Project Report
on

Personal Health Management

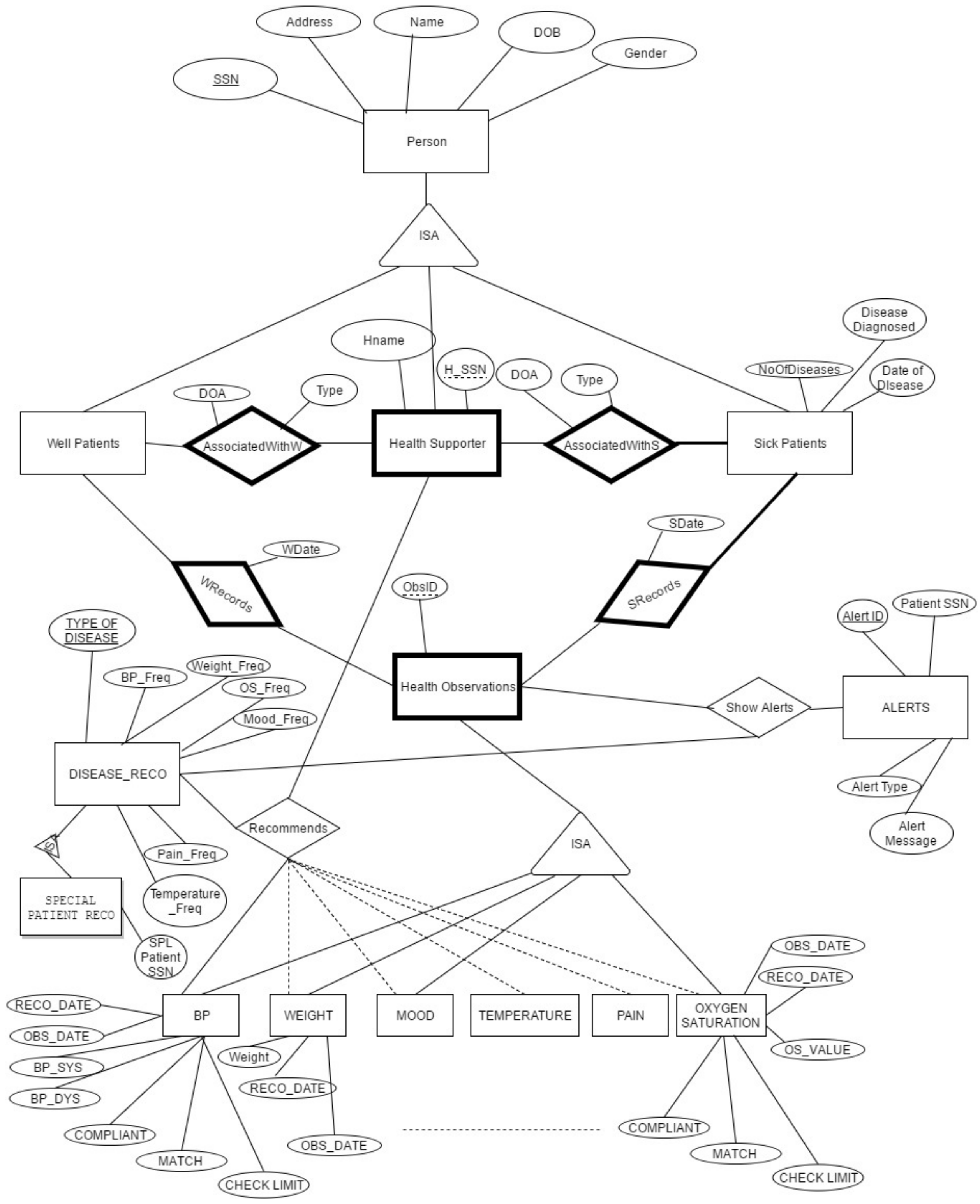
Team # -
Revanth Pathuri (rpathur@ncsu.edu)
Pranav Firake (ppfirake@ncsu.edu)
Vikas Pandey (vrpandey@ncsu.edu)
Abhishek Dudi (adudi@ncsu.edu)

Problem Statement

The purpose of Personal Health Management System is to provide a Database management system for health care domain to automate some of the processes including maintaining the patient data, updating it and include recommending the patient about their health parameters. This also includes health supporters and doctors who would monitor patients and guide them.

The goal of this application is to assist people (both healthy and otherwise) track and manage health status information. People can track their "observations about certain health indicators and activities of daily living" whose patterns can be reflective of their health status. Tracking such information can help create alerts when something is wrong or when patients become slack about health management. And it is becoming increasingly common, the users of health supporters are being encouraged to assist, motivate and encourage patients to comply with directives. Consequently, a useful personal health management application will help with recording and monitoring specific health indicators, raise alerts by notifying patient and health supporter (where appropriate)

1. Entity Relationship Diagram



2. Entities & Functional Dependencies

The following are the List of Entity and Relationship Types, their constraints. For constraints we have followed convention to underline the primary keys and dotted-underline for foreign keys.
Persons will have ISA to Well, Sick, Health Supporters.

1. **Persons**: (<u>Ssn</u>, Password, Name, Address, Dob, Gender)

   Functional Dependency:
   Ssn ->Password, Name, Address, Dob, Gender
   Primary key: Ssn
   Not null: Ssn, Password, Name

   Description: Lists of all the persons in the database including well patients, sick patients, doctors, health supporters. This entity is a superclass for all of the persons described above.

2. **SickPerson**: (Ssn, Disease, Diagnosed)
   Functional Dependency:
   Ssn, Disease -> Diagnosed
   Primary key: (Ssn, Disease)
   Not null: Diagnosed

   Description: Lists of all sick persons. This is a subclass of the Persons entity, has all the data corresponding to the sick person, the disease he is effected with and the date on which he is diagnosed with that disease.

3. **WellPerson: (**Ssn)
   Primary key: Ssn
   Description: Lists of all persons who are well patients. This is a subclass of the Persons entity, has all the ssn corresponding to the well persons in the persons entity.

4. **Doctor:** (DocId, P_Ssn)
   Primary key: (DocId, P_Ssn)
   Description: The doctor is the subclass of the Persons entity, has a DocId and patient's Ssn has the capability for recommending patients certain health monitoring techniques.

5. **Health_Supporter:** (H_Ssn, P_Ssn, DoA, Type)
   Primary key: (H_Ssn, P_Ssn)
   Functional Dependency:
   (H_Ssn, P_Ssn) -> DoA, Type
   Foreign key: H_Ssn, P_Ssn (References Persons)
   Not null: H_Ssn, P_Ssn
   Description: This table consists of all the health supporters in the system, all the health supporters can themselves be patients or persons. Health supporters are part of the superclass Persons entity. The type in the table specifies whether the supporter is of primary or secondary. The DoA is the date of authorization for that health supporter by the patient.

6. **Disease_Reco:**
   (Type**,** Weight, Os, Bp_Sys, Bp_Dys, Pain, Mood, Temp)
   Primary key: Type
   Functional Dependency:
   Type ->Weight, Os, Bp_Sys, Bp_Dys, Pain, Mood, Temp
   Description: This table is a list of all the frequency values specific to a particular disease. Each attribute would be defined with an appropriate low value and high value or a specific value recommended by the doctor.

7. **Spl_Patient_Reco:**
   (Ssn, Weight, Os, Bp_Sys, Bp_Dys, Pain, Mood, Temp**)**
   Primary key: Ssn
   Functional Dependency:
   Ssn ->Weight, Os, Bp_Sys, Bp_Dys, Pain, Mood, Temp
   Description: This table is a list of all the frequency values specific to a particular patient. Each attribute would be defined with an appropriate low value and high value or a specific value recommended by the doctor. These values are of higher priority if specified, compared to the disease specific recommendations.

8. **BP:**
   (Obsid, Reco_Date, Obs_Date, Bp_Sys, Bp_Dys, Compliant, Match, Checklimit**)**
   Primary key: Obsid
   Functional Dependency:
   Obsid->Reco_Date, Obs_Date, Bp_Sys, Bp_Dys, Compliant, Match, Checklimit
   Description: This table lists all the blood pressure recordings of all the patients, if they recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time are same, and the checklimit will be set to true if it falls in the range as recommended.

9. **MOOD (**Obsid, Reco_Date, Obs_Date, Mood, Compliant, Match, Checklimit**)**
   Primary key: Obsid
   Functional Dependency:
   Obsid->Reco_Date, Obs_Date, Mood, Compliant, Match, Checklimit
   Description: This table lists all the mood recordings of all the patients, if the recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time

are same and the checklimit will be set to true if it falls in the range as recommended.

10. **WEIGHT:**

(Obsid, Reco_Date, Obs_Date, Weight, Compliant, Match, Checklimit**)**

Primary key: Obsid

Functional Dependency:

Obsid->Reco_Date, Obs_Date, Weight, Compliant, Match, Checklimit

Description: This table lists all the weight recordings of all the patients, if the recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time are same and the checklimit will be set to true if it falls in the range as recommended.

11. **OS:**

(Obsid, Reco_Date, Obs_Date, Os, Compliant, Match, Checklimit**)**

Primary key: Obsid

Functional Dependency:

Obsid->Reco_Date, Obs_Date, Os, Compliant, Match, Checklimit

Description: This table lists all the oxygen saturation recordings of all the patients, if the recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time are same and the checklimit will be set to true if it falls in the range as recommended.

12. **TEMP:**
(Obsid, Reco_Date, Obs_Date, Mood, Compliant, Match, checklimit**)**
Primary key: Obsid
Functional Dependency:
Obsid->Reco_Date, Obs_Date, Mood, Compliant, Match, checklimit
Description: This table lists all the temperature recordings of all the patients, if the recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time are same and the checklimit will be set to true if it falls in the range as recommended.

13. **PAIN:**
(Obsid, Reco_Date, Obs_Date, Mood, Compliant, Match, checklimit**)**
Primary key: Obsid
Functional Dependency:
Obsid->Reco_Date, Obs_Date, Mood, Compliant, Match, checklimit
Description: This table lists all the pain recordings of all the patients, if the recording frequency is compliant with the recommended frequency then the complaint will be set to true and the match will be true if the recorded time and the observation time are same and the checklimit will be set to true if it falls in the range as recommended.

14. **ALERTS: (**Alert_Id, Ssn, Alert_Msg, Type**)**
Primary key: Alert_Id
Functional Dependency:
Alert_Id **->** Ssn, Alert_Msg, Type
Description: This table lists all the alert messages of all the patients, if the recording frequency is not compliant with the recommended frequency then the type of activity alert will be set to compliant. If the recorded values are out of the range, then type will be set to limit alert.

3. Relationships:

Binary Relations

15. **RECORDS** (Ssn, Obsid)
Primary key: Obsid
Functional Dependency:
Obsid->Ssn
Description: This table is a list of all the recorded observations. Primary key would be Ssn and Obsid, because a person might have multiple observations. This is a relational table, where persons table and observations table participate.

4. ER Model

1. PERSONS

```
(
        SSN: INTEGER,
        PASSWORD: VARCHAR (32),
        NAME: VARCHAR (32),
        ADDRESS: VARCHAR (64),
        DOB: DATE,
        GENDER: VARCHAR (6),
        PRIMARY KEY (SSN)
)
```

2. SICKPERSON

```
(
        SSN: INTEGER,
        DISEASE: VARCHAR (32),
        DIAGNOSED: DATE,
        PRIMARY KEY(SSN, DISEASE),
        FOREIGN KEY (SSN) REFERENCES PERSONS
)
```

3. WELLPERSON

```
(
        SSN INTEGER,
        FOREIGN KEY (SSN) REFERENCES PERSONS
)
```

4. DOCTOR

```
(
        DOCID: INTEGER,
        P_SSN INTEGER,
        FOREIGN KEY (DOCID) REFERENCES PERSONS,
        FOREIGN KEY (P_SSN) REFERENCES PERSONS,
        PRIMARY KEY (DOCID, P_SSN)
)
```

5. DISEASE_RECO

```
(
    TYPE PRIMARY KEY,
    WEIGHT_FREQ INTEGER,
    WEIGHT_LOW INTEGER,
    WEIGHT_HIGH INTEGER,
    BP_FREQ INTEGER,
    BP_SYS_LOW INTEGER,
    BP_SYS_HIGH INTEGER,
    BP_DYS_LOW INTEGER,
    BP_DYS_HIGH INTEGER,
    OS_FREQ INTEGER,
    OS_LOW INTEGER,
    OS_HIGH INTEGER,
    PAIN_FREQ INTEGER,
    PAIN_HIGH INTEGER,
    MOOD_FREQ INTEGER,
    MOOD_VAL VARCHAR (32),
    TEMP_FREQ INTEGERS
    TEMP_LOW INTEGER
    TEMP_HIGH INTEGER
)
```

6. SPCL-Patient-Reco

```
(
SSN INTEGER,
WEIGHT_FREQ INTEGER,
WEIGHT_LOW INTEGER,
WEIGHT_HIGH INTEGER,
BP_FREQ INTEGER,
BP_SYS_LOW INTEGER,
BP_SYS_HIGH INTEGER,
BP_DYS_LOW INTEGER,
BP_DYS_HIGH INTEGER,
OS_FREQ INTEGER,
OS_LOW INTEGER,
OS_HIGH INTEGER,
PAIN_FREQ INTEGER,
PAIN_HIGH INTEGER,
MOOD_FREQ INTEGER,
MOOD_VAL VARCHAR (32),
TEMP_FREQ INTEGERS
TEMP_LOW INTEGER
TEMP_HIGH INTEGER
)
```

7. RECORDS

```
(
SSN INTEGER,
OBSID INTEGER,
PRIMARY KEY (SSN, OBSID),
FOREIGN KEY (SSN) REFERENCES PERSONS,
FOREIGN KEY (OBSID) REFERENCES OBSERVATIONS)
)
```

8. BP

```
(
OBS_ID INTEGER,
RECO_DATE DATE,
BP_SYS INTEGER,
BP_DYS INTEGER,
OBS_DATE DATE,
COMPLIANT VARCHAR (5),
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

9. OS

```
(
OBS_ID INTEGER,
RECO_DATEDATE,
OBS_DATE DATE,
OS INTEGER,
COMPLIANT VARCHAR (5),
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

10. TEMP

```
(
OBS_ID INTEGER,
RECO_DATE DATE,
OBS_DATE DATE,
COMPLIANT VARCHAR (5),
TEMP INTEGER,
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

11. MOOD

```
(
OBS_ID INTEGER,
RECO_DATE DATE,
OBS_DATE DATE,
MOOD VARCHAR (32),
COMPLIANT VARCHAR (5),
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

12. PAIN

```
(
OBS_ID INTEGER,
RECO_DATE DATE,
OBS_DATE DATE,
PAIN INTEGER,
COMPLIANT VARCHAR (5),
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

13. WEIGHT

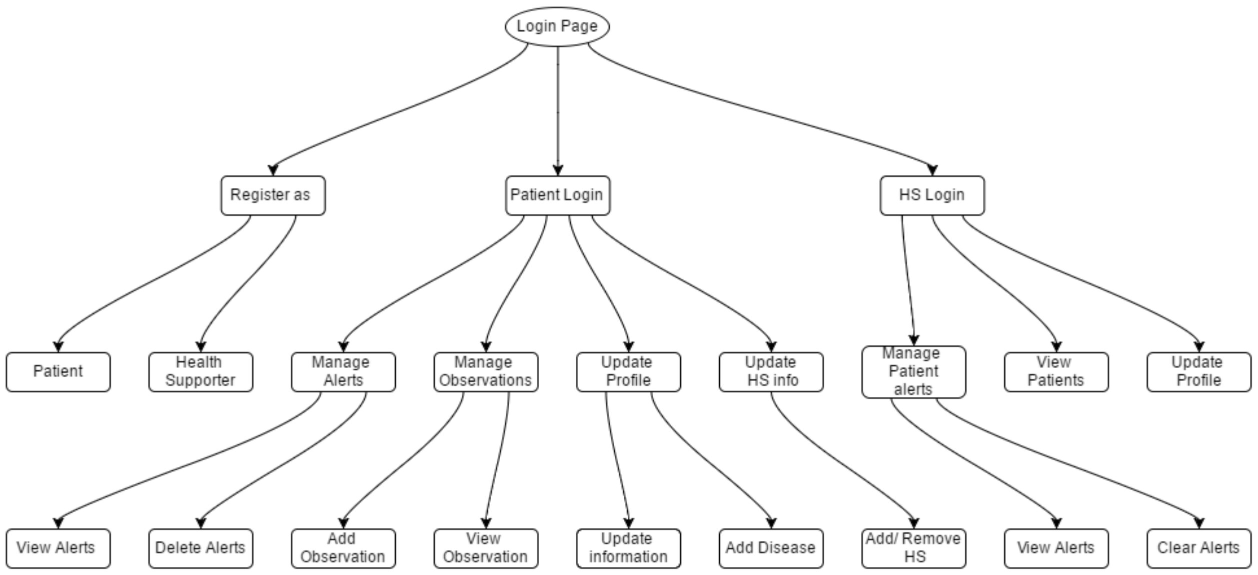
```
(
OBS_ID INTEGER,
RECO_DATE DATE,
OBS_DATE DATE,
WEIGHT INTEGER,
COMPLIANT VARCHAR (5),
MATCH VARCHAR (5),
FOREIGN KEY OBS_ID REFERENCES RECORDS
)
```

14. ALERTS

```
(
ALERT_ID INTEGER,
SSN INTEGER,
ALERT_MSG VARCHAR (32),
ALERT_TYPE VARCHAR (7),
PRIMARY KEY ALERT_ID,
FOREIGN KEY SSN REFERENCES PERSONS
)
```

5. **Constraints**:
    - Here we have considered a Person as super entity of every human being in scenario. Person can be a Sick Patient, Well Patient, Health supporter or Doctor.
    - Person can be sick only if he/she has one or more diseases.
    - We have assumed Doctor be super handler of system and he/she can edit or set recommended frequencies of various diseases and also can add specific recommendations for specific user with specific frequencies to check.
    - Person can record his/her observation with given health parameters from registration page.
    - Sick patient can have up to 2 Health supporters one being primary and other being secondary.
    - Disease Reco table will have data of recommended frequencies for specific diseases and it will also include recommended frequencies for general or well patients., he/she must be a Person entity.
    - Every person before registering as Health Supporter or Doctor .

6. Program Flow



7. Report Queries

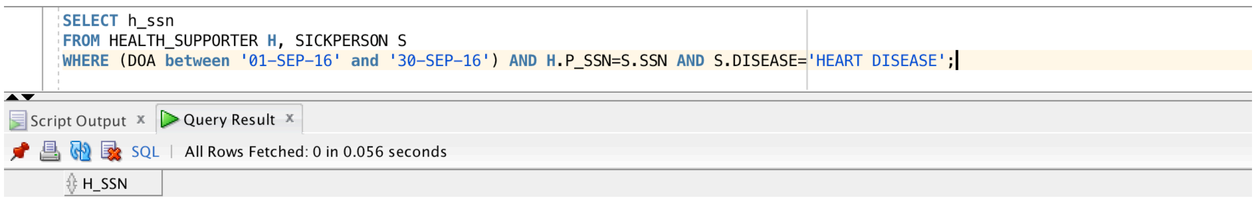
- List the number of health supporters that were authorized in the month of September 2016 by patients suffering from heart disease.

Query:

```
SELECT h_ssn
FROM HEALTH_SUPPORTER H, SICKPERSON S
WHERE (DOA between '01-SEP-16' and '30-SEP-16') AND
H.P_SSN=S.SSN AND S.DISEASE='HEART DISEASE';
```

Result:

NULL



```
SELECT h_ssn
FROM HEALTH_SUPPORTER H, SICKPERSON S
WHERE (DOA between '01-SEP-16' and '30-SEP-16') AND H.P_SSN=S.SSN AND S.DISEASE='HEART DISEASE';
```

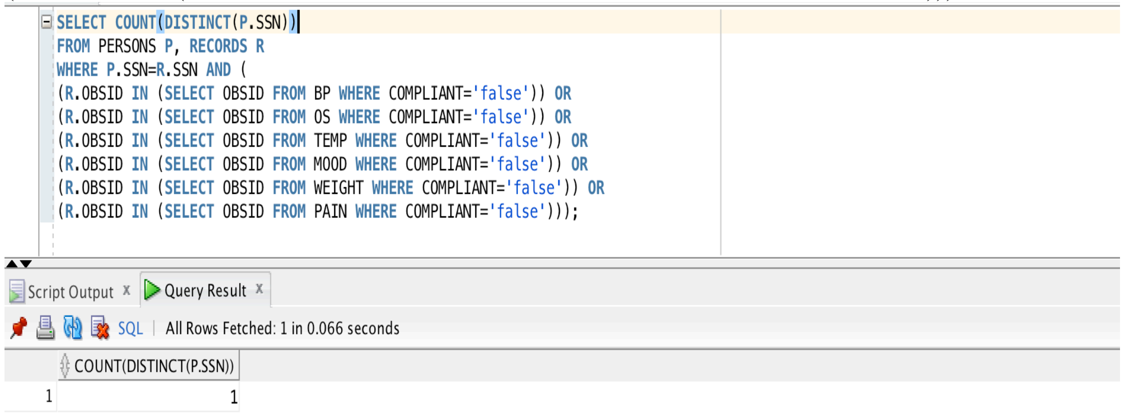
Script Output × | Query Result ×

SQL | All Rows Fetched: 0 in 0.056 seconds

```
H_SSN
```

- Give the number of patients who were not complying with the recommended frequency of recording observations.
  Query:

```
SELECT COUNT(DISTINCT(P.SSN))

FROM PERSONS P, RECORDS R

WHERE P.SSN=R.SSN AND (

(R.OBSID IN (SELECT OBSID FROM BP WHERE COMPLIANT='false')) OR

(R.OBSID IN (SELECT OBSID FROM OS WHERE COMPLIANT='false')) OR

(R.OBSID IN (SELECT OBSID FROM TEMP WHERE COMPLIANT='false')) OR

(R.OBSID IN (SELECT OBSID FROM MOOD WHERE COMPLIANT='false')) OR

(R.OBSID IN (SELECT OBSID FROM WEIGHT WHERE COMPLIANT='false')) OR

(R.OBSID IN (SELECT OBSID FROM PAIN WHERE COMPLIANT='false')));
```



```
SELECT COUNT(DISTINCT(P.SSN))
FROM PERSONS P, RECORDS R
WHERE P.SSN=R.SSN AND (
(R.OBSID IN (SELECT OBSID FROM BP WHERE COMPLIANT='false')) OR
(R.OBSID IN (SELECT OBSID FROM OS WHERE COMPLIANT='false')) OR
(R.OBSID IN (SELECT OBSID FROM TEMP WHERE COMPLIANT='false')) OR
(R.OBSID IN (SELECT OBSID FROM MOOD WHERE COMPLIANT='false')) OR
(R.OBSID IN (SELECT OBSID FROM WEIGHT WHERE COMPLIANT='false')) OR
(R.OBSID IN (SELECT OBSID FROM PAIN WHERE COMPLIANT='false')));
```

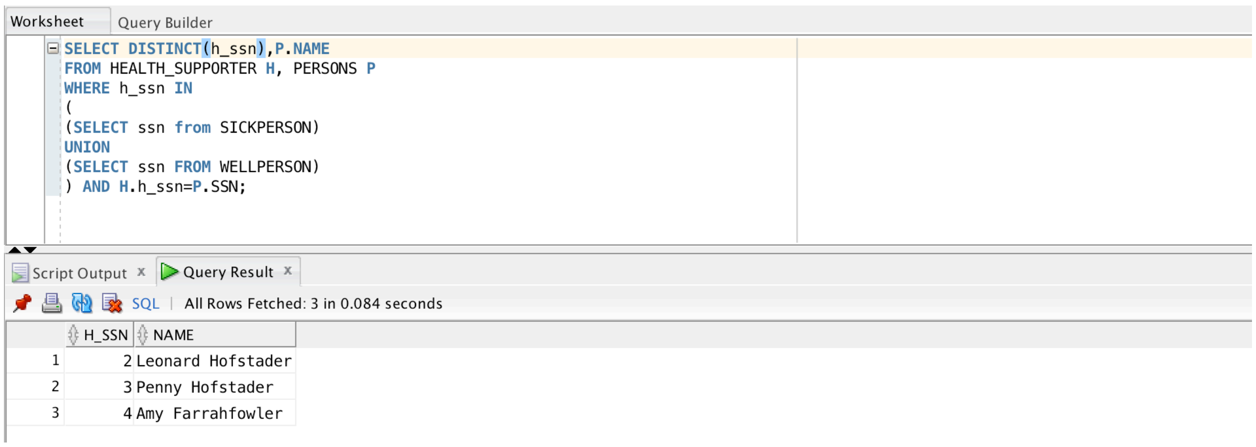
All Rows Fetched: 1 in 0.066 seconds

```
   COUNT(DISTINCT(P.SSN))
1                        1
```

Result:
1

- List the health supporters who themselves are patients.
Query:

```
SELECT DISTINCT(h_ssn),P.NAME
FROM HEALTH_SUPPORTER H, PERSONS P
WHERE h_ssn IN
(
(SELECT ssn from SICKPERSON)
UNION
(SELECT ssn FROM WELLPERSON)
) AND H.h_ssn=P.SSN;
```



```
SELECT DISTINCT(h_ssn),P.NAME
FROM HEALTH_SUPPORTER H, PERSONS P
WHERE h_ssn IN
(
(SELECT ssn from SICKPERSON)
UNION
(SELECT ssn FROM WELLPERSON)
) AND H.h_ssn=P.SSN;
```

All Rows Fetched: 3 in 0.084 seconds

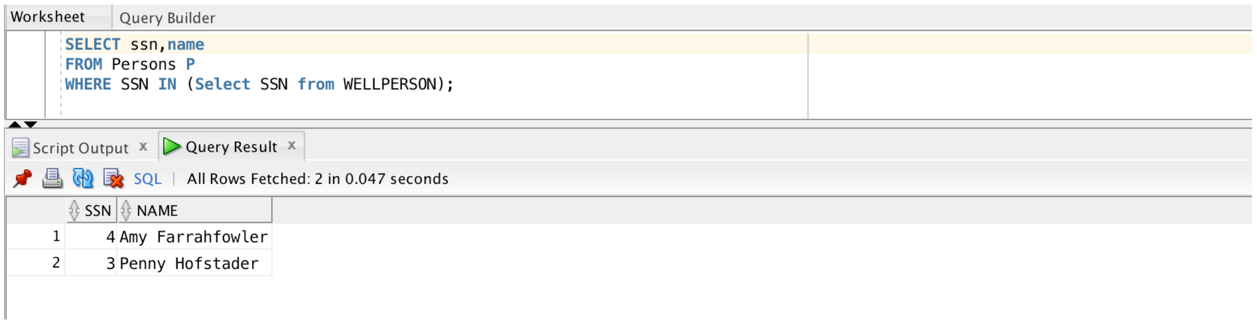
	H_SSN	NAME
1	2	Leonard Hofstader
2	3	Penny Hofstader
3	4	Amy Farrahfowler

- List the patients who are not 'sick'.

Query:

```
SELECT ssn, name
FROM Persons P
WHERE SSN IN (Select SSN from WELLPERSON);
```

Result:



```
SELECT ssn,name
FROM Persons P
WHERE SSN IN (Select SSN from WELLPERSON);
```

All Rows Fetched: 2 in 0.047 seconds

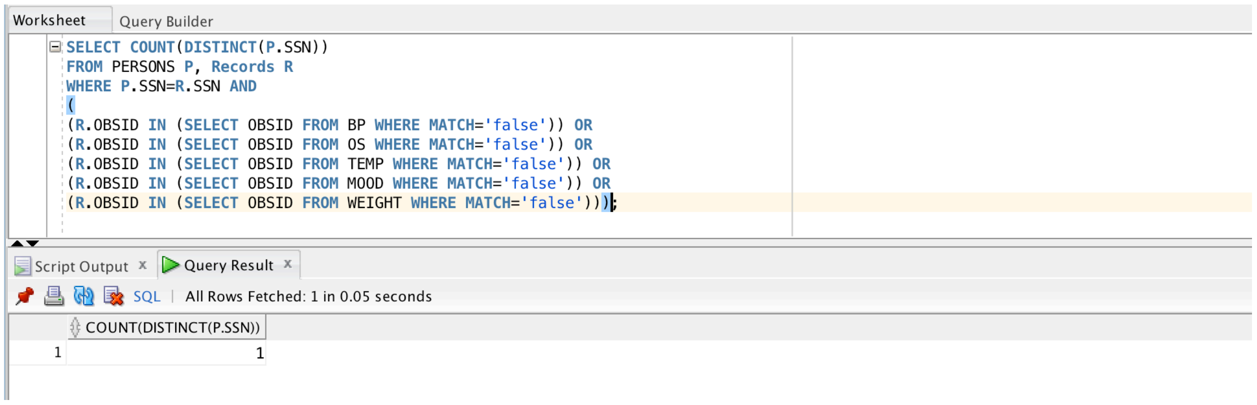
```
   SSN NAME
1    4 Amy Farrahfowler
2    3 Penny Hofstader
```

- How many patients have different observation time and recording time (of the observation)

  Query:

```
SELECT COUNT(DISTINCT(P.SSN))
FROM PERSONS P, Records R
WHERE P.SSN=R.SSN AND
(
(R.OBSID IN (SELECT OBSID FROM BP WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM OS WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM TEMP WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM MOOD WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM WEIGHT WHERE MATCH='false')) OR
 (R.OBSID IN (SELECT OBSID FROM PAIN WHERE MATCH='false'))
);
```

  Result:



```
SELECT COUNT(DISTINCT(P.SSN))
FROM PERSONS P, Records R
WHERE P.SSN=R.SSN AND
(
(R.OBSID IN (SELECT OBSID FROM BP WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM OS WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM TEMP WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM MOOD WHERE MATCH='false')) OR
(R.OBSID IN (SELECT OBSID FROM WEIGHT WHERE MATCH='false')));
```

All Rows Fetched: 1 in 0.05 seconds

```
  COUNT(DISTINCT(P.SSN))
1                      1
```