Hospital Management System

A PROJECT REPORT

Submitted by

PRATEEK [Reg No:RA2211028010001]

Under the Guidance of

Dr.G.Geetha

Designation, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

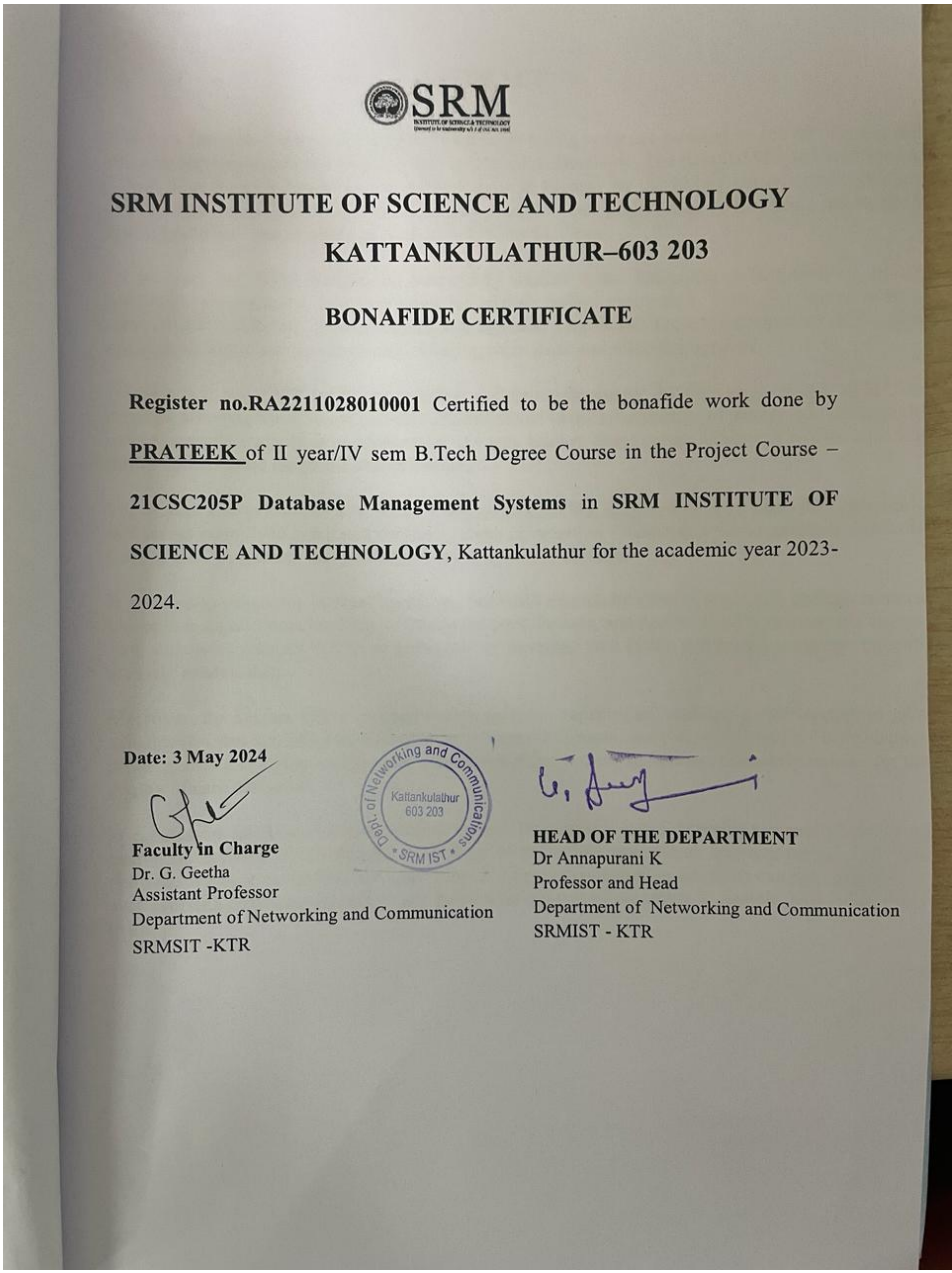
DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no.RA2211028010001 Certified to be the bonafide work done by **PRATEEK** of II year/IV sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date: 3 May 2024

[illegible]

Faculty in Charge
Dr. G. Geetha
Assistant Professor
Department of Networking and Communication
SRMSIT -KTR

Dept. of Networking and Communications
Kattankulathur
603 203
SRM IST

[illegible]

HEAD OF THE DEPARTMENT
Dr Annapurani K
Professor and Head
Department of Networking and Communication
SRMIST - KTR

ABSTRACT

In today's fast-paced world, the healthcare sector is facing immense pressure to deliver efficient and effective services to patients while managing resources effectively. The Hospital Management System (HMS) emerges as a crucial tool to address these challenges by integrating various administrative, clinical, and financial functions into a unified platform. Our project aims to develop a robust HMS that caters to the diverse needs of modern healthcare facilities.

At its core, our HMS focuses on automating routine tasks, enhancing communication among healthcare professionals, and improving the overall patient experience. Through features such as appointment scheduling, patient registration, and electronic health records management, the system streamlines administrative processes, reducing paperwork and minimizing errors.

Furthermore, our HMS facilitates seamless collaboration among different departments within the hospital, allowing for efficient resource allocation and better coordination of patient care. Integration with diagnostic equipment and laboratory systems enables real-time access to test results, ensuring timely and accurate diagnosis and treatment.

The system also prioritizes data security and confidentiality, employing robust encryption protocols and access controls to safeguard sensitive patient information. Compliance with regulatory standards such as HIPAA ensures that patient privacy is upheld at all times.

In addition to enhancing internal operations, our HMS extends its benefits to patients through features like online appointment booking, telemedicine consultations, and personalized health records access. This empowers patients to take an active role in managing their health and fosters a stronger patient-provider relationship.

Moreover, the system offers comprehensive analytics capabilities, enabling administrators to gain valuable insights into hospital performance, resource utilization, and patient outcomes. These insights drive informed decision-making and continuous improvement initiatives, ultimately enhancing the quality of care delivered.

In conclusion, our Hospital Management System represents a significant step towards modernizing healthcare delivery by leveraging technology to streamline operations, improve patient care, and optimize resource utilization. By embracing this innovative solution, hospitals can stay ahead in an ever-evolving healthcare landscape and fulfill their mission of providing accessible, high-quality care to all.

TABLE OF CONTENTS

CHAPTER 1

Entity-Relationship Model - An ER (Entity-Relationship) model is like a blueprint for organizing information in a database. It shows what things (entities) exist, what they're like, and how they're connected. It helps design a system where data is structured and relationships between different pieces of information are clear .

Components of ER Model

- ◈ Entity sets
- ◈ Relationship sets
- ◈ Attributes

Entity sets

In an Entity-Relationship (ER) model, an entity set is a collection or group of similar entities. An entity represents a real-world object, and the entity set is a set of all instances or occurrences of that type of entity. ◈ For example, consider an entity set "Student." Each individual student, such as John, Sarah, and Mike, is an instance of the "Student" entity set. The entity set defines the common properties or attributes shared by all students, like name, student ID, and date of birth.

Attributes

In an Entity-Relationship (ER) model, attributes are the properties or characteristics that describe an entity within an entity set. Attributes provide details about the entities and help define their properties.

◈ For example, consider an entity set "Person" with attributes like "Name," "Age," and "Address."

Attributes can be categorized into different types -

1. Simple attributes: Represents a single, indivisible piece of data (e.g., "Age" or "Name").

2. Composite attributes: Comprises multiple simple attributes. For instance, an attribute "Address" might be composed of sub-attributes such as "Street," "City," and "Zip Code."

3. Multi valued attributes: Can hold multiple values for a single entity. For instance, an entity "Person" may have a multi-valued attribute "Phone Numbers" to represent multiple contact numbers.

4. Derived attributes: Derived from other attributes in the database. It can be calculated or derived from existing attributes. For example, "Age" can be derived from the "Date of Birth" attribute.

5. Single valued attributes: Consider an entity set "Book" with a single-valued attribute "Title." Each book entity in this set would have a unique title, and the "Title" attribute would hold only one value for each book.

6. Key attributes: In an entity set "Employee," a key attribute might be "EmployeeID." Each employee has a unique employee ID, and if selected as the primary key, it ensures the uniqueness of each employee within the set.

Relationships

In an Entity-Relationship (ER) diagram, a relationship represents an association or connection between two or more entities. It illustrates how the data in one entity is related to the data in another entity.

Different types of relationships in ER diagram

One-to-One (1:1) Relationship: Description: Each entity in the relationship is associated with at most one instance of the other entity, and vice versa. Example: A "Person" entity is associated with one "Passport," and a "Passport" is associated with one "Person."

One-to-Many (1:N) Relationship: Description: Each entity in the relationship can be associated with multiple instances of the other entity, but each instance of the other entity is associated with at most one entity. Example: A "Department" entity is associated with multiple "Employees," but each "Employee" is associated with only one "Department."

Many-to-One (N:1) Relationship: Description: The reverse of a one-to-many relationship; each instance of one entity can be associated with multiple instances of the other entity, but each instance of the other entity is associated with at most one entity. Example: Multiple "Employees" are associated with one "Manager," but each "Manager" is associated with at most one "Employee."

Many-to-Many (N:N) Relationship: Description: Each entity in the relationship can be associated with multiple instances of the other entity, and vice versa. Example: A "Student" entity is associated with multiple "Courses," and a "Course" is associated with multiple "Students."

ENTITY AND ATTRIBUTES

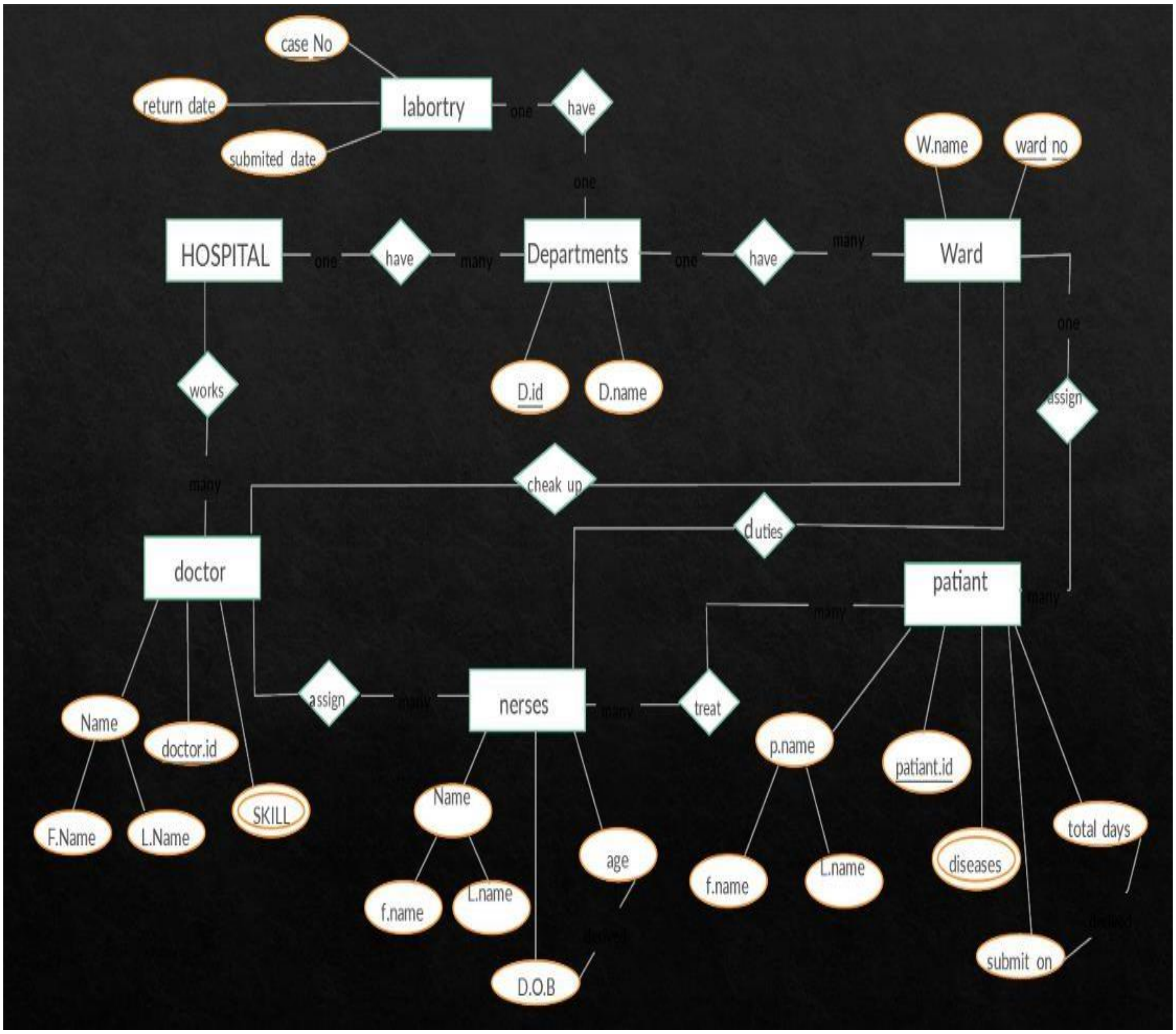
- Hospital Attributes: PhoneNumber, Address, Hospital_id, Name
- Doctors Attributes: doctor_id ,experience ,rating ,name
- Molecular(weak entity) Attributes: Accuracy , symptoms ,no._of_days
- Patients Attributes: phone_no. ,patient_id ,Name ,Address
- Primary Patient_id ,symptoms ,safety measures ,NOP
- Secondary Symptoms ,safety measures ,NOP ,patient_id
- Critical Symptoms ,safety measures ,NOP ,patient_id
- Rooms room_charge , room_id ,room_type ,NOP ,Availability
- PatientStatus StatusId ,StatusName(eg., Primary ,Secondary ,Critical

Relationship

◈ Hospital-Doctors: One-to-Many (A hospital can have many doctors, but a doctor works at one hospital).

◈ Doctors-Tests: Many-to-Many (A doctor can be associated with many tests, and a test can be associated with many doctors).

◈ Tests-Molecular: One-to-One (Each test can have one molecular test associated with it).



CHAPTER 2

Relational Model

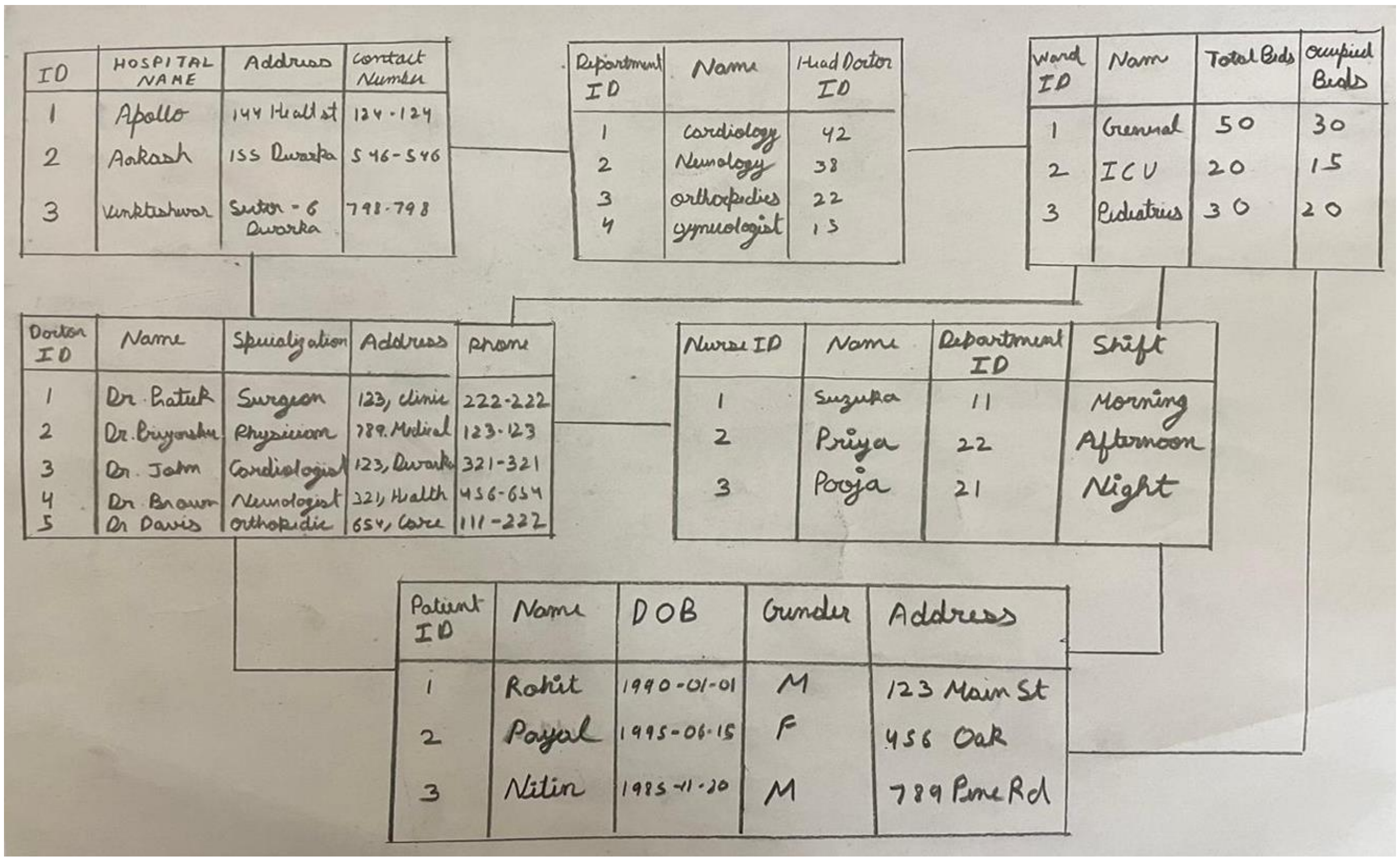
The relational model is a type of database model that stores data in tables. This model was proposed by E.F. Codd in 1970. The main components of the relational model are:

◈ Relations: In the relational model, a table is called a relation. A relation is a set of tuples. Each tuple represents an object and its attributes.

◈ Attributes: An attribute is a property or characteristic of an object. In a table, each column represents an attribute.

◈ Tuples: A tuple is a row in a table that represents a set of related data. Each tuple in a table is unique.

◈ Keys: A key is an attribute or a set of attributes that uniquely identifies a tuple in a relation. There are different types of keys like primary key, foreign key, etc.



ID	HOSPITAL NAME	Address	Contact Number
1	Apollo	144 Health st	124-124
2	Aakash	155 Dwarka	546-546
3	Venkteshwar	Sector - 6 Dwarka	798-798

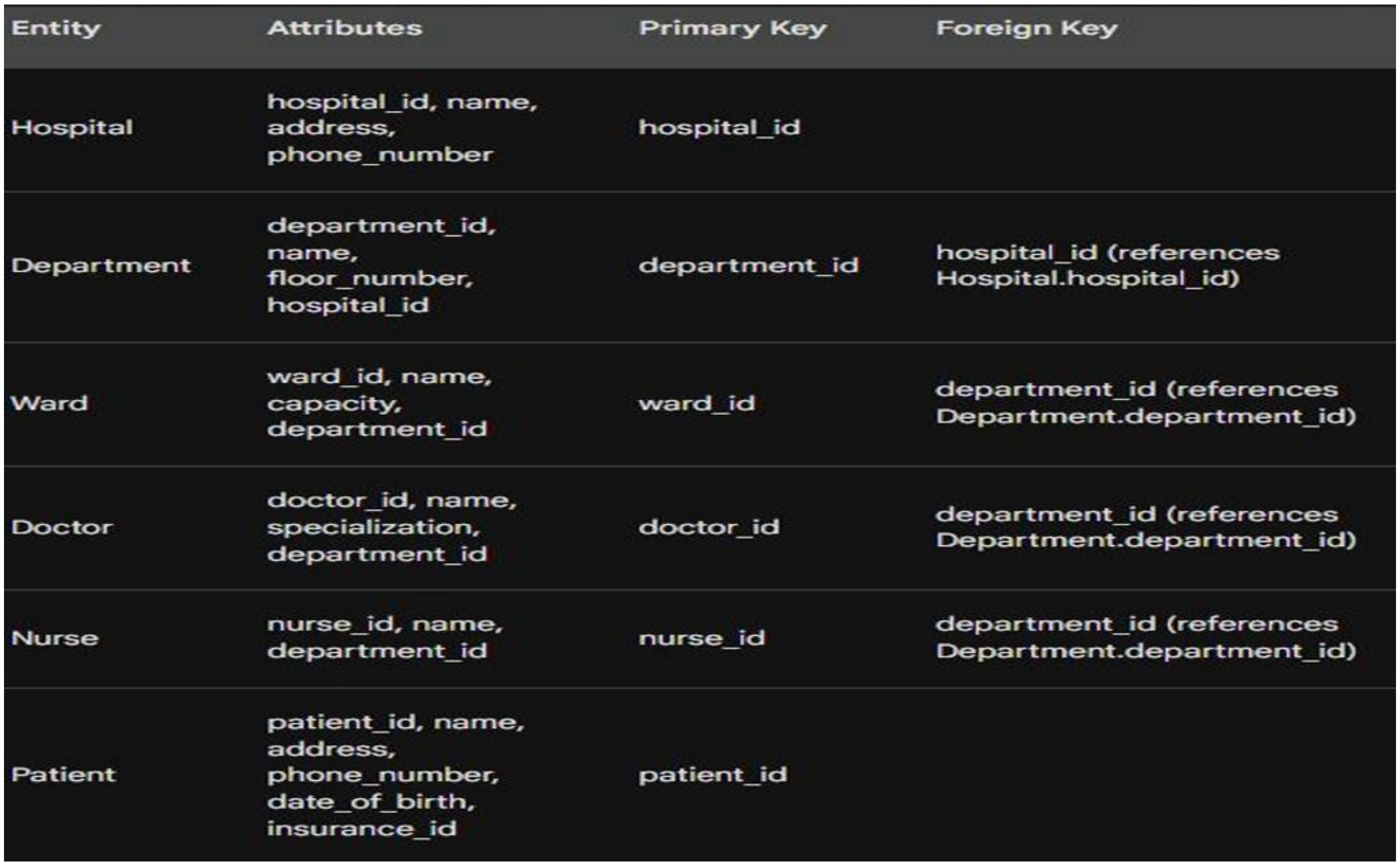
Department ID	Name	Head Doctor ID
1	Cardiology	42
2	Neurology	38
3	orthopedics	22
4	gynecologist	15

Ward ID	Name	Total Beds	Occupied Beds
1	General	50	30
2	ICU	20	15
3	Pediatrics	30	20

Doctor ID	Name	Specialization	Address	Phone
1	Dr. Prateek	Surgeon	123, clinic	222-222
2	Dr. Priyanshu	Physician	789, Medical	123-123
3	Dr. John	Cardiologist	123, Dwarka	321-321
4	Dr Brown	Neurologist	321, Health	456-654
5	Dr Davis	orthopedic	654, Care	111-222

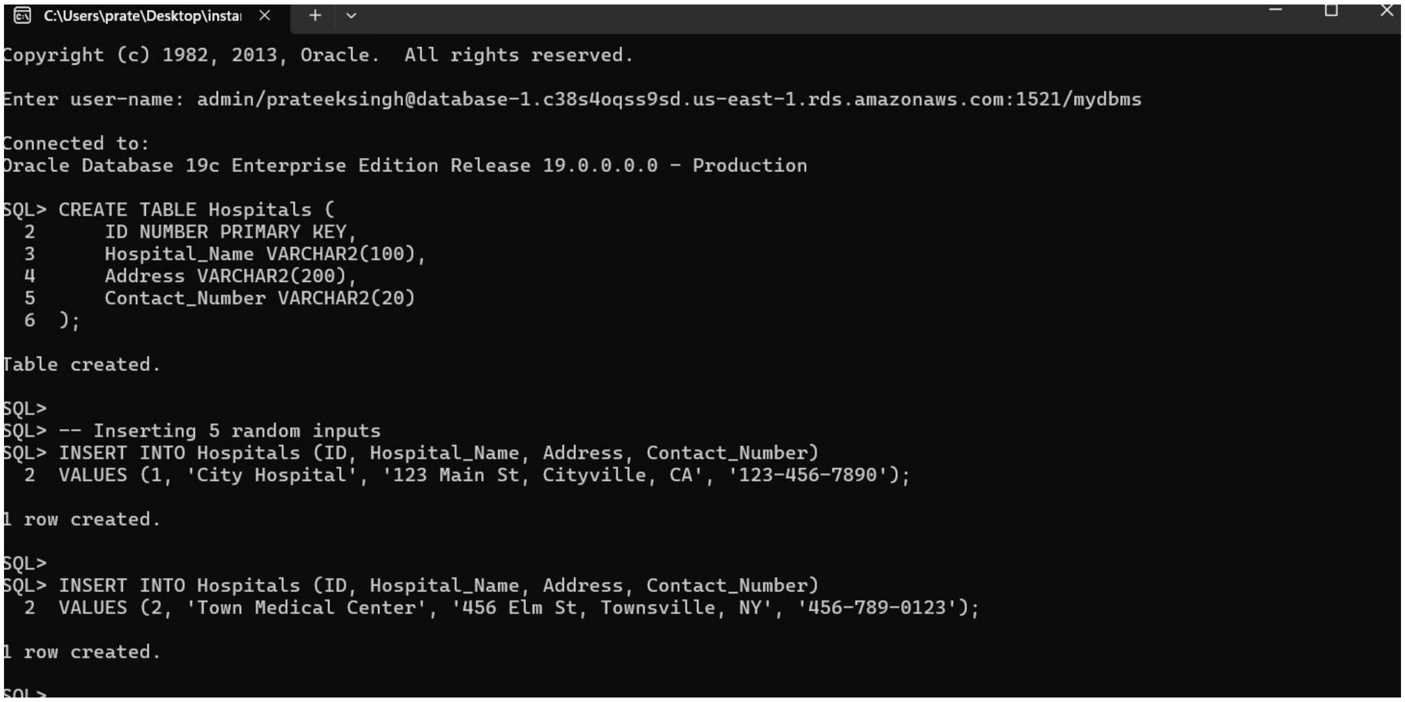
Nurse ID	Name	Department ID	Shift
1	Suzuka	11	Morning
2	Priya	22	Afternoon
3	Pooja	21	Night

Patient ID	Name	DOB	Gender	Address
1	Rohit	1990-01-01	M	123 Main St
2	Payal	1995-06-15	F	456 Oak
3	Nitin	1985-11-20	M	789 Pine Rd

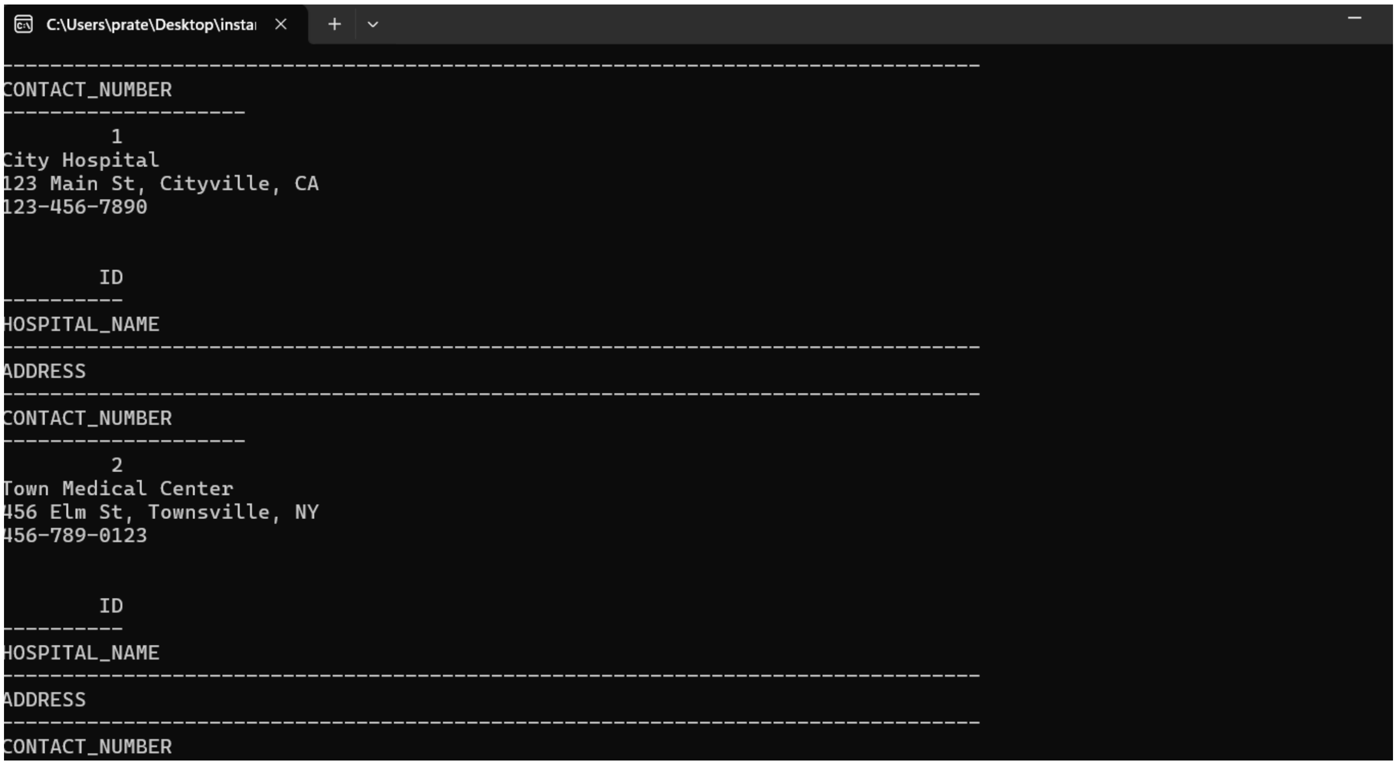


Entity	Attributes	Primary Key	Foreign Key
Hospital	hospital_id, name, address, phone_number	hospital_id	
Department	department_id, name, floor_number, hospital_id	department_id	hospital_id (references Hospital.hospital_id)
Ward	ward_id, name, capacity, department_id	ward_id	department_id (references Department.department_id)
Doctor	doctor_id, name, specialization, department_id	doctor_id	department_id (references Department.department_id)
Nurse	nurse_id, name, department_id	nurse_id	department_id (references Department.department_id)
Patient	patient_id, name, address, phone_number, date_of_birth, insurance_id	patient_id	

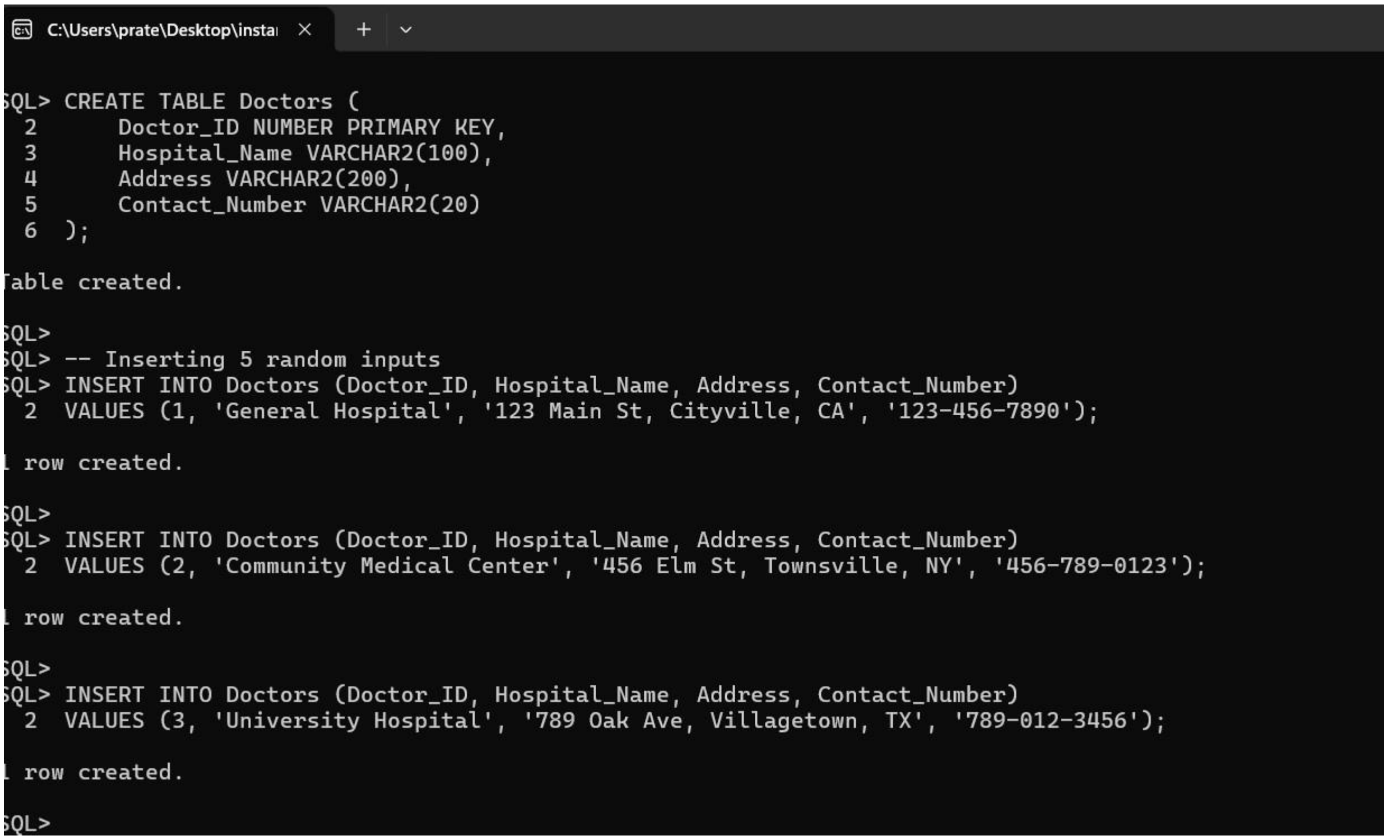
CHAPTER 3



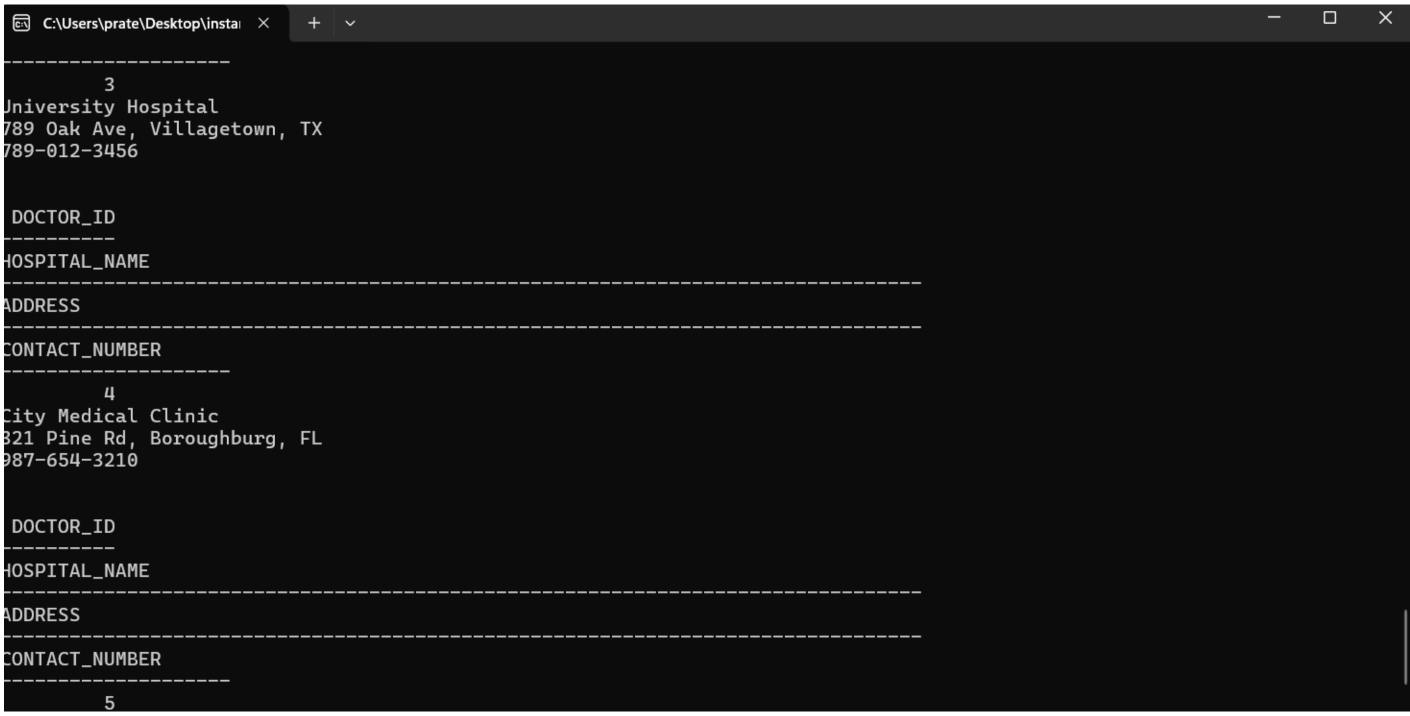
```
Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/prateeksingh@database-1.c38s4oqss9sd.us-east-1.rds.amazonaws.com:1521/mydbms

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> CREATE TABLE Hospitals (
  2      ID NUMBER PRIMARY KEY,
  3      Hospital_Name VARCHAR2(100),
  4      Address VARCHAR2(200),
  5      Contact_Number VARCHAR2(20)
  6  );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
  2  VALUES (1, 'City Hospital', '123 Main St, Cityville, CA', '123-456-7890');

1 row created.

SQL>
SQL> INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
  2  VALUES (2, 'Town Medical Center', '456 Elm St, Townsville, NY', '456-789-0123');

1 row created.

SQL>
```



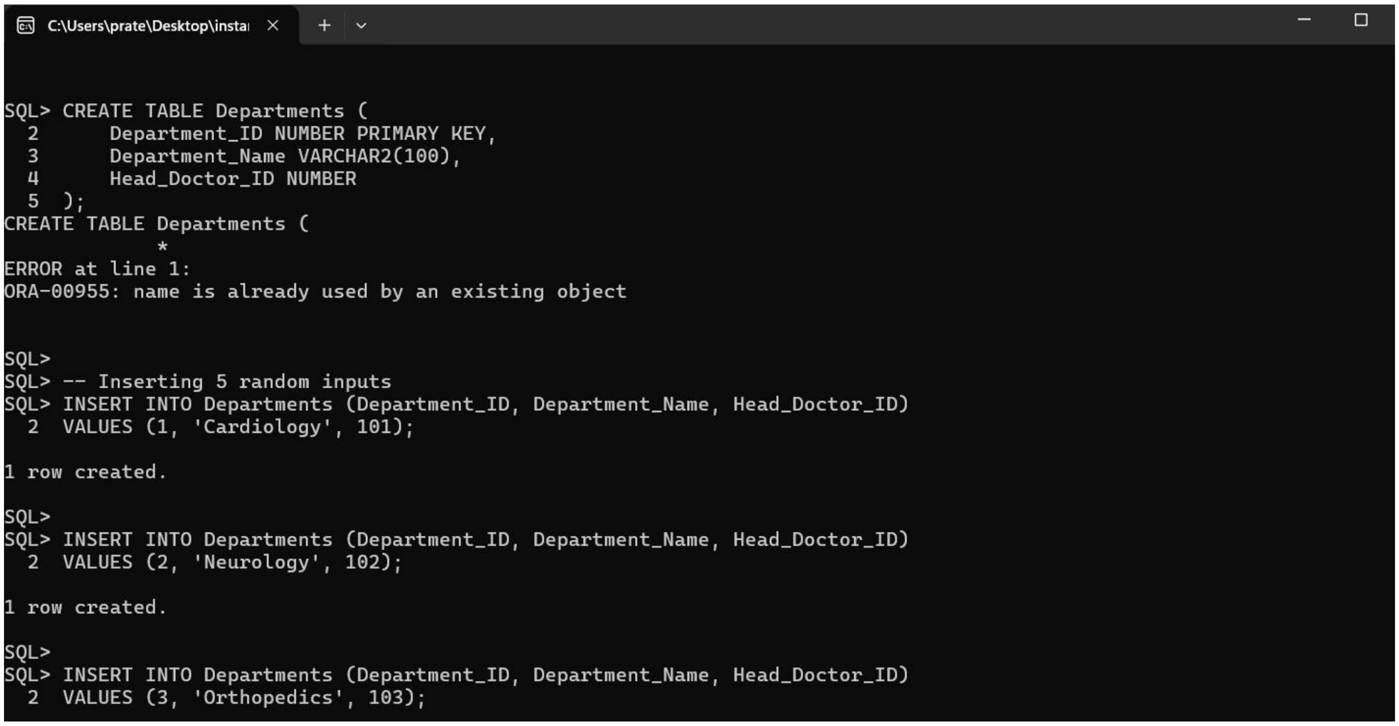
```
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
         1
City Hospital
123 Main St, Cityville, CA
123-456-7890


        ID
----------
HOSPITAL_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
         2
Town Medical Center
456 Elm St, Townsville, NY
456-789-0123


        ID
----------
HOSPITAL_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CONTACT_NUMBER
```



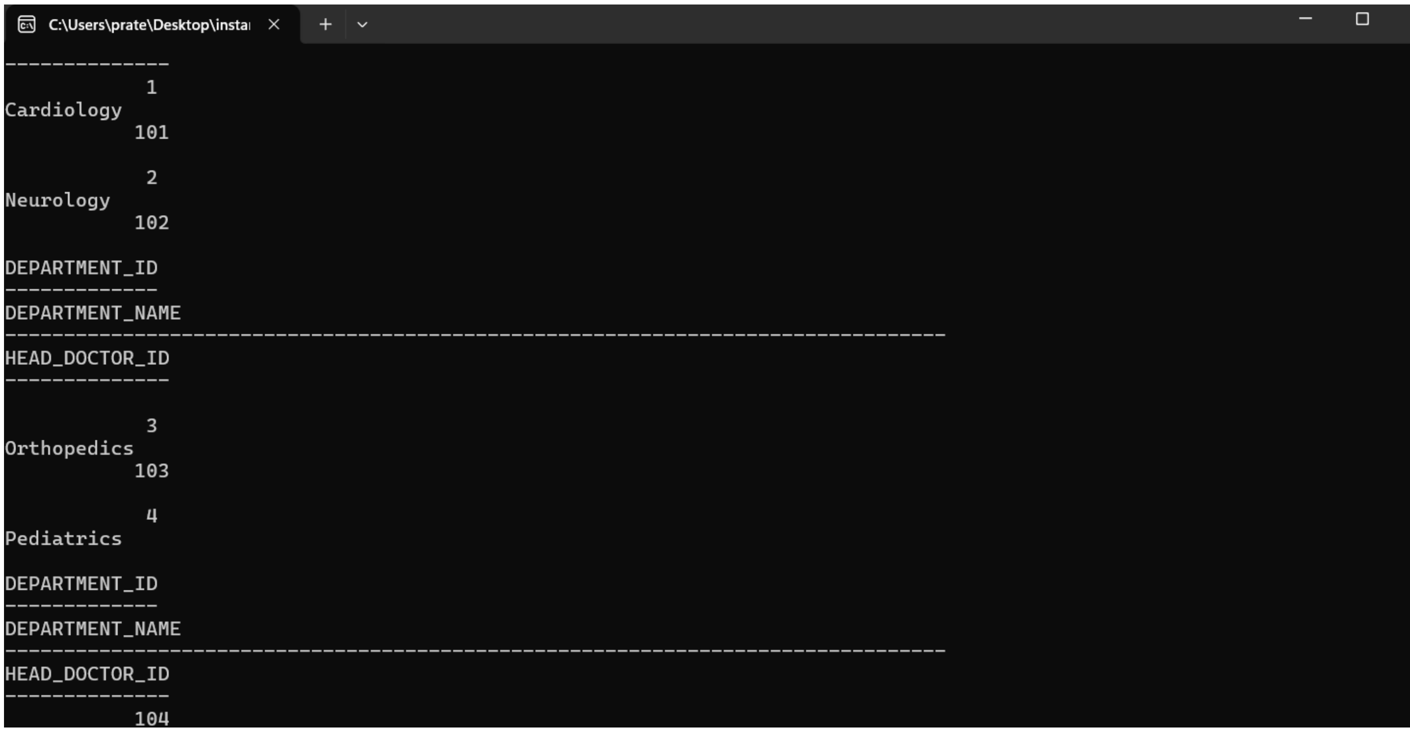
```
SQL> CREATE TABLE Doctors (
  2      Doctor_ID NUMBER PRIMARY KEY,
  3      Hospital_Name VARCHAR2(100),
  4      Address VARCHAR2(200),
  5      Contact_Number VARCHAR2(20)
  6  );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
  2  VALUES (1, 'General Hospital', '123 Main St, Cityville, CA', '123-456-7890');

1 row created.

SQL>
SQL> INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
  2  VALUES (2, 'Community Medical Center', '456 Elm St, Townsville, NY', '456-789-0123');

1 row created.

SQL>
SQL> INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
  2  VALUES (3, 'University Hospital', '789 Oak Ave, Villagetown, TX', '789-012-3456');

1 row created.

SQL>
```



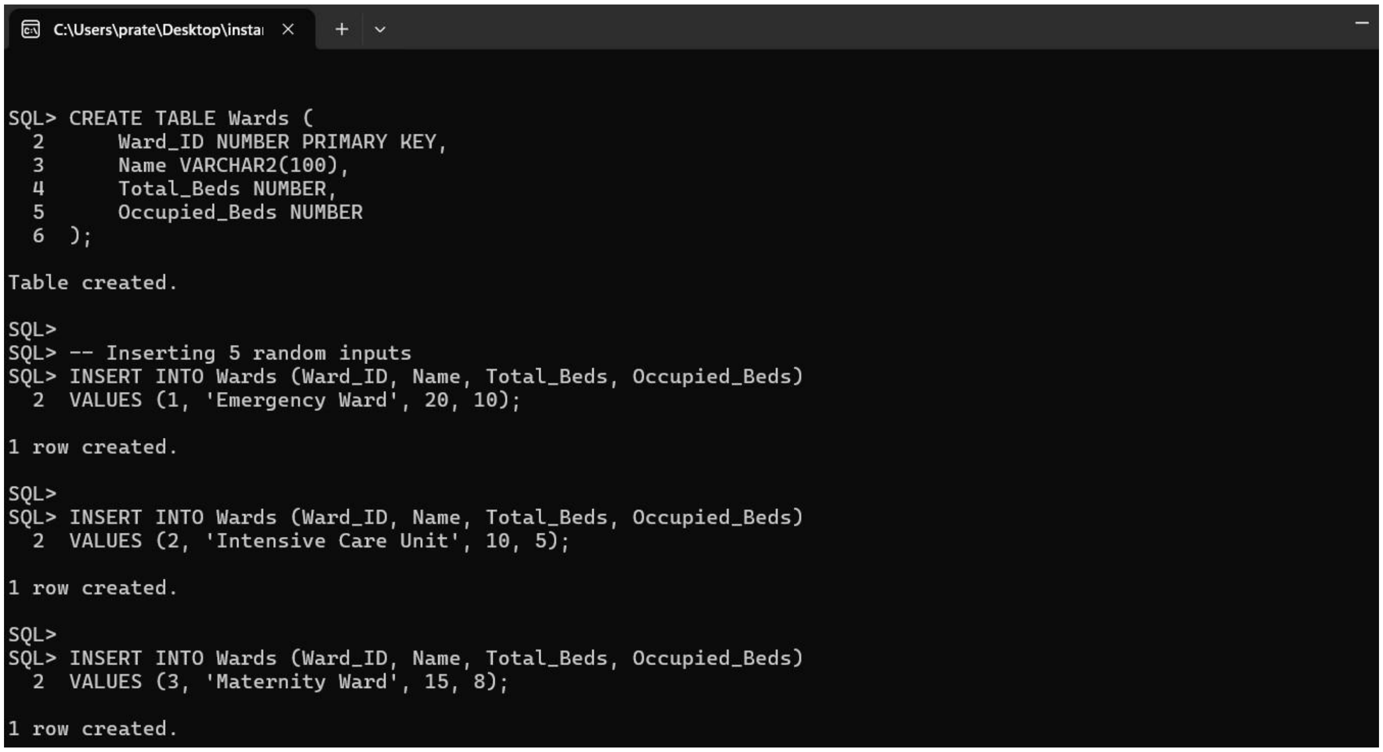
```
C:\Users\prate\Desktop\insta
--------------------
         3
University Hospital
789 Oak Ave, Villagetown, TX
789-012-3456


 DOCTOR_ID
----------
HOSPITAL_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
         4
City Medical Clinic
321 Pine Rd, Boroughburg, FL
987-654-3210


 DOCTOR_ID
----------
HOSPITAL_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
         5
```



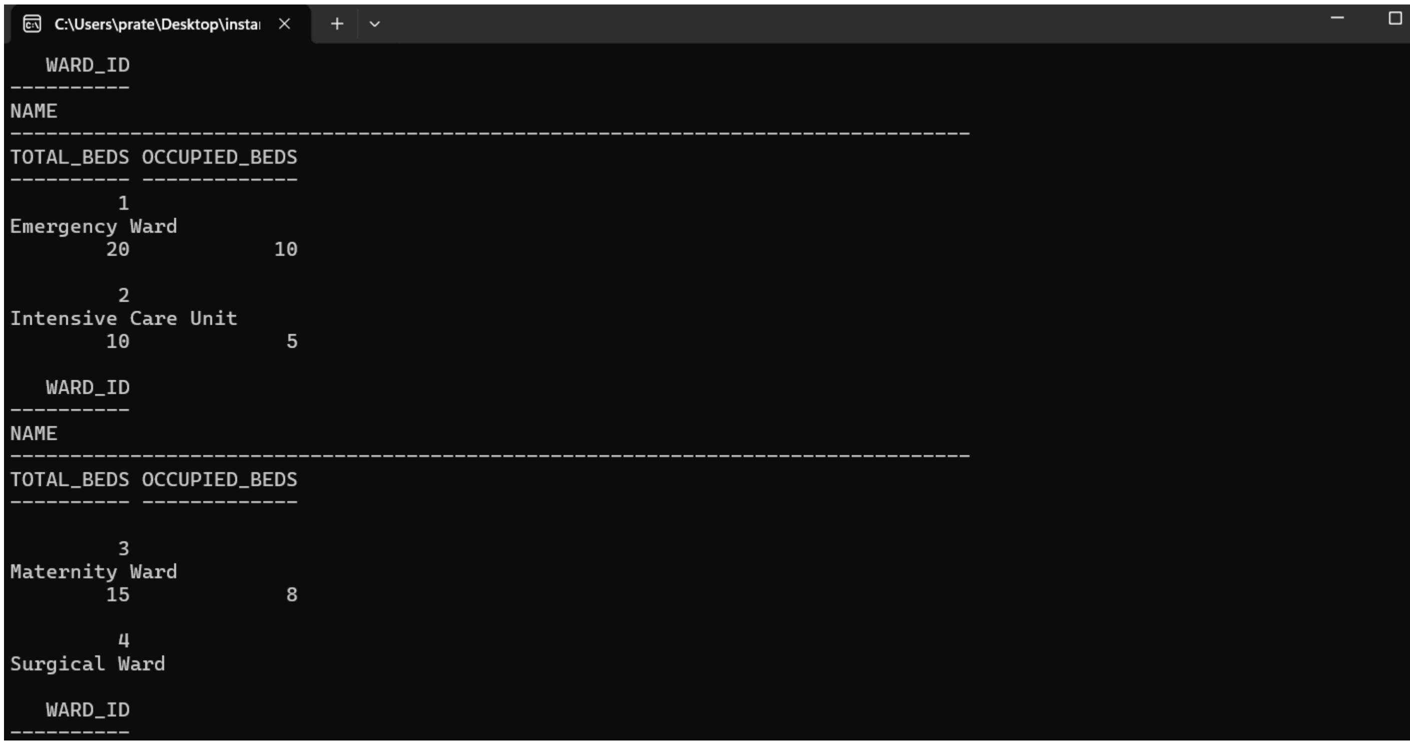
```
C:\Users\prate\Desktop\insta

SQL> CREATE TABLE Departments (
  2      Department_ID NUMBER PRIMARY KEY,
  3      Department_Name VARCHAR2(100),
  4      Head_Doctor_ID NUMBER
  5  );
CREATE TABLE Departments (
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (1, 'Cardiology', 101);

1 row created.

SQL>
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (2, 'Neurology', 102);

1 row created.

SQL>
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (3, 'Orthopedics', 103);
```



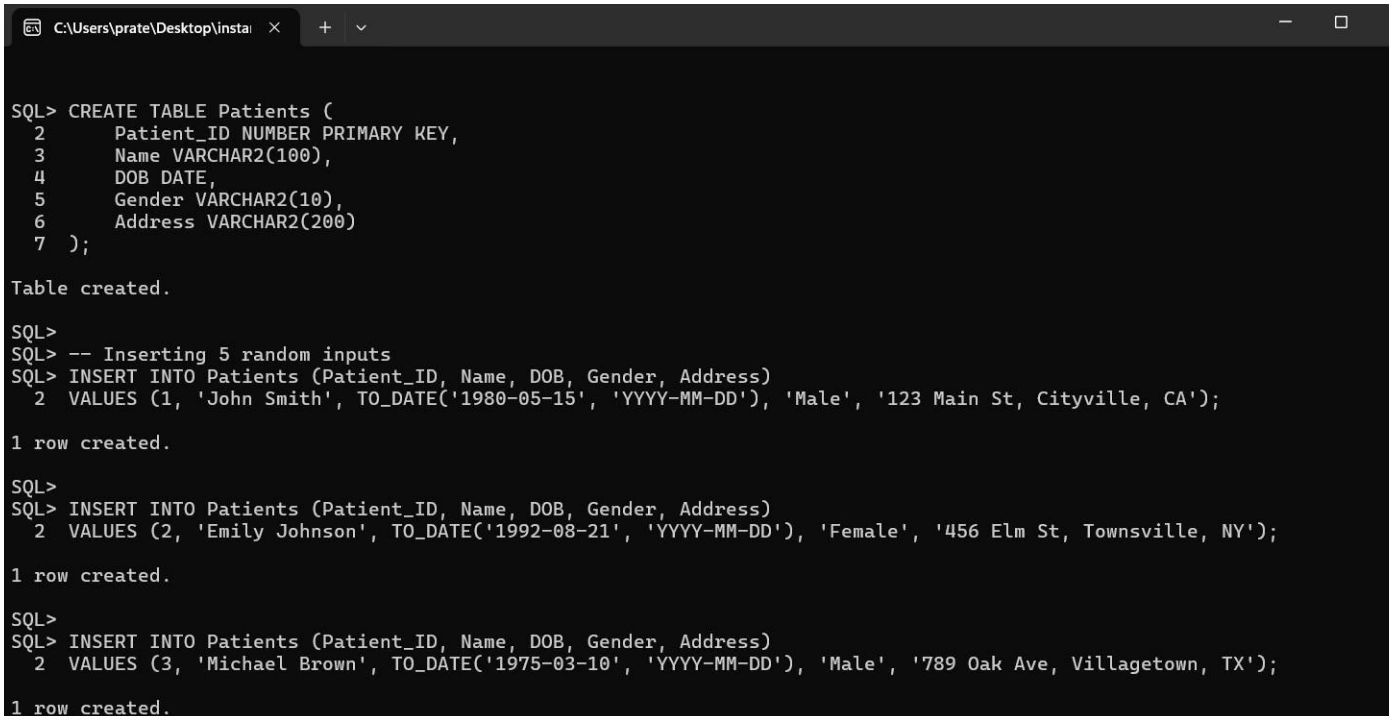
```
--------------
            1
Cardiology
          101

            2
Neurology
          102

DEPARTMENT_ID
-------------
DEPARTMENT_NAME
--------------------------------------------------------------------------------
HEAD_DOCTOR_ID
--------------

            3
Orthopedics
          103

            4
Pediatrics

DEPARTMENT_ID
-------------
DEPARTMENT_NAME
--------------------------------------------------------------------------------
HEAD_DOCTOR_ID
--------------
          104
```



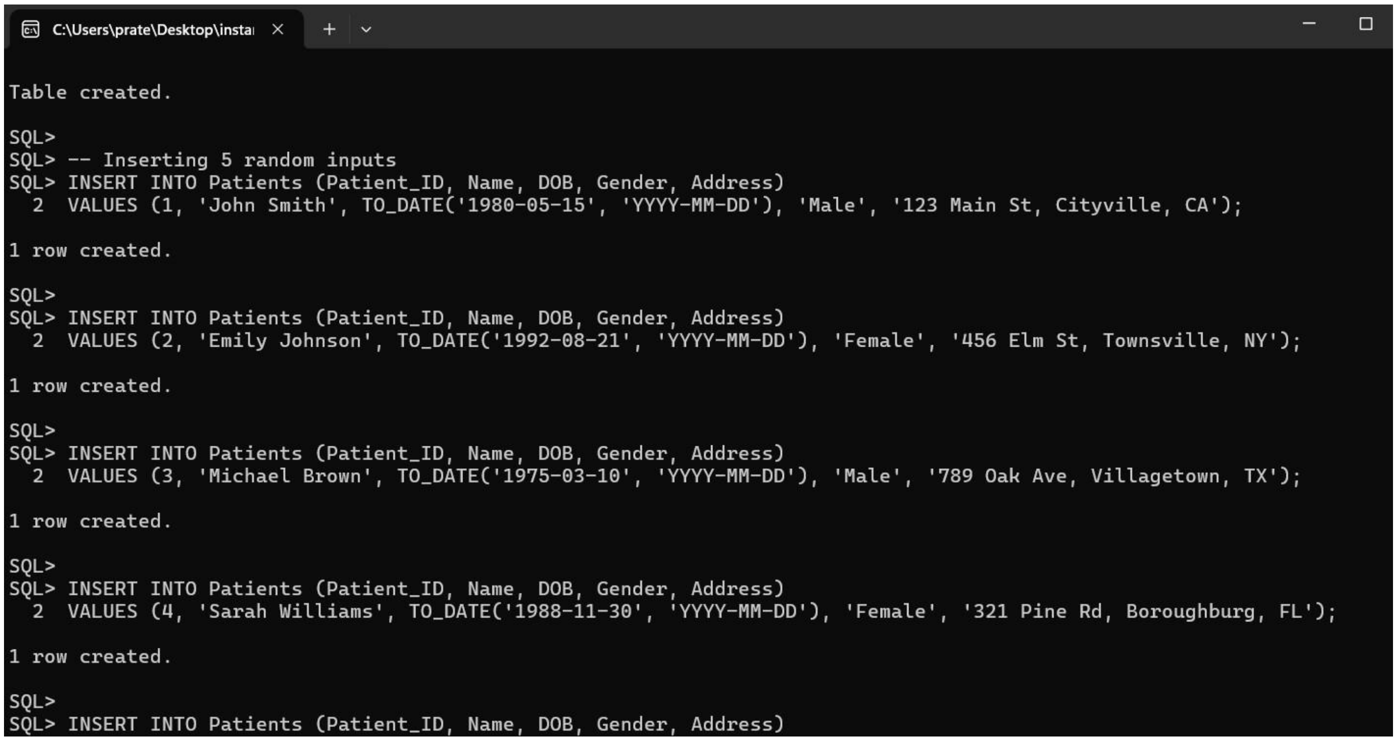
```
SQL> CREATE TABLE Wards (
  2      Ward_ID NUMBER PRIMARY KEY,
  3      Name VARCHAR2(100),
  4      Total_Beds NUMBER,
  5      Occupied_Beds NUMBER
  6  );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
  2  VALUES (1, 'Emergency Ward', 20, 10);

1 row created.

SQL>
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
  2  VALUES (2, 'Intensive Care Unit', 10, 5);

1 row created.

SQL>
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
  2  VALUES (3, 'Maternity Ward', 15, 8);

1 row created.
```



```
   WARD_ID
----------
NAME
--------------------------------------------------------------------------------
TOTAL_BEDS OCCUPIED_BEDS
---------- -------------
         1
Emergency Ward
        20            10

         2
Intensive Care Unit
        10             5

   WARD_ID
----------
NAME
--------------------------------------------------------------------------------
TOTAL_BEDS OCCUPIED_BEDS
---------- -------------

         3
Maternity Ward
        15             8

         4
Surgical Ward

   WARD_ID
----------
```

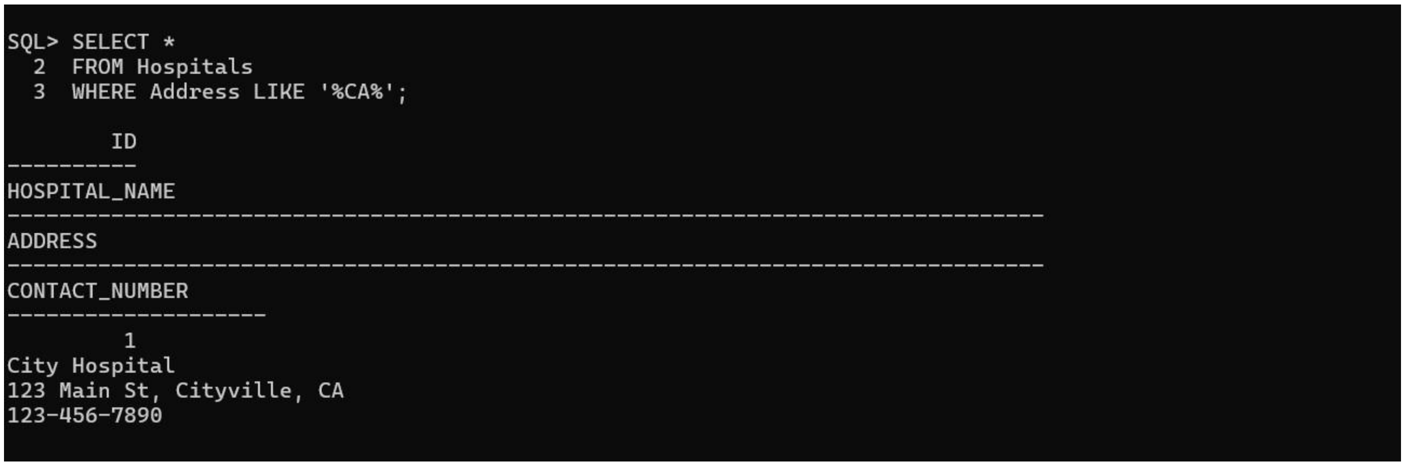


```
SQL> CREATE TABLE Patients (
         Patient_ID NUMBER PRIMARY KEY,
         Name VARCHAR2(100),
         DOB DATE,
         Gender VARCHAR2(10),
         Address VARCHAR2(200)
     );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
     VALUES (1, 'John Smith', TO_DATE('1980-05-15', 'YYYY-MM-DD'), 'Male', '123 Main St, Cityville, CA');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
     VALUES (2, 'Emily Johnson', TO_DATE('1992-08-21', 'YYYY-MM-DD'), 'Female', '456 Elm St, Townsville, NY');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
     VALUES (3, 'Michael Brown', TO_DATE('1975-03-10', 'YYYY-MM-DD'), 'Male', '789 Oak Ave, Villagetown, TX');

1 row created.
```



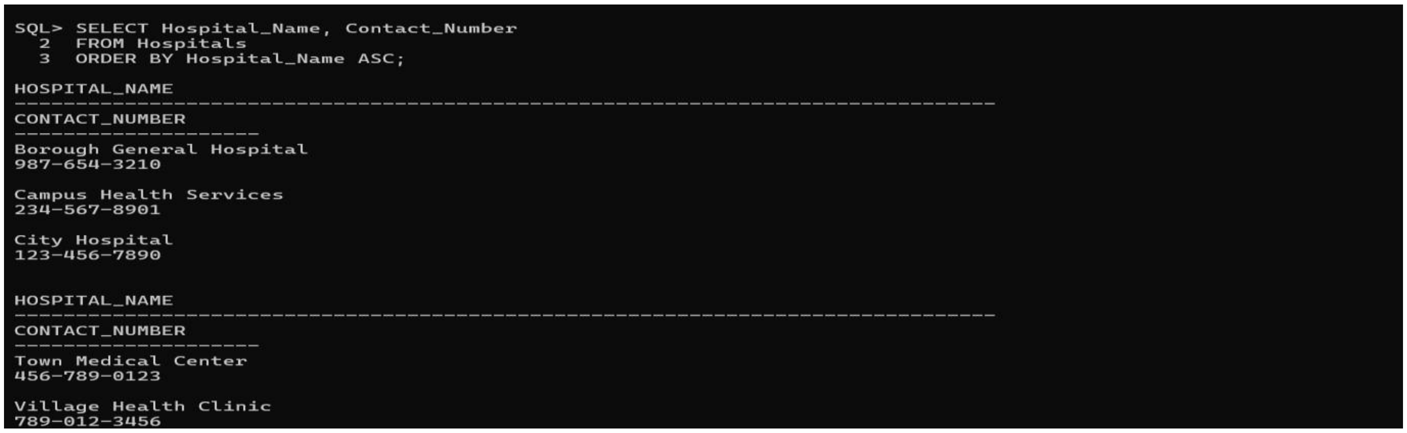
```
Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (1, 'John Smith', TO_DATE('1980-05-15', 'YYYY-MM-DD'), 'Male', '123 Main St, Cityville, CA');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (2, 'Emily Johnson', TO_DATE('1992-08-21', 'YYYY-MM-DD'), 'Female', '456 Elm St, Townsville, NY');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (3, 'Michael Brown', TO_DATE('1975-03-10', 'YYYY-MM-DD'), 'Male', '789 Oak Ave, Villagetown, TX');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (4, 'Sarah Williams', TO_DATE('1988-11-30', 'YYYY-MM-DD'), 'Female', '321 Pine Rd, Boroughburg, FL');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
```

SUBQUERY



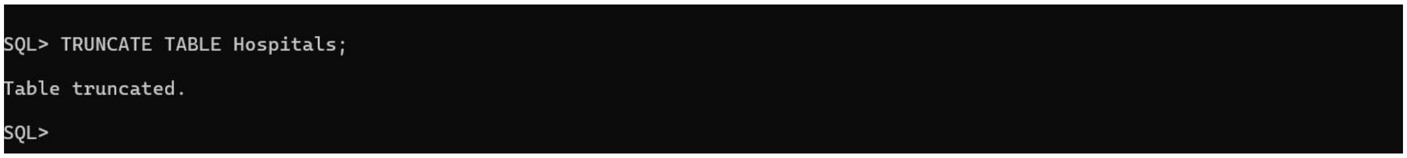
```
SQL> SELECT *
  2  FROM Hospitals
  3  WHERE Address LIKE '%CA%';

        ID
----------
HOSPITAL_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
         1
City Hospital
123 Main St, Cityville, CA
123-456-7890
```

QUERY



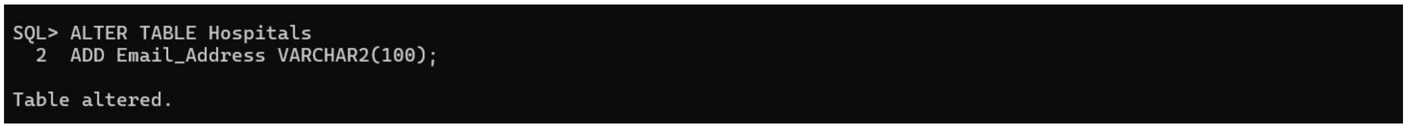
```
SQL> SELECT Hospital_Name, Contact_Number
  2  FROM Hospitals
  3  ORDER BY Hospital_Name ASC;

HOSPITAL_NAME
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
Borough General Hospital
987-654-3210

Campus Health Services
234-567-8901

City Hospital
123-456-7890


HOSPITAL_NAME
--------------------------------------------------------------------------------
CONTACT_NUMBER
--------------------
Town Medical Center
456-789-0123

Village Health Clinic
789-012-3456
```

TRUNCATE



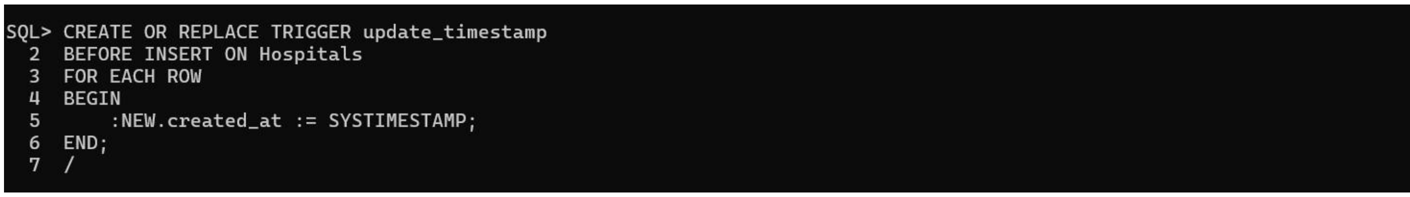
```
SQL> TRUNCATE TABLE Hospitals;

Table truncated.

SQL>
```

ALTER TABLE



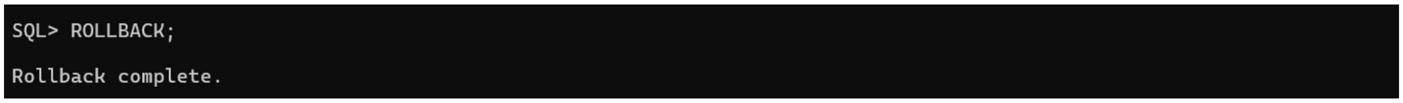
```
SQL> ALTER TABLE Hospitals
  2  ADD Email_Address VARCHAR2(100);

Table altered.
```

TRIGGER



```
SQL> CREATE OR REPLACE TRIGGER update_timestamp
  2  BEFORE INSERT ON Hospitals
  3  FOR EACH ROW
  4  BEGIN
  5      :NEW.created_at := SYSTIMESTAMP;
  6  END;
  7  /
```

ROLLBACK



```
SQL> ROLLBACK;

Rollback complete.
```

CHAPTER 4

Pitfalls and Dependencies:

1) Repeating Groups: The table contains repeating groups related to doctors and patients. Each hospital can have multiple doctors and patients associated with it.

2) Partial Dependencies: Attributes like Doctor_Name, Doctor_Specialization, Patient_ID, Patient_Name, Admission_Date, and Room_Number are dependent on the Hospital_ID. They are partially dependent on the primary key.

3) Transitive Dependencies: Attributes like Doctor_Name, Doctor_Specialization, Patient_ID, Patient_Name, Admission_Date, and Room_Number are transitively dependent on Hospital_Name and Location through Hospital_ID.

Normalization Steps:

First Normal Form (1NF): To remove repeating groups, we'll split the table into separate tables for hospital, doctor, and patient.

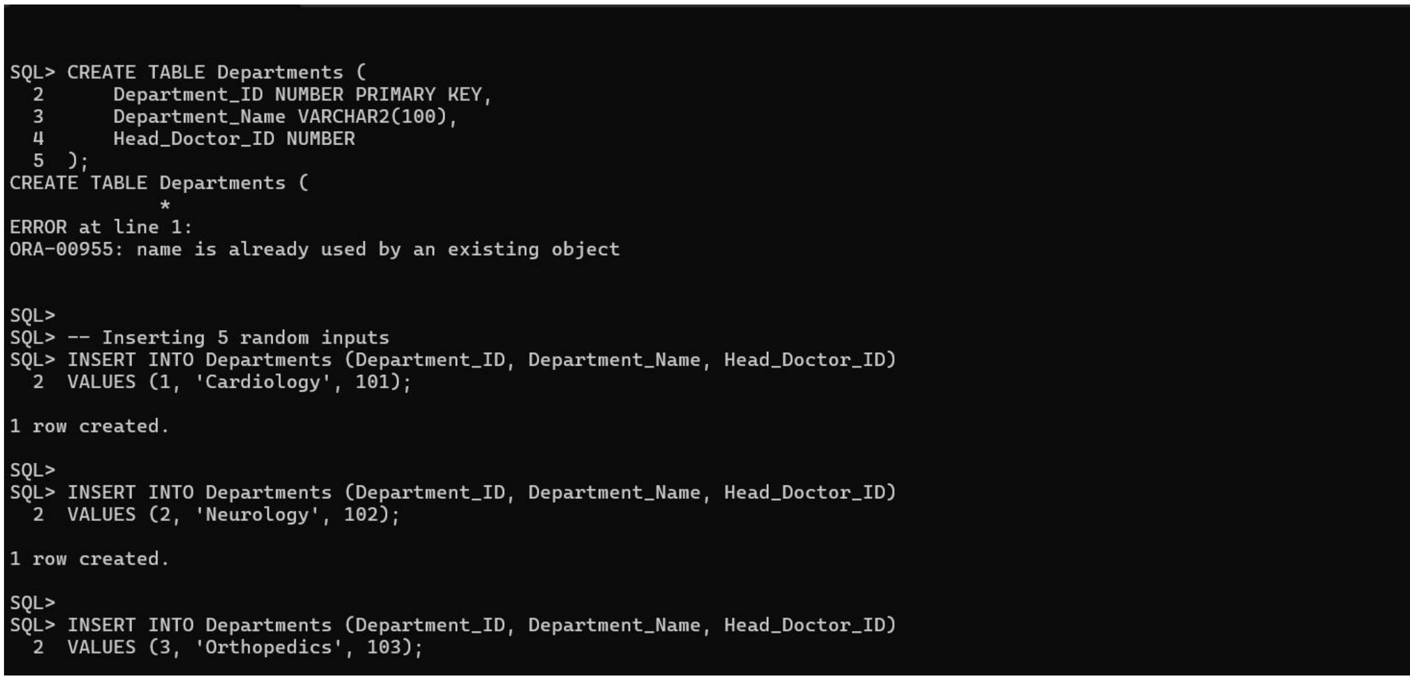
Second Normal Form (2NF): There are no partial dependencies in the first normal form. So, no further action needed for this step.

Third Normal Form (3NF): We'll remove transitive dependencies by ensuring that non-key attributes are dependent only on the primary key.

Boyce-Codd Normal Form (BCNF): Ensure that every determinant is a candidate key.

Fourth Normal Form (4NF): Ensure that there are no multi-valued dependencies.

Fifth Normal Form (5NF): Ensure that every join dependency is a consequence of the candidate keys.



```
SQL> CREATE TABLE Departments (
  2      Department_ID NUMBER PRIMARY KEY,
  3      Department_Name VARCHAR2(100),
  4      Head_Doctor_ID NUMBER
  5  );
CREATE TABLE Departments (
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (1, 'Cardiology', 101);

1 row created.

SQL>
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (2, 'Neurology', 102);

1 row created.

SQL>
SQL> INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
  2  VALUES (3, 'Orthopedics', 103);
```

Pitfalls Analysis: Redundancy: Departments might have repeated information if stored improperly.

Update Anomalies: Inconsistencies may occur if, for instance, a department's details need to be updated in multiple places.

Insertion Anomalies: Difficulties in inserting data due to missing information.

Deletion Anomalies: Unintended data loss if a department is removed.

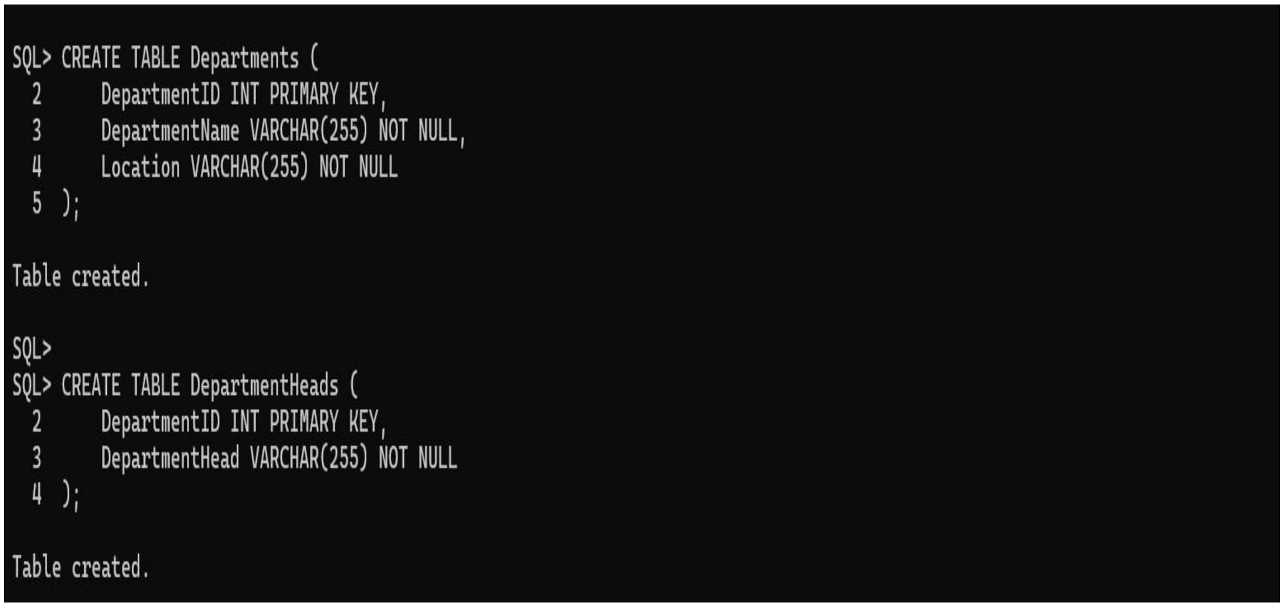
Lack of Flexibility: Hard to adapt to changes in department attributes or requirements. Identifying Dependencies: Functional Dependencies: Determine which attributes depend on each other. Transitive Dependencies: Identify attributes that depend on other non-key attributes.

Normalization: Normalization involves breaking down the tables to minimize redundancy and dependency issues. The most common normal forms are:

First Normal Form (1NF): Eliminate repeating groups and ensure each attribute contains atomic values.

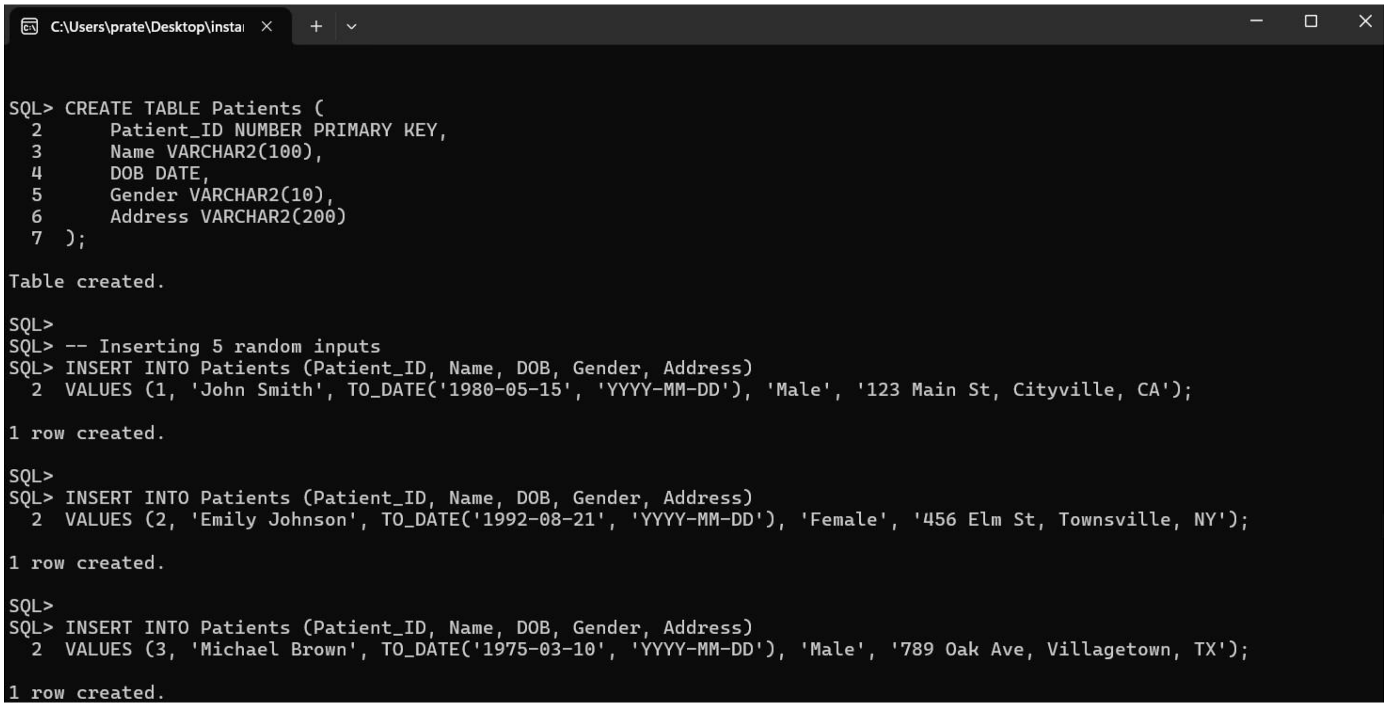
Second Normal Form (2NF): Ensure that non-key attributes are fully dependent on the primary key. This often involves breaking tables into smaller ones.

Third Normal Form (3NF): Remove transitive dependencies so that non-key attributes depend only on the primary key.



```
SQL> CREATE TABLE Departments (
  2      DepartmentID INT PRIMARY KEY,
  3      DepartmentName VARCHAR(255) NOT NULL,
  4      Location VARCHAR(255) NOT NULL
  5  );

Table created.

SQL>
SQL> CREATE TABLE DepartmentHeads (
  2      DepartmentID INT PRIMARY KEY,
  3      DepartmentHead VARCHAR(255) NOT NULL
  4  );

Table created.
```

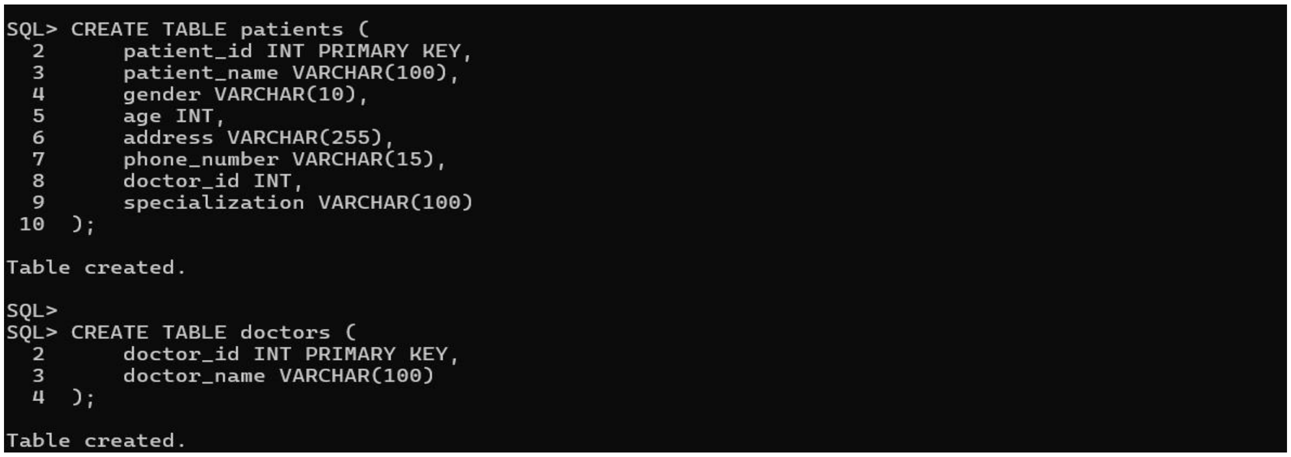
PATIENTS



```
SQL> CREATE TABLE Patients (
  2      Patient_ID NUMBER PRIMARY KEY,
  3      Name VARCHAR2(100),
  4      DOB DATE,
  5      Gender VARCHAR2(10),
  6      Address VARCHAR2(200)
  7  );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (1, 'John Smith', TO_DATE('1980-05-15', 'YYYY-MM-DD'), 'Male', '123 Main St, Cityville, CA');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (2, 'Emily Johnson', TO_DATE('1992-08-21', 'YYYY-MM-DD'), 'Female', '456 Elm St, Townsville, NY');

1 row created.

SQL>
SQL> INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)
  2  VALUES (3, 'Michael Brown', TO_DATE('1975-03-10', 'YYYY-MM-DD'), 'Male', '789 Oak Ave, Villagetown, TX');

1 row created.
```

1NF

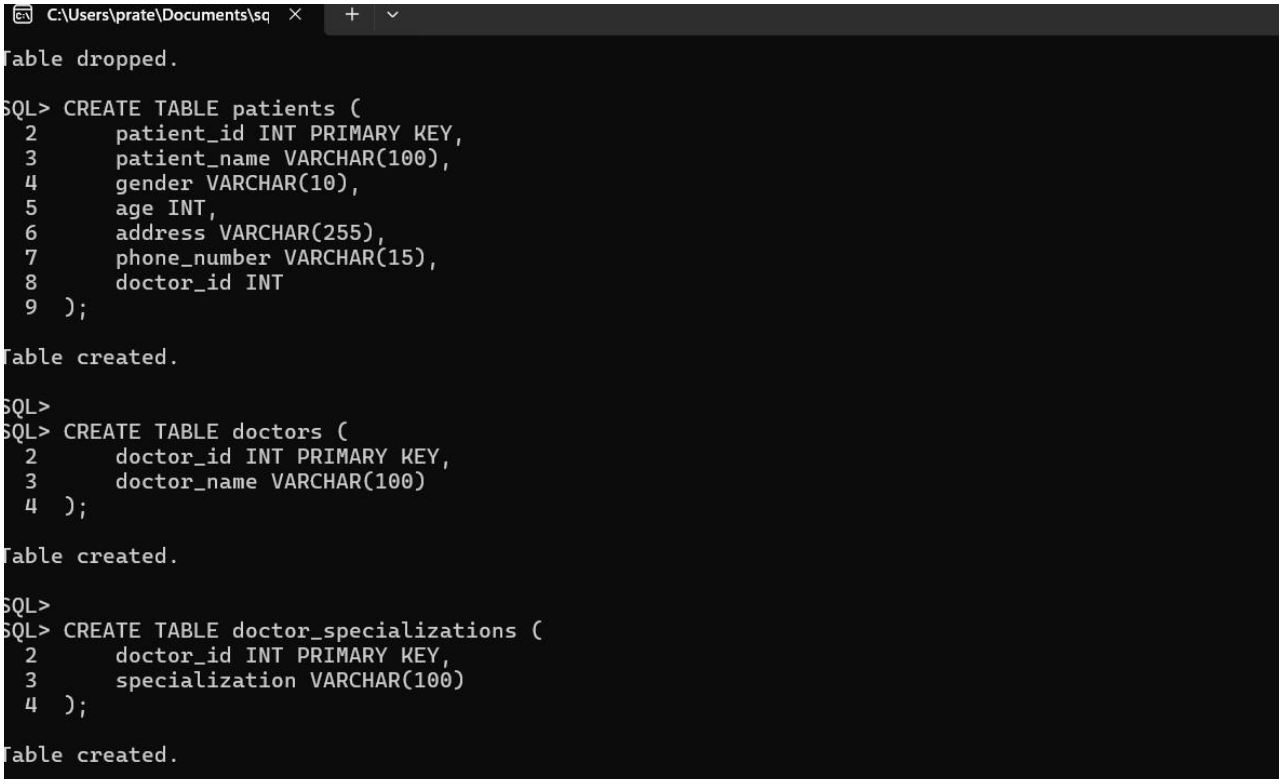
Eliminate repeating groups and create a separate table for them.



```
SQL> CREATE TABLE patients (
         patient_id INT PRIMARY KEY,
         patient_name VARCHAR(100),
         gender VARCHAR(10),
         age INT,
         address VARCHAR(255),
         phone_number VARCHAR(15),
         doctor_id INT,
         specialization VARCHAR(100)
     );

Table created.

SQL>
SQL> CREATE TABLE doctors (
         doctor_id INT PRIMARY KEY,
         doctor_name VARCHAR(100)
     );

Table created.
```

2NF

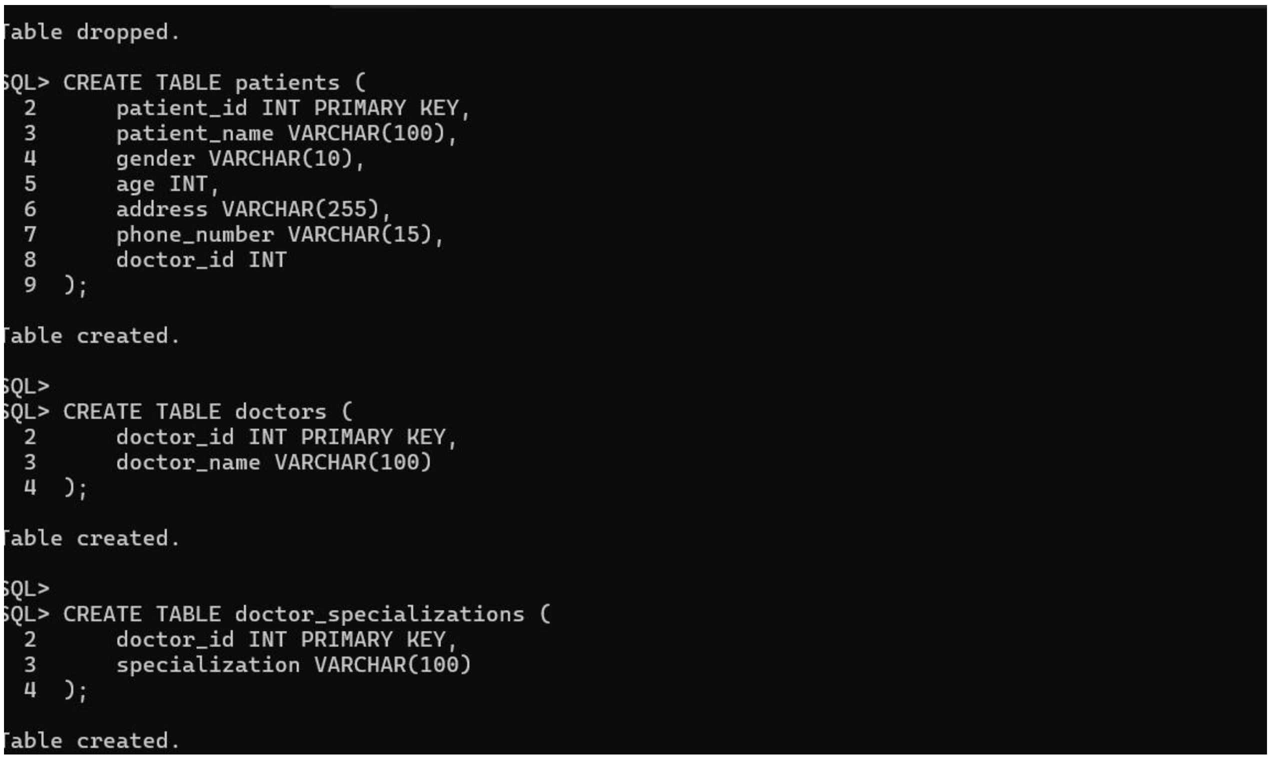
Ensure that non-key attributes are fully functional dependent on the primary key.



```
C:\Users\prate\Documents\sq

able dropped.

QL> CREATE TABLE patients (
        patient_id INT PRIMARY KEY,
        patient_name VARCHAR(100),
        gender VARCHAR(10),
        age INT,
        address VARCHAR(255),
        phone_number VARCHAR(15),
        doctor_id INT
    );

able created.

QL>
QL> CREATE TABLE doctors (
        doctor_id INT PRIMARY KEY,
        doctor_name VARCHAR(100)
    );

able created.

QL>
QL> CREATE TABLE doctor_specializations (
        doctor_id INT PRIMARY KEY,
        specialization VARCHAR(100)
    );

able created.
```

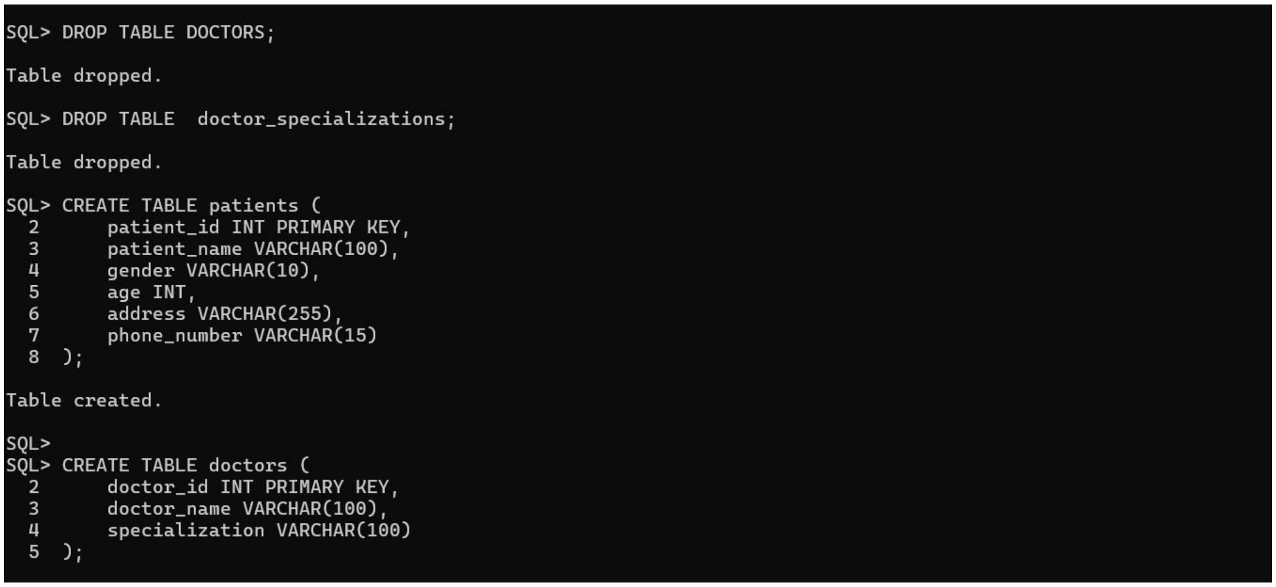
3NF

Ensure that non-key attributes are fully functional dependent on the primary key.



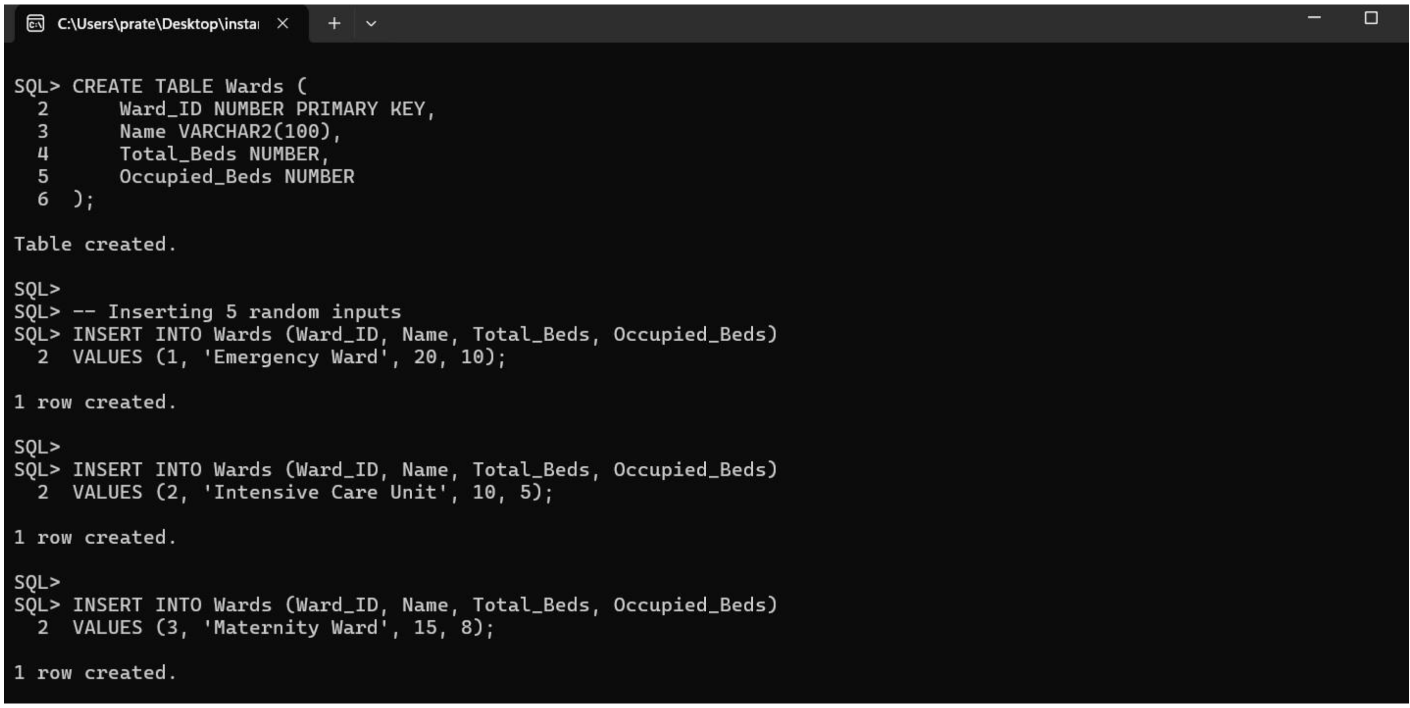
```
Table dropped.

SQL> CREATE TABLE patients (
         patient_id INT PRIMARY KEY,
         patient_name VARCHAR(100),
         gender VARCHAR(10),
         age INT,
         address VARCHAR(255),
         phone_number VARCHAR(15),
         doctor_id INT
     );

Table created.

SQL>
SQL> CREATE TABLE doctors (
         doctor_id INT PRIMARY KEY,
         doctor_name VARCHAR(100)
     );

Table created.

SQL>
SQL> CREATE TABLE doctor_specializations (
         doctor_id INT PRIMARY KEY,
         specialization VARCHAR(100)
     );

Table created.
```

4NF



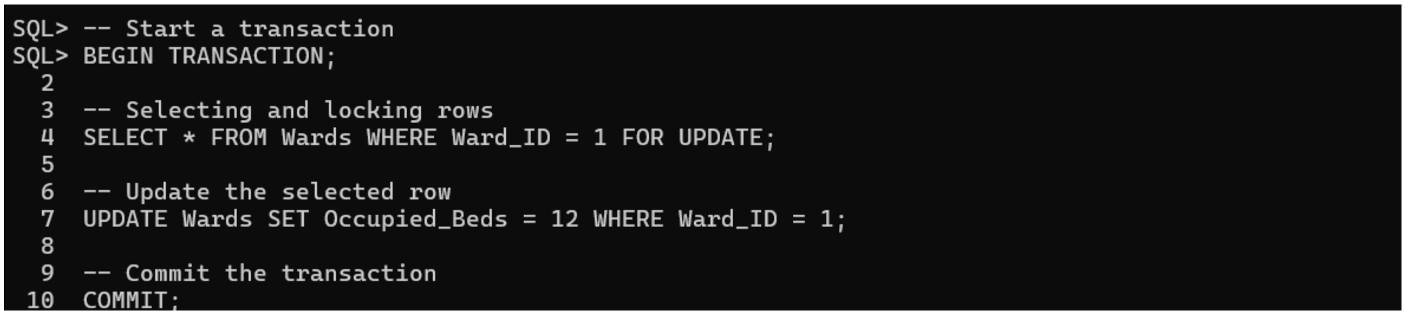
```
SQL> DROP TABLE DOCTORS;

Table dropped.

SQL> DROP TABLE  doctor_specializations;

Table dropped.

SQL> CREATE TABLE patients (
         patient_id INT PRIMARY KEY,
         patient_name VARCHAR(100),
         gender VARCHAR(10),
         age INT,
         address VARCHAR(255),
         phone_number VARCHAR(15)
     );

Table created.

SQL>
SQL> CREATE TABLE doctors (
         doctor_id INT PRIMARY KEY,
         doctor_name VARCHAR(100),
         specialization VARCHAR(100)
     );
```

CHAPTER 5



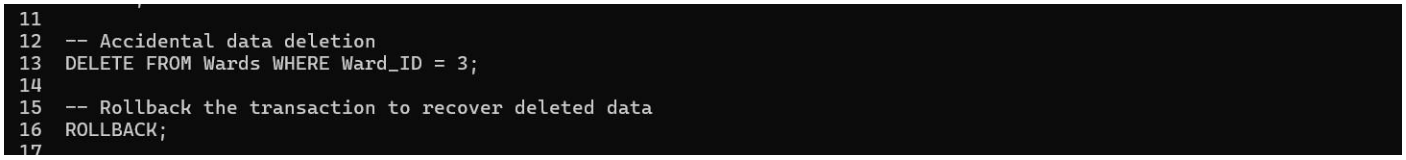
```
SQL> CREATE TABLE Wards (
         Ward_ID NUMBER PRIMARY KEY,
         Name VARCHAR2(100),
         Total_Beds NUMBER,
         Occupied_Beds NUMBER
     );

Table created.

SQL>
SQL> -- Inserting 5 random inputs
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
     VALUES (1, 'Emergency Ward', 20, 10);

1 row created.

SQL>
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
     VALUES (2, 'Intensive Care Unit', 10, 5);

1 row created.

SQL>
SQL> INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
     VALUES (3, 'Maternity Ward', 15, 8);

1 row created.
```

To implement concurrency control in the database, you can use locking mechanisms to ensure that transactions are executed safely without interference from other transactions. In Oracle, you can use various types of locks such as row-level locks, table-level locks, or even application-level locks using programming constructs like PL/SQL.



```
SQL> -- Start a transaction
SQL> BEGIN TRANSACTION;

     -- Selecting and locking rows
     SELECT * FROM Wards WHERE Ward_ID = 1 FOR UPDATE;

     -- Update the selected row
     UPDATE Wards SET Occupied_Beds = 12 WHERE Ward_ID = 1;

     -- Commit the transaction
     COMMIT;
```

In this example, the SELECT ... FOR UPDATE statement locks the selected row in the Wards table, preventing other transactions from modifying it until the current transaction is committed or rolled back. Now, regarding recovery mechanisms, Oracle provides various features for ensuring data recovery and integrity, such as: Backup and Recovery: Regularly backing up the database and transaction logs, and using Oracle Recovery Manager (RMAN) for efficient backup and recovery operations. Flashback Technology: Using features like Flashback Query, Flashback Table, and Flashback Transaction Query to recover lost data or revert the database to a previous state. Redo Logs and Undo Logs: Oracle maintains redo logs and undo logs to ensure transactional consistency and recoverability. Redo logs record all changes made to the database, while undo logs provide the information necessary to rollback transactions. High Availability Solutions: Implementing technologies like Oracle Data Guard for creating standby databases and Oracle Real Application Clusters (RAC) for clustering and failover capabilities, ensuring continuous availability and rapid recovery in case of failures. To illustrate the utilization of a recovery mechanism, let's consider an example where accidental data deletion occurs in the Wards table


```

-- Accidental data deletion
DELETE FROM Wards WHERE Ward_ID = 3;

-- Rollback the transaction to recover deleted data
ROLLBACK;
```

By issuing a ROLLBACK command, Oracle will revert the effects of the accidental DELETE operation, restoring the deleted data in the Wards table to its previous state. These are just a few examples of how you can implement concurrency control and utilize recovery mechanisms in Oracle databases. Depending on your specific requirements and environment, you may need to explore additional features and best practices for ensuring data consistency, availability, and recoverability.

CHAPTER 6

Code of the Project

```
CREATE TABLE Hospitals (
    ID NUMBER PRIMARY KEY,
    Hospital_Name VARCHAR2(100),
    Address VARCHAR2(200),
    Contact_Number VARCHAR2(20)
);


-- Inserting 5 random inputs
INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
VALUES (1, 'City Hospital', '123 Main St, Cityville, CA', '123-456-7890');


INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
VALUES (2, 'Town Medical Center', '456 Elm St, Townsville, NY', '456-789-0123');


INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
VALUES (3, 'Village Health Clinic', '789 Oak Ave, Villagetown, TX', '789-012-3456');
```

```
INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
VALUES (4, 'Borough General Hospital', '321 Pine Rd, Boroughburg, FL', '987-654-3210');

INSERT INTO Hospitals (ID, Hospital_Name, Address, Contact_Number)
VALUES (5, 'Campus Health Services', '567 Cedar Ln, Campus City, IL', '234-567-8901');

CREATE TABLE Doctors (
    Doctor_ID NUMBER PRIMARY KEY,
    Hospital_Name VARCHAR2(100),
    Address VARCHAR2(200),
    Contact_Number VARCHAR2(20)
);

-- Inserting 5 random inputs
INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
VALUES (1, 'General Hospital', '123 Main St, Cityville, CA', '123-456-7890');

INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
VALUES (2, 'Community Medical Center', '456 Elm St, Townsville, NY', '456-789-0123');

INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
VALUES (3, 'University Hospital', '789 Oak Ave, Villagetown, TX', '789-012-3456');

INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
VALUES (4, 'City Medical Clinic', '321 Pine Rd, Boroughburg, FL', '987-654-3210');
```

```
INSERT INTO Doctors (Doctor_ID, Hospital_Name, Address, Contact_Number)
VALUES (5, 'Regional Health Center', '567 Cedar Ln, Campus City, IL', '234-567-8901');




CREATE TABLE Departments (
    Department_ID NUMBER PRIMARY KEY,
    Department_Name VARCHAR2(100),
    Head_Doctor_ID NUMBER
);

-- Inserting 5 random inputs
INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
VALUES (1, 'Cardiology', 101);

INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
VALUES (2, 'Neurology', 102);

INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
VALUES (3, 'Orthopedics', 103);

INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
VALUES (4, 'Pediatrics', 104);
```

```
INSERT INTO Departments (Department_ID, Department_Name, Head_Doctor_ID)
VALUES (5, 'Oncology', 105);






CREATE TABLE Wards (
    Ward_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Total_Beds NUMBER,
    Occupied_Beds NUMBER
);

-- Inserting 5 random inputs
INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
VALUES (1, 'Emergency Ward', 20, 10);

INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
VALUES (2, 'Intensive Care Unit', 10, 5);

INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
VALUES (3, 'Maternity Ward', 15, 8);

INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
VALUES (4, 'Surgical Ward', 25, 15);
```

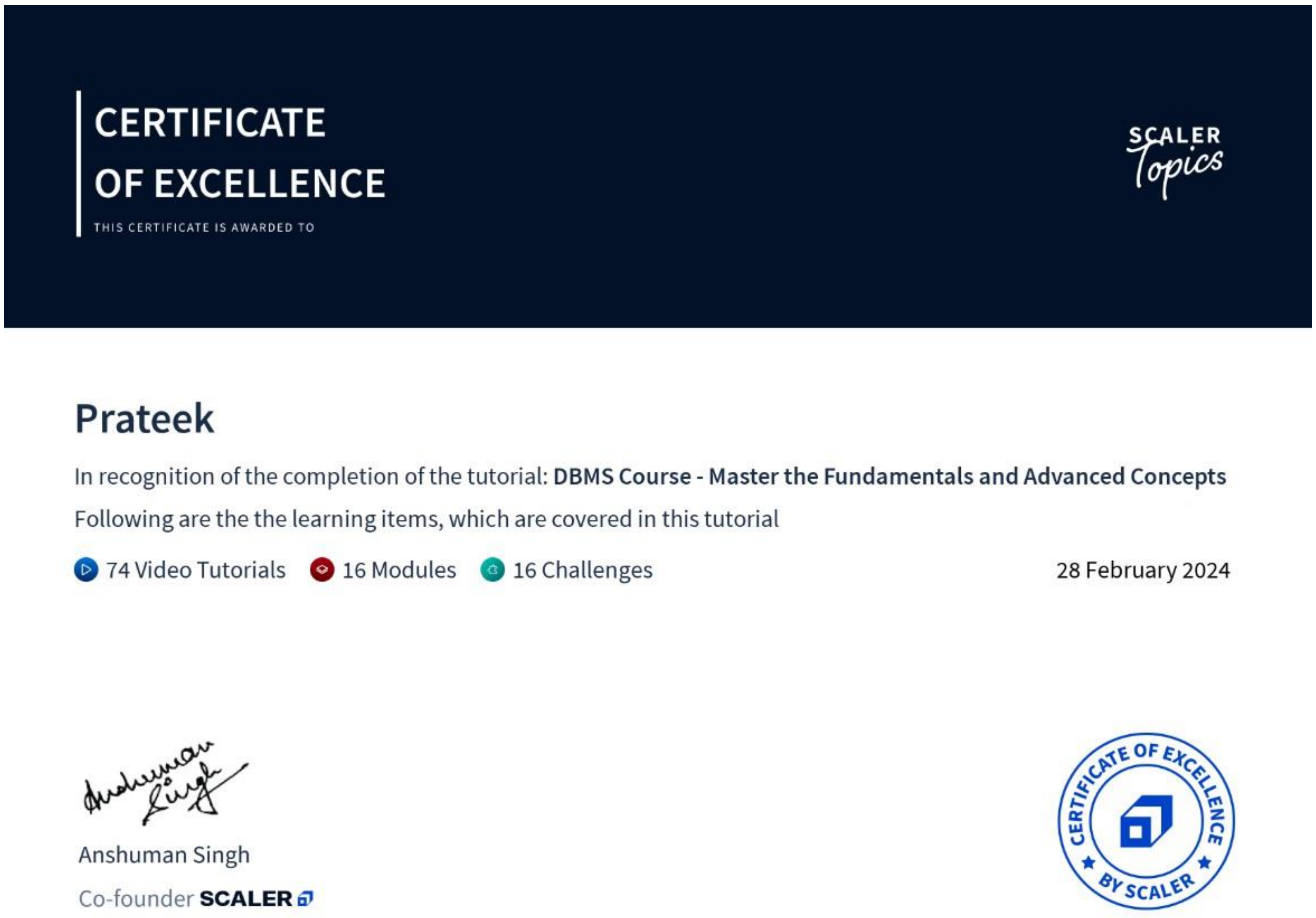
```
INSERT INTO Wards (Ward_ID, Name, Total_Beds, Occupied_Beds)
VALUES (5, 'Pediatric Ward', 30, 20);




CREATE TABLE Nurses (
    Nurse_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Department_ID NUMBER,
    Shift VARCHAR2(20)
);

-- Inserting 5 random inputs
INSERT INTO Nurses (Nurse_ID, Name, Department_ID, Shift)
VALUES (1, 'Alice Smith', 1, 'Day');

INSERT INTO Nurses (Nurse_ID, Name, Department_ID, Shift)
VALUES (2, 'John Doe', 2, 'Night');

INSERT INTO Nurses (Nurse_ID, Name, Department_ID, Shift)
VALUES (3, 'Emily Johnson', 1, 'Day');

INSERT INTO Nurses (Nurse_ID, Name, Department_ID, Shift)
VALUES (4, 'Michael Brown', 3, 'Night');
```

```
INSERT INTO Nurses (Nurse_ID, Name, Department_ID, Shift)

VALUES (5, 'Sarah Williams', 2, 'Day');




CREATE TABLE Patients (

    Patient_ID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Gender VARCHAR2(10),

    Address VARCHAR2(200)

);


-- Inserting 5 random inputs

INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)

VALUES (1, 'John Smith', TO_DATE('1980-05-15', 'YYYY-MM-DD'), 'Male', '123 Main St,
Cityville, CA');


INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)

VALUES (2, 'Emily Johnson', TO_DATE('1992-08-21', 'YYYY-MM-DD'), 'Female', '456 Elm St,
Townsville, NY');


INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)

VALUES (3, 'Michael Brown', TO_DATE('1975-03-10', 'YYYY-MM-DD'), 'Male', '789 Oak Ave,
Villagetown, TX');
```

```
INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)

VALUES (4, 'Sarah Williams', TO_DATE('1988-11-30', 'YYYY-MM-DD'), 'Female', '321 Pine Rd, Boroughburg, FL');


INSERT INTO Patients (Patient_ID, Name, DOB, Gender, Address)

VALUES (5, 'David Wilson', TO_DATE('2000-02-18', 'YYYY-MM-DD'), 'Male', '567 Cedar Ln, Campus City, IL');
```

CHAPTER 7

COURSE COMPLETION CERTIFICATE



CERTIFICATE OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Prateek

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges

28 February 2024

Anshuman Singh

Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER