Gebze Technical University
Computer Engineering

CSE 414 - 2021 Spring

DATABASE PROJECT REPORT

FURKAN
ÖZEV

161044036

Course Teacher: DR. BURCU YILMAZ

1. PROBLEM DEFINITION:

Today, we are faced with the widespread use of data in many areas and the need to control this data. A company that owns a large number of sport centers has many and many types of data. Database is a very important factor for situations such as storing, controlling and processing this data. Thus, the sport centers, members and all other information have been systematized, and an interface has been developed, facilitating access and tracking.

In this project, I realized the database system of a company that has more than one sports center. An example is MacFit.

2. OVERVIEW:

2.1 USED TECHNOLOGIES:

- Microsoft Sql Server 2018 / Management Studio (Database)

2.2 USER REQUIREMENTS:

- Storing admin user login information to access the interface and providing access and control to the db with this information.

- The information of sports centers should be kept. This information includes name, phone, address, etc.

- The information of the equipment in the sports centers should be kept. This information includes equipment code, name, price, etc.

- Members' information must be kept. This information includes name, phone, gender, registration date, gym id, address, balance etc.

- The information of the personnel working in the gyms should be kept. This information includes job code, title, etc.

- Lists of diets offered to members should be kept with their information. This information includes diet name, description, etc.

- It is necessary to detail this given diet list. These diet details, diet id, which days to apply, which times of the day to apply, food information etc. İs

- It is necessary to provide dietary relations with the member. There is a very, very much relationship here. Diet id and member id should be kept.

- There should be exercise groups served in gyms. These groups should have names, days, start and end dates.

- There is a many-to-many relationship between member and exercise. To establish this relationship, exercise id and member id must be stored.

- Rather than personal information of each member, their physical information should also be kept. This information includes memberid, measurement date, height, weight, body measurements etc.

- Payment information of each member must be kept. This information includes memberid, registration date, payment type, amount, description, etc.

- There is a many-to-many relationship between members and staff. Staff id and member id should be stored in order to keep this relationship.

- A new sport center can be added, deleted or updated.

- A new equipment can be added, deleted or updated.

- A new member can be added, deleted or updated.

- A new staff type can be added, deleted or updated.

- A new staff member can be added, deleted or their information updated.

- A new diet program can be added, deleted or updated.

- Details of these diet programs can be added, deleted or updated.

- A new exercise group can be added, deleted or updated.

- Members' physical information can be added, deleted or updated.

- Members' payment information can be added, deleted or updated.

- In addition, all this information can be filtered.

2.3 OVERVIEW:

- Sports centers can be controlled.

- Records of equipment can be checked.

- Members' personal information can be checked.

- Personnel types and personnel information can be checked.

- Information such as diet programs, information, application times can be checked. The relations of these diet programs with the members can be controlled.

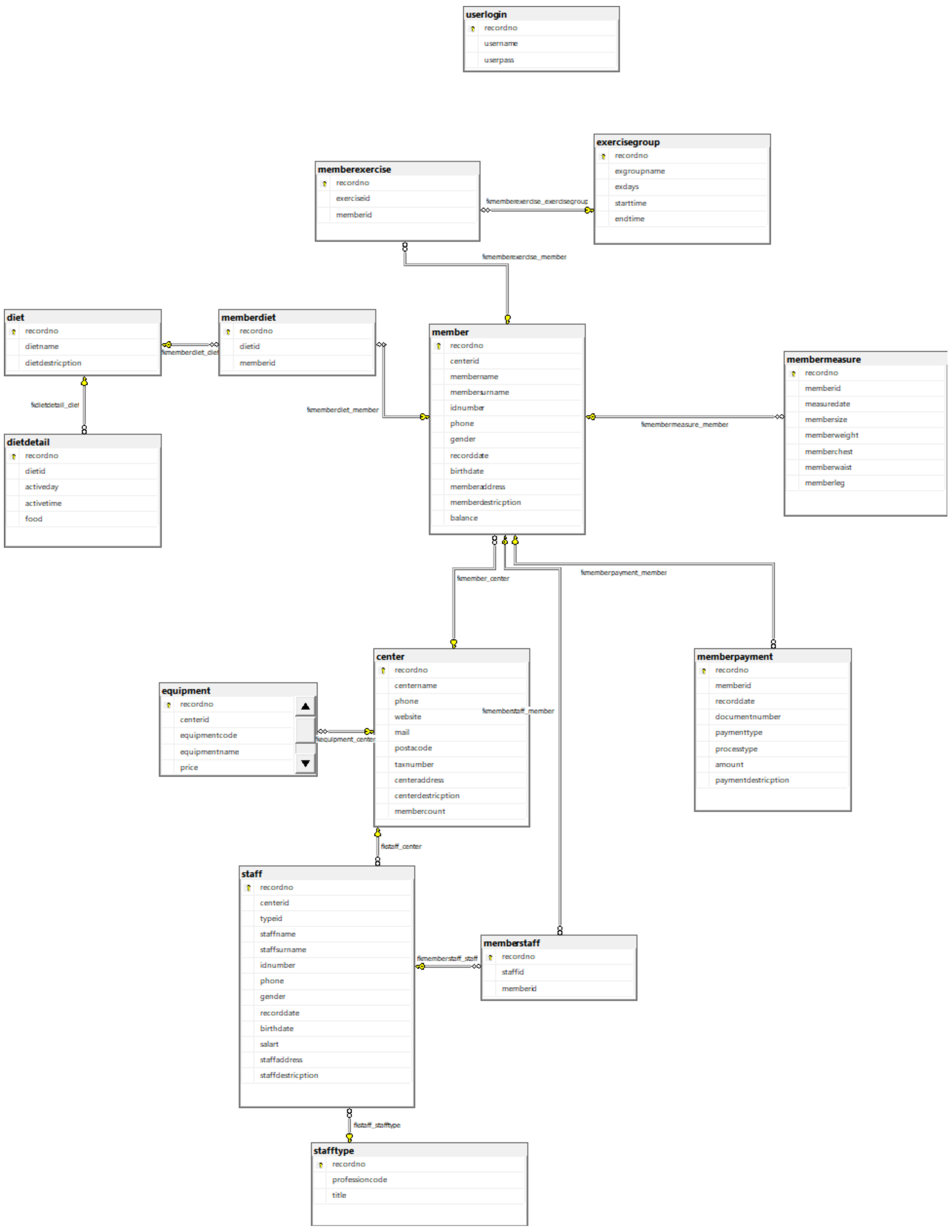
- Exercise groups information can be checked. The relations of these groups with the members can be controlled.

- Members' physical information can be checked.

- Members' payment information can be checked.

- Members' relationships with the coach can be checked.

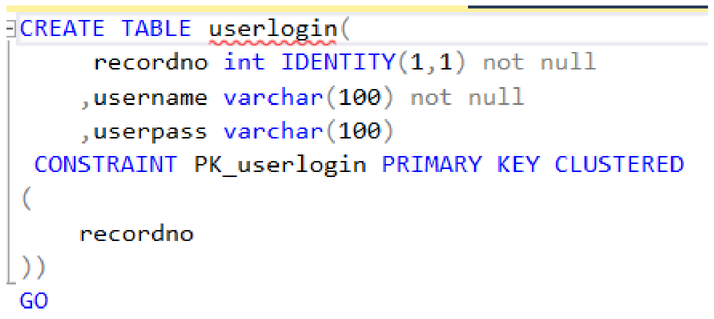
3. E-R DIAGRAM:


userlogin
recordno
username
userpass
exercisegroup
recordno
exgroupname
exdays
starttime
endtime
memberexercise
recordno
exerciseid
memberid
fkmemberexercise_exercisegroup
fkmemberexercise_member
diet
recordno
dietname
dietdestricption
memberdiet
recordno
dietid
memberid
fkmemberdiet_diet
fkdietdetail_diet
dietdetail
recordno
dietid
activeday
activetime
food
fkmemberdiet_member
member
recordno
centerid
membername
membersurname
idnumber
phone
gender
recorddate
birthdate
memberaddress
memberdestricption
balance
membermeasure
recordno
memberid
measuredate
membersize
memberweight
memberchest
memberwaist
memberleg
fkmembermeasure_member
fkmember_center
fkmemberpayment_member
center
recordno
centername
phone
website
mail
postacode
taxnumber
centeraddress
centerdestricption
membercount
fkmemberstaff_member
memberpayment
recordno
memberid
recorddate
documentnumber
paymenttype
processtype
amount
paymentdestricption
equipment
recordno
centerid
equipmentcode
equipmentname
price
fkequipment_center
fkstaff_center
staff
recordno
centerid
typeid
staffname
staffsurname
idnumber
phone
gender
recorddate
birthdate
salart
staffaddress
staffdestricption
fkmemberstaff_staff
memberstaff
recordno
staffid
memberid
fkstaff_stafftype
stafftype
recordno
professioncode
title

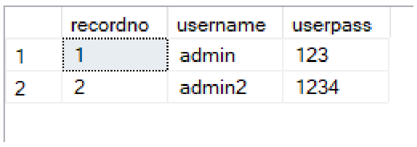
4. TABLES:

4.1 userlogin:

In order to provide database management from the interface, admin user and password information are stored.



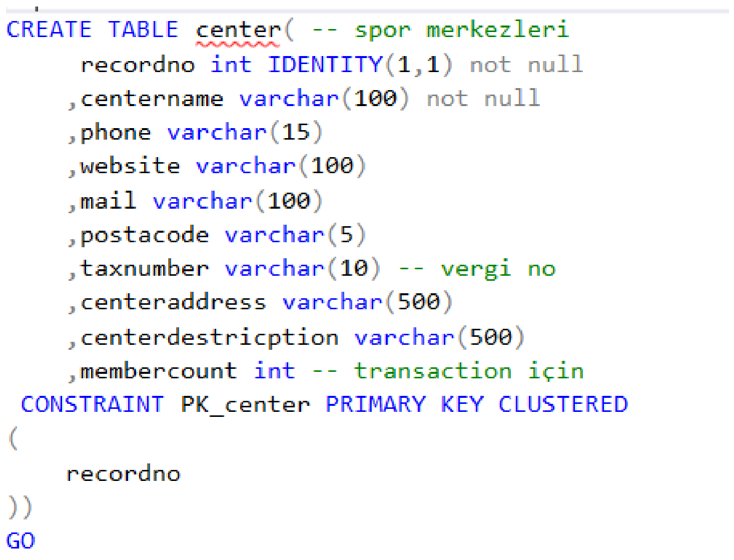
```
CREATE TABLE userlogin(
      recordno int IDENTITY(1,1) not null
     ,username varchar(100) not null
     ,userpass varchar(100)
 CONSTRAINT PK_userlogin PRIMARY KEY CLUSTERED
(
     recordno
))
GO
```



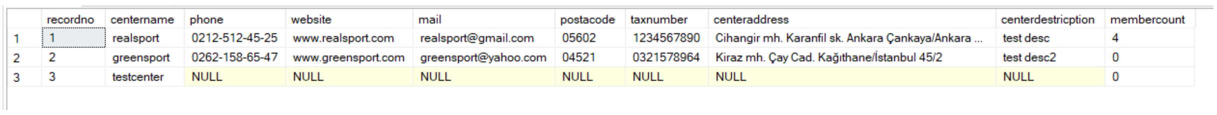
	recordno	username	userpass
1	1	admin	123
2	2	admin2	1234

4.2 center:

Detailed information of all sports centers is stored.



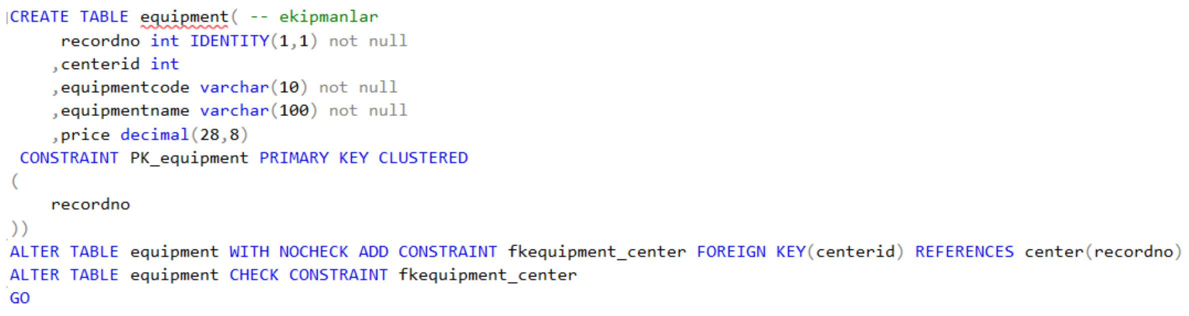
```
CREATE TABLE center( -- spor merkezleri
      recordno int IDENTITY(1,1) not null
     ,centername varchar(100) not null
     ,phone varchar(15)
     ,website varchar(100)
     ,mail varchar(100)
     ,postacode varchar(5)
     ,taxnumber varchar(10) -- vergi no
     ,centeraddress varchar(500)
     ,centerdestricption varchar(500)
     ,membercount int -- transaction için
 CONSTRAINT PK_center PRIMARY KEY CLUSTERED
(
     recordno
))
GO
```



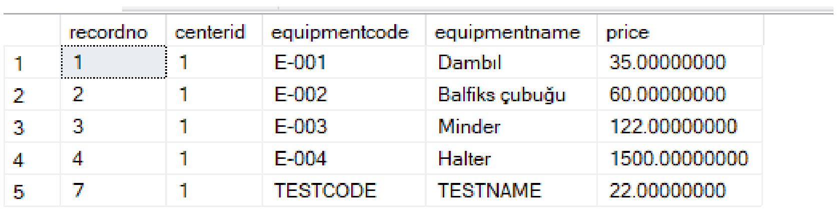
	recordno	centername	phone	website	mail	postacode	taxnumber	centeraddress	centerdestricption	membercount
1	1	realsport	0212-512-45-25	www.realsport.com	realsport@gmail.com	05602	1234567890	Cihangir mh. Karanfil sk. Ankara Çankaya/Ankara ...	test desc	4
2	2	greensport	0262-158-65-47	www.greensport.com	greensport@yahoo.com	04521	0321578964	Kiraz mh. Çay Cad. Kağıthane/İstanbul 45/2	test desc2	0
3	3	testcenter	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0

4.3 equipment:

Detailed information of all equipment in sports centers is stored. The centerid is stored to indicate which sports center the equipment belongs to.



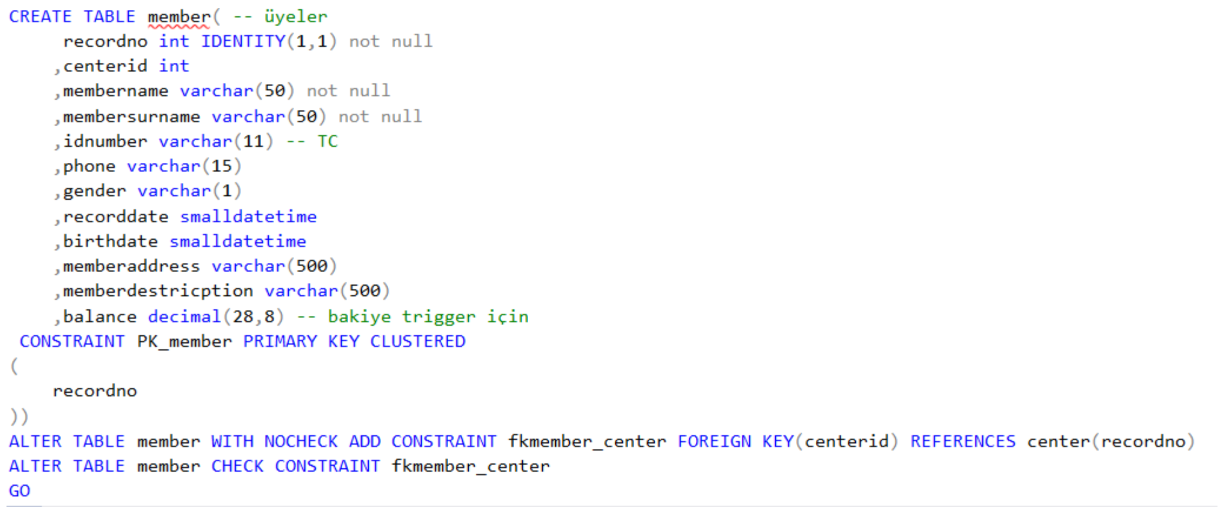
```
CREATE TABLE equipment( -- ekipmanlar
     recordno int IDENTITY(1,1) not null
    ,centerid int
    ,equipmentcode varchar(10) not null
    ,equipmentname varchar(100) not null
    ,price decimal(28,8)
 CONSTRAINT PK_equipment PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE equipment WITH NOCHECK ADD CONSTRAINT fkequipment_center FOREIGN KEY(centerid) REFERENCES center(recordno)
ALTER TABLE equipment CHECK CONSTRAINT fkequipment_center
GO
```



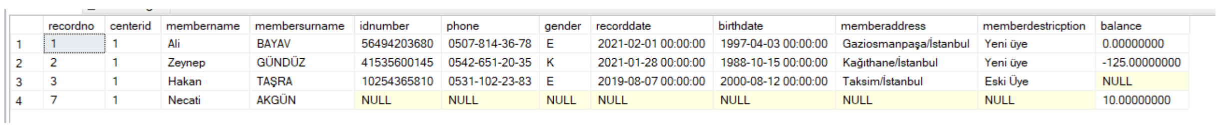
	recordno	centerid	equipmentcode	equipmentname	price
1	1	1	E-001	Dambıl	35.00000000
2	2	1	E-002	Balfiks çubuğu	60.00000000
3	3	1	E-003	Minder	122.00000000
4	4	1	E-004	Halter	1500.00000000
5	7	1	TESTCODE	TESTNAME	22.00000000

4.4 member:

All members' personal information is stored in detail. The centerid is stored to indicate which sport center the member is registered to.



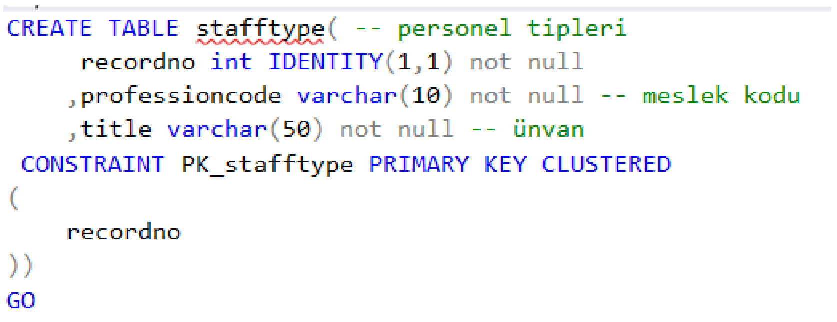
```
CREATE TABLE member( -- üyeler
     recordno int IDENTITY(1,1) not null
    ,centerid int
    ,membername varchar(50) not null
    ,membersurname varchar(50) not null
    ,idnumber varchar(11) -- TC
    ,phone varchar(15)
    ,gender varchar(1)
    ,recorddate smalldatetime
    ,birthdate smalldatetime
    ,memberaddress varchar(500)
    ,memberdestricption varchar(500)
    ,balance decimal(28,8) -- bakiye trigger için
 CONSTRAINT PK_member PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE member WITH NOCHECK ADD CONSTRAINT fkmember_center FOREIGN KEY(centerid) REFERENCES center(recordno)
ALTER TABLE member CHECK CONSTRAINT fkmember_center
GO
```



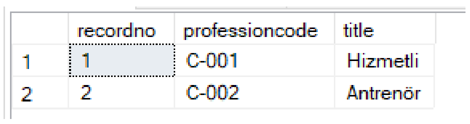
	recordno	centerid	membername	membersurname	idnumber	phone	gender	recorddate	birthdate	memberaddress	memberdestricption	balance
1	1	1	Ali	BAYAV	56494203680	0507-814-36-78	E	2021-02-01 00:00:00	1997-04-03 00:00:00	Gaziosmanpaşa/İstanbul	Yeni üye	0.00000000
2	2	1	Zeynep	GÜNDÜZ	41535600145	0542-651-20-35	K	2021-01-28 00:00:00	1988-10-15 00:00:00	Kağıthane/İstanbul	Yeni üye	-125.00000000
3	3	1	Hakan	TAŞRA	10254365810	0531-102-23-83	E	2019-08-07 00:00:00	2000-08-12 00:00:00	Taksim/İstanbul	Eski Üye	NULL
4	7	1	Necati	AKGÜN	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10.00000000

4.5 stafftype:

Personnel types are stored in detail. Information such as occupation code and title are stored.



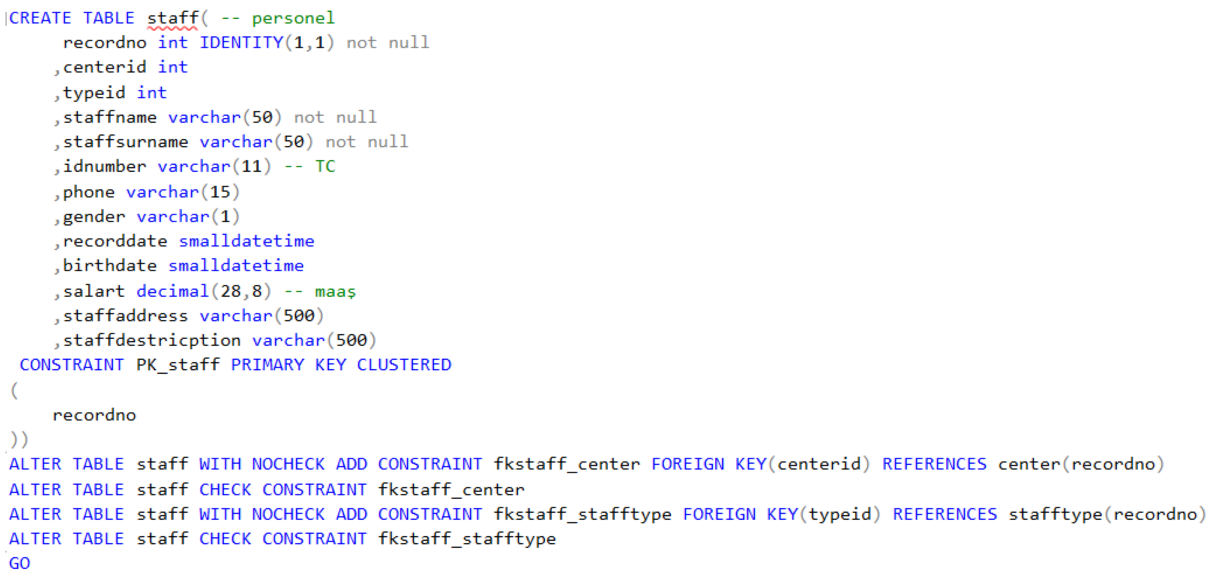
```
CREATE TABLE stafftype( -- personel tipleri
     recordno int IDENTITY(1,1) not null
    ,professioncode varchar(10) not null -- meslek kodu
    ,title varchar(50) not null -- ünvan
 CONSTRAINT PK_stafftype PRIMARY KEY CLUSTERED
(
    recordno
))
GO
```



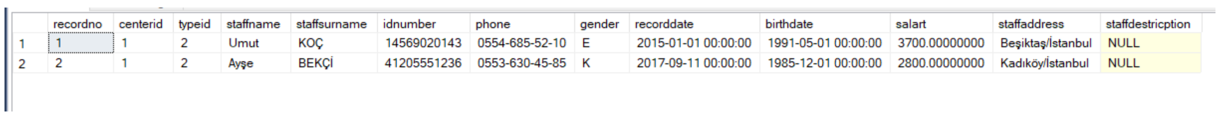
	recordno	professioncode	title
1	1	C-001	Hizmetli
2	2	C-002	Antrenör

4.6 staff:

Detailed information of the personnel is stored. The typeid is stored to indicate which type of staff the staff is. The centerid is stored to indicate which sports center the staff works at.



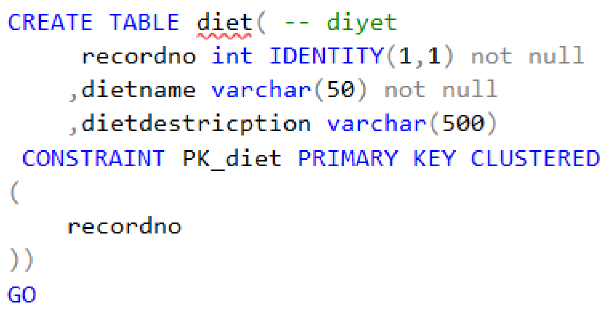
```
CREATE TABLE staff( -- personel
     recordno int IDENTITY(1,1) not null
    ,centerid int
    ,typeid int
    ,staffname varchar(50) not null
    ,staffsurname varchar(50) not null
    ,idnumber varchar(11) -- TC
    ,phone varchar(15)
    ,gender varchar(1)
    ,recorddate smalldatetime
    ,birthdate smalldatetime
    ,salart decimal(28,8) -- maaş
    ,staffaddress varchar(500)
    ,staffdestricption varchar(500)
 CONSTRAINT PK_staff PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE staff WITH NOCHECK ADD CONSTRAINT fkstaff_center FOREIGN KEY(centerid) REFERENCES center(recordno)
ALTER TABLE staff CHECK CONSTRAINT fkstaff_center
ALTER TABLE staff WITH NOCHECK ADD CONSTRAINT fkstaff_stafftype FOREIGN KEY(typeid) REFERENCES stafftype(recordno)
ALTER TABLE staff CHECK CONSTRAINT fkstaff_stafftype
GO
```



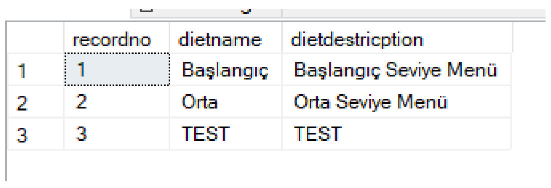
	recordno	centerid	typeid	staffname	staffsurname	idnumber	phone	gender	recorddate	birthdate	salart	staffaddress	staffdestricption
1	1	1	2	Umut	KOÇ	14569020143	0554-685-52-10	E	2015-01-01 00:00:00	1991-05-01 00:00:00	3700.00000000	Beşiktaş/İstanbul	NULL
2	2	1	2	Ayşe	BEKÇİ	41205551236	0553-630-45-85	K	2017-09-11 00:00:00	1985-12-01 00:00:00	2800.00000000	Kadıköy/İstanbul	NULL

4.7 diet:

Detailed information of diet programs is stored.



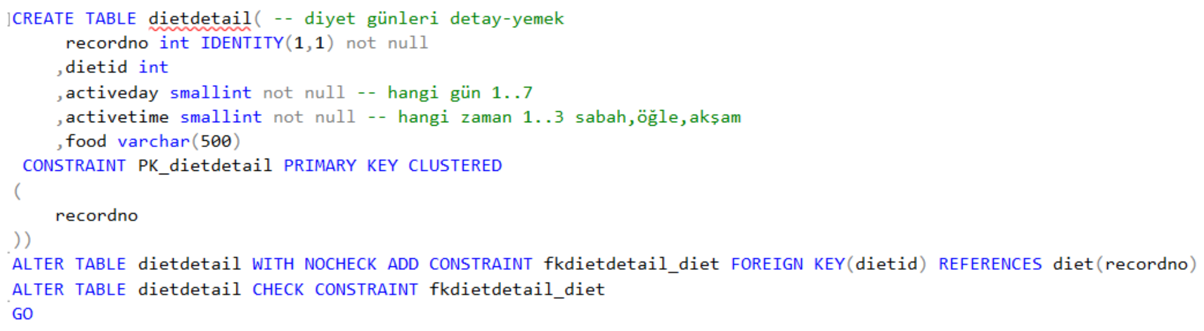
```
CREATE TABLE diet( -- diyet
      recordno int IDENTITY(1,1) not null
     ,dietname varchar(50) not null
     ,dietdestricption varchar(500)
 CONSTRAINT PK_diet PRIMARY KEY CLUSTERED
(
     recordno
))
GO
```



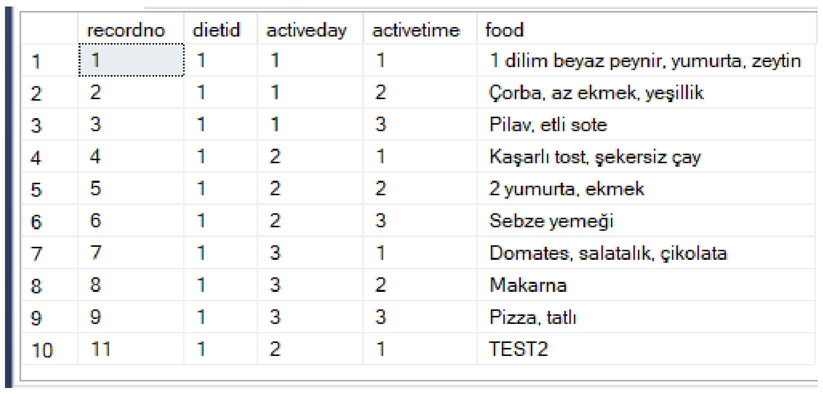
	recordno	dietname	dietdestricption
1	1	Başlangıç	Başlangıç Seviye Menü
2	2	Orta	Orta Seviye Menü
3	3	TEST	TEST

4.8 dietdetail:

More detailed information of diet programs is stored. These are the days, meals and foods on which the diet will be applied.



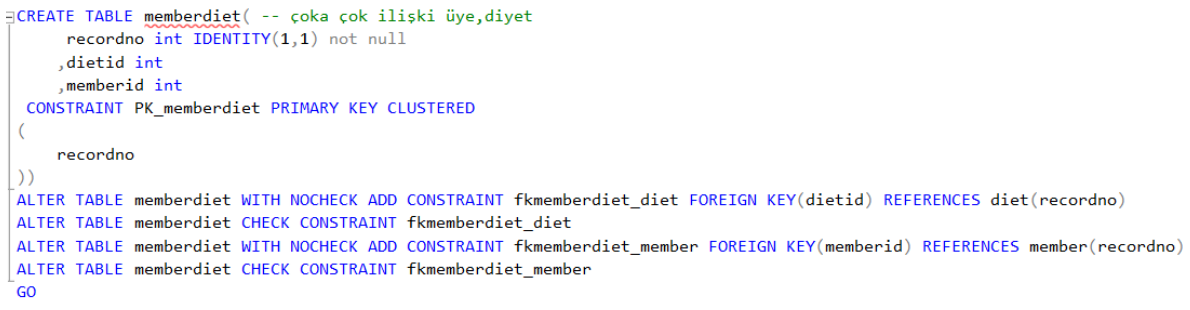
```
CREATE TABLE dietdetail( -- diyet günleri detay-yemek
      recordno int IDENTITY(1,1) not null
     ,dietid int
     ,activeday smallint not null -- hangi gün 1..7
     ,activetime smallint not null -- hangi zaman 1..3 sabah,öğle,akşam
     ,food varchar(500)
 CONSTRAINT PK_dietdetail PRIMARY KEY CLUSTERED
(
     recordno
))
ALTER TABLE dietdetail WITH NOCHECK ADD CONSTRAINT fkdietdetail_diet FOREIGN KEY(dietid) REFERENCES diet(recordno)
ALTER TABLE dietdetail CHECK CONSTRAINT fkdietdetail_diet
GO
```



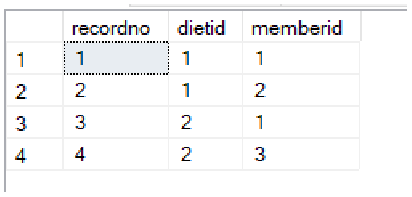
	recordno	dietid	activeday	activetime	food
1	1	1	1	1	1 dilim beyaz peynir, yumurta, zeytin
2	2	1	1	2	Çorba, az ekmek, yeşillik
3	3	1	1	3	Pilav, etli sote
4	4	1	2	1	Kaşarlı tost, şekersiz çay
5	5	1	2	2	2 yumurta, ekmek
6	6	1	2	3	Sebze yemeği
7	7	1	3	1	Domates, salatalık, çikolata
8	8	1	3	2	Makarna
9	9	1	3	3	Pizza, tatlı
10	11	1	2	1	TEST2

4.9 memberdiet:

It is the required table for assigning diet programs to members. Many-to-many relationships are provided. To ensure this, dietid and memberid are stored.



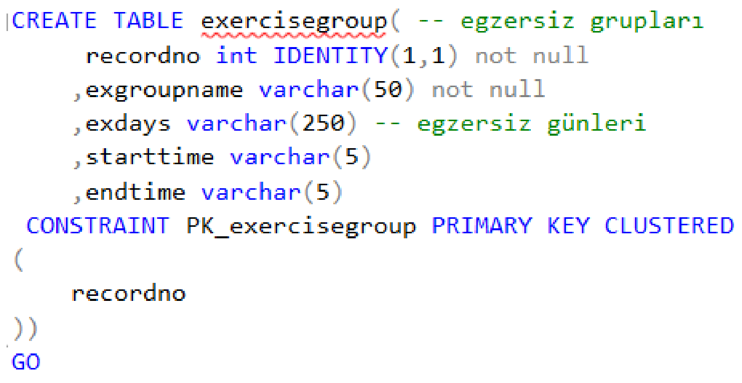
```
CREATE TABLE memberdiet( -- çoka çok ilişki üye,diyet
     recordno int IDENTITY(1,1) not null
    ,dietid int
    ,memberid int
 CONSTRAINT PK_memberdiet PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE memberdiet WITH NOCHECK ADD CONSTRAINT fkmemberdiet_diet FOREIGN KEY(dietid) REFERENCES diet(recordno)
ALTER TABLE memberdiet CHECK CONSTRAINT fkmemberdiet_diet
ALTER TABLE memberdiet WITH NOCHECK ADD CONSTRAINT fkmemberdiet_member FOREIGN KEY(memberid) REFERENCES member(recordno)
ALTER TABLE memberdiet CHECK CONSTRAINT fkmemberdiet_member
GO
```



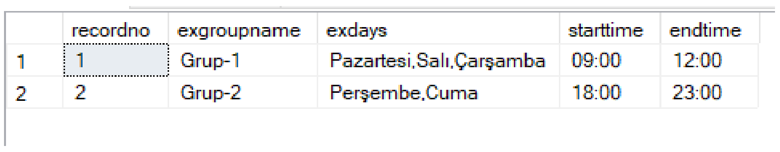
	recordno	dietid	memberid
1	1	1	1
2	2	1	2
3	3	2	1
4	4	2	3

4.10 exercisegroup:

Detailed information of exercise groups is to be stored.



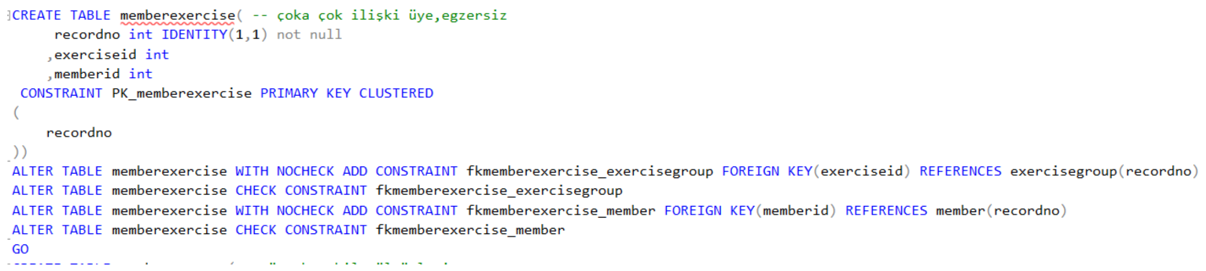
```
CREATE TABLE exercisegroup( -- egzersiz grupları
     recordno int IDENTITY(1,1) not null
    ,exgroupname varchar(50) not null
    ,exdays varchar(250) -- egzersiz günleri
    ,starttime varchar(5)
    ,endtime varchar(5)
 CONSTRAINT PK_exercisegroup PRIMARY KEY CLUSTERED
(
    recordno
))
GO
```



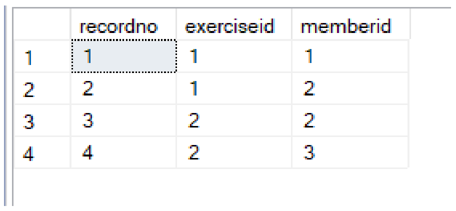
	recordno	exgroupname	exdays	starttime	endtime
1	1	Grup-1	Pazartesi,Salı,Çarşamba	09:00	12:00
2	2	Grup-2	Perşembe,Cuma	18:00	23:00

4.11 memberexercise:

It is the required table for assigning exercises to members. Many-to-many relationships are provided. To ensure this, exerciseid and memberid are stored.



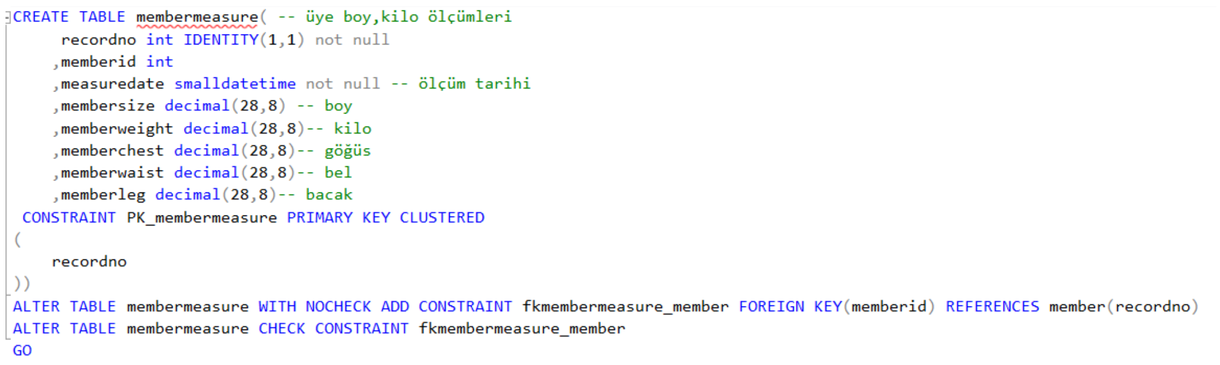
```
CREATE TABLE memberexercise( -- çoka çok ilişki üye,egzersiz
     recordno int IDENTITY(1,1) not null
    ,exerciseid int
    ,memberid int
 CONSTRAINT PK_memberexercise PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE memberexercise WITH NOCHECK ADD CONSTRAINT fkmemberexercise_exercisegroup FOREIGN KEY(exerciseid) REFERENCES exercisegroup(recordno)
ALTER TABLE memberexercise CHECK CONSTRAINT fkmemberexercise_exercisegroup
ALTER TABLE memberexercise WITH NOCHECK ADD CONSTRAINT fkmemberexercise_member FOREIGN KEY(memberid) REFERENCES member(recordno)
ALTER TABLE memberexercise CHECK CONSTRAINT fkmemberexercise_member
GO
```



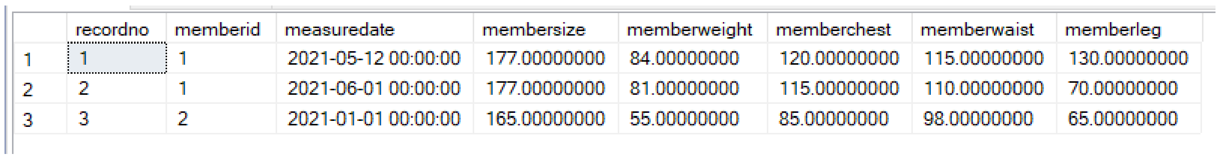
	recordno	exerciseid	memberid
1	1	1	1
2	2	1	2
3	3	2	2
4	4	2	3

4.12 membermeasure:

The physical characteristics of the members are kept in detail. The memberid is stored to indicate which member these properties belong to.



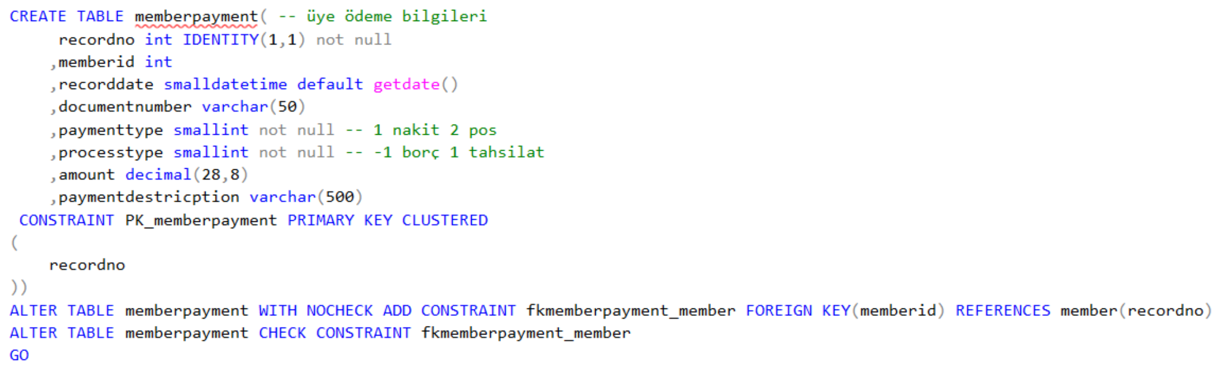
```
CREATE TABLE membermeasure( -- üye boy,kilo ölçümleri
     recordno int IDENTITY(1,1) not null
    ,memberid int
    ,measuredate smalldatetime not null -- ölçüm tarihi
    ,membersize decimal(28,8) -- boy
    ,memberweight decimal(28,8)-- kilo
    ,memberchest decimal(28,8)-- göğüs
    ,memberwaist decimal(28,8)-- bel
    ,memberleg decimal(28,8)-- bacak
 CONSTRAINT PK_membermeasure PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE membermeasure WITH NOCHECK ADD CONSTRAINT fkmembermeasure_member FOREIGN KEY(memberid) REFERENCES member(recordno)
ALTER TABLE membermeasure CHECK CONSTRAINT fkmembermeasure_member
GO
```



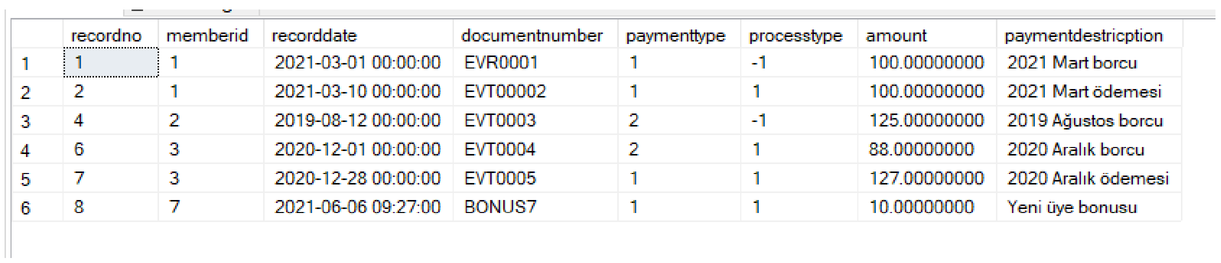
	recordno	memberid	measuredate	membersize	memberweight	memberchest	memberwaist	memberleg
1	1	1	2021-05-12 00:00:00	177.00000000	84.00000000	120.00000000	115.00000000	130.00000000
2	2	1	2021-06-01 00:00:00	177.00000000	81.00000000	115.00000000	110.00000000	70.00000000
3	3	2	2021-01-01 00:00:00	165.00000000	55.00000000	85.00000000	98.00000000	65.00000000

4.13 memberpayment:

Members' payment information is kept in detail. The memberid is stored to indicate which member this information belongs to.



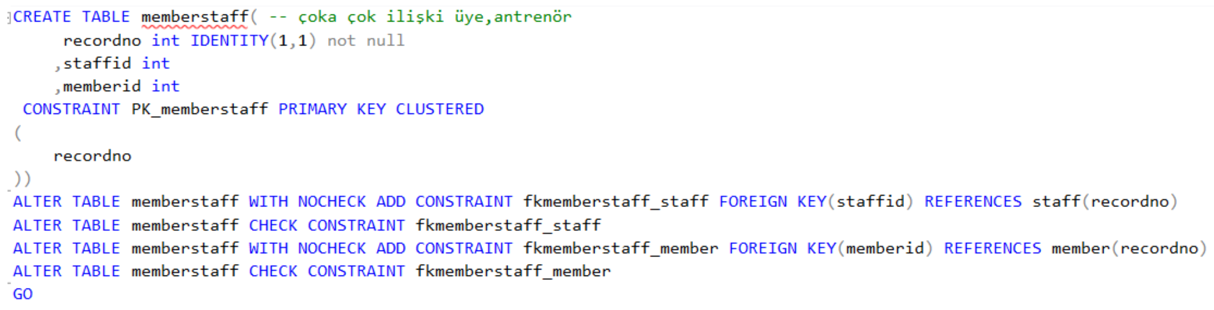
```
CREATE TABLE memberpayment( -- üye ödeme bilgileri
     recordno int IDENTITY(1,1) not null
    ,memberid int
    ,recorddate smalldatetime default getdate()
    ,documentnumber varchar(50)
    ,paymenttype smallint not null -- 1 nakit 2 pos
    ,processtype smallint not null -- -1 borç 1 tahsilat
    ,amount decimal(28,8)
    ,paymentdestricption varchar(500)
 CONSTRAINT PK_memberpayment PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE memberpayment WITH NOCHECK ADD CONSTRAINT fkmemberpayment_member FOREIGN KEY(memberid) REFERENCES member(recordno)
ALTER TABLE memberpayment CHECK CONSTRAINT fkmemberpayment_member
GO
```



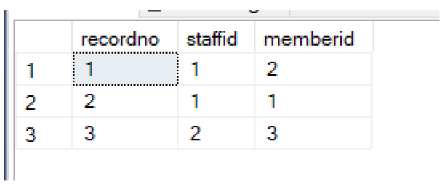
	recordno	memberid	recorddate	documentnumber	paymenttype	processtype	amount	paymentdestricption
1	1	1	2021-03-01 00:00:00	EVR0001	1	-1	100.00000000	2021 Mart borcu
2	2	1	2021-03-10 00:00:00	EVT00002	1	1	100.00000000	2021 Mart ödemesi
3	4	2	2019-08-12 00:00:00	EVT0003	2	-1	125.00000000	2019 Ağustos borcu
4	6	3	2020-12-01 00:00:00	EVT0004	2	1	88.00000000	2020 Aralık borcu
5	7	3	2020-12-28 00:00:00	EVT0005	1	1	127.00000000	2020 Aralık ödemesi
6	8	7	2021-06-06 09:27:00	BONUS7	1	1	10.00000000	Yeni üye bonusu

4.14 memberstaff:

It is the required table for assigning staffs to members. Many-to-many relationships are provided. To ensure this, staffid and memberid are stored.



```
CREATE TABLE memberstaff( -- çoka çok ilişki üye,antrenör
     recordno int IDENTITY(1,1) not null
    ,staffid int
    ,memberid int
 CONSTRAINT PK_memberstaff PRIMARY KEY CLUSTERED
(
    recordno
))
ALTER TABLE memberstaff WITH NOCHECK ADD CONSTRAINT fkmemberstaff_staff FOREIGN KEY(staffid) REFERENCES staff(recordno)
ALTER TABLE memberstaff CHECK CONSTRAINT fkmemberstaff_staff
ALTER TABLE memberstaff WITH NOCHECK ADD CONSTRAINT fkmemberstaff_member FOREIGN KEY(memberid) REFERENCES member(recordno)
ALTER TABLE memberstaff CHECK CONSTRAINT fkmemberstaff_member
GO
```



	recordno	staffid	memberid
1	1	1	2
2	2	1	1
3	3	2	3

5. NORMALIZATION:

In order for the table to comply with 1NF, there must be no multivalued attribute in the table. Every attribute in the table must have an atomic single degree. Only one value can be saved for all columns in tables. In order for the table to comply with 2NF, there must be no partial dependency in the table. Since non-prime attributes depend on a part of the candidate key, partial dependency occurs. Non-prime attributes in the table are dependent on the primary key. For the table to conform to 3NF, the transitive functional dependency must be removed. There is no transitive situation in the tables.

So, Each table cell contains a single value and each record is unique. The single column is the primary key. Like memberid, recordid, vs. It has no transitive functional dependencies. For this reason, it conforms to the Boyce Codd Normal Form.

Field types are available in the images in the tables section.

5.1 userlogin Table:

Attributes: recordno (**primary key**), username, userpass

5.2 center Table:

Attributes: recordno (**primary key**), centername, phone, website, mail, postacode, taxnumber, centeraddress, centerdestricption, membercount

5.3 equipment Table:

Attributes: recordno (**primary key**), centerid**(foreign key**), equipmentcode, equipmentname, price

centerid < center(recordno)

5.4 member Table:

Attributes: recordno (**primary key**), centerid(**foreign key**), membername, membersurname, idnumber, phone, gender, recorddate, birthdate, memberaddress, memberdestricption, balance

centerid < center(recordno)

5.5 stafftype Table:

Attributes: recordno (**primary key**), professioncode, title

5.6 staff Table:

Attributes: recordno (**primary key**), centerid(**foreign key**), typeid(**foreign key**), staffname, staffsurname, idnumber, phone, gender, recorddate, birthdae, salart, staffaddress, staffdestricption

centerid < center(recordno)

typeid < stafftype(recordno)

5.7 diet Table:

Attributes: recordno (**primary key**), dietname, dietdestricption

5.8 dietdetail Table:

Attributes: recordno (**primary key**), dietid (**foreign key**), activeday, activetime, food

dietid < diet(recordno)

5.9 memberdiet Table:

Attributes: recordno (**primary key**), dietid (**foreign key**), memberid (**foreign key**)

dietid < diet(recordno)

memberid < member(recordno)

5.10 exercisegroup Table:

Attributes: recordno (**primary key**), exgroupname, exdays, starttime, endtime

5.11 memberexercise Table:

Attributes: recordno (**primary key**), exerciseid (**foreign key**), memberid (**foreign key**)

exerciseid < exercisegroup(recordno)

memberid < member(recordno)

5.12 membermeasure Table:

Attributes: recordno (**primary key**), memberid (**foreign key**), measuredate, membersize, memberweight, memberchest, memberwaist, memberleg

memberid < member(recordno)

5.13 memberpayment Table:

Attributes: recordno (**primary key**), memberid (**foreign key**), recorddate, documentnumber, paymenttype, processtype, amount, paymentdestricption

memberid < member(recordno)

5.14 memberstaff Table:

Attributes: recordno (**primary key**), staffid (**foreign key**), memberid **(foreign key**)

staffid < staff(recordno)

memberid < member(recordno)

6. FUNCTIONAL DEPENDENCIES:

6.1 userlogin Table:

recordno -> username, userpass

6.2 center Table:

recordno -> centername, phone, website, mail, postacode, taxnumber, centeraddress, centerdestricption, membercount

6.3 equipment Table:

recordno -> centerid, equipmentcode, equipmentname, price

6.4 member Table:

recordno -> centerid, membername, membersurname, idnumber, phone, gender, recorddate, birthdate, memberaddress, member destricption, balance

6.5 stafftype Table:

recordno -> professioncode, title

6.6 staff Table:

recordno -> centerid, typeid, staffname, staffsurname, idnumber, phone, gender, recorddate, birthdate, salart, staffaddress, staffdestricption,

6.7 diet Table:

recordno -> dietname, dietdestricption

6.8 dietdetail Table:

recordno -> dietid, activeday, activetime, food

6.9 memberdiet Table:

recordno -> dietid, memberid

6.10 exercisegroup Table:

recordno -> exgroupname, exdays, starttime, endtime

6.11 memberexercise Table:

recordno -> exerciseid, memberid

6.12 membermeasure Table:

recordno -> memberid, measuredata, membersize, memberweight, memberchest, memberwaist, memberleg

6.13 memberpayment Table:

recordno -> memberid, recorddate, documentnumber, paymenttype, processtype, amount, paymentdestricption

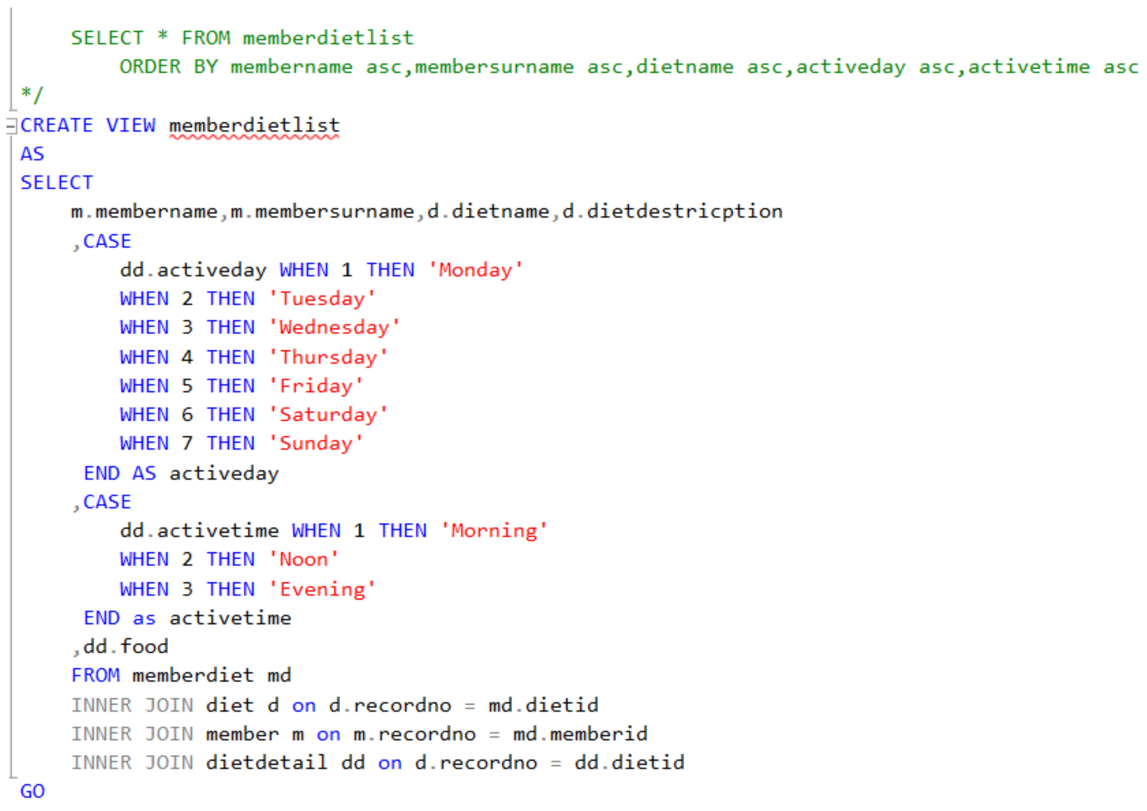
6.14 memberstaff Table:

recordno -> staffid, memberid

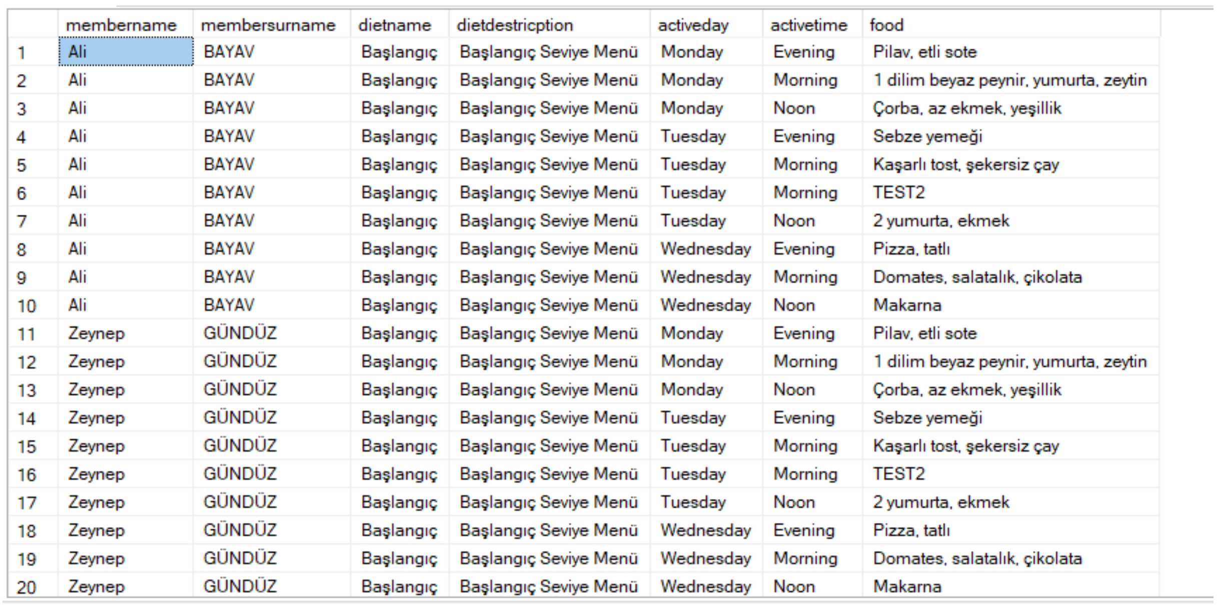
7. VIEWS:

7.1 memberdietlist View:

It is a view that shows the diet programs and details that the members should follow. member, diet, dietdetail tables were formed by combining.



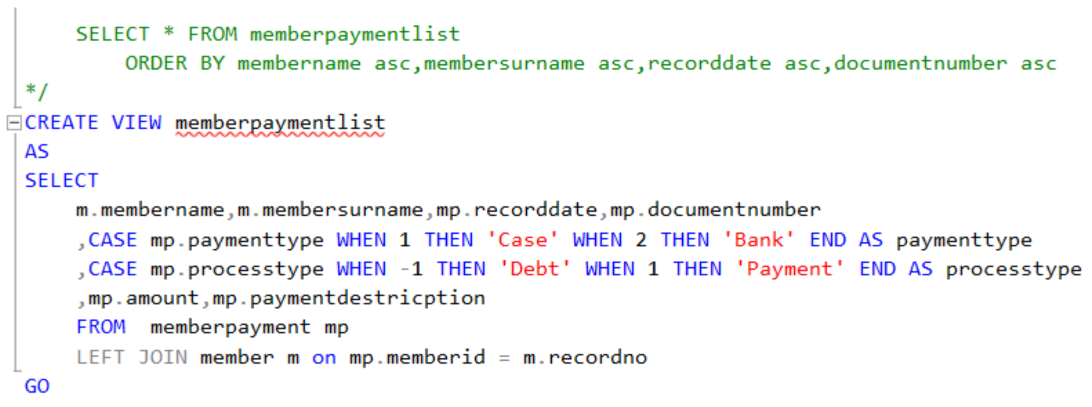
```
    SELECT * FROM memberdietlist
        ORDER BY membername asc,membersurname asc,dietname asc,activeday asc,activetime asc
*/
CREATE VIEW memberdietlist
AS
SELECT
    m.membername,m.membersurname,d.dietname,d.dietdestricption
    ,CASE
        dd.activeday WHEN 1 THEN 'Monday'
        WHEN 2 THEN 'Tuesday'
        WHEN 3 THEN 'Wednesday'
        WHEN 4 THEN 'Thursday'
        WHEN 5 THEN 'Friday'
        WHEN 6 THEN 'Saturday'
        WHEN 7 THEN 'Sunday'
     END AS activeday
    ,CASE
        dd.activetime WHEN 1 THEN 'Morning'
        WHEN 2 THEN 'Noon'
        WHEN 3 THEN 'Evening'
     END as activetime
    ,dd.food
    FROM memberdiet md
    INNER JOIN diet d on d.recordno = md.dietid
    INNER JOIN member m on m.recordno = md.memberid
    INNER JOIN dietdetail dd on d.recordno = dd.dietid
GO
```



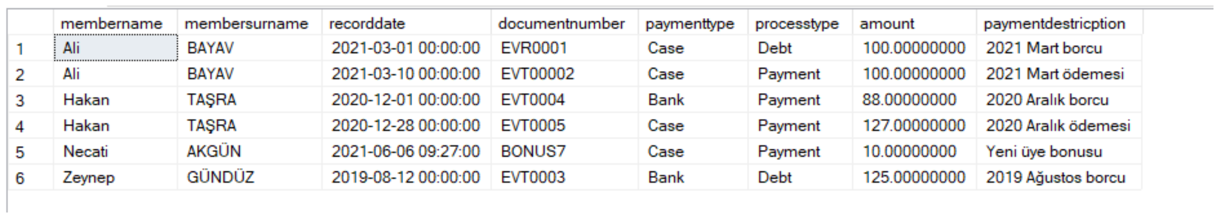
	membername	membersurname	dietname	dietdestricption	activeday	activetime	food
1	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Monday	Evening	Pilav, etli sote
2	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Monday	Morning	1 dilim beyaz peynir, yumurta, zeytin
3	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Monday	Noon	Çorba, az ekmek, yeşillik
4	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Tuesday	Evening	Sebze yemeği
5	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Tuesday	Morning	Kaşarlı tost, şekersiz çay
6	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Tuesday	Morning	TEST2
7	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Tuesday	Noon	2 yumurta, ekmek
8	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Wednesday	Evening	Pizza, tatlı
9	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Wednesday	Morning	Domates, salatalık, çikolata
10	Ali	BAYAV	Başlangıç	Başlangıç Seviye Menü	Wednesday	Noon	Makarna
11	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Monday	Evening	Pilav, etli sote
12	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Monday	Morning	1 dilim beyaz peynir, yumurta, zeytin
13	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Monday	Noon	Çorba, az ekmek, yeşillik
14	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Tuesday	Evening	Sebze yemeği
15	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Tuesday	Morning	Kaşarlı tost, şekersiz çay
16	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Tuesday	Morning	TEST2
17	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Tuesday	Noon	2 yumurta, ekmek
18	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Wednesday	Evening	Pizza, tatlı
19	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Wednesday	Morning	Domates, salatalık, çikolata
20	Zeynep	GÜNDÜZ	Başlangıç	Başlangıç Seviye Menü	Wednesday	Noon	Makarna

7.2 memberpaymentlist View:

It is a view that shows the payment details that the members should follow. member and memberpayment tables were formed by combining.



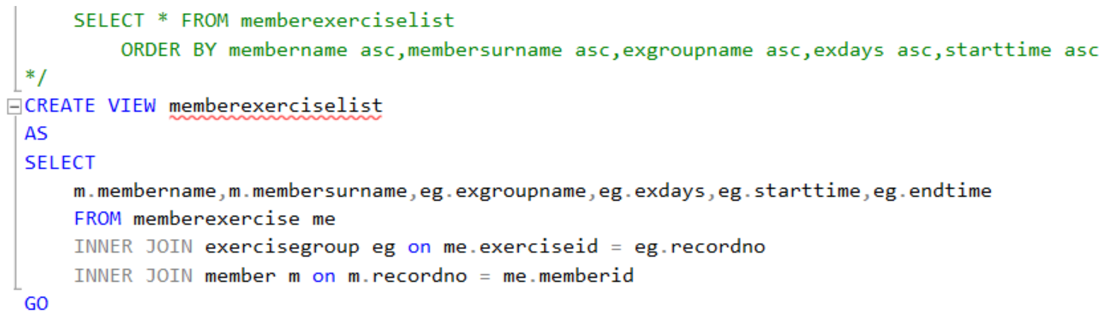
```
    SELECT * FROM memberpaymentlist
        ORDER BY membername asc,membersurname asc,recorddate asc,documentnumber asc
*/
CREATE VIEW memberpaymentlist
AS
SELECT
    m.membername,m.membersurname,mp.recorddate,mp.documentnumber
    ,CASE mp.paymenttype WHEN 1 THEN 'Case' WHEN 2 THEN 'Bank' END AS paymenttype
    ,CASE mp.processtype WHEN -1 THEN 'Debt' WHEN 1 THEN 'Payment' END AS processtype
    ,mp.amount,mp.paymentdestricption
    FROM  memberpayment mp
    LEFT JOIN member m on mp.memberid = m.recordno
GO
```



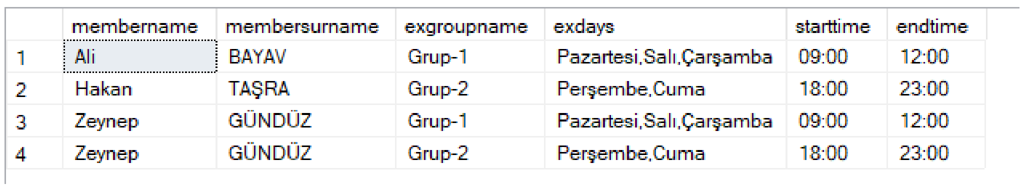
	membername	membersurname	recorddate	documentnumber	paymenttype	processtype	amount	paymentdestricption
1	Ali	BAYAV	2021-03-01 00:00:00	EVR0001	Case	Debt	100.00000000	2021 Mart borcu
2	Ali	BAYAV	2021-03-10 00:00:00	EVT00002	Case	Payment	100.00000000	2021 Mart ödemesi
3	Hakan	TAŞRA	2020-12-01 00:00:00	EVT0004	Bank	Payment	88.00000000	2020 Aralık borcu
4	Hakan	TAŞRA	2020-12-28 00:00:00	EVT0005	Case	Payment	127.00000000	2020 Aralık ödemesi
5	Necati	AKGÜN	2021-06-06 09:27:00	BONUS7	Case	Payment	10.00000000	Yeni üye bonusu
6	Zeynep	GÜNDÜZ	2019-08-12 00:00:00	EVT0003	Bank	Debt	125.00000000	2019 Ağustos borcu

7.3 memberexerciselist View:

It is a view that shows the exercise details that the members should follow. member and exercisegroup tables were formed by combining.



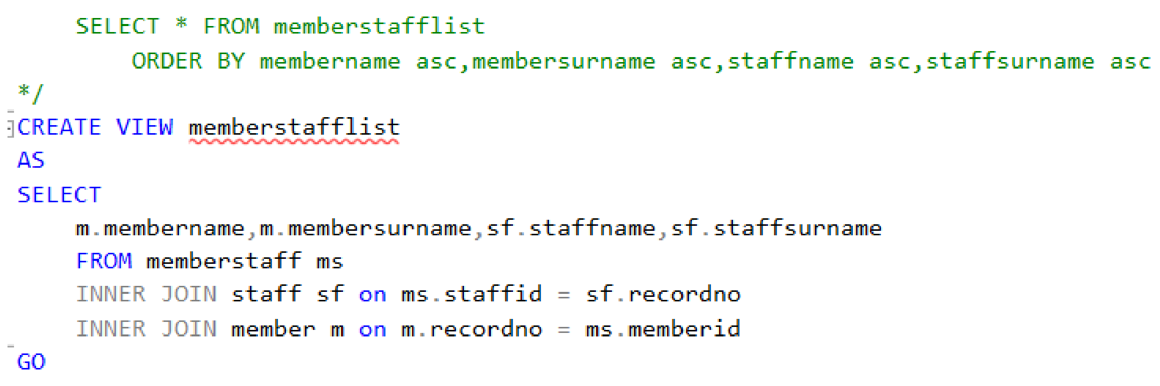
```
    SELECT * FROM memberexerciselist
        ORDER BY membername asc,membersurname asc,exgroupname asc,exdays asc,starttime asc
*/
CREATE VIEW memberexerciselist
AS
SELECT
    m.membername,m.membersurname,eg.exgroupname,eg.exdays,eg.starttime,eg.endtime
    FROM memberexercise me
    INNER JOIN exercisegroup eg on me.exerciseid = eg.recordno
    INNER JOIN member m on m.recordno = me.memberid
GO
```



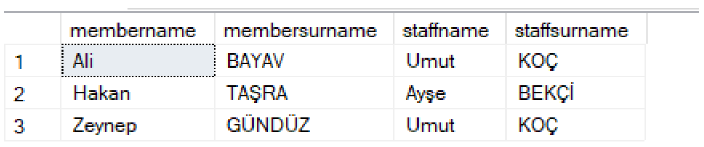
	membername	membersurname	exgroupname	exdays	starttime	endtime
1	Ali	BAYAV	Grup-1	Pazartesi,Salı,Çarşamba	09:00	12:00
2	Hakan	TAŞRA	Grup-2	Perşembe,Cuma	18:00	23:00
3	Zeynep	GÜNDÜZ	Grup-1	Pazartesi,Salı,Çarşamba	09:00	12:00
4	Zeynep	GÜNDÜZ	Grup-2	Perşembe,Cuma	18:00	23:00

7.4 memberstafflist View:

It is a view that shows the staff that the members should work together. member and staff tables were formed by combining.



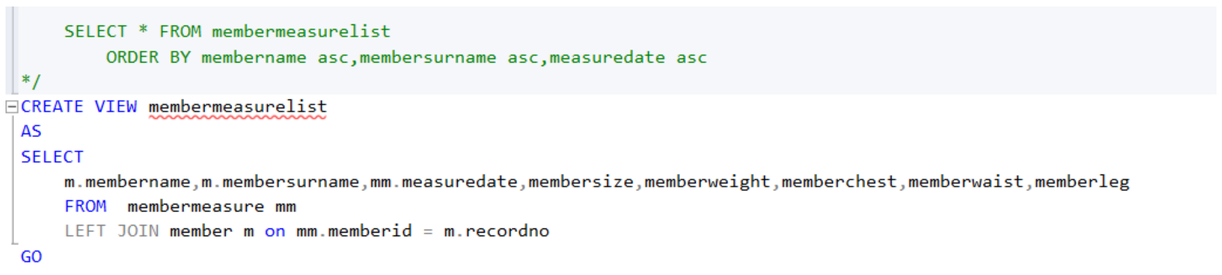
```
    SELECT * FROM memberstafflist
        ORDER BY membername asc,membersurname asc,staffname asc,staffsurname asc
*/
CREATE VIEW memberstafflist
AS
SELECT
    m.membername,m.membersurname,sf.staffname,sf.staffsurname
    FROM memberstaff ms
    INNER JOIN staff sf on ms.staffid = sf.recordno
    INNER JOIN member m on m.recordno = ms.memberid
GO
```



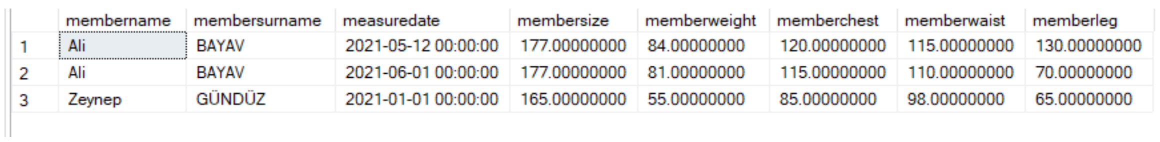
	membername	membersurname	staffname	staffsurname
1	Ali	BAYAV	Umut	KOÇ
2	Hakan	TAŞRA	Ayşe	BEKÇİ
3	Zeynep	GÜNDÜZ	Umut	KOÇ

7.5 membermeasurelist View:

A view that contains members and their physical measurements. member and membermeasure tables were formed by combining.



```
    SELECT * FROM membermeasurelist
        ORDER BY membername asc,membersurname asc,measuredate asc
*/
CREATE VIEW membermeasurelist
AS
SELECT
    m.membername,m.membersurname,mm.measuredate,membersize,memberweight,memberchest,memberwaist,memberleg
    FROM  membermeasure mm
    LEFT JOIN member m on mm.memberid = m.recordno
GO
```

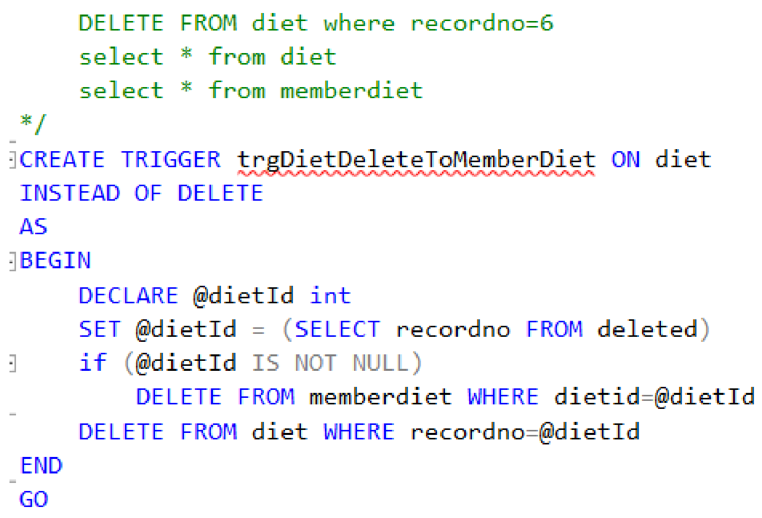


	membername	membersurname	measuredate	membersize	memberweight	memberchest	memberwaist	memberleg
1	Ali	BAYAV	2021-05-12 00:00:00	177.00000000	84.00000000	120.00000000	115.00000000	130.00000000
2	Ali	BAYAV	2021-06-01 00:00:00	177.00000000	81.00000000	115.00000000	110.00000000	70.00000000
3	Zeynep	GÜNDÜZ	2021-01-01 00:00:00	165.00000000	55.00000000	85.00000000	98.00000000	65.00000000

8. TRIGGERS:

8.1 trgDietDeleteToMemberDiet Trigger:

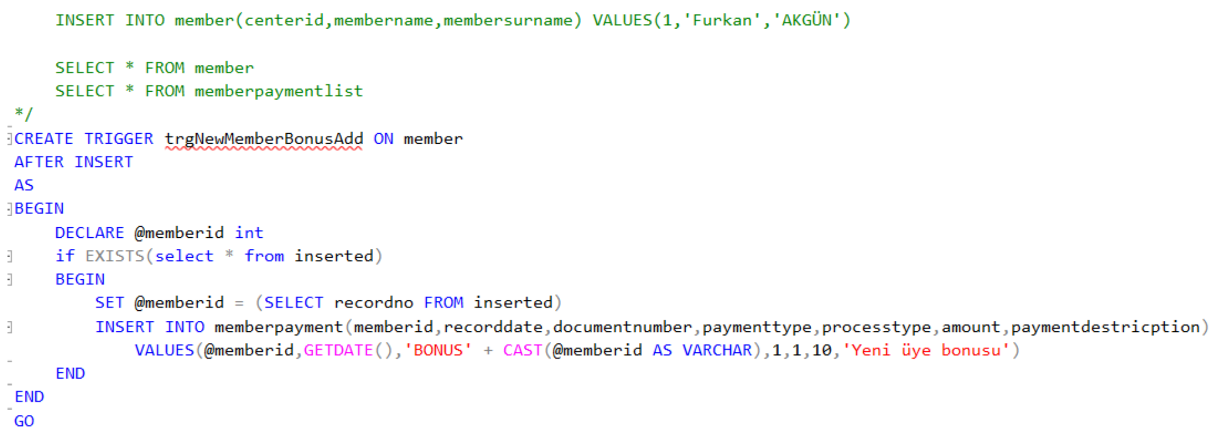
When a diet in the diet chart is deleted, if this diet is already assigned to members, they must be deleted. Therefore, that data will be deleted in the memberdiet table first. Then the diet will be deleted.



```
    DELETE FROM diet where recordno=6
    select * from diet
    select * from memberdiet
*/
CREATE TRIGGER trgDietDeleteToMemberDiet ON diet
INSTEAD OF DELETE
AS
BEGIN
    DECLARE @dietId int
    SET @dietId = (SELECT recordno FROM deleted)
    if (@dietId IS NOT NULL)
        DELETE FROM memberdiet WHERE dietid=@dietId
    DELETE FROM diet WHERE recordno=@dietId
END
GO
```

8.2 trgNewMemberBonusAdd Trigger:

When a new member is added to the member table, an automatic bonus payment is added to the member. This bonus adds to the memberpayment table.

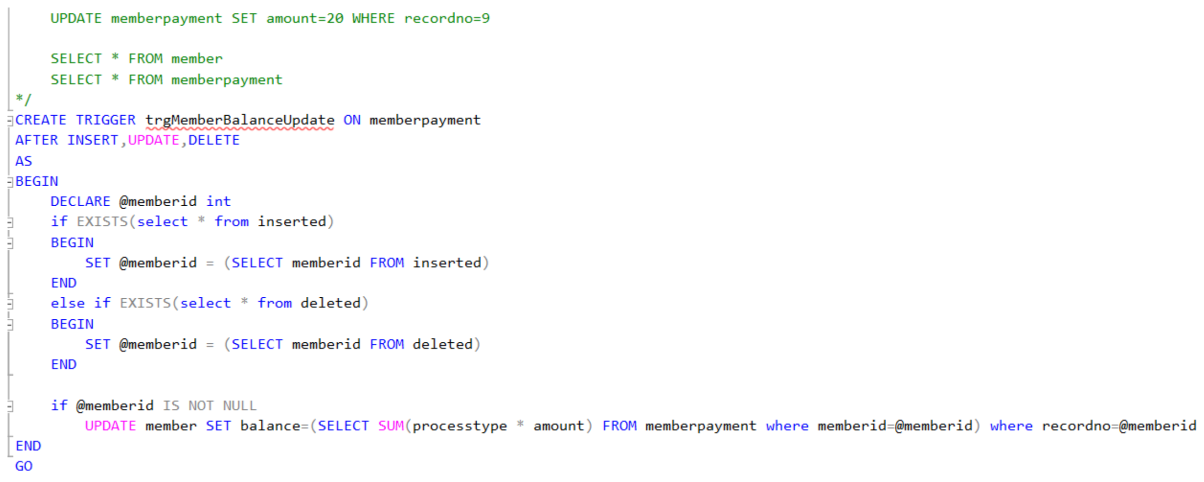


```
    INSERT INTO member(centerid,membername,membersurname) VALUES(1,'Furkan','AKGÜN')

    SELECT * FROM member
    SELECT * FROM memberpaymentlist
*/
CREATE TRIGGER trgNewMemberBonusAdd ON member
AFTER INSERT
AS
BEGIN
    DECLARE @memberid int
    if EXISTS(select * from inserted)
    BEGIN
        SET @memberid = (SELECT recordno FROM inserted)
        INSERT INTO memberpayment(memberid,recorddate,documentnumber,paymenttype,processtype,amount,paymentdestricption)
            VALUES(@memberid,GETDATE(),'BONUS' + CAST(@memberid AS VARCHAR),1,1,10,'Yeni üye bonusu')
    END
END
GO
```

8.3 trgMemberBalanceUpdate Trigger:

If a payment is added for any member in the memberpayment table, that member's balance is updated in the member table.



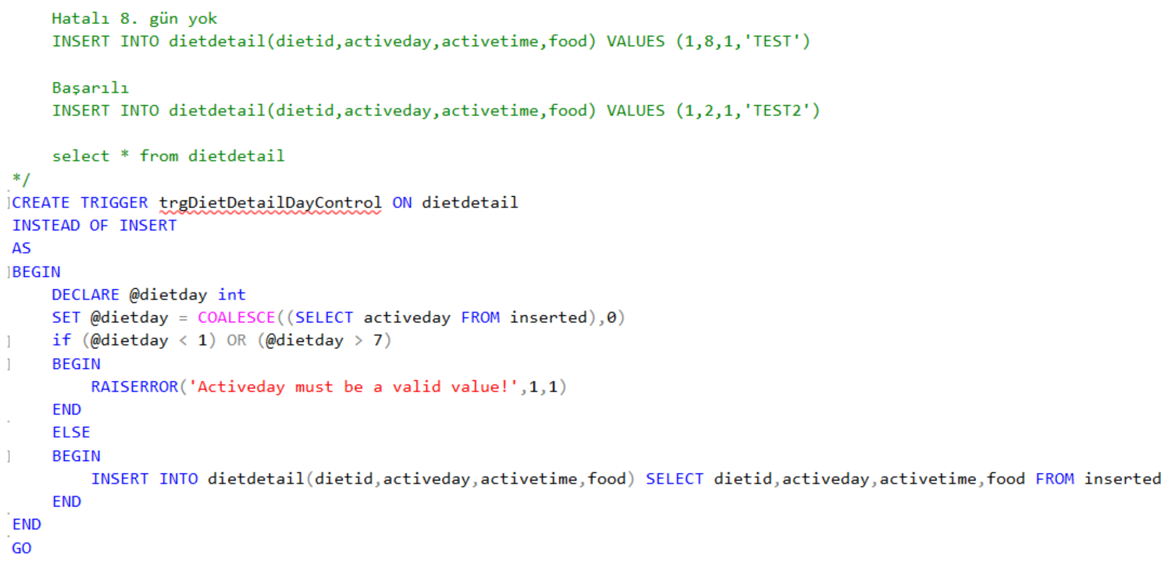
```
    UPDATE memberpayment SET amount=20 WHERE recordno=9

    SELECT * FROM member
    SELECT * FROM memberpayment
*/
CREATE TRIGGER trgMemberBalanceUpdate ON memberpayment
AFTER INSERT,UPDATE,DELETE
AS
BEGIN
    DECLARE @memberid int
    if EXISTS(select * from inserted)
    BEGIN
        SET @memberid = (SELECT memberid FROM inserted)
    END
    else if EXISTS(select * from deleted)
    BEGIN
        SET @memberid = (SELECT memberid FROM deleted)
    END

    if @memberid IS NOT NULL
        UPDATE member SET balance=(SELECT SUM(processtype * amount) FROM memberpayment where memberid=@memberid) where recordno=@memberid
END
GO
```

8.4 trgDietDetailDayControl Trigger:

While entering the diet's day information and meal information into the dietdetail table, it checks whether these information are valid.



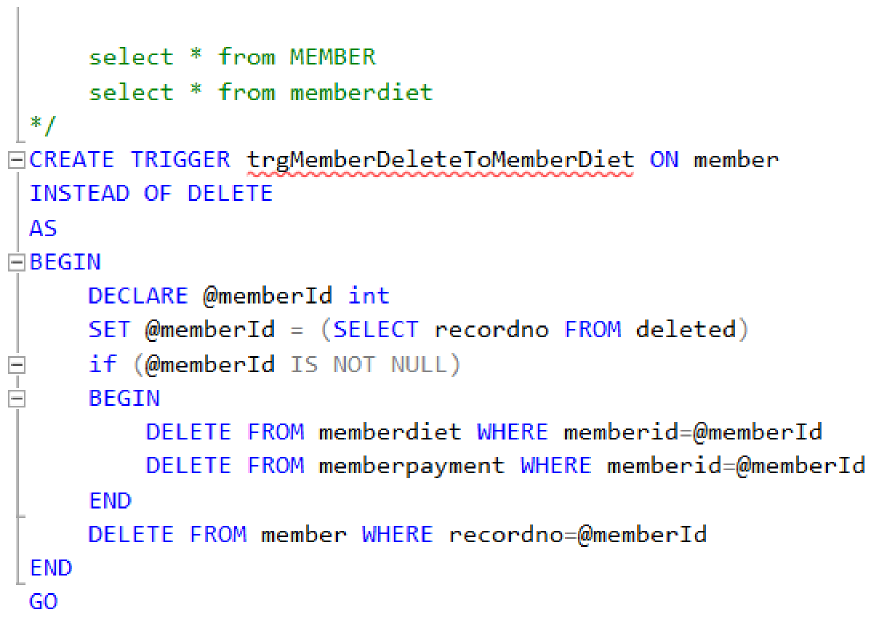
```
    Hatalı 8. gün yok
    INSERT INTO dietdetail(dietid,activeday,activetime,food) VALUES (1,8,1,'TEST')

    Başarılı
    INSERT INTO dietdetail(dietid,activeday,activetime,food) VALUES (1,2,1,'TEST2')

    select * from dietdetail
*/
CREATE TRIGGER trgDietDetailDayControl ON dietdetail
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @dietday int
    SET @dietday = COALESCE((SELECT activeday FROM inserted),0)
    if (@dietday < 1) OR (@dietday > 7)
    BEGIN
        RAISERROR('Activeday must be a valid value!',1,1)
    END
    ELSE
    BEGIN
        INSERT INTO dietdetail(dietid,activeday,activetime,food) SELECT dietid,activeday,activetime,food FROM inserted
    END
END
GO
```

8.5 trgMemberDeleteToMemberDiet Trigger:

Before a member is deleted from the member table, if a member's diet is assigned, they must be deleted. In addition, if there is a payment information belonging to the member, that information should be deleted. In such a case, firstly, deletion is done from the memberdiet and memberpayment tables. Then the member is deleted from the member table.

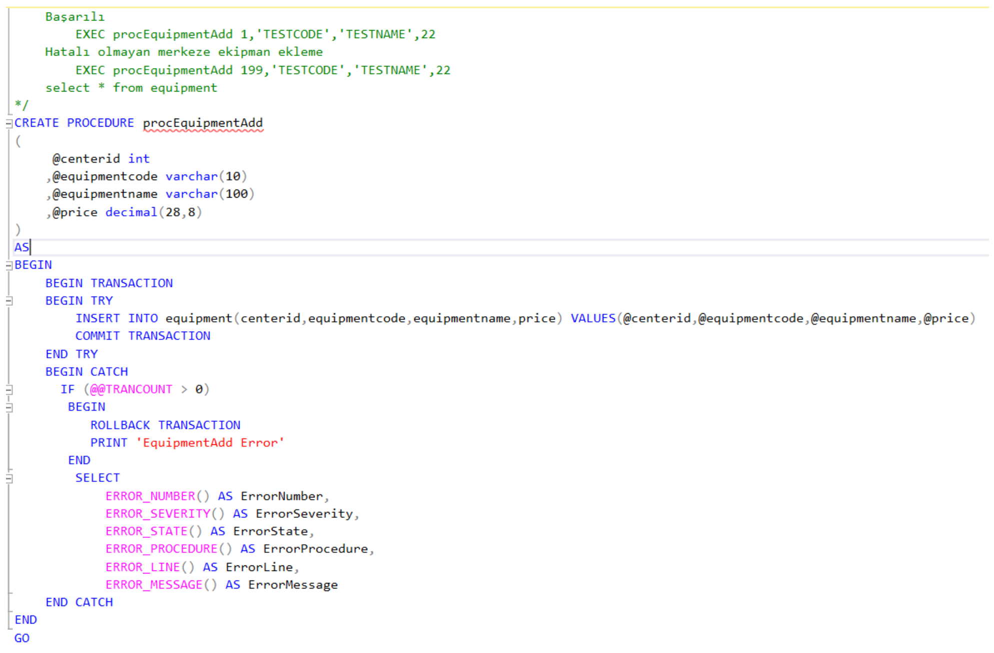


```sql
    select * from MEMBER
    select * from memberdiet
*/
CREATE TRIGGER trgMemberDeleteToMemberDiet ON member
INSTEAD OF DELETE
AS
BEGIN
    DECLARE @memberId int
    SET @memberId = (SELECT recordno FROM deleted)
    if (@memberId IS NOT NULL)
    BEGIN
        DELETE FROM memberdiet WHERE memberid=@memberId
        DELETE FROM memberpayment WHERE memberid=@memberId
    END
    DELETE FROM member WHERE recordno=@memberId
END
GO
```

9. ATOMIC TRANSACTIONS:

9.1 procEquipment Transaction:

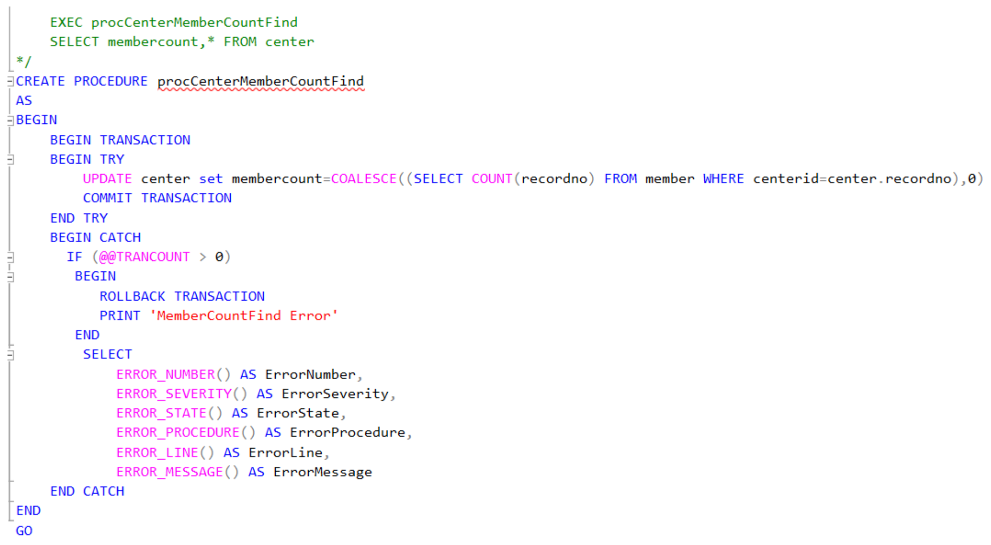
It is the transaction used when adding equipment to a sports center. If the equipment is wanted to be added to a sports center that does not exist, the rollback process is applied, otherwise the commit process is made.



```sql
    Başarılı
        EXEC procEquipmentAdd 1,'TESTCODE','TESTNAME',22
    Hatalı olmayan merkeze ekipman ekleme
        EXEC procEquipmentAdd 199,'TESTCODE','TESTNAME',22
    select * from equipment
*/
CREATE PROCEDURE procEquipmentAdd
(
     @centerid int
    ,@equipmentcode varchar(10)
    ,@equipmentname varchar(100)
    ,@price decimal(28,8)
)
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        INSERT INTO equipment(centerid,equipmentcode,equipmentname,price) VALUES(@centerid,@equipmentcode,@equipmentname,@price)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      IF (@@TRANCOUNT > 0)
       BEGIN
          ROLLBACK TRANSACTION
          PRINT 'EquipmentAdd Error'
       END
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
END
GO
```

9.2 procCenterMemberCountFind Transaction:

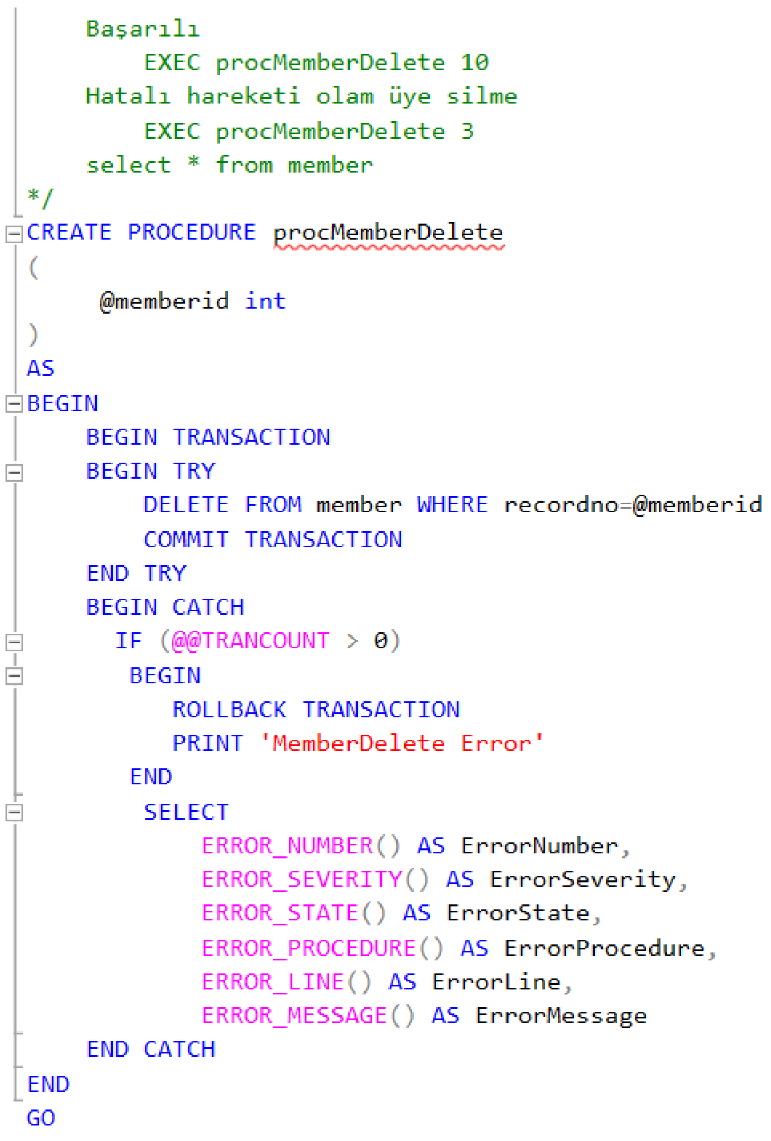
It is the transaction used when updating the number of members of sports centers. If any error is received during the update process, the rollback process is applied, otherwise the commit is performed.



```sql
    EXEC procCenterMemberCountFind
    SELECT membercount,* FROM center
*/
CREATE PROCEDURE procCenterMemberCountFind
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        UPDATE center set membercount=COALESCE((SELECT COUNT(recordno) FROM member WHERE centerid=center.recordno),0)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      IF (@@TRANCOUNT > 0)
       BEGIN
          ROLLBACK TRANSACTION
          PRINT 'MemberCountFind Error'
       END
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
END
GO
```

9.3 procMemberDelete Transaction:

It is the transaction used when deleting a member from the member table. If the member to be deleted does not exist in the table or if any error is received during the deletion process, the rollback process is applied, otherwise the commit is performed.



```
    Başarılı
        EXEC procMemberDelete 10
    Hatalı hareketi olam üye silme
        EXEC procMemberDelete 3
    select * from member
*/
CREATE PROCEDURE procMemberDelete
(
     @memberid int
)
AS
BEGIN
    BEGIN TRANSACTION
    BEGIN TRY
        DELETE FROM member WHERE recordno=@memberid
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
      IF (@@TRANCOUNT > 0)
       BEGIN
          ROLLBACK TRANSACTION
          PRINT 'MemberDelete Error'
       END
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
END
GO
```