HOSPITAL MANAGEMENT SYSTEM

UCS 310 Database Management System Project Report

END-Semester Evaluation

Submitted by:

(102117032) RIDHI THAKUR

(102117045) SHIYA MER

(102117052) HARNEET KAUR

(102117057) IRA GUPTA

BE Second Year, CSE

Submitted to:

ARCHANA SINGH

Computer Science and Engineering Department

TIET, Patiala

May 2023

INDEX

1. PROJECT OBJECTIVE :

Our project Hospital Database Management system includes registration of patients, storing their disease details into the system. It will also contain doctor's information and will digitalize the whole billing system. It has the facility to give a unique id for every patient and stores the details of every patient and staff automatically. It includes a search facility to know the current status of each room. User can search availability of a doctor and the details of a patient using the id. And the whole process conducted by Administrator.

Keywords- Hospital, Administrator, Patients, Doctor, Diseases, Staff, Treatments, Test, Lab reports, Schema.

1. BACKGROUND

A Hospital Database Management System (HDMS) is a computer or web based system that facilities managing the functioning of a hospital or any medical set up. This system will help in making the whole functioning paperless.

The hospital database includes all the necessary patient data. The disease history, test results, prescribed treatment can be accessed by doctors without much delay in order to make an accurate diagnosis and monitor the patient's health. It enables lower risks of mistakes.

A hospital is a place where Patients come up for general diseases. Hospitals Provide facilities like:

- Consultation by Doctors on Diseases.
- Diagnosis for diseases.
- Providing treatment facility.
- Facility for admitting Patients (providing beds, nursing, medicines etc.)
- Immunization for Patients/Children.

Various operational works that are done in a Hospital are:

- Recording information about the Patients that come.
- Generating bills.
- Recording information related to diagnosis given to Patients.
- Keeping record of the Immunization provided to Children/Patients.
- Keeping information about various diseases and medicines available to cure them.

These are the various jobs that need to be done in a Hospital by the operational staff and Doctors. All these works are done on papers.

The work is done as follows:

- Information about Patients is done by just writing the Patients name, age, and gender. Whenever the Patient comes up his information is stored freshly.
- Bills are generated by recording price for each facility provided to Patient on a separate sheet and at last they all are summed up.
- Diagnosis information to patients is generally recorded on the document, which contains Patients information. It is destroyed after some time period to decrease the paper load in the office.

- Immunization records of the children are maintained in pre-formatted sheets, which are kept in a file.
- Information about various diseases is not kept as any document. Doctors themselves do this job by remembering various medicines

All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and taken care of. Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them at that time.

2. IDEAS

The hospital database includes all the necessary patient data. The disease history, lab reports, prescribed treatment can be accessed by doctors without much delay in order to make an accurate diagnosis and monitor the patient's health. It enables lower risks of mistakes.

- The project maintains two levels of users:
- Administrator.
- User Level-Data Entry Operator.
- Now, I discuss the main facilities in this project are:
- Maintaining records of indoor/outdoor patients.
- Maintaining patient's test and examinations details.
- Providing different test facilities to a Doctor for doctor for diagnosis of a patients.
- Maintaining patient's prescription, medicine and diet advice details.
- Providing billing details for indoor/outdoor patients.
- Results of tests, prescription, precautions and diet advice will be automatically updated in the database.
- In this project collection of data in form different pathology labs.
- Related test reports, patient's details report, billing reports can be generated as per user requirements.

- User or administrator can search a patient's record by his id.
- Hospital Database Management System Design:

The Hospital database management system design is a database design use for managing hospital functions and events. It enables the admin to register a patient for the hospital, stores their disease details into the database. Any of the staff members, doctor & admin is able to add, view, edit, update or delete data.

- Purpose of Hospital Database Management System: The purpose of the Hospital Management System database Design is to make a secure and easy way of storing information of the patient, doctors, inpatient, outpatient, Rooms, and Bill payment.
- Features of the Hospital Database Management System: There are seven (8) common features of Hospital Management System Database Design such as Managing Administrator, Doctors, laboratory, Inpatient, Outpatient, Rooms, and Hospital Bills information.

]

Data Query Language (DQL)

SELECT-Used to retrieve certain records from one or more tables.

Data Manipulation Language (DML)

INSERT - Used to create a record UPDATE - Used to change certain records. DELETE - Used to delete certain records.

Data Definition Language (DDL)
CREATE - Used to create a new table, a view of a table, or other object in database.
ALTER - Used to modify an existing database object, such as a table.
DROP - Used to delete an entire table, a view of a table or other object in the database.

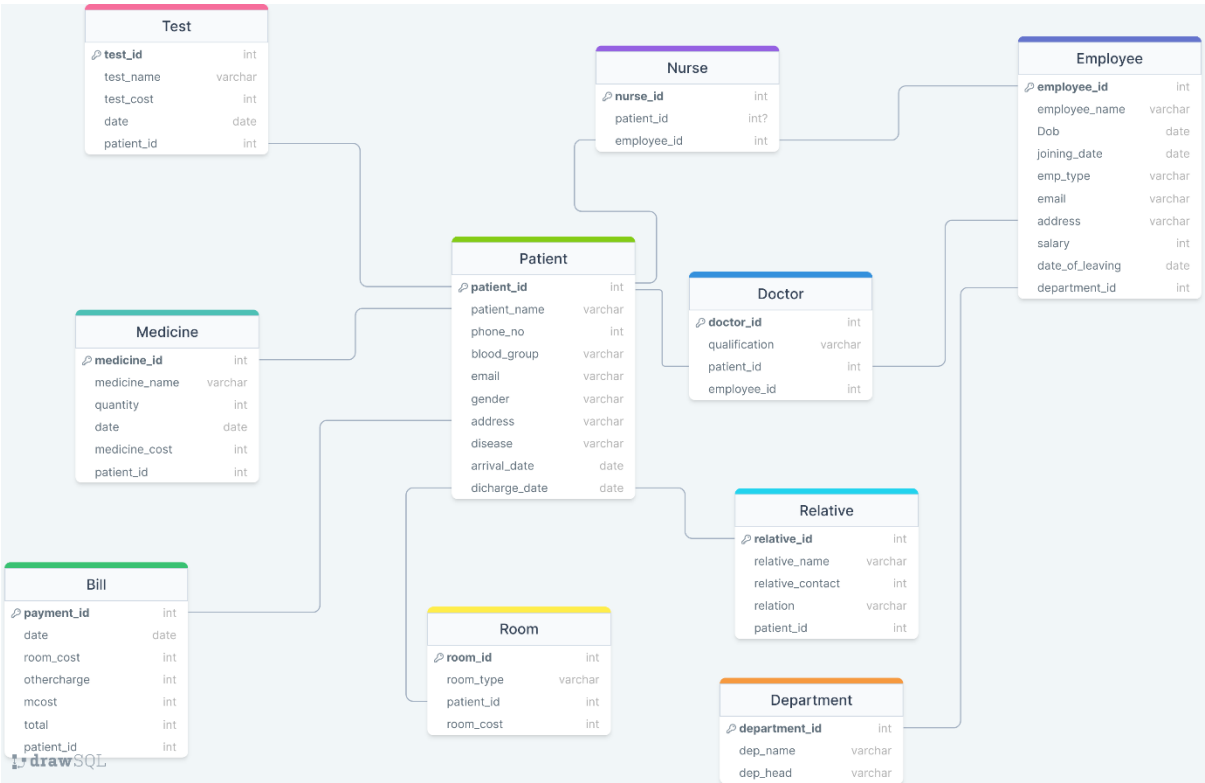
SOFTWARE REQUIREMENTS

Operating System	: 64bit WINDOWS Operating System, X64-based processor	
Database	: MYSQL	

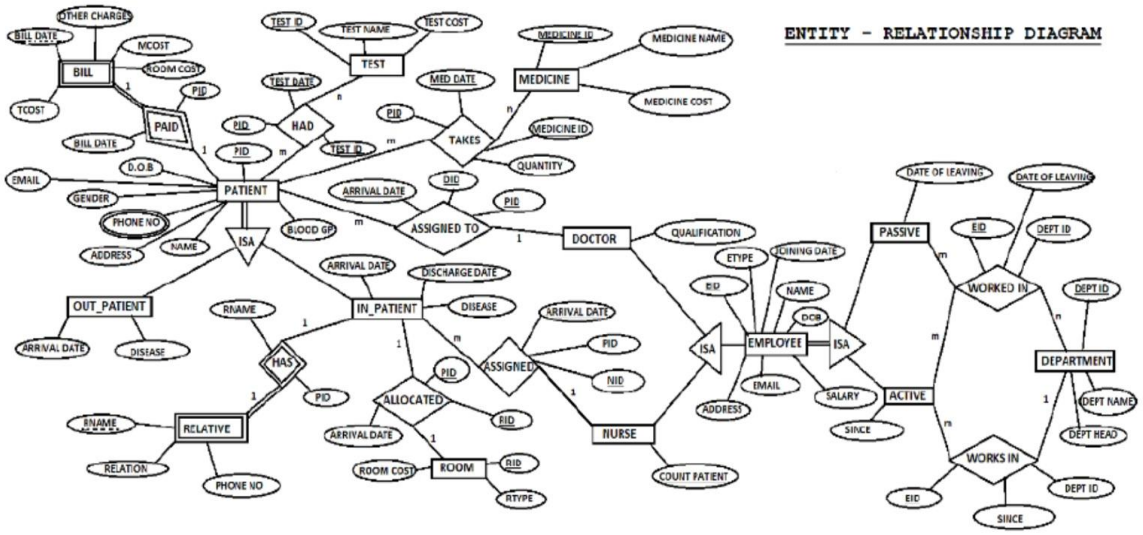
HARDWARE REQUIREMENTS

Processor	: Intel Celeron CPU N3060 @1.60GHz or Above
RAM	: 4.00 GB or Above
Hard Disk	: 1 TB
Compact Disk	: CD-ROM, CD-R, CD-RW
Input device	: Keyboard

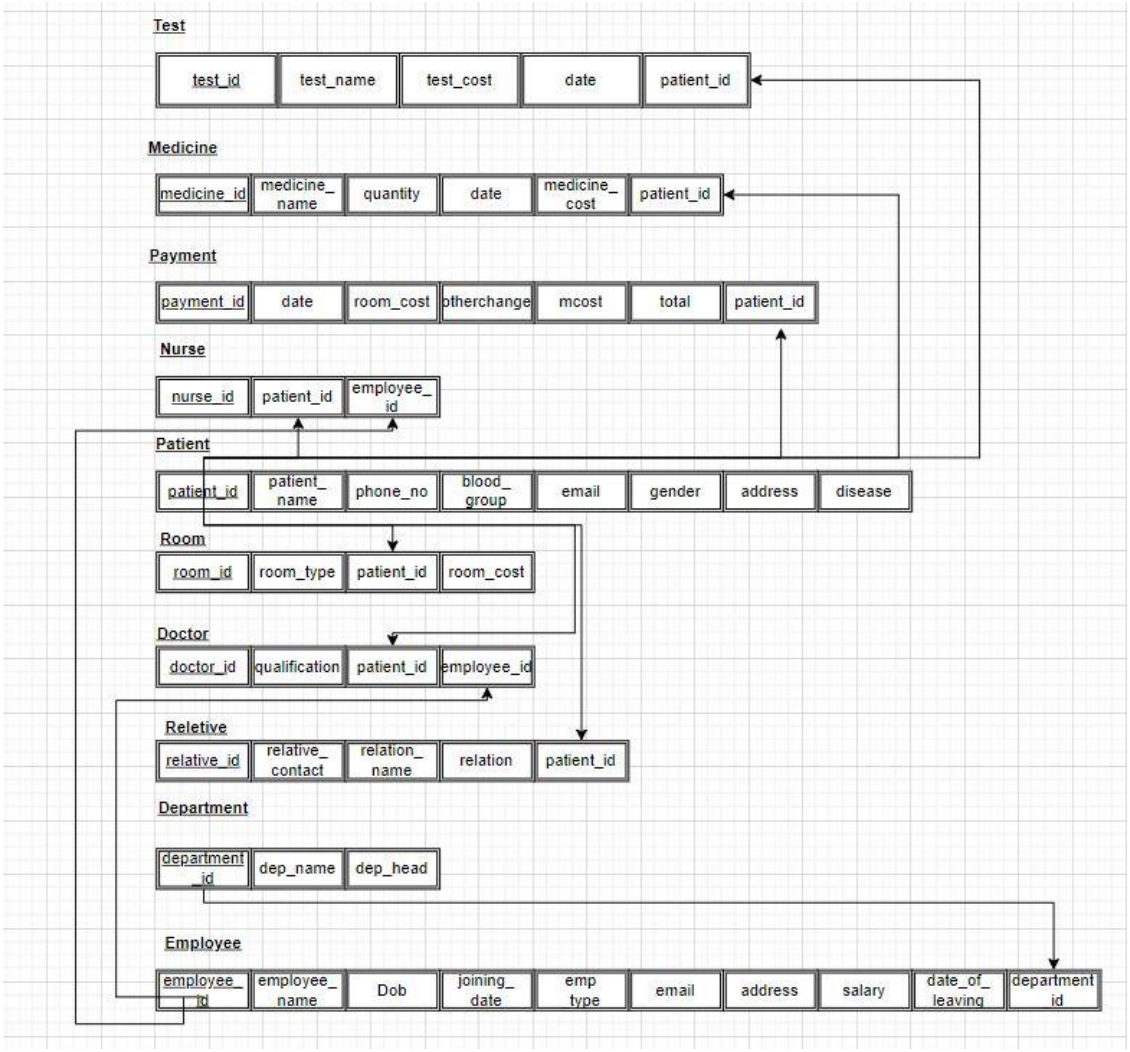
2.RELATIONAL SCHEMA :



3.ER DIAGRAM :


ENTITY - RELATIONSHIP DIAGRAM
OTHER CHARGES
BILL DATE
MCOST
BILL
ROOM COST
PID
TCOST
PAID
BILL DATE
TEST ID
TEST NAME
TEST COST
TEST
TEST DATE
HAD
PID
TST ID
MED DATE
MEDICINE
MEDICINE ID
MEDICINE NAME
MEDICINE COST
TAKES
QUANTITY
EMAIL
D.O.B
GENDER
PHONE NO
ADDRESS
NAME
PATIENT
ISA
BLOOD GP
ARRIVAL DATE
DID
ASSIGNED TO
DOCTOR
QUALIFICATION
OUT_PATIENT
ARRIVAL DATE
DISEASE
RNAME
IN_PATIENT
DISCHARGE DATE
DISEASE
HAS
ALLOCATED
ASSIGNED
NID
RELATIVE
RNAME
RELATION
PHONE NO
ROOM COST
ROOM
RID
RTYPE
NURSE
COUNT PATIENT
ISA
EMPLOYEE
ETYPE
JOINING DATE
EID
NAME
DOB
EMAIL
ADDRESS
SALARY
ISA
PASSIVE
DATE OF LEAVING
ACTIVE
SINCE
WORKED IN
DEPT ID
DEPARTMENT
DEPT NAME
DEPT HEAD
WORKS IN
SINCE

4. ER TO TABLE



5.NORMALIZATION

5.1. First Normal Form(1NF)

- All tables have a primary key

- No repeating groups or arrays

- Each column value is atomic

All tables in the given schema are already in 1NF.

5.2. Second Normal Form (2NF)

- Be in first normal form (1NF)
- No partial dependencies

The "employee" table has partial dependencies since some of its columns are dependent on only a part of the primary key (employee_id). To remove partial dependencies, we can split the table into two tables:

employee_details
- employee_id (primary key)
- employee_name
- dob
- joining_date
- emp_type
- email
- address

employee_salary
- employee_id (primary key and foreign key referencing employee_details table)
- salary
- date_of_leaving
- department_id (foreign key)

5.3. Third Normal Form(3NF)

- Be in second normal form (2NF)
- No transitive dependencies

The "bill" table has a transitive dependency since mcost column is dependent on medicine_cost and room_cost columns. To remove the transitive dependency, we can split the table into two tables:

bill_details

- payment_id (primary key)
- bill_date
- room_cost
- other_charges
- patient_id (foreign key)

medicine_details
- medicine_id (primary key)
- medicine_name
- quantity
- med_date
- medicine_cost
- patient_id (foreign key referencing bill_details table)

bill_total
- payment_id (primary key and foreign key referencing bill_details table)
- mcost
- Total

The resulting normalized schema would be:

department
- department_id (primary key)
- dep_name
- dep_head

employee_details
- employee_id (primary key)
- employee_name
- dob
- joining_date
- emp_type
- email
- address

employee_salary
- employee_id (primary key and foreign key referencing employee_details table)
- salary
- date_of_leaving
- department_id (foreign key)

patient
- patient_id (primary key)
- patient_name

- phone_no
- blood_group
- email
- gender
- address
- disease
- arrival_date
- discharge_date

medicine_details
- medicine_id (primary key)
- medicine_name
- quantity
- med_date
- medicine_cost
- patient_id (foreign key referencing bill_details table)

bill_details
- payment_id (primary key)
- bill_date
- room_cost
- other_charges
- patient_id (foreign key)

bill_total
- payment_id (primary key and foreign key referencing bill_details table)
- mcost
- Total

doctor
- doctor_id (primary key)
- qualification
- patient_id (foreign key referencing patient table)
- employee_id (foreign key referencing employee_details table)

nurse
- nurse_id (primary key)
- patient_id (foreign key referencing patient table)
- employee_id (foreign key referencing employee_details table)

relative
- relative_id (primary key)
- relative_name
- relative_contact

- relation
- patient_id (foreign key referencing patient table)

room
- room_id (primary key)
- room_type
- patient_id (foreign key referencing patient table)
- room_cost

test
- test_id (primary key)
- test_name
- test_cost
- date1
- patient_id (foreign key referencing patient table)

Note that this is just one possible normalization for the given schema and there could be other valid normalizations depending on the specific requirements and use cases of the hospital management system.

5.4. Boyce – Coded Normal Form(BCNF)

To achieve Boyce-Codd Normal Form (BCNF), we need to ensure that for every functional dependency X → Y in a table, X should be a superkey of that table.

1. department table (already in BCNF)
   - department_id (PK)
   - dep_name
   - dep_head

2. employee table (in BCNF)
   - employee_id (PK)
   - employee_name
   - dob
   - joining_date
   - emp_type
   - email
   - address
   - salary
   - date_of_leaving
   - department_id (FK)

3. patient table (in BCNF)
   - patient_id (PK)

- patient_name
- phone_no
- blood_group
- email
- gender
- address
- disease
- arrival_date
- discharge_date

4. medicine table (in BCNF)
- medicine_id (PK)
- medicine_name
- quantity
- med_date
- medicine_cost
- patient_id (FK)

5. bill table (in BCNF)
- payment_id (PK)
- bill_date
- room_cost
- other_charges
- mcost
- Total
- patient_id (FK)

6. doctor table (in BCNF)
- doctor_id (PK)
- qualification
- patient_id (FK)
- employee_id (FK)

7. nurse table (in BCNF)
- nurse_id (PK)
- patient_id (FK)
- employee_id (FK)

8. relative table (in BCNF)
- relative_id (PK)
- relative_name
- relative_contact
- relation
- patient_id (FK)

9. room table (in BCNF)
- room_id (PK)
- room_type
- patient_id (FK)
- room_cost

10. test table (in BCNF)
  - test_id (PK)
  - test_name
  - test_cost
  - date1
  - patient_id (FK)

Note: All the tables are already in BCNF, which means that each table's functional dependencies are well-formed and satisfy the requirement of the BCNF.

6. SCREENSHOTS WITH OUTPUTS

A) CREATION AND INSERTION OF TABLES

CREATION AND INSERTION OF DEPARTMENT

```
create table department(
department_id int primary key,
dep_name varchar(256) not null,
dep_head varchar(256) not null
);
insert into department values(1,'Dermatology','Dr Rajesh');
insert into department values(2,'Gynaecology','Dr Harneet');
insert into department values(3,'Psychiatric','Dr Shiya');
insert into department values(4,'Neurology','Dr Ridhi');
insert into department values(5,'Pediatric','Dr Ira');
```

DEPARTMENT_ID	DEP_NAME	DEP_HEAD
1	Dermatology	Dr Rajesh
2	Gynaecology	Dr Harneet
3	Psychiatric	Dr Shiya
4	Neurology	Dr Ridhi
5	Pediatric	Dr Ira

Table created.

TABLE DEPARTMENT

Column	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER
DEP_NAME	NOT NULL	VARCHAR2(256)
DEP_HEAD	NOT NULL	VARCHAR2(256)

CREATION AND INSERTION OF EMPLOYEE :

```
create table employee(
   employee_id int primary key,
```

```
    employee_name varchar(256) not null,
    dob date not null,
    joining_date date not null,
    emp_type varchar(256) not null,
    email varchar(256) not null,
    address varchar(256) not null,
    salary int not null,
    date_of_leaving date,
    department_id int,
    foreign key (department_id) references department(department_id)
);
insert into employee values(101,'Dr Rajesh','10-November-1965','12-November-
1987','Doctor','drrajesh@gmail.com','Patiala',150000,NULL,1);
insert into employee values(102,'Sunita','10-May-1985','18-November-
2013','Nurse','sunita@gmail.com','Rajpura',28000,NULL,1);
insert into employee values(103,'Dr Harneet','02-May-1965','10-December-
1989','Doctor','harneetkaur@gmail.com','Sangrur',200000,NULL,2);
insert into employee values(104,'Geeta','10-December-1987','12-June-
2014','Nurse','geeta@gmail.com','Patiala',27000,NULL,2);
insert into employee values(105,'Dr Shiya','20-March-1969','12-November-
1987','Doctor','shiyamer@gmail.com','Patiala',250000,NULL,3);
insert into employee values(106,'Mehek','3-October-1988','12-November-
2016','Nurse','mehekkumari@gmail.com','Delhi',27000,NULL,3);
insert into employee values(107,'Dr Ridhi','10-November-1968','12-November-
1988','Doctor','ridhithakur@gmail.com','Rajpura',290000,NULL,4);
insert into employee values(108,'Babita','18-July-1986','12-August-
2013','Nurse','babitakumari@gmail.com','Ambala',29000,NULL,4);
insert into employee values(109,'Dr Ira','10-February-1963','12-November-
1984','Doctor','iragupta@gmail.com','Sunam',270000,NULL,5);
insert into employee values(110,'Swati','10-June-1984','12-july-
2017','Nurse','swatisingla@gmail.com','Sangrur',25000,NULL,5);
```

TABLE EMPLOYEE

Column	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER
EMPLOYEE_NAME	NOT NULL	VARCHAR2(256)
DOB	NOT NULL	DATE
JOINING_DATE	NOT NULL	DATE
EMP_TYPE	NOT NULL	VARCHAR2(256)
EMAIL	NOT NULL	VARCHAR2(256)
ADDRESS	NOT NULL	VARCHAR2(256)

EMPLOYEE_ID	EMPLOYEE_NAME	DOB	JOINING_DATE	EMP_TYPE	EMAIL	ADDRESS	SALARY	DATE_OF_LEAVING	DEPARTMENT_ID
101	Dr Rajesh	10-NOV-65	12-NOV-87	Doctor	drrajesh@gmail.com	Patiala	150000	-	1
102	Sunita	10-MAY-85	18-NOV-13	Nurse	sunita@gmail.com	Rajpura	28000	-	1
103	Dr Harneet	02-MAY-65	10-DEC-89	Doctor	harneetkaur@gmail.com	Sangrur	200000	-	2
104	Geeta	10-DEC-87	12-JUN-14	Nurse	geeta@gmail.com	Patiala	27000	-	2
105	Dr Shiya	20-MAR-69	12-NOV-87	Doctor	shiyamer@gmail.com	Patiala	250000	-	3
106	Mehek	03-OCT-88	12-NOV-16	Nurse	mehekkumari@gmail.com	Delhi	27000	-	3
107	Dr Ridhi	10-NOV-68	12-NOV-88	Doctor	ridhithakur@gmail.com	Rajpura	290000	-	4
108	Babita	18-JUL-86	12-AUG-13	Nurse	babitakumari@gmail.com	Ambala	29000	-	4

CREATION AND INSERTION OF PATIENT TABLE :

```
create table patient(
    patient_id int primary key,
    patient_name varchar(256) not null,
    phone_no varchar(256) not null,
    blood_group varchar(256) not null,
    email varchar(256) not null,
    gender varchar(256) not null,
    address varchar(256) not null,
    disease varchar(256) not null,
```

```
    arrival_date date not null,
    discharge_date date not null
);
insert into patient values(1001,'Ishan','7807324202','B+','ishansharma@gmail.com','Male','Palampur','Skin
Infection','12-January-2023','15-January-2023');
insert into patient values(1003,'Isha','7087392306','O+','ishanarang@gmail.com','Female','Abohar','Pregnant','29-
April-2023','02-May-2023');
insert into patient values(1002,'Harman','9815366304','B-','harmandip@gmail.com','Male','Patiala','Anxiety','27-
May-2022','29-May-2022');
insert into patient values(1004,'Shivam','9815381061','A-','shivambector@gmail.com','Male','Chandigarh','Nerve
Dislocation','27-April-2023','30-April-2023');
insert into patient
values(1005,'Khushpreet','8699298079','A+','khushpreetgill@gmail.com','Female','Sangrur','Cold and Fever','15-
March-2023','16-March-2023');
```

TABLE PATIENT

Column	Null?	Type
PATIENT_ID	NOT NULL	NUMBER
PATIENT_NAME	NOT NULL	VARCHAR2(256)
PHONE_NO	NOT NULL	VARCHAR2(256)
BLOOD_GROUP	NOT NULL	VARCHAR2(256)
EMAIL	NOT NULL	VARCHAR2(256)
GENDER	NOT NULL	VARCHAR2(256)
ADDRESS	NOT NULL	VARCHAR2(256)

PATIENT_ID	PATIENT_NAME	PHONE_NO	BLOOD_GROUP	EMAIL	GENDER	ADDRESS	DISEASE	ARRIVAL_DATE	DISCHARGE_DATE
1001	Ishan	7807324202	B+	ishansharma@gmail.com	Male	Palampur	Skin Infection	12-JAN-23	15-JAN-23
1003	Isha	7087392306	O+	ishanarang@gmail.com	Female	Abohar	Pregnant	29-APR-23	02-MAY-23
1002	Harman	9815366304	B-	harmandip@gmail.com	Male	Patiala	Anxiety	27-MAY-22	29-MAY-22
1004	Shivam	9815381061	A-	shivambector@gmail.com	Male	Chandigarh	Nerve Dislocation	27-APR-23	30-APR-23
1005	Khushpreet	8699298079	A+	khushpreetgill@gmail.com	Female	Sangrur	Cold and Fever	15-MAR-23	16-MAR-23

CREATION AND INSERTION OF MEDICINE TABLE :

```
create table medicine(
    medicine_id int primary key,
    medicine_name varchar(256) not null,
    quantity int not null,
    med_date date not null,
    medicine_cost int not null,
    patient_id int,
    foreign key(patient_id) references patient(patient_id)
);,'JG123','AM1994','H123','Theodore','Chasseriau','France');
INSERT INTO ARTIST VALUES('ART10','NG123','PM1988','G123','Leonardo','da Vinci','Italy');
insert into medicine values(11,'Acetaminophen',50,'12-september-2035',4000,1001);
insert into medicine values(12,'Adderall',100,'14-october-2030',2000,1002);
insert into medicine values(13,'Amitriptyline',150,'25-May-2030',1000,1003);
insert into medicine values(14,'Amlodipine',200,'26-December-2029',4050,1004);
insert into medicine values(15,'Azithromycin',250,'27-January-2029',3000,1005);
```

MEDICINE_ID	MEDICINE_NAME	QUANTITY	MED_DATE	MEDICINE_COST	PATIENT_ID
11	Acetaminophen	50	12-SEP-35	4000	1001
12	Adderall	100	14-OCT-30	2000	1002
13	Amitriptyline	150	25-MAY-30	1000	1003
14	Amlodipine	200	26-DEC-29	4050	1004
15	Azithromycin	250	27-JAN-29	3000	1005

TABLE MEDICINE

Column	Null?	Type
MEDICINE_ID	NOT NULL	NUMBER
MEDICINE_NAME	NOT NULL	VARCHAR2(256)
QUANTITY	NOT NULL	NUMBER
MED_DATE	NOT NULL	DATE
MEDICINE_COST	NOT NULL	NUMBER
PATIENT_ID	-	NUMBER

CREATION AND INSERTION OF BILL TABLE

```
create table bill(
    payment_id int primary key,
    bill_date date not null,
    room_cost int not null,
    other_charges number(3) default null,
    mcost int not null,
    Total int not null,
    patient_id int,
    foreign key(patient_id) references patient(patient_id)
);

insert into bill values(991,'15-January-2023',1000,300,4000,5300,1001);
insert into bill values(992,'30-April-2023',2000,100,2000,4100,1002);
insert into bill values(993,'02-May-2003',1500,200,1000,2700,1003);
insert into bill values(994,'29-May-2022',3000,600,4050,7650,1004);
insert into bill values(995,'16-March-2023',3500,500,3000,7000,1005);
```

PAYMENT_ID	BILL_DATE	ROOM_COST	OTHER_CHARGES	MCOST	TOTAL	PATIENT_ID
991	15-JAN-23	1000	300	4000	5300	1001
992	30-APR-23	2000	100	2000	4100	1002
993	02-MAY-03	1500	200	1000	2700	1003
994	29-MAY-22	3000	600	4050	7650	1004
995	16-MAR-23	3500	500	3000	7000	1005

TABLE BILL

Column	Null?	Type
PAYMENT_ID	NOT NULL	NUMBER
BILL_DATE	NOT NULL	DATE
ROOM_COST	NOT NULL	NUMBER
OTHER_CHARGES	-	NUMBER(3,0)
MCOST	NOT NULL	NUMBER
TOTAL	NOT NULL	NUMBER
PATIENT_ID	-	NUMBER

CREATION AND INSERTION OF DOCTOR TABLE :

```
create table doctor(
    doctor_id int primary key,
    qualification varchar(50) not null,
    patient_id int,
    foreign key(patient_id) references patient(patient_id),
```

```
    employee_id int,
    foreign key(employee_id) references employee(employee_id)
);

insert into doctor values(111,'MBBS MD Dermatology',1001,101);
insert into doctor values(112,'MBBS MD Gynaecology',1002,102);
insert into doctor values(113,'MBBS MD Psychology',1003,103);
insert into doctor values(114,'MBBS MD Neurology',1004,104);
insert into doctor values(115,'MBBS MD Paediatric',1005,105);
```

DOCTOR_ID	QUALIFICATION	PATIENT_ID	EMPLOYEE_ID
111	MBBS MD Dermatology	1001	101
112	MBBS MD Gynaecology	1002	102
113	MBBS MD Psychology	1003	103
114	MBBS MD Neurology	1004	104
115	MBBS MD Paediatric	1005	105

Table created.

TABLE DOCTOR

Column	Null?	Type
DOCTOR_ID	NOT NULL	NUMBER
QUALIFICATION	NOT NULL	VARCHAR2(50)
PATIENT_ID	-	NUMBER
EMPLOYEE_ID	-	NUMBER

CREATION AND INSERTION OF NURSE TABLE :

```
create table nurse(
    nurse_id int Primary key,
    patient_id int,
    foreign key(patient_id) references patient(patient_id),
    employee_id int,
    foreign key(employee_id) references employee(employee_id)
);

insert into nurse values(31,1001,102);
insert into nurse values(32,1002,104);
insert into nurse values(33,1003,106);
insert into nurse values(34,1004,108);
insert into nurse values(35,1005,110);
```

NURSE_ID	PATIENT_ID	EMPLOYEE_ID
31	1001	102
32	1002	104
33	1003	106
34	1004	108
35	1005	110

TABLE NURSE

Column	Null?	Type
NURSE_ID	NOT NULL	NUMBER
PATIENT_ID	-	NUMBER
EMPLOYEE_ID	-	NUMBER

CREATION AND INSERTION OF RELETIVE TABLE :

```
create table reletive(
   reletive_id int Primary key,
   reletive_name varchar(50) Not null,
   reletive_contact varchar(50) not null,    relation varchar(50) not null,
   patient_id int,
   foreign key(patient_id) references patient(patient_id)
);
insert into reletive values(21,'Manvir','9856232652','Friend',1001);
insert into reletive values(23,'Swastik','9812332652','Husband',1002);
insert into reletive values(24,'Gunavri','9812342652','Wife',1003);
insert into reletive values(22,'Shivangi','9853232652','Sister',1004);
insert into reletive values(25,'Jovan','98568692652','Brother',1005);
```

TABLE RELETIVE

Column	Null?	Type
RELETIVE_ID	NOT NULL	NUMBER
RELETIVE_NAME	NOT NULL	VARCHAR2(50)
RELETIVE_CONTACT	NOT NULL	VARCHAR2(50)
RELATION	NOT NULL	VARCHAR2(50)
PATIENT_ID	-	NUMBER

RELETIVE_ID	RELETIVE_NAME	RELETIVE_CONTACT	RELATION	PATIENT_ID
21	Manvir	9856232652	Friend	1001
23	Swastik	9812332652	Husband	1002
24	Gunavri	9812342652	Wife	1003
22	Shivangi	9853232652	Sister	1004
25	Jovan	98568692652	Brother	1005

CREATION AND INSERTION OF ROOM TABLE :

```
create table room(
   room_id int primary key,
   room_type varchar(30) not null,
   patient_id int,
   foreign key(patient_id) references patient(patient_id),
   room_cost int not null
);
insert into room values(61,'Single Bed',1001,1000);
insert into room values(62,'Double Bed',1002,2000);
insert into room values(63,'Single Bed',1003,1500);
insert into room values(64,'Single Bed',1004,3000);
insert into room values(65,'Double Bed',1005,3500);
```

ROOM_ID	ROOM_TYPE	PATIENT_ID	ROOM_COST
61	Single Bed	1001	1000
62	Double Bed	1002	2000
63	Single Bed	1003	1500
64	Single Bed	1004	3000
65	Double Bed	1005	3500

TABLE ROOM

Column	Null?	Type
ROOM_ID	NOT NULL	NUMBER
ROOM_TYPE	NOT NULL	VARCHAR2(30)
PATIENT_ID	-	NUMBER
ROOM_COST	NOT NULL	NUMBER

CREATION AND INSERTION OF TEST TABLE :

```
create table test(
   test_id int Primary key,
   test_name Varchar(60) Not null,
   test_cost int Not null,
   date1 date not null,
   patient_id int,
   foreign key(patient_id) references patient(patient_id)
);
insert into test values(291,'Thyroid test',300,'15-January-2023',1001);
insert into test values(292,'Ultrasound scan',100,'30-April-2023',1002);
insert into test values(293,'MRI',200,'02-May-2003',1003);
insert into test values(294,'Cholestrol test',600,'29-May-2022',1004);
insert into test values(295,'Blood test',500,'16-March-2023',1005);
```

TEST_ID	TEST_NAME	TEST_COST	DATE1	PATIENT_ID
291	Thyroid test	300	15-JAN-23	1001
292	Ultrasound scan	100	30-APR-23	1002
293	MRI	200	02-MAY-03	1003
294	Cholestrol test	600	29-MAY-22	1004
295	Blood test	500	16-MAR-23	1005

TABLE TEST

Column	Null?	Type
TEST_ID	NOT NULL	NUMBER
TEST_NAME	NOT NULL	VARCHAR2(60)
TEST_COST	NOT NULL	NUMBER
DATE1	NOT NULL	DATE
PATIENT_ID	-	NUMBER

B)TRIGGERS :

In DEPARTMENT :

```
CREATE OR REPLACE TRIGGER trg_department
AFTER INSERT  ON department
FOR EACH ROW
BEGIN
```

```
  dbms_output.put_line('Rows is inserted in department table');
END;
```

```
1 row(s) inserted.
Rows is inserted in department table


1 row(s) inserted.
Rows is inserted in department table


1 row(s) inserted.
Rows is inserted in department table


1 row(s) inserted.
Rows is inserted in department table


1 row(s) inserted.
Rows is inserted in department table
```

In employee :

```
CREATE OR REPLACE TRIGGER trg_emp
AFTER INSERT  ON employee
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in employee');
END;
```

```
1 row(s) inserted.
Rows is inserted in employee


1 row(s) inserted.
Rows is inserted in employee


1 row(s) inserted.
Rows is inserted in employee


1 row(s) inserted.
Rows is inserted in employee


1 row(s) inserted.
Rows is inserted in employee
```

In Patient :

```
CREATE OR REPLACE TRIGGER trg_patient
AFTER INSERT  ON patient
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted patient');
END;
```

```
1 row(s) inserted.
Rows is inserted in patient


1 row(s) inserted.
Rows is inserted in patient


1 row(s) inserted.
Rows is inserted in patient


1 row(s) inserted.
Rows is inserted in patient
```

In Medicine :

```
CREATE OR REPLACE TRIGGER trg_medicine
AFTER INSERT  ON medicine
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in medicine');
END;
```

```
1 row(s) inserted.
Rows is inserted in medicine

1 row(s) inserted.
Rows is inserted in medicine

1 row(s) inserted.
Rows is inserted in medicine

1 row(s) inserted.
Rows is inserted in medicine

1 row(s) inserted.
Rows is inserted in medicine
```

In Bill :

```
CREATE OR REPLACE TRIGGER trg_bill
AFTER INSERT  ON bill
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in bill');
END;
```

```
1 row(s) inserted.
Rows is inserted in bill

1 row(s) inserted.
Rows is inserted in bill

1 row(s) inserted.
Rows is inserted in bill

1 row(s) inserted.
Rows is inserted in bill
```

In doctor :

```
CREATE OR REPLACE TRIGGER trg_doctor
AFTER INSERT  ON doctor
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in doctor');
END;
```

```
1 row(s) inserted.
Rows is inserted in doctor


1 row(s) inserted.
Rows is inserted in doctor


1 row(s) inserted.
Rows is inserted in doctor


1 row(s) inserted.
Rows is inserted in doctor


1 row(s) inserted.
Rows is inserted in doctor
```

In Nurse :

```
CREATE OR REPLACE TRIGGER trg_nurse
AFTER INSERT  ON nurse
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in nurse');
END;
```

```
1 row(s) inserted.
Rows is inserted in nurse


1 row(s) inserted.
Rows is inserted in nurse


1 row(s) inserted.
Rows is inserted in nurse


1 row(s) inserted.
Rows is inserted in nurse


1 row(s) inserted.
Rows is inserted in nurse
```

In Relatives :

```
CREATE OR REPLACE TRIGGER trg_reletive
AFTER INSERT  ON reletive
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted');
END;
```

```
1 row(s) inserted.
Rows is inserted


1 row(s) inserted.
Rows is inserted


1 row(s) inserted.
Rows is inserted


1 row(s) inserted.
Rows is inserted


1 row(s) inserted.
Rows is inserted
```

In Room :

```
CREATE OR REPLACE TRIGGER trg_room
AFTER INSERT  ON room
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in room');
END;
```

```
1 row(s) inserted.
Rows is inserted in room


1 row(s) inserted.
Rows is inserted in room


1 row(s) inserted.
Rows is inserted in room


1 row(s) inserted.
Rows is inserted in room


1 row(s) inserted.
Rows is inserted in room
```

In Test :

```
CREATE OR REPLACE TRIGGER trg_test
AFTER INSERT  ON test
FOR EACH ROW
BEGIN
  dbms_output.put_line('Rows is inserted in test');
END;
```

```
1 row(s) inserted.
Rows is inserted in test

1 row(s) inserted.
Rows is inserted in test

1 row(s) inserted.
Rows is inserted in test

1 row(s) inserted.
Rows is inserted in test

1 row(s) inserted.
Rows is inserted in test
```

C) Exceptions :

Exception handling when uniqueness property of primary key is violated:

```
BEGIN
INSERT INTO department(department_id, dep_name, dep_head)
VALUES(1, 'Marketing', 'John Smith');
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error inserting department record: ' || SQLERRM);
END;
```

```
Statement processed.
Error inserting department record: ORA-00001: unique constraint (SQL_AAACLSZEVLVFMGCOMVSLYWLZY.SYS_C00123062168) violated
```

5.3.2 Exception handling when no data is found:

```
BEGIN
UPDATE employee
SET salary = 50000
WHERE employee_id = 16;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Employee record not found');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error updating employee record: ' || SQLERRM);
END;
```

```
Statement processed.
```

5.3.3 Exception handling when no data is found or any other exception:

```
BEGIN
```

```
DELETE FROM patient
WHERE patient_id = 1;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Patient record not found');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error deleting patient record: ' || SQLERRM);
END;
```

```
Statement processed.
```

D) Cursors :

Cursor to show the details of all the employees:

```
DECLARE
CURSOR C1 IS SELECT * FROM employee order by salary desc ;
rec employee%rowtype;
begin
FOR rec in C1 loop
 dbms_output.put_line(rec.employee_id||'  '||rec.employee_name||'        '||rec.emp_type||'
'||rec.email||'         '||rec.salary);
end loop;
END;
```

```
Statement processed.
107  Dr Ridhi         Doctor       ridhithakur@gmail.com          290000
109  Dr Ira         Doctor       iragupta@gmail.com          270000
105  Dr Shiya         Doctor       shiyamer@gmail.com          250000
103  Dr Harneet         Doctor       harneetkaur@gmail.com          200000
101  Dr Rajesh         Doctor       drrajesh@gmail.com          150000
108  Babita         Nurse       babitakumari@gmail.com          29000
102  Sunita         Nurse       sunita@gmail.com          28000
104  Geeta         Nurse       geeta@gmail.com          27000
106  Mehek         Nurse       mehekkumari@gmail.com          27000
110  Swati         Nurse       swatisingla@gmail.com          25000
```

Cursor to show the details of all the employees who are doctors:

```
DECLARE
CURSOR C1 IS SELECT * FROM employee where emp_type='Doctor';
rec employee%rowtype;
begin
FOR rec in C1 loop
 dbms_output.put_line(rec.employee_id||'  '||rec.employee_name||'        '||rec.emp_type||'
'||rec.email||'         '||rec.salary);
end loop;
END;
```

```
Statement processed.
101  Dr Rajesh         Doctor       drrajesh@gmail.com          150000
103  Dr Harneet         Doctor       harneetkaur@gmail.com          200000
105  Dr Shiya         Doctor       shiyamer@gmail.com          250000
107  Dr Ridhi         Doctor       ridhithakur@gmail.com          290000
109  Dr Ira         Doctor       iragupta@gmail.com          270000
```

Cursor to show the details of all the employees who are nurses:

```
DECLARE
CURSOR C1 IS SELECT * FROM employee where emp_type='Nurse';
rec employee%rowtype;
begin
FOR rec in C1 loop
 dbms_output.put_line(rec.employee_id||'  '||rec.employee_name||'        '||rec.emp_type||'
'||rec.email||'         '||rec.salary);
end loop;
END;
```

```
Statement processed.
102  Sunita          Nurse        sunita@gmail.com          28000
104  Geeta         Nurse       geeta@gmail.com          27000
106  Mehek         Nurse       mehekkumari@gmail.com          27000
108  Babita          Nurse        babitakumari@gmail.com          29000
110  Swati         Nurse       swatisingla@gmail.com          25000
```

Cursor to show which employee works in which department:

```
DECLARE
  CURSOR emp_dept_cur IS
    SELECT e.employee_name, d.dep_name
    FROM employee e
    JOIN department d ON e.department_id = d.department_id;
BEGIN
  FOR emp_dept_rec IN emp_dept_cur LOOP
    DBMS_OUTPUT.PUT_LINE(emp_dept_rec.employee_name || ' works in ' ||
emp_dept_rec.dep_name);
  END LOOP;
END;
```

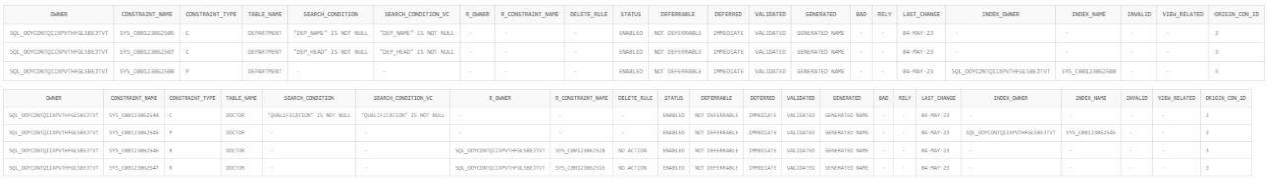
```
Statement processed.
Dr Rajesh works in Dermatology
Sunita works in Dermatology
Dr Harneet works in Gynaecology
Geeta works in Gynaecology
Dr Shiya works in Psychiatric
Mehek works in Psychiatric
Dr Ridhi works in Neurology
Babita works in Neurology
Dr Ira works in Pediatric
Swati works in Pediatric
```

E) Constraints :

```
ALTER TABLE employee ADD CONSTRAINT chk_salary CHECK (salary > 0);
ALTER TABLE patient ADD CONSTRAINT chk_blood_group CHECK (
  blood_group IN ('A+', 'A-', 'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')
);
ALTER TABLE medicine ADD CONSTRAINT chk_medicine_cost CHECK (medicine_cost >= 0);
ALTER TABLE room ADD CONSTRAINT chk_room_cost CHECK (room_cost > 0);
ALTER TABLE bill ADD CONSTRAINT chk_total CHECK (Total > 0);
ALTER TABLE employee ADD CONSTRAINT chk_joining_date CHECK (
  date_of_leaving IS NULL OR joining_date < date_of_leaving
);
```

```
Table altered.


Table altered.


Table altered.


Table altered.


Table altered.
```

```
select * from user_constraints where table_name='EMPLOYEE';
select * from user_constraints where table_name='DEPARTMENT';
select * from user_constraints where table_name='PATIENT';
select * from user_constraints where table_name='MEDICINE';
select * from user_constraints where table_name='BILL';
select * from user_constraints where table_name='DOCTOR';
select * from user_constraints where table_name='NURSE';
select * from user_constraints where table_name='RELETIVE';
select * from user_constraints where table_name='ROOM';
select * from user_constraints where table_name='TEST';
```



F) Functions and Procedures:

Procedure to insert a new department record into the department table:

```
CREATE OR REPLACE PROCEDURE insert_department(department_id IN INT, dep_name IN
VARCHAR2, dep_head IN VARCHAR2) AS
BEGIN
  INSERT INTO department VALUES (department_id, dep_name, dep_head);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Department record inserted successfully!');
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error inserting department record: ' || SQLERRM);
END;

begin
insert_department(6,'Opthalamology','Shivani');
end;
```

```
Statement processed.
Department record inserted successfully!
```

Function to calculate the total salary paid to employees in a particular department:

```
CREATE OR REPLACE FUNCTION get_department_salary(department_id IN INT) RETURN INT
AS
  total_salary INT;
BEGIN
  SELECT SUM(salary) INTO total_salary
  FROM employee
  WHERE department_id = department_id;

  RETURN total_salary;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees found in department');
    RETURN 0;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error calculating department salary: ' || SQLERRM);
    RETURN 0;
END;

declare
```

```
   sal number;
begin
sal:=get_department_salary(6);
dbms_output.put_line('Salary is: '||sal);
end;
```

```
Statement processed.
Salary is: 1296000
```

7. QUERIES

List all the patients who are currently admitted to the hospital:

SELECT * FROM patient WHERE discharge_date IS NULL;

```
no data found
```

List all the doctors and nurses along with the department they belong to:

```
SELECT e.employee_name, e.emp_type, d.dep_name
FROM employee e
JOIN department d ON e.department_id = d.department_id
WHERE e.emp_type IN ('Doctor', 'Nurse');
```

EMPLOYEE_NAME	EMP_TYPE	DEP_NAME
Dr Rajesh	Doctor	Dermatology
Sunita	Nurse	Dermatology
Dr Harneet	Doctor	Gynaecology
Geeta	Nurse	Gynaecology
Dr Shiya	Doctor	Psychiatric
Mehek	Nurse	Psychiatric
Dr Ridhi	Doctor	Neurology
Babita	Nurse	Neurology
Dr Ira	Doctor	Pediatric
Swati	Nurse	Pediatric

List all the patients along with the room they are currently occupying:

SELECT p.patient_name, r.room_type

```
FROM patient p
JOIN room r ON p.patient_id = r.patient_id
WHERE r.patient_id IS NOT NULL;
```

PATIENT_NAME	ROOM_TYPE
Ishan	Single Bed
Harman	Double Bed
Isha	Single Bed
Shivam	Single Bed
Khushpreet	Double Bed

List all the patients along with the tests they have undergone:

```
SELECT p.patient_name, t.test_name, t.date1
FROM patient p
JOIN test t ON p.patient_id = t.patient_id;
```

PATIENT_NAME	TEST_NAME	DATE1
Ishan	Thyroid test	15-JAN-23
Harman	Ultrasound scan	30-APR-23
Isha	MRI	02-MAY-03
Shivam	Cholestrol test	29-MAY-22
Khushpreet	Blood test	16-MAR-23

List all the patients along with the medicines they have been prescribed:

```
SELECT p.patient_name, m.medicine_name, m.quantity, m.med_date
FROM patient p
JOIN medicine m ON p.patient_id = m.patient_id;
```

PATIENT_NAME	MEDICINE_NAME	QUANTITY	MED_DATE
Ishan	Acetaminophen	50	12-SEP-35
Harman	Adderall	100	14-OCT-30
Isha	Amitriptyline	150	25-MAY-30
Shivam	Amlodipine	200	26-DEC-29
Khushpreet	Azithromycin	250	27-JAN-29

List all the patients along with their total bill amount:

```
SELECT p.patient_name, b.Total
FROM patient p
JOIN bill b ON p.patient_id = b.patient_id;
```

PATIENT_NAME	TOTAL
Ishan	5300
Harman	4100
Isha	2700
Shivam	7650
Khushpreet	7000

List all the relatives of a particular patient:

```
SELECT * FROM reletive WHERE patient_id = 1001;
```

RELETIVE_ID	RELETIVE_NAME	RELETIVE_CONTACT	RELATION	PATIENT_ID
21	Manvir	9856232652	Friend	1001

8.CONCLUSION

A hospital management system (HMS) is a software application designed to streamline the administration and management of healthcare facilities such as hospitals, clinics, and medical centers. The system can automate various administrative and operational tasks such as patient registration, appointment scheduling, billing and payment processing, patient record management, inventory management, and staff management.

In conclusion, a hospital management system is an essential tool for modern healthcare facilities. It can help hospitals and clinics to improve their overall efficiency, reduce administrative costs, enhance patient care and satisfaction, and increase profitability. With the integration of advanced technologies such as artificial intelligence and machine learning, hospital management systems can provide even more sophisticated solutions for healthcare providers. As healthcare continues to evolve, hospital management systems will play a crucial role in shaping the future of healthcare delivery.