Database Management Systems Lab Project

Hospital Management System

Name: Abdullah Khan

Roll No.: 111801001

Team No.: 11

Team Members:

Vishesh Munjal

Sahil J Chaudhari

Harsh Parihar

CONTENTS

CONTRIBUTIONS

- **Abdullah Khan: -** Built entity sets in ERD and assigned type, Database Schema and creating tables using SQL script, Updated SQL file, Database Creation file, Brainstorming database views
- **Sahil J Chaudhari: -** Built relational sets and relations, designing Schema Diagram, Updated SQL file, Database Creation file, Creation of views
- **Vishesh Munjal: -** Built entity sets in ERD and attributes, designing Schema Diagram, Updated SQL file, Database Creation file, Brainstorming database views
- **Harsh Parihar: -** creating tables based on database schema, Data Insertion, Brainstorming database views

1 REQUIREMENT SPECIFICATION

1.1 OBJECTIVE

Through the development of this management system, we aim to efficiently store all the data that might become useful in some form of survey or provide insight in the workflow of the present structure of the hospital along with trying to automize various task that may be requiring human input currently. The main objective of this project is thus to digitize all the interactions that may happen in a general hospital and to store that data in an efficient form so as to allow for efficient operations on this data in other downstream tasks.

1.2 USER EXPERIENCE

At the core of our database, we have 2 types entities representing actual physical people whose interaction we intend to model and optimize. These include Patient (**Inpatient or Outpatient**) and Employees (**Doctors, Nurses or Receptionists**), which predominantly interact with one another as well as some abstract entities in the database such as **Rooms, Bills, Facilities, Department, Records** while there also exists another class of entities which interact with these abstract ones to complete the behavioral model of our hospital such as **Insurance** and **Medicine.**

The User experience for any general patient involves the requirement of necessary data enlisted in the entity set description for the purpose of identification and verification. Whenever any person requires the service hospital, they will either be a recurring patient with existing record or a new patient whose record can be inserted. They will give their personal information such as name, age, contact, address, etc. to be inserted into the system. Depending upon the need for admittance, patients are classified into two groups; Inpatients and Outpatients, difference being whether they are currently admitted in the hospital. The patients then interact with the system in various forms. Each patient is received by a receptionist who manages their record for the current stay. They are then assigned a doctor for their treatment, as well as a nurse in case of admittance. If admitted, they are usually assigned a bed in a room. They may require the use of various facilities such as blood test, X-Ray etc. After successful completion of their treatment, they are required to pay a bill amounting to the expenses of their visit which might be insured depending upon individual patients.

The bill is calculated based on the costs of all the facilities used as well as the room charges if applicable, along with the medicinal drug expenses used in the treatment so far and the fee for the consultancy service of the doctors. After it their records will get updated for future refence.

The other interactions are between the employee. They belong to a particular department, and may have different duties towards the patient assigned to them based on the role they assume in the hospital. Their personal data along with these interactions are stored in the database for automation of various tasks such as availability checking of the number of beds available which can be used for patient assignment. The medical history of a patient also includes all the entities one might have interacted with during their stay, which also includes the employees working at the hospital, which can be used for shift planning of these employees so as to normalize the load on them. The interaction of the medicine and the bill can be used for inventory verification, as well as automatically determining the overall requirement of medical drugs required by hospital. The assignment of an employee to a patient allows for logging their work and the amount of service they are doing for the hospital.

With all these outcomes in mind we decide on the following model for our hospital management system which includes the following entities and relationship sets tied together by the ER diagram shown later.

1.3 ENTITY AND RELATIONSHIP SETS

1.3.1 Entity Sets

- **Doctor (DoctorID, Qualification, Specialization, Charges, Room ID):** is the entity set containing the data regarding all the doctors working at the hospital such as their qualification, Specialization as well as charges that they have for their consultancy duty, room number they are assigned for appointments. They are uniquely identified in this table by DoctorID which is a unique ID exclusive for the role of Doctor.
- **Nurse (NurseID, Experience):** is the data storage of all the nurses working for the hospital. They are uniquely identified in this set by NurseID; a UID exclusively assigned to the role of nurse.
- **Receptionist (ReceptionistID):** stores the data required for unique identification of all the receptionists present in the hospital.
- **Employees (EmployeeID, Name, Salary, Email, Sex, Address, Contact No.):** is the superclass of all the people working in the hospital including Doctors, Nurses, Receptionists etc. It is used as a central records system of all the employees of the hospital. Every employee may assume a different role but will still be present in the central employee records identified uniquely by EmployeeID; a universal UID given to all the employees. Along with these attributes and properties, it also holds any necessary details required for identification and verification as well as contact.
- **Rooms (RoomID, RoomType, Floor, WardType, No. of Beds, RoomCharges):** is the record of all the rooms either currently available for use or in use each identified by their RoomID. It also contains data about the type of room it is as well as its location given by the floor at which it is present, the wards it is situated in and the number of beds that one room may contain as one general ward room often contains multiple beds separated by curtains, and charges of room per day.
- **Inpatient (PatientID, Name, Sex, Date Admitted, Date Discharged, Contact No., Address):** represents the patients that require to be admitted in the hospital. The

attributes are required for identification purposes while the date admitted and discharged are kept for logistic purposes of maintaining the present condition of the hospital such as the vacancy of rooms, availability of medical personnel, estimated requirement of basic necessities. It is a record of personal details of the patient.

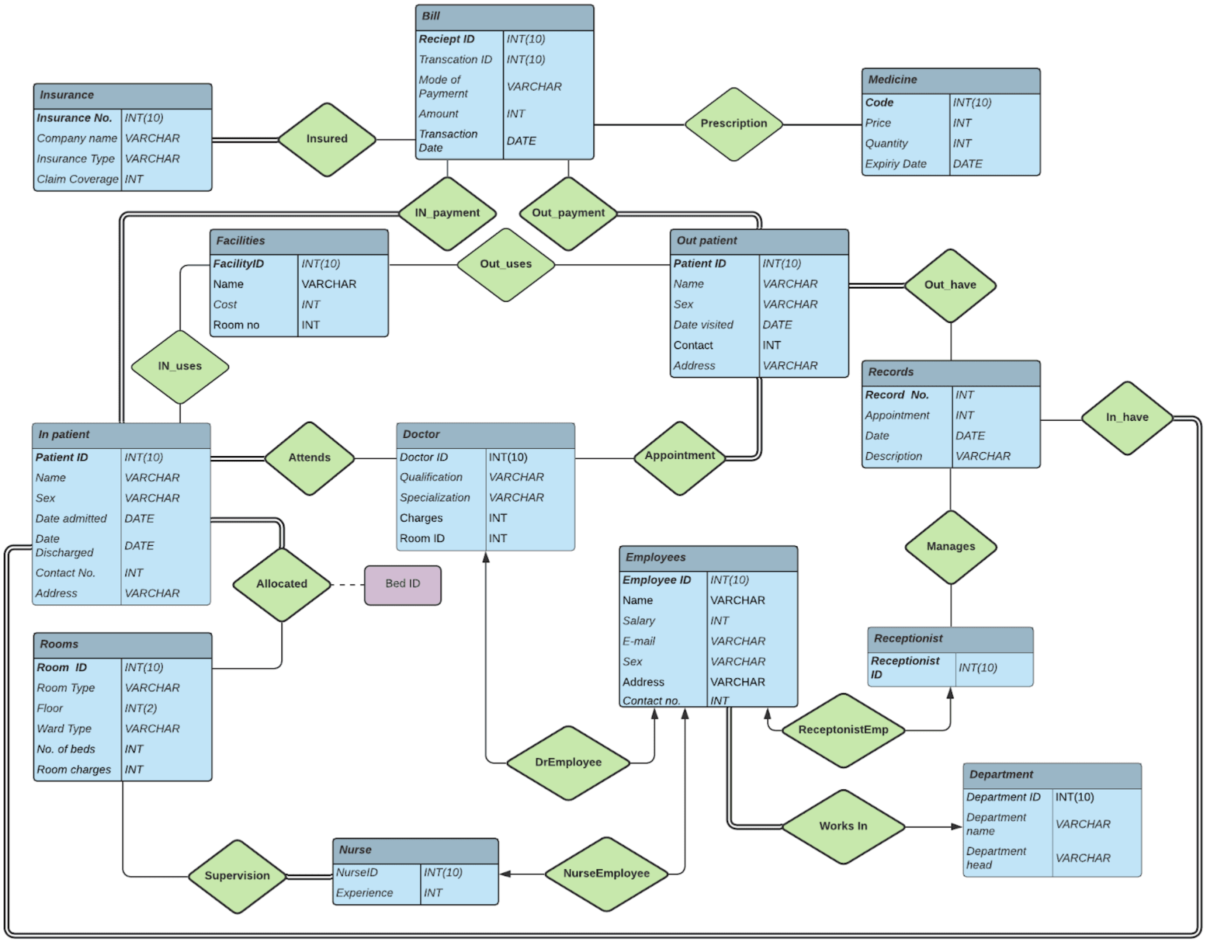
- **Outpatient (PatientID, Name, Sex, Date Admitted, Date Discharged, Contact, Address):** represents the patients that come in the daily clinic for consultation and treatment without requiring to be admitted. The other attributes are required for identification purposes, while Symptoms are required for doctor recommendation i.e., which type of doctor will be responsible for the patient. The date attribute is recorded in order to log all of this information for later use.
- **Bill (ReceiptID, TransactionID, ModeOfPayment, Amount, TransactionDate):** keeps the data related to all the payments made to the hospital. They keep the transactionID for reference purposes, amount and date for logging purposes as well as internal verification.
- **Insurance (InsuranceNo., CompanyName, InsuranceType, ClaimCoverage):** stores all the data regarding the insurance a patient might have, the amount of bill that is covered by the said insurance as well as the details of the insurance provider for logistic purposes.
- **Medicine (Code, Price, Quantity, ExpiryDate):** keeps the inventory of all the medicines available as well as their prices for billing purposes.
- **Facilities (FacilityID,FacilityType, Cost, Room_no):** stores the data available on all the facilities available such as medical tests, X-Ray etc. along with their cost and location in hospital. For future billing purposes and direction for patients.
- **Records (RecordNo., Appointment, Date, Description):** are used to store a historical medical record of all the patients treated in the hospital. RecordNo. Is used as a unique identification, while Appointment, Date and description are used for future references.
- **Department (DepartmentID, DepartmentName, DepartmentHead):** is the record of all the departments and their current heads.

1.3.2 Relationship Sets:

- **DrEmployee (Doctor --- Employee):** is a one-to-one mapping between DoctorsID and their EmpoyeeID. The DoctorID differs from the employeeID because of the exclusivity of the former. It is a one-to-one relationship as every doctor map to one employeeID.
- **NurseEmployee (Nurse --- Employee):** is a one-to-one mapping between NurseID and their EmpoyeeID. It is similar to above-described relation
- **ReceptionistEmp (Receptionist --- Employee):** is a one-to-one mapping between ReceptionistID and their EmpoyeeID. It is similar to above-described relation
- **WorksIn (Employee --- Department):** stores which department each employee works in.Its a many-to-one relationship, with every employee working in some department.

- **Allocated (Inpatient --- Rooms):** is the relationship between rooms available and the Inpatient that are being admitted based on their requirements of room type as well as Floor at which they are comfortable at staying while being near all the required facilities. Every Inpatient requires a room; thus, it is a many to many relationships with full participation from Inpatients. It also keeps an attribute ‘BedID’ as some rooms may have multiple beds and in order to distinguish them, we keep this attribute. It also helps in determining the vacancy of beds in a particular room.
- **Supervision (Nurse --- Rooms):** relates which nurse is responsible for managing and supervising the room that a patient is admitted in. It is a many to many relations because over time any nurse may attend to any room, while every nurse will be responsible for some room.
- **Attends (Doctor --- InPatient):** keeps a track of which doctor is monitoring and attending a patient and his/her treatment. It is a many-to-many relation along the same reasoning as mentioned before.
- **InUse/OutUse(In/Out Patients --- Facilities):** This stores the relation whether some patient uses any facility like X-Ray, blood test etc. later used for billing purposes. It is a many-to-many relation as any number of patients may use a particular facility and over time any number of facilities may be used by a patient.
- **InHave/OutHave(In/Out Patients --- Records):** it maps the patients to their records. Every patient will have a record which will hold all the historical details regarding his/her treatment.
- **Appointment (Doctor --- OutPatient):** is the relationship similar to attends but for the outPatient type.
- **InPayment/OutPayment (In/Out Patients --- Bill):** maps the patients to the bills of their current treatment. Patients have complete participation, while the relation is many-to-many for obvious reasons also mentioned before.
- **Prescription (Bill --- Medicine):** stores which medicine was used in a treatment billed with the ReceiptID in order to calculate the medical expense.
- **Insured (Insurance --- Bill):** relates which insurance payment was related to which bill. Every insurance waiver needs to relate to a valid bill.
- **Manages (Receptionist --- Record):** stores the log regarding which receptionist accessed which medical record as any time. It is a many-to-many relation as any receptionist may access any medical record with authorized permission.

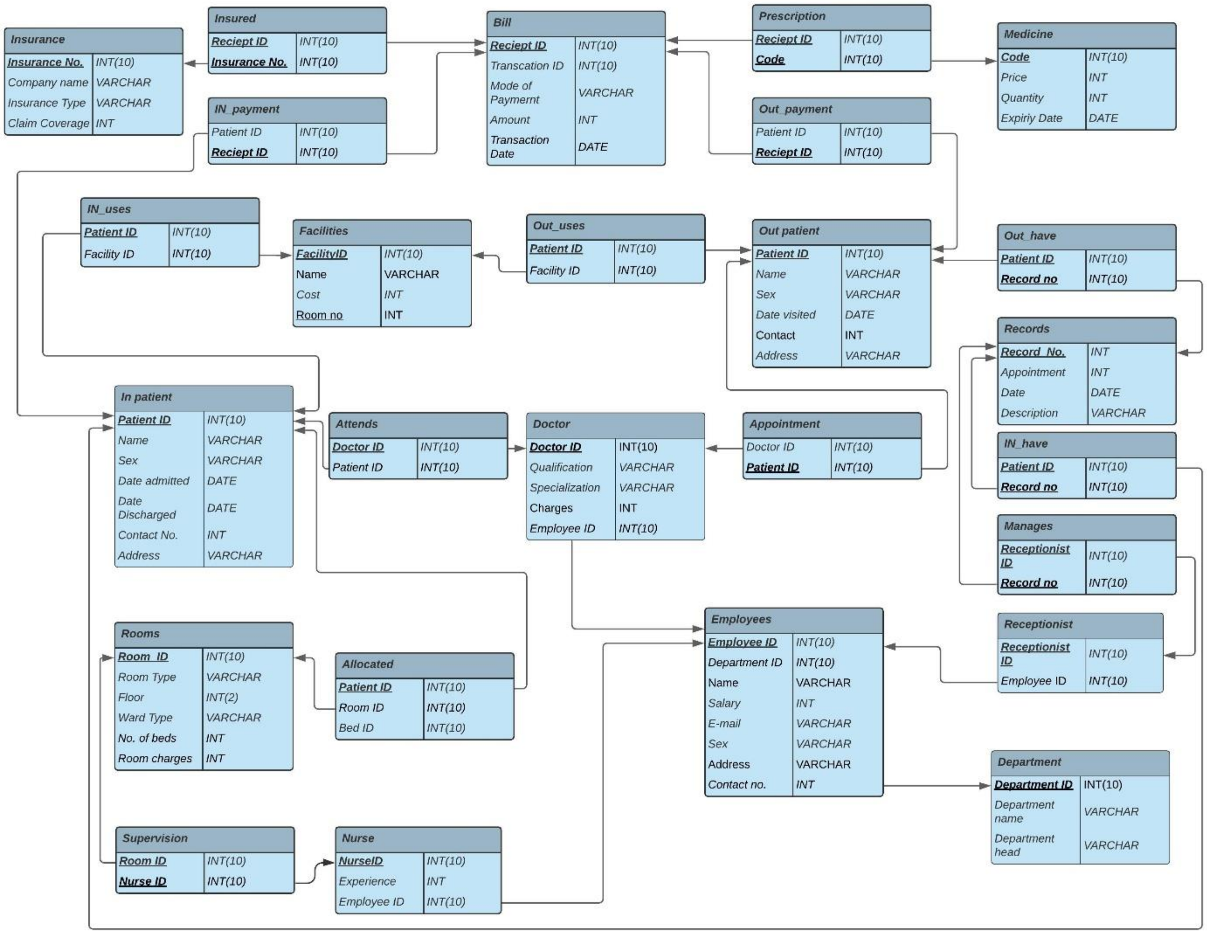
2 ER Diagram


Bill
Reciept ID INT(10)
Transcation ID INT(10)
Mode of Paymernt VARCHAR
Amount INT
Transaction Date DATE
Insurance
Insurance No. INT(10)
Company name VARCHAR
Insurance Type VARCHAR
Claim Coverage INT
Insured
Prescription
Medicine
Code INT(10)
Price INT
Quantity INT
Expiriy Date DATE
IN_payment
Out_payment
Facilities
FacilityID INT(10)
Name VARCHAR
Cost INT
Room no INT
Out_uses
Out patient
Patient ID INT(10)
Name VARCHAR
Sex VARCHAR
Date visited DATE
Contact INT
Address VARCHAR
Out_have
IN_uses
Records
Record No. INT
Appointment INT
Date DATE
Description VARCHAR
In_have
In patient
Patient ID INT(10)
Name VARCHAR
Sex VARCHAR
Date admitted DATE
Date Discharged DATE
Contact No. INT
Address VARCHAR
Attends
Doctor
Doctor ID INT(10)
Qualification VARCHAR
Specialization VARCHAR
Charges INT
Room ID INT
Appointment
Manages
Allocated
Bed ID
Employees
Employee ID INT(10)
Name VARCHAR
Salary INT
E-mail VARCHAR
Sex VARCHAR
Address VARCHAR
Contact no. INT
Receptionist
Receptionist ID INT(10)
Rooms
Room ID INT(10)
Room Type VARCHAR
Floor INT(2)
Ward Type VARCHAR
No. of beds INT
Room charges INT
ReceptionistEmp
DrEmployee
Department
Department ID INT(10)
Department name VARCHAR
Department head VARCHAR
Works In
Nurse
NurseID INT(10)
Experience INT
Supervision
NurseEmployee

3 SCHEMA DIAGRAM

A database schema, along with primary key and foreign-key constraints, can be depicted by schema diagrams. Each relation appears as a box, with the relation's name at the top in blue and the attributes listed inside the box. Primary-key attributes are shown underlined. Foreign-key constraints appear as arrows from the foreign-key attributes of the referencing relation to the primary key of the referenced relation.

We also reduce the relationship set tables using the proper rules of ER reduction to efficiently store the information. In our example we have used reduction on the relationship sets, DrEmployee, NurseEmployee, ReceptionistEmployee as well as WorksIn since these sets were either one-to-one relation or one-to-many which allowed us to store them as an additional attribute in one of the entity-set tables, without the loss of any information.

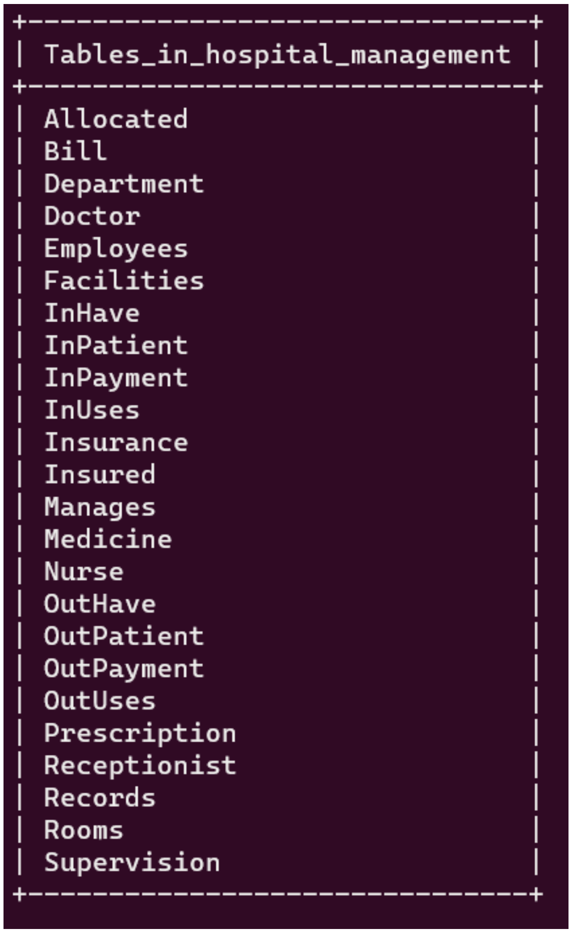


4 DATABASE SCHEMA

We used the following SQL script to create our database following our schema diagram. To replicate the results, we use the command:

$$sudo\ mysql - u < USER > \ -p\ \ < hospital.sql$$

Following are the tables created using the script [see Appendix]:



```
+-------------------------------+
| Tables_in_hospital_management |
+-------------------------------+
| Allocated                     |
| Bill                          |
| Department                    |
| Doctor                        |
| Employees                     |
| Facilities                    |
| InHave                        |
| InPatient                     |
| InPayment                     |
| InUses                        |
| Insurance                     |
| Insured                       |
| Manages                       |
| Medicine                      |
| Nurse                         |
| OutHave                       |
| OutPatient                    |
| OutPayment                    |
| OutUses                       |
| Prescription                  |
| Receptionist                  |
| Records                       |
| Rooms                         |
| Supervision                   |
+-------------------------------+
```

Tables for reference to check the correct creation of database schema

NOTE: [See Appendix for more descriptive database schema]

5 DATABASE POPULATION

We have populated the database with appropriate placeholder data, while following the dependencies and data flow of the conceptual model. To replicate the database, run the following command with the file linked at the end in Appendix. Database needs to be created and table needs to be formed previously, using the instructions given in the previous topic. Command:

$$sudo\ mysql - u < USER > -p\ hospital\_management\ \ < \ data.sql$$

Database backup created by the following commands:

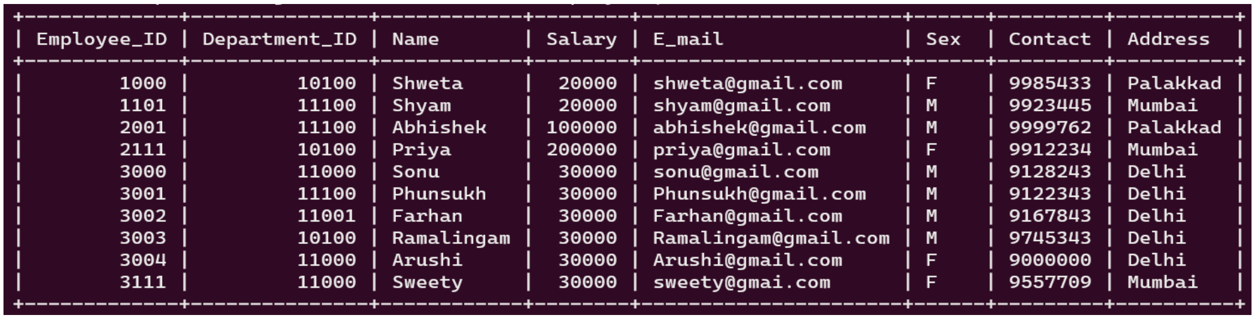
$$mysql - p - -no - data\ \ hospital\_management\ \ > hospital\_without\_data.sql$$

$$mysql - p\ hospital\_management\ \ > hospital\_with\_data.sql$$

These files can be found in the backup folder accessed by the link provided in the appendix.

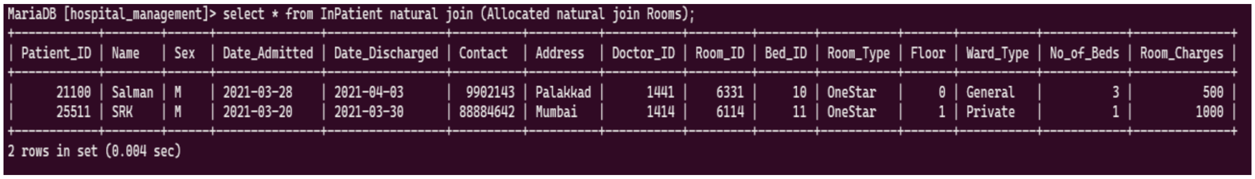
6 KEY HIGHLIGHTS OF THE DATABASE

Employees:



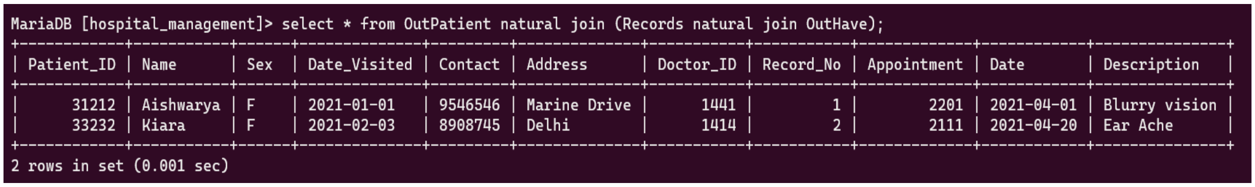
```
+-------------+---------------+------------+--------+----------------------+------+---------+----------+
| Employee_ID | Department_ID | Name       | Salary | E_mail               | Sex  | Contact | Address  |
+-------------+---------------+------------+--------+----------------------+------+---------+----------+
|        1000 |         10100 | Shweta     |  20000 | shweta@gmail.com     | F    | 9985433 | Palakkad |
|        1101 |         11100 | Shyam      |  20000 | shyam@gmail.com      | M    | 9923445 | Mumbai   |
|        2001 |         11100 | Abhishek   | 100000 | abhishek@gmail.com   | M    | 9999762 | Palakkad |
|        2111 |         10100 | Priya      | 200000 | priya@gmail.com      | F    | 9912234 | Mumbai   |
|        3000 |         11000 | Sonu       |  30000 | sonu@gmail.com       | M    | 9128243 | Delhi    |
|        3001 |         11100 | Phunsukh   |  30000 | Phunsukh@gmail.com   | M    | 9122343 | Delhi    |
|        3002 |         11001 | Farhan     |  30000 | Farhan@gmail.com     | M    | 9167843 | Delhi    |
|        3003 |         10100 | Ramalingam |  30000 | Ramalingam@gmail.com | M    | 9745343 | Delhi    |
|        3004 |         11000 | Arushi     |  30000 | Arushi@gmail.com     | F    | 9000000 | Delhi    |
|        3111 |         11000 | Sweety     |  30000 | sweety@gmai.com      | F    | 9557709 | Mumbai   |
+-------------+---------------+------------+--------+----------------------+------+---------+----------+
```

InPatients:



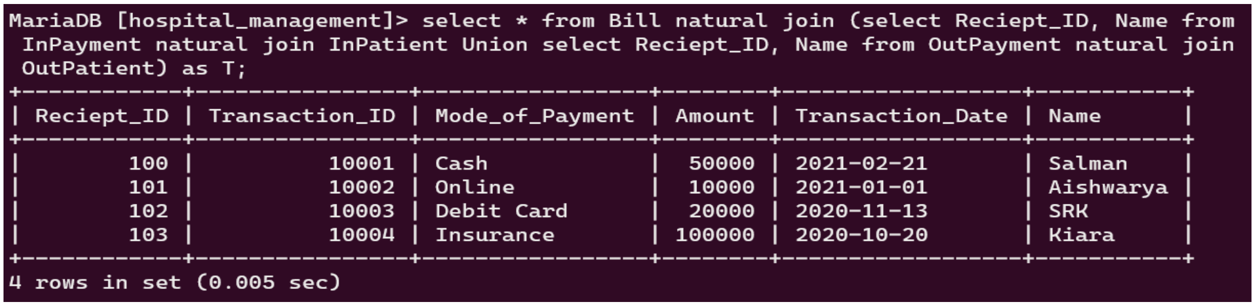
```
MariaDB [hospital_management]> select * from InPatient natural join (Allocated natural join Rooms);
+------------+--------+-----+---------------+-----------------+----------+----------+-----------+---------+--------+-----------+-------+-----------+-----------+--------------+
| Patient_ID | Name   | Sex | Date_Admitted | Date_Discharged | Contact  | Address  | Doctor_ID | Room_ID | Bed_ID | Room_Type | Floor | Ward_Type | No_of_Beds | Room_Charges |
+------------+--------+-----+---------------+-----------------+----------+----------+-----------+---------+--------+-----------+-------+-----------+-----------+--------------+
|      21100 | Salman | M   | 2021-03-28    | 2021-04-03      |  9902143 | Palakkad |      1441 |    6331 |     10 | OneStar   |     0 | General   |          3 |          500 |
|      25511 | SRK    | M   | 2021-03-20    | 2021-03-30      | 88884642 | Mumbai   |      1414 |    6114 |     11 | OneStar   |     1 | Private   |          1 |         1000 |
+------------+--------+-----+---------------+-----------------+----------+----------+-----------+---------+--------+-----------+-------+-----------+-----------+--------------+
2 rows in set (0.004 sec)
```

OutPatients:



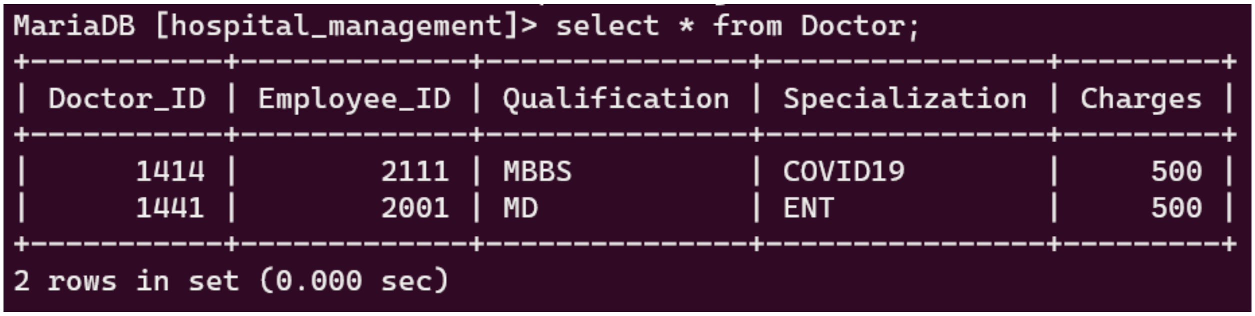
```
MariaDB [hospital_management]> select * from OutPatient natural join (Records natural join OutHave);
+------------+-----------+------+------------+---------+--------------+-----------+-----------+-------------+------------+---------------+
| Patient_ID | Name      | Sex  | Date_Visited | Contact | Address      | Doctor_ID | Record_No | Appointment | Date       | Description   |
+------------+-----------+------+------------+---------+--------------+-----------+-----------+-------------+------------+---------------+
|      31212 | Aishwarya | F    | 2021-01-01 | 9546546 | Marine Drive |      1441 |         1 |        2201 | 2021-04-01 | Blurry vision |
|      33232 | Kiara     | F    | 2021-02-03 | 8908745 | Delhi        |      1414 |         2 |        2111 | 2021-04-20 | Ear Ache      |
+------------+-----------+------+------------+---------+--------------+-----------+-----------+-------------+------------+---------------+
2 rows in set (0.001 sec)
```

Bills:



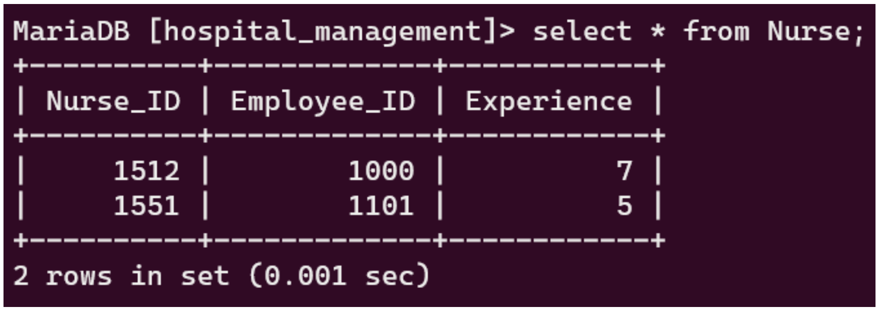
```
MariaDB [hospital_management]> select * from Bill natural join (select Reciept_ID, Name from
 InPayment natural join InPatient Union select Reciept_ID, Name from OutPayment natural join
 OutPatient) as T;
+------------+----------------+-----------------+--------+------------------+-----------+
| Reciept_ID | Transaction_ID | Mode_of_Payment | Amount | Transaction_Date | Name      |
+------------+----------------+-----------------+--------+------------------+-----------+
|        100 |          10001 | Cash            |  50000 | 2021-02-21       | Salman    |
|        101 |          10002 | Online          |  10000 | 2021-01-01       | Aishwarya |
|        102 |          10003 | Debit Card      |  20000 | 2020-11-13       | SRK       |
|        103 |          10004 | Insurance       | 100000 | 2020-10-20       | Kiara     |
+------------+----------------+-----------------+--------+------------------+-----------+
4 rows in set (0.005 sec)
```

Doctors:



```
MariaDB [hospital_management]> select * from Doctor;
+-----------+-------------+---------------+----------------+---------+
| Doctor_ID | Employee_ID | Qualification | Specialization | Charges |
+-----------+-------------+---------------+----------------+---------+
|      1414 |        2111 | MBBS          | COVID19        |     500 |
|      1441 |        2001 | MD            | ENT            |     500 |
+-----------+-------------+---------------+----------------+---------+
2 rows in set (0.000 sec)
```

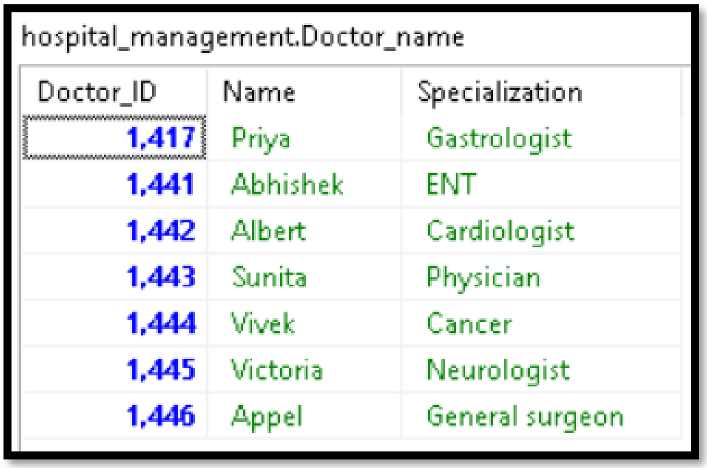
Nurses:



```
MariaDB [hospital_management]> select * from Nurse;
+----------+-------------+------------+
| Nurse_ID | Employee_ID | Experience |
+----------+-------------+------------+
|     1512 |        1000 |          7 |
|     1551 |        1101 |          5 |
+----------+-------------+------------+
2 rows in set (0.001 sec)
```

7 Database Views

create view **Doctor_name** as select D.Doctor_ID, E.Name, D.Specialization from **Doctor** as D **natural join Employees** as E;



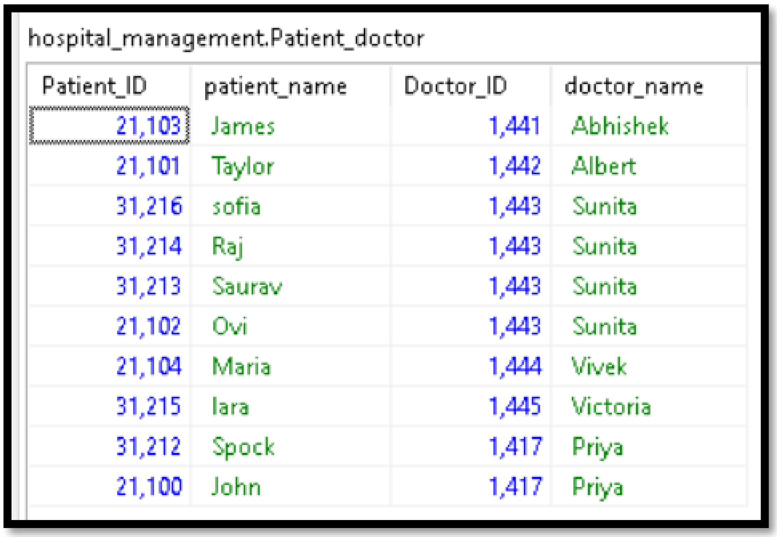
hospital_management.Doctor_name

Doctor_ID	Name	Specialization
1,417	Priya	Gastrologist
1,441	Abhishek	ENT
1,442	Albert	Cardiologist
1,443	Sunita	Physician
1,444	Vivek	Cancer
1,445	Victoria	Neurologist
1,446	Appel	General surgeon

Motivation: In any hospital management system, we always need a list of doctors along with their specialization that are currently available for consultation by the patients. This forms the basis of an appointment booking system which is crucial for any hospital.

We have stored information regarding the role of an employee (i.e., doctor in this case) in the doctor table, which however, lack the personal information regarding them such as name, and specialization; which is stored in the employee table. Thus, we use natural join to find the desired information.

```
create view Patient_doctor as (select p.Patient_ID, p.Name as
patient_name, p.Date, p.Doctor_ID , d.Name as doctor_name from
(select Patient_ID,Name,Doctor_ID, Date_Admitted as Date from
InPatient union select Patient_ID,Name,Doctor_ID, Date_Visited as
Date from OutPatient) as p inner join Doctor_name as d on
p.Doctor_ID=d.Doctor_ID);
```



hospital_management.Patient_doctor

Patient_ID	patient_name	Doctor_ID	doctor_name
21,103	James	1,441	Abhishek
21,101	Taylor	1,442	Albert
31,216	sofia	1,443	Sunita
31,214	Raj	1,443	Sunita
31,213	Saurav	1,443	Sunita
21,102	Ovi	1,443	Sunita
21,104	Maria	1,444	Vivek
31,215	lara	1,445	Victoria
31,212	Spock	1,417	Priya
21,100	John	1,417	Priya

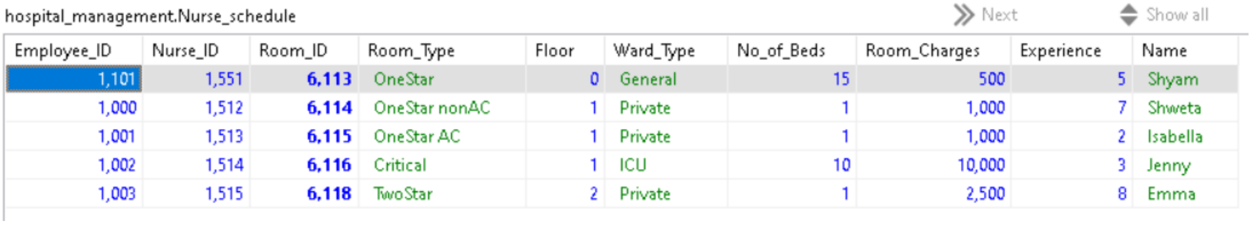
Motivation: Every hospital needs a history/log of the treatments done in their hospital. This view stores exactly that in the form of patient doctor interaction. This view stores all such interaction that happen till date. Moreover, this view is further used to get the appointments for the day. By adding the date column and querying on it using the where clause and CURDATE() function.

```
create view Appointments as

(SELECT * FROM Patient_doctor WHERE ( DATE = CURDATE() OR DATE IS
NULL ) );
```

Motivation: For proper management of today's appointments, it is necessary to keep a list of patients that have visited today or have not been discharged yet. This also allows us to keep a track of the workload on the staff.

```
create view Nurse_schedule as select * from Rooms natural join
Supervision natural join Nurse natural join (select Name, Employee_ID
from Employees) as e;
```

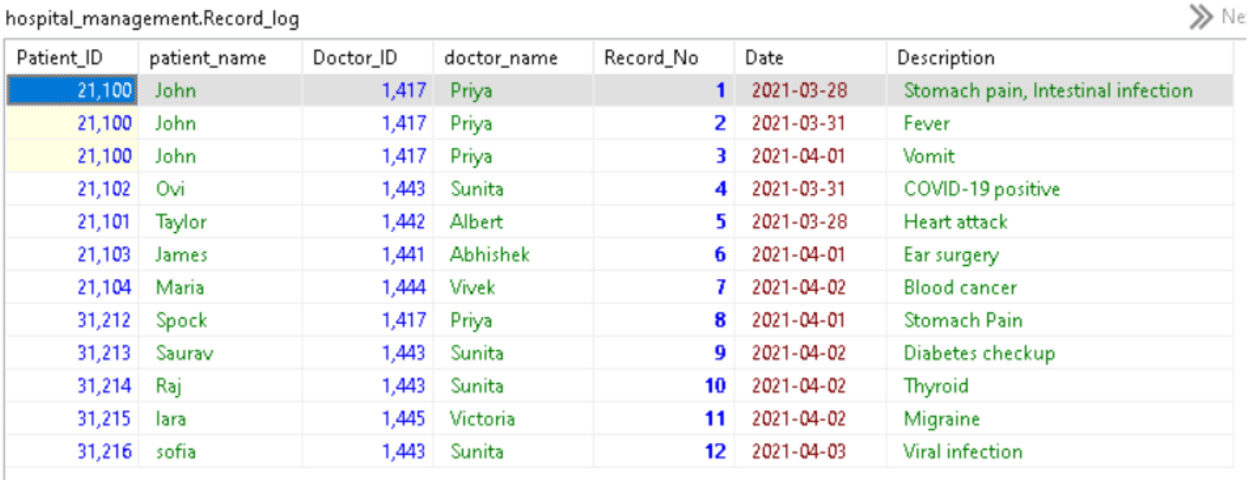


hospital_management.Nurse_schedule

Employee_ID	Nurse_ID	Room_ID	Room_Type	Floor	Ward_Type	No_of_Beds	Room_Charges	Experience	Name
1,101	1,551	6,113	OneStar	0	General	15	500	5	Shyam
1,000	1,512	6,114	OneStar nonAC	1	Private	1	1,000	7	Shweta
1,001	1,513	6,115	OneStar AC	1	Private	1	1,000	2	Isabella
1,002	1,514	6,116	Critical	1	ICU	10	10,000	3	Jenny
1,003	1,515	6,118	TwoStar	2	Private	1	2,500	8	Emma

Motivation: This view acts as a duty chart for the nursing staff indicating where their duty is at a particular day. This also allows us to chart the duty just by changing the supervision table.

create view **Record_log** as select * from **Patient_doctor** natural inner join **Records**;

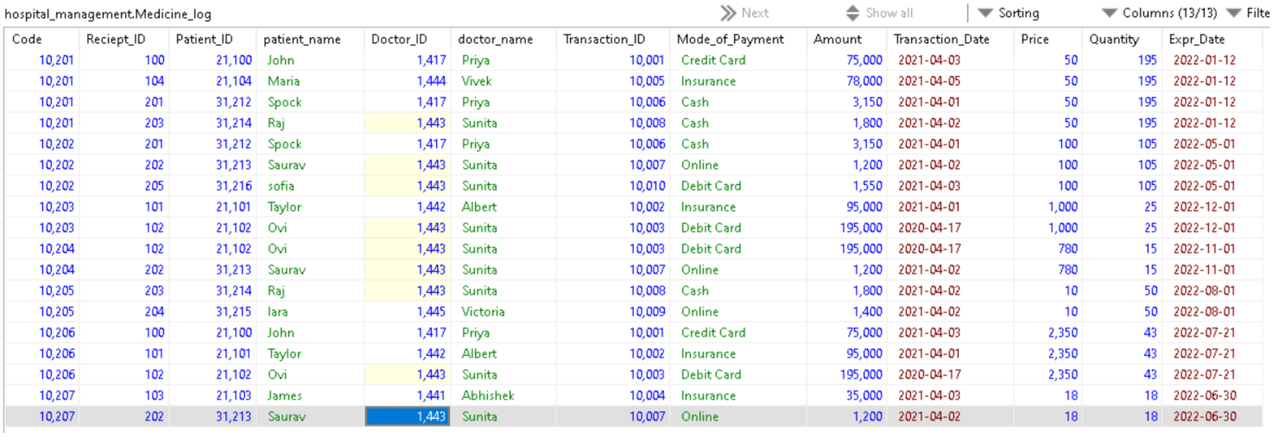


hospital_management.Record_log

Patient_ID	patient_name	Doctor_ID	doctor_name	Record_No	Date	Description
21,100	John	1,417	Priya	1	2021-03-28	Stomach pain, Intestinal infection
21,100	John	1,417	Priya	2	2021-03-31	Fever
21,100	John	1,417	Priya	3	2021-04-01	Vomit
21,102	Ovi	1,443	Sunita	4	2021-03-31	COVID-19 positive
21,101	Taylor	1,442	Albert	5	2021-03-28	Heart attack
21,103	James	1,441	Abhishek	6	2021-04-01	Ear surgery
21,104	Maria	1,444	Vivek	7	2021-04-02	Blood cancer
31,212	Spock	1,417	Priya	8	2021-04-01	Stomach Pain
31,213	Saurav	1,443	Sunita	9	2021-04-02	Diabetes checkup
31,214	Raj	1,443	Sunita	10	2021-04-02	Thyroid
31,215	lara	1,445	Victoria	11	2021-04-02	Migraine
31,216	sofia	1,443	Sunita	12	2021-04-03	Viral infection

Motivation: This view presents the records tables in a more human interpretable way, while at the same time allowing us to put restrictions based on individual user name/ID easily. Need for previous medical records is realized in many situations where they are required for future reference in some other treatment or for referring the patient to a different doctor/hospital. The privacy of such records is of utmost concern, thus, only the doctors involved and the patient could access such records unless given special permission to do so.

create view **Medicine_log** as select * from **Patient_doctor** natural inner join (select * from **InPayment union** select * from **OutPayment**) as p natural inner join **Bill** natural inner join **Prescription** natural inner join **Medicine**;



hospital_management.Medicine_log

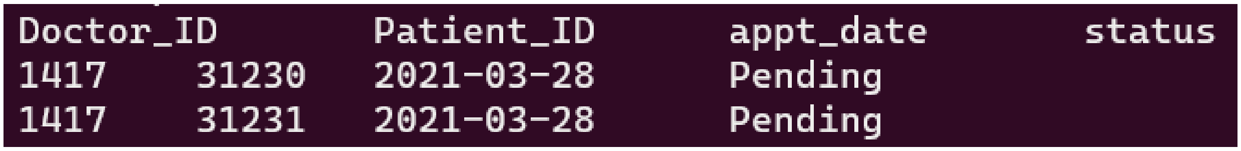
Code	Reciept_ID	Patient_ID	patient_name	Doctor_ID	doctor_name	Transaction_ID	Mode_of_Payment	Amount	Transaction_Date	Price	Quantity	Expr_Date
10,201	100	21,100	John	1,417	Priya	10,001	Credit Card	75,000	2021-04-03	50	195	2022-01-12
10,201	104	21,104	Maria	1,444	Vivek	10,005	Insurance	78,000	2021-04-05	50	195	2022-01-12
10,201	201	31,212	Spock	1,417	Priya	10,006	Cash	3,150	2021-04-01	50	195	2022-01-12
10,201	203	31,214	Raj	1,443	Sunita	10,008	Cash	1,800	2021-04-02	50	195	2022-01-12
10,202	201	31,212	Spock	1,417	Priya	10,006	Cash	3,150	2021-04-01	100	105	2022-05-01
10,202	202	31,213	Saurav	1,443	Sunita	10,007	Online	1,200	2021-04-02	100	105	2022-05-01
10,202	205	31,216	sofia	1,443	Sunita	10,010	Debit Card	1,550	2021-04-03	100	105	2022-05-01
10,203	101	21,101	Taylor	1,442	Albert	10,002	Insurance	95,000	2021-04-01	1,000	25	2022-12-01
10,203	102	21,102	Ovi	1,443	Sunita	10,003	Debit Card	195,000	2020-04-17	1,000	25	2022-12-01
10,204	102	21,102	Ovi	1,443	Sunita	10,003	Debit Card	195,000	2020-04-17	780	15	2022-11-01
10,204	202	31,213	Saurav	1,443	Sunita	10,007	Online	1,200	2021-04-02	780	15	2022-11-01
10,205	203	31,214	Raj	1,443	Sunita	10,008	Cash	1,800	2021-04-02	10	50	2022-08-01
10,205	204	31,215	lara	1,445	Victoria	10,009	Online	1,400	2021-04-02	10	50	2022-08-01
10,206	100	21,100	John	1,417	Priya	10,001	Credit Card	75,000	2021-04-03	2,350	43	2022-07-21
10,206	101	21,101	Taylor	1,442	Albert	10,002	Insurance	95,000	2021-04-01	2,350	43	2022-07-21
10,206	102	21,102	Ovi	1,443	Sunita	10,003	Debit Card	195,000	2020-04-17	2,350	43	2022-07-21
10,207	103	21,103	James	1,441	Abhishek	10,004	Insurance	35,000	2021-04-03	18	18	2022-06-30
10,207	202	31,213	Saurav	1,443	Sunita	10,007	Online	1,200	2021-04-02	18	18	2022-06-30

Motivation: Such logs help in the logistic planning and management of the hospital. They help in pharmacy inventory management, provide a confirmation of the payments, as well as find use in automatic bill generation which may be used in the future. This again has restrictive access but unlike previous view where restrictions were only in one dimension i.e., row, this includes restrictions on both the dimension based on the role of user accessing them. Patients would like to see their full payment detail, however such individual details are not necessarily shred with the inventory managers.

8 FUNCTIONS AND PROCEDURES

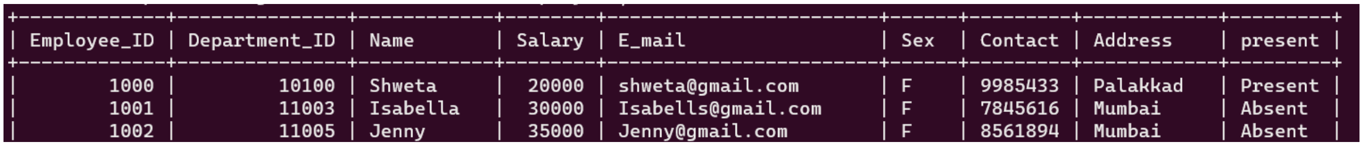
We have created the following procedures to help in application of the system.

- **Init_attendance**: is used to initialize a new column in the doctors table to store the availability of the doctors at current time
- **mark_attendance**: is used as a digital equivalent to clocking one’s shift in the hospital, this allows us to keep a track of available doctors in real time.
- **drop_attendance**: works as a digital equivalent to checkout when one’s shift is finished. This along with mark attendance completes the system of available doctors which will later be used for functionalities like booking an appointment, scheduling clinic patient etc.
- **init_appointment**: creates a table for daily appointments that the hospital may handle.
- **add_apointment**: is used primarily by the patients to book an appointment with the available doctors for a particular date.
- **mark_appointment**: is primarily used by doctors to accept an appointment. This feature is added mainly to reduce overburden on a single doctor.
- **flush_appointment**: is used by a doctor to clear out all the patients that have received a treatment already.
- **admit_patient**: is used by the doctor to admit an outpatient into the hospital as an inpatient. Since this usually requires a doctor’s permission, hence it is not given to the patients directly.



```
Doctor_ID       Patient_ID      appt_date       status
1417    31230   2021-03-28      Pending
1417    31231   2021-03-28      Pending
```

Appointment procedure being called.



```
+-------------+---------------+------------+--------+-----------------------+------+---------+-------------+---------+
| Employee_ID | Department_ID | Name       | Salary | E_mail                | Sex  | Contact | Address     | present |
+-------------+---------------+------------+--------+-----------------------+------+---------+-------------+---------+
|        1000 |         10100 | Shweta     |  20000 | shweta@gmail.com      | F    | 9985433 | Palakkad    | Present |
|        1001 |         11003 | Isabella   |  30000 | Isabells@gmail.com    | F    | 7845616 | Mumbai      | Absent  |
|        1002 |         11005 | Jenny      |  35000 | Jenny@gmail.com       | F    | 8561894 | Mumbai      | Absent  |
```

After attendance was marked for id = 1000

For billing system, our database is designed in such a manner that we can find the total bill of a patient by simple queries and thus was not implemented as a procedure.

9 WEB APP

The Following technologies were used in the backend and frontend to facilitate the formation of the web app:

- HTML, CSS: used for basic webpage structure and styling. We also used Bootstrapping for the HTML template. The twitter bootstrap was selected, allowing us to quickly create a web app with professional look.
- JavaScript: to make the webpages interactive. Allowing user to retrieve as well as give data. This is important as our aim for this project is to not just present the information already present but rather allow system to fully function through this web app.
- Node.js: was used as the serve- side backend framework. Since the majority of functionality need to be implemented using MariaDB itself, node was mainly used for URL routing and database connection.
- MariaDB: was the database used, without any choice. All the data is stored using it as well as all the functionalities given to the user or employee are implemented within MariaDB itself.

These functionality implementations include the previously mentioned procedures, which allow the functionality of appointment booking, attendance marks, etc., to name a few. Apart form these procedures we have also created a trigger which is responsible for automatically marking employees absent once their shift gets over.

9.1 REASON FOR CHOICE

The frontend doesn’t really give much choice, with HTML as the only markup structuring language, CSS being used for styling and JavaScript being used to make these webpages interactive.

There again was no choice for database, as MariaDB is a compulsory part of the project specifications.

As for backend, we mainly used Node.js due to our familiarity in using the framework due to past experiences. We already were using JavaScript to make the webpages interactive, thus the choice for using a .js framework was obvious.

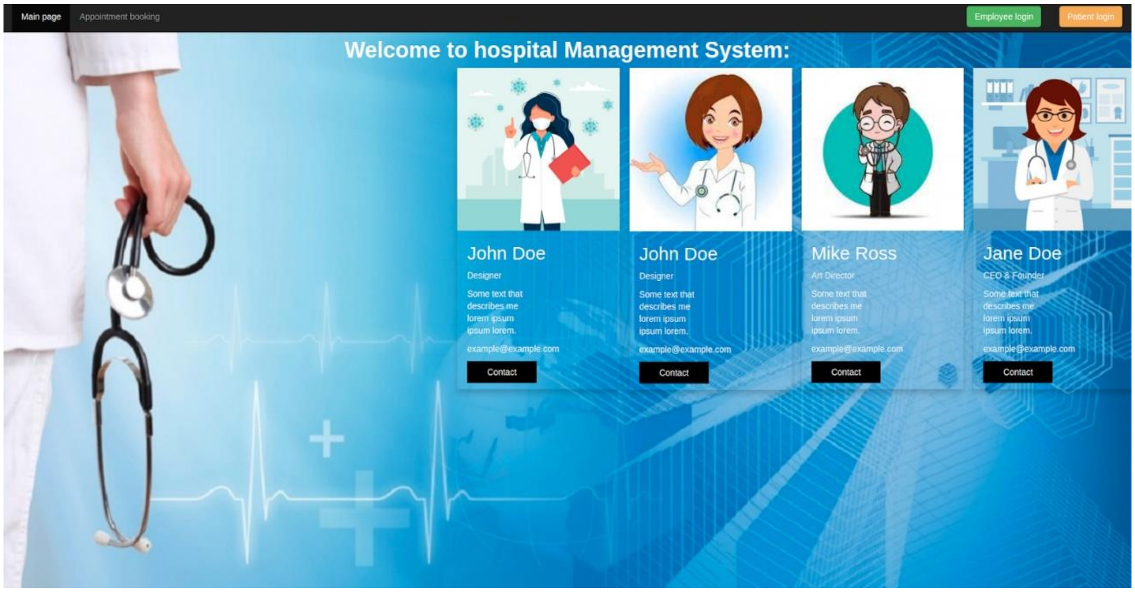
Specification: We connect to the database using the MySQL module already provided by the framework, the app runs on port 4000 (default choice).

Our main page is at ‘\main’ route, while the details of a particular type of employee is routed to ‘\<type>’ where <type> can be doctor, nurse and receptionist. The access to these pages is secured by an employee login.

9.2 Challenges Faced

Node.js doesn't really allow a good interface for interacting with HTML pages directly, thus we had to make use of the 'fs', 'Jsdom' module for interacting with data in HTML files. 'Jquery' module was used to change specific tags and classes in an HTML file.

9.3 Web App Instances



Welcome to hospital Management System:

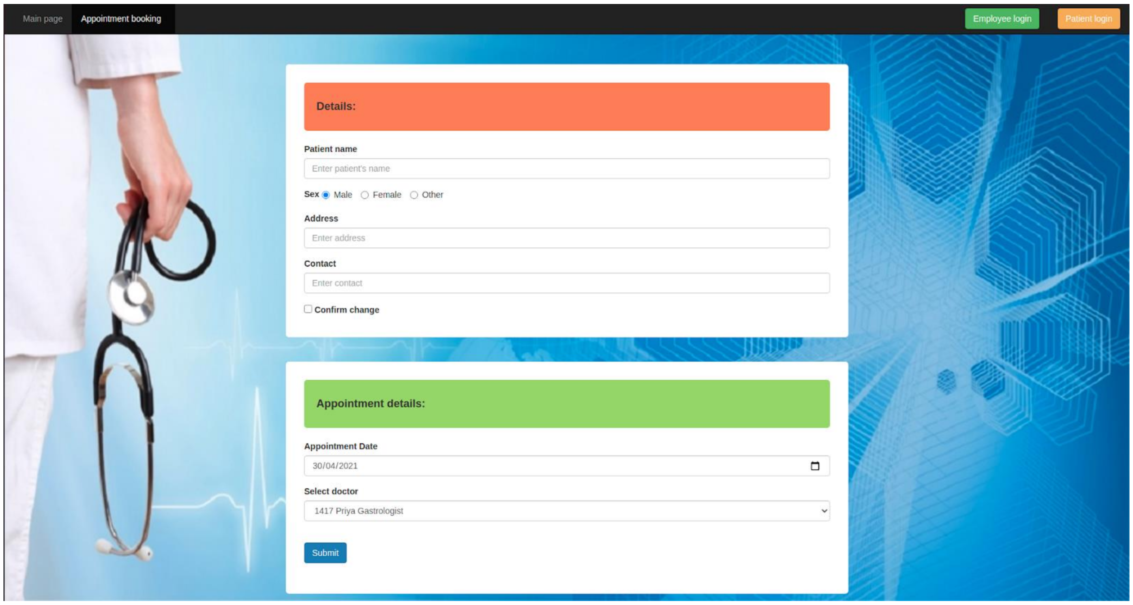
John Doe
Designer
Some text that describes me lorem ipsum ipsum lorem.
example@example.com

John Doe
Designer
Some text that describes me lorem ipsum ipsum lorem.
example@example.com

Mike Ross
Art Director
Some text that describes me lorem ipsum ipsum lorem.
example@example.com

Jane Doe
CEO & Founder
Some text that describes me lorem ipsum ipsum lorem.
example@example.com

Main



Details:

Patient name
Enter patient's name

Sex ● Male ○ Female ○ Other

Address
Enter address

Contact
Enter contact

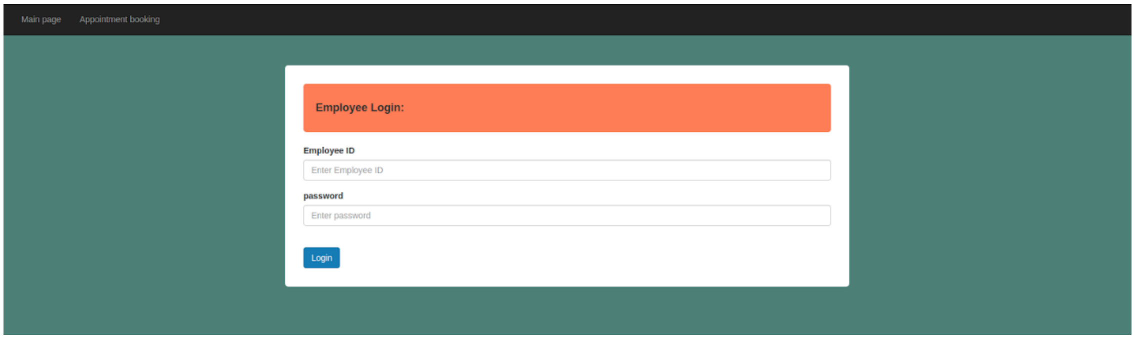
□ Confirm change

Appointment details:

Appointment Date
30/04/2021

Select doctor
1417 Priya Gastrologist

Appointment Interface



Employee Login:

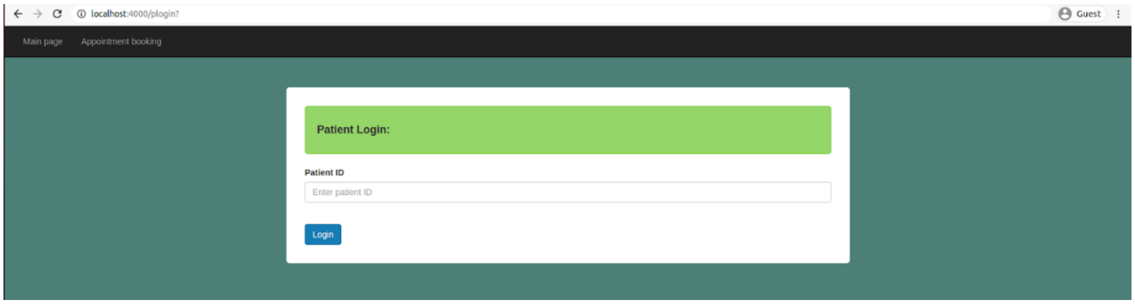
Employee ID

Enter Employee ID

password

Enter password

Employee Login

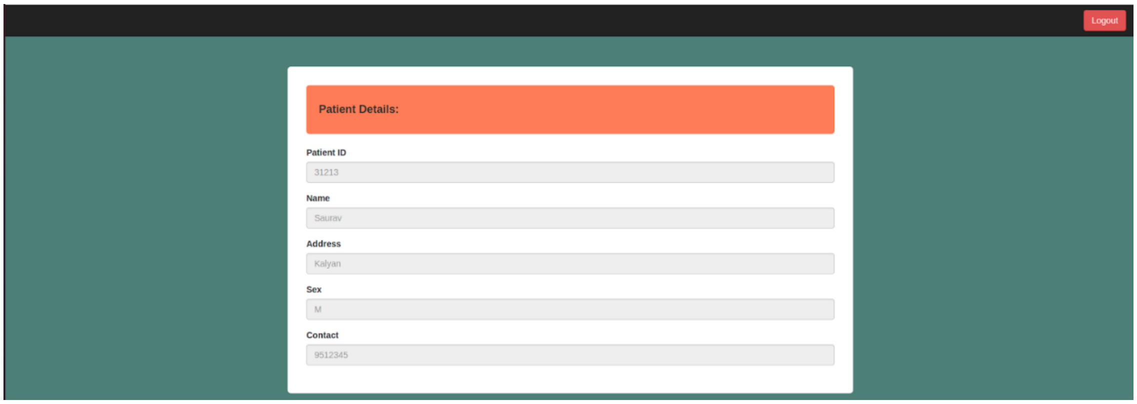


Patient Login:

Patient ID

Enter patient ID

Patient Login



Patient Details:

Patient ID

31213

Name

Saurav

Address

Kalyan

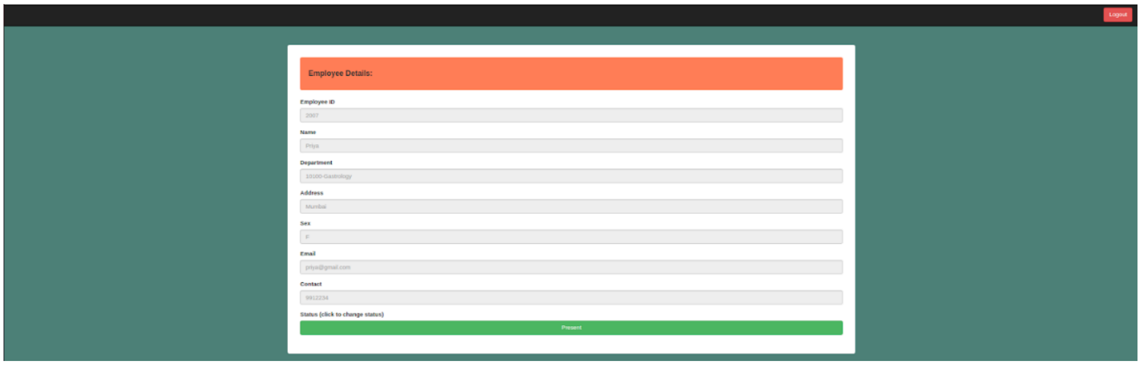
Sex

M

Contact

9512345

After Successful Patient Login



Employee Details:

Employee ID

2007

Name

Priya

Department

10100-Gastrology

Address

Mumbai

Sex

F

Email

priya@gmail.com

Contact

9912234

Status (click to change status)

After Successful Doctor Login

APPENDIX

Google Drive Link:

https://drive.google.com/drive/folders/11Xhl_cjLnTq3b6Ay3Ls-LkFiB5sfuyFO?usp=sharing

DESC of tables for reference:

Awk Command used in the terminal to generate all these descriptions:

```
for i in $(sudo mysql hospital_management -e 'SHOW TABLES' | grep -v
"Tables_in" | awk '{print $1}'); do echo "TABLE: $i"; sudo mysql
hospital_management -e "DESC $i"; done
```

Detailed Description of tables: tables.txt file

Backup files: backup/hospital_without_data.sql

backup/hospital_with_data.sql

GitLab Link: https://gitlab.com/111801054/demo/-/tree/Task4