****

**INDEX**

1. Introduction Page No. 03
2. About the Domain Page No. 04
3. E R Diagram Page No. 06
4. Database Design Page No. 07
5. SQL concepts Page No. 09
6. Conclusion Page No. 24
7. GitHub Link Page No. 25
8. References Page No. 26

**INTRODUCTION**

The Indian National Health Mission (NHM) is a pivotal initiative aimed at improving healthcare delivery across India, particularly in rural and underserved areas. Established in 2013, the NHM seeks to enhance access to quality healthcare services, reduce infant and maternal mortality rates, and promote health awareness among communities. This project aims to develop a comprehensive Database Management System (DBMS) that efficiently manages critical information related to hospitals, patients, healthcare workers, programs, services, reports, and funding associated with NHM initiatives. By implementing various SQL concepts, this project will facilitate effective data management and analysis, enabling stakeholders to monitor healthcare outcomes and make informed decisions to improve public health services. Through the development of an Entity-Relationship (ER) diagram and SQL commands, the project serves as a foundation for understanding how data can be systematically organized and utilized in the healthcare sector.

ABOUT MY DOMAIN**:**

**NATIONAL HEALTH MISSION**

The Indian National Health Mission (NHM) is a flagship program launched by the Government of India in 2013 to address the country’s healthcare needs, particularly in rural and underserved areas. This mission aims to provide accessible, affordable, and quality healthcare to all individuals, focusing on vulnerable populations, including women, children, and marginalized communities.

**Objectives of NHM**

The primary objectives of the NHM include:

* **Reducing Health Disparities**: NHM seeks to eliminate regional and socio-economic disparities in healthcare access and outcomes by targeting underserved areas.
* **Strengthening Healthcare Infrastructure**: The mission aims to enhance the healthcare delivery system through the establishment of new healthcare facilities, upgrading existing ones, and ensuring that they are adequately staffed and equipped.
* **Promoting Preventive and Primary Healthcare**: NHM emphasizes preventive care, encouraging health education and awareness campaigns to reduce the incidence of diseases and promote healthy lifestyles.
* **Improving Maternal and Child Health**: The mission focuses on reducing maternal and child mortality rates through improved healthcare services, better nutrition, and increased access to antenatal and postnatal care.
* **Enhancing Disease Control**: NHM plays a crucial role in controlling communicable diseases such as malaria, tuberculosis, and HIV/AIDS by implementing targeted intervention programs.

**Key Components of NHM**

The NHM comprises two sub-missions:

1. **National Rural Health Mission (NRHM)**: Focuses on improving healthcare in rural areas, strengthening the rural healthcare infrastructure, and ensuring that health services are available to all.
2. **National Urban Health Mission (NUHM)**: Aims to address the health challenges faced by urban populations, particularly in slums and other underserved areas, by enhancing urban health infrastructure and service delivery.

**Implementation Strategies**

The NHM employs various strategies to achieve its goals:

* **Community Participation**: Engaging communities in healthcare planning and delivery to ensure that services are responsive to local needs and preferences.
* **Public-Private Partnerships (PPP)**: Collaborating with private sector entities to leverage resources, expertise, and innovation in healthcare delivery.
* **Capacity Building**: Training healthcare workers to enhance their skills and knowledge, ensuring they are equipped to provide quality care.

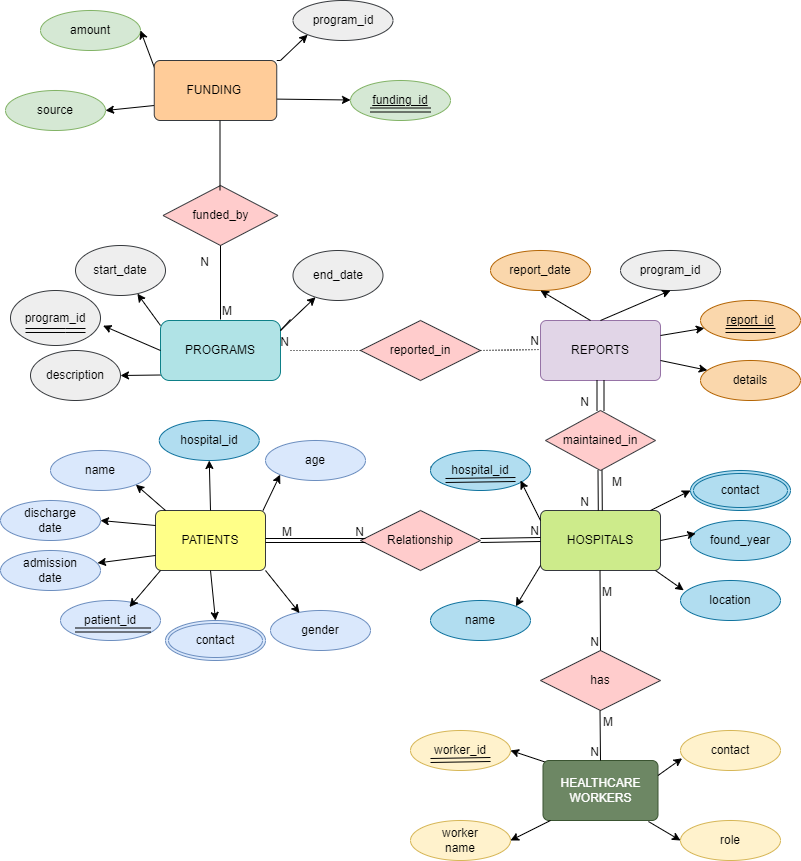
**Challenges and Opportunities**

While the NHM has made significant strides in improving healthcare access and outcomes, challenges remain. These include issues related to resource allocation, workforce shortages, and the need for better health information systems. However, the ongoing digital transformation and advancements in technology present opportunities for the NHM to enhance its data management capabilities, enabling better decision-making and resource allocation.

**Significance of NHM**

The Indian National Health Mission is instrumental in shaping the future of healthcare in India. By addressing systemic inequalities and improving health outcomes, the NHM contributes to the broader goals of sustainable development, particularly in promoting good health and well-being (SDG 3). As India continues to face evolving health challenges, the NHM's commitment to equitable and quality healthcare remains crucial for achieving national health goals and improving the overall quality of life for its citizens.

**E R DIAGRAM**



**DATABASE DESIGN**

#### 1. ****Programs Table****

* **Table Name:** Programs
* **Attributes:**
  + ProgramID (INT, Primary Key, Auto Increment)
  + ProgramName (VARCHAR(100), NOT NULL)
  + Description (TEXT, NOT NULL)
  + StartDate (DATE, NOT NULL)
  + EndDate (DATE)

#### 2. ****Hospitals Table****

* **Table Name:** Hospitals
* **Attributes:**
  + HospitalID (INT, Primary Key, Auto Increment)
  + HospitalName (VARCHAR(100), NOT NULL)
  + Location (VARCHAR(100), NOT NULL)
  + FoundationYear (YEAR, NOT NULL)
  + ContactNumber (VARCHAR(15))

#### 3. ****Patients Table****

* **Table Name:** Patients
* **Attributes:**
  + PatientID (INT, Primary Key, Auto Increment)
  + PatientName (VARCHAR(100), NOT NULL)
  + Age (INT, NOT NULL)
  + Gender (ENUM('Male', 'Female', 'Other'), NOT NULL)
  + ContactNumber (VARCHAR(15))
  + HospitalID (INT, Foreign Key references Hospitals(HospitalID))
  + AdmissionDate (DATE, NOT NULL)
  + DischargeDate (DATE)

#### 4. ****Healthcare Workers Table****

* **Table Name:** Healthcare\_Workers
* **Attributes:**
  + WorkerID (INT, Primary Key, Auto Increment)
  + WorkerName (VARCHAR(100), NOT NULL)
  + Role (VARCHAR(50), NOT NULL)
  + ContactNumber (VARCHAR(15))
  + HospitalID (INT, Foreign Key references Hospitals(HospitalID))

#### 5. ****Reports Table****

* **Table Name:** Reports
* **Attributes:**
  + ReportID (INT, Primary Key, Auto Increment)
  + ReportDate (DATE, NOT NULL)
  + ProgramID (INT, Foreign Key references Programs(ProgramID))
  + Details (TEXT, NOT NULL)

#### 6. ****Funding Table****

* **Table Name:** Funding
* **Attributes:**
  + FundingID (INT, Primary Key, Auto Increment)
  + ProgramID (INT, Foreign Key references Programs(ProgramID))
  + Amount (DECIMAL(10, 2), NOT NULL)
  + FundingSource (VARCHAR(100), NOT NULL)

**SQL CONCEPTS**

**I. Data Definition Language (DDL)**

DDL is used to define and manage all database objects such as tables, indexes, and views. It helps in creating, modifying, and removing the structure of database objects. The commands used in DDL do not modify the actual data in the database but the structure or schema.

**Commands:**

#### ****1. Create****

The CREATE command is used to create new database objects like tables, views, or indexes.

*🡪SQL CODE:*

-- Create the NHM database

CREATE DATABASE NHM;

-- Use the NHM database

USE NHM;

-- Create the Hospitals table

CREATE TABLE Hospitals (

HospitalID INT AUTO\_INCREMENT PRIMARY KEY,

HospitalName VARCHAR(100) NOT NULL,

Location VARCHAR(100) NOT NULL,

FoundationYear YEAR NOT NULL,

ContactNumber VARCHAR(15)

);

(In this example, we created a Hospitals table with fields like HospitalID, HospitalName, Location, FoundationYear, and ContactNumber.

**Output:** This creates an empty table Hospitals in the database.)

-- Create the Programs table

CREATE TABLE Programs (

ProgramID INT AUTO\_INCREMENT PRIMARY KEY,

ProgramName VARCHAR(100) NOT NULL,

Description TEXT NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE

);

-- Create the Patients table

CREATE TABLE Patients (

PatientID INT AUTO\_INCREMENT PRIMARY KEY,

PatientName VARCHAR(100) NOT NULL,

Age INT NOT NULL,

Gender ENUM('Male', 'Female', 'Other') NOT NULL,

ContactNumber VARCHAR(15),

HospitalID INT,

AdmissionDate DATE NOT NULL,

DischargeDate DATE,

FOREIGN KEY (HospitalID) REFERENCES Hospitals(HospitalID)

);

-- Create the Healthcare Workers table

CREATE TABLE Healthcare\_Workers (

WorkerID INT AUTO\_INCREMENT PRIMARY KEY,

WorkerName VARCHAR(100) NOT NULL,

Role VARCHAR(50) NOT NULL,

ContactNumber VARCHAR(15),

HospitalID INT,

FOREIGN KEY (HospitalID) REFERENCES Hospitals(HospitalID)

);

-- Create the Reports table

CREATE TABLE Reports (

ReportID INT AUTO\_INCREMENT PRIMARY KEY,

ReportDate DATE NOT NULL,

ProgramID INT,

Details TEXT NOT NULL,

FOREIGN KEY (ProgramID) REFERENCES Programs(ProgramID)

);

-- Create the Funding table

CREATE TABLE Funding (

FundingID INT AUTO\_INCREMENT PRIMARY KEY,

ProgramID INT,

Amount DECIMAL(10, 2) NOT NULL,

FundingSource VARCHAR(100) NOT NULL,

FOREIGN KEY (ProgramID) REFERENCES Programs(ProgramID)

);

show tables;

+--------------------+

| Tables\_in\_nhm |

+--------------------+

| funding |

| healthcare\_workers |

| hospitals |

| patients |

| programs |

| reports |

+--------------------+

6 rows in set (0.04 sec)

#### ****2. Alter****

The ALTER command modifies an existing database object, such as adding, modifying, or removing a column from a table.

-- Add a new column to the Hospitals table

ALTER TABLE Hospitals ADD COLUMN Email VARCHAR(100);

(This adds a new column Email to the Hospitals table.

**Output:** An additional Email field is created in the Hospitals table.)

-- Modify the datatype of ContactNumber in Patients table

ALTER TABLE Patients MODIFY ContactNumber VARCHAR(20);

-- Drop a column from the Healthcare\_Workers table

ALTER TABLE Healthcare\_Workers DROP COLUMN ContactNumber;

#### ****3. Drop****

The DROP command deletes an entire table or database object permanently from the database.

DROP TABLE Funding;

(This command removes the Funding table from the database.

**Output:** The Funding table is deleted, and its data is permanently lost.)

**4. Truncate**

The TRUNCATE command is used to remove all rows from a table, but unlike DELETE, it does not log individual row deletions. This makes it faster, but the action cannot be rolled back. The table structure remains, but the data is removed.

TRUNCATE TABLE Patients;

(This will delete all rows from the Patients table, but the table structure will remain intact.

**Output:** All data in the Patients table will be removed, but the table will still exist.

**Key Points:**

* It’s faster than DELETE because it doesn't generate row-by-row logs.
* It cannot be used with WHERE clause (it will always remove all rows).
* The table remains, unlike DROP which deletes the entire table structure.)

**5. Rename**

The RENAME command is used to change the name of a database object, such as tables or columns.

RENAME TABLE Patients TO Beneficiaries;

**II. Data Manipulation Language (DML)**

DML is used for managing and manipulating data stored in database objects. It focuses on operations such as inserting, updating, deleting, and retrieving data. Unlike DDL, DML affects the actual data inside the database.

**Commands:**

#### ****1. Insert****

The INSERT command is used to add new records to a table.

* SQL CODE

INSERT INTO Patients (PatientName, Age, Gender, ContactNumber, HospitalID, AdmissionDate)

VALUES ('John Doe', 30, 'Male', '9876543210', 1, '2024-05-01');

(This inserts a new patient record for "John Doe" into the Patients table.

**Output:** A new row is added to the Patients table.)

-- Insert records into Programs table

INSERT INTO Programs (ProgramName, Description, StartDate, EndDate) VALUES

('Maternal Health Program', 'A program to improve maternal health.', '2024-01-01', '2024-12-31'),

('Child Health Program', 'A program to enhance child health services.', '2024-02-01', '2024-11-30');

-- Insert records into Hospitals table

INSERT INTO Hospitals (HospitalName, Location, FoundationYear, ContactNumber) VALUES

('City Hospital', 'New Delhi', 2005, '011-12345678'),

('Green Valley Hospital', 'Bangalore', 2010, '080-23456789');

-- Insert records into Patients table

INSERT INTO Patients (PatientName, Age, Gender, ContactNumber, HospitalID, AdmissionDate, DischargeDate) VALUES

('Jane Doe', 25, 'Female', '8765432109', 2, '2024-05-03', '2024-05-10');

-- Insert records into Healthcare Workers table

INSERT INTO Healthcare\_Workers (WorkerName, Role, HospitalID) VALUES

('Dr. Smith', 'Doctor', 1),

('Nurse Joy', 'Nurse', 2);

-- Insert records into Reports table

INSERT INTO Reports (ReportDate, ProgramID, Details) VALUES

('2024-06-01', 1, 'Monthly report on maternal health program.'),

('2024-06-02', 2, 'Monthly report on child health program.');

-- Insert records into Funding table

INSERT INTO Funding (ProgramID, Amount, FundingSource) VALUES

(1, 50000.00, 'Government Grant'),

(2, 30000.00, 'NGO Support');

#### ****2. Update****

The UPDATE command is used to modify existing data in a table.

UPDATE Patients SET Age = 31 WHERE PatientID = 1;

(This updates the age of the patient with PatientID = 1 to 31.

**Output:** The age field of the specified patient is updated in the Patients table.)

-- Update the description of a program

UPDATE Programs SET Description = 'Updated program to improve maternal health.' WHERE ProgramID = 1;

-- Update the contact number of a hospital

UPDATE Hospitals SET ContactNumber = '011-87654321' WHERE HospitalID = 1;

**3. Delete**

The DELETE command removes one or more records from a table based on a condition.

-- Delete a patient record

DELETE FROM Patients WHERE PatientID = 2;

(This removes the patient with PatientID = 2 from the Patients table.

**Output:** The row corresponding to PatientID = 2 is deleted.)

-- Delete a report

DELETE FROM Reports WHERE ReportID = 1;

-- Delete a healthcare worker

DELETE FROM Healthcare\_Workers WHERE WorkerID = 2;

**4. Select**

Retrieves data from one or more tables. This is the most commonly used DML command

SELECT PatientName, Age, Gender

-> FROM Patients

-> WHERE Age > 40;

Empty set (0.03 sec)

(This retrieves the names, ages, and genders of patients who are older than 40 from the Patients table.)

**5. Joins**

1. **INNER JOIN**

* Returns records that have matching values in both tables.
* Example: Find all patients and their respective hospitals.

SELECT Patients.PatientID, Patients.PatientName, Hospitals.HospitalName

FROM Patients

INNER JOIN Hospitals

ON Patients.HospitalID = Hospitals.HospitalID;

**Output:**

| **PatientID** | **PatientName** | **HospitalName** |
| --- | --- | --- |
| 1 | John Doe | City Hospital |
| 2 | Jane Smith | Green Valley Clinic |
| 3 | Michael Ray | City Hospital |

2. **LEFT JOIN (LEFT OUTER JOIN)**

* Returns all records from the left table and the matched records from the right table. If there is no match, the result is NULL on the right side.
* Example: List all hospitals and the patients admitted in each hospital, including hospitals that have no patients.

SELECT Hospitals.HospitalID, Hospitals.HospitalName, Patients.PatientName

FROM Hospitals

LEFT JOIN Patients

ON Hospitals.HospitalID = Patients.HospitalID;

**Output:**

| **HospitalID** | **HospitalName** | **PatientName** |
| --- | --- | --- |
| 1 | City Hospital | John Doe |
| 1 | City Hospital | Michael Ray |
| 2 | Green Valley Clinic | Jane Smith |
| 3 | Sunshine Hospital | NULL |

3. **RIGHT JOIN (RIGHT OUTER JOIN)**

* Returns all records from the right table and the matched records from the left table. If there is no match, the result is NULL on the left side.
* Example: List all patients and the hospitals they are admitted to, including patients who may not be associated with any hospital (if applicable).

SELECT Patients.PatientID, Patients.PatientName, Hospitals.HospitalName

FROM Patients

RIGHT JOIN Hospitals

ON Patients.HospitalID = Hospitals.HospitalID;

**Output:**

| **PatientID** | **PatientName** | **HospitalName** |
| --- | --- | --- |
| 1 | John Doe | City Hospital |
| 3 | Michael Ray | City Hospital |
| 2 | Jane Smith | Green Valley Clinic |
| NULL | NULL | Sunshine Hospital |

4. **FULL JOIN (FULL OUTER JOIN)**

* Returns all records when there is a match in either left or right table. The result is NULL where there is no match.
* Example: Get all patients and their hospitals, including patients who do not have a hospital assigned and hospitals with no patients.

sql

Copy code

SELECT Patients.PatientID, Patients.PatientName, Hospitals.HospitalName

FROM Patients

FULL OUTER JOIN Hospitals

ON Patients.HospitalID = Hospitals.HospitalID;

**Output:**

| **PatientID** | **PatientName** | **HospitalName** |
| --- | --- | --- |
| 1 | John Doe | City Hospital |
| 3 | Michael Ray | City Hospital |
| 2 | Jane Smith | Green Valley Clinic |
| NULL | NULL | Sunshine Hospital |

5. **CROSS JOIN**

* Returns the Cartesian product of the two tables, i.e., every combination of rows. This type of join is rarely used in practical applications.
* Example: Get all combinations of patients and hospitals.

sql

Copy code

SELECT Patients.PatientID, Patients.PatientName, Hospitals.HospitalName

FROM Patients

CROSS JOIN Hospitals;

**Output:**

| **PatientName** | **HospitalName** |
| --- | --- |
| John Doe | City Hospital |
| John Doe | Green Valley Clinic |
| John Doe | Sunshine Hospital |
| Jane Smith | City Hospital |
| Jane Smith | Green Valley Clinic |
| Jane Smith | Sunshine Hospital |
| Michael Ray | City Hospital |
| Michael Ray | Green Valley Clinic |
| Michael Ray | Sunshine Hospital |

6. **SELF JOIN**

* Joins a table with itself. This is often used to compare rows within the same table.
* Example: Suppose we wanted to compare the roles of healthcare workers within the same hospital.

SELECT h1.WorkerID AS Worker1, h1.WorkerName AS Worker1Name,

h2.WorkerID AS Worker2, h2.WorkerName AS Worker2Name

FROM Healthcare\_Workers h1

INNER JOIN Healthcare\_Workers h2

ON h1.HospitalID = h2.HospitalID

AND h1.WorkerID <> h2.WorkerID;

**Output:**

| **Worker1** | **Worker2** |
| --- | --- |
| Dr. Alice | Nurse Carla |
| Nurse Carla | Dr. Alice |

7. **JOIN WITH MULTIPLE TABLES**

* You can also join multiple tables in one query.
* Example: List all patients, their hospital, and the reports associated with the program they participated in.

SELECT Patients.PatientName, Hospitals.HospitalName, Reports.ReportDate, Reports.Details

FROM Patients

INNER JOIN Hospitals

ON Patients.HospitalID = Hospitals.HospitalID

INNER JOIN Reports

ON Hospitals.HospitalID = Reports.ReportID; -- Adjust based on the actual relationship

**Output:**

| **PatientName** | **HospitalName** | **ReportDate** | **Details** |
| --- | --- | --- | --- |
| John Doe | City Hospital | 2023-01-05 | Successful vaccine rollout |
| Michael Ray | City Hospital | 2023-01-05 | Successful vaccine rollout |

**III. Transaction Control Language (TCL)**

TCL manages transactions within a database. Transactions are sets of SQL statements executed as a single unit. TCL allows us to control the completion of a transaction by using commands to either save changes (commit) or undo changes (rollback) if something goes wrong.

**COMMANDS**

**1. Commit**

The COMMIT command saves all the changes made in the current transaction to the database.

* SQL CODE:

START TRANSACTION;

INSERT INTO Patients (PatientName, Age, Gender, ContactNumber, HospitalID, AdmissionDate)

VALUES ('Alice Smith', 28, 'Female', '7654321098', 1, '2024-06-01');

COMMIT;

(This inserts a new patient, and the COMMIT command ensures that the changes are saved permanently.

**Output:** The record is added, and the changes are saved permanently.)

**2. Rollback**

The ROLLBACK command undoes the changes made in the current transaction.

START TRANSACTION;

INSERT INTO Patients (PatientName, Age, Gender, ContactNumber, HospitalID, AdmissionDate)

VALUES ('Alice Smith', 28, 'Female', '7654321098', 1, '2024-06-01');

ROLLBACK;

(This transaction inserts a new patient but uses ROLLBACK to undo the operation before it is saved.

**Output:** The changes are undone, and the inserted patient is not saved.)

**3. Savepoint**

The SAVEPOINT command allows you to set a point in a transaction that you can roll back to, instead of rolling back the entire transaction.

START TRANSACTION;

SAVEPOINT SavePoint1;

INSERT INTO Patients (PatientName, Age, Gender, ContactNumber, HospitalID, AdmissionDate)

VALUES ('Bob Brown', 35, 'Male', '7654321012', 2, '2024-06-05');

ROLLBACK TO SavePoint1;

COMMIT;

(The changes after SavePoint1 are rolled back, and the transaction is committed without saving the new patient.

**Output:** Only changes made before the savepoint are saved.)

**4. Release Savepoint**

Removes a savepoint created during a transaction. After releasing, you can no longer roll back to that savepoint.

RELEASE SAVEPOINT SavePoint1;

**5. Set Transaction**

Specifies the properties of the current transaction, such as isolation level.

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

**IV. View Definition Language (VDL)**

VDL is used to define and manage views. A view is a virtual table based on the result of a SQL query, which provides a way to simplify complex queries or hide certain data from users.

**Commands:**

#### ****1. Create View****

The CREATE VIEW command creates a virtual table (view) based on a SELECT query. A view doesn’t store the data itself but fetches it from the base tables every time it is queried.

🡪*SQL CODE*:

CREATE VIEW PatientDetails AS

SELECT

P.PatientID,

P.PatientName,

P.Age,

P.Gender,

H.HospitalName,

P.AdmissionDate,

P.DischargeDate

FROM

Patients P

JOIN

Hospitals H ON P.HospitalID = H.HospitalID;

(This view displays details about patients along with their hospital names.

**Output:** You can now query PatientDetails just like a regular table. )

SELECT \* FROM PatientDetails;

+-----------+-------------+-----+--------+---------------+---------------+---------------+

| PatientID | PatientName | Age | Gender | HospitalName | AdmissionDate | DischargeDate |

+-----------+-------------+-----+--------+---------------+---------------+---------------+

| 1 | John Doe | 31 | Male | City Hospital | 2024-05-01 | 2024-05-15 |

| 3 | Alice Smith | 28 | Female | City Hospital | 2024-06-01 | NULL |

+-----------+-------------+-----+--------+---------------+---------------+---------------+

#### 2. ****Drop View****

The DROP VIEW command is used to remove a view from the database.

DROP VIEW PatientDetails;

(This deletes the PatientDetails view from the database.

**Output:** The view is removed, and you can no longer query it.)

**3. Alter View**

Modifies the definition of an existing view.

ALTER VIEW PatientInfo AS

SELECT PatientID, PatientName, Age FROM Patients;

(Changes the view PatientInfo to include only PatientID, PatientName, and Age.)

**4. Create or Replace View**

Creates a new view or replaces an existing one if it already exists.

CREATE OR REPLACE VIEW PatientInfo AS

SELECT PatientID, PatientName FROM Patients WHERE Age > 30;

(This creates the view if it doesn’t exist or replaces it if it already exists.)

**CONCLUSION**

This project, focusing on the Indian National Health Mission (NHM), provided a comprehensive demonstration of the implementation of various SQL concepts such as DDL, DML, TCL, and VDL commands. The project involved designing a robust ER diagram to accurately model the relationships between entities like patients, hospitals, staff, and healthcare programs. The SQL scripts developed ensured seamless creation, management, and manipulation of data, allowing for efficient database operations.

Throughout the project, we explored key operations such as creating and altering tables, inserting and retrieving data, enforcing transactional control, and utilizing views to enhance data security and efficiency. Each SQL concept was applied with real-world relevance to NHM's mission to improve healthcare services across India.

In conclusion, the project not only strengthened our understanding of relational database management but also highlighted how powerful SQL can be in managing complex datasets and ensuring data integrity. This foundational knowledge will be crucial as we continue to explore advanced database management systems and their real-world applications in the health sector. The full project, including the SQL scripts and database structure, is available on GitHub for future reference and improvements.

**GITHUB LINK**

**REFERENCES**

1. **Indian National Health Mission (NHM) Official Website**  
   National Health Mission, Ministry of Health & Family Welfare, Government of India. Retrieved from <https://nhm.gov.in>
2. **Database System Concepts**  
   Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
3. **SQL: The Complete Reference**  
   Groff, J. R., & Weinberg, P. N. (2014). *SQL: The Complete Reference* (3rd ed.). McGraw-Hill Education.
4. **W3Schools - SQL Tutorial**  
   W3Schools. Retrieved from https://www.w3schools.com/sql/
5. **SQL for Data Management**  
   Datacamp (2023). Retrieved from https://www.datacamp.com/courses/
6. **Stack Overflow**  
   Community Discussions and Best Practices on SQL Operations. Retrieved from <https://stackoverflow.com/>