DBMS final Project Progress Report

Project Title:

Hospital Management System

Database Used: MySQL

Group members:

1. Avinesh Pratap Singh -20JE0219
2. Brijesh Kumar -20JE0279
3. Rashi Kumari -20JE0771
4. Ritik Gupta -20JE0794

GitHub Repository:

https://github.com/pratapavinesh/Hospital-Management-System

Problem Statement

Developing a web Application for hospital management to effectively manage most aspects of hospitals such as registering new Patients and Doctors, booking appointments, managing Patient records and keeping medical history of patients.

Overview

Hospitals have to deal with a lot of patients regularly and hence a lot of data. Hence it is very important for a hospital to have a DBMS with a frontend that easily allows patients to avail different services and allows doctors and administrators to manage patient.

Tech Stack Used:

Frontend: ReactJS, JavaScript

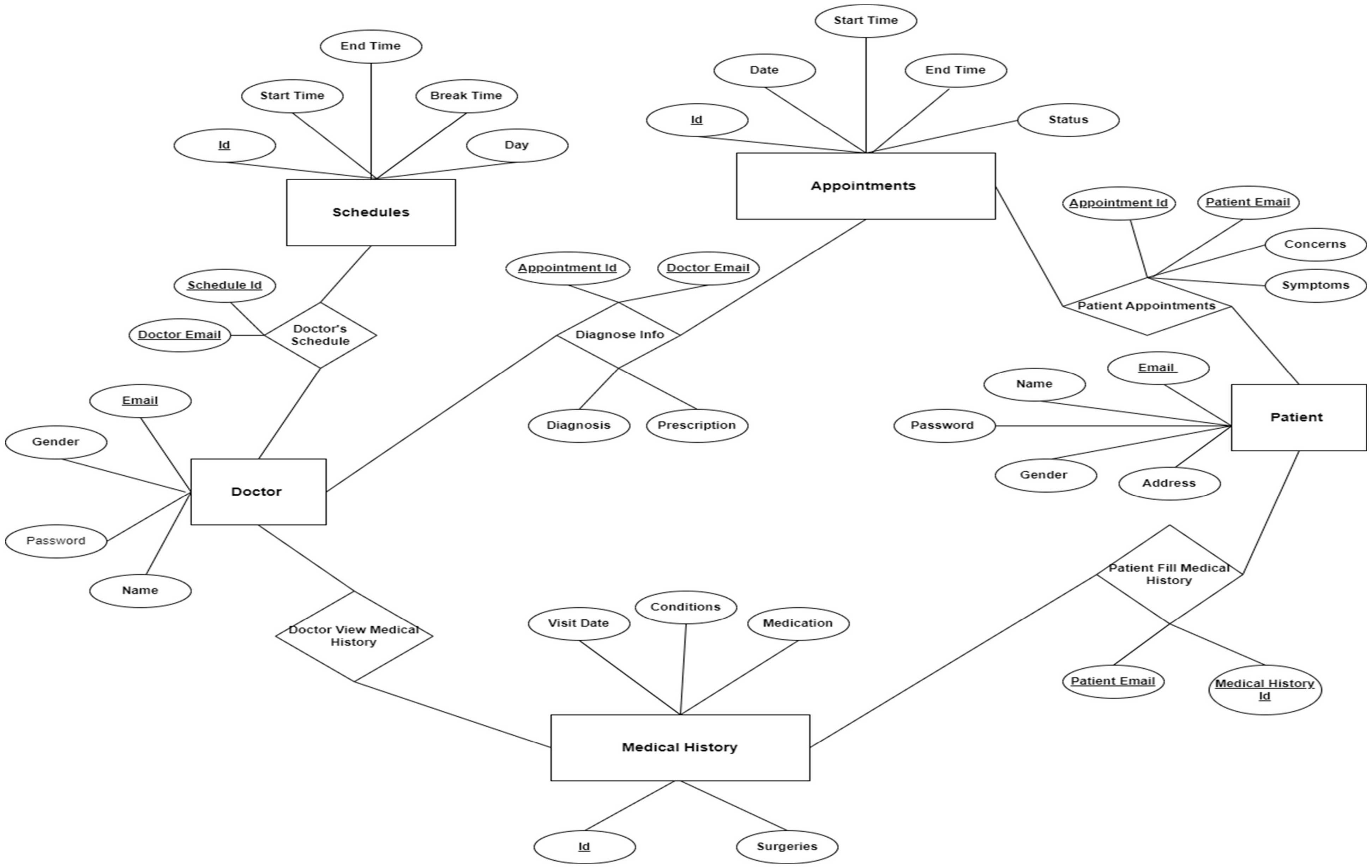
Backend: Node.js

Database: MySQL

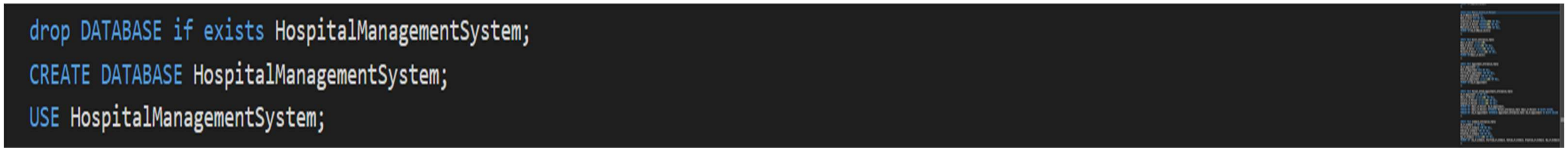
Functional Requirements

1. **For Patients:** Register, Login, Update Password, Appointments Booking, Appointments Cancelation, View of their Appointments, Updating Appointments.
2. **For Doctors:** Register, Login, Update Password, View of their Appointments, Appointments Cancellation, Patient’s Medical History View, Patient’s Profile, Updating Patient’s Medical History, Updating Prescription and Diagnosis after Appointment.
3. The system should avoid clash of appointments and allow appointments only when a doctor is not already busy or does not have a break.

ER Diagram

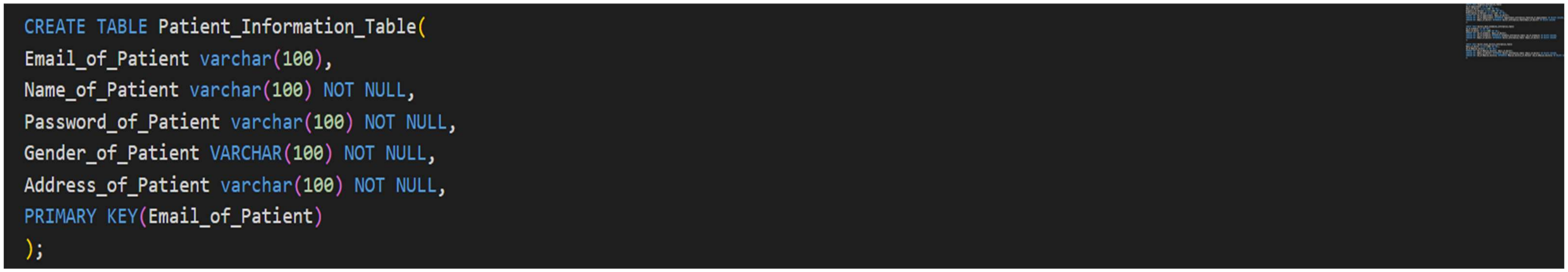

End Time
Start Time
Break Time
Id
Day
Schedules
Schedule Id
Doctor Email
Doctor's Schedule
Start Time
Date
End Time
Id
Status
Appointments
Appointment Id
Patient Email
Concerns
Symptoms
Patient Appointments
Appointment Id
Doctor Email
Diagnose Info
Diagnosis
Prescription
Email
Gender
Doctor
Password
Name
Name
Email
Password
Patient
Gender
Address
Patient Fill Medical History
Patient Email
Medical History Id
Doctor View Medical History
Visit Date
Conditions
Medication
Medical History
Id
Surgeries

1. Creation of Database:



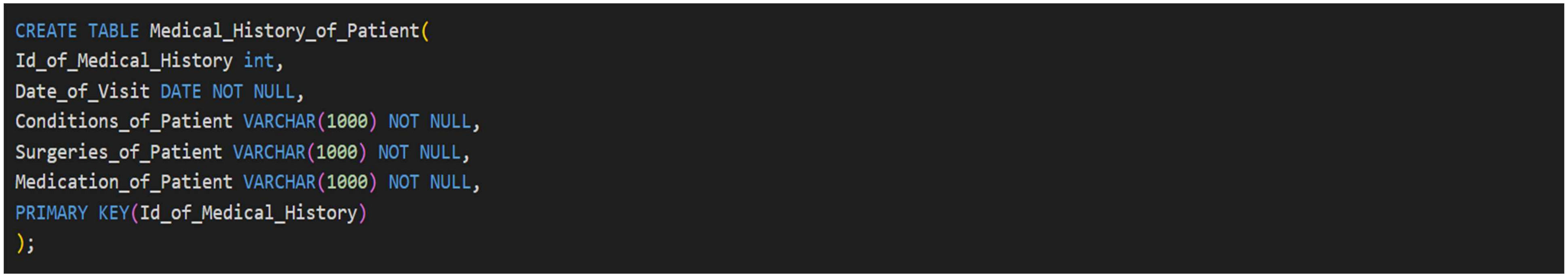
```sql
drop DATABASE if exists HospitalManagementSystem;
CREATE DATABASE HospitalManagementSystem;
USE HospitalManagementSystem;
```

2. Creation of Patient Information Table



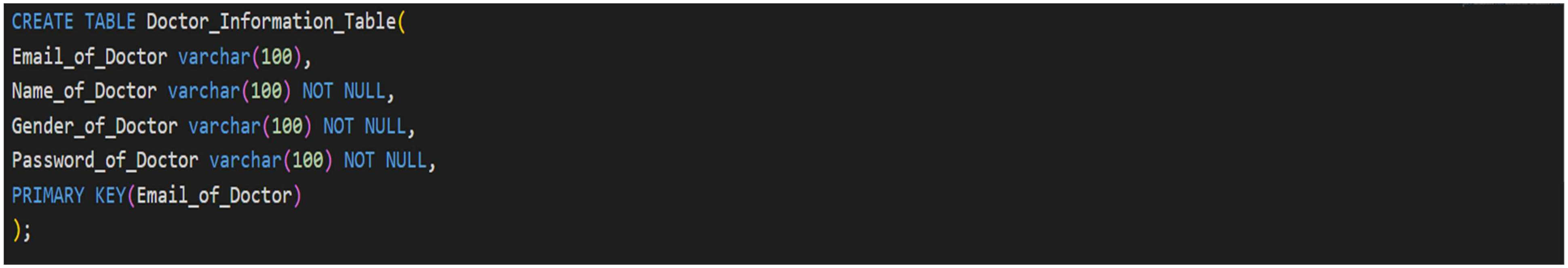
```sql
CREATE TABLE Patient_Information_Table(
Email_of_Patient varchar(100),
Name_of_Patient varchar(100) NOT NULL,
Password_of_Patient varchar(100) NOT NULL,
Gender_of_Patient VARCHAR(100) NOT NULL,
Address_of_Patient varchar(100) NOT NULL,
PRIMARY KEY(Email_of_Patient)
);
```

3. Creation of Doctor Information Table



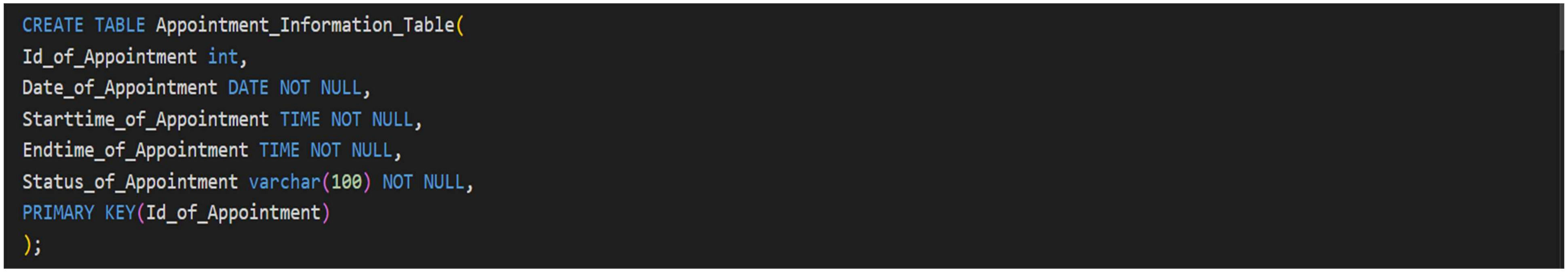
```sql
CREATE TABLE Medical_History_of_Patient(
Id_of_Medical_History int,
Date_of_Visit DATE NOT NULL,
Conditions_of_Patient VARCHAR(1000) NOT NULL,
Surgeries_of_Patient VARCHAR(1000) NOT NULL,
Medication_of_Patient VARCHAR(1000) NOT NULL,
PRIMARY KEY(Id_of_Medical_History)
);
```

4. Creation of Medical History of Patient Table



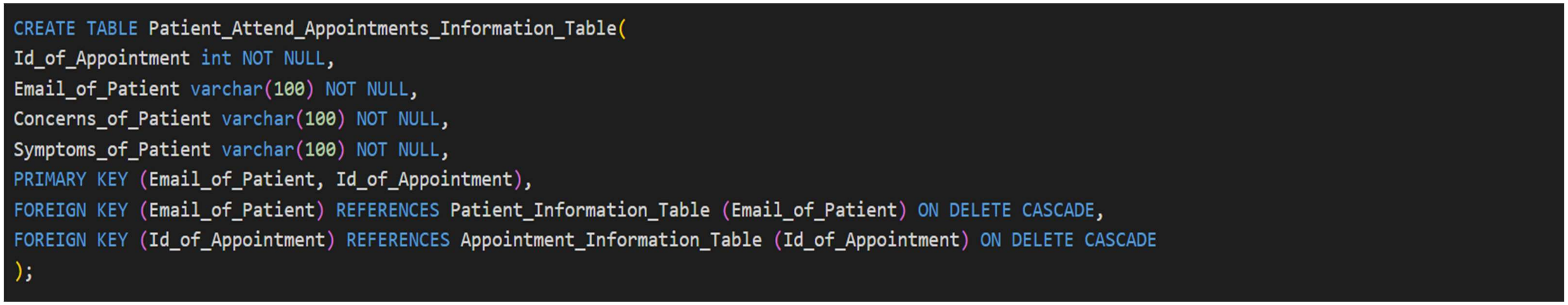
```sql
CREATE TABLE Doctor_Information_Table(
Email_of_Doctor varchar(100),
Name_of_Doctor varchar(100) NOT NULL,
Gender_of_Doctor varchar(100) NOT NULL,
Password_of_Doctor varchar(100) NOT NULL,
PRIMARY KEY(Email_of_Doctor)
);
```

5. Creation of Appointment Information Table



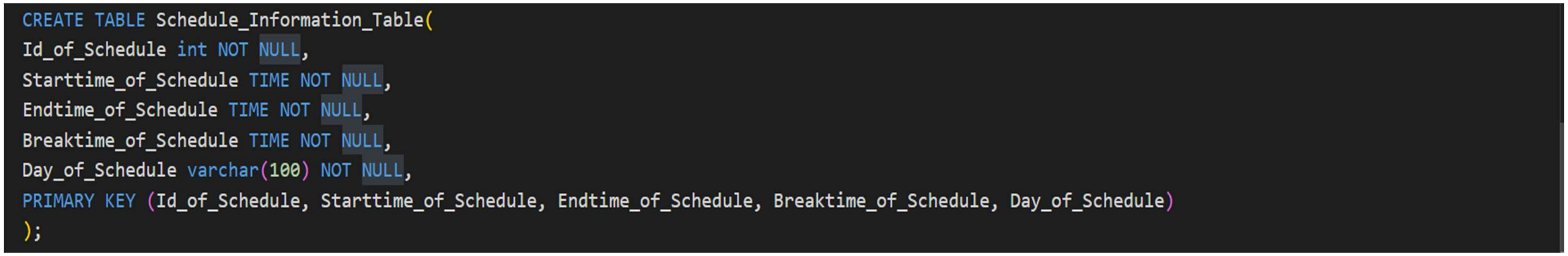
```sql
CREATE TABLE Appointment_Information_Table(
Id_of_Appointment int,
Date_of_Appointment DATE NOT NULL,
Starttime_of_Appointment TIME NOT NULL,
Endtime_of_Appointment TIME NOT NULL,
Status_of_Appointment varchar(100) NOT NULL,
PRIMARY KEY(Id_of_Appointment)
);
```

6. Creation of Patient Attended Appointment Table



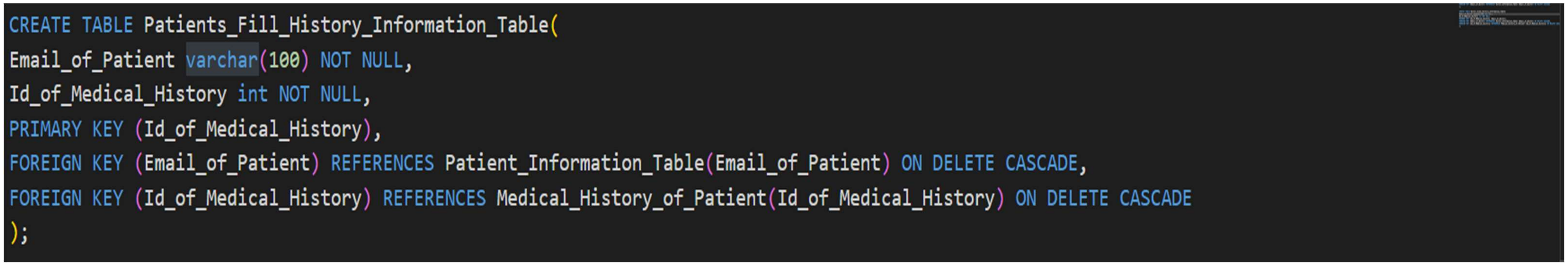
```sql
CREATE TABLE Patient_Attend_Appointments_Information_Table(
Id_of_Appointment int NOT NULL,
Email_of_Patient varchar(100) NOT NULL,
Concerns_of_Patient varchar(100) NOT NULL,
Symptoms_of_Patient varchar(100) NOT NULL,
PRIMARY KEY (Email_of_Patient, Id_of_Appointment),
FOREIGN KEY (Email_of_Patient) REFERENCES Patient_Information_Table (Email_of_Patient) ON DELETE CASCADE,
FOREIGN KEY (Id_of_Appointment) REFERENCES Appointment_Information_Table (Id_of_Appointment) ON DELETE CASCADE
);
```

7. Creation of Schedule Information Table



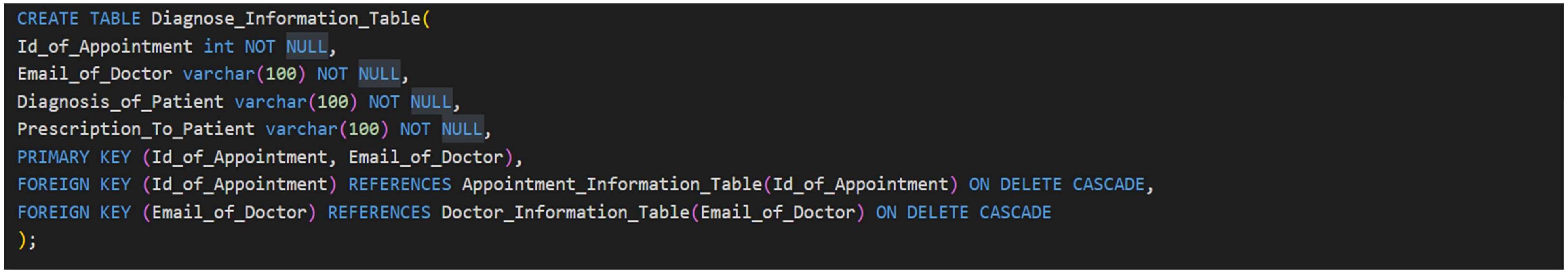
```sql
CREATE TABLE Schedule_Information_Table(
Id_of_Schedule int NOT NULL,
Starttime_of_Schedule TIME NOT NULL,
Endtime_of_Schedule TIME NOT NULL,
Breaktime_of_Schedule TIME NOT NULL,
Day_of_Schedule varchar(100) NOT NULL,
PRIMARY KEY (Id_of_Schedule, Starttime_of_Schedule, Endtime_of_Schedule, Breaktime_of_Schedule, Day_of_Schedule)
);
```

8. Creation of Patient Filling History Information Table



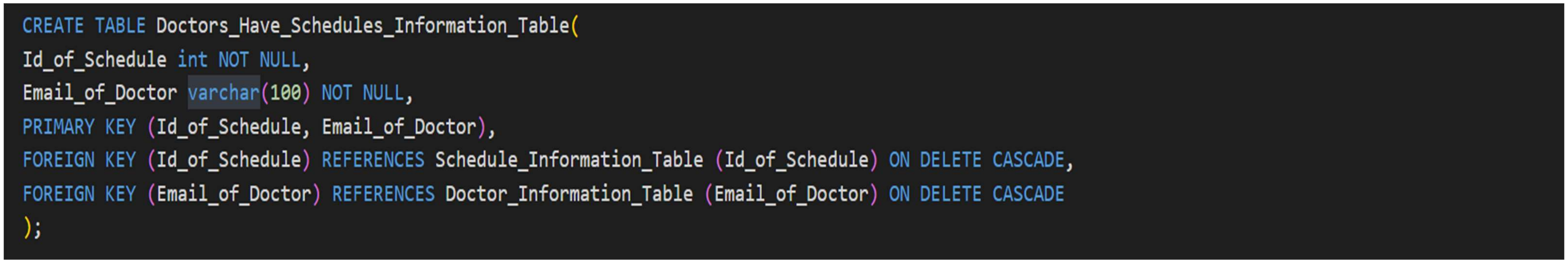
```sql
CREATE TABLE Patients_Fill_History_Information_Table(
Email_of_Patient varchar(100) NOT NULL,
Id_of_Medical_History int NOT NULL,
PRIMARY KEY (Id_of_Medical_History),
FOREIGN KEY (Email_of_Patient) REFERENCES Patient_Information_Table(Email_of_Patient) ON DELETE CASCADE,
FOREIGN KEY (Id_of_Medical_History) REFERENCES Medical_History_of_Patient(Id_of_Medical_History) ON DELETE CASCADE
);
```

9. Creation of Diagnose Information Table



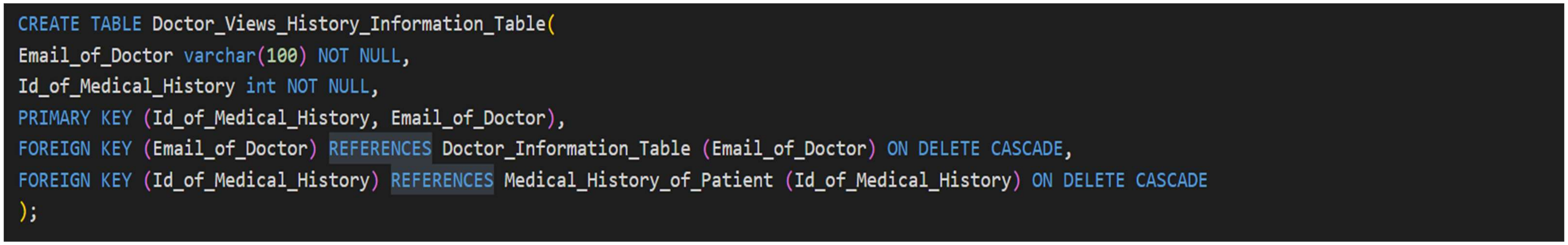
```sql
CREATE TABLE Diagnose_Information_Table(
Id_of_Appointment int NOT NULL,
Email_of_Doctor varchar(100) NOT NULL,
Diagnosis_of_Patient varchar(100) NOT NULL,
Prescription_To_Patient varchar(100) NOT NULL,
PRIMARY KEY (Id_of_Appointment, Email_of_Doctor),
FOREIGN KEY (Id_of_Appointment) REFERENCES Appointment_Information_Table(Id_of_Appointment) ON DELETE CASCADE,
FOREIGN KEY (Email_of_Doctor) REFERENCES Doctor_Information_Table(Email_of_Doctor) ON DELETE CASCADE
);
```

10. Creation of Schedules of Doctors Table



```sql
CREATE TABLE Doctors_Have_Schedules_Information_Table(
Id_of_Schedule int NOT NULL,
Email_of_Doctor varchar(100) NOT NULL,
PRIMARY KEY (Id_of_Schedule, Email_of_Doctor),
FOREIGN KEY (Id_of_Schedule) REFERENCES Schedule_Information_Table (Id_of_Schedule) ON DELETE CASCADE,
FOREIGN KEY (Email_of_Doctor) REFERENCES Doctor_Information_Table (Email_of_Doctor) ON DELETE CASCADE
);
```

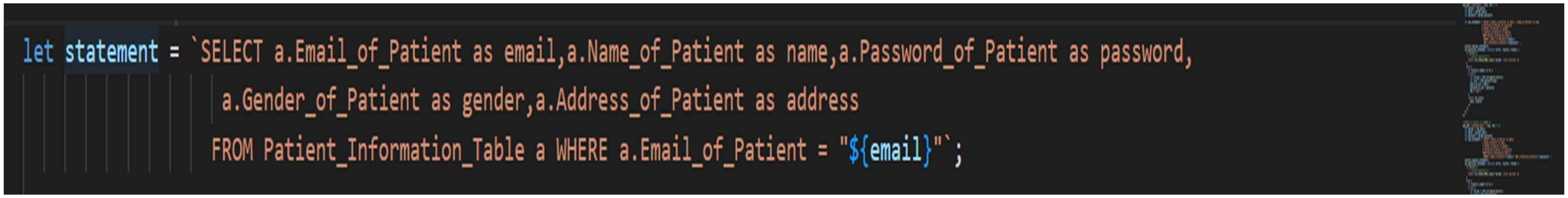
11. Creation of Doctors viewing Patient History Table



```sql
CREATE TABLE Doctor_Views_History_Information_Table(
Email_of_Doctor varchar(100) NOT NULL,
Id_of_Medical_History int NOT NULL,
PRIMARY KEY (Id_of_Medical_History, Email_of_Doctor),
FOREIGN KEY (Email_of_Doctor) REFERENCES Doctor_Information_Table (Email_of_Doctor) ON DELETE CASCADE,
FOREIGN KEY (Id_of_Medical_History) REFERENCES Medical_History_of_Patient (Id_of_Medical_History) ON DELETE CASCADE
);
```

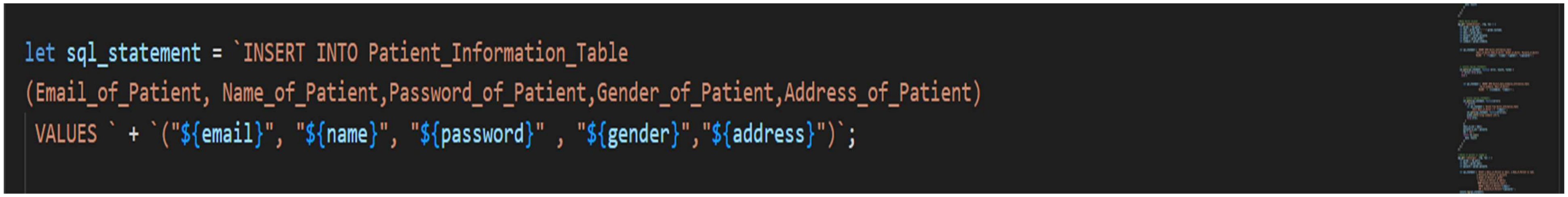
Queries for different Functionalities:

1. **Checking if Patient is already having account:**



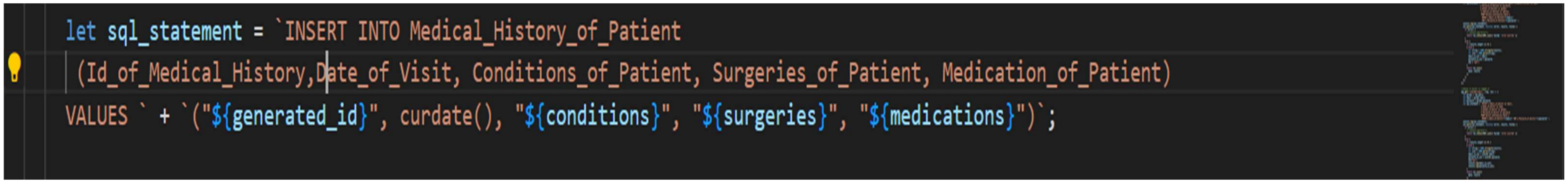
```
let statement = `SELECT a.Email_of_Patient as email,a.Name_of_Patient as name,a.Password_of_Patient as password,
                  a.Gender_of_Patient as gender,a.Address_of_Patient as address
                 FROM Patient_Information_Table a WHERE a.Email_of_Patient = "${email}"`;
```

2. **Inserting Patient Info to Create Patient's Account:**



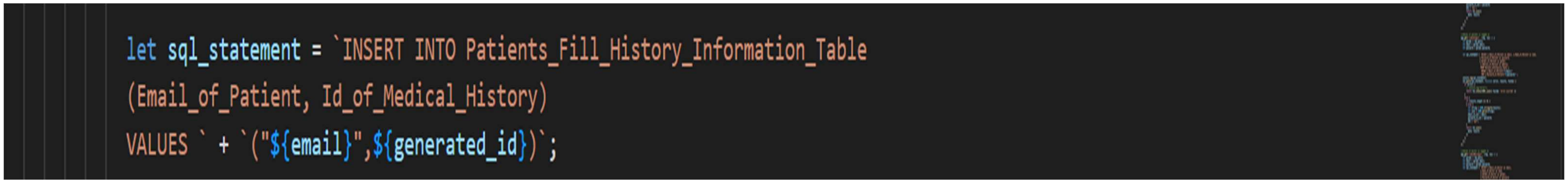
```
let sql_statement = `INSERT INTO Patient_Information_Table
(Email_of_Patient, Name_of_Patient,Password_of_Patient,Gender_of_Patient,Address_of_Patient)
 VALUES ` + `("${email}", "${name}", "${password}" , "${gender}","${address}")`;
```

3. **Inserting the Medical History of Patient Info Collected during Account Creation:**



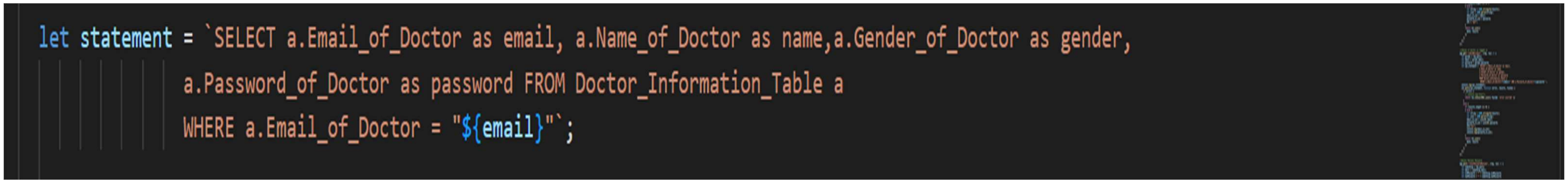
```
    let sql_statement = `INSERT INTO Medical_History_of_Patient
     (Id_of_Medical_History,Date_of_Visit, Conditions_of_Patient, Surgeries_of_Patient, Medication_of_Patient)
    VALUES ` + `("${generated_id}", curdate(), "${conditions}", "${surgeries}", "${medications}")`;
```

4. **Inserting a row into Patient Fill History Table, relating Patient History table to Patient Info Table:**



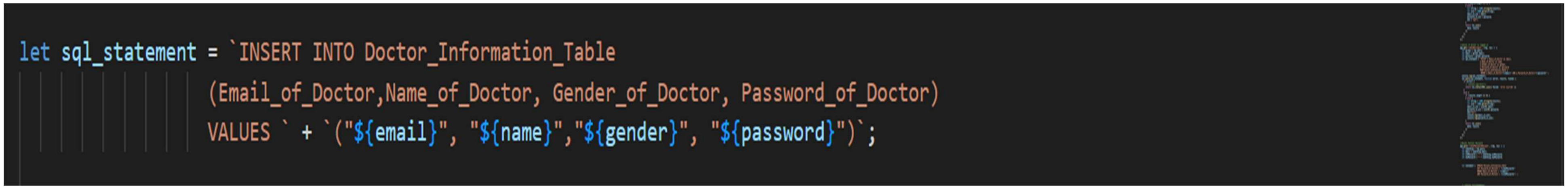
```
let sql_statement = `INSERT INTO Patients_Fill_History_Information_Table
(Email_of_Patient, Id_of_Medical_History)
VALUES ` + `("${email}",${generated_id})`;
```

5. **Check if Doctor is already having account:**



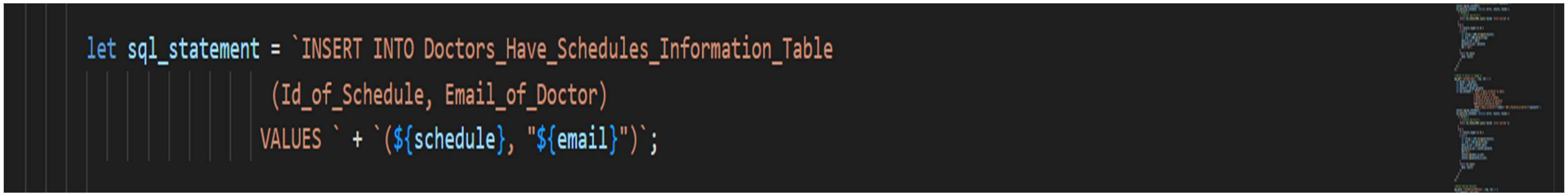
```
let statement = `SELECT a.Email_of_Doctor as email, a.Name_of_Doctor as name,a.Gender_of_Doctor as gender,
                a.Password_of_Doctor as password FROM Doctor_Information_Table a
                WHERE a.Email_of_Doctor = "${email}"`;
```

6. **Inserting Doctor Info to Create Doctor's Account:**



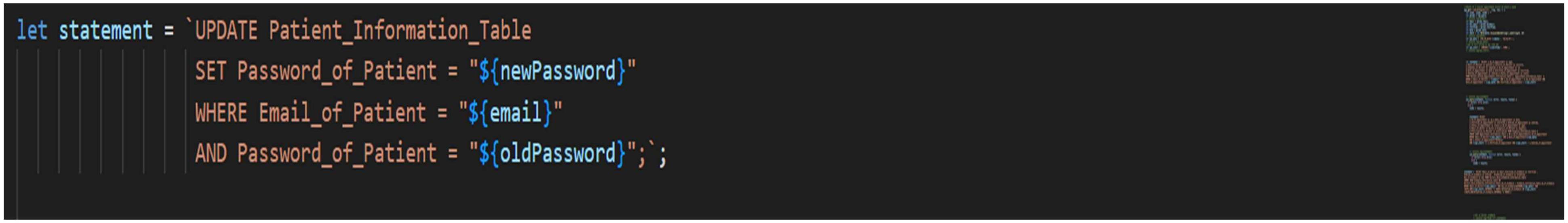
```
let sql_statement = `INSERT INTO Doctor_Information_Table
                    (Email_of_Doctor,Name_of_Doctor, Gender_of_Doctor, Password_of_Doctor)
                    VALUES ` + `("${email}", "${name}","${gender}", "${password}")`;
```

7. **Inserting Doctor's Availability Schedule Info:**



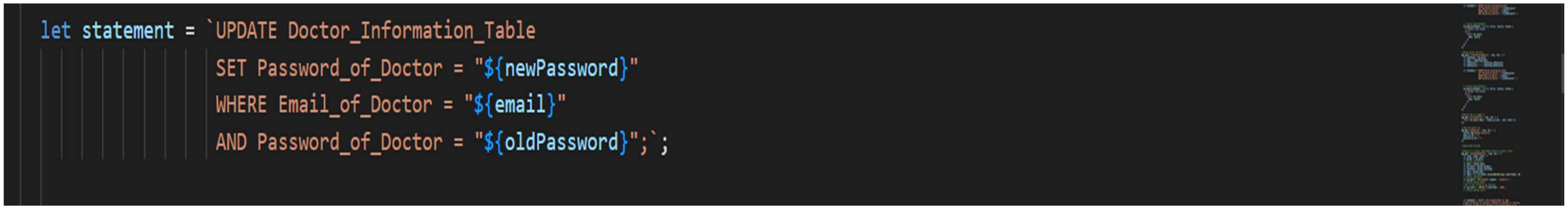
```
let sql_statement = `INSERT INTO Doctors_Have_Schedules_Information_Table
                     (Id_of_Schedule, Email_of_Doctor)
                    VALUES ` + `(${schedule}, "${email}")`;
```

8. **Updating Patient's Password:**



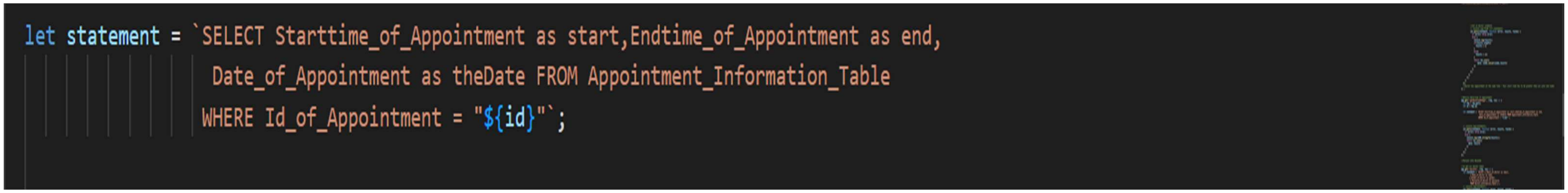
```
let statement = `UPDATE Patient_Information_Table
                SET Password_of_Patient = "${newPassword}"
                WHERE Email_of_Patient = "${email}"
                AND Password_of_Patient = "${oldPassword}";`;
```

9. **Updating Doctor's Password:**



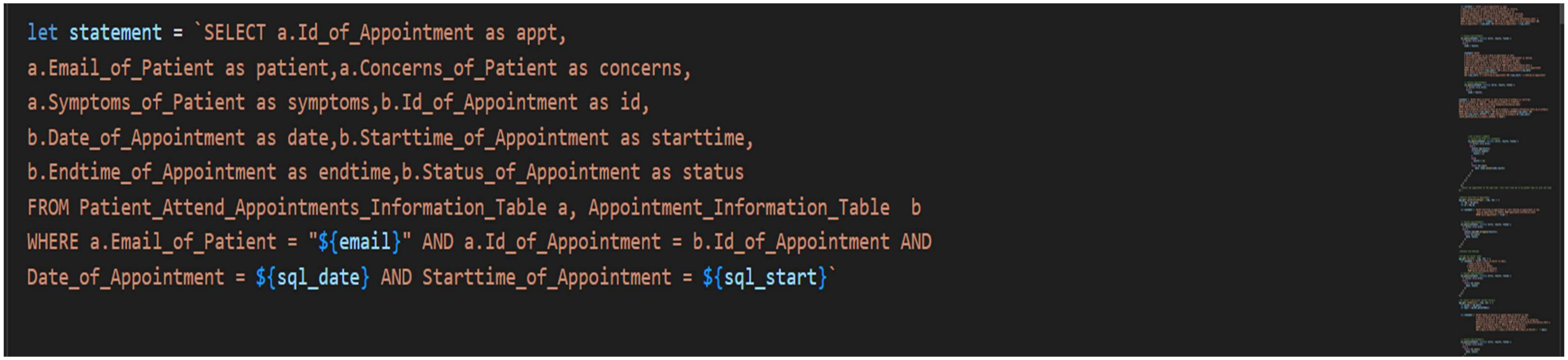
```
let statement = `UPDATE Doctor_Information_Table
                SET Password_of_Doctor = "${newPassword}"
                WHERE Email_of_Doctor = "${email}"
                AND Password_of_Doctor = "${oldPassword}";`;
```

10. Checking Date and Time of Appointment:

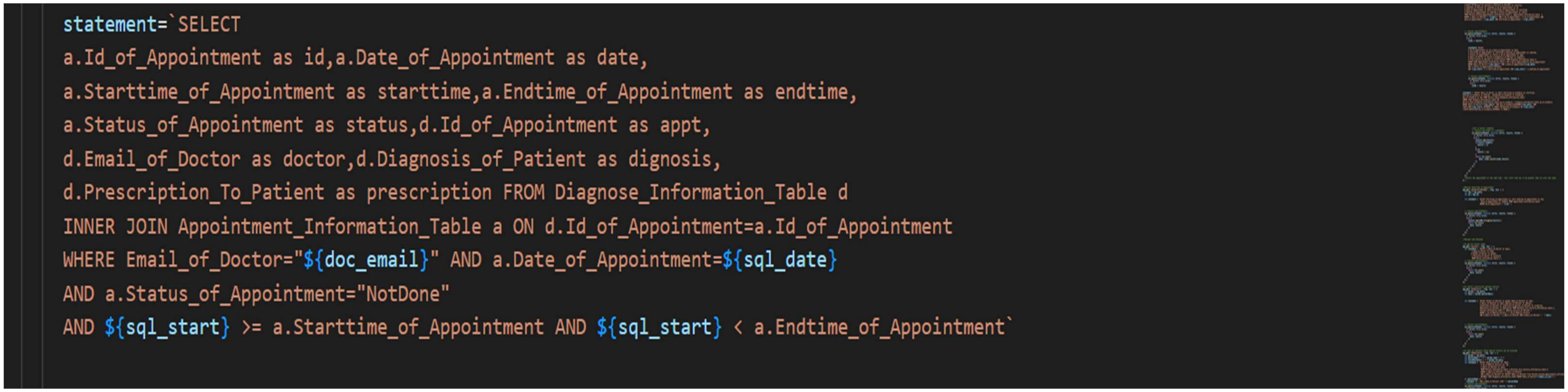


```
let statement = `SELECT Starttime_of_Appointment as start,Endtime_of_Appointment as end,
                 Date_of_Appointment as theDate FROM Appointment_Information_Table
                WHERE Id_of_Appointment = "${id}"`;
```

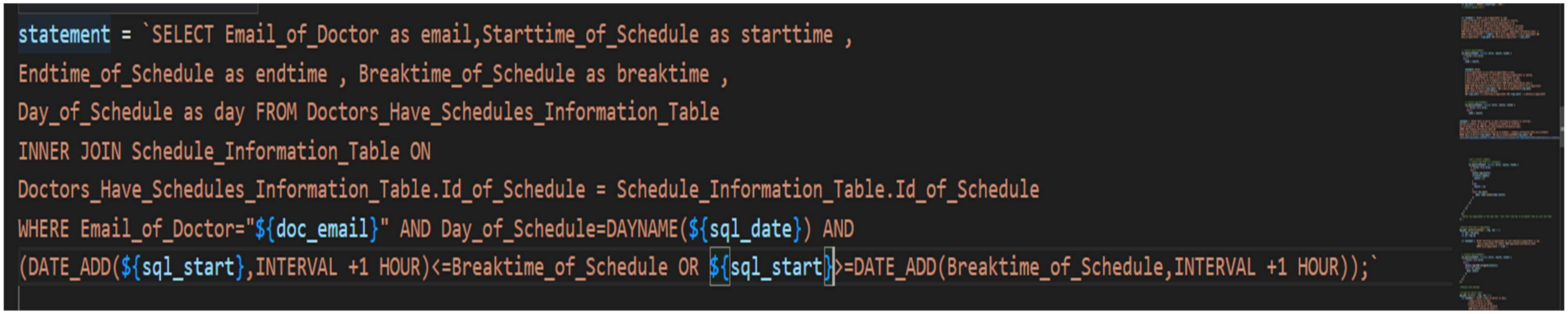
11. Checking If a Similar Appointments to avoid Clash:



```
let statement = `SELECT a.Id_of_Appointment as appt,
a.Email_of_Patient as patient,a.Concerns_of_Patient as concerns,
a.Symptoms_of_Patient as symptoms,b.Id_of_Appointment as id,
b.Date_of_Appointment as date,b.Starttime_of_Appointment as starttime,
b.Endtime_of_Appointment as endtime,b.Status_of_Appointment as status
FROM Patient_Attend_Appointments_Information_Table a, Appointment_Information_Table  b
WHERE a.Email_of_Patient = "${email}" AND a.Id_of_Appointment = b.Id_of_Appointment AND
Date_of_Appointment = ${sql_date} AND Starttime_of_Appointment = ${sql_start}`
```

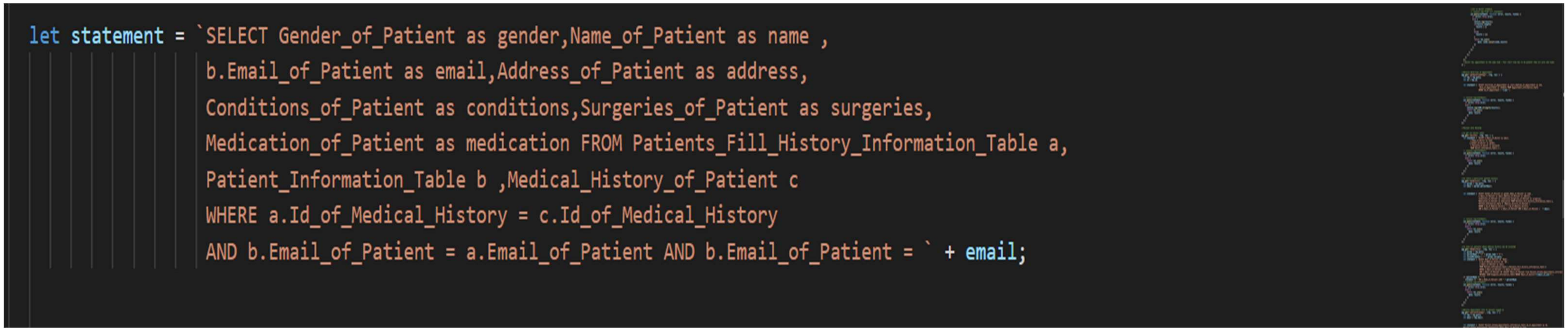


```
statement=`SELECT
a.Id_of_Appointment as id,a.Date_of_Appointment as date,
a.Starttime_of_Appointment as starttime,a.Endtime_of_Appointment as endtime,
a.Status_of_Appointment as status,d.Id_of_Appointment as appt,
d.Email_of_Doctor as doctor,d.Diagnosis_of_Patient as dignosis,
d.Prescription_To_Patient as prescription FROM Diagnose_Information_Table d
INNER JOIN Appointment_Information_Table a ON d.Id_of_Appointment=a.Id_of_Appointment
WHERE Email_of_Doctor="${doc_email}" AND a.Date_of_Appointment=${sql_date}
AND a.Status_of_Appointment="NotDone"
AND ${sql_start} >= a.Starttime_of_Appointment AND ${sql_start} < a.Endtime_of_Appointment`
```



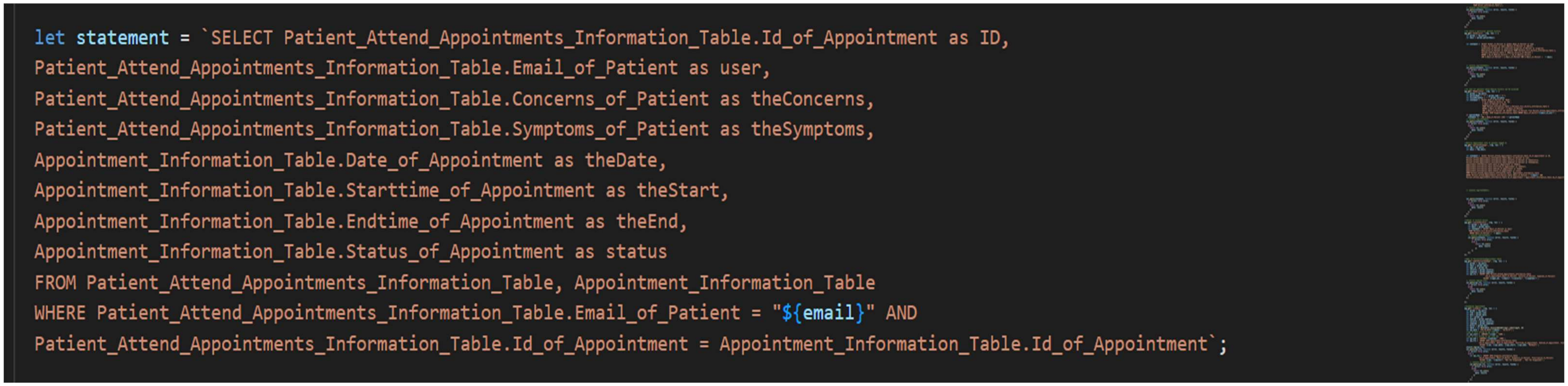
```
statement = `SELECT Email_of_Doctor as email,Starttime_of_Schedule as starttime ,
Endtime_of_Schedule as endtime , Breaktime_of_Schedule as breaktime ,
Day_of_Schedule as day FROM Doctors_Have_Schedules_Information_Table
INNER JOIN Schedule_Information_Table ON
Doctors_Have_Schedules_Information_Table.Id_of_Schedule = Schedule_Information_Table.Id_of_Schedule
WHERE Email_of_Doctor="${doc_email}" AND Day_of_Schedule=DAYNAME(${sql_date}) AND
(DATE_ADD(${sql_start},INTERVAL +1 HOUR)<=Breaktime_of_Schedule OR ${sql_start}>=DATE_ADD(Breaktime_of_Schedule,INTERVAL +1 HOUR));`
```

12. Selecting the Medical History of a particular Patient:



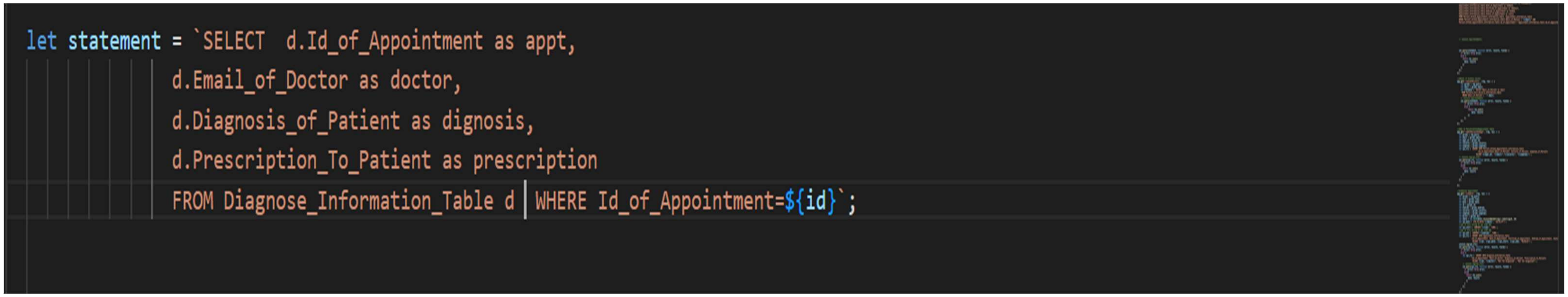
```
let statement = `SELECT Gender_of_Patient as gender,Name_of_Patient as name ,
                b.Email_of_Patient as email,Address_of_Patient as address,
                Conditions_of_Patient as conditions,Surgeries_of_Patient as surgeries,
                Medication_of_Patient as medication FROM Patients_Fill_History_Information_Table a,
                Patient_Information_Table b ,Medical_History_of_Patient c
                WHERE a.Id_of_Medical_History = c.Id_of_Medical_History
                AND b.Email_of_Patient = a.Email_of_Patient AND b.Email_of_Patient = ` + email;
```

13. Showing a Patient his/her Appointment's Info:



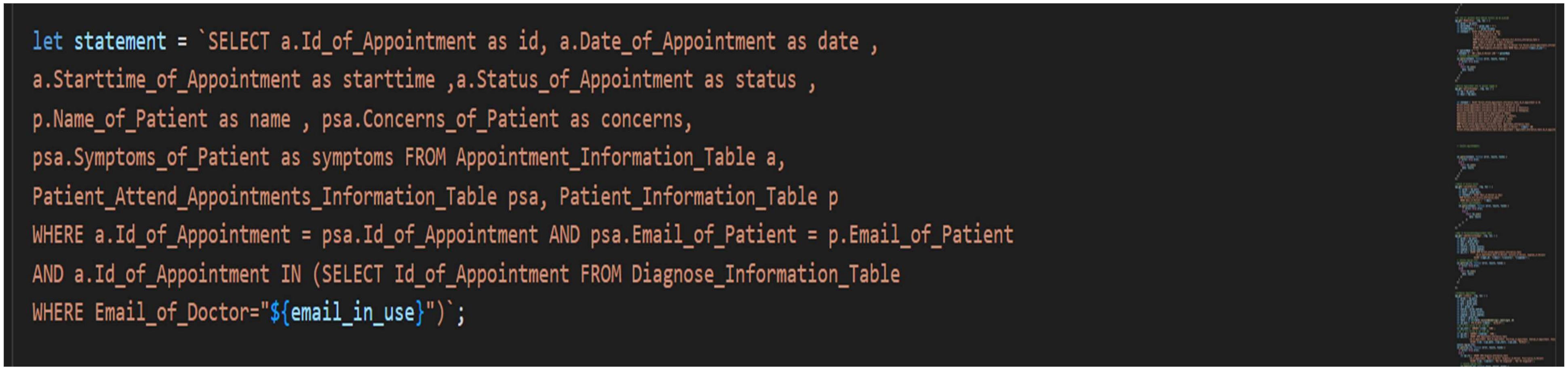
```
let statement = `SELECT Patient_Attend_Appointments_Information_Table.Id_of_Appointment as ID,
Patient_Attend_Appointments_Information_Table.Email_of_Patient as user,
Patient_Attend_Appointments_Information_Table.Concerns_of_Patient as theConcerns,
Patient_Attend_Appointments_Information_Table.Symptoms_of_Patient as theSymptoms,
Appointment_Information_Table.Date_of_Appointment as theDate,
Appointment_Information_Table.Starttime_of_Appointment as theStart,
Appointment_Information_Table.Endtime_of_Appointment as theEnd,
Appointment_Information_Table.Status_of_Appointment as status
FROM Patient_Attend_Appointments_Information_Table, Appointment_Information_Table
WHERE Patient_Attend_Appointments_Information_Table.Email_of_Patient = "${email}" AND
Patient_Attend_Appointments_Information_Table.Id_of_Appointment = Appointment_Information_Table.Id_of_Appointment`;
```

14. Showing Diagnosis Info which is filled by doctor:



```
let statement = `SELECT  d.Id_of_Appointment as appt,
             d.Email_of_Doctor as doctor,
             d.Diagnosis_of_Patient as dignosis,
             d.Prescription_To_Patient as prescription
             FROM Diagnose_Information_Table d WHERE Id_of_Appointment=${id}`;
```

15. Showing Appointments to a Doctor:



```
let statement = `SELECT a.Id_of_Appointment as id, a.Date_of_Appointment as date ,
a.Starttime_of_Appointment as starttime ,a.Status_of_Appointment as status ,
p.Name_of_Patient as name , psa.Concerns_of_Patient as concerns,
psa.Symptoms_of_Patient as symptoms FROM Appointment_Information_Table a,
Patient_Attend_Appointments_Information_Table psa, Patient_Information_Table p
WHERE a.Id_of_Appointment = psa.Id_of_Appointment AND psa.Email_of_Patient = p.Email_of_Patient
AND a.Id_of_Appointment IN (SELECT Id_of_Appointment FROM Diagnose_Information_Table
WHERE Email_of_Doctor="${email_in_use}")`;
```