10015

PGR111

Databases

Individual written home exam

Høyskolen Kristiania

FALL 2020

1.a. Identify the entities (also known as relations) that may be required for designing relational database for DentistX.

Addresses, Patients, Employees, Salary, Rooms, Cases, Appointments, Payment, TreatmentInfo.

1.b. Identify and list attributes associated with the relations identified in a.

Addresses(AddressID(PK), StreetName, StreetNumber, City, PostCode)

Patients(PatientID(PK), PFirstName, PLastName, PMobileNo, PDOB, PSex, PEmail, PPersonalID, AddressID(FK))

Employees(EmployeeID(PK), EFirstName, ELastName, EMobileNo, EDOB, ESex, EEmail, EPosition, EPersonalID, AddressID(FK))

Salary(SalaryID(PK), MonthlySalary, EmployeeID(FK))

Rooms(RoomID(PK), RoomName)

Cases(CaseNo(PK), PatientID(FK), CaseOwnerID(FK))

Appointments(ApptNo(PK), PatientID(FK), ApptStartTime, ApptEndTime, CaseOwnerID, SecondaryDentistID, HygienistID, RoomID(FK))

Payment(PaymentNo(PK), PatientID(FK), PaymentTime, PaymentAmount, PaymentMethod)

TreatmentInfo(TreatmentID(PK), ApptNo(FK), TreatmentType, Price)

1.c. Identify Candidate keys, Primary Keys and Foreign keys.

Only AddressID attribute can be primary key in **Addresses** table. There is no other candidate key or foreign key in this table.

PatientID is the primary key of **Patients** table. PMobileNo, PEmail, PPersonalID are candidate keys because these attributes have unique values. AddressID is used as foreign key in this table.

EmployeeID is the primary key of **Employees** table. Besides EMobileNo, EEmail, EPersonalID have unique values and they are candidate keys in this table. AddressID is used as foreign key here again.

SalaryID is primary key and **Salary** table has no candidate key. EmployeeID is foreign key in this table.

RoomID is primary key in **Rooms** table. RoomName is candidate key because room names are unique.

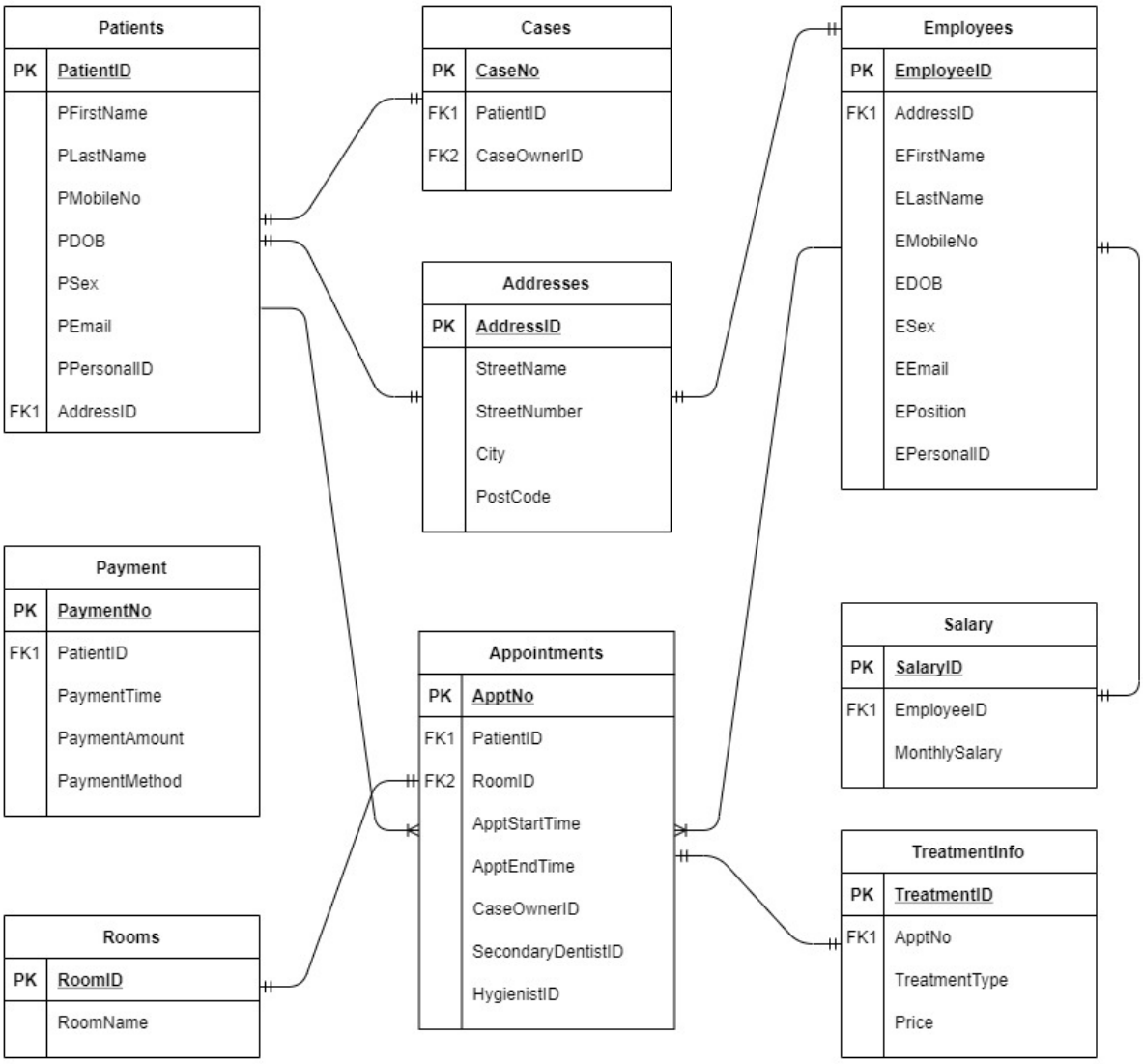
CaseNo is primary key in **Cases** table with no candidate key. PatientID and CaseOwnerID are foreign keys in this table.

ApptNo is primary key in **Appointments** table. The other columns have same values and there is no candidate key in this table. PatientID and RoomID are foreign keys in this table.

PaymentNo is the primary key of **Payment** table. The other columns can have common values in tuples and there is no candidate key. PatientID is used as foreign key in this table.

TreatmentID is selected as primary key in **TreatmentInfo** table and the other attributes may have same values in tuples. Therefore, there is no candidate key in this table. ApptNo is used as foreign key here.

1.d. Design a high-level ER model and define relations between entities.



3.a. Which Dentist treated the highest number of patients?

```
select * from Employees, (
select CaseOwnerID as ID,COUNT(*)
as Total_Count from
(select CaseOwnerID from Appointments
  UNION ALL
  select SecondaryDentistID from Appointments
  WHERE  SecondaryDentistID NOT IN (SELECT SecondaryDentistID FROM Appointments WHERE
SecondaryDentistID = 'NOTASSIGNED')
```

```
)Appointments
group by CaseOwnerID having count(*) =
(select max(t1.Total_Count) as Total_Count
from (
select CaseOwnerID,COUNT(*)
as Total_Count from
(select CaseOwnerID from Appointments
  UNION ALL
  select SecondaryDentistID from Appointments
  WHERE  SecondaryDentistID NOT IN (SELECT SecondaryDentistID FROM Appointments WHERE
SecondaryDentistID = 'NOTASSIGNED')
) Appointments
group by CaseOwnerID
) t1
) ) t2
where EmployeeID = t2.ID;
```

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

```
select YEAR(ApptStartTime) as "Year",MONTH(ApptStartTime) as "Month",count(ApptNo) as
"no_of_appointments"
from Appointments group by YEAR(ApptStartTime),MONTH(ApptStartTime)
order by YEAR(ApptStartTime);
```

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

```
select * from Patients where PatientID in
(select Appointments.PatientID from Appointments group by Appointments.PatientID
having count(Appointments.ApptNo) > 3);
```

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

```
select ApptNo

from Appointments where CaseOwnerID like 'TR%' and SecondaryDentistID like 'TR%';
```

3. e. List number of treatments performed in each room.

```
select RoomID,count(ApptNO) as "No_of_appointments"

from Appointments group by RoomID;
```

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

```
select * from Patients

where DATEDIFF(YEAR,DOB,GETDATE()) > 40;
```

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

```
select PatientID, FirstName, LastName, t1.No_of_Hours from Patients

inner join (

select PatientID as pid,sum(DATEDIFF(hour, ApptStartTime, ApptEndTime)) as No_of_Hours

from Appointments  group by PatientID) t1 on Patients.PatientID = t1.pid ;
```

4. Write a paragraph explaining your database design choices and explain why you think your database tables are in good normalized forms.

I believe I have created a good database because values stored in an attribute column are in the same domain and all the columns in tables have unique names. My database satisfies 3NF rules either. These are:

- Attribute values are single atomic values.
- Every nonprime attribute A in R must be fully functionally dependent on the primary key of R.
- No nonprime attribute of R is transitively dependent on the primary key.

I created a database which can be functional for a dentist. Tables are just enough to store every information which a dentist need. Table names and attribute names are very clear and understandable. Users can easily understand where to store data. The tables do not have many columns like lower normal forms. Applying SQL queries was a good test either and it shows that the database if fully functional and working. According to Microsoft a good database design is one that: divides your information into subject-based tables, provides access to join data, support and ensure the accuracy and integrity of information, and accommodates your data processing and reporting needs. I believe my database satisfies these features too.

5. Write a paragraph explaining your understanding of concurrency and isolation in Relational Databases.

A database's allowance ability of multiple users to affect multiple transactions is defined as database concurrency. This is one of the things which makes a storage 'database'. "Concurrency can be explained as when one user is changing data but has not yet saved that data, the database should not allow other users to query the same, unsaved data" (technopedia.com). If the transaction is not finished yet, the other users should only see the original data before transaction was started. The unsaved data is held in a temporary log file whether it has changed or not. After saving of data, it will be written to physical database storage. The lost update, the temporary update, the incorrect summary, and the unrepeatable read are common concurrency problems (Elmasri and Navathe, Database Systems, Seventh Edition).

Isolation is one of the four concurrency control protocols (Atomicity, Consistency, Isolation and Durability) and it controls how and when changes are made. Isolation aims to allow several transactions at the same time without affecting each other. Isolation levels are defined by transactions and these levels define which transaction must be isolated from data modifications made by other transactions. Following phenomena defines transaction isolation levels: (1) dirty read -reading data that has not been committed yet-, (2) non-repeatable read -reading the same row twice, and get a different value each time-, and (3) phantom read –reading the same data twice but acquiring different values- (geeksforgeeks.com). 'Transaction isolation levels control four things: (1) are locks taken when data is read? (2) what type of locks are requested, (3) how

long the read locks are held, and (4) whether a read operation referencing rows modified by another transaction'(docs.microsoft.com).

6. Would It be a good idea to use NoSQL database instead of Relational Database for DentistX? Please provide your explanation using one or two paragraphs. You can also use examples as argument to prove your opinion.

The relational model represents the database as a collection of relations which each relation resembles a table of values or a flat file of records. Non-relational databases (or No SQL databases) store their data in a non-tabular form. NoSQL databases might be based on data structures like documents.

There is a bunch of criteria which a database administrator/user should NoSQL or a relational database. These criteria are:

Consider a NoSQL datastore when:	**Consider a relational database when:**
You have high volume workloads that require large scale	Your workload volume is consistent and requires medium to large scale
Your workloads don't require ACID guarantees	ACID guarantees are required
Your data is dynamic and frequently changes	Your data is predictable and highly structured
Data can be expressed without relationships	Data is best expressed relationally
You need fast writes and write safety isn't critical	Write safety is a requirement
Data retrieval is simple and tends to be flat	You work with complex queries and reports
Your data requires a wide geographic distribution	Your users are more centralized
Your application will be deployed to commodity hardware, such as with public clouds	Your application will be deployed to large, high-end hardware

(Source: docs.microsoft.com)

I consider workload of a dentist is consistent and scale is medium to large. It cannot be considered as large scale or big data. It is highly predictable and highly structured. Attributes of tables are atomic and in a certain domain like names, mobile numbers and email. There are no open data

fields in the database where users can write whatever they want. Data is best expressed by relationality because of relations between tables and attributes. Tables are connected to each other with keys by using relationality users can reach every information stored in the database. Write safety is a requirement because the database is only suitable for relevant information. A tuple can allow to write irrelevant information, but this makes queries harder or inconclusive. Additionally, complex queries and reports may be needed by administration and relational databases are best such as I developed. The thing we needed to develop a relational database is that I deployed the database to a large hardware, a computer. Actually, it is a relative issue, but I think a computer should be considered a high-end hardware because of its complexity. The only issue we can discuss usefulness of NoSQL database is that a dentist workload doesn't require ACID guarantees. There is no money transaction on this database and mistakes can be corrected later on without any harm. According to these criteria it is logical to use a relational database as a dentist database.

DDL QUERIES

```
Create Table Addresses(
AddressID varchar(10) not null,
StreetName varchar(100) not null,
StreetNumber int not null,
City varchar(30) not null,
PostCode varchar(4) not null,
Primary Key (AddressID)
)
Create Table Patients(
PatientID varchar(10) not null,
PFirstName varchar(25) not null,
PLastName varchar(25) not null,
PMobileNo bigint not null,
PDOB date,
PSex varchar(6),
```

```
PEmail varchar(50),
PPersonalID bigint not null,
AddressID varchar(10),
Primary Key (PatientID),
Foreign Key (AddressID) references Addresses(AddressID)
)
Create Table Employees(
EmployeeID varchar(11) not null,
EmpFirstName varchar(25) not null,
ELastName varchar(25) not null,
EMobileNo bigint not null,
EDOB date,
ESex varchar(6),
EPosition varchar(30) not null,
EEmail varchar(50),
EPersonalID bigint not null,
AddressID varchar(10),
Primary Key (EmployeeID),
Foreign Key (AddressID) references Addresses(AddressID)
)
Create Table Salary(
SalaryID varchar(11) not null,
MonthlySalary bigint not null,
EmployeeID varchar(11) not null,
Primary Key (SalaryID),
Foreign Key (EmployeeID) references Employees(EmployeeID)
)
Create Table Rooms(
RoomID varchar(10) not null,
```

```
RoomName varchar(25) not null,
Primary Key (RoomID),
)
Create Table Cases(
CaseNo int not null,
PatientID varchar(10) not null,
CaseOwnerID varchar(11) not null,
Primary Key (CaseNo),
Foreign Key (PatientID) references Patients(PatientID),
Foreign Key (CaseOwnerID) references Employees(EmployeeID)
)
Create Table Appointments(
ApptNo bigint not null,
PatientID varchar(10) not null,
ApptStartTime datetime not null,
ApptEndTime datetime,
CaseOwnerID varchar(11) DEFAULT 'NOTASSIGNED',
SecondaryDentistID varchar(11) DEFAULT 'NOTASSIGNED',
HygienistID varchar(11) DEFAULT 'NOTASSIGNED',
RoomID varchar(10) not null,
Primary Key (ApptNo),
Foreign Key (PatientID) references Patients(PatientID),
Foreign Key (RoomID) references Rooms(RoomID),
)
Create Table Payment(
PaymentNo bigint not null,
PatientID varchar(10) not null,
PaymentTime datetime not null,
PaymentAmount bigint not null,
```

PaymentMethod varchar(20),

Primary Key (PaymentNo),

Foreign Key (PatientID) references Patients(PatientID),

)

Create Table TreatmentInfo(

TreatmentID varchar(10) not null,

ApptNo bigint not null,

TreatmentType varchar(100),

Price bigint not null,

Primary Key (TreatmentID),

Foreign Key (ApptNo) references Appointments(ApptNo),

)

DML QUERIES

INSERT INTO Addresses VALUES (‘AD1001’, ‘HOLTEGATA’,’25’,’OSLO’,’0256’);

INSERT INTO Addresses VALUES (‘AD1002’, ‘PROFESSOR DAHLS GATE’,’56’,’OSLO’,’0412’);

INSERT INTO Addresses VALUES (‘AD1003’, ‘DAAS GATE’,’10’,’OSLO’,’0785’);

INSERT INTO Addresses VALUES (‘AD1004’, ‘ECKERSBERGS GATE’,’14’,’OSLO’,’0985’);

INSERT INTO Addresses VALUES (‘AD1005’, ‘TIDEMANDS GATE’,’86’,’OSLO’,’0745’);

INSERT INTO Addresses VALUES (‘AD1006’, ‘GULDBERGLIA’,’48’,’OSLO’,’0356’);

INSERT INTO Addresses VALUES (‘AD1007’, ‘BEKKELIVVEIEN’,’85’,’OSLO’,’0365’);

INSERT INTO Addresses VALUES (‘AD1008’, ‘KONVENTVEIEN’,’72’,’OSLO’,’0123’);

INSERT INTO Addresses VALUES (‘AD1009’, ‘KVERNVEIEN’,’16’,’BAERUM’,’1256’);

INSERT INTO Addresses VALUES (‘AD1010’, ‘BANKVEIEN’,’23’,’ASKER’,’1347’);

INSERT INTO Appointments VALUES (‘12458’,’PA0883’,’2010-12-24- 09:00:00.000’, ’2010-12-24- 11:00:00.000’,’RO001’,’SE1013’,’NOTASSIGNED’,’NOTASSIGNED’)

INSERT INTO Appointments VALUES (‘12459’,’PA0989’,’2011-11-25- 12:00:00.000’, ’2011-11-25- 14:00:00.000’,’RO002’,’SE1020’,’NOTASSIGNED’,’NOTASSIGNED’)

INSERT INTO Appointments VALUES (‘12460’,’PA0989’,’2011-12-15- 09:00:00.000’, ’2011-12-15- 11:00:00.000’,’RO004’,’SE1020’,’NOTASSIGNED’,’HY1007’)

INSERT INTO Appointments VALUES (‘12461’,’PA1001’,’2012-01-21- 12:00:00.000’, ’2012-01-21- 14:00:00.000’,’RO003’,’TR1010’,’SE1015’,’NOTASSIGNED’)

INSERT INTO Appointments VALUES (‘12462’,’PA0883’,’2012-03-13- 09:00:00.000’, ’2012-03-13- 11:00:00.000’,’RO003’,’SE1013’,’NOTASSIGNED’,’HY1005’)

INSERT INTO Appointments VALUES (‘12463’,’PA1015’,’2012-04-14- 12:00:00.000’, ’2012-04-14- 14:00:00.000’,’RO001’,’SE1013’,’TR1007’,’HY1007’)

INSERT INTO Appointments VALUES (‘12464’,’PA1015’,’2012-05-09- 09:00:00.000’, ’2012-05-09- 11:00:00.000’,’RO002’,’TR1002’,’TR1007’,’NOTASSIGNED’)

INSERT INTO Appointments VALUES (‘12465’,’PA0883’,’2012-06-18- 12:00:00.000’, ’2012-06-18- 14:00:00.000’,’RO006’,’SE1013’,’NOTASSIGNED’,’NOTASSIGNED’)

INSERT INTO Appointments VALUES (‘12466’,’PA1001’,’2012-07-12- 09:00:00.000’, ’2012-07-12- 11:00:00.000’,’RO002’,’SE1018’,’NOTASSIGNED’,’HY1012’)

INSERT INTO Appointments VALUES (‘12467’,’PA1029’,’2012-08-28- 15:00:00.000’, ’2012-08-28- 17:00:00.000’,’RO007’,’TR1010’,’SE1009’,’HY1005’)

INSERT INTO Cases VALUES (‘1001’,’PA0883’,’SE1009’)

INSERT INTO Cases VALUES (‘1002’,’PA1015’,’SE1013’)

INSERT INTO Cases VALUES (‘1003’,’PA1025’,’SE1020’)

INSERT INTO Cases VALUES (‘1004’,’PA1026’,’TR1004’)

INSERT INTO Cases VALUES (‘1005’,’PA1029’,’TR1007’)

INSERT INTO Cases VALUES (‘1006’,’PA0989’,’SE1020’)

INSERT INTO Cases VALUES (‘1007’,’PA1001’,’TR1010’)

INSERT INTO Cases VALUES (‘1008’,’PA1018’,’SE1009’)

INSERT INTO Cases VALUES (‘1009’,’PA1032’,’TR1004’)

INSERT INTO Cases VALUES (‘1010’,’PA1033’,’SE1018’)

INSERT INTO Employees VALUES (‘RE1001’,’SCARLETT’,’JOHANSON’,’23561486’,’1984-02-24’,’FEMALE’,’RECEPTIONIST’,’SCARLETT@GMAIL.COM’,’AD1005’,’56842197598’)

INSERT INTO Employees VALUES (‘RE1002’,’JENIFER’,’LOPEZ’,’45632897’,’1979-04-18’,’FEMALE’,’RECEPTIONIST’,’JENIFER15@GMAIL.COM’,’AD1002’,’85476125349’)

INSERT INTO Employees VALUES ('HY1001','LARA','CROFT','56812432','1984-03-19','FEMALE','HYGIENIST','LARA@HOTMAIL.COM','AD1001','98564781254')

INSERT INTO Employees VALUES ('HY1006','JESSICA','BIEL','14725436','1978-11-01','FEMALE','HYGIENIST','JESSICA05@GMAIL.COM','AD1004','75489846254')

INSERT INTO Employees VALUES ('SE1003','GANDALF','THEGRAY','25413612','1825-11-24','MALE','SENIOR DENTIST','GANDALF@YAHOO.COM','AD1003','25814635789')

INSERT INTO Employees VALUES ('SE1009','TONY','STARK','89759647',1975-11-08','MALE','SENIOR DENTIST','TONY32@GMAIL.COM','AD1005','78945625468')

INSERT INTO Employees VALUES ('SE1013','BRUCE','WAYNE','65498732',1976-06-30','MALE','SENIOR DENTIST','BRUCE001@GMAIL.COM','AD1009','85296314736')

INSERT INTO Employees VALUES ('SE1015','JACK','SPARROW','41752869',1967-09-11','MALE','SENIOR DENTIST','JACK@GMAIL.COM','AD1002','71482593619')

INSERT INTO Employees VALUES ('TR1002','FRODO','BAGGINS','99825536','1977-01-10','MALE','TRAINEE DENTIST','FRODO@HOTMAIL.COM','AD1001','86124935766')

INSERT INTO Employees VALUES ('TR1011','PEREGRIN','TOK','31796541',1978-12-31','MALE','TRAINEE DENTIST','PEREGRIN@GMAIL.COM','AD1004','15948263468')

INSERT INTO Patients VALUES ('PA0883','ARNOLD','SCHWARS','25648965','1944-01-27','MALE','ARNOLD@GMAIL.COM','AD1005',' 58255351987')

INSERT INTO Patients VALUES ('PA0989','TARJEI','BOE','32658456','1956-11-25','MALE','TARJEI22@GMAIL.COM','AD1003','65231584756')

INSERT INTO Patients VALUES ('PA1001','JOHANNES','BOE','25486234','1991-04-24','MALE','JOHANNES74@GMAIL.COM','AD1003','54865923564')

INSERT INTO Patients VALUES ('PA1015','HEIDI','WENG','87986535','1991-11-19','FEMALE','HEIDI85@GMAIL.COM','AD1005','17283982937')

INSERT INTO Patients VALUES ('PA1018','INGVILD',' OSTBERG','78945612','1993-07-04','FEMALE','INGVILD@LIVE.COM','AD1005','74859658694')

INSERT INTO Patients VALUES ('PA1025','DOROTHEA','WIERER','68951524','1994-09-22','FEMALE','DOROTHEA@GMAIL.COM','AD1007','85174296326')

INSERT INTO Patients VALUES (‘PA1026’,’DENISE’,’HERMANN’,’77854215’,’1993-05-25’,’FEMALE’,’DENISE08@GMAIL.COM’,’AD1007’,’76431982645’)

INSERT INTO Patients VALUES (‘PA1029’,’ERIK’,’LESER’,’98567319’,’1990-11-28’,’MALE’,’ERIK14@LIVE.COM’,’AD1008’,’29272321687’)

INSERT INTO Patients VALUES (‘PA1032’,’JOHANNES’,’KLAEBO’,’36325214’,’1997-02-17’,’MALE’,’JOHANNES253@GMAIL.COM’,’AD1008’,’48192676588’)

INSERT INTO Patients VALUES (‘PA1033’,’EMIL’,’IVERSEN’,’45846921’,’1995-04-18’,’MALE’,’EMIL47@GMAIL.COM’,’AD1008’,’89547632298’)

INSERT INTO Payment VALUES (‘1023’,’PA0883’,’2015-06-24 11:25:00.000’,’26500’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1024’,’PA0989’,’2016-11-22 09:48:00.000’,’18900’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1025’,’PA1018’,’2018-04-13 15:57:00.000’,’9800’,’CASH’)

INSERT INTO Payment VALUES (‘1026’,’PA1026’,’2018-12-28 14:36:00.000’,’17000’,’PAY CHECK’)

INSERT INTO Payment VALUES (‘1027’,’PA1029’,’2019-07-24 10:15:00.000’,’25000’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1028’,’PA1015’,’2019-09-21 12:15:00.000’,’14000’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1029’,’PA1018’,’2019-11-17 15:16:00.000’,’26000’,’PAY CHECK’)

INSERT INTO Payment VALUES (‘1030’,’PA1001’,’2020-01-13 10:47:00.000’,’24000’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1031’,’PA1025’,’2020-03-05 11:18:00.000’,’9800’,’CREDIT CARD’)

INSERT INTO Payment VALUES (‘1032’,’PA1032’,’2020-05-29 16:18:00.000’,’15000’,’CREDIT CARD’)

INSERT INTO Rooms VALUES (‘RO001’,’ROOM A’)

INSERT INTO Rooms VALUES (‘RO002’,’ROOM B’)

INSERT INTO Rooms VALUES (‘RO003’,’ROOM C’)

INSERT INTO Rooms VALUES (‘RO004’,’ROOM D’)

INSERT INTO Rooms VALUES (‘RO005’,’ROOM E’)

INSERT INTO Rooms VALUES (‘RO006’,’ROOM F’)

INSERT INTO Rooms VALUES (‘RO007’,’ROOM G’)

INSERT INTO Salary VALUES (‘SA1001’, ‘40000’,’RE1001’)

INSERT INTO Salary VALUES (‘SA1002’, ‘41500’,’RE1002’)

INSERT INTO Salary VALUES (‘SA1003’,’48000’,’HY1001’)

INSERT INTO Salary VALUES (‘SA1004’,’43250’,’HY1006’)

INSERT INTO Salary VALUES (‘SA1005’,’95000’,’SE1003’)

INSERT INTO Salary VALUES (‘SA1006’,’78500’,’SE1009’)

INSERT INTO Salary VALUES (‘SA1007’,’76000’,’SE1013’)

INSERT INTO Salary VALUES (‘SA1008’,’82000’,’SE1015’)

INSERT INTO Salary VALUES (‘SA1009’,’51000’,’TR1002’)

INSERT INTO Salary VALUES (‘SA1010’,’50000’,’TR1011’)

INSERT INTO TreatmentInfo VALUES (‘10001’,’12458’,’Consultation’,’1500’)

INSERT INTO TreatmentInfo VALUES (‘10002’,’12459’,’Surgery’,’6500’)

INSERT INTO TreatmentInfo VALUES (‘10003’,’12460’,’Implant’,’15000’)

INSERT INTO TreatmentInfo VALUES (‘10004’,’12461’,’Hygienic Control’,’2500’)

INSERT INTO TreatmentInfo VALUES (‘10005’,’12462’,’Pull out’,’4000’)

INSERT INTO TreatmentInfo VALUES (‘10006’,’12463’,’Consultation’,’1500’)

INSERT INTO TreatmentInfo VALUES (‘10007’,’12464’,’Surgery’,’6500’)

INSERT INTO TreatmentInfo VALUES (‘10008‘,’12465’,’Implant’,’15000’)

INSERT INTO TreatmentInfo VALUES (‘10009’,’12466’,’Pull out’,’4000’)

INSERT INTO TreatmentInfo VALUES (‘10010’,’12467’,’Consultation’,’1500’)