INDEX

1.SYSTEM OVERVIEW

1.1 CURRENT SYSTEM

A Hospital Management System (HMS) is a software application that manages the overall functioning of a hospital, including patient records, appointment scheduling, billing, and inventory management. In this project, we will be designing a database management system for an HMS.

The system will be divided into various modules, each responsible for a specific set of functionalities. The key modules include:

Patient module: This module will be responsible for storing and managing patient information such as name, age, gender, contact information, medical history, and other relevant details. It will also maintain a record of the patient's appointments, visits, and treatments.

Appointment module: This module will allow patients to book appointments with doctors and other medical staff. The module will also be responsible for scheduling appointments and managing the calendar of doctors and other medical staff.

Doctor module: This module will store information about doctors, including their name, specialty, qualifications, contact information, and availability. It will also keep track of the appointments scheduled for each doctor.

Billing module: This module will handle billing and payment processes. It will generate bills for patients, track payments made by them, and manage the hospital's financial records.

Inventory module: This module will manage the hospital's inventory of medicines, medical equipment, and other supplies. It will maintain a record of the stock levels, orders, and deliveries.

Reports module: This module will generate various reports such as patient records, billing statements, inventory reports, and other relevant reports.

The database management system will be designed using MySQL, which is an open-source relational database management system. It will be used to store and manage all the data required by the HMS. The system will be designed to ensure data security and integrity, and it will comply with all relevant data protection regulations.

In conclusion, the Hospital Management System database management system project will streamline the management of a hospital's operations, including patient records, appointment scheduling, billing, and inventory management. The system will be divided into various modules, and the database management system will be designed using MySQL.

1.2 ADVANTAGES OF THE PROPOSED SYSTEM

There are several advantages of implementing a hospital management system database management system project. Some of the key advantages are:

1. **Improved Efficiency: The hospital management system database management system will streamline hospital operations, eliminating the need for manual processes and reducing the risk of errors. It will improve the efficiency of processes such as appointment scheduling, billing, and inventory management.**

2. **Enhanced Patient Care: The hospital management system will enable healthcare providers to access patient information quickly and easily, facilitating more accurate diagnoses and better patient care. It will also ensure that patients receive timely treatment and care.**

3. **Increased Productivity: The hospital management system will free up staff from administrative tasks, enabling them to focus on providing quality healthcare services. It will also reduce the time required to manage patient records and billing, resulting in increased productivity.**

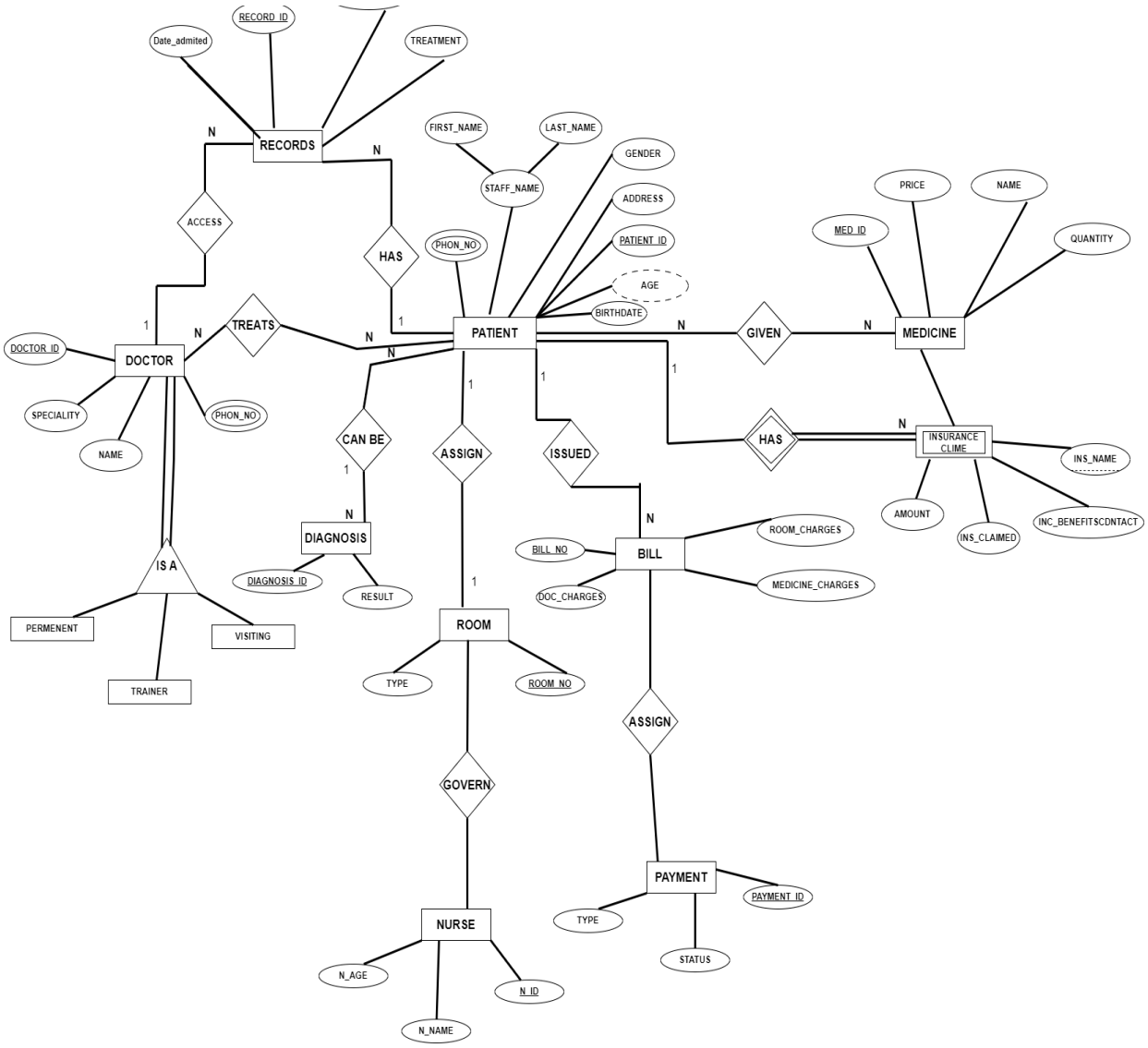
4. **Improved Communication: The hospital management system will improve communication between healthcare providers and patients. Patients can book appointments, receive reminders, and access their medical records online, while healthcare providers can access patient data from any location.**

5. **Better Data Management: The hospital management system will ensure the accuracy and security of patient data, reducing the risk of errors and unauthorized access. It will also provide easy access to patient data, enabling healthcare providers to make informed decisions.**

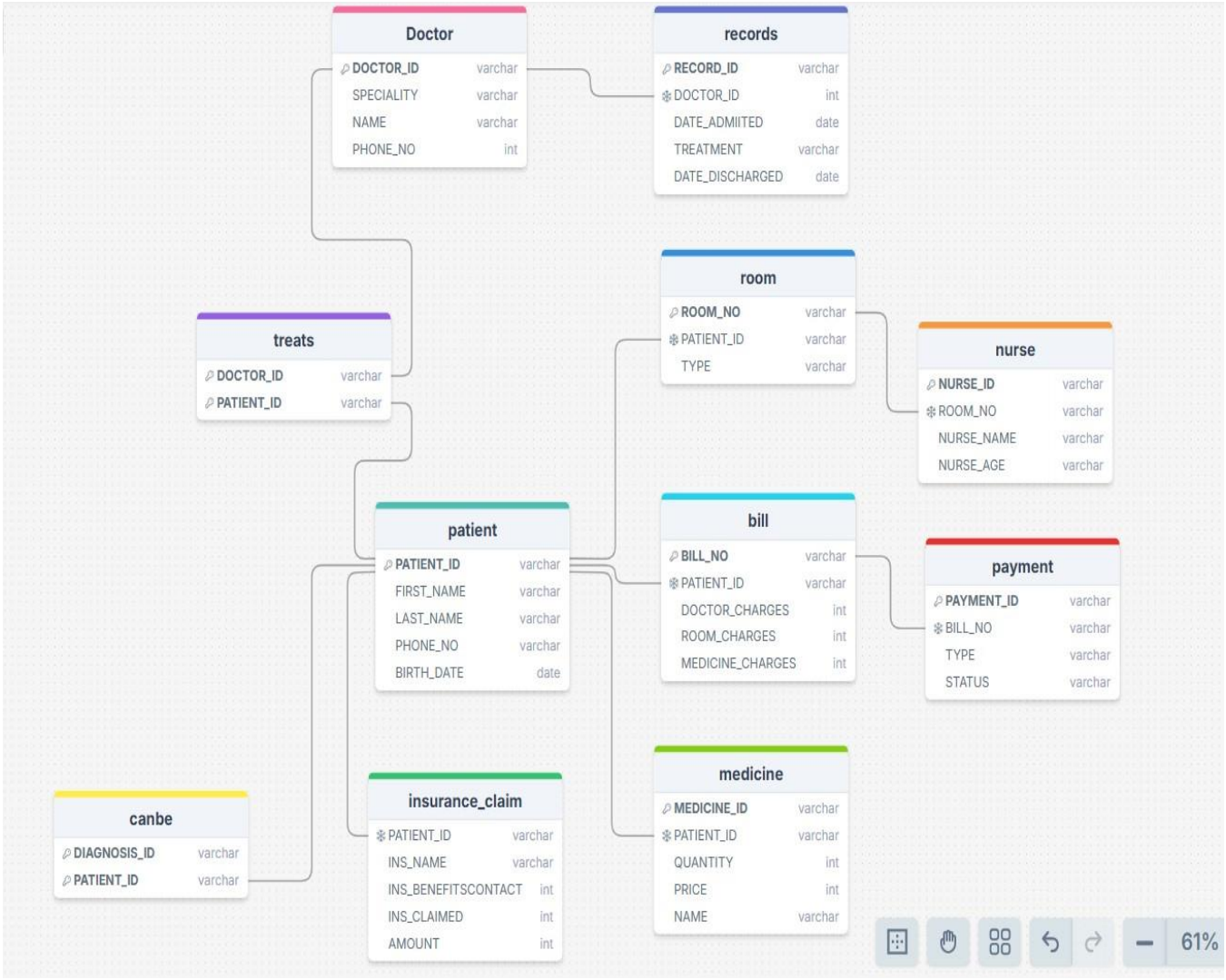
6. **Cost Savings: The hospital management system will reduce administrative costs associated with manual processes, such as data entry, paper-based record keeping, and billing. It will also reduce the risk of errors and fraud, resulting in cost savings.**

7. **In conclusion, implementing a hospital management system database management system project offers several advantages, including improved efficiency, enhanced patient care, increased productivity, improved communication, better data management, and cost savings.**

2. ENTITY-RELATIONSHIP MODEL



3. RELATIONAL SCHEMA



4. Data Dictionary

4.1 Doctor

```
postgres=# \d DOCTOR
                    Table "public.doctor"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 doctor_id  | character varying(25) |           | not null |
 speciality | character varying(25) |           |          |
 name       | character varying(25) |           |          |
 phone_no   | numeric               |           |          |
Indexes:
    "doctor_pkey" PRIMARY KEY, btree (doctor_id)
Referenced by:
    TABLE "treats" CONSTRAINT "doctor1_fk" FOREIGN KEY (doctor_id) REFERENCES doctor(doctor_id)
    TABLE "records" CONSTRAINT "doctor_fk" FOREIGN KEY (doctor_id) REFERENCES doctor(doctor_id)
```

4.2 Patient

```
postgres=# \d PATIENT
                   Table "public.patient"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 patient_id | character varying(25) |           | not null |
 first_name | character varying(25) |           |          |
 last_name  | character varying(25) |           |          |
 phone_no   | character varying(10) |           |          |
 birth_date | date                  |           |          |
Indexes:
    "patient_pkey" PRIMARY KEY, btree (patient_id)
Referenced by:
    TABLE "treats" CONSTRAINT "patient1_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
    TABLE "medicine" CONSTRAINT "patient2_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
    TABLE "bill" CONSTRAINT "patient3_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
    TABLE "insurance_clain" CONSTRAINT "patient4_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
    TABLE "canbe" CONSTRAINT "patient5_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
    TABLE "room" CONSTRAINT "patient_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.3 Nurse

```
postgres=# \d NURSE
                    Table "public.nurse"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 nurse_id   | character varying(25) |           | not null |
 room_no    | character varying(25) |           |          |
 nurse_name | character varying(25) |           |          |
 nurse_age  | character varying(25) |           |          |
Indexes:
    "nurse_pkey" PRIMARY KEY, btree (nurse_id)
```

4.4 Records

```
postgres=# \d RECORDS
                        Table "public.records"
     Column      |         Type          | Collation | Nullable | Default
-----------------+-----------------------+-----------+----------+---------
 record_id       | character varying(25) |           | not null |
 doctor_id       | character varying(25) |           |          |
 date_admitted   | date                  |           |          |
 treatment       | character varying(4000) |         |          |
 date_discharged | date                  |           |          |
Indexes:
    "records_pkey" PRIMARY KEY, btree (record_id)
Foreign-key constraints:
    "doctor_fk" FOREIGN KEY (doctor_id) REFERENCES doctor(doctor_id)
```

4.5 Room

```
postgres=# \d ROOM
                     Table "public.room"
   Column   |        Type         | Collation | Nullable | Default
------------+---------------------+-----------+----------+---------
 room_no    | character varying(4)  |         | not null |
 patient_id | character varying(25) |         |          |
 type       | character varying(25) |         |          |
Indexes:
    "room_pkey" PRIMARY KEY, btree (room_no)
Foreign-key constraints:
    "patient_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.6 Medicine

```
postgres=# \d MEDICINE
                    Table "public.medicine"
   Column    |        Type         | Collation | Nullable | Default
-------------+---------------------+-----------+----------+---------
 medicine_id | character varying(25) |         | not null |
 patient_id  | character varying(25) |         |          |
 quantity    | numeric               |         |          |
 price       | numeric               |         |          |
 name        | character varying(25) |         |          |
Indexes:
    "medicine_pkey" PRIMARY KEY, btree (medicine_id)
Foreign-key constraints:
    "patient2_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.7 Bill

```
postgres=# \d BILL
                        Table "public.bill"
     Column     |        Type         | Collation | Nullable | Default
----------------+---------------------+-----------+----------+---------
 bill_no        | character varying(25) |           | not null |
 patient_id     | character varying(25) |           |          |
 doctor_charges | numeric               |           |          |
 room_charges   | numeric               |           |          |
 medicine_charges | numeric             |           |          |
Indexes:
    "bill_pkey" PRIMARY KEY, btree (bill_no)
Foreign-key constraints:
    "patient3_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
Referenced by:
    TABLE "payment" CONSTRAINT "bill_fk" FOREIGN KEY (bill_no) REFERENCES bill(bill_no)
```

4.8 Canbe

```
postgres=# \d CANBE
                     Table "public.canbe"
   Column    |        Type         | Collation | Nullable | Default
-------------+---------------------+-----------+----------+---------
 diagnosis_id | character varying(25) |           |          |
 patient_id   | character varying(25) |           |          |
Foreign-key constraints:
    "patient5_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.9 Treats

```
postgres=# \d TREATS
                     Table "public.treats"
   Column   |        Type         | Collation | Nullable | Default
------------+---------------------+-----------+----------+---------
 doctor_id  | character varying(25) |           |          |
 patient_id | character varying(25) |           |          |
Foreign-key constraints:
    "doctor1_fk" FOREIGN KEY (doctor_id) REFERENCES doctor(doctor_id)
    "patient1_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.10 Insurance_Claim

```
postgres=# \d INSURANCE_CLAIM
                    Table "public.insurance_claim"
      Column       |        Type         | Collation | Nullable | Default
-------------------+---------------------+-----------+----------+---------
 patient_id        | character varying(25) |           |          |
 ins_name          | character varying(25) |           |          |
 ins_benefitscontact | numeric             |           |          |
 ins_claimadd      | numeric               |           |          |
 amount            | numeric               |           |          |
Foreign-key constraints:
    "patient4_fk" FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
```

4.11 Payment

```
postgres=# \d PAYMENT
                    Table "public.payment"
   Column   |        Type         | Collation | Nullable | Default
------------+---------------------+-----------+----------+---------
 payment_id | character varying(25) |           | not null |
 bill_no    | character varying(25) |           |          |
 type       | character varying(25) |           |          |
 status     | character varying(25) |           |          |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (payment_id)
Foreign-key constraints:
    "bill_fk" FOREIGN KEY (bill_no) REFERENCES bill(bill_no)
```

5. DATA IMPLEMENTATION

A) SCHEMA

5.1.1 DOCTOR

```
CREATE TABLE DOCTOR (
    DOCTOR_ID VARCHAR(25) PRIMARY KEY,
    SPECIALITY VARCHAR(25),
    NAME VARCHAR(25),
    PHONE_NO NUMERIC
);
```

5.1.2 PATIENT

```
CREATE TABLE PATIENT (
    PATIENT_ID VARCHAR(25) PRIMARY KEY,
    FIRST_NAME VARCHAR(25),
    LAST_NAME VARCHAR(25),
    PHONE_NO VARCHAR(10),
    BIRTH_DATE DATE
);
```

5.1.3 NURSE

```
CREATE TABLE NURSE (
    NURSE_ID VARCHAR(25) PRIMARY KEY,
    ROOM_NO VARCHAR(25),
    NURSE_NAME VARCHAR(25),
    NURSE_AGE VARCHAR(25)
);
```

5.1.4 RECORDS

```
CREATE TABLE RECORDS (
    RECORD_ID VARCHAR(25) PRIMARY KEY,
    DOCTOR_ID VARCHAR(25),
    DATE_ADMITTED DATE,
    TREATMENT VARCHAR(4000),
    DATE_DISCHARGED DATE,
    CONSTRAINT doctor_fk FOREIGN KEY (DOCTOR_ID) REFERENCES
DOCTOR(DOCTOR_ID)
);
```

5.1.5 ROOM

```
CREATE TABLE ROOM (
    ROOM_NO VARCHAR(4) PRIMARY KEY,
    PATIENT_ID VARCHAR(25),
    TYPE VARCHAR(25),
    CONSTRAINT patient_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID)
);
```

5.1.6 MEDICINE

```
CREATE TABLE MEDICINE (
    MEDICINE_ID VARCHAR(25) PRIMARY KEY,
    PATIENT_ID VARCHAR(25),
    QUANTITY NUMERIC,
    PRICE NUMERIC,
    NAME VARCHAR(25),
    CONSTRAINT patient2_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID)
);
```

5.1.7 BILL

```
CREATE TABLE BILL (
    BILL_NO VARCHAR(25) PRIMARY KEY,
    PATIENT_ID VARCHAR(25),
    DOCTOR_CHARGES NUMERIC,
    ROOM_CHARGES NUMERIC,
    MEDICINE_CHARGES NUMERIC,
    CONSTRAINT patient3_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID)
);
```

5.1.8 CANBE

```
CREATE TABLE CANBE (
    DIAGNOSIS_ID VARCHAR(25),
    PATIENT_ID VARCHAR(25),
    CONSTRAINT patient5_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID)
);
```

5.1.9 TREATS

```
CREATE TABLE TREATS (
    DOCTOR_ID VARCHAR(25),
    PATIENT_ID VARCHAR(25),
    CONSTRAINT patient1_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID),
    CONSTRAINT doctor1_fk FOREIGN KEY (DOCTOR_ID) REFERENCES
DOCTOR(DOCTOR_ID)
);
```

5.1.10 INSURANCE_CLAIM

```
CREATE TABLE INSURANCE_CLAIM (
    PATIENT_ID VARCHAR(25),
    INS_NAME VARCHAR(25),
    INS_BENEFITSCONTACT NUMERIC,
    INS_CLAIMADD NUMERIC,
    AMOUNT NUMERIC,
    CONSTRAINT patient4_fk FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID)
);
```

5.1.11 PAYMENT

```
CREATE TABLE PAYMENT (
    PAYMENT_ID VARCHAR(25) PRIMARY KEY,
    BILL_NO VARCHAR(25),
    TYPE VARCHAR(25),
    STATUS VARCHAR(25),
    CONSTRAINT bill_fk FOREIGN KEY (BILL_NO) REFERENCES
BILL(BILL_NO)
);
```

B) DATA INSERTION

5.2.1 DOCTOR

```
INSERT INTO DOCTOR(DOCTOR_ID, SPECIALITY, NAME, PHONE_NO)
VALUES
   ('d1', 'Orthopedics', 'Ashish Sabharwal', '9898989898'),
   ('d2', 'Internal Medicine', 'Sanjay Sachdeva', '8989898989'),
   ('d3', 'Orthopedics', 'Surbhi Anand', '1212121212'),
   ('d4', 'Pathology', 'Gagan Gautam', '2121212121'),
   ('d5', 'Dermatology', 'Sandeep Batra', '3434343434'),
   ('d6', 'Internal Medicine', 'Aditya Gupta', '4343434343'),
   ('d7', 'Orthopedics', 'Bipin S Walia', '5656565656'),
   ('d8', 'Pathology', 'Amandeep Singh Dhillon', '6565656565'),
   ('d9', 'Dermatology', 'SKS Marya', '1717171717'),
   ('d10', 'Internal Medicine', 'Gaurav Kharya', '7171717171');
```

5.2.2 PATIENT

```
INSERT INTO PATIENT(PATIENT_ID, FIRST_NAME, LAST_NAME, PHONE_NO,
BIRTH_DATE)
VALUES
   ('p1', 'Narendra', 'Modi', '1231231231', TO_DATE('10-FEB-1965', 'DD-MON-
YYYY')),
   ('p2', 'Virat', 'Kohli', '2312312312', TO_DATE('13-JAN-1980', 'DD-MON-YYYY')),
   ('p3', 'Shahrukh', 'Khan', '3123123123', TO_DATE('19-AUG-1972', 'DD-MON-
YYYY')),
   ('p4', 'Arjun', 'Allu', '4564564564', TO_DATE('22-APR-1970', 'DD-MON-YYYY')),
   ('p5', 'Salman', 'Khan', '5645645645', TO_DATE('01-FEB-1975', 'DD-MON-
YYYY')),
   ('p6', 'Vijay', 'Thalapathy', '6456456456', TO_DATE('14-JUL-1969', 'DD-MON-
YYYY')),
   ('p7', 'Amit', 'Shah', '7897897897', TO_DATE('26-MAR-1979', 'DD-MON-
YYYY')),
   ('p8', 'Kapil', 'Sharma', '8978978978', TO_DATE('30-SEP-1978', 'DD-MON-
YYYY')), -- Fixed invalid date
   ('p9', 'Ramdev', 'Baba', '9789789789', TO_DATE('10-DEC-1968', 'DD-MON-
YYYY')),
   ('p10', 'Sundar', 'Pichai', '9900887847', TO_DATE('07-OCT-1976', 'DD-MON-
YYYY'));
```

5.2.3 NURSE

```
INSERT INTO NURSE(NURSE_ID, ROOM_NO, NURSE_NAME, NURSE_AGE)
VALUES
   ('N1', 'rn1', 'Priya Patel', 29),
   ('N2', 'rn2', 'Sanjay Kumar', 32),
   ('N3', 'rn3', 'Sangeeta Shah', 27),
   ('N4', 'rn4', 'Rajesh Mehra', 35),
   ('N5', 'rn5', 'Meera Rajput', 31),
   ('N6', 'rn6', 'Arjun Singh', 28),
   ('N7', 'rn7', 'Pallavi Desai', 33),
   ('N8', 'rn8', 'Ramesh Chopra', 36),
   ('N9', 'rn9', 'Alka Bhatia', 30),
   ('N10', 'rn10', 'Neha Kapoor', 26);
```

5.2.4 RECORDS

```
INSERT INTO RECORDS (RECORD_ID, DOCTOR_ID, DATE_ADMITTED,
TREATMENT, DATE_DISCHARGED)
VALUES
   ('rec1', 'd1', TO_DATE('1-FEB-2023', 'DD-MON-YYYY'), 'Flu', TO_DATE('3-
FEB-2023', 'DD-MON-YYYY')),
   ('rec2', 'd2', TO_DATE('1-FEB-2023', 'DD-MON-YYYY'), 'Common Cold',
TO_DATE('2-FEB-2023', 'DD-MON-YYYY')),
   ('rec3', 'd3', TO_DATE('1-FEB-2023', 'DD-MON-YYYY'), 'Stomachache',
TO_DATE('4-FEB-2023', 'DD-MON-YYYY')),
   ('rec4', 'd4', TO_DATE('1-FEB-2023', 'DD-MON-YYYY'), 'Back Pain',
TO_DATE('3-FEB-2023', 'DD-MON-YYYY')),
   ('rec5', 'd5', TO_DATE('1-FEB-2023', 'DD-MON-YYYY'), 'Toothache',
TO_DATE('1-FEB-2023', 'DD-MON-YYYY')),
   ('rec6', 'd6', TO_DATE('2-FEB-2023', 'DD-MON-YYYY'), 'Asthma', TO_DATE('7-
FEB-2023', 'DD-MON-YYYY')),
   ('rec7', 'd7', TO_DATE('2-FEB-2023', 'DD-MON-YYYY'), 'Allergies',
TO_DATE('4-FEB-2023', 'DD-MON-YYYY')),
   ('rec8', 'd8', TO_DATE('2-FEB-2023', 'DD-MON-YYYY'), 'Art Therapy',
TO_DATE('4-FEB-2023', 'DD-MON-YYYY')),
   ('rec9', 'd9', TO_DATE('2-FEB-2023', 'DD-MON-YYYY'), 'COVID-19',
TO_DATE('9-FEB-2023', 'DD-MON-YYYY')),
   ('rec10', 'd10', TO_DATE('2-FEB-2023', 'DD-MON-YYYY'), 'Influenza',
TO_DATE('7-FEB-2023', 'DD-MON-YYYY'));
```

5.2.5 ROOM

```
INSERT INTO ROOM(ROOM_NO, PATIENT_ID, TYPE) VALUES
   ('rn1', 'p1', 'general'),
   ('rn2', 'p2', 'deluxe'),
   ('rn3', 'p3', 'super deluxe'),
   ('rn4', 'p4', 'general'),
   ('rn5', 'p5', 'deluxe'),
   ('rn6', 'p6', 'super deluxe'),
   ('rn7', 'p7', 'general'),
   ('rn8', 'p8', 'deluxe'),
   ('rn9', 'p9', 'super deluxe'),
   ('rn10', 'p10', 'super deluxe'),
   ('rn11', NULL, 'general'),
   ('rn12', NULL, 'super deluxe'),
   ('rn13', NULL, 'deluxe');
```

5.2.6 MEDICINE

```
INSERT INTO MEDICINE(MEDICINE_ID, PATIENT_ID, QUANTITY, PRICE,
NAME) VALUES
   ('M1', 'p1', 50, 100, 'Paracetamol'),
   ('M2', 'p2', 5, 1000, 'Ibuprofen'),
   ('M3', 'p3', 25, 240, 'Aspirin'),
   ('M4', 'p4', 30, 100, 'Metformin'),
   ('M5', 'p5', 80, 75, 'Amoxicillin'),
   ('M6', 'p6', 5, 83, 'Omeprazole'),
   ('M7', 'p7', 10, 225, 'Azithromycin'),
   ('M8', 'p8', 40, 160, 'Atorvastatin'),
   ('M9', 'p9', 25, 93, 'Clopidogrel'),
   ('M10', 'p10', 20, 34, 'Losartan');
```

5.2.7 BILL

```
INSERT INTO BILL (BILL_NO, PATIENT_ID, DOCTOR_CHARGES,
ROOM_CHARGES, MEDICINE_CHARGES) VALUES
   ('b1', 'p1', 400, 2000, 8000),
   ('b2', 'p2', 500, 3000, 2200),
   ('b3', 'p3', 600, 4000, 2800),
   ('b4', 'p4', 800, 5000, 1500),
   ('b5', 'p5', 200, 6000, 3400),
   ('b6', 'p6', 600, 4000, 800),
   ('b7', 'p7', 400, 3000, 500),
   ('b8', 'p8', 700, 5000, 900),
   ('b9', 'p9', 300, 1000, 4500),
   ('b10', 'p10', 900, 8000, 7600);
```

5.2.8 CANBE

```
INSERT INTO CANBE (DIAGNOSIS_ID, PATIENT_ID) VALUES
   ('d1', 'p1'),
   ('d10', 'p2'),
   ('d9', 'p3'),
   ('d8', 'p4'),
   ('d7', 'p5'),
   ('d2', 'p6'),
   ('d4', 'p7'),
   ('d3', 'p8'),
   ('d6', 'p9'),
   ('d5', 'p10');
```

5.2.9 TREATS

```
INSERT INTO TREATS (DOCTOR_ID, PATIENT_ID) VALUES
   ('d1', 'p1'),
   ('d2', 'p2'),
   ('d3', 'p3'),
   ('d4', 'p4'),
   ('d5', 'p5'),
   ('d6', 'p6'),
   ('d7', 'p7'),
   ('d8', 'p8'),
   ('d9', 'p9'),
   ('d10', 'p10');
```

5.2.10 INSURANCE_CLAIM

```
INSERT INTO INSURANCE_CLAIM (PATIENT_ID, INS_NAME,
INS_BENEFITSCONTACT, INS_CLAIMADD, AMOUNT) VALUES
   ('p1', 'bajaj', 1234567892, 4000, 10000),
   ('p2', 'lic', 2345678912, 3000, 12000),
   ('p3', 'sbi life', 3456789123, 4500, 11000),
   ('p4', 'reliance general', 2345678923, 4200, 15000),
   ('p5', 'bajaj', 5678901233, 4000, 13000),
   ('p6', 'lic', 9876543219, 5600, 18000),
   ('p7', 'bajaj', 8765432109, 8700, 22000),
   ('p8', 'sbi life', 7654321098, 6000, 14000),
   ('p9', 'reliance general', 5678943210, 2000, 7000),
   ('p10', 'bajaj', 987654321, 2700, 9000);
```

5.2.11 PAYMENT

```
INSERT INTO PAYMENT (PAYMENT_ID, BILL_NO, TYPE, STATUS) VALUES
   ('pay1', 'b1', 'cash', 'paid'),
   ('pay2', 'b2', 'card', 'pending'),  -- Changed NULL to 'pay2'
   ('pay3', 'b3', 'upi', 'paid'),
   ('pay4', 'b4', 'netbanking', 'pending'),  -- Changed NULL to 'pay4'
   ('pay5', 'b5', 'cash', 'paid'),
   ('pay6', 'b6', 'upi', 'paid'),
   ('pay7', 'b7', 'cheque', 'pending'),  -- Changed NULL to 'pay7'
   ('pay8', 'b8', 'upi', 'paid'),
   ('pay9', 'b9', 'netbanking', 'pending'),  -- Changed NULL to 'pay9'
   ('pay10', 'b10', 'cash', 'paid');
```

INSERTION OUTPUT:

5.2.1 DOCTOR

```
postgres=# select *from DOCTOR;
 doctor_id |    speciality    |         name          |  phone_no
-----------+------------------+-----------------------+------------
 d1        | Orthopedics      | Ashish Sabharwal      | 9898989898
 d2        | Internal Medicine| Sanjay Sachdeva       | 8989898989
 d3        | Orthopedics      | Surbhi Anand          | 1212121212
 d4        | Pathology        | Gagan Gautam          | 2121212121
 d5        | Dermatology      | Sandeep Batra         | 3434343434
 d6        | Internal Medicine| Aditya Gupta          | 4343434343
 d7        | Orthopedics      | Bipin S Walia         | 5656565656
 d8        | Pathology        | Amandeep Singh Dhillon| 6565656565
 d9        | Dermatology      | SKS Marya             | 1717171717
 d10       | Internal Medicine| Gaurav Kharya         | 7171717171
(10 rows)
```

5.2.2 PATIENT

```
postgres=# select *from PATIENT;
 patient_id | first_name | last_name  |  phone_no  | birth_date
------------+------------+------------+------------+------------
 p1         | Narendra   | Modi       | 1231231231 | 1965-02-10
 p2         | Virat      | Kohli      | 2312312312 | 1980-01-13
 p3         | Shahrukh   | Khan       | 3123123123 | 1972-08-19
 p4         | Arjun      | Allu       | 4564564564 | 1970-04-22
 p5         | Salman     | Khan       | 5645645645 | 1975-02-01
 p6         | Vijay      | Thalapathy | 6456456456 | 1969-07-14
 p7         | Amit       | Shah       | 7897897897 | 1979-03-26
 p8         | Kapil      | Sharma     | 8978978978 | 1978-09-30
 p9         | Ramdev     | Baba       | 9789789789 | 1968-12-10
 p10        | Sundar     | Pichai     | 9900887847 | 1976-10-07
(10 rows)
```

5.2.3 NURSE

```
postgres=# select *from NURSE;
 nurse_id | room_no |  nurse_name   | nurse_age
----------+---------+---------------+-----------
 N1       | rn1     | Priya Patel   | 29
 N2       | rn2     | Sanjay Kumar  | 32
 N3       | rn3     | Sangeeta Shah | 27
 N4       | rn4     | Rajesh Mehra  | 35
 N5       | rn5     | Meera Rajput  | 31
 N6       | rn6     | Arjun Singh   | 28
 N7       | rn7     | Pallavi Desai | 33
 N8       | rn8     | Ramesh Chopra | 36
 N9       | rn9     | Alka Bhatia   | 30
 N10      | rn10    | Neha Kapoor   | 26
(10 rows)
```

5.2.4 RECORDS

```
postgres=# select *from RECORDS;
 record_id | doctor_id | date_admitted |  treatment  | date_discharged
-----------+-----------+---------------+-------------+-----------------
 rec1      | d1        | 2023-02-01    | Flu         | 2023-02-03
 rec2      | d2        | 2023-02-01    | Common Cold | 2023-02-02
 rec3      | d3        | 2023-02-01    | Stomachache | 2023-02-04
 rec4      | d4        | 2023-02-01    | Back Pain   | 2023-02-03
 rec5      | d5        | 2023-02-01    | Toothache   | 2023-02-01
 rec6      | d6        | 2023-02-02    | Asthma      | 2023-02-07
 rec7      | d7        | 2023-02-02    | Allergies   | 2023-02-04
 rec8      | d8        | 2023-02-02    | Art Therapy | 2023-02-04
 rec9      | d9        | 2023-02-02    | COVID-19    | 2023-02-09
 rec10     | d10       | 2023-02-02    | Influenza   | 2023-02-07
(10 rows)
```

5.2.5 ROOM

```
postgres=# select *from ROOM;
 room_no | patient_id |     type
---------+------------+--------------
 rn1     | p1         | general
 rn2     | p2         | deluxe
 rn3     | p3         | super deluxe
 rn4     | p4         | general
 rn5     | p5         | deluxe
 rn6     | p6         | super deluxe
 rn7     | p7         | general
 rn8     | p8         | deluxe
 rn9     | p9         | super deluxe
 rn10    | p10        | super deluxe
 rn11    |            | general
 rn12    |            | super deluxe
 rn13    |            | deluxe
(13 rows)
```

5.2.6 MEDICINE

```
postgres=# select *from MEDICINE;
 medicine_id | patient_id | quantity | price |     name
-------------+------------+----------+-------+-------------
 M1          | p1         |       50 |   100 | Paracetamol
 M2          | p2         |        5 |  1000 | Ibuprofen
 M3          | p3         |       25 |   240 | Aspirin
 M4          | p4         |       30 |   100 | Metformin
 M5          | p5         |       80 |    75 | Amoxicillin
 M6          | p6         |        5 |    83 | Omeprazole
 M7          | p7         |       10 |   225 | Azithromycin
 M8          | p8         |       40 |   160 | Atorvastatin
 M9          | p9         |       25 |    93 | Clopidogrel
 M10         | p10        |       20 |    34 | Losartan
(10 rows)
```

5.2.7 BILL

```
postgres=# select *from BILL;
 bill_no | patient_id | doctor_charges | room_charges | medicine_charges
---------+------------+----------------+--------------+------------------
 b1      | p1         |            400 |         2000 |             8000
 b2      | p2         |            500 |         3000 |             2200
 b3      | p3         |            600 |         4000 |             2800
 b4      | p4         |            800 |         5000 |             1500
 b5      | p5         |            200 |         6000 |             3400
 b6      | p6         |            600 |         4000 |              800
 b7      | p7         |            400 |         3000 |              500
 b8      | p8         |            700 |         5000 |              900
 b9      | p9         |            300 |         1000 |             4500
 b10     | p10        |            900 |         8000 |             7600
(10 rows)
```

5.2.8 CANBE

```
postgres=# select *from CANBE;
 diagnosis_id | patient_id
--------------+------------
 d1           | p1
 d10          | p2
 d9           | p3
 d8           | p4
 d7           | p5
 d2           | p6
 d4           | p7
 d3           | p8
 d6           | p9
 d5           | p10
(10 rows)
```

5.2.9 TREATS

```
postgres=# select *from TREATS;
 doctor_id | patient_id
-----------+------------
 d1        | p1
 d2        | p2
 d3        | p3
 d4        | p4
 d5        | p5
 d6        | p6
 d7        | p7
 d8        | p8
 d9        | p9
 d10       | p10
(10 rows)
```

5.2.10 INSURANCE_CLAIM

```
postgres=# select *from INSURANCE_CLAIM;
 patient_id |     ins_name     | ins_benefitscontact | ins_claimadd | amount
------------+------------------+---------------------+--------------+--------
 p1         | bajaj            |          1234567892 |         4000 |  10000
 p2         | lic              |          2345678912 |         3000 |  12000
 p3         | sbi life         |          3456789123 |         4500 |  11000
 p4         | reliance general |          2345678923 |         4200 |  15000
 p5         | bajaj            |          5678901233 |         4000 |  13000
 p6         | lic              |          9876543219 |         5600 |  18000
 p7         | bajaj            |          8765432109 |         8700 |  22000
 p8         | sbi life         |          7654321098 |         6000 |  14000
 p9         | reliance general |          5678943210 |         2000 |   7000
 p10        | bajaj            |           987654321 |         2700 |   9000
(10 rows)
```

5.2.11 PAYMENT

```
postgres=# select *from PAYMENT;
 payment_id | bill_no |    type    | status
------------+---------+------------+---------
 pay1       | b1      | cash       | paid
 pay2       | b2      | card       | pending
 pay3       | b3      | upi        | paid
 pay4       | b4      | netbanking | pending
 pay5       | b5      | cash       | paid
 pay6       | b6      | upi        | paid
 pay7       | b7      | cheque     | pending
 pay8       | b8      | upi        | paid
 pay9       | b9      | netbanking | pending
 pay10      | b10     | cash       | paid
(10 rows)
```

5.3 QUERIES USING BASIC DBMS CONSTRUCTS JOIN & SUBQUERIES:

5.3.1 find no of room available according to room type?

```
postgres=# SELECT type, COUNT(*) FROM ROOM WHERE patient_id IS NULL GROUP BY type;
     type     | count
--------------+-------
 deluxe       |     1
 general      |     1
 super deluxe |     1
(3 rows)
```

5.3.2 find patient name with their doctor name?

```
postgres=# SELECT doctor.name, patient.first_name FROM doctor FULL JOIN treats ON doctor.doctor_id = treats.doctor_id FULL JOIN patient ON treats.patient_id
 = patient.patient_id;
         name         | first_name
----------------------+------------
 Ashish Sabharwal     | Narendra
 Sanjay Sachdeva      | Virat
 Surbhi Anand         | Shahrukh
 Gagan Gautam         | Arjun
 Sandeep Batra        | Salman
 Aditya Gupta         | Vijay
 Bipin S Walia        | Amit
 Amandeep Singh Dhillon | Kapil
 SKS Marya            | Ramdev
 Gaurav Kharya        | Sundar
(10 rows)
```

5.3.3 find the name of the patient who has pay payment through cash?

```
postgres=# SELECT DISTINCT first_name FROM patient NATURAL JOIN bill NATURAL JOIN payment WHERE type = 'cash';
 first_name
------------
 Narendra
 Salman
 Sundar
(3 rows)
```

5.3.4 list all patient whose first name start with A.

```
postgres=# SELECT patient_id , first_name , birth_date FROM patient WHERE FIRST_NAME LIKE 'A%';
 patient_id | first_name | birth_date
------------+------------+------------
 p4         | Arjun      | 1970-04-22
 p7         | Amit       | 1979-03-26
(2 rows)
```

5.3.5 list all doctors name in sorted form

```
postgres=# SELECT DOCTOR.NAME FROM DOCTOR ORDER BY NAME;
          name
-----------------------
 Aditya Gupta
 Amandeep Singh Dhillon
 Ashish Sabharwal
 Bipin S Walia
 Gagan Gautam
 Gaurav Kharya
 Sandeep Batra
 Sanjay Sachdeva
 SKS Marya
 Surbhi Anand
(10 rows)
```

5.3.6 list all room no with patient id which associated with room.

```
postgres=# SELECT PATIENT.PATIENT_ID, ROOM.ROOM_NO FROM PATIENT RIGHT JOIN ROOM ON ROOM.PATIENT_ID = PATIENT.PATIENT_ID;
 patient_id | room_no
------------+---------
 p1         | rn1
 p2         | rn2
 p3         | rn3
 p4         | rn4
 p5         | rn5
 p6         | rn6
 p7         | rn7
 p8         | rn8
 p9         | rn9
 p10        | rn10
            | rn11
            | rn12
            | rn13
(13 rows)
```

5.3.7 list all insurence name which is taken by patient whose name start with B.

```
postgres=# SELECT INS_NAME FROM INSURANCE_claim WHERE PATIENT_ID IN (SELECT PATIENT_ID FROM PATIENT WHERE last_name LIKE 'B%');
    ins_name
---------------
 reliance general
(1 row)
```

5.3.8 find patient name and number and medicine detail given by doctor id d1

```
postgres=# SELECT p.FIRST_NAME, p.LAST_NAME, p.PHONE_NO, m.NAME AS MEDICINE_NAME, m.QUANTITY, m.PRICE FROM patient p JOI
N treats t ON p.PATIENT_ID = t.PATIENT_ID JOIN medicine m ON p.PATIENT_ID = m.PATIENT_ID WHERE t.DOCTOR_ID = 'd1';
 first_name | last_name |  phone_no  | medicine_name | quantity | price
------------+-----------+------------+---------------+----------+-------
 Narendra   | Modi      | 1231231231 | Paracetamol   |       50 |   100
(1 row)
```

5.3.9 find patient id who paid maximum charge of medicine

```
postgres=# SELECT PATIENT_ID FROM BILL WHERE MEDICINE_CHARGES = (SELECT MAX(MEDICINE_CHARGES) FROM BILL);
 patient_id
------------
 p1
(1 row)
```

5.3.10 find no of patient treat by each doctor

```
postgres=# SELECT d.DOCTOR_ID, d.NAME AS DOCTOR_NAME, COUNT(t.PATIENT_ID) AS NUMBER_OF_PATIENTS FROM DOCTOR d JOIN TREAT
S t ON d.DOCTOR_ID = t.DOCTOR_ID GROUP BY d.DOCTOR_ID, d.NAME;
 doctor_id |      doctor_name       | number_of_patients
-----------+------------------------+--------------------
 d4        | Gagan Gautam           |                  1
 d1        | Ashish Sabharwal       |                  1
 d7        | Bipin S Walia          |                  1
 d3        | Surbhi Anand           |                  1
 d2        | Sanjay Sachdeva        |                  1
 d8        | Amandeep Singh Dhillon |                  1
 d6        | Aditya Gupta           |                  1
 d9        | SKS Marya              |                  1
 d10       | Gaurav Kharya          |                  1
 d5        | Sandeep Batra          |                  1
(10 rows)
```

5.4 FUNCTION , TRIGGERS & CURSER:

5.4.1 Create one table whose name is extra which Contain data which is deleted from Payment table.

```
CREATE TABLE extra (
   PAYMENT_ID VARCHAR(50),
   BILL_NO VARCHAR(50),
   TYPE VARCHAR(50),
   STATUS VARCHAR(50)
);
CREATE FUNCTION my_trigger_function() RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO extra VALUES (OLD.PAYMENT_ID, OLD.BILL_NO, OLD.TYPE,
OLD.STATUS);
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER my_trigger_name
BEFORE DELETE ON PAYMENT
FOR EACH ROW
EXECUTE PROCEDURE my_trigger_function();
SELECT * FROM PAYMENT;
DELETE FROM PAYMENT WHERE PAYMENT_ID = 'pay1';
SELECT * FROM extra;
```

```
postgres=# CREATE TABLE extra (
postgres(#     PAYMENT_ID VARCHAR(50),
postgres(#     BILL_NO VARCHAR(50),
postgres(#     TYPE VARCHAR(50),
postgres(#     STATUS VARCHAR(50)
postgres(# );
CREATE TABLE
postgres=# CREATE FUNCTION my_trigger_function() RETURNS TRIGGER AS $$
postgres$# BEGIN
postgres$#     INSERT INTO extra VALUES (OLD.PAYMENT_ID, OLD.BILL_NO, OLD.TYPE, OLD.STATUS);
postgres$#     RETURN OLD;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# CREATE TRIGGER my_trigger_name
postgres-# BEFORE DELETE ON PAYMENT
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE my_trigger_function();
CREATE TRIGGER
postgres=# SELECT * FROM PAYMENT;
 payment_id | bill_no |    type    | status
------------+---------+------------+---------
 pay1       | b1      | cash       | paid
 pay2       | b2      | card       | pending
 pay3       | b3      | upi        | paid
 pay4       | b4      | netbanking | pending
 pay5       | b5      | cash       | paid
 pay6       | b6      | upi        | paid
 pay7       | b7      | cheque     | pending
 pay8       | b8      | upi        | paid
 pay9       | b9      | netbanking | pending
 pay10      | b10     | cash       | paid
(10 rows)


postgres=# DELETE FROM PAYMENT WHERE PAYMENT_ID = 'pay1';
DELETE 1
postgres=# SELECT * FROM extra;
 payment_id | bill_no | type | status
------------+---------+------+--------
 pay1       | b1      | cash | paid
(1 row)
```

5.4.2 store the previous record before updated of payment in other table which name is store.

```
CREATE TABLE store (
   PAYMENT_ID VARCHAR(50),
   BILL_NO VARCHAR(50),
   TYPE VARCHAR(50),
   STATUS VARCHAR(50)
);
CREATE FUNCTION trigger_function() RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO store VALUES (OLD.PAYMENT_ID, OLD.BILL_NO, OLD.TYPE,
OLD.STATUS);
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_name
BEFORE UPDATE ON PAYMENT
FOR EACH ROW
EXECUTE PROCEDURE trigger_function();
SELECT * FROM PAYMENT;
UPDATE PAYMENT SET STATUS = 'paid' WHERE BILL_NO = 'b4';
SELECT * FROM store;
```

```
postgres=# CREATE TABLE store (
postgres(#     PAYMENT_ID VARCHAR(50),
postgres(#     BILL_NO VARCHAR(50),
postgres(#     TYPE VARCHAR(50),
postgres(#     STATUS VARCHAR(50)
postgres(# );
CREATE TABLE
postgres=#
postgres=#
postgres=#
postgres=# CREATE FUNCTION trigger_function() RETURNS TRIGGER AS $$
postgres$# BEGIN
postgres$#     INSERT INTO store VALUES (OLD.PAYMENT_ID, OLD.BILL_NO, OLD.TYPE, OLD.STATUS);
postgres$#     RETURN OLD;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=#
postgres=# CREATE TRIGGER trigger_name
postgres-# BEFORE UPDATE ON PAYMENT
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE trigger_function();
CREATE TRIGGER
postgres=#
postgres=# SELECT * FROM PAYMENT;
 payment_id | bill_no |    type    | status
------------+---------+------------+---------
 pay2       | b2      | card       | pending
 pay3       | b3      | upi        | paid
 pay4       | b4      | netbanking | pending
 pay5       | b5      | cash       | paid
 pay6       | b6      | upi        | paid
 pay7       | b7      | cheque     | pending
 pay8       | b8      | upi        | paid
 pay9       | b9      | netbanking | pending
 pay10      | b10     | cash       | paid
(9 rows)


postgres=# UPDATE PAYMENT SET STATUS = 'paid' WHERE BILL_NO = 'b4';
UPDATE 1
postgres=# SELECT * FROM store;
 payment_id | bill_no |    type    | status
------------+---------+------------+---------
 pay4       | b4      | netbanking | pending
(1 row)
```

5.4.3 Create one function total which return total price of medicine as per quantity.

```
SELECT * FROM MEDICINE;
CREATE FUNCTION total(quantity INTEGER, price INTEGER)
RETURNS INTEGER
AS $$
BEGIN
   RETURN quantity * price;
END;
$$ LANGUAGE plpgsql;
SELECT total(50, 100);
```

```
postgres=# SELECT * FROM MEDICINE;
 medicine_id | patient_id | quantity | price |     name
-------------+------------+----------+-------+--------------
 M1          | p1         |       50 |   100 | Paracetamol
 M2          | p2         |        5 |  1000 | Ibuprofen
 M3          | p3         |       25 |   240 | Aspirin
 M4          | p4         |       30 |   100 | Metformin
 M5          | p5         |       80 |    75 | Amoxicillin
 M6          | p6         |        5 |    83 | Omeprazole
 M7          | p7         |       10 |   225 | Azithromycin
 M8          | p8         |       40 |   160 | Atorvastatin
 M9          | p9         |       25 |    93 | Clopidogrel
 M10         | p10        |       20 |    34 | Losartan
(10 rows)


postgres=# CREATE FUNCTION total(quantity INTEGER, price INTEGER)
postgres-# RETURNS INTEGER
postgres-# AS $$
postgres$# BEGIN
postgres$#     RETURN quantity * price;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=#
postgres=# SELECT total(50, 100);
 total
-------
  5000
(1 row)
```

5.4.4 Create one function which return insurance name, patient id & amount from INSURANCE_CLAIM table.

```
CREATE FUNCTION get_all_users()
RETURNS TABLE (P_ID VARCHAR(25), IN_NAME VARCHAR(25), AMT NUMERIC)
AS
$$
BEGIN
  RETURN QUERY SELECT PATIENT_ID, INS_NAME, AMOUNT FROM
INSURANCE_CLAIM;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM get_all_users();
```

```
postgres=# CREATE FUNCTION get_all_users()
postgres-# RETURNS TABLE (P_ID VARCHAR(25), IN_NAME VARCHAR(25), AMT NUMERIC)
postgres-# AS
postgres-# $$
postgres$# BEGIN
postgres$#   RETURN QUERY SELECT PATIENT_ID, INS_NAME, AMOUNT FROM INSURANCE_CLAIM;
postgres$# END;
postgres$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=# -- Test the function by selecting all records
postgres=# SELECT * FROM get_all_users();
 p_id |     in_name      |  amt
------+------------------+-------
 p1   | bajaj            | 10000
 p2   | lic              | 12000
 p3   | sbi life         | 11000
 p4   | reliance general | 15000
 p5   | bajaj            | 13000
 p6   | lic              | 18000
 p7   | bajaj            | 22000
 p8   | sbi life         | 14000
 p9   | reliance general |  7000
 p10  | bajaj            |  9000
(10 rows)
```

5.4.5 Create a cursor to select Doctor with SPECIALITY='Dermatology' and fetch the next Doctor.

```
DO $$
DECLARE
   cur CURSOR FOR SELECT * FROM DOCTOR WHERE SPECIALITY = 'Dermatology';
   rec RECORD;
BEGIN
   OPEN cur;
   FETCH NEXT FROM cur INTO rec;
   IF FOUND THEN
      RAISE NOTICE 'Doctor ID: %, Name: %, Phone: %', rec.DOCTOR_ID, rec.NAME,
rec.PHONE_NO;
   ELSE
      RAISE NOTICE 'No doctor found';
   END IF;
   CLOSE cur;
END $$;
```

```
postgres=# DO $$
postgres$# DECLARE
postgres$#     cur CURSOR FOR SELECT * FROM DOCTOR WHERE SPECIALITY = 'Dermatology';
postgres$#     rec RECORD;
postgres$# BEGIN
postgres$#     OPEN cur;
postgres$#     FETCH NEXT FROM cur INTO rec;
postgres$#     IF FOUND THEN
postgres$#         RAISE NOTICE 'Doctor ID: %, Name: %, Phone: %', rec.DOCTOR_ID, rec.NAME, rec.PHONE_NO;
postgres$#     ELSE
postgres$#         RAISE NOTICE 'No doctor found';
postgres$#     END IF;
postgres$#     CLOSE cur;
postgres$# END $$;
NOTICE:  Doctor ID: d5, Name: Sandeep Batra, Phone: 3434343434
DO
postgres=#
```

REFERENCE:

- GOOGLE

COMPILER:

- POSTGRES