The (N)erds

IS4420_Group Project

Group 3

Daimon Robertson

Lehi Ariceaga

Yunjia Hu

Ravi Ghuman

Spencer Bischoff

Table of Contents

Project Overview:

Our project is a database modeling the University of Utah Hospital. The general business model includes sick or injured patients receiving medical help from our medical staff. This staff includes nurses, doctors and surgeons, as well as, non-medical employees. Our company earns profit by employing hourly and salary workers who treat patients. The hospital provides the medical staff with machinery, technology, and tools to treat patients. These patients pay along with their insurance provider, which results in a profit. In order to run the hospital more efficiently and provide more high-quality services, the information and relationship of these entities will be recorded and tracked.

University of Utah Hospital is a research and teaching hospital on the campus of the University of Utah in Salt Lake City, Utah. As a hospital, our goal is to have the highest quality health care to offer in the state of Utah. Our most important services we offer include dentistry, sports medicine, cardiology and many more.

User Requirements

We specify user requirements primarily based on staff and patient needs. By tracking medical inventory, patient room usage and patient medical history, to make our hospitals' medical processes more efficient.

1. As employees track patient info
    a. As a medical, I want to know medical records
    b. As non-medical I'd like to know stock of medicines
    c. As a nurse, I'd like to know inpatient, outpatient
    d. As a front desk, I'd like to know whether someone is an in-patient or out-patient
    e. As a doctor, I'd like to know current medications
    f. As a surgeon, I'd like to know what room the surgery is in
    g. As a surgeon, I'd like to know which patient needs surgery
    h. As a front desk, i'd like to know which patients can be visited
    i. As front desk, i'd like to know estimated discharge date
    j. As an out-patient, i'd like to know my homecare instructions
    k. As an out-patient, i'd like to know billing
    l. As front desk, i'd like to know patient insurance information
    m. As front desk, i'd like to know billing - patient information
2. Manage employee information
    a. As accountant track salaries
    b. As a manager keep track of the firing, hiring, and promoting of employees
    c. As Non-medical. setting up when medical workers need to work
    d. As front desk, I"d like to know occupancy of rooms
    e. As corporate, I'd like to know utilization rate of rooms

f. As corporate, I'd like to know revenue
g. As manager, I'd like to know my hourly employees hours worked

Business Rules

We include 20 entities, and we use business rules to make the whole data model more specific and to understand the relationship between entities more clearly.

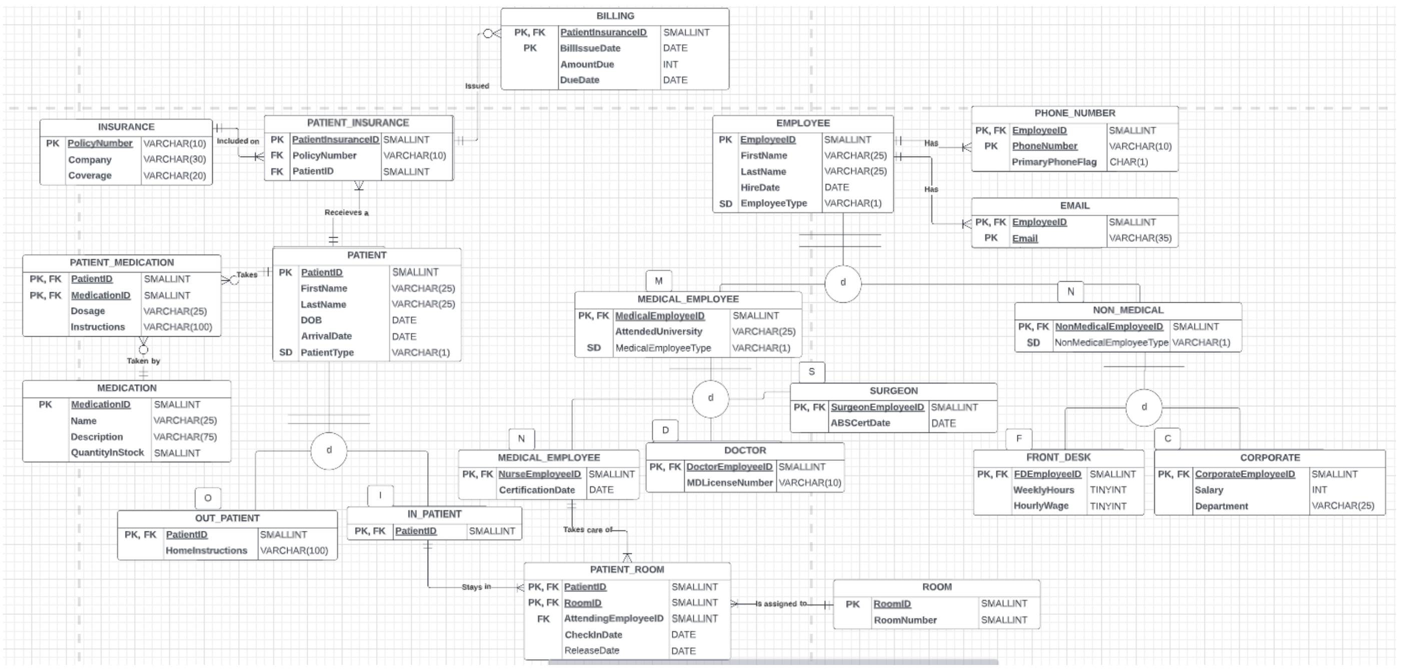
1. Insurance can only be tied to one patient and one patient can only have one insurance policy.
2. A bill can only go to one patient and one patient will only have one bill
3. Medications can be taken by many patients and many medications can be taken by each patient
4. Patient can have many phone numbers, only one phone number can be tied to a patient
5. Employees have to be either non-medical or medical employees, cannot be both
6. Non-medical employees are either manager, front desk, or corporate do not have to be one of those three
7. Medical employees are either doctor, nurse, or surgeon and do not have to be one of those three
8. Billing will have one insurance policy, each insurance goes to one billing
9. Patients have to be either in-patient or out-patient, cannot be both.
10. An employee provides at least one email but they can have more, one email only belongs to one employee.
11. An employee provides at least one phone number but they can have more, one phone number only belongs to one employee.
12. A patient provides at least one email but they can have more, one email only belongs to one patient.
13. Each medical employee has different types of certifications but they all attended a University
14. Front Desk employees have weekly hours and hourly wage, they are not salary
15. Corporate non-medical employees are part of a department and have a salary
16. Insurance Company, policy number and coverage is essential
17. In-patients will have a release date when they get released but out-patients are released the same day
18. The hire date of an employee will help with knowing how long they have been working at the hospital
19. All patient billing will have a due date
20. Home instructions are sent with out-patients only - not in-patients

Data Outputs / Business Questions

The business questions we implement are very relevant to hospitals. The utilization rate of the wards or the inventory of medical items are important measurement targets. The results of these questions allow us to better position ourselves and identify areas for improvement. And we divide these questions into the patient and employee categories, where we need to meet the most demand.

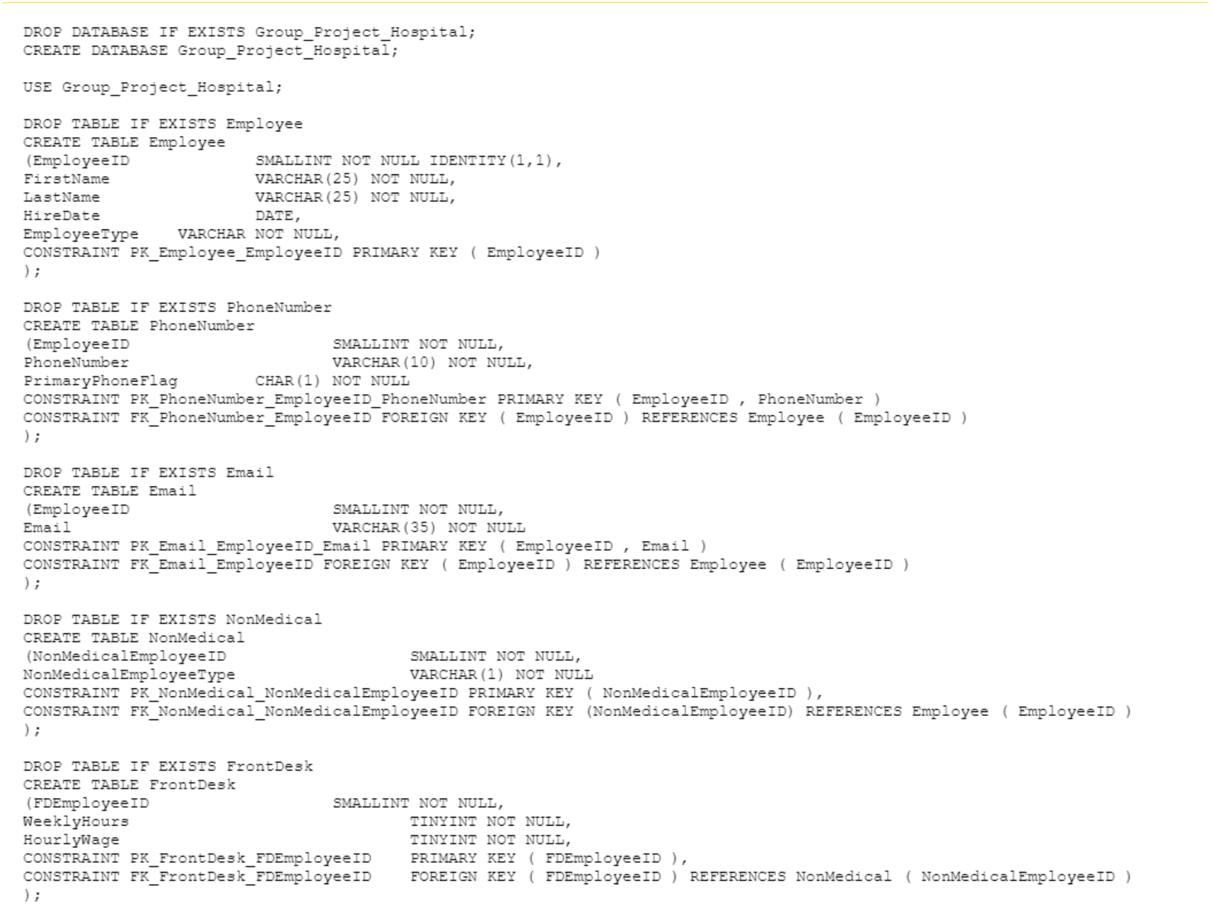
- ➔ 1. Patients:
  - ➔ Utilize billing, medication information
  - ➔ The total and average of annual bills
  - ➔ Measure the ratio of in-patient to out-patient to adjust the space of different departments
  - ➔ Find the age or other patient info for medication purposes
  - ➔ Find the correct instructions for medication for patients when they go home
  - ➔ Know the billing amount based on the insurance policy's coverage
  - ➔ Find patient by last name
  - ➔ Finding the room that a patient is in
  - ➔ Finding the contact info for a patient
- ➔ 2. Employees:
  - ➔ Utilize in patient out patient information
  - ➔ See Salary
  - ➔ Measure the room utilization
  - ➔ Managers consider promotions by finding information such as the employee's work hours.
  - ➔ Report to measure the length of time patients were treated
  - ➔ Report to find which insurance company patients used most frequently
  - ➔ Measure which medication wears off the fastest.
  - ➔ Measure utilization of doctors and nurses to see if hospitals need to fire or hire.
  - ➔ Finding the contact information of an employee

Physical Data Model (Entity-Relationship Diagram)



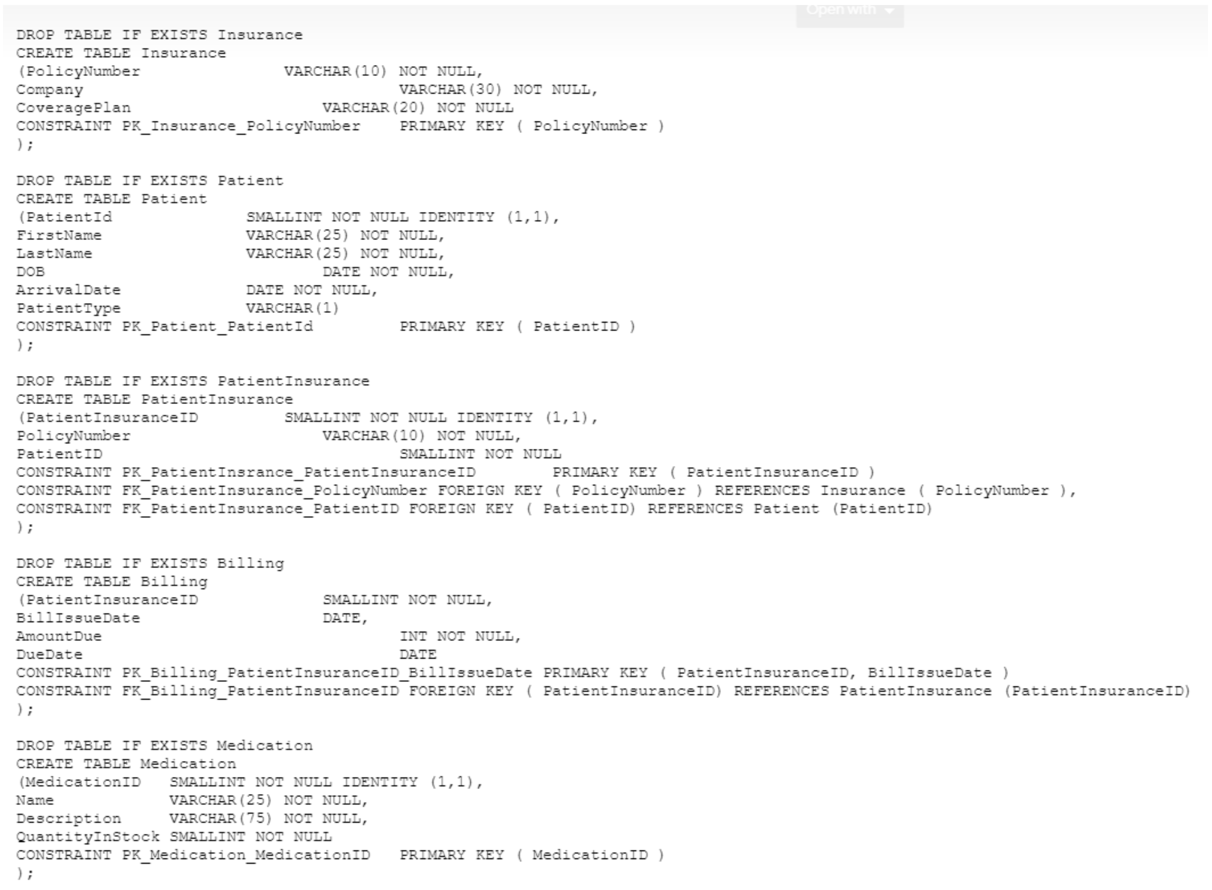
Database Implementation

Our group used the entity relationship diagram shown above to create the schema for Microsoft SQL Server. Please find the executable SQL file that will construct and populate the necessary tables in the database in the attached files. Here are a few lines of code that initialize and fill several database tables.



```
DROP DATABASE IF EXISTS Group_Project_Hospital;
CREATE DATABASE Group_Project_Hospital;

USE Group_Project_Hospital;

DROP TABLE IF EXISTS Employee
CREATE TABLE Employee
(EmployeeID             SMALLINT NOT NULL IDENTITY(1,1),
FirstName               VARCHAR(25) NOT NULL,
LastName                VARCHAR(25) NOT NULL,
HireDate                DATE,
EmployeeType    VARCHAR NOT NULL,
CONSTRAINT PK_Employee_EmployeeID PRIMARY KEY ( EmployeeID )
);

DROP TABLE IF EXISTS PhoneNumber
CREATE TABLE PhoneNumber
(EmployeeID                     SMALLINT NOT NULL,
PhoneNumber                     VARCHAR(10) NOT NULL,
PrimaryPhoneFlag        CHAR(1) NOT NULL
CONSTRAINT PK_PhoneNumber_EmployeeID_PhoneNumber PRIMARY KEY ( EmployeeID , PhoneNumber )
CONSTRAINT FK_PhoneNumber_EmployeeID FOREIGN KEY ( EmployeeID ) REFERENCES Employee ( EmployeeID )
);

DROP TABLE IF EXISTS Email
CREATE TABLE Email
(EmployeeID                     SMALLINT NOT NULL,
Email                           VARCHAR(35) NOT NULL
CONSTRAINT PK_Email_EmployeeID_Email PRIMARY KEY ( EmployeeID , Email )
CONSTRAINT FK_Email_EmployeeID FOREIGN KEY ( EmployeeID ) REFERENCES Employee ( EmployeeID )
);

DROP TABLE IF EXISTS NonMedical
CREATE TABLE NonMedical
(NonMedicalEmployeeID                   SMALLINT NOT NULL,
NonMedicalEmployeeType                  VARCHAR(1) NOT NULL
CONSTRAINT PK_NonMedical_NonMedicalEmployeeID PRIMARY KEY ( NonMedicalEmployeeID ),
CONSTRAINT FK_NonMedical_NonMedicalEmployeeID FOREIGN KEY (NonMedicalEmployeeID) REFERENCES Employee ( EmployeeID )
);

DROP TABLE IF EXISTS FrontDesk
CREATE TABLE FrontDesk
(FDEmployeeID                   SMALLINT NOT NULL,
WeeklyHours                             TINYINT NOT NULL,
HourlyWage                              TINYINT NOT NULL,
CONSTRAINT PK_FrontDesk_FDEmployeeID    PRIMARY KEY ( FDEmployeeID ),
CONSTRAINT FK_FrontDesk_FDEmployeeID    FOREIGN KEY ( FDEmployeeID ) REFERENCES NonMedical ( NonMedicalEmployeeID )
);
```

```
DROP TABLE IF EXISTS Corporate
CREATE TABLE Corporate
(CorporateEmployeeID    SMALLINT NOT NULL,
Salary                                  INT NOT NULL,
Department                              VARCHAR(25),
CONSTRAINT PK_Corporate_CorporateEmployeeID  PRIMARY KEY ( CorporateEmployeeID ),
CONSTRAINT FK_Corporate_CorporateEmployeeID  FOREIGN KEY ( CorporateEmployeeID ) REFERENCES NonMedical ( NonMedicalEmployeeID )
);


DROP TABLE IF EXISTS MedicalEmployee
CREATE TABLE MedicalEmployee
(MedicalEmployeeID      SMALLINT NOT NULL,
AttendedUniversity      VARCHAR(25) NOT NULL,
MedicalEmployeeType VARCHAR(1) NOT NULL
CONSTRAINT PK_MedicalEmployee_MedicalEmployeeID PRIMARY KEY ( MedicalEmployeeID )
CONSTRAINT FK_MedicalEmployee_MedicalEmployeeID FOREIGN KEY ( MedicalEmployeeID ) REFERENCES Employee ( EmployeeID )
);

DROP TABLE IF EXISTS RegisteredNurse
CREATE TABLE RegisteredNurse
(NurseEmployeeID        SMALLINT NOT NULL,
CertificationDate       DATE
CONSTRAINT PK_RegisteredNurse_NurseEmployeeID   PRIMARY KEY ( NurseEmployeeID )
CONSTRAINT FK_RegisteredNurse_NurseEmployeeID   FOREIGN KEY (NurseEmployeeID) REFERENCES MedicalEmployee (MedicalEmployeeID)
);

DROP TABLE IF EXISTS Doctor
CREATE TABLE Doctor
(DoctorEmployeeID       SMALLINT NOT NULL,
MDLicenseNumber         VARCHAR(10)
CONSTRAINT PK_Doctor_DoctorEmployeeID   PRIMARY KEY  (DoctorEmployeeID)
CONSTRAINT FK_Doctor_DoctorEmployeeID   FOREIGN KEY (DoctorEmployeeID) REFERENCES MedicalEmployee (MedicalEmployeeID)
);

DROP TABLE IF EXISTS Surgeon
CREATE TABLE Surgeon
(SurgeonEmployeeID      SMALLINT NOT NULL,
ABSCertDate                     DATE
CONSTRAINT PK_Surgeon_SurgeonEmployeeID PRIMARY KEY ( SurgeonEmployeeID )
CONSTRAINT FK_Surgeon_SurgeonEmployeeID FOREIGN KEY (SurgeonEmployeeID) REFERENCES MedicalEmployee (MedicalEmployeeID)
);
```

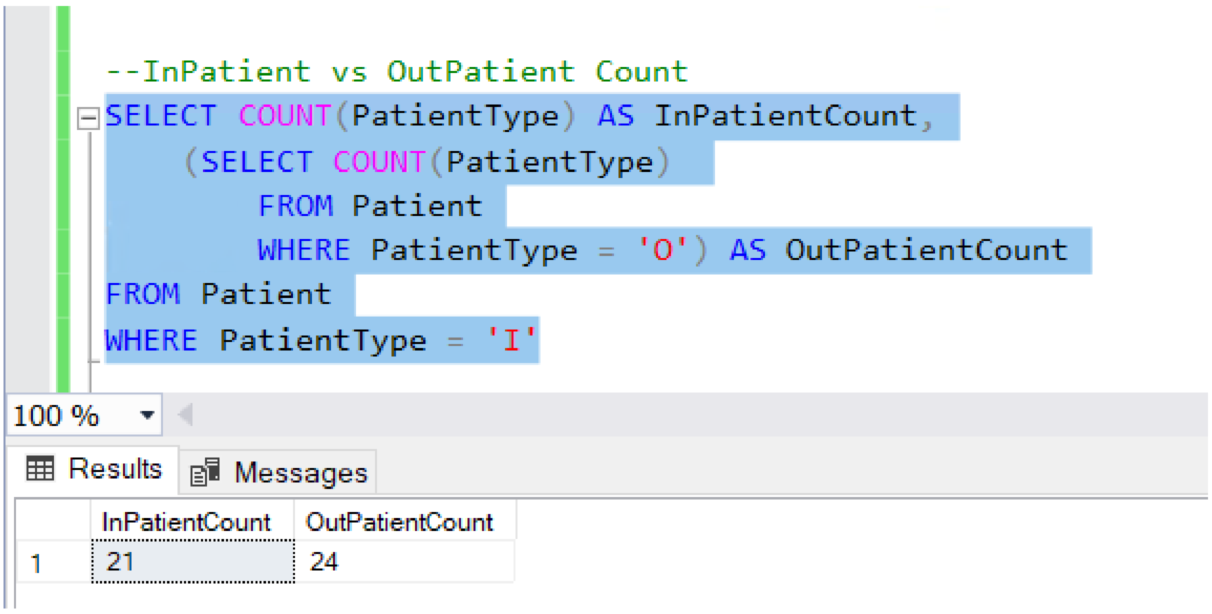


```
DROP TABLE IF EXISTS Insurance
CREATE TABLE Insurance
(PolicyNumber               VARCHAR(10) NOT NULL,
Company                                 VARCHAR(30) NOT NULL,
CoveragePlan                    VARCHAR(20) NOT NULL
CONSTRAINT PK_Insurance_PolicyNumber    PRIMARY KEY ( PolicyNumber )
);

DROP TABLE IF EXISTS Patient
CREATE TABLE Patient
(PatientId              SMALLINT NOT NULL IDENTITY (1,1),
FirstName               VARCHAR(25) NOT NULL,
LastName                VARCHAR(25) NOT NULL,
DOB                             DATE NOT NULL,
ArrivalDate             DATE NOT NULL,
PatientType             VARCHAR(1)
CONSTRAINT PK_Patient_PatientId         PRIMARY KEY ( PatientID )
);

DROP TABLE IF EXISTS PatientInsurance
CREATE TABLE PatientInsurance
(PatientInsuranceID         SMALLINT NOT NULL IDENTITY (1,1),
PolicyNumber                    VARCHAR(10) NOT NULL,
PatientID                               SMALLINT NOT NULL
CONSTRAINT PK_PatientInsrance_PatientInsuranceID        PRIMARY KEY ( PatientInsuranceID )
CONSTRAINT FK_PatientInsurance_PolicyNumber FOREIGN KEY ( PolicyNumber ) REFERENCES Insurance ( PolicyNumber ),
CONSTRAINT FK_PatientInsurance_PatientID FOREIGN KEY ( PatientID) REFERENCES Patient (PatientID)
);

DROP TABLE IF EXISTS Billing
CREATE TABLE Billing
(PatientInsuranceID             SMALLINT NOT NULL,
BillIssueDate                   DATE,
AmountDue                               INT NOT NULL,
DueDate                                 DATE
CONSTRAINT PK_Billing_PatientInsuranceID_BillIssueDate PRIMARY KEY ( PatientInsuranceID, BillIssueDate )
CONSTRAINT FK_Billing_PatientInsuranceID FOREIGN KEY ( PatientInsuranceID) REFERENCES PatientInsurance (PatientInsuranceID)
);

DROP TABLE IF EXISTS Medication
CREATE TABLE Medication
(MedicationID   SMALLINT NOT NULL IDENTITY (1,1),
Name            VARCHAR(25) NOT NULL,
Description     VARCHAR(75) NOT NULL,
QuantityInStock SMALLINT NOT NULL
CONSTRAINT PK_Medication_MedicationID   PRIMARY KEY ( MedicationID )
);
```

```
DROP TABLE IF EXISTS PatientMedication
CREATE TABLE PatientMedication
(PatientID              SMALLINT NOT NULL,
MedicationID    SMALLINT NOT NULL,
Dosage                  VARCHAR(25) NOT NULL,
Instructions    VARCHAR(100) NOT NULL
CONSTRAINT PK_PatientMedication_PatientID_MedicationID PRIMARY KEY ( PatientID , MedicationID )
CONSTRAINT FK_PatientMedication_PatientID FOREIGN KEY (PatientID) REFERENCES Patient (PatientID),
CONSTRAINT FK_PatientMedication_MedicationID FOREIGN KEY (MedicationID) REFERENCES Medication (MedicationID)
);

DROP TABLE IF EXISTS OutPatient
CREATE TABLE OutPatient
(PatientId               SMALLINT NOT NULL,
HomeInstructions VARCHAR(100) NOT NULL
CONSTRAINT PK_OutPatient_PatientID      PRIMARY KEY ( PatientID )
CONSTRAINT FK_OutPatient_PatientID FOREIGN KEY (PatientID) REFERENCES Patient (PatientID)
);

DROP TABLE IF EXISTS InPatient
CREATE TABLE InPatient
(PatientID              SMALLINT NOT NULL
CONSTRAINT PK_InPatient_PatientID       PRIMARY KEY ( PatientID )
CONSTRAINT FK_InPatient_PatientID FOREIGN KEY (PatientID) REFERENCES Patient (PatientID)
);

DROP TABLE IF EXISTS Room
CREATE TABLE Room
(RoomID         SMALLINT NOT NULL IDENTITY(1,1),
RoomNumber      SMALLINT NOT NULL
CONSTRAINT PK_Room_RoomID       PRIMARY KEY ( RoomID )
);

DROP TABLE IF EXISTS PatientRoom
CREATE TABLE PatientRoom
(PatientId                      SMALLINT NOT NULL,
RoomID                          SMALLINT NOT NULL,
AttendingEmployeeID     SMALLINT NOT NULL,
CheckInDate                     DATE,
ReleaseDate                     DATE
CONSTRAINT PK_PatientRoom_PatientID_RoomID PRIMARY KEY ( PatientID, RoomID )
CONSTRAINT FK_PatientRoom_RoomID FOREIGN KEY (RoomID) REFERENCES Room (RoomID),
CONSTRAINT FK_PatientRoom_PatientID FOREIGN KEY (PatientID) REFERENCES Patient (PatientID),
CONSTRAINT FK_AttendingEmployeeID FOREIGN KEY (AttendingEmployeeID) REFERENCES Employee (EmployeeID)
);
```

Answering Business Questions

Our query designers constructed queries, views, and stored procedures using the business questions and the necessary data outputs to produce the reports. For the collection of SQL queries, views, and stored procedures, please refer to the attached files. A handful of the most intriguing inquiries' code snippets and results are provided here.

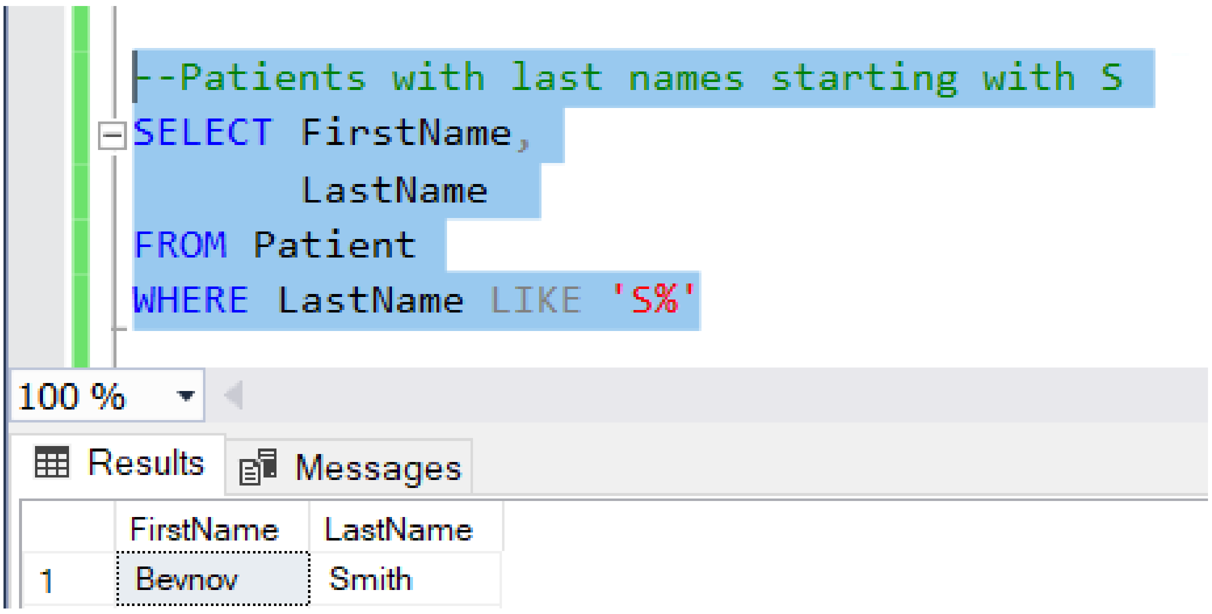


```
--InPatient vs OutPatient Count
SELECT COUNT(PatientType) AS InPatientCount,
    (SELECT COUNT(PatientType)
        FROM Patient
        WHERE PatientType = 'O') AS OutPatientCount
FROM Patient
WHERE PatientType = 'I'
```

100 %

Results | Messages

```
   InPatientCount  OutPatientCount
1  21              24
```

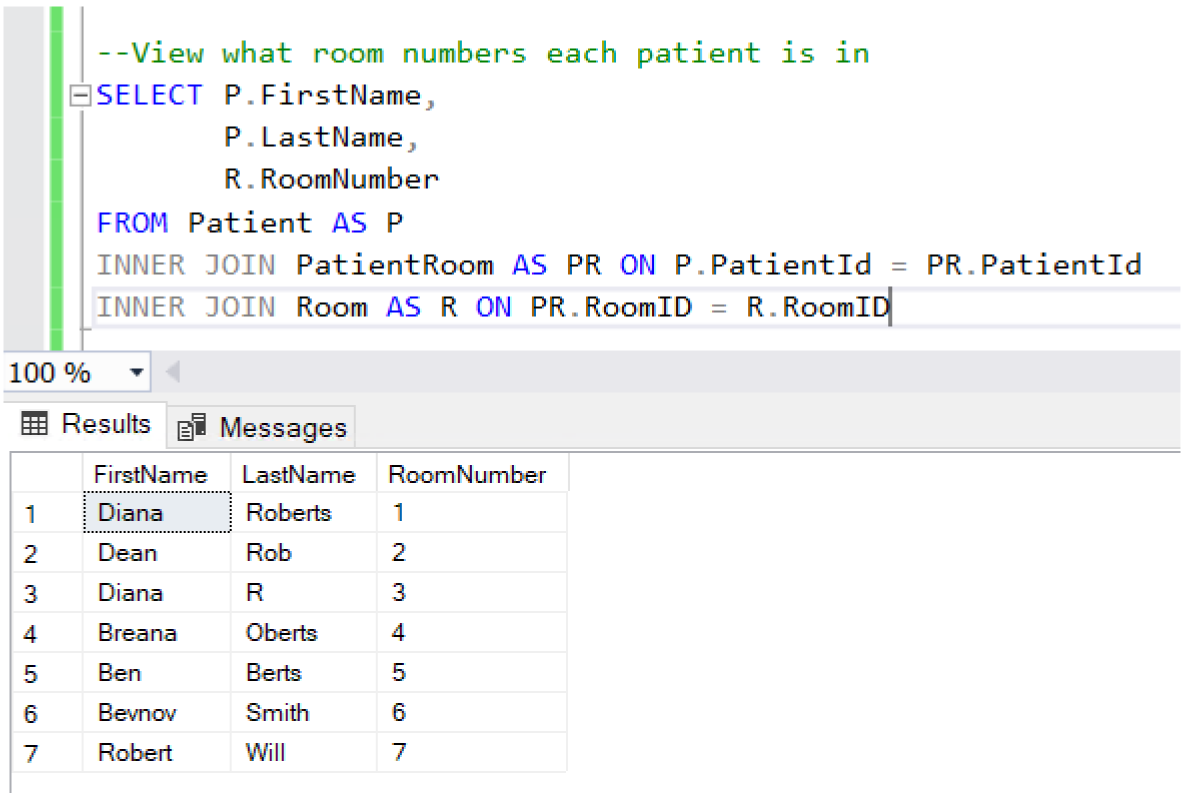


```
--Patients with last names starting with S
SELECT FirstName,
         LastName
FROM Patient
WHERE LastName LIKE 'S%'
```

100 %

Results | Messages

```
   FirstName  LastName
1  Bevnov     Smith
```

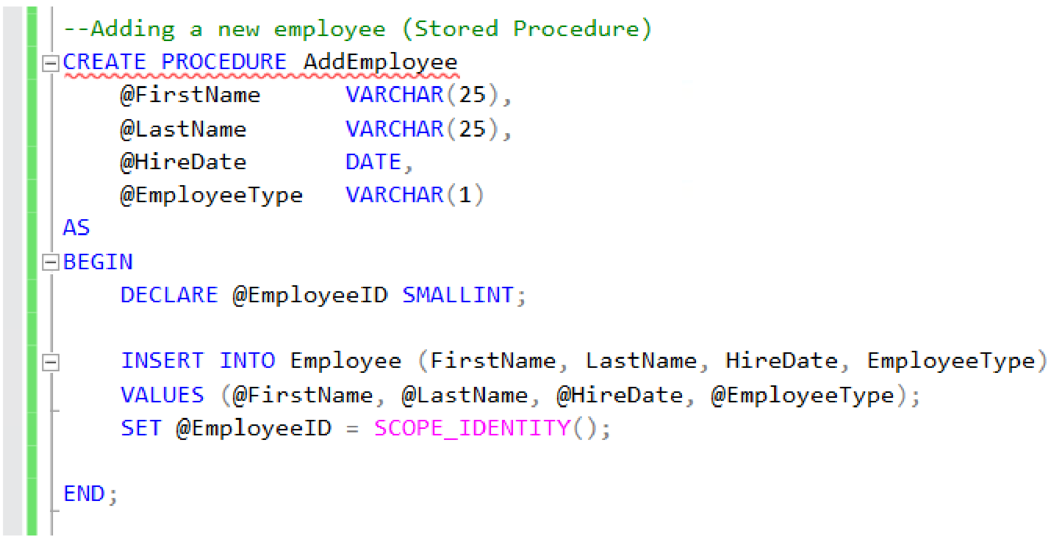


```sql
--View what room numbers each patient is in
SELECT P.FirstName,
       P.LastName,
       R.RoomNumber
FROM Patient AS P
INNER JOIN PatientRoom AS PR ON P.PatientId = PR.PatientId
INNER JOIN Room AS R ON PR.RoomID = R.RoomID
```

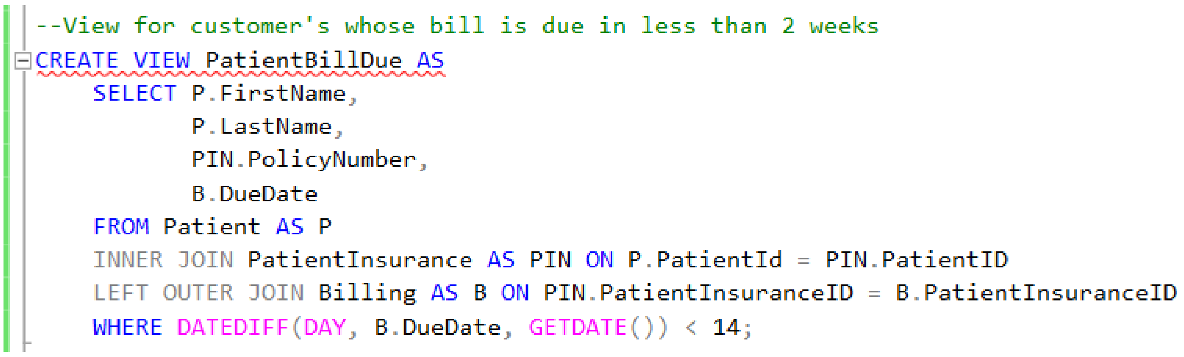
100 %

Results | Messages

	FirstName	LastName	RoomNumber
1	Diana	Roberts	1
2	Dean	Rob	2
3	Diana	R	3
4	Breana	Oberts	4
5	Ben	Berts	5
6	Bevnov	Smith	6
7	Robert	Will	7



```sql
--Adding a new employee (Stored Procedure)
CREATE PROCEDURE AddEmployee
    @FirstName      VARCHAR(25),
    @LastName       VARCHAR(25),
    @HireDate       DATE,
    @EmployeeType   VARCHAR(1)
AS
BEGIN
    DECLARE @EmployeeID SMALLINT;

    INSERT INTO Employee (FirstName, LastName, HireDate, EmployeeType)
    VALUES (@FirstName, @LastName, @HireDate, @EmployeeType);
    SET @EmployeeID = SCOPE_IDENTITY();

END;
```



```sql
--View for customer's whose bill is due in less than 2 weeks
CREATE VIEW PatientBillDue AS
    SELECT P.FirstName,
           P.LastName,
           PIN.PolicyNumber,
           B.DueDate
    FROM Patient AS P
    INNER JOIN PatientInsurance AS PIN ON P.PatientId = PIN.PatientID
    LEFT OUTER JOIN Billing AS B ON PIN.PatientInsuranceID = B.PatientInsuranceID
    WHERE DATEDIFF(DAY, B.DueDate, GETDATE()) < 14;
```

Summary and Conclusion

We chose the University hospital because some of our group members worked there and thought it would be interesting to think about the processes that the hospital goes through. Challenges we came across were how large hospital databases can actually get. We, as a group, had to come to a compromise in how in depth we all wanted to go without going into a rabbit hole. At the same time, giving adequate enough information and proper layout of a hospital's system.

We converted that data into a database schema in the form of entity relationship diagrams to determine what kind of data outputs we would need to create (ERDs). Starting with a logical ERD, we further improved it to create a physical ERD. Following completion of that, we used the ERD to create a database in Microsoft SQL Server. All 20 business inquiries and data outputs required of us are effectively addressed, or provided with reports.