

# CA1: Database Design and Development

Module Title: Databases and Business Applications

Module Code: B8IT101

Module Leader: Jennifer Byrne

Student Name: Manoj Kumar Thangaraj

Student Code: 10570753

Contents

[CA1: Database Design and Development 1](#_Toc4657376)

[1. Project Overview/Scope 3](#_Toc4657377)

[2. Entity Relationship Diagram 3](#_Toc4657378)

[3. Assumptions Made 3](#_Toc4657379)

[4 Data Dictionary 4](#_Toc4657380)

[5. Technology Used 6](#_Toc4657381)

[6. Test Plan 7](#_Toc4657382)

[7. Reflections on Learning 10](#_Toc4657383)

[8. References 10](#_Toc4657384)

[9. SQL 10](#_Toc4657385)

# Project Overview/Scope

This project provides a technical design and provision of a new database server and the development of a new database for LetsGetFit gym to store the details of the members, trainers, program. The boundaries of this project include setting up the relational database model to record the members personal details, their designed program plan, setting up their membership card in order to login into the gym. The enrollment setup is made in order to bring the members into desired plan that is available to them. The diet and program are designed exclusively for the members and recorded in the database. The trainers set up the program plan and assign to the members. The progress chart database is made in order to develop the members progress in the gym. The brief view of the payment details to store the payment made by their members.

# Entity Relationship Diagram

Diagram, schematic

Description automatically generated

# Assumptions Made

The assumptions that were made in order to design this database model are as follows:

* If customer chooses a plan, he/she will not be allowed to change the plan until its fully completed.
* A separate HR system that employs the trainers is not part to the database model.
* There is a separate system that handles the member payments and salary of trainers.
* Members that checked GDPR option are retained on the system in a dormant state after membership expires.
* The hard delete removes the personal details alone, but the business-related information is kept on record and hard delete is done manually through the stored procedure.

# Data Dictionary

### Entity: Member

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| MemberID | INT IDENTITY (100,1) (PK) | NOT NULL | Auto generated Member ID |
| MemFirstName | VARCHAR (50) | NOT NULL | Member First Name |
| MemSurName | VARCHAR (40) | NOT NULL | Member Surname |
| AddressLine1 | VARCHAR (100) | NOT NULL | Member’s Address |
| AddressLine2 | VARCHAR (100) | NULL | Address Continued |
| City | VARCHAR (30) | NOT NULL | City |
| County | VARCHAR (30) | NOT NULL | Member’s County |
| PostCode | CHAR (10) | NOT NULL | Postcode |
| MemEmailID | VARCHAR (100) | NOT NULL | Email ID |
| MemPhoneNumber | CHAR (13) | NOT NULL | Contact Number |
| GDPR | BOOLEAN | NOT NULL | GDPR Status |
| IsDeleted | CHAR (1) | NOT NULL | Active or Inactive |

### Entity: MemberEnrollment

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| EnrollmentID | INT IDENTITY (1000,1) (PK) | NOT NULL | Auto generated for Member Enrollment |
| EnrollmentPlan | CHAR 10) | NOT NULL | Member can choose the plan |
| MemberID | INT (FK) | NULL | Enrollment corresponds to the member |
| DateStarted | DATETIME | NOT NULL | Start date |
| ExpiryDate | DATETIME | NOT NULL | Expiry date |
| Duration | CHAR (15) | NOT NULL | Duration of the plan |

### Entity: MembershipCard

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| UniqueMemNum | INT IDENTITY (7000,1) (PK) | NOT NULL | Autogenerated number for the card |
| MemberID | INT(FK) | NULL | Card belongs to the member |
| Name | VARCHAR (20) | NOT NULL | Name of the member |
| Image | IMAGE | NOT NULL | Image of Member |

### Entity: Session

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Required | Description |
| SessionID | INT IDENTITY (8000,1) (PK) | NOT NULL | Autogenerated when member log in |
| UniqueMemNum | CHAR (10) (FK) | NULL | Member details |
| TimeIn | DATETIME | NOT NULL | Time of log in |
| TimeOut | DATETIME | NOT NULL | Time of log out |

### Entity: ProgressChart

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| ProgressID | INT IDENTITY (9000,1) (PK) | NOT NULL | Autogenerated ID when recording measurements |
| Member ID | INT(FK) | NULL | Progress of the member |
| Date | DATETIME | NOT NULL | Date recorded |
| BMI | FLOAT | NOT NULL | Body mass index |
| FatPercentage | FLOAT | NOT NULL | Fat percentage |
| MuscleMass | FLOAT | NOT NULL | Muscle weight |

### Entity: Trainer

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| TrainerID | INT IDENTITY (2000,1) (PK) | NOT NULL | Autogenerated ID for trainer |
| FirstName | VARCHAR (100) | NOT NULL | First Name |
| SurName | VARCHAR (50) | NOT NULL | Surname |
| PPSNumber | CHAR (8) | NOT NULL | PPS number |
| PhoneNum | VARCHAR (15) | NOT NULL | Contact number |
| EmailID | VARCHAR (50) | NOT NULL | Email ID |
| GDPR | BOOLEAN | NOT NULL | GDPR choice |
| IsDeleted | CHAR (1) | NOT NULL | Active or Inactive |

### Entity: Payment

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| PaymentID | INT IDENTITY (3000,1) (PK) | NOT NULL | Autogenerated for each payment |
| MemberID | INT(FK) | NOT NULL | Details of who made payment |
| AmountPaid | FLOAT | NOT NULL | Amount Paid |
| TimePaid | DATETIME | NOT NULL | Date Paid |

### Entity: Program

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| ProgramID | INT IDENTITY (4000,1) (PK) | NOT NULL | Autogenerated ID for each program |
| ProgramName | VARCHAR (100) | NOT NULL | Name of the program |
| Duration | VARCHAR (100) | NOT NULL | Duration of the program |
| LastUpdated | DATETIME | NOT NULL | Last Updated |
| ProgramType | VARCHAR (25) | NOT NULL | Type of the program |

### Entity: ProgramPlan

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Required** | **Description** |
| PlanID | INT IDENTITY (6000,1) (PK) | NOT NULL | Autogenerated ID for plan |
| MemberID | INT(FK) | NULL | Member details |
| ProgramID | INT(FK) | NULL | Program details |
| TrainerID | INT(FK) | NULL | Trainer details |
| WorkoutDate | DATETIME | NOT NULL | Date of the workout |
| Description | VARCHAR (100) | NOT NULL | Type of workout |

### Entity: DietPlan

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Required | Description |
| DietID | INT IDENTITY (8000,1) (PK) | NOT NULL | Autogenerated for each diet |
| DietName | VARCHAR (20) | NOT NULL | Name of the diet |
| Calories | FLOAT | NOT NULL | Calories |
| Carbhohydrates | FLOAT | NOT NULL | Amount of Carbs |
| Fat | FLOAT | NOT NULL | Amount of Fat |
| Protein | FLOAT | NOT NULL | Amount of Protein |

# Technology Used

* Microsoft SQL Server
* Lucidchart
* Draw.io

# 

# Test Plan

Table 1 – the test plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Item Tested** | **Test Run** | **Expected Result** | **Actual Result** |
| ProgramPlanDesign | EXEC ProgramPlanDesign @memid,@trainerid,@programid,@dietid,  @workoutdate,@des | It should enter a new row with the details of Member ID, Trainer ID, Program ID, Diet ID, Workout date and Description in the program plan table | The actual result was same as the expected result |
| NewMembersandPlans | EXEC NewMembersandplans @MemFirName, @MemSurName,@Add1,@Add2,@City,@County,  @Postcode,@MemEmail,@MemPhone,@GDPR,@IsDeleted  @memid,@trainerid,@programid,@dietid,@workoutdate  @des,@action | A new row of member details and a new row of program plan details for a new member | The actual result was the expected result that it did entered a new row of details in each table |
| ProgramUpdate | EXEC ProgramUpdate @programname,@duration,@lastupdated,  @programtype,@action | Depends on the action mentioned, it should be able to Insert, Delete, or Update a program | It indeed accomplished the task it was given. |
| TrainersNew | EXEC TrainersNew @firstname,@surname,  @ppsnumber,@phonenum,@emailid,@GDPR,@IsDeleted | A new row of Trainer details listed in the table | The actual result did the same. |
| DeleteMember | EXEC DeleteMember @memid,@action | A row belong to a member is deleted and the foreign key locations are deleted as well if they are in cascade. If the FK is Delete set to null, then the Member ID is alone deleted in the other tables | The actual result met the expected result. |
| SoftDeletedMember | EXEC SoftDeleteMember @memid | Sets the IsDeleted Column to ‘Y’ | The actual result is the same |

# Reflections on Learning

A simple and reliable database model was created for the LetsgetFit gym. This project gave me an opportunity to implement my skills in practice. While doing this project, I gained a deeper understanding on various Database design concepts. One shortcoming of this project was that, as mentioned in Assumptions section, the member details cannot be deleted automatically but through a manual store procedure. I ran into few problems, when trying to create a nested stored procedure for creating new member and program plans. But the invaluable discussions made during the class sessions helped me in resolving the issues.

# References

ERD Templates. Available at:

<https://www.lucidchart.com/pages/>

For ERD diagram, Available at:

<https://app.diagrams.net/>

Microsoft SQL documentation. Available at:

<https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15>

Microsoft SQL Server Notes for Professionals. Available at:

<http://goalkicker.com/MicrosoftSQLServerBook/>

## 9. SQL

The SQL codes are written below as follows:

CREATE TABLE [Member] (

[MemberID] INT IDENTITY(100,1) NOT NULL,

[MemFirstName] VARCHAR(50) NOT NULL,

[MemSurName] VARCHAR(50) NOT NULL,

[AddressLine1] VARCHAR(100) NOT NULL,

[AddressLine2] VARCHAR(100),

[City] VARCHAR(20) NOT NULL,

[County] VARCHAR(20) NOT NULL,

[Postcode] VARCHAR(10) NOT NULL,

[MemEmailID] VARCHAR(50) NOT NULL,

[MemPhoneNum] VARCHAR(15) NOT NULL,

[GDPR] CHAR(1) NOT NULL,

[IsDeleted] CHAR(1) NOT NULL,

PRIMARY KEY ([MemberID])

);

CREATE TABLE [Trainer] (

[TrainerID] INT IDENTITY(2000,1) NOT NULL,

[FirstName] VARCHAR(50) NOT NULL,

[SurName] VARCHAR(50) NOT NULL,

[PPSNumber] CHAR(8) NOT NULL,

[PhoneNum] VARCHAR(15) NOT NULL,

[EmailID] VARCHAR(30) NOT NULL,

[GDPR] CHAR(1) NOT NULL,

[IsDeleted] CHAR(1) NOT NULL,

PRIMARY KEY ([TrainerID])

);

CREATE TABLE [MemberEnrollment] (

[EnrollmentID] INT IDENTITY(1000,1) NOT NULL,

[EnrollmentPlan] CHAR(10) NOT NULL,

[DateStarted] DATETIME NOT NULL,

[ExpiryDate] DATETIME NOT NULL,

[Duration] VARCHAR(20) NOT NULL,

[MemberID] INT NULL,

PRIMARY KEY ([EnrollmentID]),

FOREIGN KEY ([MemberID]) REFERENCES [Member] ([MemberID]) ON DELETE SET NULL

);

CREATE TABLE [Payment] (

[PaymentID] INT IDENTITY(3000,1) NOT NULL,

[MemberID] INT NULL,

[AmountPaid] FLOAT NOT NULL,

[TimePaid] DATETIME NOT NULL,

PRIMARY KEY ([PaymentID]),

FOREIGN KEY ([MemberID]) REFERENCES [Member] ([MemberID]) ON DELETE SET NULL

);

CREATE TABLE [Program] (

[ProgramID] INT IDENTITY(4000,1) NOT NULL,

[ProgramName] VARCHAR(30) NOT NULL,

[Duration] VARCHAR(25) NOT NULL,

[LastUpdated] DATETIME NOT NULL,

[ProgramType] VARCHAR(25) NOT NULL,

PRIMARY KEY ([ProgramID])

);

CREATE TABLE [DietPlan] (

[DietID] INT IDENTITY(5000,1) NOT NULL,

[DietName] VARCHAR(20) NOT NULL,

[Calories] FLOAT NOT NULL,

[Carbhohydrates] FLOAT NOT NULL,

[Fat] FLOAT NOT NULL,

[Protein] FLOAT NOT NULL,

PRIMARY KEY ([DietID])

);

CREATE TABLE [ProgramPlan] (

[PlanID] INT IDENTITY(6000,1) NOT NULL,

[MemberID] INT NULL,

[TrainerID] INT NULL,

[ProgramID] INT NULL,

[DietID] INT NULL,

[WorkoutDate] DATETIME NOT NULL,

[Description] VARCHAR(100) NOT NULL,

PRIMARY KEY ([PlanID]),

FOREIGN KEY ([MemberID]) REFERENCES [Member] ([MemberID]) ON DELETE SET NULL,

FOREIGN KEY ([TrainerID]) REFERENCES [Trainer] ([TrainerID]) ON DELETE SET NULL,

FOREIGN KEY ([ProgramID]) REFERENCES [Program] ([ProgramID]) ON DELETE SET NULL,

FOREIGN KEY ([DietID]) REFERENCES [DietPlan] ([DietID]) ON DELETE SET NULL

);

CREATE TABLE [MembershipCard] (

[UniqueMemNum] INT IDENTITY(7000,1) NOT NULL,

[MemberID] INT NULL,

[Name] VARCHAR(20) NOT NULL,

[Image] IMAGE NOT NULL,

PRIMARY KEY ([UniqueMemNum]),

FOREIGN KEY ([MemberID]) REFERENCES [Member] ([MemberID]) ON DELETE CASCADE

);

CREATE TABLE [Session] (

[SessionID] INT IDENTITY(8000,1) NOT NULL,

[UniqueMemNum] INT NULL,

[TimeIN] DATETIME NOT NULL,

[TimeOUT] DATETIME NOT NULL,

PRIMARY KEY ([SessionID]),

FOREIGN KEY ([UniqueMemNum]) REFERENCES [MembershipCard] ([UniqueMemNum]) ON DELETE SET NULL

);

CREATE TABLE [ProgressChart] (

[ProgressID] INT IDENTITY(9000,1) NOT NULL,

[MemberID] INT NULL,

[Date] DATETIME NOT NULL,

[BMI] FLOAT NOT NULL,

[FatPercentage] FLOAT NOT NULL,

[MuscleMass] FLOAT NOT NULL,

PRIMARY KEY ([ProgressID]),

FOREIGN KEY ([MemberID]) REFERENCES [Member] ([MemberID]) ON DELETE SET NULL

);

INSERT INTO DietPlan

VALUES ('KetoDiet','1600','20','80','130'),

('PaleoDiet','900','20','30','180'),

('VeganDiet','1800','80','30','130'),

('MuscleBuilding','2600','180','80','200'),

('DukanDiet','1800','50','30','180'),

('UltraLowFat','1600','120','20','180'),

('AtkinsDiet','2000','30','80','200'),

('HCGDiet','2000','30','30','80'),

('ZoneDiet','2000','80','60','60'),

('IntermittentFasting','2000','100','60','150')

INSERT INTO Program

VALUES ('CoreBurnout','60 Minutes','08/31/2020','Endurance'),

('BoxerCore','60 Minutes','08/31/2020','Endurance'),

('UpperBodyCoreBlend','75 Minutes','08/31/2020','Strength'),

('ABAndButt2.0','75 Minutes','09/30/2020','Endurance'),

('AtlasAmplified','90 Minutes','09/30/2020','Strength'),

('ShoulderShaper','90 Minutes','09/30/2020','Endurance'),

('UpperBodyExpress','90 Minutes','09/30/2020','Strength'),

('GlutesAndGlory','90 Minutes','10/31/2020','Endurance'),

('QuickHIITLowerBody','30 Minutes','10/31/2020','Endurance'),

('BacksideBurner','75 Minutes','10/31/2020','Endurance')

CREATE PROC MemberNew

@MemFirName VARCHAR(50),

@MemSurName VARCHAR(50),

@Add1 VARCHAR(100),

@Add2 VARCHAR(100),

@City VARCHAR(20),

@County VARCHAR(20),

@Postcode VARCHAR(10),

@MemEmail VARCHAR(50),

@MemPhone VARCHAR(15),

@GDPR CHAR(1),

@IsDeleted CHAR(1)

AS

BEGIN

DECLARE @IDENTITYVALUE INT

BEGIN TRAN

INSERT INTO Member(MemFirstName,MemSurName,AddressLine1,AddressLine2,City,County,Postcode,MemEmailID,MemPhoneNum,GDPR,IsDeleted)

VALUES (@MemFirName,@MemSurName,@Add1,@Add2,@City,@County,@Postcode,@MemEmail,@MemPhone,@GDPR,@IsDeleted)

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

ELSE

BEGIN

SET @IDENTITYVALUE = SCOPE\_IDENTITY()

COMMIT TRAN

END

END

CREATE PROC TrainersNew

@firstname VARCHAR(50),

@surname VARCHAR(50),

@ppsnumber CHAR(8),

@phonenum VARCHAR(15),

@emailid VARCHAR(30),

@GDPR CHAR(1),

@IsDeleted CHAR(1),

@Action CHAR(6)

AS

BEGIN

DECLARE @IDENTITYVALUE INT

BEGIN TRAN

INSERT INTO Trainer (FirstName,SurName,PPSNumber,PhoneNum,EmailID,GDPR,IsDeleted)

VALUES (@firstname,@surname,@ppsnumber,@phonenum,@emailid,@GDPR,@IsDeleted)

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

ELSE

BEGIN

SET @IDENTITYVALUE = SCOPE\_IDENTITY()

COMMIT TRAN

END

END

CREATE PROC ProgramUpdate

@programid INT,

@programname VARCHAR(30),

@duration VARCHAR(25),

@lastupdated DATETIME,

@programtype VARCHAR(25),

@Action CHAR(6)

AS

DECLARE @IDENTITYVALUE INT

IF @Action IS NULL

BEGIN

PRINT 'NOTHING TO DO'

RETURN -1

END

IF (@Action='UPDATE' AND @programid IS NOT NULL)

BEGIN

IF EXISTS(SELECT 1 FROM Program WHERE ProgramID=@programid)

BEGIN TRAN

UPDATE Program

SET ProgramName=@programname,

Duration=@duration,

LastUpdated=@lastupdated,

ProgramType=@programtype

WHERE ProgramID=@programid

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

END

IF (@Action='DELETE' AND @programid IS NOT NULL)

BEGIN

PRINT 'DELETING'

IF EXISTS(SELECT 1 FROM Program WHERE ProgramID=@programid)

BEGIN TRAN

DELETE Program

WHERE ProgramID=@programid

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

END

IF (@Action='INSERT')

BEGIN

BEGIN TRAN

INSERT INTO Program(ProgramName,Duration,LastUpdated,ProgramType)

VALUES(@programname,@duration,@lastupdated,@programtype)

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

END

COMMIT TRAN

CREATE PROC DeleteMember

@Memid INT,

@Action CHAR(6)

AS

DECLARE @IDENTITYVALUE INT

IF @Action IS NULL

BEGIN

PRINT 'NOTHING TO DO'

RETURN -1

END

IF (@Action='Delete' AND @Memid IS NOT NULL)

BEGIN

PRINT 'DELETING'

IF EXISTS(SELECT 1 FROM Member WHERE MemberID=@Memid)

BEGIN TRAN

DELETE Member

WHERE MemberID=@Memid

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

ELSE

COMMIT TRAN

END

CREATE VIEW VW\_ActiveMembers

AS

SELECT M.MemberID,(M.MemFirstName+' ' +M.MemSurName) AS FullName,M.MemEmailID,M.MemPhoneNum,T.TrainerID,(T.FirstName+' '+T.SurName) AS TrainerFullName,T.PhoneNum,T.EmailID,P.ProgramName,P.Duration,D.DietName,ProgramPlan.WorkoutDate,ProgramPlan.Description

FROM Member M

JOIN ProgramPlan

ON ProgramPlan.MemberID=M.MemberID

JOIN Trainer T

ON ProgramPlan.TrainerID=T.TrainerID

JOIN Program P

ON ProgramPlan.ProgramID=P.ProgramID

JOIN DietPlan D

ON ProgramPlan.DietID=D.DietID

WHERE M.IsDeleted='N';

CREATE PROC SoftDeleteMembers

@Memid INT

AS

DECLARE @counter INT

SELECT @counter=COUNT(\*) FROM Member

WHERE MemberID=@Memid

BEGIN TRAN

UPDATE Member

SET IsDeleted='Y'

WHERE MemberID=@Memid

IF @@ROWCOUNT <> @counter

ROLLBACK

ELSE

COMMIT

CREATE VIEW VW\_Membersdeleted

AS

SELECT \* FROM Member

WHERE MEMBER.IsDeleted='Y'

select\* from Program

exec ProgramUp

create proc updatepro

@progra int,

@duration varchar(25),

@action char(6)

as

declare @identityvalue int

if @action is null

begin

print 'Nothin'

end

if (@action='update' and @progra is not null)

BEGIN

IF EXISTS(SELECT 1 FROM Program WHERE ProgramID=@progra)

BEGIN TRAN

UPDATE Program

SET Duration=@duration

WHERE ProgramID=@progra

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

END

commit tran

EXEC MemberNew 'Manoj','Kumar','09 Brookfields','Tullyvaraga','shannon','Co.Clare','V14NR23','manoj@gmail.com','+353899728680','Y','N'

EXEC MemberNew 'Santhosh','Pandian','09 Brookfields','Tullyvaraga','shannon','Co.Clare','V14NR23','santhosh@gmail.com','+35389986809','Y','N'

EXEC MemberNew 'Krishna','Ravichandran','09 Brookfields','Tullyvaraga','shannon','Co.Clare','V14NR23','krishna@gmail.com','+353899728683','N','N'

EXEC MemberNew 'Viki','Muthusamy','168 Finian Park','Tullyvaraga','shannon','Co.Clare','V14AY68','viki@gmail.com','+353899727674','Y','N'

EXEC MemberNew 'Sarita','Mahanjan','168 Finian Park','Tullyvaraga','shannon','Co.Clare','V14AY68','sarita@gmail.com','+353899768680','N','N'

EXEC MemberNew 'Karthik','Pandian','24 Ballycasey court','Ballycasey','shannon','Co.Clare','V14XO87','karthik@gmail.com','+353890890756','N','N'

EXEC MemberNew 'Abner','Johnson','4 Ballycasey Manor','Ballycasey','shannon','Co.Clare','V14IK86','abner@gmail.com','+353890890786','Y','N'

EXEC MemberNew 'Aldous','Jio','6 Ballycasey Hill','Ballycasey','shannon','Co.Clare','V14RT02','aldous@gmail.com','+353890807685','Y','N'

EXEC MemberNew 'Viki','Mukundan','09 Cronan Lawn','Tullyvaraga','shannon','Co.Clare','V14PO98','vikimu@gmail.com','+353890890759','Y','N'

EXEC MemberNew 'Vijay','Pranav','56 Cluanin','Tullyvaraha','shannon','Co.Clare','V14KJ87','Vijay@gmail.com','+353890890076','Y','N'

EXEC TrainersNew 'Abu','Abdhul','KSA08097','+3536565789','abu@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Slava','Peter','KSA08096','+3536565785','slava@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Suga','Vanesh','KSA08095','+3536565689','suga@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Shawn','Mendes','KSA08094','+3536565788','shawn@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Richard','Duchon','KSA08093','+3536563456','richard@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Ruba','Ali','KSA08092','+35365654376','ruba@gmail.com','N','N','Insert'

EXEC TrainersNew 'Lauren','Findely','KSA08091','+353658734','Lauren@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Nochtli','Peralda','KSA08090','+35365876348','nichotli@gmail.com','N','N','Insert'

EXEC TrainersNew 'Morgan','Rose','KSA08089','+3536576767','morgan@gmail.com','Y','N','Insert'

EXEC TrainersNew 'Rachel','John','KSA08098','+3536568987','rachel@gmail.com','Y','N','Insert'

INSERT INTO Payment

VALUES (101,'180','08/09/2020'),

(102,'60','07/02/2020'),

(103,'100','06/01/2020'),

(104,'60','08/01/2020'),

(105,'100','08/05/2020'),

(106,'100','09/01/2020'),

(107,'180','09/01/2020'),

(108,'180','10/01/2020'),

(109,'180','10/01/2020'),

(110,'100','06/01/2020')

INSERT INTO MembershipCard

VALUES (101,'Manoj Kumar','D:\Data Analytics\Database and Business Applications\Picture\101.jpg'),

(102,'Santhosh Pandian','D:\Data Analytics\Database and Business Applications\Picture\102.jpg'),

(103,'Krishna Ravichandran','D:\Data Analytics\Database and Business Applications\Picture\103.jpg'),

(104,'Viki Muthusamy','D:\Data Analytics\Database and Business Applications\Picture\104.jpg'),

(105,'Sarita Mahajan','D:\Data Analytics\Database and Business Applications\Picture\105.jpg'),

(106,'Karthik Pandian','D:\Data Analytics\Database and Business Applications\Picture\106.jpg'),

(107,'Abner Johnson','D:\Data Analytics\Database and Business Applications\Picture\107.jpg'),

(108,'Aldous Jio','D:\Data Analytics\Database and Business Applications\Picture\108.jpg'),

(109,'Viki Mukundan','D:\Data Analytics\Database and Business Applications\Picture\109.jpg'),

(110,'Vijay Pranav','D:\Data Analytics\Database and Business Applications\Picture\110.jpg')

INSERT INTO MemberEnrollment

VALUES ('Yearly','08/09/2020','08/07/2021','12 Months',101),

('Quarterly','07/02/2020','11/01/2020','4 Months',102),

('HalfYearly','06/01/2020','12/31/2020','6 Months',103),

('Quarterly','08/01/2020','11/30/2020','4 Months',104),

('HalfYearly','08/05/2020','02/04/2021','6 Months',105),

('Halfyearly','09/01/2020','02/28/2020','6 Months',106),

('Yearly','09/01/2020','08/31/2021','12 Months',107),

('Yearly','06/01/2020','05/31/2021','12 Months',108),

('Yearly','10/01/2020','09/30/2021','12 Months',109),

('HalfYearly','06/01/2020','12/31/2020','6 Months',110)

select \* from MemberEnrollment

SELECT \* FROM Trainer

select \* from Program

select \* from DietPlan

INSERT INTO ProgramPlan

VALUES (101,2001,4000,5000,'09/09/2020','Fat Loss Program'),

(102,2001,4002,5003,'07/03/2020','Strength Program'),

(103,2002,4002,5003,'09/10/2020','Strength Program'),

(104,2005,4006,5003,'08/01/2020','Muscle building Program'),

(101,2002,4005,5000,'09/10/2020','Muscle Building Program'),

(102,2008,4007,5003,'07/05/2020','Strength Program'),

(105,2009,4006,5009,'08/06/2020','Strength Program'),

(105,2008,4005,5009,'08/10/2020','Strength Program'),

(106,2008,4005,5009,'09/02/2020','Strength Program'),

(106,2008,4006,5009,'09/05/2020','Strength Program')

SELECT \* FROM MembershipCard

INSERT INTO Session

VALUES (7000,'10/10/2020 09:08:55','10/10/2020 10:08:45')

INSERT INTO Session

VALUES (7001,'10/11/2020 10:08:45','10/11/2020 11:08:33'),

(7002,'10/11/2020 10:08:44','10/11/2020 12:07:29'),

(7003,'11/11/2020 11:07:34','11/11/2020 13:09:23'),

(7004,'12/11/2020 15:09:56','12/11/2020 16:09:23')

INSERT INTO ProgressChart

VALUES (101,'10/10/2020','23.5','13.6','28.5'),

(102,'10/10/2020','30.6','35.5','33.5'),

(103,'10/10/2020','23.5','13.6','28.5'),

(101,'10/10/2020','23','17.6','26.5'),

(104,'10/10/2020','22','18.6','29.5'),

(105,'10/10/2020','24.5','17.6','31.5'),

(106,'10/10/2020','21.5','14.6','32.5'),

(107,'10/10/2020','26.5','20.6','34.5'),

(108,'10/10/2020','24.5','13.6','27.5'),

(109,'10/10/2020','28.5','14.6','26.5'),

(110,'10/10/2020','21.5','17.6','24.5')

CREATE PROC ProgramPlanDesign

@memid INT,

@trainerid INT,

@programid INT,

@dietid INT,

@workoutdate DATETIME,

@des VARCHAR(100)

AS

BEGIN

DECLARE @IDENTITYVALUE INT

BEGIN TRAN

INSERT INTO ProgramPlan (MemberID,TrainerID,ProgramID,DietID,WorkoutDate,Description)

VALUES (@memid,@trainerid,@programid,@dietid,@workoutdate,@des)

IF @@ERROR <> 0 OR @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WENT WRONG'

ROLLBACK TRAN

END

ELSE

BEGIN

SET @IDENTITYVALUE = SCOPE\_IDENTITY()

COMMIT TRAN

END

END

ALTER PROC NewMembersandplans

@MemFirName VARCHAR(50),

@MemSurName VARCHAR(50),

@Add1 VARCHAR(100),

@Add2 VARCHAR(100),

@City VARCHAR(20),

@County VARCHAR(20),

@Postcode VARCHAR(10),

@MemEmail VARCHAR(50),

@MemPhone VARCHAR(15),

@GDPR CHAR(1),

@IsDeleted CHAR(1),

@memid INT,

@trainerid INT,

@programid INT,

@dietid INT,

@workoutdate DATETIME,

@des VARCHAR(100),

@action CHAR(7)

AS

DECLARE @IDENTITY INT

IF @action='Insert'

BEGIN

INSERT INTO Member(MemFirstName,MemSurName,AddressLine1,AddressLine2,City,County,Postcode,MemEmailID,MemPhoneNum,GDPR,IsDeleted)

VALUES (@MemFirName,@MemSurName,@Add1,@Add2,@City,@County,@Postcode,@MemEmail,@MemPhone,@GDPR,@IsDeleted)

SET @IDENTITY = SCOPE\_IDENTITY()

IF @@ERROR = 0 AND @@ROWCOUNT = 1

BEGIN

EXEC ProgramPlanDesign @IDENTITY,@trainerid,@programid,@dietid,@workoutdate,@des

IF @@ERROR <> 0 AND @@ROWCOUNT <> 1

BEGIN

PRINT 'SOMETHING WRONG'

RETURN -1

END

END

End