Name: Tam Duy Vo

PGR111 Databases Individual Written Home Exam H2020

<u>Question 1</u>:

a. Identify the entities:

1. Employee 7. Patient 13. SeniorDentist

2. Hygiene_control 8. Problem 14. TraineeDentist

3. Dental_treatment 9. Room 15. Records

4. Treatment 10. Time_slot 16. JuniorTraineeAppointment

5. Fees 11. Job

6. Receptionist 12. Hygienist

b. Attributes and datatype for each table:

Employee	
Attribute	**Data Type**
ID	Int
First_name	Varchar(12)
Last_name	Varchar(12)
Email	Varchar(50)
Contact	Varchar(9)
address	Varchar(40)
Dental_treatment	
Attribute	**Data Type**
Hygienist_Id	Int
Senior_dentist_Id	Int
Treatment_Id	Int

Room	
Attribute	**Data Type**
Room_Id	Varchar(1)
Start_time	Time

Hygiene_control	
Attribute	**Data Type**
Hygienist_Id	Int
Treatment_Id	Int

Treatment	
Attribute	**Data Type**
Treatment_Id	Int
Problem_Id	Int
Room_Id	Varchar(11)
Date	Date
Start_time	Time
Time_slot	
Attribute	**Data Type**
Start_time	Time
End_time	Time

Problem	
Attribute	**Data Type**
Patient_Id	Int
Problem_Id	Int
Description	text

Hygienist	
Attribute	**Data Type**
Hygienist_Id	Int
TraineeDentist	
Attribute	**Data Type**
Trainee_dentist_Id	Int

JuniorTraineeAppointment	
Attribute	**Data Type**

Treatment_Id	Int
Trainee_dentist_Id	Int

SeniorDentist	
Attribute	**Data Type**
Senior_dentist_Id	Int

Receptionist	
Attribute	**Data Type**
Reptionist_Id	Int

Patient	
Attribute	**Data Type**
Patient_Id	Int
First_name	Varchar(12)
Last_name	Varchar(12)
Age	Int
Email	Varchar(40)
Contact	Varchar(10)
Address	Varchar(40)

Job	
Attribute	**Data Type**
Job_Id	Int
Title	Varchar(12)
salary	Int

Records	
Attribute	**Data Type**
Receptionist_Id	Int
Patient_Id	Int
Date	Date
Time	Time
Problem_Id	Int

Fees	
Attribute	**Data Type**
Treatment_Id	Int
Amount	Int

c. Candidate Key, Primary Key, Foreign Keys of each entity:

1. Employee
   - Primary Key (ID)
   - Candidate Key (Email), (Contact)
2. Hygiene_control
   - Primary Key (Treatment_Id)
   - Foreign Key (Hygienist_Id references Hygienist),

     (Treatment_Id references Treatment)
3. Dental_treatment
   - Primary Key (Treatment_Id)
   - Foreign Key (Senior_dentist_Id references SeniorDentist), (Hygienist_Id references Hygienist), (Treatment_Id references Treatment)

4. Treatment
    - Primary Key (Treatment_Id)
    - Foreign Key (Problem_Id references Problem), (Room_Id, Start_time references Room)
5. Fees
    - Primary Key (Treatment_Id)
    - Foreign Key (Treatment_Id references Treatment)
6. Records
    - Primary Key (Receptionist_Id, Date, Time)
    - Foreign Key (Receptionist_Id references Receptionist), (Patient_Id references Patient), (Problem_Id references Problem)
7. Patient
    - Primary Key (Patient_Id)
    - Candidate Key (Email), (Contact)
8. Problem
    - Primary Key (Problem_Id)
    - Foreign Key (Patient_Id references Patient)
9. Room
    - Primary Key (Room_Id, Start_time)
10. Time_slot
    - Primary Key (Start_time)
    - Candidate Key (End_Time)
    - Foreign Key (Start_time references Room)
11. Hygienist
    - Primary Key (Hygienist_Id)
    - Foreign Key (Hygienist_Id references Employee)
12. Senior Dentist
    - Primary Key (Senior_Dentist_Id)
    - Foreign Key (Senior_Dentist_Id references Employee)

13. Receptionist

- Primary Key (Receptionist_Id)
- Foreign Key (Receptionist_Id references Employee)

14. Trainee Dentist

- Primary Key (Trainee_Dentist_Id)
- Foreign Key (Trainee_Dentist_Id reference Employee)

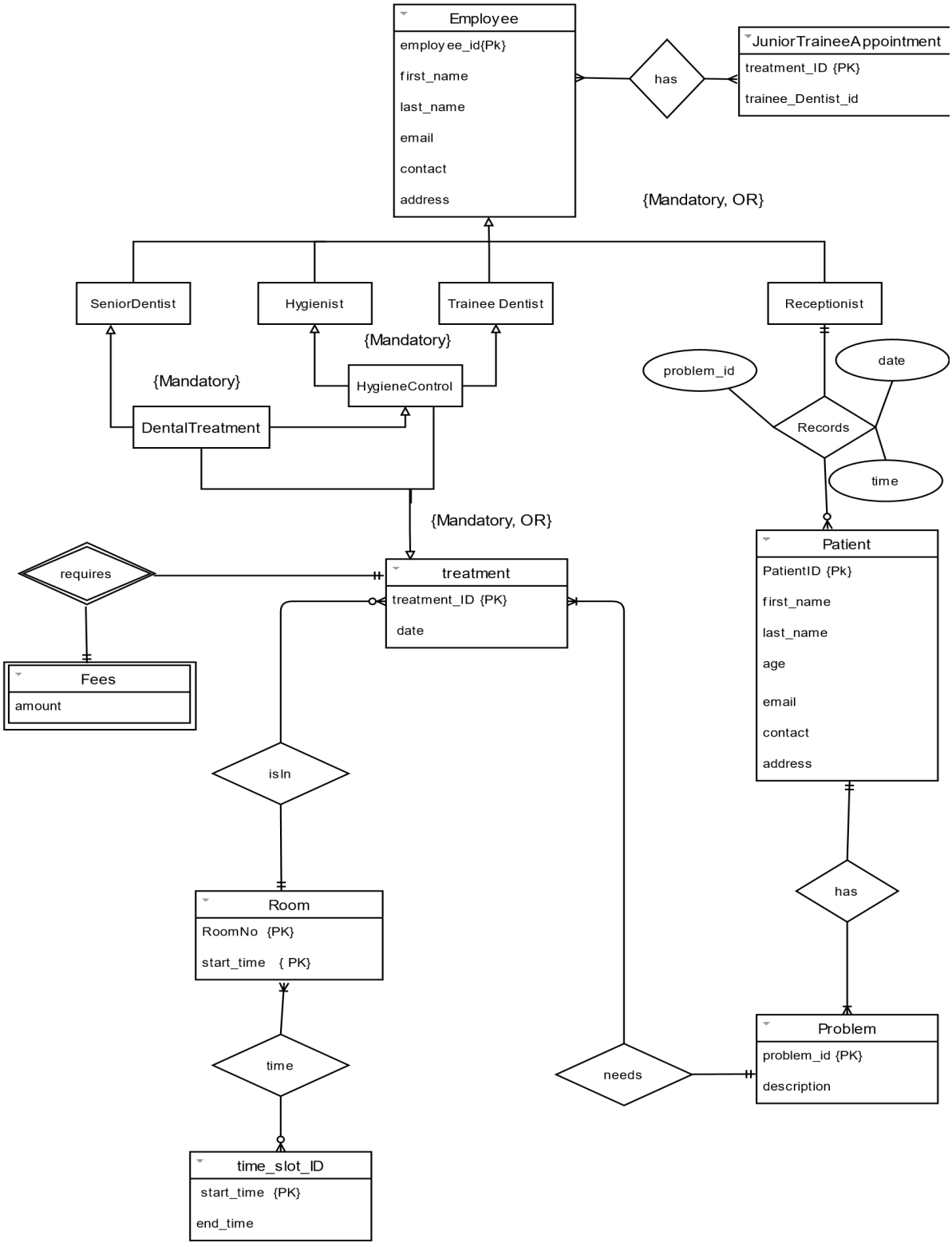
15. Job

- Primary Key (Job_Id)
- Candidate Key (Title)

16. JuniorTraineeAppointment

- Primary Key (Treatment_Id, Trainee_dentist_id)
- Foreign Key (Treatment_Id references Treatment)

d. Entity relationship Diagram of Database DentistX



e. Implementation of Database DentistX

-- 1. create table Employee

```
create table Employee(
    ID int  not null,
    First_name varchar(12) not null,
    Last_name varchar(12) not null,
    Job_Id int not null,
    Email varchar(50) not null,
    Contact varchar(9) not null,
    Address varchar(40) not null);
ALTER TABLE Employee ADD PRIMARY KEY (ID);

--2. create table Dental_treatment
create table Dental_treatment(
    Hygienist_Id int not null,
    Senior_Dentist_Id int not null,
    Treatment_Id int not null);

ALTER TABLE Dental_treatment ADD PRIMARY KEY (Treatment_Id);
ALTER TABLE Dental_treatment
  ADD CONSTRAINT Dental_treatment_ibfk_1 FOREIGN KEY (Hygienist_Id)
REFERENCES
   Hygienist(Hygienist_Id) ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Dental_treatment
  ADD CONSTRAINT Dental_treatment_ibfk_2 FOREIGN KEY (Senior_dentist_Id)
REFERENCES
   SeniorDentist(Senior_dentist_Id) ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Dental_treatment
  ADD CONSTRAINT Dental_treatment_ibfk_4 FOREIGN KEY (Treatment_Id)
REFERENCES
   Treatment(Treatment_Id) ON DELETE NO ACTION ON UPDATE CASCADE;

--3. create table Fees
create table Fees(
    Treatment_Id int not null,
    Amount int not null);

ALTER TABLE Fees ADD PRIMARY KEY (Treatment_Id);
ALTER TABLE Fees
  ADD CONSTRAINT Fees_ibfk_1 FOREIGN KEY (Treatment_Id) REFERENCES
   Treatment(Treatment_Id) ON DELETE NO ACTION ON UPDATE CASCADE;



--4. create table hygiene_control
```

```
create table Hygiene_control(
    Hygienist_Id int not null,
    Treatment_Id int not null);
ALTER TABLE Hygiene_control ADD PRIMARY KEY (Treatment_Id);
ALTER TABLE Hygiene_control
  ADD CONSTRAINT Hygiene_control_ibfk_1 FOREIGN KEY (Treatment_Id)
REFERENCES
   Treatment(Treatment_Id) ON DELETE NO ACTION ON UPDATE CASCADE,
ALTER TABLE Hygiene_control
  ADD CONSTRAINT Hygiene_control_ibfk_3 FOREIGN KEY (Hygienist_Id) REFERENCES
   Hygienist(Hygienist_Id) ON DELETE NO ACTION ON UPDATE CASCADE;

--5. create table Hygienist
create table Hygienist(
    Hygienist_Id int not null);
ALTER TABLE Hygienist ADD PRIMARY KEY (Hygienist_Id);
ALTER TABLE Hygienist
  ADD CONSTRAINT Hygienist_ibfk_1 FOREIGN KEY (Hygienist_Id) REFERENCES
   Employee(ID) ON DELETE NO ACTION ON UPDATE CASCADE;

--6. create table Receptionist
create table Receptionist(
    Receptionist_Id int not null);
ALTER TABLE Receptionist ADD PRIMARY KEY (Receptionist_Id);
ALTER TABLE Receptionist
  ADD CONSTRAINT Receptionist_ibfk_1 FOREIGN KEY (Receptionist_Id) REFERENCES
Employee(ID);

--7. create table Trainee_Dentist
create table TraineeDentist(
    Trainee_dentist_Id int not null);
ALTER TABLE TraineeDentist ADD PRIMARY KEY (Trainee_dentist_Id);
ALTER TABLE TraineeDentist
  ADD CONSTRAINT TraineeDentist_ibfk_1 FOREIGN KEY (Trainee_dentist_Id)
REFERENCES Employee(ID) ON DELETE NO ACTION ON UPDATE CASCADE;

--8. create table SeniorDentist
create table SeniorDentist(
    Senior_dentist_Id int not null);
ALTER TABLE SeniorDentist ADD PRIMARY KEY (Senior_dentist_Id);
ALTER TABLE SeniorDentist
  ADD CONSTRAINT Seniordentist_ibfk_3 FOREIGN KEY (Senior_dentist_Id)
REFERENCES Employee(ID);
```

```
--9. create table Job
create table Job(
    Job_Id int not null,
    Title varchar(20) not null,
    Salary int not null);
ALTER TABLE Job ADD PRIMARY KEY (Job_Id);

--10. create table JuniorTraineeAppointment
create table JuniorTraineeAppointment(
    Treatment_Id int not null,
    Trainee_dentist_Id int not null);
ALTER TABLE JuniorTraineeAppointment ADD PRIMARY KEY
(Treatment_Id,Trainee_dentist_Id);
ALTER TABLE JuniorTraineeAppointment
  ADD CONSTRAINT TraineeAppointment_ibfk_1 FOREIGN KEY (Trainee_dentist_Id)
REFERENCES
  TraineeDentist(Trainee_dentist_Id) ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE JuniorTraineeAppointment
  ADD CONSTRAINT TraineeAppointment_ibfk_2 FOREIGN KEY (Treatment_Id)
REFERENCES
  Treatment(Treatment_Id) ON DELETE NO ACTION ON UPDATE CASCADE;

--11. create table Patient
create table Patient(
    Patient_Id int not null,
    First_name varchar(12) not null,
    Last_name varchar(12) not null,
    Age int not null,
    Email varchar(40) not null,
    Contact varchar(10) not null,
    Address varchar (40) not null);
ALTER TABLE Patient ADD PRIMARY KEY (Patient_Id);

--12. create table Problem
create table Problem(
    Patient_Id int not null,
    Problem_Id int not null,
    Description text not null);
ALTER TABLE Problem ADD PRIMARY KEY (Problem_Id);
ALTER TABLE Problem ADD FOREIGN KEY (Patient_Id) REFERENCES
Patient(Patient_Id);
```

```
--13. create table Records
create table Records(
    Receptionist_Id int not null,
    Patient_Id int not null,
    Date date not null,
    Time time not null,
    Problem_Id int not null);
ALTER TABLE Records ADD PRIMARY KEY (Receptionist_Id, Date, Time);
ALTER TABLE Records
  ADD FOREIGN KEY (Patient_Id) REFERENCES Patient(Patient_Id),
  ADD FOREIGN KEY (Problem_Id) REFERENCES Problem(Problem_Id);
ALTER TABLE Records DROP CONSTRAINT  Records_ibfk_3;
ALTER TABLE Records
 ADD CONSTRAINT Records_ibfk_3 FOREIGN KEY (Receptionist_Id) REFERENCES
   Receptionist(Receptionist_Id) ON DELETE NO ACTION ON UPDATE CASCADE;

--14. create table Room
create table Room(
    Room_Id varchar(1) not null,
    Start_time time not null);
ALTER TABLE Room ADD PRIMARY KEY (Room_Id, Start_time);
ALTER TABLE Room
 ADD CONSTRAINT Room_ibfk_1 FOREIGN KEY (Start_time) REFERENCES
  Time_slot(Start_time) ON DELETE NO ACTION ON UPDATE CASCADE;

--15. create table Time_slot
create table Time_slot(
    Start_time time not null,
    End_time time not null);
ALTER TABLE Time_slot ADD PRIMARY KEY (Start_time);

--16. create table Treatment
create table Treatment(
    Treatment_Id int not null,
    Problem_Id int not null,
    Room_Id varchar(1) not null,
    Date date not null,
    Start_time time not null);

ALTER TABLE Treatment ADD PRIMARY KEY (Treatment_Id);
```

```
ALTER TABLE Treatment

  ADD CONSTRAINT Treatment_ibfk_1 FOREIGN KEY (Problem_Id) REFERENCES
    Problem(Problem_Id) ON DELETE NO ACTION ON UPDATE CASCADE;


ALTER TABLE Treatment
  ADD CONSTRAINT Treatment_ibfk_2 FOREIGN KEY (Room_Id, Start_time)
REFERENCES Room(Room_Id, Start_time);
```

Question 2: Create dummy records for each table

1. Table Employee:

```
INSERT INTO Employee (ID, First_name, Last_name, Email, Job_Id, Contact, Address)
VALUES
(1,'Andy','Arvind','Andy@gmail.com',1,'93376234','Rosenvolt gate 19'),
(2,'Noah','Øystein','noah@gmail.com',1,'46734123','Sognvannveien 21'),
(3,'Clara','Johansen','clara@gmail.com',2,'93412346','Vestli gate 12'),
(4,'Alesandra','Berg','alesandra@yahoo.com',2,'47723978','Rosenhoff 8A'),
(5,'Frederik','Eriksen','fredrik@gmail.com',3,'97656233','Bogstadveien 10'),
(6,'Ditmir','Kraja','ditmir@gmail.com',3,'47612908','Majourstuen gate 20'),
(7,'Julia','Valgen','julia@yahoo.com',4,'92318356','Frogner gata 10'),
(8,'Kiran','Balsen','kiran@gmail.com',4,'43612904','Linderoff 12'),
(9,'Simina','Hansen','simina@gmail.com',4,'46712600','Linderudd 11'),
(10,'Greta','Helgen','greta@yahoo.com',4,'92333456','Torgata 19A'),
(11,'Pia','Johsen','pia@gmail.com',4,'92376458','Bjørvika 12'),
(12,'Philip','Damen','philip@yahoo.com',4,'47799023','Gaustadveien 2'),
(13,'Erik','Helgeland','erik@yhaoo.com',4,'98976543','Kongleveien 23'),
(14,'Sarah','Eriksen','sarah@gmail.com',4,'45621378','Akker brygge 12'),
(15,'Egle','Lindsen','egle@gmail.com',1,'90656123','Troinheimgata 19'),
(16,'Madalina','Ioana','mada@gmail.com',1,'93367256','Grønnland 6B'),
(17,'Johnie','Andersen','johnie@gmail.com',1,'46723905','Krigsjå 4'),
(18,'Anna','Nechita','anna2gmail.com',1,'97834512','Sandvika 33'),
(19,'Helene','Berg','helene@gmail.com',1,'48900234','Kjelsjå 11'),
(20,'Andrea','Hassen','andrea@gmail.com',1,'93067542','Bislett 8A'),
(21,'Camila','Halvag','camila@gmail.com',1,'45689023','Lilestrøms 22'),
(22,'Linea','Hansen','linea@gmail.com',1,'94512789','Problemveien 15'),
(23,'Linn','Johansen','linn@gmail.com',3,'47736345','Slemdal 17'),
(24,'Mira','Devolt','mira@gmail.com',3,'46362309','Sognveien 87'),
(25,'Melodi','Hannes','melodi@gmail.com',3,'97332568','Torgata 36');
```

2. Table Hygiene_control:

```
INSERT INTO Hygiene_control (Hygienist_Id, Treatment_Id) VALUES
```

(7, 1), (8, 2), (9, 3), (12, 4), (11, 5), (13, 6),
(8, 7), (10, 8), (13, 9), (7, 10);

3. Table Dental_treatment:

```
INSERT INTO Dental_treatment (Hygienist_Id, Senior_Dentist_Id, Treatment_Id) VALUES
(9, 15, 11), (8, 22, 12), (14, 22, 13), (10, 15, 14), (7, 16, 15),
(7, 16, 16), (10, 19, 17), (12, 18, 18), (8, 1, 19), (7, 22, 20);
```

4. Table Treatment:

```
INSERT INTO Treatment(Treatment_Id, Problem_Id, Room_Id, Date, Start_time) VALUES
(1, 3, 'A', '2020-12-02', '09:00:00'), (2, 3, 'D', '2020-12-03', '09:00:00'),
(3, 3, 'C', '2020-12-05', '12:00:00'), (4, 4, 'D', '2020-11-02', '15:00:00'),
(5, 5, 'B', '2020-11-23', '09:00:00'), (6, 2, 'B', '2020-11-19', '09:00:00'),
(7, 4, 'B', '2020-12-19', '12:00:00'), (8, 1, 'G', '2020-12-01', '12:00:00'),
(9, 4, 'G', '2020-12-01', '15:00:00'), (10, 3, 'D', '2020-12-12', '09:00:00'),
(11, 5, 'F', '2020-12-12', '15:00:00'), (12, 1, 'C', '2020-12-12', '12:00:00'),
(13, 2, 'A', '2020-11-22', '09:00:00'), (14, 5, 'A', '2020-11-22', '09:00:00'),
(15, 1, 'B', '2020-12-17', '09:00:00'), (16, 1, 'C', '2020-12-18', '12:00:00'),
(17, 5, 'G', '2020-12-13', '12:00:00'), (18, 2, 'G', '2021-01-13', '12:00:00'),
(19, 1, 'B', '2020-12-24', '09:00:00'), (20, 3, 'E', '2020-12-24', '12:00:00');
```

5. Table Fees:

```
INSERT INTO Fees(Treatment_Id, Amount) VALUES
(1, 1000), (2, 500), (3, 1000), (4, 1500), (5, 900), (6, 1800), (7, 900), (9, 1900),
(10, 1300), (11, 2000), (12, 1200), (13, 1300), (14, 1600), (15, 1400), (16, 1300),
(17, 1100), (18, 1000), (19, 800), (20, 900);
```

6. Table Receptionist :

```
INSERT INTO Receptionist (Receptionist_Id) VALUES (3), (4);
```

7. Table Patient :

```
INSERT INTO Patient (Patient_Id, First_name, Last_name, Age, Email, Contact, Address)
VALUES
(1, 'Dennis', 'Høgli', 26, 'denis@gmail.com', '48936693',  'Høgskoleveien 194'),
(2, 'Ronja', 'Svendsen', 40, 'ronja@gmail.com', '43290705', 'Hellerveien 74'),
(3, 'Vilde', 'Kristianse', 44, 'vilde@yahoo.com', '97105393', 'Tjømegaten 225'),
(4, 'Jenny', 'Bredesen', 9, 'jenny@hotmail.com', '43187824', 'Adolph Bergs vei 99'),
(5, 'Tiril', 'Nilsen', 70, 'tiril@gmail.com', '91897693', 'Balders vei 215');
```

8. Table Problem:

```
INSERT INTO Problem (Patient_Id, Problem_Id, Description) VALUES
```

```
(1, 1, 'Bad Breath'),
(2, 2, 'Tooth Decay'),
(3, 3, 'Gum (Periodontal) Disease'),
(4, 4, 'Mouth Sore, Bad Breath'),
(5, 5, 'Toothaches and Dental Emergencies');
```

9. Table Room:

```
INSERT INTO Room (Room_Id, Start_time) VALUES
('A', '09:00:00'), ('A', '12:00:00'), ('A', '15:00:00'),
('B', '09:00:00'), ('B', '12:00:00'), ('B', '15:00:00'),
('C', '09:00:00'), ('C', '12:00:00'), ('C', '15:00:00'),
('D', '09:00:00'), ('D', '12:00:00'), ('D', '15:00:00'),
('E', '09:00:00'), ('E', '12:00:00'), ('E', '15:00:00'),
('F', '09:00:00'), ('F', '12:00:00'), ('F', '15:00:00'),
('G', '09:00:00'), ('G', '12:00:00'), ('G', '15:00:00');
```

10. Table Time_slot:

```
INSERT INTO Time_slot (Start_time, End_time) VALUES
('09:00:00', '11:00:00'), ('12:00:00', '14:00:00'), ('15:00:00', '17:00:00');
```

11. Table Job:

```
INSERT INTO Job (Job_Id, Title, Salary) VALUES
(1, 'Senior_Dentist', 100000),  (2, 'Receptionist', 40000),
(3, 'Trainee dentist', 60000),  (4, 'Hygienist', 70000);
```

12. Table Hygienist:

```
INSERT INTO Hygienist (Hygienist_Id) VALUES
(7), (8), (9), (10), (11), (12), (13), (14);
```

13. Table SeniorDentist:

```
INSERT INTO SeniorDentist (Senior_dentist_Id) VALUES
(1), (2), (15), (16), (17), (18), (19), (20), (21), (22);
```

14. Table TraineeDentist:

```
INSERT INTO TraineeDentist (Trainee_dentist_Id) VALUES
(5), (6), (23), (24), (25);
```

15. Table Records:

```
INSERT INTO Records (Receptionist_Id, Patient_Id, Date, Time, Problem_Id) VALUES
```

```
(3, 3, '2020-12-02', '06:00:00', 3), (3, 5, '2020-12-03', '08:00:00', 5),
(3, 3, '2020-12-04', '10:00:00', 3), (3, 3, '2020-12-05', '03:00:00', 3),
(4, 4, '2020-12-01', '07:00:00', 4);
```

16. Table JuniorTraineeAppointment:

```
INSERT INTO JuniorTraineeAppointment (Treatment_Id, Trainee_dentist_Id) VALUES
(1, 5), (1, 6), (1, 24), (2, 6), (2, 23), (5, 6),
(6, 23), (7, 5), (7, 24), (17, 24);
```

<u>Question 3:</u> SQL Queries to retrieve data from Database DentistX

a. Which Dentist treated the highest number of patients?

```
SELECT * FROM Employee WHERE ID IN (SELECT MAX(maxTreatment)
        FROM (SELECT Senior_Dentist_Id, COUNT(*) AS
        maxTreatment FROM Dental_treatment
        GROUP BY Dental_treatment.Senior_Dentist_Id) AS Result);
```

b. List number of appointments per month in order of the date and time they occurred.

```
SELECT * FROM Treatment ORDER BY Date, Start_time;
```

c. Retrieve Patient details whose treatment(s) spanned over more than 3 appointments for each treatment.

```
SELECT * FROM Patient WHERE Patient_Id IN (SELECT Patient_Id
FROM Problem WHERE Problem_Id IN (SELECT Problem_Id
FROM Treatment GROUP BY Problem_Id HAVING COUNT(Treatment_Id)>3));
```

d. Retrieve list of appointments where more than one junior/trainee dentists were assigned.

```
SELECT * FROM Treatment WHERE Treatment_Id IN (SELECT Treatment_Id
FROM JuniorTraineeAppointment GROUP BY Treatment_Id
HAVING COUNT(Trainee_dentist_Id)>1);
```

e. List number of treatments performed in each room.

```
SELECT Room_Id, COUNT (Treatment_Id) AS
"Number of treatments" FROM Treatment GROUP BY Room_Id;
```

f. Retrieve list of patients whose age is more than 40 years.

```
SELECT * FROM Patient WHERE age>40;
```

g. Calculate and present Total Hours used on each Patient in the database

```
SELECT Problem_Id, COUNT(*)*2 AS Totaltime FROM Treatment GROUP BY Problem_Id;
```

<u>Question 4</u>: Write a paragraph explaining your database design choices and explain why you think your database tables are in good normalized forms:

The database “DentistX” is a relational database that is designed in three steps. First, its ERD was constructed according to given requirements. Then it was reduced into relational schema and finally it was normalized using normalization steps. Any database is said to be in good normalized form when each table has its own primary key and it is free from partial and transitive dependency. It means when in any table a non- primary key attribute is not dependent on a part of a primary key and when any non-primary key attribute is not transitively dependent on a primary key then a database is said to be in free from partial dependency and transitive dependency respectively. The database “DentistX” is in definitely in normalized form because each table has its own primary key and no any non-primary key attribute is partially dependent on a primary key and similarly no any non-primary key attribute is transitively dependent on any non-primary key attribute making the designed database in good normalized form.

<u>Question 5</u>: Write a paragraph explaining your understanding of concurrency and isolation in Relational Databases:

Concurrency is one of the properties of a relational database called ACID properties (Atomicity, Consistency, Isolation, Durability) Concurrency in a database means many users can view the same database at the same time in a consistent form. A database is said to be consistent, if multiple users see the same view of data. For example, if two receptionists are accessing the same database at the same time for recording patient's information and making an appointment for the same doctor with two different patients on the same day and doctor has only one slot empty on that day. Now assume receptionist A fixed appointment for patient A and change was saved in the database but not reflected on reception B's screen or view of database then receptionist B would also fix appointments for patient B making database inconsistent. Therefore, for database must be concurrent if it allows multiple transactions at the same time else it would make database inconsistent which could be cause of loss of millions of dollars for a business. Isolation is another ACID property of relational database that is more related to querying process. For example, when a database allows multiple transactions to multiple users at the same time, and at 9:00 AM user A wants to see records of all patients in a database consisting of 20000 patients. Since, database is quite big it might take at least 3 minutes to show results. Then at 9:01 AM another patient is about to add to the table but change might not be committed. Now question arises if last record added should be displayed to the user or not? This has 4 possibilities called isolation levels. First, user is able to see change but problem arises when record to be added is not committed making results inconsistent called dirty read. Second, user is not able to see change since it was not committed before query was started. Third, user doesn't see change because change was not committed before transaction was started and last change is displayed in series of changes committed called serializable.

<u>Question 6</u>: Would it be a good idea to use NoSQL database instead of Relational Database for DentistX ?

No doubt relational database helps to store data efficiently, but use of NoSQL would have been the best choice to use for a number of reasons. First, it subtracts need of designing schema beforehand and allows to work with unstructured data. Second, NoSQL database can have different formats like documents format, graph format or even table format that helps to read data more easily, estimate current state of the business and even make predictions about future trends. Third, NoSQL allows to store single database on multiple servers in form of parts reducing need of purchasing expensive servers beforehand. In a nutshell, using NoSQL database helps to store more complex and unstructured data efficiently and it is cheaper to implement than relational databases.

When database is not required to design database schema beforehand, it reduces implementation costs. Because in relational database if we make any little mistake in database design and implement it before the mistake is caught then its correction cost could be higher than its implementation cost. In addition to this if user wants some modifications in the existing database, there is high rate of possibility that database require to implement it once again.

For example, while designing database "DentistX", I neglected possibility of assigning more than one trainee dentists in single appointment and found it while writing queries. In that case, I had to re-implement my database that wasted my lots of time. Mistakes can be even bigger while database design. But, this mistake could not have risen if I had used NoSQL database since it doesn't require to design schema beforehand.

Besides, as mind processes visual data more easily, graphs based would help to analyze each doctor’s progress, or check which room is usually busier than others would have been quicker than reading it in table format. In addition to this, what if clinic wants to expand its branches in the whole country making data more complex and wants modifications in its relational database, then not only database needs to be modified or even re-designed but also it might require to purchase even expensive server make existing server useless, since relational database requires to store all data on a single server, but NoSQL can be cheaper here because it can store single data on multiple servers. Therefore, from all aspects it's much better to use NoSQL database because it's cheaper, easy to implement and has capability to accommodate change.

------- End-------