**Project Overview**

This project analyzes healthcare data from CSV datasets (Patients, Doctors, Appointments, Diagnoses, and Medications) using advanced SQL queries to extract insights for better patient care, doctor performance evaluation, appointment analysis, medication patterns, and revenue tracking.

**Step 1: Create Database**

CREATE DATABASE healthcaredb;

USE healthcaredb;

**Step 2: Import CSV Files into MySQL Workbench**

**Datasets:**

* patients.csv
* doctors.csv
* appointments.csv
* diagnoses.csv
* medications.csv

**Import Method:**

1. In MySQL Workbench, go to:
   * Server → Data Import.
   * Choose "Import from Self-Contained File" and select your .csv.
2. Use the table import wizard:
   * Go to Table Data Import Wizard from the schema.
   * Browse for the CSV file.
   * Map column names from CSV headers.

Repeat for each CSV dataset.

**Step 3: Define Schemas & Add Keys**

-- Define Primary Keys

ALTER TABLE Patients ADD PRIMARY KEY (PatientID);

ALTER TABLE Doctors ADD PRIMARY KEY (DoctorID);

ALTER TABLE Appointments ADD PRIMARY KEY (AppointmentID);

ALTER TABLE Diagnoses ADD PRIMARY KEY (DiagnosisID);

ALTER TABLE Medications ADD PRIMARY KEY (MedicationID);

-- Define Foreign Keys

ALTER TABLE Appointments

ADD FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),

ADD FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID);

ALTER TABLE Diagnoses

ADD FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),

ADD FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID);

ALTER TABLE Medications

ADD FOREIGN KEY (DiagnosisID) REFERENCES Diagnoses(DiagnosisID);

**Step 4: ER Diagram (Entity-Relationship Model)**

**How to generate in MySQL Workbench:**

1. Go to Database → Reverse Engineer.
2. Select your database (healthcaredb).
3. Complete the wizard → ERD is auto-generated.
4. Export as image → include it in your PPT.

**Entities and Relationships:**

* **Patients** ↔ **Appointments** ↔ **Doctors**
* **Appointments** ↔ **Diagnoses** ↔ **Medications**

**Step 5: Advanced SQL Queries**

**1. Patient Demographics with Appointments**

SELECT

p.PatientID, p.Name, p.Gender, p.Age,

COUNT(a.AppointmentID) AS TotalAppointments

FROM Patients p

LEFT JOIN Appointments a ON p.PatientID = a.PatientID

GROUP BY p.PatientID, p.Name, p.Gender, p.Age

ORDER BY TotalAppointments DESC;

**2. Doctor Diagnoses and Treatments**

SELECT

d.DoctorID, d.Name, d.Specialization,

COUNT(DISTINCT diag.DiagnosisID) AS TotalDiagnoses,

COUNT(DISTINCT diag.Treatment) AS UniqueTreatments

FROM Doctors d

LEFT JOIN Diagnoses diag ON d.DoctorID = diag.DoctorID

GROUP BY d.DoctorID, d.Name, d.Specialization

ORDER BY TotalDiagnoses DESC;

**3. Appointment Status Trends**

SELECT Status, COUNT(\*) AS Count

FROM Appointments

GROUP BY Status;

**4. Top 5 Prescribed Medications**

SELECT MedicationName, COUNT(\*) AS PrescriptionCount

FROM Medications

GROUP BY MedicationName

ORDER BY PrescriptionCount DESC

LIMIT 5;

**5. Revenue by Completed Appointments**

SELECT

d.Name AS DoctorName, d.Specialization,

COUNT(DISTINCT a.AppointmentID) AS CompletedAppointments,

SUM(t.Cost) AS TotalRevenue

FROM appointments a

JOIN treatments t ON a.AppointmentID = t.AppointmentID

JOIN doctors d ON a.DoctorID = d.DoctorID

WHERE a.Status = 'Completed'

GROUP BY d.DoctorID, d.Name, d.Specialization;

**Joins & Analytics**

**Inner Join – Completed Appointments with Doctor & Patient**

SELECT

a.AppointmentID, p.Name AS PatientName,

d.Name AS DoctorName, d.Specialization, a.AppointmentDate

FROM Appointments a

INNER JOIN Patients p ON a.PatientID = p.PatientID

INNER JOIN Doctors d ON a.DoctorID = d.DoctorID

WHERE a.Status = 'Completed';

**Left Join – Patients without Appointments**

SELECT p.PatientID, p.Name, p.ContactNumber, p.Address

FROM Patients p

LEFT JOIN Appointments a ON p.PatientID = a.PatientID

WHERE a.AppointmentID IS NULL;

**Right Join – Diagnoses per Doctor**

SELECT

d.DoctorID, d.Name, d.Specialization,

COUNT(DISTINCT diag.DiagnosisID) AS TotalDiagnoses

FROM Diagnoses diag

RIGHT JOIN Doctors d ON diag.DoctorID = d.DoctorID

GROUP BY d.DoctorID, d.Name, d.Specialization;

**Full Join – Mismatched Appointments vs Diagnoses**

-- Appointments without Diagnoses

SELECT

a.AppointmentID, p.Name AS PatientName,

d.Name AS DoctorName, NULL AS Diagnosis

FROM appointments a

LEFT JOIN diagnoses diag ON a.AppointmentID = diag.AppointmentID

LEFT JOIN patients p ON a.PatientID = p.PatientID

LEFT JOIN doctors d ON a.DoctorID = d.DoctorID

WHERE diag.AppointmentID IS NULL

UNION

-- Diagnoses without Appointments

SELECT

diag.AppointmentID, p.Name AS PatientName,

d.Name AS DoctorName, diag.Diagnosis

FROM diagnoses diag

LEFT JOIN appointments a ON diag.AppointmentID = a.AppointmentID

LEFT JOIN patients p ON diag.PatientID = p.PatientID

LEFT JOIN doctors d ON diag.DoctorID = d.DoctorID

WHERE a.AppointmentID IS NULL;

**Rank Patients per Doctor – Window Function**

SELECT

a.DoctorID, d.Name AS DoctorName,

a.PatientID, p.Name AS PatientName,

COUNT(\*) AS TotalAppointments,

RANK() OVER (PARTITION BY a.DoctorID ORDER BY COUNT(\*) DESC) AS RankPerDoctor

FROM Appointments a

JOIN Doctors d ON a.DoctorID = d.DoctorID

JOIN Patients p ON a.PatientID = p.PatientID

GROUP BY a.DoctorID, d.Name, a.PatientID, p.Name;

**Age Group Buckets – CASE**

SELECT

CASE

WHEN Age BETWEEN 18 AND 30 THEN '18-30'

WHEN Age BETWEEN 31 AND 50 THEN '31-50'

ELSE '51+'

END AS AgeGroup,

COUNT(\*) AS PatientCount

FROM Patients

GROUP BY AgeGroup;

**Contacts Ending with '1234' – String Function**

SELECT

UPPER(Name) AS PatientName,

ContactNumber

FROM Patients

WHERE ContactNumber LIKE '%1234';

**Patients Only Prescribed "Insulin" – Subquery**

SELECT DISTINCT PatientID

FROM Diagnoses

WHERE DiagnosisID IN (

SELECT DiagnosisID

FROM Medications

GROUP BY DiagnosisID

HAVING SUM(CASE WHEN MedicationName <> 'Insulin' THEN 1 ELSE 0 END) = 0

);

**Average Prescription Duration**

SELECT

DiagnosisID,

AVG(DATEDIFF(EndDate, StartDate)) AS AvgPrescriptionDuration

FROM Medications

GROUP BY DiagnosisID;

**Doctor with Most Unique Patients**

SELECT

d.DoctorID, d.Name AS DoctorName,

d.Specialization,

COUNT(DISTINCT a.PatientID) AS UniquePatients

FROM Appointments a

JOIN Doctors d ON a.DoctorID = d.DoctorID

GROUP BY d.DoctorID, d.Name, d.Specialization

ORDER BY UniquePatients DESC

LIMIT 1;