GITHUB REPO
PROJECT DEMO LINK

The reason Hospital Clinic Solutions has decided to focus on this management system is mainly to simplify the complex system of interactions and communication between healthcare professionals and their Patients. It will provide Patients easy access to their own health data such as prescription, bills, and doctor information. I believe this kind of information is critical in today's society and should be ready and within reach of all Patients. This could make future experiences less complicated such as, applying for insurance, and even possibly harmful or life threatening events in locations where the Patient's medical records aren't transferrable or easily accessible.

As the main focus of my database is to store information regarding hospital staff and Patient records, I will not be able to use real Doctor-Patient data as it would violate HIPAA rules. Under HIPAA, which stands for "Health Insurance Portability and Accountability Act of 1996," sensitive Patient health information is protected from being disclosed without the Patient's knowledge or consent. This limitation requires us to generate my own data for the system. As a solution, I have decided to employ the Python package 'Faker' which I can use to generate fake data to export into a CSV file. I can then take that CSV file and import it into the different Entity tables through MYSQL workbench, which will generate the necessary SQL script to apply to the database.

Business Rules

- Patients can be under the care of only one Doctor
- Doctors can have one or more Patients
- When Doctor writes into Medical Record, both assigned Nurse and Patient receive access to medical record
- Nurses can set up future appointments on orders of the Doctor
- Patient can reschedule or request appointment
- Rooms can hold multiple Patients
- Each Room is assigned one Nurse
- Patients have read-only access to Medical Records
- Doctors can assign Prescriptions
- Payments are made to Bills

QUERIES

This query utilizes INNER JOIN since both the appointment table and patients table have a common key. The idpatients key in the appointment table is the same as the SSN key in the patients table. With this clause you're able to return a new table of important information from both tables.

```
SELECT patients.FullName, appointment.appointmentdate,
appointment.appointmenttime
FROM appointment
INNER JOIN patients ON appointment.idpatient = patients.SSN
ORDER BY patients.SSN
```

I've added a second INNER JOIN clause since both the appointment table and doctors table have a common key. The iddoctors key in the appointment table is the same as the SSN key in the doctors table. Paired along with a LEFT JOIN clause in order to aggregate data from the doctors table foreign key iddepartment to retrieve the names of the departments the doctors work in.

```
SELECT department.departmentName AS 'Department', doctors.FullName AS
'Doctor Name', patients.FullName AS 'Patient Name',
appointment.appointmentdate AS 'Date', appointment.appointmenttime AS 'Time'
FROM appointment
INNER JOIN doctors ON appointment.iddoctor = doctors.SSN
INNER JOIN patients ON appointment.idpatient = patients.SSN
LEFT JOIN department ON doctors.iddepartment = department.iddepartment
ORDER BY appointment.appointmentdate
```

This is a query with a subquery inside of it. I'm asking for a table of data to be returned from the nurse table, but only if those nurses are already assigned to a room. The table will also hold information regarding whether each room is full and the names of the patients within the rooms.

```
SELECT
        `r`.idroom AS 'Room Number',
        `r`.roomstatus AS 'Room Status',
        `n`.FullName AS 'Nurse In Charge',
        `p1`.FullName AS 'Patient 1',
```

```
        `p2`.FullName AS 'Patient 2',
        `p3`.FullName AS 'Patient 3',
        `p4`.FullName AS 'Patient 4'
FROM nurse `n`
INNER JOIN room `r` ON `n`.SSN = `r`.idnurse
LEFT JOIN patients `p1` ON ((`r`.idpatient1 = `p1`.SSN))
LEFT JOIN patients `p2` ON ((`r`.idpatient2 = `p2`.SSN))
LEFT JOIN patients `p3` ON ((`r`.idpatient3 = `p3`.SSN))
LEFT JOIN patients `p4` ON ((`r`.idpatient4 = `p4`.SSN))
WHERE `n`.SSN IN (SELECT DISTINCT `r`.idnurse FROM room)
```

This is a script for a trigger within the patients table. The purpose of it is to automatically generate a message if a new patient (INSERT/UPDATE) is admitted to the hospital and hasn't paid a balance.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `patients_AFTER_INSERT`
AFTER INSERT ON `patients` FOR EACH ROW BEGIN
        IF NEW.payment IS NULL THEN
      INSERT INTO messages(idpatient, message)
      VALUES(new.SSN,CONCAT('Hi ', NEW.FullName, ', please pay ymy bill.'));
        END IF;
END
```

To test this trigger, I can run this query to admit a new patient

```
SET FOREIGN_KEY_CHECKS=0;
INSERT INTO patients_zip
  SET zip=78249, city='San Antonio', state='TX',street='1 UTSA Circle';
INSERT INTO patients
        SET SSN=12345678, FullName='Retro ManishimI',
Zip=LAST_INSERT_ID();
```

This is one of the possible views in the database, it allows to view detailed reports of patients with existing prescriptions and appointments.

```
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
```

```
VIEW `patient_report` AS
   SELECT
      `r`.`idreport` AS `report id`,
      `p`.`FullName` AS `Patient Name`,
      `p`.`Age` AS `Patient Age`,
      `m`.`medicinename` AS `Medicine Name`,
      `m`.`medicinecost` AS `Medicine Cost`,
      `a`.`appointmentdate` AS `Appointment Date`,
      `a`.`appointmenttime` AS `Appointment Time`,
      `a`.`app_length` AS `Appointment Length`,
      ((`a`.`fee_per_day` * `a`.`app_length`) + `m`.`medicinecost`) AS `Bill Total`,
      `p`.`payment` AS `Payment Made`,
      (((`a`.`fee_per_day` * `a`.`app_length`) + `m`.`medicinecost`) -
`p`.`payment`) AS `Current Balance`,
      `d`.`FullName` AS `Doctor Name`
   FROM
      ((((((`report` `r`
      JOIN `patients` `p` ON ((`r`.`idpatient` = `p`.`SSN`)))
      JOIN `bill` `b` ON ((`r`.`idbill` = `b`.`idbill`)))
      LEFT JOIN `prescription` `pr` ON ((`b`.`idprescription` =
`pr`.`idprescription`)))
      LEFT JOIN `medicine` `m` ON ((`pr`.`idmedicine` = `m`.`idmedicine`)))
      LEFT JOIN `appointment` `a` ON ((`b`.`idappointment` =
`a`.`idappointment`)))
      LEFT JOIN `doctors` `d` ON ((`a`.`iddoctor` = `d`.`SSN`)))
```

To see how this view is necessary I can look at the appointment table and the prescription table.

```
SELECT * FROM hospital.appointment;
SELECT * FROM hospital.prescription;
```

If I update one of the prescriptions to a new medicine, let's say the first prescription was an error

```
UPDATE prescription
SET idmedicine = 75
WHERE idprescription= 4
```

I re-run a query to see the report view again to see the change in the medicine, cost, bill total, and current balance

```
SELECT * FROM hospital.patient_report;
```

I can also update the appointment length and see similar results

```
UPDATE appointment
SET app_length = 45
WHERE idappointment= 1
```

Let’s see the view again

```
SELECT * FROM hospital.patient_report;
```