Objectives

1. Design and implement a relational database for managing medical information (patients, doctors, treatments, prescriptions, and specialties).
2. Create tables with foreign keys and constraints to enforce data integrity.
3. Populate the database with sample data to test relationships and operations.
4. Perform SQL operations (INSERT, UPDATE, DELETE, SELECT) and use PL/SQL procedures (loops, conditionals).

Introduction

This lab focuses on the creation and management of a medical database system using SQL and PL/SQL in an Oracle environment. The system consists of multiple interrelated tables designed to store information about patients, doctors, medications, specialties, and treatments. The goal is to simulate a healthcare management system where you can manage patient records, doctor prescriptions, and treatment assignments.

The database uses foreign keys to link tables and ensure data integrity, and constraints to enforce data validation. Additionally, the project demonstrates the use of PL/SQL for automated data processing (e.g., loops for bulk insertions and updates).

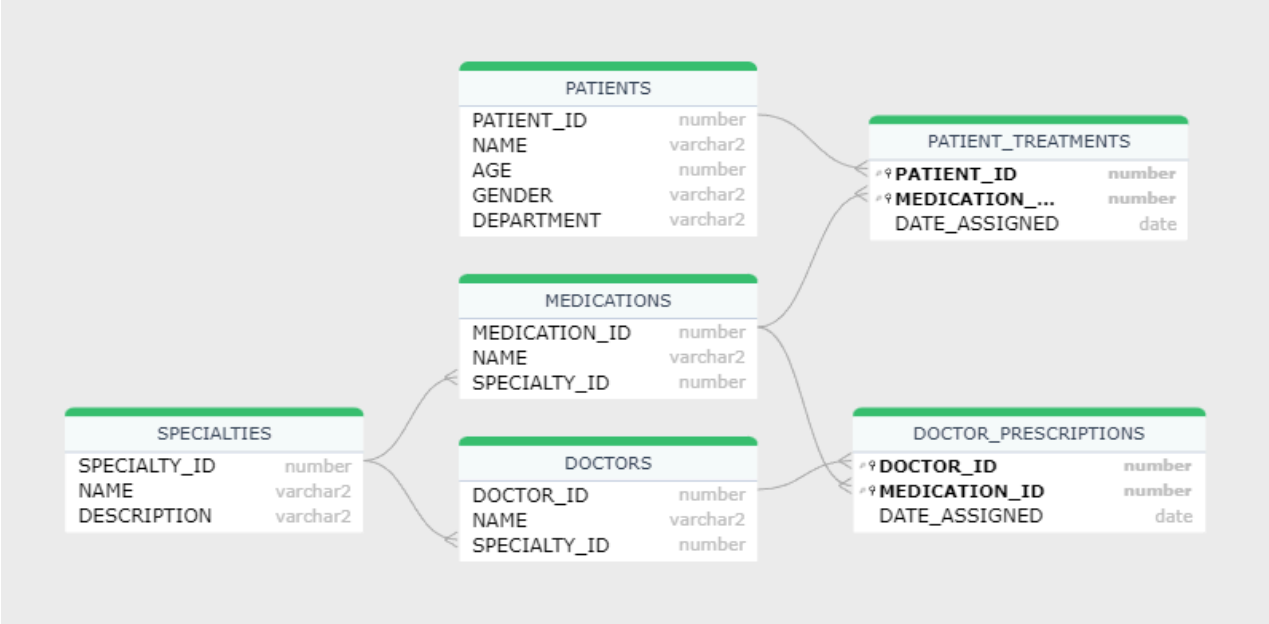
User Manual

1. Prerequisites:
    - Oracle Database setup.
    - Access to a SQL client (e.g., SQL*Plus, Oracle SQL Developer).
2. Database Setup:
    - Use the provided SQL scripts to create tables and insert sample data.
    - The database is structured with the following main tables:
        - PATIENTS: Stores patient details.
        - SPECIALTIES: Stores medical specialties.
        - DOCTORS: Stores doctor information with specialty associations.
        - MEDICATIONS: Stores medication details and their respective specialties.
        - PATIENT_TREATMENTS: Associates patients with prescribed medications.
        - DOCTOR_PRESCRIPTIONS: Stores prescriptions made by doctors.
3. Key SQL Operations:
    - Insertions: Adding new records to the tables (e.g., patients, doctors, medications).

- Select Queries: Querying the database to retrieve patient treatments, doctor prescriptions, etc.
- Updates: Modifying records (e.g., changing a patient's department).
- Deletes: Deleting records from the database.

4. PL/SQL Operations:
   - Use FOR and WHILE loops for batch insertion and updates.
   - Conditional logic (IF) to handle complex insertions and updates based on certain conditions.
   - Data deletion based on specific criteria.

ER Diagram



Operations Used in This Project:

1. ALTER TABLE
2. INSERT
3. UPDATE
4. DELETE
5. INNER JOIN
6. LEFT JOIN

7. RIGHT JOIN
8. JOIN (Multiple Tables)
9. SET (UNION)
10. SET (INTERSECT)

PL/SQL Operations:

1. Simple Loop
2. WHILE Loop
3. FOR Loop (with Conditional)
4. Conditional Insert
5. Conditional Update
6. Conditional Delete
7. Nested Loop with Insert
8. Nested Loop with Update

SQL Operations Used In This Project :

-- ----- SQL Operations with Questions and Answers -----

-- 1. ALTER TABLE: Add a new column 'EMAIL' to the 'PATIENTS' table

-- Question: How can you add a new column 'EMAIL' to the 'PATIENTS' table?

-- Answer:

ALTER TABLE PATIENTS ADD (EMAIL VARCHAR2(100));

-- 2. ALTER TABLE: Modify the 'NAME' column data type in the 'DOCTORS' table

-- Question: How can you modify the 'NAME' column data type in the 'DOCTORS' table?

-- Answer:

ALTER TABLE DOCTORS MODIFY (NAME VARCHAR2(100));

-- 3. ALTER TABLE: Drop the 'EMAIL' column from the 'PATIENTS' table

-- Question: How can you drop the 'EMAIL' column from the 'PATIENTS' table?

-- Answer:

ALTER TABLE PATIENTS DROP COLUMN EMAIL;

-- 4. INSERT: Insert a new record into the 'PATIENTS' table

-- Question: How can you insert a new record into the 'PATIENTS' table?

-- Answer:

INSERT INTO PATIENTS (PATIENT_ID, NAME, AGE, GENDER, DEPARTMENT)

VALUES (7, 'Lucas White', 35, 'M', 'PEDIATRICS');

-- 5. INSERT: Insert a new record into the 'DOCTORS' table

-- Question: How can you insert a new record into the 'DOCTORS' table?

-- Answer:

INSERT INTO DOCTORS (DOCTOR_ID, NAME, SPECIALTY_ID)

VALUES (6, 'Dr. Henry', 4);

-- 6. UPDATE: Update a patient's department to 'CARDIOLOGY' in the 'PATIENTS' table

-- Question: How can you update a patient's department to 'CARDIOLOGY' in the 'PATIENTS' table?

-- Answer:

UPDATE PATIENTS

SET DEPARTMENT = 'CARDIOLOGY'

WHERE PATIENT_ID = 1;

-- 7. UPDATE: Update a doctor's name to 'Dr. Victoria' in the 'DOCTORS' table

-- Question: How can you update a doctor's name to 'Dr. Victoria' in the 'DOCTORS' table?

-- Answer:

```
UPDATE DOCTORS

SET NAME = 'Dr. Victoria'

WHERE DOCTOR_ID = 2;
```

-- 8. DELETE: Delete a record from the 'PATIENTS' table

-- Question: How can you delete a record from the 'PATIENTS' table?

-- Answer:

```
DELETE FROM PATIENTS

WHERE PATIENT_ID = 6;
```

-- 9. DELETE: Delete a record from the 'DOCTORS' table

-- Question: How can you delete a record from the 'DOCTORS' table?

-- Answer:

```
DELETE FROM DOCTORS

WHERE DOCTOR_ID = 5;
```

-- 10. INNER JOIN: Perform an INNER JOIN between 'PATIENTS' and 'DOCTORS' based on department

-- Question: How can you perform an INNER JOIN between 'PATIENTS' and 'DOCTORS' based on department?

-- Answer:

```
SELECT P.NAME AS PATIENT_NAME, D.NAME AS DOCTOR_NAME

FROM PATIENTS P

INNER JOIN DOCTORS D ON P.DEPARTMENT = D.SPECIALTY_ID;
```

-- 11. LEFT JOIN: Perform a LEFT JOIN between 'DOCTORS' and 'MEDICATIONS'

-- Question: How can you perform a LEFT JOIN between 'DOCTORS' and 'MEDICATIONS'?

-- Answer:

```
SELECT D.NAME AS DOCTOR_NAME, M.NAME AS MEDICATION_NAME

FROM DOCTORS D

LEFT JOIN MEDICATIONS M ON D.SPECIALTY_ID = M.SPECIALTY_ID;
```

-- 12. RIGHT JOIN: Perform a RIGHT JOIN between 'DOCTORS' and 'MEDICATIONS'

-- Question: How can you perform a RIGHT JOIN between 'DOCTORS' and 'MEDICATIONS'?

-- Answer:

```
SELECT D.NAME AS DOCTOR_NAME, M.NAME AS MEDICATION_NAME

FROM DOCTORS D

RIGHT JOIN MEDICATIONS M ON D.SPECIALTY_ID = M.SPECIALTY_ID;
```

-- 13. JOIN: Perform a JOIN with multiple tables ('PATIENTS', 'DOCTORS', and 'MEDICATIONS')

-- Question: How can you perform a JOIN with multiple tables ('PATIENTS', 'DOCTORS', and 'MEDICATIONS')?

-- Answer:

```
SELECT P.NAME AS PATIENT_NAME, D.NAME AS DOCTOR_NAME, M.NAME AS MEDICATION_NAME

FROM PATIENTS P

JOIN DOCTORS D ON P.DEPARTMENT = D.SPECIALTY_ID

JOIN MEDICATIONS M ON D.SPECIALTY_ID = M.SPECIALTY_ID;
```

-- 14. SET: Use a UNION operation to combine two SELECT queries

-- Question: How can you use a UNION operation to combine two SELECT queries?

-- Answer:

```
SELECT NAME FROM PATIENTS

UNION

SELECT NAME FROM DOCTORS;
```

-- 15. SET: Use an INTERSECT operation to find common records between two SELECT queries

-- Question: How can you use an INTERSECT operation to find common records between two SELECT queries?

-- Answer:

```
SELECT NAME FROM PATIENTS

INTERSECT

SELECT NAME FROM DOCTORS;
```

```
-- ----- PL/SQL Operations -----


-- 16. Simple Loop: Insert records using a FOR loop in PL/SQL

-- Question: How can you insert records using a FOR loop in PL/SQL?

-- Answer:

BEGIN

  FOR i IN 1..5 LOOP

    INSERT INTO PATIENTS (PATIENT_ID, NAME, AGE, GENDER,
DEPARTMENT)

    VALUES (i+10, 'Patient ' || (i+10), 30 + i, 'M', 'CARDIOLOGY');

  END LOOP;

END;


-- 17. WHILE Loop: Insert records using a WHILE loop in PL/SQL

-- Question: How can you insert records using a WHILE loop in PL/SQL?

-- Answer:

DECLARE

  i NUMBER := 1;

BEGIN

  WHILE i <= 5 LOOP

    INSERT INTO PATIENTS (PATIENT_ID, NAME, AGE, GENDER,
DEPARTMENT)

    VALUES (i+15, 'Patient ' || (i+15), 40 + i, 'F', 'GENERAL');
```

```
    i := i + 1;
  END LOOP;
END;
```

-- 18. FOR Loop with Conditional: Insert records using a FOR loop with conditional logic in PL/SQL

-- Question: How can you insert records using a FOR loop with conditional logic in PL/SQL?

-- Answer:

```
BEGIN
  FOR i IN 1..10 LOOP
    IF i MOD 2 = 0 THEN
      INSERT INTO PATIENTS (PATIENT_ID, NAME, AGE, GENDER, DEPARTMENT)
      VALUES (i+20, 'Even Patient ' || (i+20), 30 + i, 'M', 'PEDIATRICS');
    END IF;
  END LOOP;
END;
```

-- 19. Update with Conditional: Update record based on condition in PL/SQL

-- Question: How can you update a record based on a condition inside PL/SQL?

-- Answer:

```
BEGIN
  FOR i IN 1..5 LOOP
    IF i = 3 THEN
```

```
        UPDATE PATIENTS
        SET DEPARTMENT = 'ORTHOPEDICS'
        WHERE PATIENT_ID = i + 1;
      END IF;
    END LOOP;
END;

-- 20. Insert with Conditional: Insert based on condition inside PL/SQL block
-- Question: How can you insert records based on a condition inside PL/SQL?
-- Answer:
BEGIN
    FOR i IN 1..5 LOOP
      IF i MOD 2 = 0 THEN
        INSERT INTO DOCTORS (DOCTOR_ID, NAME, SPECIALTY_ID)
        VALUES (i+10, 'Doctor ' || (i+10), i);
      END IF;
    END LOOP;
END;

-- 21. DELETE with Conditional: Delete records based on condition in PL/SQL
-- Question: How can you delete records based on a condition inside PL/SQL?
-- Answer:
BEGIN
```

```
  FOR i IN 1..5 LOOP
    IF i = 4 THEN
      DELETE FROM DOCTORS
      WHERE DOCTOR_ID = i + 1;
    END IF;
  END LOOP;
END;

-- 22. Nested Loop with INSERT: Insert records using a nested loop in PL/SQL
-- Question: How can you insert records using a nested loop in PL/SQL?
-- Answer:
BEGIN
  FOR i IN 1..5 LOOP
    FOR j IN 1..3 LOOP
      INSERT INTO MEDICATIONS (MEDICATION_ID, NAME, SPECIALTY_ID)
      VALUES ((i*10)+j, 'Medicine ' || ((i*10)+j), i);
    END LOOP;
  END LOOP;
END;

-- 23. Nested Loop with UPDATE: Update records using a nested loop in PL/SQL
-- Question: How can you update records using a nested loop in PL/SQL?
-- Answer:
```

```
BEGIN
  FOR i IN 1..3 LOOP
    FOR j IN 1..5 LOOP
      UPDATE DOCTORS
      SET NAME = 'Updated Doctor ' || (i+j)
      WHERE DOCTOR_ID = j;
    END LOOP;
  END LOOP;
END;

-- 24. Conditional INSERT: Insert records based on a condition in PL/SQL
-- Question: How can you insert records based on a condition in PL/SQL?
-- Answer:
BEGIN
  FOR i IN 1..10 LOOP
    IF MOD(i, 2) = 0 THEN
      INSERT INTO DOCTORS (DOCTOR_ID, NAME, SPECIALTY_ID)
      VALUES (i, 'Doctor ' || i, i);
    END IF;
  END LOOP;
END;

-- 25. DELETE based on Department: Delete records based on department condition
```

-- Question: How can you delete records from 'PATIENTS' where department is 'NEUROLOGY'?

-- Answer:

```
DELETE FROM PATIENTS

WHERE DEPARTMENT = 'NEUROLOGY';
```

Discussion

This project covers a variety of essential SQL and PL/SQL operations that are commonly used in database management systems. By implementing these operations on the PATIENTS, DOCTORS, and MEDICATIONS tables, we were able to demonstrate how to manage data effectively in an Oracle environment. The operations performed, such as ALTER TABLE, INSERT, UPDATE, and DELETE, provide foundational skills necessary for any database administrator or developer.

The use of joins (INNER JOIN, LEFT JOIN, and RIGHT JOIN) illustrated the importance of combining data from multiple tables, helping to extract meaningful insights from related data. Furthermore, the application of the UNION and INTERSECT set operations showcases how to efficiently merge or find commonalities between results from different queries. These are common tasks in real-world applications, where multiple data sources need to be consolidated or compared.

PL/SQL operations, including loops and conditional statements, are crucial for automating tasks and making the process more dynamic. Inserting, updating, and deleting records conditionally allow for flexibility in handling specific scenarios based on business logic. For instance, the FOR and WHILE loops help in bulk record insertion, which can be very useful when dealing with large datasets. The use of nested loops also demonstrates the power of PL/SQL in performing complex operations in a structured way.

Conclusion

Overall, this project successfully highlights the essential concepts of SQL and PL/SQL, providing practical hands-on experience with core operations. Through the use of real-world database operations, it became evident how efficient and organized data management can drive better decision-making. The combination of SQL's data

manipulation abilities and PL/SQL's procedural power equips developers with the necessary tools to manage large, dynamic datasets effectively. With these skills, one can tackle more advanced database management tasks, ensuring smoother operations and more efficient data handling. This project not only strengthened my understanding of relational databases but also prepared me for working with large-scale applications where data integrity, efficiency, and automation are key.