CSE 3055 DATABASE SYSTEMS PROJECT

GYMMASTER

Student 1: Mert Kelkit – 150115013

Student 2: Furkan Nakıp – 150115032

About The Project

In this project, we are planning to design a database system that can be managed with a Spring web application for a gym center. Our aim is to make easier to manage gym center's payments, inventory, training sections, memberships. There are members, administrators, staffs in the gym and each of them have different types of memberships, staff jobs.

Also each administrator, staff and member will be user in the system with different privilleges. For instance members can view training sections, view his/her body measurements and choose a training section to attend; staffs can enter a new body measurement, enter a training section to an available time that will be instructed by him/her; administrators can view/manage membership types of members, check their payments and check contacts with equipment suppliers. Each of them can login to the system with their user accounts and can complete their jobs.

Data & Requirement Analysis

The database will contain tables of membership types, staff types, user types, members, admins, staffs, users, body measurements of members, weekly training sections, log of training attendances, meber payments, suppliers and equipments.

Membership types are identified by membership type id, description, total charge and total day count that member can join gym. Usually used for checking payments of members. There are 4 membership types in this gym:

1- Trial Membership: 30 days, total charge is 0 ₺. Can be adopted by a member just one time.
2- Monthly Membership: 30 days, total charge is 100 ₺.
3- Six Months Membership: 180 days, total charge is 400 ₺.
4- Yearly Membership: 365 days, total charge is 700 ₺.

Staff types are identified by staff type id and description. These table used for checking a staff can open a training section or not. Descriptions specifies profession of staffs. There are 6 staff types in this gym:

1- Yoga Staff
2- Pilates Staff
3- Cardio Staff
4- Dietician
5- Body Building (Crossfit) Staff
6- Fitness Staff

User types are identified by user type id and descriptions. As we said each user would have different privilleges, here these user types are used to determine a user's privilleges by the system. There are 3 types of users:

1- Member User
2- Staff User
3- Admin User

Members are identified by their member id, member name – surname, mail address, address, phone number, age, gender, starting date of membership and the type of member's membership. Each member must have a membership type. One membership type can be adopted by zero or many members.

Staffs are identified by their staff id, staff name – surname, mail address, address, phone number, age, type of staff and salary of staff. Each staff must have a staff type. One staff type can be adopted by many staffs.

Admins are identified by their admin id, admin name – surname, mail address, address, phone number, age and salary.

Users are identified by their user id, username, password and user type. Each user must have a user type. One user type can be adopted by many users. We said that each admin, member and staff will have a user account in the system. Primary key of this table (user_id) will be equal to admin id if user type is admin, staff id if user type is staff, member id if user type is member. That means tables User and Member/Admin/Staff have one-to-one relationship. They have exactly same values of primary keys. This table will be used mostly with the web application.

Member payments are identified by unique payment id, member id, payment amount and payment date. Each payment must be made by a member, a member can make many payments. Payment amount will be determined by member's membership type.

Members' body measurements are identified by unique measurement id, - foreign key to member table - member id, height as meters, weight as kilograms, BMI as computed column (weight / height^2), fat rate as percentage, measurement date and some optional measurements like arm width, leg width, hip width. One measurement must be made with a member, a member can have zero or many measurements. This table mainly used for improvement of a member.

Equipments are identified by unique equipment id, equipment name, supplier id – foreign key to supplier table –, amount of an equipment, purchase date, expiration date (maintenance date) and price. Each unique equipment must be bought from a supplier. A supplier can supply many equipments.

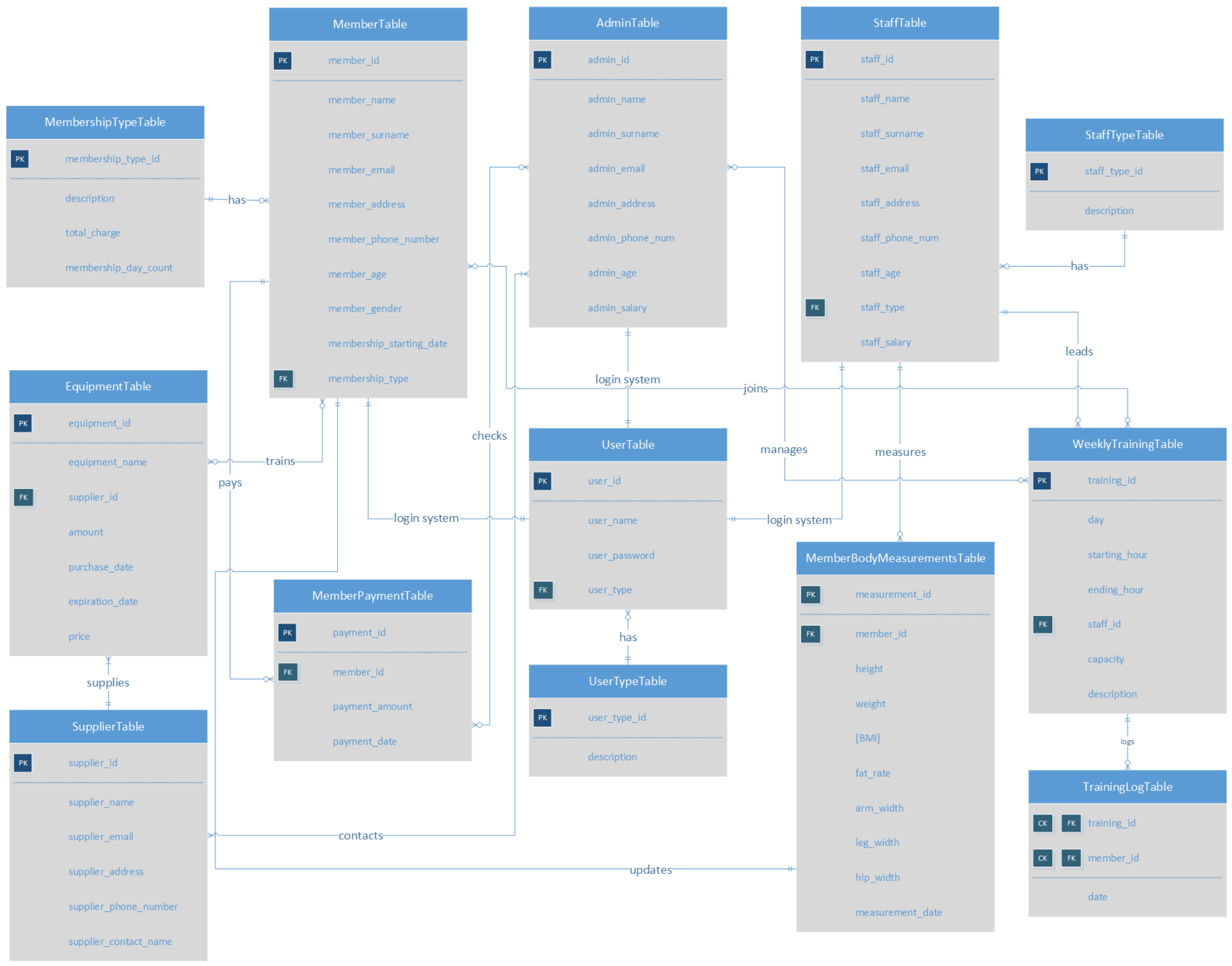
Suppliers are identified by unique supplier id, supplier name, mail address, address, phone number and contact name of supplier company.

Weekly trainings are identified by unique training id, day, starting and ending hours, staff id who instructs the training section, capacity of section and description for section e.g. "Leg training with leg press machine and squat". A weekly training must be opened by a staff. A staff may open many training sections in a week.

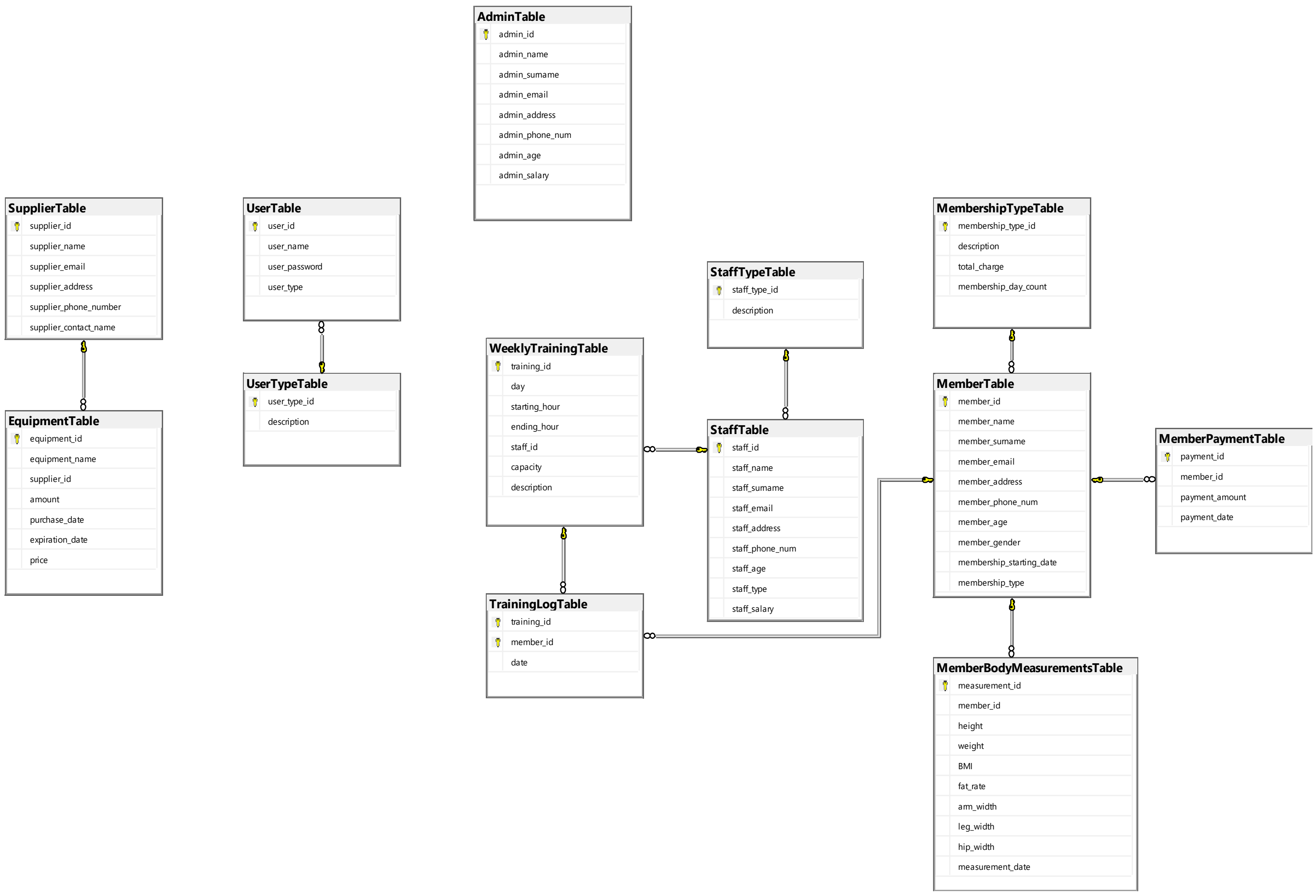
Training logs are identified by a composite key which composed of training id and member id, training date. This table has foreign key relationship with Member Table and Weekly Training Table. This table will store information about attenders of each training section.

Fictitious data is used in that database. Equipments, their prices and suppliers gathered from internet. Some of member and staff names are collected from an api service, some of them are real data given by our familiars.

E-R Diagram


MembershipTypeTable
PK membership_type_id
description
total_charge
membership_day_count
MemberTable
PK member_id
member_name
member_surname
member_email
member_address
member_phone_number
member_age
member_gender
membership_starting_date
FK membership_type
AdminTable
PK admin_id
admin_name
admin_surname
admin_email
admin_address
admin_phone_num
admin_age
admin_salary
StaffTable
PK staff_id
staff_name
staff_surname
staff_email
staff_address
staff_phone_num
staff_age
FK staff_type
staff_salary
StaffTypeTable
PK staff_type_id
description
EquipmentTable
PK equipment_id
equipment_name
FK supplier_id
amount
purchase_date
expiration_date
price
UserTable
PK user_id
user_name
user_password
FK user_type
UserTypeTable
PK user_type_id
description
WeeklyTrainingTable
PK training_id
day
starting_hour
ending_hour
FK staff_id
capacity
description
MemberBodyMeasurementsTable
PK measurement_id
FK member_id
height
weight
[BMI]
fat_rate
arm_width
leg_width
hip_width
measurement_date
MemberPaymentTable
PK payment_id
FK member_id
payment_amount
payment_date
SupplierTable
PK supplier_id
supplier_name
supplier_email
supplier_address
supplier_phone_number
supplier_contact_name
TrainingLogTable
CK FK training_id
CK FK member_id
date
has
has
has
trains
pays
checks
login system
login system
login system
joins
manages
measures
leads
logs
supplies
contacts
updates

Database Diagram



Information About Tables and Attributes

- AdminTable: Contains records of admins of the system

AdminTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
admin_id	INT	✓		Between 200000 - 300000
admin_name	VARCHAR(32)			
admin_surname	VARCHAR(32)			
admin_email	VARCHAR(50)			
admin_address	VARCHAR(128)			
admin_phone_num	VARCHAR(20)			
admin_age	SMALLINT			
admin_salary	MONEY			Unit assumed as ₺

- EquipmentTable: Containts records of equipments in the gym

EquipmentTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
equipment_id	INT	✓		Starts with 1, incremented by 1 each insertion
equipment_name	VARCHAR(128)			
supplier_id	INT		✓ *(SupplierTable)*	Shows equipment's supplier
amount	SMALLINT			
purchase_date	DATE			
expiration_date	DATE			Also shows equipment's maintenance date
price	MONEY			Unit assumed as ₺

- MemberPaymentTable: Contains information about members’ payments

MemberPaymentTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
payment_id	INT	✓		Starts with 1, incremented by 1 each insertion
member_id	INT		✓ *(MemberTable)*	Shows who made the payment
payment_amount	MONEY			Units assumed as ₺
payment_date	DATE			

- MembershipTypeTable: Contains types of memberships

MembershipTypeTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
membership_type_id	TINYINT	✓		
description	VARCHAR(128)			
total_charge	MONEY			Units assumed as ₺
membership_day_count	SMALLINT			

- MemberTable: Contains records of gym members

MemberTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
member_id	INT	✓		Between 100000-200000
member_name	VARCHAR(32)			
member_surname	VARCHAR(32)			
member_email	VARCHAR(50)			
member_address	VARCHAR(128)			
member_phone_num	VARCHAR(20)			
member_age	SMALLINT			
member_gender	VARCHAR(10)			
membership_starting_date	DATE			
membership_type	TINYINT		✓ *(MembershipTypeTable)*	

- StaffTable: Contains records of staffs working in the gym

StaffTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
staff_id	INT	✓		Between 300000-400000
staff_name	VARCHAR(32)			
staff_surname	VARCHAR(32)			
staff_email	VARCHAR(50)			
staff_address	VARCHAR(128)			
staff_phone_num	VARCHAR(20)			
staff_age	SMALLINT			
staff_type	TINYINT		✓ *(StaffTypeTable)*	Type id value between 1-6
staff_salary	MONEY			Units assumed as ₺

- MemberBodyMeasurementsTable: Contains records of members' body measurements

MemberBodyMeasurementsTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
measurement_id	INT	✓		Starts with 1, incremented by 1 each insertion
member_id	INT		✓ *(MemberTable)*	Shows whose body measurements
height	DECIMAL(8, 2)			Unit assumed as meters
weight	DECIMAL(8, 2)			Unit assumed as kilograms
BMI	DECIMAL			Computed column as weight / height^2
fat_rate	DECIMAL(8, 2)			Percentage, optional
arm_width	DECIMAL(8, 2)			Unit assumed as cm, optional
leg_width	DECIMAL(8, 2)			Unit assumed as cm, optional
hip_width	DECIMAL(8, 2)			Unit assumed as cm, optional
measurement_date	DATE			Default is GETDATE() -system date-

- StaffTypeTable: Contains type of staffs working in the gym

StaffTypeTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
staff_type_id	TINYINT	✓		
description	VARCHAR(128)			Describes staff's profession area

- SupplierTable: Contains records of equipment supplier companies to the gym

SupplierTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
supplier_id	INT	✓		Starts with 1 incremented by 1 each insertion
supplier_name	VARCHAR(64)			
supplier_email	VARCHAR(50)			
supplier_address	VARCHAR(128)			
supplier_phone_number	VARCHAR(20)			
supplier_contact_name	VARCHAR(64)			Responsible person for gym from company

- TrainingLogTable: Contains records of training section attendance

TrainingLogTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
training_id	INT	✓	✓ *(WeeklyTrainingTable)*	Composite Key together
member_id	INT	✓	✓ *(MemberTable)*	
date	DATE			Default value GETDATE() - system date-

- UserTable: Contains records of users (Members, staffs, admins with same primary keys)

UserTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
user_id	INT	✓		Same values with member_id, staff_id, admin_id in other tables
user_name	VARCHAR(32)			
user_password	VARCHAR(40)			
user_type	TINYINT		✓ *(UserTypeTable)*	A value between 1-3.

- UserTypeTable: Contains types of users

UserTypeTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
user_type_id	TINYINT	✓		1 for members, 2 for staffs, 3 for admins
description	VARCHAR(128)			

- WeeklyTrainingTable: Contains available training section records

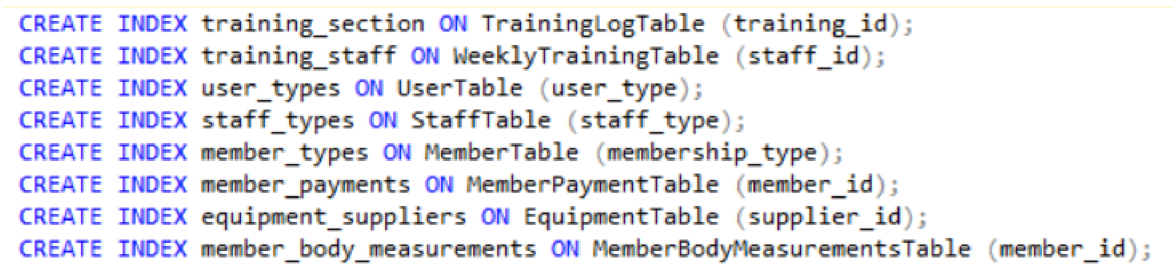
WeeklyTrainingTable				
Attribute	**Data Type**	**Primary Key**	**Foreign Key**	**Notes**
training_id	INT	✓		Starts with 1, incremented by 1 each insertion
day	VARCHAR(16)			
starting_hour	VARCHAR(8)			
ending_hour	VARCHAR(8)			
staff_id	INT		✓ *(StaffTable)*	Instructor of the training section
capacity	INT			
description	VARCHAR(128)			

** All DATE typed attributes are formatted as 'yyyy-mm-dd'*

Indices, Constraints, Defaults & Computed Columns

1. Indices

- We have primary keys for each table, so that means we directly have one clustered index for each table.
- Except for them, we created indices as shown below:



```
CREATE INDEX training_section ON TrainingLogTable (training_id);
CREATE INDEX training_staff ON WeeklyTrainingTable (staff_id);
CREATE INDEX user_types ON UserTable (user_type);
CREATE INDEX staff_types ON StaffTable (staff_type);
CREATE INDEX member_types ON MemberTable (membership_type);
CREATE INDEX member_payments ON MemberPaymentTable (member_id);
CREATE INDEX equipment_suppliers ON EquipmentTable (supplier_id);
CREATE INDEX member_body_measurements ON MemberBodyMeasurementsTable (member_id);
```

Most of these indices created by foreign key columns of tables for query optimization, these will lead us faster grouping queries.

2. Constraints

- We have foreign key constraints for all tables that contain foreign keys. All of these foreign key constraints contain has the attribute 'ON DELETE CASCADE'. For instance if a member has removed from the MemberTable, his/her body measurement will be removed from the MemberBodyMeasurementsTable.
- Except for foreign key constraints, we have a composite key constraint which is for TrainingLogTable. Both member_id from MemberTable and training_id from WeeklyTrainingTable considered as a composite key.
- We have check constraints for member_id from MemberTable, staff_id from StaffTable and admin_id from AdminTable. member_id must be between 100,000 and 200,000; staff_id must be between 300,000 and 400,000; admin_id must be between 200,000 and 300,000.
- We also have check constraints for membership_type from MemberTable, staff_type from StaffTable and user_type from UserTable. membership_type must be between 1 and 4, because we have only 4 types of memberships. staff_type must be between 1 and 6 because there is only 6 staff types. user_type must be between 1 and 3 because there is only 3 types of users (admin user, staff user, member user).
- In addition we have check constraints for phone numbers for AdminTable, StaffTable and MemberTable. A phone number must be longer or equal to 12 characters e.g. 539 501 5317 minimum length = 12. We used the formula (DATALENGTH(x_phone_num) >= 12) where x is admin, staff or member. We also record genders of each member. We assumed that given gender must be 'Male' or 'Female' with our check constraint.

3. Defaults

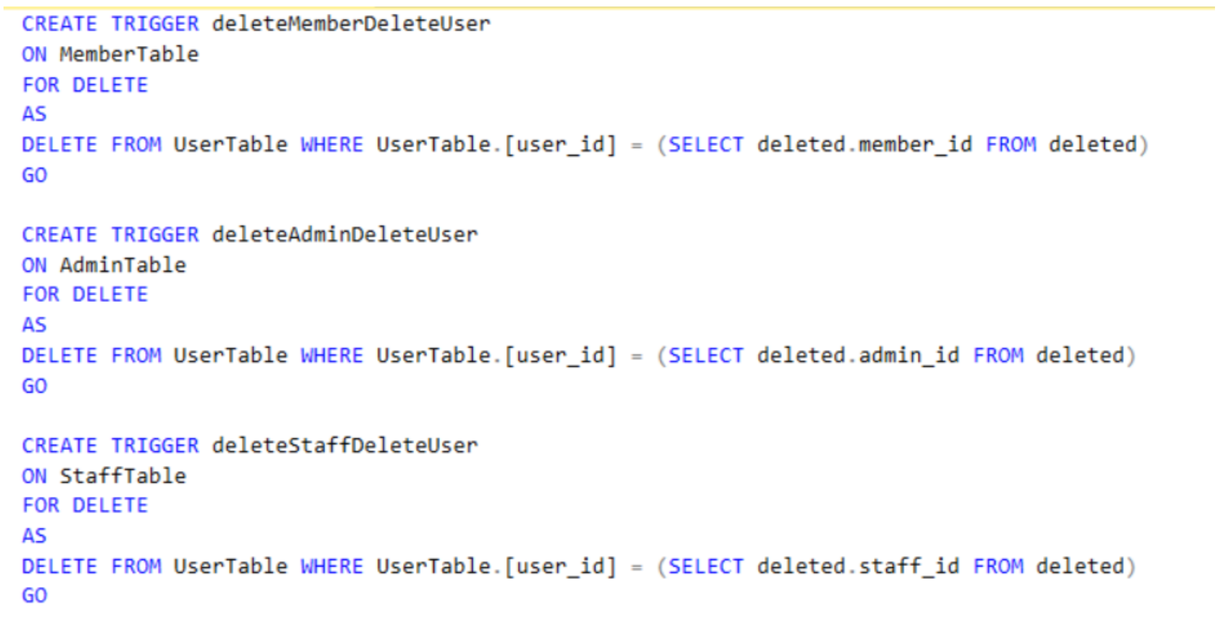
- We have default values for measurement_date from MemberBodyMeasurementsTable and date from TrainingLogTable. Both of them have GETDATE() (system date) value if values are not specified.

4. Computed Columns

- We auto computed BMI column of MemberBodyMeasurementsTable. This statistic computed as weight / (height * height). This value can be used for determining a member is overweighted, underweighted or normal weighted.

Triggers

- Delete Member/Admin/Staff -> Delete User triggers



```
CREATE TRIGGER deleteMemberDeleteUser
ON MemberTable
FOR DELETE
AS
DELETE FROM UserTable WHERE UserTable.[user_id] = (SELECT deleted.member_id FROM deleted)
GO

CREATE TRIGGER deleteAdminDeleteUser
ON AdminTable
FOR DELETE
AS
DELETE FROM UserTable WHERE UserTable.[user_id] = (SELECT deleted.admin_id FROM deleted)
GO

CREATE TRIGGER deleteStaffDeleteUser
ON StaffTable
FOR DELETE
AS
DELETE FROM UserTable WHERE UserTable.[user_id] = (SELECT deleted.staff_id FROM deleted)
GO
```

These triggers will do the job of removal of user accounts if an admin or a staff leaves the job or a member cancels his/her membership.

- Insert Member/Admin/Staff -> Insert User triggers

```sql
CREATE TRIGGER createUserOnMemberInsert ON MemberTable
FOR INSERT
AS

DECLARE @randomPassword varchar(12);
DECLARE @uname varchar(32);
DECLARE @randomNo int;

SET @randomPassword = CONVERT(varchar(256), NEWID());

SELECT @uname = (SELECT CONCAT(LOWER(inserted.member_name), LOWER(inserted.member_surname)) FROM inserted);

WHILE EXISTS(SELECT * FROM UserTable WHERE [user_name] = @uname)
BEGIN
    SET @randomNo = ABS(CHECKSUM(NewId())) % 99;
    SET @uname = CONCAT(@uname, @randomNo);
END

INSERT INTO UserTable
        ([user_id], [user_name], user_password, user_type)
    SELECT
        inserted.member_id, @uname, @randomPassword, 1
    FROM inserted;
GO
```

```sql
CREATE TRIGGER createUserOnStaffInsert ON StaffTable
FOR INSERT
AS

DECLARE @randomPassword varchar(12);
DECLARE @uname varchar(32);
DECLARE @randomNo int;

SET @randomPassword = CONVERT(varchar(256), NEWID());

SELECT @uname = (SELECT CONCAT(LOWER(inserted.staff_name), LOWER(inserted.staff_surname)) FROM inserted);

WHILE EXISTS(SELECT * FROM UserTable WHERE [user_name] = @uname)
BEGIN
    SET @randomNo = ABS(CHECKSUM(NewId())) % 99;
    SET @uname = CONCAT(@uname, @randomNo);
END

INSERT INTO UserTable
        ([user_id], [user_name], user_password, user_type)
    SELECT
        inserted.staff_id, @uname, @randomPassword, 2
    FROM inserted;
GO
```

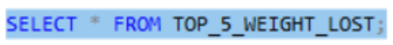
```
CREATE TRIGGER createUserOnAdminInsert ON AdminTable
FOR INSERT
AS

DECLARE @randomPassword varchar(12);
DECLARE @uname varchar(32);
DECLARE @randomNo int;

SET @randomPassword = CONVERT(varchar(256), NEWID());

SELECT @uname = (SELECT CONCAT(LOWER(inserted.admin_name), LOWER(inserted.admin_surname)) FROM inserted);

WHILE EXISTS(SELECT * FROM UserTable WHERE [user_name] = @uname)
BEGIN
    SET @randomNo = ABS(CHECKSUM(NewId())) % 99;
    SET @uname = CONCAT(@uname, @randomNo);
END

INSERT INTO UserTable
        ([user_id], [user_name], user_password, user_type)
    SELECT
        inserted.admin_id, @uname, @randomPassword, 1
    FROM inserted;
GO
```

These three triggers have the same logic. Whenever a member/admin/staff recorded to the system, thay will have automatically have user accounts belonging to them. These triggers concatenates member/admin/staff's names and surnames – appends some random number if this username exists – in order to make them their default user names. After creating user name, these triggers also generate random password with length of 12; contains uppercase letters, dashes and numbers. After these operations, user accounts are inserted with appropriate user types (1 for members, 2 for staffs, 3 for admins).

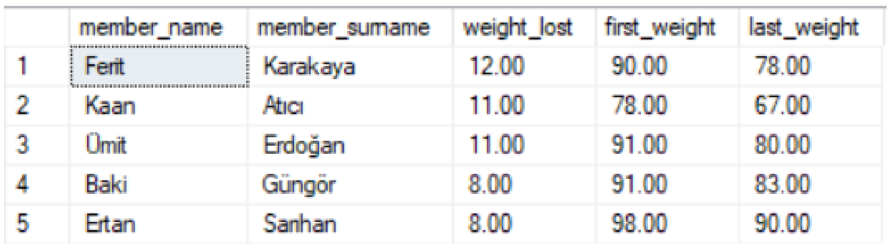
Views

- Top 5 Weight Loss View

```
CREATE VIEW TOP_5_WEIGHT_LOST
AS
SELECT TOP (5) MT.member_name, MT.member_surname, MBT1.[weight] - MBT2.[weight] AS weight_lost,
                                    MBT1.[weight] AS first_weight, MBT2.[weight] AS last_weight
FROM MemberBodyMeasurementsTable MBT1, MemberBodyMeasurementsTable MBT2, MemberTable MT
WHERE MBT1.member_id IN
    (SELECT member_id FROM MemberBodyMeasurementsTable GROUP BY(member_id) HAVING COUNT(*) > 1)
AND
MBT2.member_id IN
    (SELECT member_id FROM MemberBodyMeasurementsTable GROUP BY(member_id) HAVING COUNT(*) > 1)
AND
MBT1.measurement_date = (SELECT top 1 t2.measurement_date FROM MemberBodyMeasurementsTable t2
                            WHERE MBT1.member_id = member_id ORDER BY measurement_date)
AND
MBT2.measurement_date = (SELECT top 1 t2.measurement_date FROM MemberBodyMeasurementsTable t2
                            WHERE MBT1.member_id = member_id ORDER BY measurement_date DESC)
AND
MBT1.member_id = MBT2.member_id
AND
MBT1.member_id = MT.member_id
ORDER BY weight_lost DESC;
```



```sql
SELECT * FROM TOP_5_WEIGHT_LOST;
```

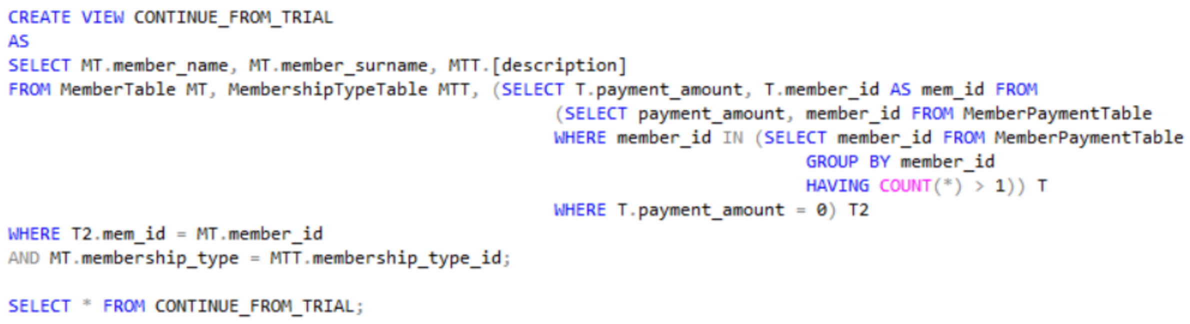


	member_name	member_surname	weight_lost	first_weight	last_weight
1	Ferit	Karakaya	12.00	90.00	78.00
2	Kaan	Atıcı	11.00	78.00	67.00
3	Ümit	Erdoğan	11.00	91.00	80.00
4	Baki	Güngör	8.00	91.00	83.00
5	Ertan	Sarıhan	8.00	98.00	90.00

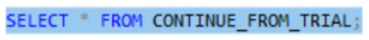
This view is for determining top 5 person that had lost weight.

First we get members that has more than one measurements in the table, in order to see the change on his/her weight. After that, we get each of these members' measurement dates in order to determine weight for the first measurement and weight for the last measurement. After that we subtract first weight from last weight determined by measurement dates in order to see the weight loss. Then sort by weight loss and get top 5 records of the view.

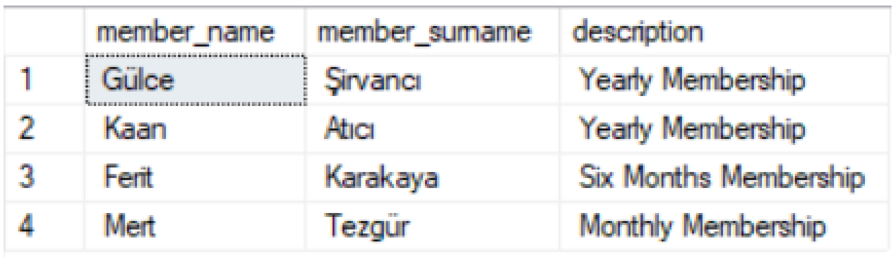
- Members That Continued Membership Starting with Trial Membership View



```sql
CREATE VIEW CONTINUE_FROM_TRIAL
AS
SELECT MT.member_name, MT.member_surname, MTT.[description]
FROM MemberTable MT, MembershipTypeTable MTT, (SELECT T.payment_amount, T.member_id AS mem_id FROM
                                              (SELECT payment_amount, member_id FROM MemberPaymentTable
                                              WHERE member_id IN (SELECT member_id FROM MemberPaymentTable
                                                                 GROUP BY member_id
                                                                 HAVING COUNT(*) > 1)) T
                                              WHERE T.payment_amount = 0) T2
WHERE T2.mem_id = MT.member_id
AND MT.membership_type = MTT.membership_type_id;

SELECT * FROM CONTINUE_FROM_TRIAL;
```



```sql
SELECT * FROM CONTINUE_FROM_TRIAL;
```



	member_name	member_surname	description
1	Gülce	Şirvancı	Yearly Membership
2	Kaan	Atıcı	Yearly Membership
3	Ferit	Karakaya	Six Months Membership
4	Mert	Tezgür	Monthly Membership

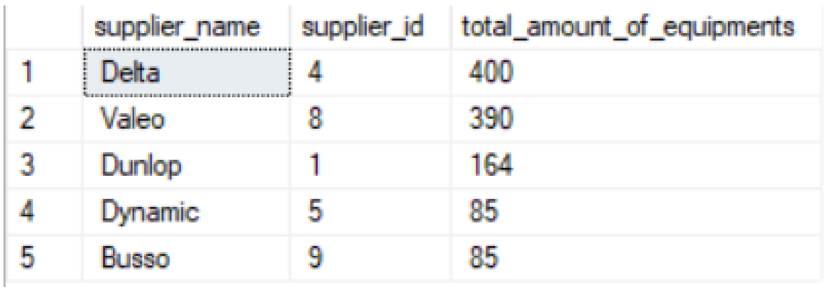
As we can see there are 4 members that continues to membership after a trial month.

In this view, first we get members that made payment more than one time, we also recorded trial memberships with a charge of 0 ₺. After that we get members that paid 0 ₺ one time and made another payment for a different membership type. From all these queries we get members that had trial membership once, after that paid for a different membership type.

- Top 5 Supplier Companies View

```
CREATE VIEW TOP_5_SUPPLIERS
AS
SELECT ST.supplier_name, ST.supplier_id, T.total_amount AS total_amount_of_equipments FROM SupplierTable ST,
(SELECT TOP 5 SUM(amount) AS total_amount, supplier_id
FROM EquipmentTable
GROUP BY supplier_id
ORDER BY total_amount DESC) T
WHERE ST.supplier_id IN (
    SELECT TOP 5 supplier_id
    FROM EquipmentTable
    GROUP BY supplier_id
    ORDER BY SUM(amount) DESC
    )
AND
ST.supplier_id = T.supplier_id;
```

```
SELECT * FROM TOP_5_SUPPLIERS;
```



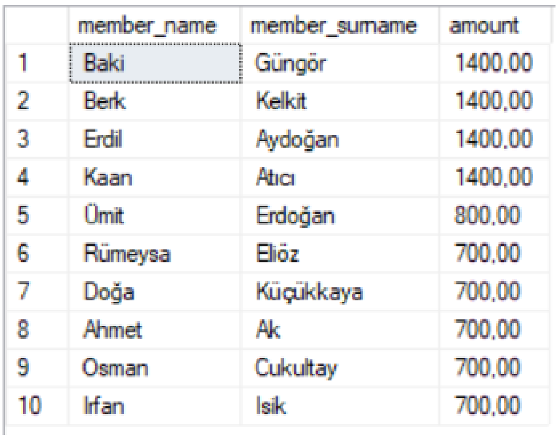
	supplier_name	supplier_id	total_amount_of_equipments
1	Delta	4	400
2	Valeo	8	390
3	Dunlop	1	164
4	Dynamic	5	85
5	Busso	9	85

We record equipments with their amounts e.g. equipment name is 5kg dumbbell, its amount is 20 and let's say we bought it from Valeo company. It means that we bought 20 equipments from Valeo. With this view, we can see top 5 supplier that supplies the most equipments. First we get top 5 supplier id's from the equipment table – with sorting by sum of amounts –, after that we get top 5 sums of equipments, grouped them by supplier id. After joining them, we get top 5 suppliers of our gym.

- Top 10 Payments View

```
CREATE VIEW TOP_10_MOST_PAYMENTS
AS
SELECT member_name, member_surname, T.amount
FROM MemberTable, (SELECT TOP 20 member_id, SUM(payment_amount) AS amount
                    FROM MemberPaymentTable
                    GROUP BY member_id
                    ORDER BY amount DESC) T
WHERE MemberTable.member_id = T.member_id;
```

```
SELECT * FROM TOP_10_MOST_PAYMENTS;
```

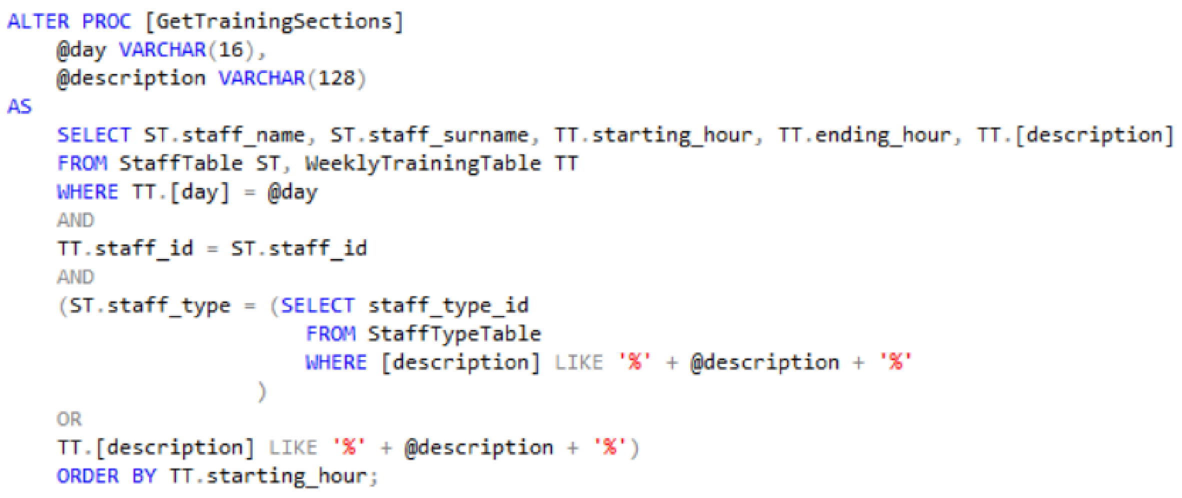


	member_name	member_surname	amount
1	Baki	Güngör	1400,00
2	Berk	Kelkit	1400,00
3	Erdil	Aydoğan	1400,00
4	Kaan	Atıcı	1400,00
5	Ümit	Erdoğan	800,00
6	Rümeysa	Eliöz	700,00
7	Doğa	Küçükkaya	700,00
8	Ahmet	Ak	700,00
9	Osman	Cukultay	700,00
10	Irfan	Isik	700,00

As we know, a member may have more than one payments to the gym. We calculated the sum of payments made to the gym member by member. After that, we sorted by total payment amount, select top 10 of it.

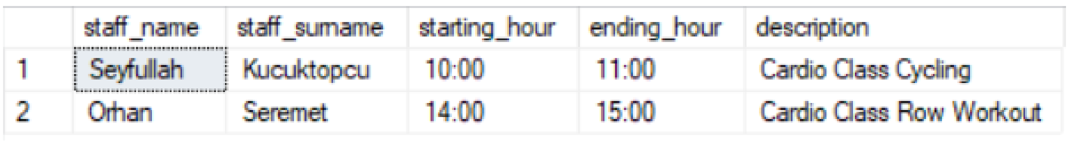
Procedures

- GetTrainingSections Procedure



```
ALTER PROC [GetTrainingSections]
    @day VARCHAR(16),
    @description VARCHAR(128)
AS
    SELECT ST.staff_name, ST.staff_surname, TT.starting_hour, TT.ending_hour, TT.[description]
    FROM StaffTable ST, WeeklyTrainingTable TT
    WHERE TT.[day] = @day
    AND
    TT.staff_id = ST.staff_id
    AND
    (ST.staff_type = (SELECT staff_type_id
                      FROM StaffTypeTable
                      WHERE [description] LIKE '%' + @description + '%'
                     )
    OR
    TT.[description] LIKE '%' + @description + '%')
    ORDER BY TT.starting_hour;
```

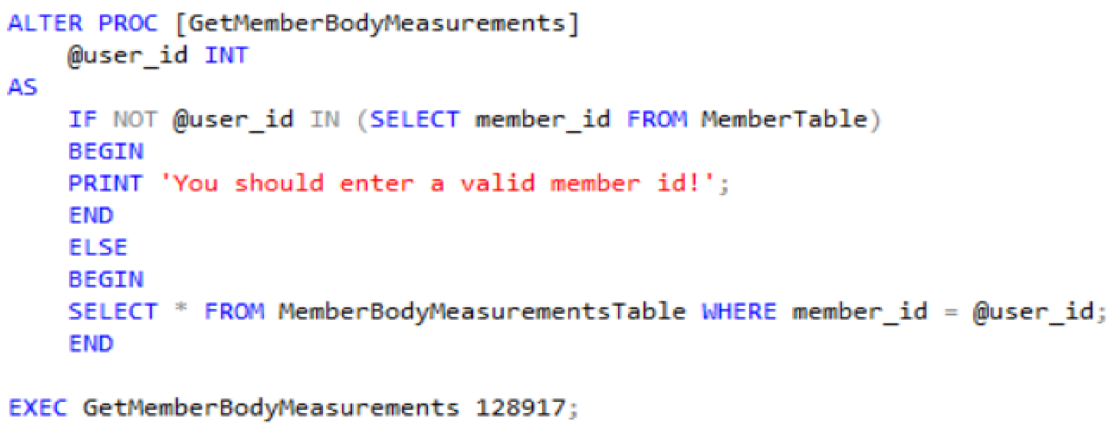
```
EXEC GetTrainingSections 'Tuesday', 'Cardio';
```



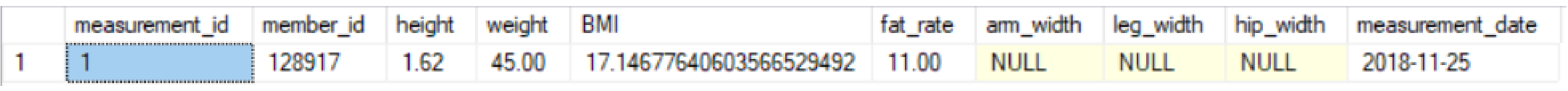
	staff_name	staff_surname	starting_hour	ending_hour	description
1	Seyfullah	Kucuktopcu	10:00	11:00	Cardio Class Cycling
2	Orhan	Seremet	14:00	15:00	Cardio Class Row Workout

On this procedure, we have 2 input parameters which are day and description. Day specifies the day of the training section, description specifies training section's focus. Procedure looks for description paramter on staff's description or training section's description. If there will be match, matched trainings will be listed.

- GetMemberBodyMeasurementsProcedure



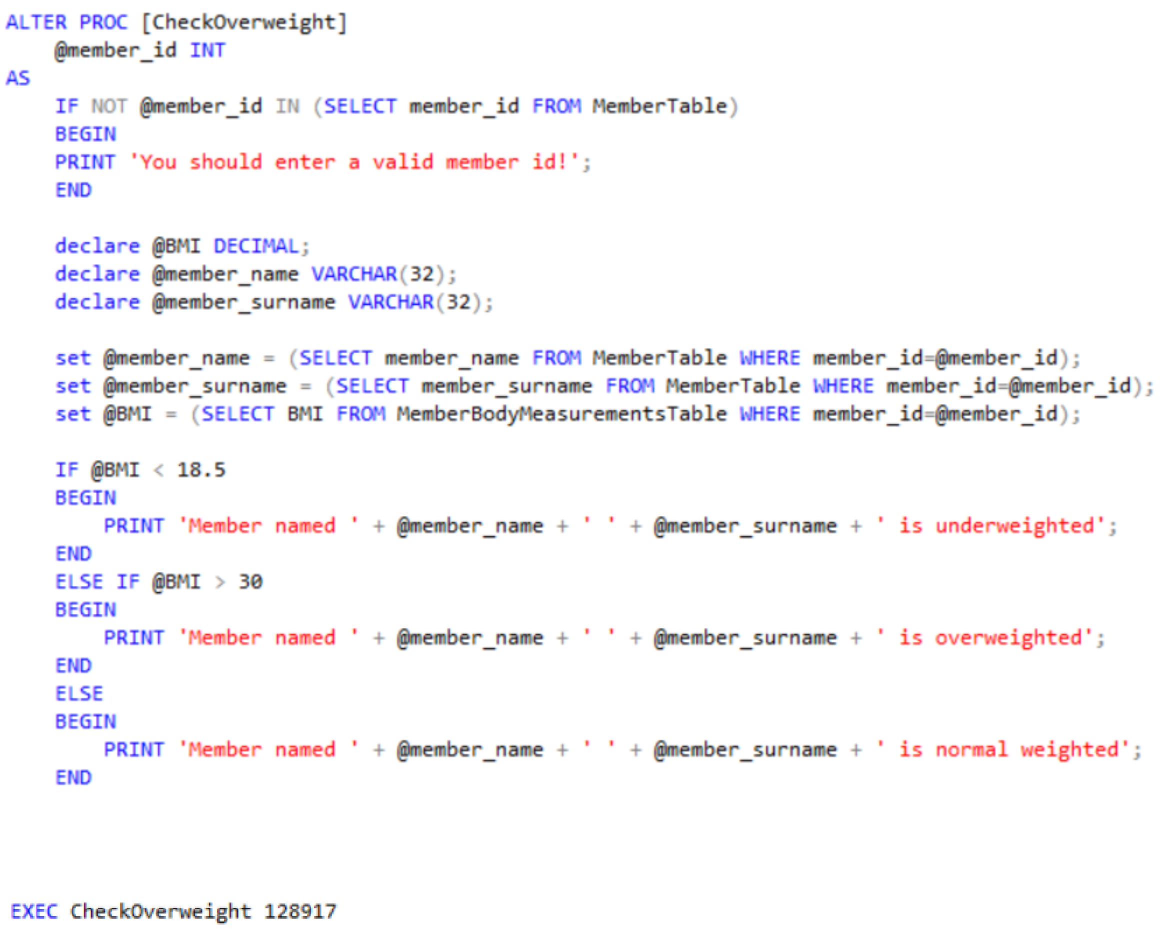
```sql
ALTER PROC [GetMemberBodyMeasurements]
    @user_id INT
AS
    IF NOT @user_id IN (SELECT member_id FROM MemberTable)
    BEGIN
    PRINT 'You should enter a valid member id!';
    END
    ELSE
    BEGIN
    SELECT * FROM MemberBodyMeasurementsTable WHERE member_id = @user_id;
    END

EXEC GetMemberBodyMeasurements 128917;
```



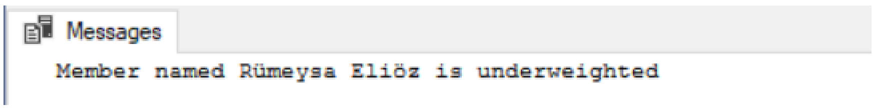
	measurement_id	member_id	height	weight	BMI	fat_rate	arm_width	leg_width	hip_width	measurement_date
1	1	128917	1.62	45.00	17.14677640603566529492	11.00	NULL	NULL	NULL	2018-11-25

With this procedure, we can access body measurements of a member with user id. Procedure first checks for presence of member, if no member exists, it’ll print a message.

- CheckOverWeight Procedure



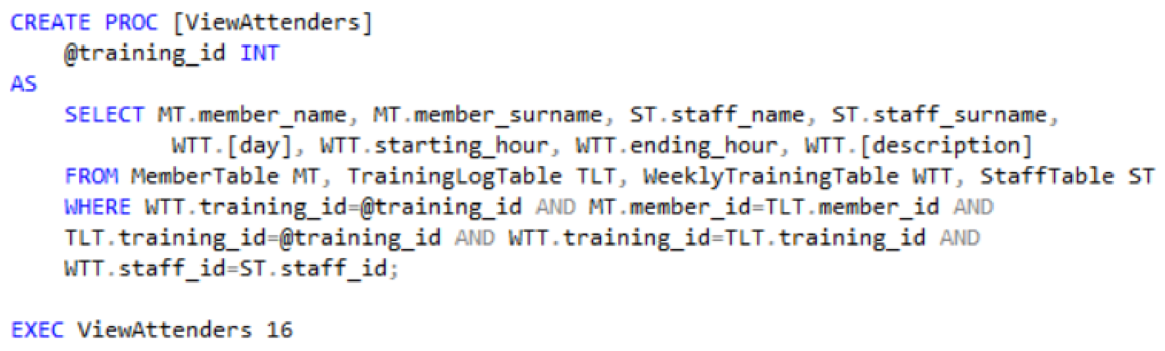
```sql
ALTER PROC [CheckOverweight]
    @member_id INT
AS
    IF NOT @member_id IN (SELECT member_id FROM MemberTable)
    BEGIN
    PRINT 'You should enter a valid member id!';
    END

    declare @BMI DECIMAL;
    declare @member_name VARCHAR(32);
    declare @member_surname VARCHAR(32);

    set @member_name = (SELECT member_name FROM MemberTable WHERE member_id=@member_id);
    set @member_surname = (SELECT member_surname FROM MemberTable WHERE member_id=@member_id);
    set @BMI = (SELECT BMI FROM MemberBodyMeasurementsTable WHERE member_id=@member_id);

    IF @BMI < 18.5
    BEGIN
        PRINT 'Member named ' + @member_name + ' ' + @member_surname + ' is underweighted';
    END
    ELSE IF @BMI > 30
    BEGIN
        PRINT 'Member named ' + @member_name + ' ' + @member_surname + ' is overweighted';
    END
    ELSE
    BEGIN
        PRINT 'Member named ' + @member_name + ' ' + @member_surname + ' is normal weighted';
    END



EXEC CheckOverweight 128917
```



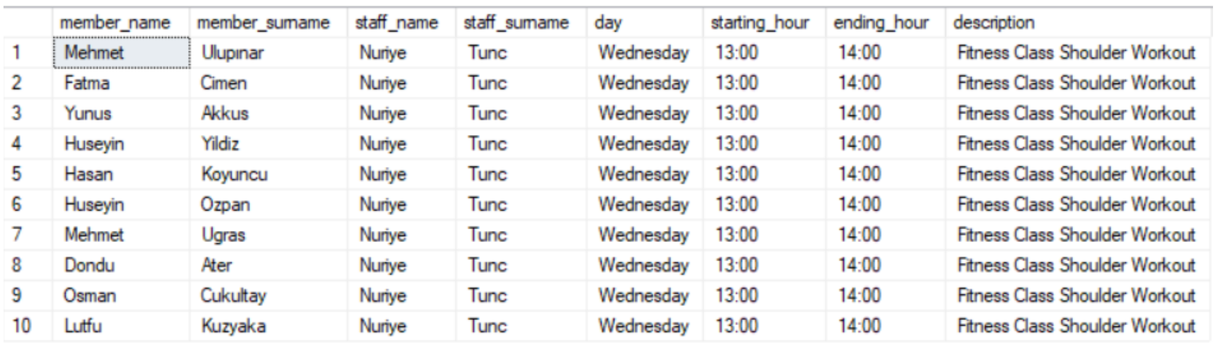
```
Messages
   Member named Rümeysa Eliöz is underweighted
```

With this procedure, we can see that if a member is underweighted, overweighted or normal weighted by giving member id. BMI which smaller than 18,5 considered as underweighted, between 18,5 and 30 considered as normal weighted, greater than 30 considered as overweighted.

- ViewAttenders Procedure



```
CREATE PROC [ViewAttenders]
    @training_id INT
AS
    SELECT MT.member_name, MT.member_surname, ST.staff_name, ST.staff_surname,
            WTT.[day], WTT.starting_hour, WTT.ending_hour, WTT.[description]
    FROM MemberTable MT, TrainingLogTable TLT, WeeklyTrainingTable WTT, StaffTable ST
    WHERE WTT.training_id=@training_id AND MT.member_id=TLT.member_id AND
    TLT.training_id=@training_id AND WTT.training_id=TLT.training_id AND
    WTT.staff_id=ST.staff_id;

EXEC ViewAttenders 16
```



	member_name	member_surname	staff_name	staff_surname	day	starting_hour	ending_hour	description
1	Mehmet	Ulupınar	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
2	Fatma	Cimen	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
3	Yunus	Akkus	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
4	Huseyin	Yildiz	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
5	Hasan	Koyuncu	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
6	Huseyin	Ozpan	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
7	Mehmet	Ugras	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
8	Dondu	Ater	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
9	Osman	Cukultay	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout
10	Lutfu	Kuzyaka	Nuriye	Tunc	Wednesday	13:00	14:00	Fitness Class Shoulder Workout

With this procedure, an authorized person can view who attended to which course by giving training id of the course as a parameter. This procedure joins 4 tables together to get important informations about the course.