Name: Sahil Jayant Chaudhari
Roll No.: 111801054
Team No.: 9

Hospital Management System

INDEX

Contribution:

Task 1:

Abdullah khan (111801001)	Built entity sets in ERD and assigned type
Sahil J. Chaudhari (111801054)	Built relational set and relations
Vishesh Munjal (111801055)	Built entity sets in ERD and attributes
Harsh Parihar (111801015)	N.A.

Task 2:

Abdullah khan (111801001)	Built database tables
Sahil J. Chaudhari (111801054)	Built schema diagram
Vishesh Munjal (111801055)	Built schema diagram
Harsh Parihar (111801015)	Built database tables

Task 3:

Abdullah khan (111801001)	Updated SQL FILE/ DATA Insertion
Sahil J. Chaudhari (111801054)	Updated SQL FILE/ DATA Insertion
Vishesh Munjal (111801055)	Updated SQL FILE/ DATA Insertion
Harsh Parihar (111801015)	DATA Insertion

Task 4:

Abdullah khan (111801001)	Brainstorming over views
Sahil J. Chaudhari (111801054)	Creation of views
Vishesh Munjal (111801055)	Brainstorming over views
Harsh Parihar (111801015)	Brainstorming over views

Task 5:

Abdullah khan (111801001)	Brainstorming over functions and procedures
Sahil J. Chaudhari (111801054)	Creation of functions and procedures
Vishesh Munjal (111801055)	Brainstorming over functions and procedures
Harsh Parihar (111801015)	Brainstorming over functions and procedures

Task 6:

Abdullah khan (111801001)	Brainstorming over sql and backend integration
Sahil J. Chaudhari (111801054)	Creation of Backend
Vishesh Munjal (111801055)	Creation of Frontend
Harsh Parihar (111801015)	Brainstorming over frontend.

Final phase:

Abdullah khan (111801001)	Preparation of presentation.
Sahil J. Chaudhari (111801054)	Brainstorming and creation of events.
Vishesh Munjal (111801055)	Brainstorming and creation of Triggers.
Harsh Parihar (111801015)	Brainstorming over demo.

Requirement specification and Purpose:

In any hospital, there are two types of patients, that is Out patient who come for OPD and In patients who are admitted for some treatment. However, for any patient, hospital management asks for some basic information which then stores it in the records of the hospital database. Such basic information includes the patient's name, sex, address, contact. Hence there will be two entities in the hospital management system named In patient and Out patient and both will content this basic information as attributes and Out patient visits for a particular time in a day hence it will also contain a date as an attribute. Since In patient is admitted to the hospital for more than one day, it will have a date of admission and date of discharge in the patient's attribute. As we are building a hospital management system, each entity must contain a primary key as an identity and hence each patient will be having a patient ID in an attribute. This will help to retrieve information about any patient at any time by simply using a patient ID in the database.

The patient is like a customer to the hospital and all doctors, nurses, and the rest of the staff are employees. Hence another crucial entity in management will be the employee. Every employee also has basic information and hence this information will be part of the employee's attributes named as name, salary, contact number, address, email id, and sex and employee id must be the primary key of this entity. Since there are various department inside hospital which include both medical department as well as management department such as finance department, cleaning department etc. and each employee is part of one department and hence employee entity has total participation and many to one relation with department entity which has its unique department ID as primary key, department name and head of department as it's attributes.

Another major part of the hospital is a doctor, both in-patient and out-patient will be consulting with the doctor and each doctor is also an employee of the hospital. Hence doctor entity is related with employee entity but the doctor will also have additional information such as qualification, specialization and every doctor also has their own charges and hence these details also must contain in doctor entity. Also, every doctor has a particular room allocated to, room number must be included in the doctor entity. As the doctor is also a strong entity, it has the doctor's id as the primary key. As an employee, the doctor shares one to one relation with the employee entity as every doctor posses a unique employee id. As both in patient and out patient consults with the doctor they both share a relationship where each patient must have at least one doctor and each doctor has multiple patients, it must be total participation of patient with the doctor. Using patient ID, we can get doctor assigned to that person and location of doctor in hospital. So in case of emergencies faster communication is feasible.

All patients in the hospital use various facilities such as sonography, X-ray, test laboratory, etc. Since hospitals have multiple rooms and sections, the facilities entity will have Its unique ID as the primary key, name of facility, charges, and room number it allocated to and both patients will share many to many relation with the facility entity as the patient may refer to multiple facilities and multiple patients opt for the same facility.

When a patient gets admitted, the patient gets allocated to the room that is under the supervision of one or more nurses. Since hospital contains various types of rooms such as general ward rooms, ICU rooms, deluxe rooms, non-AC rooms etc. and their price also varies as well as they can be at different floor hence room entity will contain unique ID as primary key and contain room type, floor of room, type of ward i.e ICU, emergency, general etc., number of beds in it and charge for bed in it as entities. Since every in patient must have room, patient have total participation relation with room entity with bed ID as attribute to relation i.e allocated. Using room entity, in database one can easily find out availability of beds at specific ward by searching with room ID and it also helps to estimate bill of patient. All rooms are under supervision of one or more nurses and hence nurse has total participation with room and share many to many relation as one nurse can also supervise particular patient in other room. Each nurse will have unique ID and also experience level as it helps to assign nurse to respected critical wards. As nurse is also an employee, entity nurse shares one to one relation with employee entity as each nurse has unique nurse ID and unique employee ID.

As appointment with doctor happens doctor note downs some symptoms and signs of patient as well as give prescription to out patient and doctor also regularly visits admitted patient and note downs condition and also refers to medication and tests that come in facilities. Even nurse also observes patient and write down record but this record is then push into database by a receptionist who manages all these records as well as patient’s visits, appointments scheduling etc. Hence record entity contains unique record number, date of record , appointment id, and description of appointment as attributes. These records help doctors and hospital management to understand behaviour and history of any patient. Record entity has many to many relation with receptionist entity which has unique ID. receptionist entity has one to one relationship with employee entity because of unique employee ID and receptionist ID. Both patient entities must have total one to many participation with record entity because patient visits multiple times to hospital or admitted for multiple days and each record have unique patient linked with. So using patient ID, one can see into that patient’s records.

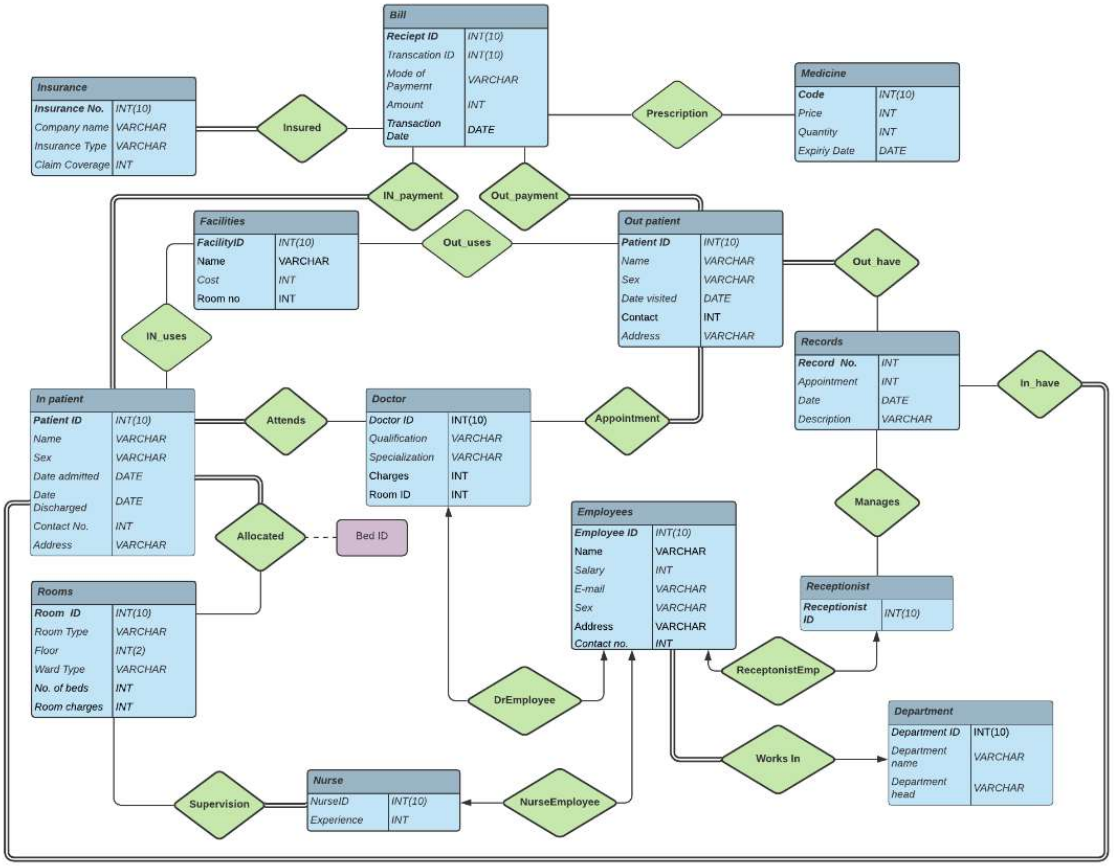
At last after appointment or during treatment of admitted patient, both have to pay hospital bills. Some patient also posses medical insurance that covers there cost and apart from hospital facilities and doctor charges, patient also have to pay for medication. Also for those who are not paying via insurance, they pay via cash, debit card or credit card and also it not necessary that appointment date must match with transaction date and sometimes one only go to buy medication without consulting with

doctor, hence bill entity must contain unique receipt ID as primary key, transaction ID for online or card payments, mode of payment that can be cash, online, debit card, credit card or insurance, total amount and transaction date. Insurance entity must contains Insurance number which is unique, company insuring, insurance type, and claim coverage according to user plans. The insurance entity shares total participation one to many with bill entity as insurance number is unique and can cover multiple bills of patient. Every hospital contains medical which has medicines with there unique code and price, hospital management also have to keep track of quantity of medication available and expiry dates in order to order medications. In bill, biller will look up in medicine entity and will calculate amount of medication according to their availability and since one medicine can be given to multiple patients and multiple medicines can be given to patient hence bill entity and medicine entity shares many to many relation. Since It is not necessary that every visit to hospital will make buy medicine, medicine entity does not have total participation with bill entity.

Benefits of such system will make job of everyone easier as database management system is fast to look and search about particular entry using unique ID or primary key of entity and relational sets. For example in order to calculate bill for particular patient, using patient ID biller will know which doctor patient consulted with using doctor entity and record entity, it will also get prices of medications via medicine entity and prices of facilities via facilities entity using patient ID, and also price of room from room ID i.e linked with patient entity and number of days admitted by dates inside patient entity. Using several instruction, biller can get accurate bill which may be insured by insurance using insurance entity. This will make clear and transparent system management system.

E-R diagram of the hospital management system:

Using requirement specification and purpose of the management system, followings are entity sets and relational set and diagram.



E-R diagram for hospital management system

Entity Sets:

Sr. no	Entity set	Attributes	Description
1	**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.
2	**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.
3	**Receptionist**	ReceptionistID	stores the data required for unique identification of all the receptionists present in the hospital.
4	**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.
5	**Rooms**	RoomID RoomType Floor WardType No. of Beds Room charges	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.

6	**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.
7	**Outpatient**	PatientID Name Sex DateVisited 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.
8	**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.
9	**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.
10	**Medicine**	Code Price Quantity, ExpiryDate	keeps the inventory of all the medicines available as well as their prices for billing purposes.

11	**Facilities**	FacilityID FacilityType Cost RoomNo	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.
12	**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.
13	**Department**	DepartmentID, DepartmentName DepartmentHead	is the record of all the departments and their current heads.

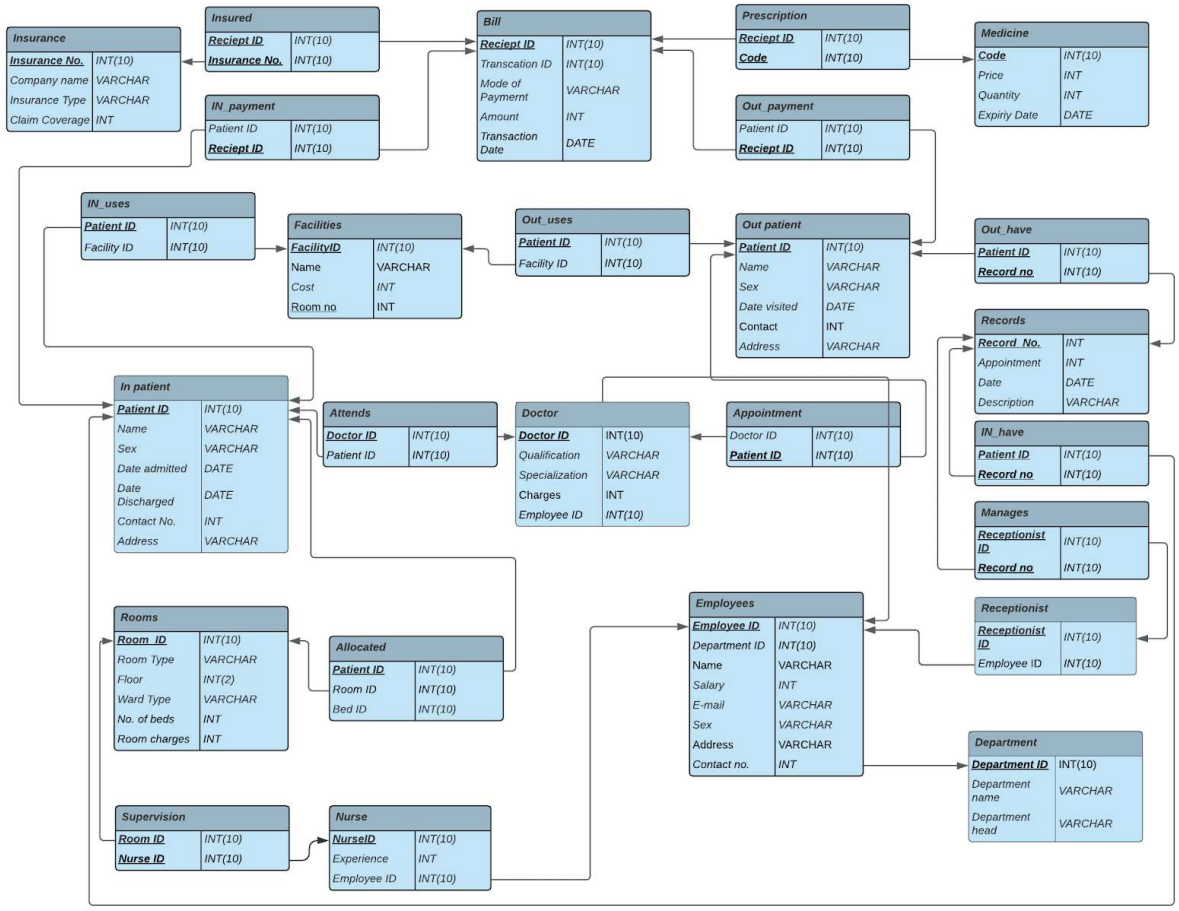
Relationship Sets:

Sr. no	Relationship set	Entity related	Description
1	**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 maps to one employeeID.
2	**NurseEmployee**	Nurse, Employee	is a one to one mapping between NurseID and their EmpoyeeID. It is similar to above described relation
3	**ReceptionistEmp**	Receptionist, Employee	is a one to one mapping between ReceptionistID and their EmpoyeeID. It is similar to above described relation
4	**Works In**	Employee, Department	stores which department each employee works in.Its a

			many-to-one relationship, with every employee working in some department.
5	**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 relationship 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.
6	**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 relation because over time any nurse may attend to any room, while every nurse will be responsible for some room.
7	**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.
8	**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.
9	**InHave/OutHave**	In/OutPatients, 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.
10	**Appointment**	Doctor, OutPatient	is the relationship similar to attends but for the outPatient type.
11	**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.
12	**Prescription**	Bill, Medicine	stores which medicine was used in a treatment billed with the ReceiptID in order to calculate the medical expense.
13	**Insured**	Insurance, Bill	relates which insurance payment was related to which bill. Every insurance waiver needs to relate to a valid bill.
14	**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.

Schema diagram of the hospital management system:



Schema diagram for hospital management system

A database schema, along with primary key and foreign-key constraints, can be depicted by schema diagrams. Figure above shows the schema diagram for our Hospital management system. Each relation appears as a box, with the relation 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 built the schema diagram based on the E-R diagram mentioned above and updated it for better performance and to avoid redundancy in model. Here we have a total 30 tables in which 13 are entity sets and 17 are relational sets.

Database based on above schema diagram is built using Mariadb where we first created a database using the CREATE function and then created a table using the CREATE TABLE command and added attributes to it with their type mentioned in schema diagram and mentioned primary keys and foreign keys.

Database creation of the hospital management system using Mariadb:

So below is a .sql file that we used to create hospital management database inside Mariadb using the following command.

```
sahil@sahil-Predator-PH315-51:~/Documents/DBMS/project$ sudo mysql < hospital.sql
```

hospital.sql file: Added in Appendix

Below is a quick view of hospital_management database.

```
MariaDB [hospital_management]> show tables;
+-------------------------------+
| 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                   |
+-------------------------------+
24 rows in set (0.001 sec)
```

Data tables in hospital_management database using MariaDB

Data insertion in hospital management system using Mariadb:

So below is a .sql file that we used to insert data inside the hospital management database inside Mariadb using the following command.

```
~/Documents/DBMS/project$ sudo mysql hospital_management < data.sql
```

data.sql file: Added in Appendix

Key parts of hospital_management database:

```
MariaDB [hospital_management]> select * from InPatient;
+------------+--------+------+---------------+-----------------+---------+-------------+-----------+
| Patient_ID | Name   | Sex  | Date_Admitted | Date_Discharged | Contact | Address     | Doctor_ID |
+------------+--------+------+---------------+-----------------+---------+-------------+-----------+
|      21100 | John   | M    | 2021-03-28    | 2021-04-03      | 9902143 | Mumbai      |      1417 |
|      21101 | Taylor | M    | 2021-03-28    | 2021-04-01      | 9905169 | Mumbai      |      1442 |
|      21102 | Ovi    | M    | 2021-03-31    | 2021-04-17      | 9912349 | Delhi       |      1443 |
|      21103 | James  | M    | 2021-04-01    | 2021-04-03      | 8348355 | Gandhinagar |      1441 |
|      21104 | Maria  | F    | 2021-04-02    | 2021-04-05      | 2015733 | Mumbai      |      1444 |
+------------+--------+------+---------------+-----------------+---------+-------------+-----------+
5 rows in set (0.001 sec)
```

Above is a screenshot of the InPatient table.

```
MariaDB [hospital_management]> select * from OutPatient;
+------------+--------+------+--------------+---------+---------+------------+
| Patient_ID | Name   | Sex  | Date_Visited | Contact | Address | Doctor_ID |
+------------+--------+------+--------------+---------+---------+------------+
|      31212 | Spock  | M    | 2021-04-01   | 9546546 | Mumbai  |      1417 |
|      31213 | Saurav | M    | 2021-04-02   | 9512345 | Kalyan  |      1443 |
|      31214 | Raj    | M    | 2021-04-02   | 9578901 | Thane   |      1443 |
|      31215 | lara   | F    | 2021-04-02   | 9514226 | Nashik  |      1445 |
|      31216 | sofia  | F    | 2021-04-03   | 1432345 | Pune    |      1443 |
+------------+--------+------+--------------+---------+---------+------------+
5 rows in set (0.001 sec)
```

Above is a screenshot of the OutPatient table.

```
MariaDB [hospital_management]> select * from (select Patient_ID, Name from InPatient) as i natural jo
in InPayment natural join Bill;
+------------+------------+--------+----------------+-----------------+--------+------------------+
| Reciept_ID | Patient_ID | Name   | Transaction_ID | Mode_of_Payment | Amount | Transaction_Date |
+------------+------------+--------+----------------+-----------------+--------+------------------+
|        100 |      21100 | John   |          10001 | Credit Card     |  75000 | 2021-04-03       |
|        101 |      21101 | Taylor |          10002 | Insurance       |  95000 | 2021-04-01       |
|        102 |      21102 | Ovi    |          10003 | Debit Card      | 195000 | 2020-04-17       |
|        103 |      21103 | James  |          10004 | Insurance       |  35000 | 2021-04-03       |
|        104 |      21104 | Maria  |          10005 | Insurance       |  78000 | 2021-04-05       |
+------------+------------+--------+----------------+-----------------+--------+------------------+
5 rows in set (0.001 sec)
```

Above is a screenshot of the InPatient table with its respective bill columns attached.

```
MariaDB [hospital_management]> select * from (select Patient_ID, Name from OutPatient) as i natural j
oin OutPayment natural join Bill;
+------------+------------+--------+----------------+-----------------+--------+------------------+
| Reciept_ID | Patient_ID | Name   | Transaction_ID | Mode_of_Payment | Amount | Transaction_Date |
+------------+------------+--------+----------------+-----------------+--------+------------------+
|        201 |      31212 | Spock  |          10006 | Cash            |   3150 | 2021-04-01       |
|        202 |      31213 | Saurav |          10007 | Online          |   1200 | 2021-04-02       |
|        203 |      31214 | Raj    |          10008 | Cash            |   1800 | 2021-04-02       |
|        204 |      31215 | lara   |          10009 | Online          |   1400 | 2021-04-02       |
|        205 |      31216 | sofia  |          10010 | Debit Card      |   1550 | 2021-04-03       |
+------------+------------+--------+----------------+-----------------+--------+------------------+
5 rows in set (0.001 sec)
```

Above is a screenshot of the OutPatient table with its respective bill columns attached.

```
MariaDB [hospital_management]> select * from (select Patient_ID, Name from InPatient) as i natural jo
in InHave natural join Records;
+------------+-----------+--------+------------+-----------------------------------+
| Patient_ID | Record_No | Name   | Date       | Description                       |
+------------+-----------+--------+------------+-----------------------------------+
|      21100 |         1 | John   | 2021-03-28 | Stomach pain, Intestinal infection |
|      21100 |         2 | John   | 2021-03-31 | Fever                             |
|      21100 |         3 | John   | 2021-04-01 | Vomit                             |
|      21101 |         5 | Taylor | 2021-03-28 | Heart attack                      |
|      21102 |         4 | Ovi    | 2021-03-31 | COVID-19 positive                 |
|      21103 |         6 | James  | 2021-04-01 | Ear surgery                       |
|      21104 |         7 | Maria  | 2021-04-02 | Blood cancer                      |
+------------+-----------+--------+------------+-----------------------------------+
7 rows in set (0.001 sec)

MariaDB [hospital_management]> select * from (select Patient_ID, Name from OutPatient) as i natural j
oin OutHave natural join Records;
+------------+-----------+--------+------------+------------------+
| Patient_ID | Record_No | Name   | Date       | Description      |
+------------+-----------+--------+------------+------------------+
|      31212 |         8 | Spock  | 2021-04-01 | Stomach Pain     |
|      31213 |         9 | Saurav | 2021-04-02 | Diabetes checkup |
|      31214 |        10 | Raj    | 2021-04-02 | Thyroid          |
|      31215 |        11 | lara   | 2021-04-02 | Migraine         |
|      31216 |        12 | sofia  | 2021-04-03 | Viral infection  |
+------------+-----------+--------+------------+------------------+
5 rows in set (0.001 sec)
```

Above is a screenshot of the InPatient and OutPatient table with its respective Records columns attached using relational table InHave and OutHave.

```
MariaDB [hospital_management]> select * from (select Patient_ID, Name from InPatient) as i natural jo
in InUses natural join Facilities;
+-------------+------------+--------+---------------+------+--------+
| Facility_ID | Patient_ID | Name   | Facility_name | Cost | RoomNo |
+-------------+------------+--------+---------------+------+--------+
|        1014 |      21100 | John   | Labroratory   | 1500 |     11 |
|        1013 |      21100 | John   | UltraSound    |  300 |     20 |
|        1016 |      21101 | Taylor | ECG           | 1000 |     13 |
|        1014 |      21102 | Ovi    | Labroratory   | 1500 |     11 |
|        1014 |      21104 | Maria  | Labroratory   | 1500 |     11 |
+-------------+------------+--------+---------------+------+--------+
5 rows in set (0.001 sec)

MariaDB [hospital_management]> select * from (select Patient_ID, Name from OutPatient) as i natural j
oin OutUses natural join Facilities;
+-------------+------------+--------+---------------+------+--------+
| Facility_ID | Patient_ID | Name   | Facility_name | Cost | RoomNo |
+-------------+------------+--------+---------------+------+--------+
|        1013 |      31212 | Spock  | UltraSound    |  300 |     20 |
|        1014 |      31213 | Saurav | Labroratory   | 1500 |     11 |
|        1014 |      31214 | Raj    | Labroratory   | 1500 |     11 |
|        1014 |      31216 | sofia  | Labroratory   | 1500 |     11 |
+-------------+------------+--------+---------------+------+--------+
4 rows in set (0.001 sec)
```

Above is a screenshot of the InPatient and OutPatient table with its respective Facilities columns attached using relational table InUses and OutUses.

```
MariaDB [hospital_management]> select * from (select Patient_ID, Name from InPatient) as i natural join Allocated natural join Rooms natural join Supervision natural join Nurse;
+----------+---------+------------+-------+--------+--------------+-------+-----------+------------+--------------+-------------+------------+
| Nurse_ID | Room_ID | Patient_ID | Name  | Bed_ID | Room_Type    | Floor | Ward_Type | No_of_Beds | Room_Charges | Employee_ID | Experience |
+----------+---------+------------+-------+--------+--------------+-------+-----------+------------+--------------+-------------+------------+
|     1551 |    6113 |      21104 | Maria |     13 | OneStar      |     0 | General   |         15 |          500 |        1101 |          5 |
|     1512 |    6114 |      21102 | Ovi   |      1 | OneStar nonAC |     1 | Private   |          1 |         1000 |        1000 |          7 |
|     1513 |    6115 |      21100 | John  |      1 | OneStar AC   |     1 | Private   |          1 |         1000 |        1001 |          2 |
|     1513 |    6115 |      21103 | James |      1 | OneStar AC   |     1 | Private   |          1 |         1000 |        1001 |          2 |
|     1514 |    6116 |      21100 | John  |      2 | Critical     |     1 | ICU       |         10 |        10000 |        1002 |          3 |
|     1514 |    6116 |      21101 | Taylor |      3 | Critical     |     1 | ICU       |         10 |        10000 |        1002 |          3 |
|     1514 |    6116 |      21102 | Ovi   |     10 | Critical     |     1 | ICU       |         10 |        10000 |        1002 |          3 |
|     1515 |    6118 |      21101 | Taylor |      1 | TwoStar      |     2 | Private   |          1 |         2500 |        1003 |          8 |
+----------+---------+------------+-------+--------+--------------+-------+-----------+------------+--------------+-------------+------------+
8 rows in set (0.001 sec)
```

Above is a screenshot of the InPatient with its respective Rooms and nurse allocation columns attached using relational table Allocated and Supervision.

```
MariaDB [hospital_management]> select * from Insurance;
+--------------+--------------+----------------+----------------+
| InsuranceNo  | CompanyName  | InsuranceType  | ClaimCoverage  |
+--------------+--------------+----------------+----------------+
|         1001 | starlite     | Health         |        1000000 |
|         1002 | ICICI        | Life           |        1000000 |
|         1003 | HDFC         | Motor          |         100000 |
|         1004 | HDFC         | Property       |        5000000 |
+--------------+--------------+----------------+----------------+
4 rows in set (0.000 sec)
```

Above is a screenshot of the Insurance table.

```
MariaDB [hospital_management]> select * from Medicine;
+--------+--------+----------+-------------+
| Code   | Price  | Quantity | Expr_Date   |
+--------+--------+----------+-------------+
| 10201  |     50 |      195 | 2022-01-12  |
| 10202  |    100 |      105 | 2022-05-01  |
| 10203  |   1000 |       25 | 2022-12-01  |
| 10204  |    780 |       15 | 2022-11-01  |
| 10205  |     10 |       50 | 2022-08-01  |
| 10206  |   2350 |       43 | 2022-07-21  |
| 10207  |     18 |       18 | 2022-06-30  |
+--------+--------+----------+-------------+
7 rows in set (0.000 sec)
```

Above is a screenshot of the Medicine table.

```
MariaDB [hospital_management]> select * from Employees natural join Department;
+---------------+-------------+-----------+--------+----------------------+-----+---------+-----------+-----------------+-----------------+
| Department_ID | Employee_ID | Name      | Salary | E_mail               | Sex | Contact | Address   | Department_name | Department_head |
+---------------+-------------+-----------+--------+----------------------+-----+---------+-----------+-----------------+-----------------+
|         10100 |        1000 | Shweta    |  20000 | shweta@gmail.com     | F   | 9985433 | Palakkad  | Gastrology      | Ramalingam      |
|         10100 |        2007 | Priya     | 200000 | priya@gmail.com      | F   | 9912234 | Mumbai    | Gastrology      | Ramalingam      |
|         10100 |        3003 | Ramalingam |  30000 | Ramalingam@gmail.com | M   | 9745343 | Delhi     | Gastrology      | Ramalingam      |
|         11000 |        3000 | Sonu      |  30000 | sonu@gmail.com       | M   | 9128243 | Delhi     | Reception       | Arushi          |
|         11000 |        3004 | Arushi    |  30000 | Arushi@gmail.com     | F   | 9000000 | Delhi     | Reception       | Arushi          |
|         11000 |        3111 | Sweety    |  30000 | sweety@gmai.com      | F   | 9557709 | Mumbai    | Reception       | Arushi          |
|         11001 |        3002 | Farhan    |  30000 | Farhan@gmail.com     | M   | 9167843 | Delhi     | EYE             | Farhan          |
|         11002 |        1003 | Emma      |  55000 | Emma@gmail.com       | F   | 6941847 | Mumbai    | Cardiology      | Albert          |
|         11002 |        2002 | Albert    | 100000 | albert@gmail.com     | M   | 6254697 | Chandigarh | Cardiology      | Albert          |
|         11003 |        1001 | Isabella  |  30000 | Isabells@gmail.com   | F   | 7845616 | Mumbai    | Internal Medicine | Sunita          |
|         11003 |        2003 | Sunita    |  60000 | SunitaMishra@gmail.com | F   | 6854684 | Gaziabad  | Internal Medicine | Sunita          |
|         11003 |        2004 | Vivek     |  90000 | Vivek@gmail.com      | M   | 9856744 | Goa       | Internal Medicine | Sunita          |
|         11004 |        2005 | Victoria  | 120000 | Victoria123@gmail.com | F   | 9867235 | Pune      | Neurology       | Victoria        |
|         11005 |        1002 | Jenny     |  35000 | Jenny@gmail.com      | F   | 8561894 | Mumbai    | GS              | Appel           |
|         11005 |        2006 | Appel     |  80000 | Appel1@gmail.com     | M   | 8754698 | Pune      | GS              | Appel           |
|         11100 |        1101 | Shyam     |  20000 | shyam@gmail.com      | M   | 9923445 | Mumbai    | ENT             | Phunsukh        |
|         11100 |        2001 | Abhishek  | 100000 | abhishek@gmail.com   | M   | 9999762 | Palakkad  | ENT             | Phunsukh        |
|         11100 |        3001 | Phunsukh  |  30000 | Phunsukh@gmail.com   | M   | 9122343 | Delhi     | ENT             | Phunsukh        |
+---------------+-------------+-----------+--------+----------------------+-----+---------+-----------+-----------------+-----------------+
18 rows in set (0.001 sec)
```

Above is a screenshot of the Employees with respective department tables.

```
MariaDB [hospital_management]> select * from Doctor;
+-----------+-------------+----------------+------------------+---------+
| Doctor_ID | Employee_ID | Qualification  | Specialization   | Charges |
+-----------+-------------+----------------+------------------+---------+
|      1417 |        2007 | MBBS           | Gastrologist     |     500 |
|      1441 |        2001 | MD             | ENT              |     500 |
|      1442 |        2002 | MBBS           | Cardiologist     |     500 |
|      1443 |        2003 | MD             | Physician        |     500 |
|      1444 |        2004 | MD             | Cancer           |     500 |
|      1445 |        2005 | MD             | Neurologist      |     500 |
|      1446 |        2006 | MD             | General surgeon  |     500 |
+-----------+-------------+----------------+------------------+---------+
7 rows in set (0.001 sec)
```

Above is a screenshot of the Doctor table.

```
MariaDB [hospital_management]> select * from Receptionist;
+------------------+-------------+
| Receptionist_ID | Employee_ID |
+------------------+-------------+
|             2122 |        3000 |
|             2222 |        3111 |
+------------------+-------------+
```

Above is a screenshot of the Receptionist table.

```
MariaDB [hospital_management]> select * from Receptionist natural inner join Manages natural inner join Records;
+-----------+----------------+-------------+------------+------------+-----------------------------------+
| Record_No | Receptionist_ID | Employee_ID | Patient_ID | Date       | Description                       |
+-----------+----------------+-------------+------------+------------+-----------------------------------+
|         1 |           2122 |        3000 |      21100 | 2021-03-28 | Stomach pain, Intestinal infection |
|         2 |           2122 |        3000 |      21100 | 2021-03-31 | Fever                             |
|         3 |           2122 |        3000 |      21100 | 2021-04-01 | Vomit                             |
|         4 |           2222 |        3111 |      21102 | 2021-03-31 | COVID-19 positive                 |
|         5 |           2122 |        3000 |      21101 | 2021-03-28 | Heart attack                      |
|         6 |           2122 |        3000 |      21103 | 2021-04-01 | Ear surgery                       |
|         7 |           2222 |        3111 |      21104 | 2021-04-02 | Blood cancer                      |
|         8 |           2122 |        3000 |      31212 | 2021-04-01 | Stomach Pain                      |
|         9 |           2222 |        3111 |      31213 | 2021-04-02 | Diabetes checkup                  |
|        10 |           2122 |        3000 |      31214 | 2021-04-02 | Thyroid                           |
|        11 |           2122 |        3000 |      31215 | 2021-04-02 | Migraine                          |
|        12 |           2222 |        3111 |      31216 | 2021-04-03 | Viral infection                   |
+-----------+----------------+-------------+------------+------------+-----------------------------------+
12 rows in set (0.001 sec)
```

Above is a screenshot of the Receptionist table with records tables using relational table manages.

Views in hospital management system using Mariadb:

In hospital management system we have created using ER-Diagram, schema and also made some changes according to the data flow and after successful insertion schema and data inside mariadb now we have created some views that will be helpful in easy access of data from more than one tables that we are going to use most frequently. Since view stores query that we have called and hence minimal space and its property of inheritance that is change in original data get reflected in view it is preferable to create important view and keep them in out database.

We have created some of view in hospital management database as follow:

1. **Doctor_name:** As in Employee table there are all employees present in hospital and it is important to keep list of doctors with their name and Doctor_ID and in Doctor table they only have their respective ID and Employee ID and Employees table have their names hence we created view called Doctor_name using joins.

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

This view is also used to create another views. This view also help user to identify different doctors available in hospital.

```
MariaDB [hospital_management]> select * from Doctor_name;
+-----------+----------+-----------------+
| Doctor_ID | Name     | Specialization  |
+-----------+----------+-----------------+
|      1417 | Priya    | Gastrologist    |
|      1441 | Abhishek | ENT             |
|      1442 | Albert   | Cardiologist    |
|      1443 | Sunita   | Physician       |
|      1444 | Vivek    | Cancer          |
|      1445 | Victoria | Neurologist     |
|      1446 | Appel    | General surgeon |
+-----------+----------+-----------------+
7 rows in set (0.001 sec)
```

2. **Patient_doctor:** In our database there are two types of patients i.e in patient and out patient and It is important to keep list of both patients together with their respective consulting doctor and since there are two different tables of patient with ID and name we will use union set operator with Doctor_names views created earlier using joins.

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

Using this view, users can view how many patients are there under certain doctor.

```
MariaDB [hospital_management]> select * from Patient_doctor;
+------------+--------------+-----------+-------------+
| Patient_ID | patient_name | Doctor_ID | doctor_name |
+------------+--------------+-----------+-------------+
|      21103 | James        |      1441 | Abhishek    |
|      21101 | Taylor       |      1442 | Albert      |
|      31216 | sofia        |      1443 | Sunita      |
|      31214 | Raj          |      1443 | Sunita      |
|      31213 | Saurav       |      1443 | Sunita      |
|      21102 | Ovi          |      1443 | Sunita      |
|      21104 | Maria        |      1444 | Vivek       |
|      31215 | lara         |      1445 | Victoria    |
|      31212 | Spock        |      1417 | Priya       |
|      21100 | John         |      1417 | Priya       |
+------------+--------------+-----------+-------------+
10 rows in set (0.002 sec)
```

3. **Record_log:** In our database there is Record table that keep record of both in and out patients and we can access them via InHave and OutHave relation table but it is very complex query we have to write every time without views and hence creating one using view we have created earlier will make it easier to reference in future.

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

Using this view user can access to history of any user in that hospital with respective consulting doctor.

```
MariaDB [hospital_management]> select * from Record_log;
+------------+--------------+-----------+-------------+-----------+------------+-----------------------------------+
| Patient_ID | patient_name | Doctor_ID | doctor_name | Record_No | Date       | Description                       |
+------------+--------------+-----------+-------------+-----------+------------+-----------------------------------+
|      21100 | John         |      1417 | Priya       |         1 | 2021-03-28 | Stomach pain, Intestinal infection |
|      21100 | John         |      1417 | Priya       |         2 | 2021-03-31 | Fever                             |
|      21100 | John         |      1417 | Priya       |         3 | 2021-04-01 | Vomit                             |
|      21102 | Ovi          |      1443 | Sunita      |         4 | 2021-03-31 | COVID-19 positive                 |
|      21101 | Taylor       |      1442 | Albert      |         5 | 2021-03-28 | Heart attack                      |
|      21103 | James        |      1441 | Abhishek    |         6 | 2021-04-01 | Ear surgery                       |
|      21104 | Maria        |      1444 | Vivek       |         7 | 2021-04-02 | Blood cancer                      |
|      31212 | Spock        |      1417 | Priya       |         8 | 2021-04-01 | Stomach Pain                      |
|      31213 | Saurav       |      1443 | Sunita      |         9 | 2021-04-02 | Diabetes checkup                  |
|      31214 | Raj          |      1443 | Sunita      |        10 | 2021-04-02 | Thyroid                           |
|      31215 | lara         |      1445 | Victoria    |        11 | 2021-04-02 | Migraine                          |
|      31216 | sofia        |      1443 | Sunita      |        12 | 2021-04-03 | Viral infection                   |
+------------+--------------+-----------+-------------+-----------+------------+-----------------------------------+
12 rows in set (0.002 sec)
```

4. **Nurse_schedule:** In our database we have assigned each patient room to particular nurse, it may have more than one nurse or nurse may assigned to more than one room and it is important to keep track of which nurse is assigned to which room and it can be done via rooms, supervision, nurse and employees tables which is very hectic to check every time via query so we have created this vies as following using joins.

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

This view will help user to identify which room is assigned to which nurse and may call respective nurse for help and nurse user get to know which rooms they have been assigned.

```
MariaDB [hospital_management]> select * from Nurse_schedule;
+-------------+----------+---------+---------------+-------+-----------+------------+--------------+------------+----------+
| Employee_ID | Nurse_ID | Room_ID | Room_Type     | Floor | Ward_Type | No_of_Beds | Room_Charges | Experience | Name     |
+-------------+----------+---------+---------------+-------+-----------+------------+--------------+------------+----------+
|        1000 |     1512 |    6114 | OneStar nonAC |     1 | Private   |          1 |         1000 |          7 | Shweta   |
|        1001 |     1513 |    6115 | OneStar AC    |     1 | Private   |          1 |         1000 |          2 | Isabella |
|        1002 |     1514 |    6116 | Critical      |     1 | ICU       |         10 |        10000 |          3 | Jenny    |
|        1003 |     1515 |    6118 | TwoStar       |     2 | Private   |          1 |         2500 |          8 | Emma     |
|        1101 |     1551 |    6113 | OneStar       |     0 | General   |         15 |          500 |          5 | Shyam    |
+-------------+----------+---------+---------------+-------+-----------+------------+--------------+------------+----------+
5 rows in set (0.093 sec)
```

5. **Medicine_log:** In hospital, doctors write multiple prescriptions to patient and it is very hectic to manage log of medicine for biller or chemist and it can be achieved by creating view that can give you medicine log with patient info and billing info with medicines given to him via join between Patient_doctor view, Bill, Prescription and Medicine tables.

```
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;
```

Using this view user can understand how many medicines given to patient and whom he is consulting with as well as billing information.

```
MariaDB [hospital_management]> select * from 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  |
+-------+-----------+------------+--------------+-----------+-------------+----------------+-----------------+--------+------------------+-------+----------+------------+
| 10201 |       100 |      21100 | John         |      1417 | Priya       |          10001 | Credit Card     |  75000 | 2021-04-03       |    50 |      195 | 2022-01-12 |
| 10201 |       104 |      21104 | Maria        |      1444 | Vivek       |          10005 | Insurance       |  78000 | 2021-04-05       |    50 |      195 | 2022-01-12 |
| 10201 |       201 |      31212 | Spock        |      1417 | Priya       |          10006 | Cash            |   3150 | 2021-04-01       |    50 |      195 | 2022-01-12 |
| 10201 |       203 |      31214 | Raj          |      1443 | Sunita      |          10008 | Cash            |   1800 | 2021-04-02       |    50 |      195 | 2022-01-12 |
| 10202 |       201 |      31212 | Spock        |      1417 | Priya       |          10006 | Cash            |   3150 | 2021-04-01       |   100 |      105 | 2022-05-01 |
| 10202 |       202 |      31213 | Saurav       |      1443 | Sunita      |          10007 | Online          |   1200 | 2021-04-02       |   100 |      105 | 2022-05-01 |
| 10202 |       205 |      31216 | sofia        |      1443 | Sunita      |          10010 | Debit Card      |   1550 | 2021-04-03       |   100 |      105 | 2022-05-01 |
| 10203 |       101 |      21101 | Taylor       |      1442 | Albert      |          10002 | Insurance       |  95000 | 2021-04-01       |  1000 |       25 | 2022-12-01 |
| 10203 |       102 |      21102 | Ovi          |      1443 | Sunita      |          10003 | Debit Card      | 195000 | 2020-04-17       |  1000 |       25 | 2022-12-01 |
| 10204 |       102 |      21102 | Ovi          |      1443 | Sunita      |          10003 | Debit Card      | 195000 | 2020-04-17       |   780 |       15 | 2022-11-01 |
| 10204 |       202 |      31213 | Saurav       |      1443 | Sunita      |          10007 | Online          |   1200 | 2021-04-02       |   780 |       15 | 2022-11-01 |
| 10205 |       203 |      31214 | Raj          |      1443 | Sunita      |          10008 | Cash            |   1800 | 2021-04-02       |    10 |       50 | 2022-08-01 |
| 10205 |       204 |      31215 | lara         |      1445 | Victoria    |          10009 | Online          |   1400 | 2021-04-02       |    10 |       50 | 2022-08-01 |
| 10206 |       100 |      21100 | John         |      1417 | Priya       |          10001 | Credit Card     |  75000 | 2021-04-03       |  2350 |       43 | 2022-07-21 |
| 10206 |       101 |      21101 | Taylor       |      1442 | Albert      |          10002 | Insurance       |  95000 | 2021-04-01       |  2350 |       43 | 2022-07-21 |
| 10206 |       102 |      21102 | Ovi          |      1443 | Sunita      |          10003 | Debit Card      | 195000 | 2020-04-17       |  2350 |       43 | 2022-07-21 |
| 10207 |       103 |      21103 | James        |      1441 | Abhishek    |          10004 | Insurance       |  35000 | 2021-04-03       |    18 |       18 | 2022-06-30 |
| 10207 |       202 |      31213 | Saurav       |      1443 | Sunita      |          10007 | Online          |   1200 | 2021-04-02       |    18 |       18 | 2022-06-30 |
+-------+-----------+------------+--------------+-----------+-------------+----------------+-----------------+--------+------------------+-------+----------+------------+
18 rows in set (0.004 sec)
```

Procedures in hospital management system using Mariadb:

In hospital management system we have created using ER-Diagram, schema and also made some changes according to the data flow and after successful insertion schema and data inside mariadb now we have created some views that will be helpful in easy access of data from more than one tables that we are going to use most frequently. We also have created some procedures that will help us in our management application. We have used procedures as all of our necessary dataflow we designed doesn't require any function that will return something, we are creating some interface that will make some additional modifications and creations of helper columns and tables for ease of transactions between queries.

Following are the procedures, our team has encountered.

Attendance:

In hospital management, one must know which employee is on duty and which one is not. E.g. While booking an appointment, the user must know which doctors are available and according to it, the user may book it. For doctors, it will be helpful to know which nurse is available and which one is not, so doctors can set patients and nurses to rooms according to the availability of nurses.

1. **init_attendance:**

   This procedure adds an insert column to the employee table which has a default value of absent that means everyone initially set as absent.

```
DELIMITER //
CREATE PROCEDURE init_attendance()
BEGIN
ALTER TABLE Employees ADD present varchar(10) NOT NULL DEFAULT('Absent');
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> call init_attendance();
Query OK, 0 rows affected (0.055 sec)

MariaDB [hospital_management]> select * from Employees;
+-------------+---------------+-----------+--------+-----------------------+------+---------+-----------+---------+
| Employee_ID | Department_ID | Name      | Salary | E_mail                | Sex  | Contact | Address   | present |
+-------------+---------------+-----------+--------+-----------------------+------+---------+-----------+---------+
|        1000 |         10100 | Shweta    |  20000 | shweta@gmail.com      | F    | 9985433 | Palakkad  | Absent  |
|        1001 |         11003 | Isabella  |  30000 | Isabells@gmail.com    | F    | 7845616 | Mumbai    | Absent  |
|        1002 |         11005 | Jenny     |  35000 | Jenny@gmail.com       | F    | 8561894 | Mumbai    | Absent  |
|        1003 |         11002 | Emma      |  55000 | Emma@gmail.com        | F    | 6941847 | Mumbai    | Absent  |
|        1101 |         11100 | Shyam     |  20000 | shyam@gmail.com       | M    | 9923445 | Mumbai    | Absent  |
|        2001 |         11100 | Abhishek  | 100000 | abhishek@gmail.com    | M    | 9999762 | Palakkad  | Absent  |
|        2002 |         11002 | Albert    | 100000 | albert@gmail.com      | M    | 6254697 | Chandigarh| Absent  |
|        2003 |         11003 | Sunita    |  60000 | SunitaMishra@gmail.com| F    | 6854684 | Gaziabad  | Absent  |
|        2004 |         11003 | Vivek     |  90000 | Vivek@gmail.com       | M    | 9856744 | Goa       | Absent  |
|        2005 |         11004 | Victoria  | 120000 | Victoria123@gmail.com | F    | 9867235 | Pune      | Absent  |
|        2006 |         11005 | Appel     |  80000 | Appel1@gmail.com      | M    | 8754698 | Pune      | Absent  |
|        2007 |         10100 | Priya     | 200000 | priya@gmail.com       | F    | 9912234 | Mumbai    | Absent  |
|        3000 |         11000 | Sonu      |  30000 | sonu@gmail.com        | M    | 9128243 | Delhi     | Absent  |
|        3001 |         11100 | Phunsukh  |  30000 | Phunsukh@gmail.com    | M    | 9122343 | Delhi     | Absent  |
|        3002 |         11001 | Farhan    |  30000 | Farhan@gmail.com      | M    | 9167843 | Delhi     | Absent  |
|        3003 |         10100 | Ramalingam|  30000 | Ramalingam@gmail.com  | M    | 9745343 | Delhi     | Absent  |
|        3004 |         11000 | Arushi    |  30000 | Arushi@gmail.com      | F    | 9000000 | Delhi     | Absent  |
|        3111 |         11000 | Sweety    |  30000 | sweety@gmai.com       | F    | 9557709 | Mumbai    | Absent  |
+-------------+---------------+-----------+--------+-----------------------+------+---------+-----------+---------+
18 rows in set (0.000 sec)
```

All rows of employee table has present status as absent.

2. **mark_attendance:**

This procedure takes employee ID as input and marks that employee with ID as present. So whenever any employee e.g doctor or nurse comes to work then they can login to application and call over mark_attendance procedure using some backend that will change their status from absent to present.

```
DELIMITER //
CREATE PROCEDURE mark_attendance(IN ID int(10))
BEGIN
IF (ID in (SELECT Employee_ID FROM Employees))
THEN UPDATE Employees SET present = 'Present' WHERE Employee_ID=ID;
END IF;
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> call mark_attendance(1000);
Query OK, 1 row affected (0.131 sec)
```

```
MariaDB [hospital_management]> call mark_attendance(1003);
Query OK, 1 row affected (0.123 sec)
```

```
MariaDB [hospital_management]> select * from Employees;
+-------------+---------------+------------+--------+-----------------------+------+---------+------------+---------+
| 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  |
|        1003 |         11002 | Emma       |  55000 | Emma@gmail.com        | F    | 6941847 | Mumbai     | Present |
|        1101 |         11100 | Shyam      |  20000 | shyam@gmail.com       | M    | 9923445 | Mumbai     | Absent  |
|        2001 |         11100 | Abhishek   | 100000 | abhishek@gmail.com    | M    | 9999762 | Palakkad   | Absent  |
|        2002 |         11002 | Albert     | 100000 | albert@gmail.com      | M    | 6254697 | Chandigarh | Absent  |
|        2003 |         11003 | Sunita     |  60000 | SunitaMishra@gmail.com| F    | 6854684 | Gaziabad   | Absent  |
|        2004 |         11003 | Vivek      |  90000 | Vivek@gmail.com       | M    | 9856744 | Goa        | Absent  |
|        2005 |         11004 | Victoria   | 120000 | Victoria123@gmail.com | F    | 9867235 | Pune       | Absent  |
|        2006 |         11005 | Appel      |  80000 | Appel1@gmail.com      | M    | 8754698 | Pune       | Absent  |
|        2007 |         10100 | Priya      | 200000 | priya@gmail.com       | F    | 9912234 | Mumbai     | Absent  |
|        3000 |         11000 | Sonu       |  30000 | sonu@gmail.com        | M    | 9128243 | Delhi      | Absent  |
|        3001 |         11100 | Phunsukh   |  30000 | Phunsukh@gmail.com    | M    | 9122343 | Delhi      | Absent  |
|        3002 |         11001 | Farhan     |  30000 | Farhan@gmail.com      | M    | 9167843 | Delhi      | Absent  |
|        3003 |         10100 | Ramalingam |  30000 | Ramalingam@gmail.com  | M    | 9745343 | Delhi      | Absent  |
|        3004 |         11000 | Arushi     |  30000 | Arushi@gmail.com      | F    | 9000000 | Delhi      | Absent  |
|        3111 |         11000 | Sweety     |  30000 | sweety@gmai.com       | F    | 9557709 | Mumbai     | Absent  |
+-------------+---------------+------------+--------+-----------------------+------+---------+------------+---------+
18 rows in set (0.001 sec)
```

We mark employees with ID 1000 and 1003 present.

3. **drop_attendance:**

This procedure takes employee ID as input and changes its present column status to absent from present. Whenever an employee is off work, it will change their status to absent using a backend of application that will help other users e.g patients will know that when a doctor is not available or when he is available.

```
DELIMITER //
CREATE PROCEDURE drop_attendance(IN ID int(10))
BEGIN
IF (ID in (SELECT Employee_ID FROM Employees))
```

```
THEN UPDATE Employees SET present = 'Absent' WHERE Employee_ID=ID;
END IF;
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> CALL drop_attendance(1000);
Query OK, 1 row affected (0.134 sec)

MariaDB [hospital_management]> select * from Employees;
+-------------+---------------+------------+--------+-----------------------+------+---------+------------+---------+
| Employee_ID | Department_ID | Name       | Salary | E_mail                | Sex  | Contact | Address    | present |
+-------------+---------------+------------+--------+-----------------------+------+---------+------------+---------+
|        1000 |         10100 | Shweta     |  20000 | shweta@gmail.com      | F    | 9985433 | Palakkad   | Absent  |
|        1001 |         11003 | Isabella   |  30000 | Isabells@gmail.com    | F    | 7845616 | Mumbai     | Absent  |
|        1002 |         11005 | Jenny      |  35000 | Jenny@gmail.com       | F    | 8561894 | Mumbai     | Absent  |
|        1003 |         11002 | Emma       |  55000 | Emma@gmail.com        | F    | 6941847 | Mumbai     | Present |
|        1101 |         11100 | Shyam      |  20000 | shyam@gmail.com       | M    | 9923445 | Mumbai     | Absent  |
|        2001 |         11100 | Abhishek   | 100000 | abhishek@gmail.com    | M    | 9999762 | Palakkad   | Absent  |
|        2002 |         11002 | Albert     | 100000 | albert@gmail.com      | M    | 6254697 | Chandigarh | Absent  |
|        2003 |         11003 | Sunita     |  60000 | SunitaMishra@gmail.com| F    | 6854684 | Gaziabad   | Absent  |
|        2004 |         11003 | Vivek      |  90000 | Vivek@gmail.com       | M    | 9856744 | Goa        | Absent  |
|        2005 |         11004 | Victoria   | 120000 | Victoria123@gmail.com | F    | 9867235 | Pune       | Absent  |
|        2006 |         11005 | Appel      |  80000 | Appel1@gmail.com      | M    | 8754698 | Pune       | Absent  |
|        2007 |         10100 | Priya      | 200000 | priya@gmail.com       | F    | 9912234 | Mumbai     | Absent  |
|        3000 |         11000 | Sonu       |  30000 | sonu@gmail.com        | M    | 9128243 | Delhi      | Absent  |
|        3001 |         11100 | Phunsukh   |  30000 | Phunsukh@gmail.com    | M    | 9122343 | Delhi      | Absent  |
|        3002 |         11001 | Farhan     |  30000 | Farhan@gmail.com      | M    | 9167843 | Delhi      | Absent  |
|        3003 |         10100 | Ramalingam |  30000 | Ramalingam@gmail.com  | M    | 9745343 | Delhi      | Absent  |
|        3004 |         11000 | Arushi     |  30000 | Arushi@gmail.com      | F    | 9000000 | Delhi      | Absent  |
|        3111 |         11000 | Sweety     |  30000 | sweety@gmai.com       | F    | 9557709 | Mumbai     | Absent  |
+-------------+---------------+------------+--------+-----------------------+------+---------+------------+---------+
18 rows in set (0.001 sec)
```

On calling drop attendance we marked employee with ID 1000 as absent.

4. **init_appointment:**
   This procedure creates an appointment table that has doctor_ID, patient_ID, Date that patients want to have appointments and status of appointment i.e appointment done or pending. This table will help doctors to know how many appointments one has and he can manage the schedule accordingly.

```
DELIMITER //
CREATE PROCEDURE init_appointment()
BEGIN
CREATE TABLE appointment(Doctor_ID int(10),
Patient_ID int(10) NOT NULL PRIMARY KEY,
appt_date DATE,
status varchar(10) NOT NULL CHECK(status IN('Pending','Done')));
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> CALL init_appointment();
Query OK, 0 rows affected (0.665 sec)

MariaDB [hospital_management]> select * from appointment;
Empty set (0.001 sec)
```

```
MariaDB [hospital_management]> DESC appointment
    -> ;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Doctor_ID  | int(10)     | YES  |     | NULL    |       |
| Patient_ID | int(10)     | NO   | PRI | NULL    |       |
| appt_date  | date        | YES  |     | NULL    |       |
| status     | varchar(10) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.002 sec)
```

We have created an empty appointment table.

5. **add_appointment:**

   This procedure takes details of patient and ID of doctor to which one want to have appointment, Then this procedure generates new patient ID for that patient for that particular appointment and insert this details to OutPatient table and it also adds generated patientID and doctor ID along with date to appointment table and mark appointment status as pending.

   This will help ease the receptionist and doctor as each patient is getting a unique ID that will be used for further transactions and also for patient it will tell how many patients are there in the queue for waiting.

```
DELIMITER //
CREATE PROCEDURE add_appointment(
   IN Name VARCHAR(20),
   IN Sex VARCHAR(4),
   IN appt_date DATE,
   IN Contact INT(10),
   IN Address VARCHAR(30),
   IN Doctor_ID int(10))
BEGIN
SET @ID = (select Patient_ID from OutPatient order by Patient_ID DESC Limit 1)+1;
INSERT INTO OutPatient VALUES(@ID,Name,Sex,appt_date,Contact,Address,Doctor_ID);
INSERT INTO appointment VALUES(Doctor_ID,@ID,appt_date,'Pending');
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> CALL add_appointment('Sahil', 'M', '2021-03-28', 9902143, 'Nashik', 1417);
Query OK, 2 rows affected (0.176 sec)

MariaDB [hospital_management]> CALL add_appointment('Vishesh', 'M', '2021-03-28', 9905169, 'Karnal', 1417);
Query OK, 2 rows affected (0.176 sec)
```

```
MariaDB [hospital_management]> select * from OutPatient;
+------------+---------+------+--------------+---------+---------+-----------+
| Patient_ID | Name    | Sex  | Date_Visited | Contact | Address | Doctor_ID |
+------------+---------+------+--------------+---------+---------+-----------+
|      31212 | Spock   | M    | 2021-04-01   | 9546546 | Mumbai  |      1417 |
|      31213 | Saurav  | M    | 2021-04-02   | 9512345 | Kalyan  |      1443 |
|      31214 | Raj     | M    | 2021-04-02   | 9578901 | Thane   |      1443 |
|      31215 | lara    | F    | 2021-04-02   | 9514226 | Nashik  |      1445 |
|      31216 | sofia   | F    | 2021-04-03   | 1432345 | Pune    |      1443 |
|      31217 | Sahil   | M    | 2021-03-28   | 9902143 | Nashik  |      1417 |
|      31218 | Vishesh | M    | 2021-03-28   | 9905169 | Karnal  |      1417 |
+------------+---------+------+--------------+---------+---------+-----------+
7 rows in set (0.001 sec)

MariaDB [hospital_management]> select * from appointment;
+-----------+------------+------------+---------+
| Doctor_ID | Patient_ID | appt_date  | status  |
+-----------+------------+------------+---------+
|      1417 |      31217 | 2021-03-28 | Pending |
|      1417 |      31218 | 2021-03-28 | Pending |
+-----------+------------+------------+---------+
2 rows in set (0.001 sec)
```

We have added two patients with names sahil and vishesh and we can see that their status is pending and both want to visit doctor with ID 1417 and both are added to OutPatient with unique IDs.

6. **mark_appointment:**
   This procedure takes a doctor ID, patient ID and appointment date and if it exists in the appointment table then mark that appointment as done. This helps others to know how many patients have done appointments with the doctor and also the doctor gets to know how many patients one has done.

```
DELIMITER //
CREATE PROCEDURE mark_appointment(IN D_ID int(10), IN P_ID int(10),
in a_date Date)
BEGIN
IF((D_ID,P_ID,a_date) IN (SELECT Doctor_ID,Patient_ID,appt_date FROM appointment))
THEN
UPDATE appointment SET status = 'Done' WHERE Doctor_ID=D_ID AND Patient_ID=P_ID
AND appt_date=a_date;
END IF;
END;
//
DELIMITER ;
```

```
MariaDB [hospital_management]> CALL mark_appointment(1417,31217,'2021-03-28');
Query OK, 1 row affected (0.141 sec)

MariaDB [hospital_management]> SELECT * FROM appointment;
+-----------+------------+------------+---------+
| Doctor_ID | Patient_ID | appt_date  | status  |
+-----------+------------+------------+---------+
|      1417 |      31217 | 2021-03-28 | Done    |
|      1417 |      31218 | 2021-03-28 | Pending |
+-----------+------------+------------+---------+
2 rows in set (0.000 sec)
```

We have marked the appointment of the patient with ID 31217 as done.

7. **admit_patient:**
   This procedure takes out patient ID and as well as predicted discharged date that can be changed later and admit it to hospital and adds its entries to In patient and also assigns him new In patient ID. This procedure, the Doctor will call when he wants the patient to be admitted, rest room allocation and billing section will be handled by respective employees.

```
DELIMITER //
CREATE PROCEDURE admit_patient(IN P_ID int(10),IN discharge_date DATE)
BEGIN
IF(P_ID in (select Patient_ID from OutPatient))
THEN
SET @ID = (select Patient_ID from InPatient order by Patient_ID DESC Limit 1)+1;
INSERT INTO InPatient SELECT @ID,Name,Sex,Date_Visited,discharge_date,Contact,
Address,Doctor_ID FROM OutPatient WHERE Patient_ID = P_ID;
END IF;
END;
//
DELIMITER;
```

```
MariaDB [hospital_management]> CALL admit_patient(31217,'2021-03-30');
Query OK, 1 row affected (0.149 sec)

MariaDB [hospital_management]> select * from InPatient;
+------------+--------+------+--------------+-----------------+---------+-------------+-----------+
| Patient_ID | Name   | Sex  | Date_Admitted | Date_Discharged | Contact | Address     | Doctor_ID |
+------------+--------+------+--------------+-----------------+---------+-------------+-----------+
|      21100 | John   | M    | 2021-03-28    | 2021-04-03      | 9902143 | Mumbai      |      1417 |
|      21101 | Taylor | M    | 2021-03-28    | 2021-04-01      | 9905169 | Mumbai      |      1442 |
|      21102 | Ovi    | M    | 2021-03-31    | 2021-04-17      | 9912349 | Delhi       |      1443 |
|      21103 | James  | M    | 2021-04-01    | 2021-04-03      | 8348355 | Gandhinagar |      1441 |
|      21104 | Maria  | F    | 2021-04-02    | 2021-04-05      | 2015733 | Mumbai      |      1444 |
|      21105 | Sahil  | M    | 2021-03-28    | 2021-03-30      | 9902143 | Nashik      |      1417 |
+------------+--------+------+--------------+-----------------+---------+-------------+-----------+
6 rows in set (0.001 sec)
```

Here we can see that we have added patient Sahil as InPatient with new unique In patient ID. These are the procedures that we thought will be most useful in implementation of databases to the backend of our hospital management application.

Similar to above procedures we have created procedures for insertion of bill entry, record entry and facility that have been included in appendix i.e google drive link.

Triggers in hospital management system using Mariadb:

Since there were multiple integrity constraints we have included and we have also added procedures there were very few changes in the database that are requiring triggers. Using triggers insertion, update and deletion validation can be done automatically. We have used triggers to avoid invalid data entry in our database.

1. **record_check:** In the record check trigger, we are applying it on the records table before insertion, we are checking whether the inserted date for record is valid or not. If patient is InPatient i.e admitted patient then record date must be between admitted date and discharged date and if patient is OutPatient then record date must be appointment date and if record date is not valid then we sending error message to console.

```
DROP TRIGGER IF EXISTS record_check;
delimiter $$
CREATE TRIGGER record_check BEFORE INSERT ON Records
FOR EACH ROW
BEGIN
IF EXISTS (SELECT Patient_ID from InPatient where Patient_ID = NEW.Patient_ID) THEN
SET @date_admit = (SELECT  Date_Admitted from InPatient where Patient_ID = NEW.Patient_ID);
SET @date_discharge = (SELECT Date_Discharged from InPatient where Patient_ID =
NEW.Patient_ID);
IF NOT (NEW.Date BETWEEN @date_admit AND @date_discharge) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR:
          DATE is not between';
END IF;
END IF;

IF EXISTS (SELECT Patient_ID from OutPatient where Patient_ID = NEW.Patient_ID) THEN
SET @date_visited = (SELECT Date_Visited from OutPatient where Patient_ID =
NEW.Patient_ID);
IF NOT (NEW.Date = @date_visited) THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR:
          DATE is not okay';
END IF;
END IF;
END; $$
delimiter ;
```

```
MariaDB [hospital_management]> insert into Records values (13,31216,'2021-04-04','Headache');
ERROR 1644 (45000): ERROR:
           DATE is not okay
MariaDB [hospital_management]> insert into Records values (13,21100,'2021-03-27','Migraine');
ERROR 1644 (45000): ERROR:
           DATE is not between
MariaDB [hospital_management]> insert into Records values (13,21100,'2021-04-04','Migraine');
ERROR 1644 (45000): ERROR:
           DATE is not between
```

In above screenshot, we tried to insert record for OutPatient with ID 31216 and for InPatient with ID 21100, but since date of visit was 3rd April 2021 for 31216, any other date than this was invalid and for 21100 valid date is between 2021-03-28 and 2021-04-03 and both 2021-03-27 and 2021-04-04 aren't. Hence we received an error.

```
MariaDB [hospital_management]> select * from OutPatient where Patient_ID = 31216;
+------------+-------+------+--------------+---------+---------+-----------+
| Patient_ID | Name  | Sex  | Date_Visited | Contact | Address | Doctor_ID |
+------------+-------+------+--------------+---------+---------+-----------+
|      31216 | sofia | F    | 2021-04-03   | 1432345 | Pune    |      1443 |
+------------+-------+------+--------------+---------+---------+-----------+
1 row in set (0.001 sec)

MariaDB [hospital_management]> select * from InPatient where Patient_ID = 21100;
+------------+------+------+---------------+-----------------+---------+---------+-----------+
| Patient_ID | Name | Sex  | Date_Admitted | Date_Discharged | Contact | Address | Doctor_ID |
+------------+------+------+---------------+-----------------+---------+---------+-----------+
|      21100 | John | M    | 2021-03-28    | 2021-04-03      | 9902143 | Mumbai  |      1417 |
+------------+------+------+---------------+-----------------+---------+---------+-----------+
1 row in set (0.001 sec)
```

But when we inserted a valid date, the record was inserted.

```
MariaDB [hospital_management]> insert into Records values (13,21100,'2021-04-02','Migraine');
Query OK, 1 row affected (0.132 sec)
```

```
|         12 |      31216 | 2021-04-03 | Viral infection                      |
|         13 |      21100 | 2021-04-02 | Migraine                             |
+------------+------------+------------+--------------------------------------+
13 rows in set (0.001 sec)
```

2. **attendance_toggle_Records:** In the attendance toggle records trigger, we will apply it on Manages table on after insertion. In this trigger, we will check whether the receptionist is present or not while inserting a record in the Records table. As whenever the receptionist is inserting a record in the Records table, a new entry is also inserted in the Manages table via procedure insert_record(). If the receptionist is not present then we are making that receptionist present.

```
DROP TRIGGER IF EXISTS attendence_toggle_Records;
delimiter $$
CREATE TRIGGER attendence_toggle_Records AFTER INSERT ON Manages
FOR EACH ROW
BEGIN
UPDATE Employees
SET present = 'Present'
WHERE Employee_ID = (SELECT Employee_ID from Receptionist where Receptionist_ID =
NEW.Receptionist_ID);
END; $$
delimiter ;
```

```
MariaDB [hospital_management]> select Employee_ID, Receptionist_ID, present from Receptionist natural inner join Employees;
+------------+----------------+---------+
| Employee_ID | Receptionist_ID | present |
+------------+----------------+---------+
|        3000 |           2122 | Absent  |
|        3001 |           2123 | Absent  |
|        3002 |           2124 | Absent  |
|        3003 |           2125 | Absent  |
|        3004 |           2126 | Absent  |
|        3111 |           2222 | Absent  |
+------------+----------------+---------+
6 rows in set (0.001 sec)
```

We can see here that all receptionists are absent.

```
MariaDB [hospital_management]> call insert_record(3000,31217,'2021-05-07','Fever');
Query OK, 3 rows affected (0.173 sec)
```

Now we are inserting a record for a patient with ID 31217 via receptionist with Receptionist ID 3000. As a receptionist were absent, a trigger got activated and made that receptionist present.

```
MariaDB [hospital_management]> select Employee_ID, Receptionist_ID, present from Receptionist natural inner join Employees;
+-------------+-----------------+---------+
| Employee_ID | Receptionist_ID | present |
+-------------+-----------------+---------+
|        3000 |            2122 | Present |
|        3001 |            2123 | Absent  |
|        3002 |            2124 | Absent  |
|        3003 |            2125 | Absent  |
|        3004 |            2126 | Absent  |
|        3111 |            2222 | Absent  |
+-------------+-----------------+---------+
```

```
|          13 |       21100 | 2021-04-02 | Migraine                                 |
|          14 |       31217 | 2021-05-07 | Fever                                    |
+-------------+-------------+------------+------------------------------------------+
14 rows in set (0.001 sec)
```

And also record got inserted for 31217.

```
|              2222 |          12 |
|              2122 |          14 |
+-------------------+-------------+
13 rows in set (0.001 sec)
```

Entry for receptionist with receipt no 14 got inserted in Manages.

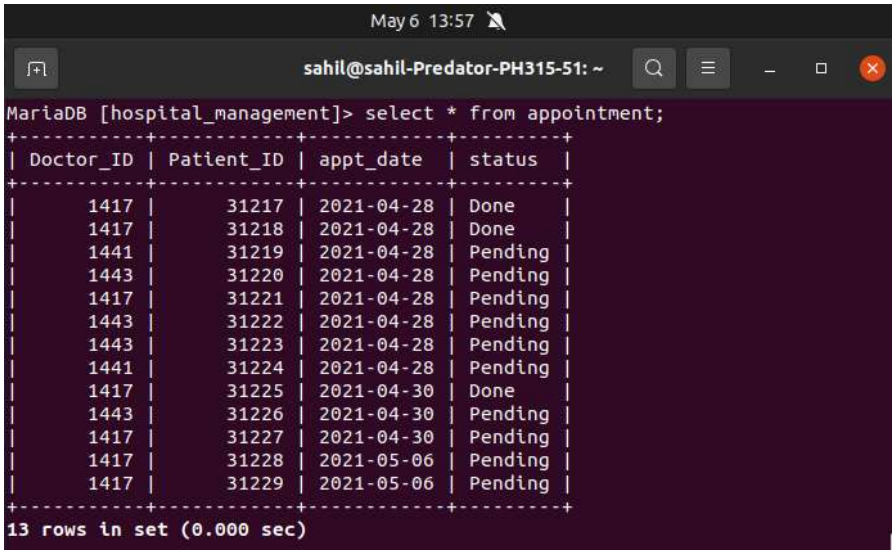
Events in hospital management system using Mariadb:

In our hospital management system, we have included some events that get triggered and some queries as events are triggered.

1. **Day_reset:** In day reset, we are triggering our event on 23 hours 59 minutes and 59 seconds each day starting from 1st may. In this event, at the end of day, we are removing those appointments that are marked as done and we are also shifting the date of all appointments that are not done i.e pending to the next date. E.g if say a patient with ID 12312 had an appointment with a doctor with ID 2007 on date 6th may, 2021 but it hasn't happened so at the end of 6th may we will shift the patient's appointment to next day i.e 7th may, 2021. So the doctor or receptionist doesn't have to manually clean and update the appointment log.

```
delimiter //
CREATE EVENT day_reset
   ON SCHEDULE EVERY 1 DAY
           STARTS '2021-05-01 23:59:59'
   DO
       BEGIN
           DELETE FROM appointment
           WHERE status = 'Done';
           SET @date = curdate();
           SET @newdate = DATE_ADD(@date,INTERVAL 1 DAY);
           UPDATE appointment
           SET appt_date = @newdate
           WHERE status = 'Pending' and appt_date <= @date;

       END //
delimiter ;
```

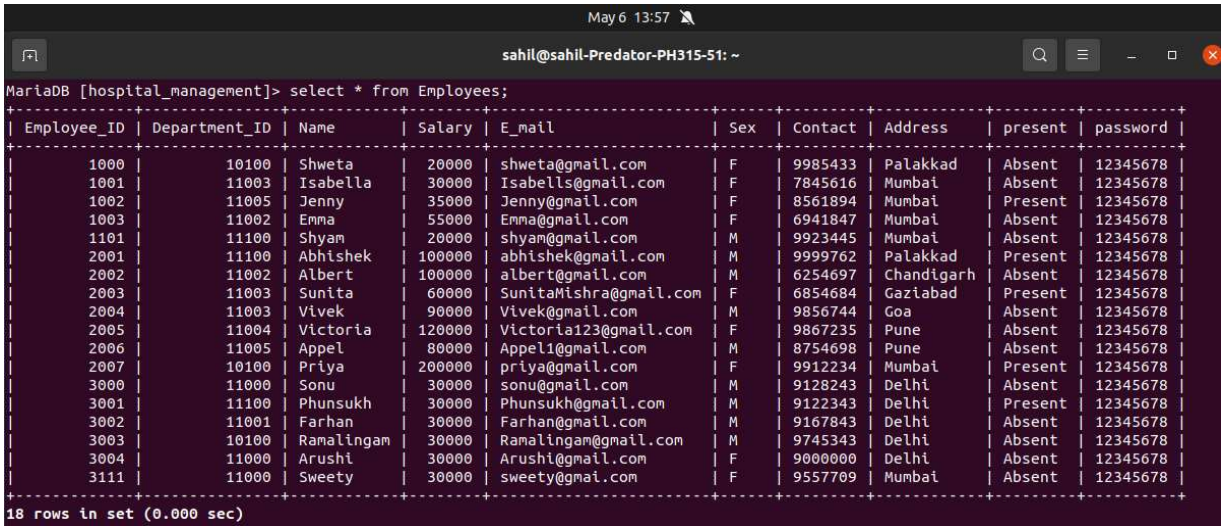


In above screenshot, we update our trigger time to 2 pm for demo, and you can see all the appointments that were done are removed and dates of all appointments that are pending were shifted to next day i.e 7th may in below screenshot.

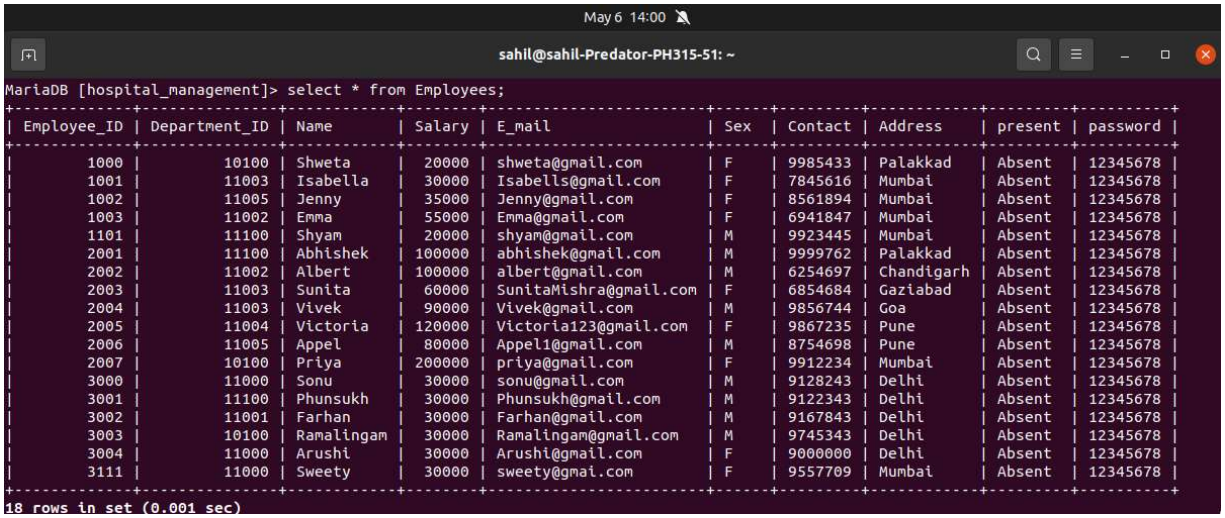
```
MariaDB [hospital_management]> select * from appointment;
+-----------+------------+------------+---------+
| Doctor_ID | Patient_ID | appt_date  | status  |
+-----------+------------+------------+---------+
|      1441 |      31219 | 2021-05-07 | Pending |
|      1443 |      31220 | 2021-05-07 | Pending |
|      1417 |      31221 | 2021-05-07 | Pending |
|      1443 |      31222 | 2021-05-07 | Pending |
|      1443 |      31223 | 2021-05-07 | Pending |
|      1441 |      31224 | 2021-05-07 | Pending |
|      1443 |      31226 | 2021-05-07 | Pending |
|      1417 |      31227 | 2021-05-07 | Pending |
|      1417 |      31228 | 2021-05-07 | Pending |
|      1417 |      31229 | 2021-05-07 | Pending |
+-----------+------------+------------+---------+
10 rows in set (0.000 sec)
```

2. **toggle:** In event toggle, we are triggering our event at the end of each shift in our hospital management system, i.e first shift 8:00 to 16:00, second shift 16:00 to 24:00 and night shift 00:00 to 8:00. It will mark all present employees as absent at end of each shift. So hospital management will not have to worry about attendance of their employees.

```
delimiter //
CREATE EVENT toggle
   ON SCHEDULE
       EVERY 8 HOUR
           STARTS '2021-05-06 08:00:00'
   DO
       BEGIN
           UPDATE Employees
           SET present = 'Absent';
       END //
delimiter ;
```

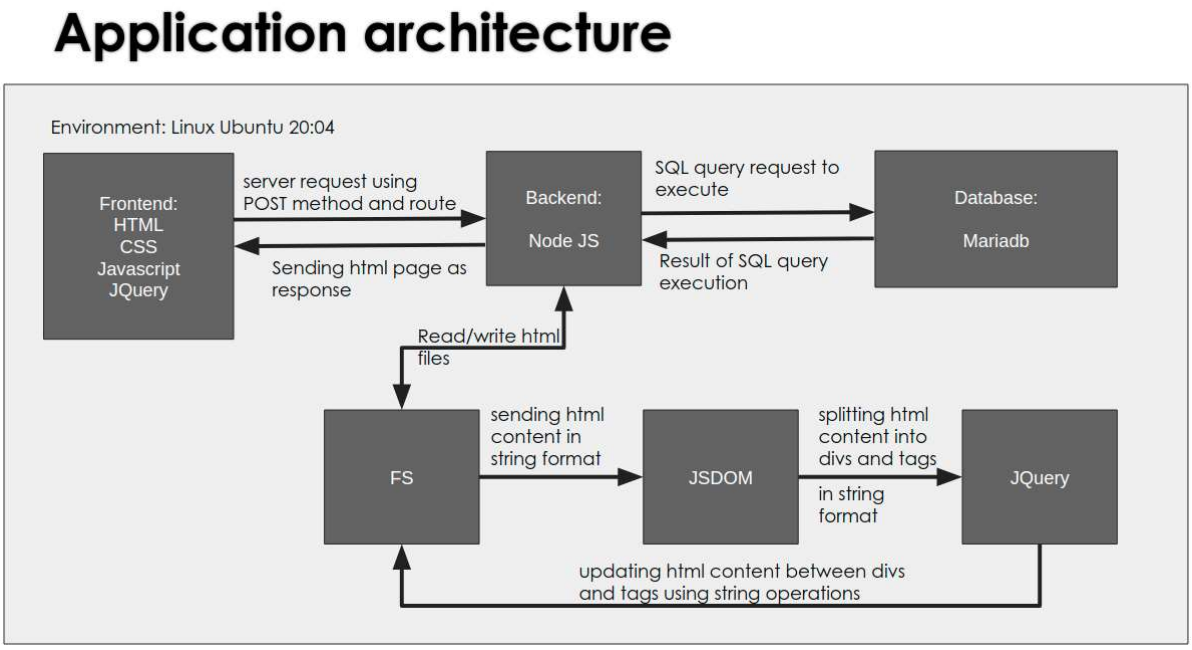


In above screenshot, we can see that there are several employees that are present at time 13:57 and for demo, we have set event trigger time at 14:00, hence when 14:00 struct, event got triggered and all employees reset back to absent status as shown in below screenshot.



Web application development:

We have used HTML, CSS, Jquery, Javascript and Twitter bootstrap for our frontend and NodeJS and Mariadb for our backend.



Frontend:

HTML, CSS:

HTML defines the structure of web page application and CSS provides visual graphics to it to beautify web pages. Without them, a web page cannot be build.

JQuery and JavaScript:

Jquery and javascript are used to interact with a web page, automate some things, create events and triggers that will take place when the user clicks some buttons or loads a webpage.

Twitter bootstrap:

Twitter bootstrap provides ready to use templates that make web pages attractive that already have predefined css classes. It helps developers to write less code.

Backend:

Node JS:

Node JS is a server side language that uses javascript language and it has a very easy to learn interface to process server side tasks and it also works very well with the MYSQL

database. NodeJS also has vast modules and plugins that give developers a wide variety of methods to approach and process tasks.

MariaDB:

MariaDB is our database server which is our heart of the project. It will store our hospital management database as well as will run all the queries and events and triggers.

```
const express = require('express')
const http = require('http');
const bodyParser = require('body-parser');
const fs = require('fs');
var mysql = require('mysql');
const app = express();
const port = 4000;
var con = mysql.createConnection({
 host: "localhost",
 user: "root",
 password: "password",
 database: "hospital_management"
});

app.listen(port, () => {
 console.log(`application listening at http://localhost:${port}`)
});

con.connect(function(err) {
 if (err) throw err;
 console.log("Connected!");
});
```

Above, First we are importing required modules and then we are creating a connection con with Mariadb server using module mysql.createConnection function in which we are providing host, user,password and database which con will be connected with. Con.connect is making connections. Var app is an application that will run the entire backend and it is an express instance and app will listen through port = 4000.

```
app.get('/plogin',(req,res)=>{
 res.sendFile('src/Patient_login.html',{root:"."})
})
```

app.get takes route i.e /plogin and on calling this route, the app will send file patient_login.html as a response to the server.

```
<form class="navbar-form navbar-right" action='/plogin'>
<button type="submit" class="btn btn-warning">Patient login</button>
</form>
```

Here we can see that in HTML we can call over this route via action attribute.

```
app.post('/facility', function(req,res){
```

```
 sql_q = 'select * from Facilities natural join ';
 sql_q += '(select * from InUses union select * from OutUses) as u where
Patient_ID = '
 sql_q += Patient_id.toString() + ';';
 con.query(sql_q,
   function(err,result,fields){
     update_facility_file(result);
     res.redirect('/patient');
 });
});
```

In above example, app.post is called by html when any button is clicked using withing form tags with method post and action will be route. Above route is /facility.

```
<form method="POST" action=""></form>
```

In this case req.body will give a JSON file that will have the value of all form elements inside the form tag. Res.redirect will take the route on which our app wants to be redirected. So /patient routes routes to Patient.html file and it will be loaded whenever the event is triggered.

Problem Faced:

NodeJS is a server side language and it does not provide any interface that can directly change HTML of HTML files, and another workaround was to create ejs files that can be rendered through nodeJS and it allow to change HTML between it but since our HTML files were lengthy hence ejs file also wasn't a good option.

Solution:

We have used following NodeJS modules:

1. fs: fs provides an interface that can read and write a variety of files.
2. Jsdom: Jsdom read HTML files and split it into HTML divs and tags in form of strings.
3. Jquery: Jquery can access specific tag or div and can change its content using string operations.

Example:

```
function raise_success_admit(pid){
 html_file = 'src/doctor.html';
 sql_q = 'select Patient_ID from InPatient order by Patient_ID DESC limit 1;';
 con.query(sql_q,
   function(err,result,fields){
     out = Object.values(JSON.parse(JSON.stringify(result)));
     newID = out[0].Patient_ID;
     fs.readFile(html_file, 'utf8', function(error, data) {
       const dom = new jsdom.JSDOM(data);
       const jquery = require('jquery')(dom.window);
       code = '\n<p>Patient with id '+pid.toString()+' admitted successfully ';
       code += 'new InPatient ID is '+newID.toString()+'</p>\n';
       jquery("#success_admit").empty();
       jquery("#success_admit").append(code);
       fs.writeFile(html_file, dom.serialize(), err => {
```

```
            // console.log('err raised success_admitfully');
        });
        });
    });
}
```

SO in the above example first we ran sql query using con.query that takes a sql query and a function that gives error and result. Then we are reading doctor.html file using fs.readFile and parsing HTML file via JSDOM.jsdom and then splitting parsed html into pieces using jquery. Then we are creating a string that we want to replace in an HTML file and then we are giving an id to jQuery that removes everything between tags with that id and then appends a function to insert our code within those tags. Finally we are overwriting the html file with our modified and edited HTML content back.

File Structure:

1. **src:** This directory contains all html files of our hospital management app.
2. **css:** This directory contains a css file.
3. **img:** This directory contains all image files that we are using for background and for doctor profile images on mainpage.html
4. **node_modules:** This directory contains all required node modules and files to run our application.
5. **app.js:** This is our main NodeJS file that takes care of the entire application.
6. **package.json:** This file configures our nodeJS files.

Screenshots of our web application:

Screenshots for a web application are here.

1. **Mainpage.html:**
   this our main page of web application. Here we have two tabs, one is main page which will redirect to mainpage via route /main and second one is appointment which will redirect to appointment.html page via route /appt and there are two buttons at right hand side that will take user to patient login and employee login respectively.

2. **Appointment.html:** this our appointment page where it is asking all necessary form fill ups to fill, for doctors we are only allowing those doctors for appointment who are present in hospital. Above tabs are the same as in the mainpage.

3. **Success.html:** This page displays success msg that also gives the user its patient login ID for login.

4. **Employee_login.html:** This page ask for employee ID and password and matched it via sql query and if succeed then redirect employee to one of following .html file according to ID: doctor, nurse, receptionist.

5. **Doctor.html:** This file shows employee details of doctor and toggle switch that employees can use to mark their attendance. When they leave they will toggle their attendance to Absent. Doctors also can view respective appointments and can mark their respective appointments as done and also can admit patients that visited the doctor. Each employee can log out whenever they want and the webpage will be redirected to the home page.

6. **Nurse.html:** This file also shows employee details and features similar to doctors. Nurses can view their schedule i.e which room they are assigned to supervise.

7. **Receptionist.html:** This file also shows employee details and features similar to doctors. Receptionists can insert billing, facility details and record details about patients by inserting patient ID.

8. **Patient_login.html:** Similar to employee login, here web page asks for patient ID and if valid patient ID is entered then it will be redirected to patient.html

9. **Patient.html:** Patient can view their basic information as well as appointment status and other details such as medication, room allocation, facility details and bill. Patients will also be provided with a logout button that will redirect users to the mainpage.

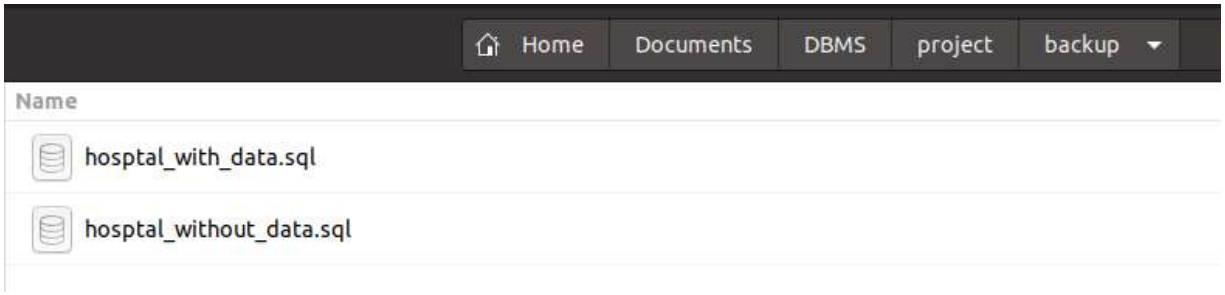
Appendix

This is a link to google drive to access hospital.sql and data.sql file that we used to create a database along with a backup folder which consists of a backup of the database. It also contains our web application files along with all necessary .sql files.
https://drive.google.com/drive/folders/11XhI_cjLnTq3b6Ay3Ls-LkFiB5sfuyFO?usp=sharing

Backup of hospital management database:

```
sahil@sahil-Predator-PH315-51:~/Documents/DBMS/project/backup$ sudo mysqldump -u root hospital_manage
ment > hosptal_with_data.sql;
sahil@sahil-Predator-PH315-51:~/Documents/DBMS/project/backup$ sudo mysqldump -u root --no-data hospi
tal_management > hosptal_without_data.sql;
```

We have created backup using commands mentioned above and below is screenshot of directory that stores backup files.



hospital.sql file: This file contains the structure of hospital management databases where first we are creating a database and then using it and then we are inserting queries to create database tables with primary keys, foreign keys and other integrity constraints.
data.sql file: This file uses datatables we created using hospital.sql file and adding data rows in it with maintaining integrity constraints.
Function.sql file: This file contains all necessary functions and procedures required for our hospital management project.
View.sql file: This file contains all necessary view required for our hospital management project.
event.sql file: This file contains all necessary events required for our hospital management project.
trigger.sql file: This file contains all necessary triggers required for our hospital management project.