WORLD HEALTHCARE ORGANSIATION

**Introduction:**

**Database System: World Health Organization (WHO)**

The **World Health Organization (WHO)** is a global health agency that plays a crucial role in promoting international public health, monitoring diseases, managing healthcare resources, and coordinating health programs across different countries. To handle such a wide scope of operations, it requires a well-structured and efficient **database management system**.

This project is designed to create a **comprehensive WHO database** that can store, manage, and analyze large amounts of health-related data. The database contains **25 interconnected tables**, each covering key areas such as:

* **Member Countries** – Information about nations associated with WHO.
* **Hospitals & Clinics** – Healthcare facilities, their services, and capacities.
* **Doctors & Staff** – Medical professionals and their specialties.
* **Diseases & Research** – Global disease data, ongoing studies, and medical research projects.
* **Patients & Cases** – Individual patient records, treatments, and recovery details.
* **Vaccination & Campaigns** – Immunization programs and worldwide health initiatives.
* **Funding & Resources** – Financial support, budgets, and resources allocated for health missions.

By combining these tables with **primary keys, foreign keys, constraints, and validations**, the system ensures **data accuracy, consistency, and reliability**.

**Real-World Application**

This database can be used by the WHO to:

* Track and analyze the **spread of diseases** across countries.
* Manage and allocate **healthcare resources** effectively.
* Maintain detailed **records of doctors, hospitals, and patients** globally.
* Monitor the success of **vaccination drives and health campaigns**.
* Support **research and policy-making** for global health improvement.

**Purpose of the Project**

The goal of this project is to design a **robust WHO database system** that demonstrates:

* **Efficient database design** with 25 tables and constraints.
* **Logical data connections** using primary and foreign keys.
* **Realistic sample records** for practical use cases.
* **Execution of SQL queries** (DDL, DML, DQL) to showcase data handling and reporting.

DATABASE DESIGN :

1.PATIENT TABLE:

CREATE TABLE Patients (

patient\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

age INT,

gender VARCHAR(10),

contact\_number VARCHAR(15),

city VARCHAR(50)

);

2,DOCTORS TABLE:

CREATE TABLE Doctors (

doctor\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

specialization VARCHAR(50),

contact\_number VARCHAR(15),

department VARCHAR(50)

);

3.APPOINTMENT TABLE;

CREATE TABLE Appointments (

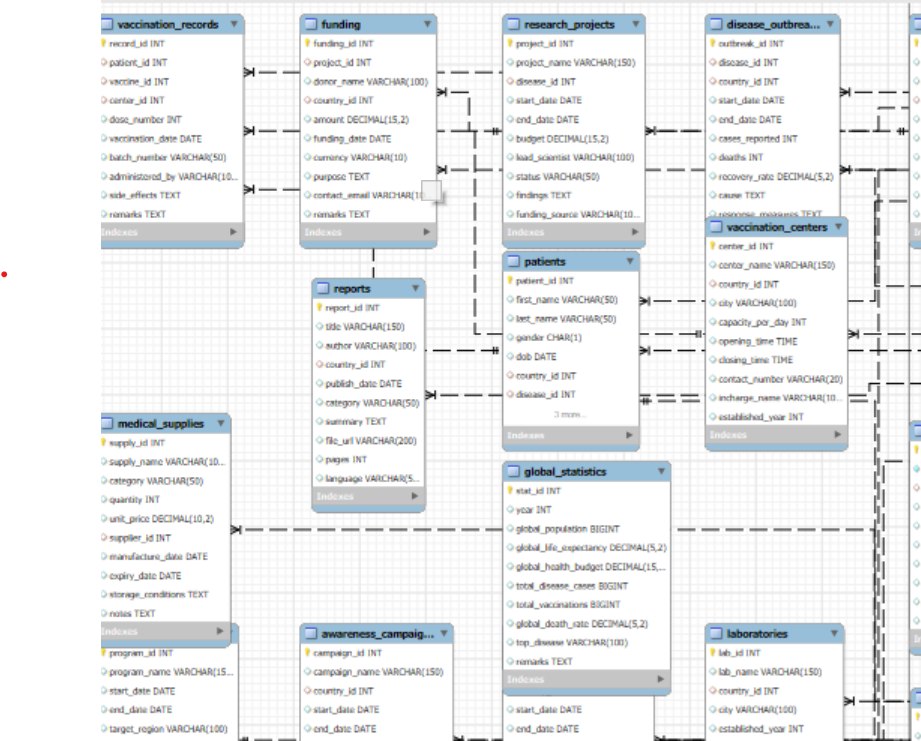
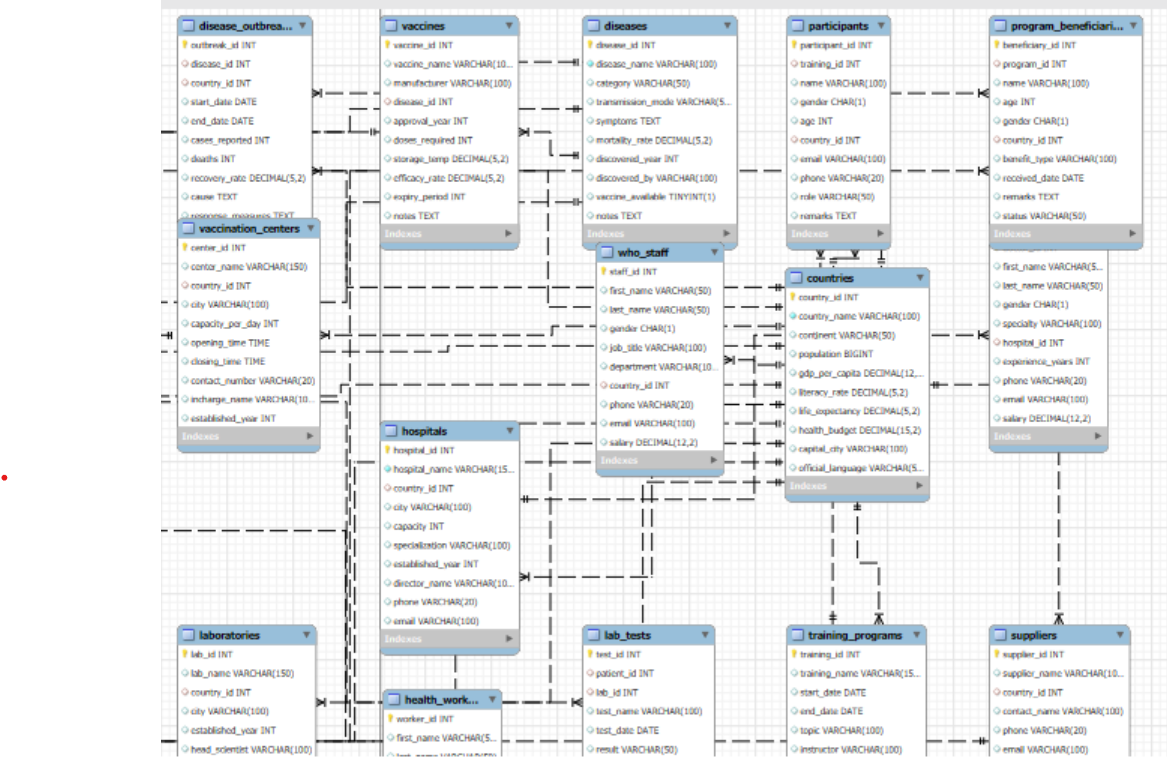
appointment\_id INT PRIMARY KEY,

patient\_id INT,

doctor\_id INT,

appointment\_date DATE);

ER-DIAGRAM:



RELATIONSHIP OF WHO

EXPLAIN ATION:

* One **country** can have many **hospitals, patients, vaccination centers, laboratories, campaigns, health programs, and fundings**.
* One **hospital** can have many **doctors, patients, appointments, and surgeries**.
* One **doctor** can treat many **patients**, write many **prescriptions**, and perform many **surgeries**.
* One **patient** can have many **appointments, lab tests, vaccination records, prescriptions, and surgeries**.
* One **disease** can affect many **patients**, have many **vaccines**, and many **research projects**.
* One **research project** can have many **fundings**.

**Queries in SQL:**

A **query** means asking the database to do something. It may be creating, inserting, updating, deleting, or retrieving data.  
Queries are divided into different types:

## DDL (Data Definition Language):

Used to **define and change the structure** of tables and databases.

Example:

* **CREATE** → Create new tables or databases.
* **ALTER** → Modify structure of a table (add/drop/rename column).
* **DROP** → Delete a table or database.
* **TRUNCATE** → Remove all records from a table quickly.

Example (Healthcare Project):

1.create:

CREATE TABLE countries (

country\_id INT PRIMARY KEY,

country\_name VARCHAR(100) NOT NULL,

);

2.alter:

Alter table countries add column age int not null;

3.drop

Drop table countries;

4.truncate:

Truncate table countries;

## DML (Data Manipulation Language):

Used to **work with data inside tables** (insert, update, delete records).

Examples:

* **INSERT** → Add new data.
* **UPDATE** → Change existing data.
* **DELETE** → Remove data.

Example:

1.INSERT :

Insert into patients (patient\_id, name, age, gender, city)

VALUES (1, 'Ravi Kumar', 35, 'Male', 'Hyderabad');

2.Update:

UPDATE patients

SET city = 'Delhi'

WHERE patient\_id = 1;

3Delete:

Delete from patients

where patient\_id = 1;

## DQL (Data Query Language):

Used to **retrieve data** (only SELECT is here).

Examples:

 **SELECT** → Fetch data from tables.

 With **WHERE**, **ORDER BY**, **GROUP BY**, **HAVING**, **LIMIT**.

Select:

SELECT \* FROM patients;

SELECT name, age

FROM patients

WHERE city = 'Hyderabad';

SELECT city, COUNT(\*) AS total\_patients FROM patients GROUP BY city;

**Analysis:**

The Healthcare System database is very useful in real life. It helps hospitals and clinics keep all information in one place. Patient records, doctor schedules, lab tests, medicines, insurance, and awareness campaigns can be managed easily.

For example:

* A hospital can quickly check a patient’s history and medicines.
* Doctors’ appointments can be tracked without confusion.
* Lab results can be connected with patients and doctors.
* Reports like “number of patients in each country” or “budget of health campaigns” can be generated in seconds.

This makes the healthcare system more organized, faster, and reliable.

**Reflection**

While making this project, I faced some problems but I solved them step by step.

* **Challenge 1**: Sometimes my INSERT queries gave errors because the column order was not correct.  
  **Solution**: I solved it by always writing column names before inserting values.
* **Challenge 2**: It was difficult to design relationships between many tables.  
  **Solution**: I made small tables first, then slowly connected them with primary keys and foreign keys.
* **Challenge 3**: The ER diagram was confusing at first.  
  **Solution**: I used simple shapes and step-by-step connections to show how patients, doctors, hospitals, and other tables are related.

Because of these steps, I was able to complete my database project successfully.