INTRODUCTION

The following points will be considered when creating the database based on the provided information.

- Patients create login credentials, which consist of a username and password. They also register with their full name, address, date of birth, and insurance with optional fields for email address and phone number.
- Through the patient portal, patients schedule appointments, and the system confirms the doctor's availability.
- The date, time, department, status, and associated doctor are saved as the appointment details.
- During appointments, doctors update patient medical records with new diagnoses and prescriptions and review previous appointments and diagnoses.
- The patient appointment's status changes to completed after the appointment, and they can provide feedback.
- Patients are required to reschedule if their appointments are cancelled.
- Patient information is retained even if they leave, with a record of the departure date.

PART 1

DATABASE DESIGN AND NORMALIZATION

In database design, there are three design approach namely Top-down, Bottom-up and Hybrid approach. With the top-down approach to database design, you begin the process by figuring out the overall structure of the database system before getting into the specifics. The bottom-up method integrates individual database components into a larger system after thoroughly designing each one separately. I will be using a combination of both top-down and bottom-up methodologies, making it a hybrid approach.

The database design is created from the client’s requirements. Tables can be created with their attributes as the columns in the tables. The tables are:

- Patient table, which holds patient data such as login credentials for the patient portal and, if desired, contact information.

- Doctors table contains contact information and a list of specializations for doctors.
- Appointments table: records all appointments, together with the patient, doctor, department, date, time, and status.
- Medical Records Table: contains all of the information related to a given appointment's medical records, such as diagnoses, medication prescribed, and allergy information.
- Reviews table: contains feedback from patients for appointments.

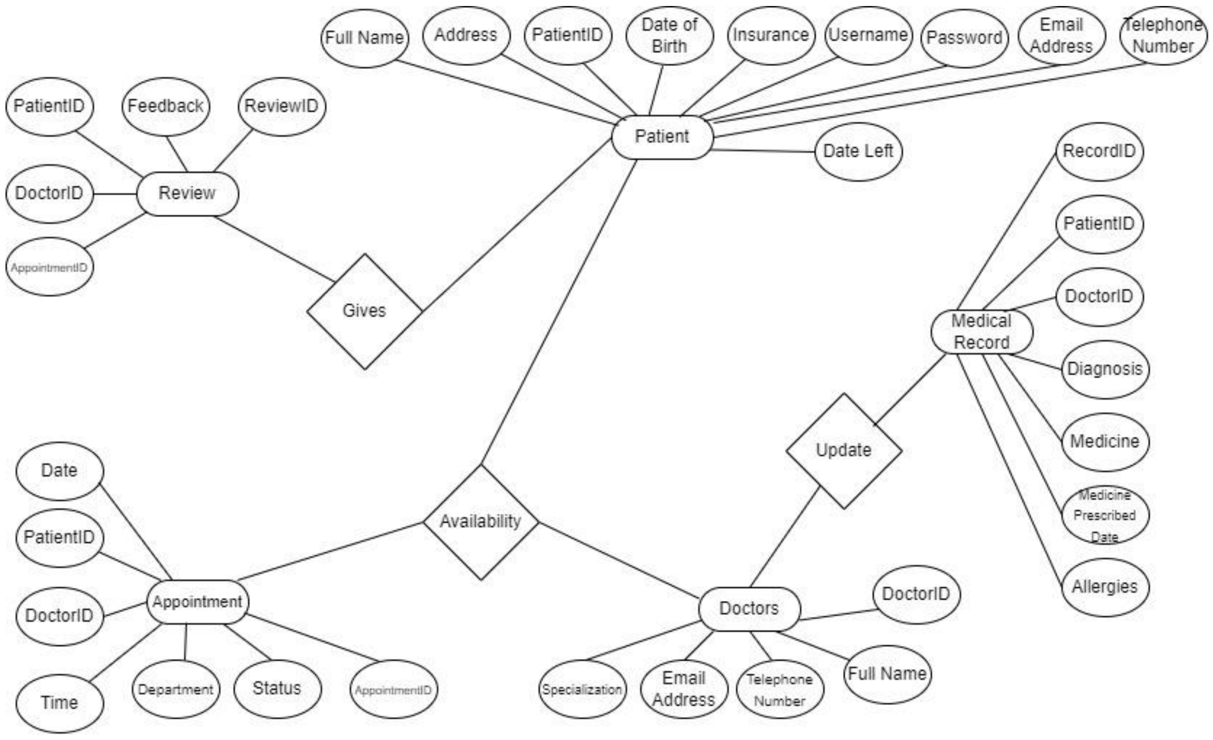


Figure 1.0

From the database design above, the entities are converted into tables. The tables are in an unnormalized form, so we normalized them to 3NF (Third Normal Form) to prevent insert, delete, and update anomalies as well as redundancies to enhance data integrity and management effectiveness.
A table cannot be in the third normal form unless it is in 1NF and 2NF first, and there cannot be any transitive dependencies between tables.
To be in First Normal Form (1NF), a table must meet these requirements:

- A primary key that uniquely identifies each row.

- All columns must contain atomic values, meaning no repeating groups or arrays within columns.

For the Second Normal Form (2NF), a table must;

- Be in 1NF
- All non-key attributes must be fully functional dependent on the primary key.
- Each column must depend on the entire primary key, and there should be no partial dependencies.

To normalize the tables to the Third Normal Form (3NF), the tables need to meet the requirements listed below.

- Fulfil the 2NF
- All attributes must depend only on the primary key and not on other non-key attributes.
- Some of the tables are split into two or more tables to achieve the required normalization.

The following tables are then created from the above ER Diagram:

Address

Column Name	**Description**	**Data Type**
AddressID	Unique identifier for address	INT (Primary Key)
Address1	The first line of the address	NVARCHAR(255)
Address2	The second line of the address	NVARCHAR(255)
City	City name	NVARCHAR(100)
Postcode	Postal code	NVARCHAR(20)
Country	Country name	NVARCHAR(100)

Table 1.0

Patient

Column Name	Description	Data Type
PatientID	Unique identifier for the patient	INT (Primary Key)
Username	Username for patient's login	NVARCHAR(50)
Password	Encrypted password for patient's login	VARBINARY(MAX)
FirstName	Patient's first name	NVARCHAR(100)
MiddleName	Patient's middle name (if applicable)	NVARCHAR(100)
LastName	Patient's last name	NVARCHAR(100)
AddressID	Foreign key referencing Address table	INT
Email	Patient's email address	NVARCHAR(255)
Telephone	Patient's telephone number	NVARCHAR(20)
Gender	Patient's gender	NVARCHAR(10)
DateOfBirth	Patient's date of birth	DATE
InsuranceNumber	Patient's insurance number	NVARCHAR(9)
StartDate	Date when the patient's record was created	DATE
EndDate	Date when the patient's record was deactivated	DATE
ReactivationDate	Date when the patient's record was reactivated	DATE

Table 1.1

Department

Column Name	Description	Data Type
DepartmentID	Unique identifier for the department	INT (Primary Key)
DepartmentName	Name of the department	NVARCHAR(255)

Table 1.2

Doctor

Column Name	**Description**	**Data Type**
DoctorID	Unique identifier for the doctor	INT (Primary Key)
FirstName	Doctor's first name	NVARCHAR(100)
MiddleName	Doctor's middle name (if applicable)	NVARCHAR(100)
LastName	Doctor's last name	NVARCHAR(100)
Telephone	Doctor's telephone number	NVARCHAR(20)
Email	Doctor's email address (computed column)	AS (computed)
Speciality	Doctor's speciality	NVARCHAR(255)
DepartmentID	Foreign key referencing Department table	INT

Table 1.3

Doctor Availability

Column Name	**Description**	**Data Type**
AvailabilityID	Unique identifier for the availability	INT (Primary Key)
DoctorID	Foreign key referencing Doctor's table	INT
DaysAvailable	Days when the doctor is available	NVARCHAR(50)
StartTime	Start time of availability	TIME
EndTime	End time of availability	TIME
Status	Status of availability (e.g., available, not available)	NVARCHAR(25)

Table 1.4

Medical Record

Column Name	Description	Data Type
RecordID	Unique identifier for the medical record	INT (Primary Key)
PatientID	Foreign key referencing Patient's table	INT
DoctorID	Foreign key referencing Doctor's table	INT
Diagnosis	Diagnosis information for the patient	NVARCHAR(MAX)
Allergies	Allergies information for the patient	NVARCHAR(MAX)
Note	Additional notes or comments on the record	NVARCHAR(MAX)

Table 1.5

Appointment

Column Name	Description	Data Type
AppointmentID	Unique identifier for the appointment	INT (Primary Key)
PatientID	Foreign key referencing Patient's table	INT
AvailabilityID	Foreign key referencing DoctorAvailability table	INT
AppointmentDate	Date of the appointment	DATE
AppointmentTime	Time of the appointment	TIME
AppointmentType	Type of appointment (e.g., regular checkup, follow-up)	NVARCHAR(100)
Status	Status of the appointment (e.g., Pending, Cancelled)	NVARCHAR(50)
Notes	Additional notes or comments for the appointment	NVARCHAR(MAX)

Table 1.6

Past Appointment

Column Name	**Description**	**Data Type**
PastAppointmentID	Unique identifier for the past appointment	INT (Primary Key)
PatientID	Foreign key referencing Patient table	INT
AvailabilityID	Foreign key referencing DoctorAvailability table	INT
AppointmentDate	Date of the appointment	DATE
AppointmentTime	Time of the appointment	TIME
AppointmentType	Type of appointment (e.g., regular checkup, follow-up)	NVARCHAR(100)
Status	Status of the appointment (e.g., Pending, Canceled)	NVARCHAR(50)
Notes	Additional notes or comments for the appointment	NVARCHAR(MAX)

Table 1.7

Medicine

Column Name	**Description**	**Data Type**
MedicineID	Unique identifier for the medicine	INT (Primary Key)
MedicineName	Name of the medicine	NVARCHAR(255)
Manufacturer	Manufacturer of the medicine	NVARCHAR(255)
Description	Description of the medicine	NVARCHAR(MAX)

Table 1.8

Prescription

Column Name	Description	Data Type
PrescriptionID	Unique identifier for the prescription	INT (Primary Key)
AppointmentID	Foreign key referencing Appointment table	INT
MedicineID	Foreign key referencing Medicine table	INT
PrescriptionDate	Date of the prescription	DATE
PrescriptionTime	Time of the prescription	TIME
Dosage	Dosage information for the medicine	NVARCHAR(100)
Notes	Additional notes or instructions for the prescription	NVARCHAR(MAX)

Table 1.9

Review

Column Name	Description	Data Type
ReviewID	Unique identifier for the review	INT (Primary Key)
PastAppointmentID	Foreign key referencing PastAppointment table	INT
ReviewDate	Date of the review	DATE
ReviewTime	Time of the review	TIME
Rating	Rating given by the patient (1 to 5)	INT
Comments	Comments or feedback provided by the patient	NVARCHAR(MAX)

Table 2.0

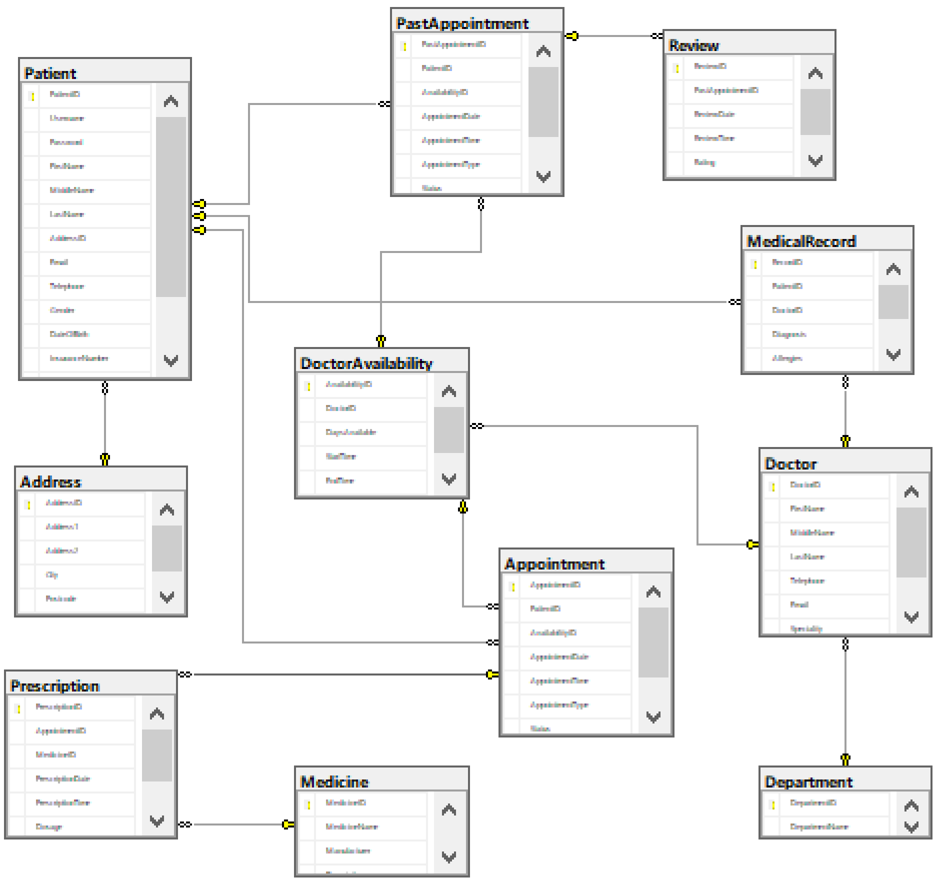


Figure 1.1

Some of the key functionalities considered are:

- **Primary Keys and Foreign Keys:** To preserve data integrity, foreign keys are used to form relationships between tables and Primary keys serve as a unique identifier for each record in a table.
- The syntax IDENTITY(seed, increment) allows specifying the starting value (seed) and the increment value (increment) for the identity column and it is used for the Primary keys to ensure that all rows have a unique value.
- **Constraints:** Where necessary constraints are used such as UNIQUE, CHECK etc.
- Computed Columns
- **Data Types:** Suitable data types for every column according to the kind of information it will store. For textual data, use NVARCHAR; for numerical identifiers, use INT; for date and time values, use DATE and TIME; etc.

<u>Cardinality</u>

Address - Patient:

One-to-Many (1 to N) relationship.

Each patient has one address (via AddressID in the Patient table).

Each address can be associated with multiple patients.

Department - Doctor:

One-to-Many (1 to N) relationship.

Each doctor belongs to one department (via DepartmentID in the Doctor table).

Each department can have multiple doctors.

Doctor - DoctorAvailability:

One-to-Many (1 to N) relationship.

Each doctor can have multiple availability slots (via DoctorID in the DoctorAvailability table).

Each availability slot belongs to one doctor.

Patient - MedicalRecord:

One-to-Many (1 To N) relationship.

Each patient can have multiple medical records (via PatientID in the MedicalRecord table).

Each medical record belongs to one patient.

Patient - Appointment:

One-to-Many (1 to N) relationship.

Each patient can have multiple appointments (via PatientID in the Appointment table).

Each appointment belongs to one patient.

DoctorAvailability - Appointment / PastAppointment:

One-to-Many (1 to N) relationship.

Each availability slot can have multiple appointments (via AvailabilityID in the Appointment table and PastAppointment table).

Each appointment belongs to one available slot.

Appointment - Prescription:

One-to-Many (1 to N) relationship.

Each appointment can have multiple prescriptions (via AppointmentID in the Prescription table).

Each prescription belongs to one appointment.

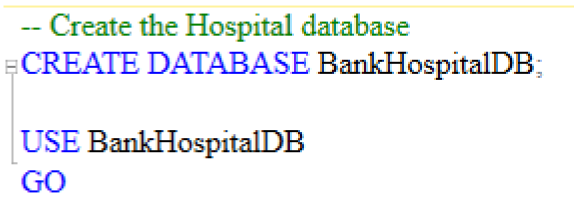
PastAppointment - Review:

One-to-One (1 to 1) relationship.

Each past appointment can have one review (via PastAppointmentID in the Review table).

Each review belongs to one past appointment.

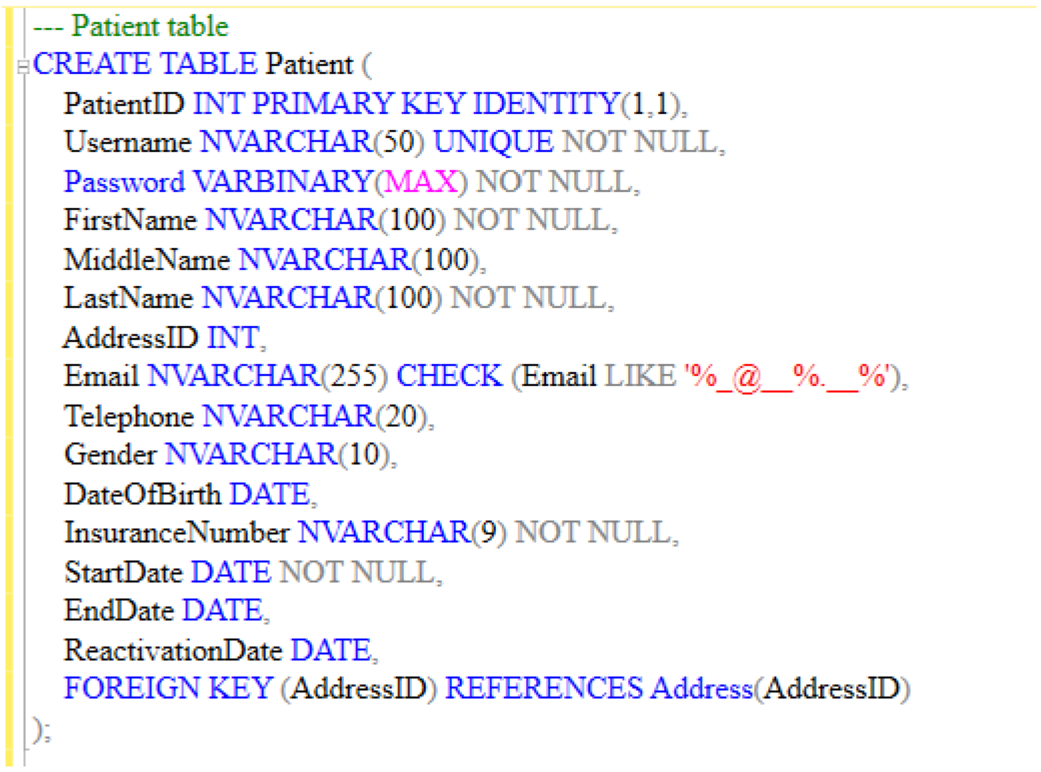
The creation of the database is depicted in the figure, and the 'GO' function guarantees that all queries will be executed in the freshly established database environment.



```
-- Create the Hospital database
CREATE DATABASE BankHospitalDB;

USE BankHospitalDB
GO
```

Figure 1.2

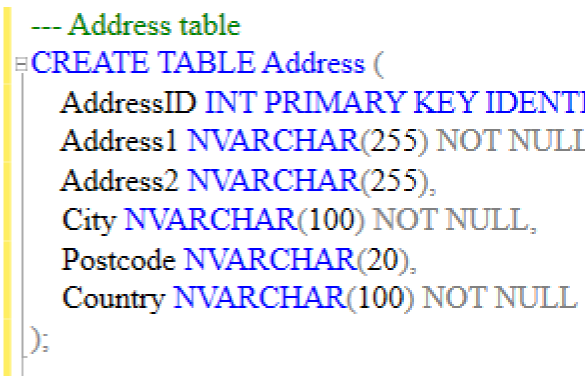
The primary key, patient ID, is in INT and is generated by the system by increasing by 1 for each input. Additionally, the username is UNIQUE to guarantee that the value is distinct across all of the table's records. Since passwords are usually hashed for security purposes, the password datatype is VARBINARY. The data types of the other columns, like FirstName, MiddleName, LastName, etc., are suitable for the kind of information they are meant to hold (e.g., DATE for date values, NVARCHAR for variable-length character data). Data consistency and integrity within the table are guaranteed by constraints such as CHECK, UNIQUE, and NOT NULL. The address table is referred to by the address ID (Foreign key) on the patient table.



```sql
--- Patient table
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY IDENTITY(1,1),
    Username NVARCHAR(50) UNIQUE NOT NULL,
    Password VARBINARY(MAX) NOT NULL,
    FirstName NVARCHAR(100) NOT NULL,
    MiddleName NVARCHAR(100),
    LastName NVARCHAR(100) NOT NULL,
    AddressID INT,
    Email NVARCHAR(255) CHECK (Email LIKE '%_@__%.__%'),
    Telephone NVARCHAR(20),
    Gender NVARCHAR(10),
    DateOfBirth DATE,
    InsuranceNumber NVARCHAR(9) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE,
    ReactivationDate DATE,
    FOREIGN KEY (AddressID) REFERENCES Address(AddressID)
);
```

Figure 1.3

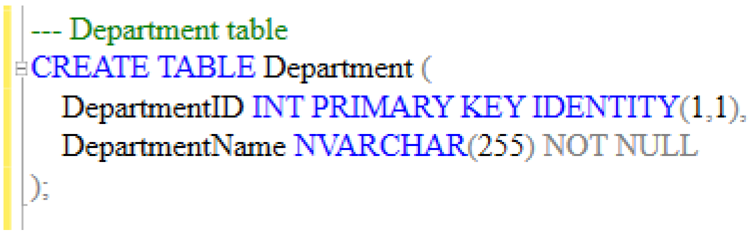
The Address table contains an automatically generated address ID which is the Primary Key with Integer data type. The other columns' nvarchar choices are made to support addresses written in non-English and special character



```sql
--- Address table
CREATE TABLE Address (
    AddressID INT PRIMARY KEY IDENTITY(1,1),
    Address1 NVARCHAR(255) NOT NULL,
    Address2 NVARCHAR(255),
    City NVARCHAR(100) NOT NULL,
    Postcode NVARCHAR(20),
    Country NVARCHAR(100) NOT NULL
);
```

Figure 1.4

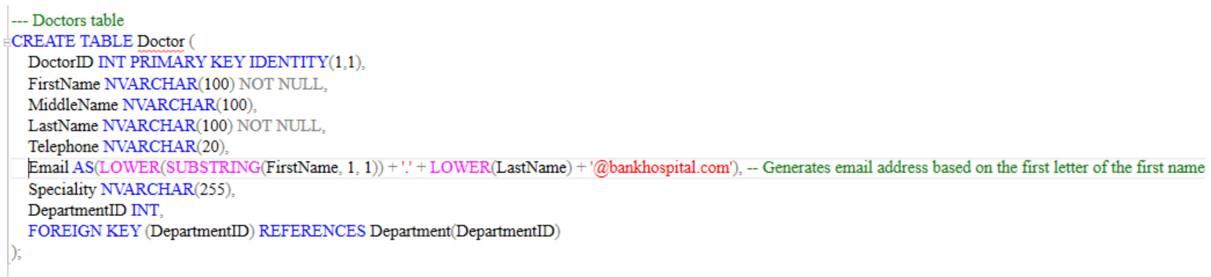
Each department is identified by a unique DepartmentID, and its corresponding name is stored in the DepartmentName field.



```sql
--- Department table
CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName NVARCHAR(255) NOT NULL
);
```

Figure 1.5

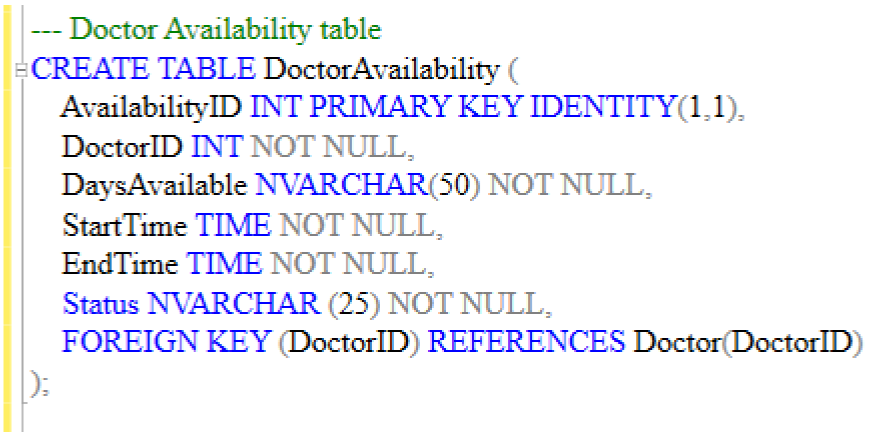
The computed column for the email address makes it easier to generate email addresses for doctors based only on their names, and the DoctorID functions as the primary key. The foreign key that creates a connection to the department table is called DepartmentID.



```sql
--- Doctors table
CREATE TABLE Doctor (
    DoctorID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(100) NOT NULL,
    MiddleName NVARCHAR(100),
    LastName NVARCHAR(100) NOT NULL,
    Telephone NVARCHAR(20),
    Email AS(LOWER(SUBSTRING(FirstName, 1, 1)) + '.' + LOWER(LastName) + '@bankhospital.com'), -- Generates email address based on the first letter of the first name
    Speciality NVARCHAR(255),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

Figure 1.6

The availability ID (INT) serves as the primary key, and the other columns contain the appropriate data types. The efficient administration of doctor availability, including the days and times they are available as well as their availability status, is made possible by this table structure. To guarantee that the DoctorID in DoctorAvailability matches a legitimate DoctorID in the Doctor table, a foreign key constraint creates a relationship with the Doctor table.

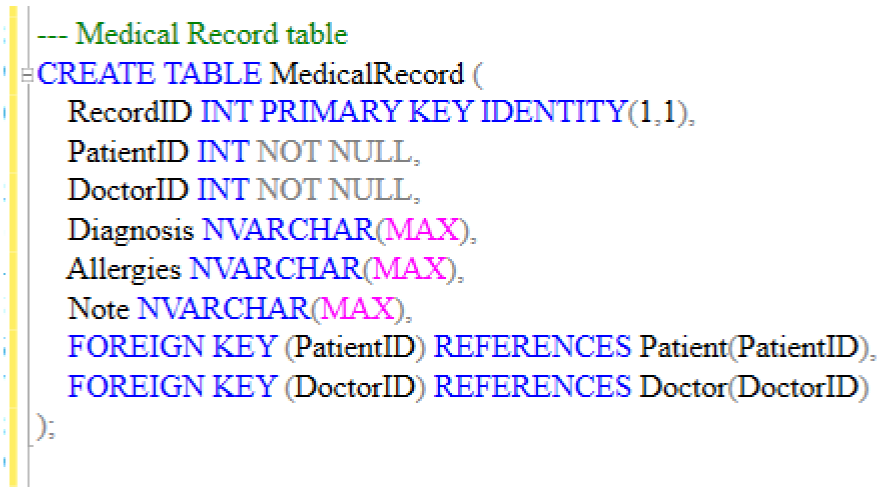


```sql
--- Doctor Availability table
CREATE TABLE DoctorAvailability (
    AvailabilityID INT PRIMARY KEY IDENTITY(1,1),
    DoctorID INT NOT NULL,
    DaysAvailable NVARCHAR(50) NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    Status NVARCHAR (25) NOT NULL,
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);
```

Figure 1.7

The primary key acts as each medical record's primary identifier and guarantees uniqueness.

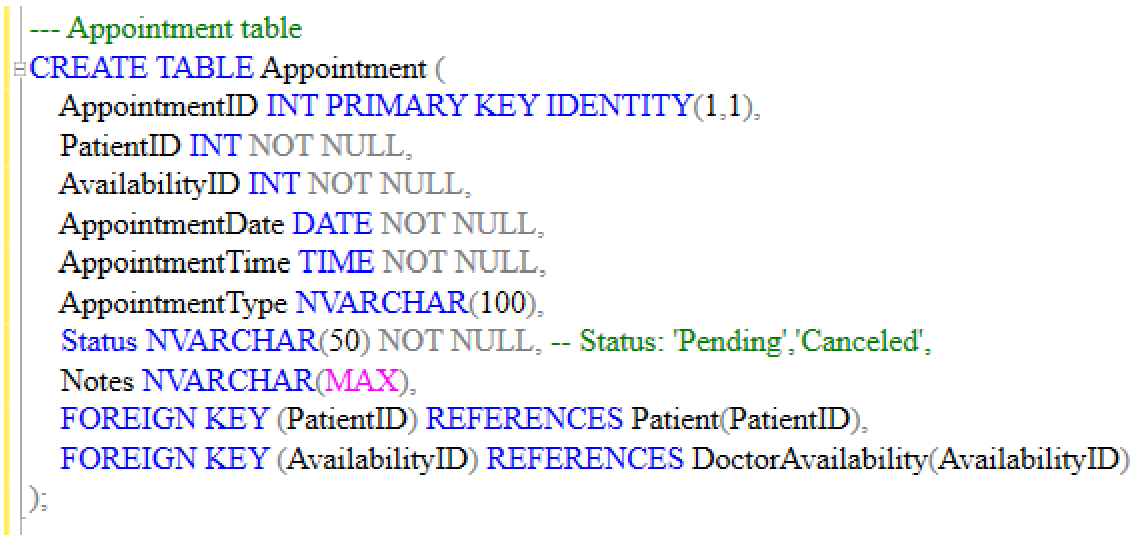
To guarantee that the PatientID in MedicalRecord corresponds to a valid PatientID in the Patient table, the foreign key creates a relationship with the Patient table. To guarantee that the DoctorID in MedicalRecord matches a legitimate DoctorID in the Doctor table, it also creates a relationship with the Doctor table.



```sql
--- Medical Record table
CREATE TABLE MedicalRecord (
    RecordID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    DoctorID INT NOT NULL,
    Diagnosis NVARCHAR(MAX),
    Allergies NVARCHAR(MAX),
    Note NVARCHAR(MAX),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);
```

Figure 1.8

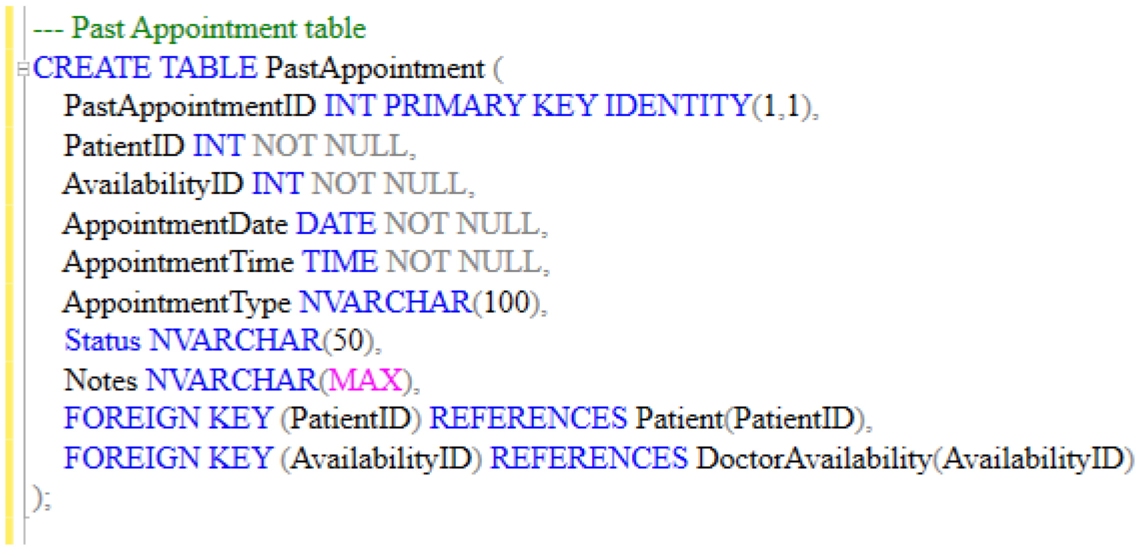
The table has the appointmentID as the Primary key, and links with the Patient and Doctor availability table via the patient and AvailabiltyID as foreign keys. This ensures that the patient can book an appointment when and if the Doctor is available



```sql
--- Appointment table
CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    AvailabilityID INT NOT NULL,
    AppointmentDate DATE NOT NULL,
    AppointmentTime TIME NOT NULL,
    AppointmentType NVARCHAR(100),
    Status NVARCHAR(50) NOT NULL, -- Status: 'Pending','Canceled',
    Notes NVARCHAR(MAX),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (AvailabilityID) REFERENCES DoctorAvailability(AvailabilityID)
);
```

Figure 1.9

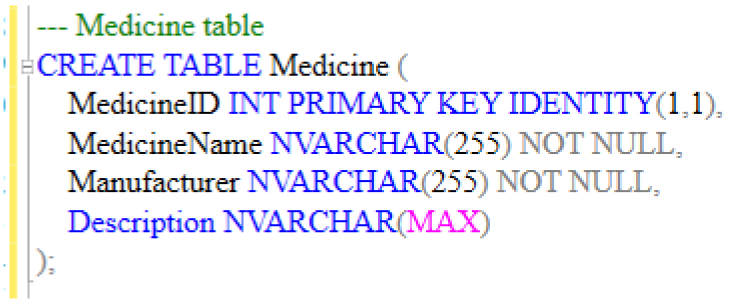
The table has the PastappointmentID as the Primary key, and links with the Patient and Doctor availability table via the PatientID and AvailabiltyID as foreign keys. This ensures that the patient can book an appointment when and if the Doctor is available



```
--- Past Appointment table
CREATE TABLE PastAppointment (
    PastAppointmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL,
    AvailabilityID INT NOT NULL,
    AppointmentDate DATE NOT NULL,
    AppointmentTime TIME NOT NULL,
    AppointmentType NVARCHAR(100),
    Status NVARCHAR(50),
    Notes NVARCHAR(MAX),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (AvailabilityID) REFERENCES DoctorAvailability(AvailabilityID)
);
```

Figure 2.0

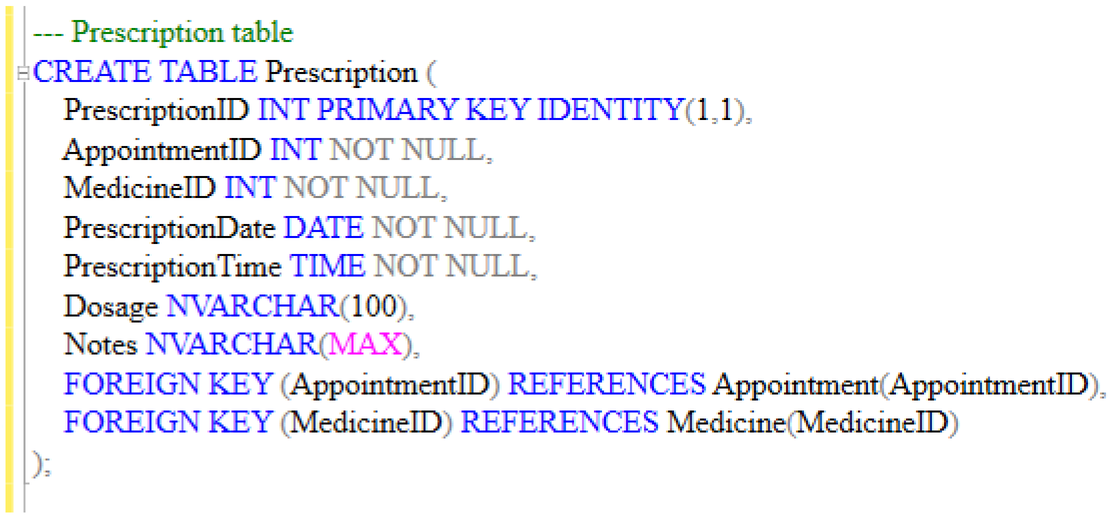
This table structure allows for the storage of information about different medicines, including their names, manufacturers, and descriptions. The primary key MedicineID ensures that each medicine is uniquely identified within the table.



```
--- Medicine table
CREATE TABLE Medicine (
    MedicineID INT PRIMARY KEY IDENTITY(1,1),
    MedicineName NVARCHAR(255) NOT NULL,
    Manufacturer NVARCHAR(255) NOT NULL,
    Description NVARCHAR(MAX)
);
```

Figure 2.1

This table structure allows for the storage of prescription information, including the appointment associated with the prescription, the medicine prescribed, the prescription date and time, dosage, and any additional notes. Having the PrescriptionID as the primary key, the table links each prescription to valid appointments and medicines Via the AppointmentID and MedicineID as the foreign keys.



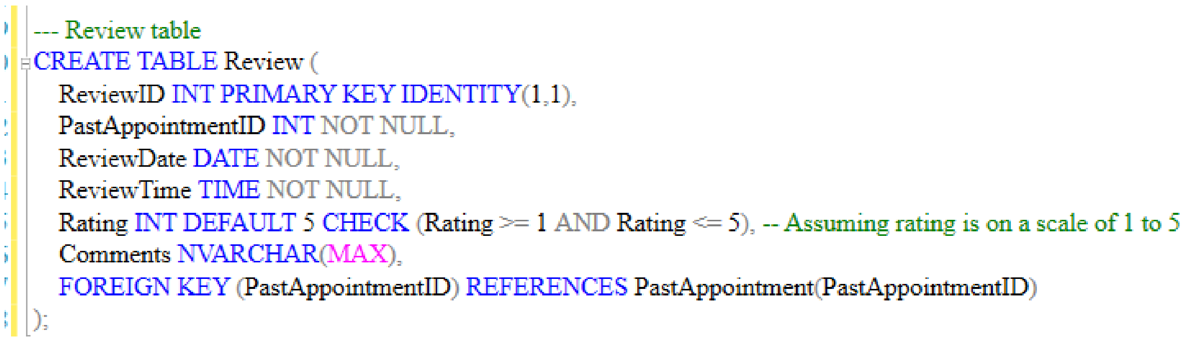
```sql
--- Prescription table
CREATE TABLE Prescription (
    PrescriptionID INT PRIMARY KEY IDENTITY(1,1),
    AppointmentID INT NOT NULL,
    MedicineID INT NOT NULL,
    PrescriptionDate DATE NOT NULL,
    PrescriptionTime TIME NOT NULL,
    Dosage NVARCHAR(100),
    Notes NVARCHAR(MAX),
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID),
    FOREIGN KEY (MedicineID) REFERENCES Medicine(MedicineID)
);
```

Figure 2.2

The primary key 'ReviewID' ensures uniqueness and serves as the primary identifier for each review.

The Foreign key constraint establishes a relationship with the PastAppointment table, ensuring that the PastAppointmentID in Review corresponds to a valid PastAppointmentID in the PastAppointment table.

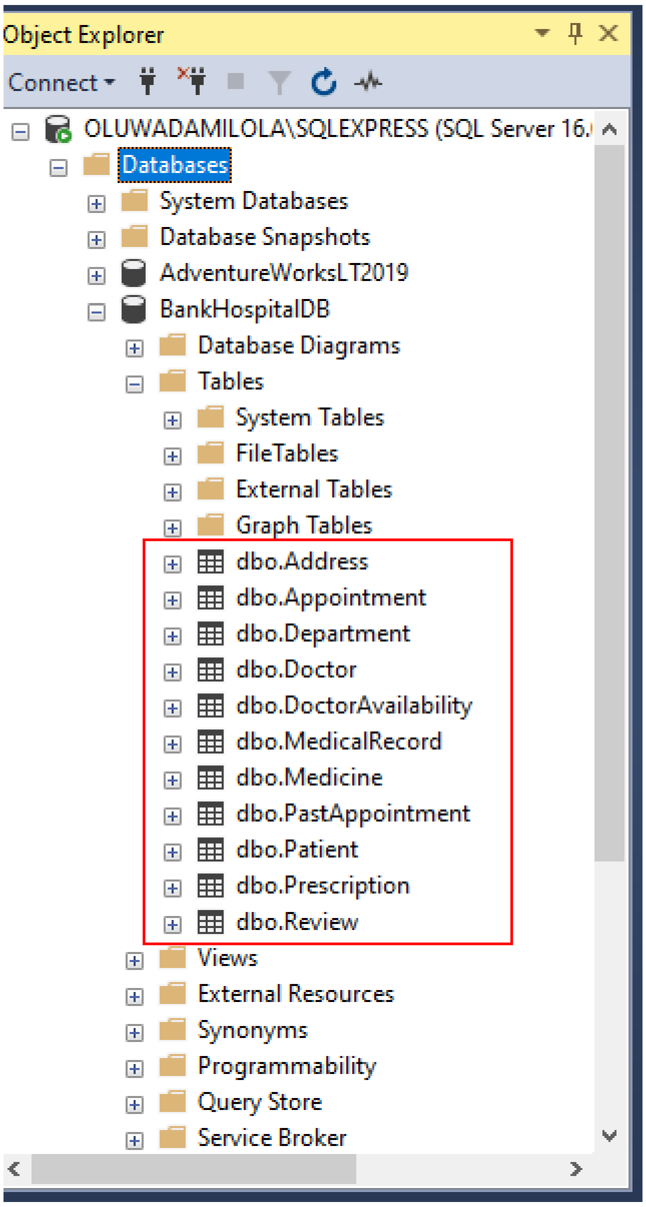
Also, the Check constraint ensures that the rating is within the specified range of 1 to 5.



```sql
--- Review table
CREATE TABLE Review (
    ReviewID INT PRIMARY KEY IDENTITY(1,1),
    PastAppointmentID INT NOT NULL,
    ReviewDate DATE NOT NULL,
    ReviewTime TIME NOT NULL,
    Rating INT DEFAULT 5 CHECK (Rating >= 1 AND Rating <= 5), -- Assuming rating is on a scale of 1 to 5
    Comments NVARCHAR(MAX),
    FOREIGN KEY (PastAppointmentID) REFERENCES PastAppointment(PastAppointmentID)
);
```

Figure 2.3

The figure below shows all the tables created in the database.



Object Explorer

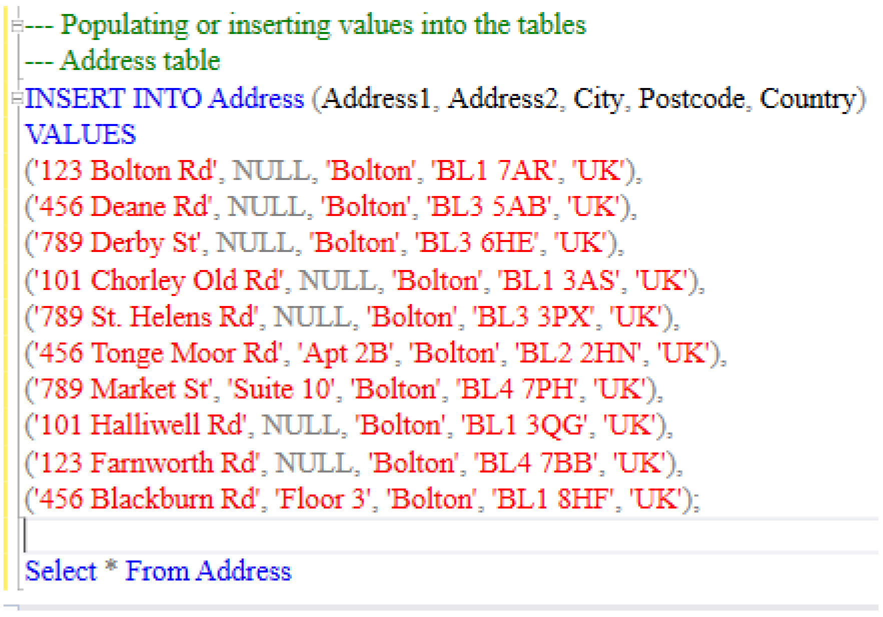
Connect

- OLUWADAMILOLA\SQLEXPRESS (SQL Server 16.
  - Databases
    - System Databases
    - Database Snapshots
    - AdventureWorksLT2019
    - BankHospitalDB
      - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - External Tables
        - Graph Tables
        - dbo.Address
        - dbo.Appointment
        - dbo.Department
        - dbo.Doctor
        - dbo.DoctorAvailability
        - dbo.MedicalRecord
        - dbo.Medicine
        - dbo.PastAppointment
        - dbo.Patient
        - dbo.Prescription
        - dbo.Review
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Query Store
      - Service Broker

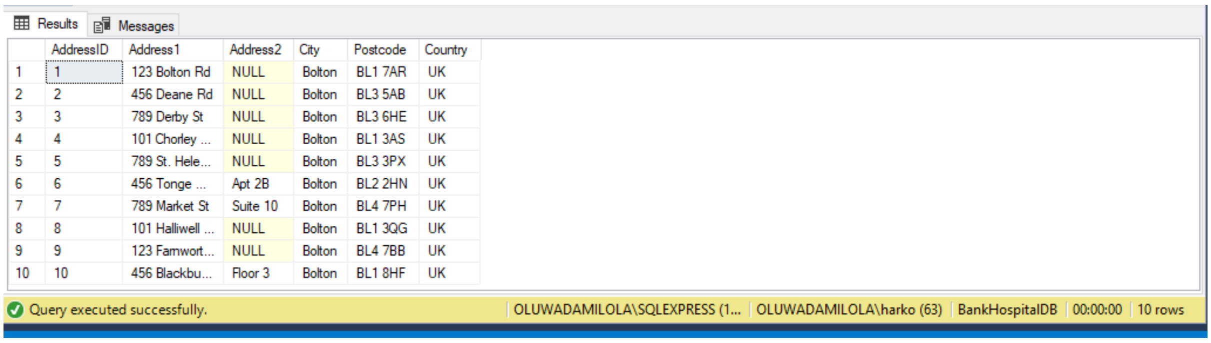
Figure 2.4

Inserting or populating the tables

After populating the tables, the SELECT function is used to display the results to confirm if the corresponding tables were properly populated as shown below.



```
--- Populating or inserting values into the tables
--- Address table
INSERT INTO Address (Address1, Address2, City, Postcode, Country)
VALUES
('123 Bolton Rd', NULL, 'Bolton', 'BL1 7AR', 'UK'),
('456 Deane Rd', NULL, 'Bolton', 'BL3 5AB', 'UK'),
('789 Derby St', NULL, 'Bolton', 'BL3 6HE', 'UK'),
('101 Chorley Old Rd', NULL, 'Bolton', 'BL1 3AS', 'UK'),
('789 St. Helens Rd', NULL, 'Bolton', 'BL3 3PX', 'UK'),
('456 Tonge Moor Rd', 'Apt 2B', 'Bolton', 'BL2 2HN', 'UK'),
('789 Market St', 'Suite 10', 'Bolton', 'BL4 7PH', 'UK'),
('101 Halliwell Rd', NULL, 'Bolton', 'BL1 3QG', 'UK'),
('123 Farnworth Rd', NULL, 'Bolton', 'BL4 7BB', 'UK'),
('456 Blackburn Rd', 'Floor 3', 'Bolton', 'BL1 8HF', 'UK');

Select * From Address
```



Results | Messages

	AddressID	Address1	Address2	City	Postcode	Country
1	1	123 Bolton Rd	NULL	Bolton	BL1 7AR	UK
2	2	456 Deane Rd	NULL	Bolton	BL3 5AB	UK
3	3	789 Derby St	NULL	Bolton	BL3 6HE	UK
4	4	101 Chorley ...	NULL	Bolton	BL1 3AS	UK
5	5	789 St. Hele...	NULL	Bolton	BL3 3PX	UK
6	6	456 Tonge ...	Apt 2B	Bolton	BL2 2HN	UK
7	7	789 Market St	Suite 10	Bolton	BL4 7PH	UK
8	8	101 Halliwell ...	NULL	Bolton	BL1 3QG	UK
9	9	123 Farnwort...	NULL	Bolton	BL4 7BB	UK
10	10	456 Blackbu...	Floor 3	Bolton	BL1 8HF	UK

```
Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

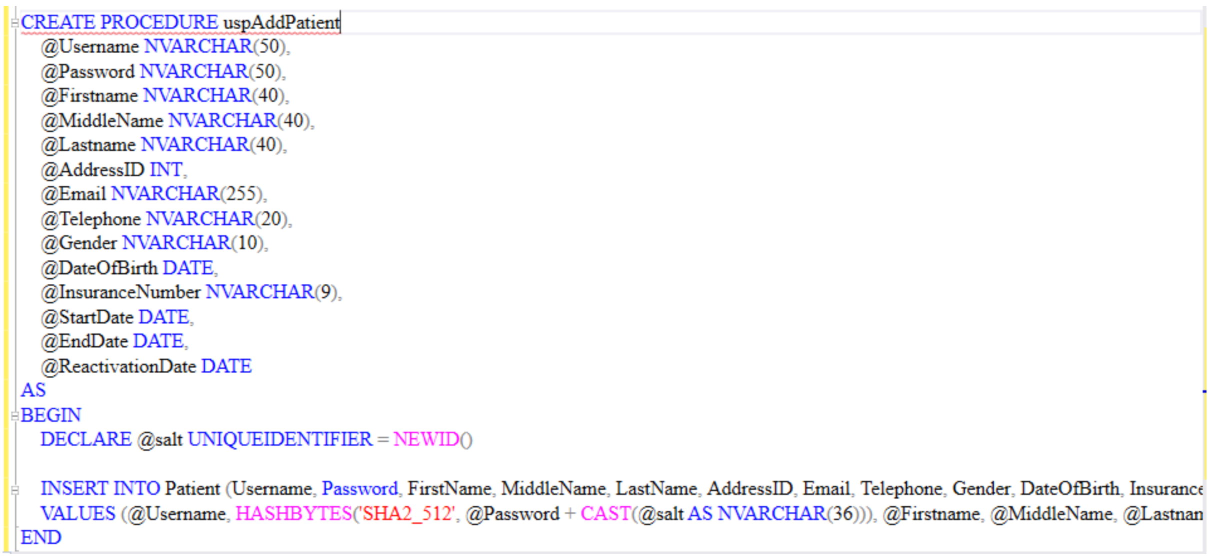
Figure 2.5

A data security function 'salt' was then used due to the sensitivity of the password in the Patient table. To initialize a variable @salt with a distinct identifier produced by the NEWID() function, use a DECLARE statement. By adding variability, this @salt value improves the password hashing process' security.

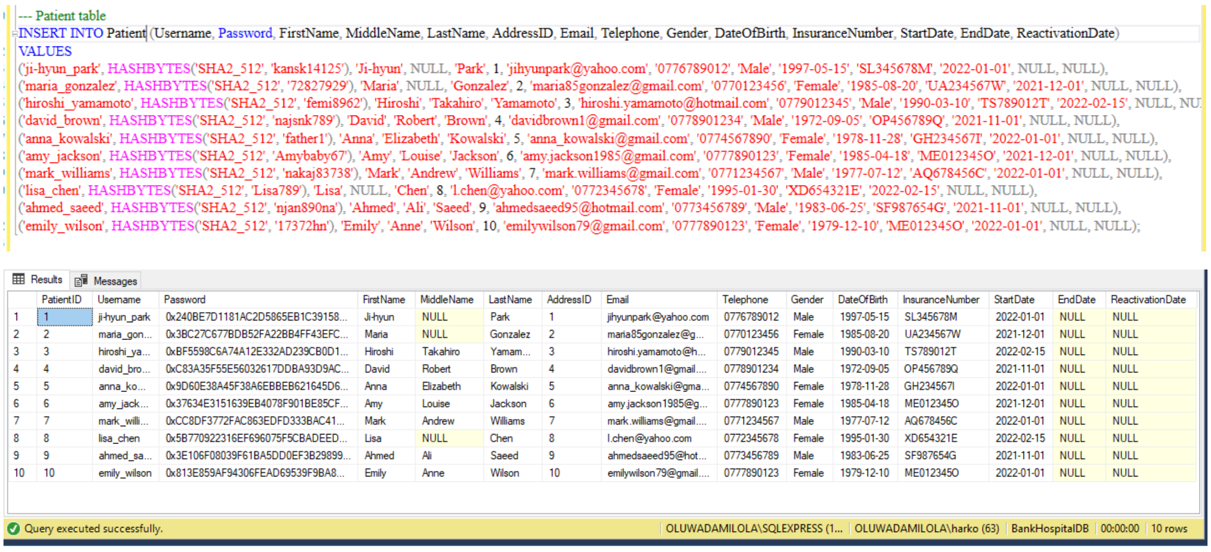
To add a new row with the specified parameters to the Patient table, use an INSERT INTO statement.

After the password has been hashed with the SHA-512 algorithm using the HASHBYTES function and the @salt value, it is cast to NVARCHAR(36) for concatenation.

The logic for handling password hashing, adding a new patient to the database, and using a salt value for extra security are all contained in this stored procedure.



```sql
CREATE PROCEDURE uspAddPatient
    @Username NVARCHAR(50),
    @Password NVARCHAR(50),
    @Firstname NVARCHAR(40),
    @MiddleName NVARCHAR(40),
    @Lastname NVARCHAR(40),
    @AddressID INT,
    @Email NVARCHAR(255),
    @Telephone NVARCHAR(20),
    @Gender NVARCHAR(10),
    @DateOfBirth DATE,
    @InsuranceNumber NVARCHAR(9),
    @StartDate DATE,
    @EndDate DATE,
    @ReactivationDate DATE
AS
BEGIN
    DECLARE @salt UNIQUEIDENTIFIER = NEWID()

    INSERT INTO Patient (Username, Password, FirstName, MiddleName, LastName, AddressID, Email, Telephone, Gender, DateOfBirth, Insurance
    VALUES (@Username, HASHBYTES('SHA2_512', @Password + CAST(@salt AS NVARCHAR(36))), @Firstname, @MiddleName, @Lastnan
END
```



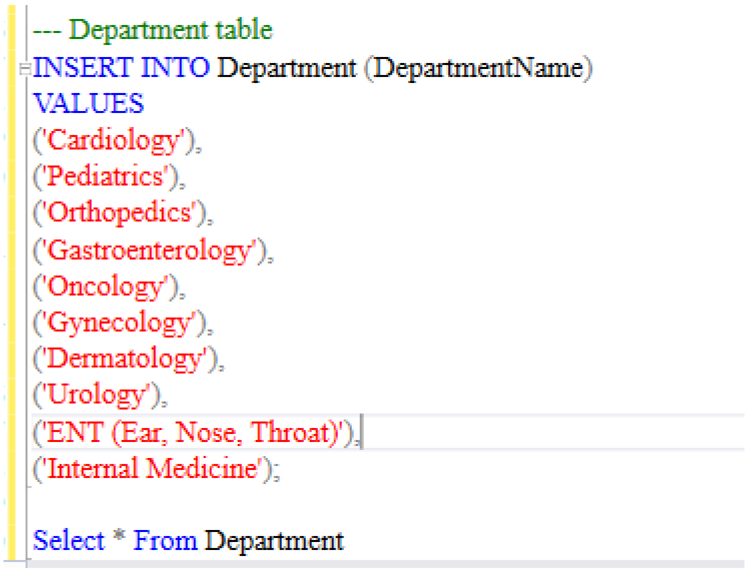
```sql
--- Patient table
INSERT INTO Patient (Username, Password, FirstName, MiddleName, LastName, AddressID, Email, Telephone, Gender, DateOfBirth, InsuranceNumber, StartDate, EndDate, ReactivationDate)
VALUES
('ji-hyun_park', HASHBYTES('SHA2_512', 'kansk14125'), 'Ji-hyun', NULL, 'Park', 1, 'jihyunpark@yahoo.com', '0776789012', 'Male', '1997-05-15', 'SL345678M', '2022-01-01', NULL, NULL),
('maria_gonzalez', HASHBYTES('SHA2_512', '72827929'), 'Maria', NULL, 'Gonzalez', 2, 'maria85gonzalez@gmail.com', '0770123456', 'Female', '1985-08-20', 'UA234567W', '2021-12-01', NULL, NULL),
('hiroshi_yamamoto', HASHBYTES('SHA2_512', 'femi8962'), 'Hiroshi', 'Takahiro', 'Yamamoto', 3, 'hiroshi.yamamoto@hotmail.com', '0779012345', 'Male', '1990-03-10', 'TS789012T', '2022-02-15', NULL, NU
('david_brown', HASHBYTES('SHA2_512', 'najsnk789'), 'David', 'Robert', 'Brown', 4, 'davidbrown1@gmail.com', '0778901234', 'Male', '1972-09-05', 'OP456789Q', '2021-11-01', NULL, NULL),
('anna_kowalski', HASHBYTES('SHA2_512', 'father1'), 'Anna', 'Elizabeth', 'Kowalski', 5, 'anna_kowalski@gmail.com', '0774567890', 'Female', '1978-11-28', 'GH234567I', '2022-01-01', NULL, NULL),
('amy_jackson', HASHBYTES('SHA2_512', 'Amybaby67'), 'Amy', 'Louise', 'Jackson', 6, 'amy.jackson1985@gmail.com', '0777890123', 'Female', '1985-04-18', 'ME012345O', '2021-12-01', NULL, NULL),
('mark_williams', HASHBYTES('SHA2_512', 'nakaj83738'), 'Mark', 'Andrew', 'Williams', 7, 'mark.williams@gmail.com', '0771234567', 'Male', '1977-07-12', 'AQ678456C', '2022-01-01', NULL, NULL),
('lisa_chen', HASHBYTES('SHA2_512', 'Lisa789'), 'Lisa', NULL, 'Chen', 8, 'l.chen@yahoo.com', '0772345678', 'Female', '1995-01-30', 'XD654321E', '2022-02-15', NULL, NULL),
('ahmed_saeed', HASHBYTES('SHA2_512', 'njan890na'), 'Ahmed', 'Ali', 'Saeed', 9, 'ahmedsaeed95@hotmail.com', '0773456789', 'Male', '1983-06-25', 'SF987654G', '2021-11-01', NULL, NULL),
('emily_wilson', HASHBYTES('SHA2_512', '17372hn'), 'Emily', 'Anne', 'Wilson', 10, 'emilywilson79@gmail.com', '0777890123', 'Female', '1979-12-10', 'ME012345O', '2022-01-01', NULL, NULL);
```

Results | Messages

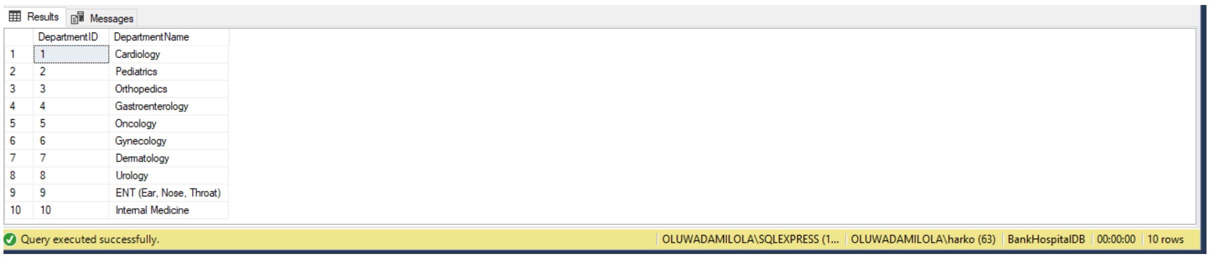
	PatientID	Username	Password	FirstName	MiddleName	LastName	AddressID	Email	Telephone	Gender	DateOfBirth	InsuranceNumber	StartDate	EndDate	ReactivationDate
1	1	ji-hyun_park	0x240BE7D1181AC2D5865EB1C39158...	Ji-hyun	NULL	Park	1	jihyunpark@yahoo.com	0776789012	Male	1997-05-15	SL345678M	2022-01-01	NULL	NULL
2	2	maria_gon...	0x3BC27C677BDB52FA22BB4FF43EFC...	Maria	NULL	Gonzalez	2	maria85gonzalez@g...	0770123456	Female	1985-08-20	UA234567W	2021-12-01	NULL	NULL
3	3	hiroshi_ya...	0xBF5598C6A74A12E332AD239CB0D1...	Hiroshi	Takahiro	Yamam...	3	hiroshi.yamamoto@h...	0779012345	Male	1990-03-10	TS789012T	2022-02-15	NULL	NULL
4	4	david_bro...	0xC83A35F55E56032617DDBA93D9AC...	David	Robert	Brown	4	davidbrown1@gmail....	0778901234	Male	1972-09-05	OP456789Q	2021-11-01	NULL	NULL
5	5	anna_ko...	0x9D60E38A45F38A6EBBEB621645D6...	Anna	Elizabeth	Kowalski	5	anna_kowalski@gma...	0774567890	Female	1978-11-28	GH234567I	2022-01-01	NULL	NULL
6	6	amy_jack...	0x37634E3151639EB4078F901BE85CF...	Amy	Louise	Jackson	6	amy.jackson1985@g...	0777890123	Female	1985-04-18	ME012345O	2021-12-01	NULL	NULL
7	7	mark_willi...	0xCC8DF3772FAC863EDFD333BAC41...	Mark	Andrew	Williams	7	mark.williams@gmail....	0771234567	Male	1977-07-12	AQ678456C	2022-01-01	NULL	NULL
8	8	lisa_chen	0x5B770922316EF696075F5CBADEED...	Lisa	NULL	Chen	8	l.chen@yahoo.com	0772345678	Female	1995-01-30	XD654321E	2022-02-15	NULL	NULL
9	9	ahmed_sa...	0x3E106F08039F61BA5DD0EF3B29899...	Ahmed	Ali	Saeed	9	ahmedsaeed95@hot...	0773456789	Male	1983-06-25	SF987654G	2021-11-01	NULL	NULL
10	10	emily_wilson	0x813E859AF94306FEAD69539F9BA8...	Emily	Anne	Wilson	10	emilywilson79@gmail....	0777890123	Female	1979-12-10	ME012345O	2022-01-01	NULL	NULL

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

Figure 2.6



```sql
--- Department table
INSERT INTO Department (DepartmentName)
VALUES
('Cardiology'),
('Pediatrics'),
('Orthopedics'),
('Gastroenterology'),
('Oncology'),
('Gynecology'),
('Dermatology'),
('Urology'),
('ENT (Ear, Nose, Throat)'),
('Internal Medicine');

Select * From Department
```

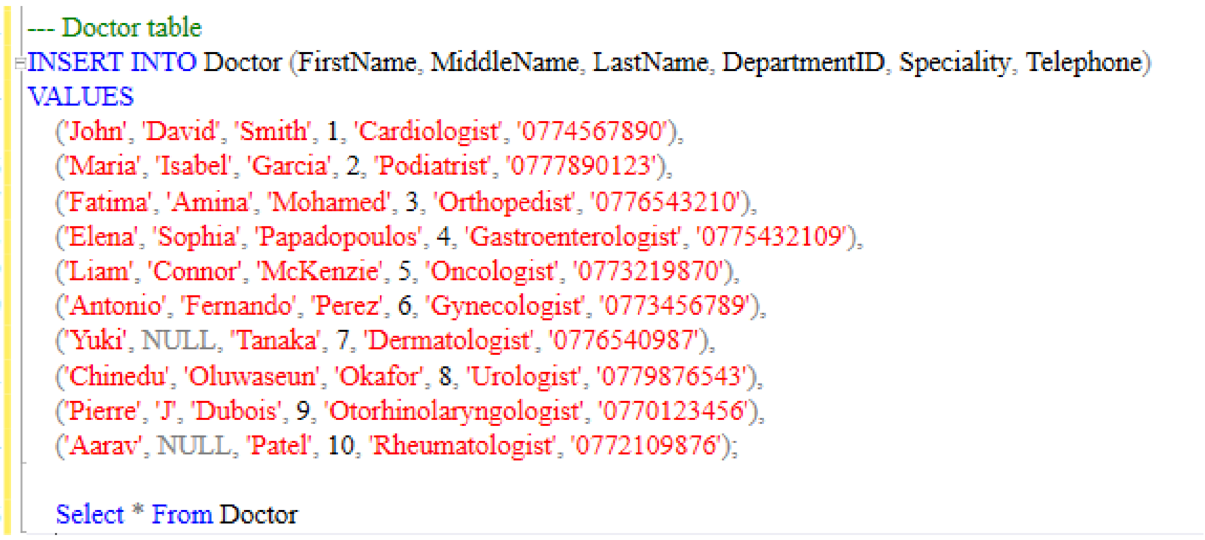


Results | Messages

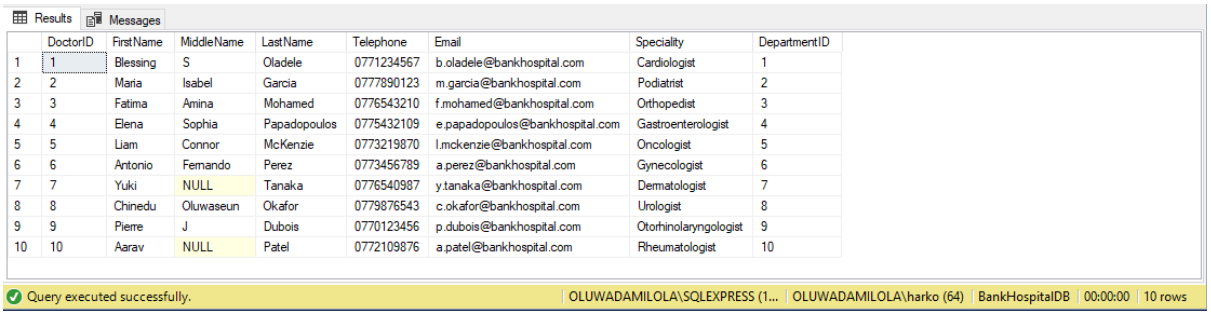
	DepartmentID	DepartmentName
1	1	Cardiology
2	2	Pediatrics
3	3	Orthopedics
4	4	Gastroenterology
5	5	Oncology
6	6	Gynecology
7	7	Dermatology
8	8	Urology
9	9	ENT (Ear, Nose, Throat)
10	10	Internal Medicine

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

Figure 2.7



```sql
--- Doctor table
INSERT INTO Doctor (FirstName, MiddleName, LastName, DepartmentID, Speciality, Telephone)
VALUES
    ('John', 'David', 'Smith', 1, 'Cardiologist', '0774567890'),
    ('Maria', 'Isabel', 'Garcia', 2, 'Podiatrist', '0777890123'),
    ('Fatima', 'Amina', 'Mohamed', 3, 'Orthopedist', '0776543210'),
    ('Elena', 'Sophia', 'Papadopoulos', 4, 'Gastroenterologist', '0775432109'),
    ('Liam', 'Connor', 'McKenzie', 5, 'Oncologist', '0773219870'),
    ('Antonio', 'Fernando', 'Perez', 6, 'Gynecologist', '0773456789'),
    ('Yuki', NULL, 'Tanaka', 7, 'Dermatologist', '0776540987'),
    ('Chinedu', 'Oluwaseun', 'Okafor', 8, 'Urologist', '0779876543'),
    ('Pierre', 'J', 'Dubois', 9, 'Otorhinolaryngologist', '0770123456'),
    ('Aarav', NULL, 'Patel', 10, 'Rheumatologist', '0772109876');

    Select * From Doctor
```

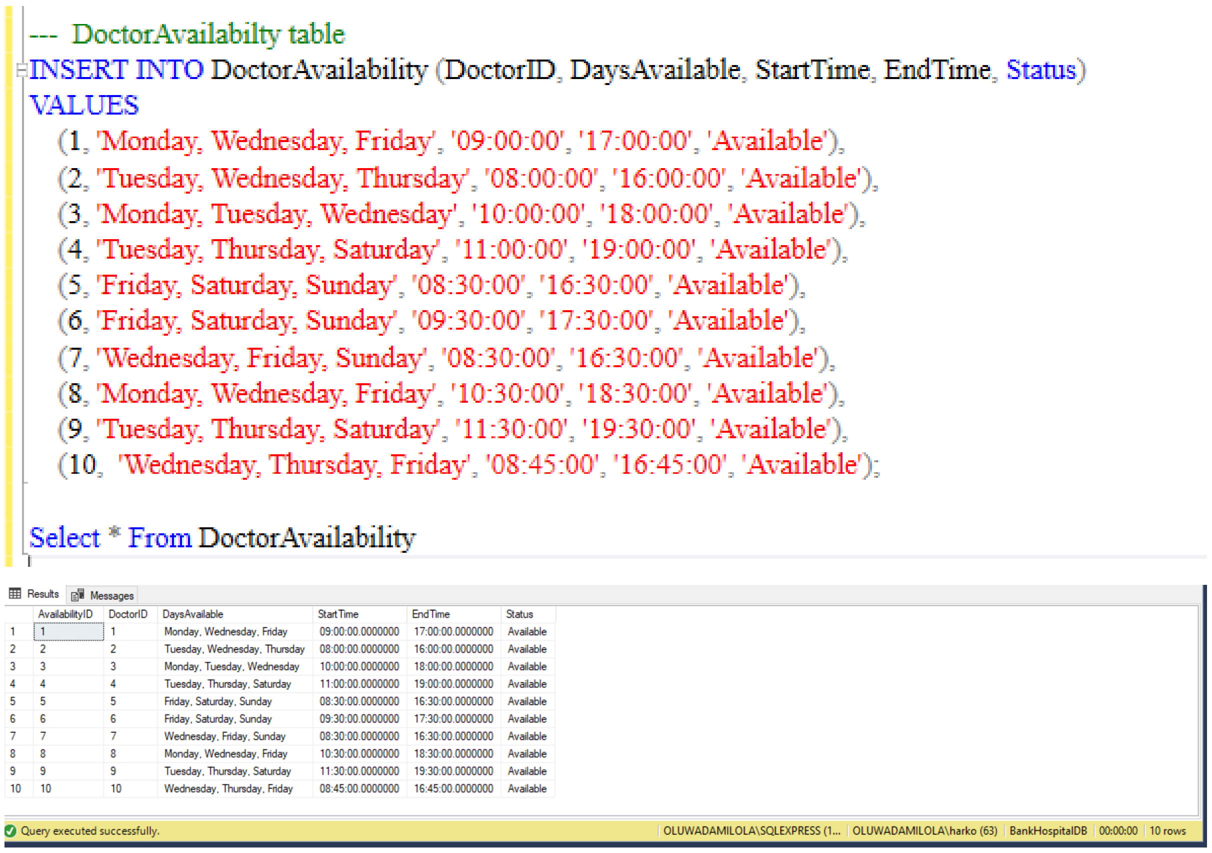


Results | Messages

	DoctorID	FirstName	MiddleName	LastName	Telephone	Email	Speciality	DepartmentID
1	1	Blessing	S	Oladele	0771234567	b.oladele@bankhospital.com	Cardiologist	1
2	2	Maria	Isabel	Garcia	0777890123	m.garcia@bankhospital.com	Podiatrist	2
3	3	Fatima	Amina	Mohamed	0776543210	f.mohamed@bankhospital.com	Orthopedist	3
4	4	Elena	Sophia	Papadopoulos	0775432109	e.papadopoulos@bankhospital.com	Gastroenterologist	4
5	5	Liam	Connor	McKenzie	0773219870	l.mckenzie@bankhospital.com	Oncologist	5
6	6	Antonio	Fernando	Perez	0773456789	a.perez@bankhospital.com	Gynecologist	6
7	7	Yuki	NULL	Tanaka	0776540987	y.tanaka@bankhospital.com	Dermatologist	7
8	8	Chinedu	Oluwaseun	Okafor	0779876543	c.okafor@bankhospital.com	Urologist	8
9	9	Pierre	J	Dubois	0770123456	p.dubois@bankhospital.com	Otorhinolaryngologist	9
10	10	Aarav	NULL	Patel	0772109876	a.patel@bankhospital.com	Rheumatologist	10

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (64) | BankHospitalDB | 00:00:00 | 10 rows
```

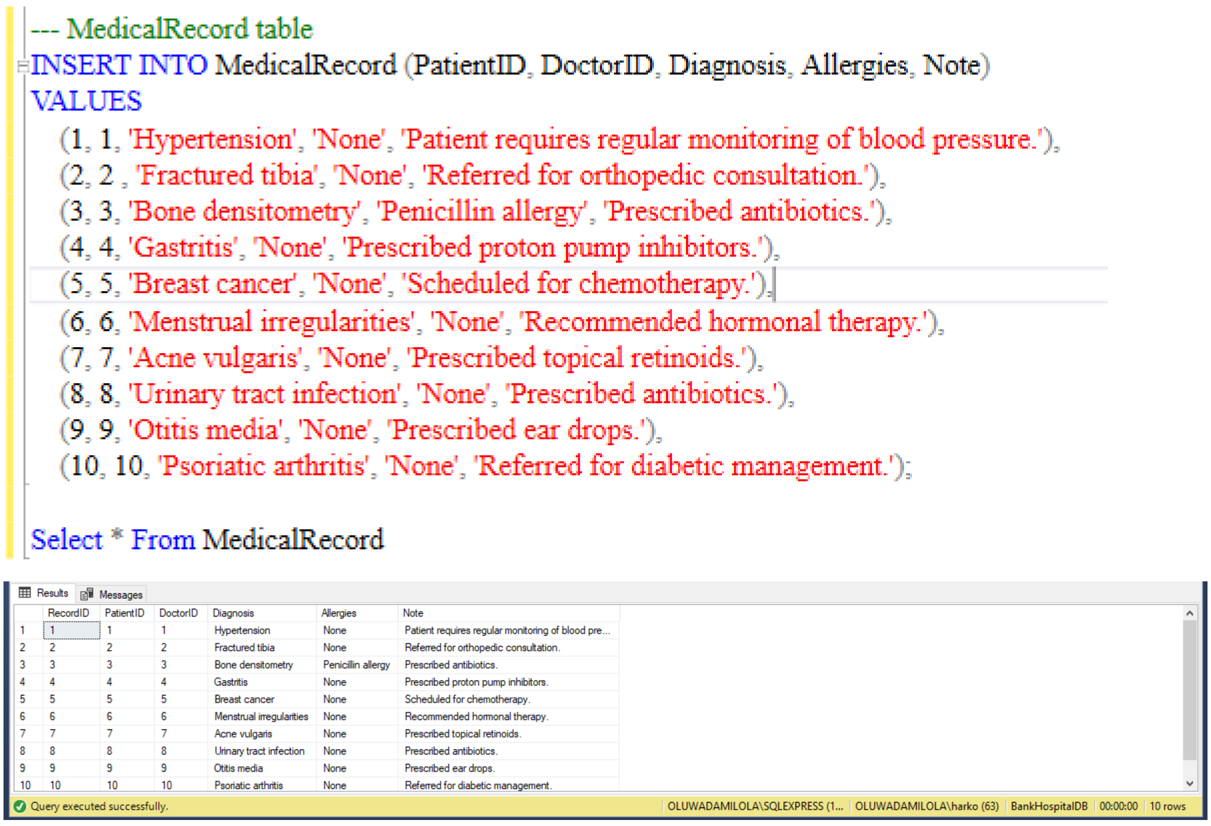
Figure 2.8



```sql
--- DoctorAvailabilty table
INSERT INTO DoctorAvailability (DoctorID, DaysAvailable, StartTime, EndTime, Status)
VALUES
    (1, 'Monday, Wednesday, Friday', '09:00:00', '17:00:00', 'Available'),
    (2, 'Tuesday, Wednesday, Thursday', '08:00:00', '16:00:00', 'Available'),
    (3, 'Monday, Tuesday, Wednesday', '10:00:00', '18:00:00', 'Available'),
    (4, 'Tuesday, Thursday, Saturday', '11:00:00', '19:00:00', 'Available'),
    (5, 'Friday, Saturday, Sunday', '08:30:00', '16:30:00', 'Available'),
    (6, 'Friday, Saturday, Sunday', '09:30:00', '17:30:00', 'Available'),
    (7, 'Wednesday, Friday, Sunday', '08:30:00', '16:30:00', 'Available'),
    (8, 'Monday, Wednesday, Friday', '10:30:00', '18:30:00', 'Available'),
    (9, 'Tuesday, Thursday, Saturday', '11:30:00', '19:30:00', 'Available'),
    (10,  'Wednesday, Thursday, Friday', '08:45:00', '16:45:00', 'Available');

Select * From DoctorAvailability
```

```
    AvailabilityID  DoctorID  DaysAvailable                 StartTime         EndTime           Status
1   1               1         Monday, Wednesday, Friday     09:00:00.0000000  17:00:00.0000000  Available
2   2               2         Tuesday, Wednesday, Thursday  08:00:00.0000000  16:00:00.0000000  Available
3   3               3         Monday, Tuesday, Wednesday    10:00:00.0000000  18:00:00.0000000  Available
4   4               4         Tuesday, Thursday, Saturday   11:00:00.0000000  19:00:00.0000000  Available
5   5               5         Friday, Saturday, Sunday      08:30:00.0000000  16:30:00.0000000  Available
6   6               6         Friday, Saturday, Sunday      09:30:00.0000000  17:30:00.0000000  Available
7   7               7         Wednesday, Friday, Sunday     08:30:00.0000000  16:30:00.0000000  Available
8   8               8         Monday, Wednesday, Friday     10:30:00.0000000  18:30:00.0000000  Available
9   9               9         Tuesday, Thursday, Saturday   11:30:00.0000000  19:30:00.0000000  Available
10  10              10        Wednesday, Thursday, Friday   08:45:00.0000000  16:45:00.0000000  Available

Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

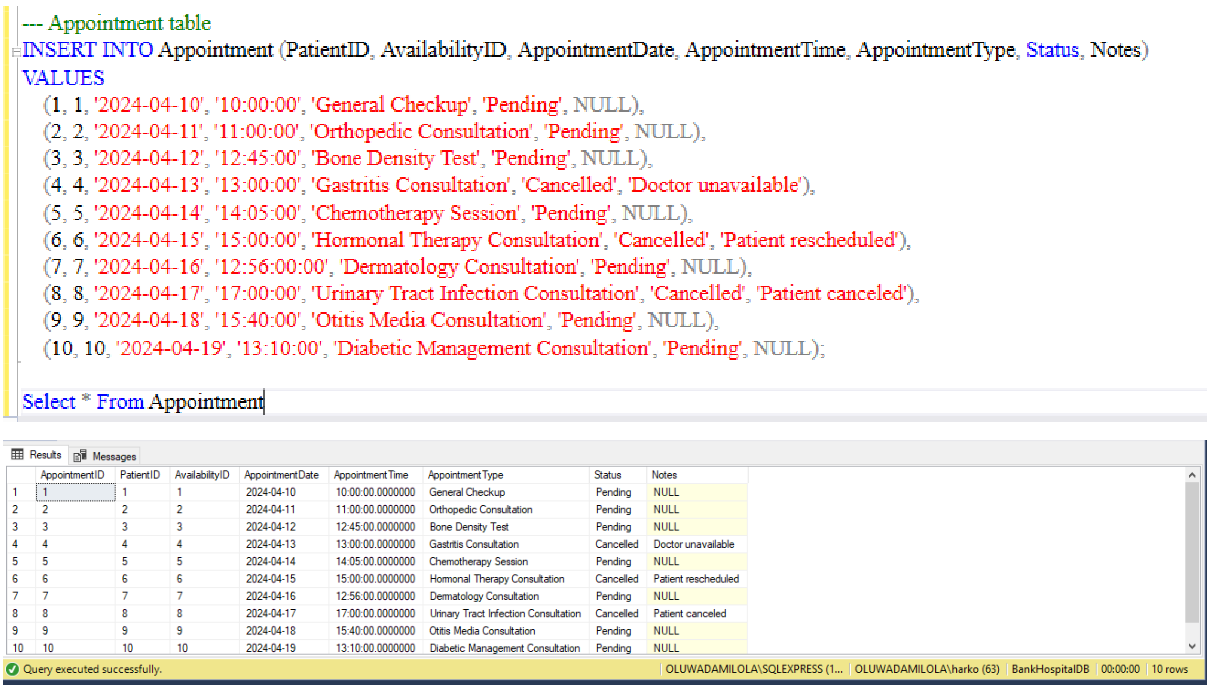
Figure 2.9



```sql
--- MedicalRecord table
INSERT INTO MedicalRecord (PatientID, DoctorID, Diagnosis, Allergies, Note)
VALUES
    (1, 1, 'Hypertension', 'None', 'Patient requires regular monitoring of blood pressure.'),
    (2, 2 , 'Fractured tibia', 'None', 'Referred for orthopedic consultation.'),
    (3, 3, 'Bone densitometry', 'Penicillin allergy', 'Prescribed antibiotics.'),
    (4, 4, 'Gastritis', 'None', 'Prescribed proton pump inhibitors.'),
    (5, 5, 'Breast cancer', 'None', 'Scheduled for chemotherapy.'),
    (6, 6, 'Menstrual irregularities', 'None', 'Recommended hormonal therapy.'),
    (7, 7, 'Acne vulgaris', 'None', 'Prescribed topical retinoids.'),
    (8, 8, 'Urinary tract infection', 'None', 'Prescribed antibiotics.'),
    (9, 9, 'Otitis media', 'None', 'Prescribed ear drops.'),
    (10, 10, 'Psoriatic arthritis', 'None', 'Referred for diabetic management.');

Select * From MedicalRecord
```

```
    RecordID  PatientID  DoctorID  Diagnosis                 Allergies           Note
1   1         1          1         Hypertension              None                Patient requires regular monitoring of blood pre...
2   2         2          2         Fractured tibia           None                Referred for orthopedic consultation.
3   3         3          3         Bone densitometry         Penicillin allergy  Prescribed antibiotics.
4   4         4          4         Gastritis                 None                Prescribed proton pump inhibitors.
5   5         5          5         Breast cancer             None                Scheduled for chemotherapy.
6   6         6          6         Menstrual irregularities  None                Recommended hormonal therapy.
7   7         7          7         Acne vulgaris             None                Prescribed topical retinoids.
8   8         8          8         Urinary tract infection   None                Prescribed antibiotics.
9   9         9          9         Otitis media              None                Prescribed ear drops.
10  10        10         10        Psoriatic arthritis       None                Referred for diabetic management.

Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

Figure 3.0



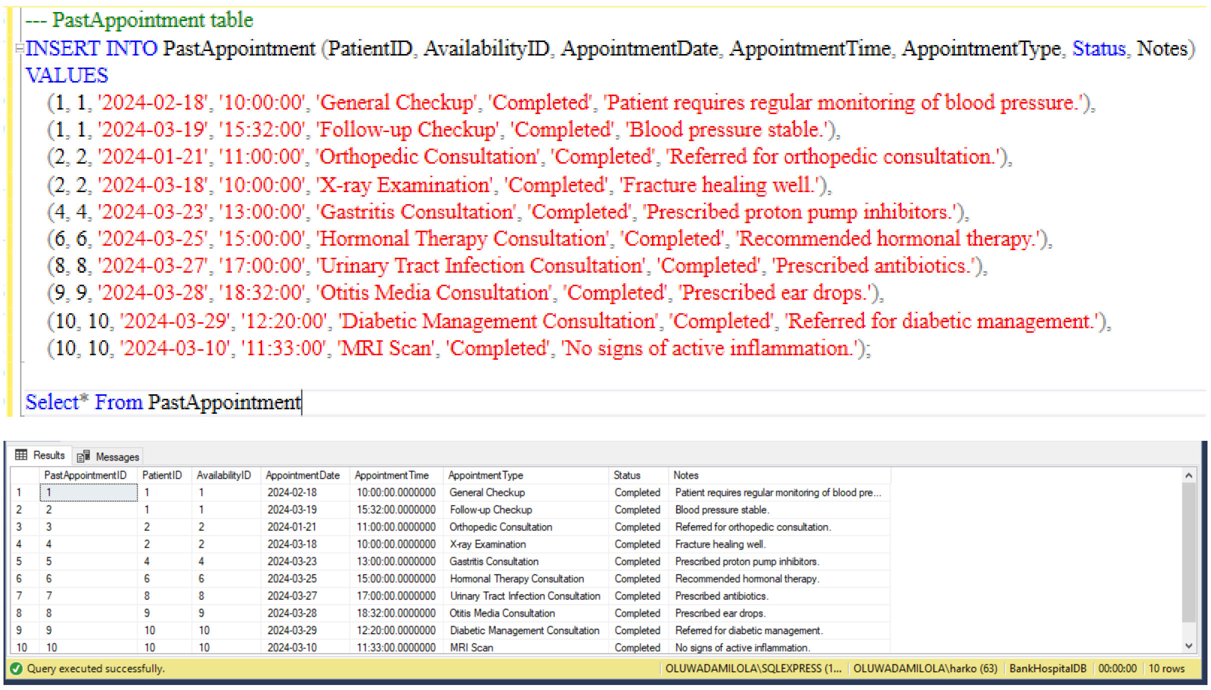
```sql
--- Appointment table
INSERT INTO Appointment (PatientID, AvailabilityID, AppointmentDate, AppointmentTime, AppointmentType, Status, Notes)
VALUES
   (1, 1, '2024-04-10', '10:00:00', 'General Checkup', 'Pending', NULL),
   (2, 2, '2024-04-11', '11:00:00', 'Orthopedic Consultation', 'Pending', NULL),
   (3, 3, '2024-04-12', '12:45:00', 'Bone Density Test', 'Pending', NULL),
   (4, 4, '2024-04-13', '13:00:00', 'Gastritis Consultation', 'Cancelled', 'Doctor unavailable'),
   (5, 5, '2024-04-14', '14:05:00', 'Chemotherapy Session', 'Pending', NULL),
   (6, 6, '2024-04-15', '15:00:00', 'Hormonal Therapy Consultation', 'Cancelled', 'Patient rescheduled'),
   (7, 7, '2024-04-16', '12:56:00:00', 'Dermatology Consultation', 'Pending', NULL),
   (8, 8, '2024-04-17', '17:00:00', 'Urinary Tract Infection Consultation', 'Cancelled', 'Patient canceled'),
   (9, 9, '2024-04-18', '15:40:00', 'Otitis Media Consultation', 'Pending', NULL),
   (10, 10, '2024-04-19', '13:10:00', 'Diabetic Management Consultation', 'Pending', NULL);

Select * From Appointment
```

Results | Messages

	AppointmentID	PatientID	AvailabilityID	AppointmentDate	AppointmentTime	AppointmentType	Status	Notes
1	1	1	1	2024-04-10	10:00:00.0000000	General Checkup	Pending	NULL
2	2	2	2	2024-04-11	11:00:00.0000000	Orthopedic Consultation	Pending	NULL
3	3	3	3	2024-04-12	12:45:00.0000000	Bone Density Test	Pending	NULL
4	4	4	4	2024-04-13	13:00:00.0000000	Gastritis Consultation	Cancelled	Doctor unavailable
5	5	5	5	2024-04-14	14:05:00.0000000	Chemotherapy Session	Pending	NULL
6	6	6	6	2024-04-15	15:00:00.0000000	Hormonal Therapy Consultation	Cancelled	Patient rescheduled
7	7	7	7	2024-04-16	12:56:00.0000000	Dermatology Consultation	Pending	NULL
8	8	8	8	2024-04-17	17:00:00.0000000	Urinary Tract Infection Consultation	Cancelled	Patient canceled
9	9	9	9	2024-04-18	15:40:00.0000000	Otitis Media Consultation	Pending	NULL
10	10	10	10	2024-04-19	13:10:00.0000000	Diabetic Management Consultation	Pending	NULL

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

Figure 3.1



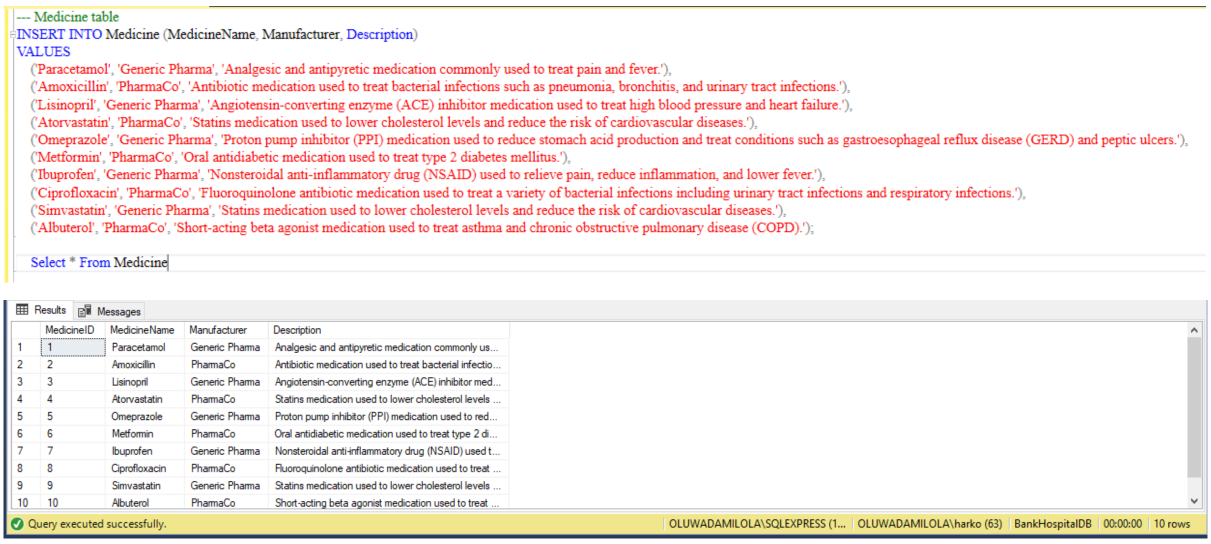
```sql
--- PastAppointment table
INSERT INTO PastAppointment (PatientID, AvailabilityID, AppointmentDate, AppointmentTime, AppointmentType, Status, Notes)
VALUES
   (1, 1, '2024-02-18', '10:00:00', 'General Checkup', 'Completed', 'Patient requires regular monitoring of blood pressure.'),
   (1, 1, '2024-03-19', '15:32:00', 'Follow-up Checkup', 'Completed', 'Blood pressure stable.'),
   (2, 2, '2024-01-21', '11:00:00', 'Orthopedic Consultation', 'Completed', 'Referred for orthopedic consultation.'),
   (2, 2, '2024-03-18', '10:00:00', 'X-ray Examination', 'Completed', 'Fracture healing well.'),
   (4, 4, '2024-03-23', '13:00:00', 'Gastritis Consultation', 'Completed', 'Prescribed proton pump inhibitors.'),
   (6, 6, '2024-03-25', '15:00:00', 'Hormonal Therapy Consultation', 'Completed', 'Recommended hormonal therapy.'),
   (8, 8, '2024-03-27', '17:00:00', 'Urinary Tract Infection Consultation', 'Completed', 'Prescribed antibiotics.'),
   (9, 9, '2024-03-28', '18:32:00', 'Otitis Media Consultation', 'Completed', 'Prescribed ear drops.'),
   (10, 10, '2024-03-29', '12:20:00', 'Diabetic Management Consultation', 'Completed', 'Referred for diabetic management.'),
   (10, 10, '2024-03-10', '11:33:00', 'MRI Scan', 'Completed', 'No signs of active inflammation.');

Select* From PastAppointment
```

Results | Messages

	PastAppointmentID	PatientID	AvailabilityID	AppointmentDate	AppointmentTime	AppointmentType	Status	Notes
1	1	1	1	2024-02-18	10:00:00.0000000	General Checkup	Completed	Patient requires regular monitoring of blood pre...
2	2	1	1	2024-03-19	15:32:00.0000000	Follow-up Checkup	Completed	Blood pressure stable.
3	3	2	2	2024-01-21	11:00:00.0000000	Orthopedic Consultation	Completed	Referred for orthopedic consultation.
4	4	2	2	2024-03-18	10:00:00.0000000	X-ray Examination	Completed	Fracture healing well.
5	5	4	4	2024-03-23	13:00:00.0000000	Gastritis Consultation	Completed	Prescribed proton pump inhibitors.
6	6	6	6	2024-03-25	15:00:00.0000000	Hormonal Therapy Consultation	Completed	Recommended hormonal therapy.
7	7	8	8	2024-03-27	17:00:00.0000000	Urinary Tract Infection Consultation	Completed	Prescribed antibiotics.
8	8	9	9	2024-03-28	18:32:00.0000000	Otitis Media Consultation	Completed	Prescribed ear drops.
9	9	10	10	2024-03-29	12:20:00.0000000	Diabetic Management Consultation	Completed	Referred for diabetic management.
10	10	10	10	2024-03-10	11:33:00.0000000	MRI Scan	Completed	No signs of active inflammation.

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 10 rows
```

Figure 3.2

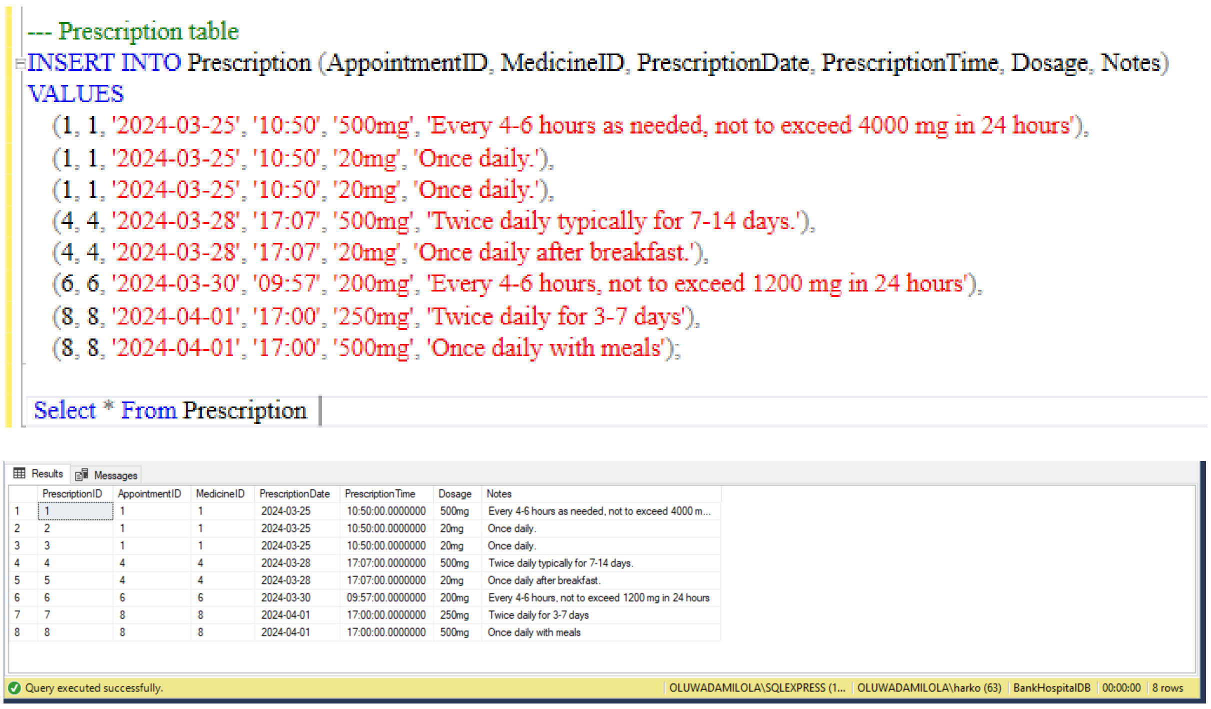


```sql
--- Medicine table
INSERT INTO Medicine (MedicineName, Manufacturer, Description)
VALUES
    ('Paracetamol', 'Generic Pharma', 'Analgesic and antipyretic medication commonly used to treat pain and fever.'),
    ('Amoxicillin', 'PharmaCo', 'Antibiotic medication used to treat bacterial infections such as pneumonia, bronchitis, and urinary tract infections.'),
    ('Lisinopril', 'Generic Pharma', 'Angiotensin-converting enzyme (ACE) inhibitor medication used to treat high blood pressure and heart failure.'),
    ('Atorvastatin', 'PharmaCo', 'Statins medication used to lower cholesterol levels and reduce the risk of cardiovascular diseases.'),
    ('Omeprazole', 'Generic Pharma', 'Proton pump inhibitor (PPI) medication used to reduce stomach acid production and treat conditions such as gastroesophageal reflux disease (GERD) and peptic ulcers.'),
    ('Metformin', 'PharmaCo', 'Oral antidiabetic medication used to treat type 2 diabetes mellitus.'),
    ('Ibuprofen', 'Generic Pharma', 'Nonsteroidal anti-inflammatory drug (NSAID) used to relieve pain, reduce inflammation, and lower fever.'),
    ('Ciprofloxacin', 'PharmaCo', 'Fluoroquinolone antibiotic medication used to treat a variety of bacterial infections including urinary tract infections and respiratory infections.'),
    ('Simvastatin', 'Generic Pharma', 'Statins medication used to lower cholesterol levels and reduce the risk of cardiovascular diseases.'),
    ('Albuterol', 'PharmaCo', 'Short-acting beta agonist medication used to treat asthma and chronic obstructive pulmonary disease (COPD).');

    Select * From Medicine
```

	MedicineID	MedicineName	Manufacturer	Description
1	1	Paracetamol	Generic Pharma	Analgesic and antipyretic medication commonly us...
2	2	Amoxicillin	PharmaCo	Antibiotic medication used to treat bacterial infectio...
3	3	Lisinopril	Generic Pharma	Angiotensin-converting enzyme (ACE) inhibitor med...
4	4	Atorvastatin	PharmaCo	Statins medication used to lower cholesterol levels ...
5	5	Omeprazole	Generic Pharma	Proton pump inhibitor (PPI) medication used to red...
6	6	Metformin	PharmaCo	Oral antidiabetic medication used to treat type 2 di...
7	7	Ibuprofen	Generic Pharma	Nonsteroidal anti-inflammatory drug (NSAID) used t...
8	8	Ciprofloxacin	PharmaCo	Fluoroquinolone antibiotic medication used to treat ...
9	9	Simvastatin	Generic Pharma	Statins medication used to lower cholesterol levels ...
10	10	Albuterol	PharmaCo	Short-acting beta agonist medication used to treat ...

```
Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1...   OLUWADAMILOLA\harko (63)   BankHospitalDB   00:00:00   10 rows
```

Figure 3.3

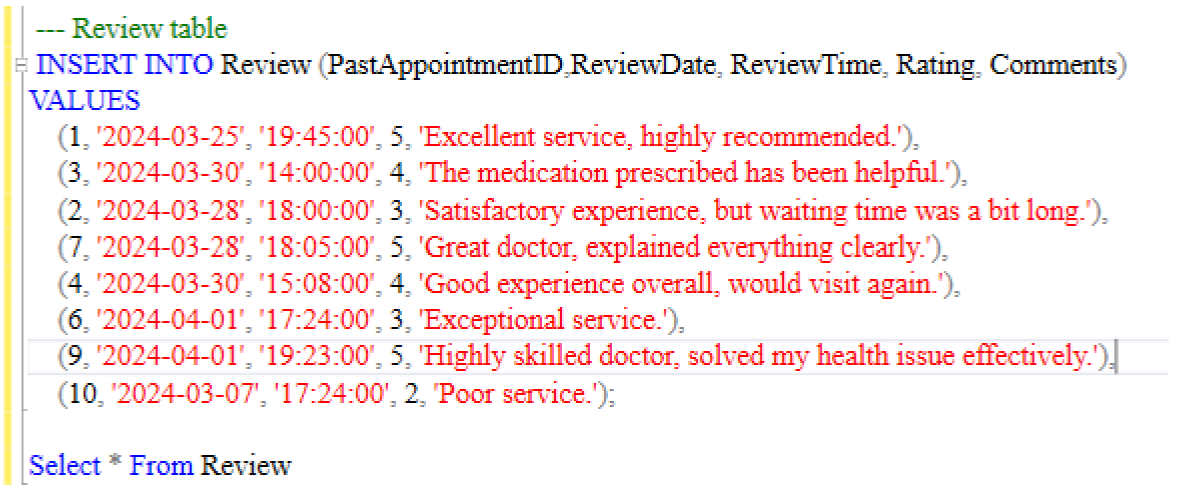


```sql
--- Prescription table
INSERT INTO Prescription (AppointmentID, MedicineID, PrescriptionDate, PrescriptionTime, Dosage, Notes)
VALUES
    (1, 1, '2024-03-25', '10:50', '500mg', 'Every 4-6 hours as needed, not to exceed 4000 mg in 24 hours'),
    (1, 1, '2024-03-25', '10:50', '20mg', 'Once daily.'),
    (1, 1, '2024-03-25', '10:50', '20mg', 'Once daily.'),
    (4, 4, '2024-03-28', '17:07', '500mg', 'Twice daily typically for 7-14 days.'),
    (4, 4, '2024-03-28', '17:07', '20mg', 'Once daily after breakfast.'),
    (6, 6, '2024-03-30', '09:57', '200mg', 'Every 4-6 hours, not to exceed 1200 mg in 24 hours'),
    (8, 8, '2024-04-01', '17:00', '250mg', 'Twice daily for 3-7 days'),
    (8, 8, '2024-04-01', '17:00', '500mg', 'Once daily with meals');

 Select * From Prescription
```

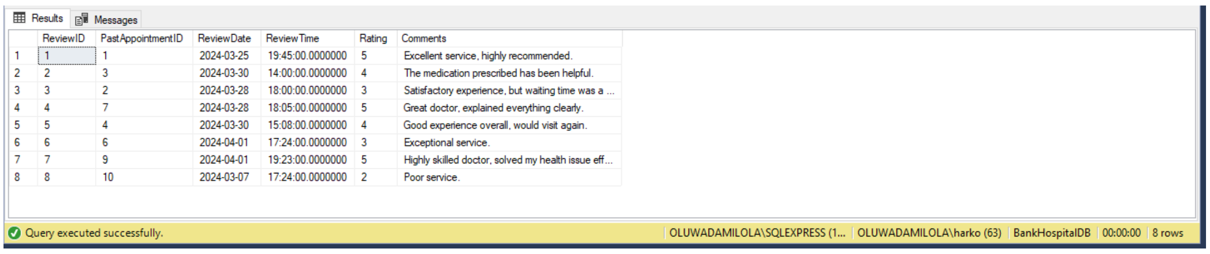
	PrescriptionID	AppointmentID	MedicineID	PrescriptionDate	PrescriptionTime	Dosage	Notes
1	1	1	1	2024-03-25	10:50:00.0000000	500mg	Every 4-6 hours as needed, not to exceed 4000 m...
2	2	1	1	2024-03-25	10:50:00.0000000	20mg	Once daily.
3	3	1	1	2024-03-25	10:50:00.0000000	20mg	Once daily.
4	4	4	4	2024-03-28	17:07:00.0000000	500mg	Twice daily typically for 7-14 days.
5	5	4	4	2024-03-28	17:07:00.0000000	20mg	Once daily after breakfast.
6	6	6	6	2024-03-30	09:57:00.0000000	200mg	Every 4-6 hours, not to exceed 1200 mg in 24 hours
7	7	8	8	2024-04-01	17:00:00.0000000	250mg	Twice daily for 3-7 days
8	8	8	8	2024-04-01	17:00:00.0000000	500mg	Once daily with meals

```
Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1...   OLUWADAMILOLA\harko (63)   BankHospitalDB   00:00:00   8 rows
```

Figure 3.4



```
--- Review table
INSERT INTO Review (PastAppointmentID,ReviewDate, ReviewTime, Rating, Comments)
VALUES
    (1, '2024-03-25', '19:45:00', 5, 'Excellent service, highly recommended.'),
    (3, '2024-03-30', '14:00:00', 4, 'The medication prescribed has been helpful.'),
    (2, '2024-03-28', '18:00:00', 3, 'Satisfactory experience, but waiting time was a bit long.'),
    (7, '2024-03-28', '18:05:00', 5, 'Great doctor, explained everything clearly.'),
    (4, '2024-03-30', '15:08:00', 4, 'Good experience overall, would visit again.'),
    (6, '2024-04-01', '17:24:00', 3, 'Exceptional service.'),
    (9, '2024-04-01', '19:23:00', 5, 'Highly skilled doctor, solved my health issue effectively.'),
    (10, '2024-03-07', '17:24:00', 2, 'Poor service.');


Select * From Review
```



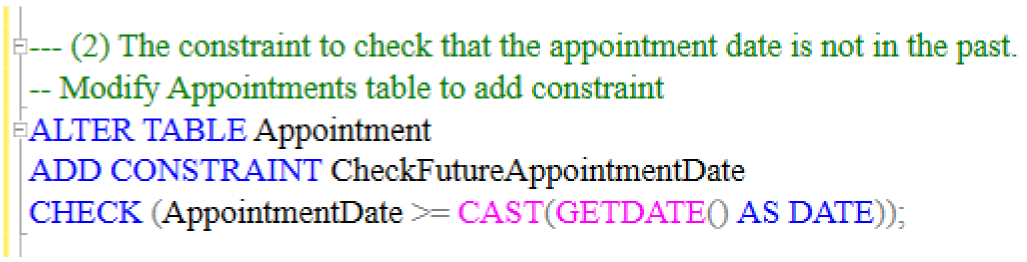
Results | Messages

	ReviewID	PastAppointmentID	ReviewDate	ReviewTime	Rating	Comments
1	1	1	2024-03-25	19:45:00.0000000	5	Excellent service, highly recommended.
2	2	3	2024-03-30	14:00:00.0000000	4	The medication prescribed has been helpful.
3	3	2	2024-03-28	18:00:00.0000000	3	Satisfactory experience, but waiting time was a ...
4	4	7	2024-03-28	18:05:00.0000000	5	Great doctor, explained everything clearly.
5	5	4	2024-03-30	15:08:00.0000000	4	Good experience overall, would visit again.
6	6	6	2024-04-01	17:24:00.0000000	3	Exceptional service.
7	7	9	2024-04-01	19:23:00.0000000	5	Highly skilled doctor, solved my health issue eff...
8	8	10	2024-03-07	17:24:00.0000000	2	Poor service.

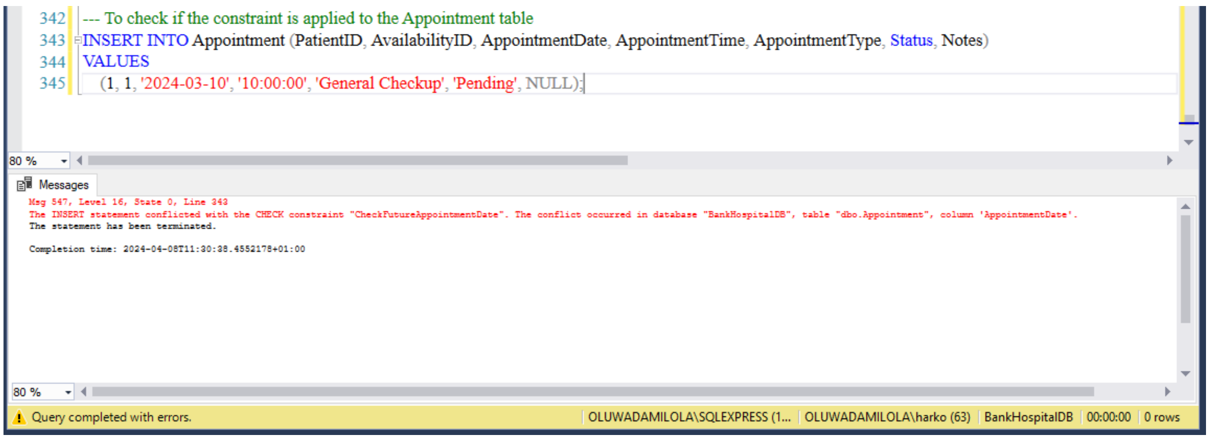
Query executed successfully. | OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 8 rows

Figure 3.5

QUESTION 2



```
--- (2) The constraint to check that the appointment date is not in the past.
-- Modify Appointments table to add constraint
ALTER TABLE Appointment
ADD CONSTRAINT CheckFutureAppointmentDate
CHECK (AppointmentDate >= CAST(GETDATE() AS DATE));
```



```
--- To check if the constraint is applied to the Appointment table
INSERT INTO Appointment (PatientID, AvailabilityID, AppointmentDate, AppointmentTime, AppointmentType, Status, Notes)
VALUES
    (1, 1, '2024-03-10', '10:00:00', 'General Checkup', 'Pending', NULL);
```

Messages

```
Msg 547, Level 16, State 0, Line 343
The INSERT statement conflicted with the CHECK constraint "CheckFutureAppointmentDate". The conflict occurred in database "BankHospitalDB", table "dbo.Appointment", column 'AppointmentDate'.
The statement has been terminated.

Completion time: 2024-04-08T11:30:38.4552178+01:00
```

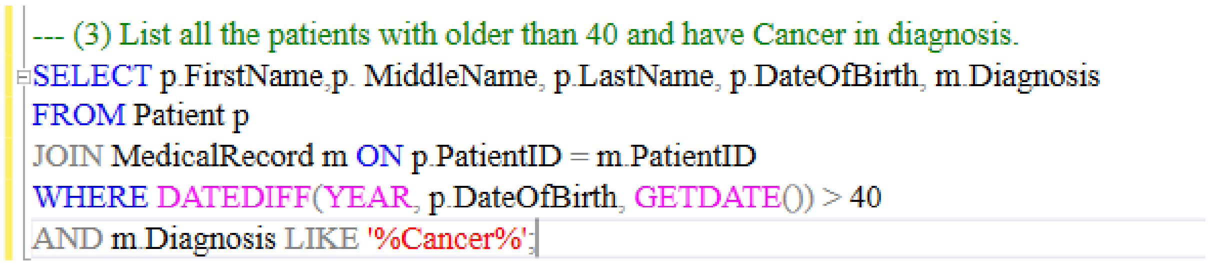
Query completed with errors. | OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 0 rows

Figure 3.6

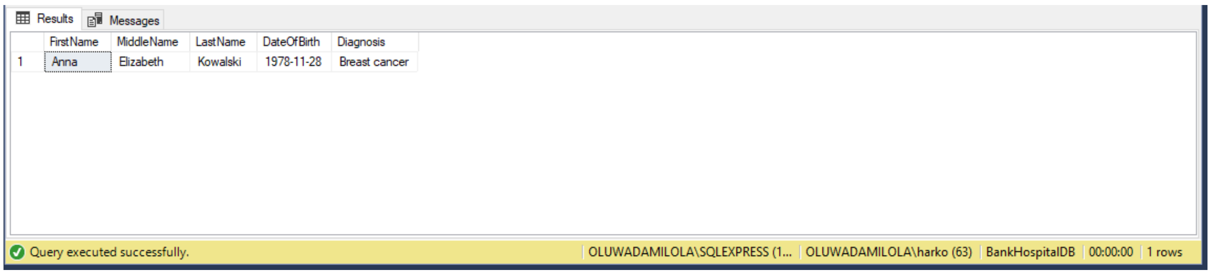
The use of the CAST(GETDATE() AS DATE) function to truncate the time portion, taking into consideration only the date part, and the GETDATE() function to retrieve the current date and time. By ensuring that the constraint verifies whether the

appointment date is from the present or the future, the use of >= effectively stops appointments from being set for earlier or past dates.
A past date was later inputted to confirm the successful execution of the constraints and the result showed that the check constraints are preventing inputting a past date.

QUESTION 3



```
--- (3) List all the patients with older than 40 and have Cancer in diagnosis.
SELECT p.FirstName,p. MiddleName, p.LastName, p.DateOfBirth, m.Diagnosis
FROM Patient p
JOIN MedicalRecord m ON p.PatientID = m.PatientID
WHERE DATEDIFF(YEAR, p.DateOfBirth, GETDATE()) > 40
AND m.Diagnosis LIKE '%Cancer%';
```



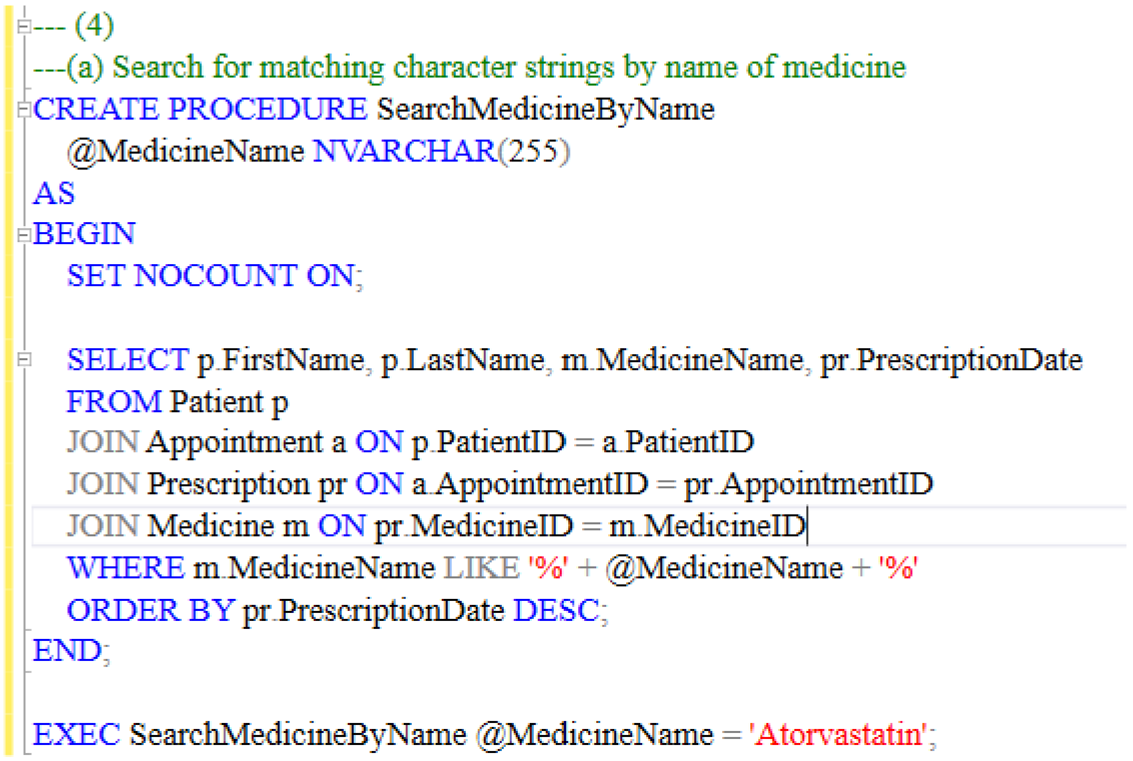
	FirstName	MiddleName	LastName	DateOfBirth	Diagnosis
1	Anna	Elizabeth	Kowalski	1978-11-28	Breast cancer

```
Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1...   OLUWADAMILOLA\harko (63)   BankHospitalDB   00:00:00   1 rows
```

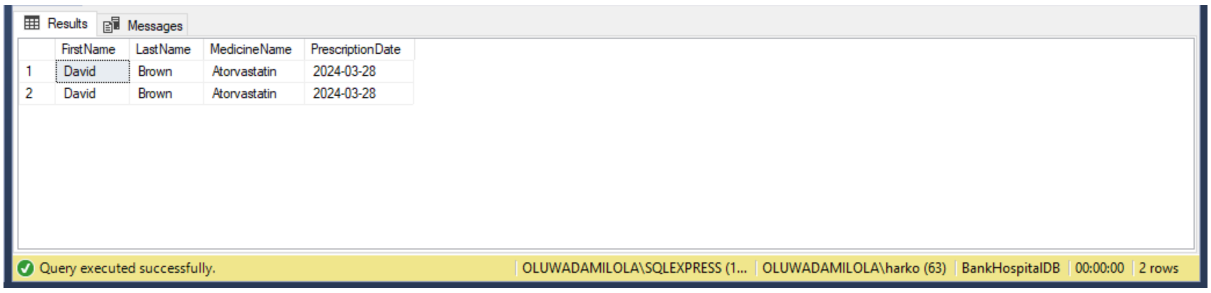
Figure 3.7
The first name, middle name, last name, date of birth, and diagnosis of patients over 40 whose medical history includes the word "cancer" are successfully retrieved by this query. It makes sure that the only patient records that are returned fit both of these requirements. This is accomplished by using the where function to join the patient table with the medical record table, using the patient ID, and the built-in date diff and getdate functions. The result displays a patient who has successfully met the conditions.

QUESTION 4

(4a)



```sql
--- (4)
---(a) Search for matching character strings by name of medicine
CREATE PROCEDURE SearchMedicineByName
    @MedicineName NVARCHAR(255)
AS
BEGIN
    SET NOCOUNT ON;

    SELECT p.FirstName, p.LastName, m.MedicineName, pr.PrescriptionDate
    FROM Patient p
    JOIN Appointment a ON p.PatientID = a.PatientID
    JOIN Prescription pr ON a.AppointmentID = pr.AppointmentID
    JOIN Medicine m ON pr.MedicineID = m.MedicineID
    WHERE m.MedicineName LIKE '%' + @MedicineName + '%'
    ORDER BY pr.PrescriptionDate DESC;
END;

EXEC SearchMedicineByName @MedicineName = 'Atorvastatin';
```

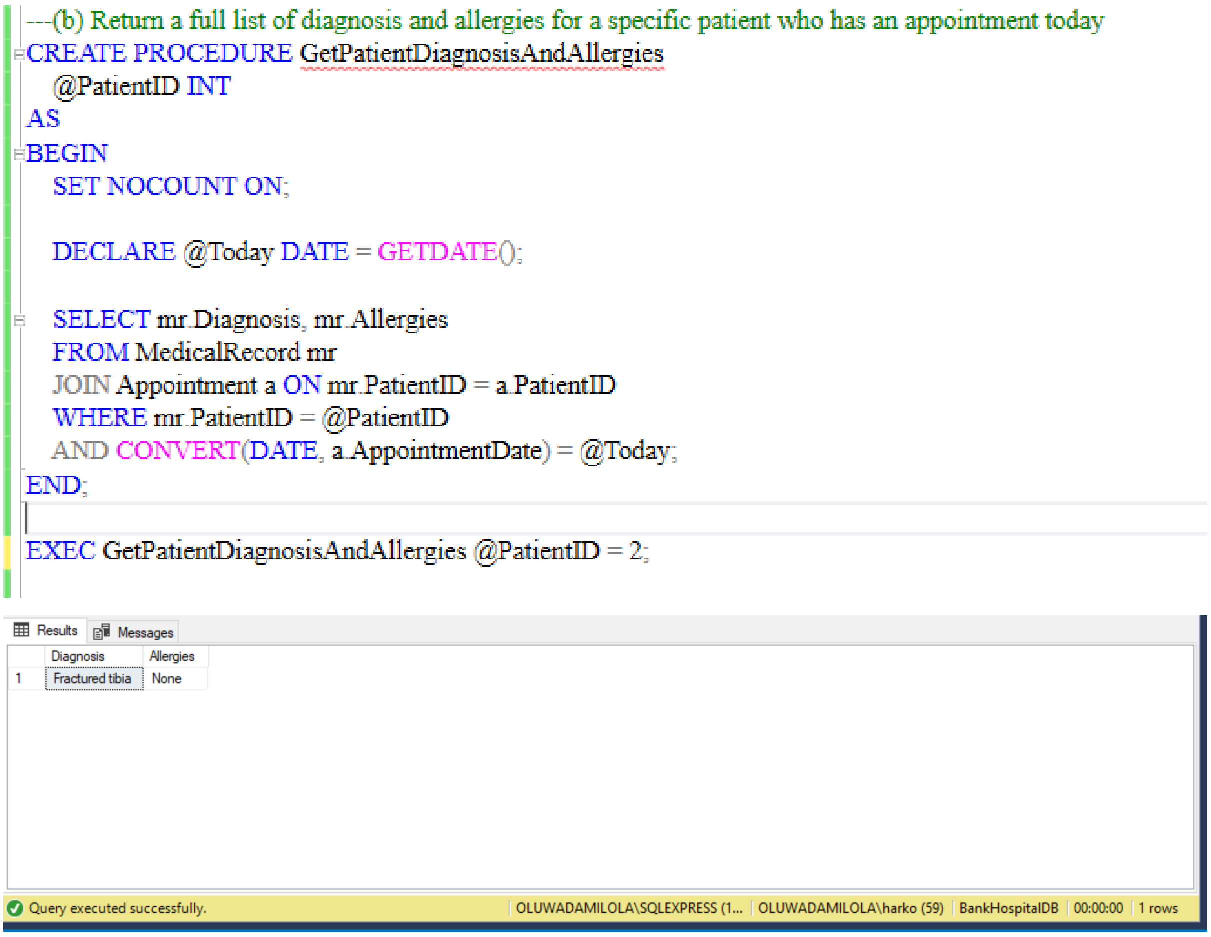


```
Results   Messages
   FirstName  LastName  MedicineName  PrescriptionDate
1  David      Brown     Atorvastatin  2024-03-28
2  David      Brown     Atorvastatin  2024-03-28

Query executed successfully.  | OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (63) | BankHospitalDB | 00:00:00 | 2 rows
```

Figure 3.8

This stored procedure retrieves information about patients who have been prescribed a medicine containing the specified medicine name parameter, ordered by the prescription date. Executing the stored procedure with the medicine name parameter set to 'Atorvastatin'. This will return data for patients who have been prescribed the medicine 'Atorvastatin', ordered by prescription date in descending order.

(4b)



```sql
---(b) Return a full list of diagnosis and allergies for a specific patient who has an appointment today
CREATE PROCEDURE GetPatientDiagnosisAndAllergies
    @PatientID INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Today DATE = GETDATE();

    SELECT mr.Diagnosis, mr.Allergies
    FROM MedicalRecord mr
    JOIN Appointment a ON mr.PatientID = a.PatientID
    WHERE mr.PatientID = @PatientID
    AND CONVERT(DATE, a.AppointmentDate) = @Today;
END;

EXEC GetPatientDiagnosisAndAllergies @PatientID = 2;
```

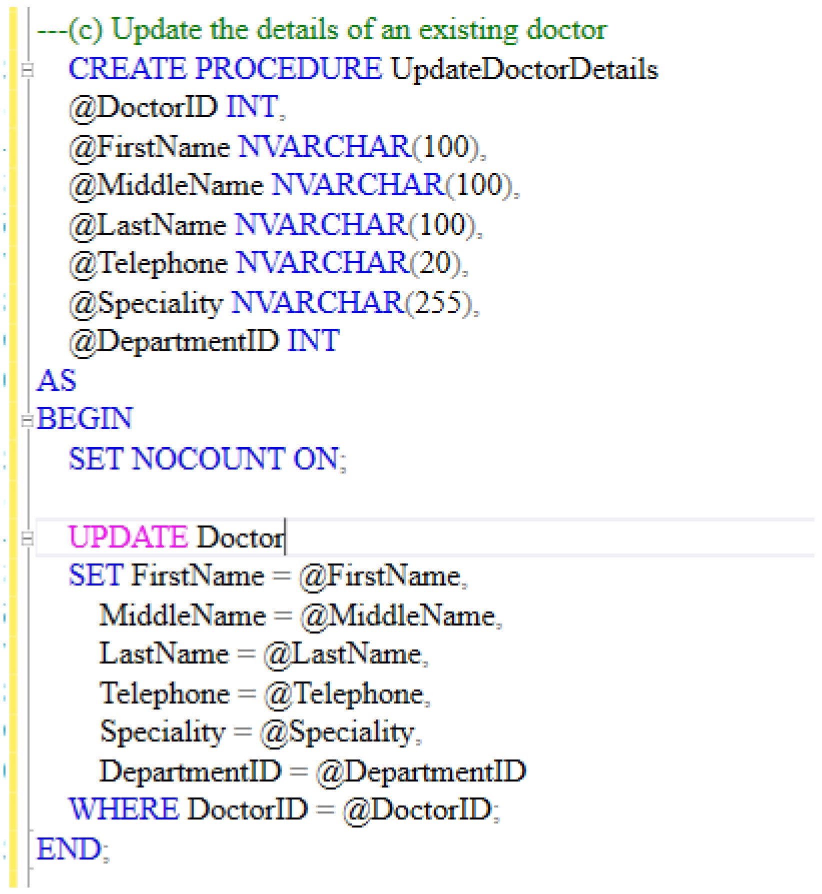
	Diagnosis	Allergies
1	Fractured tibia	None

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (59) | BankHospitalDB | 00:00:00 | 1 rows
```

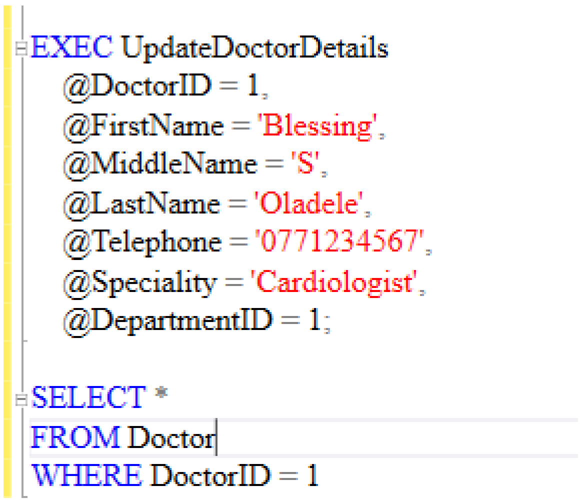
Figure 3.9

To retrieve the specific patient details, the @PatientID is entered into the stored procedure called "GetPatientDiagnosisAndAllergies." To stop the count of rows being impacted by the statement @TodayDATE, which specifies the current day, use the SET NOCOUNT ON command. To satisfy the particular requirements, the SELECT, JOIN, and WHERE functions are utilized. The complete list of diagnoses and allergies for a particular patient with an appointment on this particular day is successfully retrieved by this stored procedure. For executing the stored procedure, the Patient ID has to be inputted since it is for a particular patient recorded.

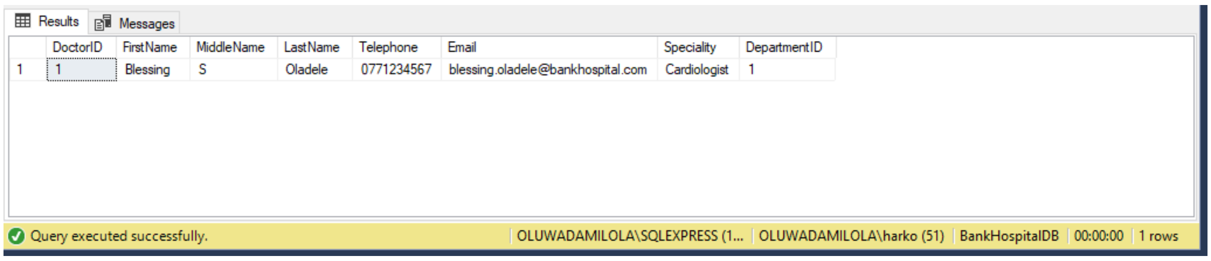
(4c)



```sql
---(c) Update the details of an existing doctor
    CREATE PROCEDURE UpdateDoctorDetails
    @DoctorID INT,
    @FirstName NVARCHAR(100),
    @MiddleName NVARCHAR(100),
    @LastName NVARCHAR(100),
    @Telephone NVARCHAR(20),
    @Speciality NVARCHAR(255),
    @DepartmentID INT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Doctor
    SET FirstName = @FirstName,
        MiddleName = @MiddleName,
        LastName = @LastName,
        Telephone = @Telephone,
        Speciality = @Speciality,
        DepartmentID = @DepartmentID
    WHERE DoctorID = @DoctorID;
END;
```



```sql
EXEC UpdateDoctorDetails
    @DoctorID = 1,
    @FirstName = 'Blessing',
    @MiddleName = 'S',
    @LastName = 'Oladele',
    @Telephone = '0771234567',
    @Speciality = 'Cardiologist',
    @DepartmentID = 1;

SELECT *
FROM Doctor
WHERE DoctorID = 1
```



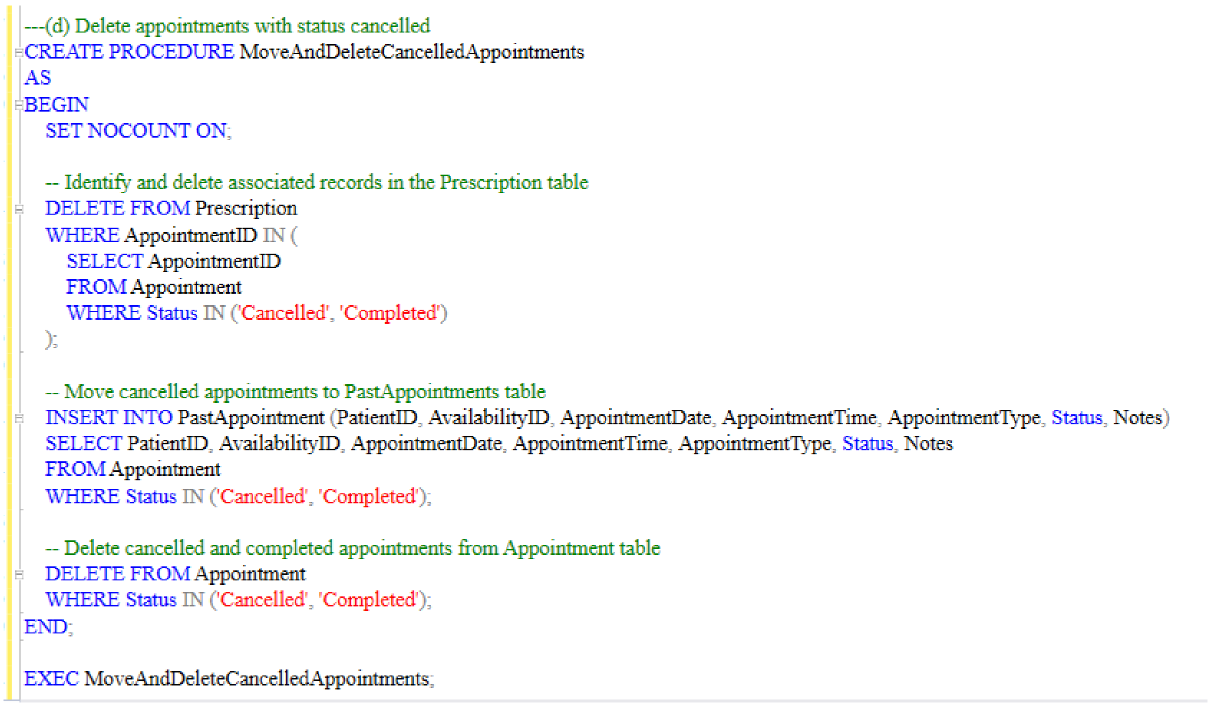
	DoctorID	FirstName	MiddleName	LastName	Telephone	Email	Speciality	DepartmentID
1	1	Blessing	S	Oladele	0771234567	blessing.oladele@bankhospital.com	Cardiologist	1

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (51) | BankHospitalDB | 00:00:00 | 1 rows
```

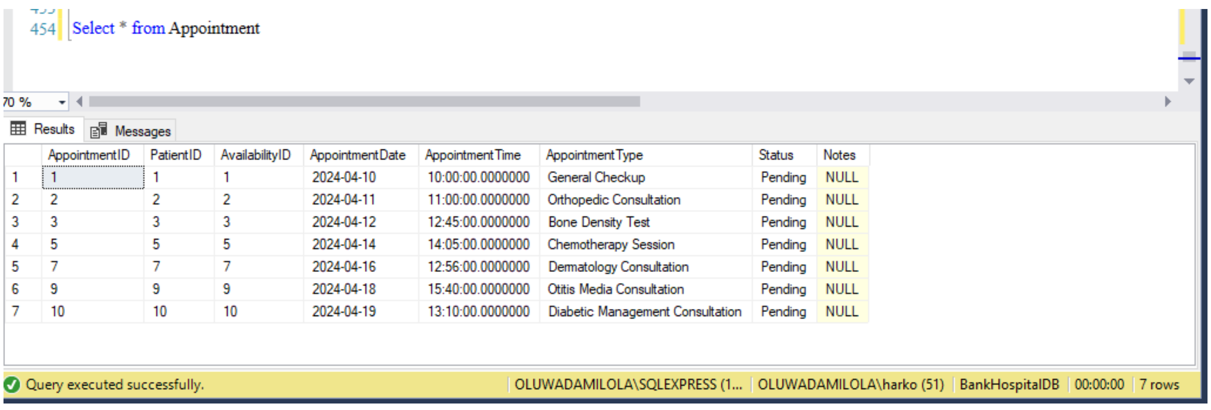
Figure 4.0

This store procedure Executes the stored procedure with specific parameter values to update the details of the doctor with ID 1. The parameter values include new values for the doctor's first name, middle name, last name, telephone number, speciality, and department ID. Overall, this SQL code effectively creates a stored procedure for updating doctor details and then executes it to update the details of a specific doctor. It also verifies the update by selecting and displaying the updated details of the doctor from the "Doctor" table.

(4d)



```sql
---(d) Delete appointments with status cancelled
CREATE PROCEDURE MoveAndDeleteCancelledAppointments
AS
BEGIN
    SET NOCOUNT ON;

    -- Identify and delete associated records in the Prescription table
    DELETE FROM Prescription
    WHERE AppointmentID IN (
        SELECT AppointmentID
        FROM Appointment
        WHERE Status IN ('Cancelled', 'Completed')
    );

    -- Move cancelled appointments to PastAppointments table
    INSERT INTO PastAppointment (PatientID, AvailabilityID, AppointmentDate, AppointmentTime, AppointmentType, Status, Notes)
    SELECT PatientID, AvailabilityID, AppointmentDate, AppointmentTime, AppointmentType, Status, Notes
    FROM Appointment
    WHERE Status IN ('Cancelled', 'Completed');

    -- Delete cancelled and completed appointments from Appointment table
    DELETE FROM Appointment
    WHERE Status IN ('Cancelled', 'Completed');
END;

EXEC MoveAndDeleteCancelledAppointments;
```



```sql
Select * from Appointment
```

	AppointmentID	PatientID	AvailabilityID	AppointmentDate	AppointmentTime	AppointmentType	Status	Notes
1	1	1	1	2024-04-10	10:00:00.0000000	General Checkup	Pending	NULL
2	2	2	2	2024-04-11	11:00:00.0000000	Orthopedic Consultation	Pending	NULL
3	3	3	3	2024-04-12	12:45:00.0000000	Bone Density Test	Pending	NULL
4	5	5	5	2024-04-14	14:05:00.0000000	Chemotherapy Session	Pending	NULL
5	7	7	7	2024-04-16	12:56:00.0000000	Dermatology Consultation	Pending	NULL
6	9	9	9	2024-04-18	15:40:00.0000000	Otitis Media Consultation	Pending	NULL
7	10	10	10	2024-04-19	13:10:00.0000000	Diabetic Management Consultation	Pending	NULL

```
Query executed successfully. | OLUWADAMILOLA\SQLEXPRESS (1... | OLUWADAMILOLA\harko (51) | BankHospitalDB | 00:00:00 | 7 rows
```

Figure 4.1

Since it isn’t ideal to delete data from a database, hence completed and cancelled appointments are moved from the appointments table to the past appointment table before being deleted from the appointment table so we can always reference it when needed.

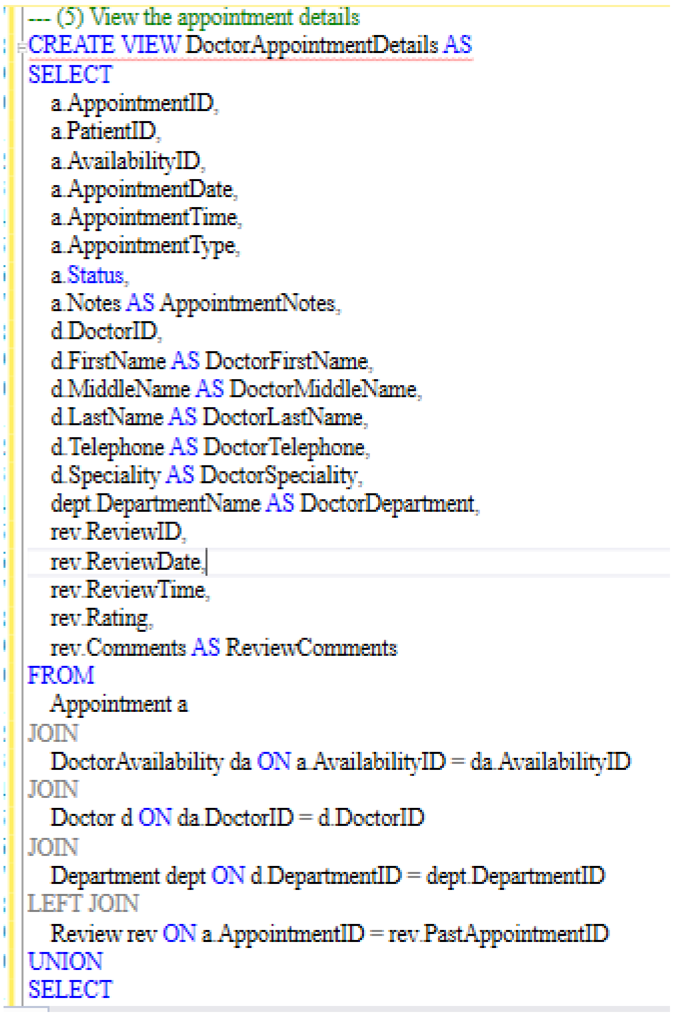
The provided SQL code creates a stored procedure named "MoveAndDeleteCancelledAppointments" to handle the deletion of appointments with a status of 'Cancelled'. After defining the stored procedure, ‘SET NOCOUNT ON’ prevents the count of rows affected by a Transact-SQL statement from being returned as part of the result set.

DELETE FROM Prescription WHERE AppointmentID IN ...: Deletes associated records in the "Prescription" table where the appointment status is 'Cancelled' or 'Completed'.

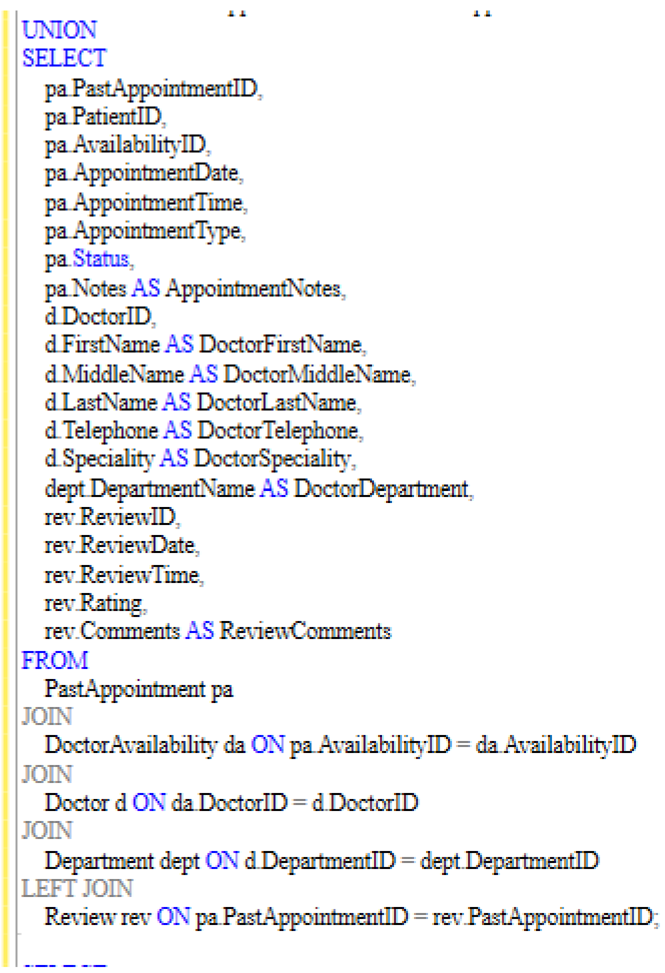
INSERT INTO PastAppointment ...: Moves cancelled appointments to the "PastAppointment" table. It selects relevant columns from the "Appointment" table where the status is 'Cancelled' or 'Completed' and inserts them into the "PastAppointment" table.

DELETE FROM Appointment WHERE Status IN ...: Deletes cancelled and completed appointments from the "Appointment" table based on the specified status criteria. Summarily, this SQL code effectively creates a stored procedure to handle the deletion of cancelled appointments, moves them to the "PastAppointment" table, and deletes them from the "Appointment" table. It then verifies the changes by selecting all records from the "Appointment" table.

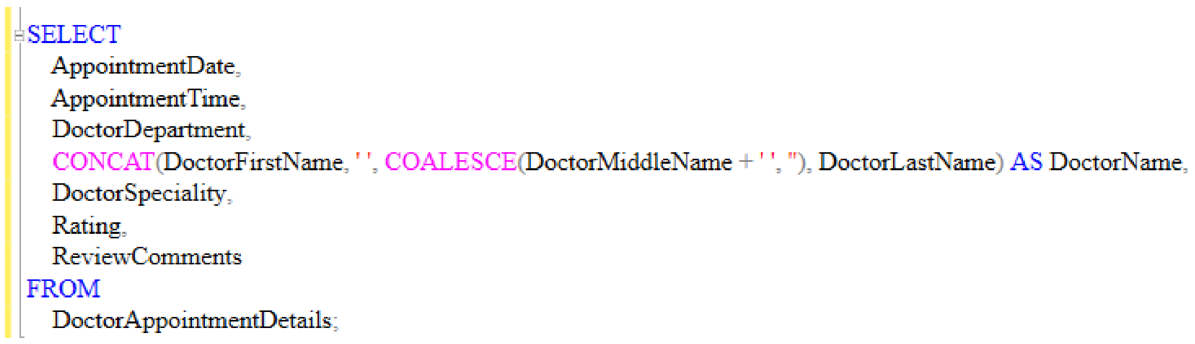
QUESTION 5



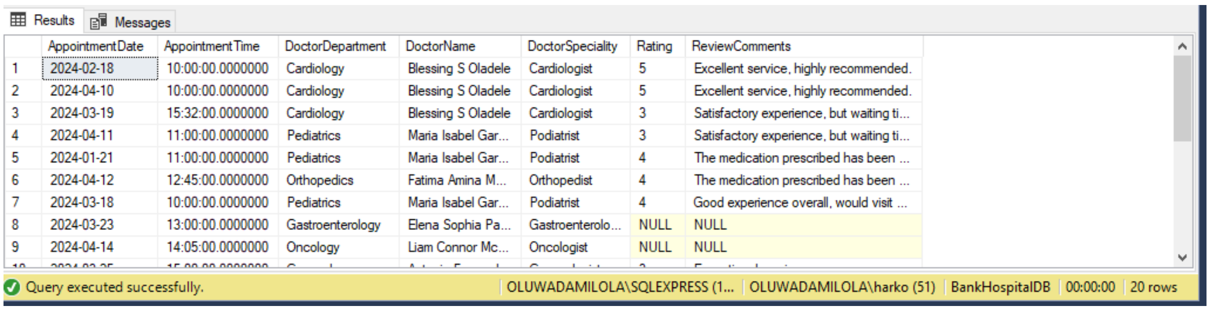
```
--- (5) View the appointment details
CREATE VIEW DoctorAppointmentDetails AS
SELECT
    a.AppointmentID,
    a.PatientID,
    a.AvailabilityID,
    a.AppointmentDate,
    a.AppointmentTime,
    a.AppointmentType,
    a.Status,
    a.Notes AS AppointmentNotes,
    d.DoctorID,
    d.FirstName AS DoctorFirstName,
    d.MiddleName AS DoctorMiddleName,
    d.LastName AS DoctorLastName,
    d.Telephone AS DoctorTelephone,
    d.Speciality AS DoctorSpeciality,
    dept.DepartmentName AS DoctorDepartment,
    rev.ReviewID,
    rev.ReviewDate,
    rev.ReviewTime,
    rev.Rating,
    rev.Comments AS ReviewComments
FROM
    Appointment a
JOIN
    DoctorAvailability da ON a.AvailabilityID = da.AvailabilityID
JOIN
    Doctor d ON da.DoctorID = d.DoctorID
JOIN
    Department dept ON d.DepartmentID = dept.DepartmentID
LEFT JOIN
    Review rev ON a.AppointmentID = rev.PastAppointmentID
UNION
SELECT
```



```sql
UNION
SELECT
    pa.PastAppointmentID,
    pa.PatientID,
    pa.AvailabilityID,
    pa.AppointmentDate,
    pa.AppointmentTime,
    pa.AppointmentType,
    pa.Status,
    pa.Notes AS AppointmentNotes,
    d.DoctorID,
    d.FirstName AS DoctorFirstName,
    d.MiddleName AS DoctorMiddleName,
    d.LastName AS DoctorLastName,
    d.Telephone AS DoctorTelephone,
    d.Speciality AS DoctorSpeciality,
    dept.DepartmentName AS DoctorDepartment,
    rev.ReviewID,
    rev.ReviewDate,
    rev.ReviewTime,
    rev.Rating,
    rev.Comments AS ReviewComments
FROM
    PastAppointment pa
JOIN
    DoctorAvailability da ON pa.AvailabilityID = da.AvailabilityID
JOIN
    Doctor d ON da.DoctorID = d.DoctorID
JOIN
    Department dept ON d.DepartmentID = dept.DepartmentID
LEFT JOIN
    Review rev ON pa.PastAppointmentID = rev.PastAppointmentID;
```



```sql
SELECT
    AppointmentDate,
    AppointmentTime,
    DoctorDepartment,
    CONCAT(DoctorFirstName, ' ', COALESCE(DoctorMiddleName + ' ', ''), DoctorLastName) AS DoctorName,
    DoctorSpeciality,
    Rating,
    ReviewComments
FROM
    DoctorAppointmentDetails;
```



	AppointmentDate	AppointmentTime	DoctorDepartment	DoctorName	DoctorSpeciality	Rating	ReviewComments
1	2024-02-18	10:00:00.0000000	Cardiology	Blessing S Oladele	Cardiologist	5	Excellent service, highly recommended.
2	2024-04-10	10:00:00.0000000	Cardiology	Blessing S Oladele	Cardiologist	5	Excellent service, highly recommended.
3	2024-03-19	15:32:00.0000000	Cardiology	Blessing S Oladele	Cardiologist	3	Satisfactory experience, but waiting ti...
4	2024-04-11	11:00:00.0000000	Pediatrics	Maria Isabel Gar...	Podiatrist	3	Satisfactory experience, but waiting ti...
5	2024-01-21	11:00:00.0000000	Pediatrics	Maria Isabel Gar...	Podiatrist	4	The medication prescribed has been ...
6	2024-04-12	12:45:00.0000000	Orthopedics	Fatima Amina M...	Orthopedist	4	The medication prescribed has been ...
7	2024-03-18	10:00:00.0000000	Pediatrics	Maria Isabel Gar...	Podiatrist	4	Good experience overall, would visit ...
8	2024-03-23	13:00:00.0000000	Gastroenterology	Elena Sophia Pa...	Gastroenterolo...	NULL	NULL
9	2024-04-14	14:05:00.0000000	Oncology	Liam Connor Mc...	Oncologist	NULL	NULL

```
Query executed successfully.    OLUWADAMILOLA\SQLEXPRESS (1...    OLUWADAMILOLA\harko (51)    BankHospitalDB    00:00:00    20 rows
```

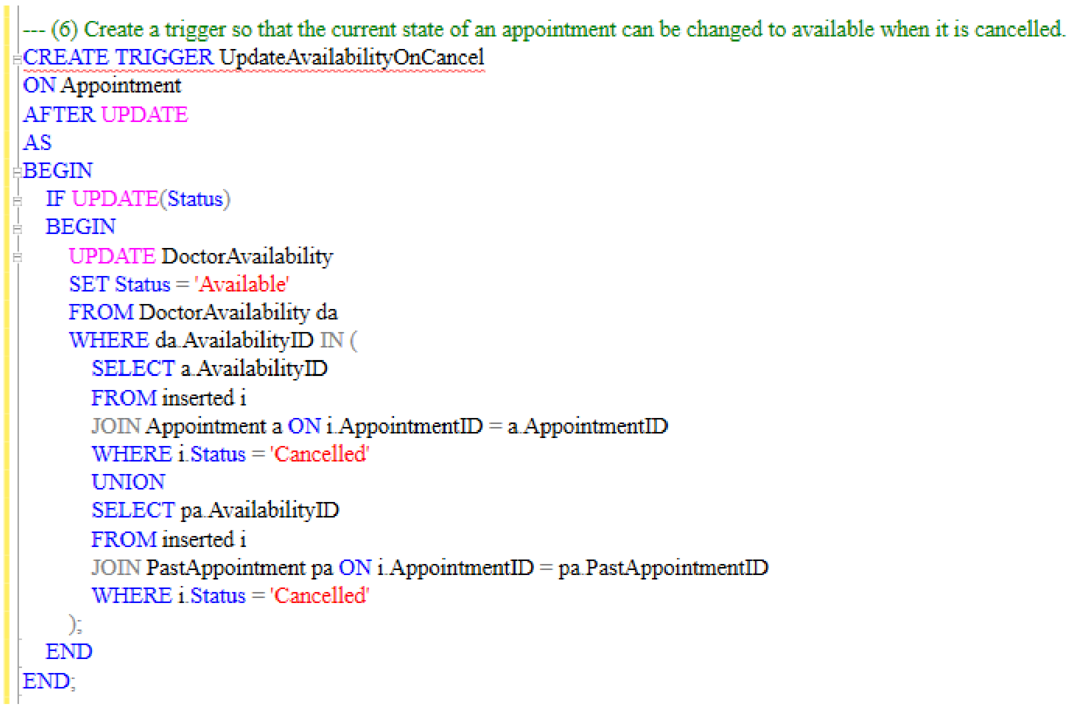
Figure 4.2

To display appointment details in an organized manner, the SQL code first creates a view to combine all of the appointment data, and then it picks out particular columns

from the view.
To obtain complete appointment information, the 'DoctorAppointmentDetails' view is created by merging information from the appointment and past appointment tables using the union operator and joining it with the Doctor, Doctor availability, department, and review table (where appropriate, inner joins were used to include review details, if available).

The "AppointmentDate," "AppointmentTime," "DoctorDepartment," and "DoctorName" appointment details are retrieved from the "DoctorAppointmentDetails" using the SELECT statement.
To represent a doctor's name in a single column, concatenate the doctor's last name, middle name (if available), and first name using the CONCAT function. To ensure correct concatenation, the COALESCE function is used to handle situations in which the middle name may be NULL.

QUESTION 6

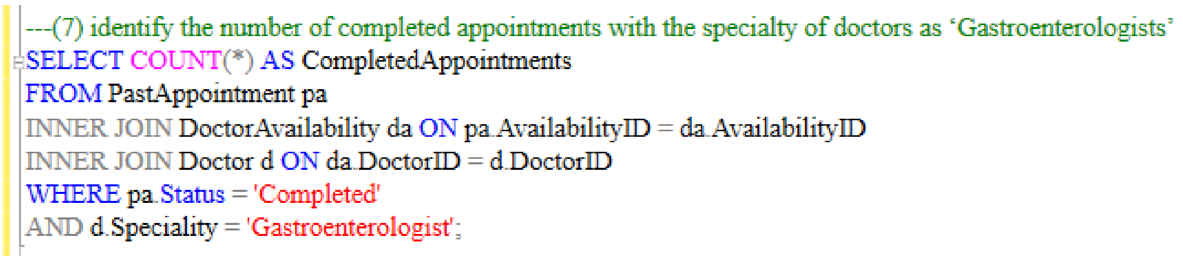


```
--- (6) Create a trigger so that the current state of an appointment can be changed to available when it is cancelled.
CREATE TRIGGER UpdateAvailabilityOnCancel
ON Appointment
AFTER UPDATE
AS
BEGIN
   IF UPDATE(Status)
   BEGIN
      UPDATE DoctorAvailability
      SET Status = 'Available'
      FROM DoctorAvailability da
      WHERE da.AvailabilityID IN (
         SELECT a.AvailabilityID
         FROM inserted i
         JOIN Appointment a ON i.AppointmentID = a.AppointmentID
         WHERE i.Status = 'Cancelled'
         UNION
         SELECT pa.AvailabilityID
         FROM inserted i
         JOIN PastAppointment pa ON i.AppointmentID = pa.PastAppointmentID
         WHERE i.Status = 'Cancelled'
      );
   END
END;
```

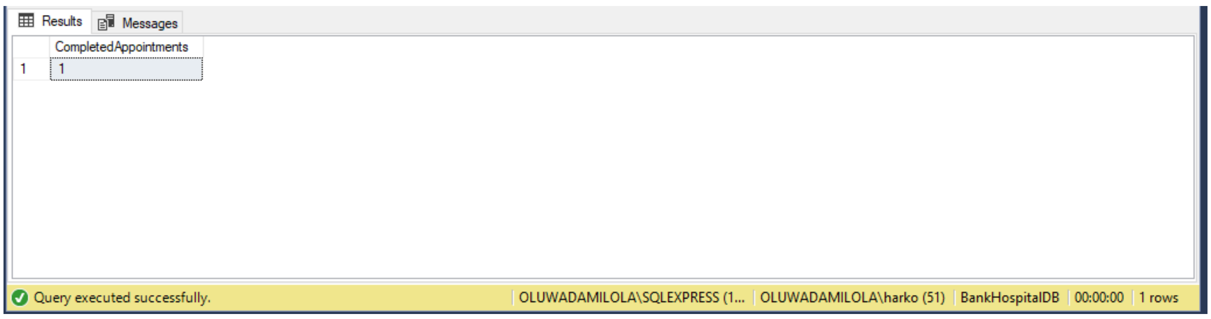
Figure 4.3
When a patient cancels an appointment or the status is changed to cancel, the trigger named Update Availability on cancel which is linked to the Appointment table updates the doctor's availability to available. To meet the requirement, it also combines the past appointment and the appointment table.

This trigger ensures that when an appointment is updated with a status of 'Cancelled', the corresponding doctor availability status is updated to 'Available'. It effectively maintains consistency in the availability status of doctors based on the status of appointments.

QUESTION 7



```
---(7) identify the number of completed appointments with the specialty of doctors as 'Gastroenterologists'
SELECT COUNT(*) AS CompletedAppointments
FROM PastAppointment pa
INNER JOIN DoctorAvailability da ON pa.AvailabilityID = da.AvailabilityID
INNER JOIN Doctor d ON da.DoctorID = d.DoctorID
WHERE pa.Status = 'Completed'
AND d.Speciality = 'Gastroenterologist';
```



```
   CompletedAppointments
1  1
```

```
Query executed successfully.   OLUWADAMILOLA\SQLEXPRESS (1...   OLUWADAMILOLA\harko (51)   BankHospitalDB   00:00:00   1 rows
```

Figure 4.4

This query effectively retrieves the count of completed appointments associated with doctors specializing in 'Gastroenterology'. It utilizes JOIN operations (Past appointment since it has the completed appointments, Doctor availability, Doctor) using the correct foreign keys and WHERE clause to apply to filter based on status and speciality.

<u>Additional Recommendations</u>

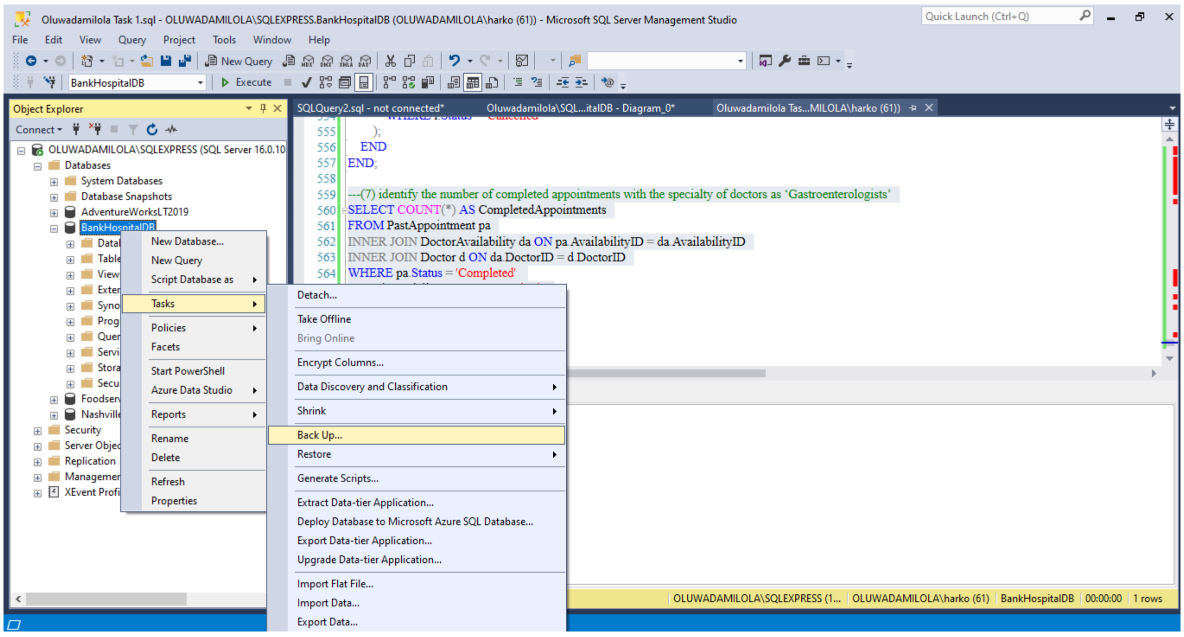
Data Integrity and Concurrency:

- The use of appropriate constraints like primary keys, foreign keys, unique constraints, NOT NULL, time stamps for dates and check constraints to enforce referential and data integrity rules at the database level.
- Taking into account the data types for every column according to the type of data that will be stored to guarantee data integrity, effective storage utilization, and compatibility with the planned operations and queries.
- In database transactions, the ACID (Atomicity, Consistency, Isolation, Durability) properties were considered.

Database Backup and Recovery:

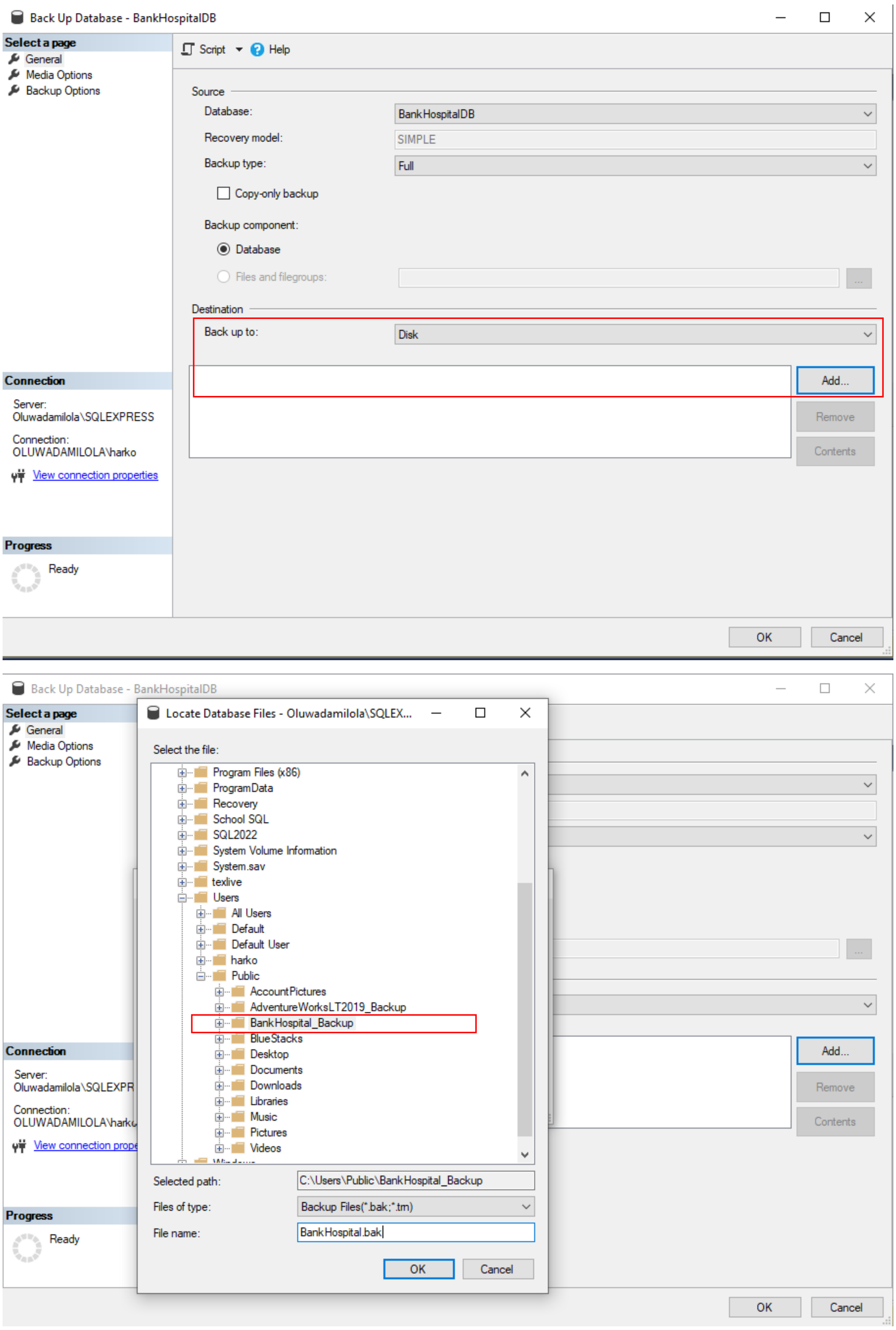
- Periodically, checkpoints will be established to minimize the quantity of data lost during system outages.
- Employees should be trained in log file comprehension, updating, and failure recovery.
- Test the backup and recovery procedure regularly to ensure that backups are dependable and successfully restorable.

My process for performing a full database backup seems straightforward. By using the Object Explorer in the database management system then easily navigate to the desired database (BankHospital database), initiate a backup task, and specify the destination for the backup file. Storing the backup on a disk in your computer provides a convenient and accessible location for recovery purposes.
Also, the approach to recovery steps, which involves selecting the backup file from the designated drive, is a standard and effective method for restoring the database to a previous state in the event of data loss or system failure.
The figures below show the step-by-step approach used in creating the backup.



```sql
        );
    END
END;

---(7) identify the number of completed appointments with the specialty of doctors as 'Gastroenterologists'
SELECT COUNT(*) AS CompletedAppointments
FROM PastAppointment pa
INNER JOIN DoctorAvailability da ON pa.AvailabilityID = da.AvailabilityID
INNER JOIN Doctor d ON da.DoctorID = d.DoctorID
WHERE pa.Status = 'Completed'
```

Figure 4.5


Back Up Database - BankHospitalDB

Select a page
General
Media Options
Backup Options

Script Help

Source
Database: BankHospitalDB
Recovery model: SIMPLE
Backup type: Full
[ ] Copy-only backup

Backup component:
(●) Database
( ) Files and filegroups:

Destination
Back up to: Disk

Add...
Remove
Contents

Connection
Server:
Oluwadamilola\SQLEXPRESS
Connection:
OLUWADAMILOLA\harko
View connection properties

Progress
Ready

OK Cancel

Locate Database Files - Oluwadamilola\SQLEX...

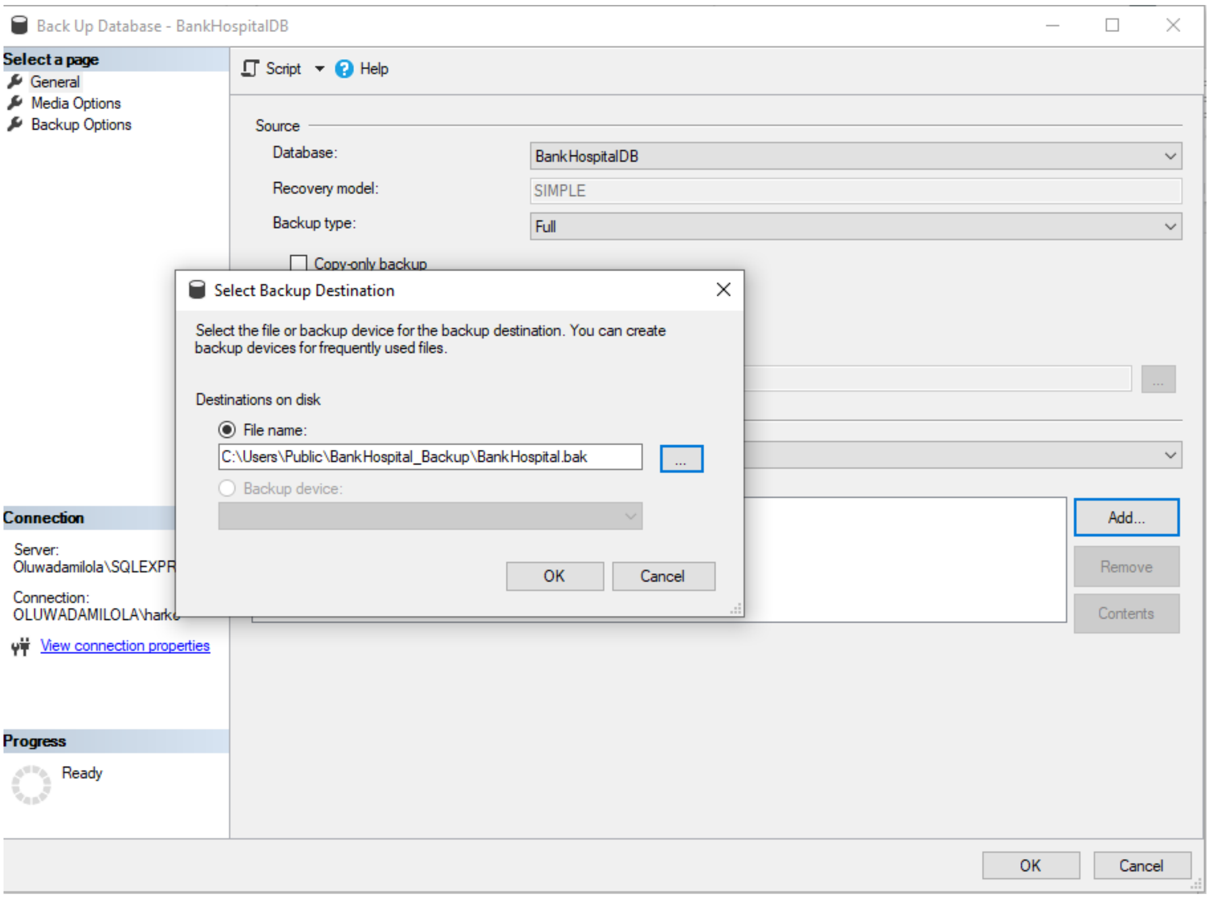
Select the file:

```
Program Files (x86)
ProgramData
Recovery
School SQL
SQL2022
System Volume Information
System.sav
texlive
Users
    All Users
    Default
    Default User
    harko
    Public
        AccountPictures
        AdventureWorksLT2019_Backup
        BankHospital_Backup
        BlueStacks
        Desktop
        Documents
        Downloads
        Libraries
        Music
        Pictures
        Videos
```

Selected path: C:\Users\Public\BankHospital_Backup
Files of type: Backup Files(*.bak;*.tm)
File name: BankHospital.bak

OK Cancel

Figure 4.6


Back Up Database - BankHospitalDB

Select a page: General, Media Options, Backup Options

Source
Database: BankHospitalDB
Recovery model: SIMPLE
Backup type: Full
[ ] Copy-only backup

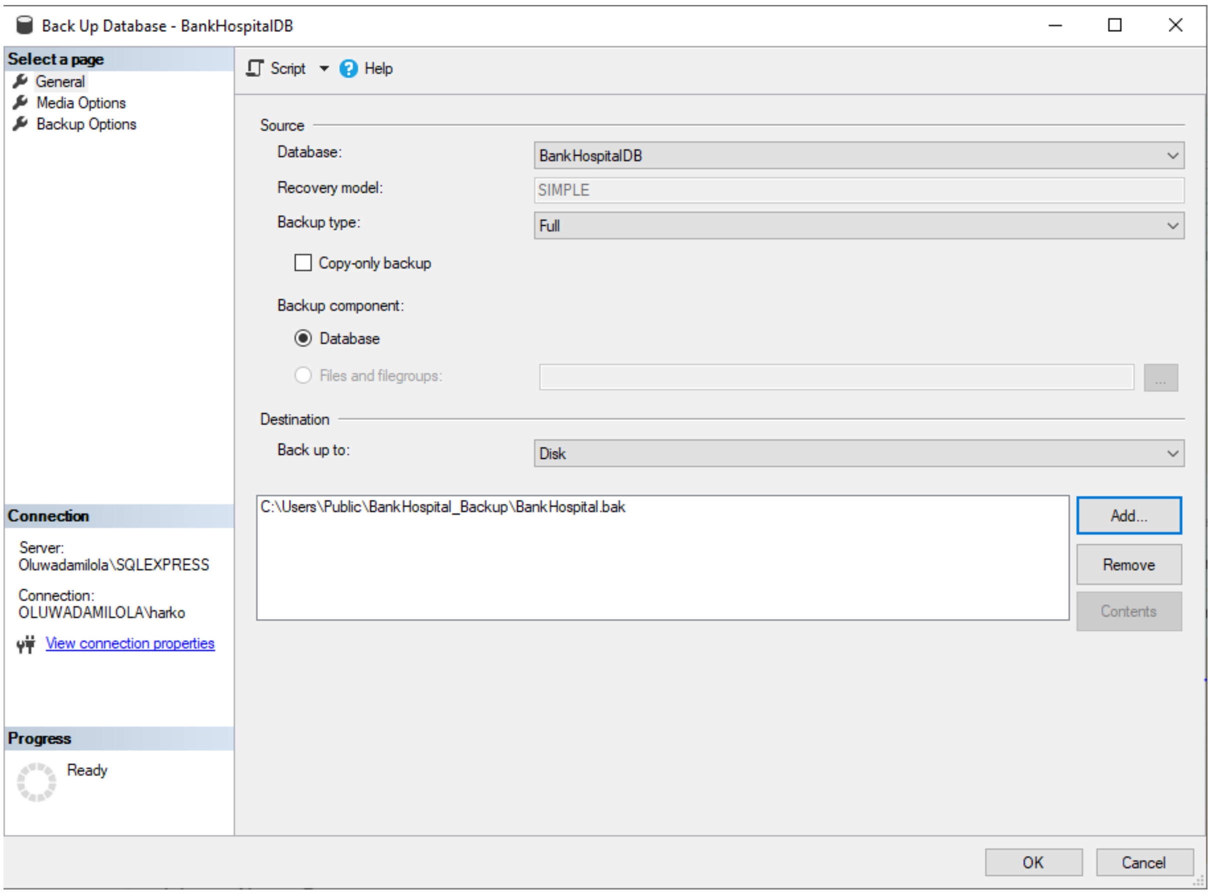
Select Backup Destination

Select the file or backup device for the backup destination. You can create backup devices for frequently used files.

Destinations on disk
[✓ File name:] C:\Users\Public\BankHospital_Backup\BankHospital.bak
Backup device:

Connection
Server: Oluwadamilola\SQLEXPR
Connection: OLUWADAMILOLA\harko

Progress: Ready


Back Up Database - BankHospitalDB

Select a page: General, Media Options, Backup Options

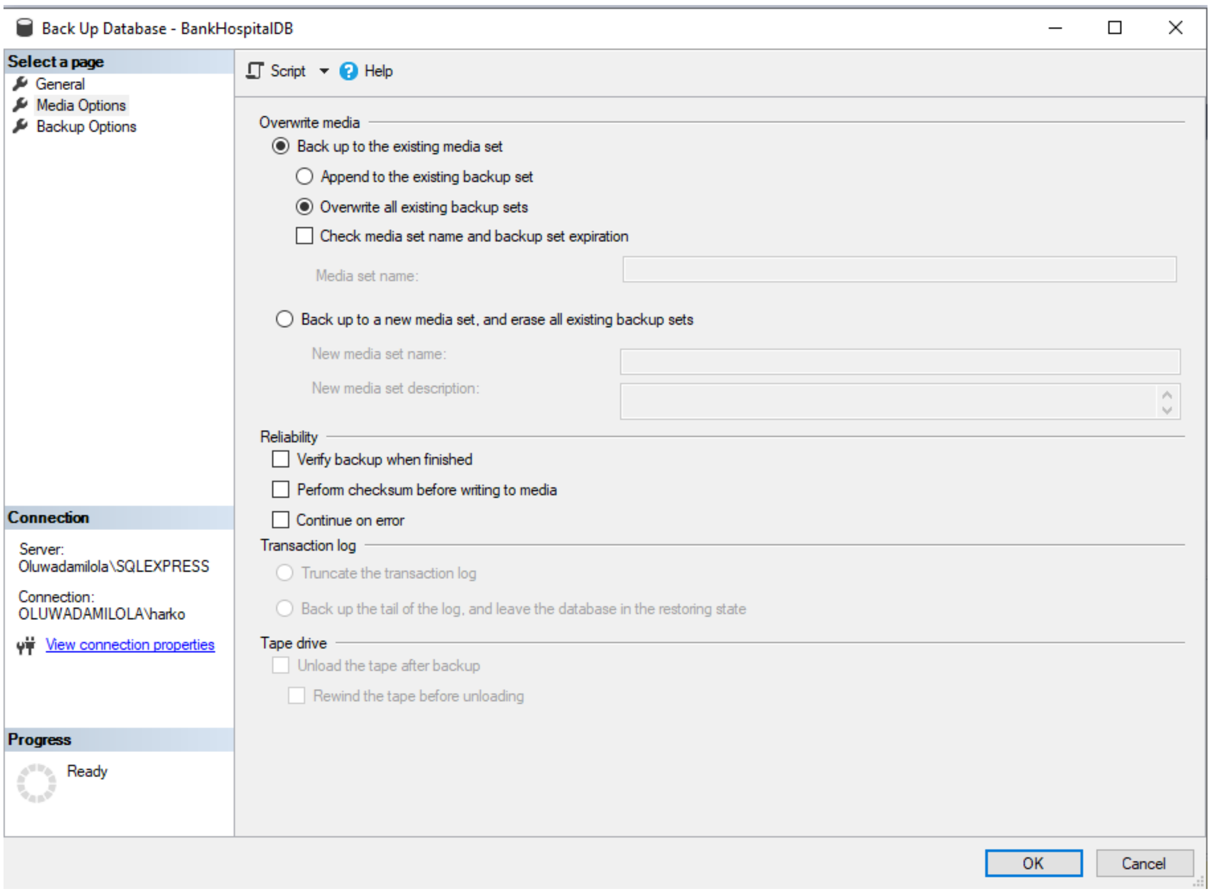
Source
Database: BankHospitalDB
Recovery model: SIMPLE
Backup type: Full
[ ] Copy-only backup
Backup component:
[✓ Database]
Files and filegroups:

Destination
Back up to: Disk
C:\Users\Public\BankHospital_Backup\BankHospital.bak

Connection
Server: Oluwadamilola\SQLEXPRESS
Connection: OLUWADAMILOLA\harko

Progress: Ready

Figure 4.7


Back Up Database - BankHospitalDB

Select a page: General; Media Options; Backup Options

Overwrite media
- (●) Back up to the existing media set
  - ( ) Append to the existing backup set
  - (●) Overwrite all existing backup sets
  - [ ] Check media set name and backup set expiration
  - Media set name:
- ( ) Back up to a new media set, and erase all existing backup sets
  - New media set name:
  - New media set description:

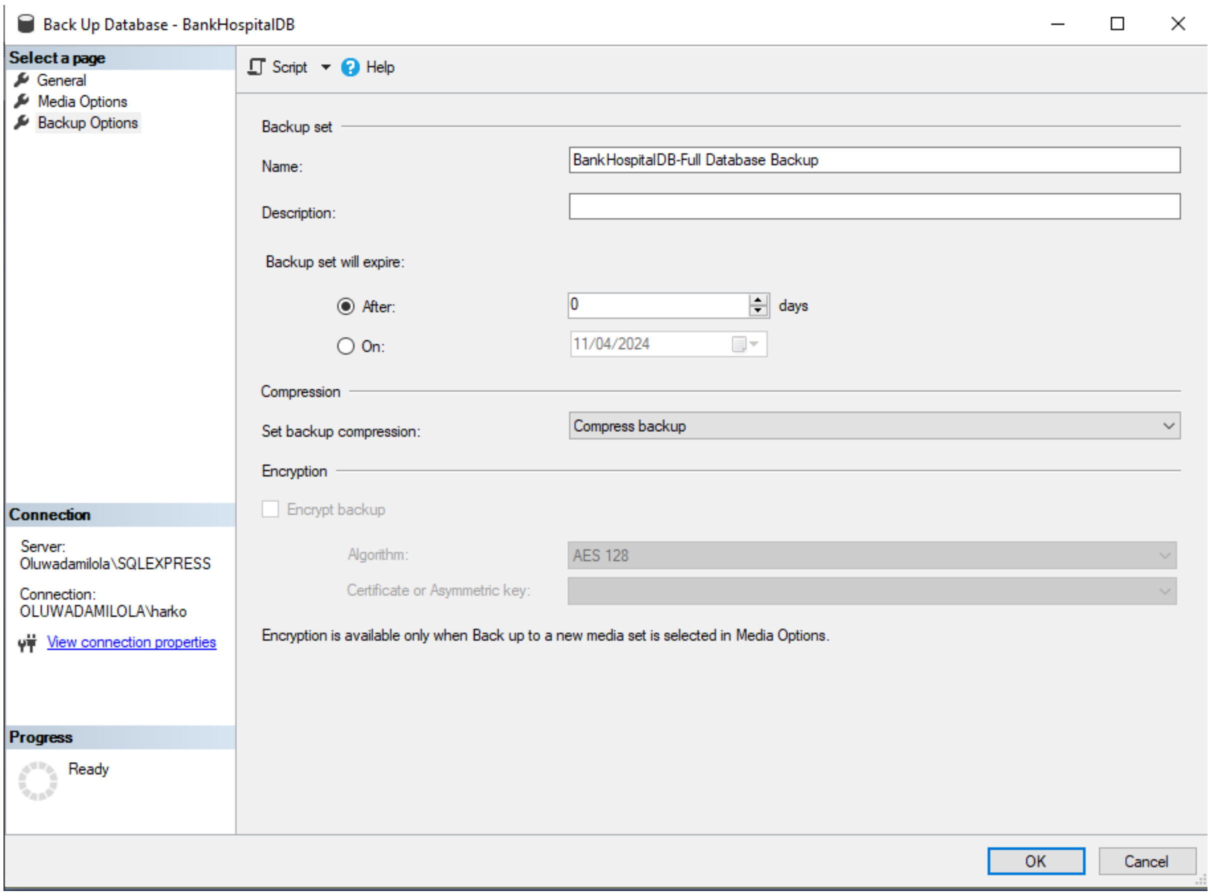
Reliability
- [ ] Verify backup when finished
- [ ] Perform checksum before writing to media
- [ ] Continue on error

Transaction log
- ( ) Truncate the transaction log
- ( ) Back up the tail of the log, and leave the database in the restoring state

Tape drive
- [ ] Unload the tape after backup
- [ ] Rewind the tape before unloading

Connection
Server: Oluwadamilola\SQLEXPRESS
Connection: OLUWADAMILOLA\harko
View connection properties

Progress: Ready


Back Up Database - BankHospitalDB

Select a page: General; Media Options; Backup Options

Backup set
- Name: BankHospitalDB-Full Database Backup
- Description:
- Backup set will expire:
  - (●) After: 0 days
  - ( ) On: 11/04/2024

Compression
- Set backup compression: Compress backup

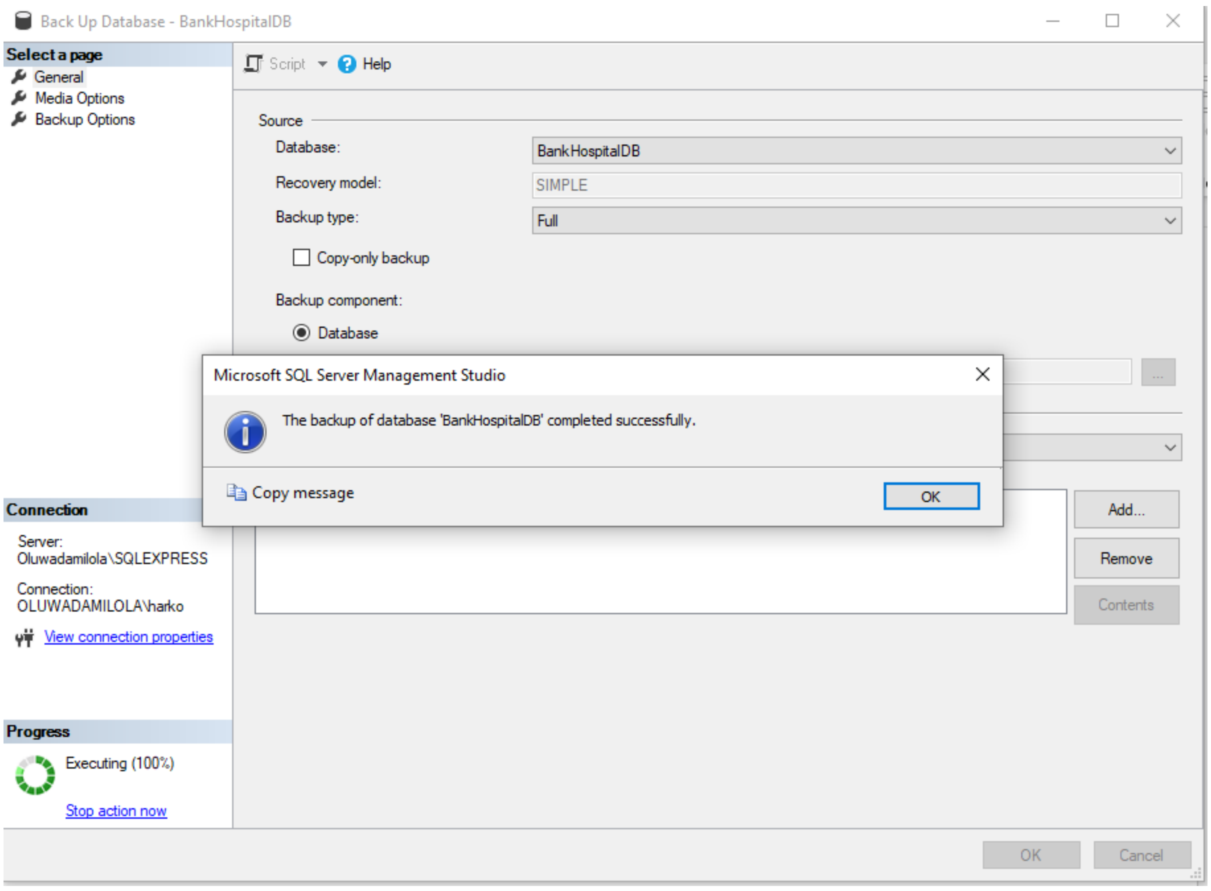
Encryption
- [ ] Encrypt backup
  - Algorithm: AES 128
  - Certificate or Asymmetric key:

Encryption is available only when Back up to a new media set is selected in Media Options.

Connection
Server: Oluwadamilola\SQLEXPRESS
Connection: OLUWADAMILOLA\harko
View connection properties

Progress: Ready

Figure 4.8



Back Up Database - BankHospitalDB

Source
Database: BankHospitalDB
Recovery model: SIMPLE
Backup type: Full
Copy-only backup
Backup component: Database

Microsoft SQL Server Management Studio

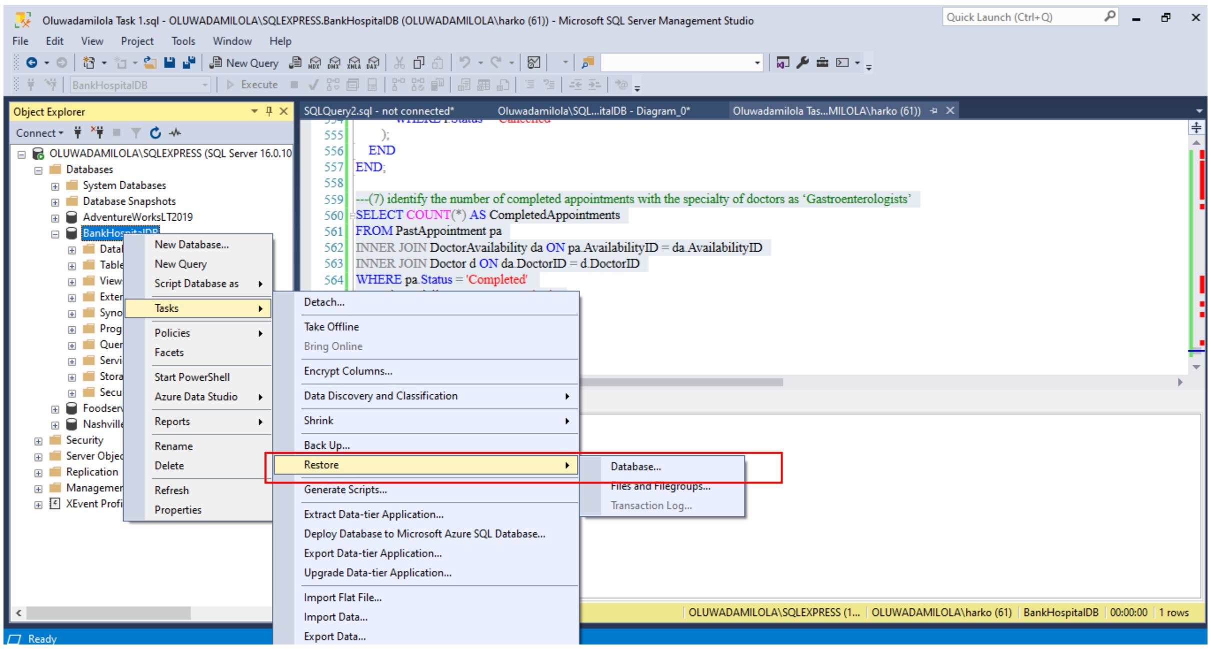
The backup of database 'BankHospitalDB' completed successfully.

Connection
Server: Oluwadamilola\SQLEXPRESS
Connection: OLUWADAMILOLA\harko

Progress
Executing (100%)

Figure 4.9

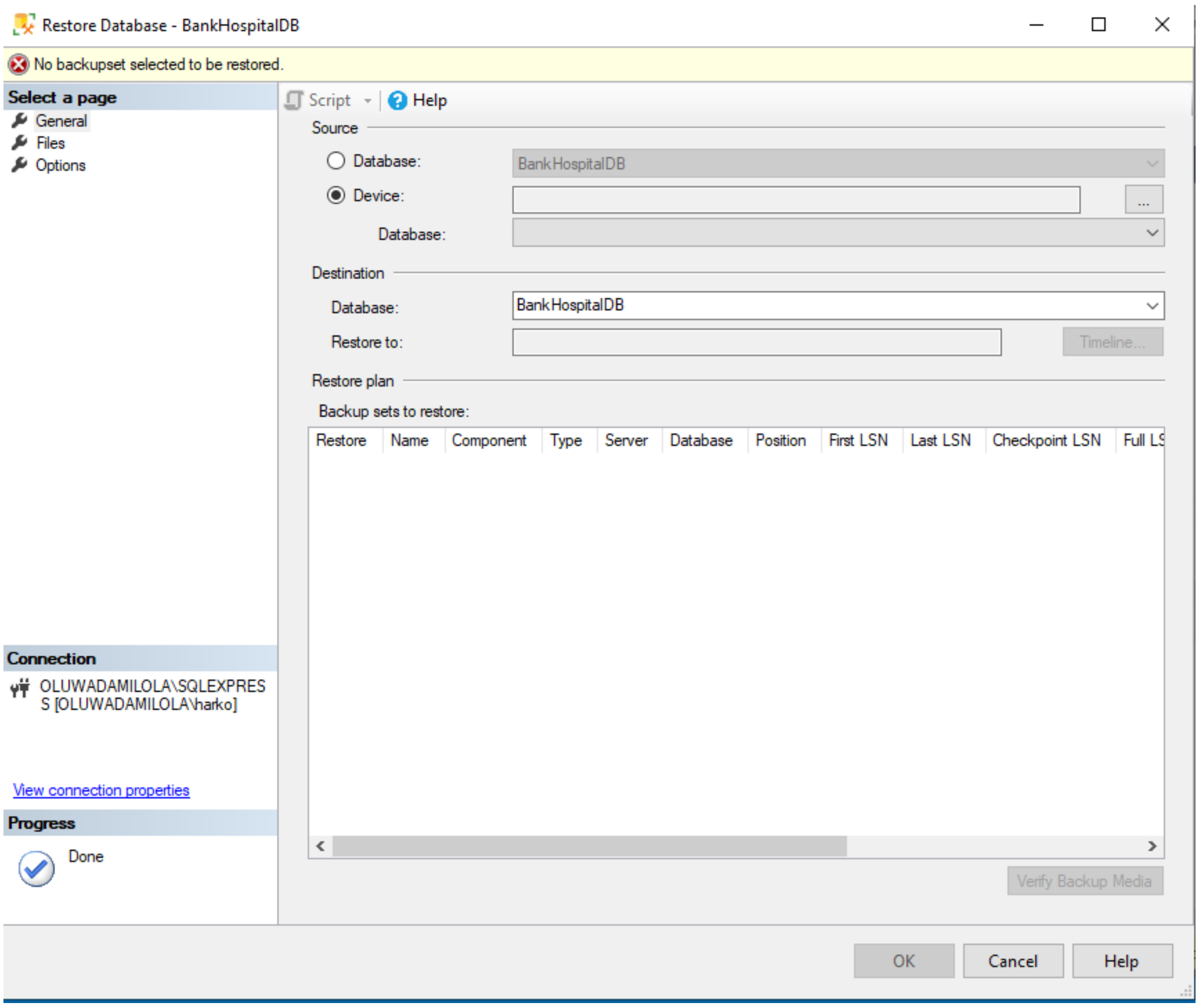
DATABASE RECOVERY



Oluwadamilola Task 1.sql - OLUWADAMILOLA\SQLEXPRESS.BankHospitalDB (OLUWADAMILOLA\harko (61)) - Microsoft SQL Server Management Studio

```
    );
  END
END;

---(7) identify the number of completed appointments with the specialty of doctors as 'Gastroenterologists'
SELECT COUNT(*) AS CompletedAppointments
FROM PastAppointment pa
INNER JOIN DoctorAvailability da ON pa.AvailabilityID = da.AvailabilityID
INNER JOIN Doctor d ON da.DoctorID = d.DoctorID
WHERE pa.Status = 'Completed'
```

Figure 5.0


Restore Database - BankHospitalDB

No backupset selected to be restored.

Select a page
- General
- Files
- Options

Source
- Database: BankHospitalDB
- [✓ Device:]
- Database:

Destination
- Database: BankHospitalDB
- Restore to:

Restore plan

Backup sets to restore:

Restore	Name	Component	Type	Server	Database	Position	First LSN	Last LSN	Checkpoint LSN	Full LS

Connection

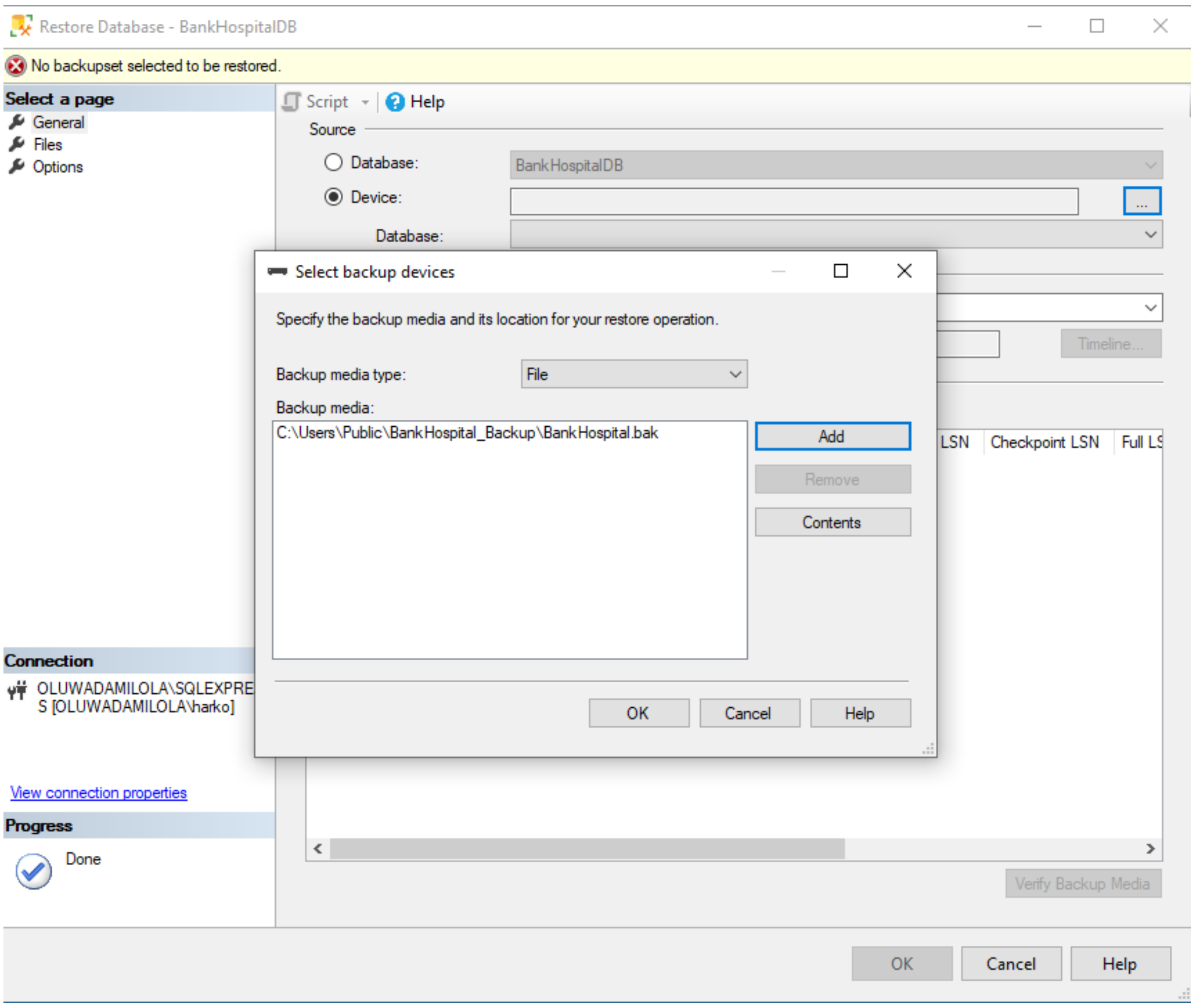
OLUWADAMILOLA\SQLEXPRESS [OLUWADAMILOLA\harko]

View connection properties

Progress

Done

Figure 5.1


Restore Database - BankHospitalDB

No backupset selected to be restored.

Select a page: General, Files, Options

Source

Database: BankHospitalDB

(●) Device:

Database:

Select backup devices

Specify the backup media and its location for your restore operation.

Backup media type: File

Backup media:

C:\Users\Public\BankHospital_Backup\BankHospital.bak

Connection

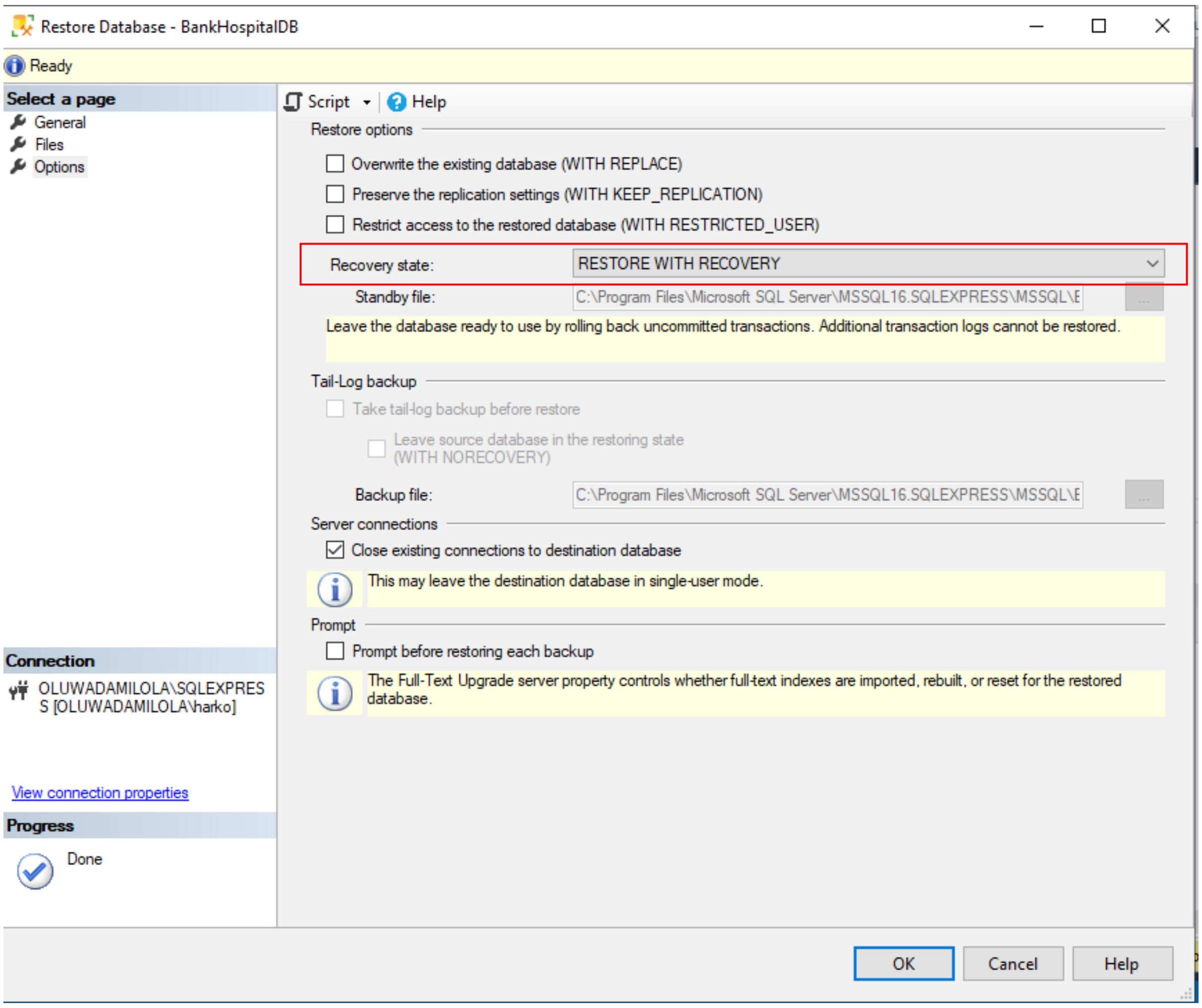
OLUWADAMILOLA\SQLEXPRES [OLUWADAMILOLA\harko]

View connection properties

Progress

Done

Figure 5.2



Restore options

☐ Overwrite the existing database (WITH REPLACE)

☐ Preserve the replication settings (WITH KEEP_REPLICATION)

☐ Restrict access to the restored database (WITH RESTRICTED_USER)

Recovery state: RESTORE WITH RECOVERY

Standby file: C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\E

Leave the database ready to use by rolling back uncommitted transactions. Additional transaction logs cannot be restored.

Tail-Log backup

☐ Take tail-log backup before restore

☐ Leave source database in the restoring state (WITH NORECOVERY)

Backup file: C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\E

Server connections

☑ Close existing connections to destination database

This may leave the destination database in single-user mode.

Prompt

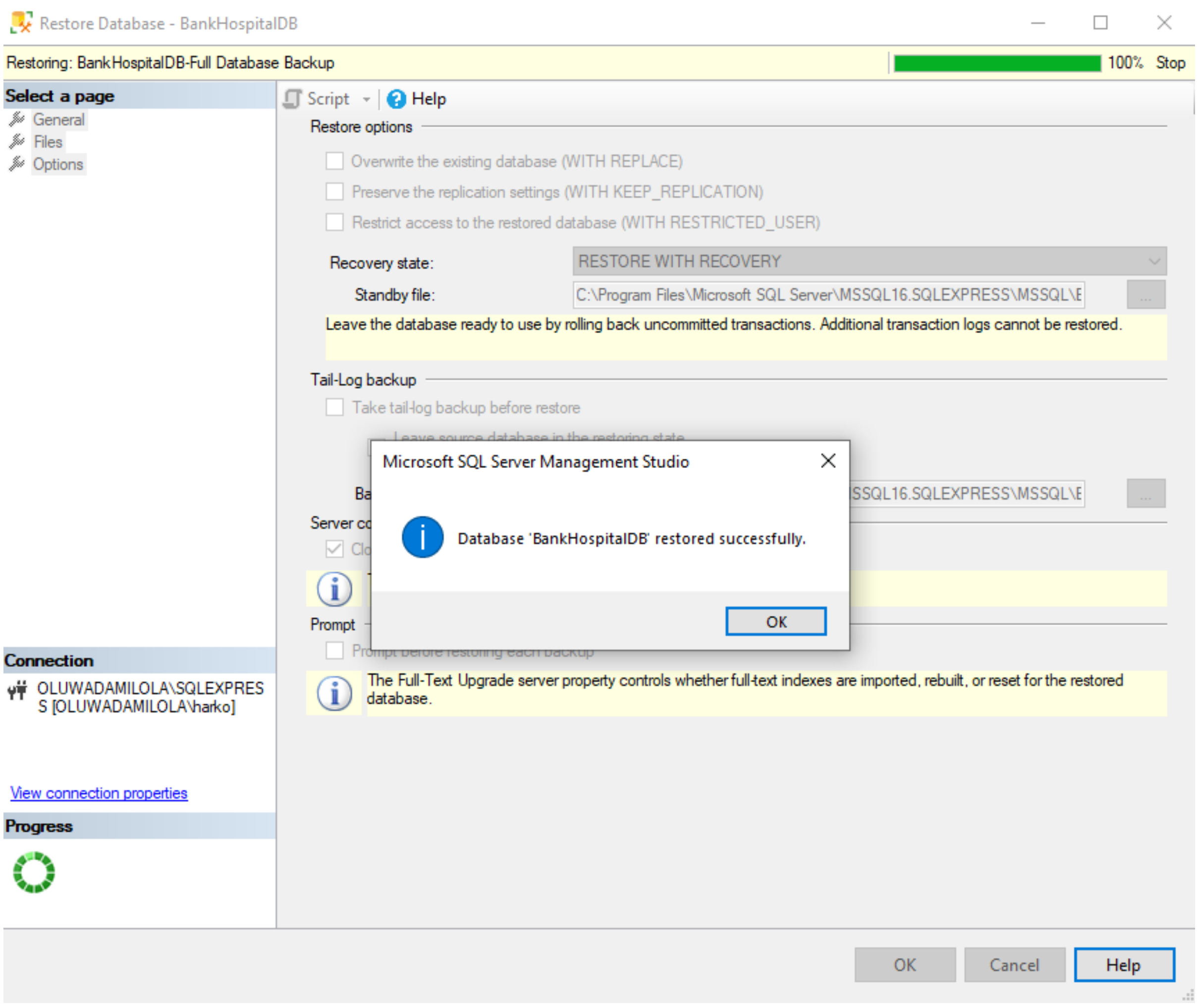
☐ Prompt before restoring each backup

The Full-Text Upgrade server property controls whether full-text indexes are imported, rebuilt, or reset for the restored database.

OLUWADAMILOLA\SQLEXPRESS [OLUWADAMILOLA\harko]

Done

Figure 5.3


Restore Database - BankHospitalDB

Restoring: BankHospitalDB-Full Database Backup 100% Stop

Select a page
- General
- Files
- Options

Script · Help

Restore options

- [ ] Overwrite the existing database (WITH REPLACE)
- [ ] Preserve the replication settings (WITH KEEP_REPLICATION)
- [ ] Restrict access to the restored database (WITH RESTRICTED_USER)

Recovery state: RESTORE WITH RECOVERY

Standby file: C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\E

Leave the database ready to use by rolling back uncommitted transactions. Additional transaction logs cannot be restored.

Tail-Log backup

- [ ] Take tail-log backup before restore

Microsoft SQL Server Management Studio

Database 'BankHospitalDB' restored successfully.

OK

The Full-Text Upgrade server property controls whether full-text indexes are imported, rebuilt, or reset for the restored database.

Connection

OLUWADAMILOLA\SQLEXPRESS [OLUWADAMILOLA\harko]

View connection properties

Progress

OK Cancel Help

Figure 5.4

Database Security:

- **Authorization:** This is achieved by the username and password that was used to create or access the patient portal.
- Creation of users and granting privileges to the right users.
- Creation of views and stored procedures for users to be able to only access the data they need.
- Hashing passwords in the event of a security breach
- Encrypt backup files to prevent unauthorized access to sensitive data during storage or transmission.

CONCLUSION

The created database has been normalized to 3NF to eliminate any redundant information or anomalies. The database tables are clearly defined, and the table names accurately indicate the types of data they contain. The design is shown above for detail, and the relationships have been established appropriately.

The hospital can improve the integrity, security, and dependability of its database system and guarantee the privacy, accessibility, and accuracy of patient and operational data by complying with the above recommendations and guidelines. The hospital will be able to continue operations even in difficult situations due to proactive backup and recovery procedures that reduce the risk of data loss and system failures.

Some of the key functionalities include;

- The ability to insert, update and delete data in the database such as the doctor update done earlier (Data Manipulation).
- Implementing constraints such as primary keys, foreign keys, check constraints, and other validation rules.
- Creating and managing tables to store different types of data
- Views have been created to make easier user interfaces by presenting particular subsets of data or carrying out complex queries.
- Stored procedures improve efficiency and security by containing frequently used operations or detailed logic into reusable code units.
- Attaching data backup and recovery procedures in place to guard against data loss and guarantee system reliability.
- Enforcing security protocols like encryption, authorization, and authentication is also necessary to protect the availability, confidentiality, and integrity of data.