

**[1. Introduction 3](#_gkkfacsgv19o)**

**[2. Restriction and assumptions 3](#_z6q2krge054h)**

**[2.1. Assumptions 3](#_30x96kp8t1v9)**

**[2.2. Restriction 3](#_cnaii9ouacmd)**

**[3. Database Design 3](#_w0g4ix6947hv)**

**[3.1. Entity-Relationship (E/R) Diagram 5](#_tugyy6bw8d6w)**

**[3.2. Database Design Considerations 6](#_ipym4xa1lyiy)**

[3.2.1. Locations Table 6](#_qj78q9rh44y9)

[3.2.2. Personnel Table 6](#_ihw2zga3i0br)

[3.2.3. Personnel\_Assignments Table 7](#_o7ewot4j5l02)

[3.2.4. CommonInfo Table 8](#_jfhu2jdiiz1l)

[3.2.5. FamilyMembers Table 9](#_98qzfcqjtlzo)

[3.2.6. ClubMembers Table 10](#_t1txbr8mnby4)

[3.2.7. Memberships Table 11](#_ou7beu8nkvpx)

[3.2.8. Payments Table 12](#_45w5i0d6z5js)

[3.2.9. Teams Table 12](#_ey43b4qu8f1)

[3.2.10. ClubMembers\_Teams Table 14](#_bu7r0z7ytxgb)

**[4. SQL scripts 15](#_vtwqxmmllt0c)**

**[4.1. DDL Statements 15](#_8zlil0u8kq41)**

[4.2.1 Period 15](#_4aalb06l6fje)

**[4.2. DML Statements (Transactions) 15](#_cdioh5dfs3y9)**

**[4.3. DML Statements (Queries) 15](#_ahpis7ksdwh4)**

**[4.4. Count \* Statements 15](#_rmx26oxnxdak)**

**[5. Conclusion 15](#_tnykpmith9kk)**

### 

# Introduction

The Montreal Youth Volleyball Club (MYVC) database system has been developed to manage club operations across multiple locations. The system is responsible for handling memberships, personnel assignments, financial transactions, family relationships, and team management. It ensures data integrity, security, and accessibility for authorized users while providing reporting and tracking features.

This database is designed to support the following main operations:

1. Registration and management of club members, families, and personnel.
2. Membership payments and donation tracking.
3. Personnel role assignments and mandate classifications.
4. Organization of teams and personnel assignments per location.
5. Comprehensive reporting on club operations, such as membership status, payments, personnel, and family relationships.

This system ensures structured data management using an ER model, which is then converted into a relational schema to maintain consistency and integrity. The system adheres to business rules and constraints to provide an optimized, scalable, and maintainable solution for club operations.

# Restriction and assumptions

## Assumptions

To ensure the database meets MYVC’s operational requirements, several design constraints and assumptions were established:

1. Single Manager Per Location: Each club location, (branch or head office), has only one general manager at any given time.
2. Exclusive Personnel Roles: Personnel can only have one role at a time, such as Coach, Treasurer, or General Manager.
3. Family Membership Structure: A club member can only be registered under one family member.
4. Annual Membership Payments: Payments for membership fees are recorded on an annual basis. Any amount paid beyond the $100 annual fee is automatically categorized as a donation.
5. Age Restrictions for Club Members: Only individuals between 11 and 18 years old can register as club members.
6. Fee Payment Requirement for Participation: A club member must fully pay the $100 annual membership fee before they can participate in any club activities.

## Restriction

To ensure data consistency and enforce business rules, the following constraints are implemented:

1. Unique Identification Requirements: Social Security Numbers (SSN) and Medicare Card numbers must be unique across all personnel and family members .
2. Unique Club Membership Identifiers: Each ClubMemberID should be globally unique.
3. Foreign Key Constraints: Relationships between tables are enforced using foreign keys.
4. Annual Fee Enforcement: Every club member must pay the full $100 annual membership fee to maintain an active status.
5. Donation Handling: Any payment exceeding $100 per year is automatically classified as a donation in the system
6. Installment Payments Allowed: Payments can be made in up to four installments per year.

## **Database Design**

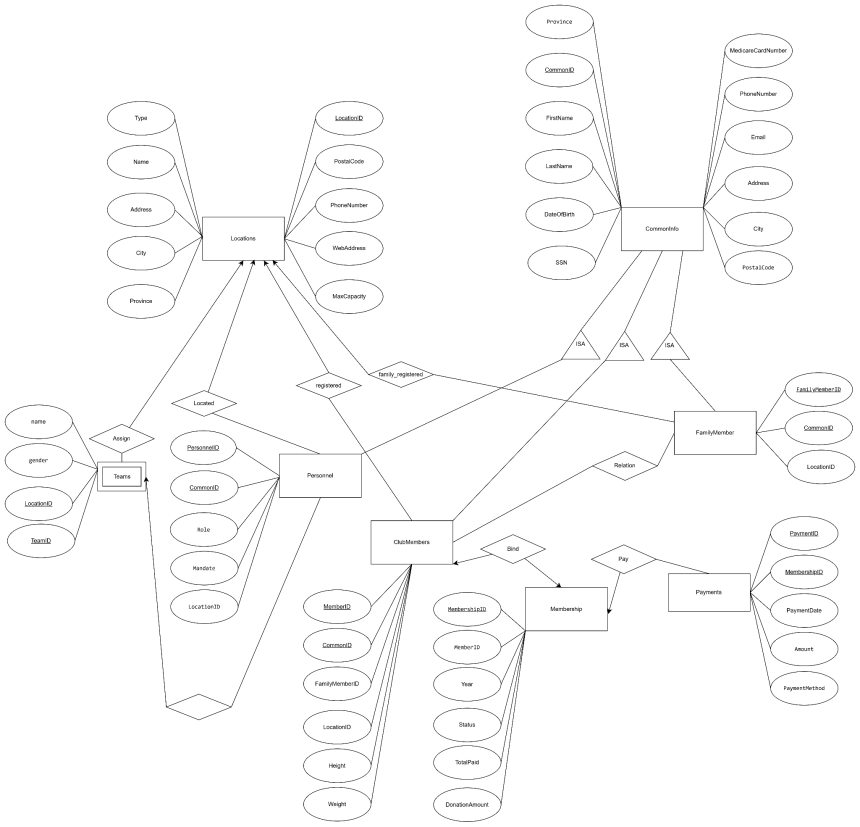


Image 1 E/R diagram

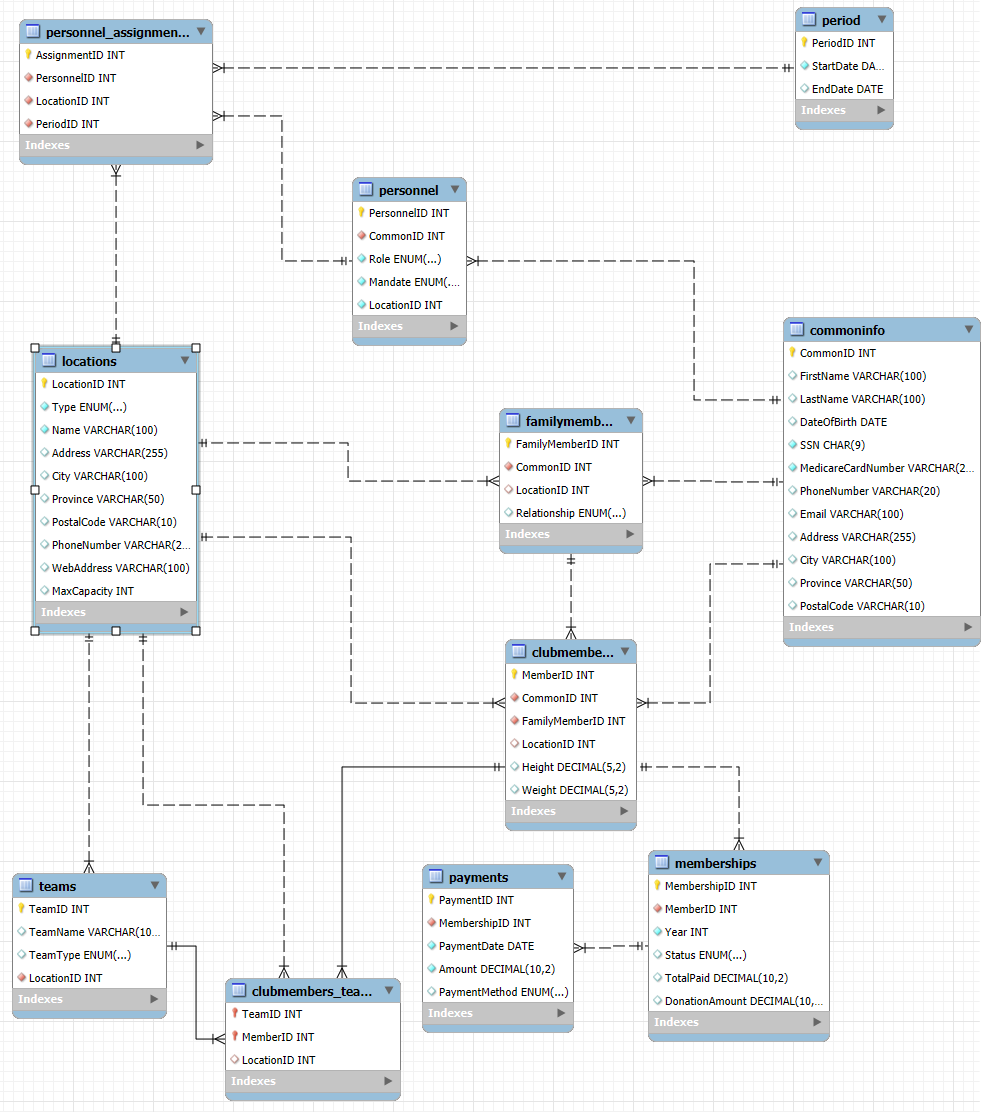


Image 2 Mysql reverse engineer E/R diagram

### Entity-Relationship (E/R) Diagram

The MYVC database is designed based on an E/R model that captures the relationships between key entities such as locations, personnel, family members, club members, teams, and payments. The key constraints and relationships include:

**One-to-Many Relationships:** A location can have multiple personnel and club members. A family member can have multiple associated club members. A team can consist of multiple club members.

**Many-to-Many Relationships:** Club members can be part of multiple teams, and teams can have multiple club members. Personnel can work at different locations during different time periods.

## Database Design Considerations

### Locations Table

Stores details about the club’s head office and branches.

Attributes: LocationID (PK), Type, Name, Address, City, Province, PostalCode, PhoneNumber, WebAddress, Capacity.

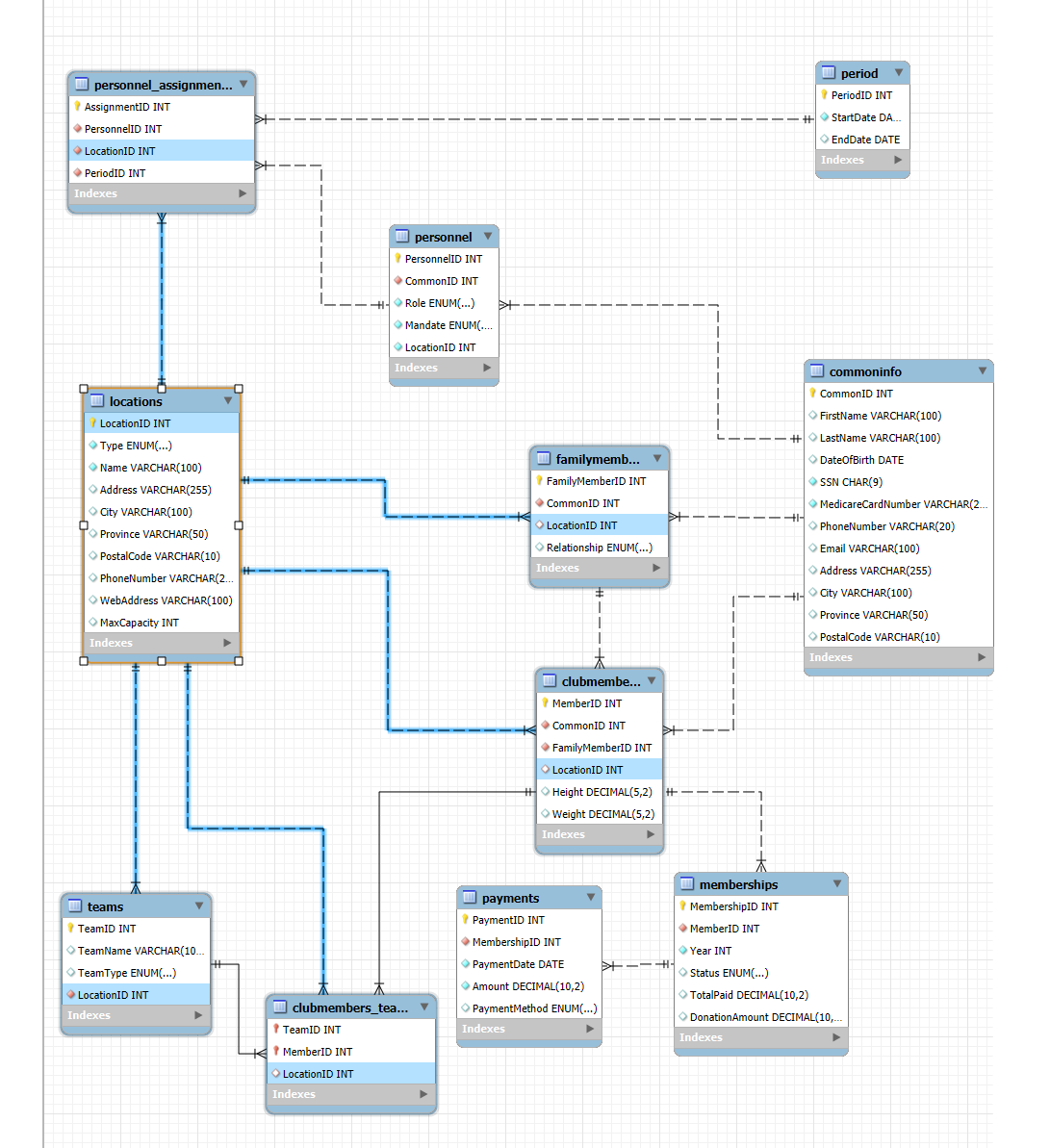


Image 3 Locations Table

In this table, LocationID is the primary key. This entity is related to 5 entities within the database schema, ensuring the data integration and removing redundancy.

### Personnel Table

Stores personnel working at different locations.

Attributes: PersonnelID (PK), CommonInfoID (FK), LocationID (FK), Role, Mandate

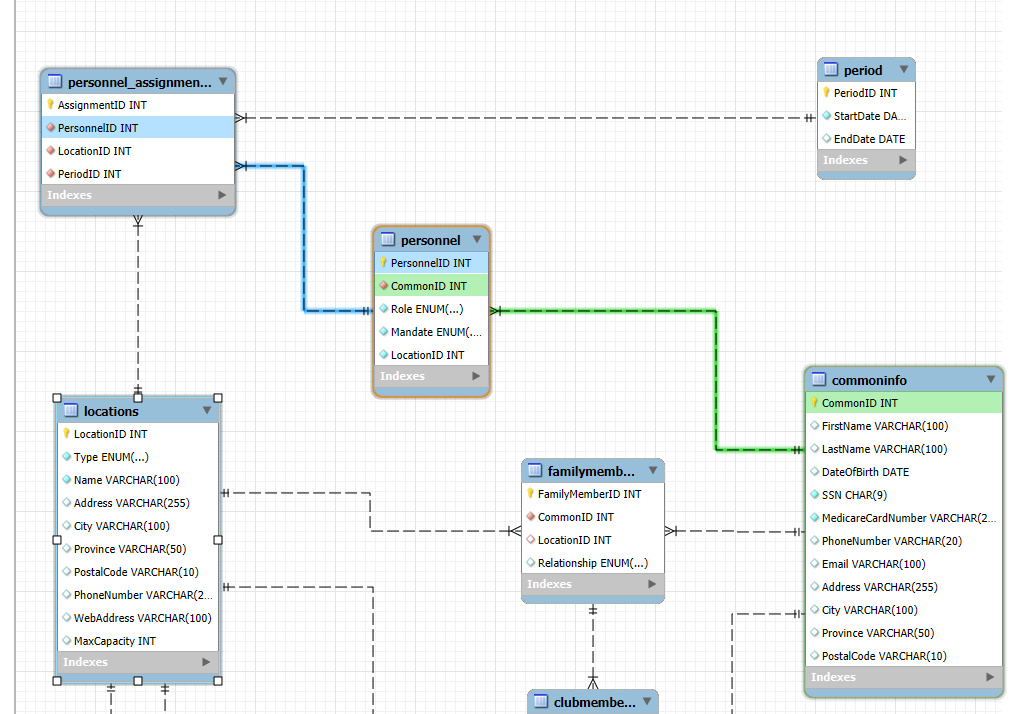


Image 4 Personnel table

For the personnel table, this is inheritance from Commoninfo with role and LocationID which indicates current location.

### Personnel\_Assignments Table

Tracks personnel working at different locations over time.

Attributes: AssignmentID (PK), PersonnelID (FK), LocationID (FK), StartDate, EndDate

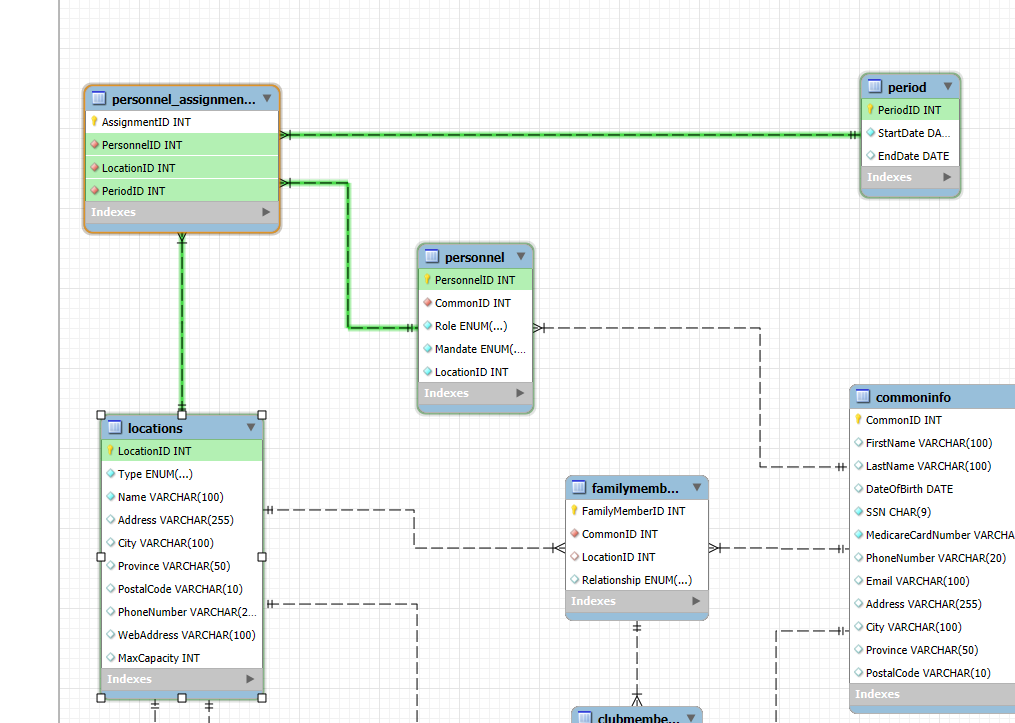


Image 5 Personnel\_Assignments

This table will store all information about the same personnel working at different locations at different periods.

### CommonInfo Table

Stores shared personal information for personnel, family members, and club members.

Attributes: CommonInfoID (PK), FirstName, LastName, SSN, MedicareCard, DOB, PhoneNumber, Email, Address, City, Province, PostalCode

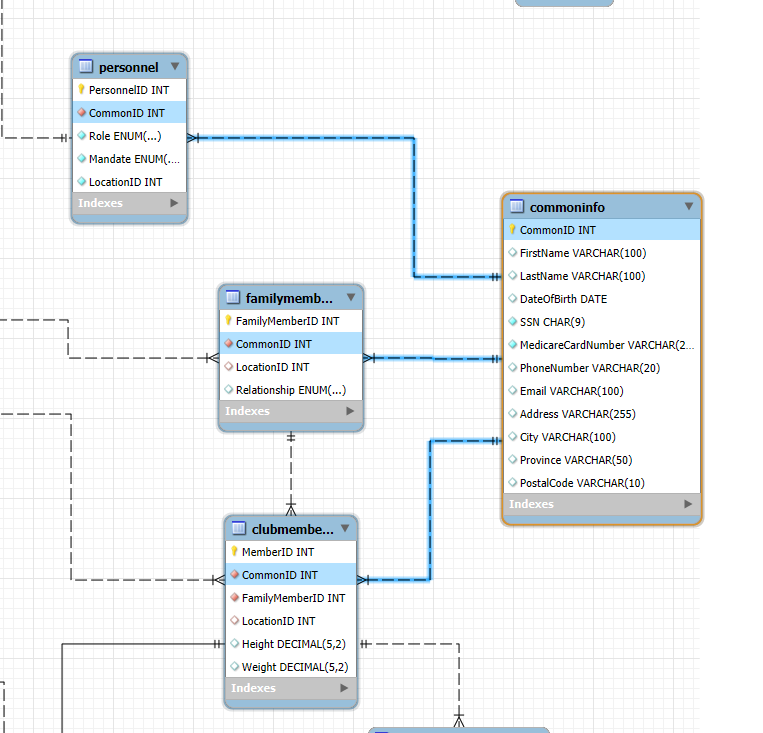


Image 6 CommonInfo Table

Stores general personal information for individuals in the system. Acts as a base entity for Personnel, FamilyMembers, and ClubMembers, which inherit common attributes such as name, contact details, and address. Referenced by Personnel, FamilyMembers, and ClubMembers.

### FamilyMembers Table

Stores information about family members who register children in the club.

Attributes: FamilyMemberID (PK), CommonInfoID (FK), LocationID (FK)

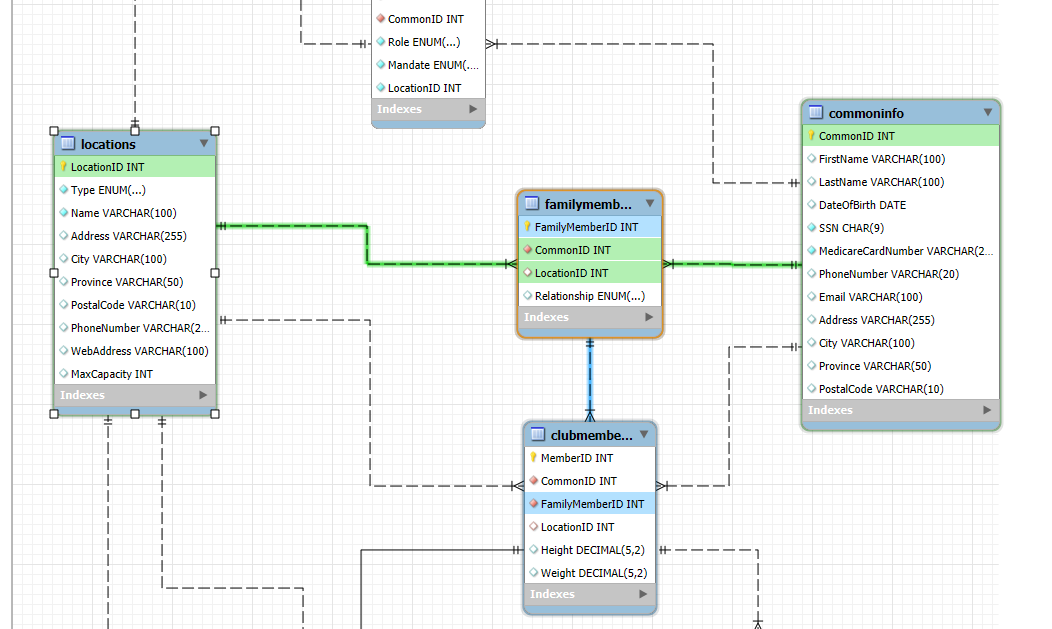


Image 7 FamilyMembers Table

Stores information about family members of club members. Inherits personal details from CommonInfo. References Locations to indicate their associated club. Referenced by ClubMembers.

### ClubMembers Table

Stores information about youth members.

Attributes: ClubMemberID (PK), CommonInfoID (FK), FamilyMemberID (FK), LocationID (FK), Height, Weight, Status (Active/Inactive)

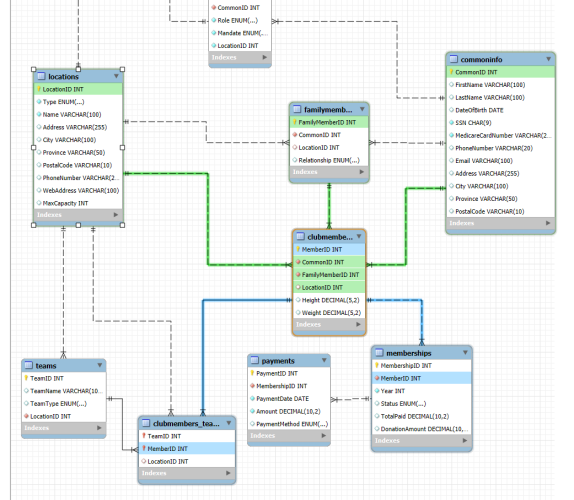


Image 8 ClubMembers Table

Stores information about club members. Inherits personal details from CommonInfo. References FamilyMembers to indicate family relationships. References Locations for the club they belong to. Referenced by Memberships and ClubMembers\_Teams.

### Memberships Table

Stores membership status and registration details.

Attributes: MembershipID (PK), ClubMemberID (FK), Status (Active/Inactive)

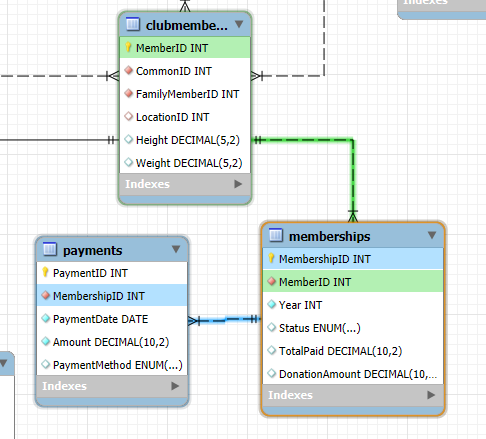


Image 9 Memberships Table

Tracks membership status and payments for club members. References ClubMembers. Referenced by Payments.

### Payments Table

Tracks membership fees and financial transactions.

Attributes: PaymentID (PK), MembershipID (FK), PaymentDate, Amount, Method

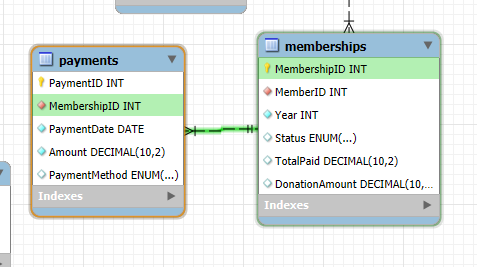


Image 10 Payments Table

Stores payment records for memberships. References Memberships to track which membership received the payment.

### Teams Table

Stores information about club teams.

Attributes: TeamID (PK), LocationID (FK), TeamName, TeamType (Boys/Girls)

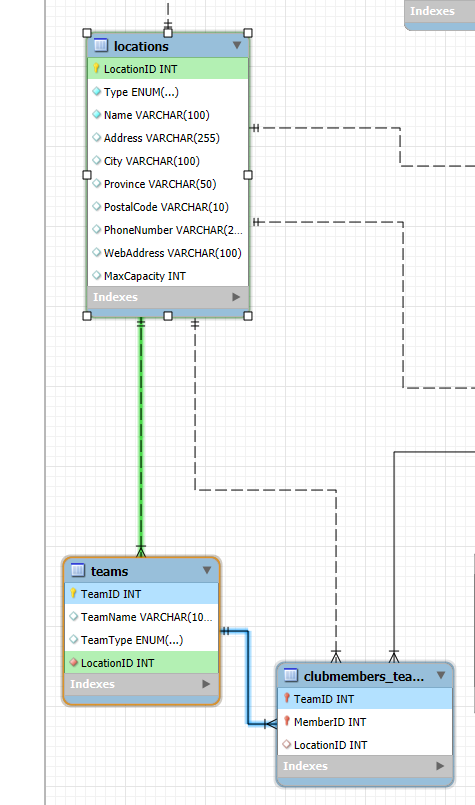


Image 11 Teams

Stores team details, including type (Boys or Girls) and associated location. References Locations. Referenced by ClubMembers\_Teams.

### ClubMembers\_Teams Table

Represents the many-to-many relationship between club members and teams.

Attributes: ClubMemberTeamID (PK), ClubMemberID (FK), TeamID (FK), LocationID (FK)

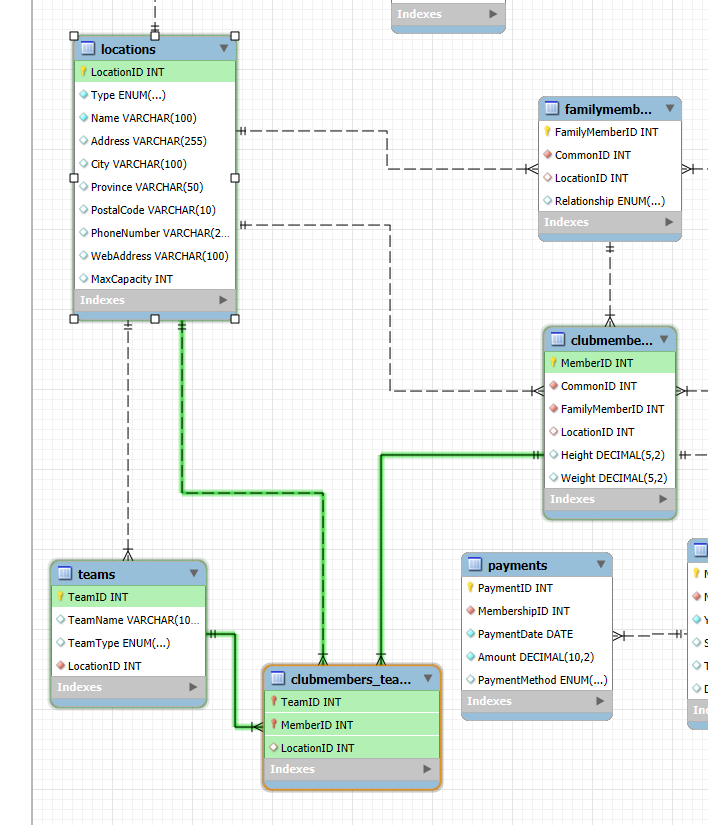


Image 12 ClubMembers\_Teams

Establishes many-to-many relationships between club members and teams. References Teams, ClubMembers, and Locations.

## **SQL scripts**

## DDL Statements

CREATE TABLE Period (

PeriodID INT AUTO\_INCREMENT PRIMARY KEY,

StartDate DATE NOT NULL,

EndDate date

);

CREATE TABLE Locations (

LocationID INT AUTO\_INCREMENT PRIMARY KEY,

Type ENUM('Head', 'Branch') NOT NULL,

Name VARCHAR(100) NOT NULL,

Address VARCHAR(255),

City VARCHAR(100),

Province VARCHAR(50),

PostalCode VARCHAR(10),

PhoneNumber VARCHAR(20),

WebAddress VARCHAR(100),

MaxCapacity INT

);

CREATE TABLE CommonInfo (

CommonID INT AUTO\_INCREMENT PRIMARY KEY,

FirstName VARCHAR(100),

LastName VARCHAR(100),

DateOfBirth DATE,

SSN CHAR(9) UNIQUE NOT NULL,

MedicareCardNumber VARCHAR(20) UNIQUE NOT NULL,

PhoneNumber VARCHAR(20),

Email VARCHAR(100),

Address VARCHAR(255),

City VARCHAR(100),

Province VARCHAR(50),

PostalCode VARCHAR(10)

);

CREATE TABLE Personnel (

PersonnelID INT AUTO\_INCREMENT PRIMARY KEY,

CommonID INT NOT NULL,

LocationID INT NOT NULL,

Role ENUM('Administrator', 'Captain', 'Coach', 'Assistant Coach', 'Other', 'General Manager', 'Deputy Manager', 'Treasurer', 'Secretary') NOT NULL,

Mandate ENUM('Volunteer', 'Salaried') NOT NULL,

FOREIGN KEY (CommonID) REFERENCES CommonInfo(CommonID),

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)

);

CREATE TABLE Personnel\_Assignments (

AssignmentID INT AUTO\_INCREMENT PRIMARY KEY,

PersonnelID INT NOT NULL,

LocationID INT NOT NULL,

PeriodID INT NOT NULL,

FOREIGN KEY (PersonnelID) REFERENCES Personnel(PersonnelID),

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),

FOREIGN KEY (PeriodID) REFERENCES Period(PeriodID)

);

CREATE TABLE FamilyMembers (

FamilyMemberID INT AUTO\_INCREMENT PRIMARY KEY,

CommonID INT NOT NULL,

LocationID INT,

Relationship ENUM('Father', 'Mother', 'Grandfather', 'Grandmother', 'Tutor', 'Partner', 'Friend', 'Other'),

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),

FOREIGN KEY (CommonID) REFERENCES CommonInfo(CommonID)

);

CREATE TABLE ClubMembers (

MemberID INT AUTO\_INCREMENT PRIMARY KEY,

CommonID INT NOT NULL,

FamilyMemberID INT NOT NULL,

LocationID INT,

Height DECIMAL(5, 2),

Weight DECIMAL(5, 2),

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),

FOREIGN KEY (CommonID) REFERENCES CommonInfo(CommonID),

FOREIGN KEY (FamilyMemberID) REFERENCES FamilyMembers(FamilyMemberID)

);

CREATE TABLE Memberships (

MembershipID INT AUTO\_INCREMENT PRIMARY KEY,

MemberID INT NOT NULL,

Year INT NOT NULL,

Status ENUM('Active', 'Inactive') DEFAULT 'Active',

TotalPaid DECIMAL(10, 2) DEFAULT 0,

DonationAmount DECIMAL(10, 2) AS (GREATEST(TotalPaid - 100, 0)) STORED,

FOREIGN KEY (MemberID) REFERENCES ClubMembers(MemberID)

);

CREATE TABLE Teams (

TeamID INT AUTO\_INCREMENT PRIMARY KEY,

TeamName VARCHAR(100),

TeamType ENUM('Boys', 'Girls'),

LocationID INT NOT NULL,

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)

);

CREATE TABLE ClubMembers\_Teams (

TeamID INT,

MemberID INT,

LocationID INT,

PRIMARY KEY (TeamID, MemberID),

FOREIGN KEY (TeamID) REFERENCES Teams(TeamID),

FOREIGN KEY (MemberID) REFERENCES ClubMembers(MemberID),

FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)

);

CREATE TABLE Payments (

PaymentID INT AUTO\_INCREMENT PRIMARY KEY,

MembershipID INT NOT NULL,

PaymentDate DATE NOT NULL,

Amount DECIMAL(10, 2) NOT NULL,

PaymentMethod ENUM('Cash', 'Debit', 'Credit'),

FOREIGN KEY (MembershipID) REFERENCES Memberships(MembershipID)

);

## DML Statements (Transactions)

**Insert Transaction**

INSERT INTO Locations (Type, Name, Address, City, Province, PostalCode, PhoneNumber, MaxCapacity)

VALUES

('Head', 'Montreal Downtown', '123 Montreal Downtown St.', 'Montreal', 'QC', 'Q1X 806', '514-555-5079', 98),

('Branch', 'Plateau Mont-Royal', '123 Plateau Mont-Royal St.', 'Montreal', 'QC', 'Q3T 611', '514-555-8635', 118),

('Branch', 'Outremont', '123 Outremont St.', 'Montreal', 'QC', 'Q4C 725', '514-555-7097', 85),

('Branch', 'Villeray', '123 Villeray St.', 'Montreal', 'QC', 'Q1D 911', '514-555-7800', 151),

('Branch', 'Verdun', '123 Verdun St.', 'Montreal', 'QC', 'Q6W 197', '514-555-6117', 192);

## DML Statements (Queries)

### Get complete details for every location

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, number of personnel, and the number of club members associated with that location. The results should be displayed sorted in ascending order by Province, then by city.

SELECT

l.Name AS LocationName,

l.Address,

l.City,

l.Province,

l.PostalCode,

l.PhoneNumber,

l.WebAddress,

l.Type AS LocationType,

l.MaxCapacity,

(SELECT FirstName FROM Personnel p JOIN CommonInfo c ON p.CommonID = c.CommonID WHERE p.Role = 'General Manager' AND p.LocationID = l.LocationID LIMIT 1) AS GeneralManagerName,

(SELECT COUNT(\*) FROM Personnel p WHERE p.LocationID = l.LocationID) AS PersonnelCount,

(SELECT COUNT(\*) FROM ClubMembers cm WHERE cm.LocationID = l.LocationID) AS ClubMembersCount

FROM Locations l

ORDER BY l.Province, l.City;

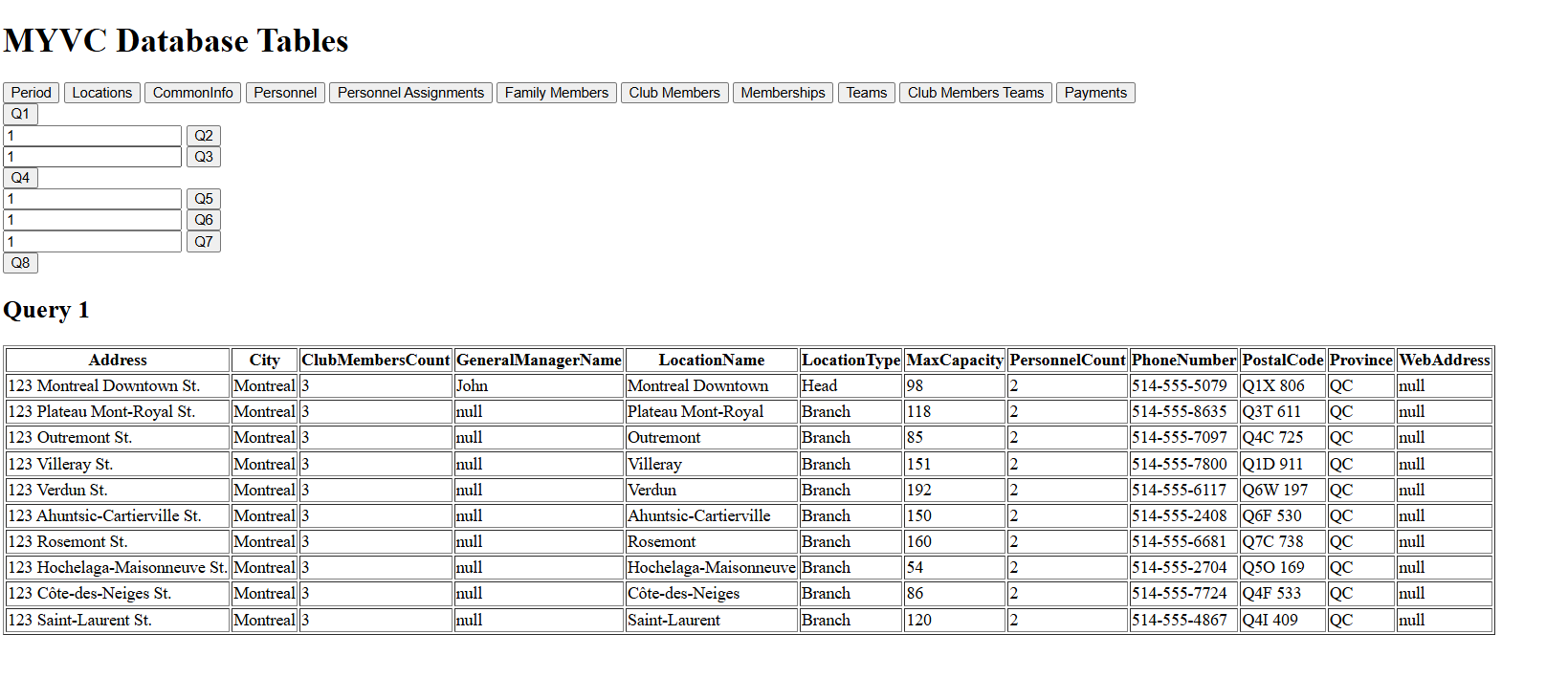


Image 13 Q1 screenshot

### For a given location, provide a report about family members with active club members

For a given location, provide a report that lists for every family member who is currently registered in the location, the number of related active club members. Information includes family members’ first name, last name, and the number of active club members that are associated with the family member

SELECT

ci.FirstName AS FamilyMemberFirstName,

ci.LastName AS FamilyMemberLastName,

COUNT(cm.MemberID) AS ActiveClubMembersCount

FROM FamilyMembers fm

JOIN CommonInfo ci ON fm.CommonID = ci.CommonID

JOIN ClubMembers cm ON fm.FamilyMemberID = cm.FamilyMemberID

JOIN Memberships m ON cm.MemberID = m.MemberID

JOIN Locations l ON fm.LocationID = l.LocationID

WHERE l.LocationID = 1

AND m.Status = 'Active'

GROUP BY fm.FamilyMemberID;

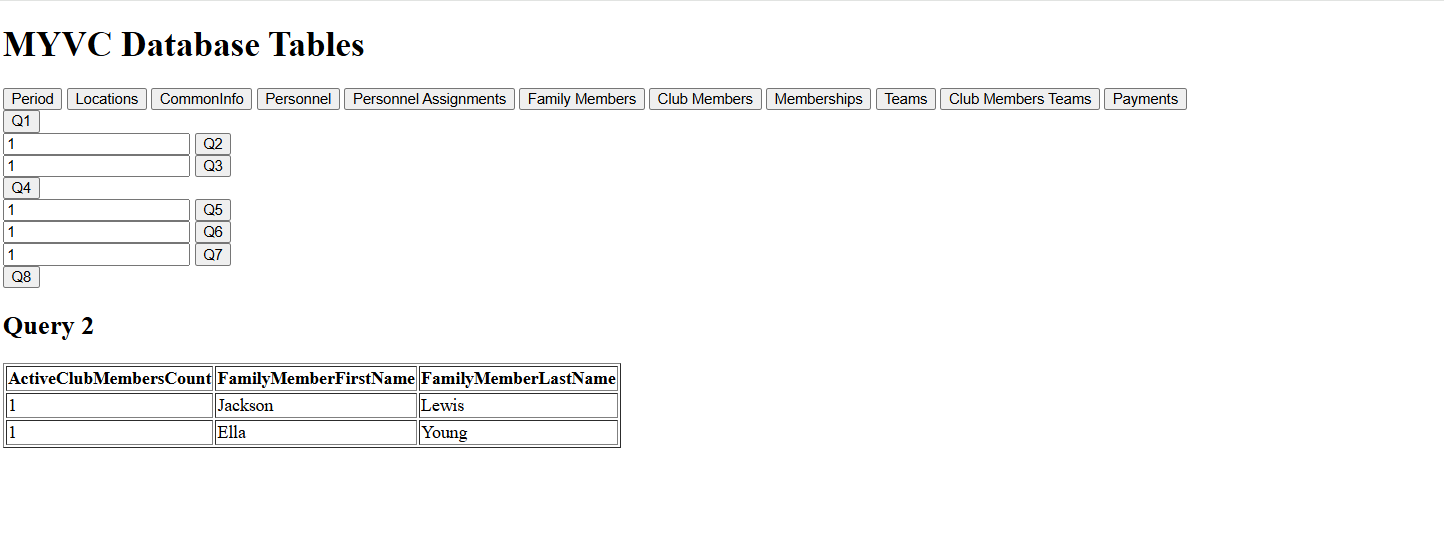


Image 14 Q2 screenshot

### **For a given location, report on personnel**

iii. For a given location, provide a report that displays information about the personnel who are currently operating in that location. The information includes first-name, last-name, date of birth, Social Security Number, Medicare card number, telephone number, address, city, province, postal code, email address, role (General manager, deputy manager, Coach, etc.) and mandate (Volunteer or Salaried).

SELECT

ci.FirstName,

ci.LastName,

ci.DateOfBirth,

ci.SSN,

ci.MedicareCardNumber,

ci.PhoneNumber,

ci.Address,

ci.City,

ci.Province,

ci.PostalCode,

ci.Email,

p.Role,

p.Mandate

FROM Personnel p

JOIN CommonInfo ci ON p.CommonID = ci.CommonID

JOIN Personnel\_Assignments pa ON p.PersonnelID = pa.PersonnelID

JOIN Locations l ON pa.LocationID = l.LocationID

WHERE l.LocationID = 1;

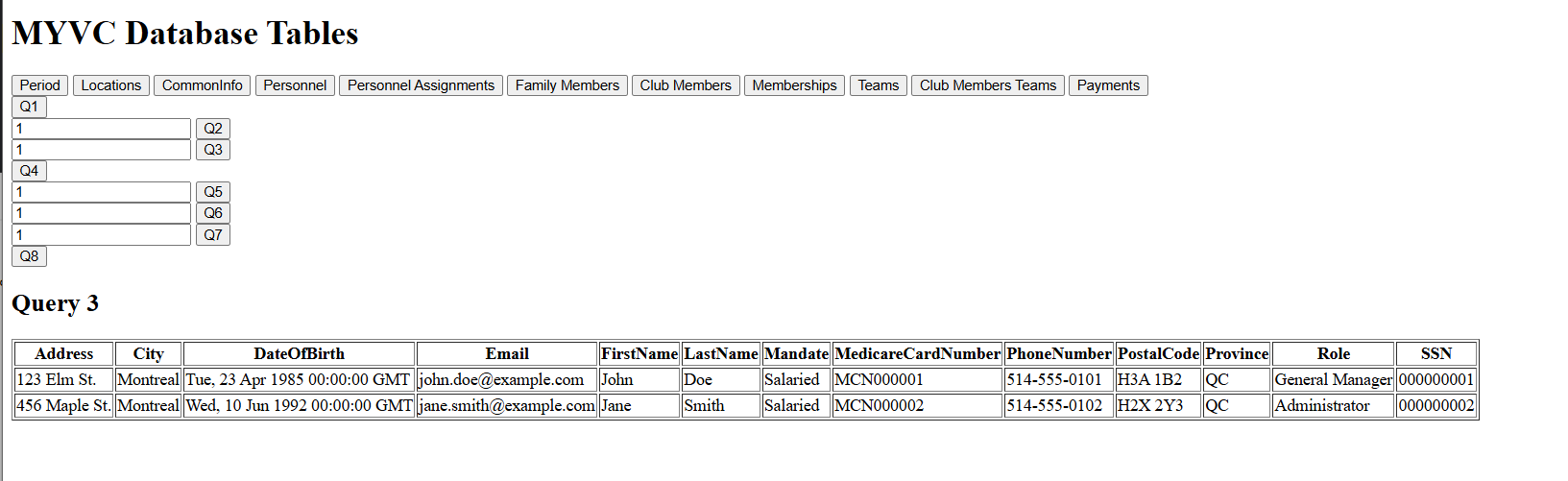


Image 15 Q3 screenshot

### Get a detailed list of all club members registered in the system

Get a detailed list of all club members registered in the system. The list should include the location name that the club member is currently associated with, the membership number of the club member, first-name, last-name, age, city, province, and status (active or inactive). The results should be displayed sorted in ascending order by location name, then by age.

SELECT

l.Name AS LocationName,

cm.MemberID AS MembershipNumber,

ci.FirstName,

ci.LastName,

TIMESTAMPDIFF(YEAR, ci.DateOfBirth, CURDATE()) AS Age,

ci.City,

ci.Province,

m.Status

FROM ClubMembers cm

JOIN CommonInfo ci ON cm.CommonID = ci.CommonID

JOIN Memberships m ON cm.MemberID = m.MemberID

JOIN Locations l ON cm.LocationID = l.LocationID

WHERE m.Status IN ('Active', 'Inactive')

ORDER BY l.Name ASC, Age ASC;

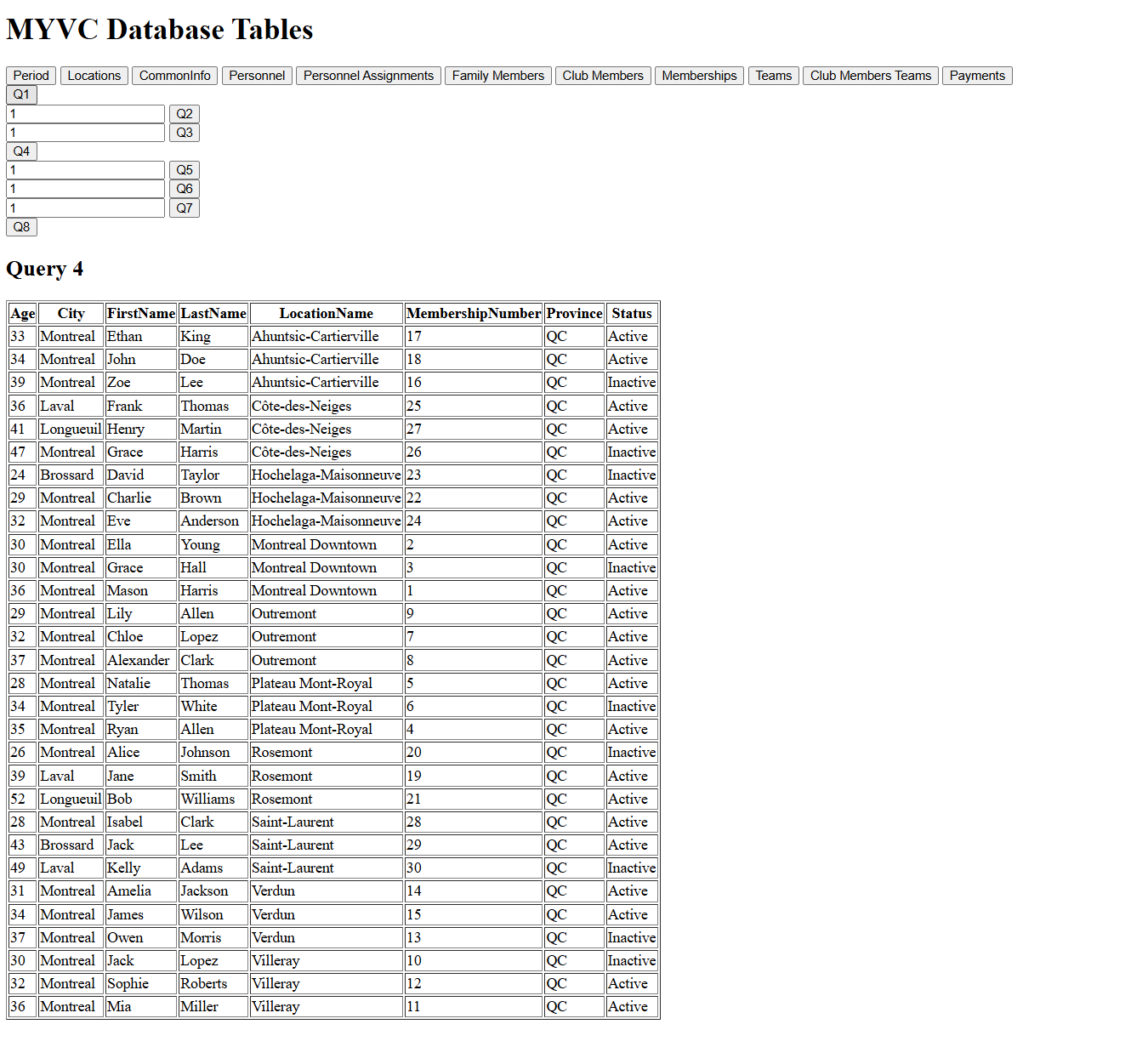


Image 16 Q4 screenshot

### For a given family member, get details of all club members associated with him/her

For a given family member, get details of all club members associated with him/her. Information includes club membership number, first-name, last-name, date of birth, Social Security Number, Medicare card number, telephone number, address, city, province, postal code, relationship with the family member, and status (active or inactive).

SELECT

m.MembershipID AS ClubMembershipNumber,

ci.FirstName,

ci.LastName,

ci.DateOfBirth,

ci.SSN,

ci.MedicareCardNumber,

ci.PhoneNumber,

ci.Address,

ci.City,

ci.Province,

ci.PostalCode,

fm.Relationship,

m.Status

FROM ClubMembers cm

JOIN CommonInfo ci ON cm.CommonID = ci.CommonID

JOIN Memberships m ON cm.MemberID = m.MemberID

JOIN FamilyMembers fm ON cm.FamilyMemberID = fm.FamilyMemberID

WHERE fm.FamilyMemberID = 1

ORDER BY m.MembershipID;

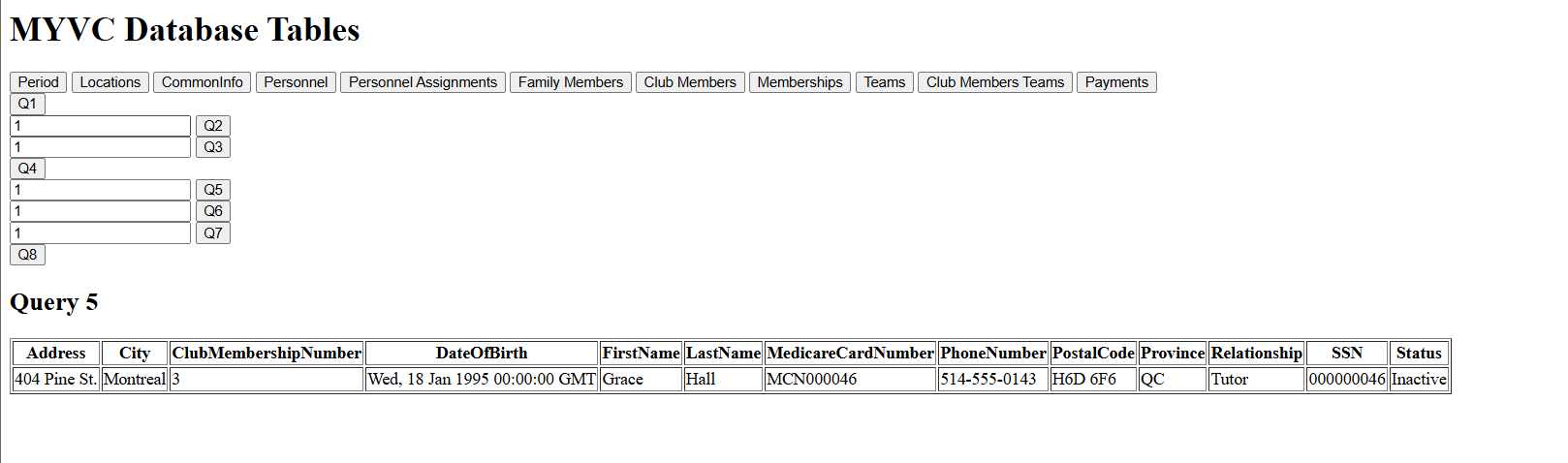


Image 17 Q5 screenshot

### For a given location, get family members with active club members associated and operator personnel

For a given location, get the list of all family members who have currently active club members associated with them and are also an operator personnel member for the same location. Information includes first-name, last-name, and phone number of the family member.

SELECT

ci.FirstName AS FamilyMemberFirstName,

ci.LastName AS FamilyMemberLastName,

ci.PhoneNumber AS FamilyMemberPhoneNumber

FROM FamilyMembers fm

JOIN CommonInfo ci ON fm.CommonID = ci.CommonID

JOIN ClubMembers cm ON fm.FamilyMemberID = cm.FamilyMemberID

JOIN Memberships m ON cm.MemberID = m.MemberID

JOIN Personnel\_Assignments pa ON pa.LocationID = fm.LocationID

JOIN Personnel p ON p.PersonnelID = pa.PersonnelID

JOIN Locations l ON fm.LocationID = l.LocationID

WHERE m.Status = 'Active'

AND p.LocationID = fm.LocationID

AND p.Role IN ('General Manager', 'Deputy Manager', 'Treasurer', 'Secretary', 'Coach', 'Assistant Coach', 'Captain') -- Operator roles

AND fm.LocationID = 1

ORDER BY ci.FirstName, ci.LastName;

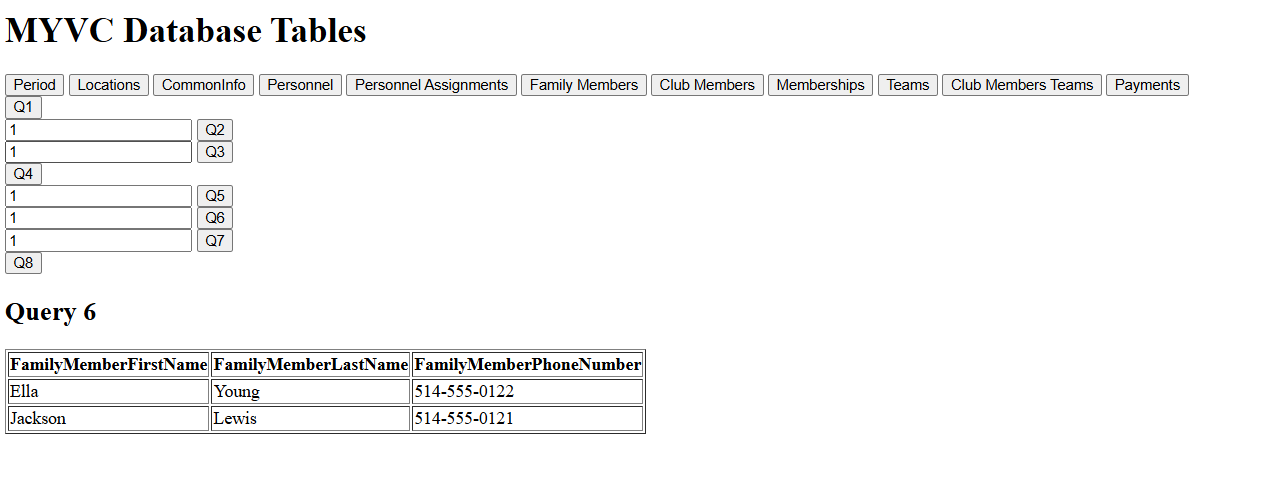


Image 18 Q6 screenshot

### For a given club member, give details of all payments for membership fees

For a given club member, give details of all payments for the membership fees. Information includes date of payment, amount of payment, and year of payment. The results should be displayed sorted in ascending order by date.

SELECT

p.PaymentDate,

p.Amount,

YEAR(p.PaymentDate) AS YearPaid

FROM Payments p

JOIN Memberships m ON p.MembershipID = m.MembershipID

WHERE m.MemberID = 1

ORDER BY p.PaymentDate;

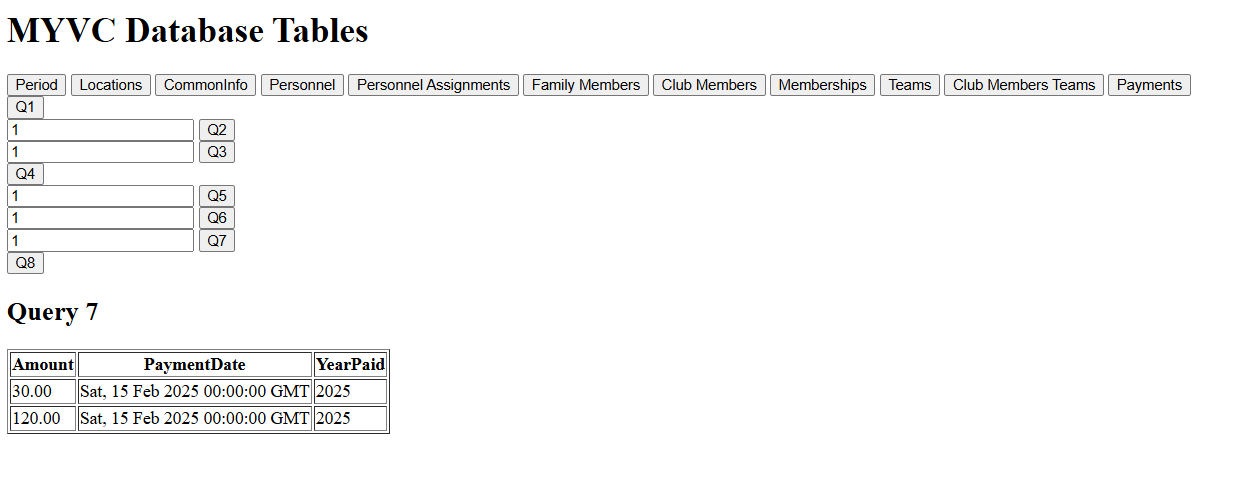


Image 19 Q7 screenshot

### Get the sum of membership fees paid and donations

Get the sum of membership fees paid and the sum of donations that are collected by the club in the year 2024.

SELECT

SUM(p.Amount) AS TotalMembershipFeesPaid,

SUM(m.DonationAmount) AS TotalDonationsCollected

FROM Payments p

JOIN Memberships m ON p.MembershipID = m.MembershipID

WHERE YEAR(p.PaymentDate) = 2025;

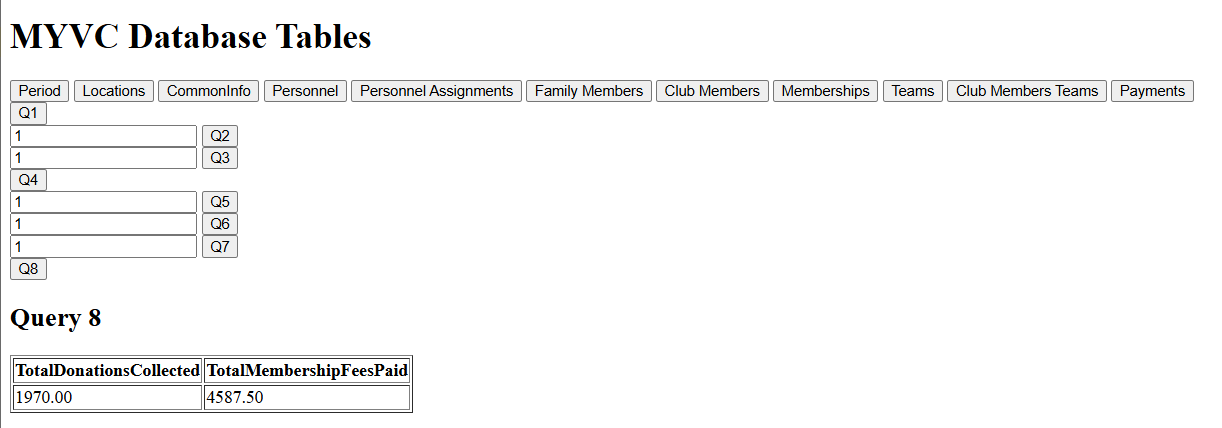


Image 20 Q8 screenshot

## Count \* Statements

SELECT

'Period' AS TableName, COUNT(\*) AS RowCount FROM Period

UNION ALL

SELECT

'Locations', COUNT(\*) FROM Locations

UNION ALL

SELECT

'CommonInfo', COUNT(\*) FROM CommonInfo

UNION ALL

SELECT

'Personnel', COUNT(\*) FROM Personnel

UNION ALL

SELECT

'Personnel\_Assignments', COUNT(\*) FROM Personnel\_Assignments

UNION ALL

SELECT

'FamilyMembers', COUNT(\*) FROM FamilyMembers

UNION ALL

SELECT

'ClubMembers', COUNT(\*) FROM ClubMembers

UNION ALL

SELECT

'Memberships', COUNT(\*) FROM Memberships

UNION ALL

SELECT

'Teams', COUNT(\*) FROM Teams

UNION ALL

SELECT

'ClubMembers\_Teams', COUNT(\*) FROM ClubMembers\_Teams

UNION ALL

SELECT

'Payments', COUNT(\*) FROM Payments;

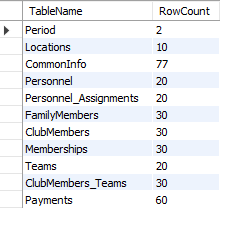


Image 21 Count screenshot

# Testing data sets

|  |  |
| --- | --- |
| Period | 2 |
| Locations | 10 |
| CommonInfo | 77 |
| Personnel | 20 |
| Personnel\_Assignments | 20 |
| FamilyMembers | 30 |
| ClubMembers | 30 |
| Memberships | 30 |
| Teams | 20 |
| ClubMembers\_Teams | 30 |
| Payments | 60 |

In our test case, we insert the above number of records to each table.

The assignments are:

10 locations with 1 head and 9 branches.

1 period

77 Commoninfo:

20 are personnels

30 are Familymemebers

30 are Clubmembers

20 Personnel:

2 at each location

Only 1 General manager working at head

30 Familymembers:

10 of them are personnel, one at each location

20 of them are father/mother, 2 at each location

30 Club members

Since each location has 3 Familymember

Each family member has 1 registered club member at the same location.

20 Team:

Each location has 2 teams(B/G)

30 ClubMembers\_Teams

Many to many relationships, relationships between Clubmember and different teams.

60 Payments:

Each club member has 2 payments.

# UI

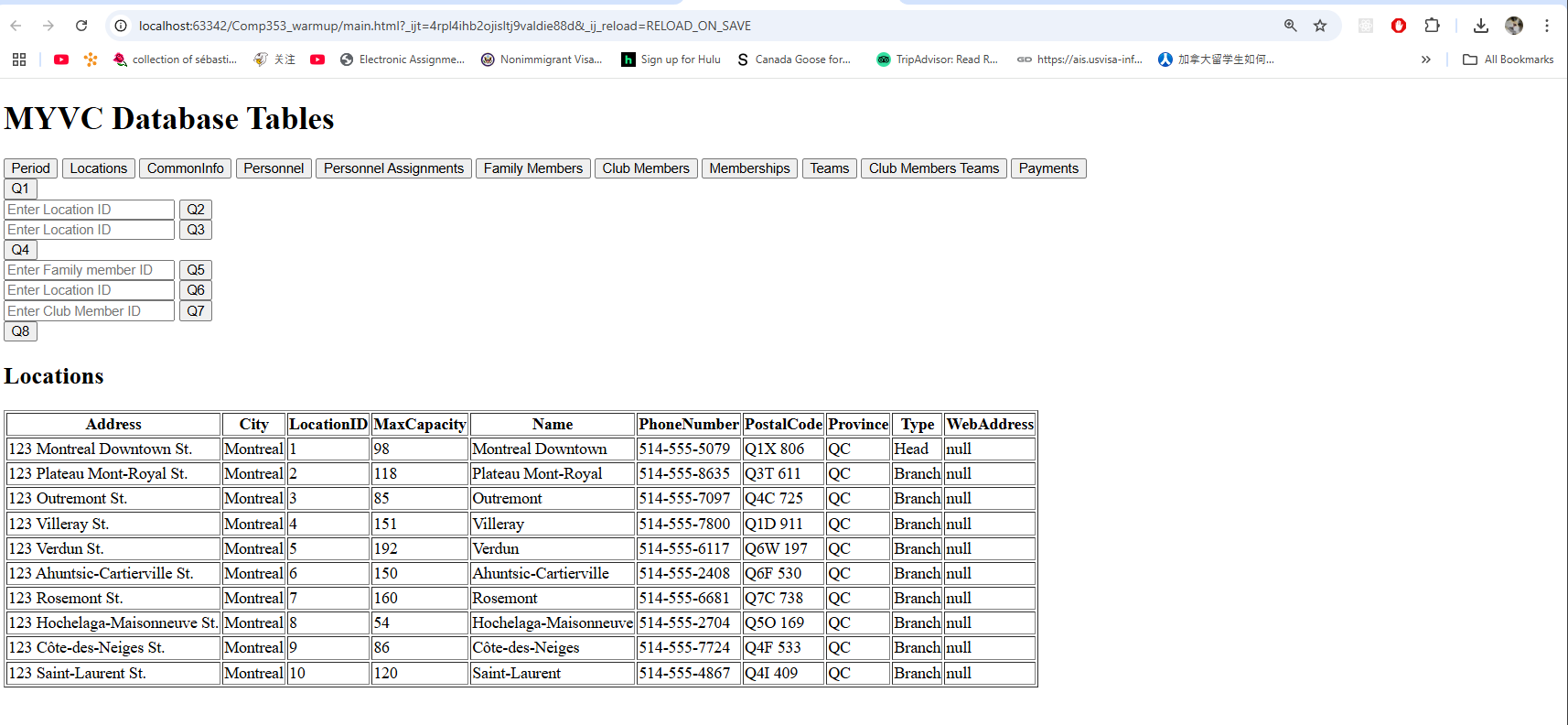


Image 22 UI web html page with python backend

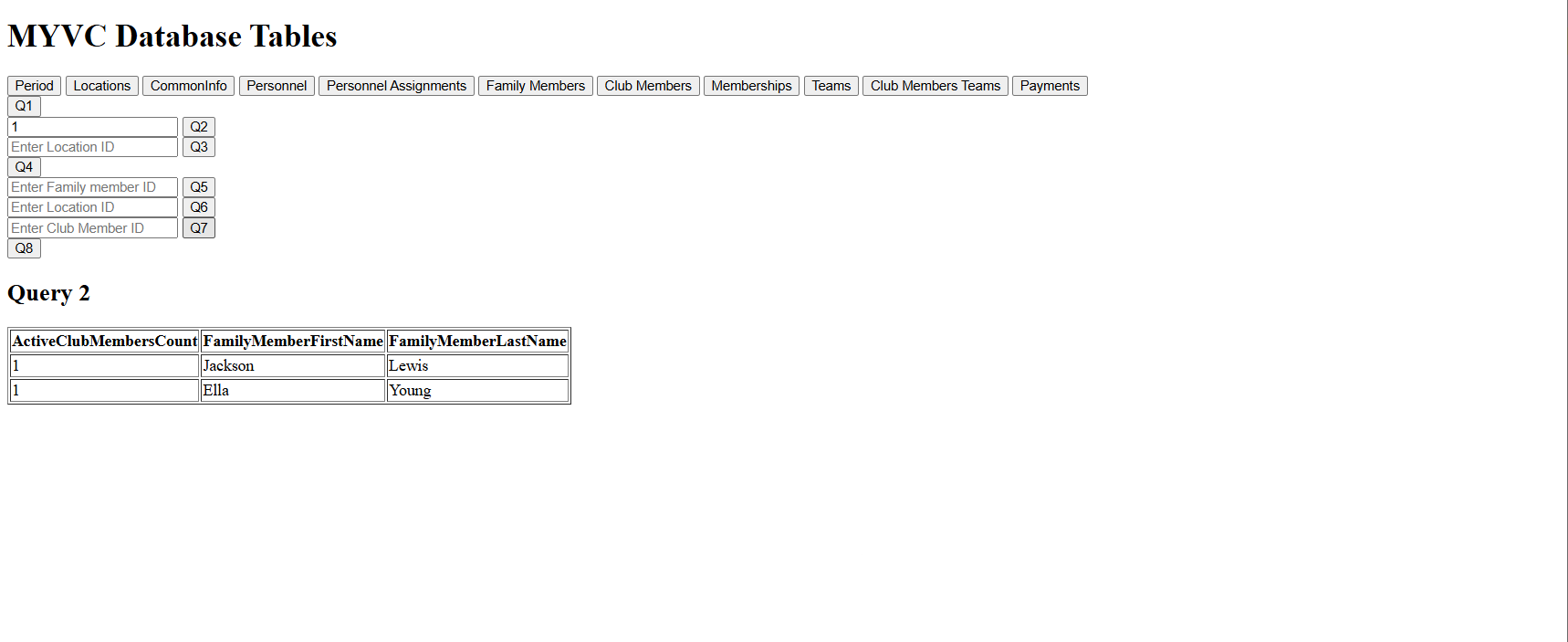


Image 23 Q2 query UI

The user interface (UI) of the MYVC Database Tables application is designed for ease of interaction with the database. It consists of a simple layout with buttons for fetching individual database tables and executing specific SQL queries. Each table has a corresponding button that, when clicked, triggers a fetch request to retrieve and display the data from the database. For SQL queries, input fields are provided to allow users to enter parameters such as Location ID, Family Member ID, and Club Member ID, depending on the query being executed. The data is then displayed in a table format, with each row representing a record from the database.

# Conclusion

The Montreal Youth Volleyball Club (MYVC) database system was successfully designed and implemented to meet the club’s operational and business requirements. Using an E/R model, the system manages membership tracking, financial transactions, family relationships, and personnel assignments across multiple locations while ensuring compliance with the business conditions

Key design constraints were enforced, including unique identifiers for personnel and members, age restrictions for club members, and financial regulations for membership fees and donations. The system maintains relational consistency through foreign key constraints and structured relationships, preventing data anomalies and inconsistencies. It ensures unique personnel identification using SSN and Medicare Card constraints, enforces age limitations (11-18 years old) for active club members, and implements membership fee tracking that requires full payment before participation in club activities while handling excess payments as donations.

To address personnel management, the system ensures that each location has only one manager at a time and that personnel can only hold one role at a time. Family relationships are structured to allow a club member to be registered under a single guardian, with the flexibility to update in case of guardian changes. The payment system allows installment-based transactions while ensuring members only become “active” once the full $100 membership fee is paid. Excess payments beyond this amount are automatically classified as donations.

Through our schema design, database constraints, and business rule enforcement, the MYVC database system provides a scalable, maintainable, and efficient solution that meets the club’s operational needs while ensuring proper data organization, financial accountability, and membership management.

# Appendix