

Design and Implementation of University Medical Center Database

CSE 581 - Intro to Database Management Systems

Submitted By Sai Kumar Palevela

Abstract - Healthcare databases are most frequently developed for the purpose of assessing the quality of healthcare, often for a specific disease or within a specific healthcare delivery system.

Accurate and comprehensive healthcare data are vitally important for a variety of purposes. These data may be used for local assessments or evaluations within a healthcare system, such as for specific outpatient conditions or inpatient hospital events. The data may also be used regionally or nationally for assessing performance within or across healthcare systems. Also, while comparisons become enormously difficult, administrative data may be used for comparing across national boundaries, to assess international differences in healthcare and disease.

In this project, we are going to design, implement a database keeping University medical Department in mind. But the principles and standards that we followed can be used to design another database of similar use case.

The implementation of the project includes creating an efficient database which can be used as backend storage for a graphical interface like web application by creating relevant tables. For making this project the efficient one I have used all the available database concepts like View, Stored Procedures, Triggers, Server Roles, Security, Encryption and User Defined Functions. Database design of my project is taken care by using Entity Relationship diagrams and by normalizing it to improve efficiency and reducing redundancy. Even though the main aim for designing is to track the medical facilities provided by university. The main goal of this project is to create a real-life environment which depicts the above model addressing the above business problems.

SAI KUMAR PALEVELA

SUID – 666053899

Contents

1) Database Name:	3
2) Design phase:	3
a) Tables Used	3
b) E/R Diagram:	6
c) Potential Integrity and Security issues	6
3) Implementation Phase	7
a) constraints	7
1. Normal Constraints:	7
2. Business Constraints	7
b) Database Normalization:	10
c) Tables Creation	11
4) Testing Phase:	22
a) Populating the database with Test information:	22
b) Views:	32
View 1:	32
View 2:	33
View 3:	34
VIEW 4:	35
c) Stored Procedures	36
Stored Procedure 1:	37
Stored Procedure 2:	38
d) Functions:	39
First Function	39
Second Function	40
e) Scripts	41
f) Triggers	42
Trigger1	42
Trigger 2	44
5) Conclusion:	46

1) Database Name:

University Medical Center Database

This total Project is mainly divided in three parts.

- Design
- Implementation
- Testing

2) Design phase:

a) Tables Used

1	Patients	
	Patient ID	PK
	Patient First Name	
	Patient Last Name	
	Age	
	Phone Number	
	State	
	City	
	Medical Staff ID	FK
2	MedicalStaff	
	MedicalStaff ID	PK
	Medical Staff First name	
	Medical Staff Last Name	
	Address	
	Phone Number	
	Speciality ID	FK
	Hospital ID	FK
	Staff Type	FK
3	StaffType	
	StaffType	PK
	Name	
4	Speciality	
	SpecialityID	PK
	Designation	
5	Hospitals	
	Hospital ID	PK

	Hospital Name	
	Address	
6	Room Type	
	RoomTypeID	PK
	Name	
7	Department	
	Department ID	PK
	Name	
	Description	
8	Rooms	
	RoomID	PK
	Room Capacity	
	DepartmentID	FK
	RoomType ID	FK
	Hospital ID	FK
	Price	
9	MedicalEquipments	
	MedicalEquipmentID	PK
	Equipment Name	
	Description	
10	RoomEquipment	
	RoomEquipment ID	PK
	Equipment ID	FK
	Room ID	FK
11	ScheduleType	
	ScheduleTypeID	PK
	Name	
	Description	
12	MedicalStaffSchedule	
	Schedule ID	
	Medical Staff ID	
	Schedule Type ID	
	Start Date Time	
	End Date Time	
	Description	
13	Appointments	
	Appointment ID	PK
	Patient ID	FK

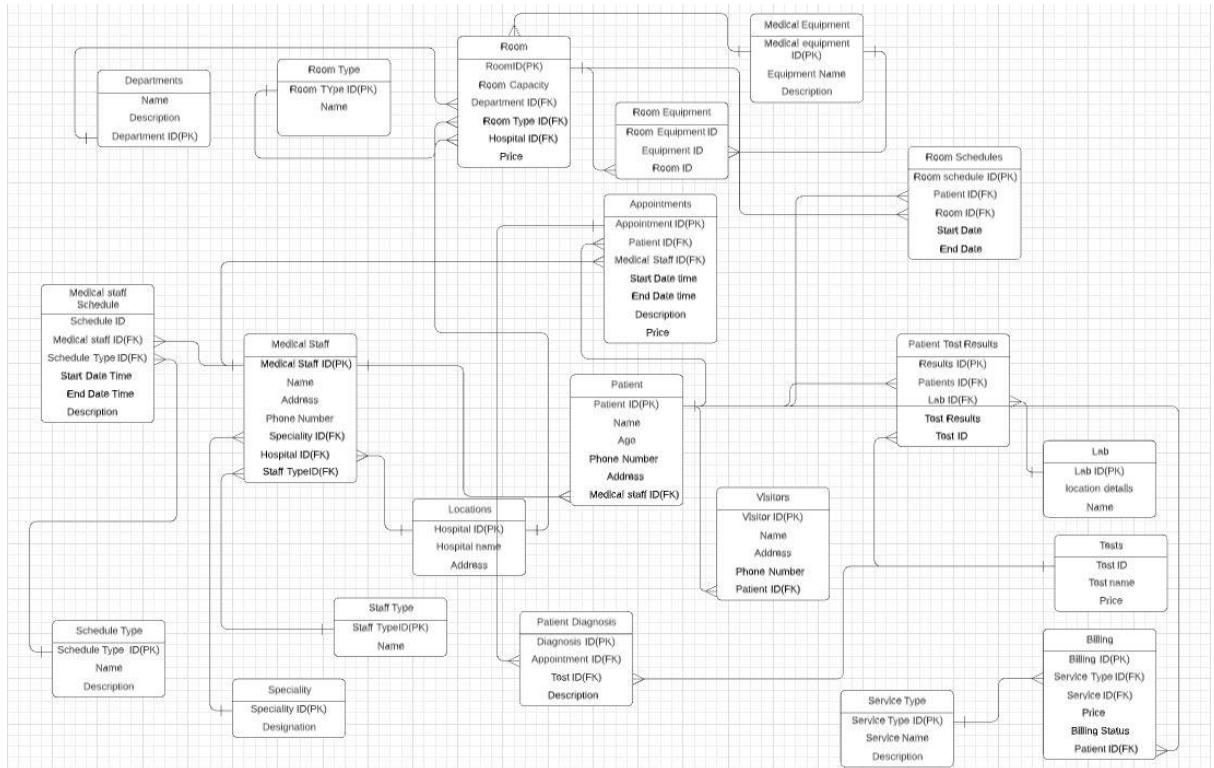
	Medical Staff ID	Fk
	Start Date time	
	End Date time	
	Description	
	Price	
14	RoomSchedules	
	Room Schedule ID	PK
	Patient ID	FK
	Room ID	FK
	Start Date	
	End Date	
15	Labs	
	Lab ID	PK
	Name	
	Location details	
16	Tests	
	Test ID	PK
	Test Name	
	Price	
17	PatientsTestResults	
	Results ID	PK
	Patients	
	Lab ID	FK
	Test Results	
	Test ID	FK
18	Visitors	
	Visitor ID	PK
	Visitor First Name	
	Visitor Last Name	
	State	
	City	
	Phone Number	
	Patient ID	FK
19	PatientDiagnosis	
	Diagnosis ID	PK
	Test ID	FK
	Description	
20	Billings	
	Billing ID	PK

	Service Type ID	FK
	Service ID	FK
	Price	
	Billing Status	
	Patient ID	FK
21	ServiceTypes	
	ServiceType ID	PK
	Service Name	
	Description	

b) E/R Diagram:

The Entity Relationship(E/R) diagram mainly focusses on how entities in the database are related to each other and how to link them.

Here is the E/R diagram given below that is used for my project.



c) Potential Integrity and Security issues

The potential security and integrity can be reduced by transforming the above database into the above-mentioned format of creating more tables i.e. Normalizing it.

For referential integrity to hold in a relational database, any column in a base table that is declared a foreign key can only contain either null values or values from a parent table's primary key or a candidate key. In other words when a foreign key value is used it must reference a valid, existing primary key in the parent table. For instance, deleting a record that contains a value referred to by a foreign key in another table would break referential integrity.

3) Implementation Phase

a) constraints

Typically, there are two types of constraints. They are:

1. **Normal Constraints:** - Normal constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted. Various types of normal constraints are: NOT NULL, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT, UNIQUE, INDEX.
2. **Business Constraints:** They include the relationships between the tables such as One-to-One, Many-to-One, Many-to-One.

In this project I have used One-to-One relations, Many-to-One relations, Many-to-Many relations.

Primary keys (PK), data types, nullable and relationships that are used in my project are listed.

1) Patients Table:

```
PatientID int IDENTITY PRIMARY KEY NOT NULL,  
PatientFirstName varchar (50) NOT NULL,  
PatientLastName varchar (50) NOT NULL,  
Age int NOT NULL,  
City varchar (50) NOT NULL,  
State char (2) NOT NULL,  
FOREIGN KEY(MedicalStaffID) REFERENCES MedicalStaff (MedicalStaffID)
```

2) MedicalStaff Table:

```
MedicalStaffID int IDENTITY PRIMARY KEY NOT NULL,  
MedicalStaffFirstName varchar (50) NOT NULL,  
MedicalStaffLastName varchar (50) NOT NULL,  
Address varchar (50) NOT NULL,  
PhoneNumber varchar (50) NULL  
FOREIGN KEY(SpecialityID) REFERENCES Speciality(SpecialityID)  
FOREIGN KEY(HospitalID) REFERENCES Hospitals(HospitalID)  
FOREIGN KEY(StaffTypeID) REFERENCES StaffType(StaffTypeID)
```

3) StaffType Table:

StaffTypeID int NOT NULL IDENTITY PRIMARY KEY,
Name varchar (50) NOT NULL

4) Specialities Table:

SpecialityID int NOT NULL IDENTITY PRIMARY KEY,
Designation varchar (20) NOT NULL

5) Hospitals Table:

HospitalID int NOT NULL IDENTITY PRIMARY KEY,
Name varchar (50) NOT NULL,
Description varchar (50),

6) Room Types Table

RoomTypeID int NOT NULL IDENTITY PRIMARY KEY,
Name varchar(50) NOT NULL,

7) Departments Table

DepartmentID INT IDENTITY PRIMARY KEY NOT NULL,
Name varchar(50) NOT NULL,
Description varchar(50)

8) Rooms Table

RoomID INT IDENTITY PRIMARY KEY NOT NULL,
RoomCapacity varchar (20) NOT NULL,
FOREIGN KEY(DepartmentID) REFERENCES Departments(DepartmentID)
FOREIGN KEY(RoomTypeID) REFERENCES RoomType(RoomTypeID)
FOREIGN KEY(HospitalID) REFERENCES Hospitals(HospitalID)

9) MedicalEquipments Table

MedicalEquipmentID INT IDENTITY PRIMARY KEY NOT NULL,
EquipmentName varchar(50) NOT NULL,
Descriptions varchar(50)

10) RoomEquipments Table

RoomEquipmentID INT NOT NULL IDENTITY PRIMARY KEY,
FOREIGN KEY(EquipmentID) REFERENCES MedicalEquipments(MedicalEquipmentID)
FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)

11) ScheduleTypes Table

ScheduleTypeID INT NOT NULL IDENTITY PRIMARY KEY,

Name varchar(50) NOT NULL,
Description varchar(50)

12) MedicalStaffSchedules Table

ScheduleID INT NOT NULL IDENTITY PRIMARY KEY,
StartTime datetime,
EndTime datetime,
Descriptions varchar(50),
FOREIGN KEY(MedicalStaffID) REFERENCES MedicalStaff(MedicalStaffID)
FOREIGN KEY(ScheduleTypeID) REFERENCES ScheduleTypes(ScheduleTypeID)

13) Appointments Table

AppointmentID INT NOT NULL IDENTITY PRIMARY KEY,
StartTime datetime,
EndTime datetime,
Description varchar(50),
Price int,
FOREIGN KEY(PatientID) REFERENCES Patients(PatientID)
FOREIGN KEY(MedicalStaffID) REFERENCES MedicalStaff(MedicalStaffID)

14) RoomSchedules Table

RoomScheduleID INT NOT NULL IDENTITY PRIMARY KEY,
StartTime datetime,
EndTime datetime,
FOREIGN KEY(PatientID) REFERENCES Patients(PatientID)
FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)

15) Labs Table

LabID INT NOT NULL IDENTITY PRIMARY KEY,
LabName varchar(50) NOT NULL,
LocationDetails varchar(50),

16) Tests Table

TestID INT NOT NULL IDENTITY PRIMARY KEY,
TestName varchar(50) NOTNULL,
Price int

17) PatientsTestResults Table

TestResultID INT NOT NULL IDENTITY PRIMARY KEY,
TestResults varchar(50) NOTNULL,

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
FOREIGN KEY (TestID) REFERENCES Tests(TestID),
FOREIGN KEY (LabID) REFERENCES Labs(LabID)

18) Visitors Table

VisitorID int IDENTITY PRIMARY KEY NOT NULL,
VisitorFirstName varchar (50) NOT NULL,
VisitorLastName varchar (50) NOT NULL,
City varchar (50) NOT NULL,
State char (2) NOT NULL,
PhoneNumber varchar(20),
FOREIGN KEY(PatientID) REFERENCES Patients (PatientID)

19) PatientDiagnosis Table

DiagnosisID int IDENTITY PRIMARY KEY NOT NULL,
Description varchar (50) NOT NULL,
FOREIGN KEY(TestID) REFERENCES Tests (TestID)

20) Billings Table

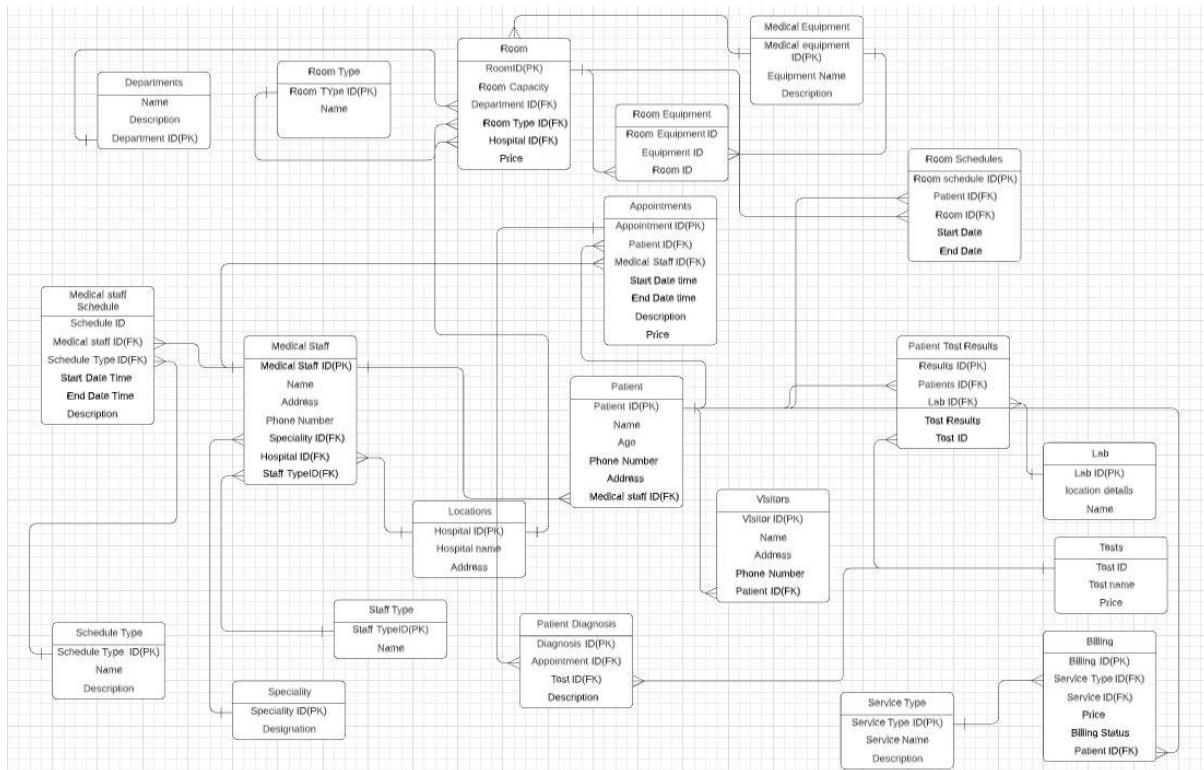
Billing ID int IDENTITY PRIMARY KEY NOT NULL,
Price Int,
BillingStatus int NOTNULL,
Service ID int,
FOREIGN KEY(PatientID) REFERENCES Patients (PatientID),
FOREIGN KEY(ServiceTypeID) REFERENCES ServiceTypes (ServiceTypeID)

21) ServiceTypes Table

ServiceTypeID int IDENTITY PRIMARY KEY NOT NULL,
ServiceName varchar(50),
Description Varchar(50)

b) Database Normalization:

For a database design to be in 3rd Normal form, it must be in First Normal Form i.e. it must not contain any non-repeating columns and should contain scalar values in all rows, and it must also be in 2nd Normal form (i.e. no partial dependencies) and all the columns must only depend on the primary key (i.e. no transitive dependencies). So here is my database design in 3rd Normal form given below:



c) Tables Creation

Below screenshots shows the creation of all the tables present in the database

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.master (DESKTOP-FN9TC92\sai (5B)) - Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, Query, Project, Tools, Window, Help. The toolbar has icons for New Query, Execute, Save, Undo, Redo, Copy, Paste, Find, Replace, and others. The Object Explorer on the left shows the database structure under "DESKTOP-FN9TC92\SQLSERVER2019 (SQL Server 15)". The main pane displays a query window titled "SQLQuery1.sql - DE...FN9TC92\sai (5B)" with the following T-SQL code:

```
IF DB_ID('UniversityMedicalDatabase') IS NOT NULL
    DROP DATABASE UniversityMedicalDatabase
GO

CREATE DATABASE UniversityMedicalDatabase
GO

--Sai Kumar Palevala
```

The status bar at the bottom indicates "96 %", "Query executed successfully.", and "Completion time: 2021-08-20T17:39:28.8542020+05:00".

Below screenshot shows the creation of “**StaffTypes**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a T-SQL script for creating the 'StaffTypes' table:

```
USE UniversityMedicalDatabase
CREATE TABLE StaffTypes(
    StaffTypeID int NOT NULL IDENTITY PRIMARY KEY,
    Name varchar (50) NOT NULL
)
```

The 'Messages' pane at the bottom right indicates that the command completed successfully with a completion time of 2021-08-20T17:44:23.9221893+05:30.

Below screenshot shows the creation of “**Specialities**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a T-SQL script for creating the 'Specialities' table:

```
USE UniversityMedicalDatabase
CREATE TABLE Specialities(
    SpecialityID int NOT NULL IDENTITY PRIMARY KEY,
    Designation varchar (20) NOT NULL
)
-- Sai Kumar Palevela
```

The 'Messages' pane at the bottom right indicates that the command completed successfully with a completion time of 2021-08-20T17:47:12.3120128+05:30.

Below screenshot shows the creation of “**Hospitals**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with various databases listed, including 'UniversityMedicalDatabase'. The right pane contains a query window titled 'SQLQuery1.sql' with the following SQL code:

```
USE UniversityMedicalDatabase
CREATE TABLE Hospitals(
    HospitalID int NOT NULL IDENTITY PRIMARY KEY,
    Name varchar (50) NOT NULL,
    Description varchar (50)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates '96 %' completion and 'Query executed successfully.' The bottom right corner shows the session details: 'DESKTOP-FN9TC92\SQLSERVER2019\sa (58)' and 'UniversityMedicalDatabase'.

Below screenshot shows the creation of “**MedicalStaff**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central pane displays a T-SQL script for creating the `MedicalStaff` table:

```

USE UniversityMedicalDatabase
CREATE TABLE MedicalStaff(
    MedicalStaffID int IDENTITY PRIMARY KEY NOT NULL,
    MedicalStaffFirstName varchar (50) NOT NULL,
    MedicalStaffLastName varchar (50) NOT NULL,
    MedicalStaffAddress varchar (50) NOT NULL,
    PhoneNumber varchar (50) NULL,
    SpecialityID Int NOT NULL,
    HospitalID Int NOT NULL,
    StaffTypeID Int NOT NULL,
    FOREIGN KEY(SpecialityID) REFERENCES Specialities(SpecialityID),
    FOREIGN KEY(HospitalID) REFERENCES Hospitals(HospitalID),
    FOREIGN KEY(StaffTypeID) REFERENCES StaffTypes(StaffTypeID)
)
-- Sai Kumar Palevela

```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-06-20T17:59:22.5640588+05:30".

Below screenshot shows the creation of “**Patients**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central pane displays a T-SQL script for creating the `Patients` table:

```

USE UniversityMedicalDatabase
CREATE TABLE Patients(
    PatientID int IDENTITY PRIMARY KEY NOT NULL,
    PatientFirstName varchar (50) NOT NULL,
    PatientLastName varchar (50) NOT NULL,
    Age int NOT NULL,
    City varchar (50) NOT NULL,
    State char (2) NOT NULL,
    MedicalStaffID Int NOT NULL,
    FOREIGN KEY(MedicalStaffID) REFERENCES MedicalStaff(MedicalStaffID)
)
-- Sai Kumar Palevela

```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-06-20T18:01:53.5964760+05:30".

Below screenshot shows the creation of “**RoomTypes**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a T-SQL script for creating a table named RoomTypes:

```
--USE UniversityMedicalDatabase
--CREATE TABLE RoomTypes(
RoomTypeID int NOT NULL IDENTITY PRIMARY KEY,
Name varchar(50) NOT NULL
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-05-20T18:03:57.3888883+05:30".

Below screenshot shows the creation of “**Departments**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a T-SQL script for creating a table named Departments:

```
--USE UniversityMedicalDatabase
CREATE TABLE Departments(
DepartmentID INT IDENTITY PRIMARY KEY NOT NULL,
Name varchar(50) NOT NULL,
Description varchar(50)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2021-05-20T18:05:03.5295888+05:30".

Below screenshot shows the creation of “Rooms” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a T-SQL script for creating the 'Rooms' table:

```
CREATE TABLE Rooms(
    RoomID INT IDENTITY PRIMARY KEY NOT NULL,
    RoomCapacity varchar(20) NOT NULL,
    DepartmentID Int NOT NULL,
    RoomTypeID Int NOT NULL,
    HospitalID Int NOT NULL,
    FOREIGN KEY(DepartmentID) REFERENCES Departments(DepartmentID),
    FOREIGN KEY(RoomTypeID) REFERENCES RoomTypes(RoomTypeID),
    FOREIGN KEY(HospitalID) REFERENCES Hospitals(HospitalID)
)
-- Sai Kumar Paleleva
```

The 'Messages' pane at the bottom right shows the command completed successfully with a completion time of 2021-06-20T18:08:11.9008816+05:30.

Below screenshot shows the creation of “MedicalEquipments” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a T-SQL script for creating the 'MedicalEquipments' table:

```
CREATE TABLE MedicalEquipments(
    MedicalEquipmentID INT IDENTITY PRIMARY KEY NOT NULL,
    EquipmentName varchar(50) NOT NULL,
    Descriptions varchar(50)
)
-- Sai Kumar Paleleva
```

The 'Messages' pane at the bottom right shows the command completed successfully with a completion time of 2021-06-20T18:10:18.0292888+05:30.

Below screenshot shows the creation of “RoomEquipments” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a SQL query window with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE RoomEquipments(
    RoomEquipmentID INT NOT NULL IDENTITY PRIMARY KEY,
    EquipmentID INT NOT NULL,
    RoomID INT NOT NULL,
    FOREIGN KEY(EquipmentID) REFERENCES MedicalEquipments(MedicalEquipmentID),
    FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)
)
-- Sai Kumar Palevela
```

The Messages pane below the query window shows the command completed successfully with a completion time of 2021-05-20T18:12:35.4919888+05:30. The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Below screenshot shows the creation of “ScheduleTypes” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a SQL query window with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE ScheduleTypes(
    ScheduleTypeID INT NOT NULL IDENTITY PRIMARY KEY,
    Name varchar(50) NOT NULL,
    Description varchar(50)
)
-- Sai Kumar Palevela
```

The Messages pane below the query window shows the command completed successfully with a completion time of 2021-05-20T19:14:05.2876862+05:30. The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Below screenshot shows the creation of “**MedicalStaffSchedules**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central pane displays a T-SQL script for creating the **MedicalStaffSchedules** table:

```
USE UniversityMedicalDatabase
CREATE TABLE MedicalStaffSchedules(
    ScheduleID INT NOT NULL IDENTITY PRIMARY KEY,
    StartTime datetime,
    EndTime datetime,
    Description varchar(50),
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Below screenshot shows the creation of “**Appointments**” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central pane displays a T-SQL script for creating the **Appointments** table:

```
USE UniversityMedicalDatabase
CREATE TABLE Appointments(
    AppointID INT NOT NULL IDENTITY PRIMARY KEY,
    StartDateTime datetime,
    EndDateTime datetime,
    Description varchar(50),
    Price int,
    PatientID Int NOT NULL,
    MedicalStaffID Int NOT NULL,
    FOREIGN KEY(PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY(MedicalStaffID) REFERENCES MedicalStaff(MedicalStaffID)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Below screenshot shows the creation of “RoomSchedules” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a SQL query window titled 'SQLQuery1.sql' with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE RoomSchedules(
    RoomScheduleID INT NOT NULL IDENTITY PRIMARY KEY,
    StartDateTime datetime,
    EndDateTime datetime,
    PatientID Int NOT NULL,
    RoomID Int NOT NULL,
    FOREIGN KEY(PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY(RoomID) REFERENCES Rooms(RoomID)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2021-06-20T18:20:11.729393+05:30.

Below screenshot shows the creation of “Labs” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including UniversityMedicalDatabase. The central pane displays a SQL query window titled 'SQLQuery1.sql' with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE Labs(
    LabID INT NOT NULL IDENTITY PRIMARY KEY,
    LabName varchar(50) NOT NULL,
    LocationDetails varchar(50),
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2021-06-20T18:21:36.1123813+05:30.

Below screenshot shows the creation of “Tests” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the 'UniversityMedicalDatabase' selected. The right pane contains a query window titled 'SQLQuery1.sql'. The query is:`USE UniversityMedicalDatabase
CREATE TABLE Tests(
 TestID INT NOT NULL IDENTITY PRIMARY KEY,
 TestName varchar(50) NOT NULL,
 Price int
)
-- Sai Kumar Palevela`

Below the query window, the 'Messages' pane shows the output:

```
Commands completed successfully.
Completion time: 2021-05-20T18:28:21.5308331+05:30
```

At the bottom, a status bar indicates 'Query executed successfully.' and '0 rows'.

Below screenshot shows the creation of “PatientTestResults” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the 'UniversityMedicalDatabase' selected. The right pane contains a query window titled 'SQLQuery1.sql'. The query is:`USE UniversityMedicalDatabase
CREATE TABLE PatientTestResults(
 TestResultID INT NOT NULL IDENTITY PRIMARY KEY,
 TestResults varchar(50) NOT NULL,
 PatientID int NOT NULL,
 TestID int NOT NULL,
 LabID int NOT NULL,
 FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
 FOREIGN KEY (TestID) REFERENCES Tests(TestID),
 FOREIGN KEY (LabID) REFERENCES Labs(LabID)
)
-- Sai Kumar Palevela`

Below the query window, the 'Messages' pane shows the output:

```
Commands completed successfully.
Completion time: 2021-05-20T18:28:30.9221214+05:30
```

At the bottom, a status bar indicates 'Query executed successfully.' and '0 rows'.

Below screenshot shows the creation of “Visitors” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a SQL query window titled 'SQLQuery1.sql' with the following code:

```
CREATE TABLE Visitors(
    VisitorID int IDENTITY PRIMARY KEY NOT NULL,
    VisitorFirstName varchar (50) NOT NULL,
    VisitorLastName varchar (50) NOT NULL,
    City varchar (50) NOT NULL,
    State char (2) NOT NULL,
    PhoneNumber varchar(20),
    PatientID int NOT NULL,
    FOREIGN KEY(PatientID) REFERENCES Patients (PatientID)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

Below screenshot shows the creation of “Diagnosis” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'UniversityMedicalDatabase'. The central pane displays a SQL query window titled 'SQLQuery1.sql' with the following code:

```
CREATE TABLE Diagnosis(
    DiagnosisID int IDENTITY PRIMARY KEY NOT NULL,
    Description varchar (50) NOT NULL,
    TestID int NOT NULL,
    FOREIGN KEY(TestID) REFERENCES Tests (TestID)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

Below screenshot shows the creation of “ServiceTypes” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including the current one, UniversityMedicalDatabase. The central pane displays a SQL query window with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE ServiceTypes(
    ServiceTypeID int IDENTITY PRIMARY KEY NOT NULL,
    ServiceName varchar(50),
    Descriptions Varchar(50)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and shows the completion time as 2021-05-20T18:32:51.6492391+05:30.

Below screenshot shows the creation of “Billings” Table.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various databases, including the current one, UniversityMedicalDatabase. The central pane displays a SQL query window with the following code:

```
USE UniversityMedicalDatabase
CREATE TABLE Billings(
    BillingID int IDENTITY PRIMARY KEY NOT NULL,
    Price Int,
    BillingStatus int NOT NULL,
    ServiceID int,
    PatientID Int NOT NULL,
    ServiceTypeID int Not NULL,
    FOREIGN KEY(PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY(ServiceTypeID) REFERENCES ServiceTypes(ServiceTypeID)
)
-- Sai Kumar Palevela
```

The status bar at the bottom indicates "Query executed successfully." and shows the completion time as 2021-05-20T18:39:12.4464462+05:30.

4) Testing Phase:

a) Populating the database with Test information:

Here I have created all the tables with the constraints and relations mentioned above in the implementation phase.

Here is a screenshot of showing all the tables that I have created in my database.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like StaffTypes, Specialities, and RoomTypes. The central pane displays a T-SQL script for creating the StaffTypes table:

```
USE UniversityMedicalDatabase
SET IDENTITY_INSERT StaffTypes ON
INSERT StaffTypes(StaffTypeID, Name) VALUES
(1, 'Doctor'),
(2, 'Nurse'),
(3, 'Junior Doctor')
SET IDENTITY_INSERT StaffTypes OFF
-- Sai Kumar Palevela
```

The Messages pane at the bottom indicates "0 rows affected" and "Completion time: 2021-05-20T19:34:02.5272484+05:30". The status bar at the bottom right shows "Query executed successfully.", "DESKTOP-FN9TC92\SQLSERVER2019\sa (58)", "UniversityMedicalDatabase", "00:00:00", and "0 rows".

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including tables like StaffTypes, Specialities, and RoomTypes. The central pane displays a T-SQL script for creating the Specialities table:

```
USE UniversityMedicalDatabase
SET IDENTITY_INSERT Specialities ON
INSERT Specialities(SpecialityID,Designation) VALUES
(1, 'Cardiologist'),
(2, 'Dentist'),
(3, 'Pulmonologist')
SET IDENTITY_INSERT Specialities OFF
-- Sai Kumar Palevela
```

The Messages pane at the bottom indicates "0 rows affected" and "Completion time: 2021-05-20T19:39:53.0000009+05:30". The status bar at the bottom right shows "Query executed successfully.", "DESKTOP-FN9TC92\SQLSERVER2019\sa (58)", "UniversityMedicalDatabase", "00:00:00", and "0 rows".

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
--USE UniversityMedicalDatabase
--SET IDENTITY_INSERT Hospitals ON
--INSERT Hospitals(HospitalID,Name,Description) VALUES
--(1, 'Mayo Clinic', 'A world-renowned medical center'),
--(2, 'Apollo', 'A modern hospital'),
--(3, 'Medicity', 'A 24-hour emergency facility')
--SET IDENTITY_INSERT Hospitals OFF
-- Sai Kumar Palevela
```

96 % Messages

(3 rows affected)

Completion time: 2021-05-20T19:43:33.4290780+05:30

96 % Messages

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sai (58) UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
--USE UniversityMedicalDatabase
--SET IDENTITY_INSERT MedicalStaff ON
--INSERT MedicalStaff(MedicalStaffID,MedicalStaffFirstName,MedicalStaffLastName,MedicalStaffAddress,PhoneNumber,SpecialityID,HospitalID,StaffTypeID) VALUES
--(1, 'James', 'Smith', 'SouthLake', '(510) 349-5056', 1, 2, 3),
--(2, 'Maria', 'Rodriguez', 'South San Frisco', '(315) 447-9825', 2, 3, 1),
--(3, 'Marry', 'Smith', 'Seattle', '(514) 223-5643', 3, 1, 2)
--SET IDENTITY_INSERT MedicalStaff OFF
-- Sai Kumar Palevela
```

96 % Messages

(3 rows affected)

Completion time: 2021-05-20T19:56:38.0044011+05:30

96 % Messages

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sai (58) UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Object Explorer Execute

UniversityMedicalDatabase

Object Explorer

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58))

```
-- USE UniversityMedicalDatabase
-- SET IDENTITY_INSERT Patients ON
-- INSERT Patients(PatientID, PatientFirstName, PatientLastName, Age, City, State, MedicalStaffID) VALUES
-- (1, 'John', 'Doe', 25, 'New York', 'NY', 1),
-- (2, 'James', 'Johnson', 30, 'Albany', 'NY', 2),
-- (3, 'Robert', 'Andrew', 45, 'San Diego', 'CA', 3)
-- SET IDENTITY_INSERT Patients OFF
-- -- Sai Kumar Palevela
```

96 % Messages

(3 rows affected)

Completion time: 2021-06-20T20:06:18.8187094+05:30

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query Object Explorer Execute

UniversityMedicalDatabase

Object Explorer

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58))

```
-- USE UniversityMedicalDatabase
-- SET IDENTITY_INSERT RoomTypes ON
-- INSERT RoomTypes(RoomTypeID, Name) VALUES
-- (1, 'ICU'),
-- (2, 'General Room'),
-- (3, 'Emergency Room')
-- SET IDENTITY_INSERT RoomTypes OFF
-- -- Sai Kumar Palevela
```

96 % Messages

(3 rows affected)

Completion time: 2021-06-20T20:18:39.0397988+05:30

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
--USE UniversityMedicalDatabase
SET IDENTITY_INSERT Departments ON
INSERT Departments(DepartmentID,Name,Description) VALUES
(1, 'Administration', ''),
(2, 'In Patient', 'W'),
(3, 'Nursing', 'O')
SET IDENTITY_INSERT Departments OFF
-- Sai Kumar Paleleva
```

96 % Messages

(3 rows affected)

Completion time: 2021-08-20T20:18:49.9838814+05:30

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
--USE UniversityMedicalDatabase
SET IDENTITY_INSERT Rooms ON
INSERT Rooms(RoomID,RoomCapacity,DepartmentID,RoomTypeID,HospitalID) VALUES
(1, 4, 1, 2, 3),
(2, 3, 3, 1, 2),
(3, 5, 2, 3, 1)
SET IDENTITY_INSERT Rooms OFF
-- Sai Kumar Paleleva
```

96 % Messages

(3 rows affected)

Completion time: 2021-08-20T20:22:11.9446817+05:30

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sa1 (58)) - Microsoft SQL Server Management Studio

```
-- USE UniversityMedicalDatabase
-- SET IDENTITY_INSERT MedicalEquipments ON
-- INSERT MedicalEquipments(MedicalEquipmentID,EquipmentName,Descriptions) VALUES
-- (1, 'Surgical Instruments', 'K'),
-- (2, 'Sterilizers', 'L'),
-- (3, 'Surgical Tablets', 'M')
-- SET IDENTITY_INSERT MedicalEquipments OFF
-- -- Sai Kumar Paleleva
```

96 %

Messages

(3 rows affected)

Completion time: 2021-08-20T20:27:11.3808882+05:00

96 %

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sa1 (58) UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sa1 (58)) - Microsoft SQL Server Management Studio

```
-- USE UniversityMedicalDatabase
-- SET IDENTITY_INSERT RoomEquipments ON
-- INSERT RoomEquipments(RoomEquipmentID,EquipmentID,RoomID) VALUES
-- (1,1,2),
-- (2,3,1),
-- (3,2,3)
-- SET IDENTITY_INSERT RoomEquipments OFF
-- -- Sai Kumar Paleleva
```

96 %

Messages

(3 rows affected)

Completion time: 2021-08-20T20:33:18.2234338+05:00

96 %

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sa1 (58) UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

Object Explorer

Connect ▾

UniversityMedicalDatabase

Databases

System Databases

Database Snapshots

AdventureWorks2016

AdventureWorksDW2016

AP

College

College2

Examples

MyGuitarShop

ProductOrders

UniversityMedicalDatabase

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Appointments

dbo.Billings

dbo.Departments

dbo.Diagnosis

dbo.Hospitals

dbo.Labs

dbo.MedicalEquipments

dbo.MedicalStaff

dbo.MedicalStaffSchedules

Columns

ScheduleID (PK, int, not null)

StartTime (datetime, null)

EndTime (datetime, null)

Descriptions (varchar(50), null)

Keys

Constraints

script.sql - DESKTOP_P-FN9TC92\sai (62) SQLQuery1.sql - DE...-FN9TC92\sai (58)*

```
-- USE UniversityMedicalDatabase
SET IDENTITY_INSERT MedicalStaffSchedules ON
INSERT MedicalStaffSchedules(ScheduleID,StartTime,EndTime,Descriptions) VALUES
(1,'2021-06-13 10:34:09','2021-06-13 14:34:09','A'),
(2,'2021-06-13 10:34:09','2021-06-13 14:34:09','B'),
(3,'2021-06-14 10:34:09','2021-06-14 14:34:09','C')
SET IDENTITY_INSERT MedicalStaffSchedules OFF
-- Sai Kumar Paleleva
```

96 %

Messages

(3 rows affected)

Completion time: 2021-06-20T22:07:14.5233289+05:00

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019 - DESKTOP-FN9TC92\sai (58) - UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

Object Explorer

Connect ▾

UniversityMedicalDatabase

Databases

System Databases

Database Snapshots

AdventureWorks2016

AdventureWorksDW2016

AP

College

College2

Examples

MyGuitarShop

ProductOrders

UniversityMedicalDatabase

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Appointments

dbo.Billings

dbo.Departments

dbo.Diagnosis

dbo.Hospitals

dbo.Labs

dbo.MedicalEquipments

dbo.MedicalStaff

dbo.MedicalStaffSchedules

Columns

ScheduleID (PK, int, not null)

StartTime (datetime, null)

EndTime (datetime, null)

Descriptions (varchar(50), null)

Keys

Constraints

script.sql - DESKTOP_P-FN9TC92\sai (62) SQLQuery1.sql - DE...-FN9TC92\sai (58)*

```
-- USE UniversityMedicalDatabase
SET IDENTITY_INSERT Appointments ON
INSERT Appointments(AppointmentID,StartTime,EndTime,Description,Price,PatientID,MedicalStaffID) VALUES
(1,'2021-06-04 10:34:09','2021-06-12 14:34:09','A',500,2,1),
(2,'2021-06-09 10:34:09','2021-06-13 14:34:09','B',1000,3,2),
(3,'2021-06-12 10:34:09','2021-06-14 14:34:09','C',1500,1,3)
SET IDENTITY_INSERT Appointments OFF
-- Sai Kumar Paleleva
```

96 %

Messages

(3 rows affected)

Completion time: 2021-06-20T22:20:22.4423334+05:00

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER2019 - DESKTOP-FN9TC92\sai (58) - UniversityMedicalDatabase 00:00:00 0 rows

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to 'DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase' (DESKTOP-FN9TC92\sai (58)). The Object Explorer sidebar shows the database structure, including 'UniversityMedicalDatabase' and its tables like 'RoomSchedules'. The main pane displays a query script named 'script.sql' (62 lines) which inserts data into the 'RoomSchedules' table. The execution results show 3 rows affected and a completion time of 2021-06-20T22:24:01.9035705+05:30. A status bar at the bottom shows 'Query executed successfully.'

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to DESKTOP-FN9TC92\SQLSERVER2019\UniversityMedicalDatabase (DESKTOP-FN9TC92\sa1 (58)) - Microsoft SQL Server Management Studio. The Object Explorer sidebar shows the database structure, including the UniversityMedicalDatabase with its tables like Labs, Appointments, Billings, Departments, Diagnosis, Hospitals, Labs, MedicalEquipments, MedicalStaff, and MedicalStaffSchedules. The main query editor window contains a script named 'script.sql' with the following content:

```
USE UniversityMedicalDatabase
SET IDENTITY_INSERT Labs ON
INSERT Labs(LabID,LabName,LocationDetails) VALUES
(1,'Blood Collection','First Floor'),
(2,'Testing','Ground Floor'),
(3,'Diagnostic','First Floor')
SET IDENTITY_INSERT Labs OFF
-- Sai Kumar Palevela
```

The results pane shows the execution completed successfully with 3 rows affected and a completion time of 2021-05-20T22:27:18.7055371+08:30.

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
USE UniversityMedicalDatabase
SET IDENTITY_INSERT Tests ON
INSERT Tests(TestID,TestName,Price)
VALUES
(1,'Sugar Test',50),
(2,'Blood',100),
(3,'CT Scan',150)
SET IDENTITY_INSERT Tests OFF
-- Sai Kumar Palevela
```

Object Explorer

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

UniversityMedicalDatabase

Databases System Databases Database Snapshots AdventureWorks2016 AdventureWorksDW2016 AP College College2 Examples MyGuitarShop ProductOrders UniversityMedicalDatabase Database Diagrams Tables System Tables FileTables External Tables Graph Tables dbo.Appointments dbo.Billings dbo.Department dbo.Diagnosis dbo.Hospitals dbo.Labs dbo.MedicalEquipments dbo.MedicalStaff dbo.MedicalStaffSchedules Columns Keys Constraints

Messages

(3 rows affected)

Completion time: 2021-05-20T22:50:04.7964159+05:30

96 %

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sai (58) UniversityMedicalDatabase 00:00:00 0 rows

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (58)) - Microsoft SQL Server Management Studio

```
USE UniversityMedicalDatabase
SET IDENTITY_INSERT PatientTestResults ON
INSERT PatientTestResults(TestResultID,TestResults,PatientID,TestID,LabID) VALUES
(1,'XYZ',1,2,3),
(2,'WXY',2,3,4),
(3,'EFG',3,1,2)
SET IDENTITY_INSERT PatientTestResults OFF
-- Sai Kumar Palevela
```

Object Explorer

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

UniversityMedicalDatabase

Databases System Databases Database Snapshots AdventureWorks2016 AdventureWorksDW2016 AP College College2 Examples MyGuitarShop ProductOrders UniversityMedicalDatabase Database Diagrams Tables System Tables FileTables External Tables Graph Tables dbo.Appointments dbo.Billings dbo.Department dbo.Diagnosis dbo.Hospitals dbo.Labs dbo.MedicalEquipments dbo.MedicalStaff dbo.MedicalStaffSchedules Columns Keys Constraints

Messages

(3 rows affected)

Completion time: 2021-05-20T22:53:42.4359964+05:30

96 %

Query executed successfully.

DESKTOP-FN9TC92\SQLSERVER20... DESKTOP-FN9TC92\sai (58) UniversityMedicalDatabase 00:00:00 0 rows

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists several databases, including the current one, UniversityMedicalDatabase. The central pane displays a T-SQL script named 'script.sql' which inserts three rows into the 'Visitors' table. The script uses the 'SET IDENTITY_INSERT' command to enable inserting values into the identity columns. The execution results in the Messages pane show a completion time of 2021-05-20T22:40:52.4382027+05:30 and 3 rows affected.

```
--USE UniversityMedicalDatabase  
SET IDENTITY_INSERT Visitors ON  
INSERT Visitors(VisitorID,VisitFirstNames,VisitLastName,City,State,PhoneNumber,PatientID) VALUES  
(1,'John','Doe','New York','NY','(212) 447-5059',1),  
(2,'Mark','Wilson','Albany','NY','(518) 992-4739',3),  
(3,'Maria','Adams','San Francisco','CA','(916) 332-4532',2)  
SET IDENTITY_INSERT Visitors OFF  
-- Sai Kumar Palevela
```

This screenshot shows the same Microsoft SQL Server Management Studio environment. A new script is being run, starting with 'SET IDENTITY_INSERT Diagnosis ON'. It inserts three rows into the 'Diagnosis' table with descriptions 'ABC', 'KLM', and 'XYZ'. The execution results in the Messages pane show a completion time of 2021-05-20T22:44:24.3513244+05:30 and 3 rows affected.

```
--USE UniversityMedicalDatabase  
SET IDENTITY_INSERT Diagnosis ON  
INSERT Diagnosis(DiagnosisID,Description,TestID) VALUES  
(1,'ABC',3),  
(2,'KLM',2),  
(3,'XYZ',1)  
SET IDENTITY_INSERT Diagnosis OFF  
-- Sai Kumar Palevela
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the UniversityMedicalDatabase. The central pane displays a T-SQL script named 'script.sql' which inserts three rows into the 'ServiceTypes' table. The script is as follows:

```

USE UniversityMedicalDatabase
SET IDENTITY_INSERT ServiceTypes ON
INSERT ServiceTypes(ServiceTypeID,ServiceName,Descriptions) VALUES
(1,'Appointment','ABC')
(2,'Room Booking','KLM')
(3,'Diagnosis','EFG')
SET IDENTITY_INSERT ServiceTypes OFF
-- Sai Kumar Paleleva

```

The execution results in the Messages pane show '3 rows affected' and a completion time of '2021-05-20T22:49:28.9493885+05:30'. A status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

This screenshot shows the same Microsoft SQL Server Management Studio environment. The Object Explorer and central pane are identical to the first screenshot. The script in the central pane is for inserting data into the 'Billings' table:

```

USE UniversityMedicalDatabase
SET IDENTITY_INSERT Billings ON
INSERT Billings(BillingID,Price,BillingStatus,ServiceID,PatientID,ServiceTypeID) VALUES
(1,2000,1,1,2,3),
(2,1000,0,2,3,1),
(3,1450,1,3,1,2)
SET IDENTITY_INSERT Billings OFF
-- Sai Kumar Paleleva

```

The execution results in the Messages pane show '3 rows affected' and a completion time of '2021-05-20T22:53:10.5847981+05:30'. A status bar at the bottom indicates 'Query executed successfully.' and '0 rows'.

b) Views:

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table, however the view itself does not store any data in it. The fields in a view are fields from one or more real tables in the database.

Here is the list of views that I have created and used in my project.

View 1:

Number of patients for each doctor

`CREATE VIEW vpatienttodoctor`

`AS`

`SELECT`

```

MS.MedicalStaffID,
MS.MedicalStaffFirstName AS Name,
Count(P.PatientID) AS Number_Of_Patients
FROM Patients P
INNER JOIN MedicalStaff MS on P.MedicalStaffID = MS.MedicalStaffID
GROUP BY MS.MedicalStaffID,
MS.MedicalStaffFirstName

```

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, under the 'UniversityMedicalDatabase' node, there is a new entry for the view 'vpatienttodoctor'. The 'Script' button next to it is highlighted. The script pane displays the T-SQL code for creating the view:

```

CREATE VIEW vpatienttodoctor
AS
SELECT
MS.MedicalStaffID,
MS.MedicalStaffFirstName AS Name,
Count(P.PatientID) AS Number_Of_Patients
FROM Patients P
INNER JOIN MedicalStaff MS on P.MedicalStaffID = MS.MedicalStaffID
GROUP BY MS.MedicalStaffID,
MS.MedicalStaffFirstName

```

In the Messages pane, it shows the command completed successfully with a completion time of 2021-05-21T08:27:18.4834120+05:30.

Below screenshot shows the result of the above created view

The screenshot shows the SQL Server Management Studio interface. The 'Script' button for the view 'vpatienttodoctor' is selected. The script pane contains the following T-SQL query:

```

SELECT * FROM vpatienttodoctor;

```

The Results pane displays the data returned by the view:

	MedicalStaffID	Name	Number_Of_Patients
1	1	James	1
2	2	Maria	1
3	3	Mary	1

The status bar at the bottom indicates 3 rows were returned.

View 2:

Total tests conducted in each lab.

CREATE VIEW vtestvslab

AS

```

SELECT
L.LabID,
L.LabName,
count(ptr.testresultid) as NumberOfTests
FROM Labs L
INNER JOIN PatientTestResults PTR ON L.LabID = PTR.LabID
GROUP BY L.LabID, L.LabName

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central Query window contains the following SQL code:

```

CREATE VIEW vttestslab
AS
SELECT
L.LabID,
L.LabName,
count(ptr.testresultid) as NumberOfTests
FROM Labs L
INNER JOIN PatientTestResults PTR ON L.LabID = PTR.LabID
GROUP BY L.LabID, L.LabName

```

The Results pane at the bottom right shows the output of the query:

```

96 %  Messages
Command completed successfully.
Completion time: 2021-05-21T09:35:29.2974800+05:30

```

A status bar at the bottom indicates "Query executed successfully."

Below screenshot shows the result of the above created view

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the UniversityMedicalDatabase. The central Query window contains the following SQL code:

```

SELECT * FROM vttestslab;

```

The Results pane at the bottom right shows the output of the query:

LabID	LabName	NumberOfTests
1	Blood Collection	1
2	Testing	1
3	Diagnostic	1

A status bar at the bottom indicates "Query executed successfully."

View 3:

Number of appointments each month at each hospital

CREATE VIEW vappointments

AS

```

SELECT
FORMAT(A.StartDateTime, 'MMMM') AS Month,
h.Name as HospitalName,
count(A.AppointmentId) as NumberOfAppointments FROM Appointments A
INNER JOIN MedicalStaff MS on A.MedicalStaffID = MS.MedicalStaffID
INNER JOIN Hospitals H ON MS.HospitalID = H.HospitalID
GROUP BY FORMAT(A.StartDateTime, 'MMMM'),
h.Name

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'UniversityMedicalDatabase' database is selected. In the center pane, a new query window titled 'SQLQuery1.sql' is open, containing the following T-SQL code:

```

CREATE VIEW vappointments
AS
SELECT
FORMAT(A.StartDateTime, 'MMMM') AS Month,
h.Name as HospitalName,
count(A.AppointmentId) as NumberOfAppointments FROM Appointments A
INNER JOIN MedicalStaff MS on A.MedicalStaffID = MS.MedicalStaffID
INNER JOIN Hospitals H ON MS.HospitalID = H.HospitalID
GROUP BY FORMAT(A.StartDateTime, 'MMMM'),
h.Name

```

Below the code, the 'Messages' pane shows the output: 'Commands completed successfully.' and 'Completion time: 2021-04-21T09:39:31.7109702+05:00'. At the bottom of the screen, a status bar indicates 'Query executed successfully.' and '0 rows'.

Below screenshot shows the result of the above created view

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'UniversityMedicalDatabase' database is selected. In the center pane, a new query window titled 'SQLQuery1.sql' is open, containing the following T-SQL code:

```

SELECT * FROM vappointments;

```

Below the code, the 'Results' pane displays the output of the query:

Month	HospitalName	NumberOfAppointments
June	Apollo	1
June	Kims	1
June	Medicity	1

At the bottom of the screen, a status bar indicates 'Query executed successfully.' and '3 rows'.

VIEW 4:

Visitor that has come to attend patient.

CREATE VIEW vvisitortopatient

AS

```

SELECT Visitors.VisitorFirstName AS VistorFirst, Patients.PatientFirstName AS
PatientFirstName FROM
Patients INNER JOIN Visitors
ON Patients.PatientID = Visitors.PatientID

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'UniversityMedicalDatabase' is selected. In the center pane, a query window titled 'SQLQuery1.sql - DESKTOP-FN9TC92\sa1 (53)*' contains the following SQL code:

```

CREATE VIEW vvisitorpatient
AS
SELECT Visitors.VisitorFirstName AS VistorFirst, Patients.PatientFirstName AS PatientFirstName
FROM Patients INNER JOIN Visitors
ON Patients.PatientID = Visitors.PatientID

```

The results pane below the query window shows the message 'Commands completed successfully.' and the completion time: 2021-08-21T08:44:04.4915149+05:30.

Below screenshot shows the result of the above created view

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'UniversityMedicalDatabase' is selected. In the center pane, a query window titled 'SQLQuery1.sql - DESKTOP-FN9TC92\sa1 (53)*' contains the following SQL code:

```

SELECT * FROM vvisitorpatient

```

The results pane below the query window displays the data from the view:

VistorFirst	PatientFirstName
Steve	David
Mark	Robert
Maria	James

The results pane shows '3 rows' at the bottom.

c) Stored Procedures

A stored procedure is a prepared SQL code that you can save, so the code can be reused repeatedly. You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Here is the list of stored procedures used in the project.

Stored Procedure 1:

```
CREATE PROCEDURE sp_GetTestResults
```

```
AS
```

```
BEGIN
```

```
SELECT P.PatientFirstName, P.PatientLastName, PT.TestResults FROM  
Patients P INNER JOIN PatientTestResults PT ON  
P.PatientID = PT.PatientID
```

```
END
```

```
-- Sai Kumar Palevela
```

The screenshot shows the Object Explorer on the left with the UniversityMedicalDatabase selected. In the center, a query window titled 'SQLQuery1.sql' contains the T-SQL code for creating the stored procedure. The code includes the CREATE PROCEDURE statement, parameters for PatientFirstName, PatientLastName, and TestResults, an AS clause, a BEGIN block with the SELECT statement, and an END block. A comment '-- Sai Kumar Palevela' is present. Below the code, the 'Messages' pane shows the command completed successfully with a completion time of 2021-05-21T17:28:41.9081936+05:30. The status bar at the bottom indicates 'Query executed successfully.'

The screenshot shows the same environment as the previous one. The query window now contains the EXEC sp_GetTestResults command. The results pane on the right displays a table with three rows of data: David Smith with XYZ, James Johnson with KLM, and Robert Andrew with EFG. The status bar at the bottom indicates 'Query executed successfully.'

	PatientFirstName	PatientLastName	TestResults
1	David	Smith	XYZ
2	James	Johnson	KLM
3	Robert	Andrew	EFG

Stored Procedure 2:

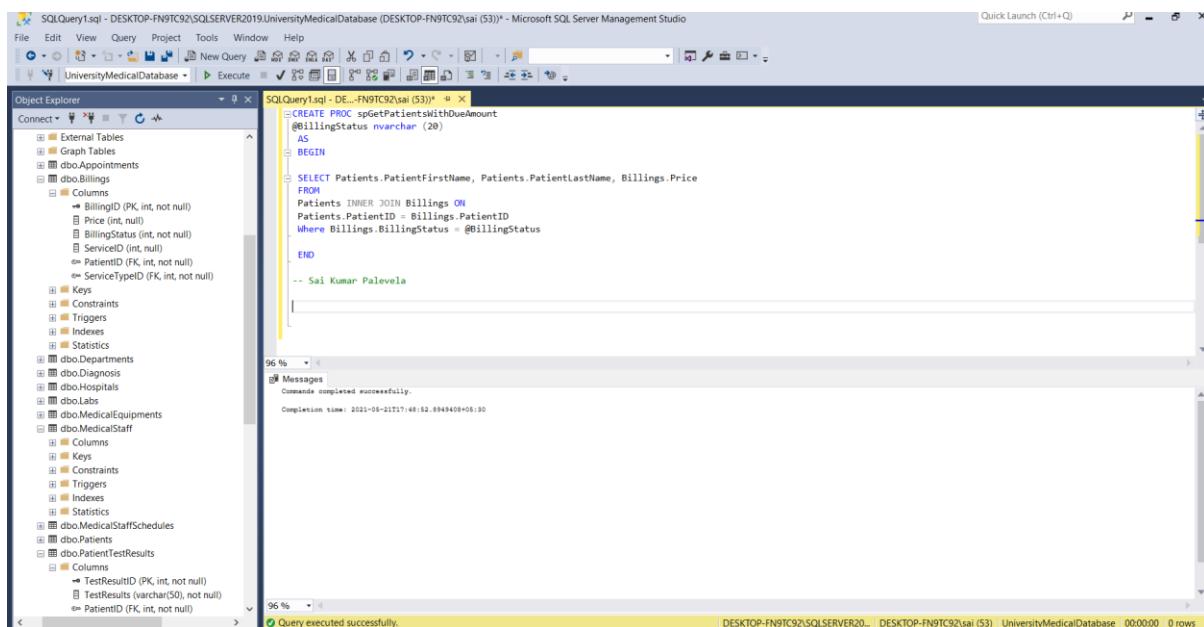
Getting Patients who have not paid their fee.

```
CREATE PROC spGetPatientsWithDueAmount
@BillingStatus nvarchar (20)
AS
BEGIN

SELECT Patients.PatientFirstName, Patients.PatientLastName, Billings.Price
FROM
Patients INNER JOIN Billings ON
Patients.PatientID = Billings.PatientID
Where Billings.BillingStatus = @BillingStatus

END
```

-- Sai Kumar Palevela



The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure, including tables like 'Patients', 'Billings', and 'PatientTestResults'. The central pane contains the T-SQL code for the stored procedure 'spGetPatientsWithDueAmount'. The code includes a CREATE PROC statement with parameters for BillingStatus and a SELECT query joining 'Patients' and 'Billings' tables. The right pane shows the 'Messages' window indicating the command completed successfully. The status bar at the bottom shows the execution time and number of rows affected.

```
CREATE PROC spGetPatientsWithDueAmount
@BillingStatus nvarchar (20)
AS
BEGIN

SELECT Patients.PatientFirstName, Patients.PatientLastName, Billings.Price
FROM
Patients INNER JOIN Billings ON
Patients.PatientID = Billings.PatientID
Where Billings.BillingStatus = @BillingStatus

END
```

-- Sai Kumar Palevela

Below is the Screenshot of Execution of above Stored Procedure.

```

SQLQuery1.sql - DESKTOP-FN9TC92\SQLSERVER2019.UniversityMedicalDatabase (DESKTOP-FN9TC92\sai (53)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
UniversityMedicalDatabase | Execute = ✓
Quick Launch (Ctrl+Q) P X
Object Explorer
Connect ▾
SQLQuery1.sql - DESKTOP-FN9TC92\sai (53)*
EXEC spGetPatientsWithDueAmount
@BillingStatus = 1
-- Sai Kumar Palevela
Results Messages
PatientFirstName PatientLastName Price
1 James Johnson 2000
2 David Smith 1490

```

Query executed successfully.

d) Functions:

Functions: (Pre-defined and User Defined Functions)

In SQL, there are many Pre-defined Aggregate function like COUNT, MIN, MAX, SUM, AVG etc. These Pre-defined functions are used in SQL SELECT expressions to calculate values and manipulate data. These functions can be used anywhere in the expressions.

Here are the List of Function used in the Project.

First Function:

```

CREATE FUNCTION NumberOfPatientsByState(@State Varchar(20))
RETURNS INT
BEGIN
RETURN(SELECT COUNT(*) FROM Patients
WHERE State = @State)
END

```

-- Sai Kumar Palevela

```

CREATE FUNCTION NumberOfPatientsByState(@State Varchar(20))
RETURNS INT
BEGIN
    RETURN(SELECT COUNT(*) FROM Patients
    WHERE State = @State)
END
-- Sai Kumar Palevela

```

The screenshot shows the Object Explorer on the left with the 'UniversityMedicalDatabase' selected. The 'SQLQuery1.sql' window on the right contains the T-SQL code for creating a function. The code defines a function named 'NumberOfPatientsByState' that takes a state name as input and returns the count of patients for that state. The function uses a SELECT statement with a WHERE clause to filter patients by state. The code is signed off by 'Sai Kumar Palevela'. The status bar at the bottom indicates 'Query executed successfully.'

```

SELECT dbo.NumberOfPatientsByState('NY')

```

The screenshot shows the same environment as the previous one. The 'SQLQuery1.sql' window now contains a single query: 'SELECT dbo.NumberOfPatientsByState('NY')'. The results pane shows a single row with a value of 1, indicating there is 1 patient in New York. The status bar at the bottom indicates 'Query executed successfully.'

Second Function:

```

CREATE FUNCTION MedicalStaffDetails()
RETURNS TABLE
RETURN(SELECT
MS.MedicalStaffFirstName, MS.MedicalStaffLastName, MS.PhoneNumber, S.Designation, ST.N
ame as Type from MedicalStaff as MS
INNER JOIN Specialities as S ON S.SpecialityID = MS.SpecialityID INNER JOIN
StaffTypes as ST ON ST.StaffTypeID = MS.StaffTypeID);

```

```

CREATE FUNCTION MedicalStaffDetails()
RETURNS TABLE
AS
SELECT MS.MedicalStaffFirstName, MS.MedicalStaffLastName, MS.PhoneNumber, S.Designation, ST.Type
FROM MedicalStaff AS MS
INNER JOIN Specialties AS S ON S.SpecialtyID = MS.SpecialtyID
INNER JOIN StaffTypes AS ST ON ST.StaffTypeID = MS.StaffTypeID;
--Sai Kumar Paleleva

```

`select * from dbo.MedicalStaffDetails()`
--Sai Kumar Paleleva

```

select * from dbo.MedicalStaffDetails()
--Sai Kumar Paleleva

```

MedicalStaffFirstName	MedicalStaffLastName	PhoneNumber	Designation	Type
James	Smith	(516) 349-5056	Cardiologist	Junior Doctor
Maria	Rodriguez	(315) 447-9825	Dentist	Doctor
Mary	Smith	(514) 223-5643	Pulmonologist	Nurse

e) Scripts

Script : create users with various security levels, password assignments, roles, encryptions

`USE UniversityMedicalDatabase;`

`GO`

`CREATE ROLE PatientEntry;`

`GRANT UPDATE`

`ON Patients`

`TO PatientEntry;`

`GO`

`CREATE ROLE AddNewVisitor`

```

GRANT INSERT, UPDATE
ON Visitors
TO AddNewVisitor
GO
CREATE LOGIN SaiKumar WITH PASSWORD = '$@!kumar123',
DEFAULT_DATABASE = UniversityMedicalDatabase;
CREATE USER Sai FOR LOGIN SaiKumar;
ALTER ROLE PatientEntry ADD MEMBER Sai;

```

The screenshot shows the SQL Server Management Studio interface with a query window containing the provided SQL code. The code creates a database, grants update permissions to the PatientEntry role, creates a login named SaiKumar, creates a user named Sai, and adds the user to the PatientEntry role. The execution status at the bottom indicates "Query executed successfully".

```

USE UniversityMedicalDatabase;
GO
CREATE ROLE PatientEntry;
GRANT UPDATE
ON Patients
TO PatientEntry;
GO
CREATE ROLE AddNewVisitor
GRANT INSERT, UPDATE
ON Visitors
TO AddNewVisitor;
GO
CREATE LOGIN SaiKumar WITH PASSWORD = '$@!kumar123',
DEFAULT_DATABASE = UniversityMedicalDatabase;
CREATE USER Sai FOR LOGIN SaiKumar;
ALTER ROLE PatientEntry ADD MEMBER Sai;
-- Sai kumar Palevela

```

f) Triggers

Trigger1:

```

CREATE TRIGGER UpdateMedicalStaffSchedule
    ON Appointments
    for insert
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    declare @appointmentid int
    select @appointmentid = AppointmentID from inserted

    INSERT INTO MedicalStaffSchedules
    select
        MedicalStaffID,
        1 as ScheduleTypeID,
        StartDateTime,
        EndDateTime,
        Description as Descriptions
    from Appointments
        WHERE AppointmentID = @appointmentid
--Sai Kumar Palevela

```

UniversityMedicalDatabase

```

CREATE TRIGGER UpdateMedicalStaffSchedule
    ON Appointments
    FOR INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    declare @appointmentid int
    select @appointmentid = AppointmentId from inserted

    INSERT INTO MedicalStaffSchedules
    select
        MedicalStaffID,
        1 as ScheduleTypeID,
        StartDateTime,
        EndDateTime,
        Description as Descriptions
    FROM Appointments
    WHERE AppointmentID = @appointmentid
END

```

Messages

Commands completed successfully.

Completion time: 2021-05-22T00:55:24.2194824+05:00

Query executed successfully.

UniversityMedicalDatabase

```

select * from MedicalStaffSchedules

```

ScheduleID	MedicalStaffID	ScheduleTypeID	StartDateTime	EndDateTime	Descriptions
1	1	1	2021-06-12 10:34:09.000	2021-06-12 14:34:09.000	A
2	2	1	2021-06-13 10:34:09.000	2021-06-13 14:34:09.000	B
3	3	1	2021-06-14 10:34:09.000	2021-06-14 14:34:09.000	C

Results

Messages

Query executed successfully.

Testing of Trigger:

The screenshot shows the Object Explorer on the left with the UniversityMedicalDatabase selected. In the center, a query window titled 'SQLQuery2.sql - DE...-FN9TC92\sa (61)' is open, displaying the following SQL code:

```
INSERT [dbo].[Appointments] ([StartTime], [EndTime], [Description], [Price], [PatientID], [MedicalStaffID])
VALUES ( CAST('2021-06-04T13:34:09.000' AS DateTime), CAST('2021-06-12T18:34:09.000' AS DateTime), 'A', 500, 2, 1)
```

The status bar at the bottom indicates 'Query executed successfully'.

The screenshot shows the Object Explorer on the left with the UniversityMedicalDatabase selected. In the center, a query window titled 'SQLQuery2.sql - DE...-FN9TC92\sa (61)' is open, displaying the following SQL code:

```
select * from MedicalStaffSchedules
```

The results pane shows a table with the following data:

ScheduleID	MedicalStaffID	ScheduleTypeID	StartTime	EndTime	Description
1	1	1	2021-06-12 10:34:09.000	2021-06-12 14:34:09.000	A
2	2	1	2021-06-13 10:34:09.000	2021-06-13 14:34:09.000	B
3	3	1	2021-06-14 10:34:09.000	2021-06-14 14:34:09.000	C
4	4	1	2021-06-04 10:34:09.000	2021-06-12 16:34:09.000	A

The status bar at the bottom indicates 'Query executed successfully'.

Trigger 2:

```
CREATE TRIGGER UpdateBillingAppointment
    ON Appointments
    for insert
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
```

```

declare @appointmentid int
select @appointmentid = AppointmentId from inserted

INSERT INTO Billings
select
    Price,
    0 as BillingStatus,
    @appointmentid as ServiceId,
    PatientID,
    1 as ServiceTypeId
from Appointments
WHERE AppointmentID = @appointmentid

END

```

The screenshot shows the Object Explorer on the left with the 'UniversityMedicalDatabase' expanded. The 'Triggers' node under 'Appointments' is selected. The main pane displays the T-SQL code for creating the trigger:

```

CREATE TRIGGER UpdateBillingAppointment
ON Appointments
FOR Insert
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    declare @appointmentid int
    select @appointmentid = AppointmentId from inserted

    INSERT INTO Billings
    select
        Price,
        0 as BillingStatus,
        @appointmentid as ServiceId,
        PatientID,
        1 as ServiceTypeId
    from Appointments
    WHERE AppointmentID = @appointmentid
END

```

The 'Messages' pane at the bottom right shows the message: "Command completed successfully." and "Completion time: 2021-06-12T01:04:28.7000000Z".

The screenshot shows the Object Explorer on the left with the 'UniversityMedicalDatabase' expanded. The 'Tables' node under 'Appointments' is selected. The main pane displays the T-SQL code for inserting data into the 'Appointments' table:

```

INSERT [dbo].[Appointments] ([StartTime], [EndTime], [Description], [Price], [PatientID], [MedicalStaffID])
VALUES ( CAST('2021-06-04T16:34:09.000' AS DateTime), CAST('2021-06-12T16:34:09.000' AS DateTime), N'A', 500, 2, 1)

```

The 'Messages' pane at the bottom right shows the message: "1 row affected." and "Completion time: 2021-06-12T01:10:13.4172333Z".

Testing Trigger:

The screenshot shows the SSMS interface. The Object Explorer on the left lists several database objects: Diagnosis, Labs, MedicalEquipments, MedicalStaff, MedicalStaffSchedules, Patients, PatientTestResults, RoomEquipments, Rooms, and RoomSchedules. The MedicalStaff node is expanded, showing columns like MedicalStaffID, MedicalStaffFirstName, MedicalStaffLastName, MedicalStaffAddress, PhoneNumbers, SpecialityID, HospitalID, and StaffTypeID. The MedicalStaffSchedules node is also expanded, showing columns like ScheduledID, StartDateTime, EndDateTime, and Descriptions. The query results window on the right displays the output of a SELECT query against the Billings table, which contains 5 rows of data.

BillingID	Price	BillingStatus	ServiceID	PatientID	ServiceTypeID
1	2000	0	2	3	1
2	2000	0	2	3	1
3	1400	1	3	1	2
4	1000	0	1002	2	1
5	1000	0	1003	2	1

5) Conclusion:

The implementation and design of database to keep track of every Patient status throughout the Diagnosis process from the time s/he got admitted to hospital till s/he got discharged from hospital.

This database is created from the scratch and all the data that I have used in my project is considered from the real-life values.

In this project, all the basic database concepts are implemented i.e. Starting with thorough understanding of the user/customer requirements and identify the objects. Then creation of tables, determining its constraints, relationships between the tables used, Entity Relationship diagram, inserting data, usage of standard queries, Normalizing the database, creation of Views, usage of Predefined and User defined functions, Stored Procedures, Triggers and ending with transactions. Finally, ends with the test cases for all the business scenarios. By doing this project I have got great command over various skills on all the database concepts also got experience on working with the real-world project from scratch.