Vincent Sastra
Agnes Lee
Partheeban Bharati

Project Summary

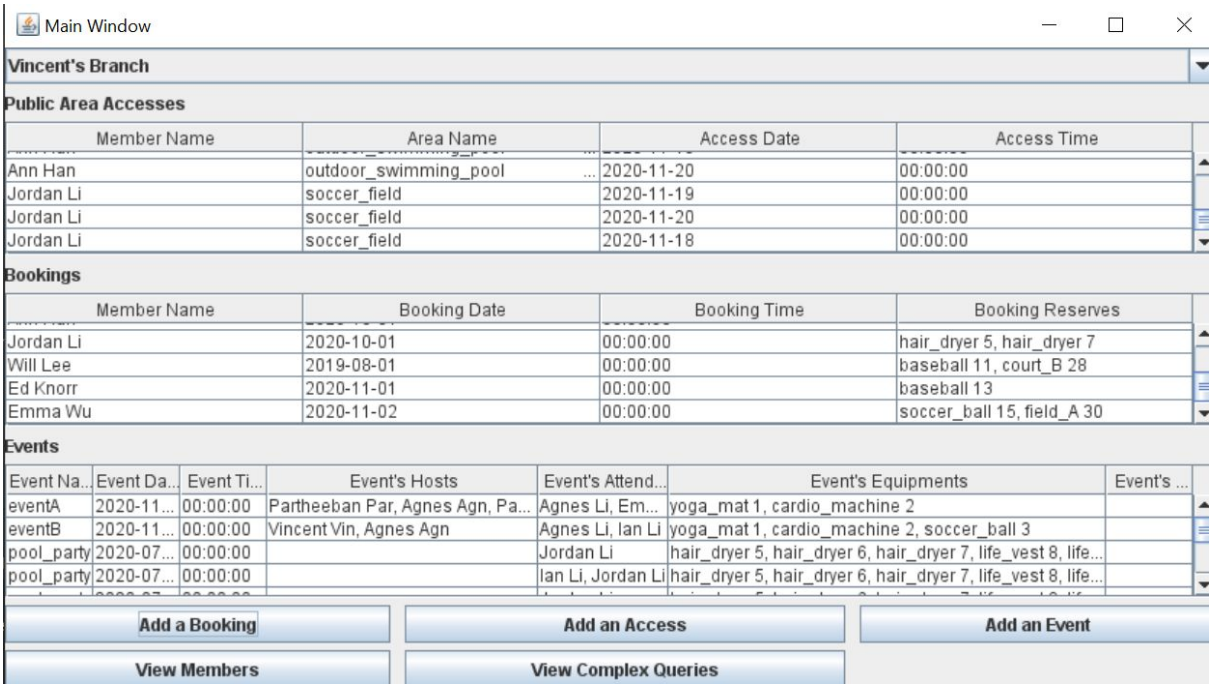
Our project is built on a recreational center DBMS and displayed with a GUI developed by Java Swing. Users can log into the system and select a branch to look at several things that the branch has, such as a list of bookables, a list of members signed up in the branch, a list of areas in the branch that a certain member can access etc. Users can also insert/ delete values (such as make/ delete bookings from the branch) to our program. Our program also delivers some complex querying methods such as displaying members that have accessed a public area before and the number of their total accesses.

Screenshots of the App

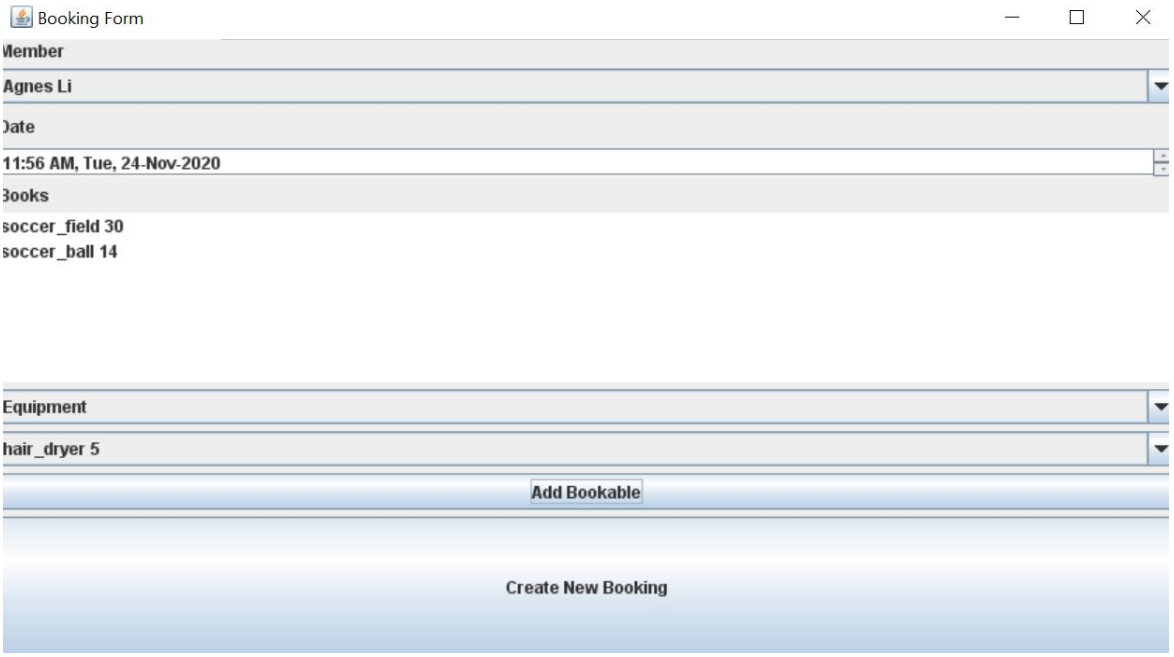
** Our program (the gui part) might run a little bit slow. The loading might take a few seconds. Please be patient with our program. Sorry about it!

Our program floors (round but always down) the timestamp into 15 minute increments. This is by design because it is impractical to record bookings more precisely for scheduling purposes **

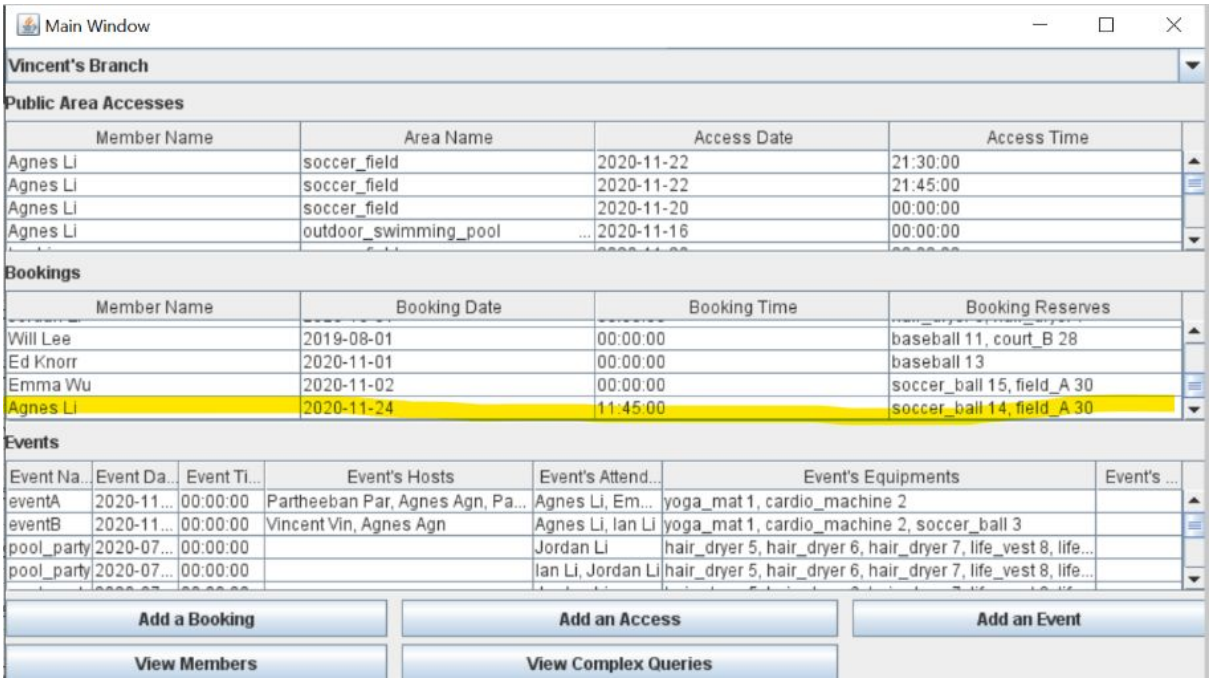
Screenshot 1 - main window of Vincent's branch



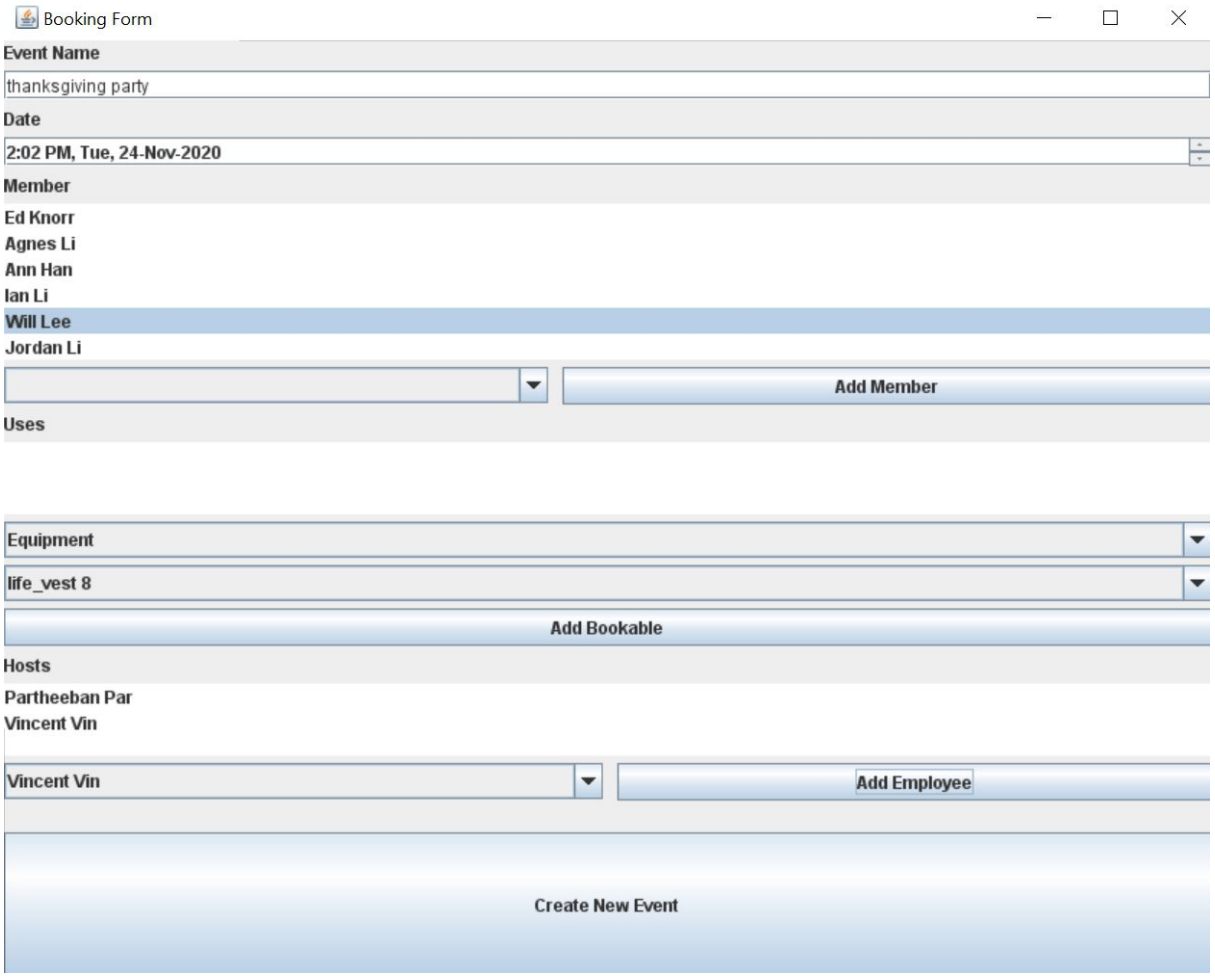
Screenshot 2 - adding a new booking by adding the member and bookables that is associated in a booking. In this case, member Agnes booked soccer ball and soccer field on Nov 24 11:56.



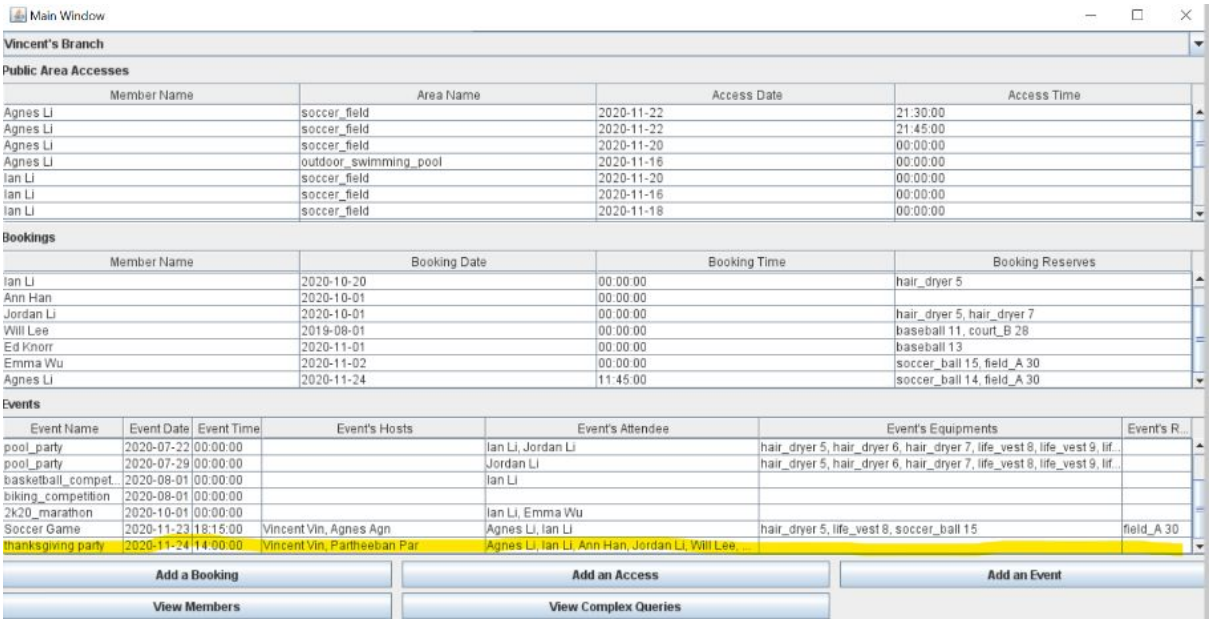
Screenshot 3 - After adding a new booking, the booking showed up in Vincent's branch!



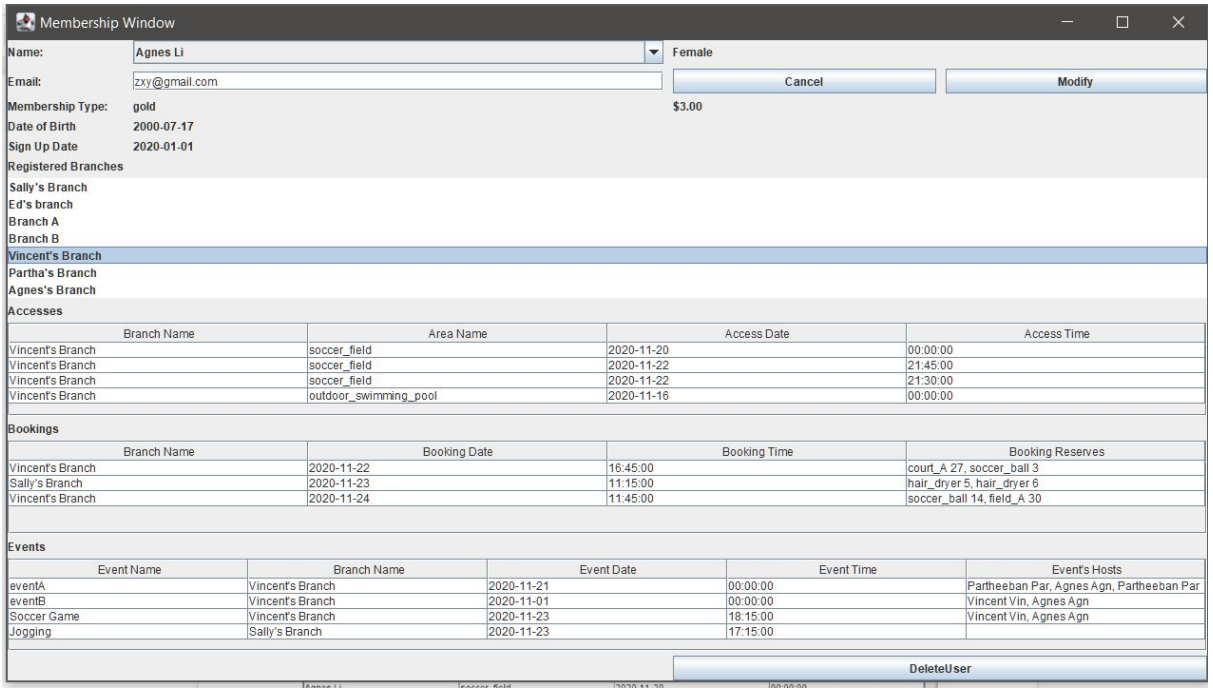
Screenshot 4 - add an event “thanksgiving party” with all members in Vincent’s branch. The addition also records which employees hosted the event and which bookables were used in the event.



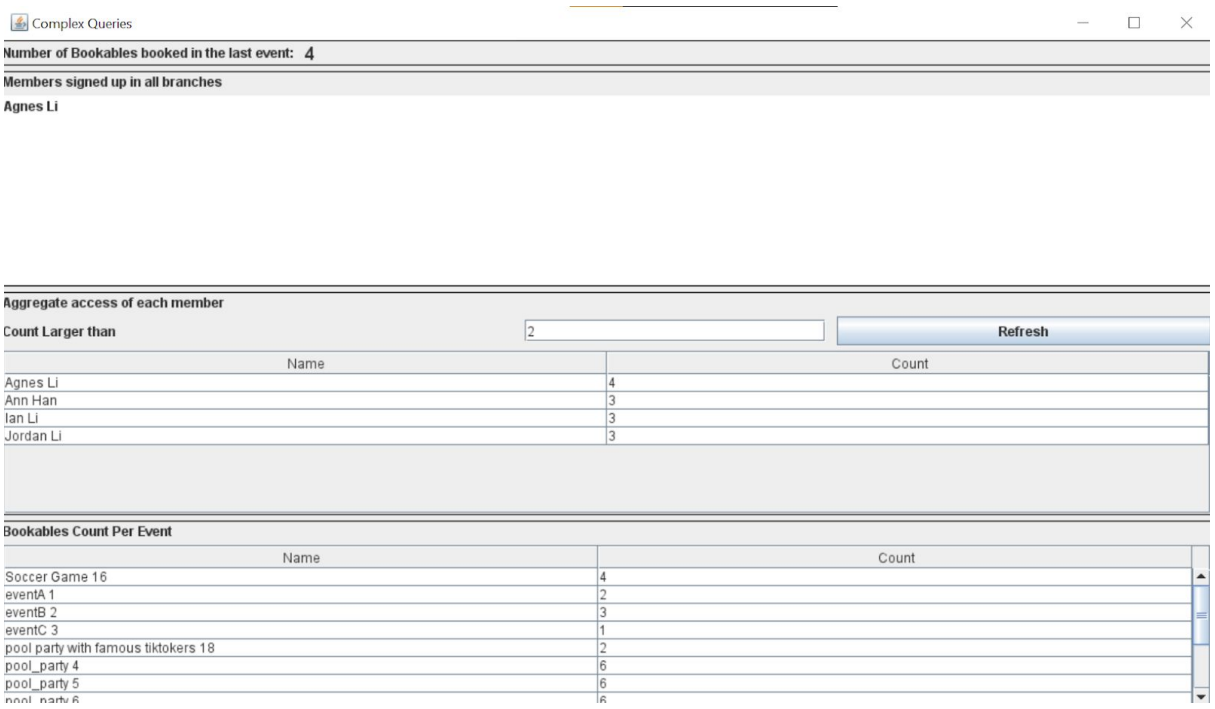
Screenshot 5 - the event is recorded at Vincent's branch!



Screenshot 6 - A view of all the details of an individual member, allowing a change to the email and the option to delete the user from the database



Screenshot 7 - view complex queries: number of bookables in the last event (nested aggregation), members signed up in all branches (division), aggregate access of each member larger than 2 (aggregation with HAVING), bookables count per event (simple aggregation with GROUP BY)



SQL Schema

```
create table BRANCH
(
   BRANCH_ID NUMBER not null
         primary key,
   BRANCH_NAME VARCHAR2(80) not null,
   BRANCH_ADDRESS VARCHAR2(80) not null
)/

create table PUBLIC_AREA
(
   AREA_ID NUMBER not null
         constraint PUBLICAREA_PK
               primary key,
   AREA_NAME CHAR(50) not null,
   AREA_IS_OUTDOOR CHAR not null,
   BRANCH_ID NUMBER
         constraint PUBLIC_AREA_BRANCH_BRANCH_ID_FK
               references BRANCH
)/

create table BOOKABLE
(
   BOOKABLE_ID NUMBER not null
         constraint BOOKABLE_PK
               primary key,
   BOOKABLE_TYPE VARCHAR2(50) not null,
   BOOKABLE_NAME VARCHAR2(50) not null,
   BRANCH_ID NUMBER not null
         constraint BOOKABLE_BRANCH_FK
               references BRANCH
                     on delete cascade
)/

create table EMPLOYEE
(
   EMPLOYEE_ID NUMBER not null
         constraint EMPLOYEE_PK
               primary key,
   EMPLOYEE_FIRST_NAME VARCHAR2(50) not null,
   EMPLOYEE_LAST_NAME VARCHAR2(50) not null,
   EMPLOYEE_EMAIL VARCHAR2(50) not null,
   EMPLOYEE_SIN NUMBER not null,
   EMPLOYEE_DOB DATE not null,
   EMPLOYEE_GENDER VARCHAR2(1),
```

```
    EMPLOYEE_ADDRESS VARCHAR2(100),
    BRANCH_ID NUMBER not null
        constraint EMPLOYEE_BRANCH__FK
            references BRANCH
)/

create unique index EMPLOYEE_EMPLOYEE_EMAIL_UINDEX
    on EMPLOYEE (EMPLOYEE_EMAIL)
/

create unique index EMPLOYEE_EMPLOYEE_SIN_UINDEX
    on EMPLOYEE (EMPLOYEE_SIN)
/

create table EQUIPMENT
(
    BOOKABLE_ID NUMBER not null
        constraint EQUIPMENT_PK
            primary key
        constraint EQUIPMENT_BOOKABLE__FK
            references BOOKABLE
                on delete cascade,
    BOOKABLE_PURCHASED_DATE DATE,
    BOOKABLE_LAST_FIXED_DATE DATE
)/

create table ROOM
(
    BOOKABLE_ID NUMBER not null
        constraint ROOM_PK
            primary key
        constraint ROOM_BOOKABLE__FK
            references BOOKABLE,
    BOOKABLE_LAST_INNOVATION DATE
)/

create table HOSTS
(
    EMPLOYEE_ID NUMBER not null,
    EVENT_ID NUMBER not null,
    primary key (EMPLOYEE_ID, EVENT_ID)
)/

create table PROGRAM
(
    EVENT_ID NUMBER not null
        primary key,
    NAME VARCHAR2(50) not null,
```

```
    START_DATE DATE not null,
    END_DATE DATE
)/

create table PRICE
(
    MEMBERSHIP_TYPE VARCHAR2(20) not null
        primary key,
    MEMBERSHIP_FEE NUMBER
)/

create table EVENT
(
    EVENT_ID NUMBER not null
        primary key,
    NAME VARCHAR2(50) not null,
    EVENT_DATETIME DATE not null,
    BRANCH_ID NUMBER not null
        constraint EVENT_BRANCH_BRANCH_ID_FK
            references BRANCH
)/

create table MEMBER
(
    MEMBER_ID NUMBER not null
        constraint MEMBER_PK
            primary key,
    MEMBER_SINCE DATE,
    MEMBERSHIP_TYPE VARCHAR2(20) not null
        constraint MEMBER_PRICE_MEMBERSHIP_TYPE_FK
            references PRICE
                on delete cascade,
    DOB DATE not null,
    FIRST_NAME VARCHAR2(50) not null,
    LAST_NAME VARCHAR2(50) not null,
    GENDER VARCHAR2(1),
    EMAIL VARCHAR2(50) not null,
    constraint MEMBER_PK_2
        unique (FIRST_NAME, LAST_NAME)
)/

create unique index MEMBER_EMAIL_UINDEX
    on MEMBER (EMAIL)
/

create table SIGN_UP
(
    MEMBER_ID NUMBER not null
```

```
        constraint SIGN_UP_MEMBER_MEMBER_ID_FK
                references MEMBER,
    BRANCH_ID NUMBER not null
        constraint SIGN_UP_BRANCH_BRANCH_ID_FK
                references BRANCH,
    primary key (MEMBER_ID, BRANCH_ID)
)/

create table BOOKING
(
    BOOKING_ID NUMBER not null
        constraint BOOKING_PK
                primary key,
    BOOKING_DATE DATE,
    MEMBER_ID NUMBER not null
        constraint BOOKING_MEMBER_MEMBER_ID_FK
                references MEMBER
                        on delete cascade,
    BRANCH_ID NUMBER not null
        constraint BOOKING_BRANCH_BRANCH_ID_FK
                references BRANCH
)/

create table "ACCESS"
(
    MEMBER_ID NUMBER not null
        constraint ACCESS_MEMBER_MEMBER_ID_FK
                references MEMBER,
    AREA_ID NUMBER not null
        constraint ACCESS_PUBLIC_AREA_AREA_ID_FK
                references PUBLIC_AREA,
    ACCESS_DATE DATE not null,
    constraint ACCESS_PK
        primary key (ACCESS_DATE, MEMBER_ID, AREA_ID)
)/

create table RESERVE
(
    BOOKING_ID NUMBER not null
        constraint RESERVE_BOOKING_BOOKING_ID_FK
                references BOOKING,
    BOOKABLE_ID NUMBER not null
        constraint RESERVE_BOOKABLE_BOOKABLE_ID_FK
                references BOOKABLE,
    constraint RESERVE_PK
        primary key (BOOKING_ID, BOOKABLE_ID)
)/
```

```
create table ATTEND
(
   MEMBER_ID NUMBER not null
         constraint ATTEND_MEMBER_MEMBER_ID_FK
               references MEMBER,
   EVENT_ID NUMBER not null
         constraint ATTEND_DAYEVENT_EVENT_ID_FK
               references EVENT,
   constraint ATTEND_PK
         primary key (MEMBER_ID, EVENT_ID)
)/

create table USE
(
   BOOKABLE_ID NUMBER not null
         constraint USE_BOOKABLE_BOOKABLE_ID_FK
               references BOOKABLE,
   EVENT_ID NUMBER not null
         constraint USE_EVENT_EVENT_ID_FK
               references EVENT,
   constraint USE_PK
         primary key (BOOKABLE_ID, EVENT_ID)
)/
```

A list of all SQL queries used.

** Note the ACCESS Table is in quotes because ACCESS is a keyword in SQL **

INSERT

```
INSERT INTO "ACCESS"  (MEMBER_ID, AREA_ID, ACCESS_DATE) " +
                "VALUES (?,?,?)

INSERT INTO RESERVE VALUES (?,?)

INSERT INTO BOOKING VALUES (?,?,?,?)

INSERT INTO ATTEND (EVENT_ID, MEMBER_ID)
                        VALUES (?,?)

INSERT INTO HOSTS (EVENT_ID, EMPLOYEE_ID)
                        VALUES (?,?)

INSERT INTO USE (EVENT_ID, BOOKABLE_ID)
                        VALUES (?,?)

INSERT INTO EVENT (EVENT_ID, NAME, EVENT_DATETIME, BRANCH_ID)
                        VALUES (?,?,?,?)
```

DELETE

```
DELETE FROM member WHERE member_id = ?
```

UPDATE

```
UPDATE member SET email = ? WHERE member_id = ?
```

SELECTION

```
SELECT *
                FROM MEMBER m
                WHERE m.MEMBER_ID = ?

SELECT *
                FROM BRANCH
                WHERE BRANCH_ID = ?

SELECT * FROM branch
```

```
SELECT *
        FROM EMPLOYEE
        WHERE BRANCH_ID = ?


SELECT *
        FROM BOOKING
        WHERE BRANCH_ID = ?
SELECT *
        FROM BOOKABLE
        WHERE BOOKABLE_ID IN (
            SELECT BOOKABLE_ID
            FROM EQUIPMENT
            ) AND BRANCH_ID = ?
SELECT *
        FROM BOOKABLE
        WHERE BOOKABLE_ID IN (
            SELECT BOOKABLE_ID
            FROM ROOM
            ) AND BRANCH_ID = ?
```

PROJECTION

```
SELECT BRANCH_NAME
        FROM branch b, sign_up s
        WHERE s.MEMBER_ID = ? AND
        b.BRANCH_ID = s.BRANCH_ID

SELECT m.FIRST_NAME, m.LAST_NAME, p.AREA_NAME, a.ACCESS_DATE
        FROM "ACCESS" a, MEMBER m, PUBLIC_AREA p
        WHERE a.MEMBER_ID = m.MEMBER_ID
        AND p.AREA_ID = a.AREA_ID "
        AND p.BRANCH_ID = ?

SELECT b.BRANCH_NAME, p.AREA_NAME, a.ACCESS_DATE
         FROM "ACCESS" a, PUBLIC_AREA p, BRANCH b
         WHERE b.BRANCH_ID = p.BRANCH_ID
         AND p.AREA_ID = a.AREA_ID "
         AND a.MEMBER_ID = ?
```

JOIN

```
SELECT *
        FROM MEMBER m, ATTEND a
        WHERE m.MEMBER_ID = a.MEMBER_ID and a.EVENT_ID = ?
```

```
SELECT *
        FROM EVENT, ATTEND
        WHERE EVENT.EVENT_ID = ATTEND.EVENT_ID
        AND MEMBER_ID = ?
SELECT *
        FROM HOSTS, EMPLOYEE
        WHERE HOSTS.EVENT_ID = ? AND
        EMPLOYEE.EMPLOYEE_ID = HOSTS.EMPLOYEE_ID

SELECT *
        FROM ATTEND, MEMBER
        WHERE ATTEND.EVENT_ID = ? AND
        MEMBER.MEMBER_ID = ATTEND.MEMBER_ID

SELECT *
        FROM MEMBER m, BOOKING b, BRANCH br
        WHERE m.MEMBER_ID = b.MEMBER_ID
        AND br.BRANCH_ID = b.BRANCH_ID
        AND b.MEMBER_ID = ?

SELECT *
        FROM RESERVE, BOOKABLE
        WHERE RESERVE.BOOKING_ID = ? AND
        RESERVE.BOOKABLE_ID = BOOKABLE.BOOKABLE_ID
```

Aggregation with GROUP BY:
Find the event name, id, and number of bookable used per event

```
SELECT NAME, G.EVENT_ID, COUNT(BOOKABLE.BOOKABLE_ID)
FROM EVENT, (SELECT EVENT.EVENT_ID, COUNT(BOOKABLE.BOOKABLE_ID)
              FROM BOOKABLE, EVENT, USE
              WHERE BOOKABLE.BOOKABLE_ID = USE.BOOKABLE_ID
              GROUP BY (EVENT.EVENT_ID)) G
WHERE G.EVENT_ID = EVENT.EVENT_ID
```

Aggregation with HAVING
Find users that have accessed public areas ? times before and their number of accesses. ? is user input. It can be 2 times, 3 times, etc.

```
SELECT *
FROM MEMBER M JOIN (SELECT a.MEMBER_ID, COUNT(a.MEMEBR_ID)
                    FROM “ACCESS” a
                    GROUP BY a.MEMBER_ID
                    HAVING COUNT(a.MEMBER_ID) >= ?
                   ) G on MEMBER_ID = G.MEMBER_ID
```

Nested aggregation with GROUP BY:

Count the number of bookables used in the latest event

```
SELECT COUNT(BOOKABLE.BOOKABLE_ID)
FROM BOOKABLE, EVENT, USE
WHERE BOOKABLE.BOOKABLE_ID = USE.BOOKABLE
        AND USE.EVENT_ID = EVENT.EVENT_ID
GROUP BY EVENT.EVENT_DATETIME
HAVING EVENT_DATETIME = (SELECT MAX(EVENT_DATETIME)
                         FROM EVENT)
```

Division
Find names of members who signed up in every branch

```
SELECT FIRST_NAME, LAST_NAME
FROM member m
WHERE NOT EXISTS (SELECT b.branch_id
                  FROM branch b
                  WHERE NOT EXISTS (SELECT r.branch_id
                                    FROM SIGN_UP r
                                    WHERE r.branch_id = b.branch_id
                                        AND m.member_id = r.member_id))
```