



# Integrated Regulatory Approach to Database Design, Development, and Optimization

By:

Adedayo Kukoyi

DEPARTMENT OF INFORMATION TECHNOLOGY,  
PURDUE UNIVERSITY GLOBAL

July 10, 2025.

## Universal Health Database Design.

### PART 1. DATABASE DEFINITION

#### INTRODUCTION

The capstone project is being carried out to develop a healthcare database for a client named Universal Health. The database would be implemented to meet the client's data requirements. The database will include entities like Patient, Appointment, Insurance, Medical History, Physician, Consent, and Procedures.

#### PURPOSE

The project's purpose is to create an affordable healthcare database for small or mid-size healthcare organizations. The aim is to design and develop a cost-effective relational database that efficiently stores and retrieves electronic health data generated by a healthcare facility. The stored data includes patient information, medical history, physician details, diagnoses, procedures, admission records, and medications. This database ensures easy access to patient medical records while maintaining data integrity and consistency. Healthcare facilities are mandated by law (e.g., HIPAA) to secure patient health records in ways that do not compromise the patient's sensitive records. A health facility that fails to implement a secure database management system in managing protected health records may be exposed to regulatory risk and attract a stiff penalty.

### PART 2. DATABASE DESIGN

#### List of Tables (Entity Types):

The Universal Health database is designed to include the tables listed below.

1. **Patient:** Contains patient demographics information.
2. **Physician:** Contains personal information of Physicians hired by the Universal Health Facility.
3. **Insurance:** Contains insurance information for the patient.
4. **Emergency Contact:** Contains personal information for emergency contact for patients.
5. **Appointment:** Contains patient appointment details.
6. **Prescription:** Contains information about prescriptions generated.
7. **Medication:** Contain information about medications.
8. **Medical History:** Contains the patient's medical history.
9. **Procedure:** Contain information about procedures.

**10. Consent:** Contains information about every consent issued by the patient.

### Universal Health Business Rules

The table below contains the database entities, Entity Pair, Cardinality, and the Entity Business Rule.

Entity	Entity Pair	Cardinality	Entity Business Rules
Patient	Emergency Contact	One-to-Many	The patient must have at least one emergency contact.
Patient	Insurance	One-to-Many	The patient must have primary insurance on file.
Patient	Appointment	One-to-Many	A patient can have multiple appointments. Appointments cannot overlap.
Patient	Prescription	One-to-Many	A patient may have prescriptions that are linked to an appointment.
Patient	Medical History	One-to-One	A patient must have a medical history.
Patient	Consent	One-to-many	A patient must have at least one consent on file. A patient can have many contents. Examples include consent for treatment, consent for procedure, etc.
Appointment	Physician	Many-to-One	A physician or a nurse can be assigned to many appointments. Appointments cannot overlap.
Prescription	Physician	Many-to-One	A physician can create many prescriptions.
Prescription	Medication	Many-to-One	A medication needs at least one prescription. Many prescriptions can be generated for one medication.
Procedure	Physician	One-to-One	At least one physician must perform a procedure.
Procedure	Consent	One-to-One	Consent must be in place for every procedure.

**Figure 2.0: Business Rule Table.**

### Entity Structure and Attributes.

The table below lists the entities in the universal health database and their different attributes.

<u>Entity Name, Attributes, Primary and Secondary Keys.</u>		
<u>Patient</u>	<u>Insurance</u>	<u>Prescription</u>
<ul style="list-style-type: none"> <li>• PatientID (PK)</li> <li>• FirstName</li> <li>• LastName</li> <li>• DOB (Date of Birth)</li> <li>• Gender</li> <li>• Phone</li> <li>• Email</li> <li>• Address</li> <li>• City</li> </ul>	<ul style="list-style-type: none"> <li>• InsuranceID (PK)</li> <li>• ProviderName</li> <li>• PolicyNumber</li> <li>• GroupNumber</li> <li>• CoverageType</li> <li>• PatientID (FK)</li> </ul>	<ul style="list-style-type: none"> <li>• PrescriptionID (PK)</li> <li>• DatePrescribed</li> <li>• DosageInstruction</li> <li>• PatientID (FK)</li> <li>• PhysicianID (FK)</li> <li>• MedicationID (FK)</li> </ul>

<ul style="list-style-type: none"> <li>• State</li> <li>• ZipCode</li> <li>• SSN</li> <li>• InsuranceID FK</li> <li>• EmergencyContactID FK</li> </ul>		
<p><b><u>EmergencyContact</u></b></p> <ul style="list-style-type: none"> <li>• EmergencyContactID (PK)</li> <li>• FirstName</li> <li>• LastName</li> <li>• Relationship</li> <li>• Phone</li> <li>• Email</li> <li>• PatientID (FK)</li> </ul>	<p><b><u>Medication</u></b></p> <ul style="list-style-type: none"> <li>• MedicationID (PK)</li> <li>• Name</li> <li>• Dosage</li> <li>• Description</li> </ul>	<p><b><u>MedicalHistory</u></b></p> <ul style="list-style-type: none"> <li>• RecordID (PK)</li> <li>• Diagnosis</li> <li>• TreatmentPlan</li> <li>• RecordDate</li> <li>• PatientID (FK)</li> </ul>
<p><b><u>Appointment</u></b></p> <ul style="list-style-type: none"> <li>• AppointmentID (PK)</li> <li>• AppointmentDate</li> <li>• Status</li> <li>• Reason</li> <li>• PatientID (FK)</li> <li>• PhysicianID (FK)</li> </ul>	<p><b><u>Procedures</u></b></p> <ul style="list-style-type: none"> <li>• ProcedureID (PK)</li> <li>• Name</li> <li>• Description</li> <li>• Cost</li> <li>• RecordID (FK)</li> </ul>	<p><b><u>Physician</u></b></p> <ul style="list-style-type: none"> <li>• PhysicianID (PK)</li> <li>• FirstName</li> <li>• LastName</li> <li>• Specialty</li> <li>• Email</li> <li>• Phone</li> </ul>
<p><b><u>Consent</u></b></p> <ul style="list-style-type: none"> <li>• ConsentID (PK)</li> <li>• ConsentType</li> <li>• ConsentDate</li> <li>• ExpirationDate</li> <li>• PatientID (FK)</li> </ul>		

Figure 2.1: Entities Structure and Attributes.

### PART 3: ERD SCHEMA DIAGRAM

#### ERD DIAGRAM

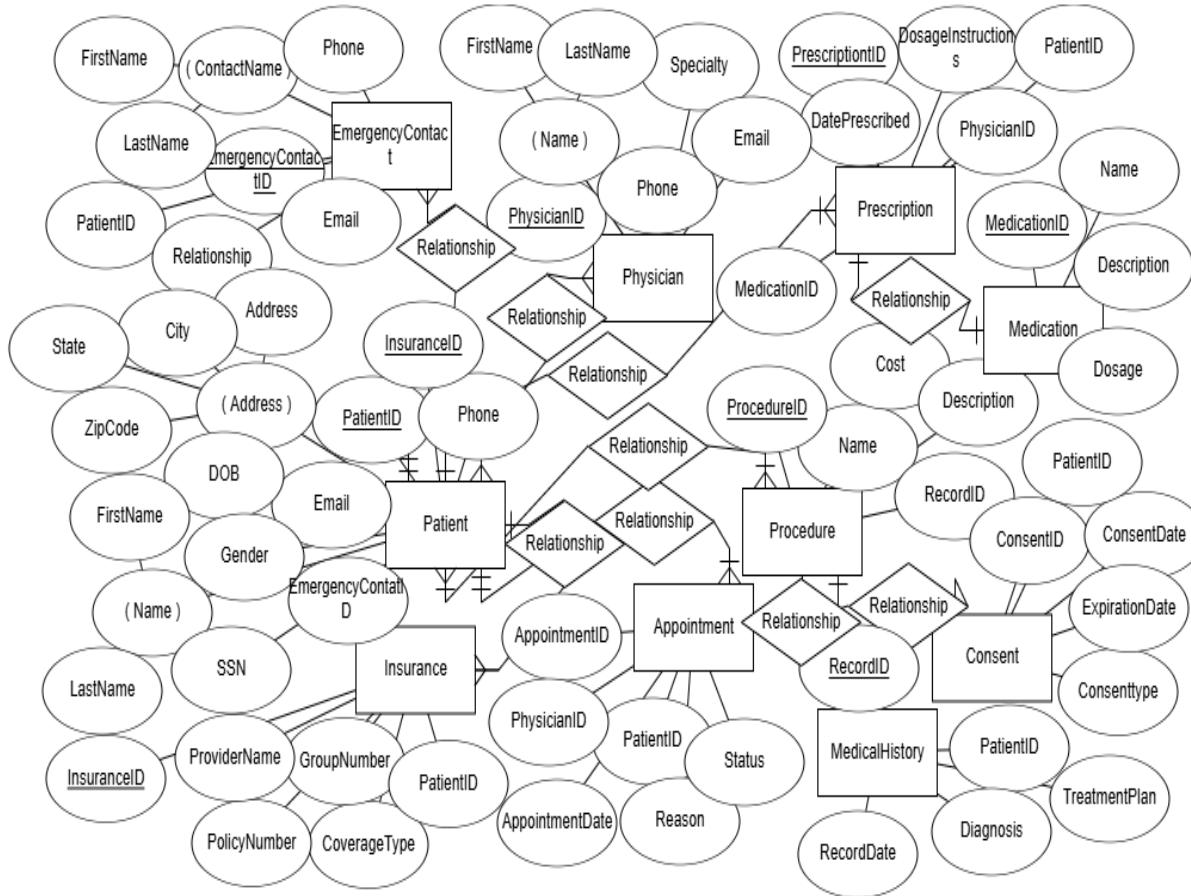
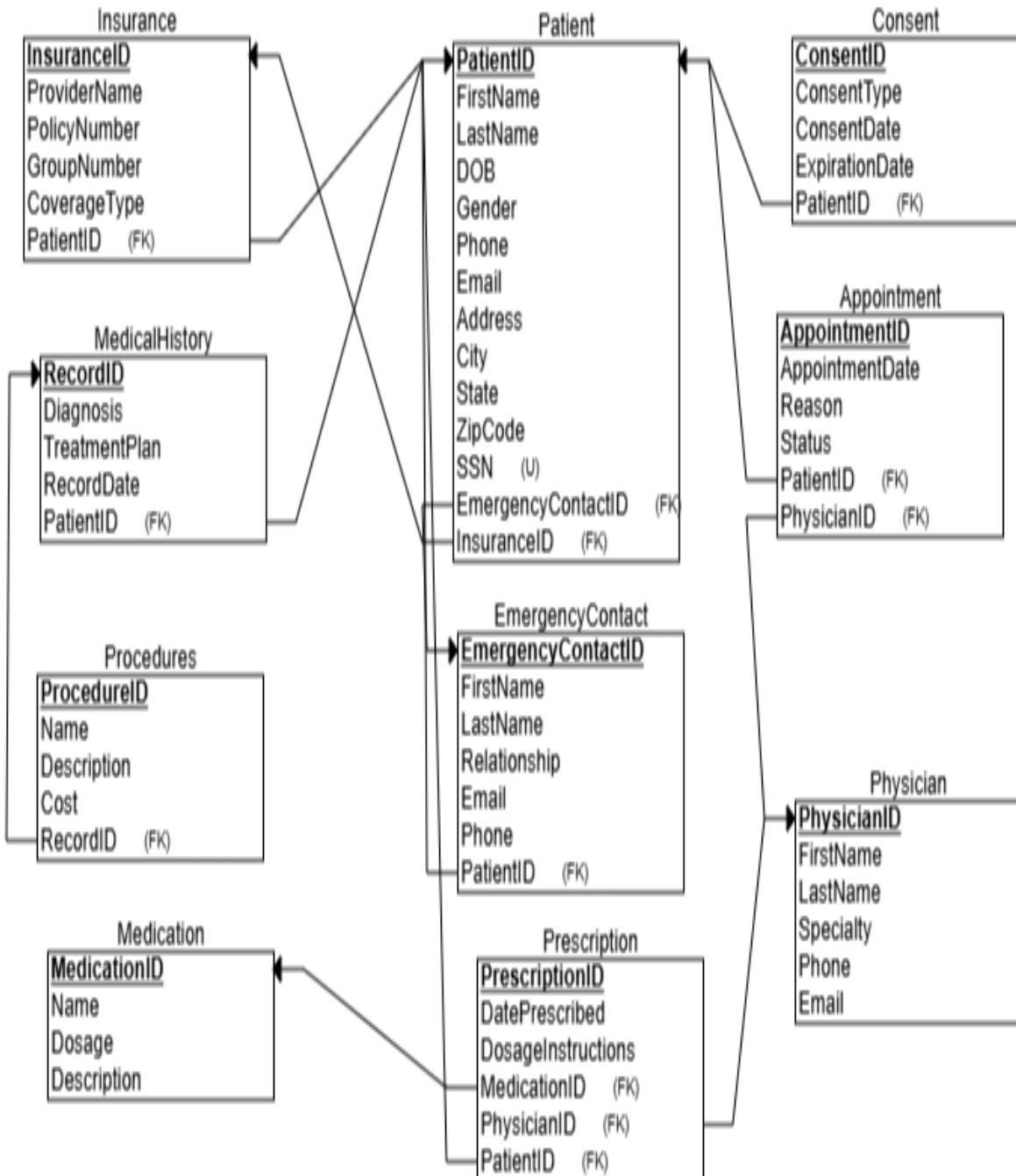


Figure 3.0 Entity Relationship Diagram.

**ERD SCHEMA DIAGRAM****Figure 3.1 Entity Relationship Schema Diagram.**

## ENTITIES, STRUCTURES & RELATIONSHIP DIAGRAM

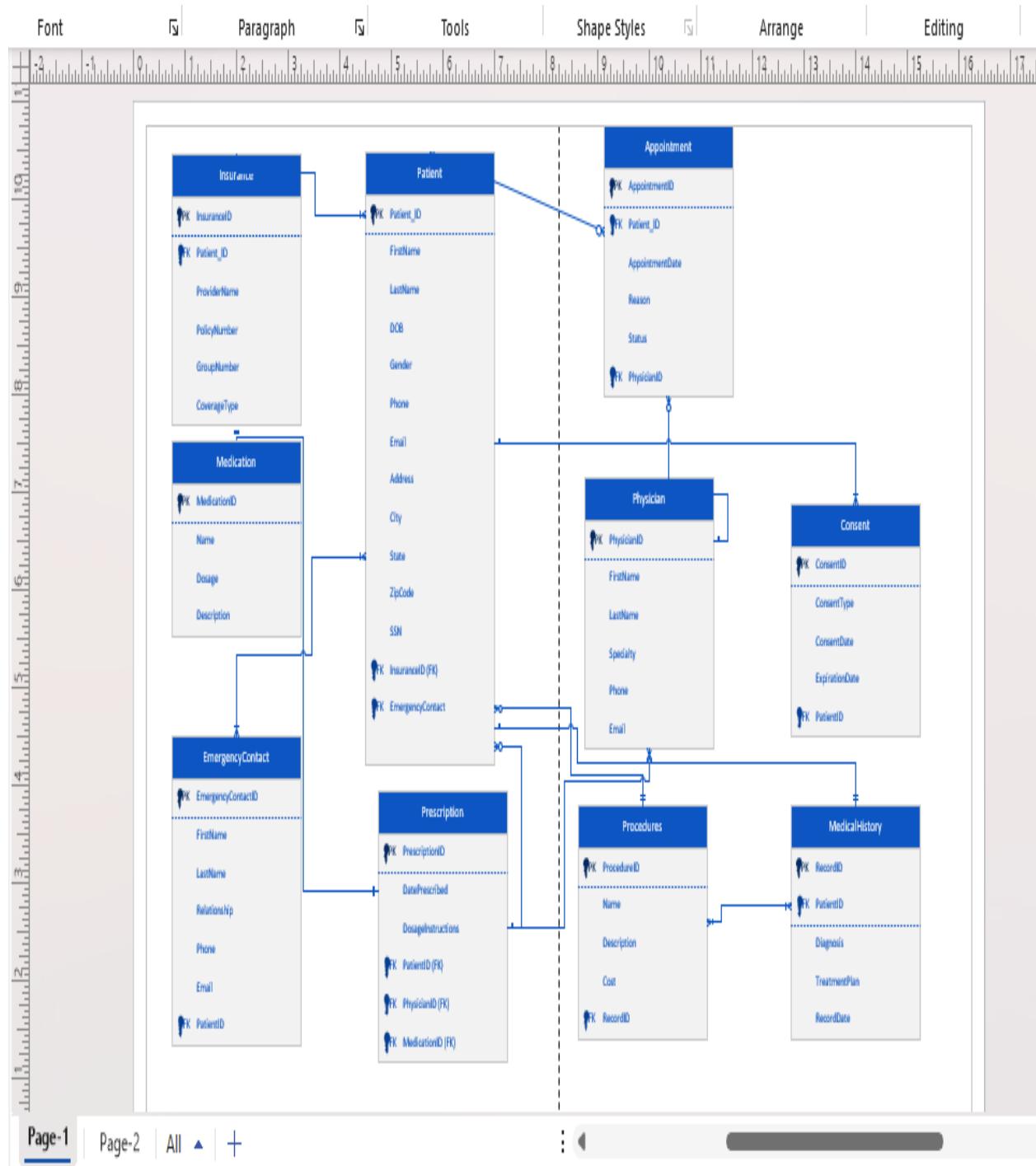


Figure 3.2: Entity Structure & Relationship Diagram.

#### PART 4: DATABASE DESIGN AND NORMALIZATION ASSERTION.

The Insurance-Patient entity pair below explains how the database was normalized.

Table 4.0 Normalization Example: using the insurance and patient pair.

InsuranceID	Provider Name	Policy Number	Group Number	Coverage Type	PatientID	PatientName (First & Last)	DOB	Email
-------------	---------------	---------------	--------------	---------------	-----------	----------------------------	-----	-------

- InsuranceID and PatientID are the Primary keys for the Insurance table and Patient table, respectively.
- Non-key attributes with different data were separated into fields.
- The insurance table non-key attribute depends on the InsuranceID.
- Please note that the PatientID is a foreign key in the Insurance table; hence, the non-key attributes of the insurance table depend on the InsuranceID key.
- Please note that the InsuranceID is a foreign key in the Patient table, hence the non-key attributes of the patient table depend on the patientID key.

The resulting tables are below:

Table 4.1 Insurance Schema

InsuranceID	PatientID	ProviderName	PolicyNumber	GroupNumber	CoverageType
-------------	-----------	--------------	--------------	-------------	--------------

Table 4.2 Patient schema

PatientID	InsuranceID	PatientFirstname	PatientLastname	DOB	Email	Address
-----------	-------------	------------------	-----------------	-----	-------	---------

#### Normalization:

The normalization table above explains how form attributes were normalized based on 1NF, 2NF, and 3NF, respectively.

- 1NF – YES <Repeat group was removed from the table column> For example, Insurance policy number and policy name were separated into different columns to eliminate redundancy.

- 2NF – YES <All partial dependency was removed, reorganized, and placed in a separate table> See the Normalization example using the Insurance: Patient pair.
- 3NF – YES <Ensure that all non-key attributes depend on a primary key> See the Normalization example using the insurance: patient entity pair.  
(Kendall and Kendall, 2020).

## **PART 5: DATABASE DOCUMENTATION**

### **Universal Health Data Dictionary**

The tables below show the characteristics of each table and a brief description of the purpose of the table in the database.

**Table 5.0** Schema Name: Appointment

**Purpose:** The table's purpose is to store patient appointment information for universal health.

Field	Description	Data Type	Allow Null	Key
AppointmentID	Appointment table primary key.	Integer	No	Primary Key
PatientID	Appointment table foreign key.	Integer	No	Foreign Key
PhysicianID	Appointment table foreign key.	Integer	No	Foreign Key
AppoinmentDate	Patient appointment date	DateTime	No	Not a Key
Reason	Reason for hospital visit	Text	No	Not a Key
Status	Patient arrival status	Varchar (20)	No	Not a Key

**Table 5.1** Schema Name: Consent

**Purpose:** The table's purpose is to store the patient's consent information.

Field	Description	Data Type	Allow Null	Key
ConsentID	Consent table primary key.	Integer	No	Primary Key
PatientID	Consent table foreign key.	Integer	No	Foreign Key

ConsentType	Type of consent collected from the patient.	Varchar (100)	No	Not a Key
ConsentDate	Date & time the patient issued consent	DateTime	No	Not a Key
ExpirationDate	Consent expiration date & time.	DateTime	No	Not a Key

**Table 5.2** Schema Name: EmergencyContact

**Purpose:** The table's purpose is to store the patient's emergency contact information.

Field	Description	Data Type	Allow Null	Key
EmergencyContactID	Emergency Contact table primary key.	Integer	No	Primary Key
PatientID	Emergency Contact table foreign key.	Integer	No	Foreign Key
FirstName	Patient's emergency contact's first name.	Varchar (20)	No	Not a Key
LastName	Patient's emergency contact's last name.	Varchar (20)	No	Not a Key
Relationship	Emergency contact relationship to the patient.	Varchar (20)	No	Not a Key
Phone	Emergency contact phone number.	Varchar (15)	No	Not a Key
Email	Emergency contact email address.	Varchar (50)	No	Not a Key

**Table 5.3** Schema Name: Insurance

**Purpose:** The table's purpose is to store the patient's insurance information.

Field	Description	Data Type	Allow Null	Key
InsuranceID	Insurance table primary key.	Integer	No	Primary Key
PatientID	Insurance table foreign key.	Integer	No	Foreign Key
ProviderName	The name of the Patient's insurance provider	Varchar (20)	No	Not a Key
PolicyNumber	Patient insurance policy number.	Varchar (20)	No	Not a Key

GroupNumber	Patient insurance group number.	Varchar (20)	No	Not a Key
CoverageType	Patient insurance coverage type.	Varchar (20)	No	Not a Key

**Table 5.4** Schema Name: Medical History

**Purpose:** The table's purpose is to store the patient's medical history.

Field	Description	Data Type	Allow Null	Key
RecordID	The medical history table's primary key.	Integer	No	Primary Key
PatientID	The medical history table foreign key.	Integer	No	Foreign Key
Diagnosis	This is a record of the patient's diagnosis.	Varchar (225)	No	Not a Key
TreatmentPlan	This is the patient's treatment plan.	Nvarchar (200)	No	Not a Key
RecordDate	Date & time the record was generated.	DateTime	No	Not a Key

**Table 5.5** Schema Name: Medication

**Purpose:** The table's purpose is to store the patient's medication information.

Field	Description	Data Type	Allow Null	Key
MedicationID	The medication table's primary key.	Integer	No	Primary Key
Name	Name of medication.	Varchar (20)	No	Not a Key
Dosage	Prescribed dosage.	Varchar (50)	No	Not a Key
Description	Description of medication.	Text	No	Not a Key

**Table 5.5** Schema Name: Patient

**Purpose:** The table's purpose is to store the patient's information.

Field	Description	Data Type	Allow Nulls	Key
PatientID	Patient table primary key.	Integer	No	Primary Key
FirstName	Patient's first name.	Varchar (20)	No	Not a Key
LastName	Patient's last name.	Varchar (20)	No	Not a Key
DOB	Patient's date of birth	Date	No	Not a Key

Gender	Patient's Gender	Char (1)	No	Not a Key
Phone	Phone number of the patient.	Varchar (15)	Yes	Not a Key
Email	Email of the patient.	Varchar (50)	Yes	Not a Key
Address	Residential & mailing address of the patient.	Varchar (100)	No	Not a Key
City	Patient City name.	Varchar (15)	No	Not a Key
State	Patient state name	Varchar (10)	No	Not a Key
ZipCode	Address Zip Code for the customer.	Varchar (5)	No	Not a Key
SSN	Patient's social security number.	Char (11)	Yes	Not a Key
InsuranceID	Patient's table foreign key.	Integer	No	Not a Key
EmergencyContactID	Patient's table foreign key.	Integer	No	Not a Key

**Table 5.6** Schema Name: Physician

**Purpose:** The table's purpose is to store personal information about the physician.

Field	Description	Data Type	Allow Nulls	Key
PhysicianID	Physician table primary key.	Integer	No	Primary Key
FirstName	Physician's first name.	Varchar (20)	No	Not a Key
LastName	Physician's last name.	Varchar (20)	No	Not a Key
Specialty	Physician's area of specialization.	Varchar (20)	No	Not a Key
Phone	Physician phone number.	Varchar (15)	No	Not a Key
Email	Physician's email address.	Varchar(50)	No	Not a Key

**Table 5.7** Schema Name: Prescription

**Purpose:** The table's purpose is to store information about the patient's prescription.

Field	Description	Data Type	Allow Nulls	Key

PrescriptionID	Prescription table primary key.	Integer	No	Primary Key
PatientID	Prescription table Foreign Key.	Integer	No	Foreign Key
PhysicianID	Prescription table Foreign Key.	Integer	No	Foreign Key
MedicationID	Prescription table Foreign Key.	Integer	No	Foreign Key
DatePrescribed	Prescription date.	Date	No	Not a Key
DosageInstructions	Dosage instructions.	Text	No	Not a Key

**Table 5.8 Schema Name: Procedures.**

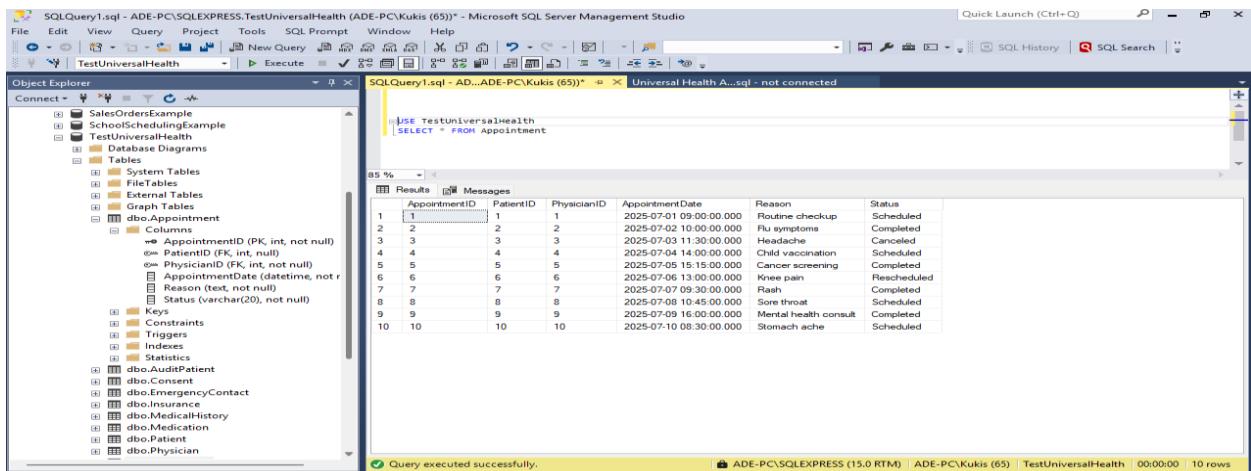
**Purpose:** The table's purpose is to store information about the patient's procedures.

Field	Description	Data Type	Allow Nulls	Key
ProcedureID	Procedures table primary key.	Integer	No	Primary Key
RecordID	Procedures table Foreign Key.	Integer	No	Foreign Key
Name	The name of the procedure.	Varchar (20)	No	Not a Key
Description	Description of the procedure.	Text	No	Not a Key
Cost	Cost of Procedure.	Decimal (10, 2)	No	Not a Key

## PART 6: SCHEMAS WITH SAMPLE DATA.

The Figure below shows the efficiency of the database in patient data retrieval with sample data.

Figure 6.0 - Appointment Schema with Sample Data.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several schemas and tables, including SalesOrdersExample, SchoolSchedulingExample, TestUniversalHealth, and various appointment-related tables like Appointments, Appointments, and Appointments. The central pane displays a query results window titled "Results" for the "Appointment" table. The query executed is:

```
USE TestUniversalHealth
SELECT * FROM Appointment
```

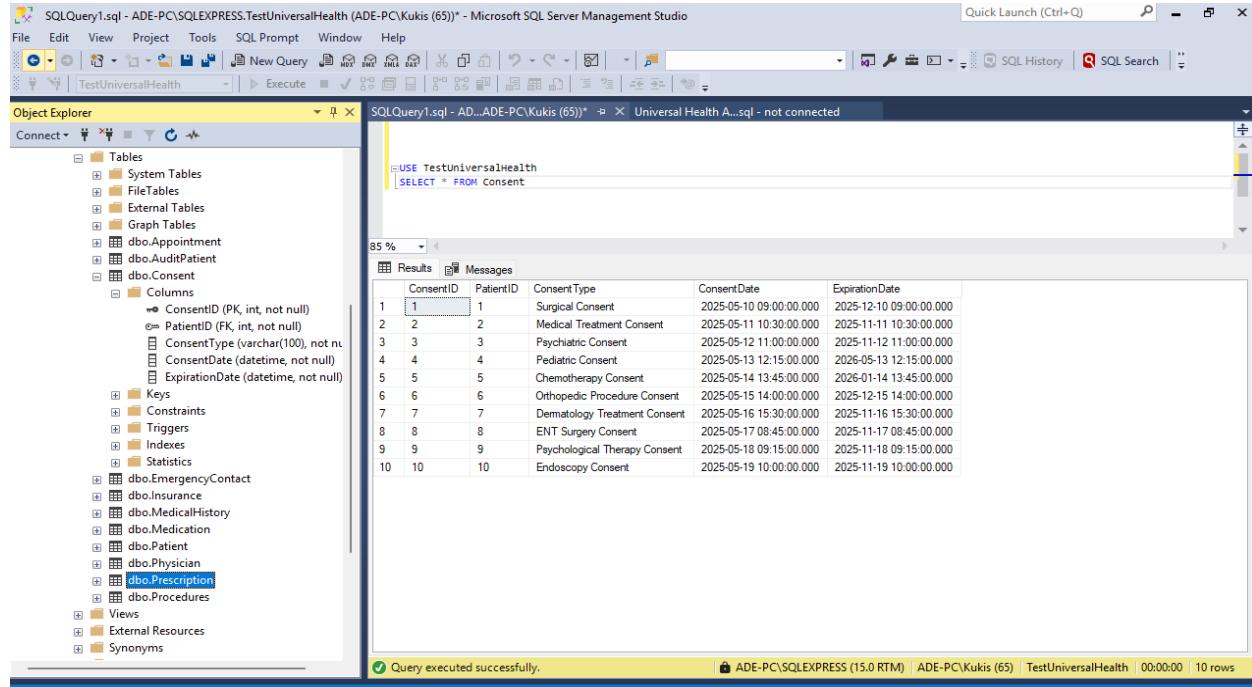
The results grid shows 10 rows of data:

AppointmentID	PatientID	PhysicianID	AppointmentDate	Reason	Status
1	1	2	2025-07-01 09:00:00.000	Routine checkup	Scheduled
2	2	2	2025-07-03 10:00:00.000	Flu symptoms	Completed
3	3	3	2025-07-03 11:30:00.000	Headache	Canceled
4	4	4	2025-07-04 14:00:00.000	Child vaccination	Scheduled
5	5	5	2025-07-05 15:15:00.000	Cancer screening	Completed
6	6	6	2025-07-06 13:00:00.000	Knee pain	Rescheduled
7	7	7	2025-07-07 09:30:00.000	Rash	Completed
8	8	8	2025-07-08 10:45:00.000	Sore throat	Scheduled
9	9	9	2025-07-09 16:00:00.000	Mental health consult	Completed
10	10	10	2025-07-10 08:30:00.000	Stomach ache	Scheduled

At the bottom of the screen, a status bar indicates "Query executed successfully." and shows the connection details: ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows.

The image above shows the Appointment test data retrieval. The query was executed in less than 1 second.

Figure 6.1 - Consent Schema with Sample Data.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including tables like `Consent`, `EmergencyContact`, and `Prescription`. The main window displays a query results grid for the `Consent` table. The query is:

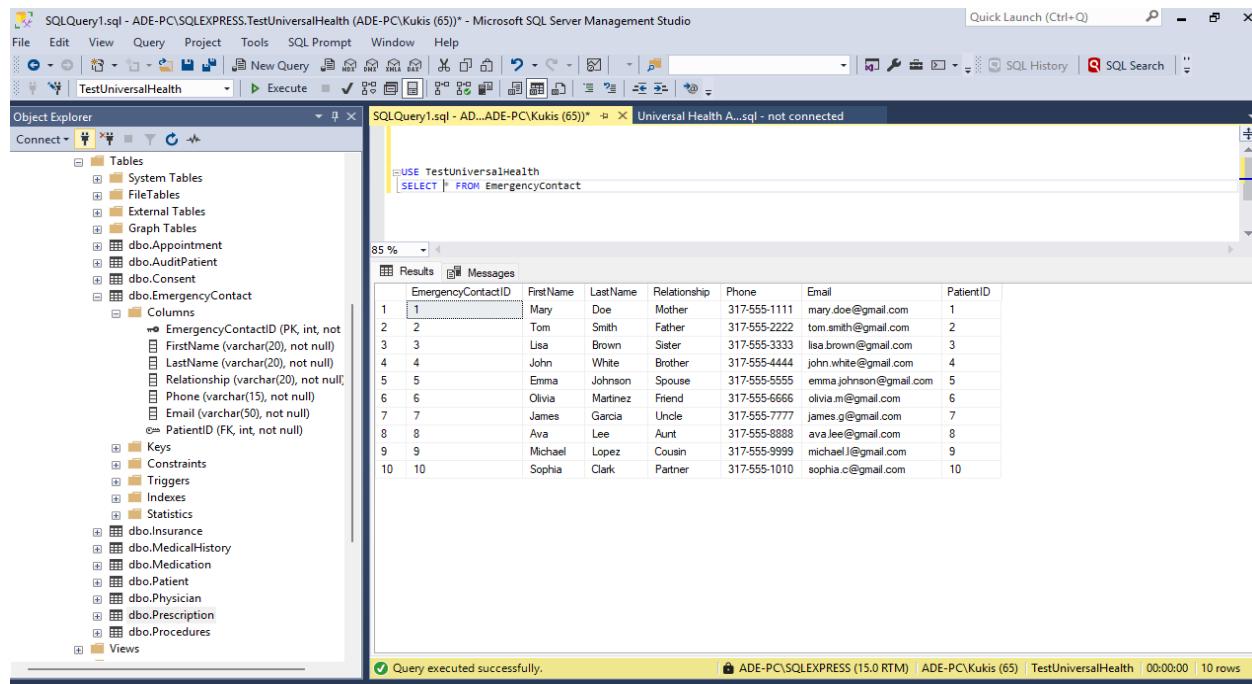
```
USE TestUniversalHealth
SELECT * FROM Consent
```

The results show 10 rows of sample data:

ConsentID	PatientID	ConsentType	ConsentDate	ExpirationDate
1	1	Surgical Consent	2025-05-10 09:00:00.000	2025-12-10 09:00:00.000
2	2	Medical Treatment Consent	2025-05-11 10:30:00.000	2025-11-11 10:30:00.000
3	3	Psychiatric Consent	2025-05-12 11:00:00.000	2025-11-12 11:00:00.000
4	4	Pediatric Consent	2025-05-13 12:15:00.000	2026-05-13 12:15:00.000
5	5	Chemotherapy Consent	2025-05-14 13:45:00.000	2026-01-14 13:45:00.000
6	6	Orthopedic Procedure Consent	2025-05-15 14:00:00.000	2025-12-15 14:00:00.000
7	7	Dermatology Treatment Consent	2025-05-16 15:30:00.000	2025-11-16 15:30:00.000
8	8	ENT Surgery Consent	2025-05-17 08:45:00.000	2025-11-17 08:45:00.000
9	9	Psychological Therapy Consent	2025-05-18 09:15:00.000	2025-11-18 09:15:00.000
10	10	Endoscopy Consent	2025-05-19 10:00:00.000	2025-11-19 10:00:00.000

Query executed successfully.

Figure 6.2 - EmergencyContact Schema with Sample Data.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including tables like `EmergencyContact`, `Patient`, and `Prescription`. The main window displays a query results grid for the `EmergencyContact` table. The query is:

```
USE TestUniversalHealth
SELECT * FROM EmergencyContact
```

The results show 10 rows of sample data:

EmergencyContactID	FirstName	LastName	Relationship	Phone	Email	PatientID
1	Mary	Doe	Mother	317-555-1111	mary.doe@gmail.com	1
2	Tom	Smith	Father	317-555-2222	tom.smith@gmail.com	2
3	Lisa	Brown	Sister	317-555-3333	lisa.brown@gmail.com	3
4	John	White	Brother	317-555-4444	john.white@gmail.com	4
5	Emma	Johnson	Spouse	317-555-5555	emma.johnson@gmail.com	5
6	Olivia	Martinez	Friend	317-555-6666	olivia.m@gmail.com	6
7	James	Garcia	Uncle	317-555-7777	james.g@gmail.com	7
8	Ava	Lee	Aunt	317-555-8888	ava.lee@gmail.com	8
9	Michael	Lopez	Cousin	317-555-9999	michael.l@gmail.com	9
10	Sophia	Clark	Partner	317-555-1010	sophia.c@gmail.com	10

Query executed successfully.

Figure 6.3 - Insurance Schema with Sample Data.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like System Tables, FileTables, External Tables, Graph Tables, dbo.Appointment, dbo.AuditPatient, dbo.Consent, dbo.EmergencyContact, dbo.Insurance, dbo.Columns, dbo.Keys, dbo.Constraints, dbo.Triggers, dbo.Indexes, dbo.Statistics, dbo.MedicalHistory, dbo.Medication, dbo.Patient, dbo.Physician, dbo.Prescription, dbo.Procedures, Views, and External Resources. The central Results pane displays the output of a query:

```
USE TestUniversalHealth
SELECT * FROM Insurance
```

The results show 10 rows of sample data:

InsuranceID	ProviderName	PolicyNumber	GroupNumber	CoverageType	PatientID
1	Aetna	POL12345	GRP1001	Full	1
2	Cigna	POL12346	GRP1002	Partial	2
3	BlueCross	POL12347	GRP1003	Full	3
4	UnitedHealth	POL12348	GRP1004	Dental	4
5	Kaiser	POL12349	GRP1005	Vision	5
6	Aetna	POL12350	GRP1006	Full	6
7	Cigna	POL12351	GRP1007	Partial	7
8	BlueCross	POL12352	GRP1008	Dental	8
9	UnitedHealth	POL12353	GRP1009	Full	9
10	Kaiser	POL12354	GRP1010	Vision	10

At the bottom, a message indicates "Query executed successfully." and shows connection details: ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows.

Figure 6.4 - MedicalHistory Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like System Tables, FileTables, External Tables, Graph Tables, dbo.Appointment, dbo.AuditPatient, dbo.Consent, dbo.EmergencyContact, dbo.Insurance, dbo.MedicalHistory, dbo.Columns, dbo.Keys, dbo.Constraints, dbo.Triggers, dbo.Indexes, dbo.Statistics, dbo.Medication, dbo.Patient, dbo.Physician, dbo.Prescription, dbo.Procedures, Views, and External Resources. The central Results pane displays the output of a query:

```
USE TestUniversalHealth
SELECT * FROM MedicalHistory
```

The results show 10 rows of sample data:

RecordID	PatientID	Diagnosis	TreatmentPlan	RecordDate
1	1	Hypertension	Daily medication and diet control.	2025-04-01 08:00:00.000
2	2	Type 2 Diabetes	Insulin and exercise.	2025-04-02 09:00:00.000
3	3	Asthma	Inhaler therapy.	2025-04-03 10:00:00.000
4	4	Allergic Rhinitis	Antihistamines and avoidance.	2025-04-04 11:00:00.000
5	5	Breast Cancer	Chemotherapy and surgery.	2025-04-05 12:00:00.000
6	6	Knee Osteoarthritis	Pain management and physiotherapy.	2025-04-06 13:00:00.000
7	7	Psoriasis	Topical treatments.	2025-04-07 14:00:00.000
8	8	Tonsillitis	Antibiotics and rest.	2025-04-08 15:00:00.000
9	9	Anxiety Disorder	Therapy and medication.	2025-04-09 16:00:00.000
10	10	Irregular Bowel Syndrome	Diet modification and medication.	2025-04-10 17:00:00.000

At the bottom, a message indicates "Query executed successfully." and shows connection details: ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows.

Figure 6.5 - Medication Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects under the 'TestUniversalHealth' database, including tables like System Tables, External Tables, Graph Tables, Appointments, AuditPatient, Consent, EmergencyContact, Insurance, MedicalHistory, Medication, and Patient. The 'Medication' table is selected, and its columns (MedicationID, Name, Dosage, Description) are visible. The 'Tables' node under 'Medication' is expanded, showing the primary key 'MedicationID' and other columns. The 'Results' tab in the center displays a query result set for the 'Medication' table. The query is:

```
USE TestUniversalHealth
SELECT * FROM Medication
```

The results show 10 rows of medication data:

MedicationID	Name	Dosage	Description
1	Aspirin	100 mg	Pain reliever and anti-inflammatory.
2	Metformin	500 mg	Used to control blood sugar levels.
3	Lisinopril	10 mg	Treats high blood pressure.
4	Ibuprofen	200 mg	Common over-the-counter NSAID.
5	Aoxicillin	250 mg	Broad-spectrum antibiotic.
6	Atorvastatin	20 mg	Lowers cholesterol.
7	Levothyroxine	50 mcg	Thyroid hormone replacement.
8	Omeprazole	40 mg	Reduces stomach acid.
9	Albuterol	90 mcg	Relieves asthma symptoms.
10	Sertraline	100 mg	Used for depression and anxiety.

Below the results, a message bar indicates: 'Query executed successfully.' and shows connection details: ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows.

Figure 6.6 - Patient Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects under the 'TestUniversalHealth' database, including tables like External Tables, Graph Tables, Appointments, AuditPatient, Consent, EmergencyContact, Insurance, MedicalHistory, Medication, and Patient. The 'Patient' table is selected, and its columns (PatientID, FirstName, LastName, DOB, Gender, Phone, Email, Address, City, State, ZipCode, SSN, Ins) are visible. The 'Tables' node under 'Patient' is expanded, showing the primary key 'PatientID' and other columns. The 'Results' tab in the center displays a query result set for the 'Patient' table. The query is:

```
USE TestUniversalHealth
SELECT * FROM Patient
```

The results show 10 rows of patient data:

PatientID	FirstName	LastName	DOB	Gender	Phone	Email	Address	City	State	ZipCode	SSN	Ins
1	John	Doe	1980-03-15	M	317-555-2010	john.doe@gmail.com	123 Maple St	Avon	Indiana	46213	123-45-6789	1
2	Jane	Smith	1990-07-22	F	317-555-2020	jane.smith@gmail.com	456 Oak St	Marion	Indiana	46224	987-65-4321	2
3	Robert	Johnson	1975-01-10	M	317-555-2030	robert.j@gmail.com	789 Pine St	Avon	Indiana	46213	111-22-3333	3
4	Emily	Davis	1985-06-30	F	317-555-2040	emily.d@gmail.com	321 Birch St	Avon	Indiana	46213	444-55-6666	4
5	Michael	Brown	1992-11-20	M	317-555-2050	mike.b@gmail.com	654 Elm St	Marion	Indiana	46224	777-88-9999	5
6	Sarah	Wilson	1978-04-12	F	317-555-2060	sarah.w@gmail.com	111 Cedar St	Marion	Indiana	46224	222-33-4444	6
7	David	Clark	1965-08-25	M	317-555-2070	david.c@gmail.com	222 Aspen St	Marion	Indiana	46224	555-66-7777	7
8	Linda	Martinez	1989-12-18	F	317-555-2080	linda.m@gmail.com	333 Walnut St	Marion	Indiana	46224	88-89-9000	8
9	James	Lewis	2000-02-01	M	317-555-2090	james.l@gmail.com	444 Chestnut St	Marion	Indiana	46224	000-11-2222	9
10	Karen	Hall	1972-09-09	F	317-555-2100	karen.h@gmail.com	555 Redwood St	Marion	Indiana	46224	333-44-5555	10

Below the results, a message bar indicates: 'Query executed successfully.' and shows connection details: ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows.

Figure 6.7 - Physician Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like System Tables, FileTables, Graph Tables, and various dbo.\* tables such as Appointment, AuditPatient, Consent, EmergencyContact, Insurance, MedicalHistory, Medication, Patient, and Physician. The Physician table is selected, and its columns (PhysicianID, FirstName, LastName, Specialty, Phone, Email) are visible. The Results tab on the right displays a query that concatenates '317-' to the phone numbers of physicians whose phones do not already start with '317-'. The output shows 10 rows of physician data.

```

USE TestUniversalHealth
SELECT * FROM Physician

UPDATE Physician
SET Phone = CONCAT('317-', Phone)
WHERE Phone NOT LIKE '317-%';

```

PhysicianID	FirstName	LastName	Specialty	Phone	Email
1	Alice	Carter	Cardiology	317-555-3010	alice.c@uh-hospital.com
2	Bob	Wilson	General	317-555-3020	bob.w@uh-hospital.com
3	Clara	Evans	Neurology	317-555-3030	clara.e@uh-hospital.com
4	Henry	Tumer	Pediatrics	317-555-3040	henry.t@uh-hospital.com
5	Olivia	Green	Oncology	317-555-3050	olivia.g@uh-hospital.com
6	Ethan	White	Orthopedics	317-555-3060	ethan.w@uh-hospital.com
7	Grace	Lee	Dermatology	317-555-3070	grace.l@uh-hospital.com
8	Daniel	Moore	ENT	317-555-3080	daniel.m@uh-hospital.com
9	Sophia	King	Psychiatry	317-555-3090	sophia.k@uh-hospital.com
10	Charles	Scott	Gastroenterology	317-555-3100	charles.s@uh-hospital.com

Figure 6.8 - Prescription Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like System Tables, FileTables, Graph Tables, and various dbo.\* tables such as Appointment, AuditPatient, Consent, EmergencyContact, Insurance, MedicalHistory, Medication, Patient, Physician, and Prescription. The Prescription table is selected, and its columns (PrescriptionID, DatePrescribed, DosageInstructions) are visible. The Results tab on the right displays a query that selects the first 10 prescriptions from the Prescription table. The output shows 10 rows of prescription data.

```

USE TestUniversalHealth
SELECT PrescriptionID, DatePrescribed, DosageInstructions FROM Prescription

```

PrescriptionID	DatePrescribed	DosageInstructions
1	2025-06-01	Take 1 tablet daily after meals.
2	2025-06-02	Take 2 tablets daily with meals.
3	2025-06-03	Take once daily in the morning.
4	2025-06-04	Take every 8 hours as needed.
5	2025-06-05	Complete full course, twice a day.
6	2025-06-06	Take 1 tablet at bedtime.
7	2025-06-07	Take before breakfast.
8	2025-06-08	Take one capsule daily.
9	2025-06-09	Inhale two puffs every 4 hours.
10	2025-06-10	Take once daily in the evening.

Figure 6.9 - Procedures Schema with Sample Data

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including Tables, Views, and Procedures. The Procedures node under Tables is expanded, showing columns: ProcedureID, RecordID, Name, Description, and Cost. A query window titled 'SQLQuery1.sql' is open, displaying the following T-SQL code:

```
USE TestUniversalHealth
SELECT * FROM Procedures
```

The results grid shows the following data:

	ProcedureID	RecordID	Name	Description	Cost
1	1	1	ECG	Check electrical heart activity.	120.00
2	2	2	HbA1c Test	Measure average blood sugar.	65.00
3	3	3	Spirometry	Evaluate lung function.	100.00
4	4	4	Allergy Test	Determine allergen sensitivities.	150.00
5	5	5	Mastectomy	Breast removal surgery.	5000.00
6	6	6	Knee MRI	Imaging for joint issues.	750.00
7	7	7	Skin Biopsy	Check for skin conditions.	200.00
8	8	8	Tonsillectomy	Remove tonsils.	2500.00
9	9	9	Psych Eval	Comprehensive mental health assessment.	300.00
10	10	10	Colonoscopy	Examine large intestine.	950.00

At the bottom of the results grid, a message states 'Query executed successfully.' The status bar at the bottom right shows 'ADE-PC\SQLEXPRESS (15.0 RTM) | ADE-PC\Kukis (65) | TestUniversalHealth | 00:00:00 | 10 rows'.

**REFERENCES**

- Kendall, E. K and Kendall, E. J (2020). *Systems Analysis and Design 10<sup>th</sup> Edition.* (pp. 409-439). Pearson Education Limited.