COMP 353 Project

# Montréal Youth Soccer Club

Drew Wagner #40042144 (Leader, d\_wagne@encs.concordia.ca)

Arvand Azarbar #40166313 (a\_azarba@encs.concordia.ca)

Parsa Ghadimi #40203370 (p\_ghadim@encs.concordia.ca)

Ricardo RAJI CHAHINE #40234410 ([r\_rajich@encs.concordia.ca](mailto:r_rajich@encs.concordia.ca))

# I-Assumptions

Please find our assumptions, notes and specifications in the respective files. In other words, each file contains its own specific assumptions. This was done to keep things professional, tidy and relevant.

For example, in the “migrations/4\_create\_teams\_events.sql” file, the following comments are added:

*-- if both teams are visiting, then choose one arbitrarily to be home*

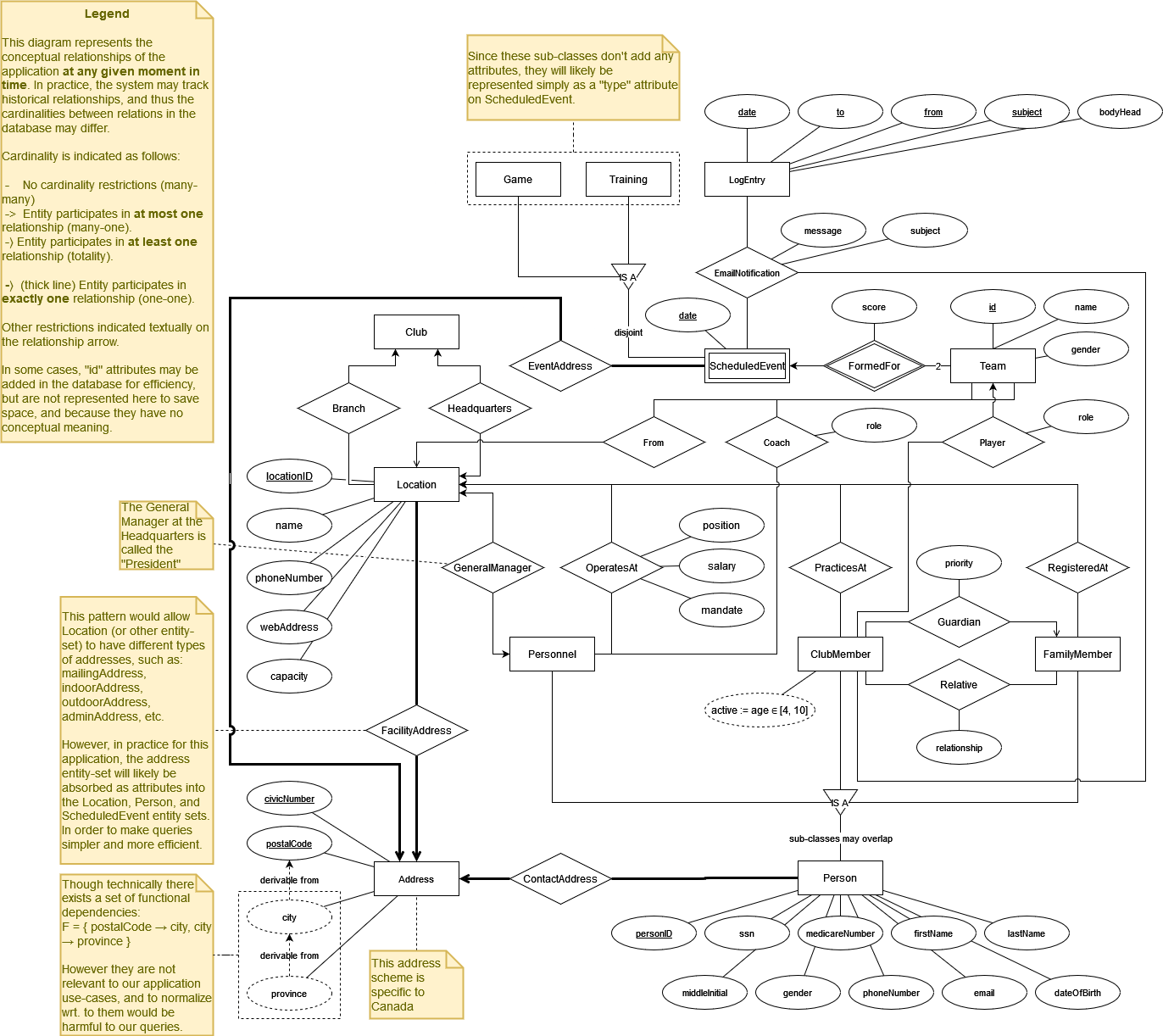
/\* Events may be international, in which case we can't rely on the canadian address system.

\* Additionally, it may be desirable to include a more descriptive location than what can be

\* done with civicNumber + postalCode.

\*/

# II-ER Diagram



# III-ER to Relations

1. Person

CREATE TABLE IF NOT EXISTS Persons (

    personID SERIAL PRIMARY KEY,

    firstName VARCHAR(50),

    lastName VARCHAR(50),

    dateOfBirth DATE,

*/\* If other, then put prefered team \*/*

    gender ENUM('m', 'f'),

    SIN CHAR(9) UNIQUE NOT NULL,

    medicareCardNumber VARCHAR(24) UNIQUE,

    telephoneNumber VARCHAR(15),

    civicNumber INT UNSIGNED,

    city VARCHAR(50),

    province ENUM(

        'NL',

        'PE',

        'NS',

        'NB',

        'QC',

        'ON',

        'MB',

        'SK',

        'AB',

        'BC',

        'YT',

        'NT',

        'NU'

    ),

    postalCode CHAR(6),

    emailAddress VARCHAR(320),

    createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

1. Locations

CREATE TABLE IF NOT EXISTS Locations (

    locationID SERIAL PRIMARY KEY,

    name VARCHAR(50),

    civicNumber INT UNSIGNED,

    city VARCHAR(50),

    province ENUM(

        'NL',

        'PE',

        'NS',

        'NB',

        'QC',

        'ON',

        'MB',

        'SK',

        'AB',

        'BC',

        'YT',

        'NT',

        'NU'

    ),

    webAddress VARCHAR(2048),

*/\* Max length generally supported by modern browsers \*/*

    postalCode CHAR(6),

    phoneNumber VARCHAR(15),

    type ENUM('Head', 'Branch'),

    capacity INT UNSIGNED,

    createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

1. Family Members and Club Members

CREATE TABLE IF NOT EXISTS

    ClubMember\_Relatives (

        clubMemberID BIGINT UNSIGNED NOT NULL,

        relativeID BIGINT UNSIGNED NOT NULL,

        relationship ENUM(

            'Father',

            'Mother',

            'GrandFather',

            'GrandMother',

            'Tutor',

            'Partner',

            'Friend',

            'Other'

        ),

        createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

*/\* Can change if a friend becomes a parent's partner \*/*

        updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

        PRIMARY KEY (clubMemberID, relativeID),

        FOREIGN KEY (clubMemberID) REFERENCES Persons (personID),

        FOREIGN KEY (relativeID) REFERENCES Persons (personID)

    );

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE TABLE IF NOT EXISTS

    ClubMember\_Guardians (

        clubMemberID BIGINT UNSIGNED NOT NULL,

        guardianID BIGINT UNSIGNED NOT NULL,

        startDate DATE NOT NULL,

        endDate DATE,

        `priority` INT UNSIGNED NOT NULL,

        PRIMARY KEY (clubMemberID, guardianID, startDate),

        FOREIGN KEY (clubMemberID) REFERENCES Persons (personID),

        FOREIGN KEY (guardianID) REFERENCES Persons (personID),

        CHECK (

            endDate IS NULL

            OR startDate < endDate

        )

    );

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE TABLE IF NOT EXISTS

    Family\_ClubMember\_Location (

        locationID BIGINT UNSIGNED NOT NULL,

        personID BIGINT UNSIGNED NOT NULL,

        startDate DATE NOT NULL,

        endDate DATE,

        PRIMARY KEY (locationID, personID, startDate),

        FOREIGN KEY (locationID) REFERENCES Locations (locationID),

        FOREIGN KEY (personID) REFERENCES Persons (personID),

        CHECK (

            endDate IS NULL

            OR startDate < endDate

        )

    );

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE VIEW IF NOT EXISTS

    Active\_Family\_ClubMember\_Location AS

SELECT

    \*

FROM

    Family\_ClubMember\_Location

WHERE

    endDate IS NULL;

CREATE VIEW IF NOT EXISTS

    Active\_ClubMember\_Guardians AS

SELECT

    \*

FROM

    ClubMember\_Guardians

WHERE

    endDate IS NULL;

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE VIEW IF NOT EXISTS

    Active\_ClubMembers AS

SELECT

    p.\*,

    locationID,

    fcl.startDate as locationStartDate,

    FIRST\_VALUE(guardianID) OVER (

        PARTITION BY

            personID

        ORDER BY

            `priority` ASC

    ) as primaryGuardianID,

    NTH\_VALUE(guardianID, 2) OVER (

        PARTITION BY

            personID

        ORDER BY

            `priority` ASC

    ) as secondaryGuardianID

FROM

    Persons p

    JOIN Active\_ClubMember\_Guardians cg ON (cg.clubMemberID = p.personID)

    JOIN Active\_Family\_ClubMember\_Location fcl ON (p.personID = fcl.personID)

WHERE

    (

        TIMESTAMPDIFF (YEAR, p.dateOfBirth, CURRENT\_DATE) BETWEEN 4 AND 10

    )

GROUP BY

    p.personID;

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE TRIGGER validate\_clubmember\_start\_date AFTER

INSERT

    ON Family\_ClubMember\_Location FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        Family\_ClubMember\_Location

    WHERE

        personID = NEW.personID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END;

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE TRIGGER validate\_guardian\_start\_date AFTER

INSERT

    ON ClubMember\_Guardians FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        ClubMember\_Guardians

    WHERE

        clubMemberID = NEW.clubMemberID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END;

1. Personnel

CREATE TABLE IF NOT EXISTS

    Personnel (

        locationID BIGINT UNSIGNED NOT NULL,

        personID BIGINT UNSIGNED NOT NULL,

        startDate DATE NOT NULL,

        endDate DATE,

        `role` ENUM('Administrator', 'Trainer', 'Other'),

        mandate ENUM('Volunteer', 'Salary'),

        isGeneralManager BOOLEAN,

        PRIMARY KEY (locationID, personID, startDate),

        FOREIGN KEY (locationID) REFERENCES Locations (locationID),

        FOREIGN KEY (personID) REFERENCES Persons (personID),

        CHECK (

            endDate IS NULL

            OR startDate < endDate

        )

    );

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE VIEW IF NOT EXISTS

    Active\_Personnel AS

SELECT

    \*

FROM

    Personnel

WHERE

    endDate IS NULL;

DELIMITER $$

*----------------------------------------------------------------------------------------------------------------------------------*

CREATE TRIGGER validate\_personnel\_start\_date AFTER

INSERT

    ON Personnel FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        Personnel

    WHERE

        personID = NEW.personID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END;

1. Location Teams

CREATE TABLE IF NOT EXISTS Teams (

    locationID BIGINT UNSIGNED NOT NULL,

    teamNumber INT UNSIGNED NOT NULL,

    `name` VARCHAR(50),

    gender ENUM('m', 'f'),

    createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

    PRIMARY KEY (locationID, teamNumber),

    FOREIGN KEY (locationID) REFERENCES Locations(locationID)

);

1. Team Members

CREATE TABLE IF NOT EXISTS

    Team\_Members (

        teamID BIGINT UNSIGNED NOT NULL,

        personID BIGINT UNSIGNED,

        startDate DATE NOT NULL,

        endDate DATE,

        `role` ENUM(

            'coach',

            'goalkeeper',

            'defender',

            'midfielder',

            'forward',

            'other'

        ),

        isCaptain BOOLEAN,

        PRIMARY KEY (teamID, personID, startDate),

        FOREIGN KEY (teamID) REFERENCES Teams (teamID),

        FOREIGN KEY (personID) REFERENCES Persons (personID),

        CHECK (

            endDate IS NULL

            OR startDate < endDate

        )

    );

1. Scheduled Events

CREATE TABLE IF NOT EXISTS

    Scheduled\_Events (

*-- if both teams are visiting, then choose one arbitrarily to be home*

        homeTeamID BIGINT UNSIGNED NOT NULL,

        visitingTeamID BIGINT UNSIGNED NOT NULL,

        `date` TIMESTAMP NOT NULL,

*/\* Events may be international, in which case we can't rely on the canadian address system.*

*\* Additionally, it may be desirable to include a more descriptive location than what can be*

*\* done with civicNumber + postalCode.*

*\*/*

        `address` TEXT,

        `type` ENUM('training', 'game'),

        homeScore INT UNSIGNED DEFAULT 0,

        visitingScore INT UNSIGNED DEFAULT 0,

        createdAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

        updatedAt TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

        PRIMARY KEY (homeTeamID, visitingTeamID, `date`),

        FOREIGN KEY (homeTeamID) REFERENCES Teams (teamID),

        FOREIGN KEY (visitingTeamID) REFERENCES Teams (teamID)

    );

# IV-Normalization and V-Functionalities

## Tables and Views

A screenshot of a computer

Description automatically generated

## Personnel

### Initial table

Personnel (personID, ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth, position, salary, mandate, startDate, endDate, locationId, locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID, province, city, postalCode,civicNumber)

### Functional dependencies:

* personID -> { ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* ssn -> { personID, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* medicareNumber -> { personID, ssn, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* { personID, locationId, startDate } -> {position, salary, mandate, endDate}
* locationId -> { locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID}
* postalCode → {city, province, civicNumber}
* City → {province}

### Decomposing to 3NF:

**Person** (personID, ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth)

**Personnel** (personID, locationId, startDate, position, salary, mandate, endDate)

**Location** (locationId, locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID)

**The following table is not in 3NF.**

**personAddress(province, city, postalCode,civicNumber)**

## ClubMember

### Initial table

ClubMember (personID, ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth, startDate, endDate, locationId, locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID, familyMemberID, familyMemberRelationship,familyMemberPriority,province, city, postalCode,civicNumber)

### Functional dependencies:

* personID -> { ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* ssn -> { personID, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* medicareNumber -> { personID, ssn, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth }
* locationId -> { locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID}
* personID, relativeID ->, familyMemberRelationship,
* personID, familyMemberPriority -> familyMemberID, familyMemberRelationship
* postalCode → {city, province, civicNumber}
* city → {province}

### Decomposing to 3NF:

The following tables all satisfy the 3NF since the LHS of all the functional dependencies are candidate keys (personID,ssn,medicareNumber, locationID, etc.)

**ClubMember** (personID, ssn, medicareNumber, firstName, lastName, middleInitial, gender, phoneNumber, email, dateOfBirth, startDate, endDate, locationId, primaryFamilyMemberID, priority)

**Location** (locationId, locationName, locationPhoneNumber, locationWebAddress, locationCapacity, locationGeneralManagerID)

**familyRelative**( personID, relativeID, relationship)

**familyMember**(personID, familyMemberID, familyMemberRelationship, priority)

**The following table is not in 3NF.**

**personAddress(province, city, postalCode,civicNumber)**

**Event**

### Initial table

Event(TeamID1,TeamID2,EventID,EventDate,EventStartTime,EventType,scoreTeam1, ScoreTeam2,province, city, postalCode,civicNumber)

### Functional dependencies:

* EventID → {TeamID1, TeamID2, EventDate, type(training,game), ,ScoreTeam1,ScoreTeam2, province, city, postalCode, civicNumber}
* PostalCode → city, civicNumber, province

### Decomposing to 3NF:

The table, Event, satisfies the 3NF since the LHS of all the functional dependencies are candidate keys.

It’s also in BCNF.

Event (Event(TeamID1,TeamID2,EventID,EventDate,EventStartTime,EventType,scoreTeam1, ScoreTeam2)

**The following table is not in 3NF.**

**personAddress(province, city, postalCode,civicNumber)**

# VI-Triggers

Trigger to set the current Personnel’s endDate to null when inserting a new Personnel with the same personID:

CREATE TRIGGER update\_personnel\_end\_date BEFORE

INSERT

    ON Personnel FOR EACH ROW

BEGIN IF (NEW.endDate IS NULL) THEN

UPDATE Personnel

SET

    endDate = NEW.startDate

WHERE

    personID = NEW.personID

    AND endDate IS NULL;

END IF;

END$$

DELIMITER ;

DELIMITER $$

Trigger to validate Personnel startDate after inserting a Personnel with the same personID:

CREATE TRIGGER validate\_personnel\_start\_date AFTER

INSERT

    ON Personnel FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        Personnel

    WHERE

        personID = NEW.personID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END$$

DELIMITER ;

Trigger to set the current Club Member/Guardian relationship’s endDate to null before inserting a new Guardian record for a given club member:

CREATE TRIGGER update\_guardian\_end\_date BEFORE

INSERT

    ON ClubMember\_Guardians FOR EACH ROW

BEGIN IF (NEW.endDate IS NULL) THEN

UPDATE ClubMember\_Guardians

SET

    endDate = NEW.startDate

WHERE

    clubMemberID = NEW.clubMemberID

    AND endDate IS NULL;

END IF;

END$$

DELIMITER ;

Trigger to validate a Club Member/Guardian relationship’s startDate before inserting a Guardian with the same personID:

CREATE TRIGGER validate\_guardian\_start\_date AFTER

INSERT

    ON ClubMember\_Guardians FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        ClubMember\_Guardians

    WHERE

        clubMemberID = NEW.clubMemberID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END$$

DELIMITER ;

Trigger to change a ClubMember’s endDate from null, before a new record(activity) for the same ClubMember is added:

   CREATE TRIGGER update\_clubmember\_location\_end\_date BEFORE

    INSERT

        ON Family\_ClubMember\_Location FOR EACH ROW

    BEGIN IF (NEW.endDate IS NULL) THEN

    UPDATE Family\_ClubMember\_Location

    SET

        endDate = NEW.startDate

    WHERE

        personID = NEW.personID

        AND endDate IS NULL;

    END IF;

    END$$

    DELIMITER ;

Trigger to update a ClubMember’s startDate after a new record(activity) for the same ClubMember is added:

CREATE TRIGGER validate\_clubmember\_start\_date AFTER

INSERT

    ON Family\_ClubMember\_Location FOR EACH ROW

BEGIN IF EXISTS (

    SELECT

        1

    FROM

        Family\_ClubMember\_Location

    WHERE

        personID = NEW.personID

        AND (

            (NEW.startDate BETWEEN startDate AND endDate)

            OR (

                NEW.endDate > startDate

                AND NEW.endDate < endDate

            )

        )

) THEN

SIGNAL SQLSTATE '45000'

SET

    MESSAGE\_TEXT = 'Start date must be later than most recent endDate';

END IF;

END$$

DELIMITER ;

# VII-Extra Features

# User Interface

A screenshot of a website

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a football club

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Automatic Data Generation

Records generation using Python.

Please find all of the relevant information and instructions, in the “**comp353\_data\_faker.ipynb”** file.