```
CREATE TABLE PERSON(
PER_ID int primary key CHECK (PER_ID >= 1000),
PER_EM varchar(25) NOT NULL CHECK ((PER_EM NOT LIKE '%[!-@]%') AND (PER_EM NOT LIKE
'%[[-`]%') AND (PER_EM NOT LIKE '%[{-~]%')),
PER_MB varchar(25) NOT NULL CHECK ((PER_MB NOT LIKE '%[!-@]%') AND (PER_MB NOT LIKE
'%[[-`]%') AND (PER_MB NOT LIKE '%[{-~]%')),
PER_GJIN varchar(1) NOT NULL CHECK (PER_GJIN IN ('M','F')),
PER_DL date NOT NULL ,
PER_NR char(13) NOT NULL CHECK (PER_NR LIKE '+3556[7-9][2-9][0-9][0-9][0-9][0-9][0-
9][0-9]'),
PER_EMAIL varchar(225) NOT NULL CHECK (PER_EMAIL LIKE '%@%'),
PER_PUN BIT NOT NULL,
PER_PAC BIT NOT NULL,
CONSTRAINT PUN_PAC CHECK (PER_PUN = 1 OR PER_PAC = 1));

CREATE TABLE PACIENT
(PER_ID  INT NOT NULL UNIQUE,
PAC_ALGJ varchar(100),
PAC_SHEN varchar(100),
PRIMARY KEY (PER_ID),
FOREIGN KEY (PER_ID) REFERENCES PERSON(PER_ID));

CREATE TABLE PUNE(
PUNE_ID TINYINT PRIMARY KEY CHECK (PUNE_ID > 0),
PUNE_TIT VARCHAR(30) NOT NULL,
PUNE_PAGE_MUJ DECIMAL(10,2) CHECK (PUNE_PAGE_MUJ >= 0));


CREATE TABLE PUNONJES(
PER_ID INT NOT NULL UNIQUE,
PUN_ADRES varchar (20),
PUN_QYTET char(3) DEFAULT 'TIR' NOT NULL CHECK (PUN_QYTET IN
('TIR','DUR','BER','SHK','XXX')),
PUN_LAB CHAR(1) NOT NULL CHECK (PUN_LAB IN ('Y','N')),
PUNE_ID TINYINT NOT NULL REFERENCES PUNE(PUNE_ID),
PRIMARY KEY (PER_ID),
FOREIGN KEY (PER_ID) REFERENCES PERSON(PER_ID));


CREATE TABLE PUNE_HISTORIK(
PER_ID INT NOT NULL,
PUNE_ID TINYINT NOT NULL,
PH_DT_FILLIM DATE NOT NULL,
PH_DT_PERFUNDIM DATE,
PRIMARY KEY (PER_ID, PUNE_ID, PH_DT_FILLIM),
FOREIGN KEY (PER_ID) REFERENCES PUNONJES(PER_ID),
FOREIGN KEY (PUNE_ID) REFERENCES PUNE(PUNE_ID),
CONSTRAINT PU_HIST1 UNIQUE(PER_ID, PUNE_ID, PH_DT_FILLIM));



CREATE TRIGGER TR_PUN_HIST
ON PUNONJES AFTER INSERT, UPDATE
AS
BEGIN
DECLARE
@COUNT_PUNE_CHANGE INT;
      IF EXISTS (SELECT * FROM DELETED) BEGIN --nqs kemi update
            SELECT @COUNT_PUNE_CHANGE = COUNT(I.PER_ID) FROM INSERTED I, DELETED D
```

```
            WHERE I.PER_ID = D.PER_ID AND I.PUNE_ID != D.PUNE_ID;
            SELECT @COUNT_PUNE_CHANGE AS 'CHANGED PUN IDS';

            IF @COUNT_PUNE_CHANGE > 0 BEGIN --if ndryshime ne llojin e punes
                  --update daten e perf te hist se meparshem te punonjesit
                  UPDATE PUNE_HISTORIK SET PH_DT_PERFUNDIM = GETDATE()
                  WHERE PER_ID IN (SELECT PER_ID FROM INSERTED)
                  AND PH_DT_PERFUNDIM IS NULL;
            END;
      END;
      --insert & update inserto te hist, punen e re te punonjesit
      INSERT INTO PUNE_HISTORIK (PER_ID, PUNE_ID, PH_DT_FILLIM)
      SELECT PER_ID, PUNE_ID, GETDATE() AS PH_DT_FILLIM
      FROM INSERTED;
END;


CREATE TABLE ARSIM (
ARS_ID TINYINT NOT NULL PRIMARY KEY CHECK (ARS_ID > 0),
ARS_FUSHA VARCHAR(30),
ARS_NIVELI varchar(30),
);


CREATE TABLE KUALIFIKIM
(
PER_ID INT NOT NULL,
ARS_ID TINYINT NOT NULL,
KL_DT_FILLIM DATE NOT NULL,
KL_DT_PERFUNDIM DATE,
KL_INSTITUCIONI VARCHAR(30) NOT NULL,
PRIMARY KEY (PER_ID, ARS_ID),
FOREIGN KEY (PER_ID) REFERENCES PUNONJES(PER_ID),
FOREIGN KEY (ARS_ID) REFERENCES ARSIM(ARS_ID),
CONSTRAINT KLF1 UNIQUE(PER_ID, ARS_ID));


CREATE TABLE LABORANT
(PER_ID  INT NOT NULL,
LBR_SPEC varchar(30),
LBR_DISP CHAR(1) NOT NULL CHECK (LBR_DISP IN ('Y','N')),
PRIMARY KEY (PER_ID),
FOREIGN KEY (PER_ID) REFERENCES PUNONJES(PER_ID));


CREATE TABLE TAKIM (
TKM_ID  INT PRIMARY KEY CHECK (TKM_ID > 0),
TKM_DT DATE NOT NULL,
TKM_KRYER BIT DEFAULT 0,
TKM_SHEN varchar(100),
PER_ID  INT NOT NULL REFERENCES PACIENT(PER_ID));

CREATE TABLE MOSTER_INFO (
MSR_INFO_ID  SMALLINT NOT NULL PRIMARY KEY CHECK (MSR_INFO_ID > 0),
MSR_INFO_LLOJ VARCHAR(30) NOT NULL,
MSR_INFO_JETGJATESI_ORE SMALLINT NOT NULL);


CREATE TABLE MOSTER
```

```
(PAC_ID  INT NOT NULL UNIQUE, --check e trasheguar
LBR_ID INT NOT NULL UNIQUE,
MSR_DT DATETIMEOFFSET(0) DEFAULT GETDATE() NOT NULL,
MSR_INFO_ID SMALLINT NOT NULL REFERENCES MOSTER_INFO(MSR_INFO_ID),
PRIMARY KEY (PAC_ID, LBR_ID, MSR_DT),
FOREIGN KEY (PAC_ID) REFERENCES PACIENT(PER_ID),
FOREIGN KEY (LBR_ID) REFERENCES LABORANT(PER_ID),
CONSTRAINT MSR1 UNIQUE(PAC_ID, LBR_ID, MSR_DT));


CREATE TRIGGER TR_TAKIM_KRYER
ON MOSTER AFTER INSERT --insert te mostra table do te thot u krye takimi
AS BEGIN
UPDATE TAKIM SET TKM_KRYER = 1 --update sinjalizuesin
WHERE TAKIM.PER_ID IN
            (SELECT I.PAC_ID FROM INSERTED I)
            AND TKM_DT = (SELECT MAX(T.TKM_DT) FROM TAKIM T --takimin me te fundit
per pacientin pasi mund tket shum
                                    WHERE T.PER_ID IN (SELECT I1.PAC_ID FROM
INSERTED I1));
END;



CREATE TABLE ANALIZE (
ANZ_ID  SMALLINT NOT NULL PRIMARY KEY CHECK (ANZ_ID > 0),
ANZ_EM VARCHAR(30) NOT NULL,
ANZ_LLOJI VARCHAR(30) NOT NULL,
ANZ_PER VARCHAR(200) NOT NULL,
ANZ_ZGJAT_ORE SMALLINT NOT NULL,
ANZ_CMIM NUMERIC(10,2) NOT NULL CHECK (ANZ_CMIM > 0));


CREATE TABLE ANALIZE_HISTORIK (
ANZ_ID  SMALLINT NOT NULL,
AH_DT_FILLIM DATE NOT NULL,
AH_DT_PERFUNDIM DATE,
AH_CMIM NUMERIC(10,2) NOT NULL CHECK (AH_CMIM > 0),
PRIMARY KEY (ANZ_ID, AH_DT_FILLIM),
FOREIGN KEY (ANZ_ID) REFERENCES ANALIZE(ANZ_ID),
CONSTRAINT ANZ_HIST1 UNIQUE(ANZ_ID, AH_DT_FILLIM));

CREATE TRIGGER TR_ANZ_HIST
ON ANALIZE AFTER INSERT, UPDATE
AS
BEGIN
DECLARE
@COUNT_CMIM_CHANGE INT;

      IF EXISTS (SELECT * FROM DELETED) BEGIN --nqs update
            SELECT @COUNT_CMIM_CHANGE = COUNT(I.ANZ_ID) FROM INSERTED I, DELETED D
            WHERE I.ANZ_ID = D.ANZ_ID AND I.ANZ_CMIM != D.ANZ_CMIM;

            SELECT @COUNT_CMIM_CHANGE AS 'CHANGED CMIM';

            IF @COUNT_CMIM_CHANGE > 0 BEGIN --nqs kemi ndryshim te cmimit
                  --vendos dt perf te hist se meparshme te kesaj analize
                  UPDATE ANALIZE_HISTORIK SET AH_DT_PERFUNDIM = GETDATE()
                  WHERE ANZ_ID IN (SELECT ANZ_ID FROM INSERTED)
```

```
                AND AH_DT_PERFUNDIM IS NULL;
            END;
      END;
      --insert & update, shto te historiku cmimin e ri pa dat perfundimi
      INSERT INTO ANALIZE_HISTORIK (ANZ_ID, AH_DT_FILLIM, AH_CMIM)
      SELECT ANZ_ID, GETDATE() AS AH_DT_FILLIM, ANZ_CMIM
      FROM INSERTED;
END;


CREATE TABLE RAPORT (
RAP_ID  INT PRIMARY KEY CHECK (RAP_ID > 0),
RAP_DT DATE NOT NULL,
RAP_PER VARCHAR(300));


CREATE TABLE FATURE
(FAT_ID  INT NOT NULL UNIQUE CHECK (FAT_ID > 0),
FAT_DT DATETIMEOFFSET(0) DEFAULT GETDATE() NOT NULL,
FAT_TOT DECIMAL(14,4) NOT NULL CHECK (FAT_TOT > 0),
PER_ID  INT NOT NULL REFERENCES PACIENT(PER_ID));


CREATE TABLE FATURE_RRESHT(
FAT_ID INT NOT NULL REFERENCES FATURE(FAT_ID),
ANZ_ID  SMALLINT NOT NULL REFERENCES ANALIZE(ANZ_ID),
PAC_ID  INT NOT NULL UNIQUE,
LBR_ID INT NOT NULL UNIQUE,
MSR_DT DATETIMEOFFSET(0) NOT NULL,
RAP_ID INT NOT NULL REFERENCES RAPORT(RAP_ID),
PRIMARY KEY (FAT_ID, ANZ_ID),
FOREIGN KEY (PAC_ID, LBR_ID, MSR_DT) REFERENCES MOSTER (PAC_ID, LBR_ID, MSR_DT),
CONSTRAINT FAT_RRE1 UNIQUE(FAT_ID, ANZ_ID));
```