DATABASE MANAGEMENT SYSTEMS PROJECT

Scenario

A doctor examines different patients. A doctor can write prescriptions to patients. There is a nurse attached to each doctor. People are also inherited. A software system is created for a hospital. The manager wants to store and manage some information such as doctors, patients, prescriptions in the hospital. The database is expected to contain information about people, doctors, patients, nurses, drugs, prescriptions, and appointments.

Business Rules

-Each doctor has name,e-mail,telephonenumber,specialty.

-Prescription has an id.

-A drug has an id, name, drug type(syrup, pill,injection).

-Appointment has an ID ,appointment date, description.

-Each patient has a name, e-mail, telephone number.

-Each room has an id, name.

-Each clinic has an id, name.

-Each nurse has an name, e-mail, phone number.

-Inthesystem,eachpersonhaspersonid,name,e-mail,telephonenumberinformation.Also,apersoncan exist in just three categories: a doctor, a nurse and a patient. One person can not be all three. Peopleareidistinguishedfromeachotherbytheirid.

-Adoctorcanmakemorethanoneappointment.Onlyonedoctorcanbevisitedforanappointment.

-Aprescriptioniscreated byadoctor.Adoctormaynot writeaprescription ormaywritemorethan one prescription.

-Adrugmaybeprescribedmorethanonce,andaprescriptionmaycontainmorethanonedrug.

-A patient can be at an appointment, but a patient can have more than one appointment.

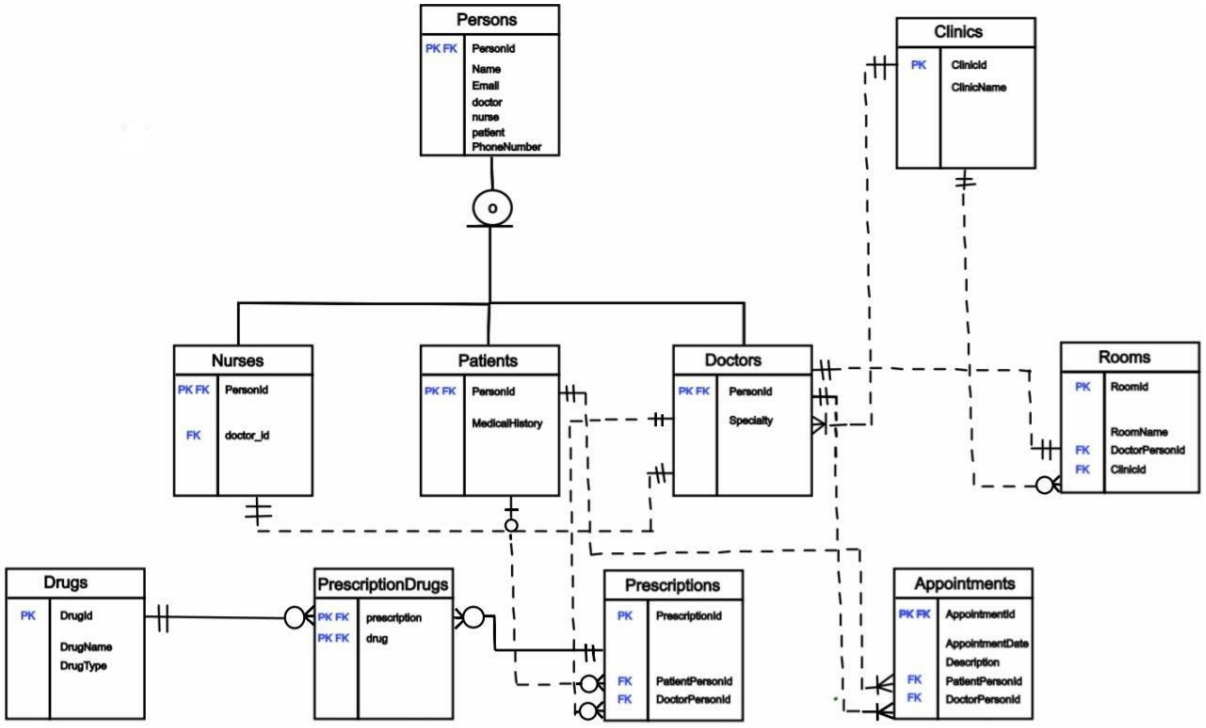
-A patient may receive no prescription a tall, or may receive a prescription. A prescription may be written for a patient, or may not be written at all.

-A room has only one doctor. A doctor has only one room.

-A clinic must have at least one doctor, but there can also be more than one doctor but a doctor can only be at a clinic.

-A nurse can only work with one doctor, a doctor can only work with one nurse.

-There are one or more rooms in a clinic. A room be longs to only one clinic.



Relational Schema (Textual Representation):

- Persons(PersonId:bigint,Name:varchar(30),Email:varchar(50),PhoneNumber:bigint, doctor:bool, nurse:bool, patient:bool)
- Nurses(PersonId:bigint,DoctorId:bigint)
- Patients(PersonId:bigint,MedicalHistory:varchar(100))
- Doctors(PersonId:bigint,Speciality:varchar(30))
- Clinics(ClinicId:bigint,ClinicName:varchar(30))
- Rooms(RoomId:bigint,RoomName:varchar(30),DoctorPersonId:bigint,ClinicId:bigint)
- Appointments(AppoinmentId:bigint,AppoinmentDate:timestamp,Description:VARCHAR(60), PatientPersonId:bigint, DoctorPersonId:bigint)
- Prescriptions(PrescriptionId:bigint,PatientPersonId:bigint,DoctorPersonId:bigint)
- PrescriptionDrugs(Prescription:bigint,Drug:bigint)
- Drugs(DrugId:bigint,DrugName:varchar(50),DrugType:varchar(30))

SQLstatementstocreatethedatabasewiththedatainit
--CreationProcesses

```
CREATETABLEIFNOTEXISTS"EFMigrationsHistory"( "MigrationId"
        character varying(150) NOT NULL, "ProductVersion" character
        varying(32) NOT NULL,
        CONSTRAINT"PK___EFMigrationsHistory"PRIMARYKEY("MigrationId")
    );

CREATETABLE"Clinics"(
        "ClinicId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "ClinicName" VARCHAR(30) NOT NULL,
        CONSTRAINT"PK_Clinic"PRIMARYKEY("ClinicId")
    );

CREATETABLE"Drugs"(
        "DrugId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "DrugName" VARCHAR(50) NOT NULL,
        "DrugType" VARCHAR(30) NOT NULL,
        CONSTRAINT"PK_Drug"PRIMARYKEY("DrugId")
    );

CREATETABLE"Persons"(
        "PersonId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "Name" VARCHAR(30) NOT NULL,
        "Email"VARCHAR(50)NOTNULL,
        "PhoneNumber"textNOTNULL,
        doctor boolean NOT NULL,
        patient boolean NOT NULL,nurse
        boolean NOT NULL,
        CONSTRAINT"PK_Person"PRIMARYKEY("PersonId")
    );

CREATETABLE"Doctors"(
        "PersonId" bigint NOT NULL,
        "Specialty"VARCHAR(30)NOTNULL,
        "ClinicId"bigintNOTNULL,
        CONSTRAINT"PK_Doctor"PRIMARYKEY("PersonId"),
        CONSTRAINT"FK_Doctor_Person_PersonId"FOREIGNKEY("PersonId")REFERENCES
"Persons" ("PersonId"),
        CONSTRAINT"FK_Doctor_Clinic_ClinicId"FOREIGNKEY("ClinicId")REFERENCES "Clinics"
("ClinicId")
    );

CREATE TABLE "Patients"
        ( "PersonId"bigintNOTNULL,
        "MedicalHistory" VARCHAR(100) NOT NULL,
        CONSTRAINT"PK_Patient"PRIMARYKEY("PersonId"),
        CONSTRAINT"FK_Patient_Person_PersonId"FOREIGNKEY("PersonId")REFERENCES
"Persons" ("PersonId")
    );

CREATETABLE"Nurses"(
        "PersonId" bigint NOT NULL,
        "DoctorPersonId"bigintNOTNULL,
        CONSTRAINT"PK_Nurse"PRIMARYKEY("PersonId"),
        CONSTRAINT"FK_Nurse_Person_PersonId"FOREIGNKEY("PersonId")REFERENCES
```

```
"Persons"("PersonId"),
        CONSTRAINT"FK_Nurse_Doctor_DoctorPersonId"FOREIGNKEY("DoctorPersonId")
REFERENCES "Doctors" ("PersonId")
    );

CREATETABLE"Rooms"(
        "RoomId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "RoomName" VARCHAR(30) NOT NULL,
        "DoctorPersonId"bigintNOTNULL,
        "ClinicId" bigint NOT NULL,
        CONSTRAINT"PK_Room"PRIMARYKEY("RoomId"),
        CONSTRAINT"FK_Room_Clinic_ClinicId"FOREIGNKEY("ClinicId")REFERENCES "Clinics"
("ClinicId"),
        CONSTRAINT"FK_Room_Doctor_DoctorPersonId"FOREIGNKEY("DoctorPersonId")
REFERENCES "Doctors" ("PersonId")
    );

CREATETABLE"Appointments"(
        "AppointmentId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "AppointmentDate" timestamp with time zone NOT NULL,
        "Description" VARCHAR(60) NOT NULL,
        "PatientPersonId"bigintNOTNULL,
        "DoctorPersonId"bigintNOTNULL,
        CONSTRAINT"PK_Appointment"PRIMARYKEY("AppointmentId"),
        CONSTRAINT"FK_Appointment_Doctor_DoctorPersonId"FOREIGNKEY("DoctorPersonId")
REFERENCES "Doctors" ("PersonId"),
        CONSTRAINT"FK_Appointment_Patient_PatientPersonId"FOREIGNKEY("PatientPersonId")
REFERENCES "Patients" ("PersonId")
    );

CREATETABLE"Prescriptions"(
        "PrescriptionId"bigintGENERATEDBYDEFAULTASIDENTITY,
        "DoctorPersonId" bigint NOT NULL,
        "PatientPersonId"bigintNOTNULL,
        CONSTRAINT"PK_Prescription"PRIMARYKEY("PrescriptionId"),
        CONSTRAINT"FK_Prescription_Doctor_DoctorPersonId"FOREIGNKEY("DoctorPersonId")
REFERENCES "Doctors" ("PersonId"),
        CONSTRAINT"FK_Prescription_Patient_PatientPersonId"FOREIGNKEY("PatientPersonId")
REFERENCES "Patients" ("PersonId")
    );

CREATETABLE"PrescriptionDrugs"( p
        rescription bigint NOT NULL,
        drug bigint NOT NULL,
        CONSTRAINT"PK_PrescriptionDrug"PRIMARYKEY(prescription,drug),
        CONSTRAINT"FK_PrescriptionDrug_Drug_drug"FOREIGNKEY(drug)REFERENCES "Drugs"
("DrugId"),
        CONSTRAINT"FK_PrescriptionDrug_Prescription_prescription"FOREIGNKEY(prescription)
REFERENCES "Prescriptions" ("PrescriptionId")
    );


                --INSERTIONPROCESSES

 --Clinics
INSERTINTO"Clinics"("ClinicName")VALUES ('General
Medicine Clinic'),
('Pediatrics Clinic'),
('Cardiology Center'),
('DermatologyClinic'),
```

('Neurology Institute'),
('Orthopedic Hospital'),
('OncologyDepartment'),
('ENT Clinic'),
('Psychiatry Clinic'),
('GastroenterologyUnit');

--Drugs
INSERTINTO"Drugs"("DrugName","DrugType")VALUES
('Drug 5', 'Tablet'),
('Drug9','Injection'),
('Drug10','Syrup'),
('Drug8','Capsule'),
('Drug5','Tablet'),
('Drug1','Tablet'),
('Drug3','Injection'),
('Drug3','Syrup'),
('Drug8','Capsule'),
('Drug9','Injection');

--Persons
INSERTINTO"Persons"("Name","Email","PhoneNumber",doctor,patient,nurse)VALUES ('Person 1',
'person1@example.com','1234567891', True, False, False),
('Person 2', 'person2@example.com','1234567892', True, False, False),
('Person 3', 'person3@example.com','1234567893', True, False, False),
('Person 4', 'person4@example.com','1234567894', False, False, True),
('Person 5', 'person5@example.com','1234567895', False, True, False),
('Person 6', 'person6@example.com','1234567896', True, False, False),
('Person 7', 'person7@example.com','1234567897', False, False, True),
('Person 8', 'person8@example.com','1234567898', False, False, True),
('Person 9', 'person9@example.com','1234567899', False, False, True),
('Person10','person10@example.com','12345678910',False,False,True);

--Doctors
INSERTINTO"Doctors"("PersonId","Specialty","ClinicId")VALUES (21,
'Dermatology', 7),
(22,'Dermatology',7),
(23,'Dermatology',10),
(24,'Pediatrics', 7),
(25,'Dermatology',10);

--Patients
INSERTINTO"Patients"("PersonId","MedicalHistory")VALUES (26,
'History 9'),
(27,'History1'),
(28,'History2'),
(29,'History9'),
(30,'History7');

--Nurses
INSERTINTO"Nurses"("PersonId","DoctorPersonId")VALUES (6,
5),
(7, 1),
(8, 4),
(9, 1),
(10, 2);

--Rooms
INSERTINTO"Rooms"("RoomName","DoctorPersonId","ClinicId")VALUES
('Room 3', 5, 7),

('Room4',4,5),
('Room4',5,10),
('Room8',3,7),
('Room3',2,3),
('Room2',3,4),
('Room3',5,3),
('Room4',2,8),
('Room4',1,4),
('Room3',3,7);

--Appointments
INSERTINTO"Appointments"("AppointmentDate","Description","PatientPersonId", "DoctorPersonId")
VALUES
(NOW()+INTERVAL'1days','Description4',7,3),
(NOW()+INTERVAL'2days','Description6',10,3),
(NOW()+INTERVAL'3days','Description2',7,3),
(NOW()+INTERVAL'4days','Description10',10,5),
(NOW()+INTERVAL'5days','Description7',7,4),
(NOW()+INTERVAL'6days','Description9',9,1),
(NOW()+INTERVAL'7days','Description6',8,3),
(NOW()+INTERVAL'8days','Description6',10,1),
(NOW()+INTERVAL'9days','Description2',9,3),
(NOW()+INTERVAL'10days','Description7',7,5);

--Prescriptions
INSERTINTO"Prescriptions"("DoctorPersonId","PatientPersonId")VALUES (1,
6),
(3, 9),
(4, 9),
(5, 9),
(1, 7),
(1, 9),
(3, 6),
(4, 6),
(3, 7),
(3, 10);

--PrescriptionDrugs
INSERTINTO"PrescriptionDrugs"(prescription,drug)VALUES (15,
14),
(12, 17),
(10, 15),
(19, 16),
(16, 14),
(17, 13),
(11, 17),
(12, 18),
(12, 19),
(15, 15);

//UPDATEProcess

--TRIGGERS

--TRIGER1

```
        --TrigerTABLE

CREATETABLE"AppointmentLog"( "LogId"
   SERIAL PRIMARY KEY,
   "AppointmentId"bigintNOTNULL,
   "LogMessage" text NOT NULL,
   "LogTime"timestampNOTNULLDEFAULTNOW()
);

CREATEORREPLACEFUNCTIONlog_appointment()
RETURNS TRIGGER AS $$
BEGIN
   INSERTINTO"AppointmentLog"("AppointmentId","LogMessage")
   VALUES (NEW."AppointmentId", 'New appointment added.');
   RETURN NEW;
END;
$$LANGUAGEplpgsql;

CREATETRIGGERafter_appointment_insert
BEFORE INSERT ON "Appointments"
FOREACHROW
EXECUTEFUNCTIONlog_appointment();


INSERTINTO"Persons"("Name","Email","PhoneNumber",doctor,patient,nurse)VALUES ('Person 11',
'person12@example.com','1234567891', False, True, False);

INSERTINTO"Persons"("Name","Email","PhoneNumber",doctor,patient,nurse)VALUES ('Person 12',
'person13@example.com','1234567891', True, False, False);


--Patient
INSERTINTO"Patients"("PersonId","MedicalHistory")VALUES (13,
'History 11');

--Doctor
INSERTINTO"Doctors"("PersonId","Specialty","ClinicId")VALUES (14,
'Dermatology', 7);

--Addanew appointment
INSERTINTO"Appointments"("AppointmentDate","Description","PatientPersonId",
"DoctorPersonId")
VALUES(NOW()+INTERVAL'2days','Routinecheck-up',13, 14);

-- Log tablosundaki kaydı kontrol et
SELECT*FROM"AppointmentLog";




--TRIGER2

CREATEORREPLACEFUNCTIONcascade_delete_patient_appointments()
RETURNS TRIGGER AS $$
BEGIN
   DELETEFROM"Appointments"WHERE"PatientPersonId"=OLD."PersonId"; RETURN
   OLD;
END;
$$LANGUAGEplpgsql;
```

```
CREATETRIGGERafter_patient_delete
BEFORE DELETE ON "Patients"
FOREACHROW
EXECUTEFUNCTIONcascade_delete_patient_appointments();

--Let'strytodeletethe"Patients"
DELETEFROM"Persons"WHERE"PersonId"=3;
DELETEFROM"Patients"WHERE"PersonId"=3;



--TRIGER3


        --TrigerTABLE

CREATETABLE"DoctorNameLog"( "LogId"
   SERIAL PRIMARY KEY,
   "DoctorPersonId" bigint NOT NULL,
   "OldName" VARCHAR(50) NOT NULL,
   "NewName"VARCHAR(50)NOTNULL,
   "ChangeTime"timestampNOTNULLDEFAULTNOW()
);


CREATEORREPLACEFUNCTIONlog_doctor_name_changes()
RETURNS TRIGGER AS $$
BEGIN
   --Ifthenamehaschanged,savetheoldandnewnamestothelogtable. IF
   NEW."Name" IS DISTINCT FROM OLD."Name" THEN
      INSERTINTO"DoctorNameLog"("DoctorPersonId","OldName","NewName","ChangeTime")
      VALUES (OLD."PersonId", OLD."Name", NEW."Name", NOW());
   END IF;
   RETURNNEW;
END;
$$LANGUAGEplpgsql;

CREATETRIGGERdoctor_name_update_trigger BEFORE
UPDATE ON "Persons"
FOREACHROW
WHEN(OLD."doctor"=TRUEANDNEW."doctor"=TRUE)
EXECUTEFUNCTIONlog_doctor_name_changes();

--Let's update
UPDATE"Persons"SET"Name"='Memati'WHERE"PersonId"=3;

--Let'scheckthelogtable
SELECT*FROM"DoctorNameLog";



--TRIGER4


        -- Triger

TABLECREATETABLE"ClinicR

eport"(
   "ReportId"SERIALPRIMARYKEY,
```

"ClinicId"bigintNOTNULL,

```
    "ClinicName"VARCHAR(50)NOTNULL,
    "LastUpdated"timestampNOTNULLDEFAULTNOW()
);



CREATEORREPLACEFUNCTIONupdate_clinic_reports()
RETURNS TRIGGER AS $$
BEGIN
    --Updatereporttableifclinicnameorinformationhas changed
    IFNEW."ClinicName"ISDISTINCTFROMOLD."ClinicName"THEN UPDATE
        "ClinicReport"
        SET"ClinicName"=NEW."ClinicName",
            "LastUpdated" = NOW()
        WHERE"ClinicId"=NEW."ClinicId";

    END IF;
    RETURNNEW;
END;
$$LANGUAGEplpgsql;

CREATETRIGGERclinic_update_trigger
BEFORE UPDATE ON "Clinics"
FOREACHROW
EXECUTEFUNCTIONupdate_clinic_reports();


--Let'supdatetheclinicname
UPDATE"Clinics"SET"ClinicName"='CityHealthCenter'WHERE"ClinicId"=6;

--Let'scheckttheclinicalreporttable
SELECT * FROM "ClinicReport";



--FUNCTIONS

--Function1

CREATE OR REPLACE FUNCTION get_patients_by_doctor(doctor_id BIGINT)
RETURNSTABLE(PersonIdBIGINT,PersonNameTEXT,MedicalHistoryTEXT)AS$$ BEGI
N
    RETURNQUERY
    SELECTp."PersonId",p."Name",pa."MedicalHistory" FROM
    "Persons" p
    JOIN"Patients"paONp."PersonId"= pa."PersonId"
    JOIN"Appointments"aONa."PatientPersonId"=pa."PersonId"
    WHERE a."DoctorPersonId" = doctor_id;
END;
$$LANGUAGEplpgsql;

SELECT*FROMget_patients_by_doctor(3);--3numaralıdoktoraaithastalar


--Function2

CREATEORREPLACEFUNCTIONget_rooms_by_clinic(clinic_idBIGINT)
RETURNS TABLE (RoomName TEXT, DoctorName TEXT) AS $$
BEGIN
    RETURNQUERY
```

```
   SELECTr."RoomName",p."Name"ASDoctorName
   FROM "Rooms" r
   JOIN"Doctors"dONr."DoctorPersonId"=d."PersonId"
   JOIN "Persons" p ON d."PersonId" = p."PersonId"
   WHERE r."ClinicId" = clinic_id;
END;
$$LANGUAGEplpgsql;

SELECT*FROMget_rooms_by_clinic(7);--7numaralıkliniğinodaları


--Function3

CREATEORREPLACEFUNCTIONget_drugs_by_patient(patient_idBIGINT)
RETURNS TABLE (DrugName TEXT, PrescriptionId BIGINT) AS $$
BEGIN
   RETURNQUERY
   SELECTd."DrugName",pd."prescription"ASPrescriptionId
   FROM "PrescriptionDrugs" pd
   JOIN"Prescriptions"pONpd."prescription"=p."PrescriptionId" JOIN
   "Drugs" d ON pd."drug" = d."DrugId"
   WHEREp."PatientPersonId"=patient_id;
END;
$$LANGUAGEplpgsql;

SELECT*FROMget_drugs_by_patient(6);--6numaralıhastanınilaçları


--Function4

CREATEORREPLACEFUNCTIONget_appointment_count_by_doctor(doctor_idBIGINT)
RETURNS INT AS $$
DECLARE
   appointment_countINT;
BEGIN
   SELECTCOUNT(*)
   INTOappointment_count
   FROM "Appointments"
   WHERE"DoctorPersonId"=doctor_id;

   RETURNappointment_count;
END;
$$LANGUAGEplpgsql;

SELECTget_appointment_count_by_doctor(3);--3numaralıdoktoruntoplamrandevuları
```

```
// Doctor Admin CRUD Processes
0 references
public IActionResult DoctorsIndex()
{
    var doctors = _context.Doctors
    .Include(p => p.Clinic)
    .Include(p => p.Person);
    return View(doctors);
}
```

```
[HttpGet]
0 references
public async Task<IActionResult> PersonsDelete(long? id)
{
    var person = await _context.Persons.FindAsync(id);
    if(person.doctor) {
        ViewBag.Index = "DoctorsIndex";
        ViewBag.aa = id;
    }
    if(person.patient) {
        ViewBag.Index = "PatientsIndex";
    }
    return View(person);
}

[HttpPost]
0 references
public async Task<IActionResult> PersonsDelete( long id)
{
    var person = await _context.Persons.FindAsync(id);
    var doctor = await _context.Doctors.FindAsync(id);
    var patient = await _context.Patients.FindAsync(id);
    Console.WriteLine( "aa");
    _context.Persons.Remove(person);
    await _context.SaveChangesAsync();

    if (person.doctor)
    {
        _context.Persons.Remove(person);
        _context.Doctors.Remove(doctor);
        return RedirectToAction("DoctorsIndex");
    }
    if (person.patient)
    {
        _context.Persons.Remove(person);
        _context.Patients.Remove(patient);
        return RedirectToAction("PatientsIndex");
    }
    return View(person);
}
```

```
[HttpGet]
0 references
public async Task<IActionResult> PersonsUpdate(long? id)
{
    var person = await _context.Persons.FindAsync(id);
    return View(person);
}

[HttpPost]
0 references
public async Task<IActionResult> PersonsUpdate(long id, Person person)
{

    person.PersonId = id;
    try
    {
        _context.Update(person);
        await _context.SaveChangesAsync();
    }
    catch (Exception)
    {
        throw;
    }
    if (person.doctor)
    {
        return RedirectToAction("DoctorsUpdate", new{ id = person.PersonId});
    }
    if (person.patient)
    {
        return RedirectToAction("PatientsUpdate", new{ id = person.PersonId});
    }

    return View(person);
}

// Person CRUD Processes

[HttpGet]
0 references
public IActionResult PersonsCreate()
{
    return View();
}

[HttpPost]
0 references
public IActionResult PersonsCreate(Person person)
{
    _context.Persons.Add(person);
    _context.SaveChanges();
    ViewBag.personid = person.PersonId;
    if (person.doctor)
    {
        return View("DoctorsCreate");
    }
    if (person.patient)
    {
        return View("PatientsCreate");
    }

    return View(person);
}
```