CS557 Project Fall 2015

Hospital Management System

Shruthi Vinjamuri

Contents

Hospital Management System

1. Purpose of the project

The purpose of the project is to design a Hospital Management System for Apollo Hospital which is a multi-facility hospital in the New York City. The hospital has multiple departments with different classes of doctors. The patients are to be routed to the concerned department to which doctor belongs and the treatment details should be tracked. The project should also handle room allocation to the patients, billing and other paper works to be taken care of. In particular, the database built for the project should maintain records of the outdoor patients, patients admitted in the hospital, the medical records of the patients by the doctors, the patients who have been operated and the patients who are already released from the hospital.

2. Software Requirements

- Programming Language used: SQL
- Database Tool: Microsoft SQL Server 2014
- RAM 8 GB

3. Project Description

The Apollo Hospital is a multi-facility hospital in the New York City.

The hospital has multiple departments and units such as Pathology, Emergency, Dental, Obstetrics and Gynecology, Cardiology, Gastroenterology, ICUs (Intensive Care Unit), Ear nose and throat (ENT), Orthopedic, Neurology, Cardiology, Diagnostic imaging and so on. There is a Patient Welcome Unit where the visiting patients can obtain a card (that is, an entry card of that patient) for check up from a concerned doctor. After making an entry in the card, they visit the concerned doctor in their room. Depending on the situation, the doctor can either prescribe medicine or admit the patients in the respective departments. The patient may choose either private or general room according to their needs. But before getting an admission in the hospital, the patient has to fulfil certain formalities for instance clearing certain bills for: room charges, meal charges and others. After completion of the treatment, the doctor releases the patient. Prior to the release, the patient has to complete certain formalities again, such as full payments of any unpaid bills, insurance paper works and some other issues (if any).

There are primarily two classes of doctors in the hospital, viz., regular doctors and call on doctors. Regular doctors are those doctors who come to the hospital daily. Call-on doctors are those doctors who are requested by the hospital to attend some patients if the concerned doctor is unavailable.

The aim of this mini-project is to design and develop a database for the aforementioned hospital to maintain the records of various departments, rooms, and doctors in the hospital. The database must also maintain records of the outdoor patients, patients admitted in the hospital, the medical records of the patients by the doctors, the patients who have been operated and the patients who are already released from the hospital.

4. Database Design

In the project statement the following design is asked to be implemented –

Patient: It keeps track of all the details about both the admitted patients and the outdoor patients. A unique ID is generated for each patient after registration. Patient ID, patient's name, patient's address, admission date, doctor's name, treatment details, room number, room type etc. are to be recorded. Also particular patient details can be viewed in the table using a separate form with an attribute patient id, doctor's name and admission date.

Admission: This records the basic patient related information, which is registered when the patient visits the hospital for the first time. Each patient is allocated with a unique patient identification number. It should also record details of all the formalities to be fulfilled by the patient.

Outdoor–Patient: This manages activities related to a patient who visits the Hospital Resident Doctor or Call-on Doctor for medical consultations, diagnosis and treatment.

Staff: It keeps track of all the details about doctors and other staff members of the hospital. Staff member's name, designation, staff ID, address, qualification, cell no, e-mail are recorded.

Doctor: An entity for the doctors is also required to record the details such as patients attended by a doctor, tests conducted, doctor ID, department and so on.

Billing: This keeps track of the bills of both the admitted patients as well as outdoor patients who come to the hospital. Prior to the release from the hospital, every patient needs to complete certain formalities of the hospital such as payment of bills (if any), test charges, operation charges (if any), doctors' charges, etc. These charges are required to be recorded in the billing details.

Department: This keeps track of department details for each patient. Patient id, department name, doctor and other related attributed are recorded here.

4.1 BCNF and Other Normal Forms verification

I have made all the domains of the attributes in the design 'atomic' to preserve First Normal Form.

- ➢ Given an instance of r (R), we say that the instance satisfies the functional dependency $\alpha \rightarrow \beta$ if for all pairs of tuples t1 and t2 in the instance such that
  $t1[\alpha] = t2[\alpha]$, it is also the case that $t1[\beta] = t2[\beta]$.

We say that the functional dependency α → β holds on schema r (R) if, in every legal instance of r (R) it satisfies the functional dependency.

As per the design given in the requirements I found the Patient relation is not in BCNF.
The functional dependencies involved in the patient relation are –

1. Patient ID -> patient's name, patient's address
2. room number -> room type

One of the more desirable normal forms that we can obtain is Boyce–Codd normal form (BCNF). It eliminates all redundancy that can be discovered based on functional dependencies. A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F+ are of the form α→β, where α ⊆ R, β ⊆ R, and at least one of the following holds:

- α→β is a trivial functional dependency (that is, β ⊆ α).
- α is a super key for schema R.

A database design is in BCNF if each member of the set of relation schemas that constitutes the design is in BCNF.

In the project requirements it is said that a unique id for each patient registered is to be maintained. Hence I am maintaining the 'Encounters' of the patient to the hospital. Here there should be proper decomposition of relations to maintain consistency of database and handle redundancy.

As per the requirement it is asked whenever a patient encounters the hospital – the respective doctor details, treatment details and room details for in-patient are to be tracked under a patient table. If all these details are tracked in the same table then say a patient is just an outdoor patient then all the in-patient columns for that patient and the room details would be 'NULLS' which is undesirable. Similarly if the patient is in-patient, out-patient columns would be 'NULLS'.

Also if the room details are included in the patient table and assume if a particular room type is to be deleted say, room_type ='General' then all the patient records with that room type would be deleted which leads to '**deletion anomaly**'. Hence this is very much undesirable to combine all these details into a single table and proper loss less decomposition preserving is required to overcome functional dependencies, update & deletion anomalies and data redundancy.

Hence to preserve the BCNF in my database design I have split the patient requirement into tables:

PATIENT_ENCOUNTER Table -

NAME	TYPE	CONSTRAINTS
ENCOUNTER_ID	BIGINT	PRIMARY KEY
DOCTOR_ID	BIGINT	PRIMARY KEY
PATIENT_ID	BIGINT	FOREIGN KEY REFERENCES ADMISSION
DEPARTMENT_ID	VARCHAR(10)	FOREIGN KEY REFERENCES DEPARTMENT_HOSP
ENCOUNTER_DATE	DATE	NOT NULL

PATIENT_TYPE	VARCHAR(10)	NOT NULL, CHECK(PATIENT_TYPE IN(‘OUTDOOR’,’INDOOR’))

ROOM Table –

NAME	TYPE	CONSTRAINTS
ROOM_NUMBER	VARCHAR(10)	PRIMARY KEY
ROOM_TYPE	VARCHAR(7)	NOT NULL, CHECK(ROOM_TYPE IN('GENERAL','PRIVATE'))
ROOM_STATUS	CHAR(1)	NOT NULL, CHECK(ROOM_STATUS IN('Y','N','y','n'))

INDOOR_PATIENT Table -

NAME	TYPE	CONSTRAINTS
ENCOUNTER_ID	BIGINT	PRIMARY KEY, FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
DOCTOR_ID	BIGINT	PRIMARY KEY, FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
ADMISSION_DATE	BIGINT	NOT NULL
RELEASE_DATE	BIGINT	
TREATMENT_DETAILS	VARCHAR(10)	
ROOM_NUMBER	DATE	NOT NULL

With the above design of three tables the patient’s encounter with doctor, the type of the patient and room allocated to the patient are tracked effectively. All the tables are now in BCNF.

I have assumed that whenever a patient is seen by a different doctor in a visit it would have new line in the PATIENT_ENCOUNTER table. Also, during a patient visit it will be categorized either as INDOOR or OUTDOOR patient type.
Keeping in mind to preserve BCNF form for the database tables, my design includes following tables and attributes:

PATIENT_ENCOUNTER Table:

NAME	TYPE	CONSTRAINTS
ENCOUNTER_ID	BIGINT	PRIMARY KEY
DOCTOR_ID	BIGINT	PRIMARY KEY
PATIENT_ID	BIGINT	FOREIGN KEY REFERENCES ADMISSION
DEPARTMENT_ID	VARCHAR(10)	FOREIGN KEY REFERENCES DEPARTMENT_HOSP
ENCOUNTER_DATE	DATE	NOT NULL
PATIENT_TYPE	VARCHAR(10)	NOT NULL, CHECK(PATIENT_TYPE IN(‘OUTDOOR’,’INDOOR’))

ROOM Table:

NAME	TYPE	CONSTRAINTS
ROOM_NUMBER	VARCHAR(10)	PRIMARY KEY
ROOM_TYPE	VARCHAR(7)	NOT NULL, CHECK(ROOM_TYPE IN('GENERAL','PRIVATE'))
ROOM_STATUS	CHAR(1)	NOT NULL, CHECK(ROOM_STATUS IN('Y','N','y','n'))

INDOOR_PATIENT Table:

NAME	TYPE	CONSTRAINTS
ENCOUNTER_ID	BIGINT	PRIMARY KEY, FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
DOCTOR_ID	BIGINT	PRIMARY KEY,FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
ADMISSION_DATE	BIGINT	NOT NULL
RELEASE_DATE	BIGINT	
TREATMENT_DETAILS	VARCHAR(10)	
ROOM_NUMBER	DATE	NOT NULL

DEPARTMENT_HOSP Table:

NAME	TYPE	CONSTRAINTS
DEPARTMENT_ID	VARCHAR(10)	PRIMARY KEY
DEPARTMENT_NAME	VARCHAR(20)	NOT NULL

ADMISSION Table:

NAME	TYPE	CONSTRAINTS
PATIENT_ID	BIGINT	PRIMARY KEY
PATIENT_NAME	CHAR(30)	NOT NULL
SSN	VARCHAR(10)	
DOB	DATE	NOT NULL
EMAIL	VARCHAR(30)	
PHONE_NUMBER	VARCHAR(10)	
INSURANCE_NUMBER	VARCHAR(10)	NOT NULL

STAFF Table –

NAME	TYPE	CONSTRAINTS
STAFF_ID	BIGINT	PRIMARY KEY
STAFF_NAME	VARCHAR(30)	NOT NULL
DESIGNATION	VARCHAR(25)	NOT NULL
STAFF_ADDRESS	VARCHAR(50)	
EMAIL	VARCHAR(30)	
CELL_NUMBER	VARCHAR(10)	

QUALIFICATION	VARCHAR(25)	NOT NULL

DOCTOR Table –

NAME	TYPE	CONSTRAINTS
DOCTOR_ID	BIGINT	PRIMARY KEY
DOCTOR_NAME	VARCHAR(30)	NOT NULL
DOCTOR_TYPE	VARCHAR(25)	NOT NULL, CHECK(DOCTOR_TYPE IN ('HOUSE-RESIDENT','ON-CALL'))
STAFF_ID	BIGINT	FOREIGN KEY REFERENCES STAFF
DEPARTMENT_ID	VARCHAR(10)	FOREIGN KEY REFERENCES DEPARTMENT_HOSP
ENCOUNTER_ID	BIGINT	PRIMARY KEY

BILLING TABLE –

NAME	TYPE	CONSTRAINTS
BILLING_ID	BIGINT	PRIMARY KEY
ENCOUNTER_ID	BIGINT	FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
DOCTOR_ID	BIGINT	FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
BILL_AMT	MONEY	
BILL_STATUS	VARCHAR(10)	CHECK(BILL_STATUS IN ('OPEN,'CLOSED','HOLD'))

OUTDOOR_PATIENT -

NAME	TYPE	CONSTRAINTS
ENCOUNTER_ID	BIGINT	PRIMARY KEY, FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
DOCTOR_ID	BIGINT	PRIMARY KEY ,FOREIGN KEY REFERENCES PATIENT_ENCOUNTER
PRESCRIPTION	VARCHAR(25)	

All these tables are in BCNF and yield a robust hospital management database. The attributes for each table along with their type and constraints on them are as mentioned.

4.2 Table Creation Scripts

The above mentioned tables can be created in the database using the following SQL scripts. I have used Microsoft SQL Server 2014 for creation of my database for Hospital Management system.
Table creation steps –

```
BEGIN TRANSACTION

          CREATE TABLE DEPARTMENT_HOSP (
          DEPARTMENT_ID VARCHAR(10) PRIMARY KEY,
          DEPARTMENT_NAME VARCHAR(20) NOT NULL
          )

          CREATE TABLE ADMISSION (
          PATIENT_ID BIGINT,
          SSN VARCHAR(10),
          PATIENT_NAME CHAR(30) NOT NULL,
          PATIENT_ADDRESS VARCHAR(50),
          DOB DATE NOT NULL,
          EMAIL VARCHAR(30),
          PHONE_NUMBER VARCHAR(10),
          INSURANCE_NUMBER VARCHAR(10),
          PRIMARY KEY(PATIENT_ID)
          )

          CREATE TABLE PATIENT_ENCOUNTER(
          ENCOUNTER_ID BIGINT,
          DOCTOR_ID BIGINT,
          PATIENT_ID BIGINT,
          PATIENT_TYPE VARCHAR(10)  NOT NULL CHECK(PATIENT_TYPE IN('OUTDOOR','INDOOR')),
          DEPARTMENT_ID VARCHAR(10),
          ENCOUNTER_DATE DATE,
          PRIMARY KEY(ENCOUNTER_ID, DOCTOR_ID),
          FOREIGN KEY(PATIENT_ID) REFERENCES ADMISSION,
          FOREIGN KEY(DEPARTMENT_ID) REFERENCES DEPARTMENT_HOSP
          )

          CREATE TABLE ROOM (
          ROOM_NUMBER VARCHAR(10),
          ROOM_TYPE VARCHAR(7) CHECK(ROOM_TYPE IN('GENERAL','PRIVATE')),
          ROOM_STATUS CHAR(1) CHECK(ROOM_STATUS IN('Y','N','y','n')),
          PRIMARY KEY(ROOM_NUMBER)
          )

          CREATE TABLE OUTDOOR_PATIENT (
          ENCOUNTER_ID BIGINT,
          DOCTOR_ID BIGINT,
          PRESCRIPTION VARCHAR(25),
          PRIMARY KEY(ENCOUNTER_ID, DOCTOR_ID),
          FOREIGN KEY(ENCOUNTER_ID, DOCTOR_ID) REFERENCES PATIENT_ENCOUNTER)
```

```
CREATE TABLE INDOOR_PATIENT(
ENCOUNTER_ID BIGINT,
DOCTOR_ID BIGINT,
ADMISSION_DATE DATE NOT NULL,
RELEASE_DATE DATE,
TREATMENT_DETAILS VARCHAR(75),
ROOM_NUMBER VARCHAR(10) NOT NULL,
PRIMARY KEY(ENCOUNTER_ID,DOCTOR_ID),
FOREIGN KEY(ENCOUNTER_ID, DOCTOR_ID) REFERENCES PATIENT_ENCOUNTER,
FOREIGN KEY(ROOM_NUMBER) REFERENCES ROOM
)

CREATE TABLE STAFF(
STAFF_ID BIGINT,
STAFF_NAME VARCHAR(30) NOT NULL,
DESIGNATION VARCHAR(25) NOT NULL,
STAFF_ADDRESS VARCHAR(50),
QUALIFICATION VARCHAR(25),
CELL_NO VARCHAR(10),
EMAIL VARCHAR(30),
PRIMARY KEY(STAFF_ID)
)

CREATE TABLE DOCTOR(
DOCTOR_ID BIGINT,
DOCTOR_NAME VARCHAR(25),
DOCTOR_TYPE VARCHAR(25) NOT NULL CHECK(DOCTOR_TYPE IN ('HOSPITAL-
RESIDENT','CALL-ON')),
STAFF_ID BIGINT,
ENCOUNTER_ID BIGINT,
DEPARTMENT_ID VARCHAR(10),
PRIMARY KEY(DOCTOR_ID,ENCOUNTER_ID),
FOREIGN KEY(DEPARTMENT_ID) REFERENCES DEPARTMENT_HOSP,
FOREIGN KEY(STAFF_ID) REFERENCES STAFF
)

CREATE TABLE BILLING(
BILLING_ID BIGINT PRIMARY KEY,
ENCOUNTER_ID BIGINT,
DOCTOR_ID BIGINT,
BILL_AMT MONEY,
BILL_STATUS VARCHAR(10) CHECK(BILL_STATUS IN ('OPEN','CLOSED','HOLD')),
FOREIGN KEY(ENCOUNTER_ID,DOCTOR_ID) REFERENCES PATIENT_ENCOUNTER)
```

COMMIT TRANSACTION

Running the above create statements in the Microsoft SQL Server 2014 creates all the required tables.

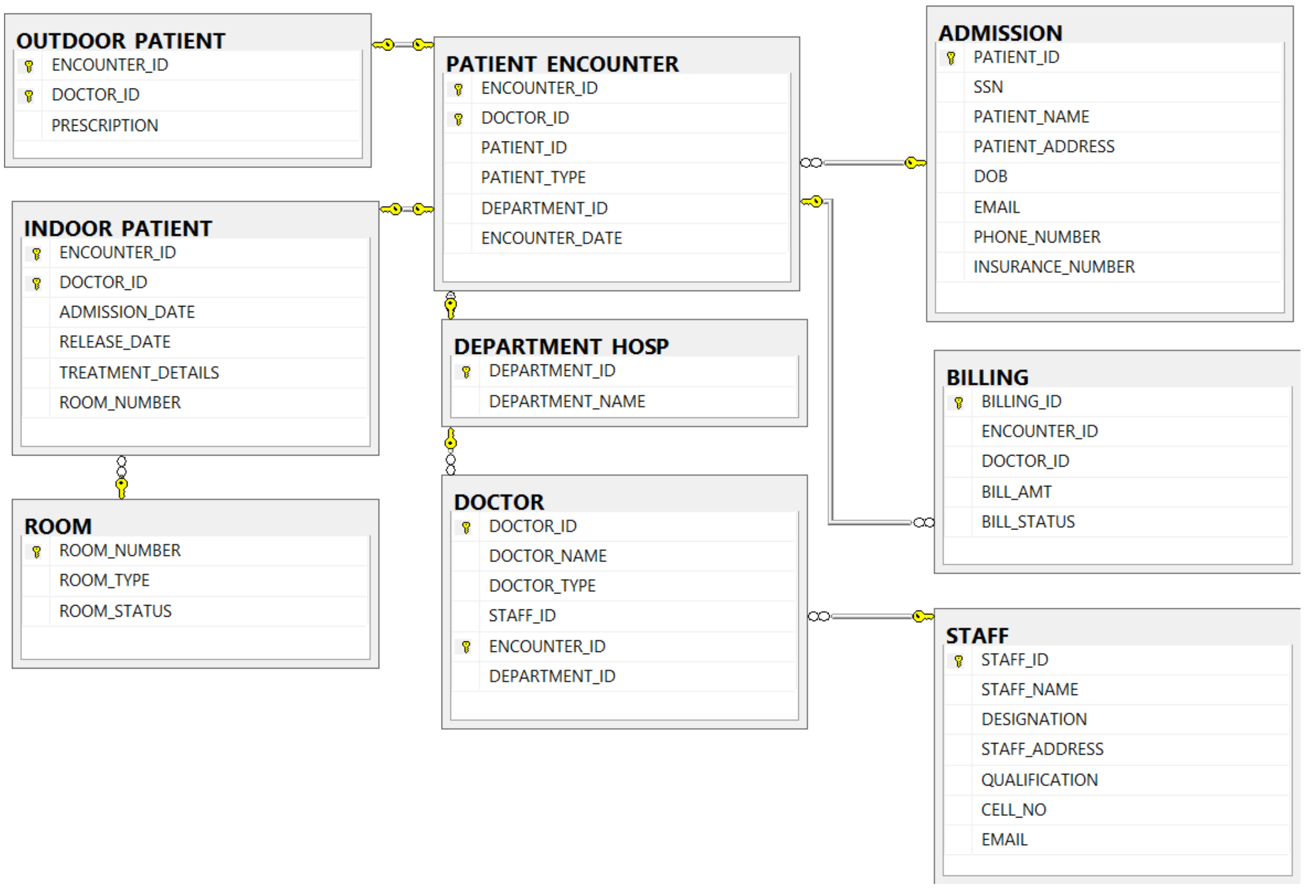
4.3 Insert Scripts

```
BEGIN TRANSACTION

        INSERT INTO DEPARTMENT_HOSP VALUES ('D101','PATHOLOGY')
        INSERT INTO DEPARTMENT_HOSP VALUES ('D102','Gynecology')
        INSERT INTO DEPARTMENT_HOSP VALUES ('D103','Child Ward')
        INSERT INTO DEPARTMENT_HOSP VALUES ('D104','NEUROLOGY')
        INSERT INTO DEPARTMENT_HOSP VALUES ('D105','CARDIOLOGY')

        INSERT INTO ADMISSION VALUES(1001,765878764,'John Brown', '786 Tryeb Ave, Apt
        #308, AP, HYD','7/8/1989','rreke@gmail.com',7362898172,9001)
        INSERT INTO ADMISSION VALUES(1002,165878764,'Black Brown', '918 Tryeb Ave,
        Apt #303, AP, HYD','4/18/1993','tere@gmail.com',6362898172,9002)
        INSERT INTO ADMISSION VALUES(1003,265878764,'John White', '991 Tryeb Ave, Apt
        #305, AP, HYD','3/12/1995','gdfge@gmail.com',8362898172,9003)
        INSERT INTO ADMISSION VALUES(1004,365878764,'Red Organge', '654 Tryeb Ave,
        Apt #301, AP, HYD','1/23/1989','gerrg@gmail.com',1362898272,9004)

        INSERT INTO PATIENT_ENCOUNTER VALUES
        (101,501,1001,'INDOOR','D101','1/1/2009')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (102,502,1001,'INDOOR','D102','10/12/2010')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (103,503,1001,'OUTDOOR','D102','12/09/2010')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (103,505,1001,'OUTDOOR','D104','12/09/2010')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (104,504,1001,'OUTDOOR','D103','12/23/2009')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (105,501,1004,'INDOOR','D101','1/1/2009')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (106,502,1002,'INDOOR','D102','12/23/2010')
        INSERT INTO PATIENT_ENCOUNTER VALUES
        (105,505,1004,'INDOOR','D104','09/12/2010')


        INSERT INTO ROOM VALUES ('R101','PRIVATE','Y')
        INSERT INTO ROOM VALUES ('R102','PRIVATE','Y')
        INSERT INTO ROOM VALUES ('R103','GENERAL','Y')
        INSERT INTO ROOM VALUES ('R104','GENERAL','Y')

        INSERT INTO OUTDOOR_PATIENT VALUES (103,503,'CROCIN')
        INSERT INTO OUTDOOR_PATIENT VALUES (103,505,'DOLO 650')
        INSERT INTO OUTDOOR_PATIENT VALUES (104,504,'COLDACT')

        INSERT INTO INDOOR_PATIENT VALUES
        (101,501,'1/1/2009','2/20/2009','DIABETES','R101')
        INSERT INTO INDOOR_PATIENT VALUES (102,502,'10/12/2010','11/12/2010','HEART
        SURGERY','R102')
        INSERT INTO INDOOR_PATIENT VALUES (106,502,'12/23/2010','1/12/2011','KIDNEY
        PROBLEM','R103')
        INSERT INTO INDOOR_PATIENT VALUES
        (105,505,'09/12/2010','10/12/2010','DIABETES','R104')
        INSERT INTO INDOOR_PATIENT VALUES
        (105,501,'1/1/2009','2/1/2009','DIABETES','R104')

        INSERT INTO STAFF VALUES (3001,'KATE SMITH','DOCTOR','437 DOCTOR BLD,
        NY', 'MBBS',7647682912,'ERFW@KDJG.COM')
        INSERT INTO STAFF VALUES (3002,'KITE SMALL','DOCTOR','412 DOCTOR BLD,
        TR','MBBS',7643482912,'ERDFGW@KDJG.COM')
        INSERT INTO STAFF VALUES (3003,'WANE JANE','DOCTOR','401 DOCTOR BLD,
        KT','MBBS',7647982912,'EGER@KDJG.COM')
        INSERT INTO STAFF VALUES (3004,'KATE JACKIE','DOCTOR','237 DOCTOR BLD,
        YW','MBBS',7653382912,'ERGW@KDJG.COM')
        INSERT INTO STAFF VALUES (3005,'WILLSON JILL','DOCTOR','33 DOCTOR BLD,
        KY','MBBS',7689382912,'DFW@KDJG.COM')
        INSERT INTO STAFF VALUES (3006,'KATE ABBE','DOCTOR','98 DOCTOR BLD,
        NY','MBBS',7642376912,'ERFW@DFDJG.COM')

        INSERT INTO DOCTOR VALUES (501,'KATE SMITH','HOSPITAL-
        RESIDENT',3001,101,'D101')
        INSERT INTO DOCTOR VALUES (502,'KITE SMALL','CALL-ON',3002,102,'D102')
        INSERT INTO DOCTOR VALUES (503,'WANE JANE','HOSPITAL-
        RESIDENT',3003,103,'D102')
        INSERT INTO DOCTOR VALUES (504,'KATE JACKIE','CALL-ON',3004,104,'D103')
        INSERT INTO DOCTOR VALUES (505,'WILSON JILL','HOSPITAL-
        RESIDENT',3005,105,'D104')
        INSERT INTO DOCTOR VALUES (502,'KITE SMALL','CALL-ON',3002,106,'D102')
        INSERT INTO DOCTOR VALUES (505,'WILSON JILL','HOSPITAL-
        RESIDENT',3005,103,'D104')
        INSERT INTO DOCTOR VALUES (501,'KATE SMITH','HOSPITAL-
        RESIDENT',3001,105,'D101')


        INSERT INTO BILLING VALUES (701, 101,501,764.89,'OPEN')
```

```
    INSERT INTO BILLING VALUES (702, 106,502,924.9,'CLOSED')
    INSERT INTO BILLING VALUES (703, 102,502,92.45,'OPEN')
    INSERT INTO BILLING VALUES (704, 103,503,43.8,'CLOSED')
    INSERT INTO BILLING VALUES (708, 103,505,123.8,'OPEN')
    INSERT INTO BILLING VALUES (705, 104,504,46.9,'OPEN')
    INSERT INTO BILLING VALUES (706, 105,501,14.39,'OPEN')
    INSERT INTO BILLING VALUES (707, 105,505,76.19,'CLOSED')

COMMIT TRANSACTION
```

4.4 ER – Diagram for the database design

The below shown ER- Diagram explains the tables, their attributes, the primary keys, primary key and foreign key relations. This ER – Diagram is derived from Microsoft SQL Server 2014 for our database.

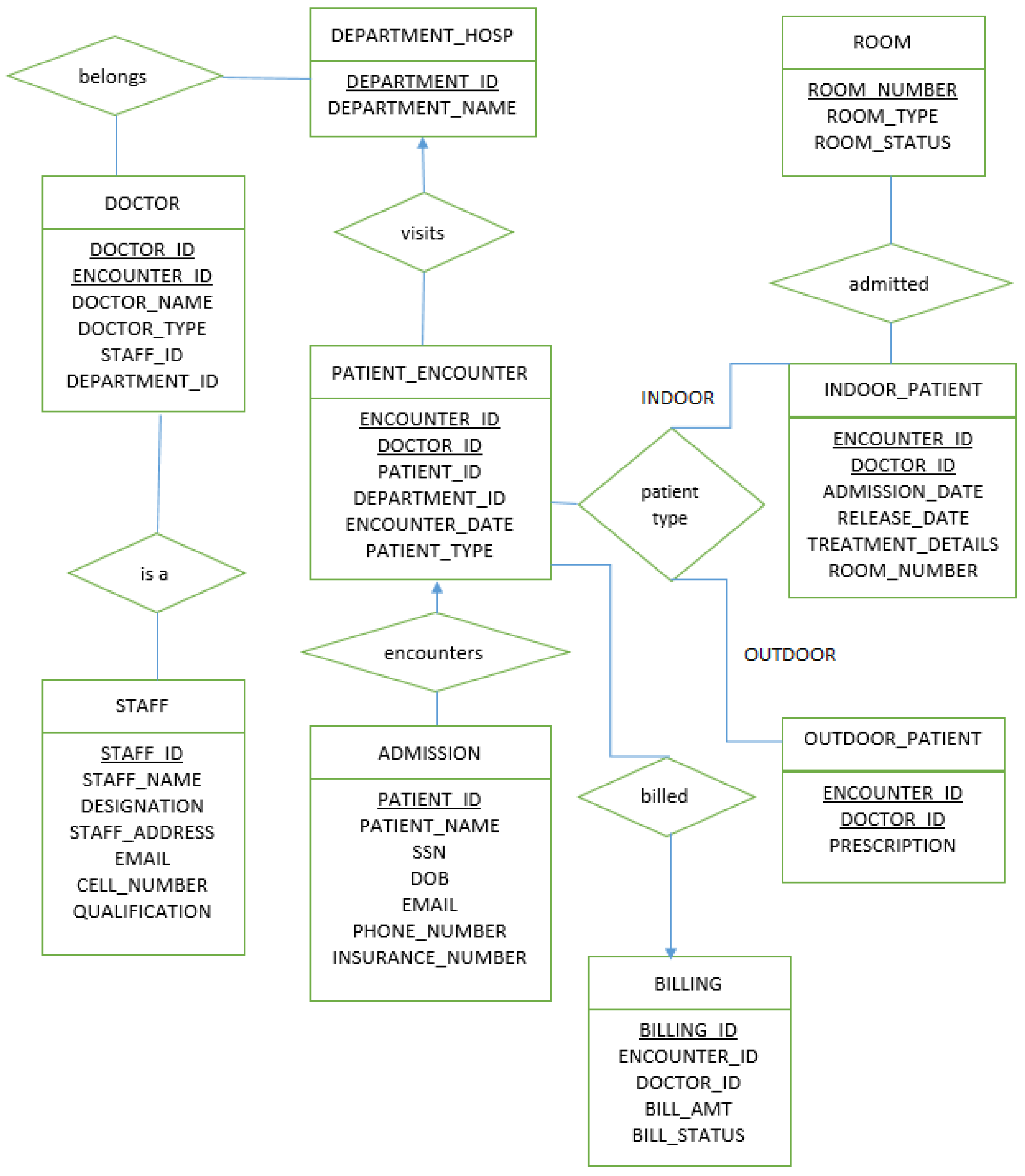


An E-R diagram can express the overall logical structure of a database graphically.
Basic Structure:
An E-R diagram consists of the following major components:

- Rectangles divided into two parts represent entity sets. The first part, contains the name of the entity set. The second part contains the names of all the attributes of the entity set.
- Diamonds represent relationship sets.
- Undivided rectangles represent the attributes of a relationship set.
- Attributes that are part of the primary key are underlined.
- Lines link entity sets to relationship sets.
- Dashed lines link attributes of a relationship set to the relationship set.
- Double lines indicate total participation of an entity in a relationship set.
- Double diamonds represent identifying relationship sets linked to weak entity sets.

Below is the ER-Diagram of our design. All the shapes in the ER-Diagram signifies the above meanings. The arrow mark from one entity set through relationship set to other entity set signifies one-to-many relationship.

This ER-Diagram very much correlates with the one obtained from Microsoft SQL Server 2014 which proves the correctness of our design practically.



4.5 SQL Query execution asked in the problem statement

I have written the queries and executed the queries on Microsoft SQL Server 2014. I have presented the SQL query, expected output and the screenshot of the actual output when run on Microsoft SQL Server 2014.

1. Table creation scripts is shown above as part of the design steps.
2. Display the details of a patient named John Brown and also display the name of doctor (s) who diagnosed him and also his total bill amount.

SQL Query:

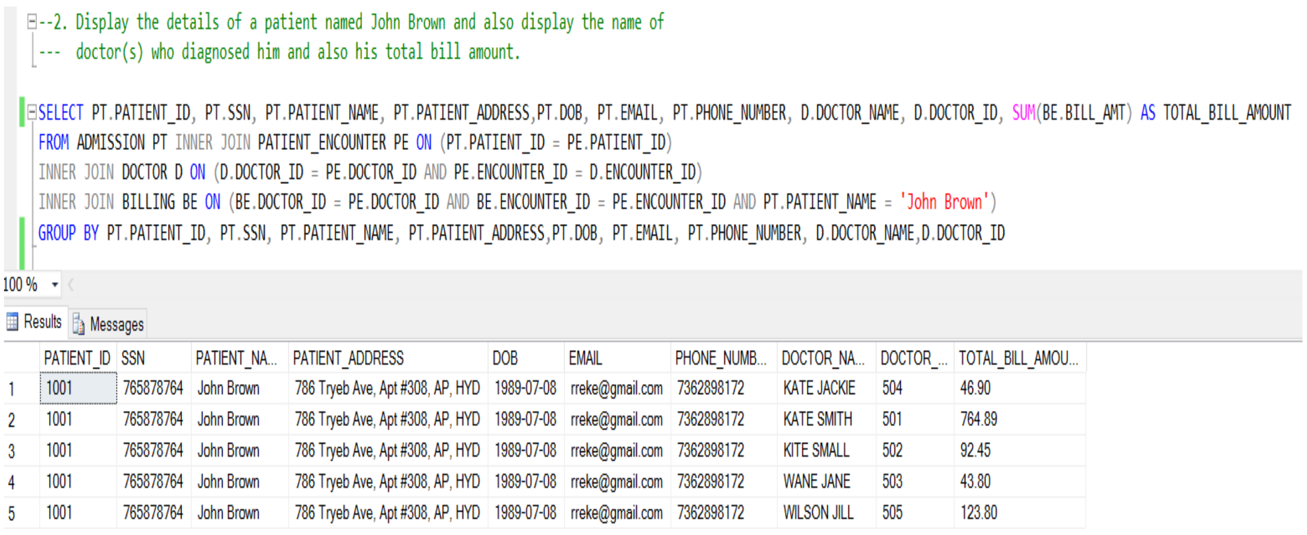
```
SELECT PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS,PT.DOB, PT.EMAIL,
PT.PHONE_NUMBER, D.DOCTOR_NAME, D.DOCTOR_ID, SUM(BE.BILL_AMT) AS
TOTAL_BILL_AMOUNT FROM ADMISSION PT INNER JOIN PATIENT_ENCOUNTER PE
ON (PT.PATIENT_ID = PE.PATIENT_ID)
INNER JOIN DOCTOR D
ON (D.DOCTOR_ID = PE.DOCTOR_ID AND PE.ENCOUNTER_ID = D.ENCOUNTER_ID)
INNER JOIN BILLING BE
ON (BE.DOCTOR_ID = PE.DOCTOR_ID AND BE.ENCOUNTER_ID = PE.ENCOUNTER_ID AND
PT.PATIENT_NAME = 'John Brown')
GROUP BY PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS, PT.DOB, PT.EMAIL,
PT.PHONE_NUMBER, D.DOCTOR_NAME,D.DOCTOR_ID
```

Expected output:

```
1001    765878764    John Brown              786 Tryeb Ave, Apt #308, AP, HYD    1989-07-
08      rreke@gmail.com    7362898172    KATE JACKIE    504    46.90
1001    765878764    John Brown              786 Tryeb Ave, Apt #308, AP, HYD    1989-07-
08      rreke@gmail.com    7362898172    KATE SMITH     501    764.89
1001    765878764    John Brown              786 Tryeb Ave, Apt #308, AP, HYD    1989-07-
08      rreke@gmail.com    7362898172    KITE SMALL     502    92.45
1001    765878764    John Brown              786 Tryeb Ave, Apt #308, AP, HYD    1989-07-
08      rreke@gmail.com    7362898172    WANE JANE      503    43.80
1001    765878764    John Brown              786 Tryeb Ave, Apt #308, AP, HYD    1989-07-
08      rreke@gmail.com    7362898172    WILSON JILL    505    123.80
```

The query should return the above five records for the Patient John Brown.

Screenshot of the output when the query is run on Microsoft SQL Server 2014



```
--2. Display the details of a patient named John Brown and also display the name of
---  doctor(s) who diagnosed him and also his total bill amount.

SELECT PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS,PT.DOB, PT.EMAIL, PT.PHONE_NUMBER, D.DOCTOR_NAME, D.DOCTOR_ID, SUM(BE.BILL_AMT) AS TOTAL_BILL_AMOUNT
FROM ADMISSION PT INNER JOIN PATIENT_ENCOUNTER PE ON (PT.PATIENT_ID = PE.PATIENT_ID)
INNER JOIN DOCTOR D ON (D.DOCTOR_ID = PE.DOCTOR_ID AND PE.ENCOUNTER_ID = D.ENCOUNTER_ID)
INNER JOIN BILLING BE ON (BE.DOCTOR_ID = PE.DOCTOR_ID AND BE.ENCOUNTER_ID = PE.ENCOUNTER_ID AND PT.PATIENT_NAME = 'John Brown')
GROUP BY PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS,PT.DOB, PT.EMAIL, PT.PHONE_NUMBER, D.DOCTOR_NAME,D.DOCTOR_ID
```

100 %

Results | Messages

	PATIENT_ID	SSN	PATIENT_NA...	PATIENT_ADDRESS	DOB	EMAIL	PHONE_NUMB...	DOCTOR_NA...	DOCTOR_...	TOTAL_BILL_AMOU...
1	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	KATE JACKIE	504	46.90
2	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	KATE SMITH	501	764.89
3	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	KITE SMALL	502	92.45
4	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	WANE JANE	503	43.80
5	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	WILSON JILL	505	123.80

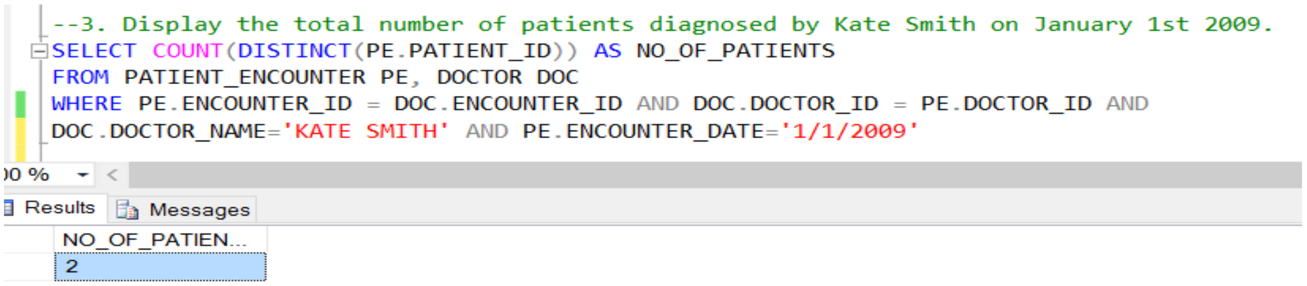
3. Display the total number of patients diagnosed by Kate Smith on January 1st 2009.

<u>SQL Query:</u>

SELECT COUNT (DISTINCT (PE.PATIENT_ID)) AS NO_OF_PATIENTS
FROM PATIENT_ENCOUNTER PE, DOCTOR DOC
WHERE PE.ENCOUNTER_ID = DOC.ENCOUNTER_ID AND DOC.DOCTOR_ID = PE.DOCTOR_ID AND DOC.DOCTOR_NAME='KATE SMITH' AND PE.ENCOUNTER_DATE='1/1/2009'

Expected Output:
2

Screenshot of the output when the query is run on Microsoft SQL Server 2014



```
--3. Display the total number of patients diagnosed by Kate Smith on January 1st 2009.
SELECT COUNT(DISTINCT(PE.PATIENT_ID)) AS NO_OF_PATIENTS
FROM PATIENT_ENCOUNTER PE, DOCTOR DOC
WHERE PE.ENCOUNTER_ID = DOC.ENCOUNTER_ID AND DOC.DOCTOR_ID = PE.DOCTOR_ID AND
DOC.DOCTOR_NAME='KATE SMITH' AND PE.ENCOUNTER_DATE='1/1/2009'
```

)0 %

Results | Messages

NO_OF_PATIEN...
2

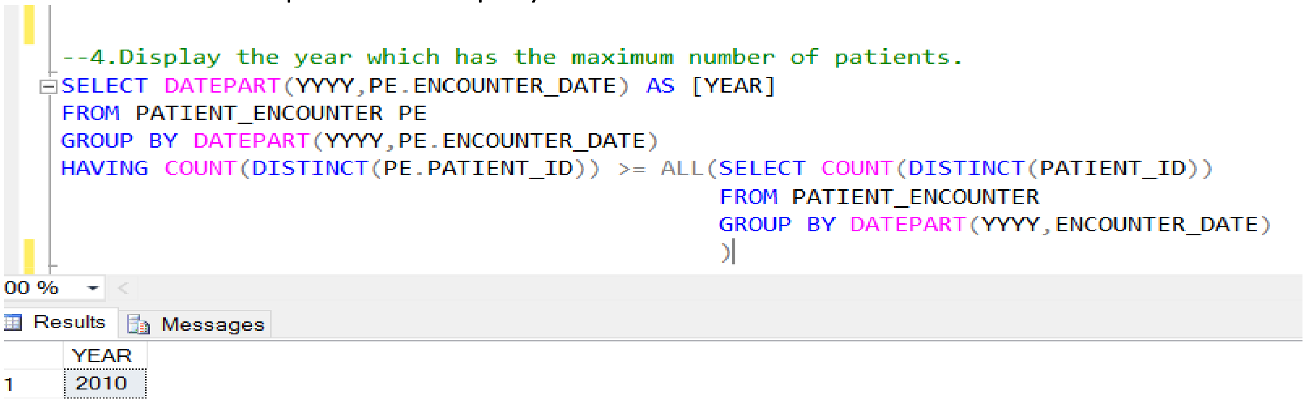
4. Display the year which has the maximum number of patients.

<u>SQL Queries:</u>

```
SELECT DATEPART (YYYY, PE.ENCOUNTER_DATE) AS [YEAR]
FROM PATIENT_ENCOUNTER PE
GROUP BY DATEPART (YYYY, PE.ENCOUNTER_DATE)
HAVING COUNT (DISTINCT (PE.PATIENT_ID)) >= ALL (SELECT COUNT (DISTINCT (PATIENT_ID))
 FROM PATIENT_ENCOUNTER
GROUP BY DATEPART (YYYY, ENCOUNTER_DATE)
 )
```

Expected Output:
2010

Screenshot of the output when the query is run on Microsoft SQL Server 2014



```
--4.Display the year which has the maximum number of patients.
SELECT DATEPART(YYYY,PE.ENCOUNTER_DATE) AS [YEAR]
FROM PATIENT_ENCOUNTER PE
GROUP BY DATEPART(YYYY,PE.ENCOUNTER_DATE)
HAVING COUNT(DISTINCT(PE.PATIENT_ID)) >= ALL(SELECT COUNT(DISTINCT(PATIENT_ID))
                                             FROM PATIENT_ENCOUNTER
                                             GROUP BY DATEPART(YYYY,ENCOUNTER_DATE)
                                             )
```

```
  YEAR
1 2010
```

5. Display the details of a patient and the supervising doctor who was admitted more than once during the year 2010.

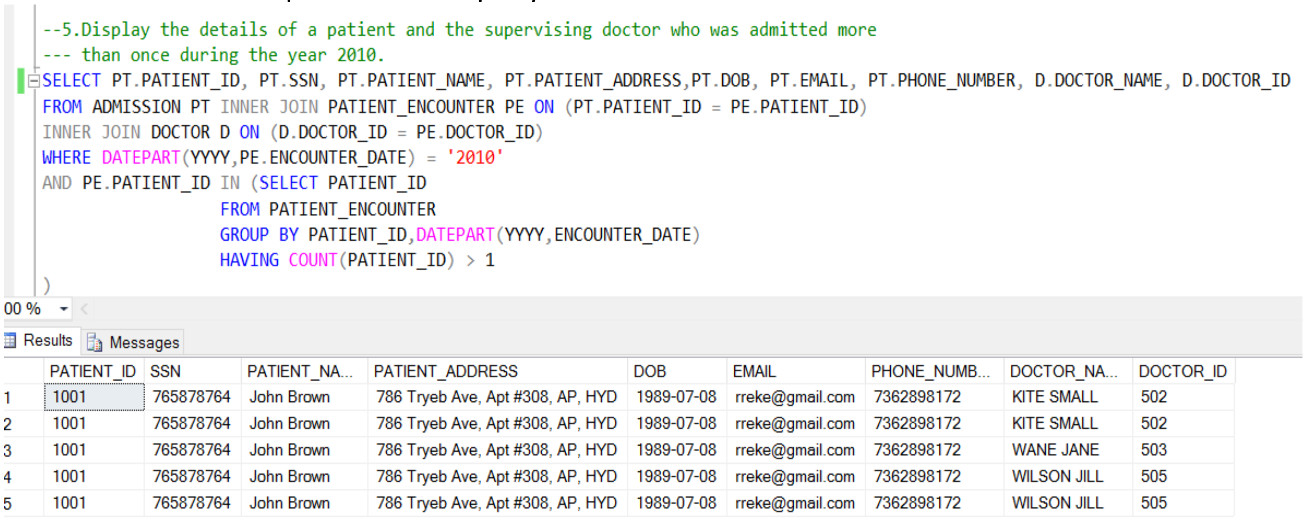
<u>SQL Query:</u>

```
SELECT PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS,PT.DOB, PT.EMAIL,
PT.PHONE_NUMBER, D.DOCTOR_NAME, D.DOCTOR_ID
FROM ADMISSION PT INNER JOIN PATIENT_ENCOUNTER PE
ON (PT.PATIENT_ID = PE.PATIENT_ID)
INNER JOIN DOCTOR D
ON (D.DOCTOR_ID = PE.DOCTOR_ID)
WHERE DATEPART (YYYY,PE.ENCOUNTER_DATE) = '2010'
AND PE.PATIENT_ID IN (SELECT PATIENT_ID FROM PATIENT_ENCOUNTER
                      GROUP BY PATIENT_ID,DATEPART (YYYY,ENCOUNTER_DATE)
                      HAVING COUNT (PATIENT_ID) > 1)
```

Expected Output:

```
1001    765878764     John Brown                    786 Tryeb Ave, Apt #308, AP, HYD     1989-07-
08      rreke@gmail.com       7362898172    KITE SMALL     502
1001    765878764     John Brown                    786 Tryeb Ave, Apt #308, AP, HYD     1989-07-
08      rreke@gmail.com       7362898172    KITE SMALL     502
1001    765878764     John Brown                    786 Tryeb Ave, Apt #308, AP, HYD     1989-07-
08      rreke@gmail.com       7362898172    WANE JANE      503
1001    765878764     John Brown                    786 Tryeb Ave, Apt #308, AP, HYD     1989-07-
08      rreke@gmail.com       7362898172    WILSON JILL    505
1001    765878764     John Brown                    786 Tryeb Ave, Apt #308, AP, HYD     1989-07-
08      rreke@gmail.com       7362898172    WILSON JILL    505
```

Screenshot of the output when the query is run on Microsoft SQL Server 2014



```
--5.Display the details of a patient and the supervising doctor who was admitted more
--- than once during the year 2010.
SELECT PT.PATIENT_ID, PT.SSN, PT.PATIENT_NAME, PT.PATIENT_ADDRESS,PT.DOB, PT.EMAIL, PT.PHONE_NUMBER, D.DOCTOR_NAME, D.DOCTOR_ID
FROM ADMISSION PT INNER JOIN PATIENT_ENCOUNTER PE ON (PT.PATIENT_ID = PE.PATIENT_ID)
INNER JOIN DOCTOR D ON (D.DOCTOR_ID = PE.DOCTOR_ID)
WHERE DATEPART(YYYY,PE.ENCOUNTER_DATE) = '2010'
AND PE.PATIENT_ID IN (SELECT PATIENT_ID
                  FROM PATIENT_ENCOUNTER
                  GROUP BY PATIENT_ID,DATEPART(YYYY,ENCOUNTER_DATE)
                  HAVING COUNT(PATIENT_ID) > 1
)
```

	PATIENT_ID	SSN	PATIENT_NA...	PATIENT_ADDRESS	DOB	EMAIL	PHONE_NUMB...	DOCTOR_NA...	DOCTOR_ID
1	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	KITE SMALL	502
2	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	KITE SMALL	502
3	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	WANE JANE	503
4	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	WILSON JILL	505
5	1001	765878764	John Brown	786 Tryeb Ave, Apt #308, AP, HYD	1989-07-08	rreke@gmail.com	7362898172	WILSON JILL	505

6. Display details of the department/units the doctors of which have billed the maximum amounts.

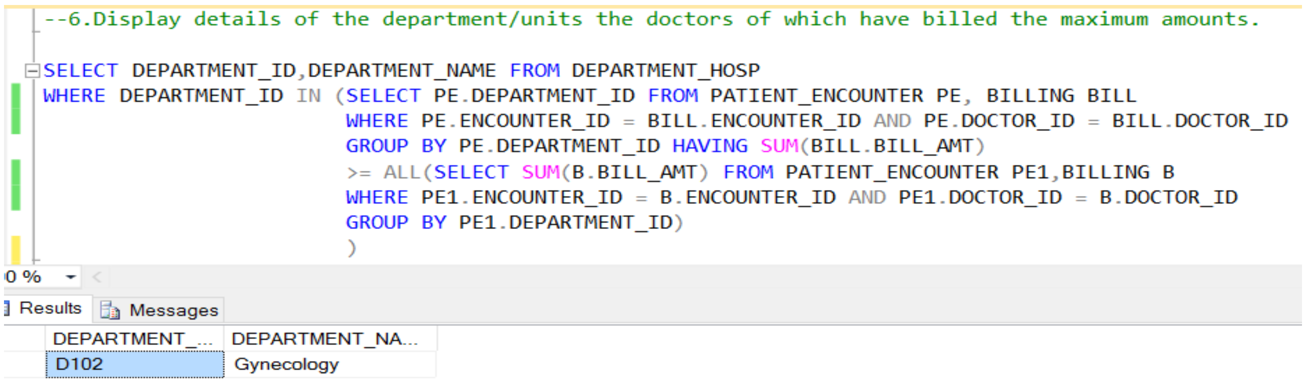
<u>SQL Query:</u>

```
SELECT DEPARTMENT_ID , DEPARTMENT_NAME FROM DEPARTMENT_HOSP
WHERE DEPARTMENT_ID IN
(SELECT PE.DEPARTMENT_ID FROM PATIENT_ENCOUNTER PE, BILLING BILL
WHERE PE.ENCOUNTER_ID = BILL.ENCOUNTER_ID AND PE.DOCTOR_ID = BILL.DOCTOR_ID
GROUP BY PE.DEPARTMENT_ID HAVING SUM (BILL.BILL_AMT) >= ALL
(SELECT SUM (B.BILL_AMT) FROM PATIENT_ENCOUNTER PE1, BILLING B
        WHERE PE1.ENCOUNTER_ID = B.ENCOUNTER_ID AND PE1.DOCTOR_ID = B.DOCTOR_ID
                GROUP BY PE1.DEPARTMENT_ID)
)
```

Expected Output:

D102 – Gynecology

Screenshot of the output when the query is run on Microsoft SQL Server 2014



```
--6.Display details of the department/units the doctors of which have billed the maximum amounts.

SELECT DEPARTMENT_ID,DEPARTMENT_NAME FROM DEPARTMENT_HOSP
WHERE DEPARTMENT_ID IN (SELECT PE.DEPARTMENT_ID FROM PATIENT_ENCOUNTER PE, BILLING BILL
                        WHERE PE.ENCOUNTER_ID = BILL.ENCOUNTER_ID AND PE.DOCTOR_ID = BILL.DOCTOR_ID
                        GROUP BY PE.DEPARTMENT_ID HAVING SUM(BILL.BILL_AMT)
                        >= ALL(SELECT SUM(B.BILL_AMT) FROM PATIENT_ENCOUNTER PE1,BILLING B
                        WHERE PE1.ENCOUNTER_ID = B.ENCOUNTER_ID AND PE1.DOCTOR_ID = B.DOCTOR_ID
                        GROUP BY PE1.DEPARTMENT_ID)
                        )
```

DEPARTMENT_...	DEPARTMENT_NA...
D102	Gynecology

5. Conclusion

The project thus presents a robust database design for Hospital Management System by following all the normalization rules with perfect selection of primary keys and foreign keys.
The project is able to output the SQL queries asked in the problem description as expected.

6. References

- DATABASE SYSTEM CONCEPTS Sixth Edition by Abraham Silberschat, Henry F. Korth, S. Sudarshan
- https://msdn.microsoft.com/en-us/library/bb545450.aspx for Microsoft SQL tutorials and basics.