J119811_assignment.pdf

by MD TANVIR SARDARR

Submission date: 31-Mar-2025 12:18AM (UTC+0100)
Submission ID: 254249179
File name: J119811_assignment.pdf_1199552_2026389306.pdf (2.35M)
Word count: 3937
Character count: 27179

SQL Database Design and Build Report for MediCareDB System

Student Assessment Number: J119811

Module Code: CO7401

Module Title: Databases – SQL and NoSQL

Word count: 1300 (Excluding References, Headings & Appendix)

Date of Submission: 31 March 2025

Contents

1. Organisation Overview

Organisation Name: MediCareDB
Function: Digital Hospital Information and Management System

Overview:

The hospital management system MediCareDB provides comprehensive support for contemporary healthcare institutions to manage patient record processing, transaction maintenance, and analytics. MediCareDB fulfills its purpose by operating within a multi-specialty hospital, with sections dedicated to Cardiology, Neurology, Pediatrics, and General Medicine. As its primary objective, the system streamlines various operational workflows, encompassing patient information management, doctor directory access, appointment scheduling, prescription writing, billing tasks, and laboratory result management.

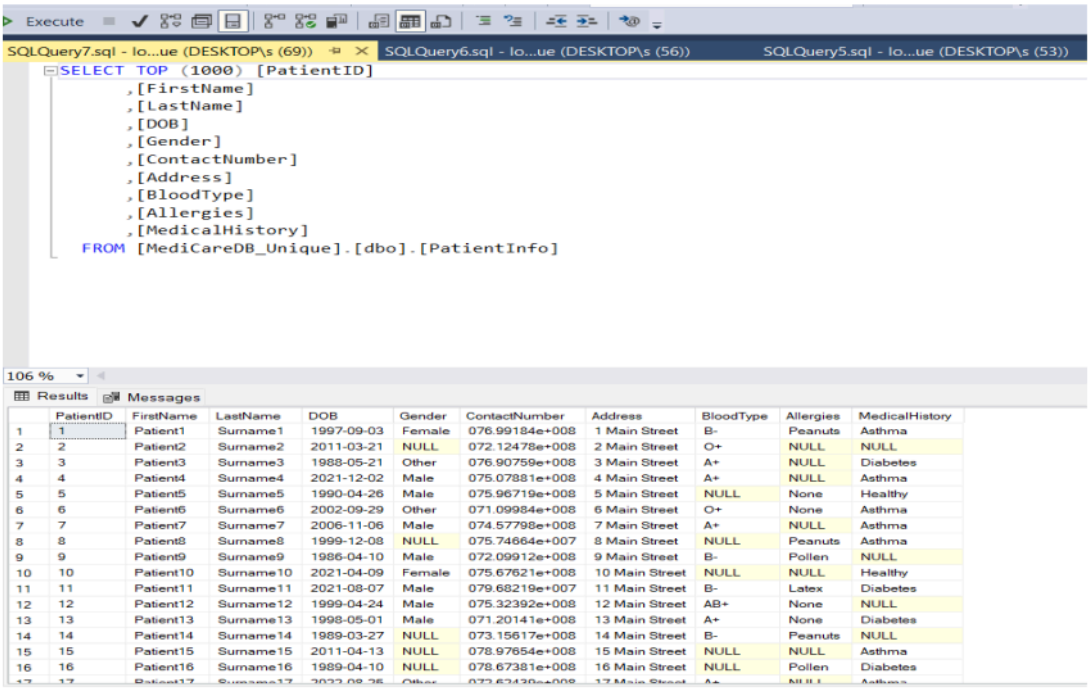
Justification:

Hospital Information Systems (HIS) play a crucial role in the efficient management of healthcare facilities. Samra et al. (2020) demonstrate that well-connected HIS platforms enhance both healthcare data storage as well as operational effectiveness and medical research capabilities together with policy development and clinical audit functionalities. MediCareDB implements a modular SQL-based solution that integrates as a unified healthcare solution for medical environments.

2. Data Modelling

2.1 Identifying Main Tables

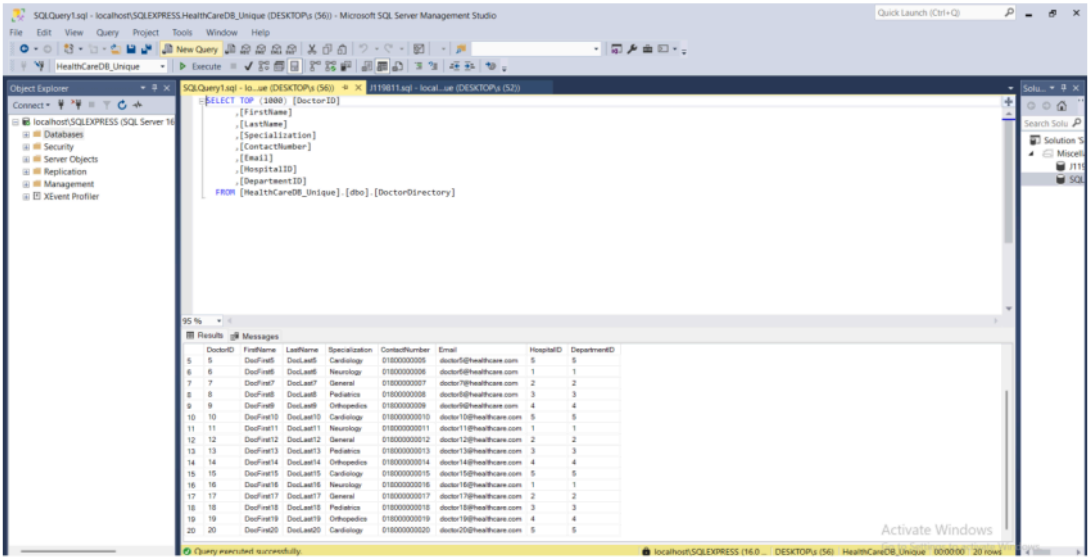
Patient Table: demographics, alongside contact information, along with medical chronicles, allergies, and blood type information are stored.



```
SELECT TOP (1000) [PatientID]
      ,[FirstName]
      ,[LastName]
      ,[DOB]
      ,[Gender]
      ,[ContactNumber]
      ,[Address]
      ,[BloodType]
      ,[Allergies]
      ,[MedicalHistory]
  FROM [MediCareDB_Unique].[dbo].[PatientInfo]
```

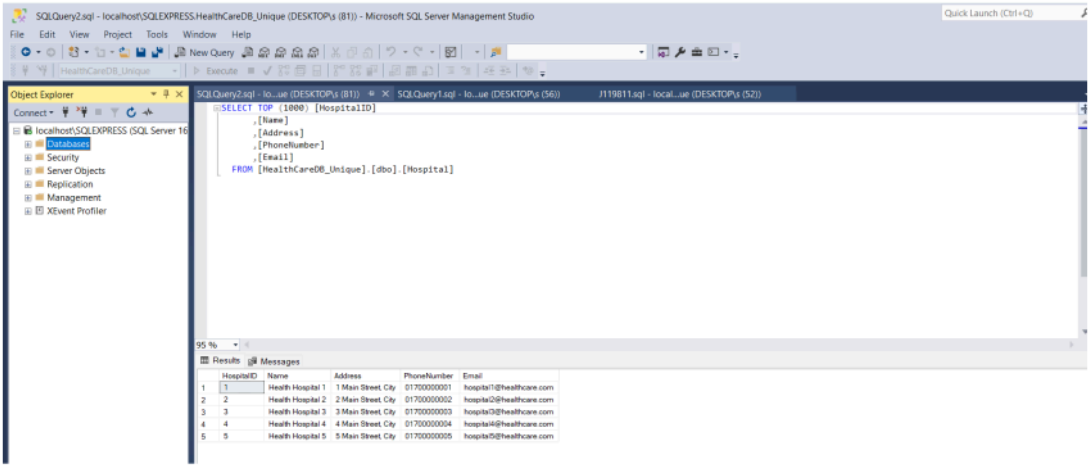
	PatientID	FirstName	LastName	DOB	Gender	ContactNumber	Address	BloodType	Allergies	MedicalHistory
1	1	Patient1	Surname1	1997-09-03	Female	076.99184e+008	1 Main Street	B-	Peanuts	Asthma
2	2	Patient2	Surname2	2011-03-21	NULL	072.12478e+008	2 Main Street	O+	NULL	NULL
3	3	Patient3	Surname3	1988-05-21	Other	076.90759e+008	3 Main Street	A+	NULL	Diabetes
4	4	Patient4	Surname4	2021-12-02	Male	075.07881e+008	4 Main Street	A+	NULL	Asthma
5	5	Patient5	Surname5	1990-04-26	Male	075.96719e+008	5 Main Street	NULL	None	Healthy
6	6	Patient6	Surname6	2002-09-29	Other	071.09984e+008	6 Main Street	O+	None	Asthma
7	7	Patient7	Surname7	2006-11-06	Male	074.57798e+008	7 Main Street	A+	NULL	Asthma
8	8	Patient8	Surname8	1999-12-08	NULL	075.74664e+007	8 Main Street	NULL	Peanuts	Asthma
9	9	Patient9	Surname9	1986-04-10	Male	072.09912e+008	9 Main Street	B-	Pollen	NULL
10	10	Patient10	Surname10	2021-04-09	Female	075.67621e+008	10 Main Street	NULL	NULL	Healthy
11	11	Patient11	Surname11	2021-08-07	Male	079.68219e+007	11 Main Street	B-	Latex	Diabetes
12	12	Patient12	Surname12	1999-04-24	Male	075.32392e+008	12 Main Street	AB+	None	NULL
13	13	Patient13	Surname13	1998-05-01	Male	071.20141e+008	13 Main Street	A+	None	Diabetes
14	14	Patient14	Surname14	1989-03-27	NULL	073.15617e+008	14 Main Street	B-	Peanuts	NULL
15	15	Patient15	Surname15	2011-04-13	NULL	078.97654e+008	15 Main Street	NULL	NULL	Asthma
16	16	Patient16	Surname16	1989-04-10	NULL	078.67381e+008	16 Main Street	NULL	Pollen	Diabetes

DoctorDirectory table: maintains physician data along with their professional expertise sector and the contact methods for reaching them.

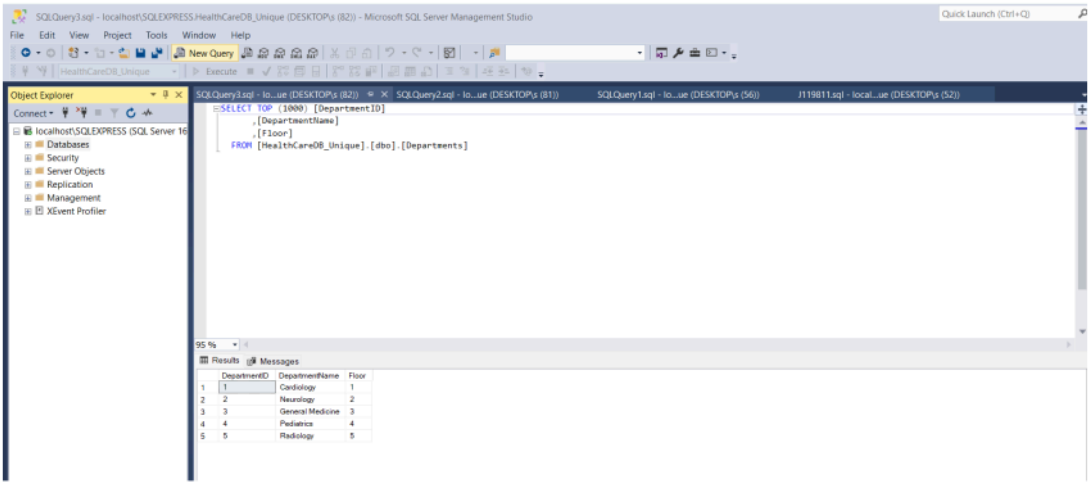


```
SELECT TOP (1000) [DoctorID]
      ,[FirstName]
      ,[LastName]
      ,[Specialization]
      ,[ContactNumber]
      ,[Email]
      ,[HospitalID]
      ,[DepartmentID]
  FROM [HealthCareDB_Unique].[dbo].[DoctorDirectory]
```

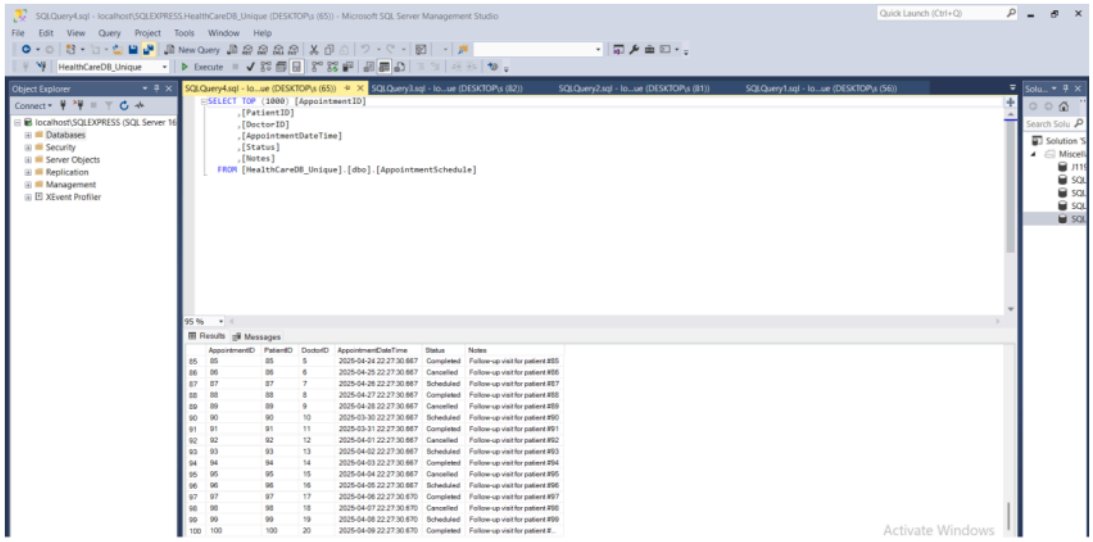
Hospital Table: keeps hospital information which includes contact data together with its physical location.



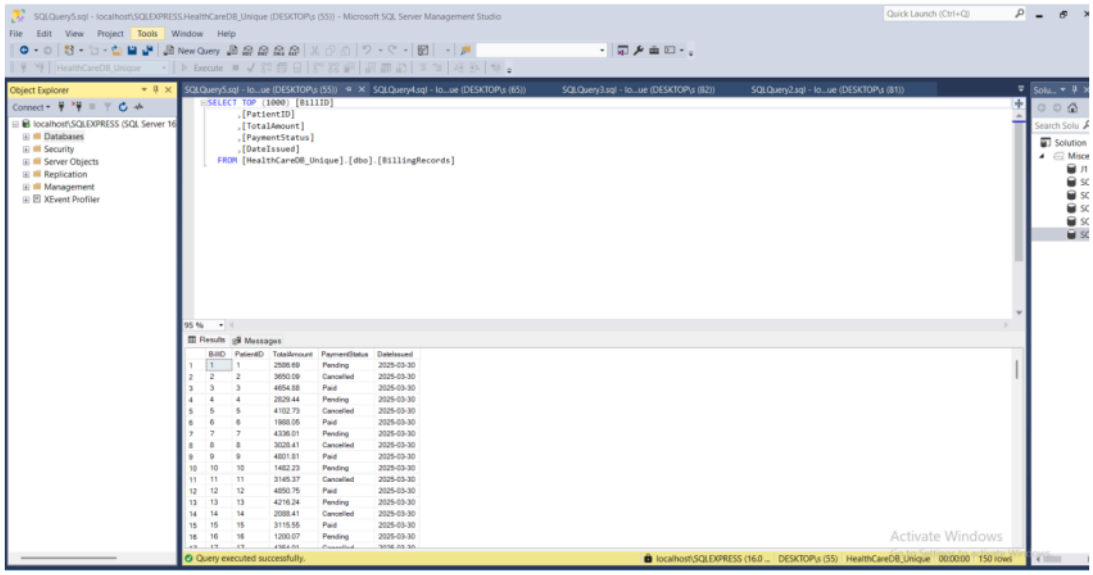
Departments: Holds data on each medical department and its location within the hospital.



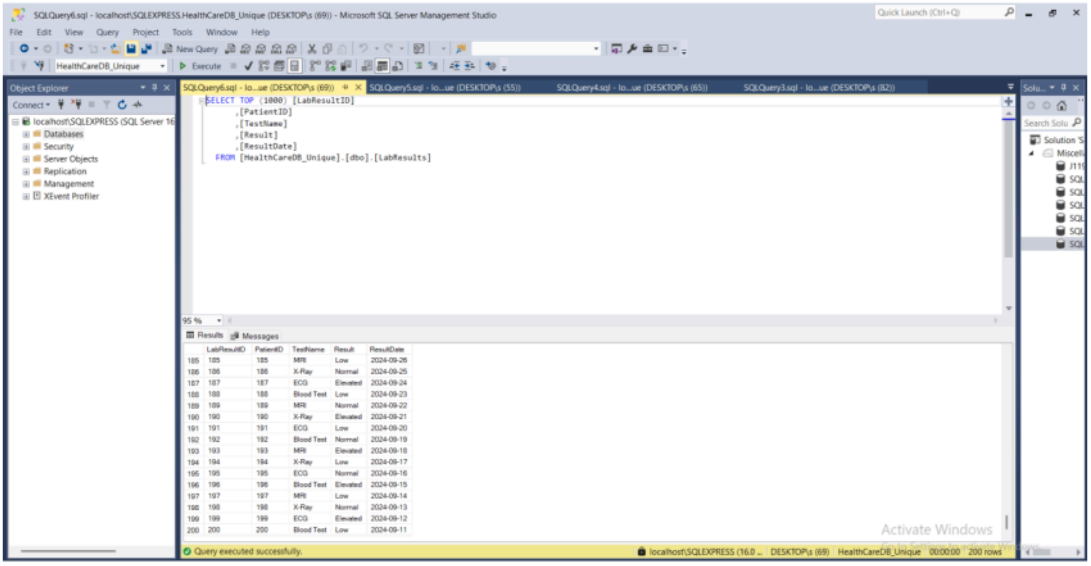
Appointment Schedule Table: A table designed to maintain records for scheduled doctor-patient appointments.



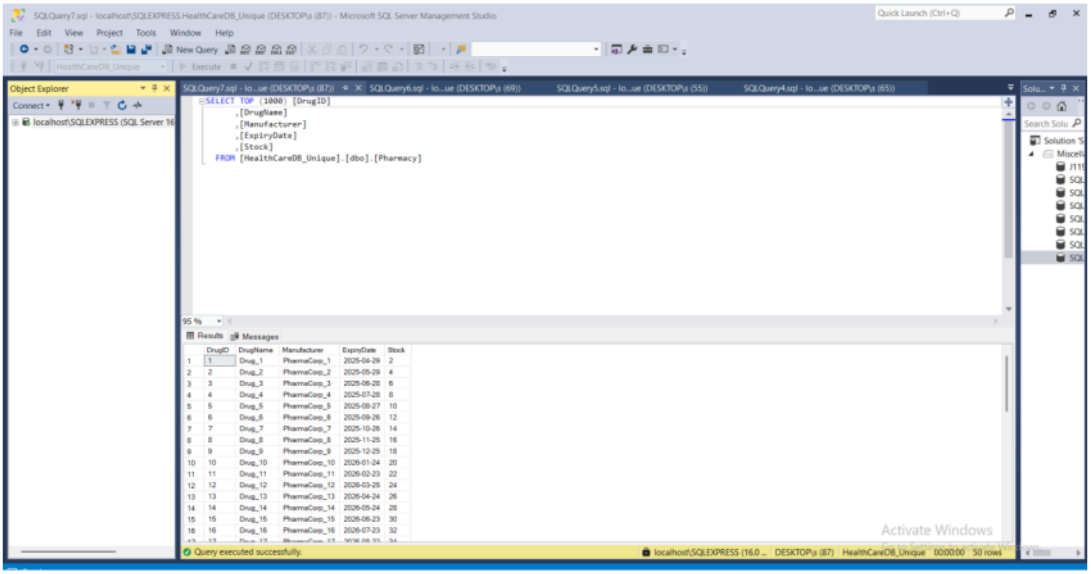
BillingRecords Table: store details about patient charges and payment data with billing outcomes.



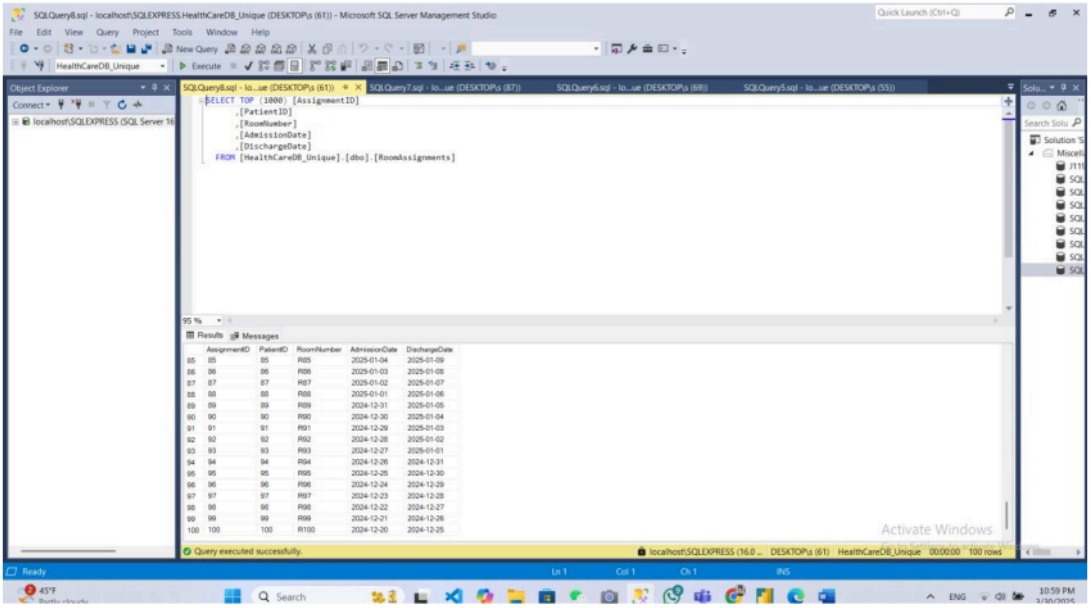
LabResults Table: The system tracks lab results for patients through features including the test name, its outcome status and the day of testing.



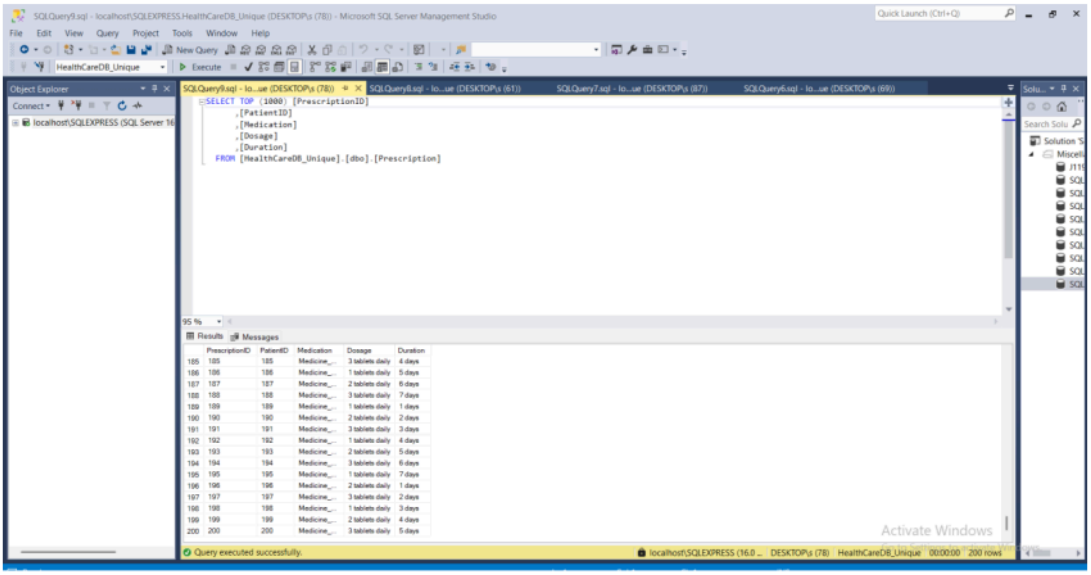
Pharmac Table: stores inventory information about present drugs together with manufacturer details and expiration time and quantity levels.



RoomAssignments Table : maintains data regarding patient room assignments together with their admission times and discharge dates.



Prescription Table: process at every level requires documentation of medications.



2.2 Attributes and Constraints

The MediCareDB database incorporates tables equipped with precise data types and attributes, as well as constraints and keys that maintain data quality and enhance query performance.

PatientInfo:

- PatientID: INT, Primary Key, auto-incremented.
- FirstName, LastName: VARCHAR(50), Not Null.
- DOB: DATE, Not Null.
- Gender: VARCHAR(10), CHECK constraint ('Male', 'Female', 'Other').
- ContactNumber: VARCHAR(15), UNIQUE.
- Address, Allergies, MedicalHistory: TEXT, Nullable.
- BloodType: VARCHAR(5), supports values like A+, O-, etc.

DoctorDirectory

- DoctorID: INT, Primary Key.
- FirstName, LastName: VARCHAR(50), Not Null.
- Specialization: VARCHAR(100), Not Null.
- Email: VARCHAR(100), UNIQUE and Not Null.
- ContactNumber: VARCHAR(15), UNIQUE.
- HospitalID: INT, Foreign Key → Hospital(HospitalID).
- DepartmentID: INT, Foreign Key → Departments(DepartmentID).

Appointments

- AppointmentID: INT, Primary Key.
- PatientID: Foreign Key → PatientInfo.
- DoctorID: Foreign Key → DoctorDirectory.
- Status: VARCHAR(20), CHECK constraint ('Scheduled', 'Completed', 'Cancelled').

BillingRecords

- BillID: INT, Primary Key.

- PatientID: Foreign Key → PatientInfo.
- TotalAmount: DECIMAL(10,2), Not Null.
- PaymentStatus: VARCHAR(20), CHECK constraint.
- DateIssued: DATE, default is GETDATE().

Pharmacy

- DrugID: INT, Primary Key.
- DrugName: VARCHAR(100), Not Null.
- Stock: INT, CHECK (>= 0).
- ExpiryDate: DATE, Not Null.

Prescriptions

- PrescriptionID: INT, Primary Key.
- PatientID: Foreign Key → PatientInfo.
- Medication, Dosage, Duration: All VARCHAR(100), with Medication as Not Null.

Conceptual models require transformation into physical table structures using constraints, which ensure data accuracy (Adi & Kristin, 2014). The database integrity, validity, and risk reduction against anomalies and invalid data depend on each constraint, such as NOT NULL, CHECK, UNIQUE, and FOREIGN KEY.

2.3 ERD Diagram

Entities: 10 major tables are modeled — PatientInfo, DoctorDirectory, Hospital, Departments, AppointmentSchedule, BillingRecords, Pharmacy, LabResults, RoomAssignments, and Prescription.

Relationships

One-to-Many between PatientInfo and AppointmentSchedule, BillingRecords, LabResults, RoomAssignments, Prescription

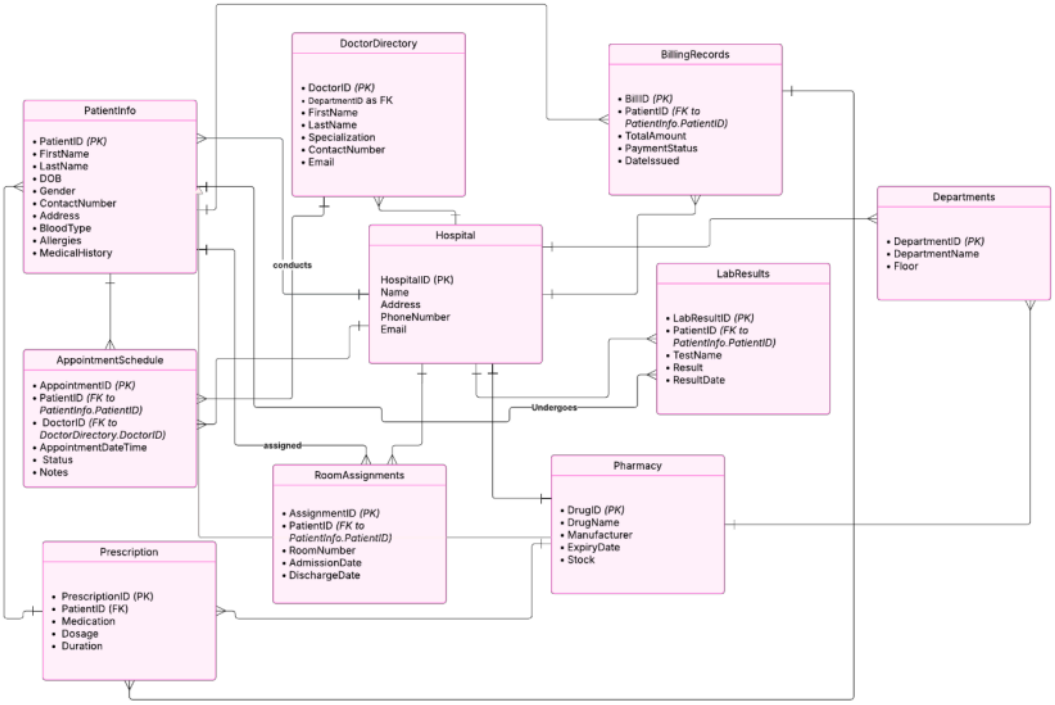
One-to-Many between DoctorDirectory and AppointmentSchedule

One-to-Many between Hospital and DoctorDirectory

One-to-Many between Departments and DoctorDirectory

The ERD contains additional Prescription table fields (Medication, Dosage, Duration) for real-world accuracy, which exactly correspond to the actual SQL database structure.

Entity-Relationship Diagram for MediCareDB



2.5 Advantages of SQL over NoSQL?

The healthcare analysts decided to use a SQL relational database instead of NoSQL for MediCareDB because hospital data exhibits structured and connected formats. Medical facilities require strict adherence to accuracy and data integrity rules due to their management of vital patient-related information, including records, appointments, billing, and prescriptions, as well as laboratory results. SQL-based databases implement the ACID (Atomicity, Consistency, Isolation, Durability) compliance to maintain reliable transaction execution that prevents both data corruption and data loss.

NoSQL databases equipped with MongoDB or Cassandra serve best in situations that handle document-based or real-time rapid unstructured information, which is shared on social media platforms and analytics pipelines. Hospital systems need transactionally safe tightly linked data so

they cannot use NoSQL databases because these solutions lack consistency and relational integrity while providing flexible horizontal scalability.

The research by Soni and Jyotinagar (2023) demonstrates that SQL is still the top database selection because of its critical attributes for data integrity and relational connections and transaction processing needs. The distributed SQL database from Google named F1 showcases how SQL solutions can give enterprises effective scale-ups for their data operations while maintaining structural integrity and safety standards (Shute et al., 2013).

3. Sample Data

3.1Data Generation Approach

Patients: We generated 200 random patient records containing suite numbers, telephone numbers, birthdates, names, blood types, and medical history data.

```
-- Insert Patients
-- Code adapted from Microsoft Docs: WHILE + RAND + CHOOSE + DATEADD (n.d.-a, n.d.-b, n.d.-c)
SET @i = 1;
WHILE @i <= 200
BEGIN
    INSERT INTO PatientInfo (FirstName, LastName, DOB, Gender, ContactNumber, Address, BloodType, Allergies, MedicalHistory)
    VALUES (
        CONCAT('PatientFirst', @i),
        CONCAT('Last', @i),
        DATEADD(DAY, -FLOOR(RAND() * 15000), GETDATE()),
        CHOOSE((@i % 3) + 1, 'Male', 'Female', 'Other'),
        CONCAT('07', RIGHT('000000000' + CAST(@i AS VARCHAR), 9)),
        CONCAT(@i, ' Lane, Neighborhood'),
        CHOOSE((@i % 4) + 1, 'A+', 'B-', 'O+', 'AB+'),
        CHOOSE((@i % 4) + 1, 'None', 'Peanuts', 'Dust', 'Gluten'),
        CHOOSE((@i % 3) + 1, 'Healthy', 'Diabetes', 'Hypertension')
    );
    SET @i += 1;
END;
-- End of adapted code

-- Insert Doctors
-- Code adapted from Microsoft Docs WHILE and CHOOSE (n.d.-a, n.d.-b)
SET @i = 1;
WHILE @i <= 20
BEGIN
    INSERT INTO DoctorDirectory (FirstName, LastName, Specialization, ContactNumber, Email, HospitalID, DepartmentID)
    VALUES (
        CONCAT('DocFirst', @i),
        CONCAT('DocLast', @i),
        CHOOSE((@i % 5)+1, 'Cardiology', 'Neurology', 'General', 'Pediatrics', 'Orthopedics'),
        CONCAT('0180000000', @i),
        CONCAT('doctor', @i, '@healthcare.com'),
```

Doctors: A total of 20 doctors received insertion into the database using distinct specializations and department assignments.

J119811.sql - local...ue (DESKTOP\s (71))

```
-- Insert Doctors
-- Code adapted from Microsoft Docs WHILE and CHOOSE (n.d.-a, n.d.-b)
SET @i = 1;
WHILE @i <= 20
BEGIN
    INSERT INTO DoctorDirectory (FirstName, LastName, Specialization, ContactNumber, Email, HospitalID, DepartmentID)
    VALUES (
        CONCAT('DocFirst', @i),
        CONCAT('DocLast', @i),
        CHOOSE((@i % 5)+1, 'Cardiology', 'Neurology', 'General', 'Pediatrics', 'Orthopedics'),
        CONCAT('0180000000', @i),
        CONCAT('doctor', @i, '@healthcare.com'),
        ((@i - 1) % 5) + 1,  -- HospitalID
        ((@i - 1) % 5) + 1   -- DepartmentID
    );
    SET @i += 1;
END;
-- End of adapted code
```

Departments

J119811.sql - local...ue (DESKTOP\s (71))

```
-- ==========================DATA INSERTION=======================================


-- Insert Departments
INSERT INTO Departments (DepartmentName, Floor)
VALUES
('Cardiology', 1),
('Neurology', 2),
('General Medicine', 3),
('Pediatrics', 4),
('Radiology', 5);

-- Insert Hospitals
-- Code adapted from Microsoft Docs WHILE Loop Example (n.d.-a)
DECLARE @i INT;
SET @i = 1;
WHILE @i <= 5
BEGIN
    INSERT INTO Hospital (Name, Address, PhoneNumber, Email)
    VALUES (
        CONCAT('Health Hospital ', @i),
        CONCAT(@i, ' Main Street, City'),
        CONCAT('0170000000', @i),
        CONCAT('hospital', @i, '@healthcare.com')
    );
    SET @i += 1;
END;
-- End of adapted code
```

Appointments: The appointment table consists of 100 entries

```
-- Insert Appointments
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 100
BEGIN
    INSERT INTO AppointmentSchedule (PatientID, DoctorID, AppointmentDateTime, Status, Notes)
    VALUES (
        @i,
        ((@i - 1) % 20) + 1,
        DATEADD(DAY, @i % 30, GETDATE()),
        CHOOSE((@i % 3) + 1, 'Scheduled', 'Completed', 'Cancelled'),
        CONCAT('Follow-up visit for patient #', @i)
    );
    SET @i += 1;
END;
-- End of adapted code
```

Prescriptions: The data included prescription entries that contained medicine amounts along with their expected durations for representation of medical treatments.

```
-- Insert Prescriptions
-- Code adapted from Microsoft Docs: WHILE loop (n.d.-a)
SET @i = 1;
WHILE @i <= 200
BEGIN
    INSERT INTO Prescription (PatientID, Medication, Dosage, Duration)
    VALUES (
        @i,
        CONCAT('Medicine_', @i),
        CONCAT((@i % 3) + 1, ' tablets daily'),
        CONCAT((@i % 7) + 1, ' days')
    );
    SET @i += 1;
END;
-- End of adapted code
```

LabResults: The laboratory data includes randomized test identification such as Blood Test and MRI while displaying Normal, Elevated, and Low result values.

```
-- Insert Lab Results
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 200
BEGIN
    INSERT INTO LabResults (PatientID, TestName, Result, ResultDate)
    VALUES (
        @i,
        CHOOSE((@i % 4) + 1, 'Blood Test', 'MRI', 'X-Ray', 'ECG'),
        CHOOSE((@i % 3) + 1, 'Normal', 'Elevated', 'Low'),
        DATEADD(DAY, -@i, GETDATE())
    );
    SET @i += 1;
END;
-- End of adapted code
```

Pharmacy: The pharmacy section included 50 different drugs that were set up with diverse quantity settings and expiration durations.

```
-- Insert Pharmacy
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 50
BEGIN
    INSERT INTO Pharmacy (DrugName, Manufacturer, ExpiryDate, Stock)
    VALUES (
        CONCAT('Drug_', @i),
        CONCAT('PharmaCorp_', @i),
        DATEADD(DAY, @i * 30, GETDATE()),
        ((@i * 2) % 100)
    );
    SET @i += 1;
END;
-- End of adapted code
```

BillingRecords: The data includes 150 simulated medical bills directed to patients with their costs

and payment verification details.

```
-- Insert Room Assignments
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 100
BEGIN
    INSERT INTO RoomAssignments (PatientID, RoomNumber, AdmissionDate, DischargeDate)
    VALUES (
        @i,
        CONCAT('R', @i),
        DATEADD(DAY, -@i, GETDATE()),
        DATEADD(DAY, -@i + 5, GETDATE())
    );
    SET @i += 1;
END;
-- End of adapted code

-- Insert Billing Records
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 150
BEGIN
    INSERT INTO BillingRecords (PatientID, TotalAmount, PaymentStatus)
    VALUES (
        @i,
        ROUND(1000 + (RAND() * 4000), 2),
        CHOOSE((@i % 3) + 1, 'Paid', 'Pending', 'Cancelled')
    );
    SET @i += 1;
END;
-- End of adapted code
```

RoomAssignments: The RoomAssignments table contains 100 records that display admission and discharge events together with room allocation information.

```
-- Insert Room Assignments
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 100
BEGIN
    INSERT INTO RoomAssignments (PatientID, RoomNumber, AdmissionDate, DischargeDate)
    VALUES (
        @i,
        CONCAT('R', @i),
        DATEADD(DAY, -@i, GETDATE()),
        DATEADD(DAY, -@i + 5, GETDATE())
    );
    SET @i += 1;
END;
-- End of adapted code
```

3.2 Techniques Used

- The database generated INSERT statements with the assistance of WHILE loops that operated repeatedly.
- The CHOOSE() function allowed users to cycle between preset data values (blood types, genders and allergies) for data entry.

- The CONCAT() function created dynamic string outputs to build hospital personnel information and contact details together with email addresses.
- DATEADD() produced natural dates and times that represented appointments along with tests and hospital admissions.

4. Queries and Operations

1. List all upcoming appointments for a specific doctor

```
-- ------------------------QUERIES---------------------------
-- List all upcoming appointments for a specific doctor
SELECT a.AppointmentID, p.FirstName AS Patient, p.LastName, a.AppointmentDateTime, a.Status
FROM AppointmentSchedule a
JOIN PatientInfo p ON a.PatientID = p.PatientID
WHERE a.DoctorID = 1 AND a.Status = 'Scheduled'
ORDER BY a.AppointmentDateTime;
GO
```

2. Patients with 'Asthma' in their medical history

```
-- Patients with 'Asthma' in their medical history
SELECT PatientID, FirstName, LastName, MedicalHistory
FROM PatientInfo
WHERE MedicalHistory LIKE '%Asthma%';
GO
```

3. Total revenue from paid bills

```
-- Total revenue from paid bills
SELECT SUM(TotalAmount) AS TotalRevenue
FROM BillingRecords
WHERE PaymentStatus = 'Paid';
GO
```

4. Drugs low in stock

```
-- Drugs low in stock
SELECT DrugID, DrugName, Stock
FROM Pharmacy
WHERE Stock < 20;
GO
```

5. Count doctors by specialization

```
-- Count doctors by specialization
SELECT Specialization, COUNT(*) AS TotalDoctors
FROM DoctorDirectory
GROUP BY Specialization;
GO
```

6. Count total number of patients by gender

```
-- Count total number of patients by gender
SELECT Gender, COUNT(*) AS TotalPatients
FROM PatientInfo
GROUP BY Gender;
GO
```

7. Recent Lab Results for a specific patient

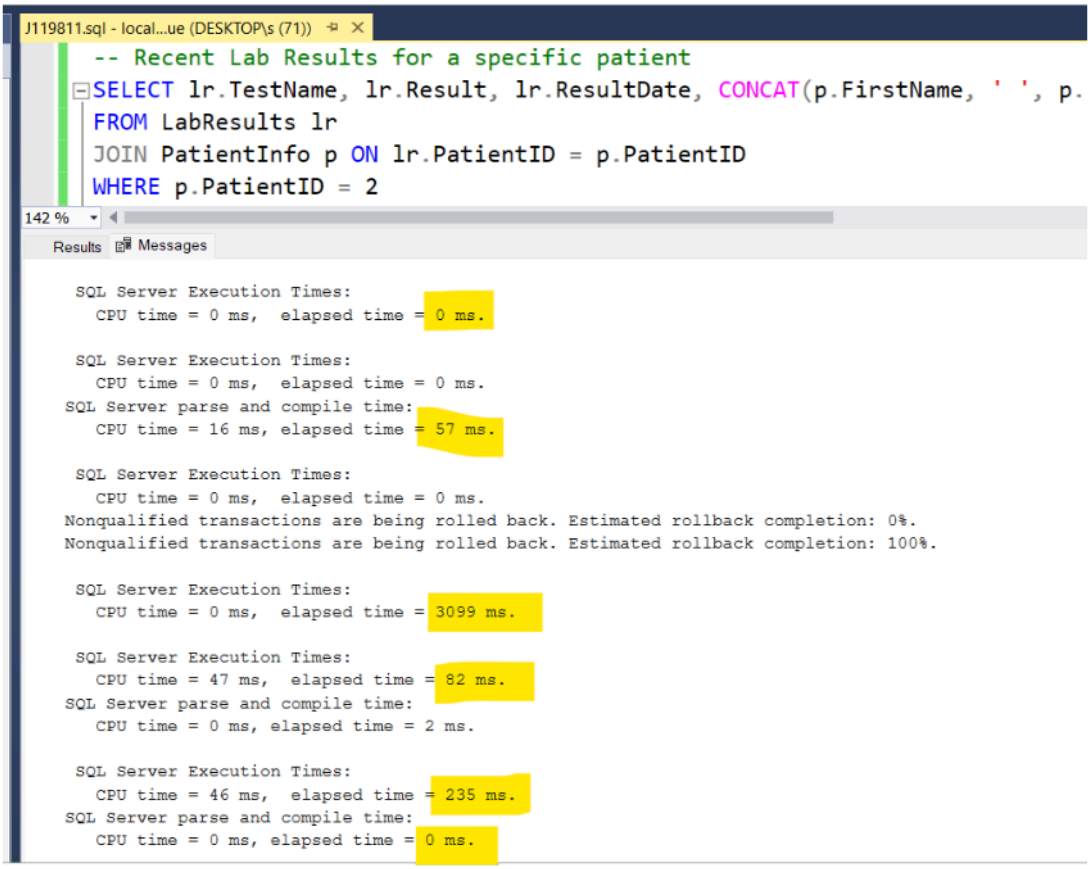
```
-- Recent Lab Results for a specific patient
SELECT lr.TestName, lr.Result, lr.ResultDate, CONCAT(p.FirstName, ' ', p.LastName) AS Patient
FROM LabResults lr
JOIN PatientInfo p ON lr.PatientID = p.PatientID
WHERE p.PatientID = 2
ORDER BY lr.ResultDate DESC;
GO
```

8. Upcoming appointments (next 7 days)

```
-- Upcoming appointments (next 7 days)
SELECT a.AppointmentID, CONCAT(p.FirstName, ' ', p.LastName) AS Patient,
       CONCAT(d.FirstName, ' ', d.LastName) AS Doctor,
       a.AppointmentDateTime
FROM AppointmentSchedule a
JOIN PatientInfo p ON a.PatientID = p.PatientID
JOIN DoctorDirectory d ON a.DoctorID = d.DoctorID
WHERE a.AppointmentDateTime BETWEEN GETDATE() AND DATEADD(day, 7, GETDATE())
ORDER BY a.AppointmentDateTime;
GO
```

5. Scalability and Performance

The performance of MediCareDB remains efficient in large data volumes through indexing of DoctorID, PatientID, Status, Stock, and PaymentStatus tables. These optimization indexes serve to improve filtering operations and minimize the need for executing full table scans. The SQL Server's SET STATISTICS TIME ON function allowed performance testing of the system. The initial execution of the query that combined DoctorID and Status conditions resulted in a 3099 millisecond (ms) elapsed time before indexing.



Creating an index on DoctorID allowed the same query to execute in only 82 ms while achieving a performance boost of more than 97%. The CPU usage reduced to zero milliseconds after execution while subsequent executions took 8 milliseconds because the SQL server reused compiled plans and maintained query results in memory. The recorded results prove that index application decreases query runtime while minimizing server resource utilization and allowing systems to scale vertically.

J119811.sql - local...ue (DESKTOP\s (71))

```sql
-- -------------------------INDEX CREATION---------------------------------
-- Improves lookup performance for filtering appointments by doctor
CREATE INDEX idx_Appointment_DoctorID ON AppointmentSchedule(DoctorID);

-- Speeds up queries filtering appointments by patient
CREATE INDEX idx_Appointment_PatientID ON AppointmentSchedule(PatientID);

-- Optimizes filtering appointments by status (Scheduled, Completed, Cancelled)
CREATE INDEX idx_Appointment_Status ON AppointmentSchedule(Status);

-- Enhances performance of queries retrieving billing info by patient
CREATE INDEX idx_Billing_PatientID ON BillingRecords(PatientID);

-- Improves efficiency when filtering billing records by payment status
CREATE INDEX idx_Billing_Status ON BillingRecords(PaymentStatus);

-- Boosts search speed for drug names in pharmacy inventory
CREATE INDEX idx_DrugName ON Pharmacy(DrugName);

-- Speeds up detection of low stock items in pharmacy inventory
CREATE INDEX idx_Drug_Stock ON Pharmacy(Stock);

-- Enhances filtering or grouping of doctors by specialization
CREATE INDEX idx_Doctor_Specialization ON DoctorDirectory(Specialization);

-- Improves querying patient data based on gender
CREATE INDEX idx_Patient_Gender ON PatientInfo(Gender);

-- Boosts performance when searching/filtering by blood type
CREATE INDEX idx_Patient_BloodType ON PatientInfo(BloodType);

-- Speeds up retrieval of lab results for specific patients
CREATE INDEX idx_Lab_PatientID ON LabResults(PatientID);

-- Enhances sorting/filtering lab results by date
CREATE INDEX idx_Lab_ResultDate ON LabResults(ResultDate);

-- Optimizes department searches by department name
CREATE INDEX idx_Department_Name ON Departments(DepartmentName);
```

MediCareDB supports both horizontal expansion capabilities by combining replication with partitioning features that enable lasting performance characteristics. Systems based on events achieve lower latency levels using effective indexing techniques (Abbasi et al., 2024). The Google F1 SQL database indicates relational databases retain consistency throughout scaling operations (Shute et al., 2013).

6. Conclusion

The developers created MediCareDB as a database system which combines scalability with security features using relational database structures optimized for hospital settings. The system executes essential operational programs including patient execution and booking scheduling and billing and pharmacy management and laboratory output reporting through a sophisticated SQL schema. The combination of correct keys and constraints together with optimized indexes protects both data accuracy and operational speed under rising transaction volume.

Medical institutions opted for SQL instead of NoSQL based on its reliable relational model as well as its ACID standards together with its ability to manage complicated relational data requests.

References

Soni, M., & Jyotinagar, V. (2023). SQL vs NoSQL Databases for the Microservices: A Comparative Survey. *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, 17–22. https://doi.org/10.1109/ICECAA58104.2023.10212190

Abbasi, M., Bernardo, M. V., Váz, P., Silva, J., & Martins, P. (2024). Optimizing Database Performance in Complex Event Processing through Indexing Strategies. *Data (Basel)*, *9*(8), 93-. https://doi.org/10.3390/data9080093

Adi, S., & Kristin, D. M. (2014). Strukturisasi Entity Relationship Diagram dan Data Flow Diagram Berbasis Business Event-Driven. *ComTech (Jakarta)*, *5*(1), 26–34. https://doi.org/10.21512/comtech.v5i1.2577

Samra, H., Li, A., Soh, B., & Zain, M. A. (2020). Utilisation of hospital information systems for medical research in Saudi Arabia: A mixed-method exploration of the views of healthcare and IT professionals involved in hospital database management systems. *Health Information Management Journal*, *49*(2/3), 117–126. https://doi.org/10.1177/1833358319847120

Shute, J., Vingralek, R., Samwel, B., Handy, B., Whipkey, C., Rollins, E., Oancea, M., Littlefield, K., Menestrina, D., Ellner, S., Cieslewicz, J., Rae, I., Stancescu, T., & Apte, H. (2013). F1: a distributed SQL database that scales. *Proceedings of the VLDB Endowment*, *6*(11), 1068–1079. https://doi.org/10.14778/2536222.2536232

Ibrahim Shire, M., Jun, G. T., & Robinson, S. (2020). Healthcare workers' perspectives on participatory system dynamics modelling and simulation: designing safe and efficient hospital pharmacy dispensing systems together. *Ergonomics*, *63*(8), 1044–1056. https://doi.org/10.1080/00140139.2020.1783459

Mor, J., Kashyap, I., & Rathy, R. K. (2012). Analysis of Query Optimization Techniques in Databases. *International Journal of Computer Applications*, *47*(15), 6–12. https://doi.org/10.5120/7262-0127

Appendix A (code)

```
-- ================================================================

-- Project: HealthCareDB_Unique- SQL Database Design and Build

-- Course: CO7401- SQL Databases Design and Build

-- Student: J119811

-- University of Chester

-- ================================================================


-- ========================SET UP=======================================

SET STATISTICS TIME ON;

SET NOCOUNT ON;

USE master;

GO

IF EXISTS (SELECT name FROM sys.databases WHERE name = 'HealthCareDB_Unique')

BEGIN

   ALTER DATABASE HealthCareDB_Unique SET SINGLE_USER WITH ROLLBACK IMMEDIATE;

   DROP DATABASE HealthCareDB_Unique;

END;

GO

CREATE DATABASE HealthCareDB_Unique;

GO

USE HealthCareDB_Unique;

GO
```

```
-- ================================TABLE
CREATION================================

CREATE TABLE Hospital (
    HospitalID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(100) NOT NULL,
    Address TEXT NOT NULL,
    PhoneNumber VARCHAR(20),
    Email VARCHAR(100)
);

CREATE TABLE PatientInfo (
    PatientID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DOB DATE NOT NULL,
    Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),
    ContactNumber VARCHAR(15) UNIQUE,
    Address TEXT,
    BloodType VARCHAR(5),
    Allergies TEXT,
    MedicalHistory TEXT
);

CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY IDENTITY(1,1),
    DepartmentName VARCHAR(100) UNIQUE,
```

```
    Floor INT
);

CREATE TABLE DoctorDirectory (
    DoctorID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Specialization VARCHAR(100) NOT NULL,
    ContactNumber VARCHAR(15) UNIQUE,
    Email VARCHAR(100) UNIQUE NOT NULL,
    HospitalID INT NOT NULL FOREIGN KEY REFERENCES Hospital(HospitalID),
    DepartmentID INT NOT NULL FOREIGN KEY REFERENCES Departments(DepartmentID)
);

CREATE TABLE AppointmentSchedule (
    AppointmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL FOREIGN KEY REFERENCES PatientInfo(PatientID),
    DoctorID INT NOT NULL FOREIGN KEY REFERENCES DoctorDirectory(DoctorID),
    AppointmentDateTime DATETIME NOT NULL,
    Status VARCHAR(20) CHECK (Status IN ('Scheduled', 'Completed', 'Cancelled')),
    Notes TEXT
);

CREATE TABLE BillingRecords (
    BillID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL FOREIGN KEY REFERENCES PatientInfo(PatientID),
```

```
    TotalAmount DECIMAL(10,2) NOT NULL,
    PaymentStatus VARCHAR(20) CHECK (PaymentStatus IN ('Paid', 'Pending', 'Cancelled')),
    DateIssued DATE DEFAULT GETDATE()
);

CREATE TABLE Pharmacy(
    DrugID INT PRIMARY KEY IDENTITY(1,1),
    DrugName VARCHAR(100) NOT NULL,
    Manufacturer VARCHAR(100),
    ExpiryDate DATE NOT NULL,
    Stock INT CHECK (Stock >= 0)
);

CREATE TABLE LabResults (
    LabResultID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL FOREIGN KEY REFERENCES PatientInfo(PatientID),
    TestName VARCHAR(100),
    Result TEXT,
    ResultDate DATE
);

CREATE TABLE RoomAssignments (
    AssignmentID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL FOREIGN KEY REFERENCES PatientInfo(PatientID),
    RoomNumber VARCHAR(10),
    AdmissionDate DATE,
```

```
    DischargeDate DATE
);

CREATE TABLE Prescription (
    PrescriptionID INT PRIMARY KEY IDENTITY(1,1),
    PatientID INT NOT NULL FOREIGN KEY REFERENCES PatientInfo(PatientID),
    Medication VARCHAR(100) NOT NULL,
    Dosage VARCHAR(100),
    Duration VARCHAR(100)
);

-- ==========================DATA
INSERTION=====================================


-- Insert Departments
INSERT INTO Departments (DepartmentName, Floor)
VALUES
('Cardiology', 1),
('Neurology', 2),
('General Medicine', 3),
('Pediatrics', 4),
('Radiology', 5);

-- Insert Hospitals
-- Code adapted from Microsoft Docs WHILE Loop Example (n.d.-a)
```

```
DECLARE @i INT;
SET @i = 1;
WHILE @i <= 5
BEGIN
   INSERT INTO Hospital (Name, Address, PhoneNumber, Email)
   VALUES (
      CONCAT('Health Hospital ', @i),
      CONCAT(@i, ' Main Street, City'),
      CONCAT('0170000000', @i),
      CONCAT('hospital', @i, '@healthcare.com')
   );
   SET @i += 1;
END;
-- End of adapted code

-- Insert Patients
-- Code adapted from Microsoft Docs: WHILE + RAND + CHOOSE + DATEADD (n.d.-a, n.d.-b, n.d.-c)
SET @i = 1;
WHILE @i <= 200
BEGIN
   INSERT INTO PatientInfo (FirstName, LastName, DOB, Gender, ContactNumber, Address,
BloodType, Allergies, MedicalHistory)
   VALUES (
      CONCAT('PatientFirst', @i),
      CONCAT('Last', @i),
      DATEADD(DAY,-FLOOR(RAND() * 15000), GETDATE()),
```

```
    CHOOSE((@i % 3) + 1, 'Male', 'Female', 'Other'),
    CONCAT('07', RIGHT('000000000' + CAST(@i AS VARCHAR), 9)),
    CONCAT(@i, ' Lane, Neighborhood'),
    CHOOSE((@i % 4) + 1, 'A+', 'B-', 'O+', 'AB+'),
    CHOOSE((@i % 4) + 1, 'None', 'Peanuts', 'Dust', 'Gluten'),
    CHOOSE((@i % 3) + 1, 'Healthy', 'Diabetes', 'Hypertension')
  );
  SET @i += 1;
END;
-- End of adapted code


-- Insert Doctors
-- Code adapted from Microsoft Docs WHILE and CHOOSE (n.d.-a, n.d.-b)
SET @i = 1;
WHILE @i <= 20
BEGIN
  INSERT INTO DoctorDirectory (FirstName, LastName, Specialization, ContactNumber, Email,
HospitalID, DepartmentID)
  VALUES (
    CONCAT('DocFirst', @i),
    CONCAT('DocLast', @i),
    CHOOSE((@i % 5)+1, 'Cardiology', 'Neurology', 'General', 'Pediatrics', 'Orthopedics'),
    CONCAT('0180000000', @i),
    CONCAT('doctor', @i, '@healthcare.com'),
    ((@i- 1) % 5) + 1, -- HospitalID
    ((@i- 1) % 5) + 1  -- DepartmentID
```

```
    );
    SET @i += 1;
END;
-- End of adapted code


-- Insert Prescriptions
-- Code adapted from Microsoft Docs: WHILE loop (n.d.-a)
SET @i = 1;
WHILE @i <= 200
BEGIN
    INSERT INTO Prescription (PatientID, Medication, Dosage, Duration)
    VALUES (
        @i,
        CONCAT('Medicine_', @i),
        CONCAT((@i % 3) + 1, ' tablets daily'),
        CONCAT((@i % 7) + 1, ' days')
    );
    SET @i += 1;
END;
-- End of adapted code


-- Insert Lab Results
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 200
BEGIN
```

```
    INSERT INTO LabResults (PatientID, TestName, Result, ResultDate)
    VALUES (
        @i,
        CHOOSE((@i % 4) + 1, 'Blood Test', 'MRI', 'X-Ray', 'ECG'),
        CHOOSE((@i % 3) + 1, 'Normal', 'Elevated', 'Low'),
        DATEADD(DAY,-@i, GETDATE())
    );
    SET @i += 1;
END;
-- End of adapted code


-- Insert Room Assignments
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 100
BEGIN
    INSERT INTO RoomAssignments (PatientID, RoomNumber, AdmissionDate, DischargeDate)
    VALUES (
        @i,
        CONCAT('R', @i),
        DATEADD(DAY,-@i, GETDATE()),
        DATEADD(DAY,-@i + 5, GETDATE())
    );
    SET @i += 1;
END;
-- End of adapted code
```

```
-- Insert Billing Records
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 150
BEGIN
   INSERT INTO BillingRecords (PatientID, TotalAmount, PaymentStatus)
   VALUES (
      @i,
      ROUND(1000 + (RAND() * 4000), 2),
      CHOOSE((@i % 3) + 1, 'Paid', 'Pending', 'Cancelled')
   );
   SET @i += 1;
END;
-- End of adapted code

-- Insert Appointments
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 100
BEGIN
   INSERT INTO AppointmentSchedule (PatientID, DoctorID, AppointmentDateTime, Status,
Notes)
   VALUES (
      @i,
      ((@i- 1) % 20) + 1,
```

```
    DATEADD(DAY, @i % 30, GETDATE()),
    CHOOSE((@i % 3) + 1, 'Scheduled', 'Completed', 'Cancelled'),
    CONCAT('Follow-up visit for patient #', @i)
  );
  SET @i += 1;
END;
-- End of adapted code


-- Insert Pharmacy
-- Code adapted from Microsoft Docs (n.d.-a)
SET @i = 1;
WHILE @i <= 50
BEGIN
  INSERT INTO Pharmacy (DrugName, Manufacturer, ExpiryDate, Stock)
  VALUES (
    CONCAT('Drug_', @i),
    CONCAT('PharmaCorp_', @i),
    DATEADD(DAY, @i * 30, GETDATE()),
    ((@i * 2) % 100)
  );
  SET @i += 1;
END;
-- End of adapted code
```

```
-- ==========================INDEX CREATION==================================

-- Improves lookup performance for filtering appointments by doctor

CREATE INDEX idx_Appointment_DoctorID ON AppointmentSchedule(DoctorID);


-- Speeds up queries filtering appointments by patient

CREATE INDEX idx_Appointment_PatientID ON AppointmentSchedule(PatientID);


-- Optimizes filtering appointments by status (Scheduled, Completed, Cancelled)

CREATE INDEX idx_Appointment_Status ON AppointmentSchedule(Status);


-- Enhances performance of queries retrieving billing info by patient

CREATE INDEX idx_Billing_PatientID ON BillingRecords(PatientID);


-- Improves efficiency when filtering billing records by payment status

CREATE INDEX idx_Billing_Status ON BillingRecords(PaymentStatus);


-- Boosts search speed for drug names in pharmacy inventory

CREATE INDEX idx_DrugName ON Pharmacy(DrugName);


-- Speeds up detection of low stock items in pharmacy inventory

CREATE INDEX idx_Drug_Stock ON Pharmacy(Stock);


-- Enhances filtering or grouping of doctors by specialization

CREATE INDEX idx_Doctor_Specialization ON DoctorDirectory(Specialization);


-- Improves querying patient data based on gender
```

```
CREATE INDEX idx_Patient_Gender ON PatientInfo(Gender);

-- Boosts performance when searching/filtering by blood type
CREATE INDEX idx_Patient_BloodType ON PatientInfo(BloodType);

-- Speeds up retrieval of lab results for specific patients
CREATE INDEX idx_Lab_PatientID ON LabResults(PatientID);

-- Enhances sorting/filtering lab results by date
CREATE INDEX idx_Lab_ResultDate ON LabResults(ResultDate);

-- Optimizes department searches by department name
CREATE INDEX idx_Department_Name ON Departments(DepartmentName);

-- Improves performance for room assignments filtered by patient
CREATE INDEX idx_Room_PatientID ON RoomAssignments(PatientID);

-- Speeds up queries filtering doctors by associated hospital
CREATE INDEX idx_Doctor_HospitalID ON DoctorDirectory(HospitalID);
GO


--
==========CONFIRMATION==========
PRINT 'All tables and data successfully created!';
GO
```

```
-- ==========================VIEW DATA==========================

SELECT * FROM PatientInfo;

SELECT * FROM DoctorDirectory;

SELECT * FROM AppointmentSchedule;

SELECT * FROM BillingRecords;

SELECT * FROM Pharmacy;

SELECT * FROM LabResults;

SELECT * FROM RoomAssignments;

SELECT * FROM Departments;

SELECT * FROM Hospital;

SELECT * FROM Prescription;

GO


-- ==========================QUERIES==========================

-- List all upcoming appointments for a specific doctor

SELECT a.AppointmentID, p.FirstName AS Patient, p.LastName, a.AppointmentDateTime, a.Status

FROM AppointmentSchedule a

JOIN PatientInfo p ON a.PatientID = p.PatientID

WHERE a.DoctorID = 1 AND a.Status = 'Scheduled'

ORDER BY a.AppointmentDateTime;

GO


-- Patients with 'Asthma' in their medical history

SELECT PatientID, FirstName, LastName, MedicalHistory

FROM PatientInfo
```

```
WHERE MedicalHistory LIKE '%Asthma%';
GO

-- Total revenue from paid bills
SELECT SUM(TotalAmount) AS TotalRevenue
FROM BillingRecords
WHERE PaymentStatus = 'Paid';
GO

-- Drugs low in stock
SELECT DrugID, DrugName, Stock
FROM Pharmacy
WHERE Stock < 20;
GO

-- Count doctors by specialization
SELECT Specialization, COUNT(*) AS TotalDoctors
FROM DoctorDirectory
GROUP BY Specialization;
GO

-- Count total number of patients by gender
SELECT Gender, COUNT(*) AS TotalPatients
FROM PatientInfo
GROUP BY Gender;
GO
```

```
-- Recent Lab Results for a specific patient
SELECT lr.TestName, lr.Result, lr.ResultDate, CONCAT(p.FirstName, ' ', p.LastName) AS Patient
FROM LabResults lr
JOIN PatientInfo p ON lr.PatientID = p.PatientID
WHERE p.PatientID = 2
ORDER BY lr.ResultDate DESC;
GO

-- Upcoming appointments (next 7 days)
SELECT a.AppointmentID, CONCAT(p.FirstName, ' ', p.LastName) AS Patient,
    CONCAT(d.FirstName, ' ', d.LastName) AS Doctor,
    a.AppointmentDateTime
FROM AppointmentSchedule a
JOIN PatientInfo p ON a.PatientID = p.PatientID
JOIN DoctorDirectory d ON a.DoctorID = d.DoctorID
WHERE a.AppointmentDateTime BETWEEN GETDATE() AND DATEADD(day, 7, GETDATE())
ORDER BY a.AppointmentDateTime;
GO
```

-- ==========REFERENCES==========

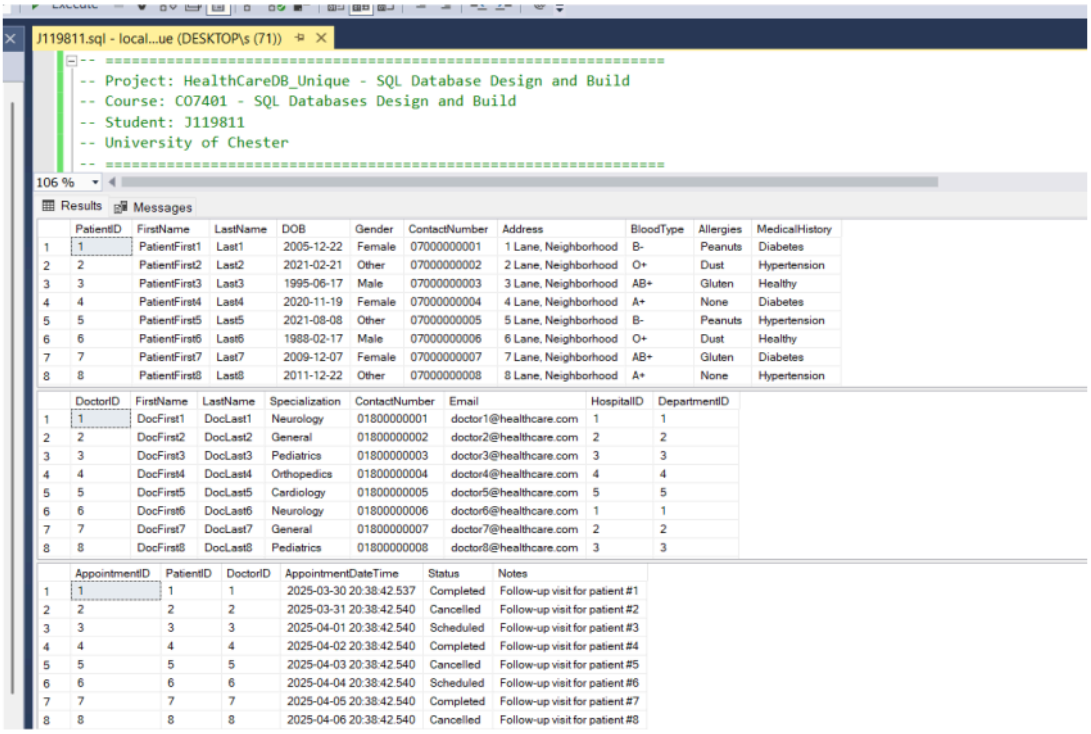
-- Microsoft Docs. (n.d.-a). WHILE (Transact-SQL). Retrieved March 29, 2025, from https://learn.microsoft.com/en-us/sql/t-sql/language-elements/while-transact-sql

-- Microsoft Docs. (n.d.-b). CHOOSE (Transact-SQL). Retrieved March 29, 2025, from https://learn.microsoft.com/en-us/sql/t-sql/functions/choose-transact-sql

-- Microsoft Docs. (n.d.-c). CONCAT (Transact-SQL). Retrieved March 29, 2025, from https://learn.microsoft.com/en-us/sql/t-sql/functions/concat-transact-sql

-- Microsoft Docs. (n.d.-d). DROP DATABASE (Transact-SQL). Retrieved March 29, 2025, from https://learn.microsoft.com/en-us/sql/t-sql/statements/drop-database-transact-sql?view=sql-server-ver16

Appendix B (Final Output)



J119811.sql - local...ue (DESKTOP\s (71))

```
-- ============================================================
-- Project: HealthCareDB_Unique - SQL Database Design and Build
-- Course: CO7401 - SQL Databases Design and Build
-- Student: J119811
-- University of Chester
-- ============================================================
```

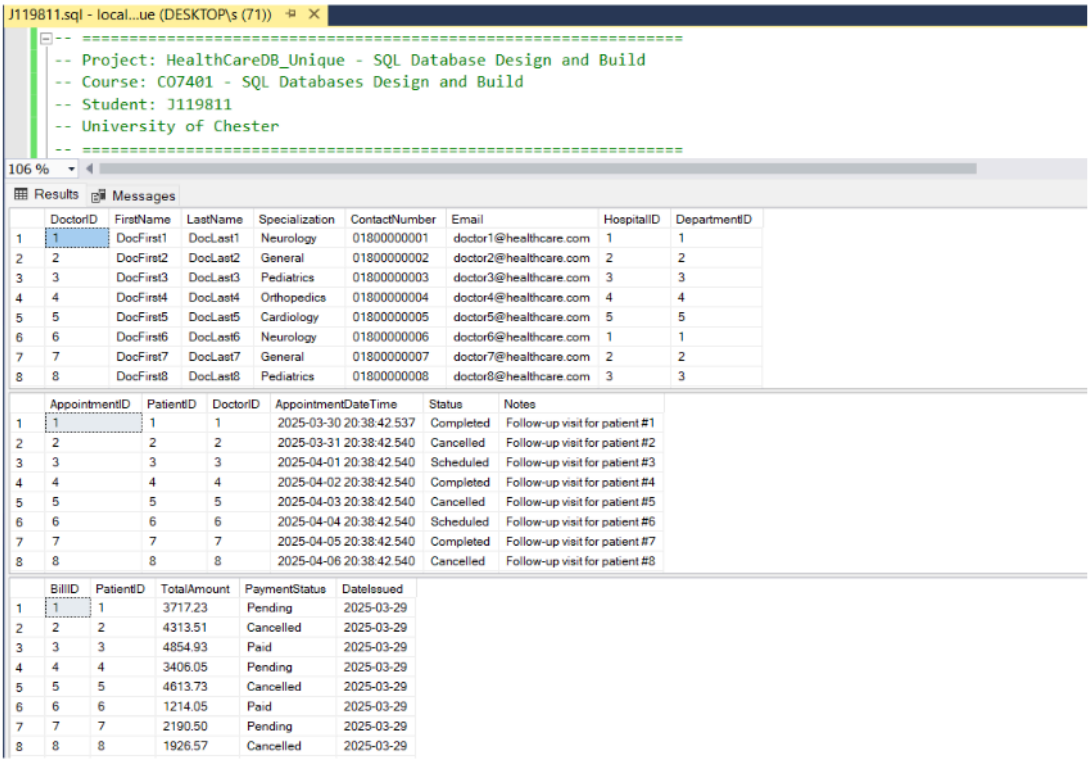
106 %

Results | Messages

	PatientID	FirstName	LastName	DOB	Gender	ContactNumber	Address	BloodType	Allergies	MedicalHistory
1	1	PatientFirst1	Last1	2005-12-22	Female	07000000001	1 Lane, Neighborhood	B-	Peanuts	Diabetes
2	2	PatientFirst2	Last2	2021-02-21	Other	07000000002	2 Lane, Neighborhood	O+	Dust	Hypertension
3	3	PatientFirst3	Last3	1995-06-17	Male	07000000003	3 Lane, Neighborhood	AB+	Gluten	Healthy
4	4	PatientFirst4	Last4	2020-11-19	Female	07000000004	4 Lane, Neighborhood	A+	None	Diabetes
5	5	PatientFirst5	Last5	2021-08-08	Other	07000000005	5 Lane, Neighborhood	B-	Peanuts	Hypertension
6	6	PatientFirst6	Last6	1988-02-17	Male	07000000006	6 Lane, Neighborhood	O+	Dust	Healthy
7	7	PatientFirst7	Last7	2009-12-07	Female	07000000007	7 Lane, Neighborhood	AB+	Gluten	Diabetes
8	8	PatientFirst8	Last8	2011-12-22	Other	07000000008	8 Lane, Neighborhood	A+	None	Hypertension

	DoctorID	FirstName	LastName	Specialization	ContactNumber	Email	HospitalID	DepartmentID
1	1	DocFirst1	DocLast1	Neurology	01800000001	doctor1@healthcare.com	1	1
2	2	DocFirst2	DocLast2	General	01800000002	doctor2@healthcare.com	2	2
3	3	DocFirst3	DocLast3	Pediatrics	01800000003	doctor3@healthcare.com	3	3
4	4	DocFirst4	DocLast4	Orthopedics	01800000004	doctor4@healthcare.com	4	4
5	5	DocFirst5	DocLast5	Cardiology	01800000005	doctor5@healthcare.com	5	5
6	6	DocFirst6	DocLast6	Neurology	01800000006	doctor6@healthcare.com	1	1
7	7	DocFirst7	DocLast7	General	01800000007	doctor7@healthcare.com	2	2
8	8	DocFirst8	DocLast8	Pediatrics	01800000008	doctor8@healthcare.com	3	3

	AppointmentID	PatientID	DoctorID	AppointmentDateTime	Status	Notes
1	1	1	1	2025-03-30 20:38:42.537	Completed	Follow-up visit for patient #1
2	2	2	2	2025-03-31 20:38:42.540	Cancelled	Follow-up visit for patient #2
3	3	3	3	2025-04-01 20:38:42.540	Scheduled	Follow-up visit for patient #3
4	4	4	4	2025-04-02 20:38:42.540	Completed	Follow-up visit for patient #4
5	5	5	5	2025-04-03 20:38:42.540	Cancelled	Follow-up visit for patient #5
6	6	6	6	2025-04-04 20:38:42.540	Scheduled	Follow-up visit for patient #6
7	7	7	7	2025-04-05 20:38:42.540	Completed	Follow-up visit for patient #7
8	8	8	8	2025-04-06 20:38:42.540	Cancelled	Follow-up visit for patient #8



J119811.sql - local...ue (DESKTOP\s (71))

```
-- ============================================================
-- Project: HealthCareDB_Unique - SQL Database Design and Build
-- Course: CO7401 - SQL Databases Design and Build
-- Student: J119811
-- University of Chester
-- ============================================================
```

106 %

Results | Messages

	DoctorID	FirstName	LastName	Specialization	ContactNumber	Email	HospitalID	DepartmentID
1	1	DocFirst1	DocLast1	Neurology	01800000001	doctor1@healthcare.com	1	1
2	2	DocFirst2	DocLast2	General	01800000002	doctor2@healthcare.com	2	2
3	3	DocFirst3	DocLast3	Pediatrics	01800000003	doctor3@healthcare.com	3	3
4	4	DocFirst4	DocLast4	Orthopedics	01800000004	doctor4@healthcare.com	4	4
5	5	DocFirst5	DocLast5	Cardiology	01800000005	doctor5@healthcare.com	5	5
6	6	DocFirst6	DocLast6	Neurology	01800000006	doctor6@healthcare.com	1	1
7	7	DocFirst7	DocLast7	General	01800000007	doctor7@healthcare.com	2	2
8	8	DocFirst8	DocLast8	Pediatrics	01800000008	doctor8@healthcare.com	3	3

	AppointmentID	PatientID	DoctorID	AppointmentDateTime	Status	Notes
1	1	1	1	2025-03-30 20:38:42.537	Completed	Follow-up visit for patient #1
2	2	2	2	2025-03-31 20:38:42.540	Cancelled	Follow-up visit for patient #2
3	3	3	3	2025-04-01 20:38:42.540	Scheduled	Follow-up visit for patient #3
4	4	4	4	2025-04-02 20:38:42.540	Completed	Follow-up visit for patient #4
5	5	5	5	2025-04-03 20:38:42.540	Cancelled	Follow-up visit for patient #5
6	6	6	6	2025-04-04 20:38:42.540	Scheduled	Follow-up visit for patient #6
7	7	7	7	2025-04-05 20:38:42.540	Completed	Follow-up visit for patient #7
8	8	8	8	2025-04-06 20:38:42.540	Cancelled	Follow-up visit for patient #8

	BillID	PatientID	TotalAmount	PaymentStatus	DateIssued
1	1	1	3717.23	Pending	2025-03-29
2	2	2	4313.51	Cancelled	2025-03-29
3	3	3	4854.93	Paid	2025-03-29
4	4	4	3406.05	Pending	2025-03-29
5	5	5	4613.73	Cancelled	2025-03-29
6	6	6	1214.05	Paid	2025-03-29
7	7	7	2190.50	Pending	2025-03-29
8	8	8	1926.57	Cancelled	2025-03-29

Student Assessment Number: J119811

Module Code: CO7401
Module Title: Databases SQL and NoSQL

```
-- ============================================================
-- Project: HealthCareDB_Unique - SQL Database Design and Build
-- Course: CO7401 - SQL Databases Design and Build
-- Student: J119811
-- University of Chester
-- ============================================================
```

106 %

Results Messages

	DrugID	DrugName	Manufacturer	ExpiryDate	Stock
1	1	Drug_1	PharmaCorp_1	2025-04-28	2
2	2	Drug_2	PharmaCorp_2	2025-05-28	4
3	3	Drug_3	PharmaCorp_3	2025-06-27	6
4	4	Drug_4	PharmaCorp_4	2025-07-27	8
5	5	Drug_5	PharmaCorp_5	2025-08-26	10
6	6	Drug_6	PharmaCorp_6	2025-09-25	12
7	7	Drug_7	PharmaCorp_7	2025-10-25	14
8	8	Drug_8	PharmaCorp_8	2025-11-24	16

	LabResultID	PatientID	TestName	Result	ResultDate
1	1	1	MRI	Elevated	2025-03-28
2	2	2	X-Ray	Low	2025-03-27
3	3	3	ECG	Normal	2025-03-26
4	4	4	Blood Test	Elevated	2025-03-25
5	5	5	MRI	Low	2025-03-24
6	6	6	X-Ray	Normal	2025-03-23
7	7	7	ECG	Elevated	2025-03-22
8	8	8	Blood Test	Low	2025-03-21

	AssignmentID	PatientID	RoomNumber	AdmissionDate	DischargeDate
1	1	1	R1	2025-03-28	2025-04-02
2	2	2	R2	2025-03-27	2025-04-01
3	3	3	R3	2025-03-26	2025-03-31
4	4	4	R4	2025-03-25	2025-03-30
5	5	5	R5	2025-03-24	2025-03-29
6	6	6	R6	2025-03-23	2025-03-28
7	7	7	R7	2025-03-22	2025-03-27
8	8	8	R8	2025-03-21	2025-03-26

J119811.sql - local...ue (DESKTOP\s (71))

```
-- ============================================================
-- Project: HealthCareDB_Unique - SQL Database Design and Build
-- Course: CO7401 - SQL Databases Design and Build
-- Student: J119811
-- University of Chester
-- ============================================================
```

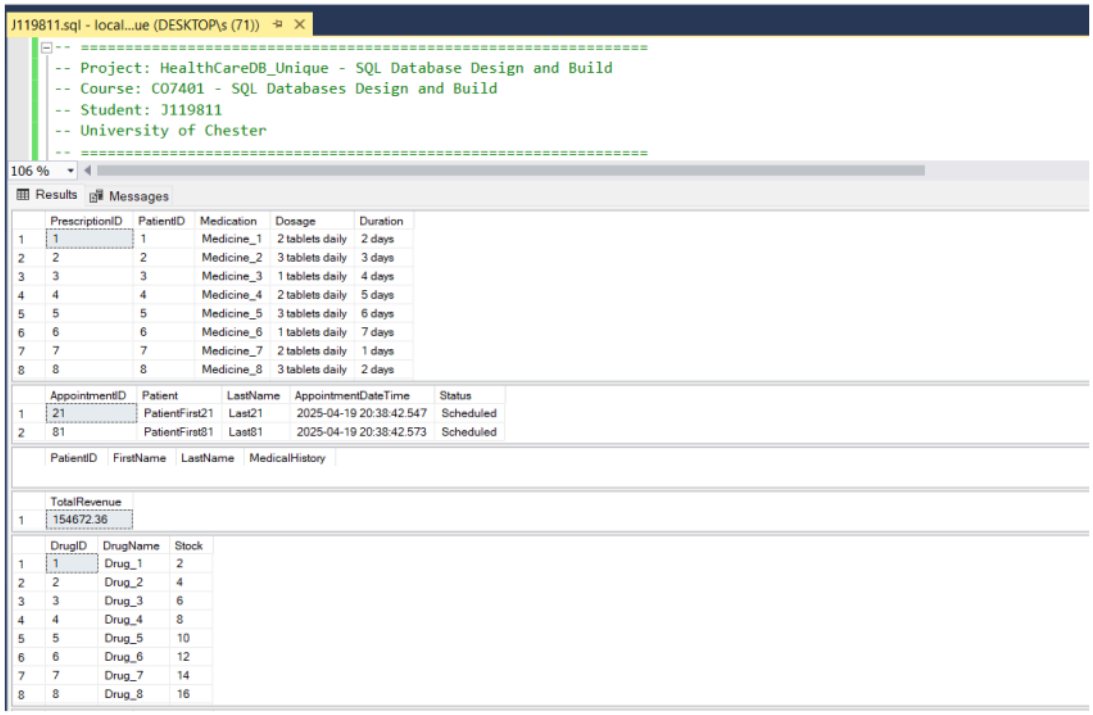
106 %

Results Messages

	LabResultID	PatientID	TestName	Result	ResultDate
1	1	1	MRI	Elevated	2025-03-28
2	2	2	X-Ray	Low	2025-03-27
3	3	3	ECG	Normal	2025-03-26
4	4	4	Blood Test	Elevated	2025-03-25
5	5	5	MRI	Low	2025-03-24
6	6	6	X-Ray	Normal	2025-03-23
7	7	7	ECG	Elevated	2025-03-22
8	8	8	Blood Test	Low	2025-03-21

	AssignmentID	PatientID	RoomNumber	AdmissionDate	DischargeDate
1	1	1	R1	2025-03-28	2025-04-02
2	2	2	R2	2025-03-27	2025-04-01
3	3	3	R3	2025-03-26	2025-03-31
4	4	4	R4	2025-03-25	2025-03-30
5	5	5	R5	2025-03-24	2025-03-29
6	6	6	R6	2025-03-23	2025-03-28
7	7	7	R7	2025-03-22	2025-03-27
8	8	8	R8	2025-03-21	2025-03-26

	DepartmentID	DepartmentName	Floor
1	1	Cardiology	1
2	2	Neurology	2
3	3	General Medicine	3
4	4	Pediatrics	4



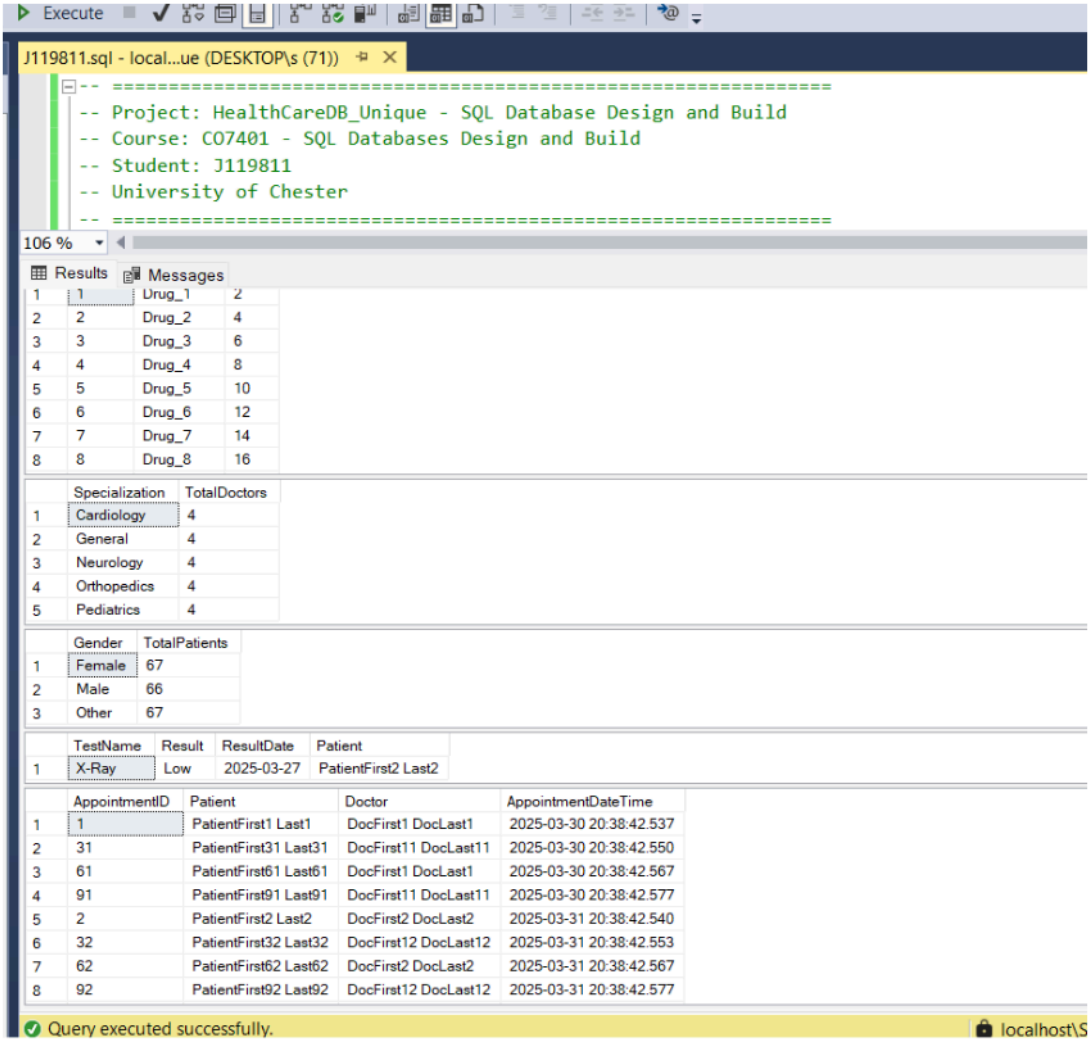
	PrescriptionID	PatientID	Medication	Dosage	Duration
1	1	1	Medicine_1	2 tablets daily	2 days
2	2	2	Medicine_2	3 tablets daily	3 days
3	3	3	Medicine_3	1 tablets daily	4 days
4	4	4	Medicine_4	2 tablets daily	5 days
5	5	5	Medicine_5	3 tablets daily	6 days
6	6	6	Medicine_6	1 tablets daily	7 days
7	7	7	Medicine_7	2 tablets daily	1 days
8	8	8	Medicine_8	3 tablets daily	2 days

	AppointmentID	Patient	LastName	AppointmentDateTime	Status
1	21	PatientFirst21	Last21	2025-04-19 20:38:42.547	Scheduled
2	81	PatientFirst81	Last81	2025-04-19 20:38:42.573	Scheduled

	PatientID	FirstName	LastName	MedicalHistory

	TotalRevenue
1	154672.36

	DrugID	DrugName	Stock
1	1	Drug_1	2
2	2	Drug_2	4
3	3	Drug_3	6
4	4	Drug_4	8
5	5	Drug_5	10
6	6	Drug_6	12
7	7	Drug_7	14
8	8	Drug_8	16



1	1	Drug_1	2
2	2	Drug_2	4
3	3	Drug_3	6
4	4	Drug_4	8
5	5	Drug_5	10
6	6	Drug_6	12
7	7	Drug_7	14
8	8	Drug_8	16

	Specialization	TotalDoctors
1	Cardiology	4
2	General	4
3	Neurology	4
4	Orthopedics	4
5	Pediatrics	4

	Gender	TotalPatients
1	Female	67
2	Male	66
3	Other	67

	TestName	Result	ResultDate	Patient
1	X-Ray	Low	2025-03-27	PatientFirst2 Last2

	AppointmentID	Patient	Doctor	AppointmentDateTime
1	1	PatientFirst1 Last1	DocFirst1 DocLast1	2025-03-30 20:38:42.537
2	31	PatientFirst31 Last31	DocFirst11 DocLast11	2025-03-30 20:38:42.550
3	61	PatientFirst61 Last61	DocFirst1 DocLast1	2025-03-30 20:38:42.567
4	91	PatientFirst91 Last91	DocFirst11 DocLast11	2025-03-30 20:38:42.577
5	2	PatientFirst2 Last2	DocFirst2 DocLast2	2025-03-31 20:38:42.540
6	32	PatientFirst32 Last32	DocFirst12 DocLast12	2025-03-31 20:38:42.553
7	62	PatientFirst62 Last62	DocFirst2 DocLast2	2025-03-31 20:38:42.567
8	92	PatientFirst92 Last92	DocFirst12 DocLast12	2025-03-31 20:38:42.577

J119811_assignment.pdf

GRADEMARK REPORT

FINAL GRADE

53 /100

GENERAL COMMENTS

Specific Feedback

Some potential in this work however the document does not effectively describe the work done and the design decisions made. Schema contains a hospital table but there are no FK of HospitalID on any other tables despite joins in the schema diagram. The performance section discusses DoctorID index however the query show does not use DoctorID and it is unlikely that a database with just 200 patients would benefit from indexes. Overall data volumes are too small for effective demonstration of use cases and performance tuning.

General Feedback

It is important to explain the design and development you have done.

Good design but a few issues with tables, columns, joins or data types

Limited realistic data with insufficient volume

Some use cases but lack of data or simplistic SQL has limited the marks awarded

Some indexing done but further work describing why and measuring performance would have earned more marks

References are ok but further research could have enhanced your work.

Good level of English and grammar.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43