UNIVERSITÉ Concordia UNIVERSITY

GINA CODY SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

Main Project

COMP353 Databases
Professor Ph.D. Khaled Jababo
Winter 2025

Team Members

Sopheaktra Lean	ID: 40225014
Rory Poonyth	ID: 40226938
Omar Sanchez Lopez	ID: 40027467
Toby Fischer	ID: 40281628

Group account: iqc353_4

"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"

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 citation[1]. 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 353 Instructor: Khaled Jababo
Name: Sopheaktra Lean, Rory Poonyth, Omar Sanchez Lopez, Toby Fischer I.D. # 40225014, 40226938, 40027467, 40281628
Signature: Pheaktra, Rory, Omar, Toby Date: March 29th 2025

[1] Rules for reference citation can be found in "Form and Style" by Patrich MacDonagh and Jack Bordan, fourth edition, May, 2000, available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf.
Approved by the ENCS Faculty Council February 10, 2012

Table of Content

1. Introduction

The Montreal Volleyball Club Database Application is implemented to manage and store information related to volleyball club operations in Montreal. This relation database system can maintain the records of the club members, locations, personnel, team formations, payments, and family associations.
The database has the functionalities such as:

- Track membership and associations: storing the information of each club members both personal and associations with multiple club locations
- Handle locations and staff: storing the information of each club locations, both the head and branch, the addresses, capacities, coaches, and managers.
- Record training and game sessions: recording the details of each scheduled training and game sessions with the teams, session, score, and player roles
- GUI: facilitating interaction of end-users with the database system

2. E/R Diagram

2.1. Entities and Attributes

2.1.1. ClubMember

This entity stores the information of registered volleyball club members:

Table: ClubMember

Columns:

club_member_number	int AI PK
first_name	varchar(50)
last_name	varchar(50)
birth_date	date
gender	enum('Male','Female')
height	decimal(5,2)
weight	decimal(5,2)
sin_number	varchar(15)
medicare_number	varchar(20)
telephone_number	varchar(20)
address	varchar(200)
city	varchar(100)
province	varchar(100)
postal_code	varchar(20)
active_status	tinyint(1)
deactivation_date	date
email_address	varchar(100)

2.1.2. FamilyMember

This entity stores the information of the family member associated to the club member:

Table: FamilyMember

Columns:

family_member_id	int AI PK
first_name	varchar(50)
last_name	varchar(50)
birth_date	date
sin_number	varchar(15)
medicare_number	varchar(20)
telephone_number	varchar(20)
address	varchar(200)
city	varchar(100)
province	varchar(100)
postal_code	varchar(20)
email_address	varchar(100)

2.1.3. ClubMember_FamilyMember

This entity stores the information of the club member associated to their family member:

Table: ClubMember_FamilyMember

Columns:

club_member_number	int PK
family_member_id	int PK
relationship	enum('Father','Mother','Grandfather','Grandmother','Uncle','Aunt','Tutor','Partner','Friend','Other')

2.1.4. SecondaryFamilyMember

This entity stores the details of secondary family members who have additional relationships with club members:

Table: SecondaryFamilyMember

Columns:

secondary_id	int AI PK
primary_family_member_id	int
first_name	varchar(50)
last_name	varchar(50)
telephone_number	varchar(20)

2.1.5. SecondaryFamilyMember_ClubMember

This entity links secondary family members to club members:

Table: SecondaryFamilyMember_ClubMember

Columns:

secondary_id	int PK
club_member_number	int PK
relationship	enum('Father','Mother','Grandfather','Grandmother','Uncle','Aunt','Tutor','Partner','Friend','Other')

2.1.6. Location

This entity stores the information of each location:

Table: Location

Columns:

location_id	int AI PK
location_type	enum('Head','Branch')
name	varchar(100)
address	varchar(200)
city	varchar(100)
province	varchar(100)
postal_code	varchar(20)
phone_number	varchar(20)
web_address	varchar(200)
max_capacity	int

2.1.7. ClubMember_Location

This entity stores the information of each club member assigned to the location:

Table: ClubMember_Location

Columns:

club_member_number	int PK
location_id	int
start_date	date PK
end_date	date

2.1.8. FamilyMember_Location

This entity stores the family member linked to the location:

Table: FamilyMember_Location

Columns:

family_member_id	int PK
location_id	int PK
start_date	date PK
end_date	date

2.1.9. Personnel

This entity stores the detailed information of each personnel:

Table: Personnel

Columns:

personnel_id	int AI PK
first_name	varchar(50)
last_name	varchar(50)
birth_date	date
sin_number	varchar(15)
medicare_number	varchar(20)
telephone_number	varchar(20)
address	varchar(200)
city	varchar(100)
province	varchar(100)
postal_code	varchar(20)
email_address	varchar(100)
mandate	enum('volunteer','salaried')

2.1.10. Personnel_Location_Role

This entity stores the information about the role and location history of every personnel

Table: Personnel_Location_Role

Columns:

personnel_id	int PK
location_id	int PK
role	enum('manager','general_manager','deputy_manager','treasurer','secretary','administrator','captain','coach','assistant_coach','other') PK
start_date	date PK
end_date	date

2.1.11. Session

This entity stores the information about training and game session:

Table: Session

Columns:

session_id	int AI PK
session_date	date
session_time	time
session_type	enum('game','training')
location_id	int
team1_id	int
team2_id	int
score_team1	int
score_team2	int
coach_pid	int

2.1.12. Team

This entity tracks teams participating in training and game sessions:

Table: Team

Columns:

team_id	int AI PK
team_name	varchar(100)
location_id	int
gender	enum('Boys','Girls')
captain_pid	int

2.1.13. Session_Player

This entity tracks players participating in sessions and their roles

Table: Session_Player

Columns:

session_id	int PK
club_member_number	int PK
team_id	int
player_position	enum('outside_hitter','opposite','setter','middle_blocker','libero','defensive_specialist','serving_specialist')

2.1.14. Payment

This entity contains the information about payment of the club member:

Table: Payment

Columns:

payment_id	int AI PK
club_member_number	int
payment_date	date
payment_amount	decimal(7,2)
method_of_payment	enum('Cash','Debit','Credit Card')
membership_year	year

2.1.15. Email_Log

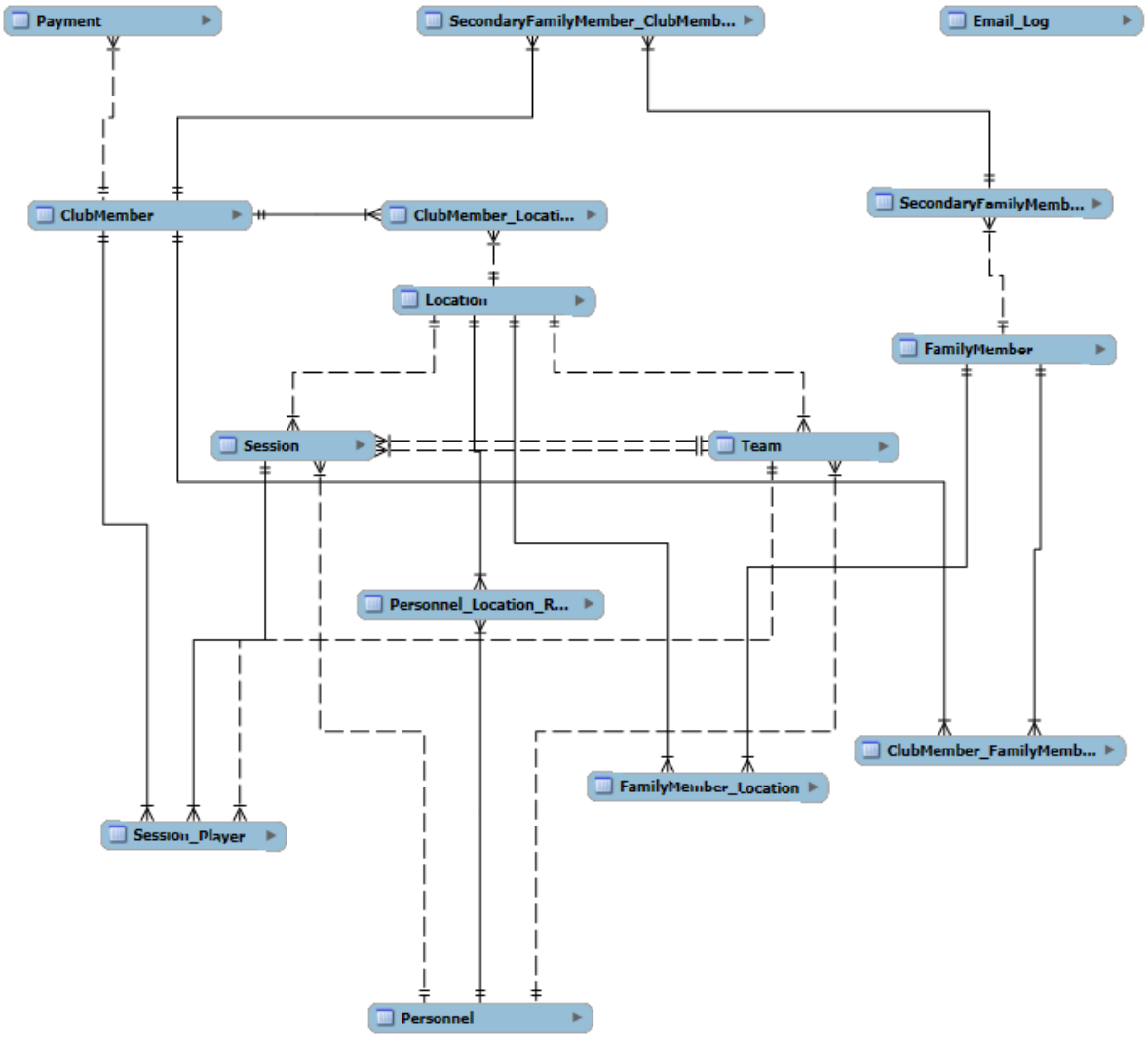
This entity tracks email sent to members:

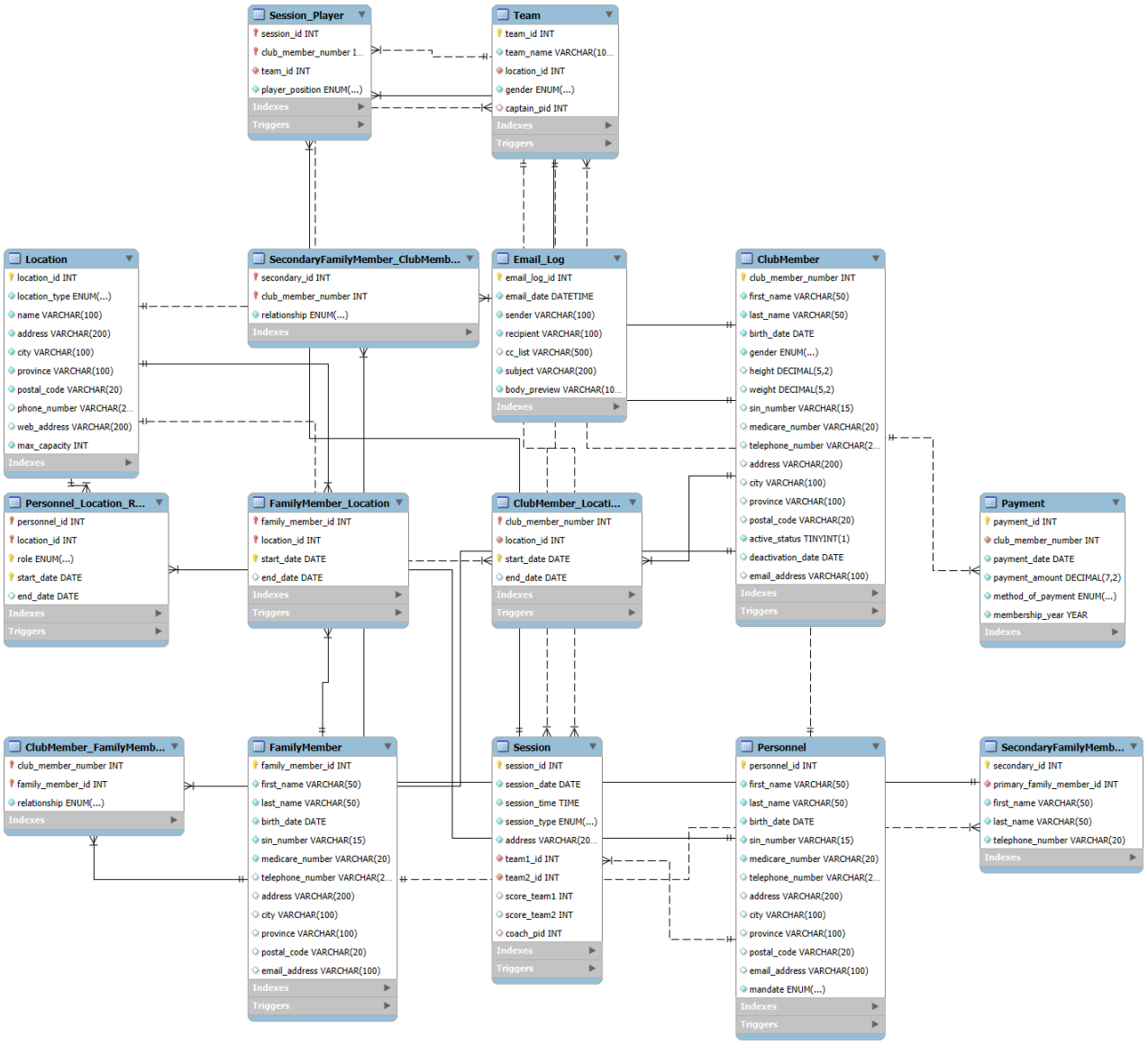
Table: Email_Log

Columns:

email_log_id	int AI PK
email_date	datetime
sender	varchar(100)
recipient	varchar(100)
cc_list	varchar(500)
subject	varchar(200)
body_preview	varchar(100)

2.2. E/R Diagram Representation





3. Normalization Analysis

3.1. Third Normal Form (3NF)

A table is in **3NF** if:

1. It is in **2NF** (no partial dependency on part of a PK, which is relevant primarily if you have composite keys).

2. **Every non-key attribute** depends on **the key, the whole key, and nothing but the key** (no transitive dependencies on non-candidate keys).

- **Bridging tables** (e.g. ClubMember_Location, Session, Personnel_Location_Role, etc.) have **composite primary keys** that **exactly** match their FKs and only store the association plus small attributes like start_date, end_date, player_role. There are no

other non-key attributes. Hence they trivially satisfy 3NF.

- **Entity tables** (e.g. ClubMember, FamilyMember, Personnel, Location) each have a **single surrogate primary key** (_id) and zero or more **unique** business keys (sin_number, etc.). All non-key attributes are about that entity, so there is no transitive dependency. That also meets 3NF.

- Some columns (e.g. sin_number is unique for Personnel) can also act as an **alternate** key. That alone does not break 3NF, since each functional dependency from sin_number to the other attributes is from a candidate key.

Hence, by inspection, **all these tables** meet the condition for **Third Normal Form**.

3.2. Boyce-Codd Normal Form (BCNF)

A table is in **BCNF** (Boyce-Codd Normal Form) if for every functional dependency X → Y, X is a candidate key. Put differently, no non-trivial FD can have a non-key determinant.

- **Bridging tables** (like Session, ClubMember_Location, Personnel_Location_Role, etc.) typically **are** in BCNF. Their entire set of attributes is basically (FK1, FK2, ..., possibly start_date, role, etc.), and the **PK** is the only determinant for all columns. No partial dependencies exist, so these are BCNF.

- **Entity tables** (e.g. Personnel): Each has a surrogate PK (e.g. personnel_id) plus a unique alternate key (sin_number). Because sin_number is also **unique**, that means sin_number is a candidate key as well. So any FD from sin_number to other columns is from a key. Hence those FD's do not break BCNF.

- **Location**: The PK is location_id. If there's no business rule that name uniquely identifies a location, there are no other FDs. So it's BCNF.

- **Email_Log**: The PK is email_log_id. Typically, no other FDs exist (the subject, sender, date, etc. all depend on the unique log entry). So it's BCNF.

- **Payment**: The PK is payment_id. We do not have a composite key for (club_member_number, membership_year, payment_date) or anything forced. As long as there's no non-key attribute that determines another non-key attribute, it's also in BCNF.

- **Team**: The PK is team_id. Potentially, (team_name) might also be unique if every session's teams have unique names. If so, that's just an **alternate key**. As long as it's declared unique, that's fine. No violation of BCNF arises.

Hence, by inspection, **all these tables** meet the condition for **BCNF**.

4. SQL Database Implementation

4.1. SQL DDL

4.1.1. ClubMember

```
CREATE TABLE `ClubMember` (
  `club_member_number` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `last_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `birth_date` date NOT NULL,
  `gender` enum('Male','Female') COLLATE utf8mb3_unicode_ci NOT NULL,
  `height` decimal(5,2) DEFAULT NULL,
  `weight` decimal(5,2) DEFAULT NULL,
  `sin_number` varchar(15) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `medicare_number` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT
NULL,
  `telephone_number` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT
NULL,
  `address` varchar(200) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `city` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `province` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `postal_code` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `active_status` tinyint(1) NOT NULL DEFAULT '1',
  `deactivation_date` date DEFAULT NULL,
  `email_address` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`club_member_number`),
  UNIQUE KEY `sin_number` (`sin_number`),
  UNIQUE KEY `medicare_number` (`medicare_number`)
) ENGINE=InnoDB AUTO_INCREMENT=2012 DEFAULT
CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci
```

4.1.2. FamilyMember

```
CREATE TABLE `FamilyMember` (
  `family_member_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `last_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `birth_date` date NOT NULL,
  `sin_number` varchar(15) COLLATE utf8mb3_unicode_ci NOT NULL,
  `medicare_number` varchar(20) COLLATE utf8mb3_unicode_ci NOT NULL,
```

```
  `telephone_number` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT
NULL,
  `address` varchar(200) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `city` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `province` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `postal_code` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `email_address` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`family_member_id`),
  UNIQUE KEY `sin_number` (`sin_number`),
  UNIQUE KEY `medicare_number` (`medicare_number`)
) ENGINE=InnoDB AUTO_INCREMENT=104 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.3. ClubMember_FamilyMember

```
CREATE TABLE `ClubMember_FamilyMember` (
  `club_member_number` int NOT NULL,
  `family_member_id` int NOT NULL,
  `relationship`
enum('Father','Mother','Grandfather','Grandmother','Uncle','Aunt','Tutor','Partner','F
riend','Other') COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`club_member_number`,`family_member_id`),
  KEY `family_member_id` (`family_member_id`),
  CONSTRAINT `ClubMember_FamilyMember_ibfk_1` FOREIGN KEY
(`club_member_number`) REFERENCES `ClubMember`
(`club_member_number`) ON DELETE CASCADE,
  CONSTRAINT `ClubMember_FamilyMember_ibfk_2` FOREIGN KEY
(`family_member_id`) REFERENCES `FamilyMember` (`family_member_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.4. SecondaryFamilyMember

```
  CREATE TABLE `SecondaryFamilyMember` (
  `secondary_id` int NOT NULL AUTO_INCREMENT,
  `primary_family_member_id` int NOT NULL,
  `first_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `last_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `telephone_number` varchar(20) COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`secondary_id`),
  KEY `SecondaryFamilyMember_ibfk_1` (`primary_family_member_id`),
```

```
  CONSTRAINT `SecondaryFamilyMember_ibfk_1` FOREIGN KEY
(`primary_family_member_id`) REFERENCES `FamilyMember`
(`family_member_id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.5. SecondaryFamilyMember_ClubMember

```
CREATE TABLE `SecondaryFamilyMember_ClubMember` (
  `secondary_id` int NOT NULL,
  `club_member_number` int NOT NULL,
  `relationship`
enum('Father','Mother','Grandfather','Grandmother','Uncle','Aunt','Tutor','Partner','F
riend','Other') COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`secondary_id`,`club_member_number`),
  KEY `SecondaryFamilyMember_ClubMember_ibfk_2`
(`club_member_number`),
  CONSTRAINT `SecondaryFamilyMember_ClubMember_ibfk_1` FOREIGN
KEY (`secondary_id`) REFERENCES `SecondaryFamilyMember`
(`secondary_id`) ON DELETE CASCADE,
  CONSTRAINT `SecondaryFamilyMember_ClubMember_ibfk_2` FOREIGN
KEY (`club_member_number`) REFERENCES `ClubMember`
(`club_member_number`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.6. Location

```
CREATE TABLE `Location` (
  `location_id` int NOT NULL AUTO_INCREMENT,
  `location_type` enum('Head','Branch') COLLATE utf8mb3_unicode_ci NOT
NULL,
  `name` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `address` varchar(200) COLLATE utf8mb3_unicode_ci NOT NULL,
  `city` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `province` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `postal_code` varchar(20) COLLATE utf8mb3_unicode_ci NOT NULL,
  `phone_number` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `web_address` varchar(200) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `max_capacity` int NOT NULL,
  PRIMARY KEY (`location_id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.7. ClubMember_Location

```
CREATE TABLE `ClubMember_Location` (
  `club_member_number` int NOT NULL,
  `location_id` int NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`club_member_number`,`start_date`),
  KEY `ClubMember_Location_ibfk_2` (`location_id`),
  CONSTRAINT `ClubMember_Location_ibfk_1` FOREIGN KEY
(`club_member_number`) REFERENCES `ClubMember`
(`club_member_number`) ON DELETE CASCADE,
  CONSTRAINT `ClubMember_Location_ibfk_2` FOREIGN KEY (`location_id`)
REFERENCES `Location` (`location_id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.8. FamilyMember_Location

```
CREATE TABLE `FamilyMember_Location` (
  `family_member_id` int NOT NULL,
  `location_id` int NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`family_member_id`,`location_id`,`start_date`),
  KEY `FamilyMember_Location_ibfk_2` (`location_id`),
  CONSTRAINT `FamilyMember_Location_ibfk_1` FOREIGN KEY
(`family_member_id`) REFERENCES `FamilyMember` (`family_member_id`)
ON DELETE CASCADE,
  CONSTRAINT `FamilyMember_Location_ibfk_2` FOREIGN KEY
(`location_id`) REFERENCES `Location` (`location_id`) ON DELETE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.9. Personnel

```
CREATE TABLE `Personnel` (
  `personnel_id` int NOT NULL AUTO_INCREMENT,
```

```
  `first_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `last_name` varchar(50) COLLATE utf8mb3_unicode_ci NOT NULL,
  `birth_date` date NOT NULL,
  `sin_number` varchar(15) COLLATE utf8mb3_unicode_ci NOT NULL,
  `medicare_number` varchar(20) COLLATE utf8mb3_unicode_ci NOT NULL,
  `telephone_number` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT
NULL,
  `address` varchar(200) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `city` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `province` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `postal_code` varchar(20) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `email_address` varchar(100) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `mandate` enum('volunteer','salaried') COLLATE utf8mb3_unicode_ci NOT
NULL,
  PRIMARY KEY (`personnel_id`),
  UNIQUE KEY `sin_number` (`sin_number`),
  UNIQUE KEY `medicare_number` (`medicare_number`)
) ENGINE=InnoDB AUTO_INCREMENT=917 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.10. Personnel_Location_Role

```
CREATE TABLE `Personnel_Location_Role` (
  `personnel_id` int NOT NULL,
  `location_id` int NOT NULL,
  `role`
enum('manager','general_manager','deputy_manager','treasurer','secretary','administ
rator','captain','coach','assistant_coach','other') COLLATE utf8mb3_unicode_ci
NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`personnel_id`,`location_id`,`role`,`start_date`),
  KEY `Personnel_Location_Role_ibfk_2` (`location_id`),
  CONSTRAINT `Personnel_Location_Role_ibfk_1` FOREIGN KEY
(`personnel_id`) REFERENCES `Personnel` (`personnel_id`) ON DELETE
CASCADE,
  CONSTRAINT `Personnel_Location_Role_ibfk_2` FOREIGN KEY
(`location_id`) REFERENCES `Location` (`location_id`) ON DELETE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.11. Session

```
CREATE TABLE `Session` (
  `session_id` int NOT NULL AUTO_INCREMENT,
  `session_date` date NOT NULL,
  `session_time` time NOT NULL,
  `session_type` enum('game','training') COLLATE utf8mb3_unicode_ci NOT
NULL,
  `location_id` int NOT NULL,
  `team1_id` int NOT NULL,
  `team2_id` int NOT NULL,
  `score_team1` int DEFAULT '0',
  `score_team2` int DEFAULT '0',
  `coach_pid` int DEFAULT NULL,
  PRIMARY KEY (`session_id`),
  KEY `team1_id` (`team1_id`),
  KEY `team2_id` (`team2_id`),
  KEY `coach_pid` (`coach_pid`),
  KEY `idx_location_id` (`location_id`),
  CONSTRAINT `fk_session_coach` FOREIGN KEY (`coach_pid`)
REFERENCES `Personnel` (`personnel_id`),
  CONSTRAINT `fk_session_location` FOREIGN KEY (`location_id`)
REFERENCES `Location` (`location_id`),
  CONSTRAINT `fk_session_team1` FOREIGN KEY (`team1_id`)
REFERENCES `Team` (`team_id`),
  CONSTRAINT `fk_session_team2` FOREIGN KEY (`team2_id`)
REFERENCES `Team` (`team_id`)
) ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.12. Session_Player

```
CREATE TABLE `Session_Player` (
  `session_id` int NOT NULL,
  `club_member_number` int NOT NULL,
  `team_id` int NOT NULL,
  `player_position`
enum('outside_hitter','opposite','setter','middle_blocker','libero','defensive_specialis
t','serving_specialist') COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`session_id`,`club_member_number`),
  KEY `team_id` (`team_id`),
  KEY `fk_sp_member` (`club_member_number`),
```

```
  CONSTRAINT `fk_sp_member` FOREIGN KEY (`club_member_number`)
REFERENCES `ClubMember` (`club_member_number`),
  CONSTRAINT `fk_sp_session` FOREIGN KEY (`session_id`) REFERENCES
`Session` (`session_id`),
  CONSTRAINT `fk_sp_team` FOREIGN KEY (`team_id`) REFERENCES
`Team` (`team_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.13. Team

```
CREATE TABLE `Team` (
  `team_id` int NOT NULL AUTO_INCREMENT,
  `team_name` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `location_id` int NOT NULL,
  `gender` enum('Boys','Girls') COLLATE utf8mb3_unicode_ci NOT NULL,
  `captain_pid` int DEFAULT NULL,
  PRIMARY KEY (`team_id`),
  KEY `location_id` (`location_id`),
  KEY `captain_pid` (`captain_pid`),
  CONSTRAINT `fk_team_captain` FOREIGN KEY (`captain_pid`)
REFERENCES `Personnel` (`personnel_id`),
  CONSTRAINT `fk_team_location` FOREIGN KEY (`location_id`)
REFERENCES `Location` (`location_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.14. Payment

```
CREATE TABLE `Payment` (
  `payment_id` int NOT NULL AUTO_INCREMENT,
  `club_member_number` int NOT NULL,
  `payment_date` date NOT NULL,
  `payment_amount` decimal(7,2) NOT NULL,
  `method_of_payment` enum('Cash','Debit','Credit Card') COLLATE
utf8mb3_unicode_ci NOT NULL,
  `membership_year` year NOT NULL,
  PRIMARY KEY (`payment_id`),
  KEY `Payment_ibfk_1` (`club_member_number`),
  CONSTRAINT `Payment_ibfk_1` FOREIGN KEY (`club_member_number`)
REFERENCES `ClubMember` (`club_member_number`) ON DELETE
CASCADE
```

```
) ENGINE=InnoDB AUTO_INCREMENT=802 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.1.15. Email_Log

```
CREATE TABLE `Email_Log` (
  `email_log_id` int NOT NULL AUTO_INCREMENT,
  `email_date` datetime NOT NULL,
  `sender` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `recipient` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  `cc_list` varchar(500) COLLATE utf8mb3_unicode_ci DEFAULT NULL,
  `subject` varchar(200) COLLATE utf8mb3_unicode_ci NOT NULL,
  `body_preview` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL,
  PRIMARY KEY (`email_log_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_unicode_ci
```

4.2. Sample Data Insertion

4.2.1. Location

```
INSERT INTO `iqc353_4`.`Location` (`location_id`, `location_type`, `name`, `address`,
`city`, `province`, `postal_code`, `phone_number`, `web_address`, `max_capacity`)
VALUES
('1', 'Head', MTL Headquarter', '123 Main St', 'Montreal', 'QC', 'H0H0H0', '514-111-1111',
'www.myvc-head.ca', '1000'),
('2', 'Branch', Laval Branch', '456 Elm Rd', 'Laval', 'QC', 'H7T2X2', '450-222-2222',
'www.myvc-laval.ca', '500'),
('3', 'Branch', Longueuil Branch', '789 Oak Ave', 'Longueuil', 'QC', 'J4K1T5',
'450-333-3333', 'www.myvc-longueuil.ca', '400'),
('4', 'Branch', Quebec City Branch', '321 Maple Blvd', 'Quebec City', 'QC', 'G1G2H8',
'418-444-4444', 'www.myvc-quebec.ca', '300'),
('5', 'Branch', Ottawa Sports Complex', '877 Sports Ave', 'Montreal', 'QC', 'H3G1M8',
'514-123-4567', 'www.myvc-ottawasports.ca', '500');
```

4.2.2. Personnel

```
INSERT INTO `iqc353_4`.`Personnel` (`personnel_id`, `first_name`, `last_name`,
`birth_date`, `sin_number`, `medicare_number`, `telephone_number`, `address`, `city`,
`province`, `postal_code`, `email_address`, `mandate`) VALUES
('1', 'John', 'Smith', '1975-11-01', 'SIN000151', 'MCN000151', '514-123-4567', '1 First St',
'Montreal', 'QC', 'H9F2L5', 'email151@definitelyarealemail.ca', 'volunteer'),
```

('2', 'Alice', 'Johnson', '1984-03-12', 'SIN000152', 'MCN000152', '416-234-5678', '123 Second Ave', 'Toronto', 'ON', 'M5A1B2', 'email152@definitelyarealemail.ca', 'salaried'),
('3', 'Michael', 'Brown', '1990-07-23', 'SIN000153', 'MCN000153', '613-345-6789', '45 Maple Rd', 'Ottawa', 'ON', 'K1A0B1', 'email153@definitelyarealemail.ca', 'volunteer'),
('4', 'Sarah', 'Taylor', '1980-11-19', 'SIN000154', 'MCN000154', '514-234-7890', '67 Oak St', 'Montreal', 'QC', 'H2X3Y9', 'email154@definitelyarealemail.ca', 'salaried'),
('5', 'David', 'Williams', '1992-01-10', 'SIN000155', 'MCN000155', '905-456-1234', '89 Pine Dr', 'Toronto', 'ON', 'L4T1C3', 'email155@definitelyarealemail.ca', 'volunteer'),
('6', 'Jessica', 'Davis', '1987-05-30', 'SIN000156', 'MCN000156', '514-567-8901', '100 Birch Rd', 'Montreal', 'QC', 'H3C4P6', 'email156@definitelyarealemail.ca', 'salaried'),
('7', 'Daniel', 'Miller', '1995-08-22', 'SIN000157', 'MCN000157', '416-678-9012', '11 Elm St', 'Toronto', 'ON', 'M6G2G1', 'email157@definitelyarealemail.ca', 'volunteer'),
('8', 'Laura', 'Moore', '1983-02-14', 'SIN000158', 'MCN000158', '613-789-0123', '56 Cedar Ave', 'Ottawa', 'ON', 'K2P1X7', 'email158@definitelyarealemail.ca', 'salaried'),
('9', 'James', 'Wilson', '1978-09-05', 'SIN000159', 'MCN000159', '905-234-5678', '23 Fir Dr', 'Toronto', 'ON', 'L5A3Z1', 'email159@definitelyarealemail.ca', 'volunteer'),
('10', 'Emily', 'Anderson', '1993-12-17', 'SIN000160', 'MCN000160', '514-678-9012', '78 Oak Ave', 'Montreal', 'QC', 'H3A1T5', 'email160@definitelyarealemail.ca', 'salaried');

4.2.3. FamilyMember

INSERT INTO `iqc353_4`.`FamilyMember` (`family_member_id`, `first_name`, `last_name`, `birth_date`, `sin_number`, `medicare_number`, `telephone_number`, `address`, `city`, `province`, `postal_code`, `email_address`) VALUES
('1', 'Joe', 'Smith', '1980-01-01', 'SIN000101', 'MCN000101', '514-123-4567', '1 First St', 'Montreal', 'QC', 'H5A1E5', 'email101@definitelyarealemail.ca'),
('2', 'Anna', 'Johnson', '1985-06-15', 'SIN000102', 'MCN000102', '416-234-5678', '2 Second Ave', 'Toronto', 'ON', 'M5A2B1', 'email102@definitelyarealemail.ca'),
('3', 'David', 'Williams', '1980-03-20', 'SIN000103', 'MCN000103', '438-345-6789', '3 Third Blvd', 'Ottawa', 'ON', 'K1A0B1', 'email103@definitelyarealemail.ca'),
('4', 'Olivia', 'Brown', '1982-11-11', 'SIN000104', 'MCN000104', '819-456-7890', '4 Fourth Rd', 'Gatineau', 'QC', 'J8X2E1', 'email104@definitelyarealemail.ca'),
('5', 'Lucas', 'Jones', '1988-04-28', 'SIN000105', 'MCN000105', '705-567-8901', '5 Fifth St', 'Hamilton', 'ON', 'L8N3A1', 'email105@definitelyarealemail.ca'),
('6', 'Sophia', 'Garcia', '1985-07-22', 'SIN000106', 'MCN000106', '514-678-9012', '6 Sixth Ave', 'Quebec City', 'QC', 'G1R4Y5', 'email106@definitelyarealemail.ca'),
('7', 'Ethan', 'Martinez', '1983-09-14', 'SIN000107', 'MCN000107', '416-789-0123', '7 Seventh Rd', 'Ottawa', 'ON', 'K1P1N2', 'email107@definitelyarealemail.ca'),
('8', 'Mia', 'Davis', '1987-02-03', 'SIN000108', 'MCN000108', '613-890-1234', '8 Eighth Blvd', 'Montreal', 'QC', 'H3B5K3', 'email108@definitelyarealemail.ca'),

```
('9', 'James', 'Rodriguez', '1981-12-05', 'SIN000109', 'MCN000109', '289-901-2345', '9
Ninth St', 'Toronto', 'ON', 'M6B3G2', 'email109@definitelyarealemail.ca'),
('10', 'Charlotte', 'Wilson', '1983-10-21', 'SIN000110', 'MCN000110', '613-234-3456', '10
Tenth Ave', 'Ottawa', 'ON', 'K2P2L4', 'email110@definitelyarealemail.ca');
```

4.2.4. ClubMember

```
INSERT INTO iqc353_4.ClubMember (club_member_number, first_name, last_name,
birth_date, gender, height, weight, sin_number, medicare_number, telephone_number,
address, city, province, postal_code, active_status, deactivation_date, email_address)
VALUES
('1', 'John', 'Doe', '2012-07-15', 'Male', '130', '40', 'SIN000001', 'MCN000001',
'514-123-4567', '1 First St', 'Montreal', 'QC', 'H1B9Q2', '1', NULL,
'email1@definitelyarealemail.ca'),
('2', 'Emma', 'Smith', '2010-05-20', 'Female', '140', '45', 'SIN000002', 'MCN000002',
'514-234-5678', '2 Second St', 'Ottawa', 'ON', 'K1A0B1', '1', NULL,
'email2@definitelyarealemail.ca'),
('3', 'Oliver', 'Johnson', '2008-04-25', 'Male', '150', '55', 'SIN000003', 'MCN000003',
'416-345-6789', '3 Third St', 'Toronto', 'ON', 'M5A1A1', '1', NULL,
'email3@definitelyarealemail.ca'),
('4', 'Sophia', 'Brown', '2013-03-18', 'Female', '135', '38', 'SIN000004', 'MCN000004',
'819-456-7890', '4 Fourth St', 'Quebec City', 'QC', 'G1A2A2', '1', NULL,
'email4@definitelyarealemail.ca'),
('5', 'Mason', 'Williams', '2011-08-10', 'Male', '160', '65', 'SIN000005', 'MCN000005',
'418-567-8901', '5 Fifth St', 'Sherbrooke', 'QC', 'J1H4B4', '1', NULL,
'email5@definitelyarealemail.ca'),
('6', 'Isabella', 'Jones', '2009-02-15', 'Female', '125', '40', 'SIN000006', 'MCN000006',
'514-678-9012', '6 Sixth St', 'Montreal', 'QC', 'H2Y1X2', '1', NULL,
'email6@definitelyarealemail.ca'),
('7', 'James', 'Garcia', '2012-09-30', 'Male', '140', '50', 'SIN000007', 'MCN000007',
'905-789-0123', '7 Seventh St', 'Toronto', 'ON', 'M6G2J2', '1', NULL,
'email7@definitelyarealemail.ca'),
('8', 'Charlotte', 'Martinez', '2010-12-25', 'Female', '130', '44', 'SIN000008', 'MCN000008',
'416-890-1234', '8 Eighth St', 'Ottawa', 'ON', 'K2P0G1', '1', NULL,
'email8@definitelyarealemail.ca'),
('9', 'Liam', 'Hernandez', '2007-03-28', 'Male', '170', '75', 'SIN000009', 'MCN000009',
'819-234-5678', '9 Ninth St', 'Gatineau', 'QC', 'J8T4V4', '0', '2025-04-01',
'email9@definitelyarealemail.ca'),
('10', 'Ava', 'Young', '2011-11-12', 'Female', '125', '38', 'SIN000010', 'MCN000010',
'705-345-6789', '10 Tenth St', 'Barrie', 'ON', 'L4N3X9', '1', NULL,
'email10@definitelyarealemail.ca');
```

4.2.5. FamilyMember_Location

```
INSERT INTO `iqc353_4`.`FamilyMember_Location` (`family_member_id`,
`location_id`, `start_date`, `end_date`) VALUES
('1', '3', '2019-03-01', '2019-03-10'),
('2', '1', '2019-04-02', '2019-04-10'),
('3', '5', '2019-05-05', '2019-05-15'),
('4', '2', '2019-06-01', '2019-06-10'),
('5', '4', '2019-07-01', '2019-07-10'),
('1', '1', '2020-01-01', NULL),
('2', '2', '2020-01-02', NULL),
('3', '3', '2020-01-03', NULL),
('4', '4', '2020-01-04', NULL),
('5', '5', '2020-01-05', NULL);
```

4.2.6. ClubMember_Location

```
INSERT INTO `iqc353_4`.`ClubMember_Location` (`club_member_number`,
`location_id`, `start_date`, `end_date`) VALUES
('1', '3', '2024-06-14', '2024-08-29'),
('2', '2', '2023-07-20', '2024-01-10'),
('3', '4', '2023-09-12', '2024-03-22'),
('4', '5', '2022-10-11', '2023-04-03'),
('5', '1', '2024-02-15', '2024-07-18'),
('1', '1', '2025-01-01', NULL),
('2', '1', '2025-02-10', NULL),
('3', '2', '2025-03-12', NULL),
('4', '2', '2024-04-23', NULL),
('5', '3', '2024-05-30', NULL);
```

4.2.7. ClubMember_FamilyMember

```
INSERT INTO iqc353_4.ClubMember_FamilyMember (club_member_number,
family_member_id, relationship) VALUES
('1', '1', 'Mother'),
('2', '1', 'Mother'),
('3', '2', 'Father'),
('4', '2', 'Father'),
('5', '3', 'Grandfather'),
('6', '3', 'Grandfather'),
('7', '4', 'Grandmother'),
('8', '4', 'Grandmother'),
('9', '5', 'Uncle'),
```

```
('10', '5', 'Uncle');
```

4.2.8. SecondaryFamilyMember

```
INSERT INTO `iqc353_4`.`SecondaryFamilyMember` (`secondary_id`,
`primary_family_member_id`, `first_name`, `last_name`, `telephone_number`) VALUES
('1', '5' 'Joe Smith', '514-123-4567'),
('2', '10' 'Sarah Johnson', '514-234-5678'),
('3', '15' 'Michael Brown', '514-345-6789'),
('4', '20' 'Emily Williams', '514-456-7890'),
('5', '25' 'David Jones', '514-567-8901'),
('6', '30' 'Laura Garcia', '514-678-9012'),
('7', '35' 'Chris Martinez', '514-789-0123'),
('8', '40' 'Rachel Rodriguez', '514-890-1234'),
('9', '45' 'James Hernandez', '514-901-2345'),
('10', '50' 'Olivia Lopez', '514-012-3456');
```

4.2.9. SecondaryFamilyMember_ClubMember

```
INSERT INTO `iqc353_4`.`SecondaryFamilyMember_ClubMember` (`secondary_id`,
`club_member_number`, `relationship`) VALUES
('1', '10', 'Mother'),
('2', '20', 'Father'),
('3', '30', 'Grandfather'),
('4', '40', 'Grandmother'),
('5', '50', 'Uncle'),
('6', '60', 'Aunt'),
('7', '70', 'Tutor'),
('8', '80', 'Partner'),
('9', '90', 'Friend'),
('10', '100', 'Other');
```

4.2.10. Personnel_Location_Role

```
INSERT INTO `iqc353_4`.`Personnel_Location_Role` (`personnel_id`, `location_id`,
`role`, `start_date`) VALUES
('1', '1', 'general_manager', '2024-01-01'),
('2', '1', 'deputy_manager', '2024-01-02'),
('3', '1', 'treasurer', '2024-01-03'),
('4', '1', 'secretary', '2024-01-04'),
('5', '1', 'administrator', '2024-01-05'),
('6', '1', 'administrator', '2024-01-06'),
```

```
('7', '2', 'general_manager', '2024-02-01'),
('8', '2', 'coach', '2024-02-02'),
('9', '2', 'administrator', '2024-02-03'),
('10', '3', 'general_manager', '2024-03-01');
```

4.2.11. Payment

```
INSERT INTO `iqc353_4`.`Payment` (`payment_id`, `club_member_number`,
`payment_date`, `payment_amount`, `method_of_payment`, `membership_year`)
VALUES
(1, 50, '2024-02-15', 30, 'Cash', 2025),
(2, 50, '2024-04-10', 35, 'Debit', 2025),
(3, 50, '2024-07-22', 35, 'Credit Card', 2025),
(4, 51, '2024-01-05', 50, 'Cash', 2025),
(5, 51, '2024-03-16', 50, 'Debit', 2025),
(6, 52, '2024-05-23', 120, 'Credit Card', 2025),
(7, 53, '2024-06-10', 25, 'Cash', 2025),
(8, 53, '2024-08-18', 75, 'Debit', 2025),
(9, 54, '2024-09-09', 45, 'Credit Card', 2025),
(10, 54, '2024-11-02', 55, 'Cash', 2025),
```

4.2.12. Session

```
INSERT INTO `iqc353_4`.`Session` (`session_id`, `session_date`, `session_time`,
`session_type`, `location_id`, `team1_id`, `team2_id`, `score_team1`, `score_team2`,
`coach_pid`) VALUES
('1', '2025-04-06', '10:30:00', 'training', '1', '1', '3', '5', '3', '30'),
('2', '2025-04-07', '14:00:00', 'game', '2', '2', '4', '2', '4', '8'),
('3', '2025-04-08', '09:15:00', 'training', '2', '7', '9', '4', '6', '24'),
('4', '2025-04-09', '16:45:00', 'game', '3', '8', '10', '5', '3', '11'),
('5', '2025-04-10', '11:30:00', 'training', '1', '11', '13', '2', '4', '30'),
('6', '2025-04-11', '13:00:00', 'game', '1', '6', '8', '6', '7', '33'),
('7', '2025-04-12', '10:45:00', 'training', '2', '12', '14', '7', '6', '8'),
('8', '2025-04-13', '15:30:00', 'game', '2', '17', '19', '3', '2', '40'),
('9', '2025-04-14', '09:00:00', 'training', '3', '18', '20', '1', '1', '24'),
('10', '2025-04-15', '12:00:00', 'game', '5', '15', '17', '4', '5', '14');
```

4.2.13. Session_Player

```
INSERT INTO `iqc353_4`.`Session_Player` (`session_id`, `club_member_number`,
`team_id`, `player_position`) VALUES
('1', '1', '1', 'opposite'),
```

```
('1', '5', '3', 'opposite'),
('1', '11', '1', 'setter'),
('1', '15', '3', 'libero'),
('1', '25', '3', 'serving_specialist'),
('1', '35', '3', 'middle_blocker'),
('1', '41', '1', 'outside_hitter'),
('1', '51', '1', 'serving_specialist'),
('1', '55', '3', 'outside_hitter'),
('1', '61', '1', 'libero');
```

4.2.14. Team

```
INSERT INTO `iqc353_4`.`Team` (`team_id`, `team_name`, `location_id`, `gender`,
`captain_pid`) VALUES
('1', 'The Thunderhawks', '1', 'Boys', '51'),
('2', 'The Night Riders', '2', 'Girls', '52'),
('3', 'The Storm Chasers', '3', 'Boys', '53'),
('4', 'The Firebirds', '4', 'Girls', '54'),
('5', 'The Ice Dragons', '5', 'Boys', '55'),
('6', 'The Silver Falcons', '1', 'Girls', '56'),
('7', 'The Phantom Wolves', '2', 'Boys', '57'),
('8', 'The Crimson Valkyries', '3', 'Girls', '58'),
('9', 'The Golden Griffins', '4', 'Boys', '59'),
('10', 'The Jade Tigers', '5', 'Girls', '60');
```

4.2.15. Email_Log

```
INSERT INTO Email_Log (email_log_id, email_date, sender, recipient, cc_list, subject,
body_preview)
VALUES
  (1, '2025-03-14 10:00:00', 'system@myvc.ca', 'amy.stone@example.com',
'coach@myvc.ca',
   'Montreal Boys U15 20-Mar-2025 18:00 training session',
   'Hello Amy Stone, your role is outside hitter...'),
  (2, '2025-03-14 10:01:00', 'system@myvc.ca', 'brian.stone@example.com', NULL,
   'Montreal Boys U15 20-Mar-2025 18:00 training session',
   'Hello Brian Stone, your role is opposite...'),
  (3, '2025-03-14 10:02:00', 'system@myvc.ca', 'cathy.miller@example.com', NULL,
   'Laval Girls U15 20-Mar-2025 18:00 training session',
   'Hello Cathy Miller, your role is libero...'),
  (4, '2025-03-01 09:00:00',  'system@myvc.ca', 'david.garcia@example.com', NULL,
   'Deactivation Notification',
```

```
 'Dear David Garcia, we regret to inform you...'),
(5, '2025-03-14 10:03:00', 'system@myvc.ca', 'elena.garcia@example.com', NULL,
 'Longueuil Boys U16 21-Mar-2025 16:00 game session',
 'Hello Elena Garcia, your role is outside hitter...');
```

5. SQL Queries and Transactions

5.1. SQL Queries for Data Retrieval

5.1.1. Create/Delete/Edit/Display a Location

5.1.1.1. Create a Location

```
INSERT INTO `iqc353_4`.`Location` (`location_id`,
`location_type`, `name`, `address`, `city`, `province`,
`postal_code`, `phone_number`, `web_address`,
`max_capacity`) VALUES
('1', 'Head MTL Headquarter', '123 Main St', 'Montreal', 'QC',
'H0H0H0', '514-111-1111', 'www.myvc-head.ca', '1000');
```

5.1.1.2. Delete a Location

```
DELETE FROM `iqc353_4`.`Location`
WHERE `location_id` = '1';
```

5.1.1.3. Edit a Location

```
UPDATE `iqc353_4`.`Location`
SET `name` = 'New Name Here'
WHERE `location_id` = '1';
```

5.1.1.4. Display a Location

```
SELECT * FROM `iqc353_4`.`Location`
WHERE `location_id` = '1';
```

5.1.2. Create/Delete/Edit/Display a Personnel

5.1.2.1. Create a Personnel

```
INSERT INTO `iqc353_4`.`Personnel` (`personnel_id`,
`first_name`, `last_name`, `birth_date`, `sin_number`,
`medicare_number`, `telephone_number`, `address`, `city`,
`province`, `postal_code`, `email_address`, `mandate`)
VALUES
('1', 'John', 'Smith', '1975-11-01', 'SIN000151', 'MCN000151',
'514-123-4567', '1 First St', 'Montreal', 'QC', 'H9F2L5',
'email151@definitelyarealemail.ca', 'volunteer');
```

5.1.2.2. Delete a Personnel

```
DELETE FROM Personnel
WHERE personnel_id = 11;
```

5.1.2.3. Edit a Personnel

```
UPDATE `iqc353_4`.`Personnel`
SET `first_name` = 'New Name Here'
WHERE `personnel_id` = '1';
```

5.1.2.4. Display a Personnel

```
SELECT * FROM `iqc353_4`.`Personnel`
WHERE `personnel_id` = '1';
```

5.1.3. Create/Delete/Edit/Display a FamilyMember (Primary)

5.1.3.1. Create a FamilyMember

```
INSERT INTO `iqc353_4`.`FamilyMember`
(`family_member_id`, `first_name`, `last_name`, `birth_date`,
`sin_number`, `medicare_number`, `telephone_number`,
`address`, `city`, `province`, `postal_code`, `email_address`)
VALUES
('1', 'Joe', 'Smith', '1980-01-01', 'SIN000101', 'MCN000101',
'514-123-4567', '1 First St', 'Montreal', 'QC', 'H5A1E5',
'email101@definitelyarealemail.ca');
```

5.1.3.2. Delete a FamilyMember

```
DELETE FROM `iqc353_4`.`FamilyMember`
WHERE `family_member_id` = '1';
```

5.1.3.3. Edit a FamilyMember

```
UPDATE `iqc353_4`.`FamilyMember`
SET `first_name` = 'New Name Here'
WHERE `family_member_id` = '1';
```

5.1.3.4. Display a FamilyMember

```
SELECT * FROM `iqc353_4`.`FamilyMember`
WHERE `family_member_id` = '1';
```

5.1.3.5. Create a secondary FamilyMember

```
INSERT INTO `iqc353_4`.`SecondaryFamilyMember`
(`secondary_id`, `primary_family_member_id`, `first_name`,
`last_name`, `telephone_number`) VALUES
('1', '5' 'Joe Smith', '514-123-4567');
```

5.1.3.6. Delete a secondary FamilyMember

```
DELETE FROM `iqc353_4`.`SecondaryFamilyMember`
WHERE `secondary_id` = '1';
```

5.1.3.7. Edit a secondary FamilyMember

```
UPDATE `iqc353_4`.`SecondaryFamilyMember`
SET `first_name` = 'New Name Here'
WHERE `secondary_id` = '1';
```

5.1.3.8. Display a secondary FamilyMember

```
SELECT * FROM `iqc353_4`.`SecondaryFamilyMember`
WHERE `secondary_id` = '1';
```

5.1.4. Create/Delete/Edit/Display a ClubMember

5.1.4.1. Create a ClubMember

```
INSERT INTO iqc353_4.ClubMember (club_member_number,
first_name, last_name, birth_date, gender, height, weight,
sin_number, medicare_number, telephone_number, address,
city, province, postal_code, active_status, deactivation_date,
email_address) VALUES
('1', 'John', 'Doe', '2012-07-15', 'Male', '130', '40', 'SIN000001',
'MCN000001', '514-123-4567', '1 First St', 'Montreal', 'QC',
'H1B9Q2', '1', NULL, 'email1@definitelyarealemail.ca');
```

5.1.4.2. Delete a ClubMember

```
DELETE FROM `iqc353_4`.`ClubMember`
WHERE `club_member_number` = '1';
```

5.1.4.3. Edit a ClubMember

```
UPDATE `iqc353_4`.`ClubMember`
SET `first_name` = 'New Name Here'
WHERE `club_member_number` = '1';
```

5.1.4.4. Display a ClubMember

```
SELECT * FROM `iqc353_4`.`ClubMember`
WHERE `club_member_number` = '1';
```

5.1.5. Create/Delete/Edit/Display a TeamFormation

5.1.5.1. Create a TeamFormation

```
INSERT INTO `iqc353_4`.`Team` (`team_id`, `team_name`,
`location_id`, `gender`, `captain_pid`) VALUES
```

('1', 'The Thunderhawks', '1', 'Boys', '51');

5.1.5.2. Delete a TeamFormation

```
DELETE FROM `iqc353_4`.`Team`
WHERE `team_id` = '1';
```

5.1.5.3. Edit a TeamFormation

```
UPDATE `iqc353_4`.`Team`
SET `team_name` = 'New Name Here'
WHERE `team_id` = '1';
```

5.1.5.4. Display a TeamFormation

```
SELECT * FROM `iqc353_4`.`Team`
WHERE `team_id` = '1';
```

5.1.6. 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).

5.1.6.1. Assign a club member to a team formation

```
INSERT INTO `iqc353_4`.`Session_Player` (`session_id`,
`club_member_number`, `team_id`, `player_position`)
VALUES
('1', '1', '1', 'opposite');
```

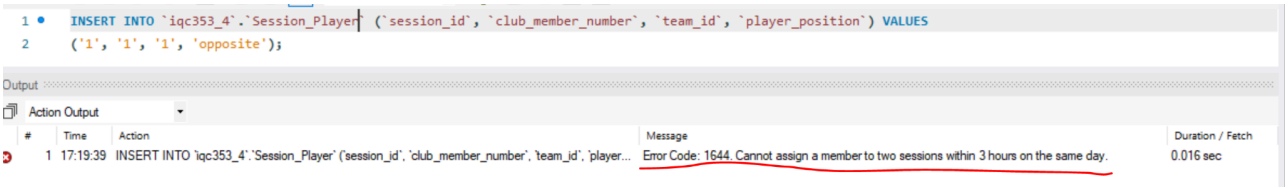
5.1.6.2. Delete a club member from a team formation

```
DELETE FROM `iqc353_4`.`Session_Player`
WHERE `session_id` = 1 AND `team_id` = 1 AND
`club_member_number` = 1;
```

5.1.6.3. Edit the member's role in a session

```
UPDATE `iqc353_4`.`Session_Player`
SET `club_member_number` = 2
WHERE `session_id` = 1
  AND `team_id` = 1
  AND `club_member_number` = 1;
```

5.1.6.4. Attempt to add a player twice within 3 hours



5.1.7. 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

```
SELECT loc.location_id,
        loc.name                AS location_name,
        loc.address,
        loc.city,
        loc.province,
        loc.postal_code,
        loc.phone_number,
        loc.web_address,
        loc.location_type,
        loc.max_capacity,
        CONCAT(p.first_name, ' ', p.last_name) AS
general_manager_name,
        COALESCE(cm_count.num_members, 0)         AS
number_of_club_members
FROM Location loc

-- find the general manager
LEFT JOIN (
        SELECT plr.location_id,
        plr.personnel_id
        FROM Personnel_Location_Role plr
        WHERE plr.role = 'general_manager'
        AND plr.end_date IS NULL
) AS gm ON gm.location_id = loc.location_id

LEFT JOIN Personnel p
  ON p.personnel_id = gm.personnel_id

-- find the count of active club members for that location
LEFT JOIN (
        SELECT cml.location_id,
        COUNT(DISTINCT cml.club_member_number) AS
num_members
        FROM ClubMember_Location cml
```

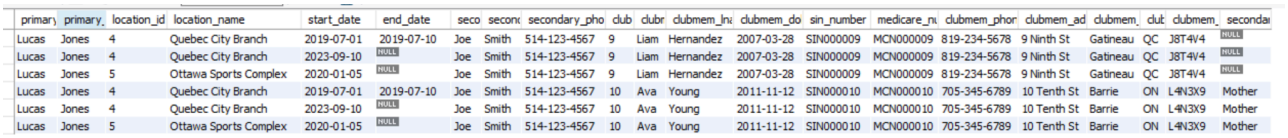
```
        JOIN ClubMember cm ON cm.club_member_number =
cml.club_member_number
                AND cm.active_status = TRUE
        WHERE cml.end_date IS NULL
        GROUP BY cml.location_id
) AS cm_count ON cm_count.location_id = loc.location_id

ORDER BY loc.province, loc.city;
```

location_id	location_name	address	city	province	postal_code	phone_number	web_address	location_type	max_capacity	general_manager_name	number_of_club_members
5	Ottawa Sports Complex	877 Sports Ave	Ottawa	ON	H3G1M8	514-123-4567	www.myvc-ottawasports.ca	Branch	500	Ethan King	19
2	Laval Branch	456 Elm Rd	Laval	QC	H7T2X2	450-222-2222	www.myvc-laval.ca	Branch	500	Maya Martínez	19
3	Longueuil Branch	789 Oak Ave	Longueuil	QC	J4K1T5	450-333-3333	www.myvc-longueuil.ca	Branch	400	Emily Anderson	20
1	MTL Headquarter	123 Main St	Montreal	QC	H0H0H0	514-111-1111	www.myvc-head.ca	Head	1000	Ella Lopez	20
4	Quebec City Branch	321 Maple Blvd	Quebec City	QC	G1G2H8	418-444-4444	www.myvc-quebec.ca	Branch	300	Eleanor Scott	20

5.1.8. 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, SocialSecurity Number, Medicare card number, telephone number, address, city, province, postal-code, and relationship with the secondary family member

```
SELECT
  fm.first_name        AS primary_fname,
  fm.last_name         AS primary_lname,
  fml.location_id,
  loc.name             AS location_name,
  fml.start_date,
  fml.end_date,

  sf.first_name        AS secondary_fname,
  sf.last_name         AS secondary_lname,
  sf.telephone_number AS secondary_phone,

  c.club_member_number,
  c.first_name         AS clubmem_fname,
  c.last_name          AS clubmem_lname,
  c.birth_date         AS clubmem_dob,
  c.sin_number,
  c.medicare_number,
  c.telephone_number  AS clubmem_phone,
  c.address            AS clubmem_address,
  c.city          AS clubmem_city,
```

```
  c.province          AS clubmem_province,
  c.postal_code       AS clubmem_postal_code,

  sc.relationship     AS secondary_relationship
FROM FamilyMember fm
-- family's location history
LEFT JOIN FamilyMember_Location fml
   ON fm.family_member_id = fml.family_member_id
LEFT JOIN Location loc
   ON loc.location_id = fml.location_id

-- secondary family member
LEFT JOIN SecondaryFamilyMember sf
   ON sf.primary_family_member_id = fm.family_member_id

-- find all club members (children) associated with the primary
LEFT JOIN ClubMember_FamilyMember cfm
   ON cfm.family_member_id = fm.family_member_id
LEFT JOIN ClubMember c
   ON c.club_member_number = cfm.club_member_number

-- relationship with the secondary family member (if any)
LEFT JOIN SecondaryFamilyMember_ClubMember sc
   ON sc.secondary_id = sf.secondary_id
   AND sc.club_member_number = c.club_member_number

WHERE fm.family_member_id = FAMILY_MEMBER_ID
ORDER BY fm.last_name, fm.first_name, c.last_name, c.first_name;
```



primary	primary_	location_id	location_name	start_date	end_date	seco	secon	secondary_pho	club	clubr	clubmem_ln	clubmem_do	sin_number	medicare_n	clubmem_phor	clubmem_ad	clubmem_	clut	clubmem_	secondar
Lucas	Jones	4	Quebec City Branch	2019-07-01	2019-07-10	Joe	Smith	514-123-4567	9	Liam	Hernandez	2007-03-28	SIN000009	MCN000009	819-234-5678	9 Ninth St	Gatineau	QC	J8T4V4	NULL
Lucas	Jones	4	Quebec City Branch	2023-09-10	NULL	Joe	Smith	514-123-4567	9	Liam	Hernandez	2007-03-28	SIN000009	MCN000009	819-234-5678	9 Ninth St	Gatineau	QC	J8T4V4	NULL
Lucas	Jones	5	Ottawa Sports Complex	2020-01-05	NULL	Joe	Smith	514-123-4567	9	Liam	Hernandez	2007-03-28	SIN000009	MCN000009	819-234-5678	9 Ninth St	Gatineau	QC	J8T4V4	NULL
Lucas	Jones	4	Quebec City Branch	2019-07-01	2019-07-10	Joe	Smith	514-123-4567	10	Ava	Young	2011-11-12	SIN000010	MCN000010	705-345-6789	10 Tenth St	Barrie	ON	L4N3X9	Mother
Lucas	Jones	4	Quebec City Branch	2023-09-10	NULL	Joe	Smith	514-123-4567	10	Ava	Young	2011-11-12	SIN000010	MCN000010	705-345-6789	10 Tenth St	Barrie	ON	L4N3X9	Mother
Lucas	Jones	5	Ottawa Sports Complex	2020-01-05	NULL	Joe	Smith	514-123-4567	10	Ava	Young	2011-11-12	SIN000010	MCN000010	705-345-6789	10 Tenth St	Barrie	ON	L4N3X9	Mother

5.1.9. 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 team's 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

```
SELECT s.session_id,
       s.session_date,
       s.session_time,
       loc.address,
       s.session_type,
       CONCAT(pc.first_name, ' ', pc.last_name) AS head_coach,
       t1.team_name AS team1_name,
       s.score_team1,
       t2.team_name AS team2_name,
       s.score_team2,
       CONCAT(cm.first_name, ' ', cm.last_name) AS player_name,
       sp.player_position,
       t3.team_name AS player_team_name  -- Added column for the
player's team
FROM Session s
JOIN Team t1
  ON s.team1_id = t1.team_id
JOIN Team t2
  ON s.team2_id = t2.team_id
LEFT JOIN Personnel pc
  ON pc.personnel_id = s.coach_pid
LEFT JOIN Session_Player sp
  ON sp.session_id = s.session_id
LEFT JOIN ClubMember cm
  ON cm.club_member_number = sp.club_member_number
JOIN Location loc  -- Join the Location table to get the address
  ON s.location_id = loc.location_id  -- Match the location_id from
Session and Location
JOIN Team t3  -- Join to get the player's team
  ON sp.team_id = t3.team_id  -- Assuming sp.team_id is the player's
team_id
WHERE s.location_id = LOCATION_ID
  AND s.session_date BETWEEN WEEK START AND WEEK END
ORDER BY s.session_date, s.session_time;
```

session_id	session_date	session_time	address	session_type	head_coach	team1_name	score_team1	team2_name	score_team2	player_name	player_position	player_team_name
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Mason Williams	defensive_specialist	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Matthew Moore	libero	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Jack Scott	outside_hitter	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Daniel Perez	setter	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	David Evans	setter	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Luke Lopez	libero	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Mason Perez	middle_blocker	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Ethan Lee	opposite	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Lucas Taylor	opposite	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Matthew Lopez	outside_hitter	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Ethan Evans	defensive_specialist	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Logan Harris	serving_specialist	The Mystic Panthers
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Elijah Wilson	middle_blocker	The Sapphire Bears
5	2025-04-10	11:30:00	123 Main St	training	Isaac Carter	The Mystic Panthers	2	The Sapphire Bears	4	Benjamin Taylor	serving_specialist	The Sapphire Bears
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Emma Smith	opposite	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Isabella Jones	defensive_specialist	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Evelyn Taylor	outside_hitter	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Mila Wilson	libero	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Charlotte Walker	libero	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Isabella Jackson	opposite	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Ella Scott	serving_specialist	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Sophia King	outside_hitter	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Ava Walker	middle_blocker	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Evelyn Johnson	middle_blocker	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Sophia White	setter	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Emma Evans	serving_specialist	The Crimson Valkyries
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Sophie Roberts	defensive_specialist	The Silver Falcons
6	2025-04-11	13:00:00	123 Main St	game	Amelia Hern...	The Silver Falcons	6	The Crimson Valky...	7	Lillian Wilson	setter	The Crimson Valkyries

5.1.10. 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

```
SELECT * FROM iqc353_4.ClubMember;SELECT
c.club_member_number,
        c.first_name,
        c.last_name
FROM ClubMember c

-- fetch distinct locations they belong to
JOIN (
  SELECT cml.club_member_number,
        COUNT(DISTINCT cml.location_id) AS loc_count,
        MIN(cml.start_date)          AS first_join_date
  FROM ClubMember_Location cml
  GROUP BY cml.club_member_number
) AS sub
  ON sub.club_member_number = c.club_member_number

WHERE c.active_status = TRUE
 AND sub.loc_count >= 3
 AND TIMESTAMPDIFF(YEAR, sub.first_join_date, CURDATE())
>= 3

ORDER BY c.club_member_number;
```

	club	first_name	last_name
▶	3	Oliver	Johnson
	11	Mason	Moore
	14	Grace	Thomas
	17	Benjamin	Harris
	20	Lily	Perez

5.1.11. 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. 1st, 2025 to Mar. 31st, 2025

```
SELECT
   l.name AS location_name,
   SUM(CASE WHEN s.session_type = 'training' THEN 1 ELSE 0
END) AS total_training_sessions,
   SUM(CASE WHEN s.session_type = 'training' THEN
sp.player_count ELSE 0 END) AS total_players_in_training_sessions,
   SUM(CASE WHEN s.session_type = 'game' THEN 1 ELSE 0 END)
AS total_game_sessions,
   SUM(CASE WHEN s.session_type = 'game' THEN sp.player_count
ELSE 0 END) AS total_players_in_game_sessions
FROM
   Location l
JOIN
   Session s ON l.location_id = s.location_id
LEFT JOIN (
   SELECT
      session_id,
      COUNT(DISTINCT club_member_number) AS player_count
   FROM
```

```
        Session_Player
    GROUP BY
        session_id
) sp ON s.session_id = sp.session_id
WHERE
    s.session_date BETWEEN START DATE AND END DATE  --
Adjust the date range as needed
GROUP BY
    l.location_id, l.name
HAVING
    SUM(CASE WHEN s.session_type = 'game' THEN 1 ELSE 0 END)
>= 2
ORDER BY
    total_game_sessions DESC;
```

location_name	total_training_sessions	total_players_in_training_sessions	total_game_sessions	total_players_in_game_sessions
Laval Branch	5	70	9	126
MTL Headquarter	7	98	5	70
Quebec City Branch	3	42	5	70
Longueuil Branch	9	126	3	42
Ottawa Sports Complex	1	14	3	42

5.1.12. 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

```
SELECT c.club_member_number,
        c.first_name,
        c.last_name,
        TIMESTAMPDIFF(YEAR, c.birth_date, CURDATE()) AS age,
        c.telephone_number,
        c.email_address,
        loc.name AS current_location_name
FROM ClubMember c

-- get current (active) location
JOIN ClubMember_Location cml
  ON c.club_member_number = cml.club_member_number
  AND cml.end_date IS NULL
JOIN Location loc
  ON loc.location_id = cml.location_id
```

```
WHERE c.active_status = TRUE
  AND NOT EXISTS (
        SELECT 1
        FROM Session_Player sp
        WHERE sp.club_member_number = c.club_member_number
  )
ORDER BY loc.name, c.club_member_number;
```

club	first_name	last_name	age	telephone_number	email_address	current_location_name
102	Sophia	Wilson	11	705-234-5682	email202@definitelyarealemail.ca	Laval Branch
103	Liam	Moore	12	705-234-5683	email203@definitelyarealemail.ca	Longueuil Branch
101	Benjamin	Taylor	13	705-234-5681	email201@definitelyarealemail.ca	MTL Headquarter
105	Ethan	Clark	14	705-234-5685	email205@definitelyarealemail.ca	Ottawa Sports Complex
104	Isabella	Lee	14	705-234-5684	email204@definitelyarealemail.ca	Quebec City Branch

5.1.13. 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 c.club_member_number,
        c.first_name,
        c.last_name,
        TIMESTAMPDIFF(YEAR, c.birth_date, CURDATE()) AS age,
        c.telephone_number,
        c.email_address,
        loc.name AS current_location_name
FROM ClubMember c

JOIN ClubMember_Location cml
   ON c.club_member_number = cml.club_member_number
   AND cml.end_date IS NULL
JOIN Location loc
   ON loc.location_id = cml.location_id

WHERE c.active_status = TRUE
  -- must have at least 1 'outside_hitter'
  AND EXISTS (
        SELECT 1
```

```
        FROM Session_Player sp
        WHERE sp.club_member_number = c.club_member_number
        AND sp.player_position = 'outside_hitter'
    )
    -- must NOT have any session with a role != 'outside_hitter'
    AND NOT EXISTS (
        SELECT 1
        FROM Session_Player sp2
        WHERE sp2.club_member_number = c.club_member_number
        AND sp2.player_position <> 'outside_hitter'
    )
ORDER BY loc.name, c.club_member_number;
```

club	first_name	last_name	age	telephone_number	email_address	current_location_name
107	Ava	Martinez	11	905-234-5679	email207@definitelyarealemail.ca	Laval Branch
108	Noah	Garcia	11	905-234-5680	email208@definitelyarealemail.ca	Longueuil Branch
106	Oliver	King	11	905-234-5678	email206@definitelyarealemail.ca	MTL Headquarter
110	James	Anderson	15	905-234-5682	email210@definitelyarealemail.ca	Ottawa Sports Complex
109	Mia	Rodriguez	16	905-234-5681	email209@definitelyarealemail.ca	Quebec City Branch

5.1.14. 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

```
SELECT c.club_member_number,
        c.first_name,
        c.last_name,
        TIMESTAMPDIFF(YEAR, c.birth_date, CURDATE()) AS age,
        c.telephone_number,
        c.email_address,
        loc.name AS current_location_name
FROM ClubMember c

JOIN ClubMember_Location cml
  ON c.club_member_number = cml.club_member_number
  AND cml.end_date IS NULL
JOIN Location loc
  ON loc.location_id = cml.location_id
```

```
WHERE c.active_status = TRUE
  AND NOT EXISTS (
        -- For each role in the 7 roles, if there's no session_player row,
fail
        SELECT 1
        FROM (
        SELECT 'outside_hitter'     AS role UNION
        SELECT 'opposite'           UNION
        SELECT 'setter'             UNION
        SELECT 'middle_blocker'           UNION
        SELECT 'libero'             UNION
        SELECT 'defensive_specialist'  UNION
        SELECT 'serving_specialist'
        ) AS roles
        WHERE NOT EXISTS (
        SELECT 1
        FROM Session_Player sp
        JOIN Session s ON sp.session_id = s.session_id
        WHERE sp.club_member_number = c.club_member_number
        AND s.session_type = 'game'
        AND sp.player_position = roles.role
        )
  )
ORDER BY loc.name, c.club_member_number;
```

club	first_name	last_name	age	telephone_number	email_address	current_location_name
4	Sophia	Brown	12	819-456-7890	email4@definitelyarealemail.ca	Laval Branch
14	Grace	Thomas	16	705-789-0123	email14@definitelyarealemail.ca	Laval Branch
54	Amelia	Taylor	13	613-234-5670	email54@definitelyarealemail.ca	Laval Branch
84	Zoe	Jackson	17	905-678-9018	email84@definitelyarealemail.ca	Laval Branch
94	Olivia	Scott	12	705-567-8909	email94@definitelyarealemail.ca	Laval Branch

5.1.15. 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

```
SELECT DISTINCT fm.first_name,
        fm.last_name,
        fm.telephone_number
FROM FamilyMember fm

-- tie family -> location if needed
```

```
JOIN FamilyMember_Location fml
   ON fml.family_member_id = fm.family_member_id
  AND fml.location_id = LOCATION_ID
  AND fml.end_date IS NULL

-- tie family -> at least one active child
JOIN ClubMember_FamilyMember cfm
   ON cfm.family_member_id = fm.family_member_id
JOIN ClubMember c
   ON c.club_member_number = cfm.club_member_number
   AND c.active_status = TRUE

JOIN ClubMember_Location cml
   ON cml.club_member_number = c.club_member_number
  AND cml.location_id = LOCATION_ID
  AND cml.end_date IS NULL

-- check if the family member is a 'captain' at that location
JOIN Personnel p
   ON p.medicare_number = fm.medicare_number
JOIN Personnel_Location_Role plr
   ON plr.personnel_id = p.personnel_id
  AND plr.role = 'captain'
  AND plr.location_id = LOCATION_ID
  AND plr.end_date IS NULL

ORDER BY fm.last_name, fm.first_name;
```

first_name	last_name	telephone_number
Sophia	Garcia	514-678-9012
Benjamin	Moore	416-345-4567
James	Rodriguez	289-901-2345
Joe	Smith	514-123-4567
Chloe	White	514-890-9012

5.1.16. 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

```
SELECT c.club_member_number,
        c.first_name,
        c.last_name,
        TIMESTAMPDIFF(YEAR, c.birth_date, CURDATE()) AS age,
        c.telephone_number,
        c.email_address,
        loc.name AS current_location
FROM ClubMember c
JOIN ClubMember_Location cml
  ON c.club_member_number = cml.club_member_number
  AND cml.end_date IS NULL
JOIN Location loc
  ON loc.location_id = cml.location_id

WHERE c.active_status = TRUE
 -- must have at least 1 game
 AND EXISTS (
        SELECT 1
        FROM Session_Player sp
        JOIN Session s ON sp.session_id = s.session_id
        WHERE sp.club_member_number = c.club_member_number
        AND s.session_type = 'game'
 )
 -- must never have a losing session:
 AND NOT EXISTS (
        SELECT 1
        FROM Session_Player sp
        JOIN Session s   ON s.session_id   = sp.session_id
        JOIN Session st2 ON st2.session_id = s.session_id -- or you can
join the "other team" approach
        WHERE sp.club_member_number = c.club_member_number
        AND s.session_type = 'game'
```

```
        -- if their team's score < other team's score => a loss
        AND (
        (s.team1_id = sp.team_id AND s.score_team1 < s.score_team2)
        OR
        (s.team2_id = sp.team_id AND s.score_team2 < s.score_team1)
        )
    )
ORDER BY loc.name, c.club_member_number;
```

club	first_name	last_name	age	telephone_number	email_address	current_location
3	Oliver	Johnson	16	416-345-6789	email3@definitelyarealemail.ca	Laval Branch
53	Luke	Mitchell	15	905-234-5671	email53@definitelyarealemail.ca	Laval Branch
64	Mason	Wilson	14	613-234-5671	email64@definitelyarealemail.ca	Laval Branch
74	Oliver	Walker	13	416-678-9018	email74@definitelyarealemail.ca	Laval Branch
56	Isabella	Jackson	14	705-234-5672	email56@definitelyarealemail.ca	Longueuil Branch
76	Liam	Lopez	17	514-567-8906	email76@definitelyarealemail.ca	Longueuil Branch
86	Emma	Evans	16	514-123-4567	email86@definitelyarealemail.ca	Longueuil Branch
96	Lillian	Wilson	15	905-234-5672	email96@definitelyarealemail.ca	Longueuil Branch
32	Lily	Adams	13	416-789-0123	email32@definitelyarealemail.ca	MTL Headquarter
57	Jackson	White	13	905-678-9015	email57@definitelyarealemail.ca	Quebec City Branch
109	Mia	Rodriguez	16	905-234-5681	email209@definitelyarealemail.ca	Quebec City Branch

5.1.17. 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 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.first_name,
        p.last_name,
        plr.start_date   AS treasurer_start_date,
        plr.end_date   AS treasurer_end_date
FROM Personnel p
JOIN Personnel_Location_Role plr
  ON p.personnel_id = plr.personnel_id
WHERE plr.role = 'treasurer'
ORDER BY p.first_name, p.last_name, plr.start_date;
```

first_name	last_name	treasurer_start_date	treasurer_end_date
Ethan	King	2012-01-02	2013-01-01
Lucas	Hernández	2010-01-01	2011-01-01
Maya	Brown	2013-01-02	2014-01-01
Michael	Brown	2024-01-03	NULL
Zoe	Lopez	2011-01-02	2012-01-01

5.1.18. 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

```
SELECT
    cm.first_name,
    cm.last_name,
    cm.telephone_number,
    cm.email_address,
    cm.deactivation_date,
    l.name AS last_location,
    COALESCE(sp.player_position, 'No role') AS last_position  --
Default to 'No role' if no position is found
FROM
    ClubMember cm
JOIN
    ClubMember_Location cml ON cm.club_member_number =
cml.club_member_number
JOIN
    Location l ON cml.location_id = l.location_id
LEFT JOIN
    Session_Player sp ON cm.club_member_number =
sp.club_member_number
LEFT JOIN
    Session s ON sp.session_id = s.session_id AND s.session_type =
'game'
WHERE
    cm.active_status = 0
    AND DATEDIFF(CURDATE(), cm.birth_date) / 365.25 > 18
    AND cml.end_date IS NULL  -- Get the last active location
```

```
    AND (
        sp.session_id IS NULL  -- Player did not participate in any game
        OR s.session_date = (
            SELECT MAX(session_date)
            FROM Session
            WHERE session_type = 'game'
            AND session_id IN (
                SELECT session_id
                FROM Session_Player
                WHERE club_member_number = cm.club_member_number
            )
        )
    )
ORDER BY
    l.name,  -- First by location name
    COALESCE(sp.player_position, 'No role'),  -- Then by player
position (role)
    cm.first_name,  -- Then by first name
    cm.last_name;  -- Finally by last name
```

first_name	last_name	telephone_number	email_address	deactivation_date	last_location	last_position
Lucas	Clark	613-234-5678	email33@definitelyarealemail.ca	2025-03-01	Laval Branch	No role
Jan	Corey	905-234-5686	email214@definitelyarealemail.ca	2025-04-01	MTL Headquarter	No role
James	Alban	905-234-5685	email213@definitelyarealemail.ca	2025-03-01	MTL Headquarter	opposite
Joe	Alan	905-234-5684	email212@definitelyarealemail.ca	2025-02-01	MTL Headquarter	setter
Liam	Hernandez	819-234-5678	email9@definitelyarealemail.ca	2025-04-01	Ottawa Sports Complex	No role

5.1.19. You should show the trigger(s) used in your system. Explain such trigger(s) and their benefits

5.1.19.1. Enforcing Age 11–18 on Insert

```
DELIMITER $$

CREATE TRIGGER check_club_member_age
BEFORE INSERT ON ClubMember
FOR EACH ROW
BEGIN
        DECLARE member_age INT;
        SET member_age = TIMESTAMPDIFF(YEAR, NEW.birth_date,
CURDATE());

        -- Block any member under 11
        IF (member_age < 11) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember must be at least 11 years
old.';
```

```
        -- If older than 18, they must be deactivated
        elseif (member_age > 18) THEN
        IF (NEW.active_status = TRUE) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember older than 18 must be
deactivated.';
        END IF;
        END IF;

        -- If 11 <= member_age <= 18, do nothing (allowed)
END$$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER check_club_member_age_update
BEFORE UPDATE ON ClubMember
FOR EACH ROW
BEGIN
        DECLARE member_age INT;
        SET member_age = TIMESTAMPDIFF(YEAR, NEW.birth_date,
CURDATE());

        -- Block any member under 11
        IF (member_age < 11) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember must be at least 11 years
old.';

        -- If older than 18, they must be deactivated
        elseif (member_age > 18) THEN
        IF (NEW.active_status = TRUE) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ClubMember older than 18 must be
deactivated.';
        END IF;
        END IF;

        -- If 11 <= member_age <= 18, do nothing (allowed)
END$$

DELIMITER ;
```

5.1.19.2. Moving Between Locations

```
DELIMITER $$

CREATE TRIGGER check_one_location_at_a_time
BEFORE INSERT ON ClubMember_Location
FOR EACH ROW
```

```
BEGIN
	-- Check if there's already a row for the same
club_member_number with end_date IS NULL
	IF NEW.end_date IS NULL AND EXISTS (
	SELECT 1
	FROM ClubMember_Location
	WHERE club_member_number = NEW.club_member_number
	AND end_date IS NULL
	) THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Cannot assign a new location without
ending the old one first.';
	END IF;
END$$

DELIMITER ;
```

5.1.19.3. Validate player assignment to a session (team_id, session_id, gender, 3 hour, active status, location all must be valid)

```
DELIMITER $$

CREATE TRIGGER check_session_player
BEFORE INSERT ON Session_Player
FOR EACH ROW
BEGIN
	-- All variable declarations first:
	DECLARE v_team_loc		INT;
	DECLARE v_team_gender		ENUM('Boys','Girls');
	DECLARE v_member_gender	ENUM('Male','Female');
	DECLARE v_active	BOOLEAN DEFAULT FALSE;
	DECLARE v_sess_date		DATE;
	DECLARE v_sess_time		TIME;


	-- (1) Ensure the chosen team_id is actually in this session

	IF NOT EXISTS (
	SELECT 1
	FROM Session s
	WHERE s.session_id = NEW.session_id
	AND (s.team1_id = NEW.team_id OR s.team2_id = NEW.team_id)
	) THEN
	SIGNAL SQLSTATE '45000'
	SET MESSAGE_TEXT = 'Invalid team_id for this session; must
match team1_id or team2_id.';
	END IF;
```

```
        -- (2) All players in the same team must match location

        -- Fetch the Team's location
        SELECT t.location_id
        INTO v_team_loc
        FROM Team t
        WHERE t.team_id = NEW.team_id;

        -- Check if the member is currently assigned to that location
        IF NOT EXISTS (
        SELECT 1
        FROM ClubMember_Location cml
        WHERE cml.club_member_number = NEW.club_member_number
        AND cml.location_id     = v_team_loc
        AND cml.end_date IS NULL
        ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Member is not assigned to the same
location as the team.';
        END IF;


        -- (3) Teams must not mix genders

        SELECT t.gender
        INTO v_team_gender
        FROM Team t
        WHERE t.team_id = NEW.team_id;

        SELECT c.gender
        INTO v_member_gender
        FROM ClubMember c
        WHERE c.club_member_number = NEW.club_member_number;

        IF (v_team_gender = 'Boys'  AND v_member_gender <> 'Male')
        OR (v_team_gender = 'Girls' AND v_member_gender <> 'Female')
THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Gender mismatch for this team.';
        END IF;


        -- (4) Prevent inactive members from participating

        SELECT c.active_status
        INTO v_active
        FROM ClubMember c
        WHERE c.club_member_number = NEW.club_member_number;

        IF v_active = FALSE THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot assign an inactive club member
to a session.';
        END IF;


        -- (5) Enforce 3-hour gap if a player is in two sessions the same day

        SELECT s.session_date, s.session_time
        INTO v_sess_date, v_sess_time
        FROM Session s
        WHERE s.session_id = NEW.session_id;

        IF EXISTS (
        SELECT 1
        FROM Session_Player sp
        JOIN Session s2
        ON s2.session_id = sp.session_id
        WHERE sp.club_member_number = NEW.club_member_number
        AND s2.session_date = v_sess_date
        AND ABS(TIMESTAMPDIFF(HOUR, s2.session_time,
v_sess_time)) < 3
        ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot assign a member to two sessions
within 3 hours on the same day.';
        END IF;
END$$

DELIMITER ;
```

5.1.19.4. The captain assigned to the team is a Personnel who actually has the role of captain at that location

```
DELIMITER $$

CREATE TRIGGER check_team_captain
BEFORE INSERT ON Team
FOR EACH ROW
BEGIN
        -- Declare variables at the top
        DECLARE cnt INT DEFAULT 0;

        -- Now the logic
        IF NEW.captain_pid IS NOT NULL THEN

        SELECT COUNT(*)
        INTO cnt
        FROM Personnel_Location_Role plr
        WHERE plr.personnel_id = NEW.captain_pid
```

```
        AND plr.location_id  = NEW.location_id
        AND plr.role     = 'captain'
        AND (plr.end_date IS NULL OR plr.end_date > NOW());

        IF cnt = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The assigned captain does not have the
Captain role at this location.';
        END IF;

        END IF;
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER check_team_captain_update
BEFORE UPDATE ON Team
FOR EACH ROW
BEGIN
        DECLARE cnt INT DEFAULT 0;
        IF NEW.captain_pid IS NOT NULL THEN

        SELECT COUNT(*)
        INTO cnt
        FROM Personnel_Location_Role plr
        WHERE plr.personnel_id = NEW.captain_pid
        AND plr.location_id  = NEW.location_id
        AND plr.role     = 'captain'
        AND (plr.end_date IS NULL OR plr.end_date > NOW());

        IF cnt = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The assigned captain does not have the
Captain role at this location.';
        END IF;
        END IF;
END$$

DELIMITER ;
```

5.1.19.5. Ensure SIN and MCN are unique for ClubMember

```
DELIMITER $$

CREATE TRIGGER check_sin_mcn_before_insert_club_member
BEFORE INSERT ON ClubMember
FOR EACH ROW
BEGIN
```

```
  -- Check if the SIN number exists in Personnel or FamilyMember tables
  IF EXISTS (SELECT 1 FROM `Personnel` WHERE `sin_number` =
NEW.`sin_number`) OR
     EXISTS (SELECT 1 FROM `FamilyMember` WHERE `sin_number` =
NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number
already exists in Personnel or FamilyMember';
  END IF;

  -- Check if the Medicare number exists in Personnel or FamilyMember
tables
  IF EXISTS (SELECT 1 FROM `Personnel` WHERE `medicare_number`
= NEW.`medicare_number`) OR
     EXISTS (SELECT 1 FROM `FamilyMember` WHERE
`medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare
number already exists in Personnel or FamilyMember';
  END IF;

END$$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER check_sin_mcn_before_update_club_member
BEFORE UPDATE ON ClubMember
FOR EACH ROW
BEGIN
  -- Check if the SIN number exists in Personnel or FamilyMember tables
(for updated value)
  IF EXISTS (SELECT 1 FROM `Personnel` WHERE `sin_number` =
NEW.`sin_number`) OR
     EXISTS (SELECT 1 FROM `FamilyMember` WHERE `sin_number` =
NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number
already exists in Personnel or FamilyMember';
  END IF;
  -- Check if the Medicare number exists in Personnel or FamilyMember
tables (for updated value)
  IF EXISTS (SELECT 1 FROM `Personnel` WHERE `medicare_number`
= NEW.`medicare_number`) OR
     EXISTS (SELECT 1 FROM `FamilyMember` WHERE
`medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare
number already exists in Personnel or FamilyMember';
  END IF;
END$$
DELIMITER ;
```

5.1.19.6. Validate the session location and coach

```
DELIMITER $$

CREATE TRIGGER check_location_and_coach_before_insert
BEFORE INSERT ON `Session`
FOR EACH ROW
BEGIN
        DECLARE role_check INT;

   DECLARE team1_location_id INT;
   DECLARE team2_location_id INT;

   -- Get the location_id of team1
   SELECT location_id INTO team1_location_id
   FROM `Team`
   WHERE `team_id` = NEW.team1_id;

   -- Get the location_id of team2
   SELECT location_id INTO team2_location_id
   FROM `Team`
   WHERE `team_id` = NEW.team2_id;

   -- Check if the session's location matches either team's location
   IF NEW.location_id != team1_location_id AND NEW.location_id
!= team2_location_id THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'The location_id must match either
team1 or team2"s location_id';
         ELSE
                SELECT COUNT(*)
                INTO role_check
                FROM `Personnel_Location_Role`
                WHERE `personnel_id` = NEW.coach_pid
                  AND `role` = 'coach'
                  AND `location_id` = NEW.location_id
                  AND `end_date` IS NULL;

                -- If no valid entry exists, raise an error
                IF role_check = 0 THEN
                        SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'The coach_pid
must have an active "coach" role at the specified location with a
NULL end_date';
                END IF;
   END IF;
```

```
END$$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER check_location_and_coach_before_update
BEFORE UPDATE ON `Session`
FOR EACH ROW
BEGIN
        DECLARE role_check INT;

   DECLARE team1_location_id INT;
   DECLARE team2_location_id INT;

   -- Get the location_id of team1
   SELECT location_id INTO team1_location_id
   FROM `Team`
   WHERE `team_id` = NEW.team1_id;

   -- Get the location_id of team2
   SELECT location_id INTO team2_location_id
   FROM `Team`
   WHERE `team_id` = NEW.team2_id;

   -- Check if the session's location matches either team's location
   IF NEW.location_id != team1_location_id AND NEW.location_id
!= team2_location_id THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'The location_id must match either
team1 or team2"s location_id';
         ELSE
                SELECT COUNT(*)
                INTO role_check
                FROM `Personnel_Location_Role`
                WHERE `personnel_id` = NEW.coach_pid
                  AND `role` = 'coach'
                  AND `location_id` = NEW.location_id
                  AND `end_date` IS NULL;

                -- If no valid entry exists, raise an error
                IF role_check = 0 THEN
                        SIGNAL SQLSTATE '45000'
```

```
                SET MESSAGE_TEXT = 'The coach_pid must have an active "coach" role at the specified location with a NULL end_date';
            END IF;
    END IF;
END$$

DELIMITER ;
```

5.1.19.7. Ensure SIN and MCN are unique for FamilyMember (or exact match from Personnel)

```
DELIMITER $$

CREATE TRIGGER before_insert_family_member
BEFORE INSERT ON FamilyMember
FOR EACH ROW
BEGIN
  -- Check if the SIN number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE `sin_number` = NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number already exists in ClubMember table';
  END IF;

  -- Check if the Medicare number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE `medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare number already exists in ClubMember table';
  END IF;

IF EXISTS (
    SELECT 1
    FROM `Personnel`
    WHERE `sin_number` = NEW.`sin_number`
       OR `medicare_number` = NEW.`medicare_number`
) THEN
    -- There is a match for either sin_number or medicare_number
    IF NOT EXISTS (
        SELECT 1
        FROM `Personnel`
        WHERE `sin_number` = NEW.`sin_number`
          AND `medicare_number` = NEW.`medicare_number`
          AND `first_name` = NEW.`first_name`
          AND `last_name` = NEW.`last_name`
          AND `birth_date` = NEW.`birth_date`
    ) THEN
        -- If the match is not exact, prevent the update
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sin number
and Medicare number must match Personnel exactly (first_name,
last_name, birth_date), or be completely unique';
        -- If the match is exact, allow the update to proceed
        -- No action required here, the update will proceed
    END IF;
    -- No match exists for either sin_number or medicare_number, so allow
the update
    -- No action required here, the update will proceed
END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER before_update_family_member
BEFORE UPDATE ON FamilyMember
FOR EACH ROW
BEGIN
  -- Check if the SIN number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE `sin_number` =
NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number
already exists in ClubMember table';
  END IF;

  -- Check if the Medicare number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE
`medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare
number already exists in ClubMember table';
  END IF;

IF EXISTS (
    SELECT 1
    FROM `Personnel`
    WHERE `sin_number` = NEW.`sin_number`
       OR `medicare_number` = NEW.`medicare_number`
) THEN
    -- There is a match for either sin_number or medicare_number
    IF NOT EXISTS (
        SELECT 1
        FROM `Personnel`
        WHERE `sin_number` = NEW.`sin_number`
          AND `medicare_number` = NEW.`medicare_number`
          AND `first_name` = NEW.`first_name`
          AND `last_name` = NEW.`last_name`
```

```
        AND `birth_date` = NEW.`birth_date`
    ) THEN
                    -- If the match is not exact, prevent the update
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sin number
and Medicare number must match Personnel exactly (first_name,
last_name, birth_date), or be completely unique';
        -- If the match is exact, allow the update to proceed
        -- No action required here, the update will proceed
    END IF;
    -- No match exists for either sin_number or medicare_number, so allow
the update
    -- No action required here, the update will proceed
END IF;

END$$

DELIMITER ;
```

5.1.19.8. Ensure SIN and MCN are unique for Personnel (or exact match from FamilyMember)

```
DELIMITER $$

CREATE TRIGGER before_insert_personnel
BEFORE INSERT ON Personnel
FOR EACH ROW
BEGIN
  -- Check if the SIN number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE `sin_number` =
NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number
already exists in ClubMember table';
  END IF;

  -- Check if the Medicare number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE
`medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare
number already exists in ClubMember table';
  END IF;

IF EXISTS (
    SELECT 1
    FROM `FamilyMember`
    WHERE `sin_number` = NEW.`sin_number`
      OR `medicare_number` = NEW.`medicare_number`
) THEN
    -- There is a match for either sin_number or medicare_number
    IF NOT EXISTS (
        SELECT 1
```

```
        FROM `FamilyMember`
        WHERE `sin_number` = NEW.`sin_number`
          AND `medicare_number` = NEW.`medicare_number`
          AND `first_name` = NEW.`first_name`
          AND `last_name` = NEW.`last_name`
          AND `birth_date` = NEW.`birth_date`
    ) THEN
                    -- If the match is not exact, prevent the update
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sin number
and Medicare number must match FamilyMember exactly (first_name,
last_name, birth_date), or be completely unique';
        -- If the match is exact, allow the update to proceed
        -- No action required here, the update will proceed
    END IF;
    -- No match exists for either sin_number or medicare_number, so allow
the update
    -- No action required here, the update will proceed
END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER before_update_personnel
BEFORE UPDATE ON Personnel
FOR EACH ROW
BEGIN
  -- Check if the SIN number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE `sin_number` =
NEW.`sin_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'SIN number
already exists in ClubMember table';
  END IF;

  -- Check if the Medicare number exists in ClubMember table
  IF EXISTS (SELECT 1 FROM `ClubMember` WHERE
`medicare_number` = NEW.`medicare_number`) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Medicare
number already exists in ClubMember table';
  END IF;

IF EXISTS (
    SELECT 1
    FROM `FamilyMember`
    WHERE `sin_number` = NEW.`sin_number`
       OR `medicare_number` = NEW.`medicare_number`
) THEN
```

```
    -- There is a match for either sin_number or medicare_number
    IF NOT EXISTS (
        SELECT 1
        FROM `FamilyMember`
        WHERE `sin_number` = NEW.`sin_number`
          AND `medicare_number` = NEW.`medicare_number`
          AND `first_name` = NEW.`first_name`
          AND `last_name` = NEW.`last_name`
          AND `birth_date` = NEW.`birth_date`
    ) THEN
                    -- If the match is not exact, prevent the update
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Sin number
and Medicare number must match FamilyMember exactly (first_name,
last_name, birth_date), or be completely unique';
        -- If the match is exact, allow the update to proceed
        -- No action required here, the update will proceed
    END IF;
    -- No match exists for either sin_number or medicare_number, so allow
the update
    -- No action required here, the update will proceed
END IF;

END$$

DELIMITER ;
```

5.1.19.9. Ensure Family Member cannot be 2x at the same place at the same time (previous entry must have an end date)

```
DELIMITER $$

CREATE TRIGGER check_duplicate_family_member
BEFORE INSERT ON FamilyMember_Location
FOR EACH ROW
BEGIN
IF NEW.end_date IS NULL AND EXISTS (
    SELECT 1
    FROM FamilyMember_Location fml
    WHERE fml.family_member_id = NEW.family_member_id
      AND fml.location_id = NEW.location_id
      AND fml.end_date IS NULL
)
THEN
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot insert
another row with end_date IS NULL.';
END IF;

END$$

DELIMITER ;
```

5.1.19.10. Ensure Personnel cannot work 2x at the same place at the same time (previous entry must have an end date)

```
DELIMITER $$

CREATE TRIGGER check_duplicate_personnel
BEFORE INSERT ON Personnel_Location_Role
FOR EACH ROW
BEGIN
IF NEW.end_date IS NULL AND EXISTS (
   SELECT 1
   FROM Personnel_Location_Role prs
   WHERE prs.personnel_id = NEW.personnel_id
     AND prs.location_id = NEW.location_id
     AND prs.end_date IS NULL
)
THEN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot insert
another row with end_date IS NULL.';
END IF;
END$$
DELIMITER ;
```

5.1.19.11. Check if payments made for current year add up to $100 or more, and modify activation status accordingly

```
DELIMITER $$

CREATE TRIGGER after_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
   DECLARE total_payment_amount DECIMAL(10, 2);
   DECLARE current_year INT;

   -- Get the current year
   SET current_year = YEAR(CURDATE());

   -- Check if the membership_year is the current year
   IF NEW.membership_year = current_year THEN
      -- Sum all payments for the same club_member_number and
membership_year
      SELECT SUM(payment_amount)
      INTO total_payment_amount
      FROM Payment
      WHERE club_member_number = NEW.club_member_number
        AND membership_year = NEW.membership_year;

      -- If the total payments are >= 100, set active_status to 1 and
deactivation_date to NULL
```

```
        IF total_payment_amount >= 100 THEN
            UPDATE ClubMember
            SET active_status = 1,
                deactivation_date = NULL
            WHERE club_member_number = NEW.club_member_number;
        ELSE
            -- If total payments are less than 100, set active_status to 0 and
deactivation_date to current date
            UPDATE ClubMember
            SET active_status = 0,
                deactivation_date = CURDATE()
            WHERE club_member_number = NEW.club_member_number;
        END IF;
    END IF;
END $$

DELIMITER ;
```

5.1.19.12. Maximum of 4 payments

```
DELIMITER $$

CREATE TRIGGER check_max_4_payments
BEFORE INSERT ON Payment
FOR EACH ROW
BEGIN
    -- Check if there are already 4 payments for the same
club_member_number and membership_year
    DECLARE payment_count INT;

    SELECT COUNT(*) INTO payment_count
    FROM Payment
    WHERE club_member_number = NEW.club_member_number
      AND membership_year = NEW.membership_year;

    IF payment_count >= 4 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot make more than four payments for
the same membership year.';
    END IF;

END$$

DELIMITER ;
```

5.1.19.13. Ensure both teams in a session have the same gender

```
DELIMITER $$

CREATE TRIGGER check_team_gender_insert
BEFORE INSERT ON Session
```

```
FOR EACH ROW
BEGIN
   DECLARE gender1 VARCHAR(10);
   DECLARE gender2 VARCHAR(10);

   -- Get the gender of team_id1
   SELECT gender INTO gender1
   FROM Team
   WHERE team_id = NEW.team1_id;

   -- Get the gender of team_id2
   SELECT gender INTO gender2
   FROM Team
   WHERE team_id = NEW.team2_id;

   -- Check if both teams have the same gender
   IF gender1 != gender2 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The genders
of team_id1 and team_id2 must be the same.';
   END IF;
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER check_team_gender_update
BEFORE UPDATE ON Session
FOR EACH ROW
BEGIN
   DECLARE gender1 VARCHAR(10);
   DECLARE gender2 VARCHAR(10);

   -- Get the gender of team_id1
   SELECT gender INTO gender1
   FROM Team
   WHERE team_id = NEW.team1_id;

   -- Get the gender of team_id2
   SELECT gender INTO gender2
   FROM Team
   WHERE team_id = NEW.team2_id;

   -- Check if both teams have the same gender
   IF gender1 != gender2 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The genders
of team_id1 and team_id2 must be the same.';
   END IF;
END$$
```

```
DELIMITER ;
```

5.1.19.14. Enforce max capacity

```
DELIMITER $$

CREATE TRIGGER check_location_capacity_before_insert
BEFORE INSERT ON ClubMember_Location
FOR EACH ROW
BEGIN
   DECLARE active_members_count INT;
   DECLARE location_max_capacity INT;

   -- Get the number of active members in the given location
   SELECT COUNT(*) INTO active_members_count
   FROM ClubMember_Location cml
   JOIN ClubMember cm ON cml.club_member_number =
cm.club_member_number
   WHERE cml.location_id = NEW.location_id
   AND cm.active_status = 1;

   -- Get the maximum capacity for the location
   SELECT max_capacity INTO location_max_capacity
   FROM Location
   WHERE location_id = NEW.location_id;

   -- Check if the number of active members exceeds the maximum capacity
   IF active_members_count >= location_max_capacity THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Location has reached its maximum capacity
for active club members.';
   END IF;
END$$

DELIMITER ;


DELIMITER $$

CREATE TRIGGER check_location_capacity_before_update
BEFORE UPDATE ON ClubMember_Location
FOR EACH ROW
BEGIN
   DECLARE active_members_count INT;
   DECLARE location_max_capacity INT;

   -- Get the number of active members in the given location
   SELECT COUNT(*) INTO active_members_count
   FROM ClubMember_Location cml
```

```
    JOIN ClubMember cm ON cml.club_member_number =
cm.club_member_number
    WHERE cml.location_id = NEW.location_id
    AND cm.active_status = 1;

    -- Get the maximum capacity for the location
    SELECT max_capacity INTO location_max_capacity
    FROM Location
    WHERE location_id = NEW.location_id;

    -- Check if the number of active members exceeds the maximum capacity
    IF active_members_count >= location_max_capacity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Location has reached its maximum capacity
for active club members.';
    END IF;
END$$

DELIMITER ;
```

5.1.20. 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)

1. Testing “Enforcing Age 11–18 on Insert” Trigger for ClubMember

(a) Attempt to insert a ClubMember younger than 11

Expected: Rejected with error "ClubMember must be at least 11 years old."

```
INSERT INTO ClubMember (first_name, last_name, birth_date, gender, sin_number,
medicare_number, active_status, email_address)

VALUES ('Tommy', 'Young', '2015-01-01', 'Male', 'SIN_TEST1', 'MED_TEST1', 1,
'tommy.young@example.com');
```

```
20:12:19    INSERT INTO ClubMember (first_name, last_name, birth_date, gender,
sin_number, medicare_number, active_status, email_address) VALUES ('Tommy', 'Young',
'2015-01-01', 'Male', 'SIN_TEST1', 'MED_TEST1', 1, 'tommy.young@example.com')
Error Code: 1644. ClubMember must be at least 11 years old.    0.016 sec
```

(b) Attempt to insert a ClubMember older than 18 with active_status = TRUE

Expected: Rejected with error "ClubMember older than 18 must be deactivated."

INSERT INTO ClubMember (first_name, last_name, birth_date, gender, sin_number, medicare_number, active_status, email_address)

VALUES ('Bob', 'Adult', '2000-03-10', 'Male', 'SIN_TEST3', 'MED_TEST3', 1, 'bob.adult@example.com');

20:22:14 INSERT INTO ClubMember (first_name, last_name, birth_date, gender, sin_number, medicare_number, active_status, email_address) VALUES ('Bob', 'Adult', '2000-03-10', 'Male', 'SIN_TEST3', 'MED_TEST3', 1, 'bob.adult@example.com') Error Code: 1644. ClubMember older than 18 must be deactivated. 0.016 sec

2. Testing "Moving Between Locations" Trigger for ClubMember_Location

(a) Attempt to insert a second active assignment for the same club_member (end_date IS NULL)

> **Expected:** Rejected with error "Cannot assign a new location without ending the old one first."

INSERT INTO ClubMember_Location (club_member_number, location_id, start_date, end_date)

VALUES (114, 2, '2025-05-01', NULL);

20:29:45 INSERT INTO ClubMember_Location (club_member_number, location_id, start_date, end_date) VALUES (114, 2, '2025-05-01', NULL) Error Code: 1644. Cannot assign a new location without ending the old one first. 0.015 sec

3. Testing "Validate Player Assignment to a Session" Trigger (Session_Player)

(a) Invalid Team ID for Session:

Attempt to assign ClubMember 2014 to a team (say, team_id = 12) that is not part of Session 53.

> **Expected:** Rejected with error "Invalid team_id for this session; must match team1_id or team2_id."

INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position)

VALUES (53, 2014, 12, 'setter');

20:47:17 INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position) VALUES (53, 2014, 12, 'setter') Error Code: 1644. Invalid team_id for this session; must match team1_id or team2_id. 0.000 sec

(b) Location Mismatch:

Assume ClubMember 2014 is not assigned to location 2 (Team's location).

> **Expected:** Rejected with error "Member is not assigned to the same location as the team."

INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position)

VALUES (53, 2014, 20, 'setter');

20:46:28 INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position) VALUES (53, 2014, 20, 'setter') Error Code: 1644. Member is not assigned to the same location as the team. 0.000 sec

(c) Gender Mismatch:

Attempt to assign ClubMember 2014 (Girl) to Team 17 (Boys team).

> **Expected:** Rejected with error "Gender mismatch for this team."

INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position)

VALUES (30, 2014, 17, 'setter');

20:50:53 INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position) VALUES (30, 2014, 17, 'setter') Error Code: 1644. Gender mismatch for this team. 0.015 sec

(d) Inactive Member:

Assume ClubMember 2014 has active_status = FALSE.

> **Expected:** Rejected with error "Cannot assign an inactive club member to a session."

INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position)

VALUES (53, 2014, 23, 'setter');

20:52:26 INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position) VALUES (53, 2014, 23, 'setter') Error Code: 1644. Cannot assign an inactive club member to a session. 0.016 sec

(e) 3-Hour Gap Violation:

Assume ClubMember 2014 is already assigned to Session 10 at 10:00:00 , then attempt to insert another assignment on the same day at 11:30:00 (less than 3 hours difference).

Expected: Rejected with error "Cannot assign a member to two sessions within 3 hours on the same day."

INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position)

VALUES (54, 2014, 23, 'setter');

20:55:23 INSERT INTO Session_Player (session_id, club_member_number, team_id, player_position) VALUES (54, 2014, 23, 'setter') Error Code: 1644. Cannot assign a member to two sessions within 3 hours on the same day. 0.000 sec

4. Testing "The Captain Assigned to the Team is a Personnel with the Captain Role" Trigger

(a) Invalid Captain Assignment on INSERT:

Expected: Rejected with error "The assigned captain does not have the Captain role at this location."

20:36:32 INSERT INTO `iqc353_4`.`Team` (`team_id`, `team_name`, `location_id`, `gender`, `captain_pid`) VALUES ('23', 'TEST TEAM G', '2', 'Girls', '60') 1644: The assigned captain does not have the Captain role at this location.

5. Testing "Ensure SIN and MCN are Unique for ClubMember" Trigger

(a) Invalid Insert:

Assume a ClubMember with sin_number = 'SIN000151' already exists in either Personnel or FamilyMember.

Expected: Rejected with error "SIN number already exists in Personnel or FamilyMember"

INSERT INTO ClubMember (first_name, last_name, birth_date, gender, sin_number, medicare_number, active_status, email_address)

VALUES ('Duplicate', 'Test', '2009-01-01', 'Male', 'DUP123', 'UNIQUE123', 1, 'dup.test@example.com');

20:59:16 INSERT INTO ClubMember (first_name, last_name, birth_date, gender, sin_number, medicare_number, active_status, email_address) VALUES ('Duplicate', 'Test', '2009-01-01', 'Male', 'SIN000151', 'UNIQUE123', 1, 'dup.test@example.com') Error Code: 1644. SIN number already exists in Personnel or FamilyMember 0.000 sec

6. Testing "Ensure FamilyMember Cannot be 2x at the Same Place at the Same Time" Trigger

(a) Invalid Insert:

Attempt to insert another FamilyMember_Location row for the same family_member_id = 50 and location_id = 2 with end_date = NULL.

> **Expected:** Rejected with error "Cannot insert another row with end_date IS NULL."

INSERT INTO FamilyMember_Location (family_member_id, location_id, start_date, end_date)

VALUES (50, 2, '2025-01-03', NULL);

21:01:47 INSERT INTO FamilyMember_Location (family_member_id, location_id, start_date, end_date) VALUES (50, 2, '2025-01-03', NULL) Error Code: 1644. Cannot insert another row with end_date IS NULL. 0.016 sec

7. Testing "Ensure Personnel Cannot Work 2x at the Same Place at the Same Time" Trigger

(a) Invalid Insert:

Attempt to insert another row for the same personnel_id = 20 and location_id = 1 with end_date IS NULL.

> **Expected:** Rejected with error "Cannot insert another row with end_date IS NULL."

INSERT INTO Personnel_Location_Role (personnel_id, location_id, role, start_date, end_date)

VALUES (70, 5, 'coach', '2025-06-01', NULL);

21:03:22 INSERT INTO Personnel_Location_Role (personnel_id, location_id, role, start_date, end_date) VALUES (70, 5, 'coach', '2025-06-01', NULL) Error Code: 1644. Cannot insert another row with end_date IS NULL. 0.000 sec

8. Testing "Check if Payments Made for Current Year Add Up to $100" Trigger

Assume the current year is 2025 and ClubMember 2014 is in use.

(a) Partial Payment (below $100):

> **Expected:** After the insert, ClubMember 2014's active_status becomes 0 (inactive) and deactivation_date is set to CURDATE().

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2014, '2025-01-15', 50.00, 'Cash', 2025);

'2014', 'Alice', 'Adolescent', '2008-06-15', 'Female', NULL, NULL, 'SIN_TEST2', 'MED_TEST2', NULL, NULL, NULL, NULL, NULL, **'0', '2025-04-04'**, 'alice.adolescent@example.com'

(b) Additional Payment (total reaches or exceeds $100):

> **Expected:** After this insert, ClubMember 2014's active_status becomes 1 (active) and deactivation_date is cleared (set to NULL).

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2014, '2025-02-15', 60.00, 'Credit Card', 2025);

'2014', 'Alice', 'Adolescent', '2008-06-15', 'Female', NULL, NULL, 'SIN_TEST2', 'MED_TEST2', NULL, NULL, NULL, NULL, NULL, **'1', NULL**, 'alice.adolescent@example.com'

9. Testing "Maximum of 4 Payments per Membership Year" Trigger

Assume ClubMember 2015 is in use for membership year 2025.

(a) Insert Four Payments (Valid):

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2015, '2025-01-10', 25.00, 'Cash', 2025);

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2015, '2025-01-20', 25.00, 'Cash', 2025);

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2015, '2025-02-10', 25.00, 'Cash', 2025);

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2015, '2025-02-20', 25.00, 'Cash', 2025);

21:07:07 INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year) VALUES (2015, '2025-02-20', 25.00, 'Cash', 2025) 1 row(s) affected 0.000 sec

(b) Attempt to Insert a 5th Payment:

> **Expected:** Rejected with error "Cannot make more than four payments for the same membership year."

INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year)

VALUES (2015, '2025-03-01', 25.00, 'Cash', 2025);

21:07:55 INSERT INTO Payment (club_member_number, payment_date, payment_amount, method_of_payment, membership_year) VALUES (2015, '2025-03-01', 25.00, 'Cash', 2025) Error Code: 1644. Cannot make more than four payments for the same membership year. 0.016 sec

10. Testing "Ensure Both Teams in a Session Have the Same Gender" Trigger

(a) Attempt to Insert a Session with Different Team Genders:

> **Expected:** Rejected with error "The genders of team_id1 and team_id2 must be the same."

INSERT INTO Session (session_date, session_time, session_type, address, location_id, team1_id, team2_id, score_team1, score_team2, coach_pid)

VALUES ('52', '2025-09-11', '08:46:00', 'game', '1', '22', '23', '40');

20:38:35 INSERT INTO `iqc353_4`.`Session` (`session_id`, `session_date`, `session_time`, `session_type`, `location_id`, `team1_id`, `team2_id`, `coach_pid`) VALUES ('52', '2025-09-11', '08:46:00', 'game', '1', '22', '23', '40') 1644: The genders of team_id1 and team_id2 must be the same.

11. Testing "Enforce Maximum Capacity for Active Members in a Location" Trigger on ClubMember_Location

Assume Location 1 has a max_capacity of 10.

(a) Attempt to Insert a 101st Active Member:

> **Expected:** Rejected with error "Location has reached its maximum capacity for active club members."

INSERT INTO ClubMember_Location (club_member_number, location_id, start_date, end_date)

VALUES (2014, 1, '2025-01-01', NULL);

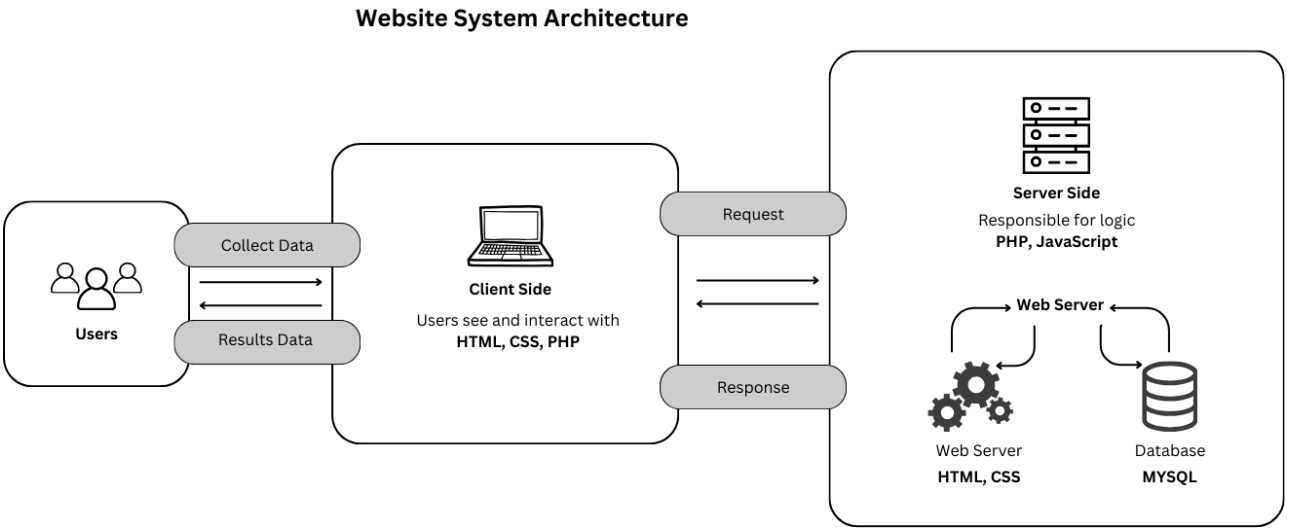
21:11:36 INSERT INTO ClubMember_Location (club_member_number, location_id, start_date, end_date) VALUES (2014, 1, '2025-01-01', NULL) Error Code: 1644. Location has reached its maximum capacity for active club members. 0.750 sec

5.1.21. 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)

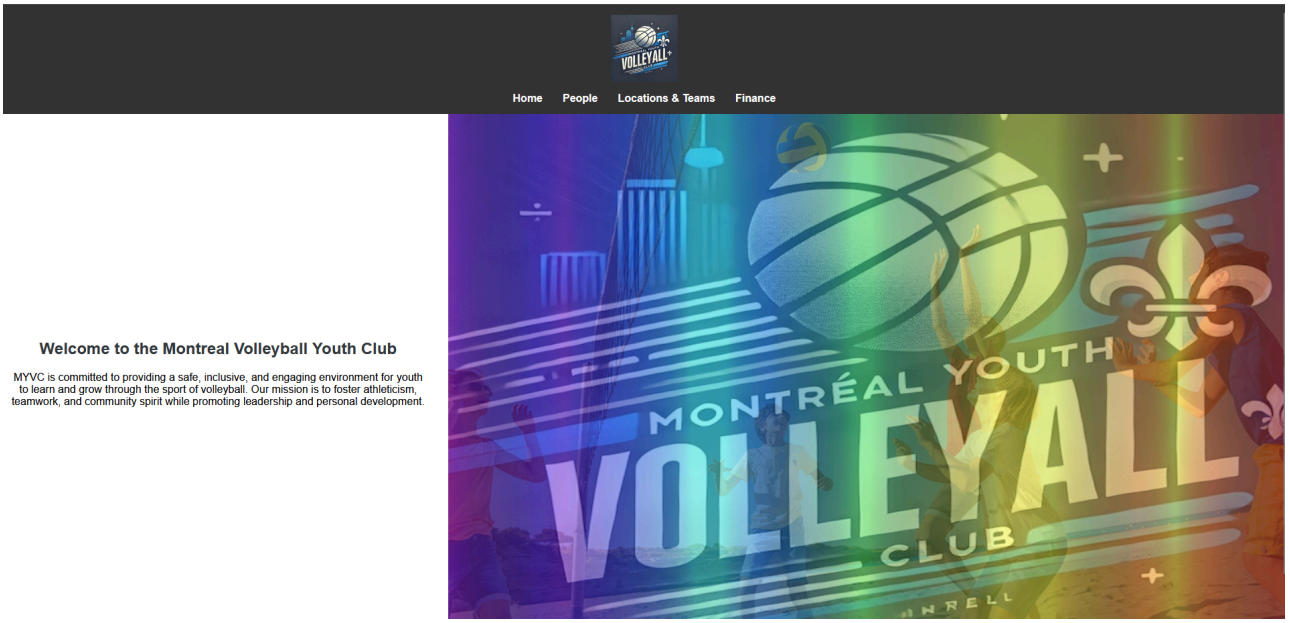
email_log_id	email_date	sender	recipient	cc_list	subject	body_preview
11	2025-02-01 00:00:00	notifications@myvc.ca	email212@definitelyarealemail.ca	email212@definitelyarealemail.ca	MYVC Account Deactivation	Dear Joe Alan, Your account has been deactivated
12	2025-04-01 00:00:00	notifications@myvc.ca	email33@definitelyarealemail.ca	email33@definitelyarealemail.ca	MYVC Account Deactivation	Dear Lucas Clark, Your account has been deactivated
13	2025-04-01 00:00:00	notifications@myvc.ca	email213@definitelyarealemail.ca	email213@definitelyarealemail.ca	MYVC Account Deactivation	Dear James Alban, Your account has been deactivated
14	2025-04-01 00:00:00	notifications@myvc.ca	email214@definitelyarealemail.ca	email214@definitelyarealemail.ca	MYVC Account Deactivation	Dear Jan Corey, Your account has been deactivated
15	2025-04-06 00:00:00	notifications@myvc.ca	email1@definitelyarealemail.ca	email1@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	John Doe, setter 06-Apr-2025
16	2025-04-06 00:00:00	notifications@myvc.ca	email5@definitelyarealemail.ca	email5@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Mason Williams, opposite 06-Apr-2025
17	2025-04-06 00:00:00	notifications@myvc.ca	email15@definitelyarealemail.ca	email15@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Alexander Jackson, libero 06-Apr-2025
18	2025-04-06 00:00:00	notifications@myvc.ca	email25@definitelyarealemail.ca	email25@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Daniel Garcia, serving_specialist 06-Apr-2025
19	2025-04-06 00:00:00	notifications@myvc.ca	email35@definitelyarealemail.ca	email35@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Daniel Perez, middle_blocker 06-Apr-2025
20	2025-04-06 00:00:00	notifications@myvc.ca	email51@definitelyarealemail.ca	email51@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Mason Perez, serving_specialist 06-Apr-2025
21	2025-04-06 00:00:00	notifications@myvc.ca	email61@definitelyarealemail.ca	email61@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Lucas Taylor, libero 06-Apr-2025
22	2025-04-06 00:00:00	notifications@myvc.ca	email66@definitelyarealemail.ca	email66@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Matthew Lopez, setter 06-Apr-2025
23	2025-04-06 00:00:00	notifications@myvc.ca	email71@definitelyarealemail.ca	email71@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Ethan Evans, defensive_specialist 06-Apr-2025
24	2025-04-06 00:00:00	notifications@myvc.ca	email91@definitelyarealemail.ca	email91@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Landon Evans, middle_blocker 06-Apr-2025
25	2025-04-06 00:00:00	notifications@myvc.ca	email95@definitelyarealemail.ca	email95@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Benjamin Taylor, defensive_specialist 06-Apr-2025
26	2025-04-06 00:00:00	notifications@myvc.ca	email206@definitelyarealemail.ca	email206@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Oliver King, outside_hitter 06-Apr-2025
27	2025-04-06 00:00:00	notifications@myvc.ca	email208@definitelyarealemail.ca	email208@definitelyarealemail.ca	Team 3 training 06-Apr-2025 10:30 AM	Noah Garcia, outside_hitter 06-Apr-2025
28	2025-04-06 00:00:00	notifications@myvc.ca	email211@definitelyarealemail.ca	email211@definitelyarealemail.ca	Team 1 training 06-Apr-2025 10:30 AM	Alan Grow, opposite 06-Apr-2025
29	2025-04-06 00:00:00	notifications@myvc.ca	email4@definitelyarealemail.ca	email4@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Sophia Brown, setter 07-Apr-2025
30	2025-04-06 00:00:00	notifications@myvc.ca	email14@definitelyarealemail.ca	email14@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Grace Thomas, opposite 07-Apr-2025
31	2025-04-06 00:00:00	notifications@myvc.ca	email24@definitelyarealemail.ca	email24@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Amelia Rodriguez, middle_blocker 07-Apr-2025
32	2025-04-06 00:00:00	notifications@myvc.ca	email38@definitelyarealemail.ca	email38@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Isabella Moore, middle_blocker 07-Apr-2025
33	2025-04-06 00:00:00	notifications@myvc.ca	email48@definitelyarealemail.ca	email48@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Harper Clark, defensive_specialist 07-Apr-2025
34	2025-04-06 00:00:00	notifications@myvc.ca	email54@definitelyarealemail.ca	email54@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Amelia Taylor, serving_specialist 07-Apr-2025
35	2025-04-06 00:00:00	notifications@myvc.ca	email58@definitelyarealemail.ca	email58@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Emma Harris, serving_specialist 07-Apr-2025
36	2025-04-06 00:00:00	notifications@myvc.ca	email67@definitelyarealemail.ca	email67@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Grace Taylor, libero 07-Apr-2025
37	2025-04-06 00:00:00	notifications@myvc.ca	email84@definitelyarealemail.ca	email84@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Zoe Jackson, defensive_specialist 07-Apr-2025
38	2025-04-06 00:00:00	notifications@myvc.ca	email88@definitelyarealemail.ca	email88@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Emily White, setter 07-Apr-2025
39	2025-04-06 00:00:00	notifications@myvc.ca	email94@definitelyarealemail.ca	email94@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Olivia Scott, middle_blocker 07-Apr-2025
40	2025-04-06 00:00:00	notifications@myvc.ca	email98@definitelyarealemail.ca	email98@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Amelia White, opposite 07-Apr-2025
41	2025-04-06 00:00:00	notifications@myvc.ca	email207@definitelyarealemail.ca	email207@definitelyarealemail.ca	Team 2 game 07-Apr-2025 02:00 PM	Ava Martinez, outside_hitter 07-Apr-2025
42	2025-04-06 00:00:00	notifications@myvc.ca	email209@definitelyarealemail.ca	email209@definitelyarealemail.ca	Team 4 game 07-Apr-2025 02:00 PM	Mia Rodriguez, outside_hitter 07-Apr-2025
43	2025-04-06 00:00:00	notifications@myvc.ca	email3@definitelyarealemail.ca	email3@definitelyarealemail.ca	Team 7 training 08-Apr-2025 09:15 AM	Oliver Johnson, setter 08-Apr-2025
44	2025-04-06 00:00:00	notifications@myvc.ca	email7@definitelyarealemail.ca	email7@definitelyarealemail.ca	Team 9 training 08-Apr-2025 09:15 AM	James Garcia, setter 08-Apr-2025
45	2025-04-06 00:00:00	notifications@myvc.ca	email13@definitelyarealemail.ca	email13@definitelyarealemail.ca	Team 7 training 08-Apr-2025 09:15 AM	Lucas Anderson, outside_hitter 08-Apr-2025

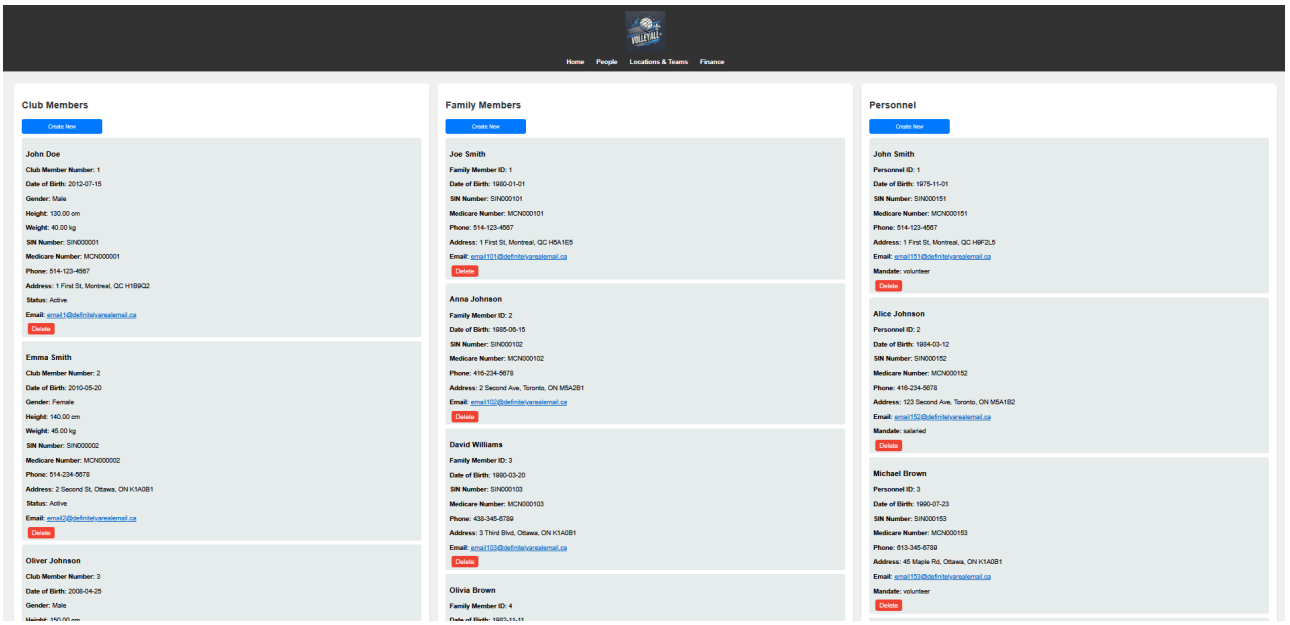
6. Web Interface Development

6.1. Website System Architecture



6.2. User Interaction Snapshots





Create Club Member

Club Member Number:

First Name:

Last Name:

Birth Date:

dd / mm / yyyy

Gender:

Male

Height:

Weight:

SIN Number:

Medicare Number:

Telephone Number:

Address:

City:

Province:

Postal Code:

Active Status:

Email:

Save User

Modify Club Member: 1

Club Member Number:

1

First Name:

John

Last Name:

Doe

Birth Date:

16 / 08 / 2012

Gender:

Male

Height:

100.00

Weight:

100.00

SIN Number:

SIN000001

Medicare Number:

MCN000001

Telephone Number:

514-123-4567

Address:

1 First St

City:

Montreal

Province:

QC

Postal Code:

H1B9Q2

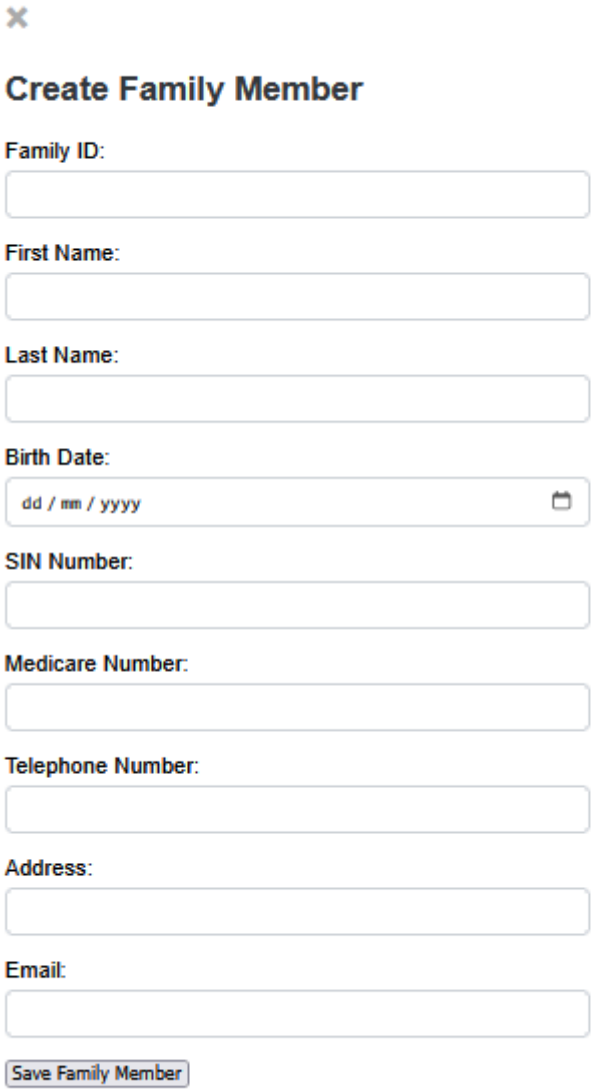
Active Status:

Active

Email:

email1@definitelyarealemail.ca

Save User


Create Family Member
Family ID:
First Name:
Last Name:
Birth Date:
dd / mm / yyyy
SIN Number:
Medicare Number:
Telephone Number:
Address:
Email:
Save Family Member

Modify Family Member: 1

Family ID:

1

First Name:

Joe

Last Name:

Smith

Birth Date:

01 / 01 / 1980

SIN Number:

SIN000101

Medicare Number:

MCN000101

Telephone Number:

514-123-4567

Address:

1 First St

Email:

email101@definitelyarealemail.ca

Save Family Member

Create Personnel Member

Personnel IDe:

First Name:

Last Name:

Birth Date:

dd / mm / yyyy

SIN Number:

Medicare Number:

Telephone Number:

Address:

City:

Province:

Postal Code:

Email:

Mandate:

Volunteer

Save Personnel Member

Modify Personnel: 1

Personnel IDe:

1

First Name:

John Test

Last Name:

Smith

Birth Date:

01 / 11 / 1975

SIN Number:

SIN000151

Medicare Number:

MCN000151

Telephone Number:

514-123-4567

Address:

1 First St

City:

Montreal

Province:

QC

Postal Code:

H9F2L5

Email:

email151@definitelyarealemail.ca

Mandate:

Volunteer

Save Personnel Member

Create Location

Location ID:

Location Type:

Head

Name:

Address:

City:

Province:

Postal Code:

Phone Number:

Web Address:

Max Capacity:

Save Location

Modify Location: 1

Location ID:

1

Location Type:

Head

Name:

MTL Headquarter

Address:

123 Main St

City:

Montreal

Province:

QC

Postal Code:

H0H0H0

Phone Number:

514-111-1111

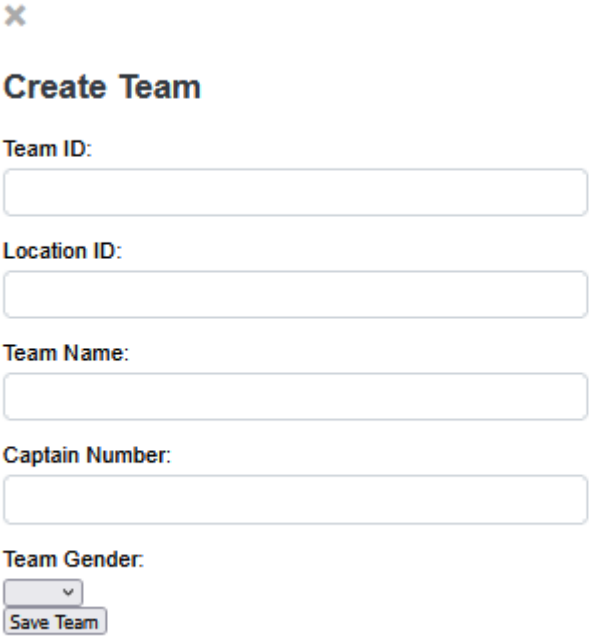
Web Address:

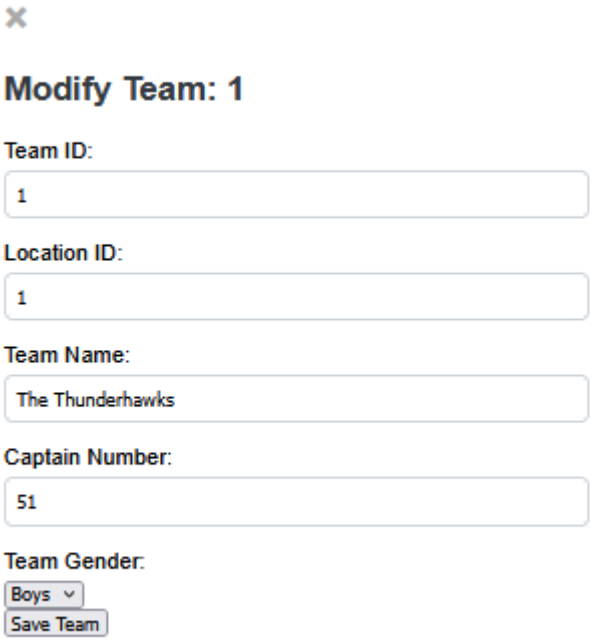
www.myvc-head.ca

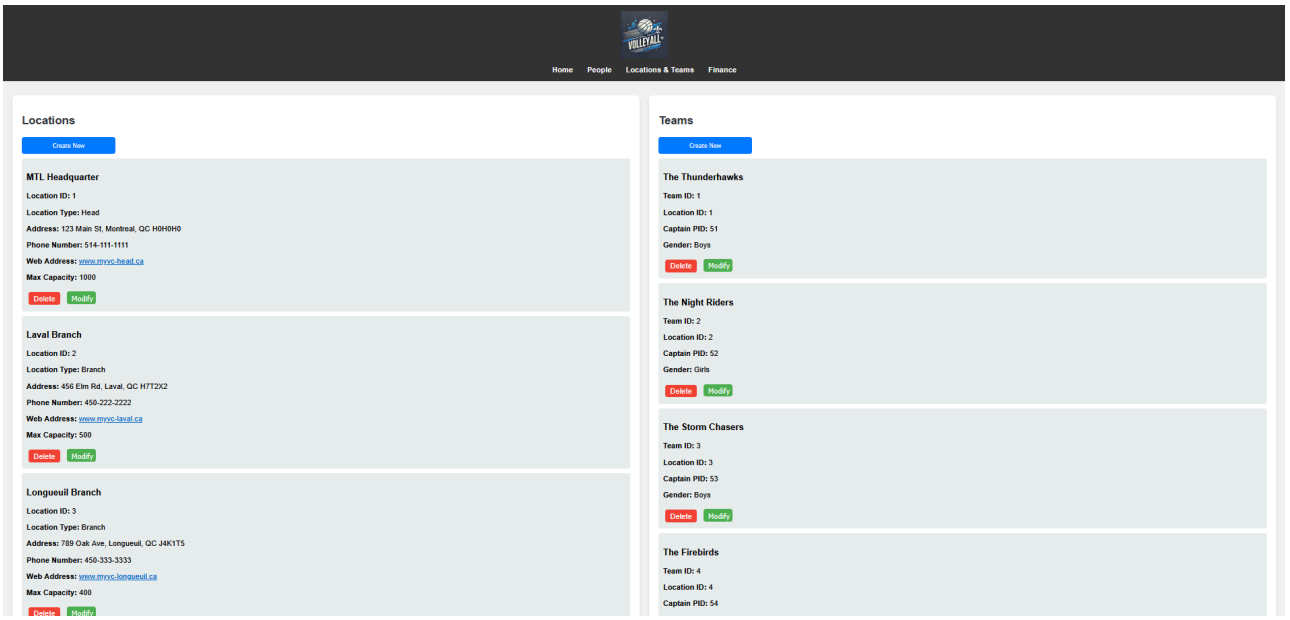
Max Capacity:

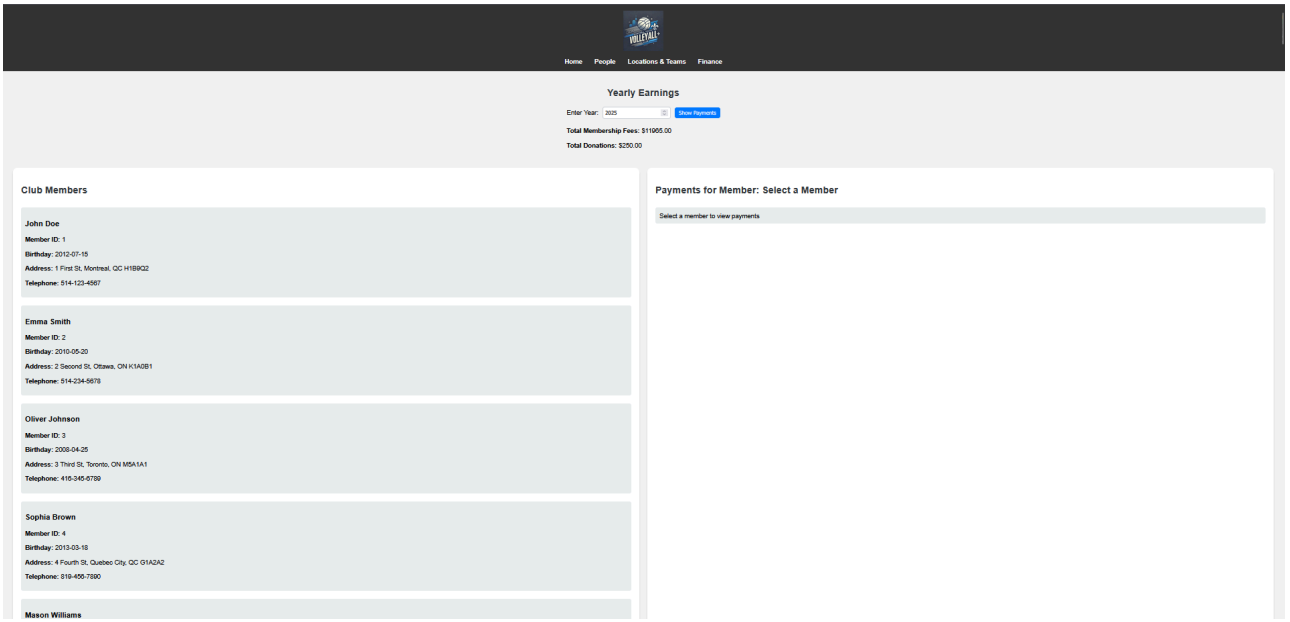
1000

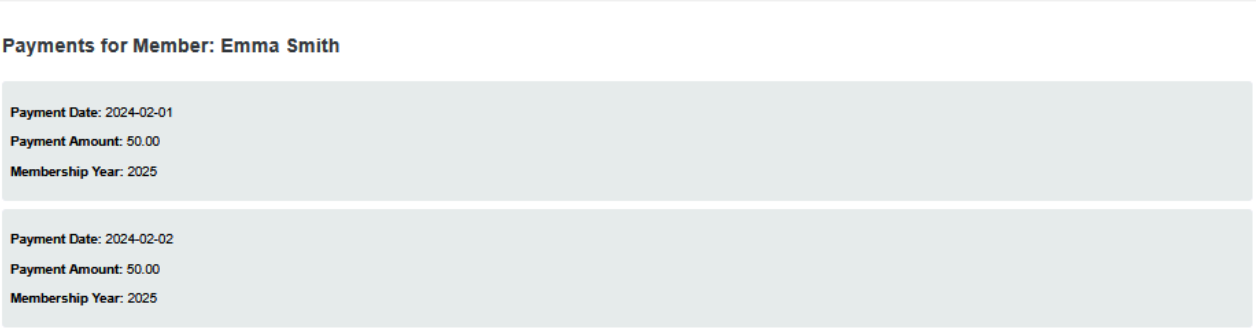
Save Location


Create Team
Team ID:
Location ID:
Team Name:
Captain Number:
Team Gender:
Save Team


Modify Team: 1
Team ID:
1
Location ID:
1
Team Name:
The Thunderhawks
Captain Number:
51
Team Gender:
Boys
Save Team


Home
People
Locations & Teams
Finance
Locations
Create New
MTL Headquarter
Location ID: 1
Location Type: Head
Address: 123 Main St, Montreal, QC H0H0H0
Phone Number: 514-111-1111
Web Address: www.myvc-head.ca
Max Capacity: 1000
Delete
Modify
Laval Branch
Location ID: 2
Location Type: Branch
Address: 456 Elm Rd, Laval, QC H7T2X2
Phone Number: 450-222-2222
Web Address: www.myvc-laval.ca
Max Capacity: 500
Delete
Modify
Longueuil Branch
Location ID: 3
Location Type: Branch
Address: 789 Oak Ave, Longueuil, QC J4K1T5
Phone Number: 450-333-3333
Web Address: www.myvc-longueuil.ca
Max Capacity: 400
Delete
Modify
Teams
Create New
The Thunderhawks
Team ID: 1
Location ID: 1
Captain PID: 51
Gender: Boys
Delete
Modify
The Night Riders
Team ID: 2
Location ID: 2
Captain PID: 52
Gender: Girls
Delete
Modify
The Storm Chasers
Team ID: 3
Location ID: 3
Captain PID: 53
Gender: Boys
Delete
Modify
The Firebirds
Team ID: 4
Location ID: 4
Captain PID: 54


Home
People
Locations & Teams
Finance
Yearly Earnings
Enter Year: 2025
Show Payments
Total Membership Fees: $11965.00
Total Donations: $250.00
Club Members
John Doe
Member ID: 1
Birthday: 2012-07-15
Address: 1 First St, Montreal, QC H1B9Q2
Telephone: 514-123-4567
Emma Smith
Member ID: 2
Birthday: 2010-05-20
Address: 2 Second St, Ottawa, ON K1A0B1
Telephone: 514-234-5678
Oliver Johnson
Member ID: 3
Birthday: 2008-04-25
Address: 3 Third St, Toronto, ON M5A1A1
Telephone: 416-345-6789
Sophia Brown
Member ID: 4
Birthday: 2013-03-18
Address: 4 Fourth St, Quebec City, QC G1A2A2
Telephone: 819-456-7890
Mason Williams
Payments for Member: Select a Member
Select a member to view payments


Payments for Member: Emma Smith
Payment Date: 2024-02-01
Payment Amount: 50.00
Membership Year: 2025
Payment Date: 2024-02-02
Payment Amount: 50.00
Membership Year: 2025

7. Conclusion

In this project, we successfully designed and implemented a comprehensive database system for the Montreal Volleyball Club. Through careful analysis and normalization, we ensured that the schema met both 3NF and BCNF requirements, optimizing data integrity and minimizing redundancy. The SQL implementation, complete with data population, queries, and transactions, demonstrates the system’s capacity to manage member records, personnel roles, locations, sessions, and payments efficiently. Furthermore, the developed web interface and extensive testing validate the system's usability and functionality, making it a robust solution for the club’s operational needs.