[redacted]

Fall 2024

Nuzhat Faizah

[redacted]

List of deliverables included in this report:

1. Entity-Relationship Diagram
2. Database
3. Data Definition Language
4. Data Manipulation Language
5. A set of SQL queries with the results
6. Normalization & Denormalization
7. Application of trigger

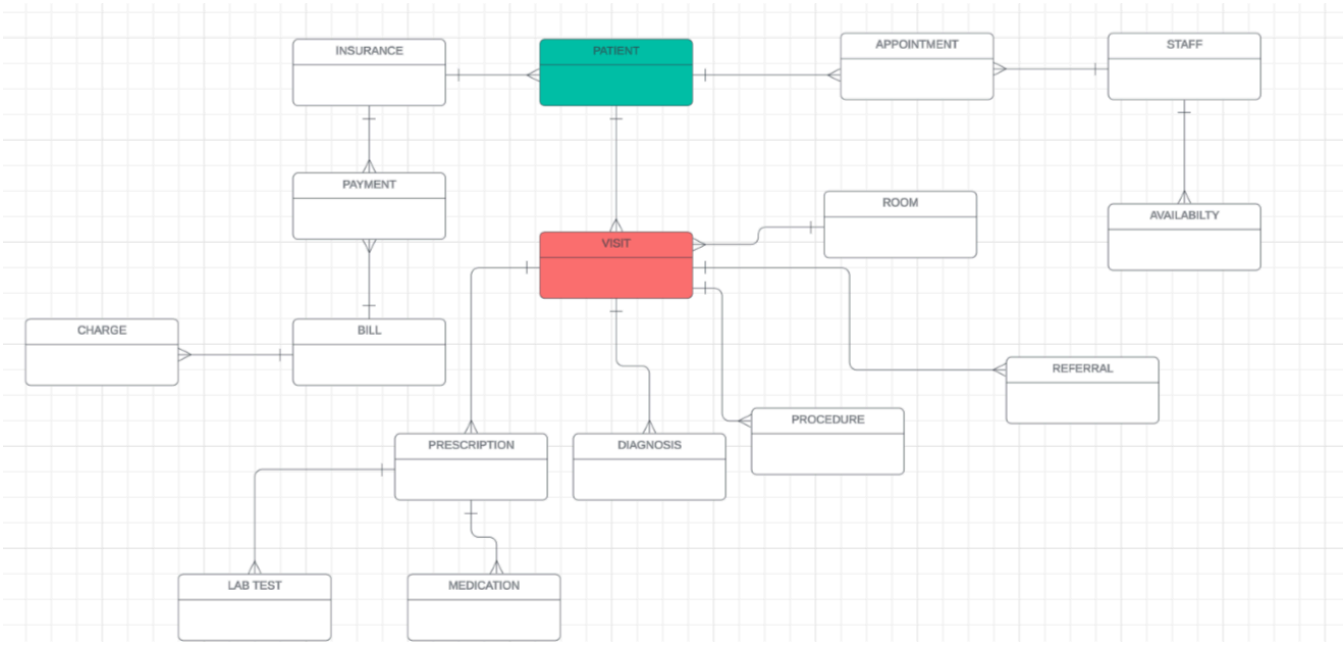


Figure 1: The Entity_Relationship Diagram for the hypothetical scenario covered in this project.

I have already created the new database in Microsoft SQL Server Management Studio. The demonstrations for DDL, DLM, SQL queries are provided below:

```
--- Creating the table for INSURANCE ---
CREATE TABLE Insurance (
    InsuranceID INT PRIMARY KEY,                -- Primary Key for Insurance
    ProviderName NVARCHAR(100) NOT NULL,        -- Insurance Provider Name
    PolicyNumber NVARCHAR(50) NOT NULL,         -- Insurance Policy Number
    CoverageDetails NVARCHAR(255)               -- Coverage Details for the Insurance Policy
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for PATIENT ---
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,                  -- Primary Key
    FirstName NVARCHAR(50) NOT NULL,            -- First Name of the Patient
    LastName NVARCHAR(50) NOT NULL,             -- Last Name of the Patient
    DateOfBirth DATE NOT NULL,                  -- Patient's Date of Birth
    Gender NVARCHAR(10) NOT NULL,               -- Gender of the Patient
    PhoneNumber NVARCHAR(20) NOT NULL,          -- Patient's Phone Number
    Email NVARCHAR(255),                        -- Patient's Email Address
    Address NVARCHAR(255),                      -- Patient's Address
    InsuranceID INT,                            -- Foreign Key referencing Insurance
    FOREIGN KEY (InsuranceID) REFERENCES Insurance(InsuranceID)
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for VISIT ---
CREATE TABLE Visit (
    VisitID INTEGER PRIMARY KEY,                -- Primary Key for Visit
    PatientID INTEGER NOT NULL,                 -- Foreign Key referencing Patient
    VisitDate DATE NOT NULL,                    -- Date of the visit
    Reason NVARCHAR(500),                       -- Reason for the visit
    ReferralID INTEGER,                         -- Foreign Key referencing Referral
    RoomID INTEGER,                             -- Foreign Key referencing Room
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (ReferralID) REFERENCES Referral(ReferralID),
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)
);
```

```sql
--- Creating the table for ROOM ---
CREATE TABLE Room (
    RoomID INTEGER PRIMARY KEY,                 -- Primary Key
    RoomNumber NVARCHAR(50) NOT NULL UNIQUE,    -- Unique Room Number
    Capacity INTEGER                            -- Capacity of the room
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for REFERRAL ---
CREATE TABLE Referral (
    ReferralID INTEGER PRIMARY KEY,
    ReferredBy TEXT NOT NULL,
    ReferredTo TEXT NOT NULL
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for APPOINTMENT ---
CREATE TABLE Appointment (
    AppointmentID INTEGER PRIMARY KEY,       -- Primary Key for Appointment
    PatientID INTEGER NOT NULL,              -- Foreign Key referencing Patient
    VisitID INTEGER,                         -- Foreign Key referencing Visit (nullable)
    AppointmentDate DATE NOT NULL,           -- Date of the appointment
    StaffID INTEGER,                         -- Foreign Key referencing Staff
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (VisitID) REFERENCES Visit(VisitID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for STAFF ---
CREATE TABLE Staff (
    StaffID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Role TEXT NOT NULL
);
```

```sql
--- Creating the table for AVAILABILITY ---
CREATE TABLE Availability (
    AvailabilityID INTEGER PRIMARY KEY,
    StaffID INTEGER NOT NULL,
    AvailableDate DATE NOT NULL,
    AvailableTime TIME NOT NULL,
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for PAYMENT ---
CREATE TABLE Payment (
    PaymentID INTEGER PRIMARY KEY,           -- Primary Key for Payment
    VisitID INTEGER NOT NULL,                -- Foreign Key referencing Visit
    AmountPaid DECIMAL(10, 2) NOT NULL,      -- Amount paid
    PaymentDate DATE NOT NULL,               -- Date of payment
    FOREIGN KEY (VisitID) REFERENCES Visit(VisitID)
);


--------------------------- SECTION BREAK -----------------------------

--- Creating the table for BILL ---
CREATE TABLE Bill (
    BillingID INT PRIMARY KEY,                   -- Primary Key for Bill
    PatientID INT NOT NULL,                      -- Foreign Key referencing Patient
    TotalAmount DECIMAL(10, 2) NOT NULL,         -- Total bill amount
    AmountPaid DECIMAL(10, 2) NOT NULL,          -- Amount already paid
    BalanceDue AS (TotalAmount - AmountPaid),    -- Calculated column for balance due
    PaymentDate DATE NOT NULL,                   -- Date of payment
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) -- Foreign Key constraint
);

--------------------------- SECTION BREAK -----------------------------

--- Creating the table for CHARGE ---
CREATE TABLE Charge (
    ChargeID INTEGER PRIMARY KEY,            -- Primary Key for Charge
    BillID INTEGER NOT NULL,                 -- Foreign Key referencing Bill
    ChargeDescription NVARCHAR(255) NOT NULL, -- Description of the charge
    ChargeAmount DECIMAL(10, 2) NOT NULL,    -- Amount of the charge
    FOREIGN KEY (BillID) REFERENCES Bill(BillingID)
```

```sql
--- Creating the table for PRESCRIPTION ---
CREATE TABLE Prescription (
    PrescriptionID INT PRIMARY KEY,          -- Primary Key for Prescription
    AppointmentID INT NOT NULL,              -- Foreign Key referencing Appointment
    MedicationName NVARCHAR(100) NOT NULL,   -- Medication name
    Dosage NVARCHAR(50) NOT NULL,            -- Dosage information
    Instructions NVARCHAR(255),              -- Instructions for the prescription
    DispenseDate DATE NOT NULL,              -- Dispense date
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
);


---------------------------- SECTION BREAK -----------------------------

--- Creating the table for MEDICATION ---
CREATE TABLE Medication (
    MedicationID INTEGER PRIMARY KEY,
    MedicationName TEXT NOT NULL,
    Dosage TEXT NOT NULL
);

---------------------------- SECTION BREAK -----------------------------

--- Creating the table for DIAGNOSIS ---
CREATE TABLE Diagnosis (
    DiagnosisID INTEGER PRIMARY KEY,         -- Primary Key for Diagnosis
    VisitID INTEGER NOT NULL,                -- Foreign Key referencing Visit
    DiagnosisDetails NVARCHAR(255) NOT NULL, -- Details of the diagnosis
    FOREIGN KEY (VisitID) REFERENCES Visit(VisitID)
);


---------------------------- SECTION BREAK -----------------------------

--- Creating the table for PROCEDURE ---
CREATE TABLE MedicalProcedure (
    ProcedureID INTEGER PRIMARY KEY,         -- Primary Key for Procedure
    VisitID INTEGER NOT NULL,                -- Foreign Key referencing Visit
    ProcedureDescription NVARCHAR(255) NOT NULL, -- Description of the procedure
    ProcedureCost DECIMAL(10, 2) NOT NULL,   -- Cost of the procedure
    FOREIGN KEY (VisitID) REFERENCES Visit(VisitID)
);
```

```sql
--- Creating the table for PROCEDURE ---
CREATE TABLE MedicalProcedure (
    ProcedureID INTEGER PRIMARY KEY,          -- Primary Key for Procedure
    VisitID INTEGER NOT NULL,                 -- Foreign Key referencing Visit
    ProcedureDescription NVARCHAR(255) NOT NULL, -- Description of the procedure
    ProcedureCost DECIMAL(10, 2) NOT NULL,    -- Cost of the procedure
    FOREIGN KEY (VisitID) REFERENCES Visit(VisitID)
);


--------------------------- SECTION BREAK -----------------------------

--- Creating the table for LAB TEST ---
CREATE TABLE LabTest (
    LabTestID INT PRIMARY KEY,                -- Primary Key for Lab Test
    TestType NVARCHAR(50) NOT NULL,           -- Type of the test (e.g., Blood Test)
    TestName NVARCHAR(100) NOT NULL,          -- Name of the test
    TestDate DATE NOT NULL,                   -- Date of the test
    TestResult NVARCHAR(255),                 -- Test result
    AppointmentID INT NOT NULL,               -- Foreign Key referencing Appointment
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
);

--------------------------- SECTION BREAK -----------------------------

--- Checking if all the tables are created ---

SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE';

SELECT name AS ObjectName, type_desc AS ObjectType
FROM sys.objects
WHERE type IN ('U');
```

	TABLE_NAME
1	MedicalProcedure
2	LabTest
3	PatientLog
4	Insurance
5	Patient
6	Referral
7	Staff
8	Availability
9	Bill
10	Medication
11	Room
12	Visit
13	Appointment
14	Payment
15	Charge
16	Prescription
17	Diagnosis

	ObjectName	ObjectType
1	MedicalProcedure	USER_TABLE
2	LabTest	USER_TABLE
3	PatientLog	USER_TABLE
4	Insurance	USER_TABLE
5	Patient	USER_TABLE
6	Referral	USER_TABLE
7	Staff	USER_TABLE
8	Availability	USER_TABLE
9	Bill	USER_TABLE
10	Medication	USER_TABLE
11	Room	USER_TABLE
12	Visit	USER_TABLE
13	Appointment	USER_TABLE
14	Payment	USER_TABLE
15	Charge	USER_TABLE
16	Prescription	USER_TABLE
17	Diagnosis	USER_TABLE

```sql
--- Ensuring foreign key relationships are correctly established ---

SELECT
    fk.name AS ForeignKeyName,
    tp.name AS ParentTable,
    tr.name AS ReferencedTable
FROM
    sys.foreign_keys AS fk
INNER JOIN
    sys.tables AS tp ON fk.parent_object_id = tp.object_id
INNER JOIN
    sys.tables AS tr ON fk.referenced_object_id = tr.object_id;
```

	ForeignKeyName	ParentTable	ReferencedTable
1	FK__Patient__Insuran__403A8C7D	Patient	Insurance
2	FK__Bill__PatientID__5441852A	Bill	Patient
3	FK__Visit__PatientID__6EF57B66	Visit	Patient
4	FK__Appointme__Patie__73BA30...	Appointm...	Patient
5	FK__Visit__Referral__6FE99F9F	Visit	Referral
6	FK__Availabil__Staff__4E88ABD4	Availability	Staff
7	FK__Appointme__Staff__75A278F5	Appointm...	Staff
8	FK__Charge__BillID__7B5B524B	Charge	Bill
9	FK__Visit__RoomID__70DDC3D8	Visit	Room
10	FK__Diagnosis__Visit__01142BA1	Diagnosis	Visit
11	FK__MedicalPr__Visit__03F0984C	MedicalPr...	Visit
12	FK__Appointme__Visit__74AE54BC	Appointm...	Visit
13	FK__Payment__VisitID__787EE5A0	Payment	Visit
14	FK__LabTest__Appoint__06CD04...	LabTest	Appointment
15	FK__Prescript__Appoi__7E37BEF6	Prescription	Appointment

```sql
--- Visualizing all the tables ---

SELECT * FROM Insurance;
SELECT * FROM Patient;
SELECT * FROM Visit;
SELECT * FROM Room;
SELECT * FROM Referral;
SELECT * FROM Appointment;
SELECT * FROM Staff;
SELECT * FROM Availability;
SELECT * FROM Payment;
SELECT * FROM Bill;
SELECT * FROM Charge;
SELECT * FROM Prescription;
SELECT * FROM Medication;
SELECT * FROM Diagnosis;
SELECT * FROM MedicalProcedure;
SELECT * FROM LabTest;
```

```sql
--- Populating all the tables with dummy data ---

INSERT INTO Insurance (InsuranceID, ProviderName, PolicyNumber, CoverageDetails)
VALUES
(1, 'Blue Cross', 'BC12345', 'Full Coverage'),
(2, 'United Health', 'UH67890', 'Partial Coverage'),
(3, 'Aetna', 'AT11223', 'Dental Coverage'),
(4, 'Cigna', 'CG44556', 'Vision Coverage');

INSERT INTO Patient (PatientID, FirstName, LastName, DateOfBirth, Gender, PhoneNumber, Email, Address, InsuranceID)
VALUES
(1, 'John', 'Doe', '1990-01-01', 'Male', '+1-123-456-7890', 'john.doe@example.com', '123 Main St', 1),
(2, 'Jane', 'Smith', '1985-05-12', 'Female', '+1-987-654-3210', 'jane.smith@example.com', '456 Elm St', 2),
(3, 'Alice', 'Johnson', '2000-08-20', 'Female', '+1-456-789-1230', 'alice.johnson@example.com', '789 Maple Ave', 3),
(4, 'Bob', 'Brown', '1975-03-15', 'Male', '+1-321-654-0987', 'bob.brown@example.com', '101 Pine Rd', 4);

INSERT INTO Room (RoomID, RoomNumber, Capacity)
VALUES
(1, 'Room-101', 2),
(2, 'Room-102', 4),
(3, 'Room-103', 1),
(4, 'Room-104', 3);

INSERT INTO Referral (ReferralID, ReferredBy, ReferredTo)
VALUES
(1, 'Dr. Smith', 'Dr. Wilson'),
(2, 'Dr. Clark', 'Dr. Johnson'),
(3, 'Dr. Brown', 'Dr. Lee'),
(4, 'Dr. Davis', 'Dr. Miller');

INSERT INTO Visit (VisitID, PatientID, VisitDate, Reason, ReferralID, RoomID)
VALUES
(1, 1, '2024-01-15', 'Annual Checkup', 1, 1),
(2, 2, '2024-02-20', 'Follow-up', 2, 2),
(3, 3, '2024-03-25', 'Dental Cleaning', 3, 3),
(4, 4, '2024-04-10', 'Eye Test', 4, 4);

INSERT INTO Staff (StaffID, Name, Role)
VALUES
(1, 'Dr. Sarah Miller', 'Physician'),
(2, 'Dr. James Wilson', 'Dentist'),
(3, 'Dr. Emily Clark', 'Optometrist'),
(4, 'Nurse Anna Johnson', 'Nurse');
```

```sql
INSERT INTO Appointment (AppointmentID, PatientID, VisitID, AppointmentDate, StaffID)
VALUES
(1, 1, 1, '2024-01-15', 1),
(2, 2, 2, '2024-02-20', 2),
(3, 3, 3, '2024-03-25', 3),
(4, 4, 4, '2024-04-10', 4);

INSERT INTO Availability (AvailabilityID, StaffID, AvailableDate, AvailableTime)
VALUES
(1, 1, '2024-01-10', '09:00:00'),
(2, 2, '2024-01-12', '10:00:00'),
(3, 3, '2024-01-15', '14:00:00'),
(4, 4, '2024-01-20', '16:00:00');

INSERT INTO Payment (PaymentID, VisitID, AmountPaid, PaymentDate)
VALUES
(1, 1, 150.00, '2024-01-16'),
(2, 2, 200.00, '2024-02-21'),
(3, 3, 250.00, '2024-03-26'),
(4, 4, 300.00, '2024-04-11');

INSERT INTO Bill (BillingID, PatientID, TotalAmount, AmountPaid, PaymentDate)
VALUES
(1, 1, 200.00, 150.00, '2024-01-16'),
(2, 2, 300.00, 200.00, '2024-02-21'),
(3, 3, 250.00, 250.00, '2024-03-26'),
(4, 4, 400.00, 300.00, '2024-04-11');

INSERT INTO Charge (ChargeID, BillID, ChargeDescription, ChargeAmount)
VALUES
(1, 1, 'Consultation Fee', 50.00),
(2, 2, 'Dental Cleaning', 100.00),
(3, 3, 'Eye Test', 50.00),
(4, 4, 'Therapy Session', 100.00);

INSERT INTO Prescription (PrescriptionID, AppointmentID, MedicationName, Dosage, Instructions, DispenseDate)
VALUES
(1, 1, 'Amoxicillin', '500mg', 'Take twice daily', '2024-01-16'),
(2, 2, 'Ibuprofen', '200mg', 'Take as needed', '2024-02-21'),
(3, 3, 'Acetaminophen', '500mg', 'Take every 6 hours', '2024-03-26'),
(4, 4, 'Latanoprost', '1 drop', 'Apply nightly', '2024-04-11');
```

```
INSERT INTO Medication (MedicationID, MedicationName, Dosage)
VALUES
(1, 'Amoxicillin', '500mg'),
(2, 'Ibuprofen', '200mg'),
(3, 'Acetaminophen', '500mg'),
(4, 'Latanoprost', '1 drop');

INSERT INTO Diagnosis (DiagnosisID, VisitID, DiagnosisDetails)
VALUES
(1, 1, 'Healthy'),
(2, 2, 'Mild Pain'),
(3, 3, 'Dental Plaque'),
(4, 4, 'Normal Vision');

INSERT INTO MedicalProcedure (ProcedureID, VisitID, ProcedureDescription, ProcedureCost)
VALUES
(1, 1, 'General Checkup', 100.00),
(2, 2, 'Tooth Cleaning', 200.00),
(3, 3, 'Eye Exam', 150.00),
(4, 4, 'Therapy Session', 300.00);

INSERT INTO LabTest (LabTestID, TestType, TestName, TestDate, TestResult, AppointmentID)
VALUES
(1, 'Blood Test', 'Complete Blood Count', '2024-01-15', 'Normal', 1),
(2, 'X-Ray', 'Chest X-Ray', '2024-02-20', 'Clear', 2),
(3, 'Urine Test', 'Urinalysis', '2024-03-25', 'No Abnormalities', 3),
(4, 'Vision Test', 'Eye Acuity Test', '2024-04-10', '20/20', 4);
```

After inserting all the dummy data, I ran the same codes for visualization to check if the insertion was executed. The results are shown below:

	InsuranceID	ProviderName	PolicyNumber	CoverageDetails
1	1	Blue Cross	BC12345	Full Coverage
2	2	United Health	UH67890	Partial Coverage
3	3	Aetna	AT11223	Dental Covera...
4	4	Cigna	CG44556	Vision Coverage

	PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Email	Address	InsuranceID
1	1	John	Doe	1990-01-01	Male	+1-123-456-7899	john.doe@example.com	123 Main St	1
2	2	Jane	Smith	1985-05-12	Female	+1-987-654-3210	jane.smith@example.com	456 Elm St	2
3	3	Alice	Johnson	2000-08-20	Female	+1-456-789-1230	alice.johnson@example...	789 Mapl...	3
4	4	Bob	Brown	1975-03-15	Male	+1-321-654-0987	bob.brown@example.c...	101 Pine Rd	4

	VisitID	PatientID	VisitDate	Reason	ReferralID	RoomID
1	1	1	2024-01-15	Annual Checkup	1	1
2	2	2	2024-02-20	Follow-up	2	2
3	3	3	2024-03-25	Dental Cleaning	3	3
4	4	4	2024-04-10	Eye Test	4	4

	RoomID	RoomNumber	Capacity
1	1	Room-101	2
2	2	Room-102	4
3	3	Room-103	1
4	4	Room-104	3

	ReferralID	ReferredBy	ReferredTo
1	1	Dr. Smith	Dr. Wilson
2	2	Dr. Clark	Dr. Johnson
3	3	Dr. Brown	Dr. Lee
4	4	Dr. Davis	Dr. Miller

	AppointmentID	PatientID	VisitID	AppointmentDate	StaffID
1	1	1	1	2024-01-15	1
2	2	2	2	2024-02-20	2
3	3	3	3	2024-03-25	3
4	4	4	4	2024-04-10	4

	StaffID	Name	Role
1	1	Dr. Sarah Miller	Physician
2	2	Dr. James Wilson	Dentist
3	3	Dr. Emily Clark	Optom...
4	4	Nurse Anna Jo...	Nurse

	DiagnosisID	VisitID	DiagnosisDetails
1	1	1	Healthy
2	2	2	Mild Pain
3	3	3	Dental Plaque
4	4	4	Normal Vision

	ProcedureID	VisitID	ProcedureDescription	ProcedureCost
1	1	1	General Checkup	100.00
2	2	2	Tooth Cleaning	200.00
3	3	3	Eye Exam	150.00
4	4	4	Therapy Session	300.00

	LabTestID	TestType	TestName	TestDate	TestResult	AppointmentID
1	1	Blood Test	Complete Blood Count	2024-01-15	Normal	1
2	2	X-Ray	Chest X-Ray	2024-02-20	Clear	2
3	3	Urine Test	Urinalysis	2024-03-25	No Abn...	3
4	4	Vision Test	Eye Acuity Test	2024-04-10	20/20	4

	AvailabilityID	StaffID	AvailableDate	AvailableTime
1	1	1	2024-01-10	09:00:00
2	2	2	2024-01-12	10:00:00
3	3	3	2024-01-15	14:00:00
4	4	4	2024-01-20	16:00:00

	PaymentID	VisitID	AmountPaid	PaymentDate
1	1	1	150.00	2024-01-16
2	2	2	200.00	2024-02-21
3	3	3	250.00	2024-03-26
4	4	4	300.00	2024-04-11

	BillingID	PatientID	TotalAmount	AmountPaid	BalanceDue	PaymentDate
1	1	1	200.00	150.00	50.00	2024-01-16
2	2	2	300.00	200.00	100.00	2024-02-21
3	3	3	250.00	250.00	0.00	2024-03-26
4	4	4	400.00	300.00	100.00	2024-04-11

	ChargeID	BillID	ChargeDescription	ChargeAmount
1	1	1	Consultation Fee	50.00
2	2	2	Dental Cleaning	100.00
3	3	3	Eye Test	50.00
4	4	4	Therapy Session	100.00

	PrescriptionID	AppointmentID	MedicationName	Dosage	Instructions	DispenseDate
1	1	1	Amoxicillin	500mg	Take twice daily	2024-01-16
2	2	2	Ibuprofen	200mg	Take as needed	2024-02-21
3	3	3	Acetaminophen	500mg	Take every 6 ...	2024-03-26
4	4	4	Latanoprost	1 drop	Apply nightly	2024-04-11

	MedicationID	MedicationName	Dosage
1	1	Amoxicillin	500mg
2	2	Ibuprofen	200mg
3	3	Acetaminophen	500mg
4	4	Latanoprost	1 drop

```sql
--- Running multiple queries ---

SELECT * FROM Insurance;   --(1)Verification
SELECT * FROM Patient;
SELECT * FROM Visit;

SELECT p.FirstName, p.LastName, v.VisitDate, r.RoomNumber  --(2)Testing relationships
FROM Visit v
JOIN Patient p ON v.PatientID = p.PatientID
JOIN Room r ON v.RoomID = r.RoomID;

SELECT p.FirstName, p.LastName, b.TotalAmount, b.BalanceDue, d.DiagnosisDetails  --(3)Testing complex relationships
FROM Bill b
JOIN Patient p ON b.PatientID = p.PatientID
JOIN Diagnosis d ON d.VisitID IN (
    SELECT VisitID FROM Visit WHERE PatientID = p.PatientID
);

SELECT p.FirstName, p.LastName, SUM(b.AmountPaid) AS TotalPayments  -- (4)Total payments grouped by patients
FROM Bill b
JOIN Patient p ON b.PatientID = p.PatientID
GROUP BY p.FirstName, p.LastName;

SELECT p.FirstName, p.LastName, i.ProviderName, b.TotalAmount -- (5)Combining data across multiple tables.
FROM Patient p
JOIN Insurance i ON p.InsuranceID = i.InsuranceID
JOIN Bill b ON p.PatientID = b.PatientID;

SELECT StaffID, COUNT(*) AS AppointmentsHandled -- (6)Using aggregrate queries
FROM Appointment
GROUP BY StaffID;

SELECT PatientID, TotalAmount, -- (7)Filtering, subqueries and calculated field
(TotalAmount - AmountPaid) AS OutstandingBalance
FROM Bill
WHERE TotalAmount > AmountPaid;
```

```sql
--- Running multiple queries ---

SELECT * FROM Insurance;    --(1)Verification
SELECT * FROM Patient;
SELECT * FROM Visit;
```

100 %

Results | Messages

	InsuranceID	ProviderName	PolicyNumber	CoverageDetails
1	1	Blue Cross	BC12345	Full Coverage
2	2	United Health	UH67890	Partial Coverage
3	3	Aetna	AT11223	Dental Coverage
4	4	Cigna	CG44556	Vision Coverage

	PatientID	FirstName	LastName	DateOfBirth	Gender	PhoneNumber	Email	Address	InsuranceID
1	1	John	Doe	1990-01-01	Male	+1-123-456-7899	john.doe@example.com	123 Main St	1
2	2	Jane	Smith	1985-05-12	Female	+1-987-654-3210	jane.smith@example.com	456 Elm St	2
3	3	Alice	Johnson	2000-08-20	Female	+1-456-789-1230	alice.johnson@example.com	789 Maple Ave	3
4	4	Bob	Brown	1975-03-15	Male	+1-321-654-0987	bob.brown@example.com	101 Pine Rd	4

	VisitID	PatientID	VisitDate	Reason	ReferralID	RoomID
1	1	1	2024-01-15	Annual Checkup	1	1
2	2	2	2024-02-20	Follow-up	2	2
3	3	3	2024-03-25	Dental Cleaning	3	3
4	4	4	2024-04-10	Eye Test	4	4

```sql
SELECT p.FirstName, p.LastName, v.VisitDate, r.RoomNumber  --(2)Testing relationships
FROM Visit v
JOIN Patient p ON v.PatientID = p.PatientID
JOIN Room r ON v.RoomID = r.RoomID;
```

100 %

Results | Messages

	FirstName	LastName	VisitDate	RoomNumber
1	John	Doe	2024-01-15	Room-101
2	Jane	Smith	2024-02-20	Room-102
3	Alice	Johnson	2024-03-25	Room-103
4	Bob	Brown	2024-04-10	Room-104

```sql
SELECT p.FirstName, p.LastName, b.TotalAmount, b.BalanceDue, d.DiagnosisDetails  --(3)Testing complex relationships
FROM Bill b
JOIN Patient p ON b.PatientID = p.PatientID
JOIN Diagnosis d ON d.VisitID IN (
    SELECT VisitID FROM Visit WHERE PatientID = p.PatientID
);
```

100 %

Results | Messages

	FirstName	LastName	TotalAmount	BalanceDue	DiagnosisDetails
1	John	Doe	200.00	50.00	Healthy
2	Jane	Smith	300.00	100.00	Mild Pain
3	Alice	Johnson	250.00	0.00	Dental Plaque
4	Bob	Brown	400.00	100.00	Normal Vision

```sql
SELECT p.FirstName, p.LastName, SUM(b.AmountPaid) AS TotalPayments  -- (4)Total payments grouped by patients
FROM Bill b
JOIN Patient p ON b.PatientID = p.PatientID
GROUP BY p.FirstName, p.LastName;
```

100 %

Results | Messages

	FirstName	LastName	TotalPayments
1	Bob	Brown	300.00
2	John	Doe	150.00
3	Alice	Johnson	250.00
4	Jane	Smith	200.00

```sql
SELECT p.FirstName, p.LastName, i.ProviderName, b.TotalAmount -- (5)Combining data across multiple tables.
FROM Patient p
JOIN Insurance i ON p.InsuranceID = i.InsuranceID
JOIN Bill b ON p.PatientID = b.PatientID;
```

100 %

Results | Messages

	FirstName	LastName	ProviderName	TotalAmount
1	John	Doe	Blue Cross	200.00
2	Jane	Smith	United Health	300.00
3	Alice	Johnson	Aetna	250.00
4	Bob	Brown	Cigna	400.00

```sql
SELECT StaffID, COUNT(*) AS AppointmentsHandled -- (6)Using aggregrate queries
FROM Appointment
GROUP BY StaffID;
```

100 %

Results | Messages

	StaffID	AppointmentsHandled
1	1	1
2	2	1
3	3	1
4	4	1

```sql
SELECT PatientID, TotalAmount, -- (7)Filtering, subqueries and calculated field
(TotalAmount - AmountPaid) AS OutstandingBalance
FROM Bill
WHERE TotalAmount > AmountPaid;
```

100 %

Results | Messages

	PatientID	TotalAmount	OutstandingBalance
1	1	200.00	50.00
2	2	300.00	100.00
3	4	400.00	100.00

Normalizaton & Denormalization

A. The database I have designed is normalized. Hence, I have not included any code for normalization.

I. All the data are atomic with no repeated groups. The Full Name is decomposed to First Name and Last Name in the Patient Table.

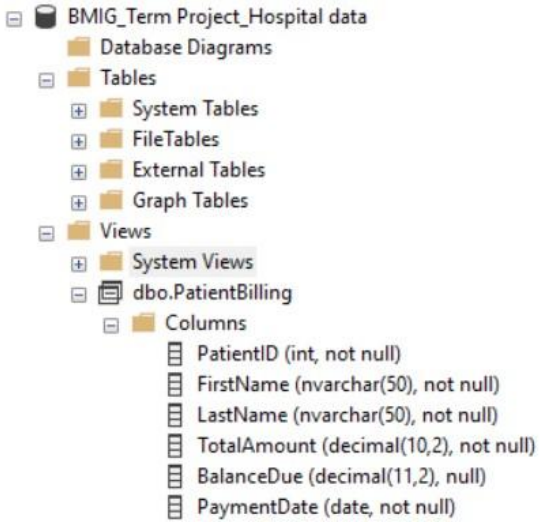
II. All the non-key columns, and attributes completely depend on the primary key.

III. There is zero redundant and duplicate data, as each table represents a single entity.

Although, denormalization was not necessary, I still tried for practice:

```sql
--- Denormalization ---

-- First Attempt
CREATE VIEW PatientBilling AS
SELECT p.PatientID, p.FirstName, p.LastName, b.TotalAmount, b.BalanceDue, b.PaymentDate
FROM Patient p
JOIN Bill b ON p.PatientID = b.PatientID;
```



```sql
-- Second Attempt

-- Altering Staff table to replace TEXT/NTEXT with NVARCHAR(MAX)
ALTER TABLE Staff
ALTER COLUMN Name NVARCHAR(255);
ALTER TABLE Staff
ALTER COLUMN Role NVARCHAR(255);

CREATE OR ALTER VIEW StaffAppointments AS
SELECT
    s.Name AS StaffName,
    s.Role,
    COUNT(a.AppointmentID) AS TotalAppointments
FROM Staff s
LEFT JOIN Appointment a ON s.StaffID = a.StaffID
GROUP BY s.Name, s.Role;

-- Checking the view manually
SELECT name
FROM sys.views
WHERE name = 'StaffAppointments';
SELECT * FROM StaffAppointments;
```

100 %

Results | Messages

	name
1	StaffAppointments

	StaffName	Role	TotalAppointments
1	Dr. James Wilson	Dentist	1
2	Nurse Anna Johnson	Nurse	1
3	Dr. Emily Clark	Optometrist	1
4	Dr. Sarah Miller	Physician	1

```sql
--- Trying triggers ---

-- What if I want to log any updates to the PATIENT Table?

 -- Creating the table as it did not exist before
CREATE TABLE PatientLog (
    LogID INT IDENTITY PRIMARY KEY,     -- Auto-incrementing ID for the log
    PatientID INT,                      -- ID of the patient being logged
    Action NVARCHAR(50),                -- Action performed (e.g., "UPDATE")
    ActionDate DATETIME DEFAULT GETDATE() -- Timestamp of the action
);

--  Creating the trigger
CREATE TRIGGER trg_Patient_Update
ON Patient
AFTER UPDATE
AS
BEGIN
    -- Inserting a log entry for updated rows
    INSERT INTO PatientLog (PatientID, Action, ActionDate)
    SELECT PatientID, 'UPDATE', GETDATE()
    FROM inserted;
END;

-- Testing the trigger
UPDATE Patient
SET PhoneNumber = '+1-123-456-7899'
WHERE PatientID = 1;

-- Checking the PatientLog table at last
SELECT * FROM PatientLog;

--------------------------- THE END -----------------------------
```

	LogID	PatientID	Action	ActionDate
1	1	1	UPDATE	2024-12-05 07:52:07.823
2	2	1	UPDATE	2024-12-05 10:56:56.067