Note: For the following queries, assume that **$variable** is a placeholder for the corresponding variable described by the name. Also assume that **variable\_id.nextval** refers to a sequence generated & used.

Note: For each functionality/feature, consider the listed queries to be done in that order.

**User Module**

The user module used 7 main functions:

1. **validatePersonID**
   1. “SELECT \* FROM persons WHERE person\_id= $person\_id”
2. **validateEmail**
   1. “SELECT \* FROM persons WHERE email = $email”
3. **removeSubscriptions**
   1. “SELECT \* FROM subscriptions WHERE person\_id= $person\_id”
   2. “DELETE FROM subscriptions WHERE person\_id = $person\_id”
4. **removeUser**
   1. “DELETE FROM users WHERE user\_name = $username”
5. **removeUserFromPersonID**
   1. “DELETE FROM users WHERE person\_id = $person\_id”
6. **removePerson**
   1. “DELETE FROM persons WHERE person\_id = $person\_id”
7. **validateUsername**
   1. “SELECT \* FROM users WHERE user\_name = $username”

To add a new person:

1. **validateEmail** was used to ensure the unique constraint was not violated
2. “INSERT INTO persons VALUES (person\_id.nextval, $first\_name, $last\_name, $address, $email, $phone\_number)”

To update a person:

1. **validatePersonID** was used to ensure the an existing person id was entered before updating a person
2. “UPDATE persons SET first\_name = $first\_name, last\_name = $last\_name, address = $address, email = $email, phone = $phone WHERE person\_id = $person\_id”
   1. Only the fields that were entered in the form are included in the query

To remove a person:

1. **validatePersonID** was used to ensure the an existing person id was entered before removing a person
2. **removeSubscriptions** was used to remove any subscriptions the person had before removing the person ensure the integrity constraint was not violated
3. **removeUserFromPersonID** was used to remove any user accounts associated with the person to ensure the integrity constraint was not violated
4. **removePerson** was used to remove the person

To display the contents of the persons table:

1. The function **get\_all\_persons** was used to first retrieve all the records in the person table – “SELECT \* FROM persons”

To add a new user account:

1. **validateUsername** was used to ensure the unique constraint was not violated
2. **validatePersonID** was used to ensure the an existing person id was entered
3. “INSERT INTO users VALUES ($username, $password, $role, $person\_id, to\_date($date\_registered, 'dd-mm-yyyy hh24:mi:ss'))”

To update a user account:

1. **validateUsername** was used to ensure an existing username was entered before updating
2. “UPDATE users SET password = $password, role = $role WHERE user\_name = $username”
   1. Only the fields that were entered in the form are included in the query

To remove a user account:

1. **validateUsername** was used to ensure an existing username was entered before removing
2. **removeUser** was used to remove the user account

To display the contents of the users table:

1. The function **get\_all\_users** was used to first retrieve all the records in the users table – “SELECT \* FROM users”

**Sensor Module**

The sensor module used 3 main functions:

1. **validateSensor**
   1. “SELECT \* FROM sensors WHERE sensor\_id = $sensor\_id”
2. **removeSubscriptionsFromSensorID**
   1. “DELETE FROM subscriptions WHERE sensor\_id = $sensor\_id”
3. **removeSensor**
   1. “DELETE FROM sensors WHERE sensor\_id = $sensor\_id”

To add a sensor:

1. “INSERT INTO sensors VALUES ( sensor\_id.nextval, $location, $type, $description)”

To remove a sensor:

1. **validateSensor** was used to ensure an existing sensor id was entered before removing
2. **removeSubscriptionsFromSensorID** was used to remove any subscriptions associated with the sensor to ensure the integrity constraint was not violated
3. **removeSensor** was used to remove the sensor

To display the contents of the sensors table:

1. The function **get\_all\_sensors** was used to first retrieve all the records in the sensors table – “SELECT \* FROM sensors”

**Search Module**

To search:

We did 3 separate queries: one for each of images, audio recordings & scalar data

1. For Images:

“SELECT i.\* FROM images i, sensors s, subscriptions t WHERE t.person\_id = $person\_id AND s.sensor\_id = t.sensor\_id AND i.sensor\_id = s.sensor\_id AND s.sensor\_type = $sensor\_type AND s.location = $sensor\_location AND ( lower(i.description) LIKE $keyword1 OR lower(s.description) LIKE $keyword1 OR lower(i.description) LIKE $keyword2 OR lower(s.description) LIKE $keyword2) AND to\_date(i.date\_created, 'dd-mm-YYYY hh24:mi:ss') >= to\_date(to\_date($start\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss') AND to\_date(i.date\_created, 'dd-mm-YYYY hh24:mi:ss') <= to\_date(to\_date($end\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss')”

1. For audio recordings:

“SELECT a.\* FROM audio\_recordings a, sensors s, subscriptions t WHERE t.person\_id = $person\_id AND s.sensor\_id = t.sensor\_id AND a.sensor\_id = s.sensor\_id AND s.sensor\_type = $sensor\_type AND s.location = $sensor\_location AND ( lower(a.description) LIKE ‘%$keyword1%’ OR lower(s.description) LIKE '%$keyword1%' OR lower(a.description) LIKE ‘%$keyword%’ OR lower(s.description) LIKE ‘%$keyword%’) AND to\_date(a.date\_created, 'dd-mm-YYYY hh24:mi:ss') >= to\_date(to\_date($start\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss') AND to\_date(a.date\_created, 'dd-mm-YYYY hh24:mi:ss') <= to\_date(to\_date($end\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss')”

1. For scalar data:

“SELECT c.id, c.sensor\_id, to\_char(c.date\_created, 'dd/mm/YYYY hh24:mi:ss') as date\_created, c.value FROM scalar\_data c, sensors s, subscriptions t WHERE t.person\_id = $person\_id AND s.sensor\_id = t.sensor\_id AND c.sensor\_id = s.sensor\_id AND s.sensor\_type = $sensor\_type AND s.location = $sensor\_location AND ( lower(s.description) LIKE '%$keyword1%' OR lower(s.description) LIKE '%$keyword2%') AND to\_date(c.date\_created, 'dd-mm-YYYY hh24:mi:ss') >= to\_date(to\_date($start\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss') AND to\_date(c.date\_created, 'dd-mm-YYYY hh24:mi:ss') <= to\_date(to\_date($end\_date, 'YYYY-mm-dd'), 'dd-mm-YYYY hh24:mi:ss')”

* $start\_date & $end\_date are assumed to be in the form YYYY-mm-dd, or example ‘2015-11-23’
* “lower(i.description) LIKE $keyword1 OR lower(s.description) LIKE $keyword1” is done for every keyword and connected via an “OR”