Faculty of Engineering and Computer Science
Expectations of Originality

This form sets out the requirements for originality for work submitted by students in the Faculty of Engineering and Computer Science. Submissions such as assignments, lab reports, project reports, computer programs and take-home exams must conform to the requirements stated on this form and to the Academic Code of Conduct. The course outline may stipulate additional requirements for the course.

1. Your submissions must be your own original work. Group submissions must be the original work of the students in the group.
2. Direct quotations must not exceed 5% of the content of a report, must be enclosed in quotation marks, and must be attributed to the source by a numerical reference citation1 . Note that engineering reports rarely contain direct quotations.
3. Material paraphrased or taken from a source must be attributed to the source by a numerical reference citation.
4. Text that is inserted from a web site must be enclosed in quotation marks and attributed to the web site by numerical reference citation.
5. Drawings, diagrams, photos, maps or other visual material taken from a source must be attributed to that source by a numerical reference citation.
6. No part of any assignment, lab report or project report submitted for this course can be submitted for any other course.
7. In preparing your submissions, the work of other past or present students cannot be consulted, used, copied, paraphrased or relied upon in any manner whatsoever.
8. Your submissions must consist entirely of your own or your group's ideas, observations, calculations, information and conclusions, except for statements attributed to sources by numerical citation.
9. Your submissions cannot be edited or revised by any other student.
10. For lab reports, the data must be obtained from your own or your lab group's experimental work.
11. . For software, the code must be composed by you or by the group submitting the work, except for code that is attributed to its sources by numerical reference.

You must write one of the following statements on each piece of work that you submit: For individual work: **"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"**, with your signature, I.D. #, and the date. For group work: **"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"**, with the signatures and I.D. #s of all the team members and the date.

A signed copy of this form must be submitted to the instructor at the beginning of the semester in each course.

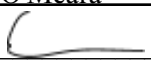
I certify that I have read the requirements set out on this form, and that I am aware of these requirements. I certify that all the work I will submit for this course will comply with these requirements and with additional requirements stated in the course outline.

Course Number: COMP 333 - Databases Instructor: Khaled Jababo

Name: Réa Mourad I.D. # 40310288

Signature:  Date: April 10th, 2025

Name: Elizabeth O'Meara I.D. # 40065959

Signature:  Date: April 10th, 2025

Name: Amani-Myriam Maamar I.D. # 40191681

Signature: AMM Date: April 4th, 2025

Name: Anh Thy Vu I.D. # 40270849

Signature:  Date: April 1st, 2025

Name: Yani Zahouani I.D. # 40285973

Signature: yz Date: Apr 4th 2025

Main Project

Presented to:
Dept. of Computer Science & Software Engineering
Concordia University

Course:
COMP 353 - Databases

Professor:
Khaled Jababo

By:
Brc353_4
Réa Mourad (40310288)
Elizabeth O'Meara (40065959)
Amani-Myriam Maamar (40191681)
Anh Thy Vu (40270849)
Yani Zahouani (40285973)

Due:
April 14, 2025

TABLE OF CONTENT

1. Contributions

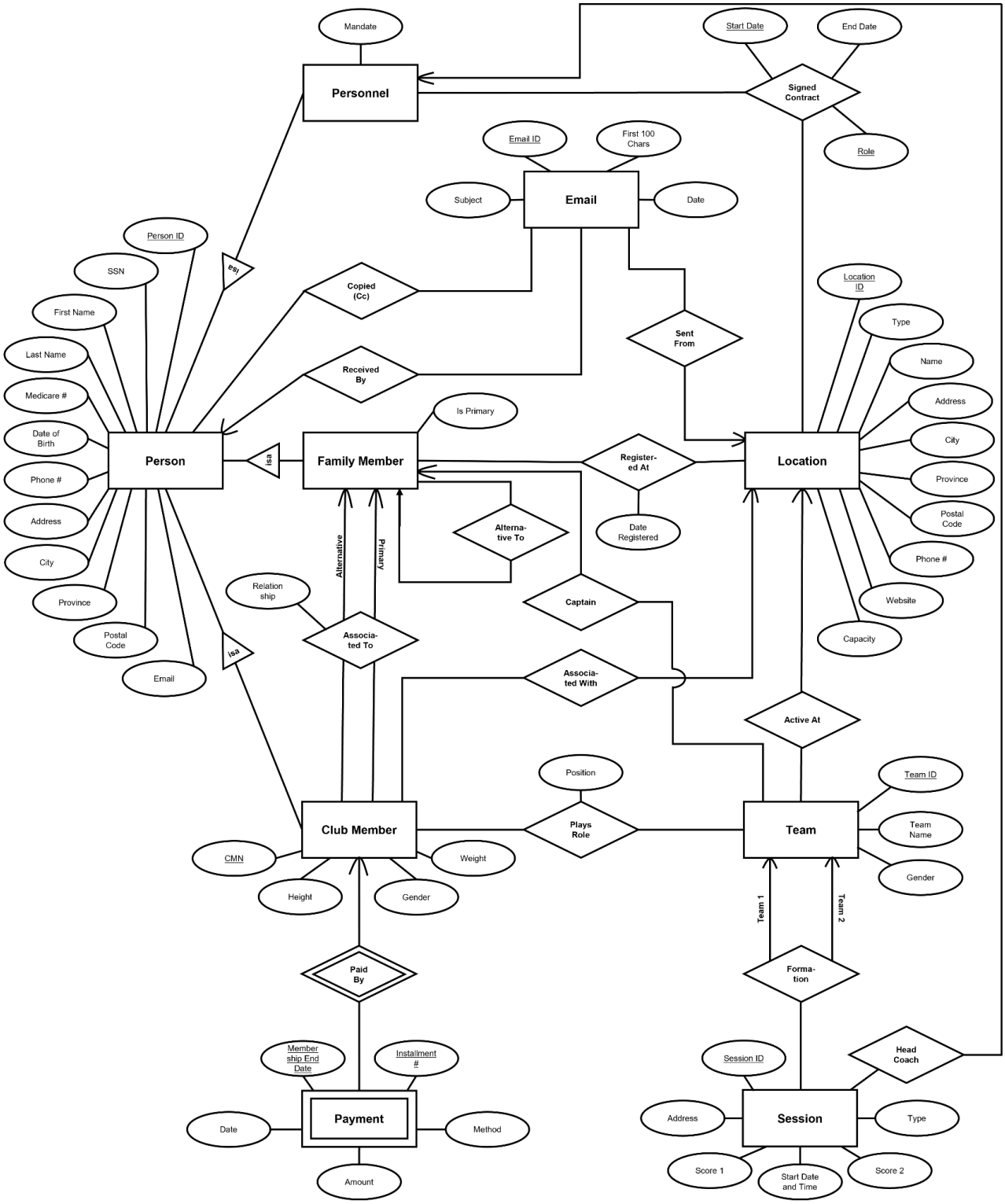
Full Name	Student ID	Task Completed
Réa Mourad	40310288	● Responsible for Queries 13-16 ● Implemented the pages for Queries 13-16
Elizabeth O'Meara	40065959	● Created all the SQL relation schemas ● Created all the triggers ● Populated all the schemas ● Coordinated the display for each queries
Amani-Myriam Maamar	40191681	● Designed ER diagram and relation schemas ● Responsible for normalization ● Responsible for Queries 6, 17, 18 ● Implemented the Team Assignment (Assign/Edit/Delete) ● Implemented the Session pages (display/add/edit/delete)
Anh Thy Vu	40270849	● Responsible for Queries 1-5 ● Implemented most of the GUI (Server Connection, CSS styling, Website Design) ● Implemented the Club Member Pages (Display/Add/Edit/Delete) ● Implemented the Family Member Pages (Display/Add/Edit/Delete) ● Implemented the Personnel Pages (Display/Add/Edit/Delete) ● Implemented the Locations Pages (Display/Add/Edit/Delete) ● Implemented the Team Formation Pages (Display/Add/Edit/Delete) ● Implement the Email Logs
Yani Zahouani	40285973	● Designed Queries 7-12 ● Implemented dynamic components for Query 9, 11 and 15 ● Implemented styling and Tables for Queries 7-12 ● Implemented Sub Queries to check for validity of dynamic placeholder value in database

Teamwork:

- Debugging website and SQL queries
- Participated in team meeting

2. Design

2.1 ER Diagram



2.2 Relations

Constraints:

- There can only be one Location of type “Head.” The rest must be of type “Branch.”
- In the Person entity, SSN and Medicare # must be unique and cannot be NULL.
- The Club Members of a Team must have the same gender and be associated with the same Location during assignment.
- A Club Member may not participate in two Sessions with a start time difference less than 3 hours.

Relation Schema:

Location (LocationID, Type, Name, Address, City, Province, PostalCode, PhoneNumber, Website, Capacity)

Person (PersonID, SSN, FirstName, LastName, MedicareNumber, DateOfBirth, Email, PhoneNumber, Address, City, Province, PostalCode)

Personnel ($\underline{\text{EmployeeID}}_{\text{Person}}$, Mandate)

FamilyMember ($\underline{\text{PersonID}}_{\text{Person}}$, isPrimary, $\text{AlternativeFamilyID}_{\text{FamilyMember}}$)

ClubMember (CMN, Gender, Height, Weight, $\text{LocationID}_{\text{Location}}$, $\text{PrimaryFamilyID}_{\text{FamilyMember}}$, PrimaryRelationship, AlternativeFamilyID, AlternativeRelationship, $\text{PersonID}_{\text{Person}}$)

Email ($\underline{\text{EmailID}}$, $\text{LocationID}_{\text{Location}}$, $\text{RecipientID}_{\text{Person}}$, Subject, Date, First100Chars)

Payment ($\underline{\text{CMN}}_{\text{ClubMember}}$, Date, Amount, Method, InstallmentNumber, MembershipEndDate)

Team ($\underline{\text{TeamID}}$, TeamName, Gender, $\text{LocationID}_{\text{Location}}$, $\text{Captain}_{\text{FamilyMember}}$)

Session ($\underline{\text{SessionID}}$, Type, Address, StartDateTime, $\text{HeadCoachID}_{\text{Personnel}}$, $\text{Team1ID}_{\text{Team}}$, $\text{Team2ID}_{\text{Team}}$, Score1, Score2)

RegisteredAt ($\underline{\text{FamilyID}}_{\text{FamilyMember}}$, $\underline{\text{LocationID}}_{\text{Location}}$, DateRegistered)

Role ($\underline{\text{CMN}}_{\text{ClubMember}}$, $\underline{\text{TeamID}}_{\text{Team}}$, Position)

Contract ($\underline{\text{EmployeeID}}_{\text{Personnel}}$, $\underline{\text{LocationID}}_{\text{Location}}$, StartDate, EndDate, Role)

Copied ($\underline{\text{EmailID}}_{\text{Email}}$, $\underline{\text{CopiedID}}_{\text{Person}}$)

(Primary keys are underlined, and foreign keys are denoted by subscripts representing the referenced table.)

3. Normalization

The only nontrivial functional dependency applicable to our database is: PostalCode → (City, Province). However, since all communication and operations are conducted via email, this dependency has minimal impact on the application's functionality and is not particularly relevant in our context. As there are no other nontrivial functional dependencies applicable to the relations in our database, all relations are already in both BCNF and 3NF. Therefore, no further normalization or decomposition is necessary.

4. SQL Relation Schema Creation

4.1 Person

```sql
-- Create the Person table
CREATE TABLE Person (
    PersonID INT PRIMARY KEY AUTO_INCREMENT,
    SSN INT(9) UNSIGNED UNIQUE,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    MedicareNumber VARCHAR(50) UNIQUE,
    DateOfBirth DATE,
    Email VARCHAR(255),
    PhoneNumber BIGINT(10) UNSIGNED NOT NULL,
    Address VARCHAR(255),
    City VARCHAR(100),
    Province ENUM('AB', 'NL', 'NS', 'PE', 'BC', 'QC', 'ON', 'MB', 'SK', 'YT', 'NT', 'NU'),
    PostalCode VARCHAR(10)
);
```

4.2 FamilyMember

```sql
CREATE TABLE FamilyMember (
    PersonID INT PRIMARY KEY,
    isPrimary BOOLEAN NOT NULL,
    AlternativeFamilyID INT NULL,
    FOREIGN KEY (PersonID) REFERENCES Person(PersonID),
    FOREIGN KEY (AlternativeFamilyID) REFERENCES FamilyMember(PersonID),
    CONSTRAINT chk_alternative_family_id CHECK (
        (AlternativeFamilyID IS NULL) OR (isPrimary = TRUE AND AlternativeFamilyID IS NOT NULL)
    )
);
```

4.3 ClubMember

```
-- Create the Club Member table
CREATE TABLE ClubMember (
    CMN INT PRIMARY KEY AUTO_INCREMENT,
    Gender ENUM('M', 'F') NOT NULL,
    Height DECIMAL(5, 2) NOT NULL,
    Weight DECIMAL(5, 2) NOT NULL,
    LocationID INT,
    PrimaryFamilyID INT NOT NULL,
    Relationship ENUM('Father', 'Mother', 'Grandfather', 'Grandmother', 'Uncle', 'Aunt', 'Tutor', 'Partner', 'Friend', 'Other') NOT NULL,
    AlternativeFamilyID INT NULL,
AlternativeRelationship ENUM('Father', 'Mother', 'Grandfather', 'Grandmother', 'Uncle', 'Aunt', 'Tutor', 'Partner', 'Friend', 'Other') NULL,
    PersonID INT UNIQUE,
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID),
    FOREIGN KEY (PrimaryFamilyID) REFERENCES FamilyMember(PersonID),
    FOREIGN KEY (PersonID) REFERENCES Person(PersonID)
);
```

4.4 Personnel

```
-- Create the Personnel table
CREATE TABLE Personnel (
    EmployeeID INT PRIMARY KEY,
    Mandate ENUM('Salaried', 'Volunteer') NOT NULL,
   FOREIGN KEY (EmployeeID) REFERENCES Person(PersonID)
);
```

4.5 Location

```
-- Create the Location table
CREATE TABLE Location (
    LocationID INT PRIMARY KEY AUTO_INCREMENT,
    Type ENUM('Head', 'Branch') NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(100) NOT NULL,
    Province ENUM('AB', 'NL', 'NS', 'PE', 'BC', 'QC', 'ON', 'MB', 'SK', 'YT', 'NT', 'NU') NOT NULL,
    PostalCode VARCHAR(10) NOT NULL,
    PhoneNumber BIGINT(10) UNSIGNED NOT NULL,
    Website VARCHAR(255) NOT NULL,
    Capacity INT NOT NULL
);
```

4.6 Contract

```
-- Create the Email table
CREATE TABLE Email (
    EmailID INT PRIMARY KEY AUTO_INCREMENT,
    LocationID INT NOT NULL,
    RecipientID INT NOT NULL,
    Subject VARCHAR(100) NOT NULL,
    Date DATETIME NOT NULL,
    First100Chars VARCHAR(100) NOT NULL,
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID),
    FOREIGN KEY (RecipientID) REFERENCES Person(PersonID)


);
```

4.7 Copied

```
-- Create the Copied table
CREATE TABLE Copied (
    EmailID INT,
    CopiedID INT,
    PRIMARY KEY (EmailID, CopiedID),
    FOREIGN KEY (EmailID) REFERENCES Email(EmailID),
    FOREIGN KEY (CopiedID) REFERENCES Person(PersonID)
);
```

4.8 Email

```sql
-- Create the Email table
CREATE TABLE Email (
    EmailID INT PRIMARY KEY AUTO_INCREMENT,
    LocationID INT NOT NULL,
    RecipientID INT NOT NULL,
    Subject VARCHAR(100) NOT NULL,
    Date DATETIME NOT NULL,
    First100Chars VARCHAR(100) NOT NULL,
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID),
    FOREIGN KEY (RecipientID) REFERENCES Person(PersonID)


);
```

4.9 Payment

```sql
-- Create the Payment table
CREATE TABLE Payment (
    CMN INT,
    Date DATE,
    Amount DECIMAL(6, 2),
    Method ENUM('Cash', 'Credit Card', 'Debit'),
    InstallmentNumber INT,
    MembershipEndDate DATE,
    PRIMARY KEY (CMN, InstallmentNumber, MembershipEndDate),
    FOREIGN KEY (CMN) REFERENCES ClubMember(CMN),
    CONSTRAINT chk_installment_number CHECK (InstallmentNumber BETWEEN 1 AND 4)
);
```

4.10 RegisteredAt

```sql
-- Create the RegisteredAt table
CREATE TABLE RegisteredAt (
    FamilyID INT,
    LocationID INT,
    DateRegistered DATE,
    PRIMARY KEY (FamilyID, LocationID),
    FOREIGN KEY (FamilyID) REFERENCES FamilyMember(PersonID),
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
);
```

4.11 Role

```sql
-- Create the Role table
CREATE TABLE Role (
    CMN INT,
    TeamID INT,
    Position ENUM('Outside Hitter', 'Opposite', 'Setter', 'Middle Blocker', 'Libero', 'Defensive Specialist', 'Serving Specialist') NOT NULL,
    PRIMARY KEY (CMN, TeamID),
    FOREIGN KEY (CMN) REFERENCES ClubMember(CMN),
    FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);
```

4.12 Session

```sql
-- Create the Session table
CREATE TABLE Session (
    SessionID INT PRIMARY KEY AUTO_INCREMENT,
    Type ENUM('Training', 'Game') NOT NULL,
    Address VARCHAR(255) NOT NULL,
    StartDateTime DATETIME NOT NULL,
    HeadCoachID INT NOT NULL,
    Team1ID INT NOT NULL,
    Team2ID INT NOT NULL,
    Score1 INT,
    Score2 INT,
    FOREIGN KEY (HeadCoachID) REFERENCES Personnel(EmployeeID),
    FOREIGN KEY (Team1ID) REFERENCES Team(TeamID),
    FOREIGN KEY (Team2ID) REFERENCES Team(TeamID),
    CONSTRAINT chk_team_ids_different CHECK (Team1ID != Team2ID)
);
```

4.13 Team

```sql
-- Create the Team table
CREATE TABLE Team (
    TeamID INT PRIMARY KEY AUTO_INCREMENT,
    TeamName VARCHAR(255) NOT NULL,
    Gender ENUM('M', 'F') NOT NULL,
    LocationID INT NOT NULL,
    Captain INT NULL,
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID),
    FOREIGN KEY (Captain) REFERENCES FamilyMember(PersonID)
);
```

5. SQL Queries Creation

1. Create/Delete/Edit/Display a Location.

- Create -

```
$locationQuery = "
    INSERT INTO Location (Type, Name, Address, City, Province, PostalCode, PhoneNumber, Website, Capacity)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
";
```

```
$contractQuery = "
    INSERT INTO Contract (EmployeeID, LocationID, StartDate, EndDate, Role)
    VALUES (?, LAST_INSERT_ID(), ?, NULL, ?)
";
```

- Delete -

```
$deleteQuery = "
    DELETE FROM Location WHERE LocationID = ?
";
```

- Edit -

```
$locationQuery = "
    UPDATE Location SET
        Type = ?,
        Name = ?,
        Address = ?,
        City = ?,
        Province = ?,
        PostalCode = ?,
        PhoneNumber = ?,
        Website = ?,
        Capacity = ?
    WHERE
        LocationID = ?
";
```

- Display -

```
$query = "
    SELECT
        l.*,
        CONCAT(p.FirstName, ' ', p.LastName) AS FullName
    FROM
        Location l
    LEFT JOIN
        Contract c ON l.LocationID = c.LocationID AND c.Role = 'General Manager'
    LEFT JOIN
        Person p ON p.PersonID = c.EmployeeID
    ORDER BY Province ASC, City ASC
";
```

2. Create/Delete/Edit/Display a Personnel.

- Create -

```
$personQuery = "
    INSERT INTO Person (SSN, FirstName, LastName, MedicareNumber, DateOfBirth, Email, PhoneNumber, Address, City, Province, PostalCode)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
";
```

```
$personnelQuery = "
    INSERT INTO Personnel (EmployeeID, Mandate)
    VALUES (?, ?)
";
```

- Delete -

```
$deletePersonnelQuery = "
    DELETE FROM Personnel WHERE EmployeeID = ?
";
```

```
$deletePersonQuery = "
    DELETE FROM Person WHERE PersonID = ?
";
```

- Edit -

```
$updatePersonQuery = "
    UPDATE Person SET
        SSN = ?,
        FirstName = ?,
        LastName = ?,
        MedicareNumber = ?,
        DateOfBirth = ?,
        Email =  ?,
        PhoneNumber = ?,
        Address = ?,
        City = ?,
        Province = ?,
        PostalCode = ?
    WHERE
        PersonID = ?
";
```

```
$updatePersonnelQuery = "
    UPDATE Personnel SET
        Mandate  = ?
    WHERE
        EmployeeID = ?
";
```

- Display -

```
$query = "
    SELECT
        p.PersonID,
        p.FirstName,
        p.LastName,
        p.Email,
        p.PhoneNumber,
        p.MedicareNumber,
        p.SSN,
        p.DateOfBirth,
        p.Address,
        p.City,
        p.Province,
        p.PostalCode,
        pers.Mandate
    FROM
        Personnel pers
    JOIN
        Person p ON pers.EmployeeID = p.PersonID
    ORDER BY
        p.PersonID
";
```

3. Create/Delete/Edit/Display a FamilyMember (Primary/Secondary).

- Create -

Primary

```
$personQuery = "
    INSERT INTO Person (SSN, FirstName, LastName, MedicareNumber, DateOfBirth, Email, PhoneNumber, Address, City, Province, PostalCode)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
";
```

```
$familyQuery = "
    INSERT INTO FamilyMember (PersonID, isPrimary)
    VALUES (?, ?)
";
```

Secondary

```
$updateFamilyQuery = "
    UPDATE FamilyMember SET
        isPrimary  = ?
    WHERE
        PersonID = ?
";
```

- Delete -

```
$deleteFamilyQuery = "
    DELETE FROM FamilyMember WHERE PersonID = ?
";
```

```
$deletePersonQuery = "
    DELETE FROM Person WHERE PersonID = ?
";
```

- Edit -

```
$updatePersonQuery = "
    UPDATE Person SET
        SSN = ?,
        FirstName = ?,
        LastName = ?,
        MedicareNumber = ?,
        DateOfBirth = ?,
        Email =  ?,
        PhoneNumber = ?,
        Address = ?,
        City = ?,
        Province = ?,
        PostalCode = ?
    WHERE
        PersonID = ?
";
```

```
$updateFamilyQuery = "
    UPDATE FamilyMember SET
        isPrimary  = ?
    WHERE
        PersonID = ?
";
```

- Display -

```
$query = "
    SELECT
        p.PersonID,
        fm.isPrimary,
        p.FirstName,
        p.LastName,
        p.Email,
        p.PhoneNumber,
        p.MedicareNumber,
        p.DateOfBirth,
        p.Address,
        p.City,
        p.Province,
        p.PostalCode
    FROM
        FamilyMember fm
    JOIN
        Person p ON fm.PersonID = p.PersonID
    ORDER BY
        p.PersonID
";
```

4. Create/Delete/Edit/Display a ClubMember.

- Create -

```
$personQuery = "
    INSERT INTO Person (SSN, FirstName, LastName, MedicareNumber, DateOfBirth, Email, PhoneNumber, Address, City, Province, PostalCode)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
";
```

```
$clubMemberQuery = "
    INSERT INTO ClubMember (Gender, Height, Weight, LocationID, PrimaryFamilyID, Relationship, AlternativeFamilyID, AlternativeRelationship, PersonID)
    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
";
```

- Delete -

```
$deleteClubMemberQuery = "
    DELETE FROM ClubMember WHERE PersonID = ?
";
```

```
$deletePersonQuery = "
    DELETE FROM Person WHERE PersonID = ?
";
```

- Edit -

```
$updatePersonQuery = "
    UPDATE Person SET
        SSN = ?,
        FirstName = ?,
        LastName = ?,
        MedicareNumber = ?,
        DateOfBirth = ?,
        Email = ?,
        PhoneNumber = ?,
        Address = ?,
        City = ?,
        Province = ?,
        PostalCode = ?
    WHERE
        PersonID = ?
";
```

```
$updateClubMemberQuery = "
    UPDATE ClubMember SET
        Gender = ?,
        Height = ?,
        Weight = ?,
        LocationID = ?,
        PrimaryFamilyID = ?,
        Relationship = ?,
        AlternativeFamilyID = ?,
        AlternativeRelationship = ?
    WHERE
        PersonID = ?
";
```

- Display -

```
$query = "
    SELECT
        p.PersonID,
        cm.CMN,
        p.FirstName,
        p.LastName,
        cm.Gender,
        cm.Height,
        cm.Weight,
        p.PhoneNumber,
        p.Email,
        p.DateOfBirth,
        p.Address,
        p.City,
        p.Province,
        p.PostalCode
    FROM
        ClubMember cm
    JOIN
        Person p ON cm.PersonID = p.PersonID
    ORDER BY
        p.PersonID
";
```

5. Create/Delete/Edit/Display a TeamFormation.

- Create -

```
$teamQuery = "
    INSERT INTO Team (TeamName, Gender, LocationID, Captain)
    VALUES (?, ?, ?, ?)
";
```

- Delete -

```
$deleteQuery = "
    DELETE FROM Team WHERE TeamID = ?
";
```

- Edit -

```
$updateQuery = "
    UPDATE Team Set
        TeamName = ?,
        Gender = ?,
        LocationID = ?,
        Captain = ?
    WHERE
        TeamID = ?
";
```

- Display -

```
$query = "
    SELECT
        t.TeamID,
        t.TeamName,
        t.Gender,
        t.LocationID,
        l.Name AS LocationName,
        t.Captain AS CaptainID,
        CONCAT(p.FirstName, ' ', p.LastName) AS CaptainName,
        (SELECT COUNT(*) FROM Role r WHERE r.TeamID = t.TeamID) AS PlayerCount
    FROM
        Team t
    LEFT JOIN Location l ON t.LocationID = l.LocationID
    LEFT JOIN Person p ON t.Captain = p.PersonID
    ORDER BY
        t.TeamID
";
```

6. Assign/Delete/Edit a club member to a team formation. (Attempt to assign a conflicting assignment for a club member in two team formations on the same day).

- Add Session -

```
$teamQuery = "
    INSERT INTO Session(Type, Address, StartDateTime, HeadCoachId, Team1ID, Team2ID)
    VALUES (?, ?, ?, ?, ?, ?)
";
```

- Delete Session -

```
$deleteQuery = "
    DELETE FROM Session WHERE SessionID = ?
";
```

- Edit Session -

```
$updateQuery = "
    UPDATE Session SET
        Type = ?, Address = ?,
        StartDateTime = ?,
        HeadCoachId = ?,
        Team1ID = ?,
        Team2ID = ?
    WHERE SessionID = ?
";
```

- Assign Player to Team -

```
$addQuery = "
    INSERT INTO Role (CMN, TeamID, Position)
    VALUES (?, ?, ?);
";
```

- Delete Player in Team -

```
$deleteQuery = "DELETE FROM Role
                WHERE TeamID = ? AND CMN = ?";
```

- Edit Player Position in Team -

```
$updateQuery = "
    UPDATE Role Set
        Position = ?
    WHERE
        TeamID = ? AND CMN = ?
";
```

7. Get complete details for every location in the system. Details include address, city, province, postal-code, phone number, web address, type (Head, Branch), capacity, general manager name, and the number of club members associated with that location. The results should be displayed sorted in ascending order by Province, then by city.

```
$query = "
    SELECT
        l.*,
        CONCAT(p.FirstName, ' ', p.LastName) AS FullName
    FROM
        Location l
    LEFT JOIN
        Contract c ON l.LocationID = c.LocationID AND c.Role = 'General Manager'
    LEFT JOIN
        Person p ON p.PersonID = c.EmployeeID
    ORDER BY Province ASC, City ASC
";
```

8. For a given family member, get details of all the locations that she/he was/is associated with, the secondary family member and all the club members associated with the primary family member. Information includes first name, last name and phone number of the secondary family member, and for every associated club member, the location name, the club membership number, first-name, last-name, date of birth, Social Security Number, Medicare card number, telephone number, address, city, province, postal-code, and relationship with the secondary family member.

```
$locationQuery = "
    SELECT
        l.Name AS LocationName,
        l.Address,
        l.City,
        l.Province,
        l.PostalCode
    FROM FamilyMember AS fm
    JOIN RegisteredAt as ra
        ON fm.PersonID = ra.FamilyID
    JOIN Location AS l
        ON ra.LocationID = l.LocationID
    WHERE fm.PersonID = $Q8Id
";
```

```
$secondaryQuery = "
    SELECT
        secondary_person.FirstName AS SecondaryFirstName,
        secondary_person.LastName AS SecondaryLastName,
        secondary_person.PhoneNumber AS SecondaryPhoneNumber
    FROM FamilyMember AS fm
    LEFT JOIN FamilyMember AS secondary_fm
        ON fm.AlternativeFamilyID = secondary_fm.PersonID
    LEFT JOIN Person AS secondary_person
        ON secondary_fm.PersonID = secondary_person.PersonID
    WHERE fm.PersonID = $Q8Id
";
```

```
$membersQuery = "
    SELECT
        cm.CMN AS ClubMembershipNumber,
        person.FirstName AS ClubMemberFirstName,
        person.LastName AS ClubMemberLastName,
        person.DateOfBirth,
        person.SSN,
        person.MedicareNumber,
        person.PhoneNumber AS ClubMemberPhoneNumber,
        person.Address,
        person.City,
        person.Province,
        person.PostalCode,
        cm.Relationship AS RelationshipToSecondaryFamilyMember
    FROM FamilyMember AS fm
    JOIN ClubMember AS cm
        ON cm.PrimaryFamilyID = fm.PersonID
    JOIN Person AS person
        ON cm.PersonID = person.PersonID
    WHERE
        fm.PersonID = $Q8Id
";
```

9. For a given location and week, get details of all the teams' formations recorded in the system. Details include, head coach first name and last name, start time of the training or game session, address of the session, nature of the session (training or game), the teams name, the score (if the session is in the future, then score will be null), and the first name, last name and role (goalkeeper, defender, etc.) of every player in the team. Results should be displayed sorted in ascending order by the start day then by the start time of the session.

```
$query = "
    SELECT
        Person.FirstName AS HeadCoachFirstName,
        Person.LastName AS HeadCoachLastName,
        S.StartDateTime AS SessionStartTime,
        S.Address AS SessionAddress,
        S.Type AS SessionType,
        T.TeamName AS TeamName,
        S.Score1 AS Team1Score,
        S.Score2 AS Team2Score,
        Player.FirstName AS PlayerFirstName,
        Player.LastName AS PlayerLastName,
        R.Position AS PlayerRole
    FROM Session S
    JOIN Personnel P ON S.HeadCoachID = P.EmployeeID
    JOIN Person ON P.EmployeeID = Person.PersonID
    JOIN Team T ON T.TeamID = S.Team1ID OR T.TeamID = S.Team2ID
    JOIN Role R ON R.TeamID = T.TeamID
    JOIN ClubMember CM ON CM.CMN = R.CMN
    JOIN Person Player ON Player.PersonID = CM.PersonID
    JOIN Location L ON L.LocationID = T.LocationID
    WHERE L.Name = '$location'
    AND S.StartDateTime BETWEEN '$start_date' AND '$end_date'
    ORDER BY DATE(S.StartDateTime) ASC, TIME(S.StartDateTime) ASC;
";
```

10. Get details of club members who are currently active and have been associated with at least three different locations and are members for at most three years. Details include Club membership number, first name and last name. Results should be displayed sorted in ascending order by club membership number.

```
$query = "
SELECT DISTINCT
    CM.CMN AS ClubMembershipNumber,
    P.FirstName,
    P.LastName
FROM ClubMember CM
JOIN Person P ON CM.PersonID = P.PersonID
JOIN RegisteredAt RA ON CM.PrimaryFamilyID = RA.FamilyID
WHERE CM.CMN IN (
    SELECT CMN
    FROM RegisteredAt
    GROUP BY CMN
    HAVING COUNT(DISTINCT LocationID) >= 3
)
AND TIMESTAMPDIFF(YEAR, RA.DateRegistered, CURDATE()) <= 3
ORDER BY CM.CMN ASC;
```

11. For a given period of time, give a report on the teams' formations for all the locations. For each location, the report should include the location name, the total number of training sessions, the total number of players in the training sessions, the total number of game sessions, the total number of players in the game sessions. Results should only include locations that have at least two game sessions. Results should be displayed sorted in descending order by the total number of game sessions. For example, the period of time could be from Jan. 1 st, 2025 to Mar. 31st, 2025.

```
$query = "
    SELECT
        L.Name AS LocationName,
        COUNT(CASE WHEN S.Type = 'Training' THEN S.SessionID END) AS TotalTrainingSessions,
        SUM(CASE WHEN S.Type = 'Training' THEN (SELECT COUNT(*)
                                                FROM Role R
                                                WHERE R.TeamID = S.Team1ID OR R.TeamID = S.Team2ID) END) AS TotalPlayersInTraining,
        COUNT(CASE WHEN S.Type = 'Game' THEN S.SessionID END) AS TotalGameSessions,
        SUM(CASE WHEN S.Type = 'Game' THEN (SELECT COUNT(*)
                                            FROM Role R
                                            WHERE R.TeamID = S.Team1ID OR R.TeamID = S.Team2ID) END) AS TotalPlayersInGames
    FROM Location L
    JOIN Session S ON L.LocationID = S.Team1ID OR L.LocationID = S.Team2ID
    WHERE S.StartDateTime BETWEEN '$start_date' AND '$end_date'
    GROUP BY L.LocationID
    HAVING COUNT(CASE WHEN S.Type = 'Game' THEN S.SessionID END) >= 2
    ORDER BY TotalGameSessions DESC;
";
```

12. Get a report on all active club members who have never been assigned to any formation team session. The list should include the club member's membership number, first name, last name, age, date of joining the club, phone number, email and current location name. The results should be displayed sorted in ascending order by location name then by club membership number.

```
$query = "
    SELECT
        CM.CMN AS MembershipNumber,
        P.FirstName,
        P.LastName,
        TIMESTAMPDIFF(YEAR, P.DateOfBirth, CURDATE()) AS Age,
        MIN(RA.DateRegistered) AS DateOfJoining,
        P.PhoneNumber,
        P.Email,
        L.Name AS CurrentLocationName
    FROM ClubMember CM
    JOIN Person P ON CM.PersonID = P.PersonID
    JOIN RegisteredAt RA ON CM.PrimaryFamilyID = RA.FamilyID
    JOIN Location L ON CM.LocationID = L.LocationID
    WHERE CM.CMN NOT IN (
        SELECT DISTINCT R.CMN
        FROM Role R
    )
    GROUP BY CM.CMN, P.FirstName, P.LastName, P.DateOfBirth, P.PhoneNumber, P.Email, L.Name
    ORDER BY L.Name ASC, CM.CMN ASC;
";
```

13. Get a report on all active club members who have only been assigned as outside hitter in all the formation team sessions they have been assigned to. They must be assigned to at least one formation session as an outside hitter. They should have never been assigned to any formation session with a role different than outside hitter. The list should include the club member's membership number, first name, last name, age, phone number, email and current location name. The results should be displayed sorted in ascending order by location name then by club membership number.

```
SELECT
    ClubMember.CMN,
    Person.FirstName,
    Person.LastName,
    FLOOR(DATEDIFF(CURDATE(), Person.DateOfBirth)/365) AS age,
    Person.PhoneNumber,
    Person.Email,
    Location.name AS location_name
FROM Person, ClubMember, Location, Payment
WHERE
    -- joins
    Person.PersonID = ClubMember.PersonID
    AND Location.LocationID = ClubMember.LocationID
    AND ClubMember.CMN = Payment.CMN
    -- club member are still active
    AND Payment.MembershipEndDate  >= CURDATE()
    AND FLOOR(DATEDIFF(CURDATE(), Person.DateOfBirth)/365) <= 18
    -- they're in a session as an outside hitter
    AND EXISTS (
        SELECT 1
        FROM Role r1
        JOIN Session ON r1.TeamID = Session.Team1ID OR r1.TeamID = Session.Team2ID
        WHERE r1.CMN = ClubMember.CMN
        AND r1.Position = 'Outside Hitter'
    )
    -- they should never be assigned any other role in any session
    AND NOT EXISTS (
        SELECT 1
        FROM Role r2
        JOIN Session ON r2.TeamID = Session.Team1ID OR r2.TeamID = Session.Team2ID
        WHERE r2.CMN = ClubMember.CMN
        AND r2.Position != 'Outside Hitter'
    )

GROUP BY ClubMember.CMN
ORDER BY
    location_name ASC,
    ClubMember.CMN ASC
```

14. Get a report on all active club members who have been assigned at least once to every role throughout all the formation team game sessions. The club member must be assigned to at least one formation game session as an outside hitter, opposite, setter, middle blocker, libero, defensive specialist, and serving specialist. The list should include the club member's membership number, first name, last name, age, phone number, email and current location name. The results should be displayed sorted in ascending order by location name then by club membership number.

```
$query = "
    SELECT
        cm.CMN,
        p.FirstName,
        p.LastName,
        FLOOR(DATEDIFF(CURDATE(), p.DateOfBirth)/365) AS age,
        p.PhoneNumber,
        p.Email,
        l.Name AS location_name
    FROM ClubMember cm
    JOIN Person p ON cm.PersonID = p.PersonID
    JOIN Location l ON cm.LocationID = l.LocationID
    JOIN Payment pay ON cm.CMN = pay.CMN
    WHERE
        pay.MembershipEndDate >= CURDATE()
        AND FLOOR(DATEDIFF(CURDATE(), p.DateOfBirth)/365) <= 18
        AND cm.CMN IN (
            SELECT r.CMN
            FROM Role r
            JOIN Session s ON r.TeamID = s.Team1ID OR r.TeamID = s.Team2ID
            GROUP BY r.CMN
            HAVING
                COUNT(DISTINCT r.Position) = 7
        )
    ORDER BY
        location_name ASC,
        cm.CMN ASC
";
```

15. For the given location, get the list of all family members who have currently active club members associated with them and are also captains for the same location. Information includes first name, last name, and phone number of the family member. A family member is considered to be a captain if she/he is assigned as a captain to at least one team formation session in the same location.

```
$stmt = $conn->prepare("
    SELECT DISTINCT
        captain.FirstName AS CaptainFirstName,
        captain.LastName AS CaptainLastName,
        captain.PhoneNumber
    FROM FamilyMember fm
    JOIN Person captain ON fm.PersonID = captain.PersonID
    JOIN Team t ON t.Captain = fm.PersonID
    JOIN ClubMember cm ON (cm.PrimaryFamilyID = fm.PersonID OR cm.AlternativeFamilyID = fm.PersonID)
    JOIN Location l ON t.LocationID = l.LocationID
    WHERE (fm.isPrimary = TRUE OR cm.AlternativeFamilyID IS NOT NULL)
    AND l.Name = ?
");
```

16. Get a report of all active club members who have never lost a game in which they played. A club member is considered to win a game if she/he has been assigned to a game session and is assigned to the team that has a score higher than the score of the other team. The club member must be assigned to at least one formation game session. The list should include the club member's membership number, first name, last name, age, phone number, email and current location name. The results should be displayed sorted in ascending order by location name then by club membership number.

```
    $query = "
        SELECT
    cm.CMN,
    p.FirstName,
    p.LastName,
    FLOOR(DATEDIFF(CURDATE(), p.DateOfBirth)/365) AS age,
    p.PhoneNumber,
    p.Email,
    l.Name AS LocationName
FROM
    ClubMember cm
    JOIN Person p ON cm.PersonID = p.PersonID
    JOIN Location l ON cm.LocationID = l.LocationID
    JOIN Payment pay ON cm.CMN = pay.CMN
WHERE
    -- Member is active
    pay.MembershipEndDate >= CURDATE()

    --  has been assigned to at least one formation game session
    AND EXISTS (
        SELECT 1
        FROM Role r
        JOIN Team t ON r.TeamID = t.TeamID
        JOIN Session s ON (s.Team1ID = t.TeamID OR s.Team2ID = t.TeamID)
        WHERE r.CMN = cm.CMN
    )

    -- has never lost a game they played in
    AND NOT EXISTS (
        SELECT 1
        FROM Role r
        JOIN Team t ON r.TeamID = t.TeamID
        JOIN Session s ON (s.Team1ID = t.TeamID OR s.Team2ID = t.TeamID)
        WHERE r.CMN = cm.CMN
        AND (
            (t.TeamID = s.Team1ID AND s.Score1 < s.Score2 AND s.Score1 IS NOT NULL AND s.Score2 IS NOT NULL)
            OR
            (t.TeamID = s.Team2ID AND s.Score2 < s.Score1 AND s.Score1 IS NOT NULL AND s.Score2 IS NOT NULL)
        )
        AND s.Type = 'Game'
        )
        GROUP BY cm.CMN
        ORDER BY
            l.Name ASC,
            cm.CMN ASC;
    ";
```

17. Get a report of all the personnel who were treasurer of the club at least once or is currently a treasurer of the club. The report should include the treasurer's first name, last name, start date as a treasurer and last date as treasurer. If the last date as treasurer is null means that the personnel is the current treasurer of the club. Results should be displayed sorted in ascending order by first name then by last name then by start date as a treasurer.

```
SELECT
  p.FirstName,
  p.LastName,
  c.StartDate,
  c.EndDate
  -- joins
FROM Person p
JOIN Contract c ON p.PersonID = c.EmployeeID
    -- selects employees who have taken the role of treasurer at any time
WHERE c.Role = 'Treasurer'
ORDER BY p.FirstName, p.LastName, c.StartDate;
```

18. Get a report on all club members who were deactivated by the system because they became over 18 years old. Results should include the club member' first name, last name, telephone number, email address, deactivation date, last location name and last role when deactivated. Results should be displayed sorted in ascending order by location name, then by role, then by first name then by last name.

```
DROP VIEW IF EXISTS DeactivatedMembers, TeamsLastSession;

-- find all deactivated members over 18
CREATE VIEW DeactivatedMembers AS
SELECT
    c.CMN,
    p.FirstName,
    p.LastName,
    p.PhoneNumber,
    p.Email,
    DATE_ADD(LAST_DAY(DATE_ADD(p.DateOfBirth, INTERVAL 18 YEAR)),
    INTERVAL 1 DAY) AS DeactivationDate,
    c.LocationID
-- joins
FROM Person p
JOIN ClubMember c ON c.PersonID = p.PersonID
JOIN Location l ON c.LocationID = l.LocationID
-- condition: age over 18
WHERE p.DateOfBirth <= DATE_SUB(CURDATE(), INTERVAL 18 YEAR);
```

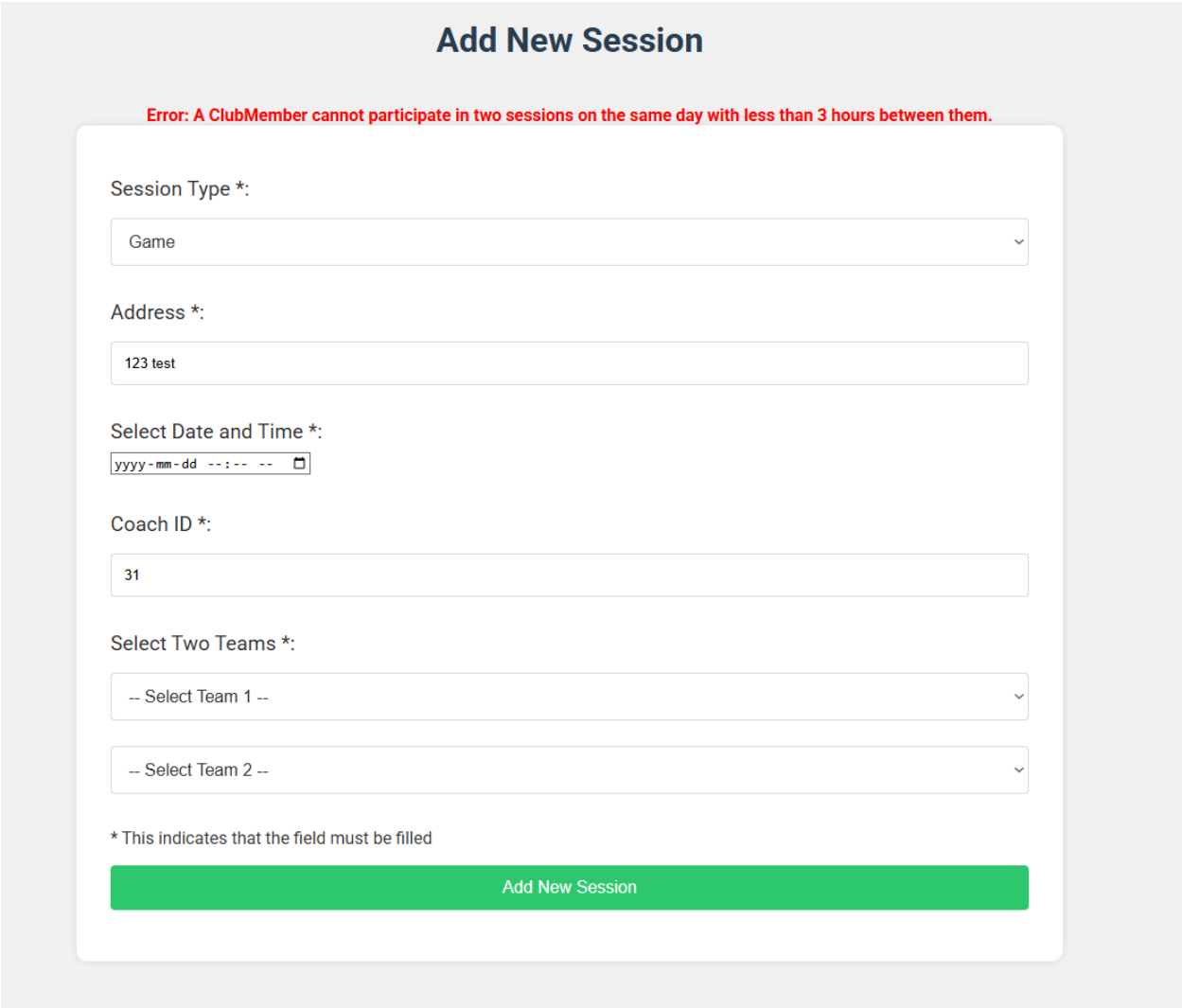
```sql
-- find the most recent/last session a team had
CREATE VIEW TeamsLastSession AS
SELECT t.TeamID,
        max(s.StartDateTime) AS LastSession
-- joins
FROM Team t, Session s
WHERE t.TeamID=s.Team1ID OR t.TeamID=s.Team2ID
GROUP BY t.TeamID;

  -- display deactivated members and their last location and role
  SELECT DISTINCT
      dm.FirstName,
      dm.LastName,
      dm.PhoneNumber,
      dm.Email,
      dm.DeactivationDate,
      l.Name AS LocationName,
      r.Position
  FROM DeactivatedMembers dm
  -- joins
  JOIN Role r ON dm.CMN = r.CMN
  JOIN TeamsLastSession tls ON r.TeamID = tls.TeamID
  JOIN (
      -- finds most recent session a member had
      SELECT r2.CMN, MAX(tls2.LastSession) AS MaxSession
      FROM Role r2
      JOIN TeamsLastSession tls2 ON r2.TeamID = tls2.TeamID
      GROUP BY r2.CMN
  ) latest ON r.CMN = latest.CMN AND tls.LastSession = latest.MaxSession
  JOIN Location l ON dm.LocationID = l.LocationID
  ORDER BY
      l.Name,
      r.Position,
      dm.FirstName,
      dm.LastName;
```

20. You need to demonstrate the integrity of all the requirements provided in the description. Example, the system should not allow a user to assign a player on two different formation sessions at the same time or on a conflicting time (less than three hours difference).

Since this tuples exist within Session:

2025-04-10 10:00:00	123 Main St	Training	Emma Taylor	Raptors (Edited)	25	2	Titans	Edit Cancel

We cannot add any session that uses either team within the next 3 hours.



21. You need to demonstrate the generation of emails and the logs of the emails produced by the system (at least one email for team formation, and one email for club member deactivation).

- Display -

```
$query = "
    SELECT
        l.Name AS LocationName,
        CONCAT(p.FirstName, ' ', p.LastName) AS FullName,
        e.Subject,
        DATE(e.Date) AS Date,
        DATE_FORMAT(e.Date, '%h:%i %p') AS Time,
        e.First100Chars AS Body
    FROM
        Email e
    JOIN
        Person p ON e.RecipientID = p.PersonID
    JOIN
        Location l ON e.LocationID = l.LocationID
    ORDER BY
        e.Date DESC
";
```

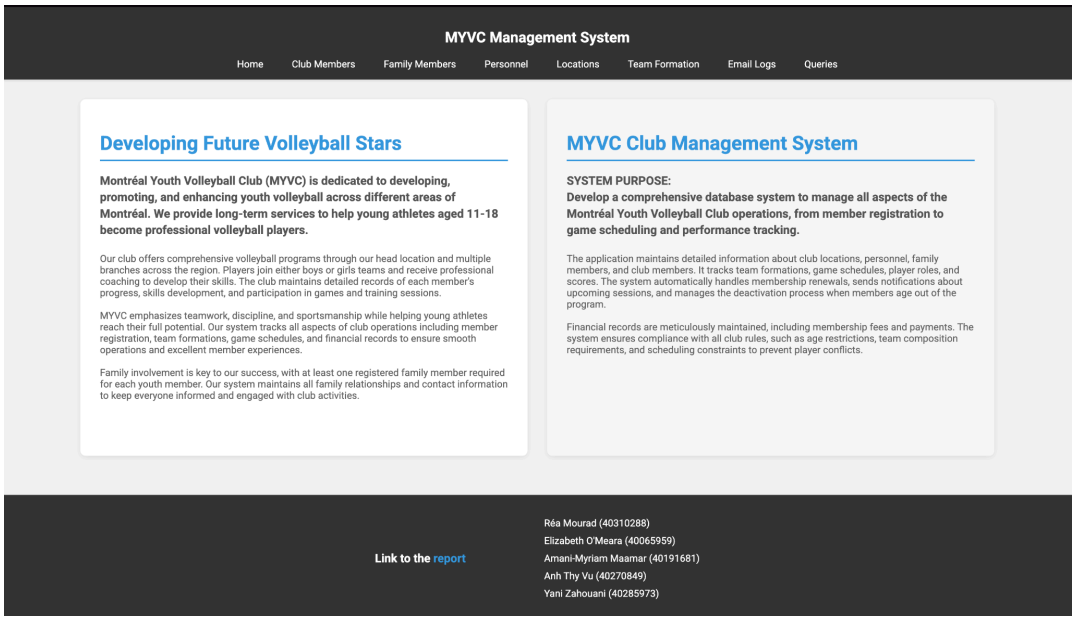
- Generate Email -

```
$emailQuery = "
    INSERT INTO Email (locationID, recipientID, Subject, Date, First100Chars) VALUES (?, ?, ?, ?, ?)
";
```

6. GUI Snapshot

6.1 Display

- Home Page -

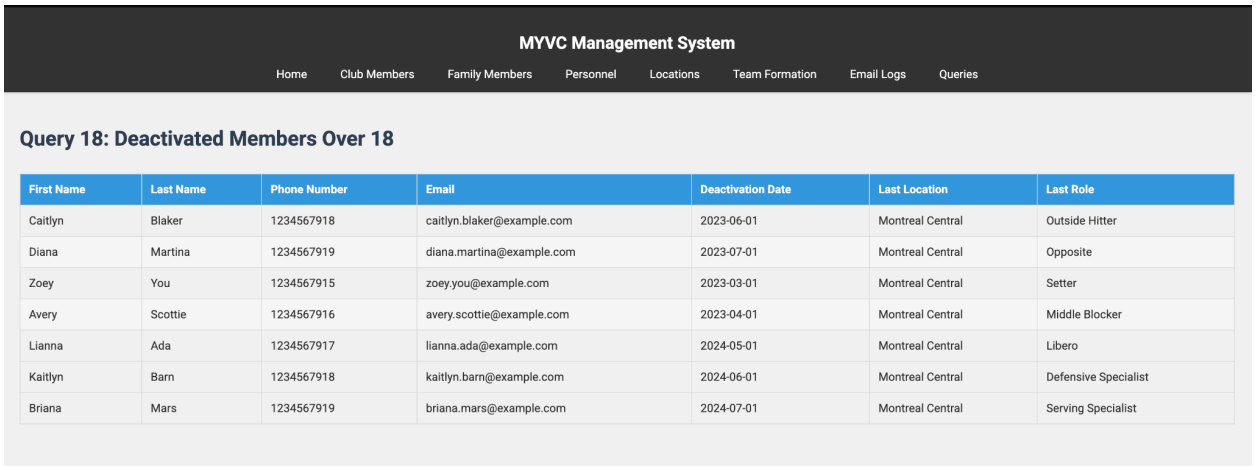


- Queries -

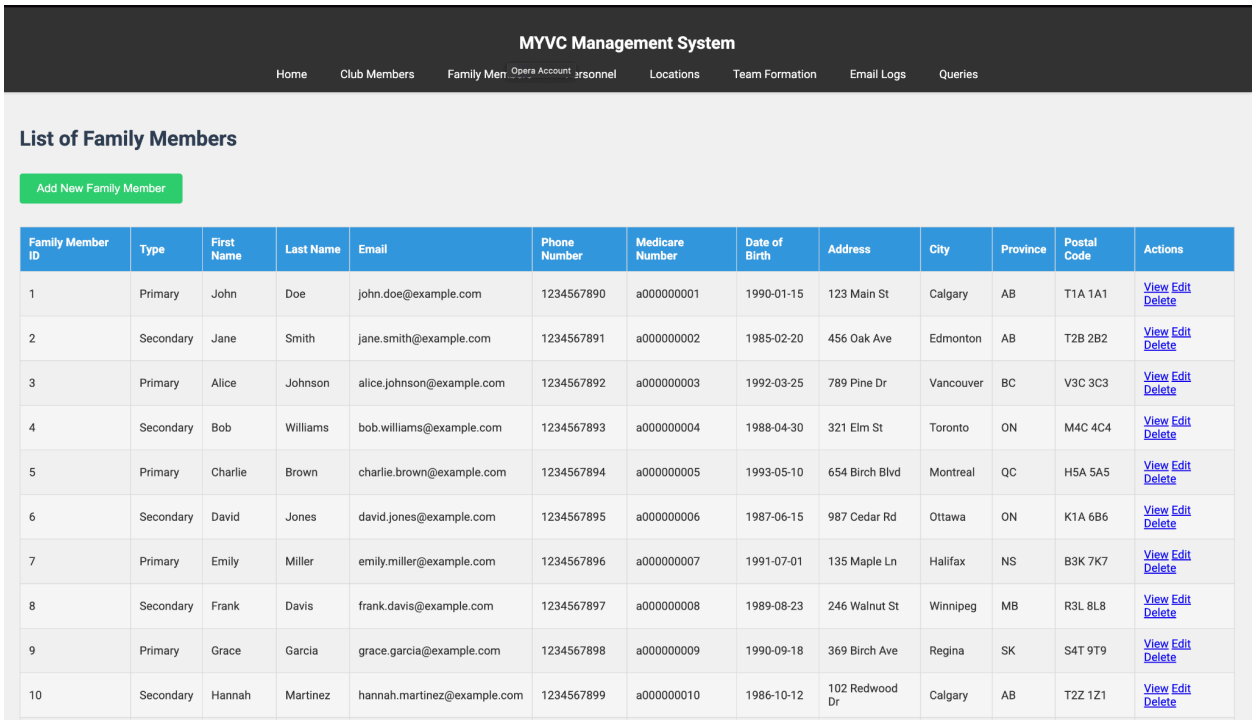
* All tables are displayed in a consistent format. Those who require CRUD functionality include additional forms to simplify the process.

(i.e., Display of all Family Members registered in our database)

- Without CRUD -



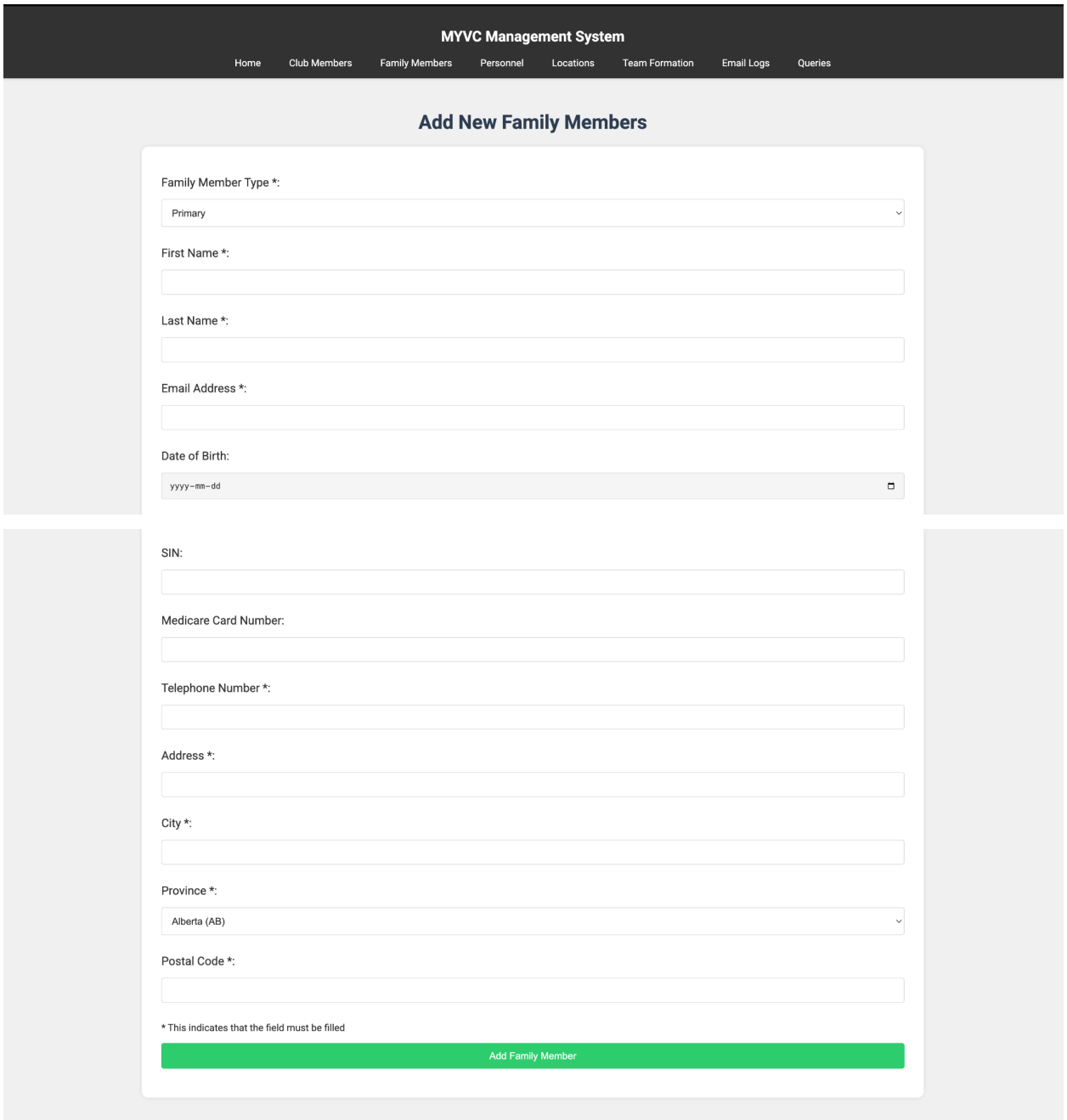
- With CRUD -


MYVC Management System
Home
Club Members
Family Men
Opera Account
ersonnel
Locations
Team Formation
Email Logs
Queries
List of Family Members
Add New Family Member
Family Member ID
Type
First Name
Last Name
Email
Phone Number
Medicare Number
Date of Birth
Address
City
Province
Postal Code
Actions
1 Primary John Doe john.doe@example.com 1234567890 a000000001 1990-01-15 123 Main St Calgary AB T1A 1A1 View Edit Delete
2 Secondary Jane Smith jane.smith@example.com 1234567891 a000000002 1985-02-20 456 Oak Ave Edmonton AB T2B 2B2 View Edit Delete
3 Primary Alice Johnson alice.johnson@example.com 1234567892 a000000003 1992-03-25 789 Pine Dr Vancouver BC V3C 3C3 View Edit Delete
4 Secondary Bob Williams bob.williams@example.com 1234567893 a000000004 1988-04-30 321 Elm St Toronto ON M4C 4C4 View Edit Delete
5 Primary Charlie Brown charlie.brown@example.com 1234567894 a000000005 1993-05-10 654 Birch Blvd Montreal QC H5A 5A5 View Edit Delete
6 Secondary David Jones david.jones@example.com 1234567895 a000000006 1987-06-15 987 Cedar Rd Ottawa ON K1A 6B6 View Edit Delete
7 Primary Emily Miller emily.miller@example.com 1234567896 a000000007 1991-07-01 135 Maple Ln Halifax NS B3K 7K7 View Edit Delete
8 Secondary Frank Davis frank.davis@example.com 1234567897 a000000008 1989-08-23 246 Walnut St Winnipeg MB R3L 8L8 View Edit Delete
9 Primary Grace Garcia grace.garcia@example.com 1234567898 a000000009 1990-09-18 369 Birch Ave Regina SK S4T 9T9 View Edit Delete
10 Secondary Hannah Martinez hannah.martinez@example.com 1234567899 a000000010 1986-10-12 102 Redwood Dr Calgary AB T2Z 1Z1 View Edit Delete

6.2 Add forms

* The add forms are consistent across the website for all queries requiring CRUD operations. Their implementation allows users to insert new tuples into the corresponding relations.

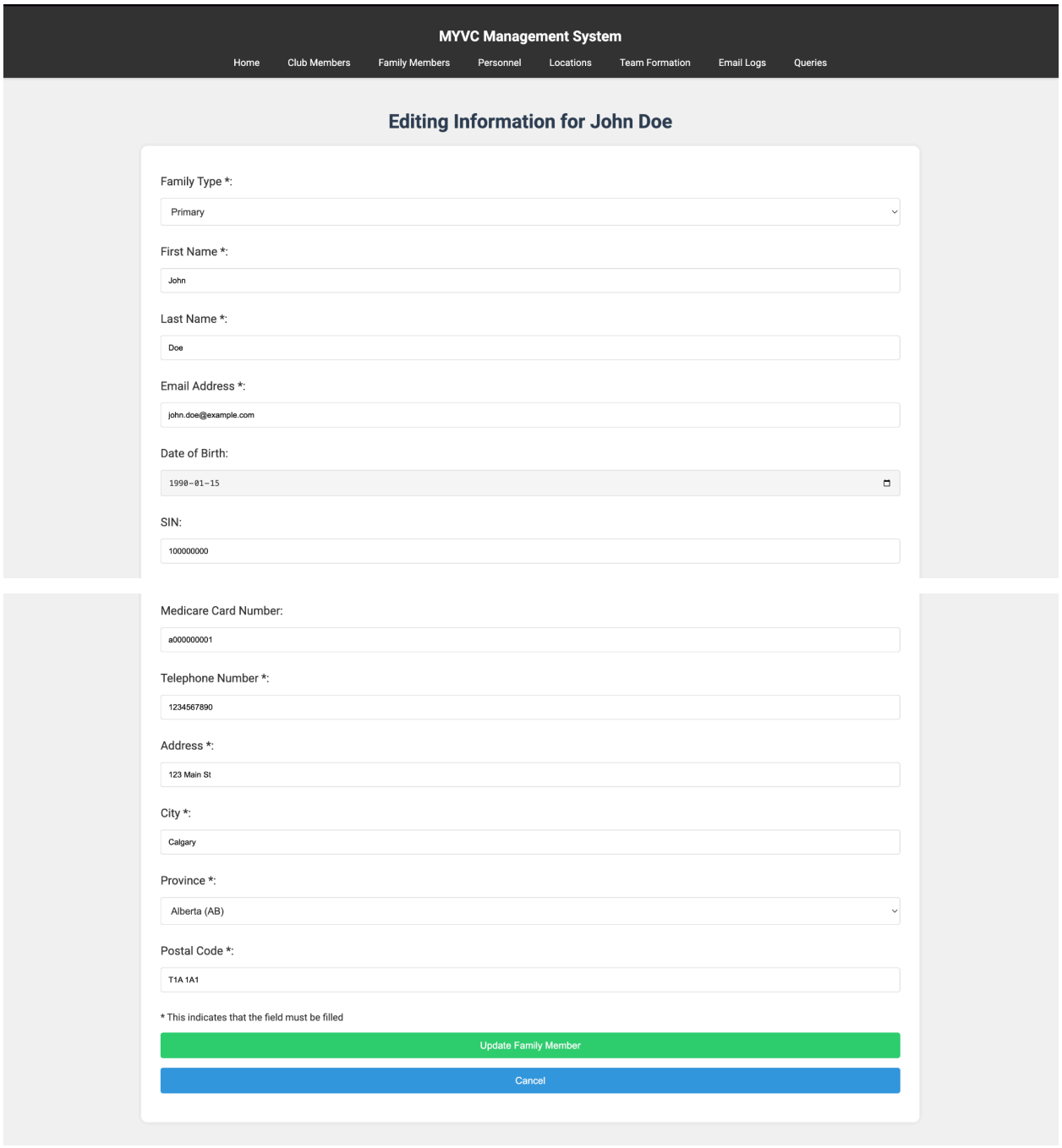
(i.e., Add form for inserting a new Family Member)



6.3 Edit Forms

* The edit forms are consistent throughout the website for all queries requiring CRUD functionality. They allow users to update existing tuples in the corresponding relations.

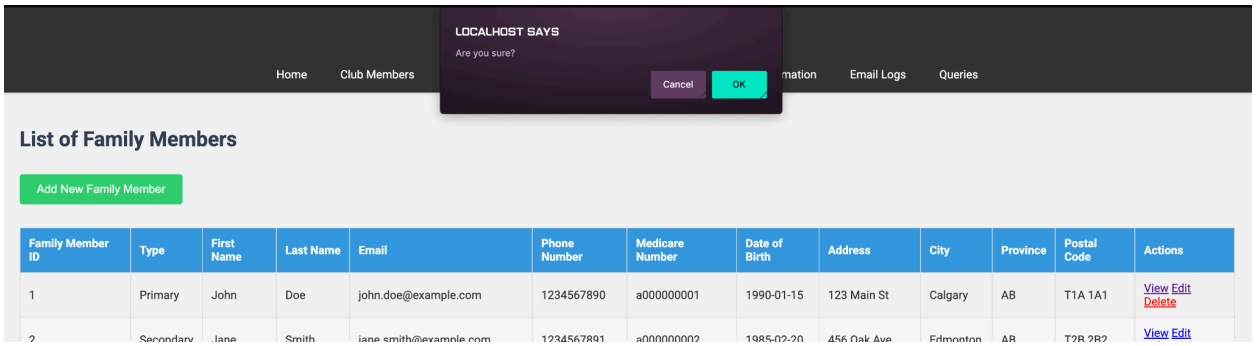
(i.e., Edit John Doe's information in FamilyMember)



6.4 Deleted Alert

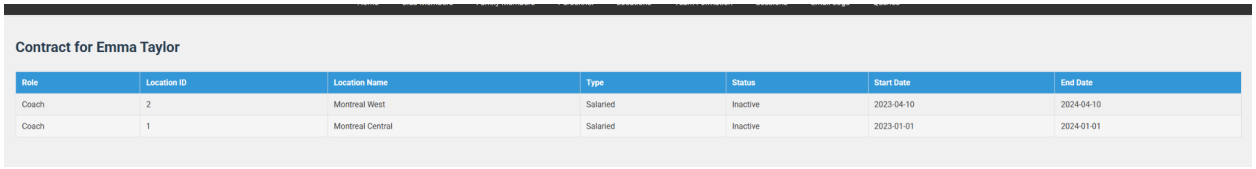
* This alert appears to confirm the action when a user attempts to remove a tuple from the corresponding relation.

(i.e., Remove John Doe in FamilyMember)

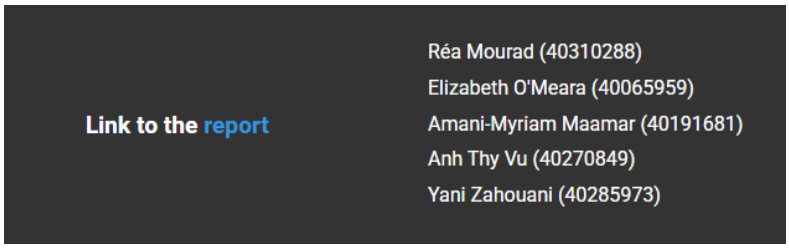


6.5 Additional Functionality

- Implemented dynamic display connected to each employee, showing their associated mandates. -

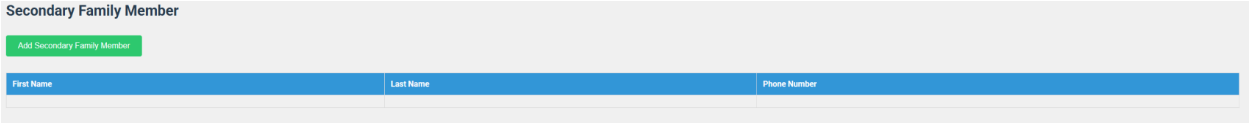


- Link the final report in the website -

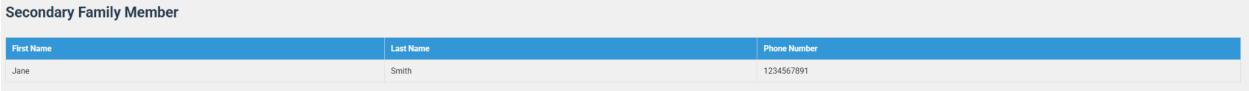


- Implemented a conditional display to restrict users from adding multiple secondary family member -

Without Secondary



With Secondary



7. Triggers

```
-- A club member inherits the alternate family member of their primary family member
DELIMITER $$
CREATE TRIGGER trg_check_club_member_alternate_family_id
BEFORE INSERT ON ClubMember
FOR EACH ROW
BEGIN
    DECLARE primaryAltFamilyID INT;

    -- Fetch the AlternateFamilyID of the associated PrimaryFamilyID
    SELECT AlternativeFamilyID INTO primaryAltFamilyID
    FROM FamilyMember
    WHERE PersonID = NEW.PrimaryFamilyID;

    -- Check if the ClubMember's AlternateFamilyID matches the PrimaryFamilyID's AlternateFamilyID
    IF NEW.AlternativeFamilyID IS NOT NULL AND NEW.AlternativeFamilyID != primaryAltFamilyID THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember AlternateFamilyID must match the PrimaryFamilyIDs AlternateFamilyID.';
    END IF;

    -- If the ClubMember's AlternateFamilyID is NULL, set it to match the PrimaryFamilyID's AlternateFamilyID
    IF NEW.AlternativeFamilyID IS NULL THEN
        SET NEW.AlternativeFamilyID = primaryAltFamilyID;
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER trg_update_club_member_alternate_family_id
AFTER UPDATE ON FamilyMember
FOR EACH ROW
BEGIN
    -- Check if the AlternativeFamilyID has been updated
    IF OLD.AlternativeFamilyID != NEW.AlternativeFamilyID THEN
        -- Update the AlternativeFamilyID of all ClubMembers associated with the updated PrimaryFamilyID
        UPDATE ClubMember
        SET AlternativeFamilyID = NEW.AlternativeFamilyID
        WHERE PrimaryFamilyID = NEW.PersonID;
    END IF;
END$$
DELIMITER ;
```

```sql
-- a club member must have a primary family member (isPrimary = true) as their primary family member
DELIMITER $$
CREATE TRIGGER trg_check_primary_family_member
BEFORE INSERT ON ClubMember
FOR EACH ROW
BEGIN
    DECLARE isPrimaryValue BOOLEAN;

    -- Check if the referenced FamilyMember has isPrimary set to TRUE
    SELECT isPrimary INTO isPrimaryValue
    FROM FamilyMember
    WHERE PersonID = NEW.PrimaryFamilyID;

    IF isPrimaryValue IS NULL OR isPrimaryValue = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The referenced FamilyMember in ClubMember must have isPrimary set to TRUE.';
    END IF;
END$$
DELIMITER ;


CREATE TRIGGER trg_check_update_primary_family_member
BEFORE UPDATE ON ClubMember
FOR EACH ROW
BEGIN
    DECLARE isPrimaryValue BOOLEAN;

    -- Check if the referenced FamilyMember has isPrimary set to TRUE
    SELECT isPrimary INTO isPrimaryValue
    FROM FamilyMember
    WHERE PersonID = NEW.PrimaryFamilyID;

    -- Raise an error if the referenced FamilyMember is not primary
    IF isPrimaryValue IS NULL OR isPrimaryValue = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The referenced FamilyMember in ClubMember must have isPrimary set to TRUE.';
    END IF;
END$$

DELIMITER ;



-- a club member must be between 11 and 18 at time of registration
DELIMITER $$
CREATE TRIGGER trg_check_club_member_age
BEFORE INSERT ON ClubMember
FOR EACH ROW
BEGIN
    DECLARE memberAge INT;

    -- Calculate the age of the ClubMember based on their DateOfBirth in the Person table
    SELECT TIMESTAMPDIFF(YEAR, DateOfBirth, CURDATE()) INTO memberAge
    FROM Person
    WHERE PersonID = NEW.PersonID;

    -- Ensure the age is between 11 and 18
    IF memberAge < 11 OR memberAge > 18 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember must be between 11 and 18 years of age.';
    END IF;
END$$
DELIMITER ;
```

```
-- all players in the same team must be associated with the same location
DELIMITER $$

CREATE TRIGGER trg_check_team_location_consistency
BEFORE INSERT ON Role
FOR EACH ROW
BEGIN
    DECLARE teamLocationID INT;
    DECLARE clubMemberLocationID INT;

    -- Get the LocationID of the Team
    SELECT LocationID INTO teamLocationID
    FROM Team
    WHERE TeamID = NEW.TeamID;

    -- Get the LocationID of the ClubMember
    SELECT LocationID INTO clubMemberLocationID
    FROM ClubMember
    WHERE CMN = NEW.CMN;

    -- Check if the ClubMember's LocationID matches the Team's LocationID
    IF teamLocationID IS NOT NULL AND clubMemberLocationID IS NOT NULL AND teamLocationID != clubMemberLocationID THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'All ClubMembers associated with the same Team must have the same Location.';
    END IF;
END$$

DELIMITER ;
```

```
-- all players in the same team must be of the same gender
DELIMITER $$

CREATE TRIGGER trg_check_consistency_before_role_insert
BEFORE INSERT ON Role
FOR EACH ROW
BEGIN
    DECLARE team_gender ENUM('M', 'F');

    -- Get the gender of the team that the club member is associated with
    SELECT Gender INTO team_gender
    FROM Team
    WHERE TeamID = NEW.TeamID;

    -- Check if the gender of the club member matches the team gender
    IF (SELECT Gender FROM ClubMember WHERE CMN = NEW.CMN) != team_gender THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The club member gender does not match the team gender.';
    END IF;
END;
$$

DELIMITER ;
```

```
-- the two teams in the same formation must be of the same gender
DELIMITER $$

CREATE TRIGGER trg_check_gender_in_session_before_insert
BEFORE INSERT ON Session
FOR EACH ROW
BEGIN
    DECLARE gender_team1 ENUM('M', 'F');
    DECLARE gender_team2 ENUM('M', 'F');

    -- Get the gender of Team1
    SELECT Gender INTO gender_team1
    FROM Team
    WHERE TeamID = NEW.Team1ID;

    -- Get the gender of Team2
    SELECT Gender INTO gender_team2
    FROM Team
    WHERE TeamID = NEW.Team2ID;

    -- Check if both teams have the same gender
    IF gender_team1 != gender_team2 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Both teams in the same session must have the same gender.';
    END IF;
END;
$$

DELIMITER ;
```

```
-- if a player is in two team sessions on the same day, the time between the starts must be at least 3 hours
DELIMITER $$
CREATE TRIGGER trg_check_session_time_gap
BEFORE INSERT ON Session
FOR EACH ROW
BEGIN
    DECLARE team1CMN INT;
    DECLARE team2CMN INT;
    DECLARE overlappingSessionCount INT;

    -- Check if there is any overlap for Team1
    SELECT COUNT(*)
    INTO overlappingSessionCount
    FROM Session S
    JOIN Role R1 ON S.Team1ID = R1.TeamID OR S.Team2ID = R1.TeamID
    JOIN Role R2 ON NEW.Team1ID = R2.TeamID OR NEW.Team2ID = R2.TeamID
    WHERE R1.CMN = R2.CMN
      AND DATE(S.StartDateTime) = DATE(NEW.StartDateTime)
      AND ABS(TIMESTAMPDIFF(HOUR, S.StartDateTime, NEW.StartDateTime)) < 3;

    -- If there is an overlap, raise an error
    IF overlappingSessionCount > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A ClubMember cannot participate in two sessions on the same day with less than 3 hours between them.';
    END IF;
END$$
DELIMITER ;
```

```sql
-- only one location is marked as the 'Head'
DELIMITER $$

CREATE TRIGGER trg_ensure_one_head_location
BEFORE INSERT ON Location
FOR EACH ROW
BEGIN
    DECLARE headCount INT;

    -- Check if there is already a location marked as 'Head'
    SELECT COUNT(*)
    INTO headCount
    FROM Location
    WHERE Type = 'Head';

    -- If a 'Head' location already exists and the new entry is also 'Head', raise an error
    IF headCount > 0 AND NEW.Type = 'Head' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Only one location can be marked as "Head".';
    END IF;
END$$

DELIMITER ;

-- Only one location can be marked as head on update
DELIMITER $$

CREATE TRIGGER trg_ensure_one_head_location_update
BEFORE UPDATE ON Location
FOR EACH ROW
BEGIN
    DECLARE headCount INT;

    -- Check if there is already a location marked as 'Head', excluding the current one being updated
    SELECT COUNT(*)
    INTO headCount
    FROM Location
    WHERE Type = 'Head' AND LocationID != OLD.LocationID;

    -- If a 'Head' location already exists and the updated entry is also 'Head', raise an error
    IF headCount > 0 AND NEW.Type = 'Head' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Only one location can be marked as "Head".';
    END IF;
END$$

DELIMITER ;
```

```sql
-- Upon creation of a session, the team's captain must be a family members of a club member

DELIMITER $$

CREATE TRIGGER trg_check_team_captain_in_session
BEFORE INSERT ON Session
FOR EACH ROW
BEGIN
    DECLARE team1CaptainValid INT;
    DECLARE team2CaptainValid INT;

    -- Check if Team1's Captain is the PrimaryFamilyID of a ClubMember with a Role in Team1
    SELECT COUNT(*)
    INTO team1CaptainValid
    FROM ClubMember CM
    JOIN Role R ON CM.CMN = R.CMN
    WHERE CM.PrimaryFamilyID = (SELECT Captain FROM Team WHERE TeamID = NEW.Team1ID)
      AND R.TeamID = NEW.Team1ID;

    -- If Team1's Captain is invalid, raise an error
    IF team1CaptainValid = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The Captain of Team1 must be the PrimaryFamilyID of a ClubMember with a Role in the team.';
    END IF;

    -- Check if Team2's Captain is the PrimaryFamilyID of a ClubMember with a Role in Team2
    SELECT COUNT(*)
    INTO team2CaptainValid
    FROM ClubMember CM
    JOIN Role R ON CM.CMN = R.CMN
    WHERE CM.PrimaryFamilyID = (SELECT Captain FROM Team WHERE TeamID = NEW.Team2ID)
      AND R.TeamID = NEW.Team2ID;

    -- If Team2's Captain is invalid, raise an error
    IF team2CaptainValid = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The Captain of Team2 must be the PrimaryFamilyID of a ClubMember with a Role in the team.';
    END IF;
END$$

DELIMITER ;



-- A team captain must be the parent of someone assigned to the team on update
DELIMITER $$

CREATE TRIGGER trg_check_team_captain
BEFORE UPDATE ON Team
FOR EACH ROW
BEGIN
    DECLARE validCaptain INT;

    -- Check if the Captain is the PrimaryFamilyID of a ClubMember with a Role in the Team
    SELECT COUNT(*)
    INTO validCaptain
    FROM ClubMember CM
    JOIN Role R ON CM.CMN = R.CMN
    WHERE CM.PrimaryFamilyID = NEW.Captain
      AND R.TeamID = NEW.TeamID;

    -- If no valid captain is found, raise an error
    IF validCaptain = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The Captain must be the PrimaryFamilyID of a ClubMember with a Role in the Team.';
    END IF;
END$$

DELIMITER ;
```

```sql
-- check that the addition of a club member to a team doesn't cause a time conflict
DELIMITER $$

CREATE TRIGGER trg_check_club_member_session_time_gap
BEFORE INSERT ON Role
FOR EACH ROW
BEGIN
    DECLARE overlappingSessionCount INT;

    -- Check if the ClubMember is already associated with a session that overlaps with a session of the new team
    SELECT COUNT(*)
    INTO overlappingSessionCount
    FROM Session S1
    JOIN Role R1 ON S1.Team1ID = R1.TeamID OR S1.Team2ID = R1.TeamID
    JOIN Session S2 ON S2.Team1ID = NEW.TeamID OR S2.Team2ID = NEW.TeamID
    WHERE R1.CMN = NEW.CMN
      AND DATE(S1.StartDateTime) = DATE(S2.StartDateTime)
      AND ABS(TIMESTAMPDIFF(HOUR, S1.StartDateTime, S2.StartDateTime)) < 3;

    -- If there is an overlap, raise an error
    IF overlappingSessionCount > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A ClubMember cannot be associated with two sessions less than 3 hours apart.';
    END IF;
END$$

DELIMITER ;
```