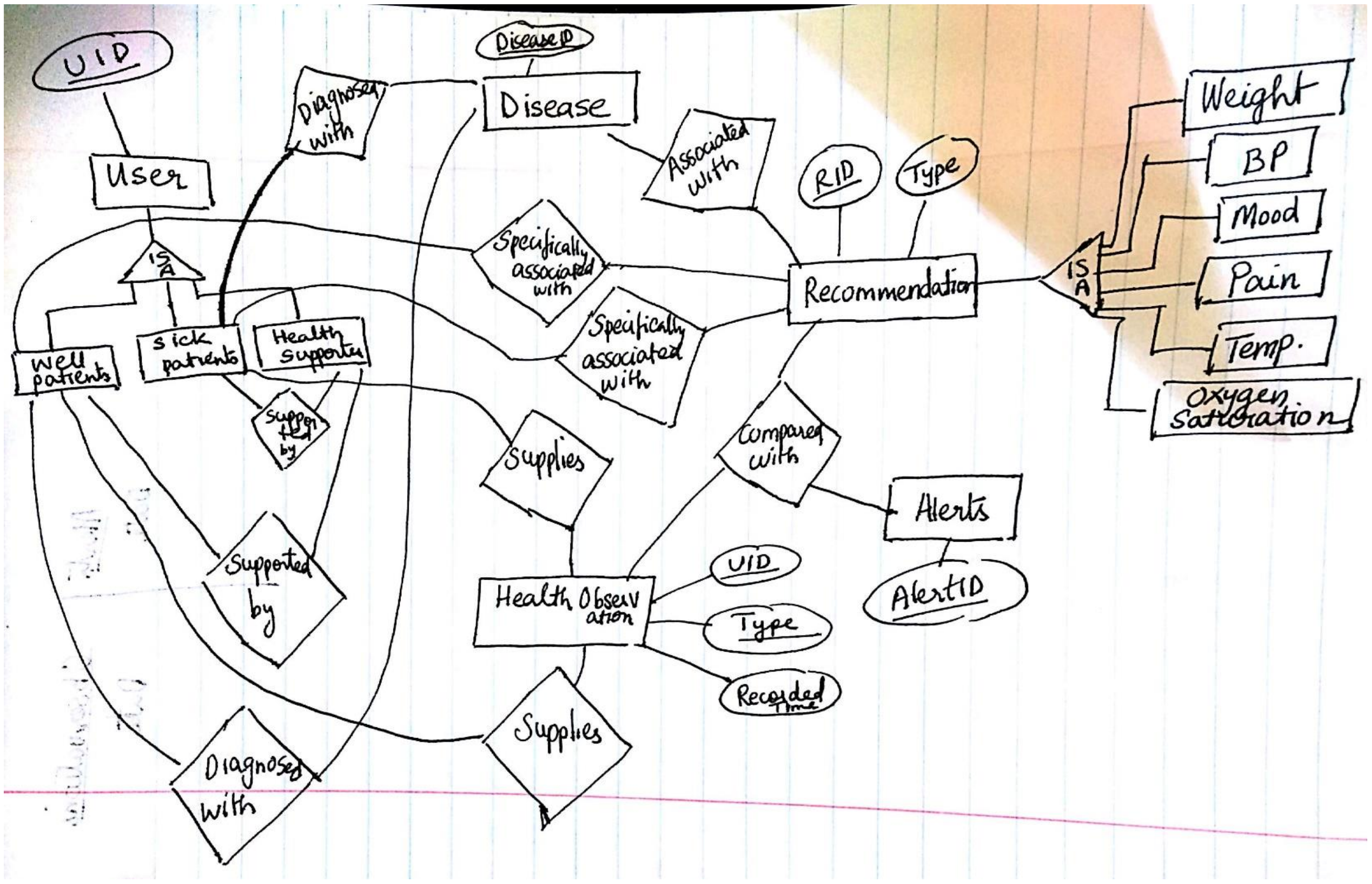
CSC 540 PROJECT 1

IMPLEMENTATION OF A PERSONAL HEALTH MANAGEMENT SYSTEM

BY-
KAUSHIK RAJAN (kaushi – 200111140)
KESHAV PARTHASARATHY (kpartha2 – 200106239)
SHARATH SREENIVASAN (ssreeni – 200109355)
NIKHIL RAMESH (nramesh2 – 200110404)

ER DIAGRAM



Explanation of ER diagram :

The primary person entity is called USER which is the parent class for everybody who uses the system. Each User is uniquely identified by a **userid**.
There are 3 children for this parent class : namely “SICKPATIENT”, “WELLPATIENT”, and “HEALTHSUPPORTER” . These 3 children have an ISA Relationship with USER meaning they have the same primary key.

“DISEASE” is a separate entity indexed by a **DiseaseID**. The SICKPATIENTS entity is connected to DISEASE through the relationship “DIAGNOSEDWITH”. The connection is such that every sick patient has **1 or more diseases**. WELLPATIENTS is also connected to DISEASES through a similar relationship but it is such that every patient has **0 or more diseases.**

Every DISEASE is AssociatedWith a RECOMMENDATION and has a foreign key referencing RECOMMENDATION. Each RECOMMENDATION has a primary key of **recommendationid**. The Recommendation tuple merely identifies that as a General or Specific , in the case of specific, there is a Patient field which references SICKPATIENTS/WELLPATIENTS.

Each value of the Recommendation is stored in a separate table namely WEIGHT /BLOODPRESSURE /OXYGENSATURATION /PAIN /MOOD /TEMPERATURE each of which has an ISA relationship with RECOMMENDATION and all of which have a primary key **recommendationid.** This allows for easy access and viewing of the limits of each of the recommendation limit attributes.

HEALTHOBSERVATIONS are associated with a USER through a SUPPLIES relation, each tuple identified by a combination of **PatientID, ObservedTime, ObservationType.** This was done because a combination of these 3 attributes enables us to uniquely identify a record in the Observations table.

HEALTHOBSERVATIONS may be COMPAREDWITH RECOMMENDATIONS in order to generate ALERTS. ALERTS are an entity which may be identified through an **AlertID**.

The relationship SUPPORTEDBY describes the association between a PATIENT and a HEALTHSUPPORTER and has a primary key of **PatientID, SupporterType** . SupporterType can take values “Primary”/” Secondary” ensuring that everyone has at most 2 Health Supporters. To further enforce the constraint of every SICKPATIENT having a HealthSupporter, SICKPATIENTS has a non-null attribute referencing HealthSupporter.

Table List and Description :

User Table

Description

A table that contains a list of all the users in the system. This includes a Unique identifier as well as contact information for each User in the system.

Attributes

Userid, Username, Password, DOB, Gender and Address

SQL

```
Create Table Patients
(
USERID VARCHAR (3) NOT NULL,
USERNAME VARCHAR (20),
PASSWORD VARCHAR (20),
DOB DATE,
GENDER CHAR (1),
ADDRESS VARCHAR (50)
)
```

Primary Key(s): **UserID**

Functional Dependencies :
Userid => Username, Password, DOB, Gender and Address

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Sick Patients

Description
This table contains a list of all the users who are diagnosed with at least 1 disease. This Entity maintains an ISA relationship with the User Entity.

Attributes
Userid, DiseaseId, HealthSupporterID

SQL

```
Create Table Sickpatients
(
USERID VARCHAR(3) NOT NULL,
DISEASEID INTEGER NOT NULL,
HEALTHSUPPORTERID VARCHAR (3) NOT NULL
)
```

Primary Key(s): **UserID, DiseaseID**

Functional Dependencies
Userid, DiseaseID =>HealthSupporterID

Normal Form

A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications
The attributes were chosen so as to ensure that every sick patient had at least one diagnosis and one health supporter. They key was chosen as mentioned to allow for multiple diagnoses for the same Patient.

Well Patients

Description
This table contains a list of all the users who may be diagnosed with some disease. This Entity maintains an ISA relationship with the User Entity.

Attributes
Userid, DiseaseID, HealthSupporterID

SQL
```
Create Table Wellpatients
(
USERID VARCHAR (3) NOT NULL,
DISEASEID INTEGER,
HEALTHSUPPORTERID VARCHAR (3)
)
```

Primary Key(s): **UserID**

Functional Dependencies
Userid => DiseaseID, HealthSupporterID

Normal Form
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications
Since both DiseaseID and HealthSupporter can be NULL values there is no need to include them in the Primary Key. Furthermore, all Well Patients are diagnosed with Disease “WELL” so as to access the corresponding Recommendations associated with being a “well” Patient.

Disease

Description
This table contains a list of all the diseases. Each Disease is identified by a unique identifier.

Attributes
DiseaseId, Disease Name, Recommendationid

SQL

```
Create Table Disease
(
DiseaseID integer NOT NULL,
DiseaseName varchar(20),
RecommendationID integer
)
```

Functional Dependencies
DiseaseID => DiseaseName, RecommendationID

Normal Form
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications
Recommendation ID is included in this table so that all users belonging to specific disease class may access the inherited General Recommendation that is associated with this disease.

SupportedBy

Description
This table describes the “Supporting” relationship between a patient and a HealthSupporter. This is a table for the relationship “SupportedBy”

Attributes :
HealthSupporterID ,PatientID , SupporterType, AuthorizationDate

SQL :
```
Create Table SupportedBy
(
HealthSupporterID varchar(3),
PatientID varchar(3) NOT NULL,
SupporterType Varchar(10) NOT NULL,
AuthorizationDate Date
```

)

Primary Key(s): **PatientID, Supportertype**

Functional Dependencies :
PatientID,SupporterType =>HealthSupporterID,AuthorizationDate

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
By constraining SupporterType to only having values of "Primary"/"Secondary" We ensure that every Patient can have at most 1 Primary and 1 Secondary Health Supporter, and it makes intuitive sense that we may identify who the Health Supporter is if we know the Patient and the SupporterType.

Recommendations

Description
This table acts as an interface for the Details of each Recommendation and each tuple is uniquely identified by an index and also contains a parameter which specifies it to be General or Specific.

Attributes :
RecommendationID, RecommendationType, PatientID

SQL :
```
Create Table Recommendation
(
RecommendationID integer NOT NULL,
RecommendationType varchar(10),
PatientID varchar(3)
)
```

Primary Key(s): RecommendationID

Functional Dependencies :
RecommendationID => RecommendationType, PatientID

Normal Form :

A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
The Recommendations all have a type of either General or Specific. Specific Recommendations also must have a PatientID which signifies to which Patient that Recommendation is intended. PatientID may be left Blank for General Recommendations

Observation

Description
This table holds all the observations that Patients take and enter into the system. Users may view their own observations or those of whom they are a supporter for.

Attributes :
PatientID,ObservedTime, RecordedTime, Value, ObservationType

SQL :

```
Create table observation
(
patientid varchar(3) NOT NULL,
observationtime timestamp NOT NULL,
recordingtime timestamp,
value varchar(10),
observationtype varchar(25) NOT NULL
)
```

Primary Key(s): **PatientID, ObservationTime, ObservationType**

Functional Dependencies :
PatientID,ObservedTime,ObservationType =>RecordedTime, Value

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
This is the most intuitively correct primary key, we can obtain the value if we know who recorded what and when.

Weight

Description
This table holds all the Recommended Weight Limit for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID, lower,upper, frequency

SQL :

```
Create Table Weight
(
RecommendationID integer NOT NULL,
Lower integer,
Upper integer,
Frequency integer
)
```

Primary Key(s): **RecommendationID**

Functional Dependencies :
RecommendationID =>lower,upper, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

BloodPressure

Description
This table holds all the Recommended BloodPressure Limits for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID,syslower,sysupper,dialower, diahigher, frequency

SQL :

```
Create table BP
(
RecommendationID integer NOT NULL,
lowersys integer,
highsys integer,
lowerdia integer,
highdia integer,
frequency integer
)
```

Primary Key: **RecommendationID**

Functional Dependencies :
RecommendationID =>lowersys,highsys,lowerdia, highdia, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

OxygenSaturation

Description
This table holds all the Recommended OxygenSaturation Limits for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID,lower,upper, frequency

SQL :
```
Create Table Oxygensaturation
(
RecommendationID integer NOT NULL,
Lower integer,
Upper integer,
Frequency integer
)
```

Primary Key(s): **RecommendationID**

Functional Dependencies :
RecommendationID =>lower,upper, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

Pain

Description
This table holds all the Recommended Pain Limits for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID, painvalue, frequency

SQL :

Primary Key(s): **RecommendationID**

Functional Dependencies :
RecommendationID =>painvalue, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

Mood

Description
This table holds all the Recommended Mood for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID, Mood, frequency

SQL :

```
Create Table Mood
(
RecommendationID integer NOT NULL,
mood varchar(10),
Frequency integer
)
```

Primary Key(s): **RecommendationID**

Functional Dependencies :
RecommendationID =>Mood, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

Temperature

Description
This table holds all the Recommended TemperatureLimits for a specific Recommendation. This entity maintains an ISA relationship with Recommendation

Attributes :
RecommendationID,lower,upper, frequency

SQL :

```
Create Table Temperature
(
RecommendationID integer NOT NULL,
Lower integer,
Upper integer,
Frequency integer
)
```

Primary Key(s): **RecommendationID**

Functional Dependencies :
RecommendationID =>lower,upper, frequency

Normal Form :
A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :
We split this into a separate entity so as to increase the ease of access for alerts.

Alerts

Description
This table holds all the Alerts that have been generated due to various reasons

Attributes :
AlertID,UserID,DateOfAlert,AlertType, AlertMessage,ObservationType

SQL :

```
Create table alerts
(
        AlertID integer NOT NULL, -- (AUTO INCREMENTED BY 1)
        Userid varchar (3),
        DateofAlert date,
        Alerttype varchar(20),
        Alertmessage varchar(50),
        Observationtype varchar(20)
)
```

Primary Key(s); **AlertID**

Functional Dependencies :
AlertID => UserID,DateOfAlert,AlertType, AlertMessage, ObservationType

Normal Form :

A database table is in **BCNF** if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Here all dependencies are on the candidate keys.

Justifications :

This is inserted into on 2 occasions :1. Upon outside-the-limit-observations being added to the observations table. 2. Upon a User who is being “Less that frequent” logs in. The View Alerts Page pulls from this table to display.

Constraints that couldn’t be implemented in the database

1. Alerts :
   The actual generation of the alerts had to happen in the application code as there was a requirement to store some information in variables. So to check for discrepancies the observation data and the dates all needed to be stored o that they could be compared and accordingly inserts were made into the Alerts Table.

2. HealthSupporter can only access Patients information after the authorization date:
   This constraint had to be implemented in the application code in the form of a query as it is an access constraint and not a data level constraint. The query utilized in the application code made use of the AuthorizationDate while attempting to access the “Supportee’s” data.

3. The overriding of the General Recommendation by the Specific Recommendation :
   This part also had to happen in the application code as multiple recommendations could apply to the same patient in the form of both general recommendations as well as specific recommendations. The application code had to check for a specific recommendation associated with a patient before searching for general one by going for the associated disease.

4. Alert Dismissal :
   This is also a UI based system which had to be handled in the application Code. We needed to have a conditional as alerts could only be deleted in specific scenarios so we required if statements to check.

Queries

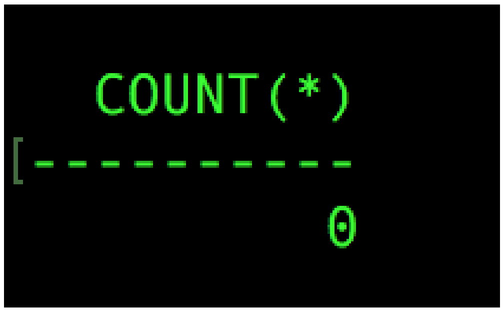
Note: All the result sets have been generated based on the sample data.

1. List the number of health supporters that were authorized in the month of September 2016 by patients suffering from heart disease.

Query:

select count(*) from supportedby where patientid in (select s.userid from sickpatients s, disease d where s.diseaseid = d.diseaseid and d.diseasename = 'Heart Disease') and authorizationdate >= '01-SEP-16' AND authorizationdate <= '30-SEP-16';

Result:

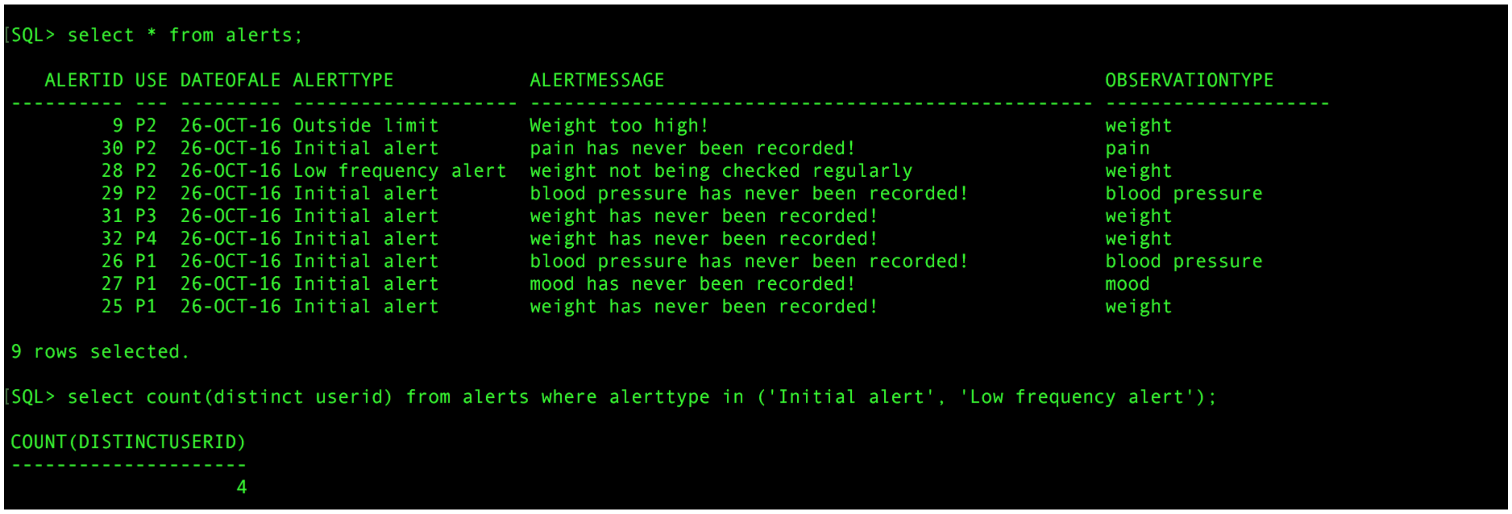


```
  COUNT(*)
[----------
         0
```

2. Give the number of patients who were not complying with the recommended frequency of recording observations.

Assumption: People having no records in the observation table will trigger an alert for not complying with the recommended frequency of recording observations. This alert is called an '**Initial Alert'**

Query: select count (distinct userid) from alerts where alerttype in ('Initial alert', 'Low frequency alert');



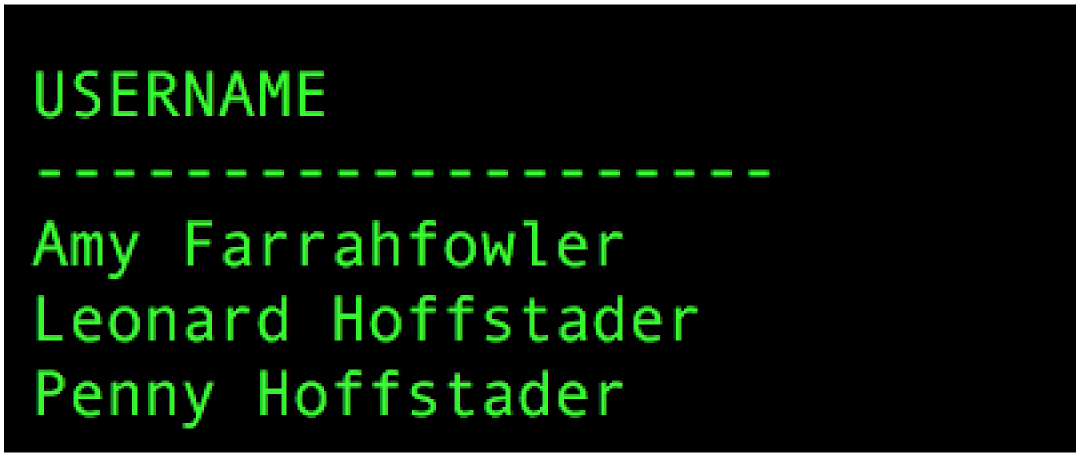
```
[SQL> select * from alerts;

   ALERTID USE DATEOFALE ALERTTYPE            ALERTMESSAGE                                       OBSERVATIONTYPE
---------- --- --------- -------------------- -------------------------------------------------- --------------------
         9 P2  26-OCT-16 Outside limit        Weight too high!                                   weight
        30 P2  26-OCT-16 Initial alert        pain has never been recorded!                      pain
        28 P2  26-OCT-16 Low frequency alert  weight not being checked regularly                 weight
        29 P2  26-OCT-16 Initial alert        blood pressure has never been recorded!            blood pressure
        31 P3  26-OCT-16 Initial alert        weight has never been recorded!                    weight
        32 P4  26-OCT-16 Initial alert        weight has never been recorded!                    weight
        26 P1  26-OCT-16 Initial alert        blood pressure has never been recorded!            blood pressure
        27 P1  26-OCT-16 Initial alert        mood has never been recorded!                      mood
        25 P1  26-OCT-16 Initial alert        weight has never been recorded!                    weight

9 rows selected.

[SQL> select count(distinct userid) from alerts where alerttype in ('Initial alert', 'Low frequency alert');

COUNT(DISTINCTUSERID)
---------------------
                    4
```

3. List the health supporters who themselves are patients.

Query:

select username from patients where userid in (select distinct healthsupporterid from supportedby where healthsupporterid in (select userid from sickpatients UNION select userid from wellpatients));

Result:



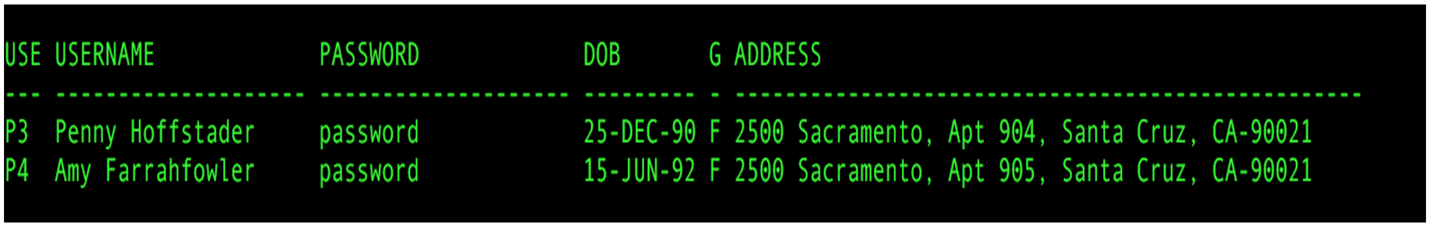
```
USERNAME
--------------------
Amy Farrahfowler
Leonard Hoffstader
Penny Hoffstader
```

<u>4. List the patients who are not ‘sick’.</u>

Query:

Select * from patients where userid in (select userid from wellpatients);

Result:



```
USE USERNAME             PASSWORD             DOB       G ADDRESS
--- -------------------- -------------------- --------- - --------------------------------------------------
P3  Penny Hoffstader     password             25-DEC-90 F 2500 Sacramento, Apt 904, Santa Cruz, CA-90021
P4  Amy Farrahfowler     password             15-JUN-92 F 2500 Sacramento, Apt 905, Santa Cruz, CA-90021
```

<u>5. How many patients have different observation time and recording time (of the observation).</u>

Query:

select count (distinct patientid) from observation where observationtime <> recordingtime;

Result:

```
     COUNT
----------
         1
```

UI Components and flow

The first window of the UI gives the user an option to either sign up or login. Should the user choose to signup, the UI accepts information like UserID, Username address etc. and allows the user to sign up. This module creates an associated entry in the database. The login functionality allows the user to login using his/her userid and password. The application will then validate the information with the database and if the userid and password match, the user will be able to login and view the homepage.

The homepage has the following options displayed:

1. Enter Observation Data - This section allows the user to enter his/her own observations.

2. View Observation Data – This section has two functionalities:
   - View own observation data: The person who is logged in can view his/her own observation data.
   - View Supportee's observation data: The person who is logged in can view the observation data for the users that he/she is supporting.

3. View Alerts – This section has the following functionalities:
   - View own alerts: The person who is logged in can view his/her own alerts.
   - View Supportee's alerts: The person who is logged in can view the alerts for all the users that he/she is supporting.
   - Delete Supportee's alerts: The person who is logged in can clear the alerts for the users that he/she is supporting.

4. Update Patients – This section has the following functionalities:
   - Update the patient information – The person logged in can update his/her username, password and address information.
   - Delete patient information – The person logged in can delete his/her own account.

5. Add Specific Recommendations – The health supporter can use this option to add specific health recommendations for the patient. Both primary and secondary health supporters can avail of this option.