CS 432
Databases

Assignment I

Database Design and ER Diagrams

QUERY CRAFTERS

Group Members

Het Trivedi	21110226
Saurabh Kumar Sah	21110188
Shreya Patel	21110155
Riya Jain	21110178
Twinkle Devda	21110228
Ishika Raj	21110081
Saumya Jaiswal	21110186
Darsh Dalal	21110049

January 29th, 2024

2.1.1 Description of the Database System

The database functions as a comprehensive **Dispensary Management System**, intricately crafted to oversee various aspects of healthcare operations. It efficiently handles the inventory of medicines, monitoring stock levels, expiry dates, and dispensation records, capturing details of those administering medications. Additionally, the system adeptly maintains an extensive repository of patient information, encompassing crucial details like names, unique identifiers, consultation dates, existing allergies, and detailed medical histories. It also records pertinent information about medical practitioners, including identification or registration numbers, names, and prescribed treatments.

Patients' medical journeys are meticulously documented, chronicling administered medications, diagnoses, recommended tests, and subsequent results. This holistic approach ensures seamless data integration, offering a comprehensive overview of patient care. Furthermore, the system provides valuable insights for healthcare providers, offering data on prescription details. This facilitates quick access to patient records, aiding informed decision-making and enhancing overall healthcare workflow efficiency. The complex web of information within the database contributes to optimizing dispensary management and patient care, creating a streamlined and organized healthcare environment.

2.1.2

Stakeholders involved in this system are doctors, pharmacists, patients, medical employees, and suppliers. The stakeholders are diverse, with healthcare administrators benefiting from improved operational efficiency and streamlined inventory management for better resource allocation. Medical practitioners gain enhanced decision-making capabilities through organized patient data, while patients experience an elevated standard of care with meticulous health journey monitoring. The functional requirement of the database is multi-faceted and crucial for seamlessly integrating and managing diverse data sets, including detailed medicine inventory, comprehensive patient records, and practitioner information. Its efficiency lies in organizing, updating, and swiftly retrieving information, ensuring a well-organized and efficient healthcare environment.

Questions asked-

- What is the purpose of their database?
- What type of records do they store?
- How do they store records of COVID patients?
- How do they keep track of the data for campus outsiders in conditions like Inter IIT and National games?
- Is any data stored for the emergency case separately or will it be stored as the normal cases?
- How is the reimbursement procedure done?
- Which type of software are they currently using? Did campus students make it?
- Do they store the record for the blood tests done by the laboratory?
- What kind of entries are done by the doctors to the system?
- How do they keep track of the bills and medicines provided by the medical agencies?

The names of the individuals with whom we have interacted-

1. Ms. Parulben Christian, Assistant Staff Nurse
2. Mr. Mukesh Sharma, Senior Staff Nurse

2.2.1.1 Entities and their Attributes

1. **Entity**: Doctor
   Attributes: Doctor_ID, Name (First_Name, Middle_Name, Last_Name), DOB, Age(), Phone_Number, Specialization, Experience, Address (Street (Street_Number, Street_Name, Apartment_Number), City, Pincode), Gender, Availability (Day, Time), Email

2. **Entity:** Patient
   Attributes: Patient_ID, Name(First_Name, Middle_Name, Last_Name), DOB, Age(), Phone_Number, Address (Street (Street_Number, Street_Name, Apartment_Number), City, Pincode), Gender, Allergies, Email

3. **Entity:** Medical_Test
   Attributes: Test_Type, Lab_Name

4. **Entity:** Emergency (Weak Entity Set)
   Attributes: Patient_Name (First_Name, Middle_Name, Last_Name), License_Number (FK), Disease, Date, Time

5. **Entity:** Hospital
   Attributes: License_Number, Name, Address (Street (Street_Number, Street_Name, Apartment_Number), City, Pincode), Phone_Number

6. **Entity:** Insurance
   Attributes: Insurance_Number, Patient_ID (FK), Issue_Date, Expiry_date, Wallet_Balance, Reimbursement_Status

7. **Entity:** Prescription
   Attributes: Prescription_ID, Patient_ID (FK), Date, Item_ID (FK), Diagnosis

8. **Entity:** OPD
   Attributes: Serial Number, Date, Time, Patient Name (First Name, Middle Name, Last Name), Doctor Name (First Name, Middle Name, Last Name), Case Type

9. **Entity:** Supplier
   Attributes: Supplier_ID, Agency_Name, Phone_Number, Address (Street (Street_Number, Street_Name, Apartment_Number), City, Pincode), Bank_Details (Account_number, Bank_name, IFSC_code, Branch)

10. **Entity:** Medicines_ Equipments
    Attributes: Item_ID, Medicine_Name, Composition, Expiry_Date, Stock

11. **Entity:** Purchase_Order
    Attributes: Bill_Number, Item_ID (FK), Supplier_ID (FK), Quantity, Amount_Paid

12. **Entity:** Staff
    Attributes: Staff_ID, Name (First_Name, Middle_Name, Last_Name), Join_Date, Phone_Number, DOB, Age(), Address (Street (Street_Number, Street_Name, Apartment_Number), City, Pincode), Email, Gender, Salary

Relationship Sets

Sr.No.	Participating Entities	Relationship	Descriptive Attribute	Cardinality	Participation Constraints
1	Doctor → Prescription	Prescribes	-	One to Many	Doctor: Partial Prescription: Total
2	Doctor → Emergency	Recommendation	-	Many to Many	Doctor: Partial Emergency: Total
3	OPD → Prescription	Appointment	-	One to One	OPD: Total Prescription: Total
4	Medical_Test → Prescription	Med_Report	Test_Date, Result	Many to Many	Medical Test: Partial Prescription: Partial
5	Prescription → Patient	Prescribed_to	-	Many to One	Prescription: Total Patient: Partial
6	Prescription → Medicines_Equipments	Medication	-	Many to Many	Prescription: Partial Medicines &: Partial Equipments
7	Emergency → Patient	Urgent	-	Many to One	Emergency: Total Patient: Partial
8	Emergency → Hospital	Referred_to	-	Many to One	Emergency: Total Hospital: Partial
9	Insurance → Patient	Claimed_by	-	One to One	Insurance: Total Patient: Partial
10	Insurance → Prescription	Claimed_for	-	One to One	Insurance: Total Prescription: Partial
11	Supplier → Medicines _Equipments → Purchase_Order	Supply_transaction	-	One to Many to Many	Supplier: Partial Medicines & Equipment: Total Purchase Order: Total
12	Purchase_Order → Staff	Ordered_by	Order_date	Many to One	Purchase Order: Total Staff: Partial

2.2.1.2 Justifications-

c. At least one primary key and one foreign key.

Ans:

- **Prescription Entity:** Prescription_ID serves as the PRIMARY KEY, which uniquely identifies each prescription, and Patient_ID is an attribute of the Prescription entity, but it is also the primary key of the 'Patient' entity set. Hence, it's a FOREIGN KEY for the prescription entity set. This establishes a connection between a prescription and the patient it is associated with.

- **Insurance Entity:** Insurance_no is the PRIMARY KEY uniquely identifying each insurance entry. Also, Patient_ID here is a FOREIGN KEY as it is the primary key of a Patient entity set. This defines a relationship between an insurance entry and the patient insured with that insurance.

- **Purchase_Order Entity:** Bill_No is the PRIMARY KEY to identify each purchase uniquely. Item_ID and Supplier_ID are two attributes, which are the PRIMARY KEY of Medicine_Equipments and Supplier entity sets, respectively. Hence, these two are the FOREIGN KEYS of the "Purchase_Order" entity. This defines a ternary relationship between Purchase_Order, Medicine_Equipments, and Supplier entities.

d. At least one "one-to-one" relationship.

Ans:

- Relationship **"Appointment"** between Prescription and OPD entity sets is a *"one to one"* relationship, as for an OPD visit, there is only one fresh issuance of a new prescription, and a unique prescription is issued for only a single OPD entry.

- Relationship **"Claimed_by"** between the Insurance and Patient entity sets is a "one to one" relationship because only one patient can be insured for a single insurance registration and vice-versa.

e. At least any/both of (one-to-many, many-to-one) relationships.

Ans:

- The relationship "**Prescribes**" between the Doctor and Prescription entity sets is a "one-to-many" relationship because a single doctor can issue multiple prescriptions, but one prescription can be issued by only one doctor.

- The relationship "**Ordered_by**" between the Purchase_Order and Staff entity sets is a "many-to-one" relationship because multiple purchase orders can be placed by different staff members, but each purchase order is placed by only one staff member.

- The relationship "**Referred_to**" between the Emergency and Hospital entity sets is a "many-to-one" relationship because multiple emergency cases can be referred to or have a destination hospital, but each emergency case is associated with only one hospital as its destination.

f. At least one many-to-many relationship.

Ans:

- The relationship **“Medication”** between Prescription and Medicines_Equipments is a “many-to-many” relationship because one prescription can involve the prescription of multiple medicines and equipments. Conversely, a single medicine or equipments item can be part of multiple prescriptions.

- The relationship **“Med_Report”** between Prescription and Medical_Test is a “many-to-many” relationship because a single prescription may contain multiple medical tests, addressing various diagnostic needs for a patient. Conversely, multiple patients' prescriptions might include a particular medical test.

- The relationship **“Recommendation”** between Doctor and Emergency Cases is a “many-to-many” relationship because a single doctor can make recommendations for multiple emergency cases, addressing different medical situations. Simultaneously, a specific emergency case might involve recommendations from multiple doctors.

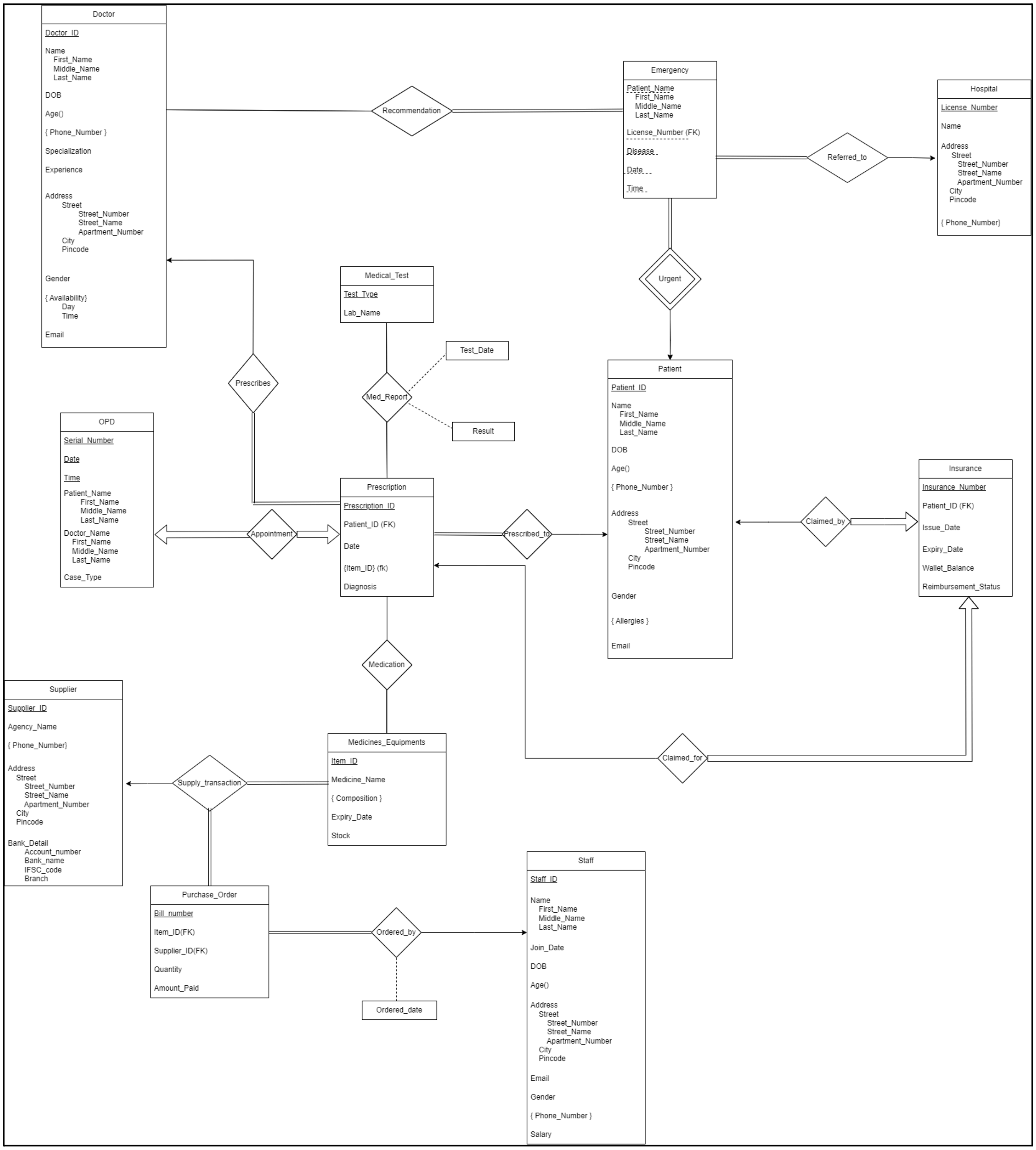
g. At least one of each (total & partial) participation constraint.

Ans:

- In the relationship "**Referred_to**" between Emergency and Hospital entity sets, there is **total** participation on the side of Emergency, signifying that every emergency case must be associated with a destination hospital. Conversely, on the side of the Hospital, there is **partial** participation, indicating that not every hospital needs to be linked to an emergency case. This accommodates scenarios where certain hospitals may not receive emergency cases.

- The relationship **"Ordered_by"** between the Purchase_Order and Staff entity sets exhibits **total** participation on the side of Purchase Order, meaning that every purchase order must be associated with a management staff member. This ensures that there are no instances of a purchase order without a responsible staff member. On the other hand, there is **partial** participation on the side of the Staff, allowing for scenarios where not every staff member needs to be involved in placing an order.

2.2.2 ER Diagram (Link)



2.3.1 Relational Schemas

Entity Sets to Relational Schemas

1. **Doctor** (Doctor_ID, First_name, Middle_Name, Last_Name, DOB, Gender, Email, Specialization, Experience, Street_Number, Street_Name, Apartment_Number, City, Pincode)

   PRIMARY KEY = Doctor_ID
   Constraints:
   - Name: NOT NULL.
   - DOB: NOT NULL
   - Email: UNIQUE
   - Specialization: NOT NULL
   - Experience: NOT NULL
   - Pincode: CHECK (>=100000 and <=999999).

Why is it needed?
To have a coherent database, name, DOB, specialization and experience must be present. And if email is present it should be unique as everyone has different email IDs. And as pin code only has six digits, CHECK constraint is needed.

2. **Doctor_Contact** (Doctor_ID, Phone_Number)

   PRIMARY KEY = (Doctor_ID, Phone_Number)
   FOREIGN KEY = Doctor_ID
   Constraints : Phone Number - CHECK(<10000000000 and >=1000000000)

Why is it needed?
Phone number has to be of ten digits so the CHECK constraint is needed.

3. **Doctor_Availability** (Doctor_ID, Date, Time)

   PRIMARY KEY = (Doctor_ID, Date, Time)’
   FOREIGN KEY= Doctor_ID
   Constraints: Date - NOT NULL

Why is it needed?
For checking the availability of a doctor, the date is necessary, so it should be NOT NULL.

4. **Patient** (Patient_ID, First_name, Middle_Name, Last_Name, DOB, Age, Street_Number, Street_Name, Apartment_Number, City, Pincode, Gender, Allergies, Email)

   PRIMARY KEY = Patient_ID
   Constraints:
   - Name: NOT NULL

- DOB: NOT NULL
- Email: UNIQUE
- Pincode: CHECK (>=100000 and <=999999).

Why is it needed?
To identify a patient, name and DOB have to be present, the DOB entry must be of Date format. And if email is present it should be unique as everyone has different email IDs. And as the pin code only has six digits, a CHECK constraint is needed.

5. **Patient_Contact** (Patient_ID, Phone_Number)

   PRIMARY KEY= (Patient_ID, Phone_Number)
   FOREIGN KEY = Patient_ID
   Constraints : Phone_Number - CHECK(<10000000000 and >=1000000000)

Why is it needed?
Phone number has to be ten digit so the CHECK constraint is needed.

6. **Patient_Allergy** (Patient_ID, Allergies)

   PRIMARY KEY= (Patient_ID, Allergies)
   FOREIGN KEY = Patient_ID

7. **Medicines_Equipments** (Item_ID, Medicine_Name, Composition, Expiry_Date, Stock)

   PRIMARY KEY = Item_ID
   Constraints:
   - Medicine_Name: NOT NULL.
   - Expiry_Date: NOT NULL.
   - Stock: CHECK (>=0).

Why is it needed?
Medicine name will help in the categorization of medicines based on their name so it is needed. Expiry Date is also necessary because if it will help in the timely incarnation of expired medicines. A Stock entity set is crucial for maintaining real-time visibility into inventory stock, so a CHECK is added.

8. **Emergency** (Patient_ID (PK), First_Name (PK), Middle_Name (PK), Last_Name (PK), License_Number (FK) and (PK), Disease (PK), Date (PK), Time (PK))

   PRIMARY KEY = Patient_ID
   FOREIGN KEY = License_Number
   Constraints:
   - Name: NOT NULL.
   - Disease: NOT NULL
   - Date: NOT NULL

Why it is needed?
When reducing a weak entity set to a relational schema, we take all the attributes of the weak entity set and the primary key of the corresponding strong entity as the attributes of the relational schema and we make all these attributes as the primary keys. Also, a description of the emergency case or illness is necessary for smooth medication hence disease should be not null.

9. **Staff** (Staff_ID, First_Name, Middle_Name, Last_Name, Join_Date, DOB, Age, Street_Number, Street_Name, Apartment_Number, City, Pincode, Email, Gender, Salary)

    PRIMARY KEY = Staff_ID

    Constraints:

    - Name: NOT NULL.
    - Join_Date: NOT NULL
    - DOB: NOT NULL
    - Pincode: CHECK (>=100000 and <=999999).
    - Email: UNIQUE
    - Salary: NOT NULL, CHECK (>=0).

Why it is needed?
To have a coherent database, name, DOB, Join Date and Salary must be present. And if email is present it should be unique as everyone has different email IDs. And as pin code only has six digits, CHECK constraint is needed.

10. **Staff_Contact** (Staff_ID, Phone_Number)

    PRIMARY KEY= (Staff_ID, Phone_Number)
    FOREIGN KEY = Staff_ID
    Constraints :
    - Phone Number - Check(<10000000000 and >=1000000000)

Why it is needed?
The constraint on Phone Number ensures that it falls within the valid range of phone numbers, with a CHECK constraint validating it to be less than 10 billion and greater than or equal to 1 billion. This ensures that phone numbers stored in the database adhere to standard formatting rules and are within acceptable numerical ranges for communication purposes.

11. **Medical Test** (Test_Type, Lab_Name)

    PRIMARY KEY = Test_Type

    Constraints:

    - Lab Name: NOT NULL.

Why it is needed?

NOT NULL constraint on Lab Name ensures essential information is always provided. These constraints support the precise categorization of medical tests and facilitate efficient management within the healthcare system.

12. **Prescription** (Prescription_ID, Patient_ID (FK), Date, Item_ID (FK), Diagnosis)

    PRIMARY KEY = Prescription_ID

    FOREIGN KEY = Patient_ID, Item_ID

    Constraints:

    - Date: NOT NULL
    - Diagnosis: NOT NULL

Why it is needed?
The Date and Diagnosis attributes can't be NULL as every prescription must have some defined values in these attributes.

13. **Prescription_Item** (Prescription_ID , Item_ID)

    PRIMARY KEY = (Prescription_ID, Item_ID)

    FOREIGN KEY = Prescription_ID , Item_ID

14. **Insurance** (Insurance_Number, Patient_ID (FK), Issue_Date, Expiry_Date, Wallet_Balance, Reimbursement_Status)

    PRIMARY KEY = Insurance_Number

    FOREIGN KEY = Patient_ID

    Constraints:

    - Issue_Date: NOT NULL
    - Expiry_Date: NOT NULL
    - Wallet_Balance: NOT NULL. CHECK(>0)
    - Reimbursement_Status: NOT NULL. DEFAULT Pending.

Why is it needed?
NOT NULL constraints on Issue Date, Expiry Date, Wallet Balance, and Reimbursement Status ensure essential information is always provided. The CHECK constraint on Wallet Balance validates positive values, crucial for financial transactions. The DEFAULT constraint on Reimbursement Status sets the default value to Pending, providing clarity on reimbursement status.

15. **Supplier** (Supplier_ID, Agency_Name, Street_Number, Street_Name, Apartment_Number, City, Pincode, Account_number, Bank_name, IFSC_code, Branch)

    PRIMARY KEY = Supplier_ID

    Constraints:

- Agency_Name: NOT NULL.
- Pincode: CHECK (>=100000 and <=999999)
- Account_number: NOT NULL, CHECK(>0)
- Bank_name: NOT NULL
- IFSC_code: NOT NULL
- Branch: NOT NULL

Why it is needed?
NOT NULL constraints on Agency Name, Bank name, IFSC code, and Branch guarantee essential details are always provided. The CHECK constraint on Pincode ensures valid geographic data. Account number is constrained to be non-negative, and vital for financial transactions.

16. **Supplier_Contact** (Supplier_ID, Phone_Number)

PRIMARY KEY= (Supplier_ID, Phone_Number)
FOREIGN KEY = Supplier_ID
Constraints : Phone_Number -CHECK(<10000000000 and >=1000000000)

Why it is needed?
The CHECK constraint on Phone Number validates its format, ensuring valid contact numbers are stored.

17. **Purchase Order** (Bill_Number, Item_ID (FK), Supplier_ID (FK), Quantity, Amount_Paid)

PRIMARY KEY = Bill_Number

FOREIGN KEY = Item_ID, Supplier_ID

Constraints:

- Quantity: NOT NULL, CHECK (>0)
- Amount_Paid: NOT NULL, CHECK (>0)

Why it is needed?

NOT NULL constraints on Quantity and Amount_Paid ensure essential transaction details are always provided. CHECK constraints validate Quantity and Amount_Paid to be greater than zero, preventing invalid entries.

18. **OPD** (Serial_Number, Date, Time, First_name_patient, Middle_name_patient, Last_name_patient, First_name_doctor, Middle_name_doctor, Last_name_doctor, Case_Type)

PRIMARY KEY = (Serial_Number, Date, Time)

Constraints:

- Case_Type: NOT NULL, (New or Old).

Why is it needed?

The Case Type attribute is constrained to be NOT NULL, providing a description of the visit type (New or Old). These constraints support precise record management and categorization of outpatient cases within the healthcare system.

19. **Hospital** (License_Number, Name, Street_Number, Street_Name, Apartment_Number, City, Pincode)

    PRIMARY KEY = License_Number

    Constraints:

    - Name: NOT NULL.
    - Pincode: CHECK (>=100000 and <=999999).

Why is it needed?
NOT NULL constraints on Name and address components guarantee essential details are always provided. The CHECK constraint on Pincode validates geographic data.

20. **Hospital_Contact** (Hospital_ID, Phone_Number)

    PRIMARY KEY= (Hospital_ID, Phone_Number)
    FOREIGN KEY = Hospital_ID
    Constraints:

    - Phone_Number : CHECK(<10000000000 and >=1000000000)

Why is it needed?
The CHECK constraint on Phone Number validates its format, ensuring valid contact numbers are stored.

Relationship Sets to Relational Schemas

1) Prescribes (Prescription_ID, Doctor_ID)

- PRIMARY KEY → Prescription_ID
- FOREIGN KEY → Prescription_ID, Doctor_ID
- Mapping Cardinality → One-to-Many (Doctor-Prescription)
- The mapping cardinality is one-to-many in the relationship between the "Doctor" and "Prescription" entity sets because a doctor can issue multiple prescriptions, but each prescription is issued by only one doctor.
- Why it is needed? - The Prescription_ID is chosen as the primary key because, during the representation of binary many-to-one relationship set, we chose the primary key of the entity set on the many side of the relationship set as the primary key of the schema.

2) Recommendation (Patient_ID, First_Name, Middle_Name, Last_Name, License_Number, Disease, Date, Time, Doctor_ID)

- PRIMARY KEY → Patient_ID, First_Name, Middle_Name, Last_Name, License_Number, Disease, Date, Time, Doctor_ID

- FOREIGN KEY → Patient_ID, Doctor_ID, License_Number
- Mapping Cardinality → Many-to-Many (Doctor-Emergency)
- The mapping cardinality is many-to-many between the "Doctor" and "Emergency" entities because multiple doctors can be involved in handling various emergency cases, and conversely, a single emergency case may require the expertise of multiple doctors.
- Why it is needed? - Here, the "Emergency" is a weak entity set, dependent on the "Patient" strong entity set. The primary key of "Emergency" is a combination of partial keys from the weak and primary key from the strong entity set, namely (First_Name, Middle_Name, Last_Name, License_Number, Disease, Date, Time) U (Patient_ID). Additionally, because the "Recommendation" relationship is many-to-many between "Emergency" and "Doctor," the final primary key includes the Primary Key of both “Emergency” and “Doctor”, hence it also includes Doctor_ID.

3) Appointment (Prescription_Id, Serial_Number)

- PRIMARY KEY → Prescription_ID
- FOREIGN KEY→ Prescription_ID, Serial_Number
- Mapping Cardinality → One-to-One (OPD-Prescription)
- In this relationship between the "OPD" and "Prescription" entity sets, the mapping cardinality is one-to-one, signifying that each appointment is uniquely linked to a specific prescription. This design reflects the scenario where each prescription, associated with a patient visit to the OPD, corresponds to a single appointment.
- Why it is needed? - The Prescription_ID is chosen as the primary key because, during the representation of binary one-to-one relationship set, we chose the primary key of the entity set on any side of the relationship set as the primary key of the schema.

4) Med_Report (Test_Type, Prescription_ID, Test_Date, Result)

- PRIMARY KEY→ Test_Type, Prescription_ID
- FOREIGN KEY → Test_Type, Prescription_ID
- Mapping Cardinality → Many to Many (Medical_Test - Prescription)
- The Cardinality between "Medical_Test" and "Prescription" entities is Many to Many as multiple medical tests can be associated with multiple prescriptions, and vice versa.
- Why it is needed? - Both the Test_Type and Prescription_ID are the primary key as in binary many to many relationship, we take union of the primary keys of both the participating entities.
- Here, Test_Date, Result are the descriptive attributes that provide additional information about each medical test.

5) Prescribed_to (Prescription_ID, Patient_ID)

- PRIMARY KEY → Prescription_ID
- FOREIGN KEY→ Prescription_ID, Patient_ID
- Mapping Cardinality → Many-to-one (Prescription-Patient)
- This design reflects the scenario where a patient may have multiple prescriptions over time, leading to a many-to-one relationship between prescriptions and patients.
- Why it is needed? - The Prescription_ID is chosen as the primary key because, during the representation of binary many-to-one relationship set, we chose the primary key of the entity set on the many side of the relationship set as the primary key of the schema.

6) Medication (Prescription_ID, Item_ID)

- PRIMARY KEY→ Prescription_ID , Item_ID
- FOREIGN KEY → Prescription_ID , Item_ID
- Mapping Cardinality → Many-to-Many (Prescription - Medicines_Equipments)
- The "Medication" relationship is characterized as "Many to Many" because multiple prescriptions can include multiple medicines and equipments, and conversely, a single medicine or equipment may be prescribed in multiple prescriptions.
- Why it is needed? - Since its many-to-many, the primary key of schema is union of primary keys of both the participating entities.

7) Urgent()

- There is no need to actually make a relational schema for this relationship set because this actually relates the weak entity set (Emergency) to its corresponding strong entity set (Patient) and we know that the schema for the relationship set linking a weak entity set to its corresponding strong entity set is redundant.

8) Referred_to (Patient_ID, First_Name, Middle Name, Last Name, License_Number, Disease, Date, Time, License_Number)

- PRIMARY KEY → Patient_ID, First_Name, Middle_Name, Last_Name, License_Number, Disease, Date, Time
- FOREIGN KEY → Patient_ID, License_Number
- Mapping Cardinality → Many-to-one (Emergency-Hospital)
- The cardinality signifies that multiple emergency cases can be associated with a single hospital, whereas a single emergency can be dealt with in a single hospital.
- Why it is needed? - The Primary Key of the entity set "Emergency" is chosen as the primary key of the schema because, during the representation of binary many-to-one relationship set, we chose the primary key of the entity set on the many side of the relationship set as the primary key of the schema.

9) Claimed by (Patient_ID, Insurance_Number)

- PRIMARY KEY → Patient_ID
- FOREIGN KEY → Patient_ID, Insurance_Number
- Mapping Cardinality → One-to-One (Patient - Insurance)
- This relationship between the "Patient" and "Insurance" entity sets has a one-to-one mapping cardinality, indicating that each patient can be uniquely associated with a single insurance claim.
- Why it is needed? - The Patient_ID is chosen as the primary key because, during the representation of binary one-to-one relationship set, we chose the primary key of the entity set on any side of the relationship set as the primary key of the schema.

10) Claimed_for (Prescription_ID, Insurance_Number)

- PRIMARY KEY → Prescription_ID
- FOREIGN KEY → Prescription_ID, Insurance_Number

- Mapping Cardinality → One-to-One (Prescription - Insurance)
- This relationship between the "Prescription" and "Insurance" entity sets has a one-to-one mapping cardinality, indicating that each prescription can be uniquely associated with a single insurance claim. This design aligns with the typical scenario where a prescription is linked to a specific instance of an insurance claim.
- Why it is needed? - The Prescription_ID is chosen as the primary key because, during the representation of binary one-to-one relationship set, we chose the primary key of the entity set on any side of the relationship set as the primary key of the schema.

11) Supply_transaction (Supplier_ID, Item_ID, Bill_Number)

- PRIMARY KEY → Item_ID, Bill_Number
- FOREIGN KEY → Supplier_ID, Item_ID, Bill_Number
- Mapping Cardinality → One-to-Many-to-Many (Supplier, Medicines & Equipments, Purchase Order)
- Why it is needed? - The Item_ID and Bill_Number are chosen as the primary key because, during the representation of the ternary one-to-many-to-many relationship set, we chose the primary key of the entity sets on 'many' sides of the relationship set as the primary key of the schema.
- In ternary relationship sets, cardinality is considered in the context of two entities as a collection relative to the third entity.
    - For (Supplier, Medicines & Equipment) to Purchase Order, for instance, a single supplier can supply a particular medicine or equipment in multiple purchase orders. Hence cardinality of the Purchase Order is "Many" relative to (Supplier, Medicines & Equipment).
    - For (Supplier, Purchase Order) to Medicines & Equipments, for instance, for a given supplier and a particular purchase order, multiple medicines and equipment can be supplied. Hence cardinality of Medicines & Equipments is "Many" relative to (Supplier, Purchase Order).
    - For (Purchase Order, Medicines & Equipments) to Supplier, for instance, for a given purchase order and a specific medicine or equipment, only one supplier is associated. Simply, a particular purchase order, paired with a specific medicine or equipment, is linked to a single supplier, hence cardinality of Supplier is "One" relative to (Purchase Order, Medicines & Equipments).

12) Ordered_by (Bill_Number, Staff_ID, Ordered_date)

- PRIMARY KEY → Bill_Number
- FOREIGN KEY→ Bill_Number, Staff_ID
- Mapping Cardinality → Many-to-one (Purchase Order - Staff)
- This design accommodates the common occurrence in which a staff member may be responsible for placing multiple purchase orders.
- Why it is needed? - The Bill_Number is chosen as the primary key because, during the representation of binary many-to-one relationship set, we chose the primary key of the entity set on the many side of the relationship set as the primary key of the schema

Contributions

Name	Group Number	Contribution
Shreya Patel	G1	Medical Center visit and Extraction of detail. Identification of entity sets and attributes, Wrote about the description of the database system and all the entities and attributes involved in the system.
Twinkle Devda	G1	Medical Center visit and Extraction of detail, Justifications, Compilation of final report, Documentation and proofreading.
Ishika Raj	G1	Medical Center visit and Extraction of detail, Justifications, Compilation of final report, Documentation and proofreading.
Het Trivedi	G1	Medical Center visit, Identification of entity sets and relation, Wrote about the description of the database system and contribution, Created the ER diagram.
Saurabh Kumar Sah	G2	Identification of attributes and entity sets, Created ER diagram for the system, Wrote definition of the entity set and attributes, and also their key constraint.
Darsh Dalal	G2	Identification of entity sets and relationship sets and deciding the participation (total/partial) of entity sets in relationship sets; designed a rough ER-Diagram; wrote justification for design requirement (c-g); reduced the relationship sets to relational schemas for which the primary key, foreign key, and mapping cardinality were figured out and explained; reduced redundancies; proof reading of the final doc
Riya Jain	G2	Identification of relations between entity sets and their cardinalities, Wrote definition of the entity set and attributes and their constraint and the entities and attributes involved in the system
Saumya Jaiswal	G2	Identification of entity sets, their attributes and formulation of relationship sets. Helped in writing justification for design features

		incorporated (c-g). Did reduction of relationship sets into relational schemas and decided the structure and attributes for the same, including the primary keys, foreign keys. Listed all mapping cardinalities and wrote their description. Proofreading of the complete document.

References:

1. Classroom slides [course webpage]
2. ER Diagram [Wikipedia]
3. How to use draw.io [Tutorials]
4. Ternary Relationship Description [Webpage]