe@example.com', '1234567890', '1990-05-15', '2024-03-02', '2024-04-01', 'Premium', NULL),
    ('Jane', 'Smith', 'F', 'jane.smith@example.com', '9876543210', '1985-08-20', '2024-04-03', '2024-05-02', 'Premium', NULL),
    ('Rahul', 'Kumar', 'M', 'rahul.kumar@example.com', '5551234567', '1995-11-10', '2024-04-03', '2024-05-02', 'Standard', 1),
    ('Emily', 'Davis', 'F', 'emily.davis@example.com', '4449876543', '1988-04-25', '2024-04-04', '2024-05-03', 'Standard', 2),
    ('Juan', 'Dela Cruz', 'M', 'juan.delacruz@example.com', '6667890123', '1992-09-30', '2024-04-06', '2024-05-05', 'Standard', 1);
GO
-- Insert value to Payments table
INSERT INTO Payments (PaymentDate, Amount, PaymentMethod, MemberID)
VALUES
    ('2024-04-03', 100.00, 'Credit Card', 2),
    ('2024-04-04', 50.00, 'Cash', 4),
    ('2024-04-05', 50.00, 'Debit Card', 5),
    ('2024-04-07', 50.00, 'Credit Card', 6),
    ('2024-04-09', 100.00, 'Cash', 10),
    ('2024-04-10', 100.00, 'Cash', 12),
    ('2024-04-10', 50.00, 'Cash', 1);
GO
-- Insert value to Attendances table
INSERT INTO Attendances (AttendanceDate, CheckInTime, CheckOutTime, MemberID)
VALUES
    ('2024-04-06', '07:30:00', NULL, 1),
    ('2024-04-06', '08:15:00', '09:30:00', 2),
    ('2024-04-06', '08:30:00', NULL, 3),
    ('2024-04-06', '08:45:00', NULL, 4),
    ('2024-04-06', '09:00:00', NULL, 5);
GO
```

Inserting Values (2)

```
-- Insert value to Equipments table
INSERT INTO Equipments (EquipmentName, EquipmentType)
VALUES
    ('Dumbbells', 'Strength'),
    ('Barbell', 'Strength'),
    ('Treadmill', 'Cardio'),
    ('Exercise Bike', 'Cardio'),
    ('Elliptical Machine', 'Cardio')
GO
-- Insert value to EquipmentUsages table
INSERT INTO EquipmentUsages (MemberID, UsageDate, UsageStart, UsageEnd, EquipmentID)
VALUES
    (1, '2024-04-05', '08:00:00', '08:30:00', 1),
    (2, '2024-04-05', '08:30:00', '09:15:00', 2),
    (3, '2024-04-05', '09:00:00', '09:30:00', 3),
    (4, '2024-04-05', '09:15:00', '09:45:00', 4),
    (5, '2024-04-05', '09:30:00', '10:15:00', 5),
    (1, '2024-04-06', '08:00:00', '08:30:00', 1),
    (2, '2024-04-06', '08:30:00', '09:15:00', 2),
    (3, '2024-04-06', '09:00:00', '09:30:00', 3),
    (4, '2024-04-06', '09:15:00', '09:45:00', 4),
    (5, '2024-04-06', '09:30:00', '10:15:00', 5);
GO
   Insert value to FitnessRecords table
INSERT INTO FitnessRecords (RecordDate, MemberID, CalorieIntake, WeightLbs, HeightInches)
VALUES
    ('2024-04-05', 1, 1500, 170.5, 70),
    ('2024-04-05', 2, 3000, 170.5, 70),
    ('2024-04-05', 3, 3000, 150.3, 68),
    ('2024-04-05', 4, 4000, 150.3, 68),
    ('2024-04-05', 5, 1800, 120.2, 65),
    ('2024-04-06', 1, 1500, 168.5, 70),
    ('2024-04-06', 2, 3000, 169.5, 70),
    ('2024-04-06', 3, 1500, 120.2, 68),
    ('2024-04-06', 4, 3000, 150.3, 68),
    ('2024-04-06', 5, 1800, 120.2, 65);
GO
```

Physical Model



3

DATABASE PROGRAMMING & QUERYING

xx

Sample View Definitions (1)

```sql
-- This view allows finance staff to monitor payment statuses, track membership durations.
CREATE 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;
GO
```

	MemberID	FirstName	LastName	MembershipStart	MembershipEnd	PaymentID	PaymentDate	Amount	PaymentMethod
2	1	John	Doe	2023-08-04	2024-05-09	18	2024-04-10	50.00	Cash
3	2	Jane	Smith	2024-03-29	2024-05-11	12	2024-04-03	100.00	Credit Card
4	2	Jane	Smith	2024-03-29	2024-05-11	22	2024-04-12	50.00	Credit Card
5	3	Rahul	Kumar	2024-01-05	2024-05-02	3	2024-01-05	50.00	Debit Card
6	4	Emily	Davis	2024-04-04	2024-05-03	13	2024-04-04	50.00	Cash
7	5	Juan	Dela Cruz	2024-04-05	2024-05-05	14	2024-04-05	50.00	Debit Card
8	6	Yuma	Lawerence	2024-04-09	2024-05-08	15	2024-04-07	50.00	Credit Card
9	10	Demetra	Commander	2024-04-09	2024-05-08	16	2024-04-09	100.00	Cash
10	12	Kelly	Panner	2024-04-10	2024-05-09	17	2024-04-10	100.00	Cash
11	13	Filmer	Cromly	2024-04-12	2024-05-11	21	2024-04-12	100.00	Credit Card

Sample View Definitions (2)

```sql
-- This view allows members to review their membership status and track their latest progress.
CREATE 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;
GO
```

	MemberID	FirstName	LastName	Gender	DateOfBirth	MembershipStart	MembershipEnd	InitialRecordDate	InitialWeightLbs	LatestRecordDate	LatestWeightLbs
1	1	John	Doe	M	1990-05-15	2023-08-04	2024-05-09	2023-08-04	151.00	2024-04-06	151.50
2	2	Jane	Smith	F	1985-08-20	2024-03-29	2024-05-11	2024-03-29	170.50	2024-04-06	169.50
3	3	Rahul	Kumar	M	1995-11-10	2024-01-05	2024-05-02	2024-01-05	150.30	2024-04-06	120.20
4	4	Emily	Davis	F	1988-04-25	2024-04-04	2024-05-03	2024-04-05	150.30	2024-04-06	150.30
5	5	Juan	Dela Cruz	M	1992-09-30	2024-04-05	2024-05-05	2024-04-05	120.20	2024-04-06	120.20
6	6	Yuma	Lawerence	M	1998-08-17	2024-04-09	2024-05-08	NULL	NULL	NULL	NULL
7	10	Demetra	Commander	F	1999-02-14	2024-04-09	2024-05-08	NULL	NULL	NULL	NULL
8	12	Kelly	Panner	M	2004-09-30	2024-04-10	2024-05-09	2024-04-10	160.50	2024-04-10	160.50
9	13	Filmer	Cromly	M	1995-08-24	2024-04-12	2024-05-11	2024-04-12	150.00	2024-04-12	150.00

Sample View Definitions (3)

```sql
-- This view allows trainers to tailor workouts, monitor nutrition intake, and optimize training plans.
CREATE VIEW vwTrainerNutritionTraining
AS
    SELECT  fr.RecordDate, m.MemberID,
            m.FirstName, m.LastName, m.DateOfBirth, m.Gender,
            dbo.GetCalorieMaintenance(fr.RecordDate, m.MemberID) AS CalorieMaintenance,
            fr.CalorieIntake, fr.WeightLbs, fr.HeightInches,
            dbo.GetBMIValue(fr.HeightInches, fr.WeightLbs) AS BMIValue,
            dbo.GetBMIClass(dbo.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;
GO
```

	RecordDate	MemberID	FirstName	LastName	DateOfBirth	Gender	CalorieMaintenance	CalorieIntake	WeightLbs	HeightInches	BMIValue	BMIClass	EquipmentName	EquipmentType	UsageStart	UsageEnd
339	2024-04-05	4	Emily	Davis	1988-04-25	F	2459.85	4000.00	150.30	68.00	22.85	Normal	Exercise Bike	Cardio	09:15:00.0000000	09:45:00.0000000
340	2024-04-05	5	Juan	Dela Cruz	1992-09-30	M	2252.54	1800.00	120.20	65.00	20.00	Normal	Elliptical Machi...	Cardio	09:30:00.0000000	10:15:00.0000000
341	2024-04-06	1	John	Doe	1990-05-15	M	2770.63	2600.00	151.50	70.00	21.74	Normal	Dumbbells	Strength	08:00:00.0000000	08:30:00.0000000
342	2024-04-06	2	Jane	Smith	1985-08-20	F	2753.58	3000.00	169.50	70.00	24.32	Normal	Barbell	Strength	08:30:00.0000000	09:15:00.0000000
343	2024-04-06	3	Rahul	Kumar	1995-11-10	M	2266.10	1500.00	120.20	64.00	20.63	Normal	Treadmill	Cardio	09:00:00.0000000	09:30:00.0000000
344	2024-04-06	3	Rahul	Kumar	1995-11-10	M	2266.10	1500.00	120.20	64.00	20.63	Normal	Exercise Bike	Cardio	08:45:00.0000000	09:30:00.0000000
345	2024-04-06	4	Emily	Davis	1988-04-25	F	2459.85	3000.00	150.30	68.00	22.85	Normal	Exercise Bike	Cardio	09:15:00.0000000	09:45:00.0000000
346	2024-04-06	5	Juan	Dela Cruz	1992-09-30	M	2252.54	1800.00	120.20	65.00	20.00	Normal	Elliptical Machi...	Cardio	09:30:00.0000000	10:15:00.0000000
347	2024-04-10	12	Kelly	Panner	2004-09-30	M	2926.60	2500.00	160.50	60.50	30.83	Obese	Dumbbells	Strength	22:23:40.2566667	22:28:28.0133333
348	2024-04-12	13	Filmer	Cromly	1995-08-24	M	2766.75	2500.00	150.00	68.00	22.80	Normal	Barbell	Strength	05:12:08.9466667	06:03:02.9466667

Sample Function Definitions

```sql
-- Function to calculate the BMI Value
CREATE FUNCTION GetBMIValue(@HeightInches DECIMAL(10,2), @WeightLbs DECIMAL(10,2))
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @BMI DECIMAL(10,2);
    SET @BMI = (@WeightLbs / POWER(@HeightInches, 2)) * 703;
    RETURN @BMI;
END;
GO
-- Function to determine the BMI classification
CREATE FUNCTION GetBMIClass(@BMI DECIMAL(10,2))
RETURNS NVARCHAR(20)
AS
BEGIN
    DECLARE @BMICategory NVARCHAR(50) = NULL;
    -- Determine BMI category
    IF @BMI < 18.5
        SET @BMICategory = 'Underweight';
    ELSE IF @BMI >= 18.5 AND @BMI < 25
        SET @BMICategory = 'Normal';
    ELSE IF @BMI >= 25 AND @BMI < 30
        SET @BMICategory = 'Overweight';
    ELSE IF @BMI >= 30
        SET @BMICategory = 'Obese';

    RETURN @BMICategory;
END;
GO
```

Sample Stored Proc. Definitions (1)

```sql
-- This stored procedure is responsible for adding a new member to the database.
CREATE PROCEDURE spAddMember
    @FirstName          NVARCHAR(50),
    @LastName           NVARCHAR(50),
    @Gender             CHAR(1),
    @Email              NVARCHAR(50) = NULL,
    @Phone              NVARCHAR(20),
    @DateOfBirth        DATE,
    @TrainerMbrID       INT = NULL
AS
BEGIN
    BEGIN TRY
        -- Check if mandatory fields are provided
        IF (@FirstName IS NULL OR @LastName IS NULL OR @Gender IS NULL
            OR @Phone IS NULL OR @DateOfBirth IS NULL)
            THROW 50001, 'Mandatory fields cannot be null.', 1;
        -- Insert member into Members table
        INSERT INTO Members
            (FirstName, LastName, Gender, Email, Phone, DateOfBirth, TrainerMbrID)
        VALUES
            (@FirstName, @LastName, @Gender, @Email, @Phone, @DateOfBirth, @TrainerMbrID);
        -- Print success message
        DECLARE @tmpMemberID INT;
        SELECT @tmpMemberID = MAX(MemberID) FROM Members;
        PRINT 'Member successfully added. Your Member ID is: '  + CONVERT(varchar, @tmpMemberID);
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred: ' + CONVERT(varchar, ERROR_MESSAGE());
    END CATCH
END
GO
```

Sample Stored Proc. Definitions (2)

```sql
-- This stored procedure is responsible for adding a payment record then updating the membership info of the member.
CREATE PROCEDURE spAddPaymentUpdateMembership
    @MemberID       INT,
    @PaymentAmount  MONEY,
    @PaymentMethod  NVARCHAR(50)
AS
BEGIN
    BEGIN TRY
        -- Check if mandatory fields are provided
        IF (@MemberID IS NULL OR @PaymentAmount IS NULL OR @PaymentMethod IS NULL)
            THROW 50001, 'MemberID and PaymentAmount cannot be null.', 1;
        -- Check existence of TrainerID in Member Record
        DECLARE @TrainerMbrID INT;
        SELECT @TrainerMbrID = TrainerMbrID
        FROM Members
        WHERE MemberID = @MemberID;
        -- Check if any inserted records violate the rule
        IF @TrainerMbrID IS NOT NULL AND @PaymentAmount <> 100
            THROW 50114, 'Payment is incorrect.', 1;

        -- Add payment record
        DECLARE @PaymentDate DATE = GETDATE(); -- Assume payment date is current date
        INSERT INTO Payments
            (PaymentDate, Amount, PaymentMethod, MemberID)
        VALUES
            (@PaymentDate, @PaymentAmount, @PaymentMethod, @MemberID);
        -- Calculate MembershipEnd date (one day before the end of the month following MembershipStart date)
        DECLARE @MembershipStart    DATE = @PaymentDate;
        DECLARE @MembershipEnd      DATE = DATEADD(DAY, -1, DATEADD(MONTH, 1, @PaymentDate));
        -- Check if membership is standard or premium
        DECLARE @MembershipType NVARCHAR(10);
        IF @PaymentAmount = 50
            SET @MembershipType = 'Standard'
        IF @PaymentAmount = 100
            SET @MembershipType = 'Premium'
        -- Check if member is new or just renewing
        DECLARE @OldMembershipStart DATE;
        SELECT @OldMembershipStart = MembershipStart
        FROM Members
        WHERE MemberID = @MemberID;
        IF @OldMembershipStart IS NOT NULL
            SET @MembershipStart = @OldMembershipStart;
        -- Update MembershipStart, MembershipEnd, and MembershipType columns in Members table
        UPDATE Members
        SET MembershipStart = @MembershipStart,
            MembershipEnd   = @MembershipEnd,
            MembershipType  = @MembershipType
        WHERE MemberID      = @MemberID;
        -- Print success message
        PRINT 'Payment record added and membership details updated.';
    END TRY
    BEGIN CATCH
        PRINT 'Error occurred: ' + CONVERT(varchar, ERROR_MESSAGE());
    END CATCH
END
GO
```

Members Table

```
EXEC spAddMember
    @FirstName = 'Filmer',
    @LastName = 'Cromly',
    @Gender = 'M',
    @Email = 'fcromly3@symantec.com',
    @Phone = '2168969541',
    @DateOfBirth = '1995-08-24',
    @TrainerMbrID = 2;
```

```
(1 row affected)
Member successfully added. Your Member ID is: 13
```

	MemberID	FirstName	LastName	Gender	Email	Phone	DateOfBirth	MembershipStart	MembershipEnd	MembershipType	TrainerMbrID
1	1	John	Doe	M	john.doe@example.com	1234567890	1990-05-15	2024-04-10	2024-05-09	Standard	NULL
2	2	Jane	Smith	F	jane.smith@example.com	9876543210	1985-08-20	2024-04-03	2024-05-02	Premium	NULL
3	3	Rahul	Kumar	M	rahul.kumar@example.com	5551234567	1995-11-10	2024-04-03	2024-05-02	Standard	1
4	4	Emily	Davis	F	emily.davis@example.com	4449876543	1988-04-25	2024-04-04	2024-05-03	Standard	2
5	5	Juan	Dela Cruz	M	juan.delacruz@example.com	6667890123	1992-09-30	2024-04-06	2024-05-05	Standard	1
6	6	Yuma	Lawerence	M	ylawerence0@miibeian.gov.cn	1587588503	1998-08-17	2024-04-09	2024-05-08	Standard	1
7	10	Demetra	Commander	F	dcommander1@answers.com	2383570976	1999-02-14	2024-04-09	2024-05-08	Premium	1
8	12	Kelly	Panner	M	kpanner2@dell.com	9164311758	2004-09-30	2024-04-10	2024-05-09	Premium	1
9	13	Filmer	Cromly	M	fcromly3@symantec.com	2168969541	1995-08-24	NULL	NULL	NULL	2

Payments Table

```
EXEC spAddPaymentUpdateMembership
    @MemberID = 13,
    @PaymentAmount = 100,
    @PaymentMethod = 'Credit Card';
```

```
(1 row affected)

(1 row affected)
Payment record added and membership details updated.
```

	PaymentID	PaymentDate	Amount	PaymentMethod	MemberID
6	15	2024-04-07	50.00	Credit Card	6
7	16	2024-04-09	100.00	Cash	10
8	17	2024-04-10	100.00	Cash	12
9	18	2024-04-10	50.00	Cash	1
10	21	2024-04-12	100.00	Credit Card	13

	MemberID	FirstName	LastName	Gender	Email	Phone	DateOfBirth	MembershipStart	MembershipEnd	MembershipType	TrainerMbrID
5	5	Juan	Dela Cruz	M	juan.delacruz@example.com	6667890123	1992-09-30	2024-04-06	2024-05-05	Standard	1
6	6	Yuma	Lawerence	M	ylawerence0@miibeian.gov.cn	1587588503	1998-08-17	2024-04-09	2024-05-08	Standard	1
7	10	Demetra	Comman...	F	dcommander1@answers.com	2383570976	1999-02-14	2024-04-09	2024-05-08	Premium	1
8	12	Kelly	Panner	M	kpanner2@dell.com	9164311758	2004-09-30	2024-04-10	2024-05-09	Premium	1
9	13	Filmer	Cromly	M	fcromly3@symantec.com	2168969541	1995-08-24	2024-04-12	2024-05-11	Premium	2

Attendances Table

```
EXEC spMemberCheckIn
    @MemberID = 13;
```

```
(1 row affected)
Member successfully checked in.
```

	AttendanceID	AttendanceDate	CheckInTime	CheckOutTime	MemberID
339	357	2024-04-06	08:15:00.0000000	09:15:00.0000000	2
340	358	2024-04-06	08:45:00.0000000	09:15:00.0000000	4
341	359	2024-04-06	09:00:00.0000000	09:15:00.0000000	5
342	360	2024-04-10	22:23:40.2566667	22:28:28.0133333	12
343	362	2024-04-12	04:53:21.5666667	NULL	13

```
EXEC spMemberCheckOut
    @MemberID = 13;
```

```
(1 row affected)
Member successfully checked out.
```

	AttendanceID	AttendanceDate	CheckInTime	CheckOutTime	MemberID
339	357	2024-04-06	08:15:00.0000000	09:15:00.0000000	2
340	358	2024-04-06	08:45:00.0000000	09:15:00.0000000	4
341	359	2024-04-06	09:00:00.0000000	09:15:00.0000000	5
342	360	2024-04-10	22:23:40.2566667	22:28:28.0133333	12
343	362	2024-04-12	04:53:21.5666667	06:24:39.7566667	13

Equipments & Usages Table

```
EXEC spAddEquipmentUsageStart
    @MemberID = 13,
    @EquipmentID = 2;
```

```
(1 row affected)
Equipment usage start time successfully added.
```

	EquipmentID	EquipmentName	EquipmentType
1	1	Dumbbells	Strength
2	2	Barbell	Strength
3	3	Treadmill	Cardio
4	4	Exercise Bike	Cardio
5	5	Elliptical Machine	Cardio

	UsageDate	MemberID	EquipmentID	UsageStart	UsageEnd
348	2024-04-06	5	5	09:30:00.0000000	10:15:00.0000000
349	2024-04-09	1	1	20:05:08.0800000	20:42:56.3500000
350	2024-04-09	1	2	20:30:46.8366667	20:42:50.5566667
351	2024-04-10	12	1	22:23:40.2566667	22:28:28.0133333
352	2024-04-12	13	2	05:12:08.9466667	NULL

Equipment Usages Table

```
EXEC spAddEquipmentUsageEnd
    @MemberID = 13,
    @EquipmentID = 2;
```

```
(1 row affected)
Equipment usage end time successfully updated.
```

	UsageDate	MemberID	EquipmentID	UsageStart	UsageEnd
348	2024-04-06	5	5	09:30:00.0000000	10:15:00.0000000
349	2024-04-09	1	1	20:05:08.0800000	20:42:56.3500000
350	2024-04-09	1	2	20:30:46.8366667	20:42:50.5566667
351	2024-04-10	12	1	22:23:40.2566667	22:28:28.0133333
352	2024-04-12	13	2	05:12:08.9466667	06:03:02.9466667

Fitness Records Table

```
EXEC spAddFitnessRecord
    @MemberID = 13,
    @CalorieIntake = 2500,
    @WeightLbs = 150,
    @HeightInches = 68;
```

```
(1 row affected)
Fitness record successfully added. BMI is: Normal
Current BMI value is: 22.80
```

	RecordDate	MemberID	CalorieIntake	WeightLbs	HeightInches
343	2024-04-06	3	1500.00	120.20	64.00
344	2024-04-06	4	3000.00	150.30	68.00
345	2024-04-06	5	1800.00	120.20	65.00
346	2024-04-10	12	2500.00	160.50	60.50
347	2024-04-12	13	2500.00	150.00	68.00

Sample Trigger Definitions (1)

```sql
-- This trigger ensures records outside of gym hours are deleted from the Attendances table.
CREATE TRIGGER tgDeleteAttendanceOutsideGymHours
    ON Attendances
    AFTER INSERT
AS
BEGIN
    -- Define the fixed open and close times for the gym
    DECLARE @GymOpenTime    TIME = '04:30:00';
    DECLARE @GymCloseTime   TIME = '23:30:00';
    -- Define the last check-in as 1 hour before the gym's closing time
    DECLARE @GymLastCheckIn TIME = DATEADD(HOUR, -1, @GymCloseTime);
    -- Check if the inserted attendance record falls OUTSIDE of gym hours
    IF EXISTS (
        SELECT 1 -- Arbitrary value to check existence
        FROM Inserted AS i
        WHERE i.CheckInTime < @GymOpenTime OR i.CheckInTime > @GymLastCheckIn
    )
    BEGIN
        THROW 50113, 'Check-in is outside gym hours.', 1;
        ROLLBACK TRAN;
    END
END
GO
```

```
(0 rows affected)
Error occurred: Check-in is outside gym hours.
```

Sample Trigger Definitions (2)

```sql
-- This trigger ensures that members cannot check in multiple times without clocking out first.
CREATE TRIGGER tgPreventCheckinsWithoutCheckout
    ON Attendances
    INSTEAD OF INSERT
AS
BEGIN
    -- Check if any inserted records violate the rule
    IF EXISTS (
        SELECT 1 -- Arbitrary value to check existence
        FROM Inserted AS i
        JOIN Attendances AS a
            ON i.MemberID = a.MemberID
        WHERE (i.AttendanceDate = a.AttendanceDate
            AND i.CheckInTime IS NOT NULL
            AND a.CheckOutTime IS NULL) -- Member hasn't clocked out yet
    )
        THROW 50114, 'Please check-out first.', 1;
    ELSE
    BEGIN
        -- Perform the actual insert operation
        INSERT INTO Attendances
            (AttendanceDate, CheckInTime, MemberID)
            (SELECT i.AttendanceDate, i.CheckInTime, i.MemberID
                FROM Inserted AS i);
    END
END
GO
```

```
(0 rows affected)
Error occurred: Please check-out first.
```

THANKS!