Database Design Term Project (CS 6360)

Shruti Jaiswal (SXJ170027), Trung Hieu Tran (TXT171930), Prateek Sarna (PXS180012), Vineet Vats (VXV180008)

1. Problem Description

Dallas Care is a hospital and medical care center. Dallas Care would like one relational database to be able to smoothly carry out their work in an organized way. The hospital has following modules: Person, Employee, Patient, Visitors, Pharmacy, Treatment, Rooms, Records and Medical Bill Payment. A Person can be an Employee or a Class 1 Patient. Details of a person such as Person ID, Name (First, Middle, Last), Address, Gender, Date of Birth, and Phone number (one person can have more than one phone number) are recorded. A person ID should be in the format, 'PXXX', where XXX can be a value between 100 and 999. A Class 1 patient is a person who visits the hospital just for a doctor consultation. A person can be both an employee and a Class 1 patient. Employee is further classified as Doctors, Nurses or Receptionists. The start date of the employee is recorded. The specialization of the doctor is stored and doctors are further classified into Trainee, Permanent or Visiting. Every Class 1 patient consults a doctor. A Class 1 patient can consult at most one doctor but one doctor can be consulted by more than one Class 1 patient. A Class 2 patient is a someone who is admitted into the hospital. A Class 2 patient can be an Employee or a Class 1 Patient or both. A doctor attends Class 2 patients. One doctor can attend many Class 2 patients but a Class 2 patient can be attended to by at most 2 doctors. The date of patient being admitted into the hospital is recorded. A Visitor log is maintained for the Class2 Patients, which stores information such as patient ID, visitor ID, visitor name, visitor's address, and visitor's contact information. Pharmacy details such as Medicine code, Name, Price, Quantity and Date of expiration is recorded. The database also stores the information of the various kinds of treatments that are offered in the hospital. The treatment details such as ID, name, duration and associated medicines are recorded. When a treatment is assigned to a Class 2 patient, the treatment details, medicine details and patient details are recorded so that the doctor can easily access this information. Nurses governs rooms. Each nurse can govern more than one room, but each room has only one nurse assigned to it. The room details such as room ID, room type and duration is recorded. Each Class 2 patient is assigned a room on being admitted to the hospital. A records database is maintained by the receptionist who keeps record of information such as record ID, patient ID, date of visit, appointment and description. The receptionist also records the payment information with the patient's ID, date of payment and the total amount due. Payment is further classified into Cash or Insurance. A person can pay by cash, or by insurance or pay via a combination of both. The cash amount is recorded if a person pays by cash. For Insurance, the insurance details such as Insurance ID, Insurance Provider, Insurance coverage and the amount is recorded.

2. Project Questions

2.1. Is the ability to model superclass/subclass relationships likely to be important in a hospital environment such as Dallas Care? Why or why not?

It is important to have the superclass/subclass relationship in a hospital environment to model the concept of Inheritance as the attribute of a PERSON will be inherited by EMPLOYEE, CLASS1_PATIENT and CLASS2_PATIENT, DOCTOR, NURSE, RECEPTIONIST etc. If we don't use the concept of superclass/subclass, we would have to add the attributes of PERSON with each relation such as EMPLOYEE, and CLASS2_PATIENT in order to store their basic details which will lead to redundancy.

2.2. Can you think of 5 more business rules (other than the one explicitly described above) that are likely to be used in a medical care environment? Add your rules to the above requirement to be implemented.

The Business rules which are likely to be used in a medical care environment are listed below:

1. Room_type

```
ALTER TABLE room
ADD CONSTRAINT room_check2
CHECK (room_type IN ('VIP', 'Single', 'Three-bedded',
'Double-bedded', 'General'));
```

2. Person_gender

```
ALTER TABLE person
ADD CONSTRAINT person_check1
CHECK (gender in ('M', 'F'));
```

3. Room_no

```
ALTER TABLE room
ADD CONSTRAINT room_check1
CHECK (room_no > 0 AND room_no <= 50);
```

4. Medicine_price

```
ALTER TABLE medicine
ADD CONSTRAINT medicine_check2 CHECK (price > 0.0);
```

5. Medicine_quantity

```
ALTER TABLE medicine
ADD CONSTRAINT medicine_check1 CHECK (quantity > 0);
```

2.3. Justify using a Relational DBMS like Oracle for this project.

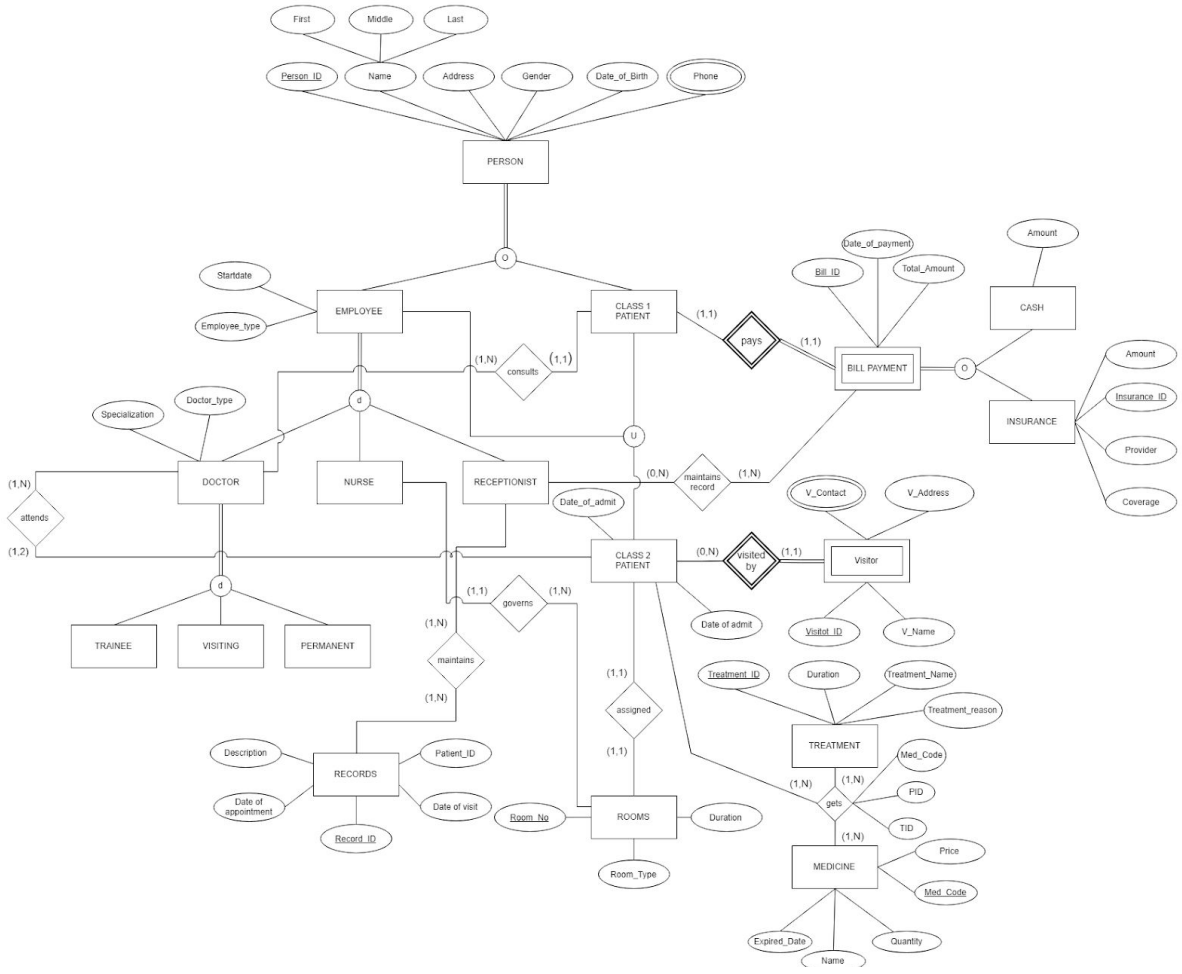
Using a relational DBMS would prove to be useful for a project like Dallas Care. Relational database has a lot of advantages to justify this statement. Some of them are listed below.

- Data Security: With an RDBMS you can hide sensitive tables and give them their authorization codes, providing a layer of protection for your data.
- Data Integrity: the structure of the relational database preserves the integrity of the data and makes it easier to meet compliance regulations.
- RDBMS Standards: Relational databases adhere to ACID properties to ensure the reliability of transactions.
- SQL Standard: SQL is a standardized language well understood by many applications, and many of the alternative database options provide SQL interfaces.
- Ease of Use: The use of tables to store data in columns and rows makes it easy to access and manage data.
- Performance: An RDBMS uses indexes to sort data and speed up performance, and supports both desktop and web applications.
- Development and Support: The large players — Oracle, Microsoft, SAP — have a vested interest in continuing to develop and evolve their database offering to meet modern standards.

3. EER Diagram with all Assumptions

(A higher resolution image has been provided separately as well)*

Dallas Care Hospital and Medical Care Center



- Entities Types:
  PERSON, BILL PAYMENT, VISITOR, RECORDS, ROOMS, MEDICINE, TREATMENT, CASH, INSURANCE, EMPLOYEE, CLASS 1 PATIENT, CLASS 2 PATIENT, DOCTOR, NURSE, RECEPTIONIST.

- Relationship Types:
  pays, consults, attends, maintains, assigned, visited by, maintains record, gets, governs

- Identification:
  All entity types and Relationship types are identified by names. Each entity of entity type is identified differently by:
  PERSON: Person_ID
  BILL PAYMENT: Bill_ID
  VISITOR: Key(Visitor_ID, Person_ID)
  RECORDS: Record_ID
  ROOMS: Room_No
  MEDICINE: Med_Code
  TREATMENT: Treatment_ID
  INSURANCE: Insurance_ID

- Specialization/Generalization:
  - EMPLOYEE and CLASS 1 PATIENT is a specialization of entity PERSON. This is a total overlapping specialization.
  - DOCTOR, NURSE and RECEPTIONIST are a specialization of EMPLOYEE based on the attribute – Designation. This is a total disjoint specialization.
  - CLASS 2 PATIENT is a union of EMPLOYEE and CLASS 1 PATIENT.
  - CASH and INSURANCE is a specialization of BILL PAYMENT based on the attribute – Payment_Type. This is a total overlapping specialization.

- Assumptions made for EER Diagram:
  - The Attribute Phone no of Entity PERSON is multivalued, as a person can have more than one phone number.
  - A Class 2 patient can be visited by N number of visitors but a visitor cannot visit more than one patient. Visitor is a weak entity dependent on a Class 2 patient.
  - A person can pay only one bill and a bill can be paid by only one person.
  - An individual room is assigned for each Class 2 patient and a patient cannot be assigned more than 1 room.
  - A receptionist can maintain records of multiple patients and a record can be maintained by multiple receptionists.
  - The Bill Payment information can be maintained by multiple receptionists and a receptionist can maintain multiple payment information of the patients.
  - A Class 2 patient can get 1 or more treatments and the same treatment can be given to one more patients.
  - A medicine can be given to 1 or more Class 2 patient and a Class 2 patient can get more than one medicine.

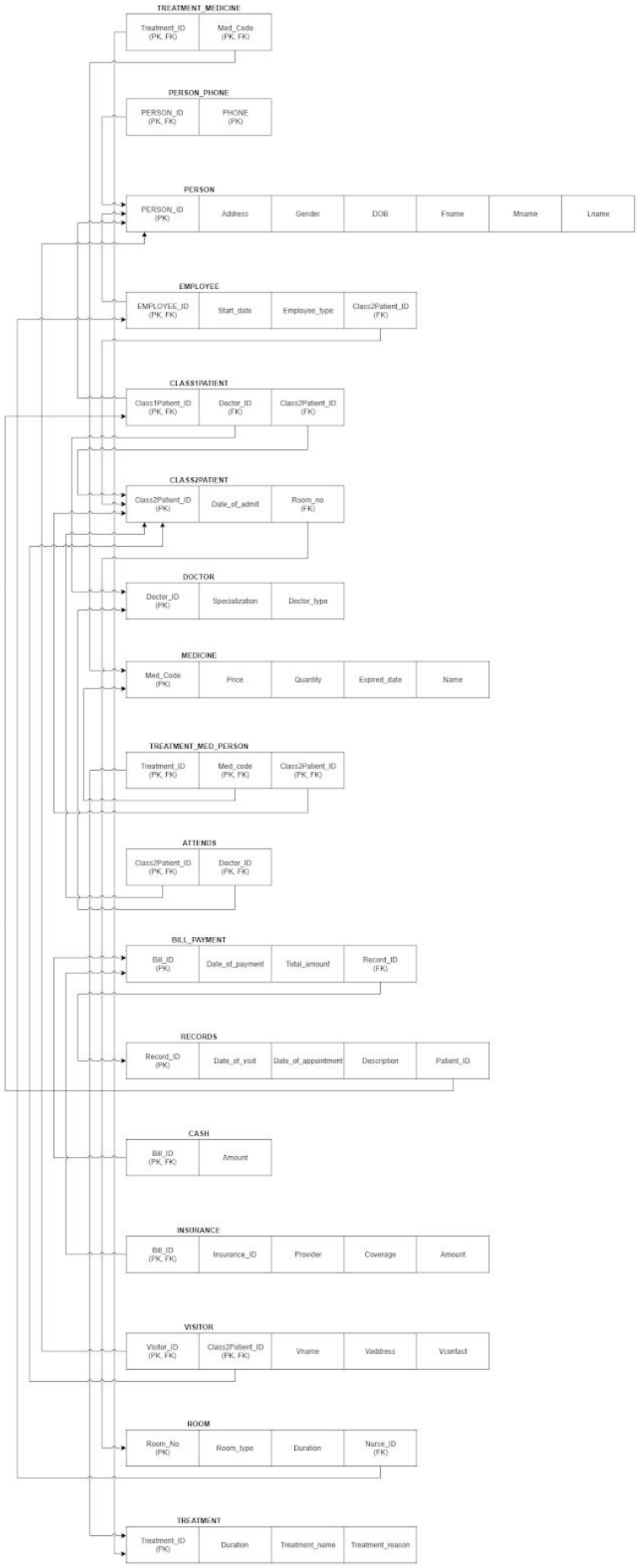
- Bill Payment depends on person. If a person doesn't exist there will be no patient for whom a bill can be generated.

4. Relational Schema in Third Normal Form:

4.1. Relational Schema

We have modified our relational model for this database. We have made one big change and certain minor changes to follow the third normal form. These are listed below:

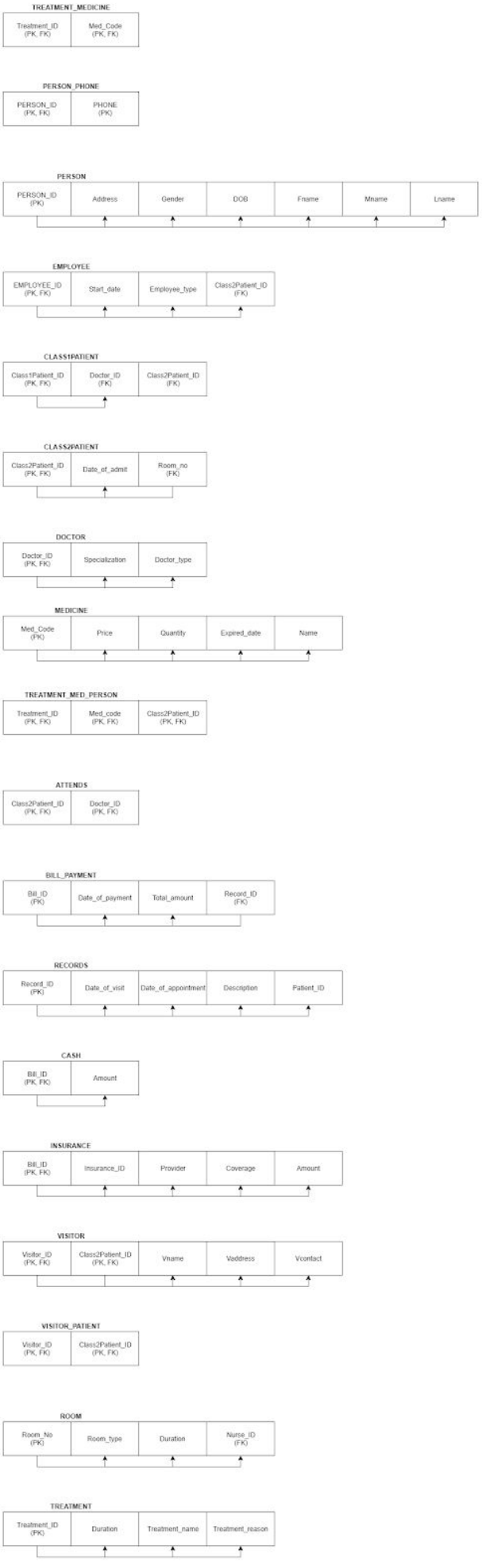
- The BILL_PAYMENT table in the previous phase was generating too many null values so to simplify the database, we have converted into three tables that are – BILL_PAYMENT, CASH, INSURANCE. Previously we were using Single relation with one type attribute but now we are using multiple relations.
- We have added a name attribute to the MEDICINE Table to satisfy the query.
- We have added a Treatment_Reason attribute to the TREATMENT Table to satisfy the query.
- For converting the union of Class2Patient we have used 8(a) rule this time instead of 8(b).



(A higher resolution image has been provided separately as well)*

4.2. Format for Every Relation

(A higher resolution image has been provided separately as well)*


TREATMENT_MEDICINE
Treatment_ID (PK, FK)
Med_Code (PK, FK)
PERSON_PHONE
PERSON_ID (PK, FK)
PHONE (PK)
PERSON
PERSON_ID (PK)
Address
Gender
DOB
Fname
Mname
Lname
EMPLOYEE
EMPLOYEE_ID (PK, FK)
Start_date
Employee_type
Class2Patient_ID (FK)
CLASS1PATIENT
Class1Patient_ID (PK, FK)
Doctor_ID (FK)
Class2Patient_ID (FK)
CLASS2PATIENT
Class2Patient_ID (PK, FK)
Date_of_admit
Room_no (FK)
DOCTOR
Doctor_ID (PK, FK)
Specialization
Doctor_type
MEDICINE
Med_Code (PK)
Price
Quantity
Expired_date
Name
TREATMENT_MED_PERSON
Treatment_ID (PK, FK)
Med_code (PK, FK)
Class2Patient_ID (PK, FK)
ATTENDS
Class2Patient_ID (PK, FK)
Doctor_ID (PK, FK)
BILL_PAYMENT
Bill_ID (PK)
Date_of_payment
Total_amount
Record_ID (FK)
RECORDS
Record_ID (PK)
Date_of_visit
Date_of_appointment
Description
Patient_ID
CASH
Bill_ID (PK, FK)
Amount
INSURANCE
Bill_ID (PK, FK)
Insurance_ID
Provider
Coverage
Amount
VISITOR
Visitor_ID (PK, FK)
Class2Patient_ID (PK, FK)
Vname
Vaddress
Vcontact
VISITOR_PATIENT
Visitor_ID (PK, FK)
Class2Patient_ID (PK, FK)
ROOM
Room_No (PK)
Room_type
Duration
Nurse_ID (FK)
TREATMENT
Treatment_ID (PK)
Duration
Treatment_name
Treatment_reason

5. All Requested SQL Statements:

5.1. Creation of Database with SQL Statements

5.1.1. Table Creation

ATTENDS

```
CREATE TABLE "TXT171930"."ATTENDS"
( "CLASS2PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DOCTOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "ATTENDS_PK" PRIMARY KEY ("CLASS2PATIENT_ID",
"DOCTOR_ID"),
CONSTRAINT "ATTENDS_FK1" FOREIGN KEY ("CLASS2PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE,
CONSTRAINT "ATTENDS_FK2" FOREIGN KEY ("DOCTOR_ID")
REFERENCES "TXT171930"."DOCTOR" ("DOCTOR_ID") ENABLE
)
```

BILL_PAYMENT

```
CREATE TABLE "TXT171930"."BILL_PAYMENT"
( "BILL_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DATE_OF_PAYMENT" DATE NOT NULL ENABLE,
"TOTAL_AMOUNT" FLOAT(126) NOT NULL ENABLE,
"RECORD_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "BILL_PAYMENT_PK" PRIMARY KEY ("BILL_ID"),
CONSTRAINT "BILL_PAYMENT_FK1" FOREIGN KEY ("RECORD_ID")
REFERENCES "TXT171930"."RECORDS" ("RECORD_ID") ENABLE
)
```

CASH

```
CREATE TABLE "TXT171930"."CASH"
( "BILL_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"AMOUNT" FLOAT(126),
CONSTRAINT "CASH_PK" PRIMARY KEY ("BILL_ID"),
CONSTRAINT "CASH_FK1" FOREIGN KEY ("BILL_ID")
REFERENCES "TXT171930"."BILL_PAYMENT" ("BILL_ID") ENABLE
)
```

CLASS1PATIENT

```
CREATE TABLE "TXT171930"."CLASS1PATIENT"
( "CLASS1PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DOCTOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"CLASS2PATIENT_ID" VARCHAR2(20 BYTE),
CONSTRAINT "CLASS1PATIENT_PK" PRIMARY KEY
("CLASS1PATIENT_ID"),
CONSTRAINT "CLASS1PATIENT_FK1" FOREIGN KEY
("CLASS1PATIENT_ID")
REFERENCES "TXT171930"."PERSON" ("PERSON_ID") ENABLE,
CONSTRAINT "CLASS1PATIENT_FK2" FOREIGN KEY ("DOCTOR_ID")
REFERENCES "TXT171930"."DOCTOR" ("DOCTOR_ID") ENABLE,
CONSTRAINT "CLASS1PATIENT_FK3" FOREIGN KEY
```

```
("CLASS2PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE
)
```

CLASS2PATIENT

```
CREATE TABLE "TXT171930"."CLASS2PATIENT"
( "CLASS2PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DATE_OF_ADMIT" DATE NOT NULL ENABLE,
"ROOM_NO" NUMBER(*,0) NOT NULL ENABLE,
CONSTRAINT "CLASS2PATIENT_PK" PRIMARY KEY
("CLASS2PATIENT_ID"),
CONSTRAINT "CLASS2PATIENT_FK1" FOREIGN KEY ("ROOM_NO")
REFERENCES "TXT171930"."ROOM" ("ROOM_NO") ENABLE
)
```

DOCTOR

```
CREATE TABLE "TXT171930"."DOCTOR"
( "DOCTOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DOCTOR_TYPE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"SPECIALIZATION" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "DOCTOR_PK" PRIMARY KEY ("DOCTOR_ID") CONSTRAINT
"DOCTOR_FK1" FOREIGN KEY ("DOCTOR_ID")
REFERENCES "TXT171930"."EMPLOYEE" ("EMPLOYEE_ID") ENABLE
)
```

EMPLOYEE

```
CREATE TABLE "TXT171930"."EMPLOYEE"
( "EMPLOYEE_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"START_DATE" DATE NOT NULL ENABLE,
"EMPLOYEE_TYPE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"CLASS2PATIENT_ID" VARCHAR2(20 BYTE),
CONSTRAINT "EMPLOYEE_PK" PRIMARY KEY ("EMPLOYEE_ID"),
CONSTRAINT "EMPLOYEE_FK1" FOREIGN KEY ("EMPLOYEE_ID")
REFERENCES "TXT171930"."PERSON" ("PERSON_ID") ENABLE,
CONSTRAINT "EMPLOYEE_FK2" FOREIGN KEY ("CLASS2PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE
)
```

INSURANCE

```
CREATE TABLE "TXT171930"."INSURANCE"
( "BILL_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"INSURANCE_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"PROVIDER" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"COVERAGE" FLOAT(126) NOT NULL ENABLE,
"AMOUNT" FLOAT(126) NOT NULL ENABLE,
CONSTRAINT "INSURANCE_PK" PRIMARY KEY ("BILL_ID"),
CONSTRAINT "INSURANCE_FK1" FOREIGN KEY ("BILL_ID")
```

```
REFERENCES "TXT171930"."BILL_PAYMENT" ("BILL_ID") ENABLE
)
```

MEDICINE

```
CREATE TABLE "TXT171930"."MEDICINE"
( "MED_CODE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"PRICE" FLOAT(126) NOT NULL ENABLE,
"QUANTITY" NUMBER(*,0) NOT NULL ENABLE,
"EXPIRED_DATE" DATE NOT NULL ENABLE,
"NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
CONSTRAINT "MEDICINE_PK" PRIMARY KEY ("MED_CODE")
)
```

PERSON

```
CREATE TABLE "TXT171930"."PERSON"
( "PERSON_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"ADDRESS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"GENDER" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DOB" DATE NOT NULL ENABLE,
"FNAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"MNAME" VARCHAR2(20 BYTE),
"LNAME" VARCHAR2(20 BYTE),
CONSTRAINT "PERSON_PK" PRIMARY KEY ("PERSON_ID")
)
```

PERSON_PHONE

```
CREATE TABLE "TXT171930"."PERSON_PHONE"
( "PERSON_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"PHONE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "PERSON_PHONE_PK" PRIMARY KEY ("PERSON_ID",
"PHONE"),
CONSTRAINT "PERSON_PHONE_FK1" FOREIGN KEY ("PERSON_ID")
REFERENCES "TXT171930"."PERSON" ("PERSON_ID") ENABLE
)
```

RECORDS

```
CREATE TABLE "TXT171930"."RECORDS"
( "RECORD_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DATE_OF_VISIT" DATE NOT NULL ENABLE,
"DATE_OF_APPOINTMENT" DATE,
"DESCRIPTION" VARCHAR2(100 BYTE),
"PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "RECORDS_PK" PRIMARY KEY ("RECORD_ID"),
CONSTRAINT "RECORDS_FK1" FOREIGN KEY ("PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE
)
```

ROOM

```
CREATE TABLE "TXT171930"."ROOM"
( "ROOM_NO" NUMBER(*,0) NOT NULL ENABLE,
"ROOM_TYPE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DURATION" NUMBER(*,0) NOT NULL ENABLE,
"NURSE_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "ROOM_PK" PRIMARY KEY ("ROOM_NO"),
CONSTRAINT "ROOM_FK1" FOREIGN KEY ("NURSE_ID")
REFERENCES "TXT171930"."EMPLOYEE" ("EMPLOYEE_ID") ENABLE
)
```

TREATMENT

```
CREATE TABLE "TXT171930"."TREATMENT"
( "TREATMENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DURATION" NUMBER(*,0) NOT NULL ENABLE,
"TREATMENT_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"TREATMENT_REASON" VARCHAR2(100 BYTE),
CONSTRAINT "TREATMENT_PK" PRIMARY KEY ("TREATMENT_ID")
)
```

TREATMENT_MED_PERSON

```
CREATE TABLE "TXT171930"."TREATMENT_MED_PERSON"
( "TREATMENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"MED_CODE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"CLASS2PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "TREATMENT_MED_PERSON_PK" PRIMARY KEY
("TREATMENT_ID", "MED_CODE", "CLASS2PATIENT_ID"),
CONSTRAINT "TREATMENT_MED_PERSON_FK1" FOREIGN KEY
("TREATMENT_ID")
REFERENCES "TXT171930"."TREATMENT" ("TREATMENT_ID") ENABLE,
CONSTRAINT "TREATMENT_MED_PERSON_FK2" FOREIGN KEY
("MED_CODE")
REFERENCES "TXT171930"."MEDICINE" ("MED_CODE") ENABLE,
CONSTRAINT "TREATMENT_MED_PERSON_FK3" FOREIGN KEY
("CLASS2PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE
)
```

TREATMENT_MEDICINE

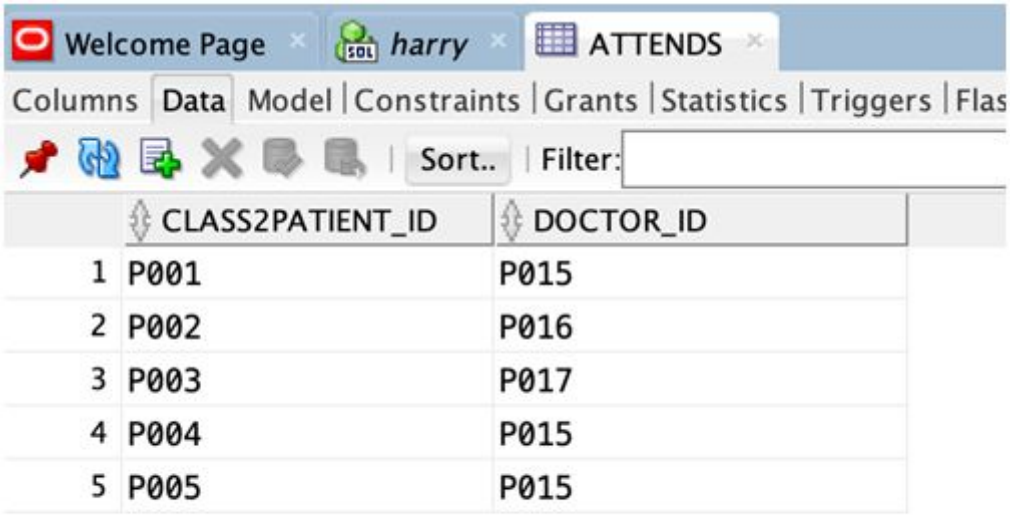
```
CREATE TABLE "TXT171930"."TREATMENT_MEDICINE"
( "TREATMENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"MED_CODE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "TREATMENT_MEDICINE_PK" PRIMARY KEY
("TREATMENT_ID", "MED_CODE"),
CONSTRAINT "TREATMENT_MEDICINE_FK1" FOREIGN KEY
("TREATMENT_ID")
REFERENCES "TXT171930"."TREATMENT" ("TREATMENT_ID") ENABLE,
CONSTRAINT "TREATMENT_MEDICINE_FK2" FOREIGN KEY ("MED_CODE")
REFERENCES "TXT171930"."MEDICINE" ("MED_CODE") ENABLE
```

)

VISITOR

```
CREATE TABLE "TXT171930"."VISITOR"
( "VISITOR_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"CLASS2PATIENT_ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"VNAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"VADDRESS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"VCONTACT" VARCHAR2(100 BYTE) NOT NULL ENABLE,
CONSTRAINT "VISITOR_PK" PRIMARY KEY ("VISITOR_ID",
"CLASS2PATIENT_ID"),
CONSTRAINT "VISITOR_FK1" FOREIGN KEY ("VISITOR_ID")
REFERENCES "TXT171930"."PERSON" ("PERSON_ID") ENABLE,
CONSTRAINT "VISITOR_FK2" FOREIGN KEY ("CLASS2PATIENT_ID")
REFERENCES "TXT171930"."CLASS2PATIENT" ("CLASS2PATIENT_ID")
ENABLE
)
```

5.1.2. Database State



	CLASS2PATIENT_ID	DOCTOR_ID
1	P001	P015
2	P002	P016
3	P003	P017
4	P004	P015
5	P005	P015

Welcome Page | harry | BILL_PAYMENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitio

Sort.. | Filter:

	BILL_ID	DATE_OF_PAYMENT	TOTAL_AMOUNT	RECORD_ID
1	1	20-NOV-18	100	6
2	2	17-NOV-17	200	5
3	3	15-JUN-18	300	4
4	4	30-JUN-16	400	1
5	5	26-JAN-17	500	2
6	6	17-MAR-15	600	3

Welcome Page | harry | CASH

Columns | Data | Model | Constraints | Grants | Statistics

Sort.. | Filter:

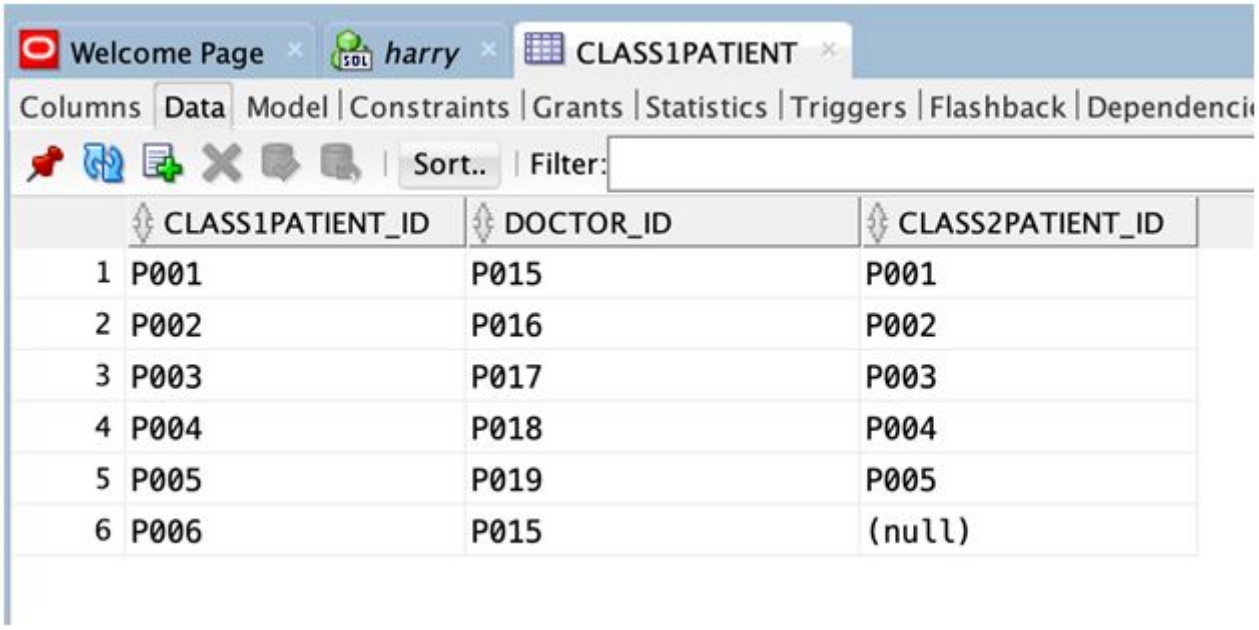
	BILL_ID	AMOUNT
1	1	100
2	2	200
3	3	300
4	6	100

Welcome Page | harry | CLASS2PATIENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies

Sort.. | Filter:

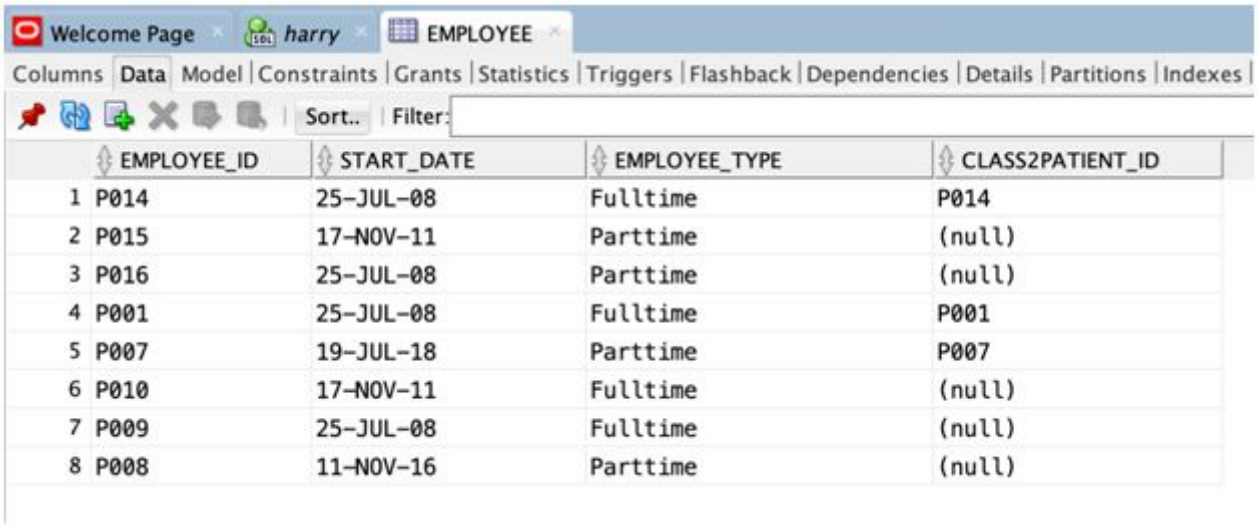
	CLASS2PATIENT_ID	DATE_OF_ADMIT	ROOM_NO
1	P014	02-NOV-18	2
2	P007	16-AUG-18	5
3	P001	09-NOV-18	1
4	P002	25-JUL-18	2
5	P003	02-NOV-18	3
6	P004	05-NOV-15	1
7	P005	18-JUL-15	5


Welcome Page | harry | CLASS1PATIENT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependenci

Sort.. | Filter:

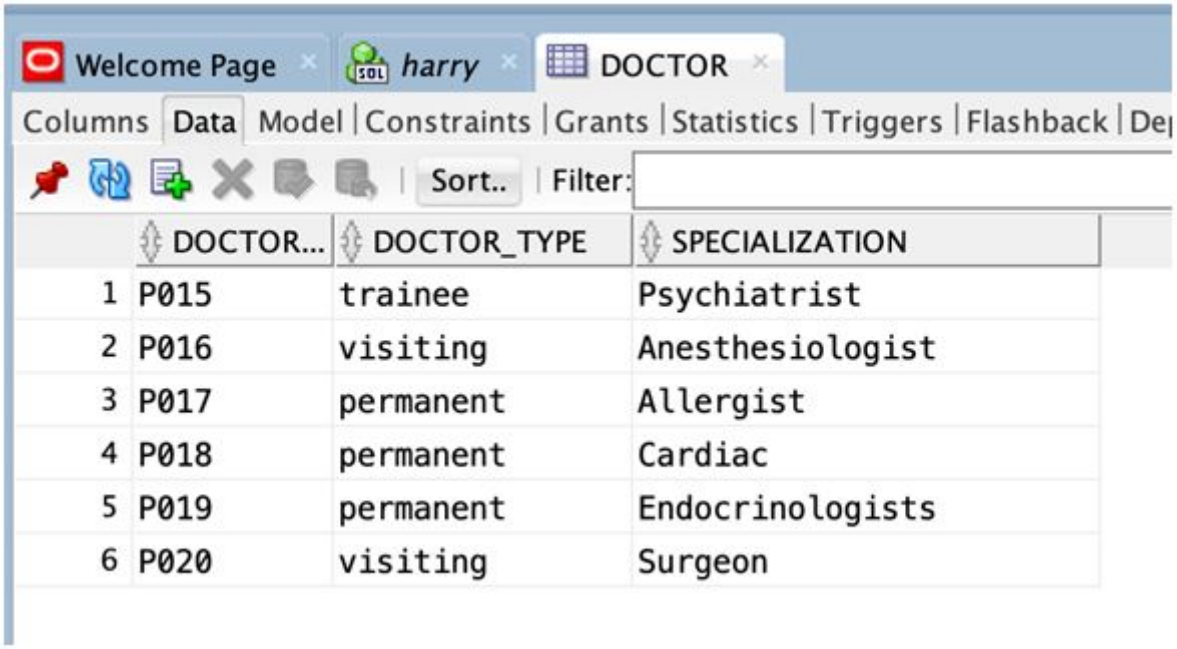
	CLASS1PATIENT_ID	DOCTOR_ID	CLASS2PATIENT_ID
1	P001	P015	P001
2	P002	P016	P002
3	P003	P017	P003
4	P004	P018	P004
5	P005	P019	P005
6	P006	P015	(null)


Welcome Page | harry | EMPLOYEE

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes

Sort.. | Filter:

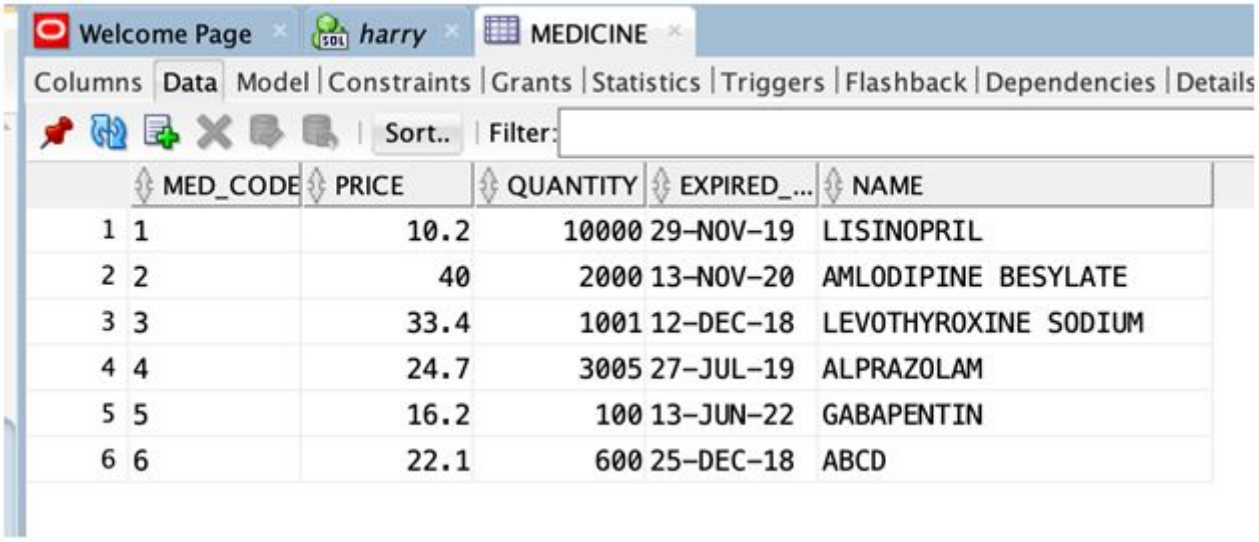
	EMPLOYEE_ID	START_DATE	EMPLOYEE_TYPE	CLASS2PATIENT_ID
1	P014	25-JUL-08	Fulltime	P014
2	P015	17-NOV-11	Parttime	(null)
3	P016	25-JUL-08	Parttime	(null)
4	P001	25-JUL-08	Fulltime	P001
5	P007	19-JUL-18	Parttime	P007
6	P010	17-NOV-11	Fulltime	(null)
7	P009	25-JUL-08	Fulltime	(null)
8	P008	11-NOV-16	Parttime	(null)


Welcome Page | harry | DOCTOR

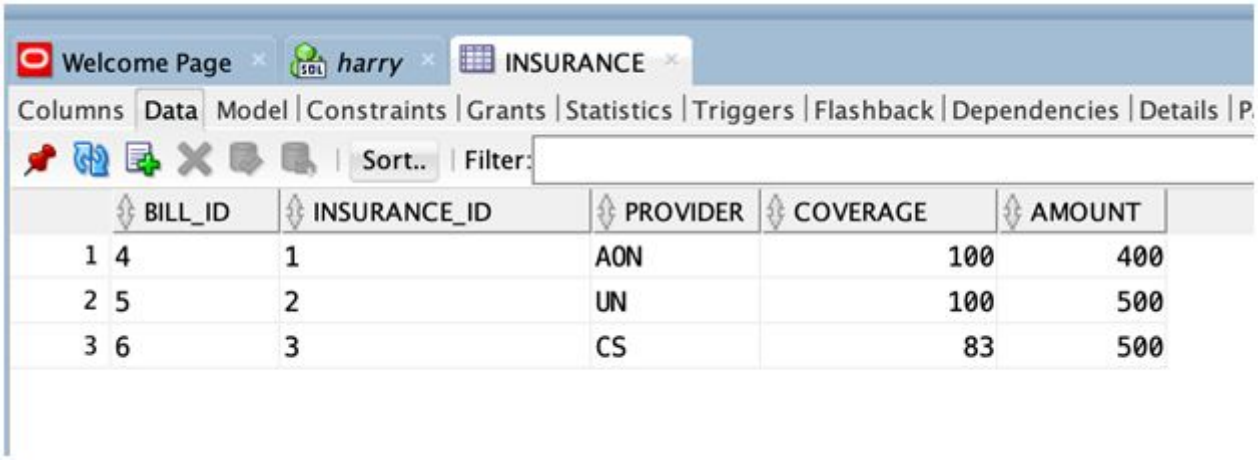
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | De

Sort.. | Filter:

	DOCTOR...	DOCTOR_TYPE	SPECIALIZATION
1	P015	trainee	Psychiatrist
2	P016	visiting	Anesthesiologist
3	P017	permanent	Allergist
4	P018	permanent	Cardiac
5	P019	permanent	Endocrinologists
6	P020	visiting	Surgeon

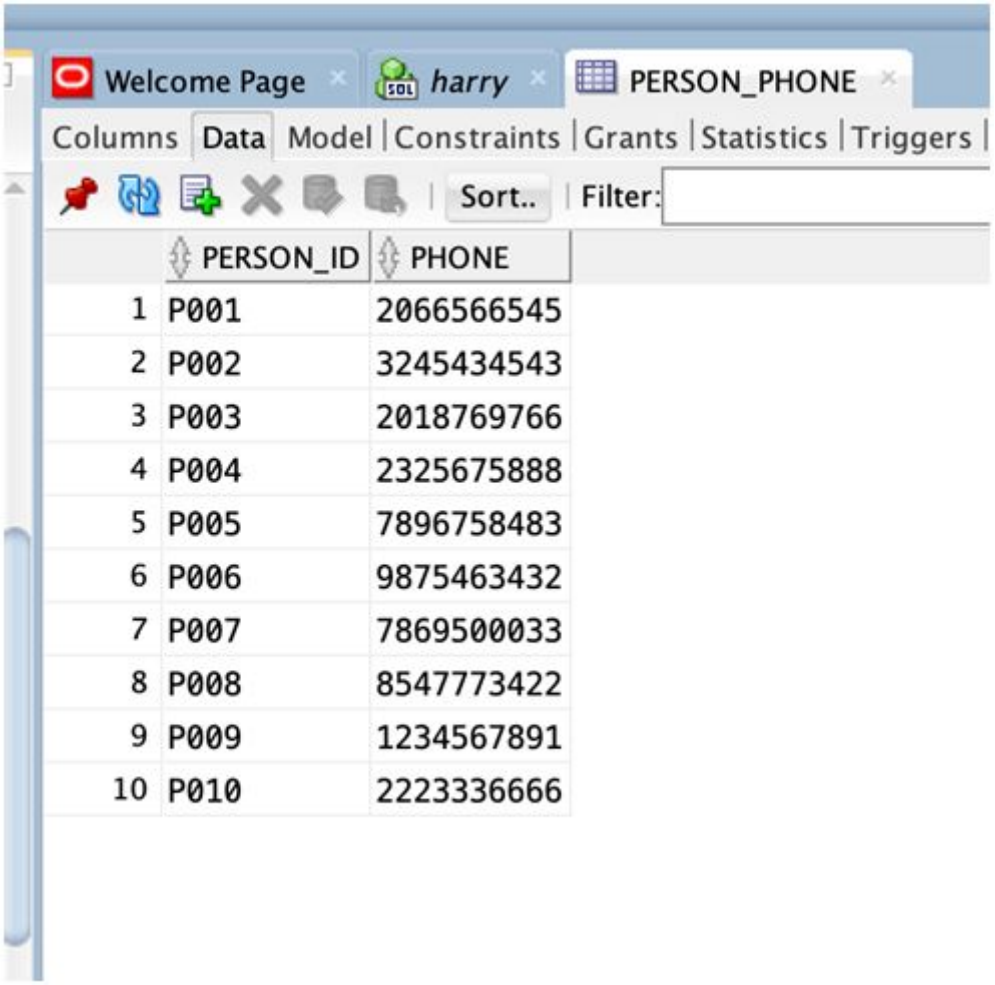


	MED_CODE	PRICE	QUANTITY	EXPIRED_...	NAME
1	1	10.2	10000	29-NOV-19	LISINOPRIL
2	2	40	2000	13-NOV-20	AMLODIPINE BESYLATE
3	3	33.4	1001	12-DEC-18	LEVOTHYROXINE SODIUM
4	4	24.7	3005	27-JUL-19	ALPRAZOLAM
5	5	16.2	100	13-JUN-22	GABAPENTIN
6	6	22.1	600	25-DEC-18	ABCD

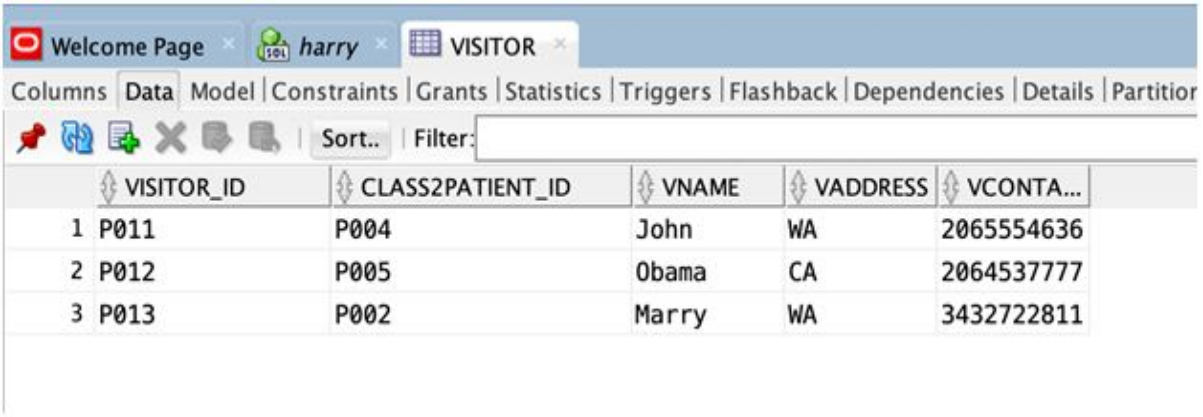


	BILL_ID	INSURANCE_ID	PROVIDER	COVERAGE	AMOUNT
1	4	1	AON	100	400
2	5	2	UN	100	500
3	6	3	CS	83	500

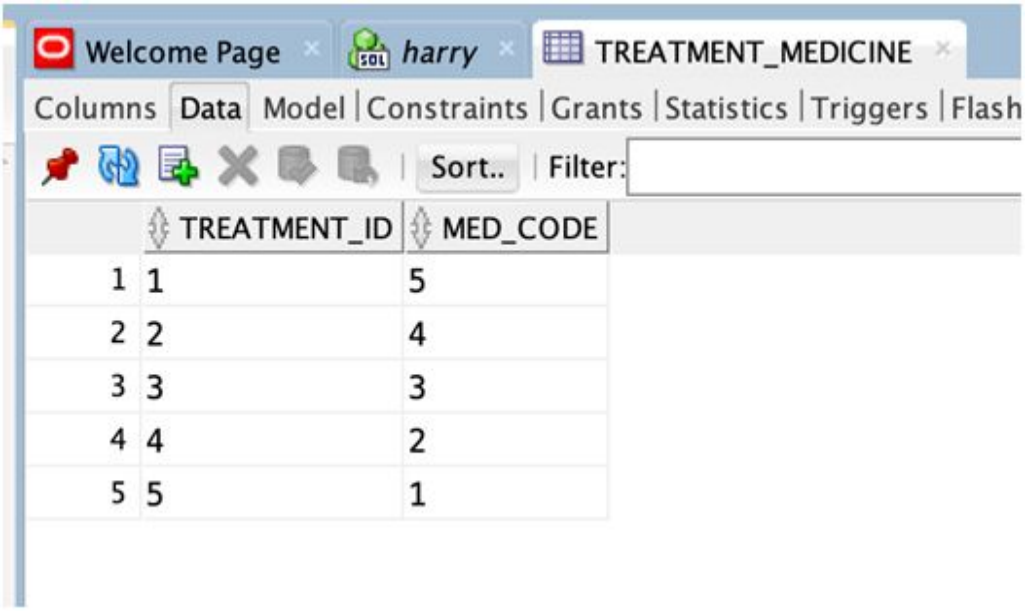
	PERSON_ID	ADDRESS	GENDER	DOB	FNAME	MNAME	LNAME
1	P014	CA	F	19-AUG-87	John	A	Barack
2	P015	OA	M	18-MAY-78	Delgado	B	Carla
3	P016	OA	M	30-JUN-91	Szeto	C	Jose
4	P017	NYC	M	23-APR-92	Kozhezva	E	Thao
5	P018	NYC	F	19-AUG-87	Carla	D	Delgado
6	P019	CA	F	22-NOV-96	Delgado	F	Kozhezva
7	P020	TX	M	28-NOV-98	Jose	G	Ian
8	P001	TX	M	23-APR-92	Andrew	H	Ngo
9	P002	OA	M	16-JUN-94	Alex	Y	Szeto
10	P003	TX	F	24-NOV-84	Alexandria	K	Kozhezva
11	P004	CA	F	03-NOV-18	Jose	H	Delgado
12	P005	CA	M	28-SEP-18	Thao	A	Nguyen
13	P006	OA	F	17-APR-18	Ian	B	Mascardo
14	P007	TX	M	30-JUN-91	Carla	C	Abenojar
15	P008	OA	F	08-NOV-18	Amine	D	Benelbar
16	P009	WA	M	26-NOV-18	Ari	E	Cohn
17	P010	WA	F	18-MAY-78	Carla	F	Ramires
18	P011	OA	F	22-NOV-96	John	G	Richard
19	P012	CA	F	19-NOV-92	Obama	H	Barack
20	P013	WA	M	19-AUG-87	Marry	F	Jane



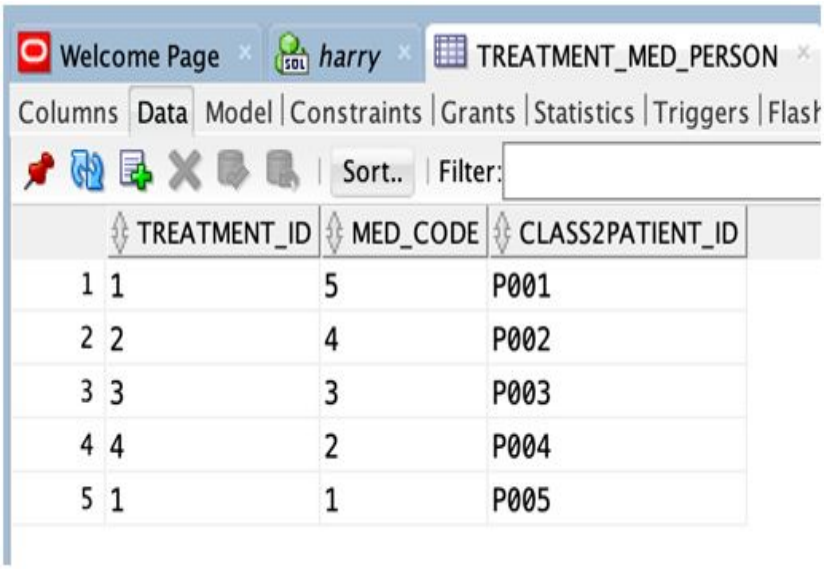
	PERSON_ID	PHONE
1	P001	2066566545
2	P002	3245434543
3	P003	2018769766
4	P004	2325675888
5	P005	7896758483
6	P006	9875463432
7	P007	7869500033
8	P008	8547773422
9	P009	1234567891
10	P010	2223336666



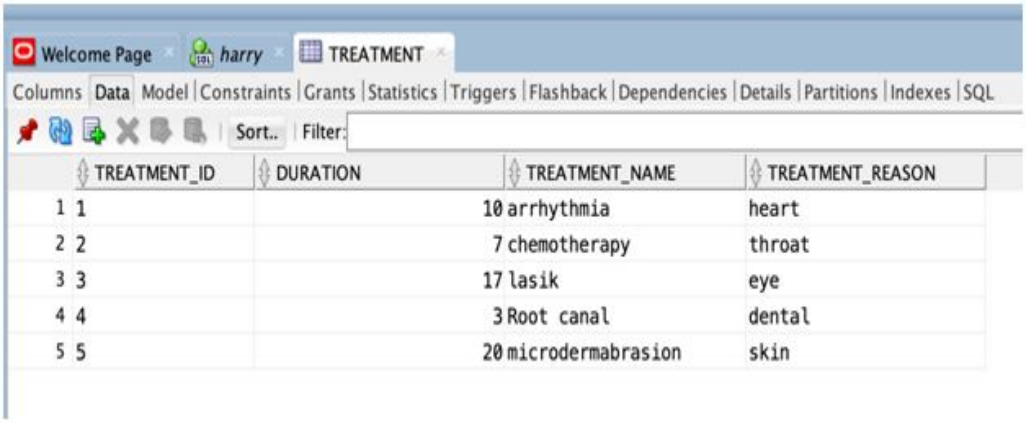
	VISITOR_ID	CLASS2PATIENT_ID	VNAME	VADDRESS	VCONTA...
1	P011	P004	John	WA	2065554636
2	P012	P005	Obama	CA	2064537777
3	P013	P002	Marry	WA	3432722811



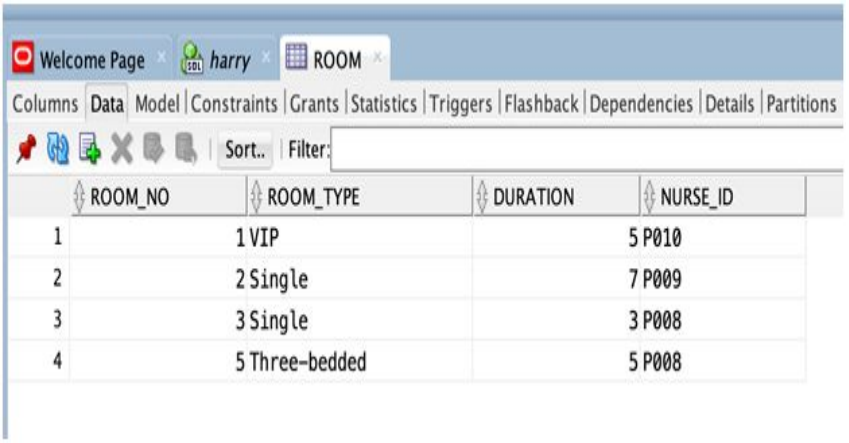
	TREATMENT_ID	MED_CODE
1	1	5
2	2	4
3	3	3
4	4	2
5	5	1



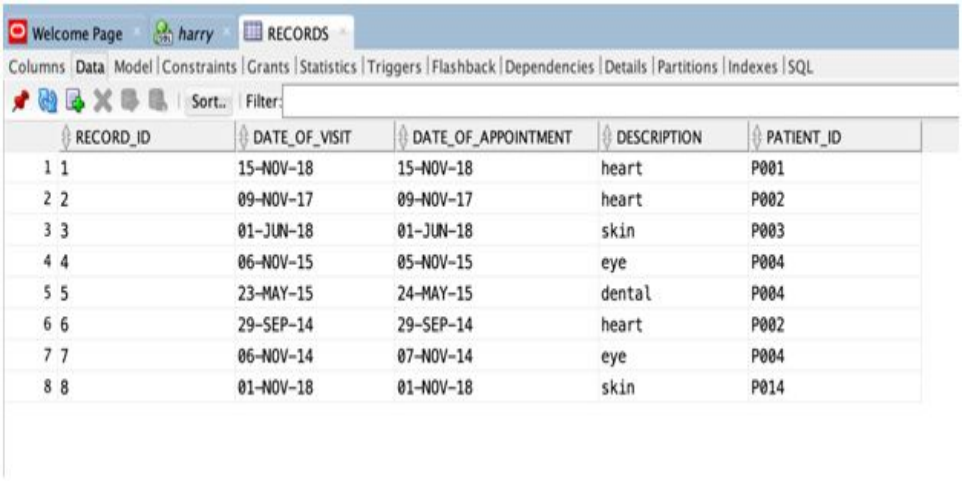
	TREATMENT_ID	MED_CODE	CLASS2PATIENT_ID
1	1	5	P001
2	2	4	P002
3	3	3	P003
4	4	2	P004
5	1	1	P005



	TREATMENT_ID	DURATION	TREATMENT_NAME	TREATMENT_REASON
1	1	10	arrhythmia	heart
2	2	7	chemotherapy	throat
3	3	17	lasik	eye
4	4	3	Root canal	dental
5	5	20	microdermabrasion	skin



	ROOM_NO	ROOM_TYPE	DURATION	NURSE_ID
1	1	VIP	5	P010
2	2	Single	7	P009
3	3	Single	3	P008
4	5	Three-bedded	5	P008



	RECORD_ID	DATE_OF_VISIT	DATE_OF_APPOINTMENT	DESCRIPTION	PATIENT_ID
1	1	15-NOV-18	15-NOV-18	heart	P001
2	2	09-NOV-17	09-NOV-17	heart	P002
3	3	01-JUN-18	01-JUN-18	skin	P003
4	4	06-NOV-15	05-NOV-15	eye	P004
5	5	23-MAY-15	24-MAY-15	dental	P004
6	6	29-SEP-14	29-SEP-14	heart	P002
7	7	06-NOV-14	07-NOV-14	eye	P004
8	8	01-NOV-18	01-NOV-18	skin	P014

5.2. Creation of Views

a. TopDoctor- This view returns the First Name, Last Name and Date of Joining of those doctors who have made more than 5 Class 1 patients and over 10 Class 2 patients.

```
CREATE VIEW TopDoctor AS
SELECT p.fname, p.lname, e.start_date
FROM person p, employee e
WHERE p.person_id=e.employee_id AND e.employee_id IN
    ( SELECT doctor_id
      FROM attends
      GROUP BY doctor_id
      HAVING COUNT(class2patient_id) > 2
      INTERSECT
      SELECT doctor_id
      FROM class1patient
      GROUP BY doctor_id
      HAVING COUNT(class1patient_id) >1
    );
```

b. TopTreatment- This view returns the treatment name of the most common treatment in Dallas Care along with the bill payment amount when a person receives that treatment.

```
CREATE VIEW TMPTOPTREATMENT AS
    SELECT tmp.treatment_id, tmp.class2patient_id
```

```
    FROM treatment_med_person tmp
    WHERE tmp.treatment_id IN
        ( SELECT treatment_id
          FROM treatment_med_person
          GROUP BY treatment_id
          HAVING COUNT(class2patient_id) = ( SELECT MAX(m)
          FROM (SELECT
          COUNT(class2patient_id) AS m, treatment_id
          FROM treatment_med_person
          GROUP BY treatment_id))
        );

CREATE VIEW TopTreatment AS
 SELECT tr.treatment_name, b.total_amount
 FROM BILL_PAYMENT b, RECORDS r, TMPTOPTREATMENT t, TREATMENT
tr
 WHERE b.record_id = r.record_id AND r.patient_id =
t.class2patient_id
AND tr.treatment_id = t.treatment_id;
```

c. ReorderMeds- This view returns the medicines that need to be reordered. A medicine needs to be reordered if the expiration date is 1 month from current date or quantity is less than 1000.

```
CREATE VIEW ReorderMeds AS
    SELECT p.name
    FROM medicine p
    WHERE p.quantity < 1000 or (p.expired_date -
TRUNC(SYSDATE) < 30) ;
```

d. PotentialPatient- This view returns the name, phone number and ID of patients who visited the hospital more than 3 times as a Class 1 patient but has not been admitted yet.

```
CREATE VIEW PotentialPatient AS
SELECT p.fname, p.lname, ph.phone
FROM person p, person_phone ph
WHERE p.person_id=ph.person_id AND p.person_id IN
    (   SELECT class1patient_id
        FROM class1patient
        WHERE class2patient_id IS NULL AND class1patient_id
IN
            (
                SELECT patient_id
                FROM records
                GROUP BY patient_id
                HAVING COUNT(*) > 3
            )
    );
```

e. MostFrequentIssues - This view returns the maximum frequency of the reason that patients visit the hospital for and the associated treatment for the same. For example, if patients visit the hospital mostly complaining about heart issues then what are the treatment associated with heart issues.

```
CREATE VIEW MostFrequentIssues AS
SELECT t.treatment_name, t.treatment_reason
FROM treatment t
WHERE t.treatment_id IN
(
    SELECT treatment_id
    FROM treatment_med_person
    GROUP BY treatment_id
    HAVING COUNT(class2patient_id) = ( SELECT MAX(m)
    FROM (SELECT
    COUNT(class2patient_id) AS m, treatment_id FROM
treatment_med_person
    GROUP BY treatment_id
     ))
);
```

5.3. Creation of SQL Queries

a. For each Doctor class, list the start date and specialization of the doctor.

```
SELECT start_date, specialization
FROM doctor, employee
WHERE doctor_id = employee_id;
```

b. Find the names of employees who have been admitted to the hospital within 3 months of joining.

```
SELECT p.fname, p.lname, p.person_id
FROM person p
WHERE p.person_id IN
    ( SELECT e.employee_id
      FROM employee e, class2patient c1
      WHERE e.class2patient_id IS NOT NULL AND
            e.class2patient_id=c1.class2patient_id  AND
            (c1.date_of_admit - e.start_date) < 90
    );
```

c. Find the average age and class (trainee, visiting or permanent) of top 5 doctors in the hospital.

```
SELECT d.doctor_type, ROUND(AVG((TRUNC(SYSDATE) -
p.dob)/365)) as AGE
FROM doctor d, person p
WHERE d.doctor_id = p.person_id AND d.doctor_id IN
(SELECT doctor_id
    FROM attends
    GROUP BY doctor_id
    HAVING COUNT(class2patient_id) > 2
```

```
    UNION
    SELECT doctor_id
    FROM class1patient
    GROUP BY doctor_id
    HAVING COUNT(person_id) > 2
)
GROUP BY d.doctor_type
```

d. Find the name of medicines associated with the most common treatment in the hospital.

```
SELECT p.name
FROM treatment_med_person ta, treatment t, medicine p
WHERE
    ta.treatment_id = t.treatment_id AND
    ta.med_code = p.med_code AND
    t.treatment_name IN
    ( SELECT treatment_name FROM MostFrequentIssues
    );
```

e. Find all the doctors who have not had a patient in the last 5 months. (Hint: Consider the date of payment as the day the doctor has attended a patient/been consulted by a patient.

```
SELECT c2.doctor_id
FROM records r, attends c2
WHERE
   r.patient_id = c2.class2patient_id AND r.patient_id NOT IN
   ( SELECT r1.patient_id
     FROM records r1
     WHERE
        (
          SELECT TRUNC(SYSDATE) - r1.date_of_visit Days FROM
Dual) < 65
        )
   UNION
   SELECT c1.doctor_id
   FROM records r, class1patient c1
   WHERE
      c1.class1patient_id is NULL AND
      r.patient_id = c1.class1patient_id AND
      r.patient_id NOT IN
           ( SELECT r1.patient_id
             FROM records r1
             WHERE
                (
                  SELECT TRUNC(SYSDATE) - r1.date_of_visit
Days FROM Dual) < 65
                )
```

f. Find the total number of patients who have paid completely using insurance and the name of the insurance provider.

```
SELECT COUNT(i1.provider), i1.provider
FROM insurance i1, records r, bill_payment b
WHERE
   b.record_id = r.record_id AND
   b.bill_id = i1.bill_id AND
   i1.bill_id IN
            (
              SELECT i.bill_id
              FROM insurance i
              WHERE i.bill_id NOT IN
                ( SELECT bill_id
                  FROM cash
                )
            )
GROUP BY i1.provider;
```

g. Find the most occupied room in the hospital and the duration of the stay.

```
SELECT r.room_no, r.duration
FROM room r
WHERE r.room_no IN
   (
    SELECT c2.room_no
    FROM class2patient c2
    GROUP BY c2.room_no
    HAVING COUNT(c2.class2patient_id) =
        (
          SELECT MAX(m)
          FROM(
           SELECT c1.room_no, COUNT(c1.class2patient_id) AS m
           FROM class2patient c1
           GROUP BY c1.room_no)
        )
   );
```

h. Find the year with the maximum number of patient visiting the hospital and the reason for their visit.

```
SELECT r2.description AS REASON, EXTRACT(year from
r2.date_of_visit) AS YEAR
FROM RECORDS r2
WHERE EXTRACT(year from r2.date_of_visit) = (
SELECT EXTRACT(year from date_of_visit) AS y
FROM RECORDS r
GROUP BY EXTRACT(year from date_of_visit)
HAVING COUNT(PATIENT_ID) =
(SELECT MAX(cc)
FROM(
```

```
SELECT EXTRACT(year from date_of_visit), COUNT(PATIENT_ID) AS
cc
FROM RECORDS r
GROUP BY EXTRACT(year from date_of_visit)
)))
```

i. Find the duration of the treatment that is provided the least to patients.

```
SELECT t.treatment_name, t.duration
FROM treatment t
WHERE t.treatment_id IN(
    SELECT treatment_id
    FROM treatment_med_person
    GROUP BY treatment_id
    HAVING COUNT(class2patient_id) = ( SELECT MIN(m)
    FROM (SELECT
    COUNT(class2patient_id) AS m, treatment_id FROM
treatment_med_person
    GROUP BY treatment_id
     )));
```

j. List the total number of patients that have been admitted to the hospital after the most current employee has joined.

```
SELECT COUNT(class2patient_id) cnt
FROM class2patient
WHERE
    date_of_admit -
    (SELECT MAX(START_DATE) dat
     FROM EMPLOYEE
    ) > 0;
```

k. List all the patient records of those who have been admitted to the hospital within a week of being consulted by a doctor.

```
SELECT patient_id
FROM class2patient c2, records c1
WHERE
   c2.class2patient_id = c1.patient_id AND
   (c2.date_of_admit - c1.date_of_visit) BETWEEN 0 AND 7;
```

l. Find the total amount paid by patients for each month in the year 2017.

```
SELECT EXTRACT(month from date_of_payment) as Month,
SUM(b.total_amount) AS Total_amount
FROM bill_payment b
WHERE EXTRACT(YEAR from date_of_payment) = 2017
GROUP BY EXTRACT(month from date_of_payment);
```

m. Find the name of the doctors of patients who have visited the hospital only once for consultation and have not been admitted to the hospital.

```
SELECT Fname
FROM Person
```

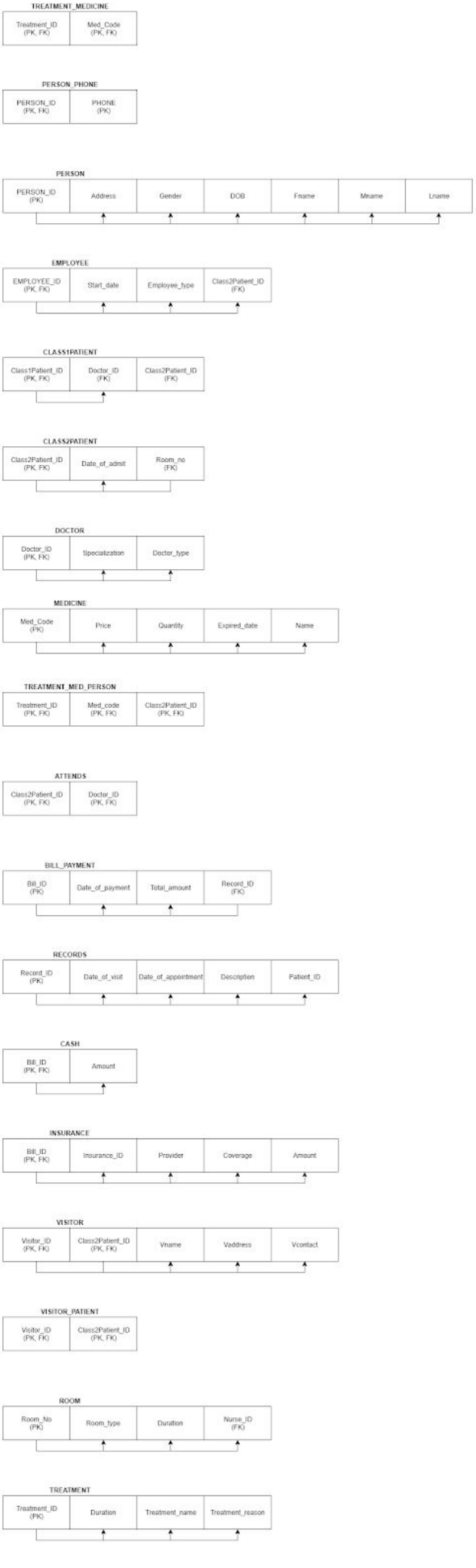
```
WHERE person_id IN
   ( SELECT DISTINCT(doctor_id)
     FROM class1patient
     WHERE class1patient_id NOT IN
        (
          SELECT class2patient_id
          FROM class2patient)
   );
```

n. Find the name and age of the potential patients in the hospital.

```
SELECT person_id,DOB, (fname || ' ' || mname || ' ' || lname)
AS name
FROM person
WHERE person_id IN
    (
      SELECT class1patient_id
      FROM class1patient
    );
```

6. Dependency Diagram:

(A higher resolution image has been provided separately as well)*


TREATMENT_MEDICINE
Treatment_ID (PK, FK)
Med_Code (PK, FK)
PERSON_PHONE
PERSON_ID (PK, FK)
PHONE (PK)
PERSON
PERSON_ID (PK)
Address
Gender
DOB
Fname
Mname
Lname
EMPLOYEE
EMPLOYEE_ID (PK, FK)
Start_date
Employee_type
Class2Patient_ID (FK)
CLASS1PATIENT
Class1Patient_ID (PK, FK)
Doctor_ID (FK)
Class2Patient_ID (FK)
CLASS2PATIENT
Class2Patient_ID (PK, FK)
Date_of_admit
Room_no (FK)
DOCTOR
Doctor_ID (PK, FK)
Specialization
Doctor_type
MEDICINE
Med_Code (PK)
Price
Quantity
Expired_date
Name
TREATMENT_MED_PERSON
Treatment_ID (PK, FK)
Med_code (PK, FK)
Class2Patient_ID (PK, FK)
ATTENDS
Class2Patient_ID (PK, FK)
Doctor_ID (PK, FK)
BILL_PAYMENT
Bill_ID (PK)
Date_of_payment
Total_amount
Record_ID (FK)
RECORDS
Record_ID (PK)
Date_of_visit
Date_of_appointment
Description
Patient_ID
CASH
Bill_ID (PK, FK)
Amount
INSURANCE
Bill_ID (PK, FK)
Insurance_ID
Provider
Coverage
Amount
VISITOR
Visitor_ID (PK, FK)
Class2Patient_ID (PK, FK)
Vname
Vaddress
Vcontact
VISITOR_PATIENT
Visitor_ID (PK, FK)
Class2Patient_ID (PK, FK)
ROOM
Room_No (PK)
Room_type
Duration
Nurse_ID (FK)
TREATMENT
Treatment_ID (PK)
Duration
Treatment_name
Treatment_reason

7. Conclusion:

This report summarizes all the necessary descriptions and solutions for the Dallas Care database, including EER diagrams, relational schemas in third normal form, SQL statements for creation of database and views, as well as other corresponding queries and a dependency diagram. This database has been implemented in Oracle and has a database state for query testing. Here, we also explain the reasoning behind choosing a relational database as well as implementing a superclass-subclass relationship.