College of Science and Technology

Royal University of Bhutan

University Assignment 1

Year 2025

Advanced Database System - DIS404

Tashi Wangchuk
BE Information Technology
Fourth Year First Semester
02220173
Tutor: Mr. Pema Galey

ROYAL UNIVERSITY OF BHUTAN

COLLEGE OF SCIENCE AND TECHNOLOGY
PHUENTSHOLING: BHUTAN

PLAGIARISM DECLARATION FORM

Student Name: Tashi Wangchuk

Student No: 02220173

Module No and Title of the module: DIS404 – Advanced Database System

Section H2 of the Royal University of Bhutan's Wheel of Academic Law provides the following definition of academic dishonesty:

Section H2 of the Royal University of Bhutan's Wheel of Academic Law provides the following definition of academic dishonesty:

"Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

Collusion: the representation of a piece of unauthorized group work as the work of a single candidate.

Commissioning: submitting an assignment done by another person as the student's own work.

Duplication: the inclusion in coursework of material identical or substantially like material which has already been submitted for any other assessment within the University.

False declaration: making a false declaration to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.

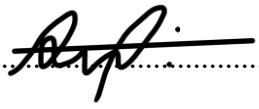
Falsification of data: presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which have been invented, altered or copied by the student.

Plagiarism: the unacknowledged use of another's work as if it were one's own.

Examples are verbatim copying another's work without acknowledgement paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement ideas or intellectual data in any form presented as one's own without acknowledging the source(s) making significant use of unattributed digital images such as graphs, tables, photographs. taken from test books, articles, films, plays, handouts, internet, or any other source, whether published or unpublished submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work use of any material without prior permission of copyright from appropriate authority or owner of the materials used"

Student Declaration.

I confirm that I have read and understood the above definitions of academic dishonesty. I declare that I have not committed any academic dishonesty when completing the attached piece of work.

Signature of Student:.....  ..... Date: 17/04/2024

Hospital Management System

The chosen organization is a mid-sized hospital that handles patient registrations, doctor assignments, appointment scheduling, and medical record tracking. The database ensures data integrity, supports queries for reporting (e.g., patient history, doctor availability), and integrates with applications for real time access.

Database Schema:

Below is the relational schema for the HMS database. Primary keys (PK) and foreign keys (FK) are indicated.

I. Patient
- patient_id (INT, PK, AUTO_INCREMENT)
- name (VARCHAR(100), NOT NULL)
- dob (DATE)
- gender (ENUM('Male', 'Female', 'Other'))
- address (VARCHAR(255))
- phone (VARCHAR(15), UNIQUE)

II. Doctor
- doctor_id (INT, PK, AUTO_INCREMENT)
- name (VARCHAR(100), NOT NULL)
- specialty (VARCHAR(50))
- phone (VARCHAR(15), UNIQUE)

III. Appointment
- appointment_id (INT, PK, AUTO_INCREMENT)
- patient_id (INT, FK references Patient(patient_id), ON DELETE CASCADE)
- doctor_id (INT, FK references Doctor(doctor_id), ON DELETE SET NULL)
- appointment_date (DATE, NOT NULL)
- appointment_time (TIME, NOT NULL)
- status (ENUM('Scheduled', 'Completed', 'Cancelled'), DEFAULT 'Scheduled')

IV. Medical Record
- record_id (INT, PK, AUTO_INCREMENT)
- patient_id (INT, FK references Patient(patient_id), ON DELETE CASCADE)

- ❖ doctor_id (INT, FK references Doctor(doctor_id), ON DELETE SET NULL)
- ❖ diagnosis (TEXT)
- ❖ treatment (TEXT)
- ❖ record_date (DATE, NOT NULL)

Created the database based on above database schema using the SQL script (Postgres SQL). Insert ed 15 sample data for testing (15 patients, 15 doctors, 15 appointments, 15 medical records). The implementation of the database tables and relationships are as follows:

Patient

Query | Query History

```sql
CREATE TYPE gender_enum AS ENUM('Male', 'Female', 'Other');

CREATE TABLE patient(
    patient_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    dob DATE,
    gender gender_enum,
    address VARCHAR(255),
    phone VARCHAR(15) UNIQUE

);
```

```sql
INSERT INTO patient (name, dob, gender, address, phone) VALUES
('Tashi Wangchuk', '1985-04-12', 'Male', 'Thimphu, Bhutan', '17100001'),
('Pema Choden', '1990-06-25', 'Female', 'Paro, Bhutan', '17100002'),
('Kezang Dorji', '1978-12-03', 'Male', 'Phuentsholing, Bhutan', '17100003'),
('Sonam Wangmo', '1988-09-15', 'Female', 'Thimphu, Bhutan', '17100004'),
('Sangay Tshering', '2000-01-20', 'Male', 'Trongsa, Bhutan', '17100005'),
('Chimi Dorji', '1995-05-10', 'Female', 'Wangdue, Bhutan', '17100006'),
('Dorji Wangchuk', '1982-03-22', 'Male', 'Bumthang, Bhutan', '17100007'),
('Dechen Wangmo', '1998-11-11', 'Female', 'Mongar, Bhutan', '17100008'),
('Kinley Dorji', '1975-07-30', 'Male', 'Gelegphu, Bhutan', '17100009'),
('Tshering Pem', '1992-08-05', 'Female', 'Trashigang, Bhutan', '17100010'),
('Tandin Wangchuk', '1987-02-14', 'Male', 'Paro, Bhutan', '17100011'),
('Sangay Choden', '2001-12-01', 'Female', 'Thimphu, Bhutan', '17100012'),
('Jigme Dorji', '1993-06-17', 'Male', 'Haa, Bhutan', '17100013'),
('Kezang Wangmo', '1989-04-29', 'Female', 'Lhuntse, Bhutan', '17100014'),
('Phuntsho Tshering', '1980-10-10', 'Male', 'Samdrup Jongkhar, Bhutan', '17100015');

SELECT * FROM patient;
```

Data Output | Messages | Notifications

SQL

	patient_id [PK] integer	name character varying (100)	dob date	gender gender_enum	address character varying (255)	phone character varying (15)
1	1	Tashi Wangchuk	1985-04-...	Male	Thimphu, Bhutan	17100001
2	2	Pema Choden	1990-06-...	Female	Paro, Bhutan	17100002
3	3	Kezang Dorji	1978-12-...	Male	Phuentsholing, Bhutan	17100003
4	4	Sonam Wangmo	1988-09-...	Female	Thimphu, Bhutan	17100004
5	5	Sangay Tshering	2000-01-...	Male	Trongsa, Bhutan	17100005
6	6	Chimi Dorji	1995-05-...	Female	Wangdue, Bhutan	17100006
7	7	Dorji Wangchuk	1982-03-...	Male	Bumthang, Bhutan	17100007
8	8	Dechen Wangmo	1998-11-...	Female	Mongar, Bhutan	17100008
9	9	Kinley Dorji	1975-07-...	Male	Gelegphu, Bhutan	17100009
10	10	Tshering Pem	1992-08-...	Female	Trashigang, Bhutan	17100010
11	11	Tandin Wangchuk	1987-02-...	Male	Paro, Bhutan	17100011
12	12	Sangay Choden	2001-12-...	Female	Thimphu, Bhutan	17100012
13	13	Jigme Dorji	1993-06-...	Male	Haa, Bhutan	17100013
14	14	Kezang Wangmo	1989-04-...	Female	Lhuntse, Bhutan	17100014
15	15	Phuntsho Tshering	1980-10-...	Male	Samdrup Jongkhar, Bhut...	17100015

Figure 1: Patient table with 15 sample inputs

Doctor

```
CREATE TABLE doctor(
    doctor_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    speciality VARCHAR(50),
    phone VARCHAR(15) UNIQUE
);
```

```
INSERT INTO doctor (name, speciality, phone) VALUES
('Dr. Karma Wangchuk', 'Cardiology', '17220001'),
('Dr. Pema Choden', 'Dermatology', '17220002'),
('Dr. Tashi Dorji', 'Neurology', '17220003'),
('Dr. Sonam Wangmo', 'Pediatrics', '17220004'),
('Dr. Sangay Tshering', 'Orthopedics', '17220005'),
('Dr. Chimi Dorji', 'General Medicine', '17220006'),
('Dr. Dorji Wangchuk', 'ENT', '17220007'),
('Dr. Dechen Wangmo', 'Gynecology', '17220008'),
('Dr. Kinley Dorji', 'Ophthalmology', '17220009'),
('Dr. Tshering Pem', 'Cardiology', '17220010'),
('Dr. Tandin Wangchuk', 'Neurology', '17220011'),
('Dr. Sangay Choden', 'Dermatology', '17220012'),
('Dr. Jigme Dorji', 'Orthopedics', '17220013'),
('Dr. Kezang Wangmo', 'Pediatrics', '17220014'),
('Dr. Phuntsho Tshering', 'General Medicine', '17220015');

SELECT * FROM doctor;
```

Data Output | Messages | Notifications

	doctor_id [PK] integer	name character varying (100)	speciality character varying (50)	phone character varying (15)
1	1	Dr. Karma Wangchuk	Cardiology	17220001
2	2	Dr. Pema Choden	Dermatology	17220002
3	3	Dr. Tashi Dorji	Neurology	17220003
4	4	Dr. Sonam Wangmo	Pediatrics	17220004
5	5	Dr. Sangay Tshering	Orthopedics	17220005
6	6	Dr. Chimi Dorji	General Medicine	17220006
7	7	Dr. Dorji Wangchuk	ENT	17220007
8	8	Dr. Dechen Wangmo	Gynecology	17220008
9	9	Dr. Kinley Dorji	Ophthalmology	17220009
10	10	Dr. Tshering Pem	Cardiology	17220010
11	11	Dr. Tandin Wangchuk	Neurology	17220011
12	12	Dr. Sangay Choden	Dermatology	17220012
13	13	Dr. Jigme Dorji	Orthopedics	17220013
14	14	Dr. Kezang Wangmo	Pediatrics	17220014
15	15	Dr. Phuntsho Tshering	General Medicine	17220015

Figure 2: Doctor table with 15 sample inputs

Appointment

```
60  CREATE TYPE status_enum AS ENUM('Scheduled', 'Completed', 'Cancelled')
61
62  CREATE TABLE appointment(
63      appointment_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
64      patient_id INT REFERENCES patient(patient_id) ON DELETE CASCADE,
65      doctor_id INT REFERENCES doctor(doctor_id) ON DELETE SET NULL,
66      appointment_date DATE NOT NULL,
67      appointment_time TIME NOT NULL,
68      status status_enum,
69      status status_enum DEFAULT 'Scheduled'
70  );
```

```sql
INSERT INTO appointment (patient_id, doctor_id, appointment_date, appointment_time, status) VALUES
(1, 1, '2025-08-21', '09:00:00', 'Completed'),
(2, 2, '2025-08-22', '10:45:00', 'Scheduled'),
(3, 3, '2025-08-23', '11:05:00', 'Completed'),
(4, 4, '2025-08-24', '14:30:00', 'Scheduled'),
(5, 5, '2025-08-26', '15:20:00', 'Cancelled'),
(6, 6, '2025-08-27', '09:15:00', 'Scheduled'),
(7, 7, '2025-08-28', '10:10:00', 'Completed'),
(8, 8, '2025-08-29', '11:50:00', 'Scheduled'),
(9, 9, '2025-08-30', '13:05:00', 'Scheduled'),
(10, 10, '2025-09-01', '14:45:00', 'Completed'),
(11, 11, '2025-09-02', '09:20:00', 'Scheduled'),
(12, 12, '2025-09-03', '10:35:00', 'Completed'),
(13, 13, '2025-09-05', '11:00:00', 'Scheduled'),
(14, 14, '2025-09-06', '13:55:00', 'Scheduled'),
(15, 15, '2025-09-07', '15:05:00', 'Scheduled');

SELECT * FROM appointment;
```

Data Output | Messages | Notifications

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	1	1	1	2025-08-21	09:00:00	Completed
2	2	2	2	2025-08-22	10:45:00	Scheduled
3	3	3	3	2025-08-23	11:05:00	Completed
4	4	4	4	2025-08-24	14:30:00	Scheduled
5	5	5	5	2025-08-26	15:20:00	Cancelled
6	6	6	6	2025-08-27	09:15:00	Scheduled
7	7	7	7	2025-08-28	10:10:00	Completed
8	8	8	8	2025-08-29	11:50:00	Scheduled
9	9	9	9	2025-08-30	13:05:00	Scheduled
10	10	10	10	2025-09-01	14:45:00	Completed
11	11	11	11	2025-09-02	09:20:00	Scheduled
12	12	12	12	2025-09-03	10:35:00	Completed
13	13	13	13	2025-09-05	11:00:00	Scheduled
14	14	14	14	2025-09-06	13:55:00	Scheduled
15	15	15	15	2025-09-07	15:05:00	Scheduled

Figure 3: Appointment table with 15 sample inputs

Medical Record

```sql
CREATE TABLE medical_record(
    record_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    patient_id INT REFERENCES patient(patient_id) ON DELETE CASCADE,
    doctor_id INT REFERENCES doctor(doctor_id) ON DELETE SET NULL,
    diagnosis TEXT,
    treatment TEXT,
    record_date DATE NOT NULL
);
```

```sql
INSERT INTO medical_record (patient_id, doctor_id, diagnosis, treatment, record_date) VALUES
(1, 1, 'Hypertension', 'Amlodipine 5mg daily', '2025-08-21'),
(2, 2, 'Acne', 'Topical cream twice daily', '2025-08-22'),
(3, 3, 'Migraine', 'Painkillers as needed', '2025-08-23'),
(4, 4, 'Common Cold', 'Rest and fluids', '2025-08-24'),
(5, 5, 'Fracture', 'Cast for 6 weeks', '2025-08-25'),
(6, 6, 'Flu', 'Antiviral medication', '2025-08-26'),
(7, 7, 'Ear Infection', 'Antibiotics', '2025-08-27'),
(8, 8, 'Pregnancy Checkup', 'Routine checkup', '2025-08-28'),
(9, 9, 'Cataract', 'Surgery scheduled', '2025-08-29'),
(10, 10, 'Hypertension', 'Medication adjustment', '2025-08-30'),
(11, 11, 'Migraine', 'Lifestyle changes', '2025-09-01'),
(12, 12, 'Skin Rash', 'Topical ointment', '2025-09-02'),
(13, 13, 'Bone Fracture', 'Physiotherapy after cast', '2025-09-03'),
(14, 14, 'Routine Checkup', 'General health advice', '2025-09-04'),
(15, 15, 'Diabetes', 'Insulin therapy', '2025-09-05');

SELECT * FROM medical_record;
```

Data Output | Messages | Notifications

	record_id [PK] integer	patient_id integer	doctor_id integer	diagnosis text	treatment text	record_date date
1	1	1	1	Hypertension	Amlodipine 5mg daily	2025-08-21
2	2	2	2	Acne	Topical cream twice dai...	2025-08-22
3	3	3	3	Migraine	Painkillers as needed	2025-08-23
4	4	4	4	Common Cold	Rest and fluids	2025-08-24
5	5	5	5	Fracture	Cast for 6 weeks	2025-08-25
6	6	6	6	Flu	Antiviral medication	2025-08-26
7	7	7	7	Ear Infection	Antibiotics	2025-08-27
8	8	8	8	Pregnancy Check...	Routine checkup	2025-08-28
9	9	9	9	Cataract	Surgery scheduled	2025-08-29
10	10	10	10	Hypertension	Medication adjustment	2025-08-30
11	11	11	11	Migraine	Lifestyle changes	2025-09-01
12	12	12	12	Skin Rash	Topical ointment	2025-09-02
13	13	13	13	Bone Fracture	Physiotherapy after cast	2025-09-03
14	14	14	14	Routine Checkup	General health advice	2025-09-04
15	15	15	15	Diabetes	Insulin therapy	2025-09-05

Figure 4: Medical Record table with 15 sample inputs

ER Diagram

Simple textual notation and description for the Entity-Relationship (ER) diagram for the HMS is given below.

1. Entities:
    - Patient (Attributes: patient_id [PK], name, dob, gender, address, phone)
    - Doctor (Attributes: doctor_id [PK], name, specialty, phone)
    - Appointment (Attributes: appointment_id [PK],
    - appointment_date, appointment_time, status)
    - MedicalRecord (Attributes: record_id [PK], diagnosis, treatment, record_date)
2. Relationships:
    - Patient --(has)-- Appointment (1:N, Participation: Patient optional, Appointment mandatory)
    - Doctor --(conducts)-- Appointment (1:N, Participation: Doctor optional, Appointment optional)
    - Patient --(owns)-- MedicalRecord (1:N, Participation: Patient mandatory, MedicalRecord optional)
    - Doctor --(creates)-- MedicalRecord (1:N, Participation: Doctor optional, MedicalRecord optional)

Following is the ER diagram for the HMS drawn using crow's foot notation:

Tool used: Draw.io - draw.io

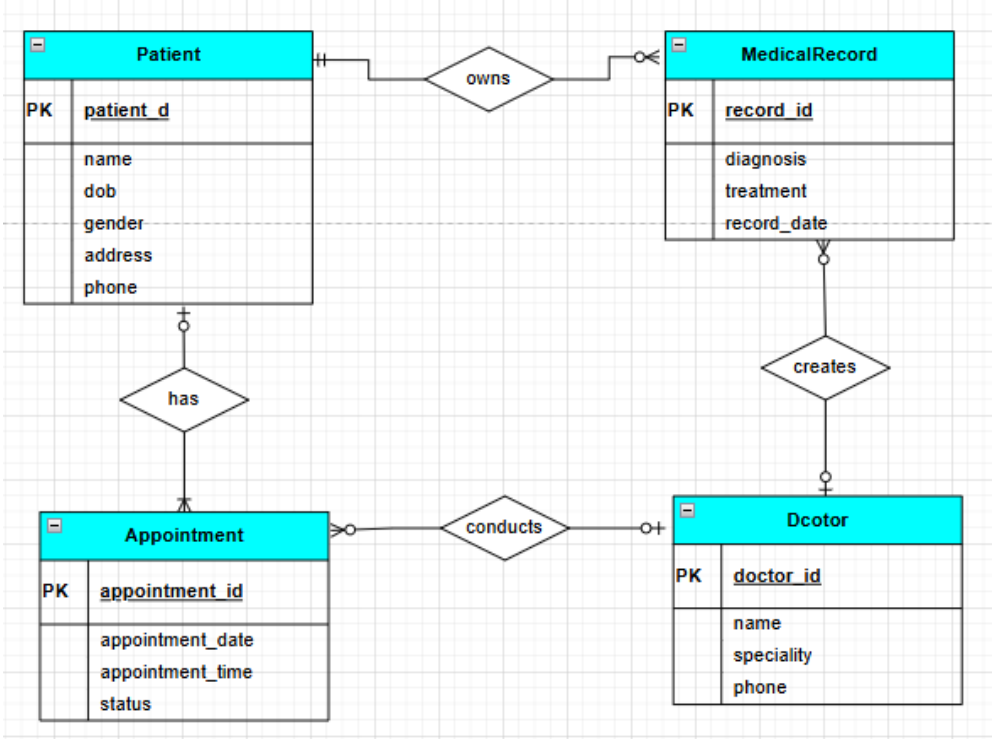


Figure 5: ER diagram for HMS

Use case diagram

The functional features are described focusing on key actors and interactions.

A. Actors:

- Admin (DBA): Manages users and schema.
- Receptionist: Handles registrations and scheduling.
- Doctor: Views/updates records.
- Patient: Views personal data (limited access).

B. Use Cases:

- Register Patient (Receptionist): Add new patient details.
- Schedule Appointment (Receptionist, Patient): Book slots with doctors.
- View Doctor Availability (Receptionist, Patient): Check free slots.
- Update Medical Record (Doctor): Add diagnosis/treatment.
- Generate Report (Admin, Doctor): Query patient history or aggregates.

- ❖ Cancel Appointment (Receptionist, Patient): Update status.
- ❖ Backup Database (Admin): Ensure data integrity.

C. Extensions/Includes:

- ❖ Schedule Appointment includes View Doctor Availability.
- ❖ Update Medical Record extends View Medical Record (if exists).

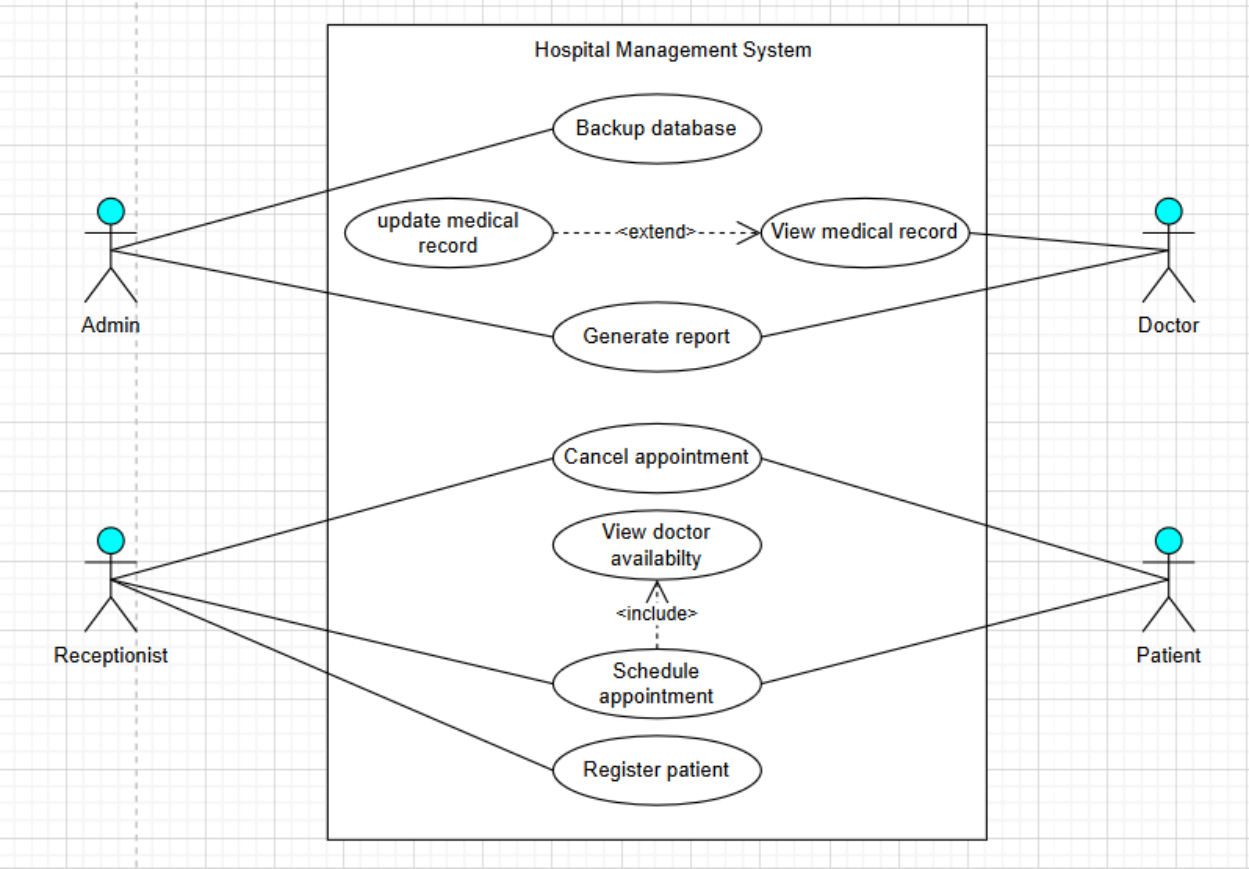


Figure 6: Use case diagram for HMS

Note: The position of the actors to the left or right does not determine if it is a primary or a secondary actor. The position of the actors is used as given above only for clean and clear visualization.

Tool : draw.io

Knowledge testing questions

1. Write a nested query to find patients who have more appointments than the average number of appointments per patient. Use aggregates.

```
SELECT p.patient_id, p.name, COUNT(a.appointment_id) AS total_appointments
FROM patient p
JOIN appointment a ON p.patient_id = a.patient_id
GROUP BY p.patient_id, p.name
HAVING COUNT(a.appointment_id) > (
    SELECT AVG(appointment_count)
    FROM (
        SELECT COUNT(*) AS appointment_count
        FROM appointment
        GROUP BY patient_id
    ) AS sub
);
```

Data Output | Messages | Notifications

patient_id [PK] integer	name character varying (100)	total_appointments bigint

The table is empty meaning there are no patients who have more appointments than the average number of appointments per patient

2. Write the query to exclude patients with NULL phone numbers using IS NOT NULL.

```
SELECT *
FROM patient
WHERE phone IS NOT NULL;
```

Query | Query History

Data Output | Messages | Notifications

	patient_id [PK] integer	name character varying (100)	dob date	gender gender_enum	address character varying (255)	phone character varying (15)
1	1	Tashi Wangchuk	1985-04-...	Male	Thimphu, Bhutan	17100001
2	2	Pema Choden	1990-06-...	Female	Paro, Bhutan	17100002
3	3	Kezang Dorji	1978-12-...	Male	Phuentsholing, Bhutan	17100003
4	4	Sonam Wangmo	1988-09-...	Female	Thimphu, Bhutan	17100004
5	5	Sangay Tshering	2000-01-...	Male	Trongsa, Bhutan	17100005
6	6	Chimi Dorji	1995-05-...	Female	Wangdue, Bhutan	17100006
7	7	Dorji Wangchuk	1982-03-...	Male	Bumthang, Bhutan	17100007
8	8	Dechen Wangmo	1998-11-...	Female	Mongar, Bhutan	17100008
9	9	Kinley Dorji	1975-07-...	Male	Gelegphu, Bhutan	17100009
10	10	Tshering Pem	1992-08-...	Female	Trashigang, Bhutan	17100010
11	11	Tandin Wangchuk	1987-02-...	Male	Paro, Bhutan	17100011
12	12	Sangay Choden	2001-12-...	Female	Thimphu, Bhutan	17100012
13	13	Jigme Dorji	1993-06-...	Male	Haa, Bhutan	17100013
14	14	Kezang Wangmo	1989-04-...	Female	Lhuntse, Bhutan	17100014
15	15	Phuntsho Tshering	1980-10-...	Male	Samdrup Jongkhar, Bhut...	17100015

All the patients had their phone numbers in the entry.

3. Write the Query for "Schedule Appointment," where the system checks doctor availability using a query on Appointment table, then inserts a new row.

Data Output Messages Notifications

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
5	5	5	5	2025-08-26	15:20:00	Cancelled
6	6	6	6	2025-08-27	09:15:00	Scheduled
7	7	7	7	2025-08-28	10:10:00	Completed
8	8	8	8	2025-08-29	11:50:00	Scheduled
9	9	9	9	2025-08-30	13:05:00	Scheduled
10	10	10	10	2025-09-01	14:45:00	Completed
11	11	11	11	2025-09-02	09:20:00	Scheduled
12	12	12	12	2025-09-03	10:35:00	Completed
13	13	13	13	2025-09-05	11:00:00	Scheduled
14	14	14	14	2025-09-06	13:55:00	Scheduled
15	15	15	15	2025-09-07	15:05:00	Scheduled

```
INSERT INTO appointment (doctor_id, patient_id, appointment_date, appointment_time, status)
SELECT 1, patient_id, '2025-08-27', '10:00:00', 'Scheduled'
FROM patient
WHERE patient_id = 1
AND NOT EXISTS (
    SELECT 1
    FROM appointment
    WHERE doctor_id = 1
      AND appointment_date = '2025-08-27'
      AND appointment_time = '10:00:00'
      AND status = 'Scheduled'
);
```

Data Output Messages Notifications

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
6	6	6	6	2025-08-27	09:15:00	Scheduled
7	7	7	7	2025-08-28	10:10:00	Completed
8	8	8	8	2025-08-29	11:50:00	Scheduled
9	9	9	9	2025-08-30	13:05:00	Scheduled
10	10	10	10	2025-09-01	14:45:00	Completed
11	11	11	11	2025-09-02	09:20:00	Scheduled
12	12	12	12	2025-09-03	10:35:00	Completed
13	13	13	13	2025-09-05	11:00:00	Scheduled
14	14	14	14	2025-09-06	13:55:00	Scheduled
15	15	15	15	2025-09-07	15:05:00	Scheduled
16	17	1	1	2025-08-27	10:00:00	Scheduled

After checking that doctor (doctor_id = 1) was available on 2025-08-27, a new appointment (appointment_id = 16) was scheduled with the doctor.

4. Write a query using UNION to list all unique phone numbers from Patient and Doctor tables.

```
SELECT phone
FROM patient
WHERE phone IS NOT NULL

UNION

SELECT phone
FROM doctor
WHERE phone IS NOT NULL;
```

Data Output | Messages | Not

	phone character varying (15)
1	17220001
2	17220011
3	17100008
4	17100015
5	17100001
6	17100013
7	17220004
8	17220014
9	17220015
10	17100012
11	17100009
12	17220007
13	17220003
14	17100011
15	17100004
16	17220010
17	17100014
18	17220006
19	17220002
20	17100002
21	17220008
22	17220005
23	17220012
24	17100005
25	17220013
26	17220009
27	17100006
28	17100010
29	17100007
30	17100003

All the phone numbers were found unique.

5. Use INTERSECT to find common specialties between doctors and any mentioned in medical records (assume adding a 'required_specialty' column if needed).

Adding the ‘required_speciality’ column in medical record table:

```
ALTER TABLE medical_record
ADD COLUMN required_specialty VARCHAR(50);
```

```sql
UPDATE medical_record
SET required_specialty = CASE record_id
    WHEN 1 THEN 'Cardiology'
    WHEN 2 THEN 'Dermatology'
    WHEN 3 THEN 'Neurology'
    WHEN 4 THEN 'General Medicine'
    WHEN 5 THEN 'Orthopedics'
    WHEN 6 THEN 'General Medicine'
    WHEN 7 THEN 'ENT'
    WHEN 8 THEN 'Gynecology'
    WHEN 9 THEN 'Ophthalmology'
    WHEN 10 THEN 'Cardiology'
    WHEN 11 THEN 'Neurology'
    WHEN 12 THEN 'Dermatology'
    WHEN 13 THEN 'Orthopedics'
    WHEN 14 THEN 'General Medicine'
    WHEN 15 THEN 'Endocrinology'
END;
```

Data Output | Messages | Notifications

	record_id [PK] integer	patient_id integer	doctor_id integer	diagnosis text	treatment text	record_date date	required_specialty character varying (50)
1	1	1	1	Hypertension	Amlodipine 5mg daily	2025-08-21	Cardiology
2	2	2	2	Acne	Topical cream twice dai...	2025-08-22	Dermatology
3	3	3	3	Migraine	Painkillers as needed	2025-08-23	Neurology
4	4	4	4	Common Cold	Rest and fluids	2025-08-24	General Medicine
5	5	5	5	Fracture	Cast for 6 weeks	2025-08-25	Orthopedics
6	6	6	6	Flu	Antiviral medication	2025-08-26	General Medicine
7	7	7	7	Ear Infection	Antibiotics	2025-08-27	ENT
8	8	8	8	Pregnancy Check...	Routine checkup	2025-08-28	Gynecology
9	9	9	9	Cataract	Surgery scheduled	2025-08-29	Ophthalmology
10	10	10	10	Hypertension	Medication adjustment	2025-08-30	Cardiology
11	11	11	11	Migraine	Lifestyle changes	2025-09-01	Neurology
12	12	12	12	Skin Rash	Topical ointment	2025-09-02	Dermatology
13	13	13	13	Bone Fracture	Physiotherapy after cast	2025-09-03	Orthopedics
14	14	14	14	Routine Checkup	General health advice	2025-09-04	General Medicine
15	15	15	15	Diabetes	Insulin therapy	2025-09-05	Endocrinology

```sql
SELECT speciality
FROM doctor

INTERSECT

SELECT required_specialty
FROM medical_record;
```

Data Output | Messages | Notifications

	speciality character varying (50)
1	Cardiology
2	ENT
3	General Medicine
4	Dermatology
5	Orthopedics
6	Gynecology
7	Ophthalmology
8	Neurology

6. Use MINUS to find patients without any medical records. Implement HMS and discuss handling duplicates.

   In Postgres SQL we have except instead of minus. Except automatically deals with the duplicates. It will only return a distint values although there are multiple entries for the same entity.

```
SELECT patient_id, name
FROM patient

EXCEPT

SELECT p.patient_id, p.name
FROM patient p
JOIN medical_record m ON p.patient_id = m.patient_id;
```

Data Output | Messages | Notifications

	patient_id integer	name character varying (100)

   All the patients tend to have entry in medical records.

7. Write an INNER JOIN query to list appointments with patient and doctor names.

```sql
SELECT
    a.appointment_id,
    p.name AS patient_name,
    d.name AS doctor_name,
    a.appointment_date,
    a.appointment_time,
    a.status
FROM appointment a
INNER JOIN patient p ON a.patient_id = p.patient_id
INNER JOIN doctor d ON a.doctor_id = d.doctor_id;
```

Data Output | Messages | Notifications

	appointment_id integer	patient_name character varying (100)	doctor_name character varying (100)	appointment_date date	appointment_time time without time zone	status status_enum
1	1	Tashi Wangchuk	Dr. Karma Wangchuk	2025-08-21	09:00:00	Completed
2	2	Pema Choden	Dr. Pema Choden	2025-08-22	10:45:00	Scheduled
3	3	Kezang Dorji	Dr. Tashi Dorji	2025-08-23	11:05:00	Completed
4	4	Sonam Wangmo	Dr. Sonam Wangmo	2025-08-24	14:30:00	Scheduled
5	5	Sangay Tshering	Dr. Sangay Tshering	2025-08-26	15:20:00	Cancelled
6	6	Chimi Dorji	Dr. Chimi Dorji	2025-08-27	09:15:00	Scheduled
7	7	Dorji Wangchuk	Dr. Dorji Wangchuk	2025-08-28	10:10:00	Completed
8	8	Dechen Wangmo	Dr. Dechen Wangmo	2025-08-29	11:50:00	Scheduled
9	9	Kinley Dorji	Dr. Kinley Dorji	2025-08-30	13:05:00	Scheduled
10	10	Tshering Pem	Dr. Tshering Pem	2025-09-01	14:45:00	Completed
11	11	Tandin Wangchuk	Dr. Tandin Wangchuk	2025-09-02	09:20:00	Scheduled
12	12	Sangay Choden	Dr. Sangay Choden	2025-09-03	10:35:00	Completed
13	13	Jigme Dorji	Dr. Jigme Dorji	2025-09-05	11:00:00	Scheduled
14	14	Kezang Wangmo	Dr. Kezang Wangmo	2025-09-06	13:55:00	Scheduled
15	15	Phuntsho Tshering	Dr. Phuntsho Tshering	2025-09-07	15:05:00	Scheduled
16	17	Tashi Wangchuk	Dr. Karma Wangchuk	2025-08-27	10:00:00	Scheduled

8. Use LEFT OUTER JOIN to list all patients and their appointments (including those without appointments).

```sql
SELECT
    p.patient_id,
    p.name AS patient_name,
    a.appointment_id,
    a.appointment_date,
    a.appointment_time,
    a.status
FROM patient p
LEFT OUTER JOIN appointment a
    ON p.patient_id = a.patient_id
ORDER BY p.patient_id;
```

Data Output Messages Notifications

	patient_id integer	patient_name character varying (100)	appointment_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	1	Tashi Wangchuk	1	2025-08-21	09:00:00	Completed
2	1	Tashi Wangchuk	17	2025-08-27	10:00:00	Scheduled
3	2	Pema Choden	2	2025-08-22	10:45:00	Scheduled
4	3	Kezang Dorji	3	2025-08-23	11:05:00	Completed
5	4	Sonam Wangmo	4	2025-08-24	14:30:00	Scheduled
6	5	Sangay Tshering	5	2025-08-26	15:20:00	Cancelled
7	6	Chimi Dorji	6	2025-08-27	09:15:00	Scheduled
8	7	Dorji Wangchuk	7	2025-08-28	10:10:00	Completed
9	8	Dechen Wangmo	8	2025-08-29	11:50:00	Scheduled
10	9	Kinley Dorji	9	2025-08-30	13:05:00	Scheduled
11	10	Tshering Pem	10	2025-09-01	14:45:00	Completed
12	11	Tandin Wangchuk	11	2025-09-02	09:20:00	Scheduled
13	12	Sangay Choden	12	2025-09-03	10:35:00	Completed
14	13	Jigme Dorji	13	2025-09-05	11:00:00	Scheduled
15	14	Kezang Wangmo	14	2025-09-06	13:55:00	Scheduled
16	15	Phuntsho Tshering	15	2025-09-07	15:05:00	Scheduled

9. Use SELF JOIN on Appointment to find conflicting appointments for the same doctor on the same date.

```
SELECT
    a1.doctor_id,
    a1.appointment_date,
    a1.appointment_time AS time1,
    a2.appointment_time AS time2
FROM appointment a1
JOIN appointment a2
    ON a1.doctor_id = a2.doctor_id
   AND a1.appointment_date = a2.appointment_date
   AND a1.appointment_id < a2.appointment_id;
```

Data Output Messages Notifications

	doctor_id integer	appointment_date date	time1 time without time zone	time2 time without time zone

There are no conflicting appointments for the same doctor on the same date.

10. Implement a FULL JOIN to combine Patient and MedicalRecord, explaining NULL handling.

```sql
SELECT
    p.patient_id,
    p.name AS patient_name,
    m.record_id,
    m.diagnosis,
    m.treatment,
    m.record_date
FROM patient p
FULL JOIN medical_record m
    ON p.patient_id = m.patient_id
ORDER BY p.patient_id;
```

Data Output | Messages | Notifications

	patient_id integer	patient_name character varying (100)	record_id integer	diagnosis text	treatment text	record_date date
1	1	Tashi Wangchuk	1	Hypertension	Amlodipine 5mg daily	2025-08-21
2	2	Pema Choden	2	Acne	Topical cream twice dai...	2025-08-22
3	3	Kezang Dorji	3	Migraine	Painkillers as needed	2025-08-23
4	4	Sonam Wangmo	4	Common Cold	Rest and fluids	2025-08-24
5	5	Sangay Tshering	5	Fracture	Cast for 6 weeks	2025-08-25
6	6	Chimi Dorji	6	Flu	Antiviral medication	2025-08-26
7	7	Dorji Wangchuk	7	Ear Infection	Antibiotics	2025-08-27
8	8	Dechen Wangmo	8	Pregnancy Check...	Routine checkup	2025-08-28
9	9	Kinley Dorji	9	Cataract	Surgery scheduled	2025-08-29
10	10	Tshering Pem	10	Hypertension	Medication adjustment	2025-08-30
11	11	Tandin Wangchuk	11	Migraine	Lifestyle changes	2025-09-01
12	12	Sangay Choden	12	Skin Rash	Topical ointment	2025-09-02
13	13	Jigme Dorji	13	Bone Fracture	Physiotherapy after cast	2025-09-03
14	14	Kezang Wangmo	14	Routine Checkup	General health advice	2025-09-04
15	15	Phuntsho Tshering	15	Diabetes	Insulin therapy	2025-09-05

```sql
-- If a patient has no medical record → the record_id, diagnosis, etc. will be NULL.
-- If a medical record doesn't match any patient → the patient_id, patient_name, etc. will be NULL.
```

11. Create an assertion to ensure no doctor has more than 10 appointments per day.

    Postgres does not support assertions directly, so we user a before insert trigger to check the count of the appointments.

```
CREATE OR REPLACE FUNCTION limit_doctor_appointments()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT COUNT(*)
        FROM appointment
        WHERE doctor_id = NEW.doctor_id
          AND appointment_date = NEW.appointment_date) >= 10 THEN
        RAISE EXCEPTION 'Doctor % already has 10 appointments on %',
                        NEW.doctor_id, NEW.appointment_date;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER limit_appointments_trigger
BEFORE INSERT ON appointment
FOR EACH ROW
EXECUTE FUNCTION limit_doctor_appointments();
```

12. Write a trigger that automatically sets appointment status to 'Completed' after inserting a medical record for that appointment.

Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No: 1

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	1	1	1	2025-08-21	09:00:00	Completed
2	2	2	2	2025-08-22	10:45:00	Scheduled
3	3	3	3	2025-08-23	11:05:00	Completed
4	4	4	4	2025-08-24	14:30:00	Scheduled
5	5	5	5	2025-08-26	15:20:00	Cancelled
6	6	6	6	2025-08-27	09:15:00	Scheduled
7	7	7	7	2025-08-28	10:10:00	Completed
8	8	8	8	2025-08-29	11:50:00	Scheduled
9	9	9	9	2025-08-30	13:05:00	Scheduled
10	10	10	10	2025-09-01	14:45:00	Completed
11	11	11	11	2025-09-02	09:20:00	Scheduled
12	12	12	12	2025-09-03	10:35:00	Completed
13	13	13	13	2025-09-05	11:00:00	Scheduled
14	14	14	14	2025-09-06	13:55:00	Scheduled
15	15	15	15	2025-09-07	15:05:00	Scheduled
16	17	1	1	2025-08-27	10:00:00	Scheduled

Let us consider appointment with ‘appointment id = 2’ whose status reflected above as scheduled.

Now, let us create a function which will automatically update the status of the appointment to ‘completed’ once we insert the details of the appointment into the medical record table.

```
CREATE OR REPLACE FUNCTION complete_appointment()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE appointment
    SET status = 'Completed'
    WHERE appointment_id = NEW.appointment_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

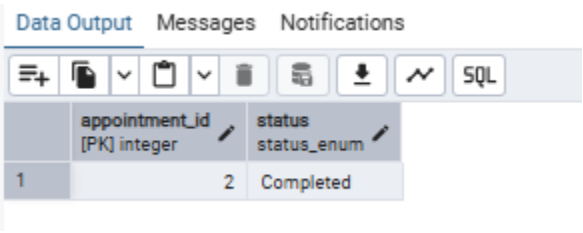
Then, create a trigger to force the action.

```
CREATE TRIGGER set_appointment_completed
AFTER INSERT ON medical_record
FOR EACH ROW
EXECUTE FUNCTION complete_appointment();
```

Test: Inserting appointment (appointment_id = 2) into medical record table.

```
INSERT INTO medical_record (patient_id, doctor_id, appointment_id, diagnosis, treatment, record_date)
VALUES (2, 2, 2, 'Hypertension Follow-up', 'Medication review', '2025-08-22');

SELECT appointment_id, status
FROM appointment
WHERE appointment_id = 2;
```



After inserting the record into the medical record table, we can see that the status of the appointment is automatically set to ‘completed’.

13. Create a view 'PatientAppointments' showing patient names and their appointment details.

```sql
CREATE VIEW PatientAppointments AS
SELECT
    p.patient_id,
    p.name AS patient_name,
    a.appointment_id,
    a.appointment_date,
    a.appointment_time,
    a.status
FROM patient p
JOIN appointment a
    ON p.patient_id = a.patient_id
ORDER BY p.patient_id, a.appointment_date, a.appointment_time;

select * from PatientAppointments;
```

Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No: 1

	patient_id integer	patient_name character varying (100)	appointment_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	1	Tashi Wangchuk	1	2025-08-21	09:00:00	Completed
2	1	Tashi Wangchuk	17	2025-08-27	10:00:00	Scheduled
3	2	Pema Choden	2	2025-08-22	10:45:00	Completed
4	3	Kezang Dorji	3	2025-08-23	11:05:00	Completed
5	4	Sonam Wangmo	4	2025-08-24	14:30:00	Scheduled
6	5	Sangay Tshering	5	2025-08-26	15:20:00	Cancelled
7	6	Chimi Dorji	6	2025-08-27	09:15:00	Scheduled
8	7	Dorji Wangchuk	7	2025-08-28	10:10:00	Completed
9	8	Dechen Wangmo	8	2025-08-29	11:50:00	Scheduled
10	9	Kinley Dorji	9	2025-08-30	13:05:00	Scheduled
11	10	Tshering Pem	10	2025-09-01	14:45:00	Completed
12	11	Tandin Wangchuk	11	2025-09-02	09:20:00	Scheduled
13	12	Sangay Choden	12	2025-09-03	10:35:00	Completed
14	13	Jigme Dorji	13	2025-09-05	11:00:00	Scheduled
15	14	Kezang Wangmo	14	2025-09-06	13:55:00	Scheduled
16	15	Phuntsho Tshering	15	2025-09-07	15:05:00	Scheduled

14. Attempt to update the view (e.g., change status) and explain if/why it succeeds or fails.

```sql
UPDATE PatientAppointments
SET status = 'Cancelled'
WHERE appointment_id = 1;
```

Data Output | Messages | Notifications

```
ERROR:  cannot update view "patientappointments"
Views that do not select from a single table or view are not automatically updatable.

SQL state: 55000
Detail: Views that do not select from a single table or view are not automatically updatable.
Hint: To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
```

My view joins multiple tables, so Postgres does not allow direct updates on such views. The easiest way to solve this problem is by making the updates in the original table.

```
UPDATE appointment
SET status = 'Cancelled'
WHERE appointment_id = 1;

select * from PatientAppointments;
```

Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No: 1 of 1

	patient_id integer	patient_name character varying (100)	appointment_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	1	Tashi Wangchuk	1	2025-08-21	09:00:00	Cancelled
2	1	Tashi Wangchuk	17	2025-08-27	10:00:00	Scheduled
3	2	Pema Choden	2	2025-08-22	10:45:00	Completed
4	3	Kezang Dorji	3	2025-08-23	11:05:00	Completed
5	4	Sonam Wangmo	4	2025-08-24	14:30:00	Scheduled
6	5	Sangay Tshering	5	2025-08-26	15:20:00	Cancelled

15. Identify potential issues like SQL injection in appointment scheduling and propose defensive techniques.

    Some of the potential issues in appointment scheduling are as follows:

    - The data is vulnerable to SQL injection when user inserts input directly through SQL code. The attackers can change or alter important user information.
    - Mistakes while entering information like date or time can also lead to errors.
    - There are also possibilities of double booking for the doctor

    Some of the defensive techniques for the above-mentioned issues are:

    - Using prepared statements with parameterized queries which will prevent users from altering the SQL queries.
    - Implementing input validation which will check the correctness of each of the user input (value and format)
    - Implementing checks ensuring that ethe doctor is not booked again on a particular date and time.

16. Using JDBC, write Java code to dynamically query appointments by date range (user input).

```java
import java.sql.*;
import java.util.Scanner;

public class SimpleAppointmentQuery {

    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Enter start date (YYYY-MM-DD): ");
        String startDate = scanner.nextLine();
        System.out.print(s:"Enter end date (YYYY-MM-DD): ");
        String endDate = scanner.nextLine();

        String url = "jdbc:postgresql://localhost:5432/HMS_Database";
        String user = "postgres";
        String password = "12";

        try (Connection conn = DriverManager.getConnection(url, user, password);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(
                 "SELECT * FROM appointment WHERE appointment_date BETWEEN '"
                 + startDate + "' AND '" + endDate + "'")) {
```

Java program to take in the start date and end date as user inputs, connect to the database and print all important fields in the appointment table.

```java
            while (rs.next()) {
                System.out.println("Appointment ID: " + rs.getInt(columnLabel:"appointment_id")
                        + ", Patient ID: " + rs.getInt(columnLabel:"patient_id")
                        + ", Doctor ID: " + rs.getInt(columnLabel:"doctor_id")
                        + ", Date: " + rs.getDate(columnLabel:"appointment_date")
                        + ", Time: " + rs.getTime(columnLabel:"appointment_time")
                        + ", Status: " + rs.getString(columnLabel:"status"));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Appointments in the date range {2025-08-10} to {2025-08-28}

```
PS E:\Tashi\Final Year\DIS404\Assignment> java -cp ".;E:\Tashi\Final Year\DIS404\Assignment\postgresql-42.7.7.jar" SimpleAppointmentQuery.java
Enter start date (YYYY-MM-DD): 2025-08-10
Enter end date (YYYY-MM-DD): 2025-08-28
Appointment ID: 3, Patient ID: 3, Doctor ID: 3, Date: 2025-08-23, Time: 11:05:00, Status: Completed
Appointment ID: 4, Patient ID: 4, Doctor ID: 4, Date: 2025-08-24, Time: 14:30:00, Status: Scheduled
Appointment ID: 5, Patient ID: 5, Doctor ID: 5, Date: 2025-08-26, Time: 15:20:00, Status: Cancelled
Appointment ID: 6, Patient ID: 6, Doctor ID: 6, Date: 2025-08-27, Time: 09:15:00, Status: Scheduled
Appointment ID: 7, Patient ID: 7, Doctor ID: 7, Date: 2025-08-28, Time: 10:10:00, Status: Completed
Appointment ID: 17, Patient ID: 1, Doctor ID: 1, Date: 2025-08-27, Time: 10:00:00, Status: Scheduled
Appointment ID: 2, Patient ID: 2, Doctor ID: 2, Date: 2025-08-22, Time: 10:45:00, Status: Completed
Appointment ID: 1, Patient ID: 1, Doctor ID: 1, Date: 2025-08-21, Time: 09:00:00, Status: Cancelled
PS E:\Tashi\Final Year\DIS404\Assignment>
```

17. Create a stored procedure 'AddAppointment' that inserts an appointment and checks doctor availability.

Data Output | Messages | Notifications

Showing rows: 1 to 16 | Page No:

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	3	3	3	2025-08-23	11:05:00	Completed
2	4	4	4	2025-08-24	14:30:00	Scheduled
3	5	5	5	2025-08-26	15:20:00	Cancelled
4	6	6	6	2025-08-27	09:15:00	Scheduled
5	7	7	7	2025-08-28	10:10:00	Completed
6	8	8	8	2025-08-29	11:50:00	Scheduled
7	9	9	9	2025-08-30	13:05:00	Scheduled
8	10	10	10	2025-09-01	14:45:00	Completed
9	11	11	11	2025-09-02	09:20:00	Scheduled
10	12	12	12	2025-09-03	10:35:00	Completed
11	13	13	13	2025-09-05	11:00:00	Scheduled
12	14	14	14	2025-09-06	13:55:00	Scheduled
13	15	15	15	2025-09-07	15:05:00	Scheduled
14	17	1	1	2025-08-27	10:00:00	Scheduled
15	2	2	2	2025-08-22	10:45:00	Completed
16	1	1	1	2025-08-21	09:00:00	Cancelled

Creating a procedure called ‘AddAppointment’.

```
CREATE OR REPLACE PROCEDURE AddAppointment(
    p_patient_id INT,
    p_doctor_id INT,
    p_appointment_date DATE,
    p_appointment_time TIME,
    OUT result_message TEXT
)
LANGUAGE plpgsql
AS $$
DECLARE
    conflict_count INT;
BEGIN
    -- Check if doctor has an appointment at the same date and time
    SELECT COUNT(*) INTO conflict_count
    FROM appointment
    WHERE doctor_id = p_doctor_id
      AND appointment_date = p_appointment_date
      AND appointment_time = p_appointment_time;

    IF conflict_count > 0 THEN
        result_message := 'Doctor is not available at this time.';
    ELSE
        INSERT INTO appointment(patient_id, doctor_id, appointment_date, appointment_time, status)
        VALUES (p_patient_id, p_doctor_id, p_appointment_date, p_appointment_time, 'Scheduled');
        result_message := 'Appointment added successfully.';
    END IF;
END;
$$;
```

Calling the ‘AddAppointment’ procedure and adding new appointment.

```
DO $$
DECLARE
    msg TEXT;
BEGIN
    CALL AddAppointment(
        1,
        2,
        '2025-08-30'::DATE,
        '10:00:00'::TIME,
        msg
    );
    RAISE NOTICE 'Result: %', msg;
END
$$;


select * from appointment;
```

Data Output | Messages | Notifications

Showing row

	appointment_id [PK] integer	patient_id integer	doctor_id integer	appointment_date date	appointment_time time without time zone	status status_enum
1	3	3	3	2025-08-23	11:05:00	Completed
2	4	4	4	2025-08-24	14:30:00	Scheduled
3	5	5	5	2025-08-26	15:20:00	Cancelled
4	6	6	6	2025-08-27	09:15:00	Scheduled
5	7	7	7	2025-08-28	10:10:00	Completed
6	8	8	8	2025-08-29	11:50:00	Scheduled
7	9	9	9	2025-08-30	13:05:00	Scheduled
8	10	10	10	2025-09-01	14:45:00	Completed
9	11	11	11	2025-09-02	09:20:00	Scheduled
10	12	12	12	2025-09-03	10:35:00	Completed
11	13	13	13	2025-09-05	11:00:00	Scheduled
12	14	14	14	2025-09-06	13:55:00	Scheduled
13	15	15	15	2025-09-07	15:05:00	Scheduled
14	17	1	1	2025-08-27	10:00:00	Scheduled
15	2	2	2	2025-08-22	10:45:00	Completed
16	1	1	1	2025-08-21	09:00:00	Cancelled
17	18	1	2	2025-08-30	10:00:00	Scheduled

We can see that a new appointment (appointment_id = 18) with specified details has been added after checking doctor's availability on '2025-08-30', '10:00:00' which was TRUE.

References

Documentation | PGJDBC. (n.d.). https://jdbc.postgresql.org/documentation/

GeeksforGeeks. (2025, July 15). *Difference between Assertions and Triggers in DBMS*. GeeksforGeeks. https://www.geeksforgeeks.org/dbms/difference-between-assertions-and-triggers-in-dbms/

GeeksforGeeks. (2025, August 1). *Introduction to SQL*. GeeksforGeeks. https://www.geeksforgeeks.org/sql/introduction-to-sql/

GeeksforGeeks. (2025, July 23). *Use Case Diagram Unified Modeling Language (UML)*. GeeksforGeeks. https://www.geeksforgeeks.org/system-design/use-case-diagram/

Perera, N., & Creately. (2025, May 6). *Understanding Crow's Foot Notation: Symbols & Usage Guide*. Creately. https://www.creately.com/guides/crows-foot-notation/

SQL triggers. (2025, February 8). SQL Tutorial. https://www.sqltutorial.org/sql-triggers/

W3Schools.com. (n.d.). https://www.w3schools.com/sql/