**FUNDAMENTALS OF DATA MANAGEMENT**

**COS20015**

**NAME: SYED OMAIR MAQDOOM MOHIUDDIN**

**STUDENT ID: 102873768**

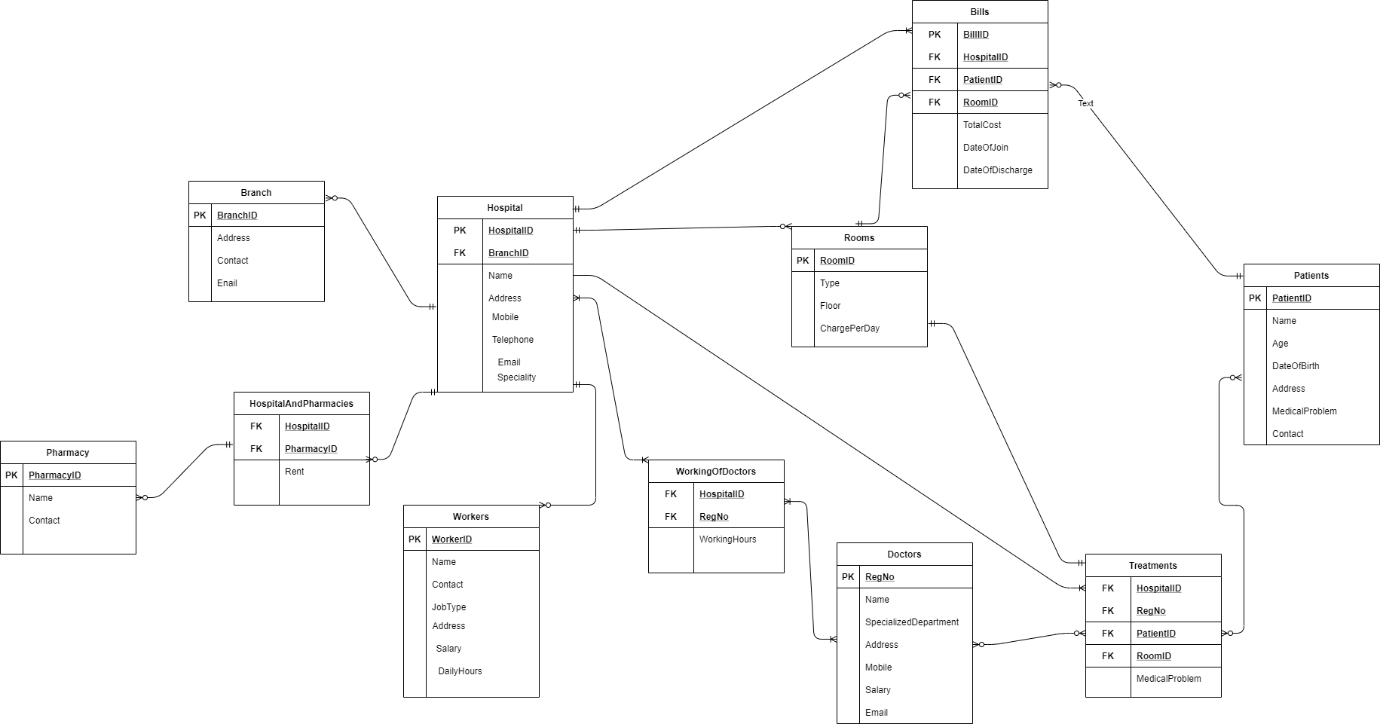
**TUTOR: Harindu Korala**

**Design Report**

This is a report for creating a hospital database. The data storage chosen was MYSQL. In the data base we have hospital branches if it contains any, doctors and workers who work for the hospital. Patients who visit hospital and are treated by 1 or many doctors. If they are admitted in hospital then the room allotted to them, bill at the time of discharge. Every hospital has one or more specialities, doctors are specialized in one or more department of medication. Each hospital may or may not have a pharmacy. So, looking at all these aspects I created a database to store all the information of the hospital.

Every table in the database has primary and foreign key constraints. They also contain indexes to retrieve the data from the database more quickly than otherwise.

**ER Diagram for the Database:**



**Normalisation:**

As you can see from the Entity relation diagram, it is in the 3NF. Normalization is like filtering some dirty water to remove all the dirt from the water. In filtering water, we have many filters and phases, in the same way we have many phases in normalisation (1NF, 2NF, 3Nf, …). Normalization helps us by getting rid of Redundancy. Redundancy is something which we have more than once. For instance, A patient can visit the hospital multiple times, or a doctor can treat a patient multiple times, so, this patient with their id will have multiple rows with similar data.

So, this redundant data leads to various problems like, INSERT anomalies, UPDATE anomalies, DELETE anomalies and many other. Using Normalisation, we can reduce data redundancy. Redundancy is expressed in terms of dependencies. Various Normal Forms are defined to have certain types of dependency. 1NF there are no composite or multi-valued attributes, in other words every attribute in the table should be single valued attributes. All the tables in the database are in 1NF.

To be in 2NF any relation should be in 1NF and it should not contain no Partial dependency. So, identification of functional dependency is necessary for second normal form. 2NF handles the UPDATE anomalies. Any primary key in case of 2NF cannot be a composite key in case it arises any partial dependency.

**Consider a relation R (W, X, Y, Z) having functional dependency (WX->YZ, XY->Z}**

**Closure of (WX) = {W, X, Y, Z}. So, the Super key will be “WX”.**

**In WX 🡪 YZ (WX is the Super Key and Y, Z is non-prime)**

**In XY 🡪 Z (XY is non-prime and Z is non=prime which is allowed in 2NF)**

**The above relation is in 2Nf because there is no prime attribute is deriving non-prime attribute that is there is no partial functional dependency.**

**Although it is not in 3NF because non=prime attribute is deriving non-prime attribute.**

So, removing all the partial dependencies from the database gave the 2NF.

To get the last normalised form, the database should be in 1Nf as well as 2NF. In 3NF non-prime attributes are only allowed to be functionally dependent on Super Key (combination of different attributes to from a primary key) of relation. No transitive dependency on a Super key or candidate key. 3Nf virtually eliminates all the redundancies. The goal of the third normal form is to ensure referential integrity.

**Consider a relation R(X ,Y, Z) having functional dependency (XY->Z, Z->X}**

**Closure of (XY) = {X, Y, Z}**

**Closure of (YZ) = {X, Y, Z}**

**So, the Candidate Keys are {XY, YZ}**

**XY 🡪 Z (prime deriving prime)**

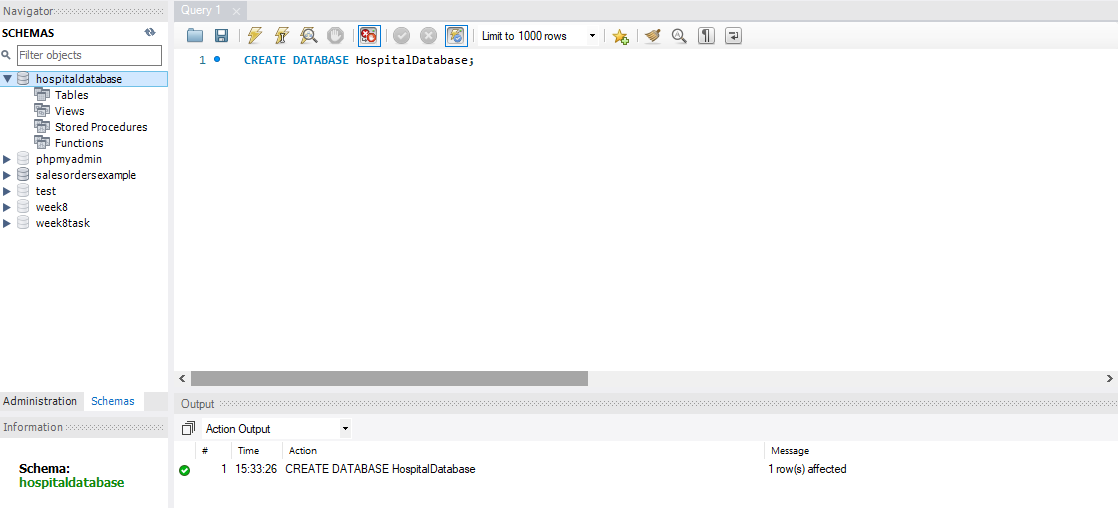
**Z🡪 X (prime deriving prime).**

**There is no partial functional dependency and no transitive dependency. It is in 3NF.**

From the above data, we can conclude that the database is in 3NF.

**Creation of Database:**

Command: **CREATE DATABASE HospitalDatabase;**



**Addition of Tables:**

1. Creating Branch table

Command:

**CREATE TABLE Branch (**

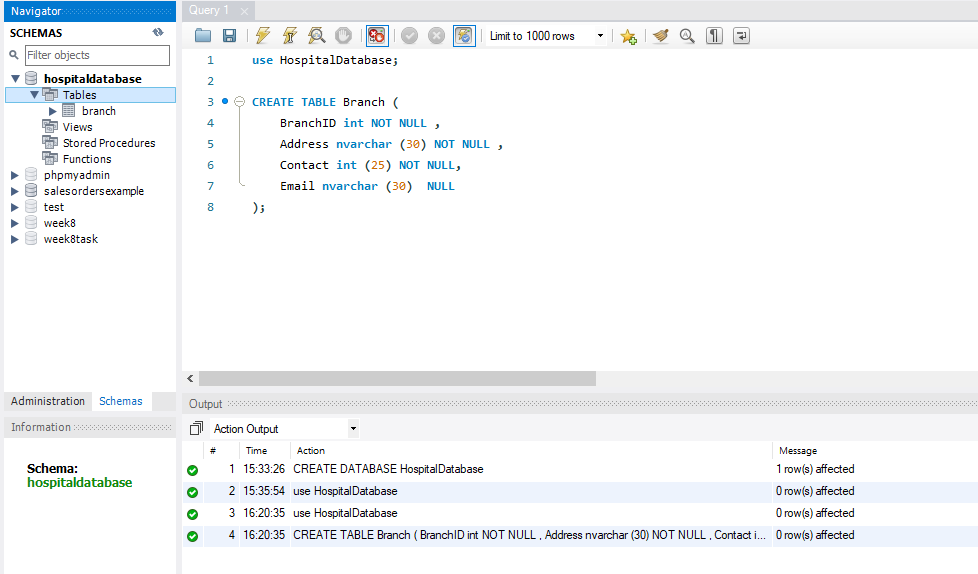
**BranchID int NOT NULL ,**

**Address nvarchar (30) NOT NULL ,**

**Contact int (25) NOT NULL,**

**Email nvarchar (30) NULL**

**);**



1. Creating Hospital Table

Command:

**CREATE TABLE Hospital (**

**HospitalID int NOT NULL,**

**BranchID int NOT NULL ,**

**Name nvarchar (30) NOT NULL,**

**Address nvarchar (30) NOT NULL ,**

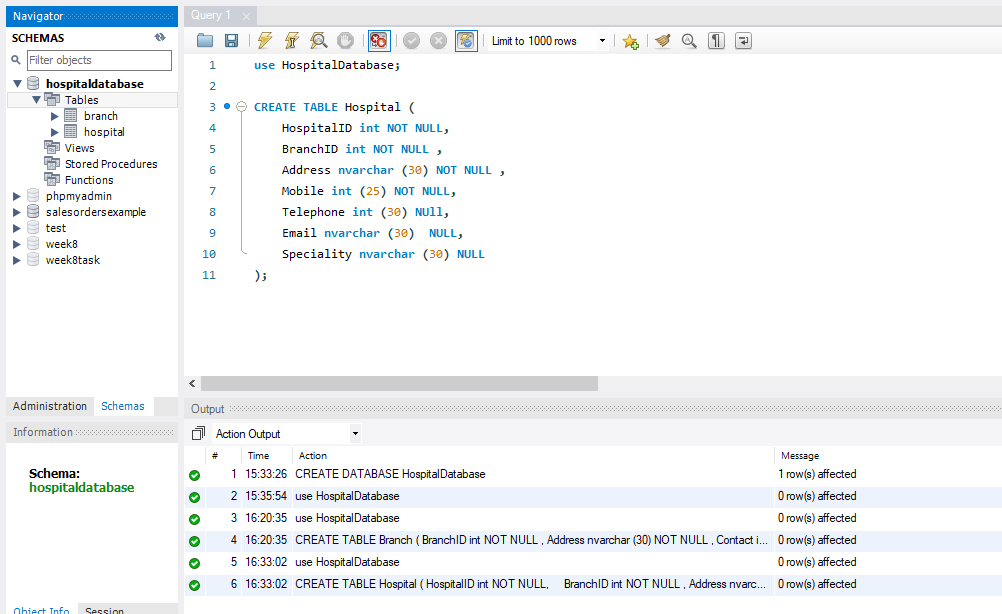
**Mobile int (25) NOT NULL,**

**Telephone int (30) NULL,**

**Email nvarchar (30) NULL,**

**Speciality nvarchar (30) NULL**

**);**



1. Creating Doctors Table

Command:

**CREATE TABLE Doctors (**

**RegNo int NOT NULL,**

**Name nvarchar (30) NOT NULL,**

**SpecializedDepartment nvarchar (20) NULL ,**

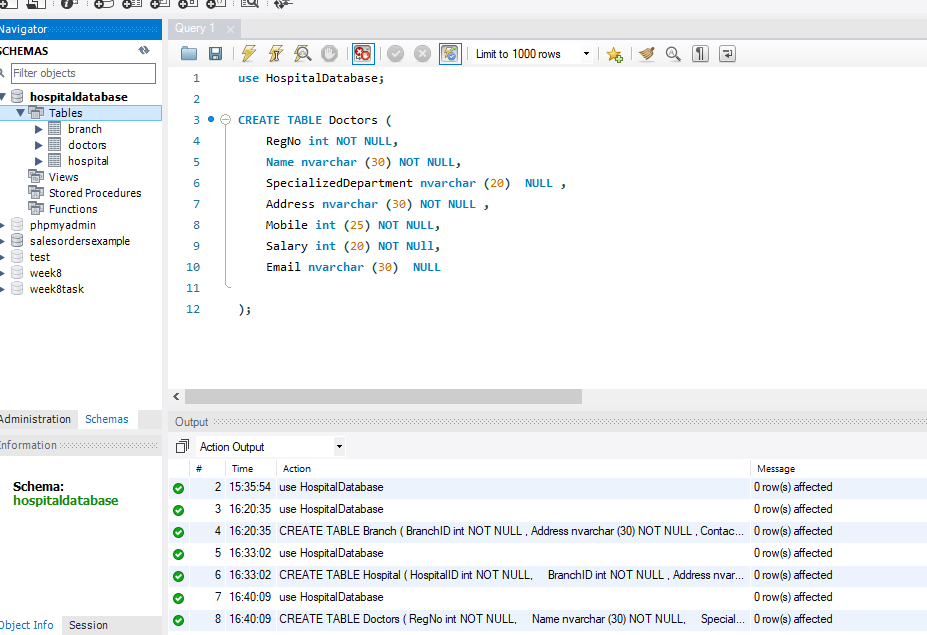
**Address nvarchar (30) NOT NULL ,**

**Mobile int (25) NOT NULL,**

**Salary int (20) NOT NUll,**

**Email nvarchar (30) NULL**

**);**



1. Creating Patients Table

Command:

**CREATE TABLE Patients (**

**PatientID int NOT NULL,**

**Age int NOT NULL ,**

**DateOfBirth DATE NOT NULL,**

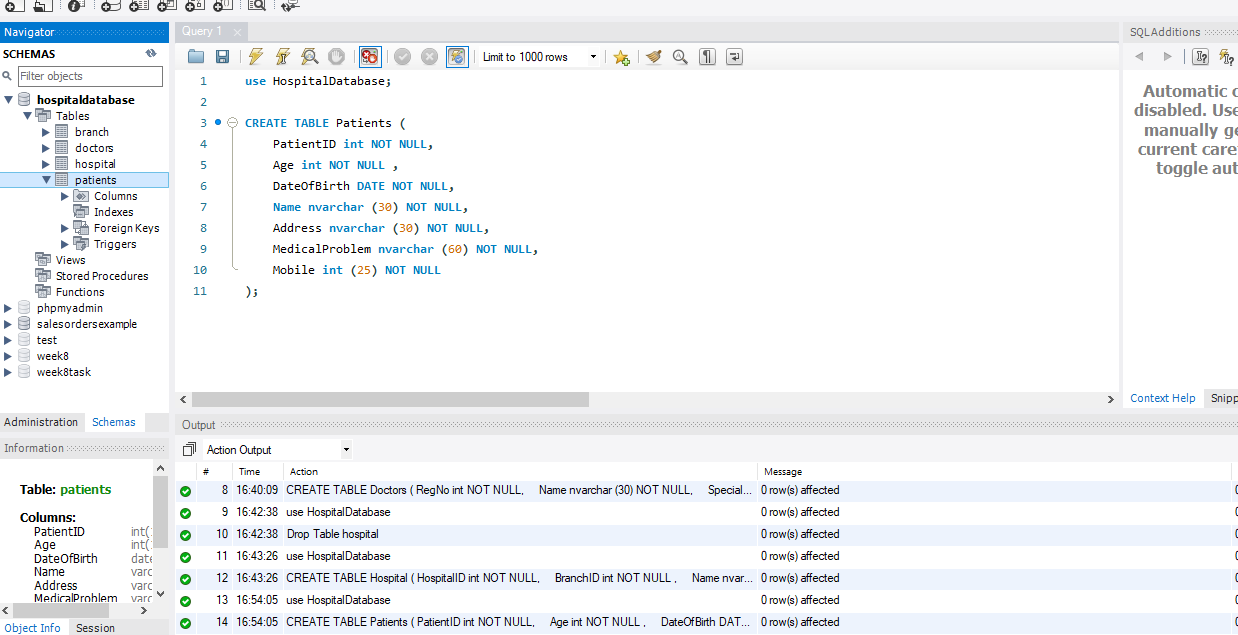
**Name nvarchar (30) NOT NULL,**

**Address nvarchar (30) NOT NULL,**

**MedicalProblem nvarchar (60) NOT NULL,**

**Mobile int (25) NOT NULL**

**);**



1. Creating Rooms/Wards Table

Command:

**CREATE TABLE Rooms (**

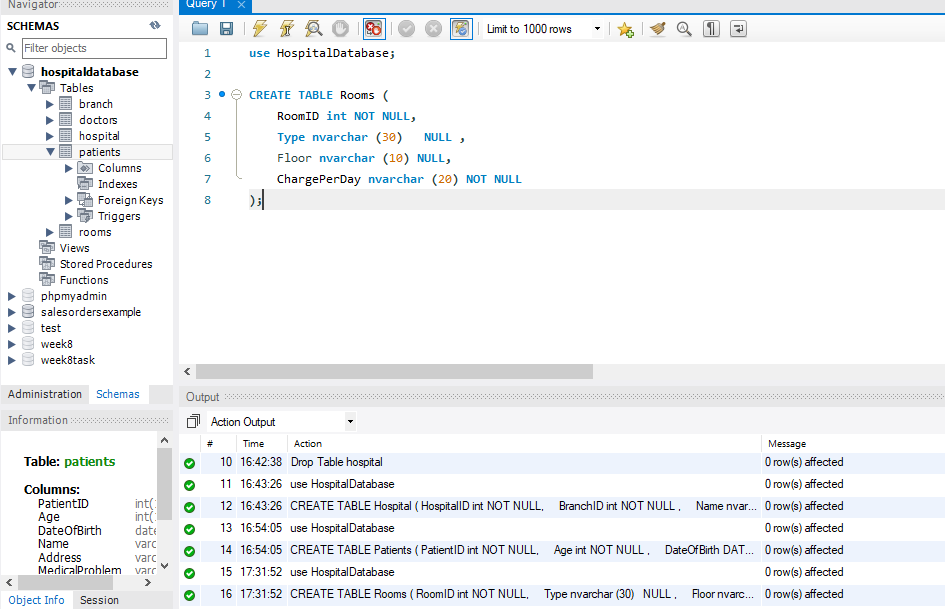
**RoomID int NOT NULL,**

**Type nvarchar (30) NULL ,**

**Floor nvarchar (10) NULL,**

**ChargePerDay nvarchar (20) NOT NULL**

**);**



1. **Create Pharmacy Table**

Command:

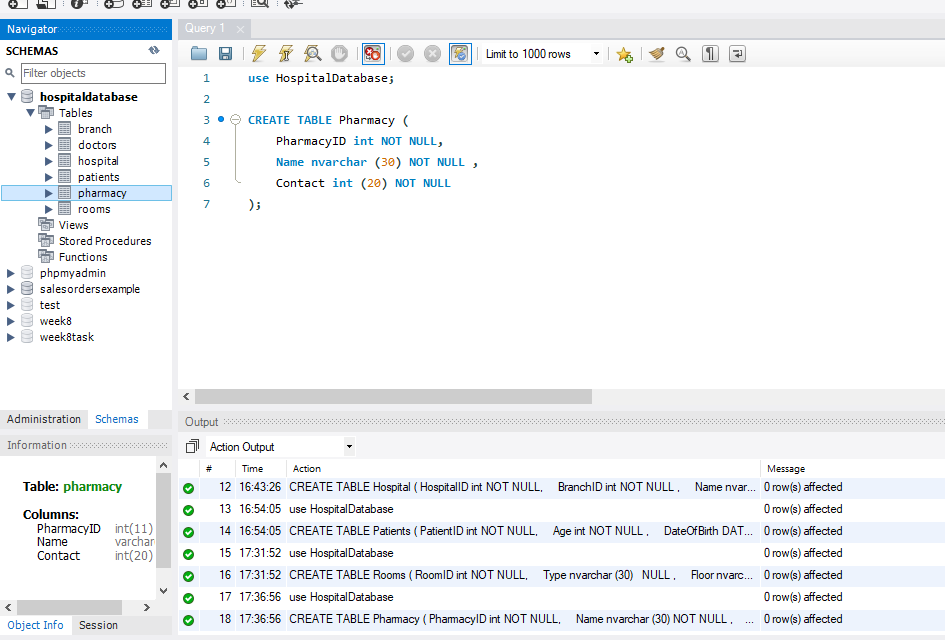
**CREATE TABLE Pharmacy (**

**PharmacyID int NOT NULL,**

**Name nvarchar (30) NOT NULL ,**

**Contact int (20) NOT NULL**

**);**



1. **Create Workers Table**

Command:

**CREATE TABLE Workers (**

**WorkerID int NOT NULL,**

**Name nvarchar (30) NOT NULL ,**

**Contact int (20) NOT NULL,**

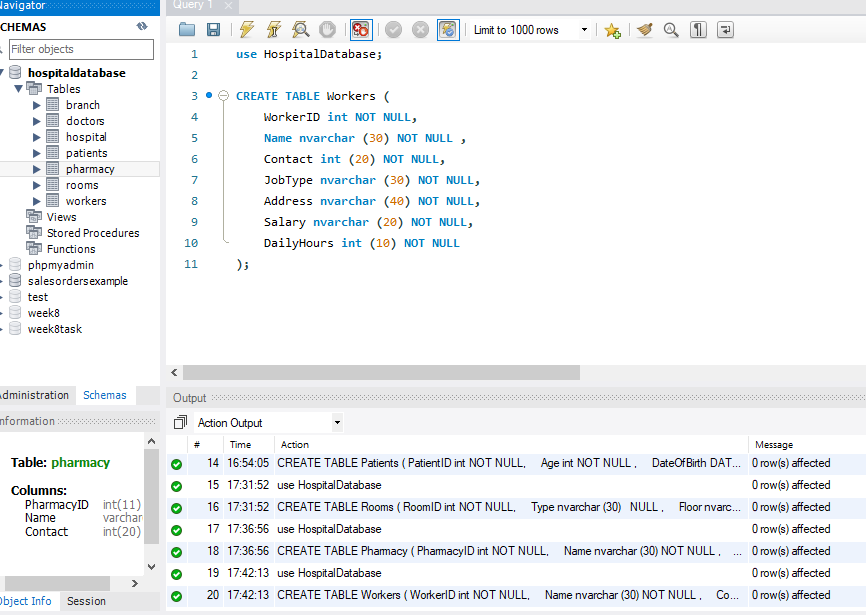
**JobType nvarchar (30) NOT NULL,**

**Address nvarchar (40) NOT NULL,**

**Salary nvarchar (20) NOT NULL,**

**DailyHours int (10) NOT NULL**

**);**



1. **Create Bills Table**

Command:

**CREATE TABLE Bills (**

**BillID int NOT NULL,**

**PaitentID int (11) NOT NULL ,**

**HospitalID int (11) NOT NULL,**

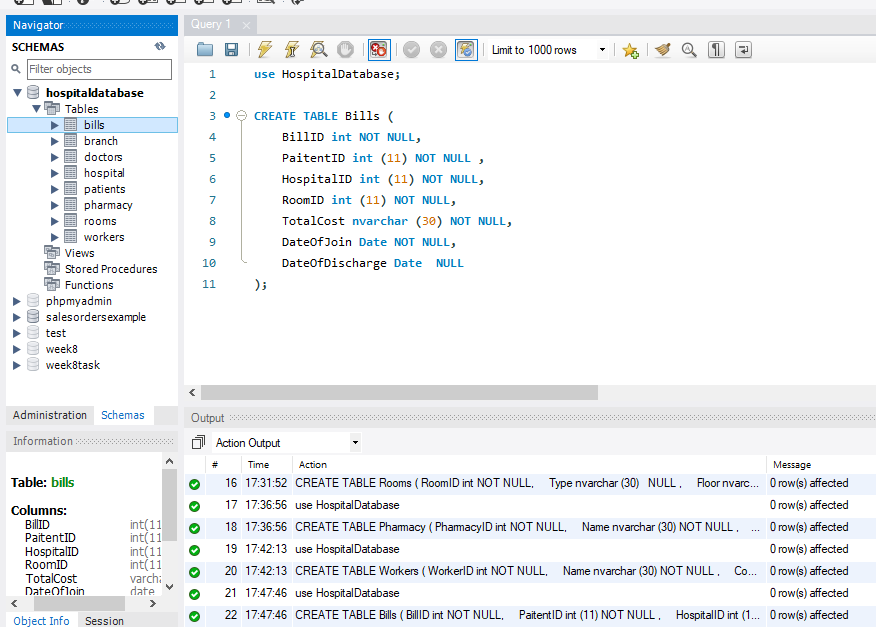
**RoomID int (11) NOT NULL,**

**TotalCost nvarchar (30) NOT NULL,**

**DateOfJoin Date NOT NULL,**

**DateOfDischarge Date NULL**

**);**



1. **Creating Pharmacy and Hospital Relation Table**

Command:

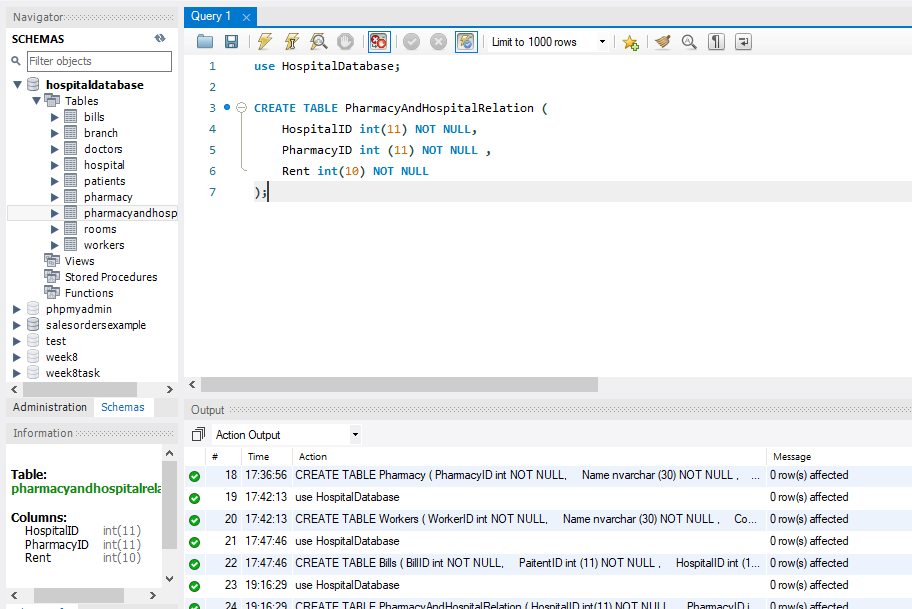
**CREATE TABLE PharmacyAndHospitalRelation (**

**HospitalID int(11) NOT NULL,**

**PharmacyID int (11) NOT NULL ,**

**Rent int(10) NOT NULL**

**);**



1. **Creating Treatments Table**

Command:

**CREATE TABLE Treatments (**

**RegNo int(11) NOT NULL,**

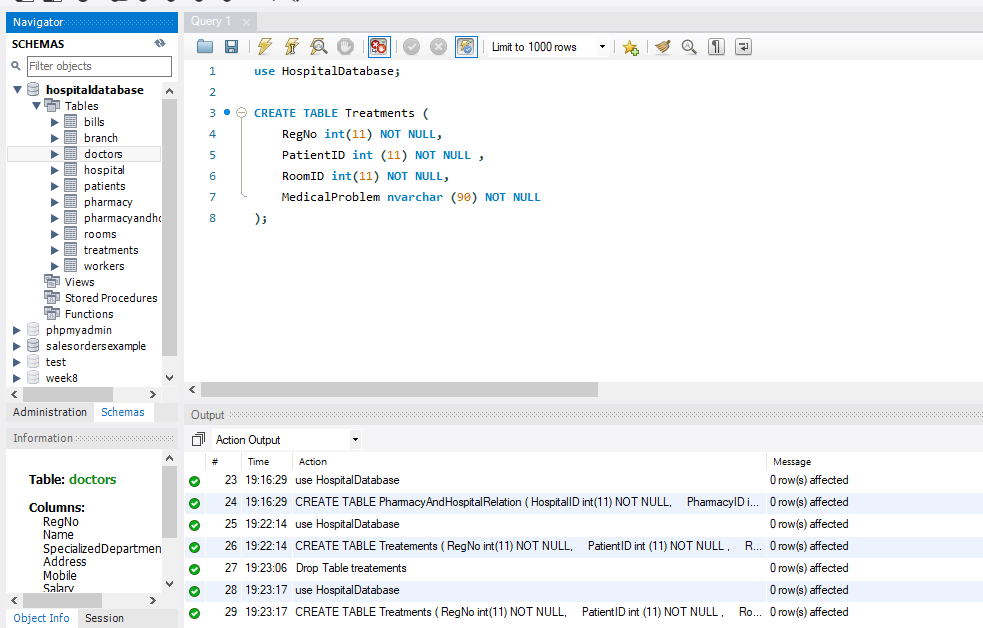
**PatientID int (11) NOT NULL ,**

**RoomID int(11) NOT NULL,**

**HospitalID int(11) NOT NULL,**

**MedicalProblem nvarchar (90) NOT NULL**

**);**



1. **Create WorkingOfDoctors Table**

Command:

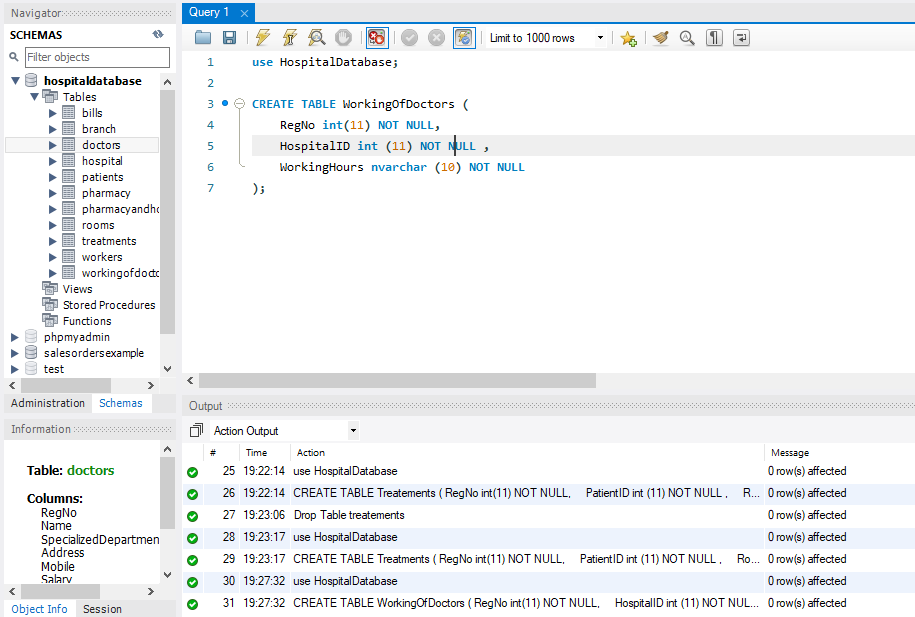
**CREATE TABLE WorkingOfDoctors (**

**RegNo int(11) NOT NULL,**

**HospitalID int (11) NOT NULL ,**

**WorkingHours nvarchar (10) NOT NULL**

**);**



**Primary and Foreign Key Constraints:**

Adding Primary Keys:

* **ALTER TABLE Branch ADD**

**CONSTRAINT Branch\_PK PRIMARY KEY**

**(**

**BranchID**

**) ;**

* **ALTER TABLE Doctors ADD**

**CONSTRAINT Doctors\_PK PRIMARY KEY**

**(**

**RegNo**

**) ;**

* **ALTER TABLE Patients ADD**

**CONSTRAINT Patients\_PK PRIMARY KEY**

**(**

**PatientID**

**) ;**

* **ALTER TABLE Pharmacy ADD**

**CONSTRAINT Pharmacy\_PK PRIMARY KEY**

**(**

**PharmacyID**

**) ;**

* **ALTER TABLE Rooms ADD**

**CONSTRAINT Rooms\_PK PRIMARY KEY**

**(**

**RoomID**

**) ;**

* **ALTER TABLE Workers ADD**

**CONSTRAINT Workers\_PK PRIMARY KEY**

**(**

**WorkerID**

**) ;**

* **ALTER TABLE Bills ADD**

**CONSTRAINT Bills\_PK PRIMARY KEY**

**(**

**BillID**

**) ;**

* **ALTER TABLE Hospital ADD**

**CONSTRAINT Hospital\_PK PRIMARY KEY**

**(**

**HospitalID**

**) ;**

Adding Foreign Keys:

* **ALTER TABLE Hospital ADD**

**CONSTRAINT Hospital\_FK01 FOREIGN KEY**

**(**

**BranchID**

**) REFERENCES Branch (**

**BranchID**

**);**

* **ALTER TABLE pharmacyandhospitalrelation ADD**

**CONSTRAINT PharmacyAndHospital\_FK01 FOREIGN KEY**

**(**

**HospitalID**

**) REFERENCES Hospital (**

**HospitalID**

**),**

**ADD CONSTRAINT PharmacyAndHospital\_FK02 FOREIGN KEY**

**(**

**PharmacyID**

**) REFERENCES Pharmacy (**

**PharmacyID**

**);**

* **ALTER TABLE Treatments ADD**

**CONSTRAINT Treatments\_FK01 FOREIGN KEY**

**(**

**HospitalID**

**) REFERENCES Hospital (**

**HospitalID**

**),**

**ADD CONSTRAINT Treatments\_FK02 FOREIGN KEY**

**(**

**PatientID**

**) REFERENCES Patients (**

**PatientID**

**),**

**ADD CONSTRAINT Treatments\_FK03 FOREIGN KEY**

**(**

**RegNo**

**) REFERENCES Doctors (**

**RegNo**

**),**

**ADD CONSTRAINT Treatments\_FK04 FOREIGN KEY**

**(**

**RoomID**

**) REFERENCES Rooms (**

**RoomID**

**);**

* **ALTER TABLE Bills ADD**

**CONSTRAINT Bills\_FK01 FOREIGN KEY**

**(**

**HospitalID**

**) REFERENCES Hospital (**

**HospitalID**

**),**

**ADD CONSTRAINT Bills\_FK02 FOREIGN KEY**

**(**

**PaitentID**

**) REFERENCES Patients (**

**PatientID**

**),**

**ADD CONSTRAINT Bills\_FK03 FOREIGN KEY**

**(**

**RoomID**

**) REFERENCES Rooms (**

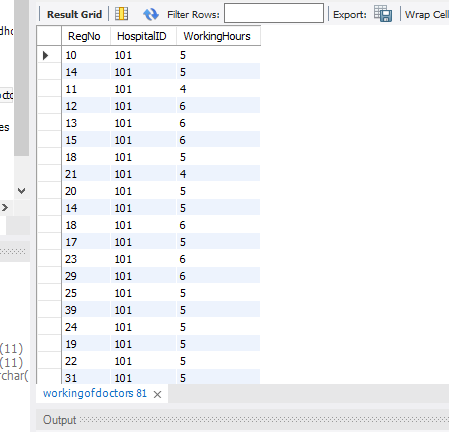
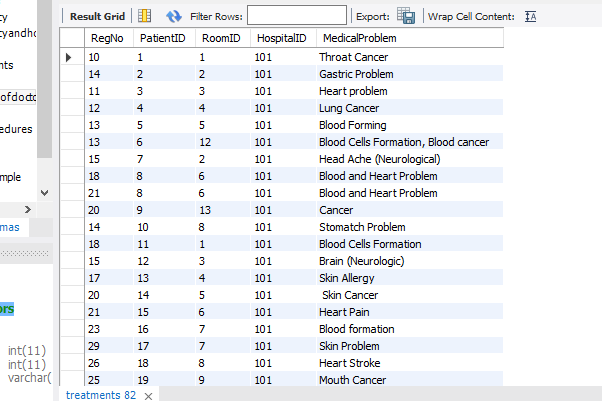
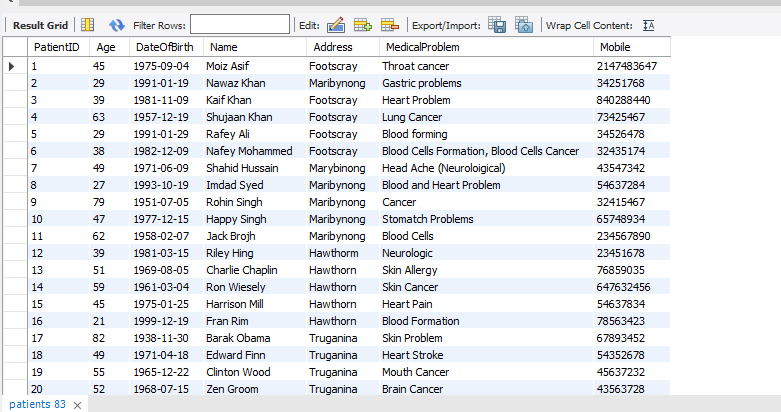
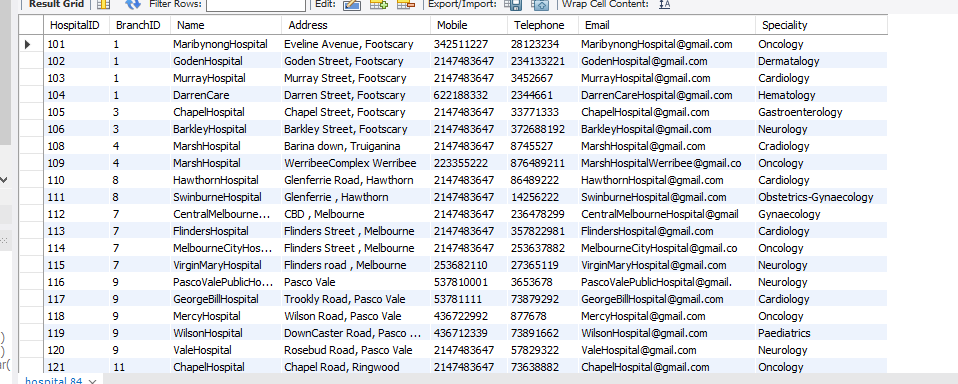
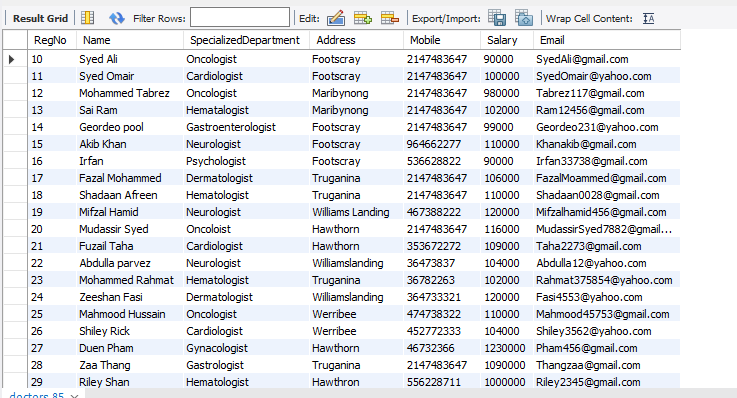
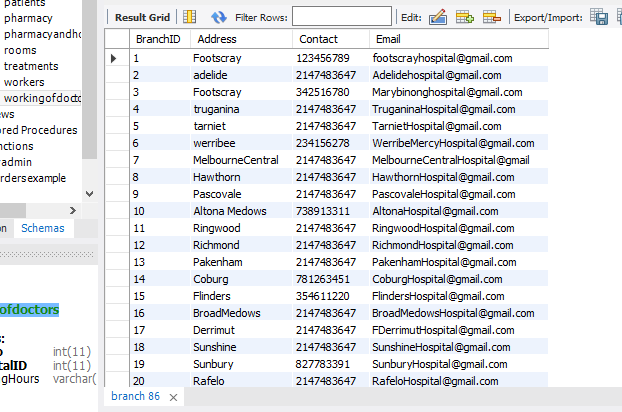
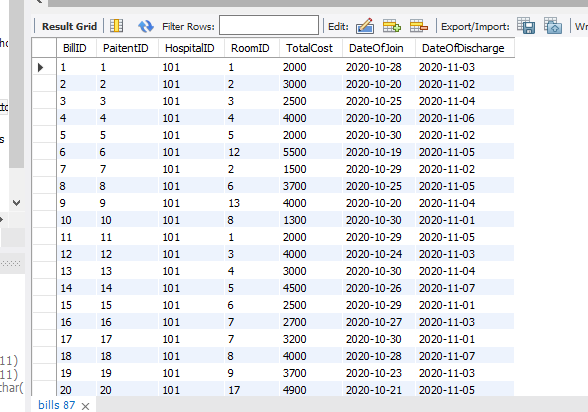
**RoomID**

**);**

**Adding Indexes:**

* **CREATE INDEX BranchID ON Branch(BranchID);**
* **CREATE INDEX RegNo ON Doctors(RegNo);**
* **CREATE INDEX HospitalID ON Hospital(HospitalID);**
* **CREATE INDEX PatientID ON Patients(PatientID);**
* **CREATE INDEX PharmacyID ON Pharmacy(PharmacyID);**
* **CREATE INDEX PharmacyAndHospital ON pharmacyandhospitalrelation(PharmacyID, HospitalID);**
* **CREATE INDEX RoomsID ON Rooms(RoomID);**
* **CREATE INDEX TreatmentsIndex ON Treatments(RegNo, PatientID, RoomID, HospitalID);**
* **CREATE INDEX WorkingDoctorIndex ON workingofdoctors(RegNo, HospitalID);**
* **CREATE INDEX WorkersIndex ON workers(WorkerID);**

**Adding Data in the Database:**

* 
* 
* 
* 
* 
* 
* 

**Commands to show typical use cases of the database:**

1. **Shows the patients details according to their age in descending order**

**SELECT**

**Name, Age, DateOfBirth, Mobile, Address**

**FROM**

**Patients**

**INNER JOIN**

**Bills**

**ON**

**Patients.PatientID = Bills.PaitentID**

**INNER JOIN**

**treatments**

**ON**

**treatments.RoomID = Bills.RoomID**

**GROUP BY**

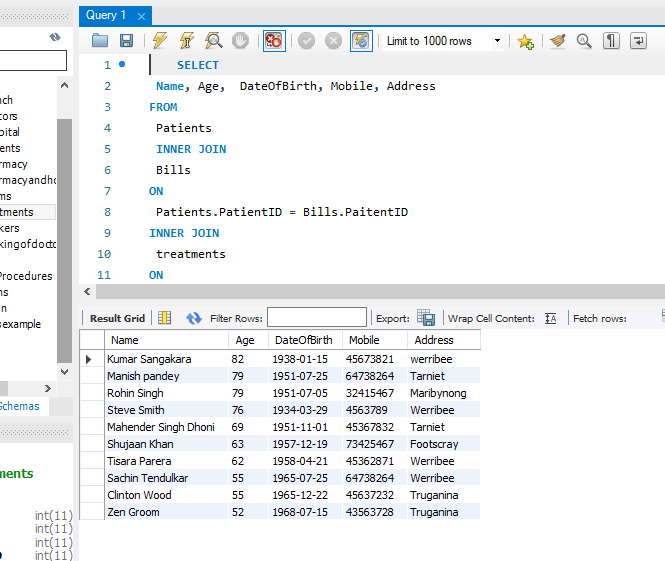
**treatments.RoomID**

**ORDER BY**

**Age**

**DESC**

**LIMIT 10**



1. **Shows the number of patients treated by any doctor**

**SELECT**

**Name,**

**count(PatientID) AS Number\_Of\_Patients\_Treated**

**FROM**

**Doctors**

**INNER JOIN**

**Treatments**

**ON**

**Treatments.RegNo = Doctors.RegNo**

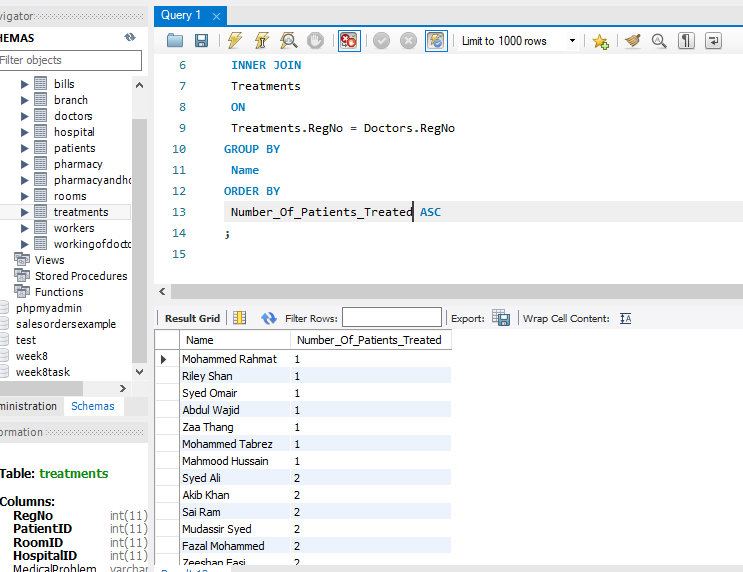
**GROUP BY**

**Name**

**ORDER BY**

**Number\_Of\_Patients\_Treated ASC**

**;**



1. **Shows the details of doctors who treated the patients with medical problem “heart Pain”**

**SELECT**

**Name as Treated\_BY**

**,**

**SpecializedDepartment as Departemnt,**

**Mobile as Contact**

**FROM**

**doctors**

**INNER JOIN**

**treatments**

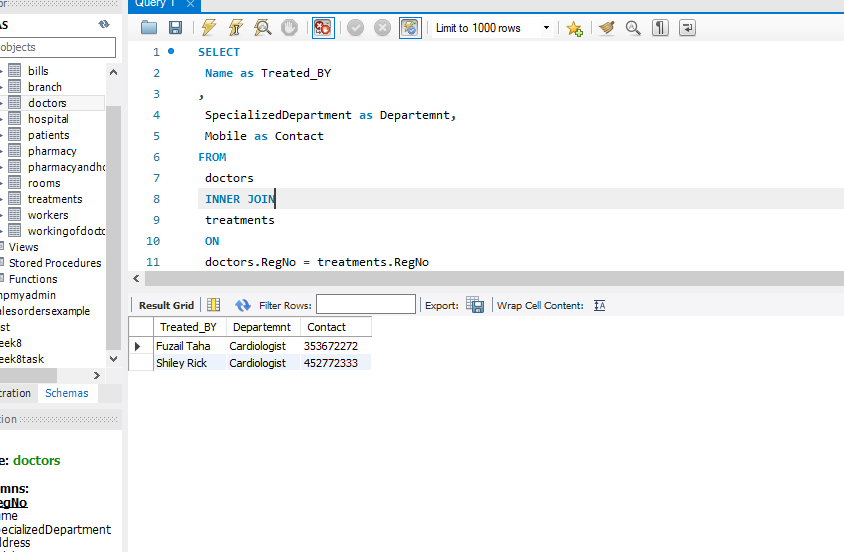
**ON**

**doctors.RegNo = treatments.RegNo**

**WHERE**

**treatments.MedicalProblem = 'Heart Pain'**

**;**



1. **Shows the Maximum Cost of Bill Paid by the Patient**

**SELECT**

**Name ,**

**TotalCost as BillTotal**

**FROM**

**Bills**

**INNER JOIN**

**Rooms**

**ON**

**Rooms.RoomID = Bills.BillID**

**INNER JOIN patients**

**ON**

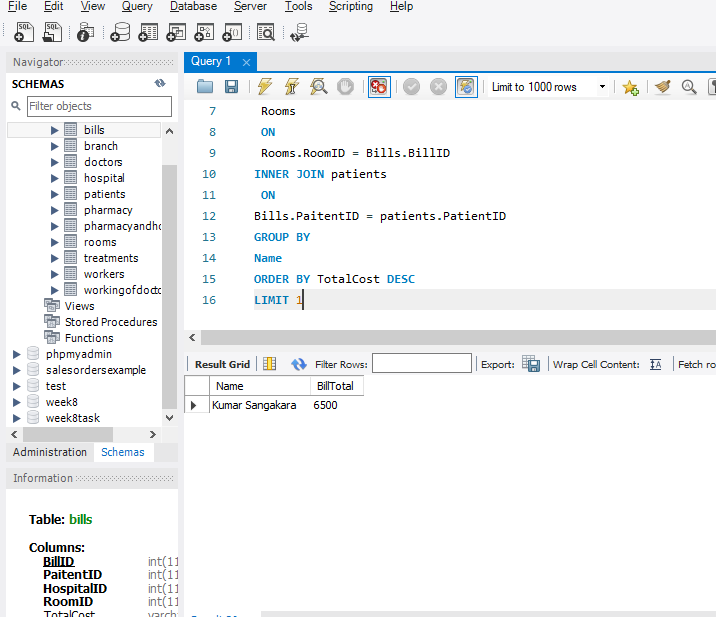
**Bills.PaitentID = patients.PatientID**

**GROUP BY**

**Name**

**ORDER BY TotalCost DESC**

**LIMIT 1**



1. **Count of hospitals which a doctor works for**

**SELECT**

**Name,**

**count(HospitalID) AS Working\_For\_Number\_Of\_Hospitals**

**FROM**

**Doctors**

**INNER JOIN**

**workingofdoctors**

**ON**

**workingofdoctors.RegNo = Doctors.RegNo**

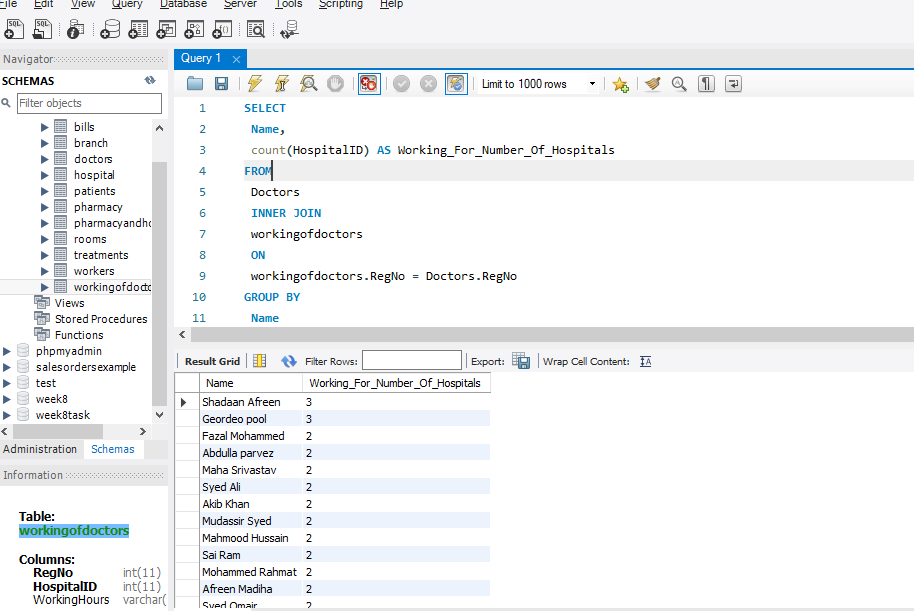
**GROUP BY**

**Name**

**ORDER BY**

**Working\_For\_Number\_Of\_Hospitals DESC**

**;**



**Main Usage of Database:**

This database can be used as a hospital database to store data.In this we can store the details of different branches of hospitals, details of he doctors who work for different branches and hospitals. Also, details of patients, records of the operations and treatments, records of the workers and their shifts. You can easily see the pays and salaries of the workers and doctors. Easily update or change the details of bills if required.