Tutorial 1: Hospital Management System(HMS)

Database Management Systems

SWAYAM: NPTEL-NOC MOOCs Jan-Apr, 2018

Partha Pratim Das
SrijoniMajumdar
Gurunath Reddy M
Himadri B G S Bhuyan

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
ppd@cse.iitkgp.ernet.in

22-Mar-2018

In this tutorial, an end to end implementation of a schema for a specific system is shown along with illustrations of querying using the schema. A system is described by specifying the functional and non functional requirements. Based on this description, the major entities are identified and modeled. Further the relationships are modeled to form the initial schema. The schema is further refined by removing redundancies through normalization. Also based on the query requirements, the schema is remodeled to facilitate querying. Finally an illustration of various queries to extract required information from the system is shown using MYSQL.

The five major workflows are identified as

1. System Specification
2. Design of Initial Schema
3. Schema refinement using functional dependencies and normalization
4. Schema refinement using query requirements
5. Illustration of querying the system using MYSQL

1. SYSTEM SPECIFICATIONS

Description and functionalities of the system

The relations and associations for the management of a hospital are discussed below.

Whenever a new patient is either admitted or comes for outdoor checkup, a unique patient id is generated after storing the name, address and date of birth of the patient. For further visits, the patient uses his unique id.

There are several departments in the Hospital. A department characterized by a unique id, name, floor number and total workers.

The doctors have a unique employee id, and their name, address, contact number, qualifications are stored with this id.

Similarly other workers like nurses, ward boys, ambulance drivers also have a unique employee id. Also each worker is characterized by name, address and type.

Doctors and workers can be associated with multiple departments with different schedules.

Whenever a patient is admitted in the hospital various details are recorded. The patient id, name and address are stored, the department number and name in which the patient is admittedalong with the bed number and room number is also stored. Also for every patient, a senior doctor and junior doctor are appointed. The details of the doctors, like name, id, contact number is recorded. The prescribed medicines are also stored for a patient.

In case of outdoor checkups, the patient id is stored, along with the department number, the employee id of the doctor, the prescription and the date of checkup.

For emergency duty every night, the employee_id of doctor and the nurse is stored along with the date.

HMS should support the following operations / query:

1. Add / Remove /Update records for department and patients
2. Add / Remove / Update worker for doctors and workers
3. Finding the number of doctors working for department ‘Gastroenterology’
4. Find the number of junior doctors in the hospital
5. Find the bed number of the patient with id ‘123’
6. Find the number of patients admitted in the department ‘Oncology’
7. Find the department having the highest worker count
8. List the departments in the second floor.
9. and so on

2. DESIGN OF THE INITIAL SCHEMA

In this section, the major entities will be identified along with the possible attributes. Further the relationships within these entities will be modeled.

Entities and Relationships

Entity: patient

Whenever a new patient is either admitted or comes for outdoor checkup, a unique patient id is generated after storing the name, address and date of birth of the patient. For further visits, the patient uses his unique id.

Entity

- **patient**

Attributes:

- patient_id
- patient_name
- dt_birth
- patient_address

Key: patient_id

Entity: doctors

The doctors have a unique employee id, and their name, address, contact number, qualifications are stored with this id.

Entity

- **doctor**

Attributes:

- employee_id
- name is composite
- address
- qualifications is composite

Key: employee_id

Entity: department

There are several departments in the Hospital. A department is characterized by a unique id, name, floor number and total workers.

Entity

- **department**

Attributes:

- department_num
- department_name
- total_worker_count
- floor name

Key: department_num

Entity: workers

Similarly other workers like nurses, ward boys, ambulance drivers also have a unique employee id. Also each worker is characterized by name, address and type.

Entity

- **workers**

Attributes:

- employee_id
- name
- Address
- type - 'N' for nurses, 'W' for ward boys, 'AD' for ambulance drivers

Key: employee_id

Relationship: admitted

Whenever a patient is admitted in the hospital various details are recorded. The patient id, name and address are stored, the department number and name in which the patient is admitted along with the bed number and room number is also stored. Also for every patient, a senior doctor and junior doctor are appointed. The details of the doctors, like name, id, contact number is recorded. The prescribed medicines are also stored for a patient.

Relationship

- **admitted**

Involved Entities:

- **patient**
- **department**
- **doctors**

Attributes from entities:

- patient_id
- patient_name
- patient_address
- department_num
- department_name
- senior_doctor_name
- senior_doctor_employee_id
- senior_doctor_prescription
- senior_doctor_contact_number
- junior_doctor_name
- junior_doctor_employee_id
- junior_doctor_prescription
- junior_doctor_contact_number

Relationship Attributes:

- date_admission
- date_discharge

Key: patient_id, department_num, date_admission

Relationship: outdoor

In case of outdoor checkups, the patient id is stored, along with the department number, the employee id of the doctor, the prescription and the date of checkup.

Relationship

- **admitted**

Involved Entities:

- **patient**
- **department**
- **doctors**

Attributes from entities:

- patient_id
- department_num
- doctor_id

Relationship Attributes:

- date_checkup
- prescription

Key: patient_id, department_num, date_checkup

Relationship:works_for

Doctors and workers can be associated with multiple departments with different schedules.

Relationship

- **works_for**

Involved Entities:

- **workers**
- **department**
- **doctors**

Attributes from entities:

- employee_id
- department_num

Relationship Attributes:

- schedule

Key: employee_id ,department_num

Relationship:emergency

For emergency duty every night, the employee_id of doctor and the nurse is stored along with the date.

Relationship

- **emergency**

Involved Entities:

- **workers AS nurses**
- **doctors**

Attributes from entities:

- doctor_id (employee_id of doctors)
- nurse_id (employee_id of nurses)

Relationship Attributes:

- date

Key: date

Initial Schema

- **admitted**(patient_id, patient_name, patient_address, department_num, department_name, senior_doctor_name, senior_doctor_employee_id, senior_doctor_prescription, senior_doctor_contact_number, junior_doctor_name, junior _doctor_employee_id, junior_doctor_prescription, junior_doctor_contact_number, date_admission, date_discharge)
- **patient (**patient_id, patient_name, dt_birth, patient_address)
- **doctors (**employee_id, name, address, qualifications)
- **department (**department_num, department_name, total_worker_count, floor)
- **workers**(employee_id, name, address, type)
- **works_for** (department_num, employee_id, schedule)
- **outdoor**(patient_id, department_num, doctor_id, prescription, date_checkup)
- **emergency**(doctor_id, nurse_id, date)

3. SCHEMA REFINEMENT USING DEPENDENCIES AND NORMALISATION

In this section, initial schema will be refined by analyzing functional dependencies and normalizing the schemas to remove the redundancies.

Analyzing Functional Dependencies and Schema Refinement

- **admitted**(patient_id, patient_name, patient_address, department_num, department_name, senior_doctor_name, senior_doctor_employee_id, senior_doctor_prescription, senior_doctor_contact_number, junior_doctor_name, junior _doctor_employee_id, junior_doctor_prescription, junior_doctor_contact_number, date_admission, date_discharge)

  Functional Dependencies

  - patient_id→patient_name, patient_addressviolates 2NF
  - department_num→department_nameviolates 2NF
  - patient_id, department_num, date_admission→date_discharge, senior_doctor_id, senior_doctor_prescription, junior_doctor_id, junior_doctor_prescription
  - senior_doctor_id→senior_doctor_name, senior_doctor_contact_numberviolates 3NF
  - junior_doctor_id→junior _doctor_name, junior _doctor_contact_numberviolates 3NF

  RESULTS IN

  - **admitted**(patient_id, department_num,date_admission, date_discharge, senior_doctor_id, senior_doctor_prescription, junior_doctor_id, junior_doctor_prescription)
  - **patient**(patient_id, patient_name, patient_address) - already there
  - **department(**department_num, department_name) -already there
  - **senior_doctor(**senior_doctor_id , senior_doctor_name, senior_doctor_contact_number)
  - **junior_doctor (**junior_doctor_id , junior _doctor_name, junior _doctor_contact_number)

  Junior doctor and senior doctor can be merged into doctors relation with the introduction of new column 'grade'

  WHICH FURTHER RESULTS TO

Relationship:admitted

- **admitted**(patient_id, department_num ,date_admission, date_discharge, doctor_id, doctor_grade, prescription)- 1NF, 2NF, 3NF, BCNF

- Involved entities: patient(attribute: patient_id); department(attributes: department_num); doctor(doctor_id)
- Attributes: date_admission, date_discharge,prescription

Changes to **Entity Set: doctors**

- **doctors (**employee_id, name, address, qualifications, grade) 1NF, 2NF, 3NF, BCNF
- **grade can be junior or senior**

➢ **patient (**patient_id, patient_name, dt_birth, patient_address)

Functional Dependencies

- patient_id→patient_name, dt_birth, patient_address - 1NF, 2NF, 3NF, BCNF

➢ **doctors (**employee_id, name, address, contact_number , qualifications, grade)

Functional Dependencies

- employee_id→ name, address, contact_number , qualifications, grade - 1NF, 2NF, 3NF, BCNF

➢ **department (**department_num, department_name, total_worker_count, floor)

Functional Dependencies

- department_num→department_name, total_worker_count, floor - 1NF, 2NF, 3NF, BCNF

➢ **workers**(employee_id, name, address, type)

Functional Dependencies

- employee_id→name, address, type - 1NF, 2NF, 3NF, BCNF

➢ **works_for** (department_num, employee_id, schedule)

Functional Dependencies

- department_num, employee_id→schedule - 1NF, 2NF, 3NF, BCNF

➢ **outdoor**(patient_id, department_num, doctor_id, prescription, date_checkup)

Functional Dependencies

- patient_id, department_num, date_checkup→doctor_id, prescription - 1NF, 2NF, 3NF, BCNF

- **emergency**(doctor_id, nurse_id, date)

Functional Dependencies

- date→doctor_id, nurse_id- 1NF, 2NF, 3NF, BCNF

We can keep a check constraint to enter the employee id for only type ='N' from workers table

OR

We can create a view on workers named as nurse, which will only select the records from worker where type = 'N' i.e nurses

Entity (View)

- **nurses**

Attributes:

- nurse_id (employee_id from workers table where type - 'N' for nurses)
- name
- Address

Key: nurse_id

Refined Schema

- **admitted**(patient_id, department_num ,date_admission, date_discharge, doctor_id, doctor_grade, prescription)
- **patient (**patient_id, patient_name, dt_birth, patient_address)
- **doctors (**employee_id, name, address, contact_number , qualifications, grade)
- **department (**department_num, department_name, total_worker_count, floor)
- **workers**(employee_id, name, address, type)
- **nurses**(nurse_id, name, address)
- **works_for** (department_num, employee_id, schedule)
- **outdoor**(patient_id, department_num, doctor_id, prescription, date_checkup)
- **emergency**(doctor_id, nurse_id, date)

4. SCHEMA REFINEMENT USING QUERY REQUIREMENTS OF THE SYSTEM

Based on query requirements, the entities will be remodeled in this section.

Analyzing Query Requirements and Schema Refinement

Query: Finding the number of doctors working for department 'Gastroenterology'

Will require querying from tables works_for with doctors and workers

So if we can introduce a new columns in the works_for relation, where the type could specify whether the employee_id belongs to a doctor, or a nurse, or a ward boy and so on..

Relationship: works_for

Relationship

- **works_for**

Involved Entities:

- **workers**
- **department**
- **doctors**

Attributes from entities:

- employee_id
- department_num

Relationship Attributes:

- schedule
- employee_type 'D' for doctors, N' for nurses, 'W' for ward boys, 'AD' for ambulance drivers

Key: employee_id ,department_num

Refined Schema

- **admitted**(patient_id, department_num ,date_admission, date_discharge, doctor_id, doctor_grade, prescription): FK: patient_id, department_num
- **patient (**patient_id, patient_name, dt_birth, patient_address)
- **doctors (**employee_id, name, address, contact_number , qualifications, grade)
- **department (**department_num, department_name, total_worker_count, floor)
- **workers**(employee_id, name, address, type)
- **nurses**(nurse_id, name, address)
- **works_for** (department_num, employee_id, employee_type, schedule) FK: employee_id, department_num
- **outdoor**(patient_id, department_num, date_checkup ,doctor_id, prescription) FK: patient_id, department_num
- **emergency**(date, doctor_id, nurse_id)FK: (doctor_id) employee_id, nurse_id

The underlined attributes are the primary keys of the relation. The foreign keys are mentioned alongside the relation as FK.

5. ILLUSTRATION OF QUERYING THE SYSTEM USING MYSQL

In this section, various instances of querying the system using MYSQL will be shown. The results are also shown.

(Note: When you will be trying the queries from this section, please don't copy the queries from this document and paste it directly in the mysql prompt, as it might not run, due to insertion of unwanted hidden special characters while copying. Please refer the queries, but type them in the prompt)

MySQL

Install SQL in Ubuntu machine by issuing the below command in terminal (**$ i**s shell prompt)

$ sudo apt-get install mysql-server

You could able to see the following prompts

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 4

Server version: 5.7.21-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Note:

If prompts for a password, give your own password

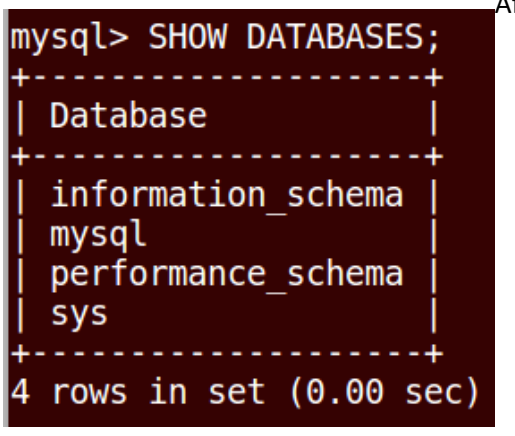
Once MySQL is installed, access the MySQL shell by

$ mysql -u root -p

To check the available databases issue (**$** will turn to **mysql> prompt**)

mysql> SHOW DATABASES;

After the above command, your screen will looks like this



To create a new database use

CREATE DATABASE database_name;

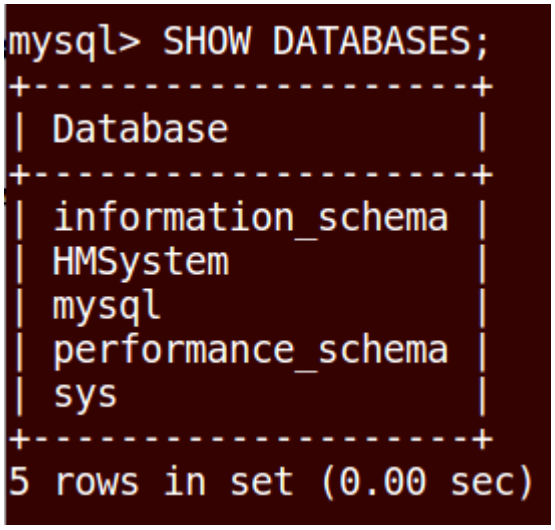
For our **H**ospital **M**anagement **System (HMSystem)**, we create our database by

mysql>CREATE DATABASE HMSystem;

Now,

mysql>SHOW DATABASES;

will results in



From the above terminal output, we can see that our **HMSystem** is added to the Database

Accessing our newly created database system

To create the tables in the newly created database **HMSystem**, we need to open it.

mysql> USE HMSystem;

Your command prompt should say **Database changed**

To check the tables in **HMSystem** use

mysql> SHOW tables;

Your command prompt should say **Empty set (0.00 sec)** because we have not yet added any tables.

Now, let add tables into **HMSystem:**

We can add the **patient** tables as follows

mysql> CREATE TABLE patient (

patient_idINT UNSIGNED NOT NULL,

patient_nameVARCHAR(50) NOT NULL,

dt_birth DATE NOT NULL,

patient_addressVARCHAR(100) NOT NULL

PRIMARY KEY (patient_id));

To see the created table

mysql> SHOW tables

results in

```
+--------------------+
| Tables_in_HMSystem |
+--------------------+
| patient            |
+--------------------+
1 row in set (0.00 sec)
```

To see the fields, types and constraints

mysql> DESCRIBE patient;

```
mysql> DESCRIBE patient;
+-----------------+------------------+------+-----+---------+-------+
| Field           | Type             | Null | Key | Default | Extra |
+-----------------+------------------+------+-----+---------+-------+
| patient_id      | int(10) unsigned | NO   | PRI | NULL    |       |
| patient_name    | varchar(50)      | NO   |     | NULL    |       |
| dt_birth        | date             | NO   |     | NULL    |       |
| patient_address | varchar(100)     | NO   |     | NULL    |       |
+-----------------+------------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Similarly, we can add the remaining tables as given below

```
mysql>CREATE TABLE doctors (
                    employee_id INT UNSIGNED NOT NULL,
                    name VARCHAR(50) NOT NULL,
                    address VARCHAR(100) NOT NULL,
                    contact_numberVARCHAR(50),
                    qualifications VARCHAR(10),
                    grade VARCHAR(10),
                    PRIMARY KEY (employee_id) );


mysql> CREATE TABLE workers (
                    employee_id INT UNSIGNED NOT NULL,
                    name VARCHAR(50) NOT NULL,
                    address VARCHAR(100),
                     type VARCHAR(5),
                     PRIMARY KEY (employee_id) );

mysql> CREATE TABLE department (
                    department_num INT UNSIGNED NOT NULL,
                    department_nameVARCHAR(50) NOT NULL,
                    total_worker_count INT,
                    floor INT,
                    PRIMARY KEY (department_num) );
```

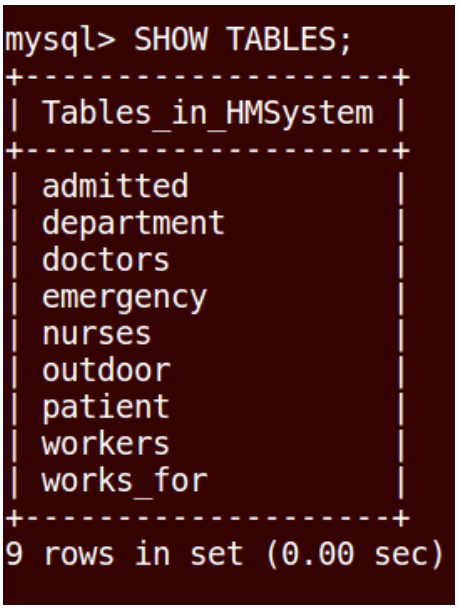
```
mysql> CREATE TABLE nurses (
                    nurse_id INT UNSIGNED NOT NULL,
                     name VARCHAR(50) NOT NULL,
                     address VARCHAR(100),
                     PRIMARY KEY (nurse_id) );


mysql>  CREATE TABLE works_for (
department_num INT UNSIGNED,
          employee_id INT UNSIGNED NOT NULL,
          employee_typeVARCHAR(50) NOT NULL,
            schedule DATE,
            PRIMARY KEY (employee_id, department_num),
            FOREIGN KEY (employee_id) REFERENCES doctors(employee_id) ON DELETE CASCADE,
            FOREIGN KEY (employee_id) REFERENCES workers(employee_id) ON DELETE CASCADE,
FOREIGN KEY (employee_id) REFERENCES doctors(employee_id) ON DELETE CASCADE,
FOREIGN KEY (department_num) REFERENCES department(department_num) ON DELETE
                                                                                CASCADE  );


mysql>CREATE TABLE emergency (
          date DATE NOT NULL,
          doctor_id INT UNSIGNED,
          nurse_id INT UNSIGNED,
          PRIMARY KEY (date),
          FOREIGN KEY (doctor_id) REFERENCES doctors(employee_id) ON DELETE CASCADE,
          FOREIGN KEY (nurse_id) REFERENCES nurses(nurse_id) ON DELETE CASCADE );
```

```
mysql> CREATE TABLE admitted (
        patient_id INT UNSIGNED NOT NULL,
        department_num INT UNSIGNED NOT NULL,
        date_admission DATE NOT NULL,
        date_discharge DATE,
        doctor_id INT UNSIGNED,
        doctor_gradeVARCHAR(10),
        prescription VARCHAR(200),
        PRIMARY KEY (patient_id, department_num, date_admission),
        FOREIGN KEY (patient_id) REFERENCES patient(patient_id) ON DELETE CASCADE,
        FOREIGN KEY (department_num) REFERENCES department(department_num) ON DELETE
                                                                        CASCADE );
```

After adding all the tables, SHOW TABLES should display the tables available in HMSystemas shown below



```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_HMSystem |
+--------------------+
| admitted           |
| department         |
| doctors            |
| emergency          |
| nurses             |
| outdoor            |
| patient            |
| workers            |
| works_for          |
+--------------------+
9 rows in set (0.00 sec)
```

Adding Tuples into Patient Relation

mysql>**INSERT INTO patient (patient_id, patient_name, dt_birth, patient_address) VALUES (1200, "Ananya Mukherjee", '1994-12-27', "Rajarhat");**

mysql>**INSERT INTO patient (patient_id, patient_name, dt_birth, patient_address) VALUES (1215, "Srikanth Reddy", '1989-09-09', "Saltlake");**

mysql> **INSERT INTO patient (patient_id, patient_name, dt_birth, patient_address) VALUES (1252, "RamyaGovindan", '1982-01-13', "Santoshpur");**

mysql>**INSERT INTO patient (patient_id, patient_name, dt_birth, patient_address) VALUES (1234, "Rita Chatterjee", '2012-07-18', "Calcutta greens");**

mysql>**INSERT INTO patient (patient_id, patient_name, dt_birth, patient_address) VALUES(1256, "Sanjoy Sen", '1980-08-11', "Golf greens");**

We can check the added tuples in the **patient** relation by

mysql>**SELECT * FROM patients;**

```
mysql> SELECT * FROM patient;
+------------+------------------+------------+-----------------+
| patient_id | patient_name     | dt_birth   | patient_address |
+------------+------------------+------------+-----------------+
|       1200 | Ananya Mukherjee | 1994-12-27 | Rajarhat        |
|       1215 | Srikanth Reddy   | 1989-09-09 | Saltlake        |
|       1234 | Rita Chatterjee  | 2012-07-18 | Calcutta greens |
|       1252 | Ramya Govindan   | 1982-01-13 | Santoshpur      |
|       1256 | Sanjoy Sen       | 1980-08-11 | Golf greens     |
+------------+------------------+------------+-----------------+
5 rows in set (0.00 sec)
```

Similarly, we can add the tuples into the other relations as follows

For **doctor**:

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (332936, "SrishtiSanyal", "Saltlake", 19434943401, "MD", "Senior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (332989, "Sireesha Sen", "Bangur", 19434943423, "MD", "Junior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (332978, "Damini Sen", "Gariahat", 19434943402, "MBBS", "Junior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (334136, "Nilanjan Roy", "Jodhpur Park", 19434943403, "MD", "Senior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (324571, "Ravi Verma", "Ballygunge", 19434943404, "MBBS", "Junior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (341235, "RomilaRanganath", "Ruby", 19434943441, "MD", "Senior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (340005, "Pradeep Raj", "Tumkur", 19430003441, "MD", "Senior");

INSERT INTO doctors (employee_id, name, address, contact_number, qualifications, grade) values (311115, "Gurunath Reddy", "Jayanagar", 19430001111, "MD", "Senior");

```
mysql> SELECT * FROM doctors;
+-------------+------------------+--------------+----------------+----------------+--------+
| employee_id | name             | address      | contact_number | qualifications | grade  |
+-------------+------------------+--------------+----------------+----------------+--------+
|      311115 | Gurunath Reddy   | Jayanagar    | 19430001111    | MD             | Senior |
|      324571 | Ravi Verma       | Ballygunge   | 19434943404    | MBBS           | Junior |
|      332936 | Srishti Sanyal   | Saltlake     | 19434943401    | MD             | Senior |
|      332978 | Damini Sen       | Gariahat     | 19434943402    | MBBS           | Junior |
|      332989 | Sireesha Sen     | Bangur       | 19434943423    | MD             | Junior |
|      334136 | Nilanjan Roy     | Jodhpur Park | 19434943403    | MD             | Senior |
|      340005 | Pradeep Raj      | Tumkur       | 19430003441    | MD             | Senior |
|      341235 | Romila Ranganath | Ruby         | 19434943441    | MD             | Senior |
+-------------+------------------+--------------+----------------+----------------+--------+
8 rows in set (0.00 sec)
```

For**department** :

INSERT INTO department(department_num, department_name, total_worker_count, floor) VALUES (1, "Gynaecology", 3, 2);

INSERT INTO department(department_num, department_name, total_worker_count, floor) VALUES (2, "General Surgery", 4,2);

INSERT INTO department(department_num, department_name, total_worker_count, floor) VALUES (3, "General Medicine", 2,1);

INSERT INTO department(department_num, department_name, total_worker_count, floor) VALUES (4, "Paediatrics", 2, 1);

INSERT INTO department(department_num, department_name, total_worker_count, floor) VALUES (5, "Gastroenterology", 1, 1);

```
mysql> SELECT * FROM department;
+-----------------+-------------------+---------------------+-------+
| department_num | department_name   | total_worker_count | floor |
+-----------------+-------------------+---------------------+-------+
|              1 | Gynaecology       |                  3 |     2 |
|              2 | General Surgery   |                  4 |     2 |
|              3 | General Medicine  |                  2 |     1 |
|              4 | Paediatrics       |                  2 |     1 |
|              5 | Gastroenterology  |                  1 |     1 |
+-----------------+-------------------+---------------------+-------+
5 rows in set (0.00 sec)
```

For **Workers**:

INSERT INTO workers (employee_id, name, address, type) VALUES (451234, "Ram Roy", "Park Circus Road 1", "AD");

INSERT INTO workers (employee_id, name, address, type) VALUES (451212, "Rahul Sen", "Park Circus Road 1", "W");

INSERT INTO workers (employee_id, name, address, type) VALUES (451244, "Sourya Das", "Park Circus Road 2", "W");

INSERT INTO workers (employee_id, name, address, type) VALUES (461789, "Sai Ajay", "Park Circus Road 2", "W");

INSERT INTO workers (employee_id, name, address, type) VALUES (411134, "Rohan Singh", "Park Circus Road 3", "W");

INSERT INTO workers (employee_id, name, address, type) VALUES (422234, "Rahul Mukherjee", "Park Circus Road 3", "W");

INSERT INTO workers(nurse_id, name, address, type) VALUES ('101001', "Lakshmi", "Yeshwantpur", "N");

INSERT INTO workers(employee_id, name, address, type) VALUES ('101002', "Bindu", "Electronic City", "N");

INSERT INTO workers(employee_id, name, address, type) VALUES ('101003', "Rashmi", "ISRO Satellite City", "N");

INSERT INTO workers(employee_id, name, address, type) VALUES ('101004', "Mahima", "Saint Jhones", "N");

```
mysql> select * from workers;
+-------------+-----------------+---------------------+------+
| employee_id | name            | address             | type |
+-------------+-----------------+---------------------+------+
|      101001 | Lakshmi         | Yeshwantpur         | N    |
|      101002 | Bindu           | Electronic City     | N    |
|      101003 | Rashmi          | ISRO Satellite City | N    |
|      101004 | Mahima          | Saint Jhones        | N    |
|      411134 | Rohan Singh     | Park Circus Road 3  | N    |
|      422234 | Rahul Mukherjee | Park Circus Road 3  | W    |
|      451212 | Rahul Sen       | Park Circus Road 1  | W    |
|      451234 | Ram Roy         | Park Circus Road 1  | AD   |
|      451244 | Sourya Das      | Park Circus Road 2  | W    |
|      461789 | Sai Ajay        | Park Circus Road 2  | W    |
+-------------+-----------------+---------------------+------+
10 rows in set (0.00 sec)
```

For **emergency** relationship

INSERT INTO emergency (date, doctor_id, nurse_id) VALUES ('2017-10-21', 324571, 101001);

INSERT INTO emergency (date, doctor_id, nurse_id) VALUES ('2016-10-21', 341235, 101003);

INSERT INTO emergency (date, doctor_id, nurse_id) VALUES ('2017-11-25', 332989, 101004);

```
mysql> select * from emergency;
+------------+-----------+----------+
| date       | doctor_id | nurse_id |
+------------+-----------+----------+
| 2016-10-21 |    341235 |   101003 |
| 2017-10-21 |    324571 |   101001 |
| 2017-11-25 |    332989 |   101004 |
+------------+-----------+----------+
3 rows in set (0.00 sec)
```

WHERE clause usage:

To select all workers who are all wardboys in the hospital.

```
mysql> SELECT * from workers WHERE type="W";
+-------------+-----------------+--------------------+------+
| employee_id | name            | address            | type |
+-------------+-----------------+--------------------+------+
|      411134 | Rohan Singh     | Park Circus Road 3 | W    |
|      422234 | Rahul Mukherjee | Park Circus Road 3 | W    |
|      451212 | Rahul Sen       | Park Circus Road 1 | W    |
|      451244 | Sourya Das      | Park Circus Road 2 | W    |
|      461789 | Sai Ajay        | Park Circus Road 2 | W    |
+-------------+-----------------+--------------------+------+
5 rows in set (0.00 sec)
```

For **nurses (optional as a relation):**

INSERT INTO nurses(nurse_id, name, address) VALUES ('101001', "Lakshmi", "Yeshwantpur");

INSERT INTO nurses(nurse_id, name, address) VALUES ('101002', "Bindu", "Electronic City");

INSERT INTO nurses(nurse_id, name, address) VALUES ('101003', "Rashmi", "ISRO Satellite City");

INSERT INTO nurses(nurse_id, name, address) VALUES ('101004', "Mahima", "Saint Jhones");

```
mysql> select * from nurses;
+----------+---------+---------------------+
| nurse_id | name    | address             |
+----------+---------+---------------------+
|   101001 | Lakshmi | Yeshwantpur         |
|   101002 | Bindu   | Electronic City     |
|   101003 | Rashmi  | ISRO Satellite City |
|   101004 | Mahima  | Saint Jhones        |
+----------+---------+---------------------+
4 rows in set (0.00 sec)
```

Updating tuple using **UPDATE** clause

To update the type of worker to “N” (nurse) with employee id 411134

UPDATE workers

SET type = "N"

WHERE employee_id = 411134;

```
mysql> UPDATE workers
    -> SET type = "N"
    -> WHERE employee_id = 411134;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * from workers;
+-------------+-----------------+--------------------+------+
| employee_id | name            | address            | type |
+-------------+-----------------+--------------------+------+
|      411134 | Rohan Singh     | Park Circus Road 3 | N    |
|      422234 | Rahul Mukherjee | Park Circus Road 3 | W    |
|      451212 | Rahul Sen       | Park Circus Road 1 | W    |
|      451234 | Ram Roy         | Park Circus Road 1 | AD   |
|      451244 | Sourya Das      | Park Circus Road 2 | W    |
|      461789 | Sai Ajay        | Park Circus Road 2 | W    |
+-------------+-----------------+--------------------+------+
6 rows in set (0.00 sec)
```

Using **LIKE** clause:

To display the names of the department which starts with "**Gen**"

SELECT * from department

WHERE department_name LIKE 'Gen%';

```
mysql> SELECT * from department  WHERE department_name LIKE 'Gen%';
+----------------+-----------------+--------------------+-------+
| department_num | department_name  | total_worker_count | floor |
+----------------+-----------------+--------------------+-------+
|              2 | General Surgery  |                  4 |     2 |
|              3 | General Medicine |                  2 |     1 |
+----------------+-----------------+--------------------+-------+
2 rows in set (0.00 sec)
```

Using **ORDER BY** clause:

To display the department names in the ascending oder in the department realtion

SELECT * from department

ORDER BY department_name ASC;

```
mysql> SELECT * from department
    -> ORDER BY department_name ASC;
+----------------+-----------------+--------------------+-------+
| department_num | department_name  | total_worker_count | floor |
+----------------+-----------------+--------------------+-------+
|              5 | Gastroenterology |                  1 |     1 |
|              3 | General Medicine |                  2 |     1 |
|              2 | General Surgery  |                  4 |     2 |
|              1 | Gynaecology      |                  3 |     2 |
|              4 | Paediatrics      |                  2 |     1 |
+----------------+-----------------+--------------------+-------+
5 rows in set (0.00 sec)
```

Using **Joins**

To obtain the names of nurses who have attended the emergency.

SELECT name

FROM emergency, workers

WHERE employee_id = nurse_id;

```
mysql> SELECT name
    -> FROM emergency, workers
    -> WHERE employee_id = nurse_id;
+---------+
| name    |
+---------+
| Lakshmi |
| Rashmi  |
| Mahima  |
+---------+
3 rows in set (0.00 sec)
```

Using **GROUP BY** clause:

To count the number of nurses, wardboys, ambulance drivers

SELECT COUNT(name), type

FROM workers

GROUP BY type;

```
mysql> select count(name), type
    -> from workers
    -> group by type;
+-------------+------+
| count(name) | type |
+-------------+------+
|           1 | AD   |
|           5 | N    |
|           4 | W    |
+-------------+------+
3 rows in set (0.00 sec)
```

The **UNION** clause

To make union of **doctor** names and **worker** name

SELECT name FROM doctors

UNION

SELECT name FROM workers;

```
mysql> select name from doctors
    -> union
    -> select name from workers;
+------------------+
| name             |
+------------------+
| Gurunath Reddy   |
| Ravi Verma       |
| Srishti Sanyal   |
| Damini Sen       |
| Sireesha Sen     |
| Nilanjan Roy     |
| Pradeep Raj      |
| Romila Ranganath |
| Lakshmi          |
| Bindu            |
| Rashmi           |
| Mahima           |
| Rohan Singh      |
| Rahul Mukherjee  |
| Rahul Sen        |
| Ram Roy          |
| Sourya Das       |
| Sai Ajay         |
+------------------+
18 rows in set (0.00 sec)
```

Using **VIEW** clause:

To create view nurse on workers from table based on type = “N”

create view nurse as

select employee_id, name, address, type

from workers

where type = "N";

To query the nurse view

select * from nurse;

```
mysql> create view nurse as
    -> select employee_id, name, address, type
    -> from workers
    -> where type = "N";
Query OK, 0 rows affected (0.05 sec)

mysql> select * from nurse;
+-------------+-------------+----------------------+------+
| employee_id | name        | address              | type |
+-------------+-------------+----------------------+------+
|      101001 | Lakshmi     | Yeshwantpur          | N    |
|      101002 | Bindu       | Electronic City      | N    |
|      101003 | Rashmi      | ISRO Satellite City  | N    |
|      101004 | Mahima      | Saint Jhones         | N    |
|      411134 | Rohan Singh | Park Circus Road 3   | N    |
+-------------+-------------+----------------------+------+
5 rows in set (0.00 sec)
```

Using **WHERE EXISTS** clause

SELECT name FROM doctors

WHERE EXISTS (SELECT name FROM doctors, emergency WHERE emergency.doctor_id = doctors.employee_id);

```
mysql> select name
    -> from doctors
    -> where exists (select name from doctors, emergency where emergency.doctor_id = doctors.employee_id);
+-----------------+
| name            |
+-----------------+
| Gurunath Reddy  |
| Ravi Verma      |
| Srishti Sanyal  |
| Damini Sen      |
| Sireesha Sen    |
| Nilanjan Roy    |
| Pradeep Raj     |
| Romila Ranganath |
+-----------------+
8 rows in set (0.00 sec)
```