HEALTH CARE CENTER DATABASE

Database Design Report

Instructor: Prof. Ehat Ercanli

Course Name: CSE 581 Introduction to Database Management System

Sonal Patil [SUID 997435672]

Email: spatil06@syr.edu

Date: 05-07-2018

TABLE OF CONTENTS

1. ABSTRACT

Health is wealth! Healthcare plays an important role in the society. People get ill, affected by some virus or injured with accidents. At this critical time, health care centers are the ones where people seek for help. The main moto of any Healthcare center is to treat, diagnose people with health issues and take care of them. Also, prevent the spreading of some major viruses. Healthcare centers fulfill this moto by hiring well educated medical perfectionist employees which can include doctors, nurses etc.

Every person on this planet has some goals/dreams about his/her life. And they all want to pursue them before their life ends. These all things that we dream in our life can only be fulfilled if we have a good health. Keeping our self away from all infection & disease is practically impossible as we live in society with full of different people. We can get ill or injured and to recover from these health issues we need a person who will know how to treat this issue and this need can get fulfilled by a health care center. Because they are there to help you reover from your health issues.

This importance of Healthcare center into a human being' life make it more responsible and to handle these responsibilities very neatly and carefully, one must have proper organization of many things and one of them is Data. Data of any real-world entity plays an important role in managing that entity. Similarly, in health care center, there are employees (e.g. Doctor, Nurse, Ward-boy etc.), patients, their diagnosis, treatment details, medical devices, visitors and many more. The data related to all these entities must be kept organized, secured and easy to access to health care center users.

The design of health care center database will definitely help in managing the data but in bonus it will also help in running the health care center processes faster and everyone who is seeking for service will get served without hassle. The implementation of the database will also help in retrieving archived records of previous cases, events. To do this, we need to get all the health care center related information and then organize it into tables. Applying the normalization forms will help in reducing data redundancy.

2.1. Design Introduction

It is always a good practice to design a database before implementing it. The designing will reduce a lot of big issues that can get occurred later when end user starts to use the application. While designing a database, we accurately collect the data, update and track them on regular basis. The health care center database will help in keeping track of all the data which is handled in the system through various events. This either can be patient health history or employee salary info or patient's visitor info.

Gathering the information is the very basic and first step of designing the database. The health care center must have information related to its employees and patients. These are the very basic entities which anyone can consider when they'll think about the health care center. In addition to these, there can be a separate entity for the visitors who will visit the patients. This entity is optional because many times it up to the health care center whether they want to store visitors' data or not, but it is a good practice to do that. Then there comes the information related to medical devices getting used in the center, medicals prescribed by the doctor to patients, procedures like X-ray machine, city scanner etc. getting used for a patient and at the end charging a bill to a patient depending upon the all the previous health care center facilities used by that patient. Patient's health insurance related data is also very important entity which helps while patient is making a payment. Until now, all the entities mentioned are related to health care center, but health care center physical structure is also an entity which helps us keeping all the information related to rooms, offices included in the health care center.

Patient can be referred by other doctors outside the current health care center in such cases we need to store the information of who referred that patient. The details we store about the referral doctor can help the new assigned doctor in future when he needs to consult to previous doctor regrading some health issue of that patient. Storing the referral doctor information in advance will simplify the task later. General information of patient should be stored in separate so that anyone who is working on that case may be a doctor or nurse or assistant can have quick access to basic information.

The whole idea of designing the health care center database is to have a simplified format of the complex and huge set of data so that the user of this database application can access it very quickly.

2.2. Database Design Process

While designing a database on a big scale, you need to have a correct database design which avoids multiple big issues that may cause later when the end user starts to use the application. How to decide whether you have designed correct database or not? When I designed my Health care center database, I followed below steps which takes you to the correct database design.

1. Gathering the information
   - When I got the task of health care center database design, the first step I did was to collect the information about health care center, how it works, what are the real-world entities, what kind of data will get stored in each entity, what is the nature of application, who will use the database etc.
2. What is the nature of application?
   - The normalization of the database most of the times depend on this rule. If you are using the database application just to analyze the data then you don't need to be strict on the normalization but if your design will get used to insert, update, delete information in the database then you need to have a normalized database. The health care center database end user will use this database to create new entries for the new patients, update previous patient's details or delete some wrong entries. Hence, I have decided to normalize my health care center database.
3. Identifying the main entities [tables]
   - After gathering the information related to health care center, it's now my responsibility to arrange all those entities in proper manner, identifying the main & sub-main entities. Main entities considered as the tables and sub-main entities which will have the single information, placed under the main entities as the fields [columns]. For example, Patient will be the main entity i.e. table and his name, primary care doctor, his health history will be the fields of the Patient table i.e. columns. At this step, I had many duplicates in many tables due to same data getting inserted by one user, getting viewed by another user and getting modified by another user.
4. Break your data into logical pieces –
   - In this step, I tried to break the fields into logical pieces so that end user can make a clean simplified query to get the information instead of the complex queries sending out not needed data with the result data. For example, dividing the address in many fields like AddressLine1, AddressLine2, City, State, zip code etc. So that if the end user only needs to find out the patients in particular area he can perform the simple query.
5. Decide on primary & foreign keys
   - These keys play very important role in the database table relations. When you need to normalize the database, those rules mostly depend on the primary key. Primary key the one which decides how the non-key fields gets addressed in the table. And the foreign key describes the relationship between two tables when you decide to divide one table into multiple derived tables to avoid the redundancy of the data. In the health care center database, I have assigned primary key to almost all tables and also foreign keys when there is a relation.
   -

6. Watch for partial dependencies
    - All the fields should entirely depend on the primary key. If there are fields which depends partially on the primary key, then you need to remove so that the database will be in the normalized form. For that, you can remove those fields from the current table and place in the sub table and connect them by giving the reference by foreign key.
7. Treat duplicate non-uniform data as your biggest enemy
    - Duplicates are dangerous in many ways. They can take more space on the disk, they can create the confusion if a data at one place gets modified by one end user and not gets updated at other places in database and other end user is looking at the wrong old one data. One way to remove duplicates is to divide the entity into sub entity i.e. a master table sub table and give the reference using foreign keys. This solution can remove the duplicates mentioned in step 3 so that what I have decided in the step 2 i.e. to normalize the database will get achieved.
8. Don’t forget indexes in the design
    - Some indexing design is useful during database modeling, even if indexes may change during actual deployment and usage. Indexes are important when considering queries on the data. When modeling, you should consider how the data will be queried. Take care not to over-index. Indexing revolves around query optimization.

2.3. Database Diagram [Entity Relation Model] -

The following database diagram is my design for the health care center database which shows the entities [tables] and their fields [columns] that are included in the health care center database. It also shows the relation between two tables and indicates primary & foreign key columns. The next section describes this diagram in detail.

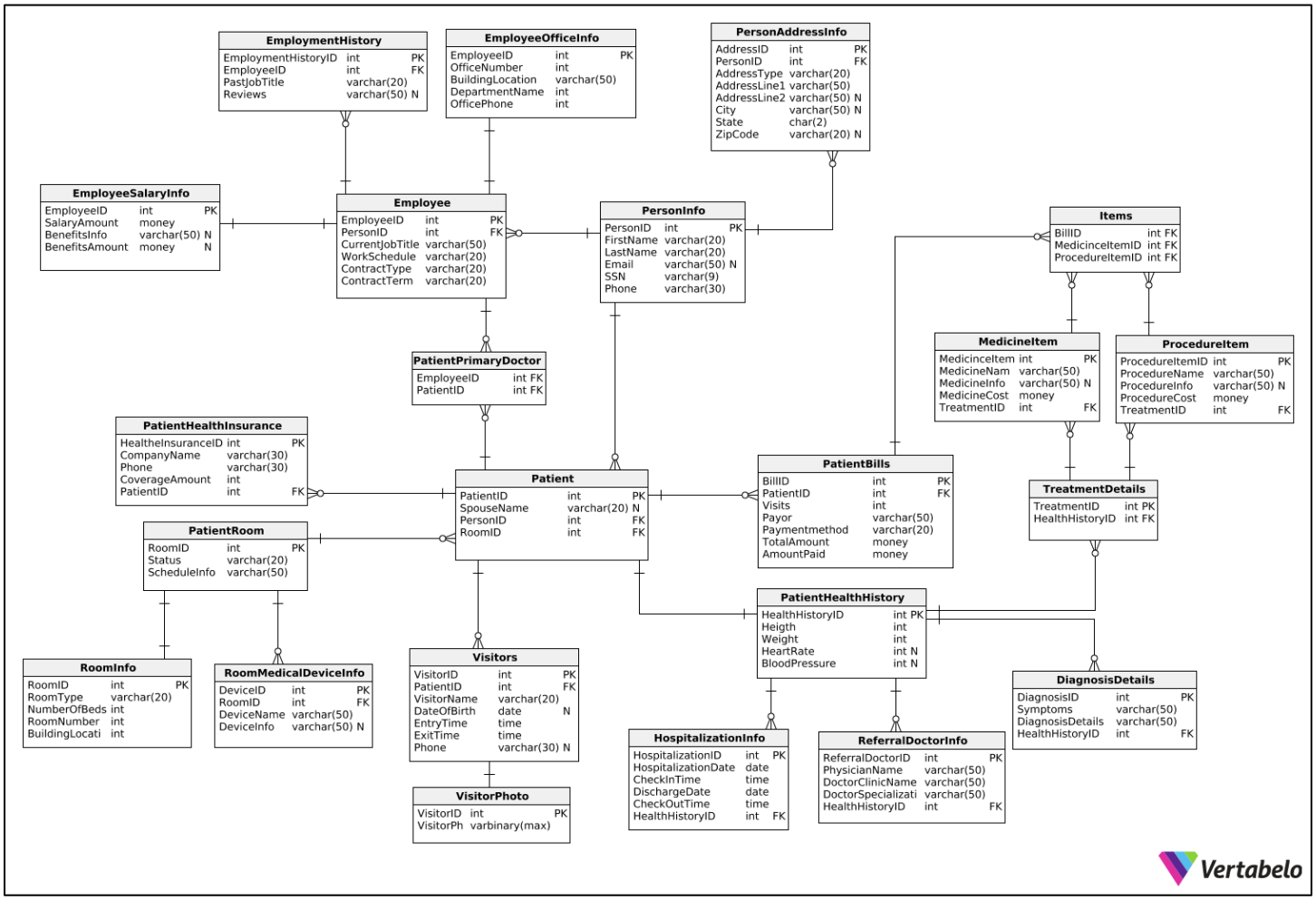


Figure 1 Health Care Center Database Design

2.4. Database Design Description -

The section 2.3 shows my health care center database design diagram which is entity relation model. This diagram shows the main entities in the database and relations between them. I have designed this health care center database considering all the minor real-world scenarios. Below is description of that diagram

1. **PersonInfo –**
   This table holds the common personal data for both Employees & Patients.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is PersonID. Each person will have his own unique ID.
   Foreign Key – No Foreign key for this table.
   Null/Not Null Values – For this table, I have chosen to accept null values for only Email column because not everybody will have email info with them or they might not want to share it.

2. **PersonAddressInfo –**
   This table holds the common in detail address of a person i.e. of Employees & Patients.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is AddressID. Each person will have his own unique address ID.
   Foreign Key – PersonID is Foreign key for this table. This describes the one:many relationship. This means One Person can have multiple Address. For example, an Employee can have 2 homes old one & new one and he choose to keep both the addresses in his personal info.
   Null/Not Null Values – For this table, I have chosen to accept null values for AddressLine2 column which is deigned to hold the remaining part of an address if the address exceeds the AddressLine1 cloumn capacity. This case might occur or might not. The other two columns City & ZipCode can also accept the null values because of the lack of information from the person side.

3. **Employee –**
   This table holds data of Employees working at the health care center.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is EmployeeID. Each employee will have his own unique ID.
   Foreign Key – PersonID is Foreign key for this table. This describes the one:many relationship. This means Person table can hold personal data of multiple Employees.
   Null/Not Null Values – For this table, All the information is the basic information which an employee gets when hired so all fields must have some data while creating a new entry of a new employee.

4. **EmployeeOfficeInfo –**
   This table holds the office information of Employees.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is EmployeeID. Here, I have created one:one relationship between EmployeeOfficeInfo and Employee table. This means each employee will have one office assigned to him/her.
   Foreign Key – EmployeeID act as both Primary Key & Foreign Key as per above given explanation.

Null/Not Null Values – For this table, to go to a particular office location we need information from all the fields so while creating an entry in this table one must enter all the columns with data.

5. **EmployeeSalaryInfo –**
   This table holds the salary information of an Employee.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is EmployeeID. Here, I have created one:one relationship between EmployeeSalaryInfo and Employee table. This means each employee will have one salary info record assigned to him/her.
   Foreign Key – EmployeeID act as both Primary Key & Foreign Key as per above given explanation.
   Null/Not Null Values – For this table, I have chosen to accept null values for BenefitsInfo & BenefitsAmount columns because not everybody will get benefits especially the new hired employees.

6. **EmploymentHistory –**
   This table holds the previous employment history of an Employee.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is EmploymentHistoryID. Each person will have his own unique employment history ID.
   Foreign Key – EmployeeID is Foreign key for this table. This describes the one:many relationship. This means one employee can have multiple employment history records.
   Null/Not Null Values – For this table, I have chosen to accept null values for only Reviews column because if an employee is joining health care center as his first employment then he won't have any previous employment experience and hence no reviews.

7. **Patient –**
   This table holds the data related to Patients.
   Primary Key – In this table, uniquely identifying key i.e. Primary key is PatientID. Each patient will have his own unique ID.
   Foreign Key – PersonID is first Foreign Key for this table. This describes one:many relationship between Person table & Patient table which means Person table can holds data of multiple patients. RoomID is another Foreign key for his table. This describes one:many relationship between Patient table & PatientRoom table which means one room can be assigned to multiple patients if it is Sharing room type.
   Null/Not Null Values – For this table, I have chosen to accept null values for only SpouseName column because not every patient will be married.

8. **PrimaryDoctors –**
   This table holds the data of doctors assigned to a patient.
   Primary Key – In this table, there is no uniquely identifying key i.e. Primary key because this table is used as the linking table between Employee & Patient table. This describes many:many relationship between Employee & Patient table.
   Foreign Key – EmployeeID & PatientID are two foreign keys of this table which means one employee can serve to multiple patients and one patient can be consulting many different doctors at the health care

center. For example, Dr. X can have A, B & C patients [multiple patients] and patient A can visit Dr. X for Oral problems, Dr. Y for cardiac problems.

Null/Not Null Values – For this table, there are no other fields than foreign keys and those two columns can't be null. So this table doesn't have a single column that accepts null values.

9. **PatientHealthInsurance –**

This table holds the health insurance related data of a Patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is HealthInsuranceID. Each patient will have his own unique health insurance ID.

Foreign Key – PatientID is the foreign key for this table. This describes one:many relationship between Patient & PatientHealthInsurance table. This means one patient can have multiple health insurances. As per the real-world search, I have found out this information that one can have multiple health insurances taken for himself and hospitals also accept multiple health insurances for that patient.

Null/Not Null Values – For this table, I have chosen not to accept any null values for any column because all the information is needed for the proper validation of one's health insurance.

10. **Visitors –**

This table holds the data related to visitors who comes to visit a Patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is VisitorID. Each visitor will have his own unique ID.

Foreign Key – PatientID is the foreign key for this table. This describes one:many relationship between Patient & Visitors table. This means one patient can have multiple visitors.

Null/Not Null Values – For this table, I have chosen to accept null values for only DateOfBirth column because not everybody will want to share their birth dates for a visit.

11. **VisitorPhoto**

This table holds photo data of a Visitor.

Primary Key – In this table, uniquely identifying key i.e. Primary key is VisitorID. Here, I have created one:one relationship between Visitors and VisitorPhoto table. This means each visitor will have one photo taken of him/her. The only reason of creating this table is because the photo data will be large and not regularly accessed.

Foreign Key – VisitorID act as both Primary Key & Foreign Key as per above given explanation.

Null/Not Null Values – For this table, I have chosen not to accept null values as the healthcare will take the photo of visitor on arrival and this is the only non-key column and it won't be null.

12. **PatientRoom –**

This table holds the data related to room assigned for a Patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is RoomID. Each room will have his own unique ID.

Foreign Key – No Foreign key for this table.

Null/Not Null Values – For this table, I have chosen not to accept null values as all the information is important one & needed one.

13. RoomInfo –

This table holds the detailed data related of a room.

Primary Key – In this table, uniquely identifying key i.e. Primary key is RoomID. Here, I have created one:one relationship between RoomInfo and PatientRoom table. This means each room will have one unique room info.

Foreign Key – RoomID act as both Primary Key & Foreign Key as per above given explanation.

Null/Not Null Values – For this table, I have chosen not to accept null values as all the information is important one & needed one.

14. RoomMedicalDeviceInfo –

This table holds the data of medical devices present in each room.

Primary Key – In this table, uniquely identifying key i.e. Primary key is DeviceID. Each device will have his own unique ID.

Foreign Key – RoomID is foreign key for this table. This describes one:many relationship between PateintRoom & RoomMedicalDeviceInfo table which means one room can have multiple devices in it.

Null/Not Null Values – For this table, I have chosen not to accept null values as all the information is important one & needed one.

15. PatientBills –

This table holds the data written on a bill assigned to a Patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is BillID. Each bill will have his own unique ID.

Foreign Key – PatientID is foreign key for this table. This describes one:many relationship between Patient & PetientBills which means one patient can have multiple bills assigned to him.

Null/Not Null Values – For this table, I have chosen not to accept null values as all the information is important one & needed one.

16. Items –

This table holds linking connection between a bill and items placed that bill.

Primary Key – In this table, there is no uniquely identifying key i.e. Primary key.

Foreign Key – This table is used as linking table between PatientBills & MedicineItem, ProcedureItem. This means PatientBills & MedicineItem and PatientBills & ProcedureItem both table combination has many:many relationship with each other. One bill can have multiple medicines and multiple procedures. Similarly, one medicine or one procedure can be one multiple bills. That's why this table have BillID, MedicineItemID, ProcedureItemID as foreign keys.

Null/Not Null Values – For this table, I have chosen not to accept null values as all fields are keys.

17. MedicineItem –

This table holds the medicine item data.

Primary Key – In this table, uniquely identifying key i.e. Primary key is MedicineItemID. Each medicine will have his own unique ID.

Foreign Key – TreatmentID is foreign key for this table. This describes one:many relationship between TreatmentDetails & MedicineItem table which means one treatment given by a doctor can have multiple medicines listed.

Null/Not Null Values – For this table, I have chosen to accept null values for only MedicineInfo column because not having info of a medicine doesn't affect the purchaser & buyer.

18. ProcedureItem –

This table holds the procedure item data.

Primary Key – In this table, uniquely identifying key i.e. Primary key is ProcedureItemID. Each procedure will have his own unique ID.

Foreign Key – TreatmentID is foreign key for this table. This describes one:many relationship between TreatmentDetails & ProcedureItem table which means one treatment given by a doctor can have multiple procedures listed.

Null/Not Null Values – For this table, I have chosen to accept null values for only ProcedureInfo column because not having info of a procedure doesn't affect the purchaser & buyer.

19. PatientHealthHistory –

This table holds the health history of a Patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is HealthHistoryID. Here, I have created one:one relationship between Patient and PatientHealthHistory table. This means each patient will have one unique health history record.

Foreign Key – HealthHistoryID act as both Primary Key & Foreign Key as per above given explanation.

Null/Not Null Values – For this table, I have chosen to accept null values for only HeartRate & BloodPressure columns because this information is not needed if patient has arrived first time for a dental check-up.

20. TreatmentDetails –

This table holds the data related to treatments given by a doctor for a patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is TreatmentID. Each patient will have his own unique treatment ID.

Foreign Key – HealthHistoryID is foreign key for this table. This describes one:many relationship between PatientHealthHistory & TreatmentDetails which means one health history can have multiple treatments performed by a doctor on a patient.

Null/Not Null Values – For this table, I have chosen not to accept null values as all fields are keys. There is no non-key column in this table.

21. DiagnosisDetails –

This table holds the data related to diagnosis done by a doctor on a patient.

Primary Key – In this table, uniquely identifying key i.e. Primary key is DiagnosisID. Each patient will have his own unique diagnosis ID.

Foreign Key – HealthHistoryID is foreign key for this table. This describes one:many relationship between PatientHealthHistory & DiagnosisDetails which means one health history can have multiple diagnosis done & found by a doctor for a patient.

Null/Not Null Values – For this table, I have chosen not to accept null values as all information is important & needed one.

22. ReferralDoctorsInfo –

This table holds the data related to a doctor outside of current health care center who has referred a patient to see new doctor in this health care center.

Primary Key – In this table, uniquely identifying key i.e. Primary key is ReferralDoctorID. Each patient will have his own unique referral doctor ID.

Foreign Key – HealthHistoryID is foreign key for this table. This describes one:many relationship between PatientHealthHistory & ReferralDoctorInfo which means one health history can have multiple referral doctors. For example, if a patient has visited multiple doctors outside of this health care center and all those doctors have suggested the same doctor in this health care center.

Null/Not Null Values – For this table, I have chosen not to accept null values as all information is important & needed one.

23. HospitalizationInfo

This table holds the data related to patient's health care center entry & exit details.

Primary Key – In this table, uniquely identifying key i.e. Primary key is HospitalizationID. Each patient will have his own unique hospitalization ID.

Foreign Key – HealthHistoryID is foreign key for this table. This describes one:many relationship between PatientHealthHistory & HospitalizationInfo which means one single patient health history can have multiple hospitalization details. For example, if a patient has to perform some surgeries regarding his oral health then he might have to hospitalize multiple times i.e. one time for each surgery and then he can have multiple entry & discharge info in hospitalization table.

Null/Not Null Values – For this table, I have chosen not to accept null values as all information is important & needed one.

3. IMPLEMENTATION

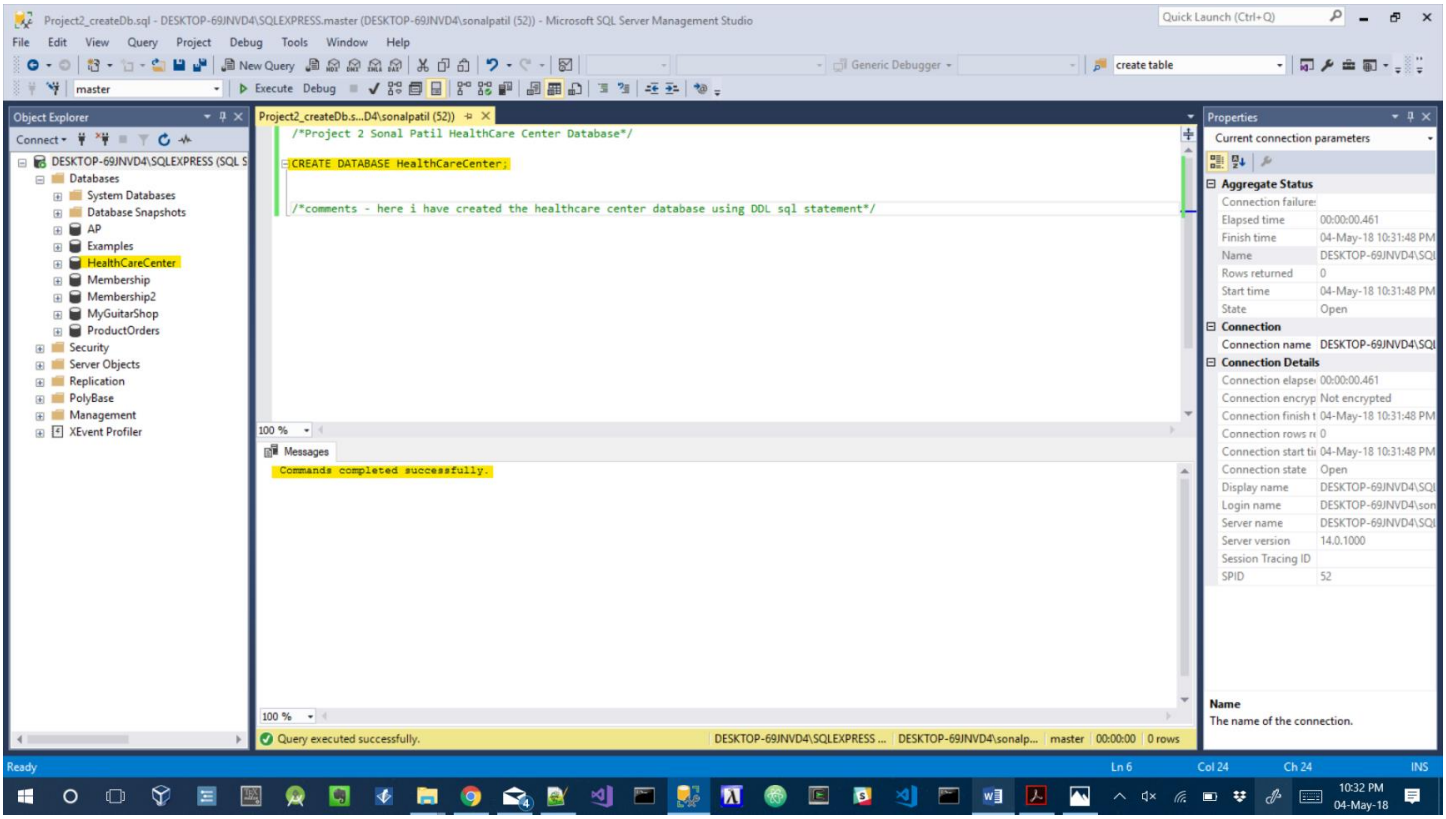
Following is the database implementation performed step by step:

3.1. Creating Database –

Code –

```
CREATE DATABASE HealthCareCenter;
/*comments - here I have created the healthcare center database using DDL sql statement*/
```

Result Screenshot –



3.2. Creating Database Tables & assigning Columns –

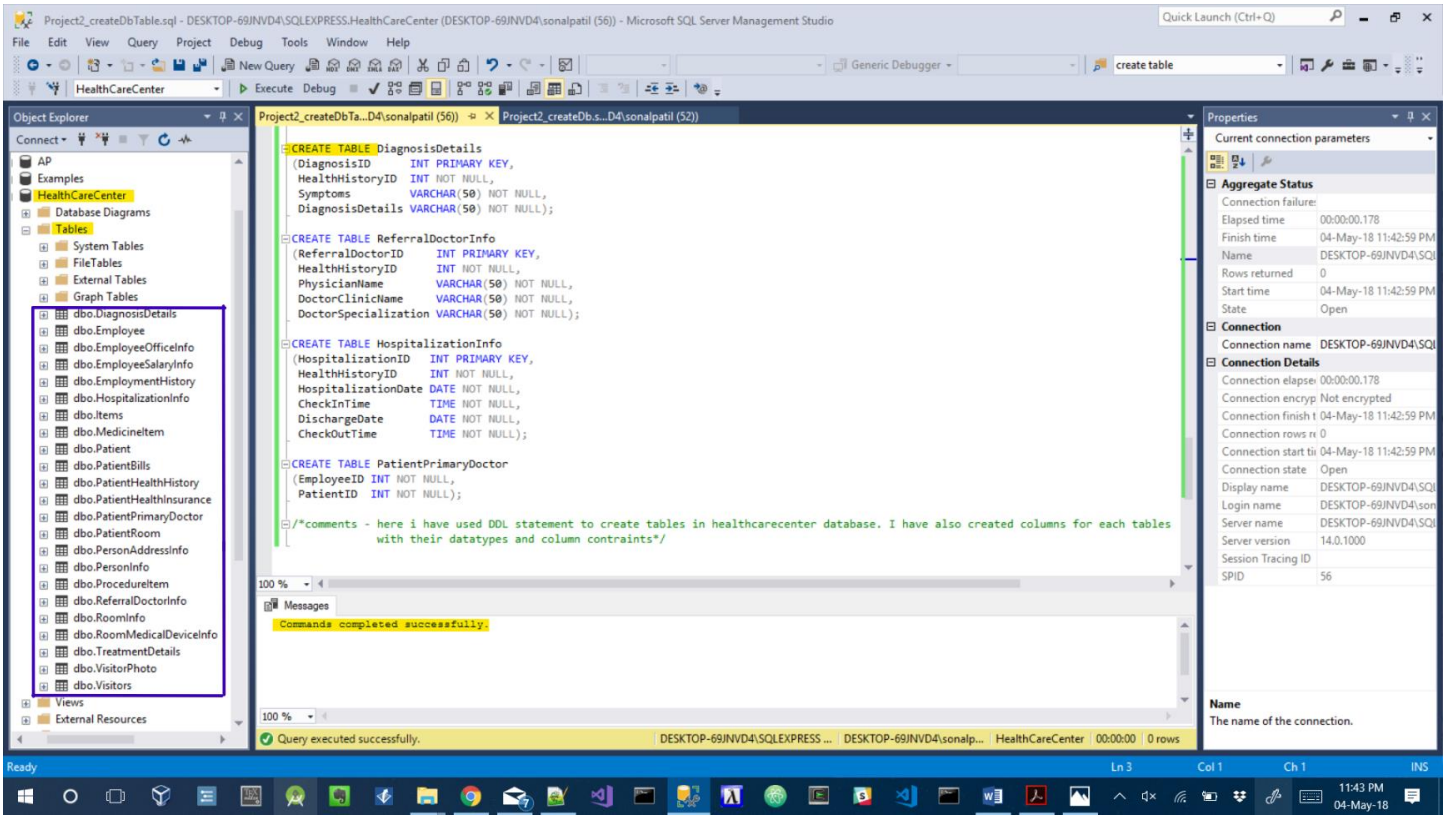
Code –

```
USE HealthCareCenter;
CREATE TABLE PersonInfo
(PersonID  INT          PRIMARY KEY,
 FirstName VARCHAR(20) NOT NULL,
 LastName  VARCHAR(20) NOT NULL,
 Email     VARCHAR(50) NULL,
 SSN       VARCHAR(9)  NOT NULL CHECK (SSN IS NOT NULL),
 Phone     VARCHAR(30) NOT NULL);

CREATE TABLE PersonAddressInfo
(AddressID    INT         PRIMARY KEY,
 PersonID     INT         NOT NULL,
 AddressType  VARCHAR(20) NOT NULL,
 AddressLine1 VARCHAR(50) NOT NULL,
 AddressLine2 VARCHAR(50) NULL,
 City         VARCHAR(50) NULL,
 State        CHAR(2)     NOT NULL,
 ZipCode      VARCHAR(20) NULL);

CREATE TABLE Employee
(EmployeeID      INT         PRIMARY KEY,
 PersonID        INT         NOT NULL,
 CurrentJobTitle VARCHAR(50) NOT NULL,
 WorkSchedule    VARCHAR(20) NOT NULL,
 ContractType    VARCHAR(20) NOT NULL,
 ContractTerm    VARCHAR(20) NOT NULL);

CREATE TABLE EmployeeOfficeInfo
(EmployeeID       INT         PRIMARY KEY IDENTITY(1,1),
 OfficeNumber     INT         NOT NULL CHECK (OfficeNumber > 0),
 BuildingLocation VARCHAR(50) NOT NULL,
 DepartmentName   VARCHAR(20) NOT NULL,
 OfficePhone      VARCHAR(30) NOT NULL);

CREATE TABLE EmployeeSalaryInfo
(EmployeeID     INT         PRIMARY KEY IDENTITY(1,1),
 SalaryAmount   MONEY       NOT NULL DEFAULT 0 CHECK (SalaryAmount >= 0),
 BenefitsInfo   VARCHAR(50) NULL,
 BenefitsAmount MONEY       NULL DEFAULT 0);

CREATE TABLE EmploymentHistory
(EmploymentHistoryID INT PRIMARY KEY,
 EmployeeID          INT NOT NULL,
 PastJobTitle        VARCHAR(20) NOT NULL,
 Reviews             VARCHAR(50) NULL);

CREATE TABLE Patient
(PatientID  INT PRIMARY KEY,
 SpouseName VARCHAR(50) NULL,
 PersonID   INT NOT NULL,
 RoomID     INT NOT NULL);

CREATE TABLE PatientRoom
(RoomID             INT         PRIMARY KEY,
 Status       VARCHAR(20) NOT NULL,
 ScheduleInfo VARCHAR(50) NOT NULL);

CREATE TABLE RoomInfo
```

```
(RoomID           INT         PRIMARY KEY IDENTITY(1,1),
 RoomType         VARCHAR(20) NOT NULL,
 NumberOfBeds     INT         NOT NULL,
 RoomNumber       INT         NOT NULL CHECK (RoomNumber > 0),
 BuildingLocation VARCHAR(50) NOT NULL);

CREATE TABLE RoomMedicalDeviceInfo
(DeviceID   INT         PRIMARY KEY,
 RoomID     INT         NOT NULL,
 DeviceName VARCHAR(30) NOT NULL,
 DeviceInfo VARCHAR(50) NULL);

CREATE TABLE PatientHealthInsurance
(HealthInsuranceID INT         PRIMARY KEY,
 PatientID         INT         NOT NULL,
 CompnayName       VARCHAR(30) NOT NULL,
 Phone             VARCHAR(30) NOT NULL,
 CoverageAmount    MONEY       NOT NULL DEFAULT 0 CHECK (CoverageAmount >= 0));

CREATE TABLE Visitors
(VisitorID INT PRIMARY KEY,
 PatientID   INT         NOT NULL,
 VisitorName VARCHAR(50) NOT NULL,
 DateOfBirth DATE        NULL,
 Phone       VARCHAR(30) NULL,
 EntryTime   TIME        NOT NULL,
 ExitTime    TIME        NOT NULL);

CREATE TABLE VisitorPhoto
(VisitorID    INT            PRIMARY KEY IDENTITY(1,1),
 VisitorPhoto VARBINARY(MAX) NOT NULL);

CREATE TABLE PatientBills
(BillID        INT         PRIMARY KEY,
 PatientID     INT         NOT NULL,
 Visits        INT         NOT NULL CHECK (Visits > 0),
 Payor         VARCHAR(20) NOT NULL,
 PaymentMethod VARCHAR(20) NOT NULL,
 TotalAmount   MONEY       NOT NULL DEFAULT 0 CHECK (TotalAmount >= 0),
 AmountPaid    MONEY       NOT NULL DEFAULT 0 CHECK (AmountPaid >= 0));

CREATE TABLE Items
(BillID          INT NOT NULL, MedicineItemID  INT NOT NULL, ProcedureItemID INT NOT NULL);

CREATE TABLE MedicineItem
(MedicineItemID INT         PRIMARY KEY,
 TreatmentID    INT         NOT NULL,
 MedicineName   VARCHAR(30) NOT NULL,
 MedicineInfo   VARCHAR(50) NULL,
 MedicineCost   MONEY       NOT NULL DEFAULT 0 CHECK (MedicineCost >= 0));

CREATE TABLE ProcedureItem
(ProcedureItemID INT         PRIMARY KEY,
 TreatmentID     INT         NOT NULL,
 ProcedureName   VARCHAR(30) NOT NULL,
 ProcedureInfo   VARCHAR(50) NULL,
 ProcedureCost   MONEY       NOT NULL DEFAULT 0 CHECK (ProcedureCost >= 0));

CREATE TABLE PatientHealthHistory
(HealthHistoryID INT PRIMARY KEY IDENTITY(1,1),
 Height          INT NOT NULL CHECK (Height > 0),
 Weight          INT NOT NULL CHECK (Weight > 0),
```

```
 HeartRate       INT NULL DEFAULT 0,
 BloodPressrue   INT NULL DEFAULT 0);

CREATE TABLE TreatmentDetails
(TreatmentID     INT PRIMARY KEY, HealthHistoryID INT NOT NULL);

CREATE TABLE DiagnosisDetails
(DiagnosisID      INT PRIMARY KEY,
 HealthHistoryID  INT NOT NULL,
 Symptoms         VARCHAR(50) NOT NULL,
 DiagnosisDetails VARCHAR(50) NOT NULL);

CREATE TABLE ReferralDoctorInfo
(ReferralDoctorID     INT PRIMARY KEY,
 HealthHistoryID      INT NOT NULL,
 PhysicianName        VARCHAR(50) NOT NULL,
 DoctorClinicName     VARCHAR(50) NOT NULL,
 DoctorSpecialization VARCHAR(50) NOT NULL);

CREATE TABLE HospitalizationInfo
(HospitalizationID   INT PRIMARY KEY,
 HealthHistoryID     INT NOT NULL,
 HospitalizationDate DATE NOT NULL,
 CheckInTime         TIME NOT NULL,
 DischargeDate       DATE NOT NULL,
 CheckOutTime        TIME NOT NULL);

CREATE TABLE PatientPrimaryDoctor
(EmployeeID INT NOT NULL, PatientID  INT NOT NULL);
/*comments - here i have used DDL statement to create tables in healthcarecenter database. I have also
created columns for each tables with their datatypes and column contraints*/
```

Result Screenshot –



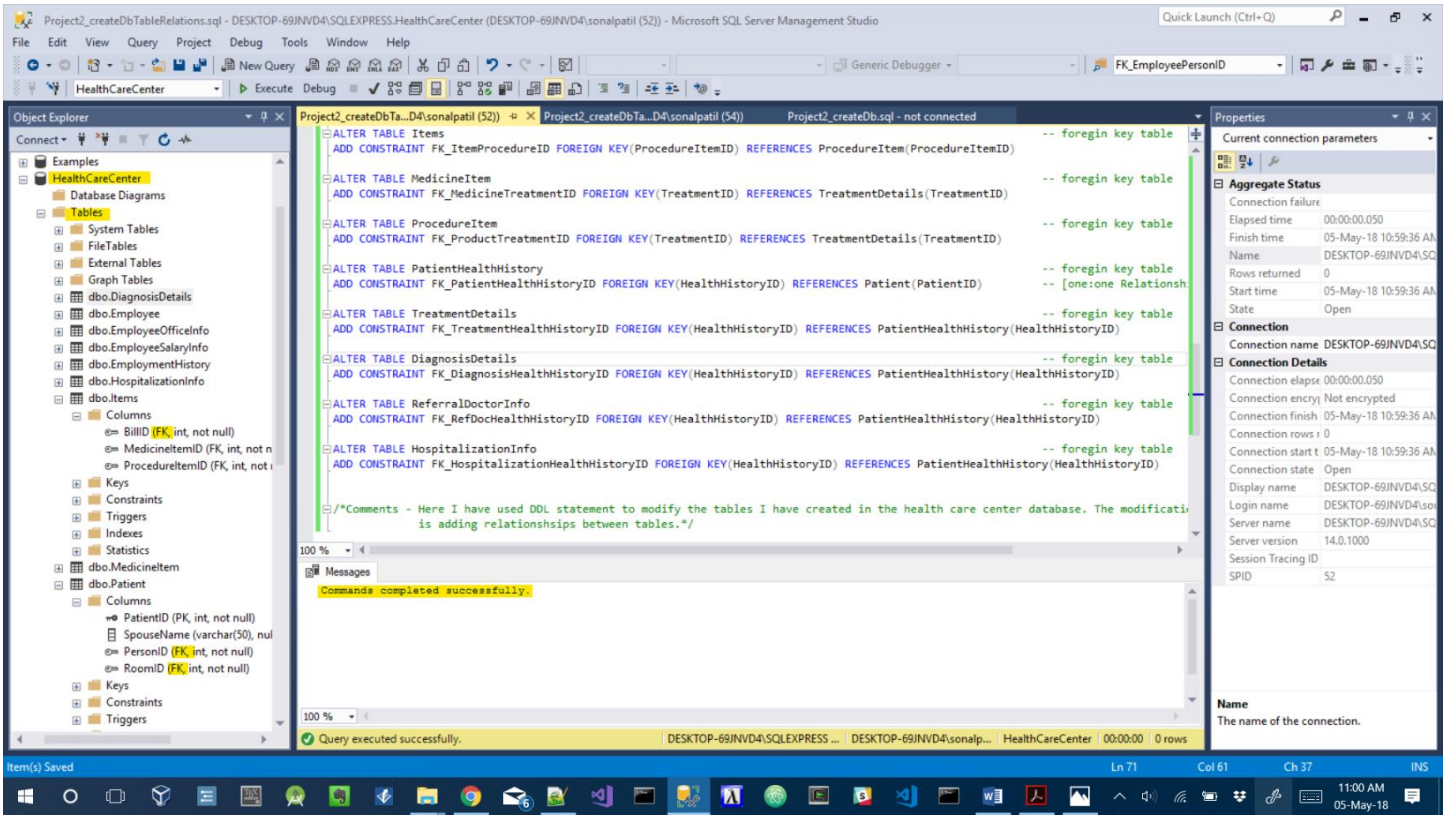
3.3. Creating Relationship between Tables –

Code –

```
USE HealthCareCenter;

ALTER TABLE PersonAddressInfo                                                    -- foregin key table
ADD CONSTRAINT FK_PersonAddressID FOREIGN KEY(PersonID) REFERENCES PersonInfo(PersonID)

ALTER TABLE Employee                                                             -- foregin key table
ADD CONSTRAINT FK_EmployeePersonID FOREIGN KEY(PersonID) REFERENCES PersonInfo(PersonID)

ALTER TABLE Patient                                                              -- foregin key table
ADD CONSTRAINT FK_PatientPersonID FOREIGN KEY(PersonID) REFERENCES PersonInfo(PersonID)

ALTER TABLE EmployeeOfficeInfo                                                   -- foregin key table
ADD CONSTRAINT FK_EmployeeOfficeID FOREIGN KEY(EmployeeID) REFERENCES Employee(EmployeeID) -- [one:one]

ALTER TABLE EmployeeSalaryInfo                                                   -- foregin key table
ADD CONSTRAINT FK_EmployeeSalaryID FOREIGN KEY(EmployeeID) REFERENCES Employee(EmployeeID) -- [one:one]

ALTER TABLE EmploymentHistory                                                    -- foregin key table
ADD CONSTRAINT FK_EmployeeID FOREIGN KEY(EmployeeID) REFERENCES Employee(EmployeeID)

ALTER TABLE PatientPrimaryDoctor                                                 -- foregin key table
ADD CONSTRAINT FK_DoctorID FOREIGN KEY(EmployeeID) REFERENCES Employee(EmployeeID)

ALTER TABLE PatientPrimaryDoctor                                                 -- foregin key table
ADD CONSTRAINT FK_PatientID FOREIGN KEY(PatientID) REFERENCES Patient(PatientID)

ALTER TABLE Patient                                                              -- foregin key table
ADD CONSTRAINT FK_PatientRoomID FOREIGN KEY(RoomID) REFERENCES PatientRoom(RoomID)

ALTER TABLE RoomInfo                                                             -- foregin key table
ADD CONSTRAINT FK_RoomID FOREIGN KEY(RoomID) REFERENCES PatientRoom(RoomID)              -- [one:one]

ALTER TABLE RoomMedicalDeviceInfo                                                -- foregin key table
ADD CONSTRAINT FK_RoomDeviceID FOREIGN KEY(RoomID) REFERENCES PatientRoom(RoomID)

ALTER TABLE PatientHealthInsurance                                               -- foregin key table
ADD CONSTRAINT FK_PatientInsuranceID FOREIGN KEY(PatientID) REFERENCES Patient(PatientID)

ALTER TABLE Visitors                                                             -- foregin key table
ADD CONSTRAINT FK_PatientVisitorID FOREIGN KEY(PatientID) REFERENCES Patient(PatientID)

ALTER TABLE VisitorPhoto                                                         -- foregin key table
ADD CONSTRAINT FK_VisitorID FOREIGN KEY(VisitorID) REFERENCES Visitors(VisitorID)        -- [one:one]

ALTER TABLE PatientBills                                                         -- foregin key table
ADD CONSTRAINT FK_PatientBillID FOREIGN KEY(PatientID) REFERENCES Patient(PatientID)

ALTER TABLE Items                                                                -- foregin key table
ADD CONSTRAINT FK_BillID FOREIGN KEY(BillID) REFERENCES PatientBills(BillID)

ALTER TABLE Items                                                                -- foregin key table
ADD CONSTRAINT FK_ItemMedicineID FOREIGN KEY(MedicineItemID) REFERENCES MedicineItem(MedicineItemID)

ALTER TABLE Items                                                                -- foregin key table
ADD CONSTRAINT FK_ItemProcedureID FOREIGN KEY(ProcedureItemID) REFERENCES
ProcedureItem(ProcedureItemID)
```

```
ALTER TABLE MedicineItem                                                    -- foregin key table
ADD CONSTRAINT FK_MedicineTreatmentID FOREIGN KEY(TreatmentID) REFERENCES TreatmentDetails(TreatmentID)

ALTER TABLE ProcedureItem                                                   -- foregin key table
ADD CONSTRAINT FK_ProductTreatmentID FOREIGN KEY(TreatmentID) REFERENCES TreatmentDetails(TreatmentID)

ALTER TABLE PatientHealthHistory                                            -- foregin key table
ADD CONSTRAINT FK_HealthHistoryID FOREIGN KEY(HealthHistoryID) REFERENCES Patient(PatientID)–[one:one]

ALTER TABLE TreatmentDetails                                                -- foregin key table
ADD CONSTRAINT FK_TreatmentHealthHistoryID FOREIGN KEY(HealthHistoryID) REFERENCES
PatientHealthHistory(HealthHistoryID)

ALTER TABLE DiagnosisDetails                                                -- foregin key table
ADD CONSTRAINT FK_DiagnosisHealthHistoryID FOREIGN KEY(HealthHistoryID) REFERENCES
PatientHealthHistory(HealthHistoryID)

ALTER TABLE ReferralDoctorInfo                                              -- foregin key table
ADD CONSTRAINT FK_RefDocHealthHistoryID FOREIGN KEY(HealthHistoryID) REFERENCES
PatientHealthHistory(HealthHistoryID)

ALTER TABLE HospitalizationInfo                                             -- foregin key table
ADD CONSTRAINT FK_HospitalizationHealthHistoryID FOREIGN KEY(HealthHistoryID) REFERENCES
PatientHealthHistory(HealthHistoryID)

/*Comments - Here I have used DDL statement to modify the tables I have created in the health care
center database. The modification is adding relationshsips between tables.*/
```

Result Screenshot –



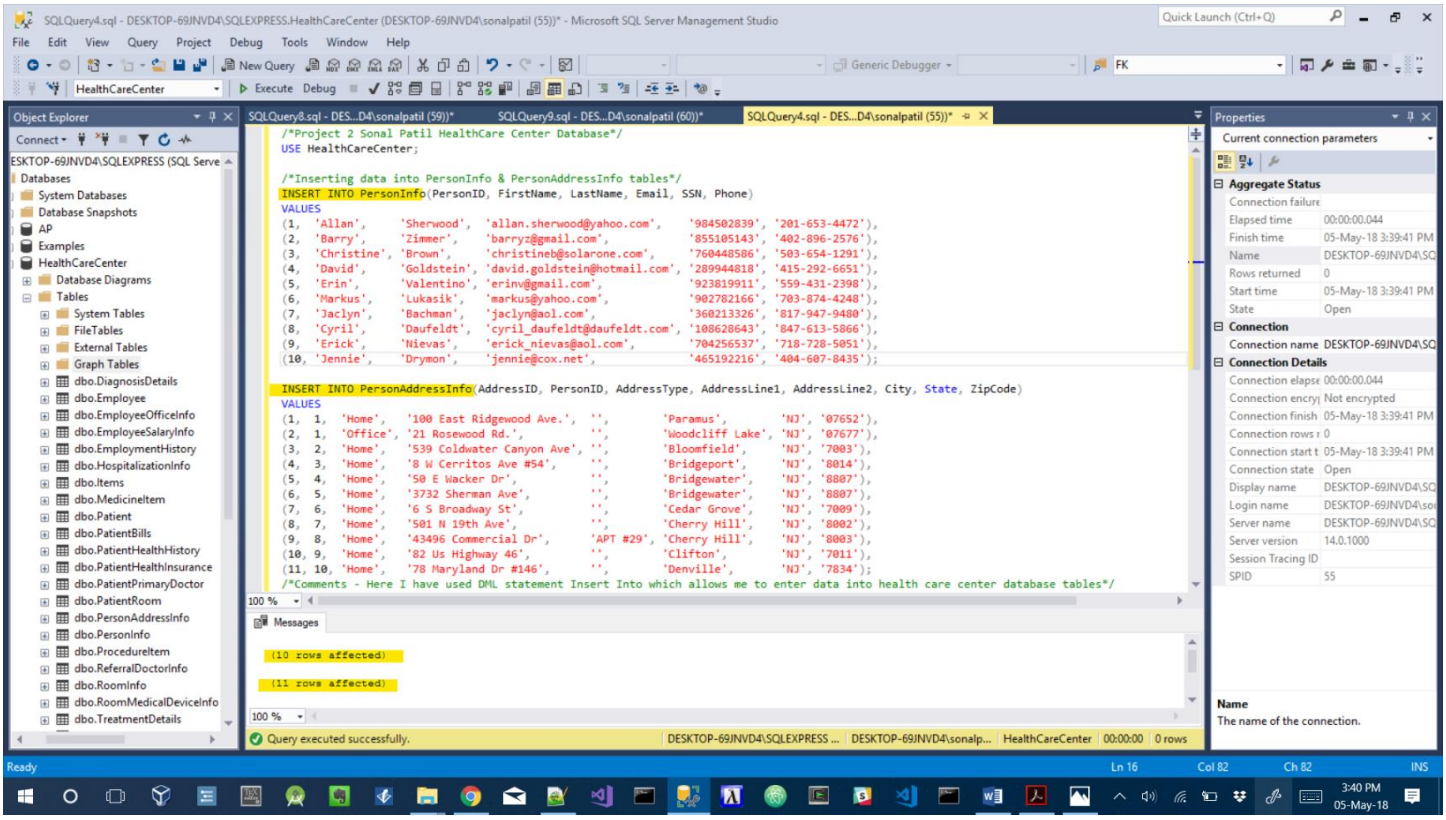
3.4. Inserting Data into Database Tables & Showing inserted data –

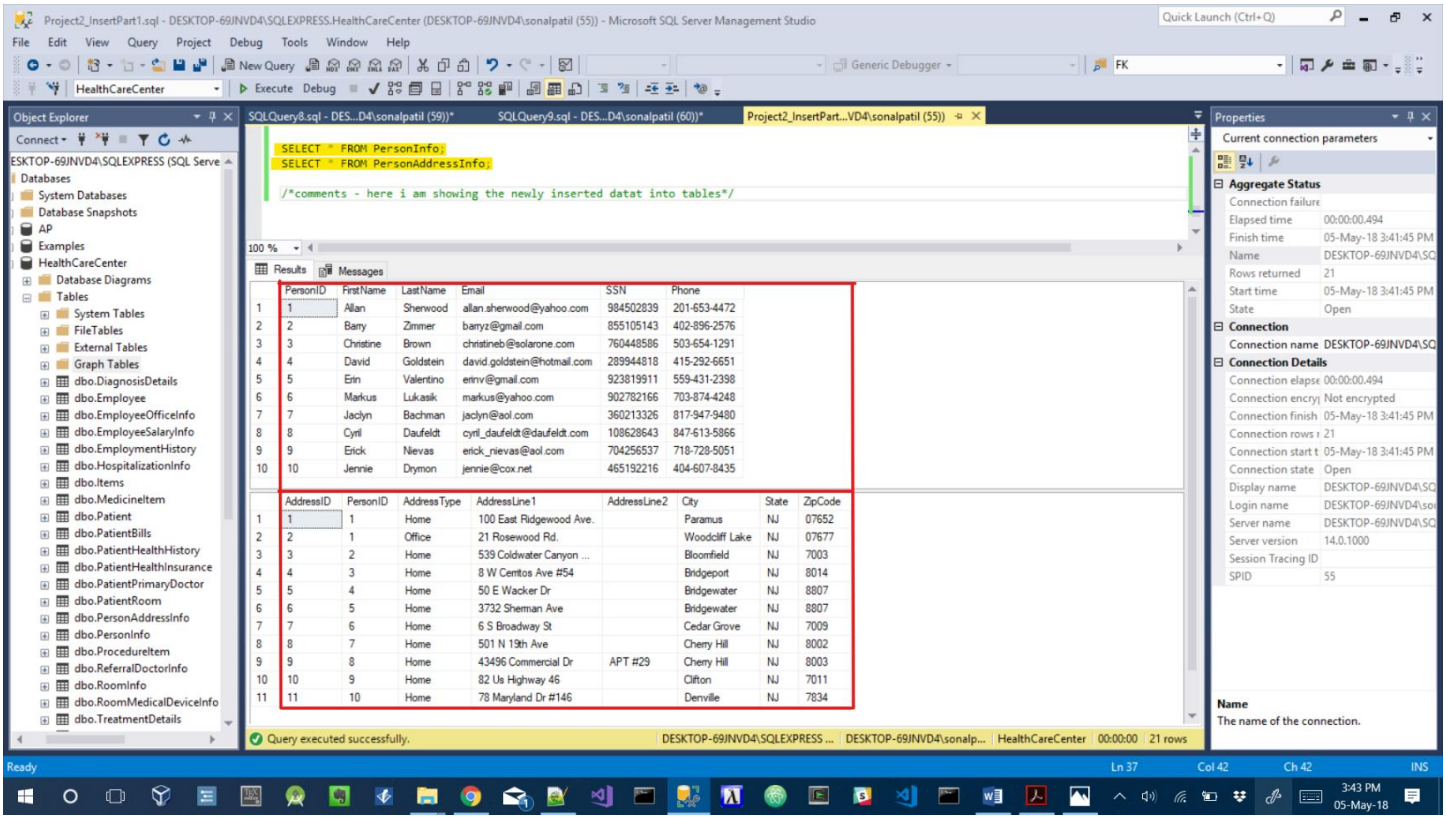
1. **Inserting data into Tables –** PersonInfo & PersonAddressInfo

Code –

```
USE HealthCareCenter;

/*Inserting data into PersonInfo & PersonAddressInfo tables*/
INSERT INTO PersonInfo(PersonID, FirstName, LastName, Email, SSN, Phone)
VALUES
(1,  'Allan',     'Sherwood',  'allan.sherwood@yahoo.com',    '984502839', '201-653-4472'),
(2,  'Barry',     'Zimmer',    'barryz@gmail.com',            '855105143', '402-896-2576'),
(3,  'Christine', 'Brown',     'christineb@solarone.com',     '760448586', '503-654-1291'),
(4,  'David',     'Goldstein', 'david.goldstein@hotmail.com', '289944818', '415-292-6651'),
(5,  'Erin',      'Valentino', 'erinv@gmail.com',             '923819911', '559-431-2398'),
(6,  'Markus',    'Lukasik',   'markus@yahoo.com',            '902782166', '703-874-4248'),
(7,  'Jaclyn',    'Bachman',   'jaclyn@aol.com',              '360213326', '817-947-9480'),
(8,  'Cyril',     'Daufeldt',  'cyril_daufeldt@daufeldt.com', '108628643', '847-613-5866'),
(9,  'Erick',     'Nievas',    'erick_nievas@aol.com',        '704256537', '718-728-5051'),
(10, 'Jennie',    'Drymon',    'jennie@cox.net',              '465192216', '404-607-8435');

INSERT INTO PersonAddressInfo(AddressID, PersonID, AddressType, AddressLine1, AddressLine2, City,
State, ZipCode)
VALUES
(1,  1,  'Home',   '100 East Ridgewood Ave.',  '',        'Paramus',        'NJ', '07652'),
(2,  1,  'Office', '21 Rosewood Rd.',          '',        'Woodcliff Lake', 'NJ', '07677'),
(3,  2,  'Home',   '539 Coldwater Canyon Ave', '',        'Bloomfield',     'NJ', '7003'),
(4,  3,  'Home',   '8 W Cerritos Ave #54',     '',        'Bridgeport',     'NJ', '8014'),
(5,  4,  'Home',   '50 E Wacker Dr',           '',        'Bridgewater',    'NJ', '8807'),
(6,  5,  'Home',   '3732 Sherman Ave',         '',        'Bridgewater',    'NJ', '8807'),
(7,  6,  'Home',   '6 S Broadway St',          '',        'Cedar Grove',    'NJ', '7009'),
(8,  7,  'Home',   '501 N 19th Ave',           '',        'Cherry Hill',    'NJ', '8002'),
(9,  8,  'Home',   '43496 Commercial Dr',      'APT #29', 'Cherry Hill',    'NJ', '8003'),
(10, 9,  'Home',   '82 Us Highway 46',         '',        'Clifton',        'NJ', '7011'),
(11, 10, 'Home',   '78 Maryland Dr #146',      '',        'Denville',       'NJ', '7834');

/*Comments - Here I have used DML statement Insert Into which allows me to enter data into health care
center database tables*/

SELECT * FROM PersonInfo;
SELECT * FROM PersonAddressInfo;


/*comments - here I am showing the newly inserted data into tables*/
```

Result Screenshots –



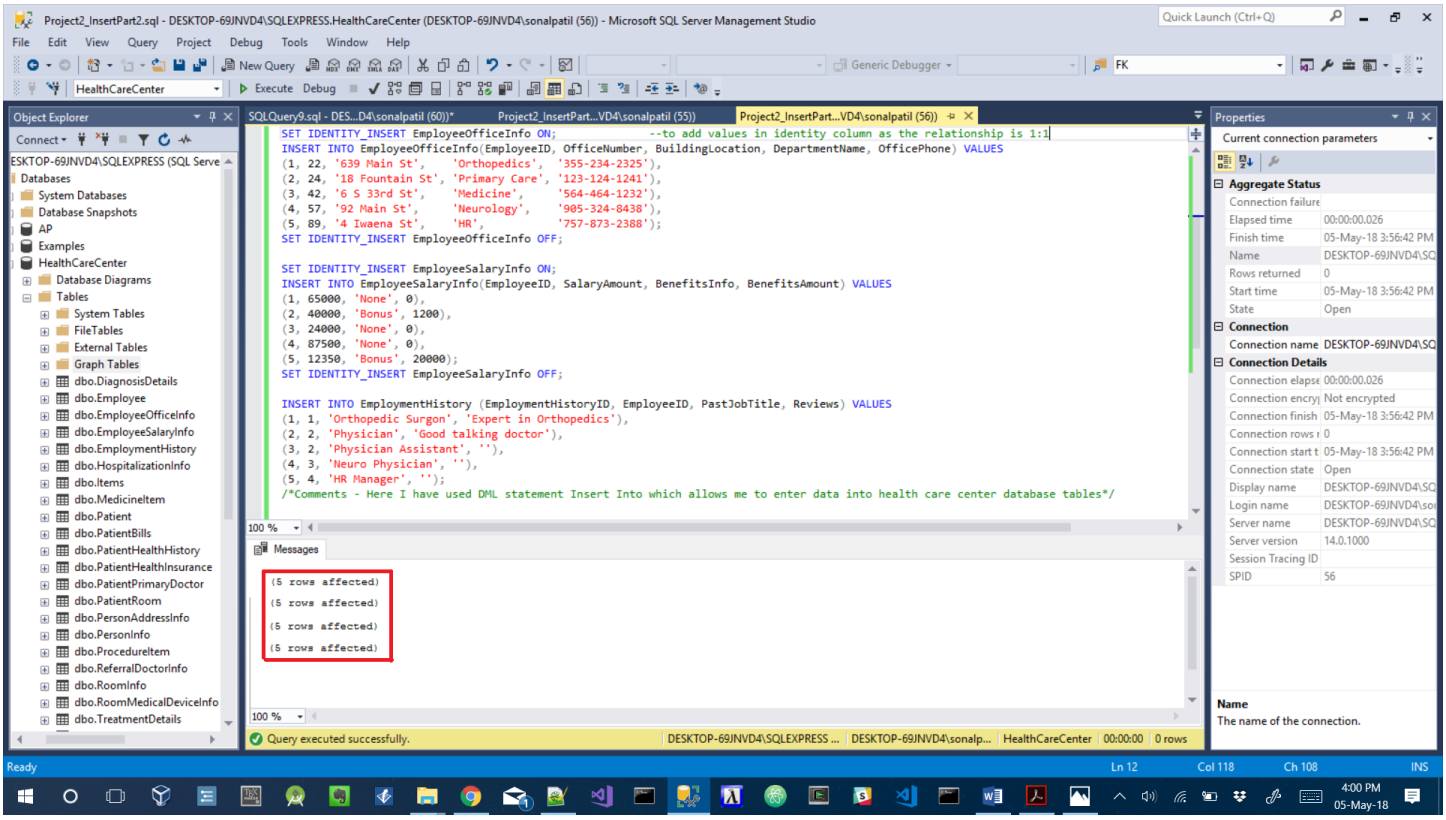


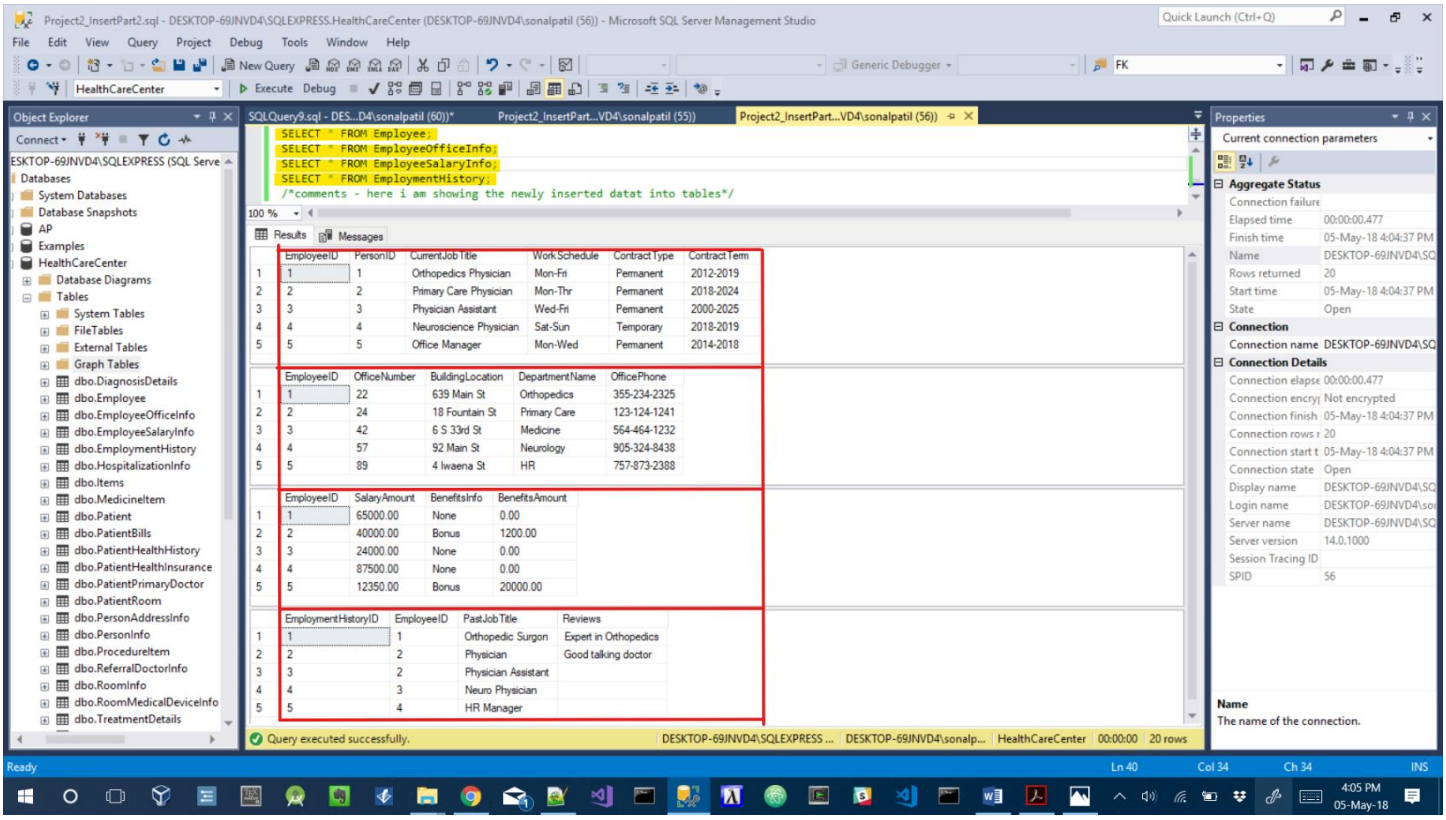
2. **Inserting data into Tables –** Employee,EmployeeSalaryInfo, EmployeeOfficeInfo & EmploymentHistory

Code –

```
USE HealthCareCenter;

/*Inserting data into Employee, EmployeeSalaryInfo, EmployeeOfficeInfo & EmploymentHistory tables*/
INSERT INTO Employee (EmployeeID, PersonID, CurrentJobTitle, WorkSchedule, ContractType, ContractTerm)
VALUES
(1, 1, 'Orthopedics Physician',   'Mon-Fri', 'Permanent', '2012-2019'),
(2, 2, 'Primary Care Physician',  'Mon-Thr', 'Permanent', '2018-2024'),
(3, 3, 'Physician Assistant',     'Wed-Fri', 'Permanent', '2000-2025'),
(4, 4, 'Neuroscience Physician',  'Sat-Sun', 'Temporary', '2018-2019'),
(5, 5, 'Office Manager ',         'Mon-Wed', 'Permanent', '2014-2018');

SET IDENTITY_INSERT EmployeeOfficeInfo ON;                          --to add values in identity column
as the relationship is 1:1
INSERT INTO EmployeeOfficeInfo(EmployeeID, OfficeNumber, BuildingLocation, DepartmentName, OfficePhone)
VALUES
(1, 22, '639 Main St',    'Orthopedics',  '355-234-2325'),
(2, 24, '18 Fountain St', 'Primary Care', '123-124-1241'),
(3, 42, '6 S 33rd St',    'Medicine',     '564-464-1232'),
(4, 57, '92 Main St',     'Neurology',    '905-324-8438'),
(5, 89, '4 Iwaena St',    'HR',           '757-873-2388');
SET IDENTITY_INSERT EmployeeOfficeInfo OFF;

SET IDENTITY_INSERT EmployeeSalaryInfo ON;
INSERT INTO EmployeeSalaryInfo(EmployeeID, SalaryAmount, BenefitsInfo, BenefitsAmount) VALUES
(1, 65000, 'None', 0),
(2, 40000, 'Bonus', 1200),
(3, 24000, 'None', 0),
(4, 87500, 'None', 0),
(5, 12350, 'Bonus', 20000);
SET IDENTITY_INSERT EmployeeSalaryInfo OFF;

INSERT INTO EmploymentHistory (EmploymentHistoryID, EmployeeID, PastJobTitle, Reviews) VALUES
(1, 1, 'Orthopedic Surgon', 'Expert in Orthopedics'),
(2, 2, 'Physician', 'Good talking doctor'),
(3, 2, 'Physician Assistant', ''),
(4, 3, 'Neuro Physician', ''),
(5, 4, 'HR Manager', '');
/*Comments - Here I have used DML statement Insert Into which allows me to enter data into health care
center database tables*/

SELECT * FROM Employee;
SELECT * FROM EmployeeOfficeInfo;
SELECT * FROM EmployeeSalaryInfo;
SELECT * FROM EmploymentHistory;

/*comments - here I am showing the newly inserted data into tables*/
```

Result Screenshots –



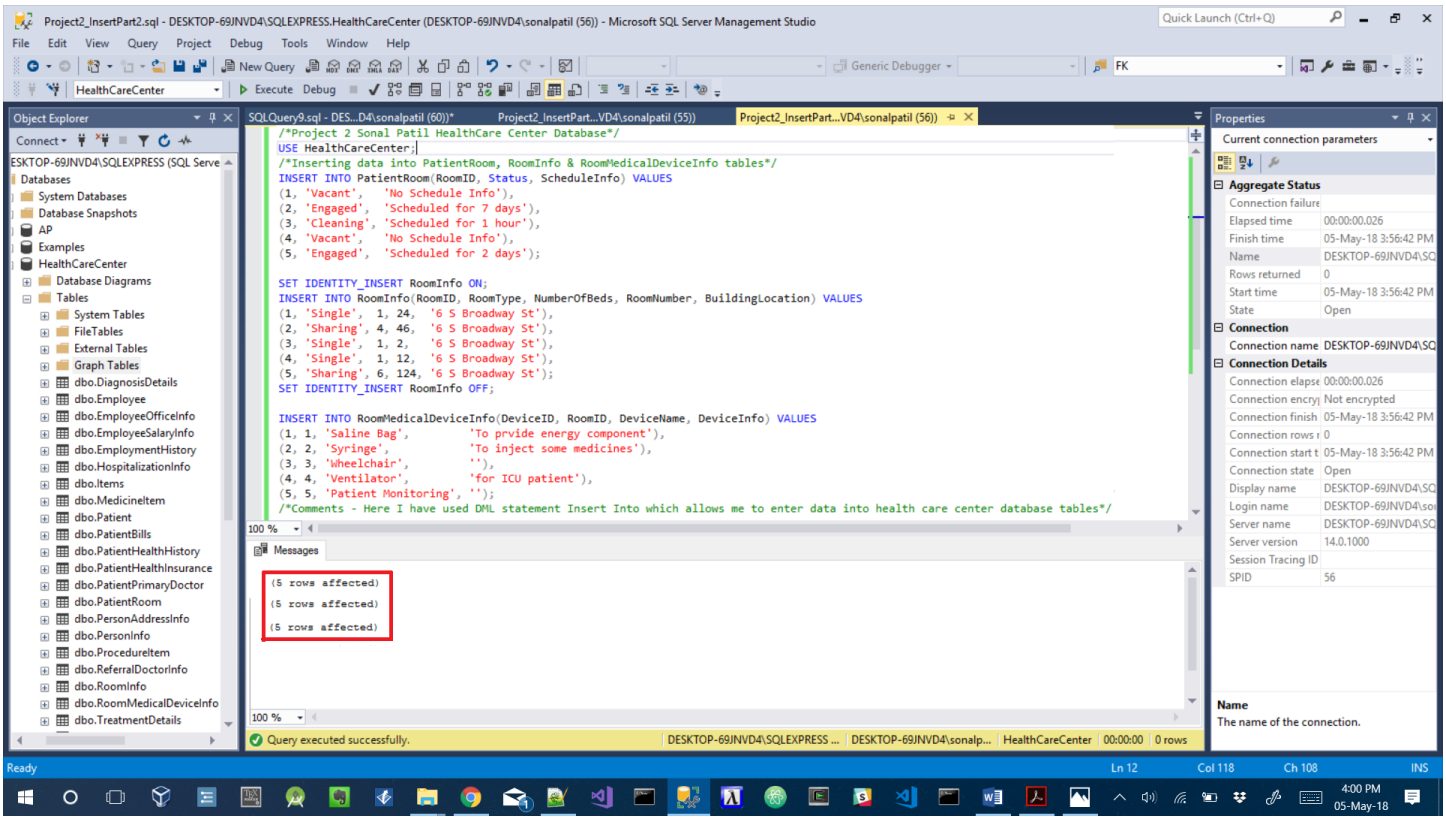


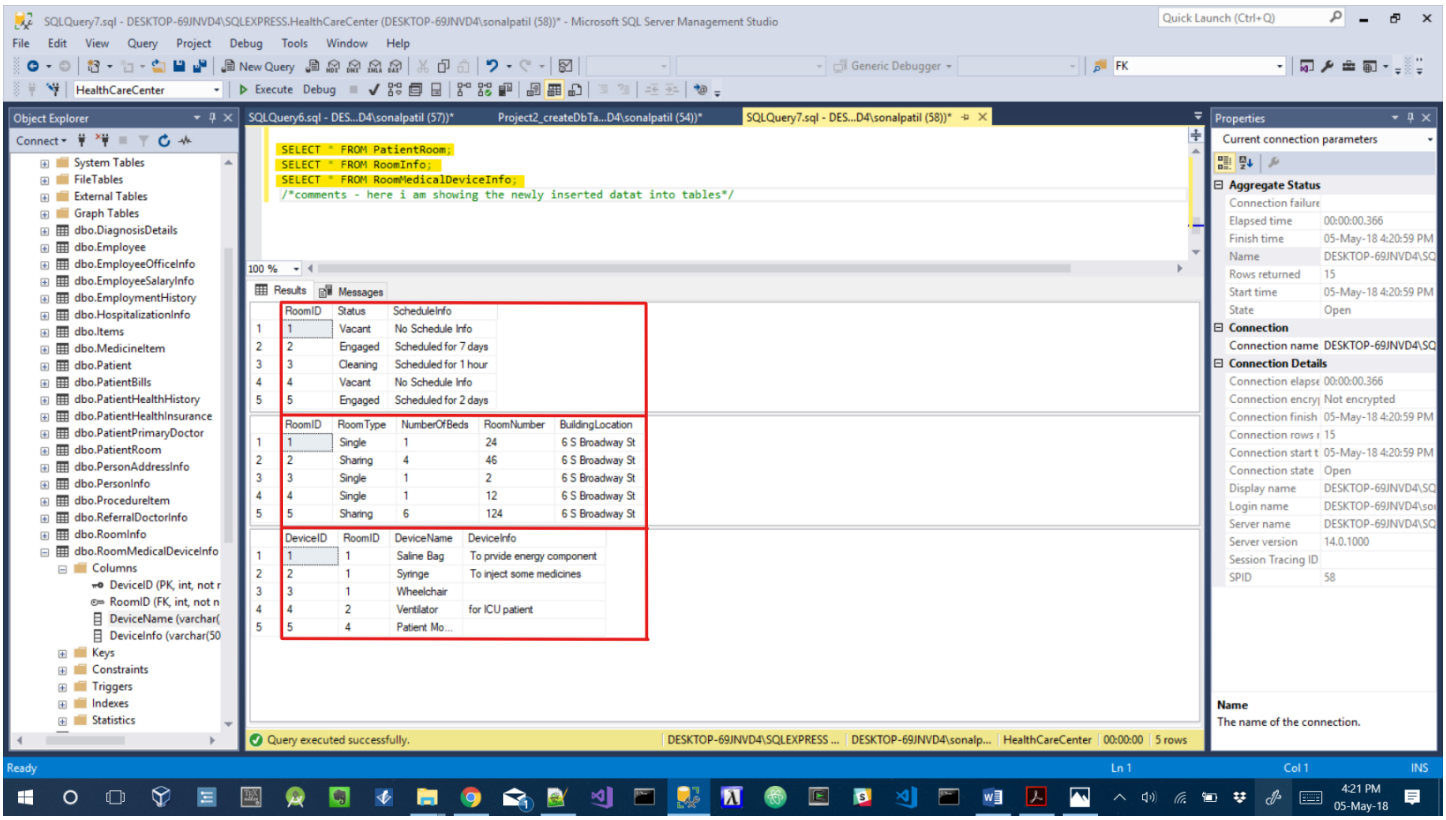
3. **Inserting data into Tables –** PatientRoom, RoomInfo & RoomMedicalDeviceInfo

Code –

```
USE HealthCareCenter;

/*Inserting data into PatientRoom, RoomInfo & RoomMedicalDeviceInfo tables*/
INSERT INTO PatientRoom(RoomID, Status, ScheduleInfo) VALUES
(1, 'Vacant',   'No Schedule Info'),
(2, 'Engaged',  'Scheduled for 7 days'),
(3, 'Cleaning', 'Scheduled for 1 hour'),
(4, 'Vacant',   'No Schedule Info'),
(5, 'Engaged',  'Scheduled for 2 days');

SET IDENTITY_INSERT RoomInfo ON;
INSERT INTO RoomInfo(RoomID, RoomType, NumberOfBeds, RoomNumber, BuildingLocation) VALUES
(1, 'Single',  1, 24,  '6 S Broadway St'),
(2, 'Sharing', 4, 46,  '6 S Broadway St'),
(3, 'Single',  1, 2,   '6 S Broadway St'),
(4, 'Single',  1, 12,  '6 S Broadway St'),
(5, 'Sharing', 6, 124, '6 S Broadway St');
SET IDENTITY_INSERT RoomInfo OFF;

INSERT INTO RoomMedicalDeviceInfo(DeviceID, RoomID, DeviceName, DeviceInfo) VALUES
(1, 1, 'Saline Bag',         'To prvide energy component'),
(2, 2, 'Syringe',            'To inject some medicines'),
(3, 3, 'Wheelchair',         ''),
(4, 4, 'Ventilator',         'for ICU patient'),
(5, 5, 'Patient Monitoring', '');
/*Comments - Here I have used DML statement Insert Into which allows me to enter data into health care
center database tables*/

SELECT * FROM PatientRoom;
SELECT * FROM RoomInfo;
SELECT * FROM RoomMedicalDeviceInfo;

/*comments - here I am showing the newly inserted data into tables*/
```

Result Screenshots –



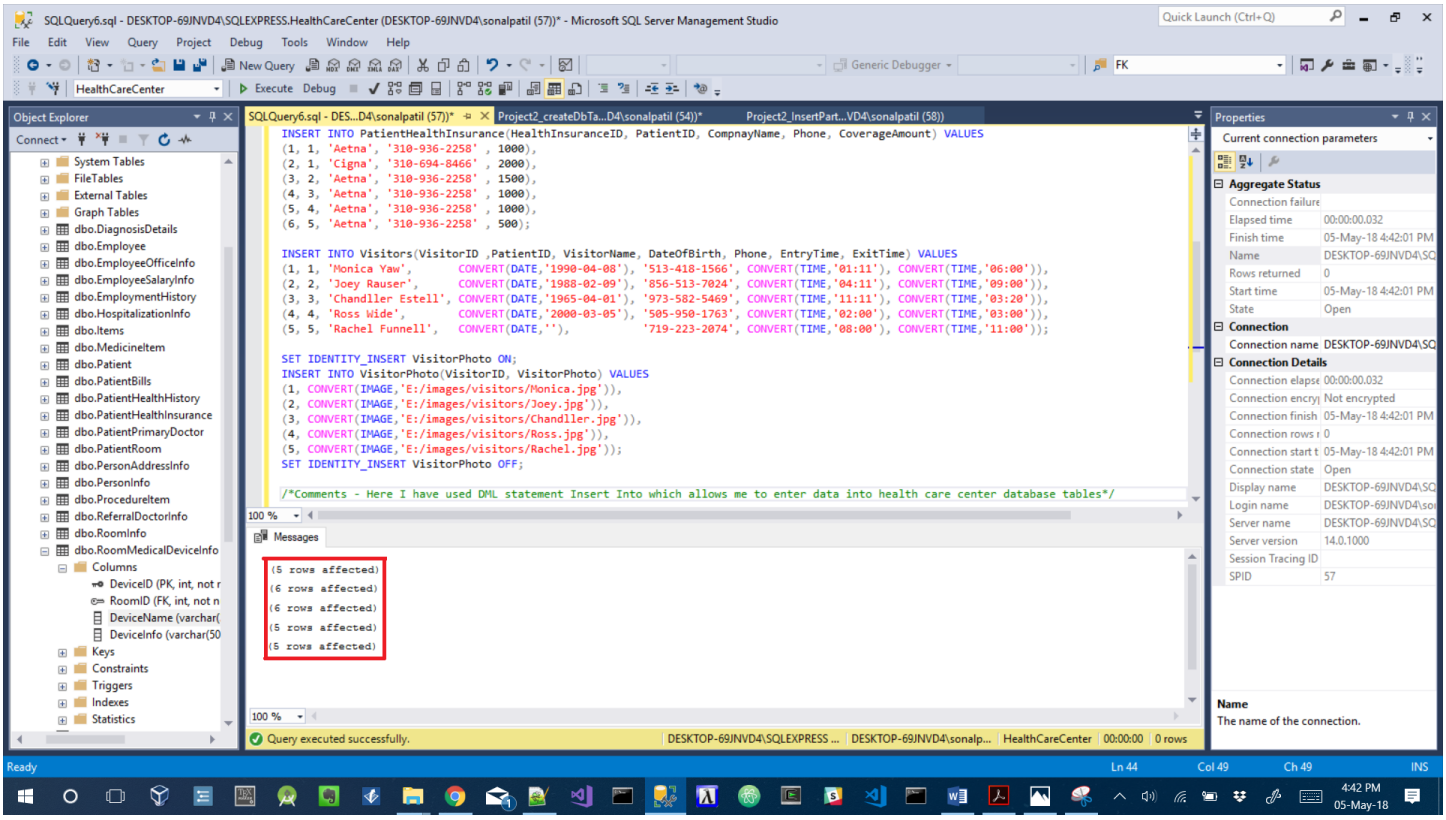


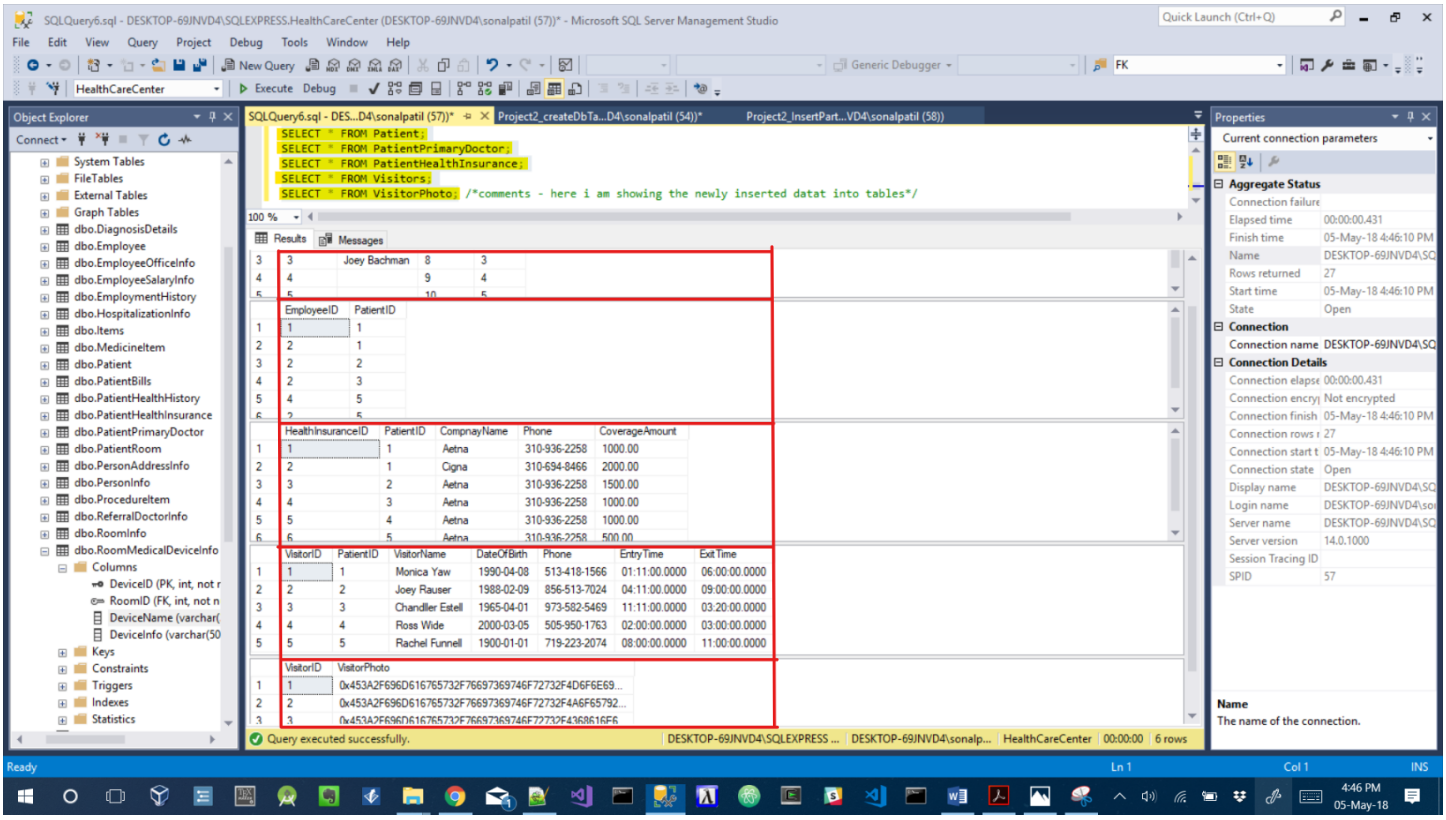
4. **Inserting data into Tables –** Patient, PatientPrimaryDoctor, PatientHealthInsurance, Visitors & VisitorsPhoto

Code –

```
USE HealthCareCenter;
/*Inserting data into Patient, PatientPrimaryDoctor, PatientHealthInsurance, Visitors & VisitorsPhoto tables*/

INSERT INTO Patient(PatientID, SpouseName, PersonID, RoomID) VALUES
(1, 'Erica Lukasik', 6,  1),
(2, '',              7,  2),
(3, 'Joey Bachman',  8,  3),
(4, '',              9,  4),
(5, '',              10, 5);

INSERT INTO PatientPrimaryDoctor(EmployeeID, PatientID) VALUES – many:many relation so linking table
(1,1),
(2,1),
(2,2),
(2,3),
(4,5),
(2,5);

INSERT INTO PatientHealthInsurance(HealthInsuranceID, PatientID, CompnayName, Phone, CoverageAmount)
VALUES
(1, 1, 'Aetna', '310-936-2258' , 1000),
(2, 1, 'Cigna', '310-694-8466' , 2000),
(3, 2, 'Aetna', '310-936-2258' , 1500),
(4, 3, 'Aetna', '310-936-2258' , 1000),
(5, 4, 'Aetna', '310-936-2258' , 1000),
(6, 5, 'Aetna', '310-936-2258' , 500);

INSERT INTO Visitors(VisitorID ,PatientID, VisitorName, DateOfBirth, Phone, EntryTime, ExitTime) VALUES
(1, 1, 'Monica Yaw',       CONVERT(DATE,'1990-04-08'), '513-418-1566', CONVERT(TIME,'01:11'),
CONVERT(TIME,'06:00')),
(2, 2, 'Joey Rauser',      CONVERT(DATE,'1988-02-09'), '856-513-7024', CONVERT(TIME,'04:11'),
CONVERT(TIME,'09:00')),
(3, 3, 'Chandller Estell', CONVERT(DATE,'1965-04-01'), '973-582-5469', CONVERT(TIME,'11:11'),
CONVERT(TIME,'03:20')),
(4, 4, 'Ross Wide',        CONVERT(DATE,'2000-03-05'), '505-950-1763', CONVERT(TIME,'02:00'),
CONVERT(TIME,'03:00')),
(5, 5, 'Rachel Funnell',   CONVERT(DATE,''),           '719-223-2074', CONVERT(TIME,'08:00'),
CONVERT(TIME,'11:00'));

SET IDENTITY_INSERT VisitorPhoto ON;
INSERT INTO VisitorPhoto(VisitorID, VisitorPhoto) VALUES
(1, CONVERT(IMAGE,'E:/images/visitors/Monica.jpg')),
(2, CONVERT(IMAGE,'E:/images/visitors/Joey.jpg')),
(3, CONVERT(IMAGE,'E:/images/visitors/Chandller.jpg')),
(4, CONVERT(IMAGE,'E:/images/visitors/Ross.jpg')),
(5, CONVERT(IMAGE,'E:/images/visitors/Rachel.jpg'));
SET IDENTITY_INSERT VisitorPhoto OFF;

/*Comments - Here I have used DML statement Insert Into which allows me to enter data into  tables*/
SELECT * FROM Patient;
SELECT * FROM PatientPrimaryDoctor;
SELECT * FROM PatientHealthInsurance;
SELECT * FROM Visitors;
SELECT * FROM VisitorPhoto; /*comments - here i am showing the newly inserted data into tables*/
```

Result Screenshots –



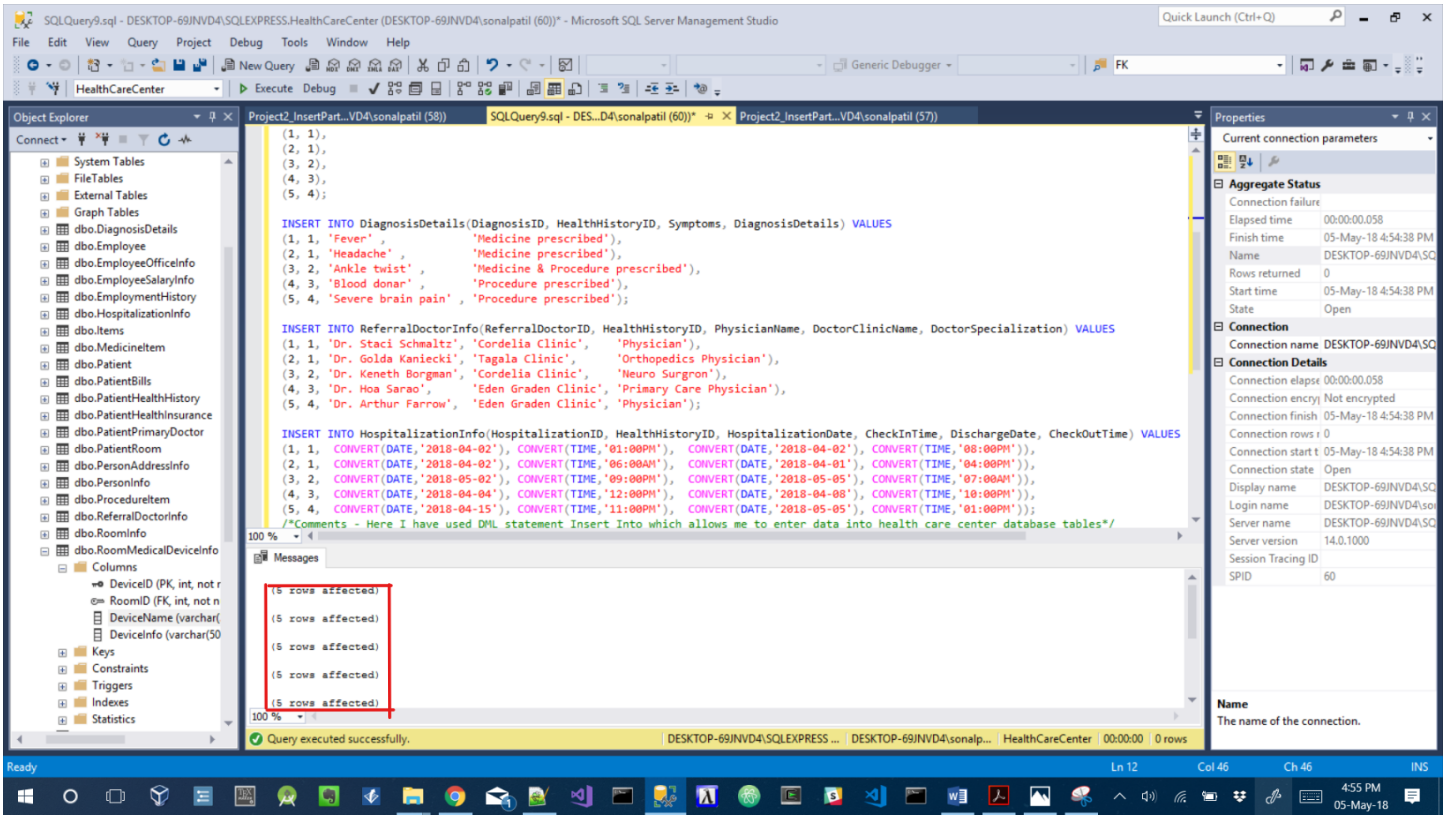


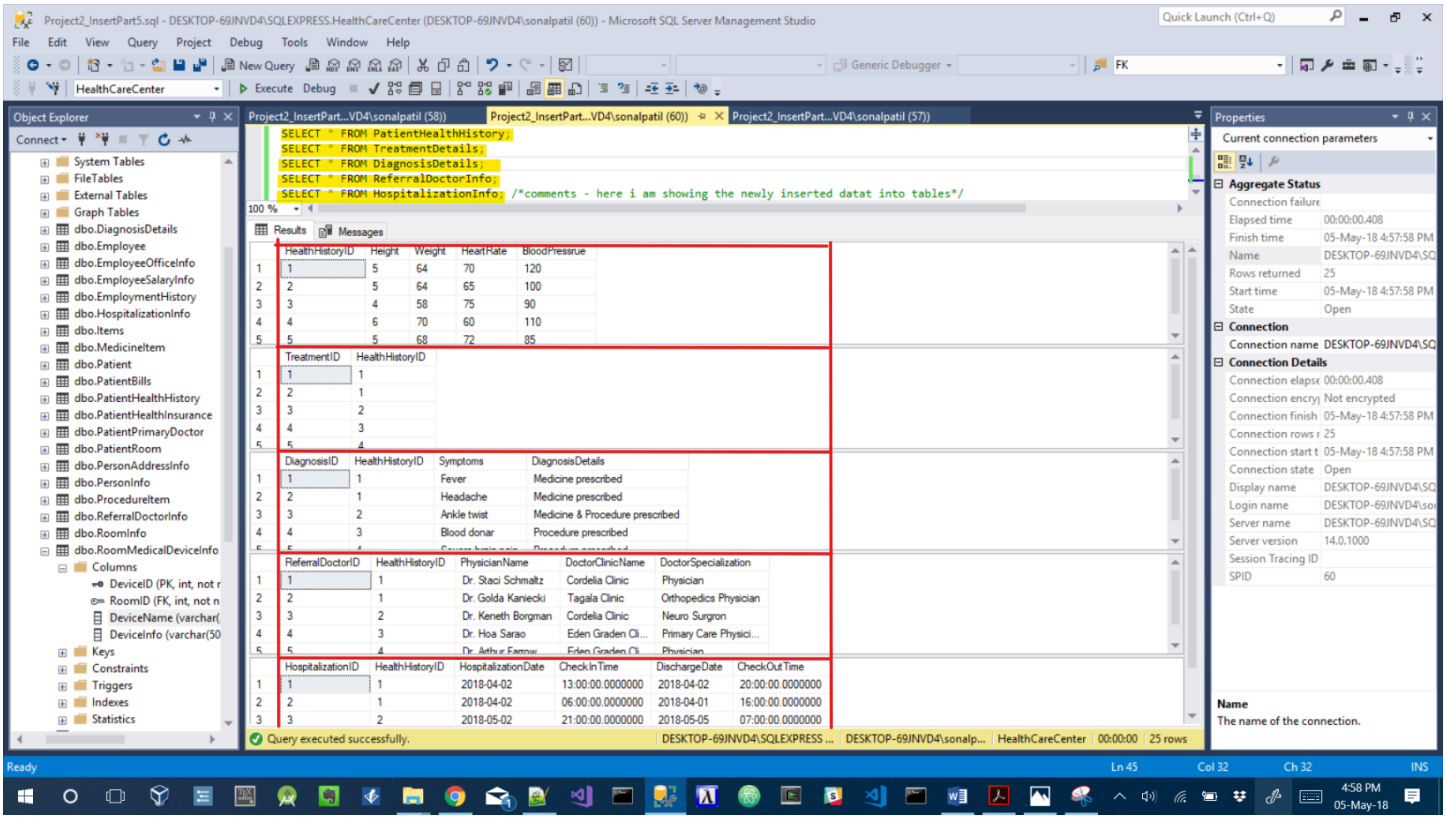
5. **Inserting data into Tables –** PatientHealthHistory, TreatmentDetails, DiagnosisDetails, HospitalizationInfo & ReferralDoctorInfo

Code –

```
USE HealthCareCenter;
/*Inserting data into PatientHealthHistory, TreatmentDetails, DiagnosisDetails, HospitalizationInfo & ReferralDoctorInfo tables*/

SET IDENTITY_INSERT PatientHealthHistory ON;
INSERT INTO PatientHealthHistory(HealthHistoryID, Height, Weight, HeartRate, BloodPressrue) VALUES
(1, 5.5 , 64, 70, 120),
(2, 5.0 , 64, 65, 100),
(3, 4.5 , 58, 75, 90),
(4, 6.0 , 70, 60, 110),
(5, 5.8 , 68, 72, 85);
SET IDENTITY_INSERT PatientHealthHistory OFF;

INSERT INTO TreatmentDetails(TreatmentID, HealthHistoryID) VALUES
(1, 1),
(2, 1),
(3, 2),
(4, 3),
(5, 4);

INSERT INTO DiagnosisDetails(DiagnosisID, HealthHistoryID, Symptoms, DiagnosisDetails) VALUES
(1, 1, 'Fever' ,             'Medicine prescribed'),
(2, 1, 'Headache' ,          'Medicine prescribed'),
(3, 2, 'Ankle twist' ,       'Medicine & Procedure prescribed'),
(4, 3, 'Blood donar' ,       'Procedure prescribed'),
(5, 4, 'Severe brain pain' , 'Procedure prescribed');

INSERT INTO ReferralDoctorInfo(ReferralDoctorID, HealthHistoryID, PhysicianName, DoctorClinicName, DoctorSpecialization) VALUES
(1, 1, 'Dr. Staci Schmaltz', 'Cordelia Clinic',    'Physician'),
(2, 1, 'Dr. Golda Kaniecki', 'Tagala Clinic',      'Orthopedics Physician'),
(3, 2, 'Dr. Keneth Borgman', 'Cordelia Clinic',    'Neuro Surgron'),
(4, 3, 'Dr. Hoa Sarao',      'Eden Graden Clinic', 'Primary Care Physician'),
(5, 4, 'Dr. Arthur Farrow',  'Eden Graden Clinic', 'Physician');

INSERT INTO HospitalizationInfo(HospitalizationID, HealthHistoryID, HospitalizationDate, CheckInTime, DischargeDate, CheckOutTime) VALUES
(1, 1,  CONVERT(DATE,'2018-04-02'), CONVERT(TIME,'01:00PM'),  CONVERT(DATE,'2018-04-02'), CONVERT(TIME,'08:00PM')),
(2, 1,  CONVERT(DATE,'2018-04-02'), CONVERT(TIME,'06:00AM'),  CONVERT(DATE,'2018-04-01'), CONVERT(TIME,'04:00PM')),
(3, 2,  CONVERT(DATE,'2018-05-02'), CONVERT(TIME,'09:00PM'),  CONVERT(DATE,'2018-05-05'), CONVERT(TIME,'07:00AM')),
(4, 3,  CONVERT(DATE,'2018-04-04'), CONVERT(TIME,'12:00PM'),  CONVERT(DATE,'2018-04-08'), CONVERT(TIME,'10:00PM')),
(5, 4,  CONVERT(DATE,'2018-04-15'), CONVERT(TIME,'11:00PM'),  CONVERT(DATE,'2018-05-05'), CONVERT(TIME,'01:00PM'));
/*Comments - Here I have used DML statement Insert Into which allows me to enter data into health care center database tables*/

SELECT * FROM PatientHealthHistory;
SELECT * FROM TreatmentDetails;
SELECT * FROM DiagnosisDetails;
SELECT * FROM ReferralDoctorInfo;
SELECT * FROM HospitalizationInfo;
/*comments - here i am showing the newly inserted data into tables*/
```

Result Screenshots –



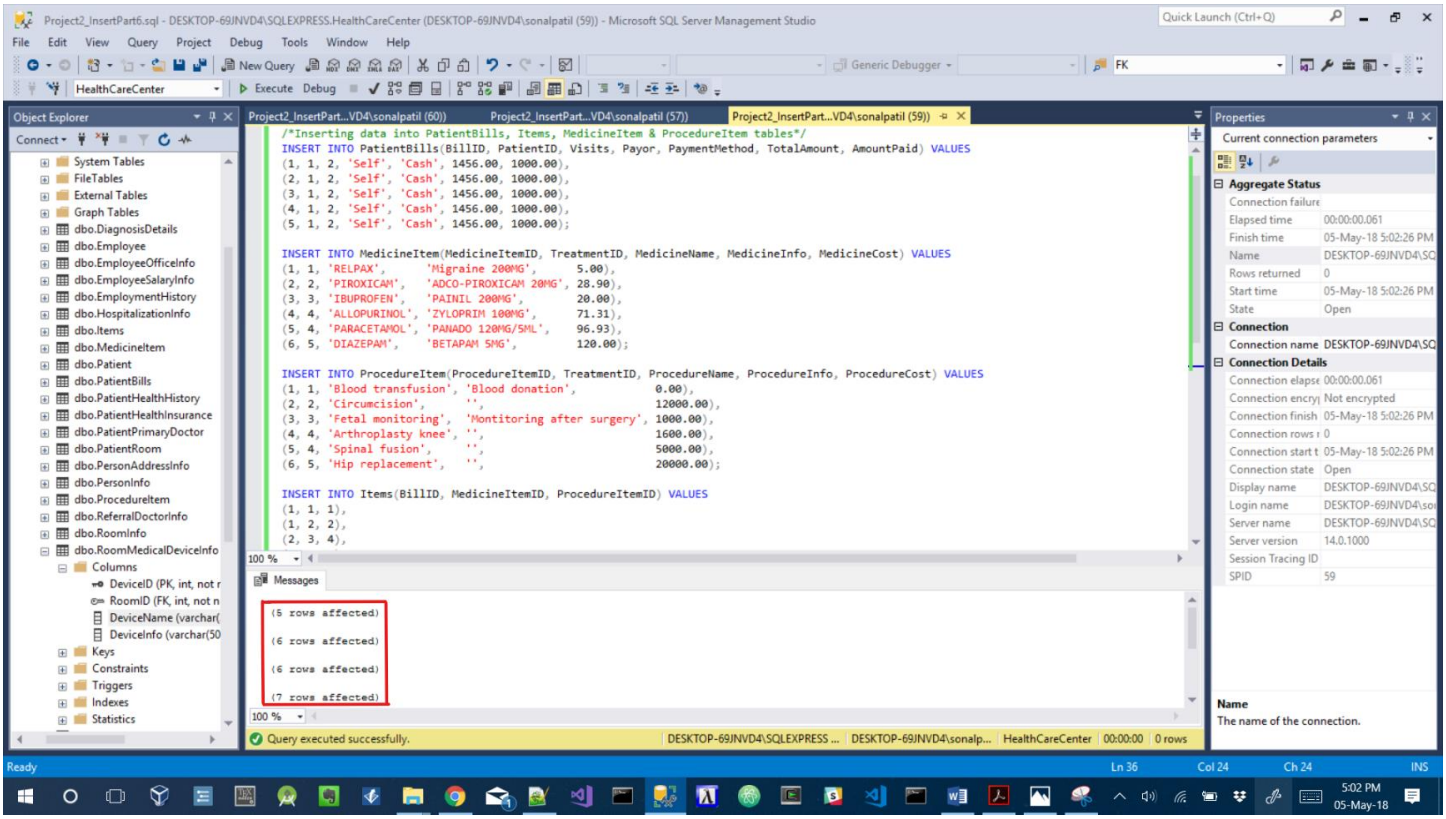


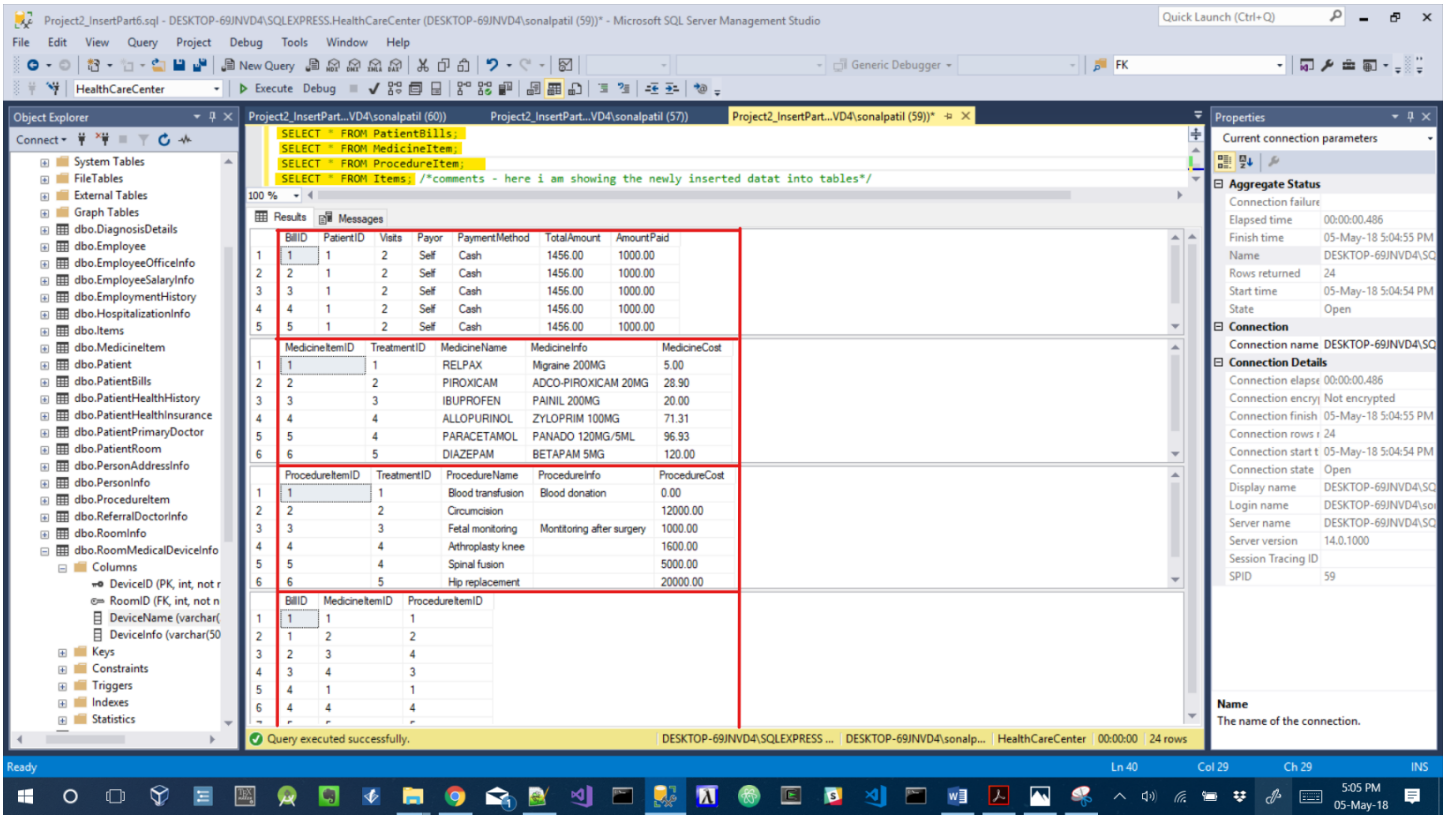
6. **Inserting data into Tables –** PatientBills, Items, MedicineItem & ProcedureItem

Code –

```
USE HealthCareCenter;

/*Inserting data into PatientBills, Items, MedicineItem & ProcedureItem tables*/
INSERT INTO PatientBills(BillID, PatientID, Visits, Payor, PaymentMethod, TotalAmount, AmountPaid)
VALUES
(1, 1, 2, 'Self', 'Cash', 1456.00, 1000.00),
(2, 1, 2, 'Self', 'Cash', 1456.00, 1000.00),
(3, 1, 2, 'Self', 'Cash', 1456.00, 1000.00),
(4, 1, 2, 'Self', 'Cash', 1456.00, 1000.00),
(5, 1, 2, 'Self', 'Cash', 1456.00, 1000.00);

INSERT INTO MedicineItem(MedicineItemID, TreatmentID, MedicineName, MedicineInfo, MedicineCost) VALUES
(1, 1, 'RELPAX',       'Migraine 200MG',      5.00),
(2, 2, 'PIROXICAM',    'ADCO-PIROXICAM 20MG', 28.90),
(3, 3, 'IBUPROFEN',    'PAINIL 200MG',        20.00),
(4, 4, 'ALLOPURINOL', 'ZYLOPRIM 100MG',      71.31),
(5, 4, 'PARACETAMOL', 'PANADO 120MG/5ML',    96.93),
(6, 5, 'DIAZEPAM',     'BETAPAM 5MG',         120.00);

INSERT INTO ProcedureItem(ProcedureItemID, TreatmentID, ProcedureName, ProcedureInfo, ProcedureCost)
VALUES
(1, 1, 'Blood transfusion', 'Blood donation',            0.00),
(2, 2, 'Circumcision',      '',                          12000.00),
(3, 3, 'Fetal monitoring',  'Montitoring after surgery', 1000.00),
(4, 4, 'Arthroplasty knee', '',                          1600.00),
(5, 4, 'Spinal fusion',     '',                          5000.00),
(6, 5, 'Hip replacement',   '',                          20000.00);

INSERT INTO Items(BillID, MedicineItemID, ProcedureItemID) VALUES
(1, 1, 1),
(1, 2, 2),
(2, 3, 4),
(3, 4, 3),
(4, 1, 1),
(4, 4, 4),
(5, 5, 5);
/*Comments - Here I have used DML statement Insert Into which allows me to enter data into health care
center database tables*/

SELECT * FROM PatientBills;
SELECT * FROM MedicineItem;
SELECT * FROM ProcedureItem;

SELECT * FROM Items; /*comments - here i am showing the newly inserted data into tables*/
```

Result Screenshot –





4. TESTING

I have performed the testing on my created health care database using different Advanced SQL skills. The tests I performed with code & result are shown below -

4.1. 1st Scenario – View

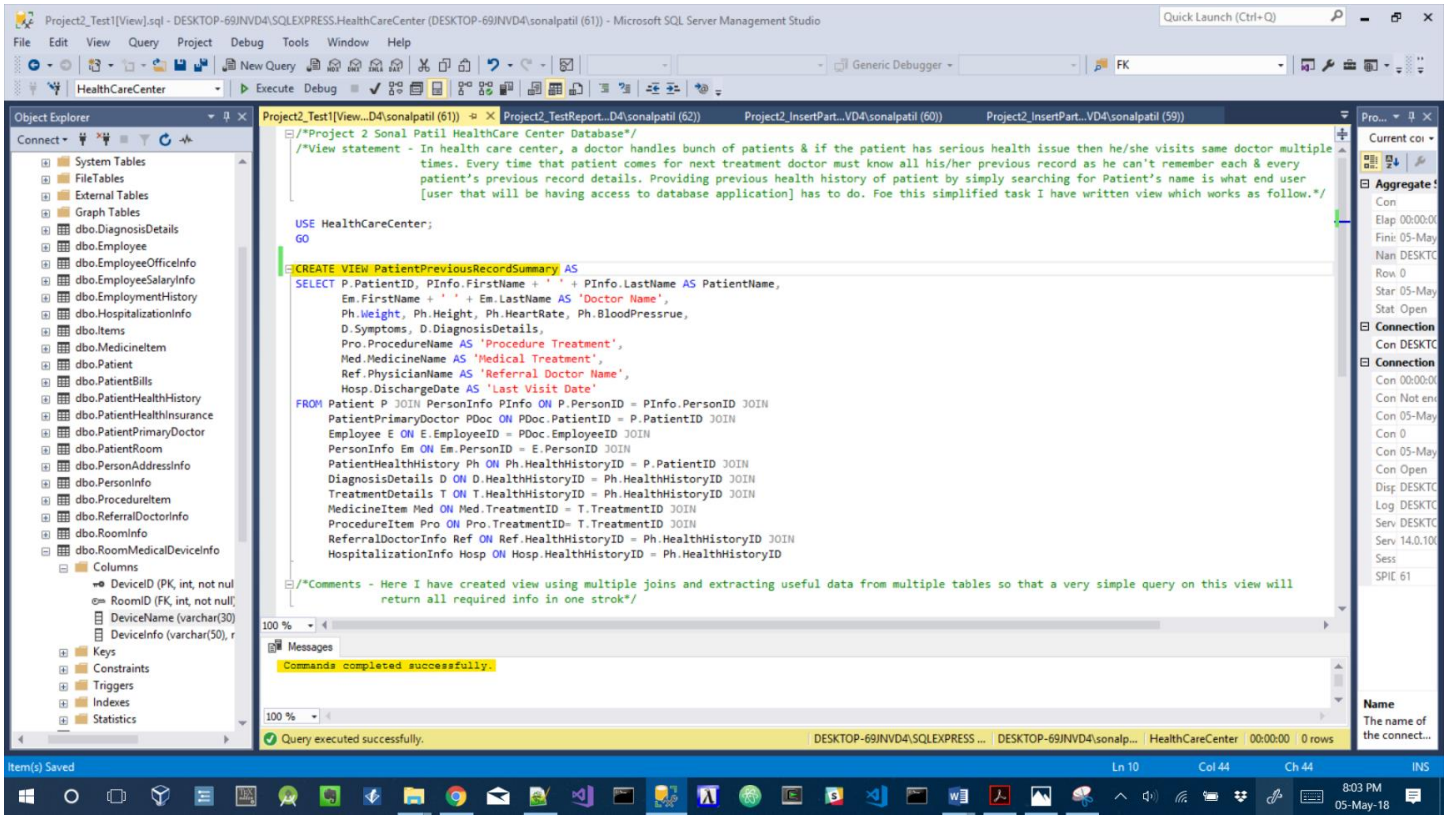
Scenario Statement –

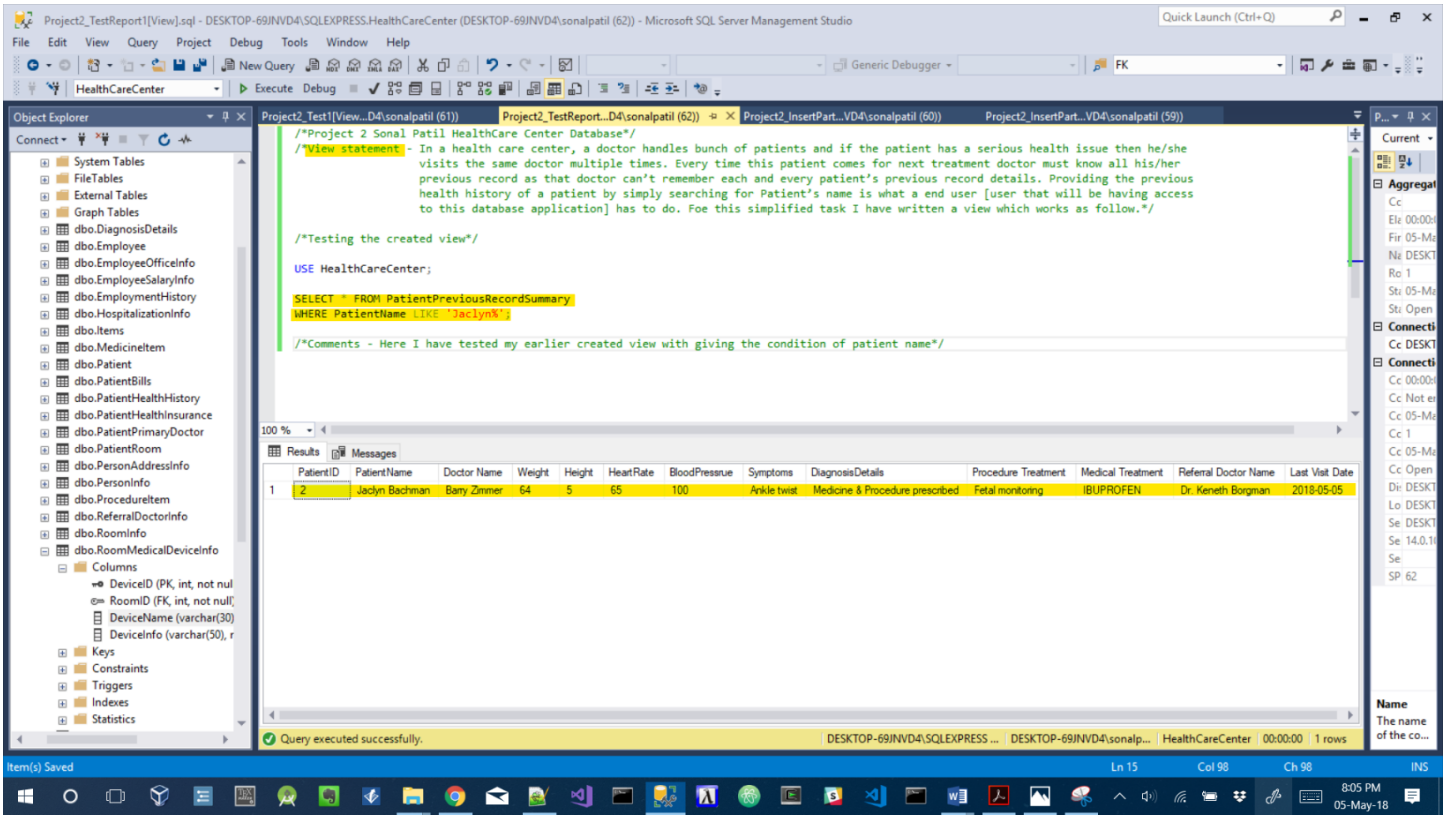
In a health care center, a doctor handles bunch of patients and if the patient has a serious health issue then he/she visits the same doctor multiple times. Every time this patient comes for the next treatment doctor must know all his/her previous record as that doctor can't remember each and every patient's previous record details. Providing the previous health history of a patient by simply searching for Patient's name is what a end user [user that will be having access to this database application] has to do. Foe this simplified task I have written a view which works as follow.

Code –

```
USE HealthCareCenter;
GO

CREATE VIEW PatientPreviousRecordSummary AS
SELECT P.PatientID, PInfo.FirstName + ' ' + PInfo.LastName AS PatientName,
          Em.FirstName + ' ' + Em.LastName AS 'Doctor Name',
          Ph.Weight, Ph.Height, Ph.HeartRate, Ph.BloodPressrue,
          D.Symptoms, D.DiagnosisDetails,
          Pro.ProcedureName AS 'Procedure Treatment',
          Med.MedicineName AS 'Medical Treatment',
          Ref.PhysicianName AS 'Referral Doctor Name',
          Hosp.DischargeDate AS 'Last Visit Date'
FROM Patient P JOIN PersonInfo PInfo ON P.PersonID = PInfo.PersonID JOIN
        PatientPrimaryDoctor PDoc ON PDoc.PatientID = P.PatientID JOIN
        Employee E ON E.EmployeeID = PDoc.EmployeeID JOIN
        PersonInfo Em ON Em.PersonID = E.PersonID JOIN
        PatientHealthHistory Ph ON Ph.HealthHistoryID = P.PatientID JOIN
        DiagnosisDetails D ON D.HealthHistoryID = Ph.HealthHistoryID JOIN
        TreatmentDetails T ON T.HealthHistoryID = Ph.HealthHistoryID JOIN
        MedicineItem Med ON Med.TreatmentID = T.TreatmentID JOIN
        ProcedureItem Pro ON Pro.TreatmentID= T.TreatmentID JOIN
        ReferralDoctorInfo Ref ON Ref.HealthHistoryID = Ph.HealthHistoryID JOIN
        HospitalizationInfo Hosp ON Hosp.HealthHistoryID = Ph.HealthHistoryID

/*Comments - Here I have created view using multiple joins and extracting useful data from multiple
tables so that a very simple query on this view will return all required info in one strok*/

/*Testing the created view*/

USE HealthCareCenter;

SELECT * FROM PatientPreviousRecordSummary
WHERE PatientName LIKE 'Jaclyn%';

/*Comments - Here I have tested my earlier created view with giving the condition of patient name*/
```

Result Screenshot –





4.2. 2nd Scenario – Stored Procedure

Scenario Statement –

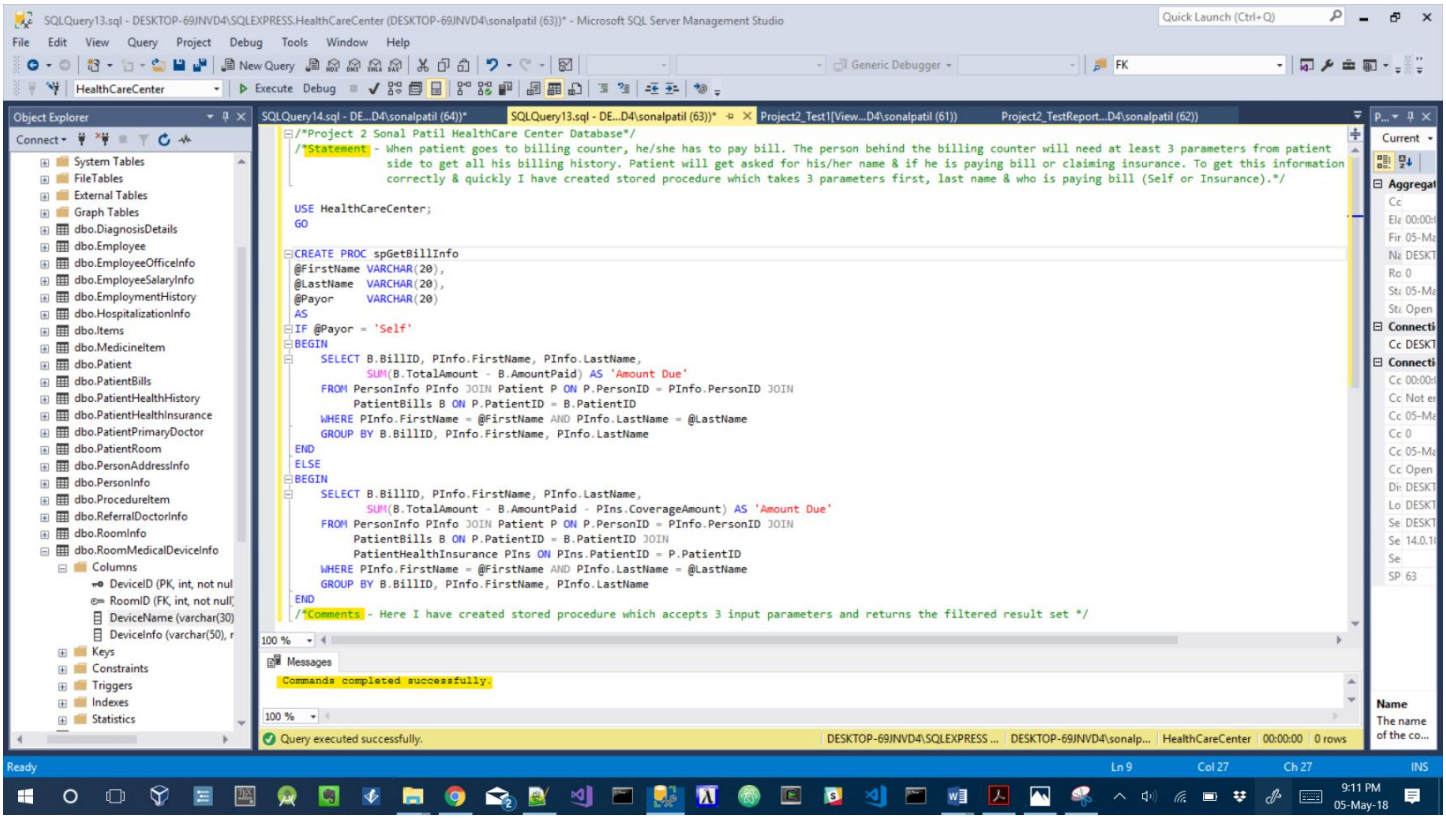
When a patient goes to billing counter of health care center, he/she has to pay the bill. The person behind the billing counter will need at least 3 parameters from patient side to get all his billing history. Patient will get asked for his/her name and if he is paying the bill or claiming the insurance.

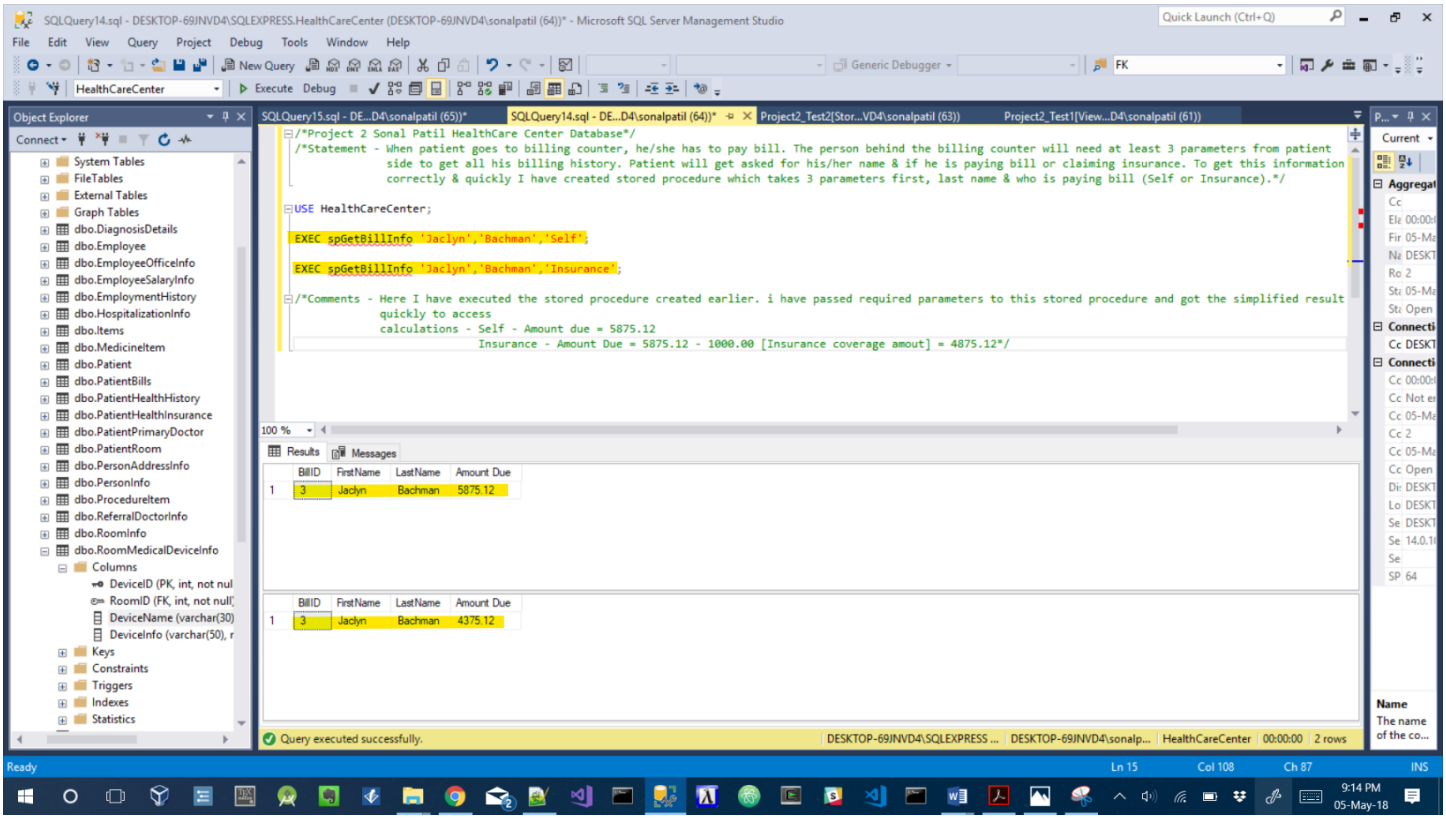
To get this information correctly and quickly I have created a stored procedure named spGetBillInfo which takes three parameters as input which are first name, last name and who is paying the bill (Self or Insurance). Depending on parameters provided I have displayed the results.

Code –

```
USE HealthCareCenter;
GO

CREATE PROC spGetBillInfo
@FirstName VARCHAR(20),
@LastName  VARCHAR(20),
@Payor     VARCHAR(20)
AS
IF @Payor = 'Self'
BEGIN
       SELECT B.BillID, PInfo.FirstName, PInfo.LastName,
                 SUM(B.TotalAmount - B.AmountPaid) AS 'Amount Due'
       FROM PersonInfo PInfo JOIN Patient P ON P.PersonID = PInfo.PersonID JOIN
              PatientBills B ON P.PatientID = B.PatientID
       WHERE PInfo.FirstName = @FirstName AND PInfo.LastName = @LastName
       GROUP BY B.BillID, PInfo.FirstName, PInfo.LastName
END
ELSE
BEGIN
       SELECT B.BillID, PInfo.FirstName, PInfo.LastName,
                 SUM(B.TotalAmount - B.AmountPaid - PIns.CoverageAmount) AS 'Amount Due'
       FROM PersonInfo PInfo JOIN Patient P ON P.PersonID = PInfo.PersonID JOIN
              PatientBills B ON P.PatientID = B.PatientID JOIN
              PatientHealthInsurance PIns ON PIns.PatientID = P.PatientID
       WHERE PInfo.FirstName = @FirstName AND PInfo.LastName = @LastName
       GROUP BY B.BillID, PInfo.FirstName, PInfo.LastName
END
/*Comments - Here I have created stored procedure which accepts 3 input parameters and returns the
filtered result set */

/*Testing the created stored procedure*/
USE HealthCareCenter;

EXEC spGetBillInfo 'Jaclyn','Bachman','Self';

EXEC spGetBillInfo 'Jaclyn','Bachman','Insurance';

/*Comments - Here I have executed the stored procedure created earlier. i have passed required
parameters to this stored procedure and got the simplified result quickly to access
calculations - Self - Amount due = 5875.12
               Insurance - Amount Due = 5875.12 - 1000.00 [Insurance coverage amount] = 4875.12*/
```

Result Screenshot –





4.3. 3rd Scenario – Function

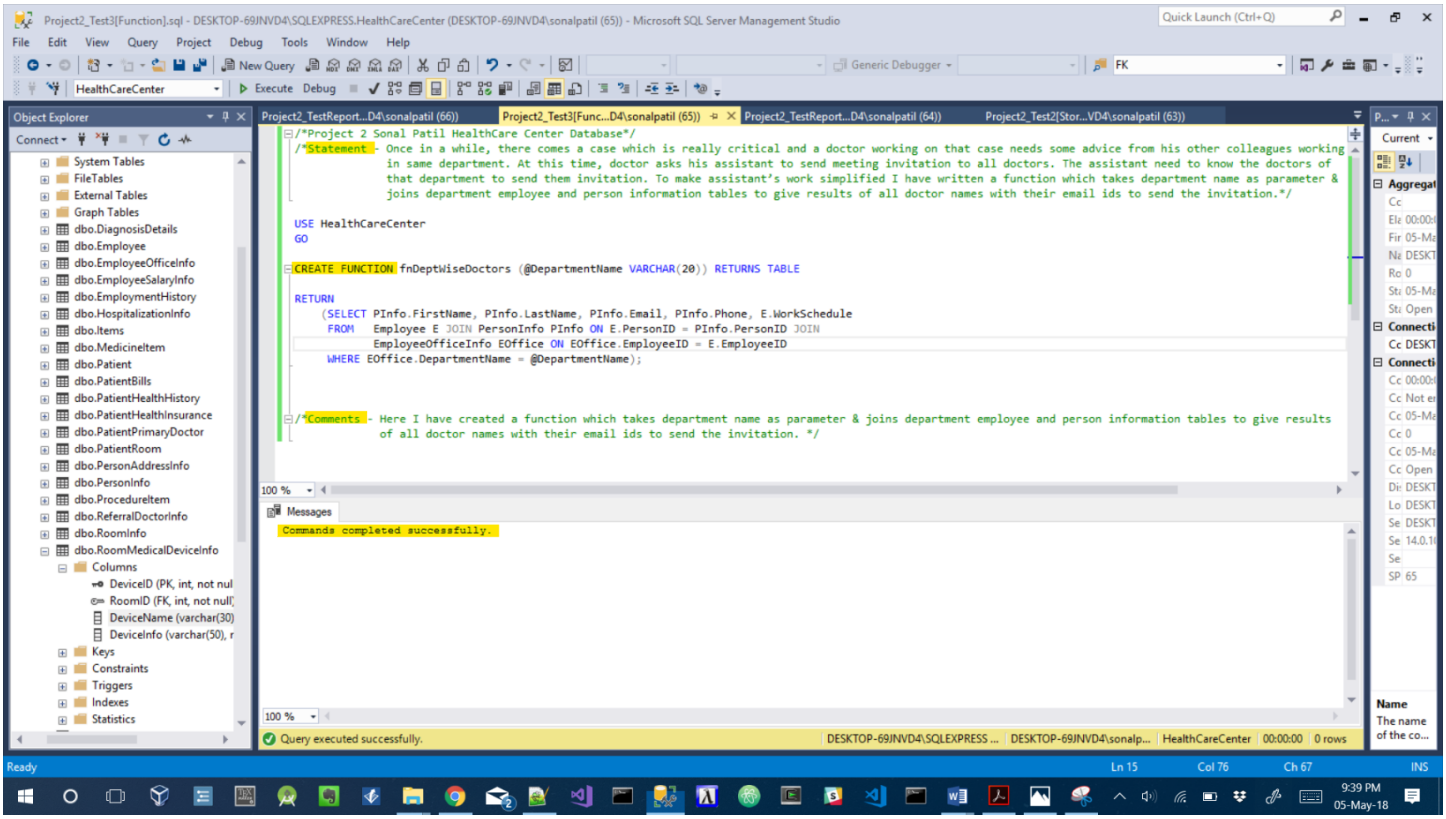
Scenario Statement –

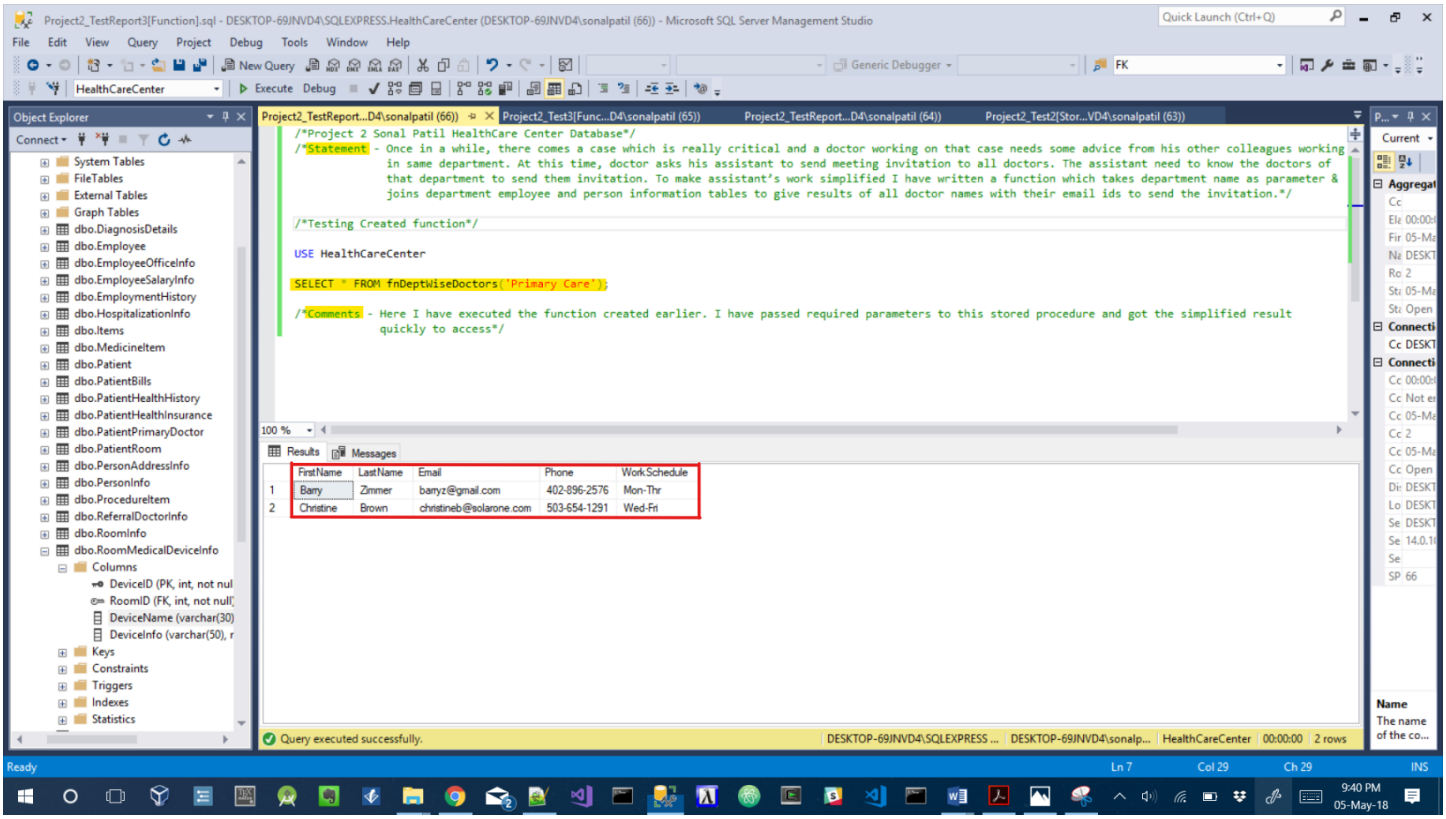
Once in a while, there comes a case which is really critical and a doctor working on that case needs some advice from his other colleagues working in the same department. At this time, doctor asks his assistant to send a meeting invitation to all the doctors. The assistant need to know the doctors of that department to send them invitation. To make assistant's work simplified I have written a function which takes department name as parameter and joins department employee and person information tables to give results of all doctor names with their email ids to send the invitation.

Code –

```
USE HealthCareCenter
GO

CREATE FUNCTION fnDeptWiseDoctors (@DepartmentName VARCHAR(20)) RETURNS TABLE

RETURN
       (SELECT PInfo.FirstName, PInfo.LastName, PInfo.Email, PInfo.Phone, E.WorkSchedule
        FROM   Employee E JOIN PersonInfo PInfo ON E.PersonID = PInfo.PersonID JOIN
                     EmployeeOfficeInfo EOffice ON EOffice.EmployeeID = E.EmployeeID
        WHERE EOffice.DepartmentName = @DepartmentName);



/*Comments - Here I have created a function which takes department name as parameter & joins department
employee and person information tables to give results of all doctor names with their email ids to send
the invitation. */

/*Testing Created function*/

USE HealthCareCenter

SELECT * FROM fnDeptWiseDoctors('Primary Care');

/*Comments - Here I have executed the function created earlier. I have passed required parameters to
this stored procedure and got the simplified result quickly to access*/
```

Result Screenshot –





4.4. 4th Scenario – Script

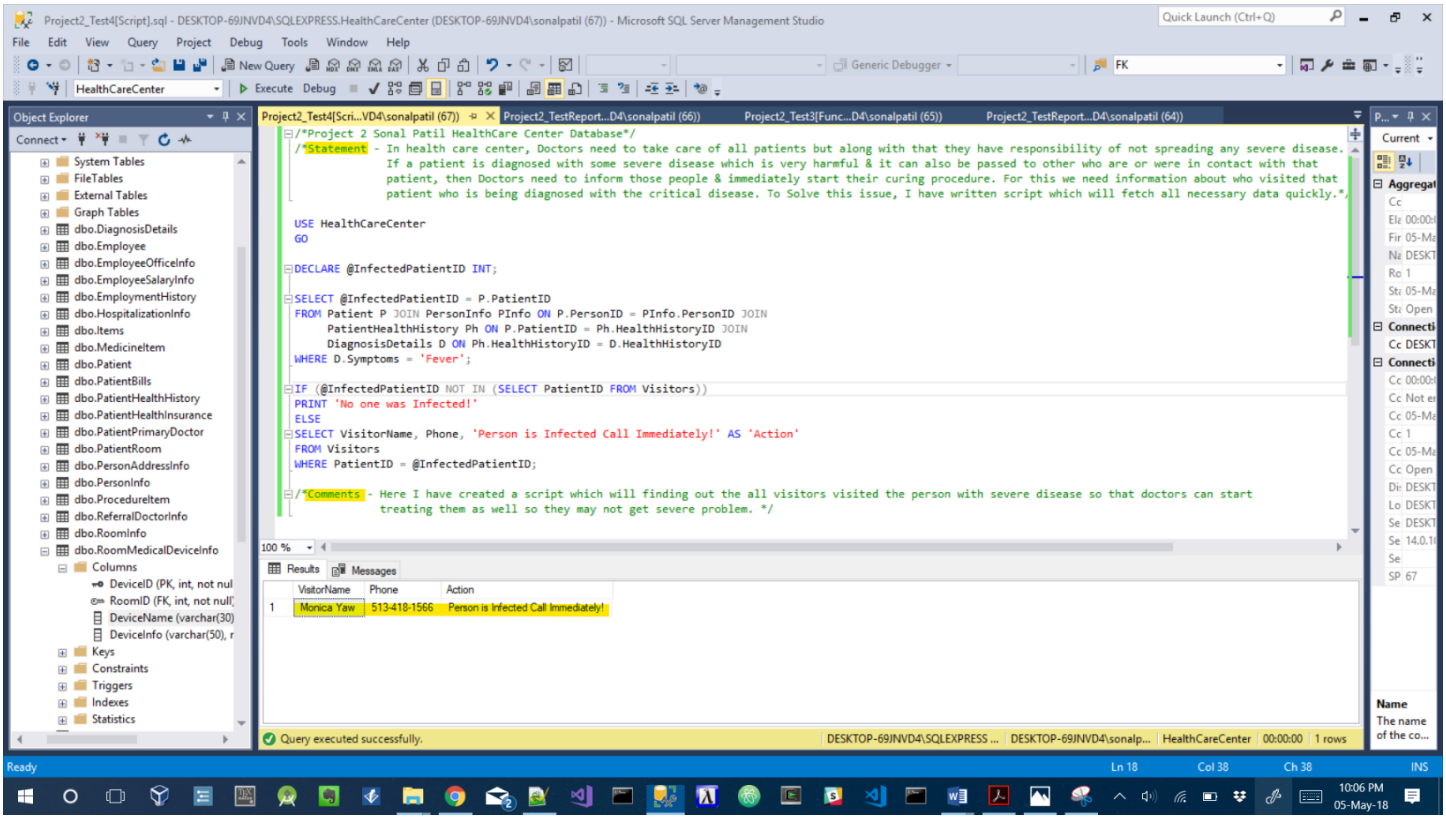
Scenario Statement –

In health care center, Doctors need to take care of all the patients but along with that they have responsibility of not spreading any severe disease. If a patient is diagnosed with some severe disease which is very harmful, and it can also be passed to other who are or were in contact with that patient, then Doctors need to inform those people and immediately start the curing procedure. For this we need the information about who visited that patient who is being diagnosed with the critical disease. To Solve this issue, I have written a script which will fetch all the necessary data quickly.

Code –

```
USE HealthCareCenter
GO
DECLARE @InfectedPatientID INT;
SELECT @InfectedPatientID = P.PatientID
FROM Patient P JOIN PersonInfo PInfo ON P.PersonID = PInfo.PersonID JOIN
     PatientHealthHistory Ph ON P.PatientID = Ph.HealthHistoryID JOIN
     DiagnosisDetails D ON Ph.HealthHistoryID = D.HealthHistoryID
WHERE D.Symptoms = 'Fever';

IF (@InfectedPatientID NOT IN (SELECT PatientID FROM Visitors)) PRINT 'No one was Infected!'
ELSE
SELECT VisitorName, Phone, 'Person is Infected Call Immediately!' AS 'Action'
FROM Visitors
WHERE PatientID = @InfectedPatientID;

/*Comments - Here I have created a script which will finding out the all visitors visited the person
with severe disease so that doctors can start treating them to avoid more severe problem. */
```

Result Screenshot –



5. CONCLUSION

Project Analysis –

The health care center database designed for this project meets almost all needs of a primary level Health care center. The advanced health care center may include more entities which are not included in this database, but the good point of my database design is that it can be expanded further as requirements increase.

Thinking critically at the start of database design always helps later when you start using the designed database. It removes all the duplication and result is a normalized data structure.

Another thing I learnt from this project is writing predefined routines [Functions/Stored Procedures/Views] help to retrieve the data faster and speed up the process.

Remarks –

In this project, I learned how to design a database from scratch when you have given the requirements or jobs one real world entity does. Here, how health care center works & what are the basic entities you should consider for this project was given as a part of requirement and then I designed the database matching those requirements.

It also helped me to understand how to transform real world entities data into database tables, how to relate them with each other using relationships available to us, how to split data into different tables which helps in triggering column specific queries. Also gave me knowledge of database design engine 'Vertabelo'.