LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC6001KP Advanced Database Systems Development

Assessment Weightage & Type

25% Individual Coursework

Year and Semester

2018-19 Autumn / 2018-19 Spring

Student Name: Animesh Gautam

London Met ID: 18029830

College ID:NP01CP4A180083

Assignment Due Date:

Assignment Submission Date:

Title (Where Required):

Word Count (Where Required):

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Introduction

Sukraraj Tropical and Infectious Disease hospital is the national referral hospital which was established in 1993 in Kathmandu Nepal. It provides all the services related to tropical and infectious disease with a capacity of 100 beds. It receives patients from all over the country and have services like out-patient, laboratory, X-ray and 24 hrs emergency. It also runs DOTS clinic and HIV/AIDS counseling and clinic every day. The hospital provides training to undergraduate and postgraduate medicine students including foreign students on infectious and tropical diseases, HIV/AIDS, Rabies, Snake bites etc. Its forte is treatment of major disease like typhoid fever, leishmaniasis(kala-azar), malaria, tetanus, rabies, snake bite, many animal bite (dog bite, monkey bite, leopard bite, bear bite, rate bite), meningitis, encephalitis etc.

The aim of this hospital is to provide better facilities, quality services and treatment for all, provide training to all medical students including foreigners and to enhance and upgrade the current services in a very systematic and efficient manner in upcoming days. The objectives of this hospital are:

- Expansion of services in the field of Infectious and Tropical disease in regional, zone and district levels.
- Improvement in the quality of existing services.
- Teaching and training for undergraduate and post graduate students (nation and international).
- Research activities.
- Community services including mobile services. (Khanal, 2016)

Current Business Activities and Operations

For any treatment or checkup in Sukraraj Tropical and Infectious Disease hospital first the patient must book their appointment. Patient gives his/her name for the booking. The appointment date and time is given by the staff to the patient. On the date of appointment full details of patient is taken like age, gender, address, then the details are checked to determine whether the patient is new or regular and kept in the record of hospital. Then at the time of appointment the patient is taken to a ward where the

checkup/treatment is done by a staff which can either be a certified or uncertified doctor/nurse/assistant. Then the patient pays the specific amount for the check/treatment he /she has gone through. If any of the certified staff which maybe doctor or nurse or assistant working in the same hospital comes for the checkup then he/she gets the treatment for free else if other uncertified staff comes for treatment then he/she should pay required amount like other patient. For that first the record system of hospital is checked to find whether he/she is certified or uncertified staff, then the details of that staff is added as a patient in the record system and rest of the process goes on for the treatment.

Current Business Rules

Every hospital does not have the same rule to run their business but might have some common rules. So, some of the business rule that I have created are as follows:

- A person can be a staff or a patient or both.
- A staff can be a certified or uncertified doctor/nurse/assistant.
- Each patient can be a regular/new.
- Certified doctor/nurse/assistant will get the treatment for free.
- Uncertified doctor/nurse/assistant will have to pay the amount required like other patient.
- Each patient can have one or more appointment.
- Each doctor/nurse/assistant can look after one or more appointment.
- Each person can have one or more address but one address can be of only one person.
- Each person can have one or more contacts like Phone number, Email address.
- A Treatment can be done in one appointment.
- An Appointment takes place in one ward.
- A Payment can be done of one appointment.

Identification of Entities and Attributes

Entities are defined as tables that hold specific information. Every entity in a database must have a different name. Entities are represented by mean of their properties, called attributes. All attributes have value.

Entites	Attributes
Person	Person_ID(P.K), Person_Type,Person_Name,Person_Gender, Person_Age.
Staff	Staff_ID(P.K), Person_ID(F.K), Staff_Type,Staff_Job.
Patient	Patient_ID(P.K), Person_ID(F.K), Patient_Type.
Address	Address_ID(P.K), Person_ID(F.K),Country,Zone,Street,Street_Number,Mailing_Address_Number.
Contact	Contact_ID(P.K),Address_ID(F.K),Person_ID(F.K),Fax_Number,Phone_Number,Email_Address, Landline_Number.
Appointment	Appointment_ID(P.K), Person_ID(F.K),Ward_No(F.K),Payment_ID(F.K), Appointment_Booking_Date,Appointment_Date,Appointment_Time, Staff_ID
Ward	Ward_No(P.K),Ward_Name,Ward_Block
Treatment	Appointment_ID(F.K),Person_ID(F.K),Treatment
Payment	Payment_ID(P.K),Payment_Date,Payment_Amount

Table 1-Entites and Attributes

Initial E-R Diagram

Entity-Relationship (ER) model of the data

The ER (Entity Relational Model) is a high-level conceptual data model diagram that helps us to analyze data requirements to produce a well-designed database. ER diagram displays the relationships of entity set stored in a database that helps us explain the logical structure of database. So, it is considered a best practice to complete ER modeling before implementing database. There are three basic components of ER Diagram: Entities, Attributes, Relationships. (Guru99, 2019)

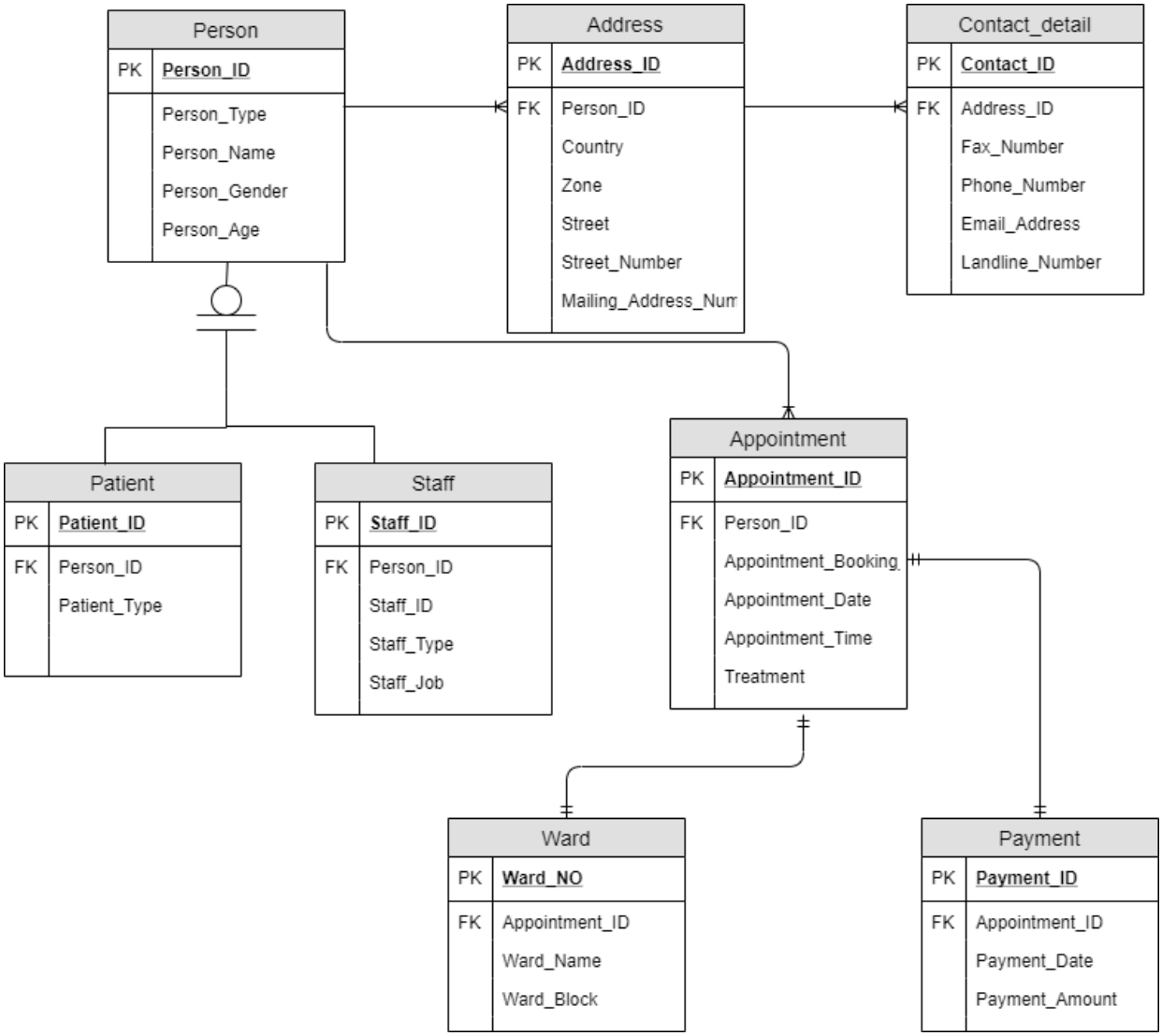

Figure 1-Initial E-R Diagram

Assumptions

Normalization

Normalization is a technique of splitting a large table into smaller tables and defining relationship between then to increase the clarity in organizing data. It is a systematic approach of decomposing tables to eliminate data redundancy and dependency of data. It is a multi-step process that puts data into tabular form, removing duplicate data from the relation table. Normalization should be done in such a way that the database design must be efficient, should be free of update, insertion and deletion anomalies and the data is stored logically. (Ahlawat, 2019)

1. **Un-Normalized Form (UNF)**

   The un-normalized form of normalization keeps all the data under a single entity where the repeating group are kept inside curly brackets and

repeating data are kept outside with a single key attribute to represent the entity.

Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age, { Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number{ Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number}},{ Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Treatment, Ward_No, Ward_Name, Ward_Block, Payment_ID, Payment_Date, Payment_Amount}, Staff_ID, Staff_Type, Staff_Job, Patient_ID, Patient_Type)

2. First Normal Form (1NF)

In 1NF the repeating groups are removed to separate Relation (Entity). The repeating group and the repeating data is separated from un-normalized form.

Person(<u>Person_ID</u>, Person_Type, Person_Name, Person_Gender, Person_Age, Staff_ID, Staff_Type, Staff_Job, Patient_ID, Patient_Type)

Person_Address(<u>Person_ID</u>*, <u>Address_ID</u>, Country, Zone, Street, Street_Number, Mailing_Address_Number)

Person_Contact(<u>Person_ID*,</u> <u>Address_ID*,</u> <u>Contact_ID,</u> Fax_Number, Phone_Number, Email_Address, Landline_Number)

Person_Appointment(<u>Person_ID*,</u> <u>Appointment_ID,</u> Appointment_Booking_Date, Appointment_Date, Appointment_Time, Treatment, Ward_No, Ward_Name, Ward_Block, Payment_ID, Payment_Date, Payment_Amount)

3. Second Normal Form (2NF)

In 2NF, the partial dependencies are removed in an entity. The partial dependencies should be first checked and then removed accordingly.

Partial Dependency

Partial Dependency occurs when a non-prime attribute is fuctionally dependent on part of a candidate key.

Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age, Staff_ID, Staff_Type, Staff_Job, Patient_ID, Patient_Type).

Since, Person has only one key attribute, it is in 2NF.

Person_Address(Person_ID*, Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number).

Address_ID -> Country, Zone, Street, Street_Number, Mailing_Address_Number
Person_ID* -> X
Addrerss_ID, Person_ID* -> X

Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID*)

Since, the address can be of any person, we kept the Person_ID as foreign key.

Person_Contact(Person_ID*, Address_ID*, Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number)

Contact_ID -> Fax_Number, Phone_Number, Email_Address, Landline_Number.

Person_ID* -> X

Address_ID* -> X

Person_ID*, Address_ID*, Contact_ID -> X

Contact_detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Person_ID*, Address_ID*)

The contact_details is based on any person with a particular address so we kept Person_ID and Address_ID in order to determine the contact of a person living in an address.

Person_Appointment(Person_ID*, Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time,

Treatment, Ward_No, Ward_Name, Ward_Block, Payment_ID,

Payment_Date, Payment_Amount)

Appointment_ID -> Appointment_Booking_Date, Appointment_Date,

Appointment_Time, Ward_No, Ward_Name, Ward_Block, Payment_ID,

Payment_Date, Payment_Amount

Person_ID* -> X

Person_ID*, Appointment_ID -> Treatment

Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Ward_No, Ward_Name, Ward_Block, Payment_ID, Payment_Date, Payment_Amount, Person_ID*)

Appoitnment_treatment(Person_ID*, Appointment_ID, Treatment)

4. Third Normal Form

In 3NF, transitive dependencies are removed in an entity. 3NF is used to reduce data duplication and achieve the data integrity.

Transitive Dependency

Transitive Dependency exists when there is an intermediate dependency. For example:

A, B and C be the three attributes and let the functional dependencies exists.

A -> B

B -> C

Then it can be stated that the following transitive dependency also holds

A -> B -> C

Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age, Staff_ID, Staff_Type, Staff_Job, Patient_ID, Patient_Type)

Person_ID -> Staff_ID -> Staff_Type, Staff_Job
Person_ID ->Person_Type, Person_Name, Person_Gender, Person_Age
Staff_ID -> Staff_Type, Staff_Job

Person_ID -> Patient_ID -> Patient_Type
Person_ID ->Person_Type, Person_Name, Person_Gender, Person_Age
Patient_ID -> Patient_Type

Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age)
Staff(Staff_ID, Staff_Type, Staff_Job, Person_ID*)

Patient(Patient_ID, Patient_Type, Person_ID*)

Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID*)

Since, there is no transitive dependencies it is already in 3NF.

Contact_detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Person_ID*, Address_ID*)

Since, there is no transitive dependencies it is already in 3NF.

Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Ward_No, Ward_Name, Ward_Block, Payment_ID, Payment_Date, Payment_Amount, Person_ID*)

Appointment_ID -> Ward_No -> Ward_Name, Ward_Block

Appointment_ID -> Appointment_Booking_Date, Appointment_Date, Appointment_Time

Ward_No -> Ward_Name, Ward_Block

Appointment_ID -> Payment_ID -> Payment_Date, Payment_Amount

Appointment_ID -> Appointment_Booking_Date, Appointment_Date, Appointment_Time

Payment_ID -> Payment_Date, Payment_Amount

Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time)

Appointment_Ward(Ward_No, Ward_Name, Ward_Block)

Appointment_Payment(Payment_Date, Payment_Amount)

Appoitnment_treatment(Person_ID*, Appointment_ID, Treatment)

Since, there is no transitive dependencies it is already in 3NF.

The final tables are:

Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age)

Staff(Staff_ID, Staff_Type, Staff_Job, Person_ID*)

Patient(Patient_ID, Patient_Type, Person_ID*)

Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID*)

Contact_detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Person_ID*, Address_ID*)

Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time)

Ward(Ward_No, Ward_Name, Ward_Block)

Payment(Payment_Date, Payment_Amount)

Treatment(Person_ID*, Appointment_ID, Treatment)

Entity Relation Diagram

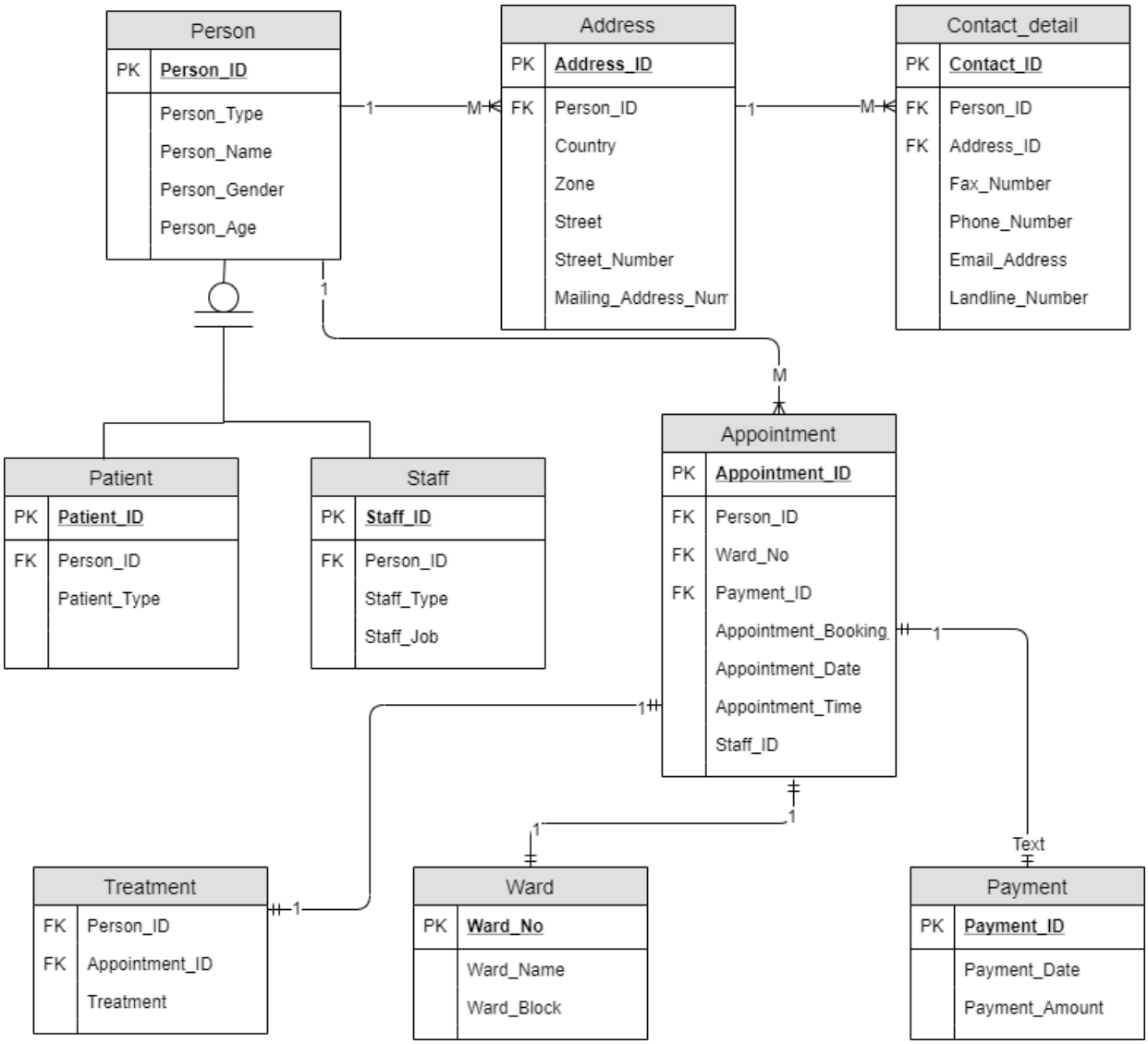

Figure 2- Final E-R Diagram

From the diagram, we can explain that a person can be a staff or patient or both. A person can have one or more address and the person with one or more address can also have one or more contact. A person can also take one or more appointments. An appointment can have only one treatment. An appointment also takes place in only one ward. A payment done by the patient, is of one appointment.

Database Implementation

Tables Generation (DDL Scripts)

```
create table Person(
        Person_ID number(2) NOT NULL,
        Person_Type varchar(10),
        Person_Name varchar(50),
        Person_Gender varchar(6),
        Person_Age number(3),
        CONSTRAINT Person_ID
        PRIMARY KEY (Person_ID));
```

```
SQL> create table Person(
  2   Person_ID number(2) NOT NULL,
  3   Person_Type varchar(10),
  4   Person_Name varchar(50),
  5   Person_Gender varchar(6),
  6   Person_Age number(3),
  7   CONSTRAINT Person_ID
  8   PRIMARY KEY (Person_ID));

Table created.
```

Figure 3- Table Generation of Person

```
create table Staff(
        Staff_ID varchar(5) NOT NULL,
        Staff_Type varchar(20),
        Staff_Job varchar(20),
        Person_ID number(2) NOT NULL,
        CONSTRAINT Staff_ID
        PRIMARY KEY(Staff_ID),
        CONSTRAINT Staff_Person_ID_Fk FOREIGN KEY(Person_ID) REFERENCES
        Person(Person_ID));
```

```
SQL> create table Staff(
  2  Staff_ID varchar(5) NOT NULL,
  3  Staff_Type varchar(20),
  4  Staff_Job varchar(20),
  5  Person_ID number(2) NOT NULL,
  6  CONSTRAINT Staff_ID
  7  PRIMARY KEY(Staff_ID),
  8  CONSTRAINT Staff_Person_ID_Fk FOREIGN KEY(Person_ID) REFERENCES Person(Person_ID));

Table created.
```

Figure 4- Table Generation of Staff

```
create table Patient(
        Patient_ID varchar(5) NOT NULL,
        Patient_Type varchar(30),
        Person_ID number(2) NOT NULL,
        CONSTRAINT Patient_ID
        PRIMARY KEY(Patient_ID),
        CONSTRAINT Patient_Person_ID_Fk FOREIGN KEY(Person_ID)
        REFERENCES Person(Person_ID));
```

```
SQL> create table Patient(
  2  Patient_ID varchar(5) NOT NULL,
  3  Patient_Type varchar(30),
  4  Person_ID number(2) NOT NULL,
  5  CONSTRAINT Patient_ID
  6  PRIMARY KEY(Patient_ID),
  7  CONSTRAINT Patient_Person_ID_Fk FOREIGN KEY(Person_ID) REFERENCES Person(Person_ID));

Table created.
```

Figure 5-Table Generation of Patient

```
create table Address(
    Address_ID varchar(5) NOT NULL,
    Country varchar(10),
    Zone varchar(10),
    Street varchar(30),
    Street_Number number(7),
    Mailing_Address_Number number(7),
    Person_ID number(2) ,
    CONSTRAINT Address_ID PRIMARY KEY (Address_ID),
    CONSTRAINT Address_Person_ID_FK FOREIGN KEY (Person_ID) REFERENCES Person(Person_ID));
```

```
SQL> create table Address(
  2      Address_ID varchar(5) NOT NULL,
  3      Country varchar(10),
  4      Zone varchar(10),
  5      Street varchar(30),
  6      Street_Number number(7),
  7      Mailing_Address_Number number(7),
  8      Person_ID number(2) ,
  9      CONSTRAINT Address_ID PRIMARY KEY (Address_ID),
 10      CONSTRAINT Address_Person_ID_FK FOREIGN KEY (Person_ID) REFERENCES  Person(Person_ID));

Table created.
```

Figure 6-Table Generation of Address

```
create table Contact_Detail(Person_ID number(2),
                    Address_ID varchar(5) NOT NULL,
                    Contact_ID varchar(5),
                    Fax_Number varchar(10),
                    Phone_Number number(15),
                    Email_Address varchar(50),
                    Landline_Number number(10),
                    CONSTRAINT Contact_ID PRIMARY KEY (Contact_ID),
                    CONSTRAINT Contact_Detail_Person_ID_FK FOREIGN KEY (Person_ID) REFERENCES Person(Person_ID),
                    CONSTRAINT Contact_Detail_Address_ID_FK FOREIGN KEY (Address_ID) REFERENCES Address(Address_ID));
```

```
SQL> create table Contact_Detail(Person_ID number(2),
  2     Address_ID varchar(5) NOT NULL,
  3     Contact_ID varchar(5),
  4     Fax_Number varchar(10),
  5     Phone_Number number(15),
  6     Email_Address varchar(50),
  7     Landline_Number number(10),
  8     CONSTRAINT Contact_ID PRIMARY KEY (Contact_ID),
  9     CONSTRAINT Contact_Detail_Person_ID_FK FOREIGN KEY    (Person_ID) REFERENCES Person(Person_ID),
 10     CONSTRAINT Contact_Detail_Address_ID_FK FOREIGN KEY (Address_ID)    REFERENCES Address(Address_ID));

Table created.
```

Figure 7-Table Generation of Address

```
create table Ward(
        Ward_No varchar(5),
        Ward_Name varchar(20),
        Ward_Block varchar(5),
        CONSTRAINT Ward_No PRIMARY KEY (Ward_No));
```

```
SQL> create table Ward(
  2  Ward_No varchar(5),
  3  Ward_Name varchar(20),
  4  Ward_Block varchar(5),
  5  CONSTRAINT Ward_No PRIMARY KEY (Ward_No));

Table created.
```

Figure 8-Table Generation of Ward

```
create table Payment(
        Payment_ID varchar(5),
        Payment_Date date,
        Payment_Amount number(7),
        CONSTRAINT Payment_ID PRIMARY KEY (Payment_ID));
```

```
SQL> create table Payment(
  2  Payment_ID varchar(5),
  3  Payment_Date date,
  4  Payment_Amount number(7),
  5  CONSTRAINT Payment_ID PRIMARY KEY (Payment_ID));

Table created.
```

Figure 9-Table Generation of Payment

```
create table Appointment(
        Person_ID number(2) NOT NULL,
        Ward_No varchar(5),
        Payment_ID varchar(5),
        Staff_ID  varchar(5),
        Appointment_ID varchar(5),
        Appointment_Booking_Date date,
        Appointment_Date date,
        Appointment_Time varchar(10),
        CONSTRAINT Appointment_ID PRIMARY KEY (Appointment_ID),
        CONSTRAINT Appointment_Ward_No_FK FOREIGN KEY (Ward_No)
REFERENCES Ward(Ward_No),
        CONSTRAINT Appointment_Payment_ID_FK FOREIGN KEY (Payment_ID)
REFERENCES Payment(Payment_ID),
        CONSTRAINT Appointment_Person_ID_FK FOREIGN KEY (Person_ID)
REFERENCES Person(Person_ID));
```

```
SQL>  create table Appointment(
  2  Person_ID number(2) NOT NULL,
  3   Ward_No varchar(5),
  4   Payment_ID varchar(5),
  5   Staff_ID  varchar(5),
  6   Appointment_ID varchar(5),
  7   Appointment_Booking_Date date,
  8   Appointment_Date date,
  9   Appointment_Time varchar(10),
 10   CONSTRAINT Appointment_ID PRIMARY KEY (Appointment_ID),
 11   CONSTRAINT Appointment_Ward_No_FK FOREIGN KEY (Ward_No) REFERENCES Ward(Ward_No),
 12   CONSTRAINT Appointment_Payment_ID_FK FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID),
 13   CONSTRAINT Appointment_Person_ID_FK FOREIGN KEY (Person_ID) REFERENCES Person(Person_ID));

Table created.
```

Figure 10- Table Generation of Appointment

```
create table Treatment(
        Appointment_ID varchar(5),
        Person_ID number(2) ,
        Treatment varchar(50),
        CONSTRAINT Treatment_Appointment_ID_FK FOREIGN KEY
(Appointment_ID) REFERENCES    Appointment(Appointment_ID),
        CONSTRAINT Treatment_Person_ID_FK FOREIGN KEY (Person_ID)
REFERENCES Person(Person_ID));
```

```
SQL>  create table Treatment(
  2  Appointment_ID varchar(5),
  3  Person_ID number(2) ,
  4  Treatment varchar(50),
  5  CONSTRAINT Treatment_Appointment_ID_FK FOREIGN KEY (Appointment_ID) REFERENCES Appointment(Appointment_ID),
  6  CONSTRAINT Treatment_Person_ID_FK FOREIGN KEY (Person_ID) REFERENCES Person(Person_ID));

Table created.
```

Figure 11-Table Generation of Treatment

Populate DB tables

Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (1,'Patient', 'Ram', 'Male', '35');

```
SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (1,'Patient', 'Ram', 'Male', '35');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (2,'Patient', 'kishwor', 'Male', '20');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (3,'Staff', 'Riya', 'Female', '22');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (4,'Staff', 'Sunil', 'Male', '25');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (5,'Patient', 'Sabita', 'Female', '34');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (6,'Patient', 'Suraj', 'Male', '55');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (7,'Patient', 'Phul Kumari', 'Female', '60');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (8,'Staff', 'Siddhant', 'Male', '25');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (9,'Staff', 'Rita', 'Female', '27');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (10,'Staff', 'Birendra', 'Male', '30');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (11,'Staff', 'Prasidha', 'Male', '43');

1 row created.

SQL> Insert into Person(Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age) values (12,'Staff', 'Ashray', 'Male', '32');

1 row created.
```

Figure 12-Populating Person Table

Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S001','Certified','Doctor',10);

```
SQL>       Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S001','Certified','Doctor',10);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S002','Certified','Nurse',3);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S003','Certified','Nurse',9);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S004','Certified','Doctor',8);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S005','Uncertified','Doctor',11);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S006','Uncertified','Nurse',4);

1 row created.

SQL> Insert into Staff (Staff_ID, Staff_Type,Staff_Job,Person_ID) values ('S007','Certified','Assitant',12);

1 row created.
```

Figure 13--Populating Staff Table

Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P001','Regular',1);

```
SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P001','Regular',1);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P002','Regular_Uncertified_Nurse',5);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P003','New',2);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P004','Regular_Certified_Doctor',6);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P005','Regular',7);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P006','New_Uncertified_Doctor',11);

1 row created.

SQL> Insert into Patient (Patient_ID, Patient_Type,Person_ID) values ('P007','New_Certified_Nurse',3);

1 row created.
```

Figure 14--Populating Patient Table

Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A01', 'Nepal', 'Bagmati', 'Brahmakumari', '3056', 44600, 3);

```
SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A01', 'Nepal', 'Bagmati', 'Brahmakumari', '3056', 44600, 3);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A02', 'Nepal', 'Bagmati', 'SiddhartTole', '4372', 44600, 5);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A03', 'Nepal', 'Janakpur', 'RamChowk', '5811', 78351, 7);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A04', 'Nepal', 'Bagmati', 'Nayabazar', '36121', 33455, 2);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A05', 'Nepal', 'Narayani', 'Sihpur', '47823', 65193, 3);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A06', 'Nepal', 'Bagmati', 'Ranibari', '68317', 34661, 6);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A07', 'Nepal', 'Bagmati', 'DamlaGalli', '83512', 99881, 1);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A08', 'Nepal', 'Bagmati', 'TufanChowk', '79131', 68300, 9);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A09', 'Nepal', 'Bagmati', 'GaneshChowk', '28225', 97110, 12);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A010', 'Nepal', 'Bagmati', 'BalajuHeight', '14160', 82660, 7);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A011', 'Nepal', 'Bagmati', 'KaldharaChowk', '54321', 73909, 10);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A012', 'Nepal', 'Bagmati', 'Bohoratar', '11037', 58503, 4);

1 row created.
```

Figure 15--Populating Address Table

```
SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A012', 'Nepal', 'Bagmati', 'Bohoratar', '11037', 58503, 4);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A013', 'Nepal', 'Bagmati', 'HighVision', '25460', 95478, 8);

1 row created.

SQL> Insert into Address(Address_ID, Country, Zone, Street, Street_Number, Mailing_Address_Number, Person_ID) values('A014', 'Nepal', 'Bagmati', 'Kapurdhara', '94475', 52304, 11);

1 row created.

SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C01', '544-7748', 9841567341,'prasidha120@gmail.com', 443351, 'A014', 11);

1 row created.

SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C02', '845-7752', 9845714011,'rya325@gmail.com', 443250, 'A05', 3);

1 row created.
```

Figure 16--Populating Address Table

Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C01', '544-7748', 9841567341,'prasidha120@gmail.com', 443351, 'A014', 11);

```
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C01', '544-7748', 9841567341,'prasidha120@gmail.com', 443351, 'A014', 11);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C02', '845-7752', 9845714011,'rya325@gmail.com', 443250, 'A05', 3);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C03', '221-5410', 9865738721, 'Rita777@gmail.com', 443621, 'A08', 9);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C04', '543-5106', 9821532534,'phulKmr@gmail.com', 448551, 'A010', 7);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C05', '132-2356', 9841234636,'srj9988@gmail.com', 448428, 'A06', 6);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C06', '356-3004', 9848426523,'sunill1@gmail.com', 449872, 'A012', 4);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C07', '467-5623', 9843696503,'sabita4321@gmail.com', 448417, 'A02', 5);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C08', '145-9413', 9865212351,'ram22@gmail.com', 448752, 'A07', 1);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C09', '567-6548', 9812457862,'riya444@gmail.com', 448457, 'A01', 3);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C010', '544-7512', 9889763246,'ashrayy97@gmail.com', 447884, 'A09', 12);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C011', '234-8187', 9848792531,'sid6582@gmail.com', 443354, 'A013', 8);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C012', '926-8412', 9846248426,'phulkumari8@gmail.com', 443875, 'A03', 7);
1 row created.
```

Figure 17--Populating Contact_Detail Table

```
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C012', '926-8412', 9846248426,'phulkumari8@gmail.com', 443875, 'A03', 7);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C013', '842-2351',9846745681,'bire101@gmail.com', 443303, 'A011', 10);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C014', '322-1243',9365866286,'kishwor37@gmail.com', 440057, 'A04', 2);
1 row created.
SQL> Insert into Contact_Detail(Contact_ID, Fax_Number, Phone_Number, Email_Address, Landline_Number, Address_ID, Person_ID) values ('C015', '953-2157',9841879955,'rita24@gmail.com', 446587, 'A08', 9);
1 row created.
```

Figure 18-Populating Contact_Detail Table

Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (12, 'General', 'B');

```
SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (12, 'General', 'B');

1 row created.

SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (22, 'X-Ray', 'A');

1 row created.

SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (10, 'Emergency', 'C');

1 row created.

SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (15, 'ENT', 'B');

1 row created.

SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (5, 'ICU', 'E');

1 row created.

SQL> Insert into Ward(Ward_No,Ward_Name,Ward_Block) values (20, 'Cardiology', 'A');

1 row created.
```

Figure 19-Populating Ward Table

Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA1', '24-Dec-2019', 5000);

```
SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA1', '21-Dec-2019', 5000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA2', '21-Dec-2019', 0);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA3', '22-Dec-2019', 10000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA4', '22-Dec-2019', 7000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA5', '22-Dec-2019', 2000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA6', '23-Dec-2019', 3000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA7', '23-Dec-2019', 15000);

1 row created.

SQL> Insert into Payment(Payment_ID, Payment_Date, Payment_Amount) values ('PA8', '24-Dec-2019', 1000);

1 row created.
```

Figure 20-Populating Payment Table

Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP1', '20-Dec-2019', '21-Dec-2019', '11:00am', 3, 'S001', 10, 'PA2');

```
SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP1', '20-Dec-2019', '21-Dec-2019', '11:00am', 3, 'S0
01', 10, 'PA2');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP2', '20-Dec-2019', '21-Dec-2019', '1:00pm', 7, 'S00
2', 22, 'PA3');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP3', '21-Dec-2019', '22-Dec-2019', '10:00am', 1, 'S0
01', 15, 'PA4');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP4', '21-Dec-2019', '22-Dec-2019', '12:00pm', 11, 'S
003', 3, 'PA6');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP5', '22-Dec-2019', '23-Dec-2019', '11:00am', 6, 'S0
05', 7, 'PA7');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP6', '23-Dec-2019', '24-Dec-2019', '10:00am', 5, 'S0
06', 5, 'PA5');

1 row created.

SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP7', '23-Dec-2019', '24-Dec-2019', '12:00pm', 2, 'S0
07', 20, 'PA8');

1 row created.
```

Figure 21-Populating Appointment Table

```
SQL> Insert into Appointment(Appointment_ID, Appointment_Booking_Date, Appointment_Date, Appointment_Time, Person_ID, Staff_ID, Ward_No, Payment_ID) values ('AP8', '23-Dec-2019', '24-Dec-2019', '2:00pm', 1, 'S00
4', 12, 'PA1');

1 row created.
```

Figure 22-Populating Appointment Table

Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP1', 3, 'Rabies');

```
SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP1', 3, 'Rabies');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP2', 7, 'Malaria');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP3', 1, 'Tetanus');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP4', 4, 'SnakeBite');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP5', 6, 'Typhoid');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP6', 5, 'Meningitis');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP7', 2, 'Encephalitis');

1 row created.

SQL> Insert into Treatment(Appointment_ID, Person_ID, Treatment) values ('AP8', 1, 'Fever');

1 row created.
```

Figure 24- Populating Treatment Table

```
SQL> update treatment set Person_ID=11 where Appointment_ID='AP4';

1 row updated.
```

Figure 23-Modifying values of Treatment Table

Due to wrong input of data in the Person_ID column of Treatment table the data had to be modified.

Database Querying

A query is a request for data or information from database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs.

Information Queries

```
SQL> select person.person_id, person.person_name, person.person_type, person.person_gender, person.person_age,patient.patient_id, patient.patient_type from person join patient on (person.person_id=patient.person
_id);

 PERSON_ID PERSON_NAME                                        PERSON_TYP PERSON PERSON_AGE PATIE PATIENT_TYPE
---------- ---------------------------------------------------- ---------- ------ ---------- ----- ------------------------------
         1 Ram                                                  Patient    Male           35 P001  Regular
         2 kishwor                                              Patient    Male           20 P003  New
         3 Riya                                                 Staff      Female         22 P007  New_Certified_Nurse
         5 Sabita                                               Patient    Female         34 P002  Regular_Uncertified_Nurse
         6 Suraj                                                Patient    Male           55 P004  Regular_Certified_Doctor
         7 Phul Kumari                                          Patient    Female         60 P005  Regular
        11 Prasidha                                             Staff      Male           43 P006  New_Uncertified_Doctor

7 rows selected.
```

Figure 25-Information Query 1

This query connects person and patient table and displays person id, person_name, person_type, person_gender, person_age, patient_id, patient_type. Overall it shows all the patient, regular and new.

```
SQL> select person.person_id, person.person_name, person.person_type, person.person_gender, person.person_age,patient.patient_id, patient.patient_type, address.address_id, address.country, address.zone, address.
street, address.street_number, address.mailing_address_number from person join patient on person.person_id=patient.person_id join address on person.person_id= address.person_id;

 PERSON_ID PERSON_NAME                                        PERSON_TYP PERSON PERSON_AGE PATIE PATIENT_TYPE                  ADDRE COUNTRY    ZONE       STREET                         STREET_NUMBER MAILING_AD
DRESS_NUMBER
---------- ---------------------------------------------------- ---------- ------ ---------- ----- ------------------------------ ----- ---------- ---------- ------------------------------ ------------- ----------
------------
         3 Riya                                                 Staff      Female         22 P007  New_Certified_Nurse            A01   Nepal      Bagmati    Brahmakumari                            3056
 44600
         5 Sabita                                               Patient    Female         34 P002  Regular_Uncertified_Nurse      A02   Nepal      Bagmati    SiddhartTole                            4372
 44600
         7 Phul Kumari                                          Patient    Female         60 P005  Regular                        A03   Nepal      Janakpur   RamChowk                                5811
          78351
         2 kishwor                                              Patient    Male           20 P003  New                            A04   Nepal      Bagmati    Nayabazar                              36121
          33455
         3 Riya                                                 Staff      Female         22 P007  New_Certified_Nurse            A05   Nepal      Narayani   Sihpur                                 47823
 65193
         6 Suraj                                                Patient    Male           55 P004  Regular_Certified_Doctor       A06   Nepal      Bagmati    Ranibari                               68317
 34661
         1 Ram                                                  Patient    Male           35 P001  Regular                        A07   Nepal      Bagmati    DamlaGalli                             83512
          99881
         7 Phul Kumari                                          Patient    Female         60 P005  Regular                        A010  Nepal      Bagmati    BalajuHeight                           14160
          82660
        11 Prasidha                                             Staff      Male           43 P006  New_Uncertified_Doctor         A014  Nepal      Bagmati    Kapurdhara                             94475
 52304

9 rows selected.
```

Figure 26-Information Query 2

This query connects person, patient and address table and displays all the details of patient with their address.

```
SQL>  select p.person_id, p.person_type, p.person_name,s.staff_id, s.staff_type, s.staff_job,a.appointment_id, p.payment_id, (p.payment_amount)*12/100 Amount_Recieved from appointment a join person p on a.person
_id=p.person_id join staff s on a.staff_id=s.staff_id join payment p on a.payment_id=p.payment_id where s.staff_type='Certified' and s.staff_job='Doctor' ;

 PERSON_ID PERSON_TYP PERSON_NAME                                        STAFF STAFF_TYPE          STAFF_JOB           APPOI PAYME AMOUNT_RECIEVED
---------- ---------- ---------------------------------------------------- ----- ------------------- ------------------- ----- ----- --------------
         3 Staff      Riya                                                 S001  Certified           Doctor              AP1   PA2              0
         1 Patient    Ram                                                  S001  Certified           Doctor              AP3   PA4            840
         1 Patient    Ram                                                  S004  Certified           Doctor              AP8   PA1            600
```

Figure 27-Information Query 3

This query connects person, staff, appointment and payment table, displaying all the certified doctor that have/has conducted an appointment and the amount he/she received.

```
SQL> select person.person_id, person.person_name, person.person_type, person.person_gender, person.person_age, staff.staff_id, staff.staff_type, staff.staff_job, patient.patient_id, patient.patient_type from per
son join staff on person.person_id=staff.person_id join patient on staff.person_id=patient.person_id;

 PERSON_ID PERSON_NAME                                          PERSON_TYP PERSON PERSON_AGE STAFF STAFF_TYPE          STAFF_JOB           PATIE PATIENT_TYPE
---------- -------------------------------------------------- ---------- ------ ---------- ----- -------------------- -------------------- ----- ------------------------------
         3 Riya                                               Staff      Female         22 S002  Certified            Nurse                P007  New_Certified_Nurse
        11 Prasidha                                           Staff      Male           43 S005  Uncertified          Doctor               P006  New_Uncertified_Doctor
```

Figure 28-Information Query 4

This query connects person, staff, and patient table that displays the information of all staff that are also patient.

Transaction Queries

```
SQL>  select p.person_id, p.person_type, p.person_name, pa.patient_id, pa.patient_type, a.appointment_id, s.staff_id, s.staff_type, s.staff_job, t.treatment, payment.payment_amount from person p join patient pa
on p.person_id=pa.person_id join appointment a on p.person_id=a.person_id join staff s on p.person_id=s.person_id join treatment t on p.person_id=t.person_id join payment on a.payment_id=payment.payment_id where
 s.staff_type='Uncertified' and s.staff_job='Doctor';

 PERSON_ID PERSON_TYP PERSON_NAME                                          PATIE PATIENT_TYPE                 APPOI STAFF STAFF_TYPE          STAFF_JOB           TREATMENT
    PAYMENT_AMOUNT
---------- ---------- -------------------------------------------------- ----- ------------------------------ ----- ----- -------------------- -------------------- --------------------------------------------------
--- --------------
        11 Staff      Prasidha                                           P006  New_Uncertified_Doctor        AP4   S005  Uncertified          Doctor              SnakeBite
         3000
```

Figure 29-Transaction Query 1

This query connects person, patient, appointment, staff, treatment and payments table. It displays the information of a person who is uncertified doctor of the hospital, took an appointment for a treatment and the amount he/she paid for the treatment.

```
SQL> select a.appointment_id, a.appointment_booking_date, a.appointment_date, a.appointment_time, w.ward_no, w.ward_name, w.ward_block from appointment a join ward w on a.ward_no= w.ward_no where w.ward_name='Em
ergency';

APPOI APPOINTME APPOINTME APPOINTMEN WARD_ WARD_NAME           WARD_
----- --------- --------- ---------- ----- -------------------- -----
AP1   20-DEC-19 21-DEC-19 11:00am    10    Emergency            C
```

Figure 30-Transaction Query 2

This query connects two tables, appointment and ward and displays all the appointment that have been conducted in emergency ward.

```
SQL> select p.person_name, p.person_type, s.staff_id, s.staff_type, s.staff_job, a.appointment_id, a.appointment_date, a.appointment_time from staff s join appointment a on s.staff_id= a.staff_id join person p o
n s.person_id=p.person_id where a.appointment_date='24-DEC-2019';

PERSON_NAME                                        PERSON_TYP STAFF STAFF_TYPE          STAFF_JOB            APPOI APPOINTME APPOINTMEN
-------------------------------------------------- ---------- ----- -------------------- -------------------- ----- --------- ----------
Sunil                                              Staff      S006  Uncertified          Nurse                AP6   24-DEC-19 10:00am
Ashray                                             Staff      S007  Certified            Assitant             AP7   24-DEC-19 12:00pm
Siddhant                                           Staff      S004  Certified            Doctor               AP8   24-DEC-19 2:00pm
```

Figure 31-Transaction Query 3

This query connects person, staff, and appointment table, displaying all the staffs that have conducted an appointment on a given date.

```
SQL> select p.person_name, pa.patient_id, pa.patient_type, a.appointment_id, a.appointment_booking_date, a.appointment_date from person p join patient pa on p.person_id=pa.person_id join appointment a on p.perso
n_id=a.person_id where a.appointment_date='22-DEC-2019';

PERSON_NAME                                        PATIE PATIENT_TYPE                    APPOI APPOINTME APPOINTME
-------------------------------------------------- ----- ------------------------------- ----- -------- --------
Ram                                                P001  Regular                         AP3   21-DEC-19 22-DEC-19
Prasidha                                           P006  New_Uncertified_Doctor         AP4   21-DEC-19 22-DEC-19
```

Figure 32-Transaction Query 4

This query connects three table: person, patient and appointment. It displays all patients that booked an appointment on a given date.

Dump File

```
C:\WINDOWS\system32>E:

E:\>exp PatientRecordSystem/coursework file=coursework.dmp

Export: Release 11.2.0.2.0 - Production on Mon Dec 23 22:39:26 2019

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user PATIENTRECORDSYSTEM
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user PATIENTRECORDSYSTEM
About to export PATIENTRECORDSYSTEM's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export PATIENTRECORDSYSTEM's tables via Conventional Path ...
. . exporting table                        ADDRESS         14 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                    APPOINTMENT          8 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 CONTACT_DETAIL         15 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        PATIENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        PAYMENT          8 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         PERSON         12 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                          STAFF          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                      TREATMENT          8 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                           WARD          8 rows exported
EXP-00091: Exporting questionable statistics.
```

Figure 33- Creation of Dump File

```
EXP-00091: Exporting questionable statistics.
. . exporting table                      TREATMENT          8 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                           WARD          8 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.
```

Figure 34-Creation of Dump File

Critical Evaluation

While doing our coursework we gathered a lot of experience on how the hospital work, how hospital records their data, different business rules. We got a better skill regarding analyzation of case scenario, identification of attributes. While analyzing the case I was able to make a rough estimate on how would our coursework model look like and initial entities and attributes. While identifying entities and attributes, I was able to make an initial E-R diagram and show the relationship between the tables.

After identifying entities and attributes, and making E-R diagram, normalization process was done and 3NF result set was generated. Using different queries like create, insert and select tables were created, values were inserted and displayed. Many difficulties were faced while creating the coursework like implementation of scenario, normalization, data insertion and data display. These all difficulties were resolved by consulting with module teachers, researching, going through lecture slide and taking references from book.

Critical Assessment of coursework

Overall the coursework included the techniques like analyzing the scenario, normalization technique, table creation and insertion technique, and data display technique. These techniques were enhanced during the completion of the coursework. While undertaking this database project the process involved were analyzation of scenario, identification of entities and attributes, creation of E-R diagram, normalization, table generation, table population and querying. After the completion of coursework, we had a better understanding of these techniques. Database model is very helpful in real life situation and also relates to other modules like Emerging, Java, Linux and python. We had a better skills regarding creation of this database which is very helpful for future employment as it is applicable in real life scenario.