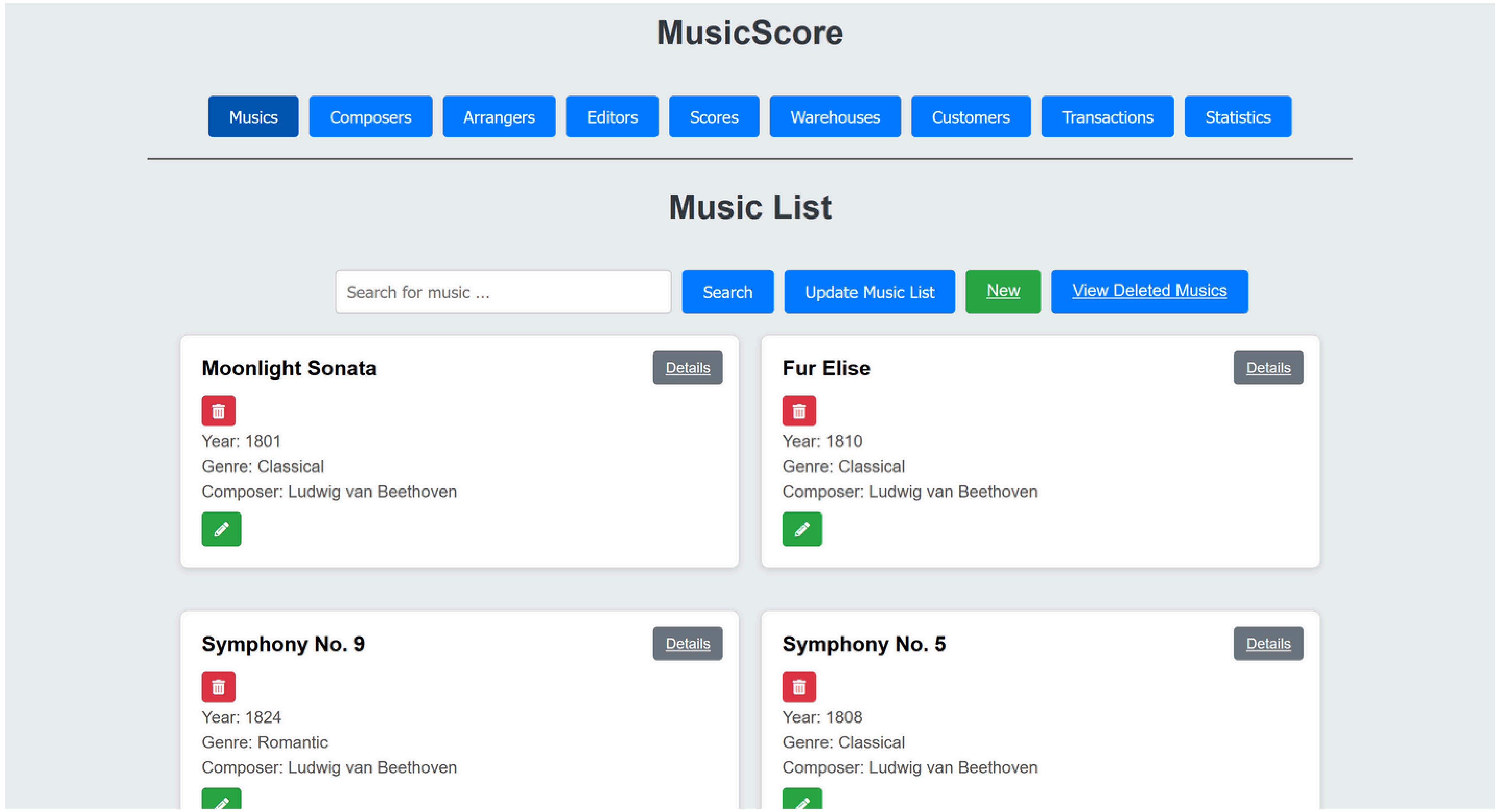
DataBase



Music List

Moonlight Sonata
Year: 1801
Genre: Classical
Composer: Ludwig van Beethoven

Fur Elise
Year: 1810
Genre: Classical
Composer: Ludwig van Beethoven

Symphony No. 9
Year: 1824
Genre: Romantic
Composer: Ludwig van Beethoven

Symphony No. 5
Year: 1808
Genre: Classical
Composer: Ludwig van Beethoven

João Monteiro - 114547
Jorge Domingues - 113278
P5G4

Introduction

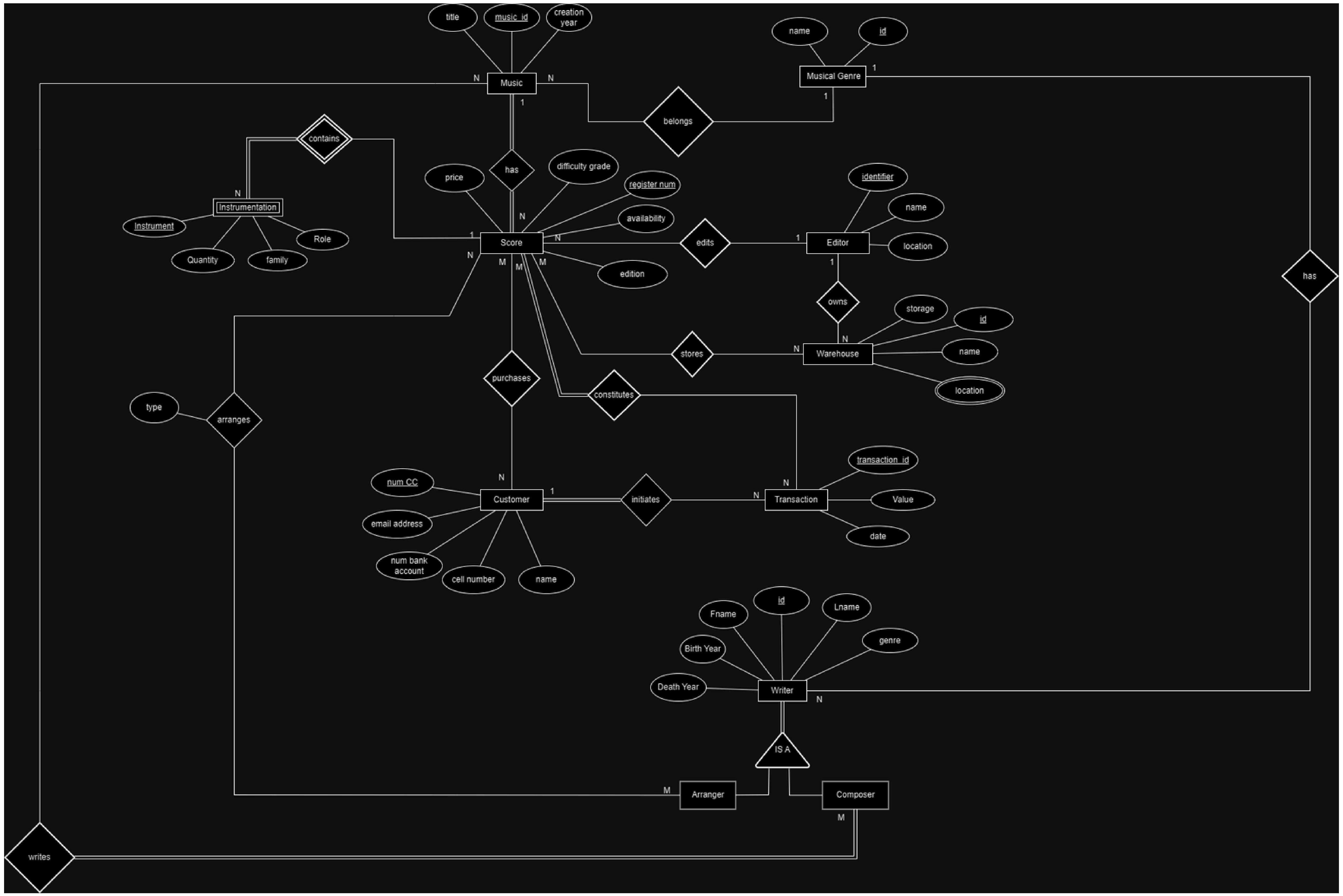
Entities:

- Music
- Score
- Writer (Composer or Arranger)
- Editor
- Warehouse
- Customer
- Transaction

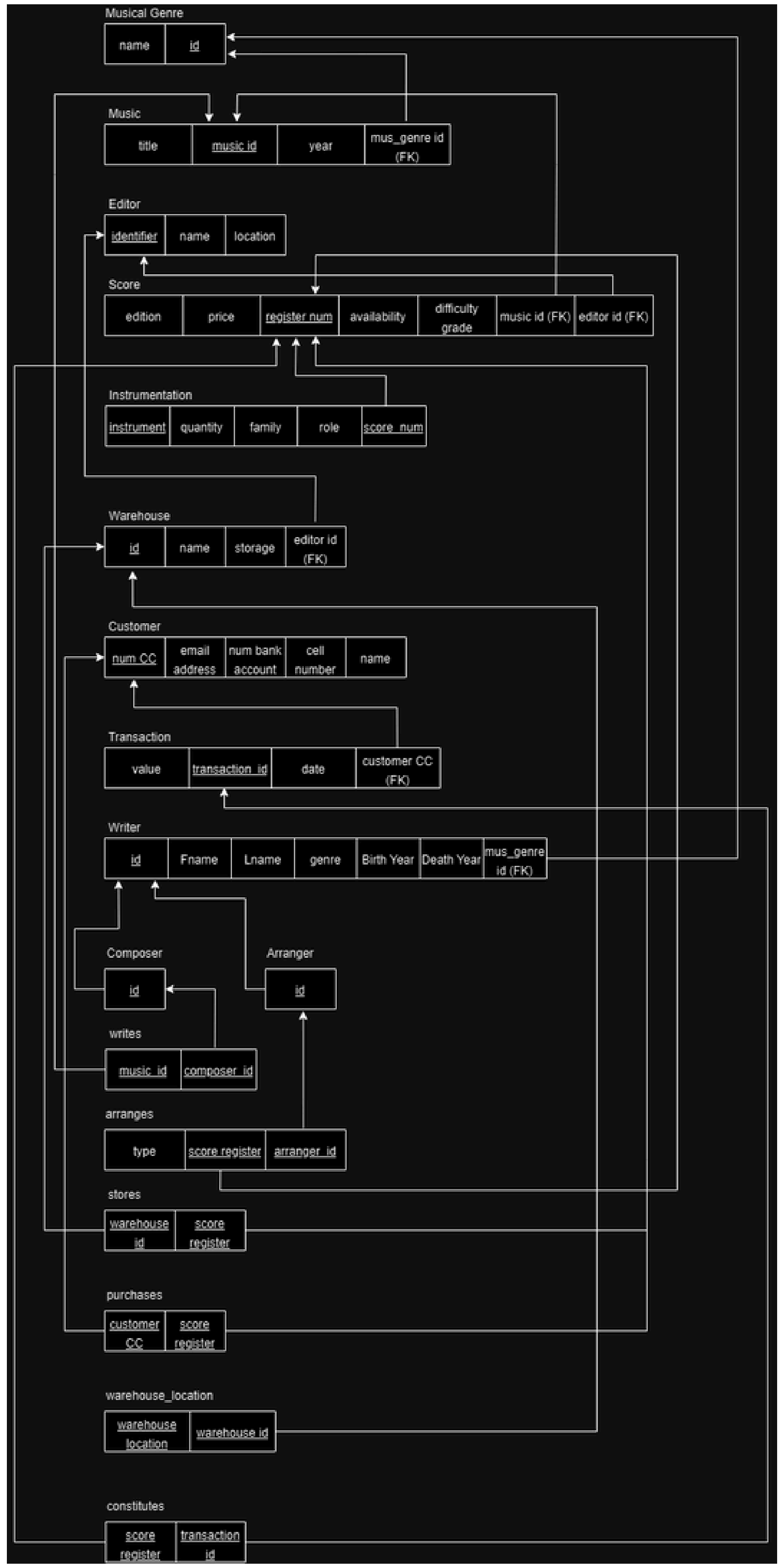
Functionalities:

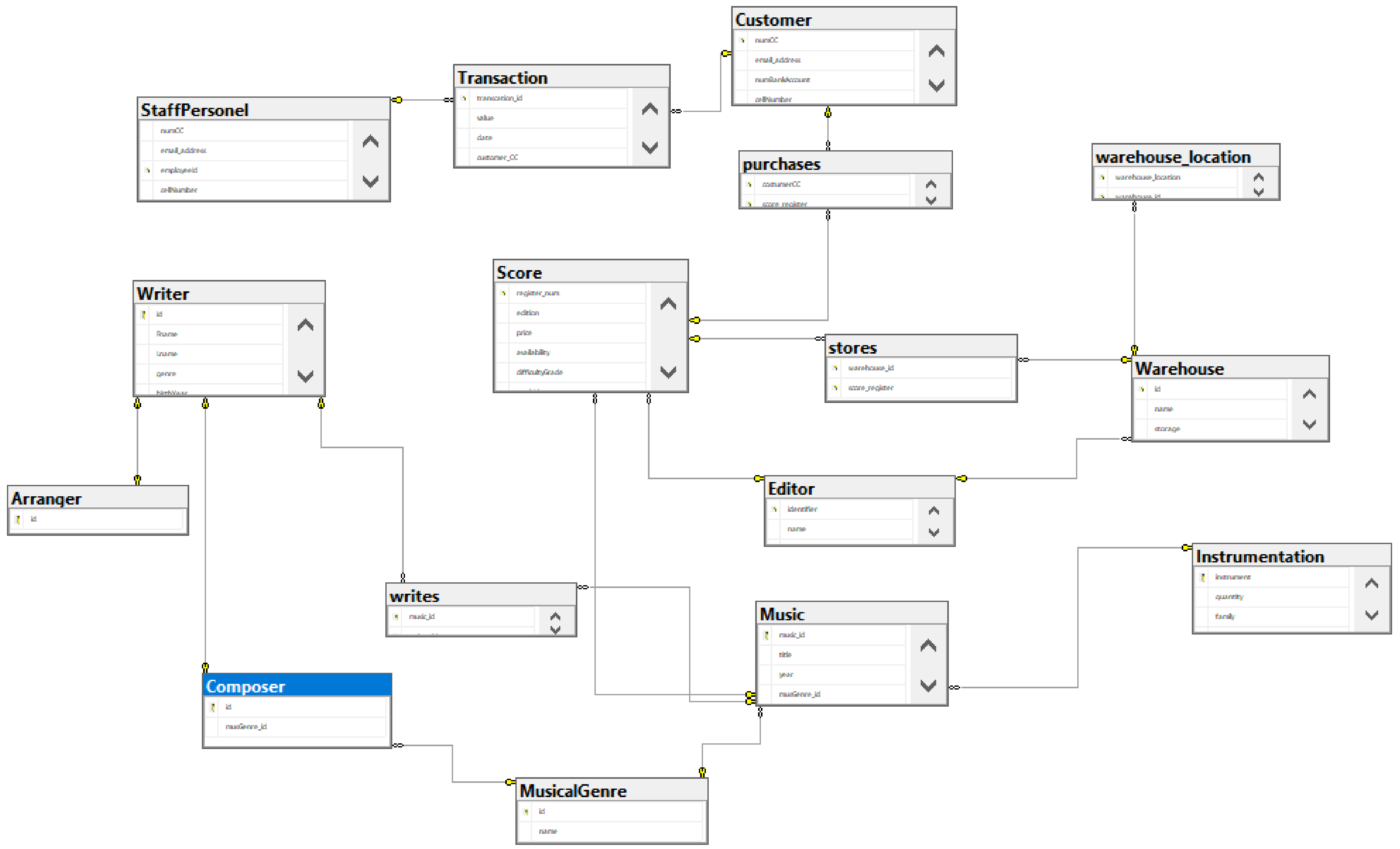
- Add entities
- Search entities
- Edit entities
- Buy scores
- See bought scores by customer
- Check scores availability
- Check warehouse content
- Others

Entity-Relationship Diagrams

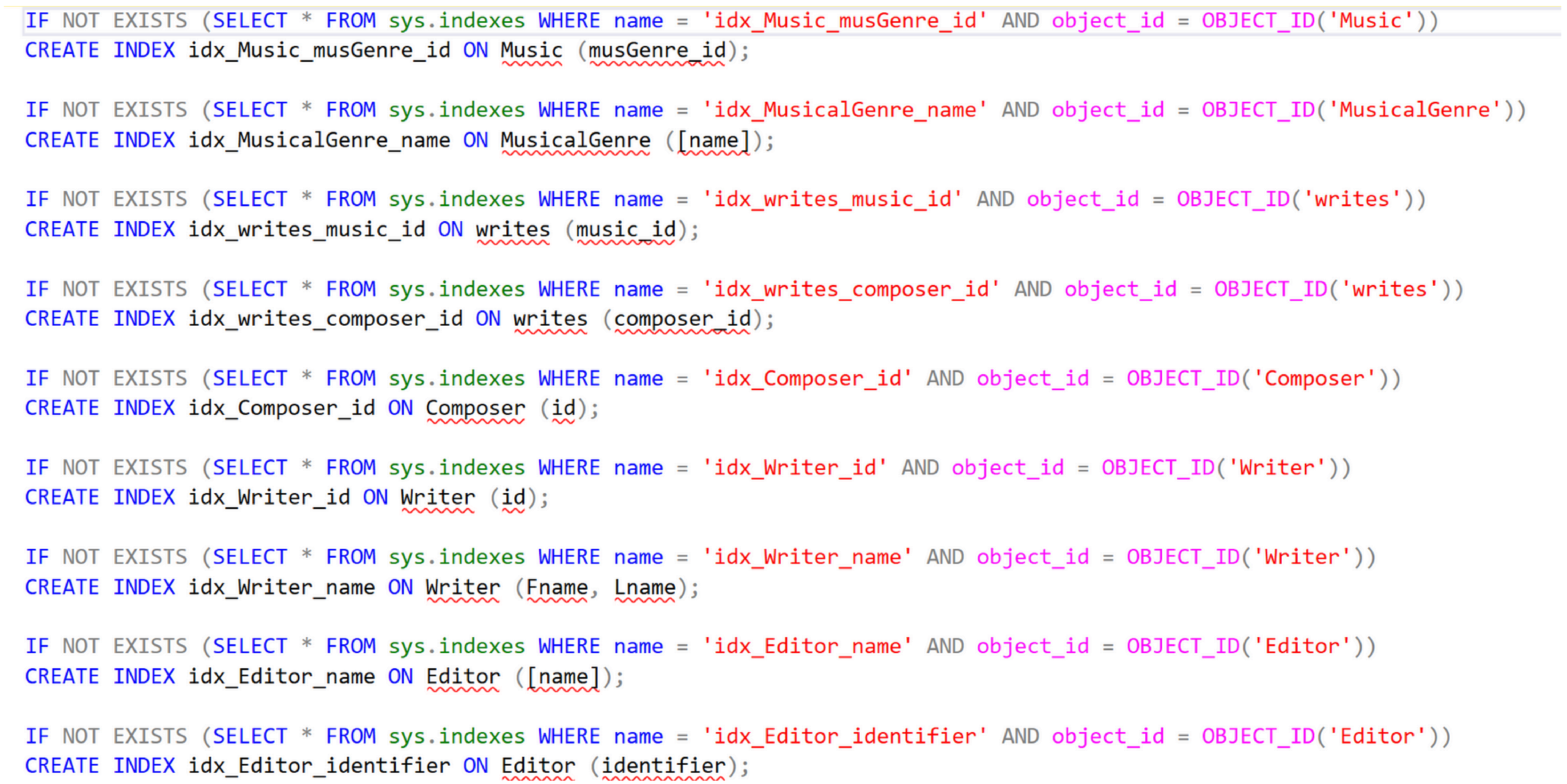


Relational Diagram


Musical Genre
name | id
Music
title | music id | year | mus_genre id (FK)
Editor
identifier | name | location
Score
edition | price | register num | availability | difficulty grade | music id (FK) | editor id (FK)
Instrumentation
instrument | quantity | family | role | score num
Warehouse
id | name | storage | editor id (FK)
Customer
num CC | email address | num bank account | cell number | name
Transaction
value | transaction id | date | customer CC (FK)
Writer
id | Fname | Lname | genre | Birth Year | Death Year | mus_genre id (FK)
Composer
id
Arranger
id
writes
music id | composer id
arranges
type | score register | arranger id
stores
warehouse id | score register
purchases
customer CC | score register
warehouse_location
warehouse location | warehouse id
constitutes
score register | transaction id



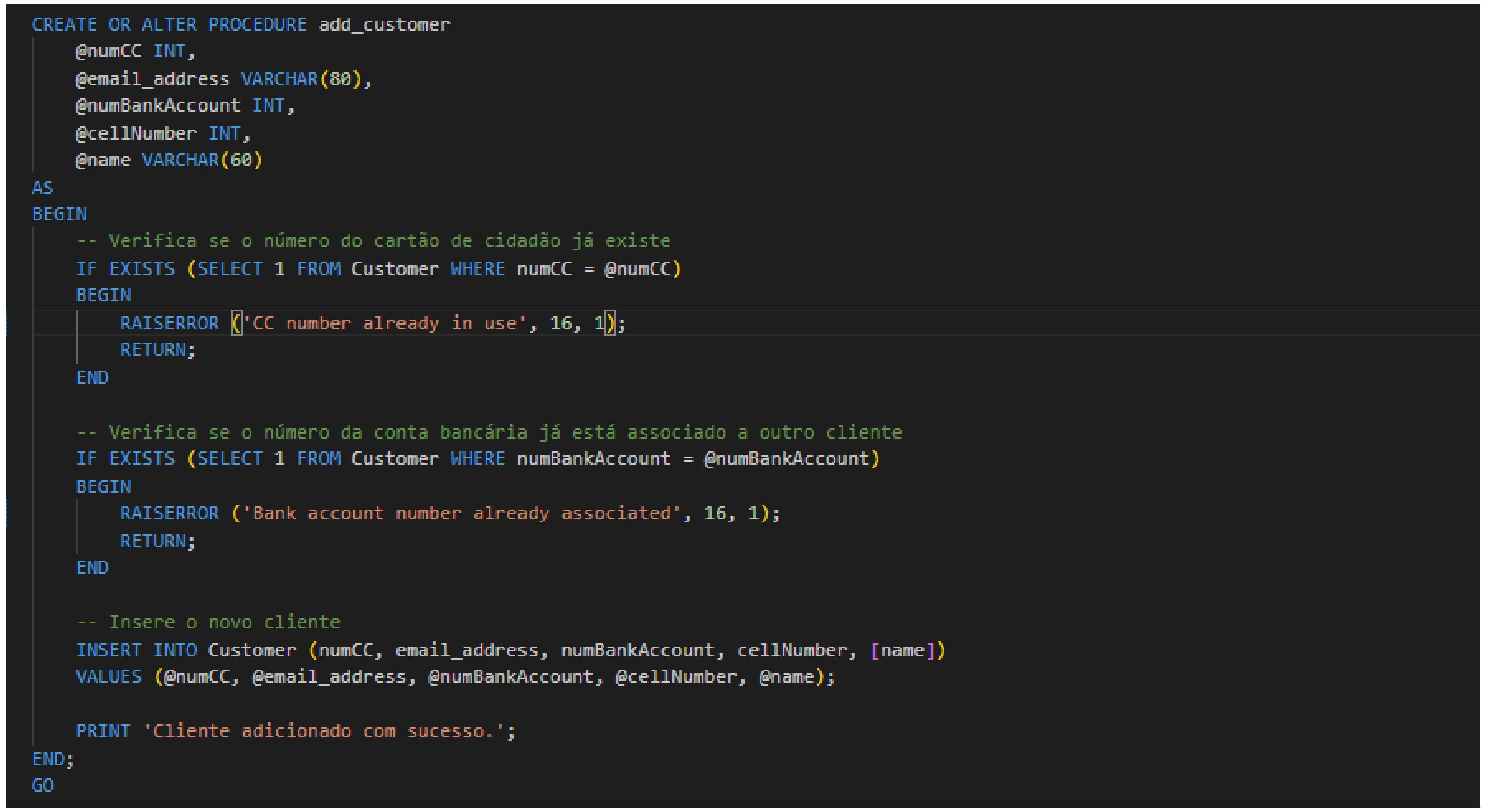
Indexes



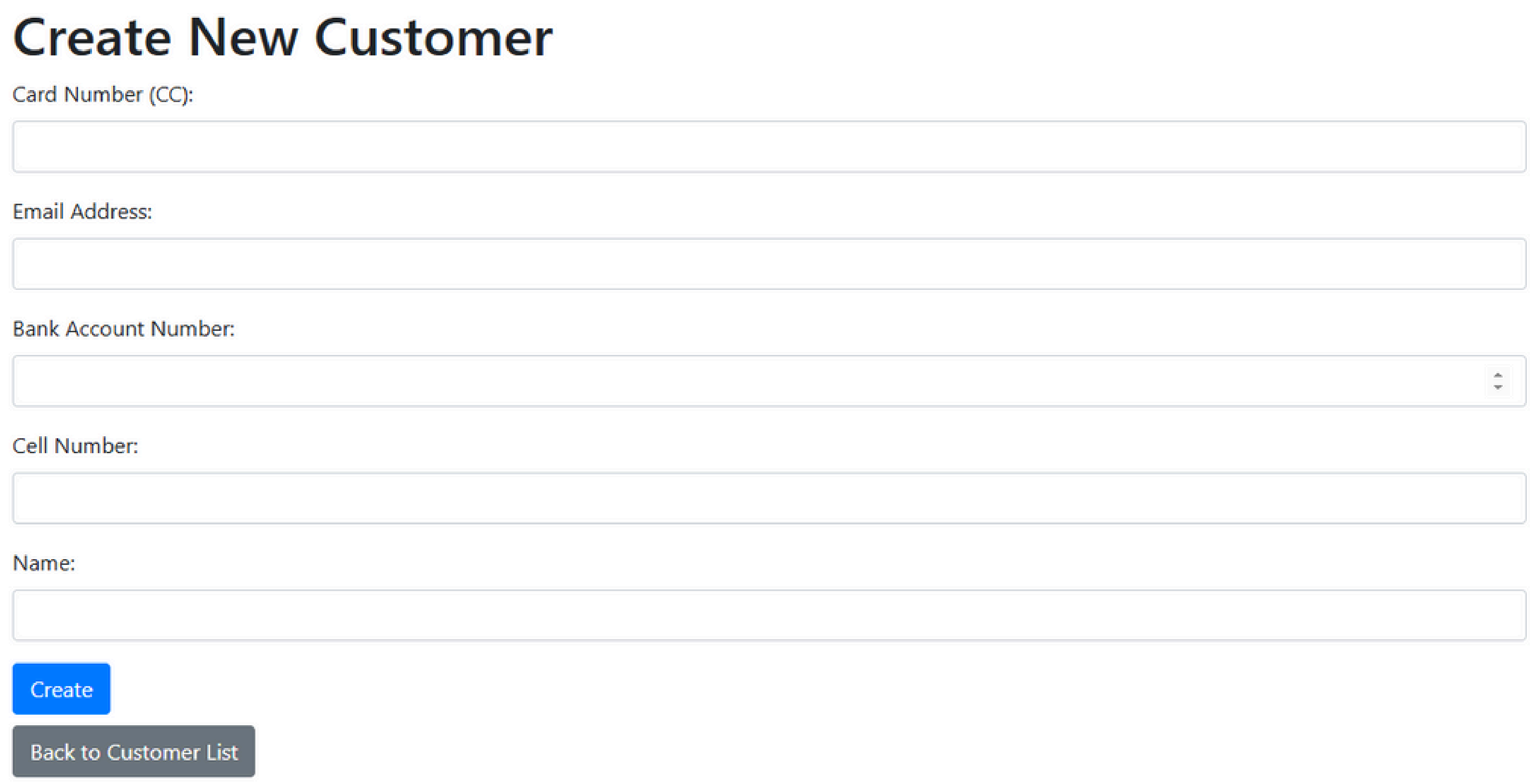
```sql
IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Music_musGenre_id' AND object_id = OBJECT_ID('Music'))
CREATE INDEX idx_Music_musGenre_id ON Music (musGenre_id);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_MusicalGenre_name' AND object_id = OBJECT_ID('MusicalGenre'))
CREATE INDEX idx_MusicalGenre_name ON MusicalGenre ([name]);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_writes_music_id' AND object_id = OBJECT_ID('writes'))
CREATE INDEX idx_writes_music_id ON writes (music_id);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_writes_composer_id' AND object_id = OBJECT_ID('writes'))
CREATE INDEX idx_writes_composer_id ON writes (composer_id);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Composer_id' AND object_id = OBJECT_ID('Composer'))
CREATE INDEX idx_Composer_id ON Composer (id);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Writer_id' AND object_id = OBJECT_ID('Writer'))
CREATE INDEX idx_Writer_id ON Writer (id);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Writer_name' AND object_id = OBJECT_ID('Writer'))
CREATE INDEX idx_Writer_name ON Writer (Fname, Lname);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Editor_name' AND object_id = OBJECT_ID('Editor'))
CREATE INDEX idx_Editor_name ON Editor ([name]);

IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'idx_Editor_identifier' AND object_id = OBJECT_ID('Editor'))
CREATE INDEX idx_Editor_identifier ON Editor (identifier);
```

Stored Procedure

Add Customer



```sql
CREATE OR ALTER PROCEDURE add_customer
    @numCC INT,
    @email_address VARCHAR(80),
    @numBankAccount INT,
    @cellNumber INT,
    @name VARCHAR(60)
AS
BEGIN
    -- Verifica se o número do cartão de cidadão já existe
    IF EXISTS (SELECT 1 FROM Customer WHERE numCC = @numCC)
    BEGIN
        RAISERROR ('CC number already in use', 16, 1);
        RETURN;
    END

    -- Verifica se o número da conta bancária já está associado a outro cliente
    IF EXISTS (SELECT 1 FROM Customer WHERE numBankAccount = @numBankAccount)
    BEGIN
        RAISERROR ('Bank account number already associated', 16, 1);
        RETURN;
    END

    -- Insere o novo cliente
    INSERT INTO Customer (numCC, email_address, numBankAccount, cellNumber, [name])
    VALUES (@numCC, @email_address, @numBankAccount, @cellNumber, @name);

    PRINT 'Cliente adicionado com sucesso.';
END;
GO
```



Create New Customer

Card Number (CC):

Email Address:

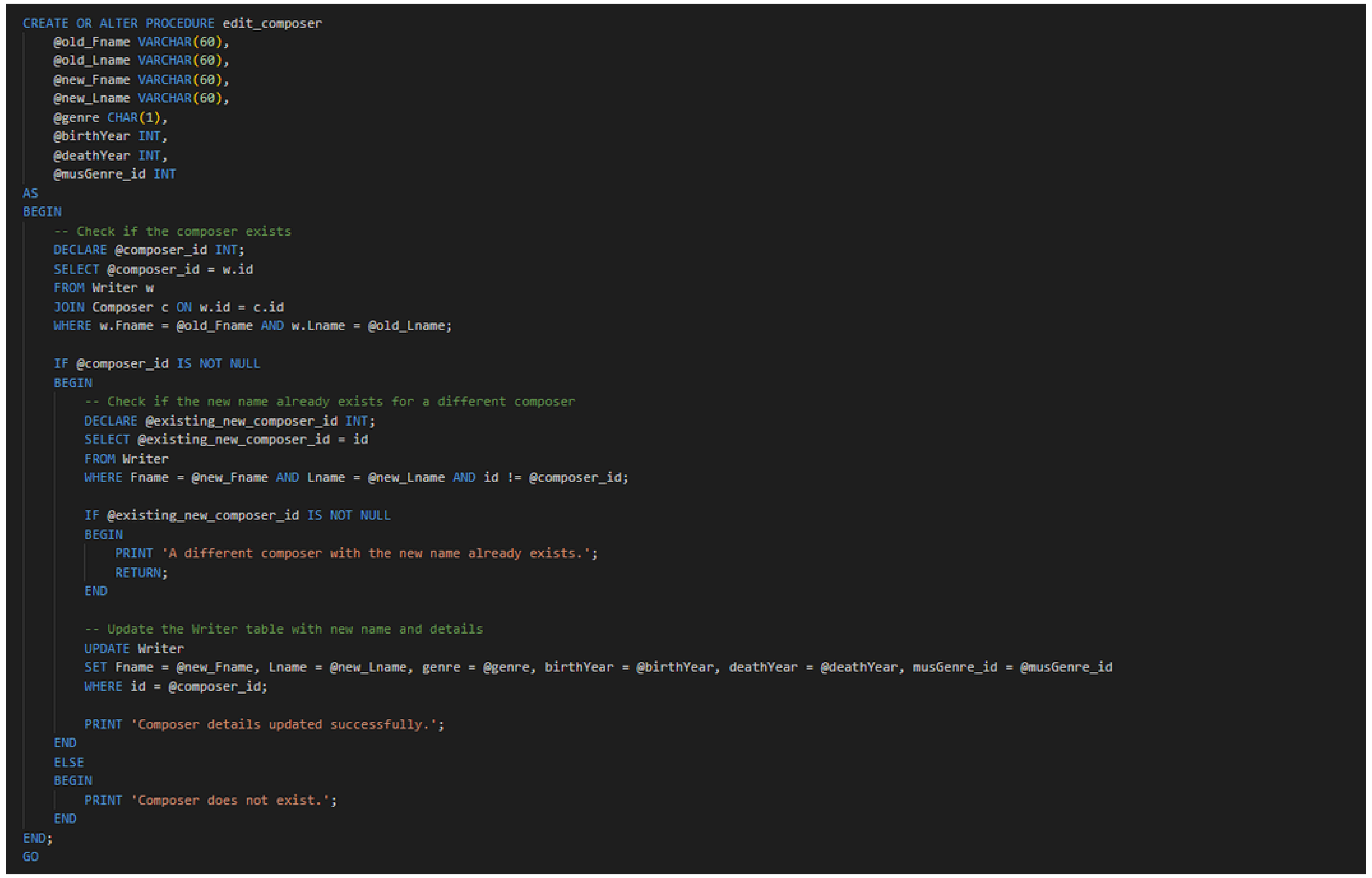
Bank Account Number:

Cell Number:

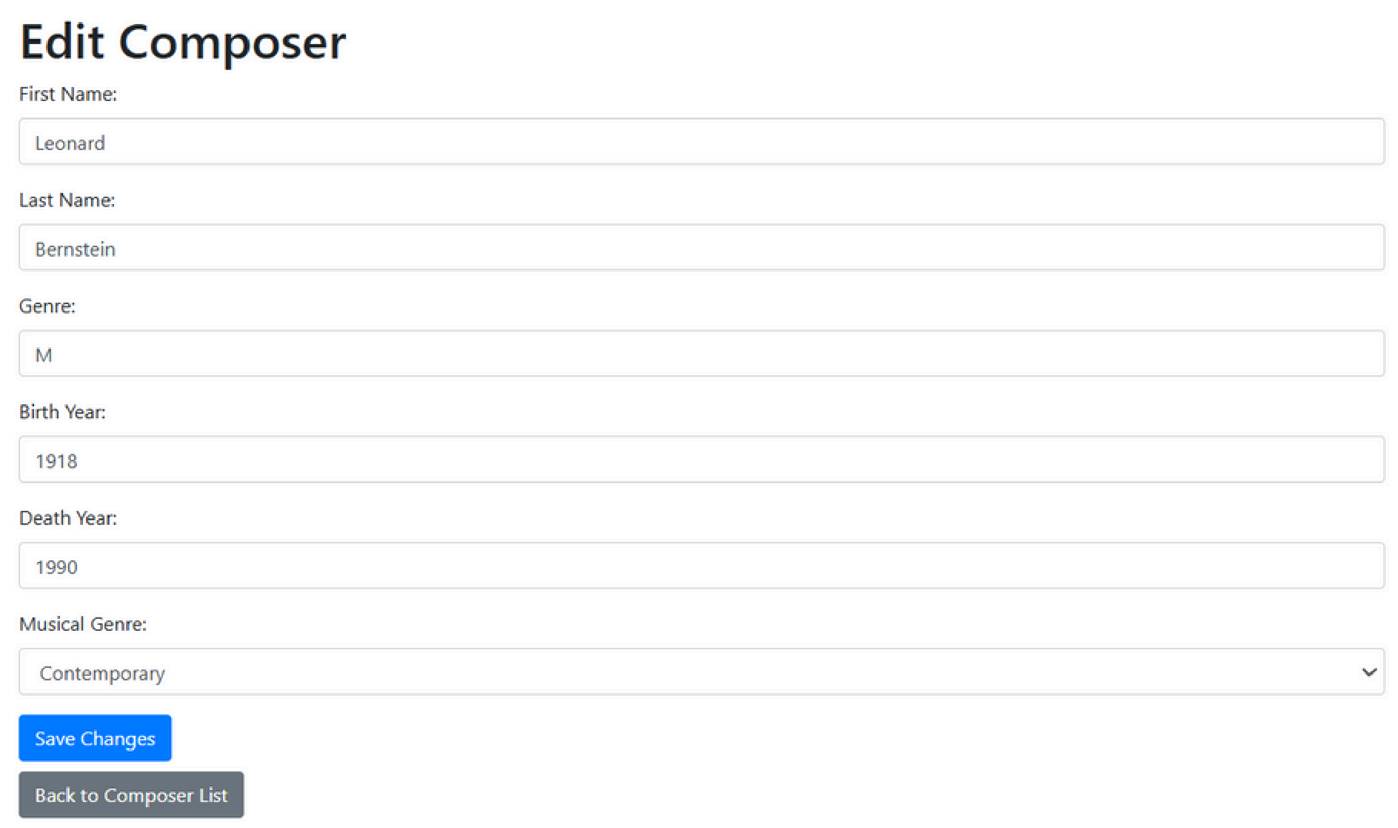
Name:

Stored Procedure

Edit Composer



```sql
CREATE OR ALTER PROCEDURE edit_composer
    @old_Fname VARCHAR(60),
    @old_Lname VARCHAR(60),
    @new_Fname VARCHAR(60),
    @new_Lname VARCHAR(60),
    @genre CHAR(1),
    @birthYear INT,
    @deathYear INT,
    @musGenre_id INT
AS
BEGIN
    -- Check if the composer exists
    DECLARE @composer_id INT;
    SELECT @composer_id = w.id
    FROM Writer w
    JOIN Composer c ON w.id = c.id
    WHERE w.Fname = @old_Fname AND w.Lname = @old_Lname;

    IF @composer_id IS NOT NULL
    BEGIN
        -- Check if the new name already exists for a different composer
        DECLARE @existing_new_composer_id INT;
        SELECT @existing_new_composer_id = id
        FROM Writer
        WHERE Fname = @new_Fname AND Lname = @new_Lname AND id != @composer_id;

        IF @existing_new_composer_id IS NOT NULL
        BEGIN
            PRINT 'A different composer with the new name already exists.';
            RETURN;
        END

        -- Update the Writer table with new name and details
        UPDATE Writer
        SET Fname = @new_Fname, Lname = @new_Lname, genre = @genre, birthYear = @birthYear, deathYear = @deathYear, musGenre_id = @musGenre_id
        WHERE id = @composer_id;

        PRINT 'Composer details updated successfully.';
    END
    ELSE
    BEGIN
        PRINT 'Composer does not exist.';
    END
END;
GO
```



Edit Composer

First Name: Leonard

Last Name: Bernstein

Genre: M

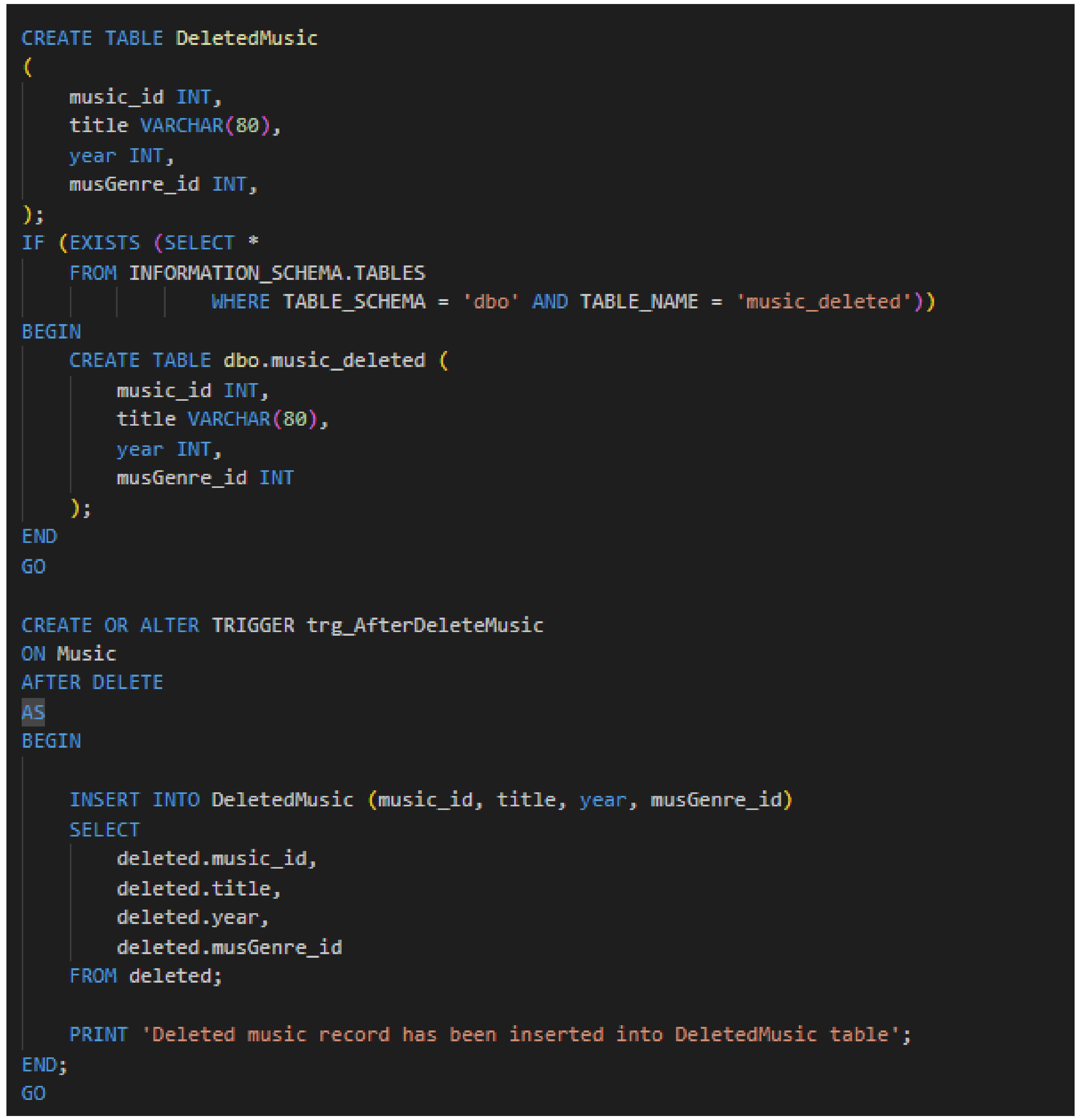
Birth Year: 1918

Death Year: 1990

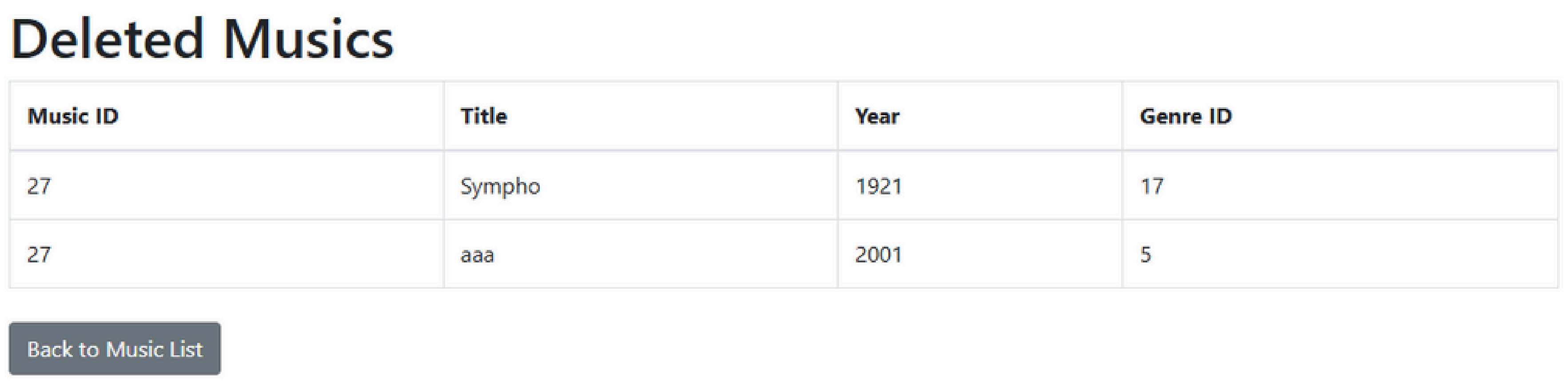
Musical Genre: Contemporary

Trigger

Deleted Musics



```
CREATE TABLE DeletedMusic
(
    music_id INT,
    title VARCHAR(80),
    year INT,
    musGenre_id INT,
);
IF (EXISTS (SELECT *
    FROM INFORMATION_SCHEMA.TABLES
                WHERE TABLE_SCHEMA = 'dbo' AND TABLE_NAME = 'music_deleted'))
BEGIN
    CREATE TABLE dbo.music_deleted (
        music_id INT,
        title VARCHAR(80),
        year INT,
        musGenre_id INT
    );
END
GO

CREATE OR ALTER TRIGGER trg_AfterDeleteMusic
ON Music
AFTER DELETE
AS
BEGIN

    INSERT INTO DeletedMusic (music_id, title, year, musGenre_id)
    SELECT
        deleted.music_id,
        deleted.title,
        deleted.year,
        deleted.musGenre_id
    FROM deleted;

    PRINT 'Deleted music record has been inserted into DeletedMusic table';
END;
GO
```



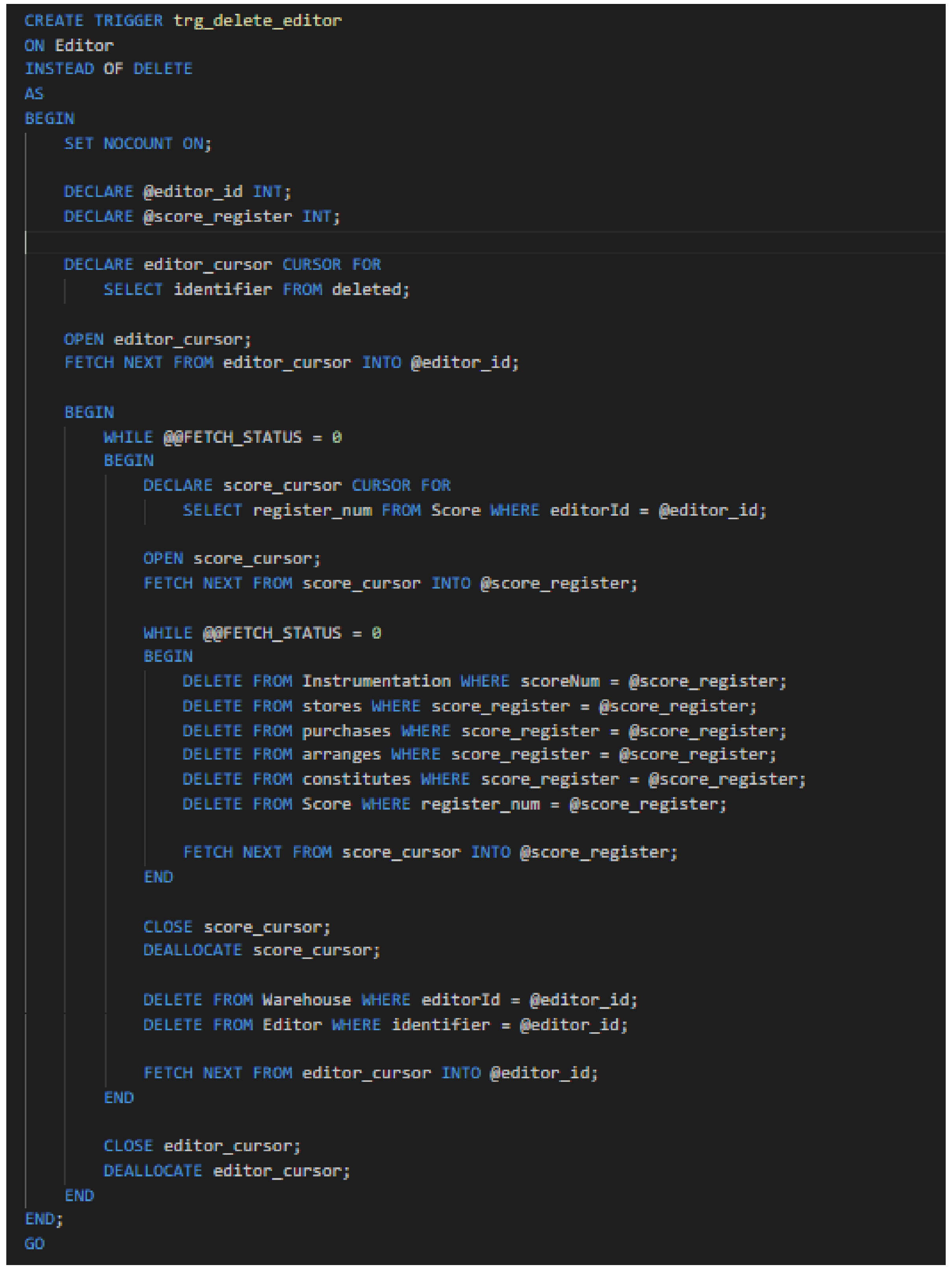
Deleted Musics

Music ID	Title	Year	Genre ID
27	Sympho	1921	17
27	aaa	2001	5

Back to Music List

Trigger

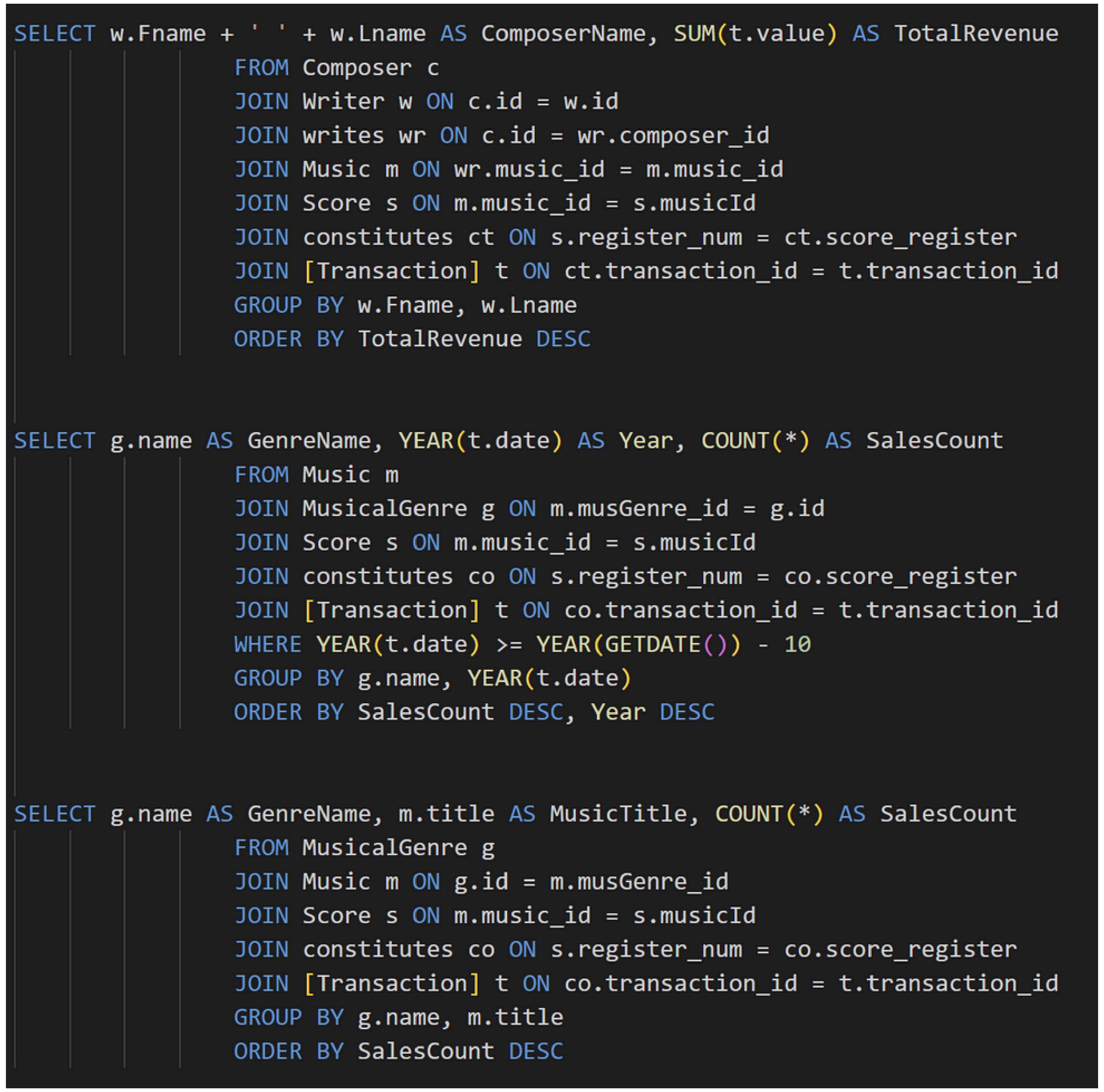
Delete Editor



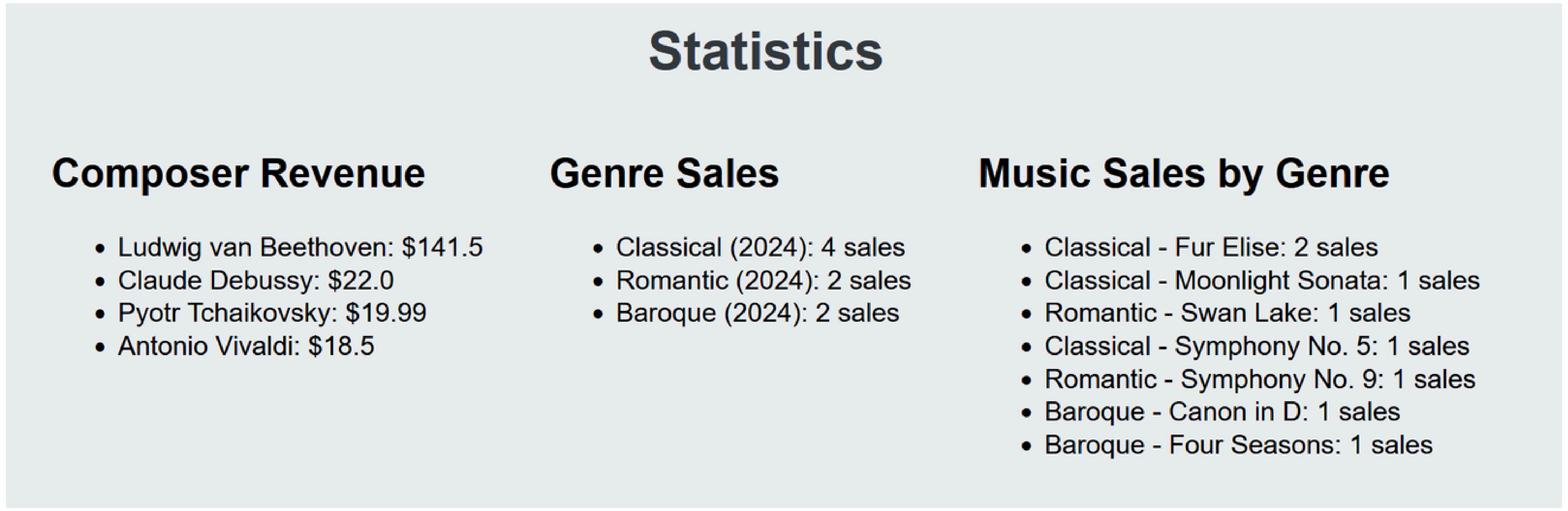
```sql
CREATE TRIGGER trg_delete_editor
ON Editor
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @editor_id INT;
    DECLARE @score_register INT;

    DECLARE editor_cursor CURSOR FOR
        SELECT identifier FROM deleted;

    OPEN editor_cursor;
    FETCH NEXT FROM editor_cursor INTO @editor_id;

    BEGIN
        WHILE @@FETCH_STATUS = 0
        BEGIN
            DECLARE score_cursor CURSOR FOR
                SELECT register_num FROM Score WHERE editorId = @editor_id;

            OPEN score_cursor;
            FETCH NEXT FROM score_cursor INTO @score_register;

            WHILE @@FETCH_STATUS = 0
            BEGIN
                DELETE FROM Instrumentation WHERE scoreNum = @score_register;
                DELETE FROM stores WHERE score_register = @score_register;
                DELETE FROM purchases WHERE score_register = @score_register;
                DELETE FROM arranges WHERE score_register = @score_register;
                DELETE FROM constitutes WHERE score_register = @score_register;
                DELETE FROM Score WHERE register_num = @score_register;

                FETCH NEXT FROM score_cursor INTO @score_register;
            END

            CLOSE score_cursor;
            DEALLOCATE score_cursor;

            DELETE FROM Warehouse WHERE editorId = @editor_id;
            DELETE FROM Editor WHERE identifier = @editor_id;

            FETCH NEXT FROM editor_cursor INTO @editor_id;
        END

        CLOSE editor_cursor;
        DEALLOCATE editor_cursor;
    END
END;
GO
```

QUERYS

Statistics



```sql
SELECT w.Fname + ' ' + w.Lname AS ComposerName, SUM(t.value) AS TotalRevenue
                FROM Composer c
                JOIN Writer w ON c.id = w.id
                JOIN writes wr ON c.id = wr.composer_id
                JOIN Music m ON wr.music_id = m.music_id
                JOIN Score s ON m.music_id = s.musicId
                JOIN constitutes ct ON s.register_num = ct.score_register
                JOIN [Transaction] t ON ct.transaction_id = t.transaction_id
                GROUP BY w.Fname, w.Lname
                ORDER BY TotalRevenue DESC


SELECT g.name AS GenreName, YEAR(t.date) AS Year, COUNT(*) AS SalesCount
                FROM Music m
                JOIN MusicalGenre g ON m.musGenre_id = g.id
                JOIN Score s ON m.music_id = s.musicId
                JOIN constitutes co ON s.register_num = co.score_register
                JOIN [Transaction] t ON co.transaction_id = t.transaction_id
                WHERE YEAR(t.date) >= YEAR(GETDATE()) - 10
                GROUP BY g.name, YEAR(t.date)
                ORDER BY SalesCount DESC, Year DESC


SELECT g.name AS GenreName, m.title AS MusicTitle, COUNT(*) AS SalesCount
                FROM MusicalGenre g
                JOIN Music m ON g.id = m.musGenre_id
                JOIN Score s ON m.music_id = s.musicId
                JOIN constitutes co ON s.register_num = co.score_register
                JOIN [Transaction] t ON co.transaction_id = t.transaction_id
                GROUP BY g.name, m.title
                ORDER BY SalesCount DESC
```



Statistics

Composer Revenue

- Ludwig van Beethoven: $141.5
- Claude Debussy: $22.0
- Pyotr Tchaikovsky: $19.99
- Antonio Vivaldi: $18.5

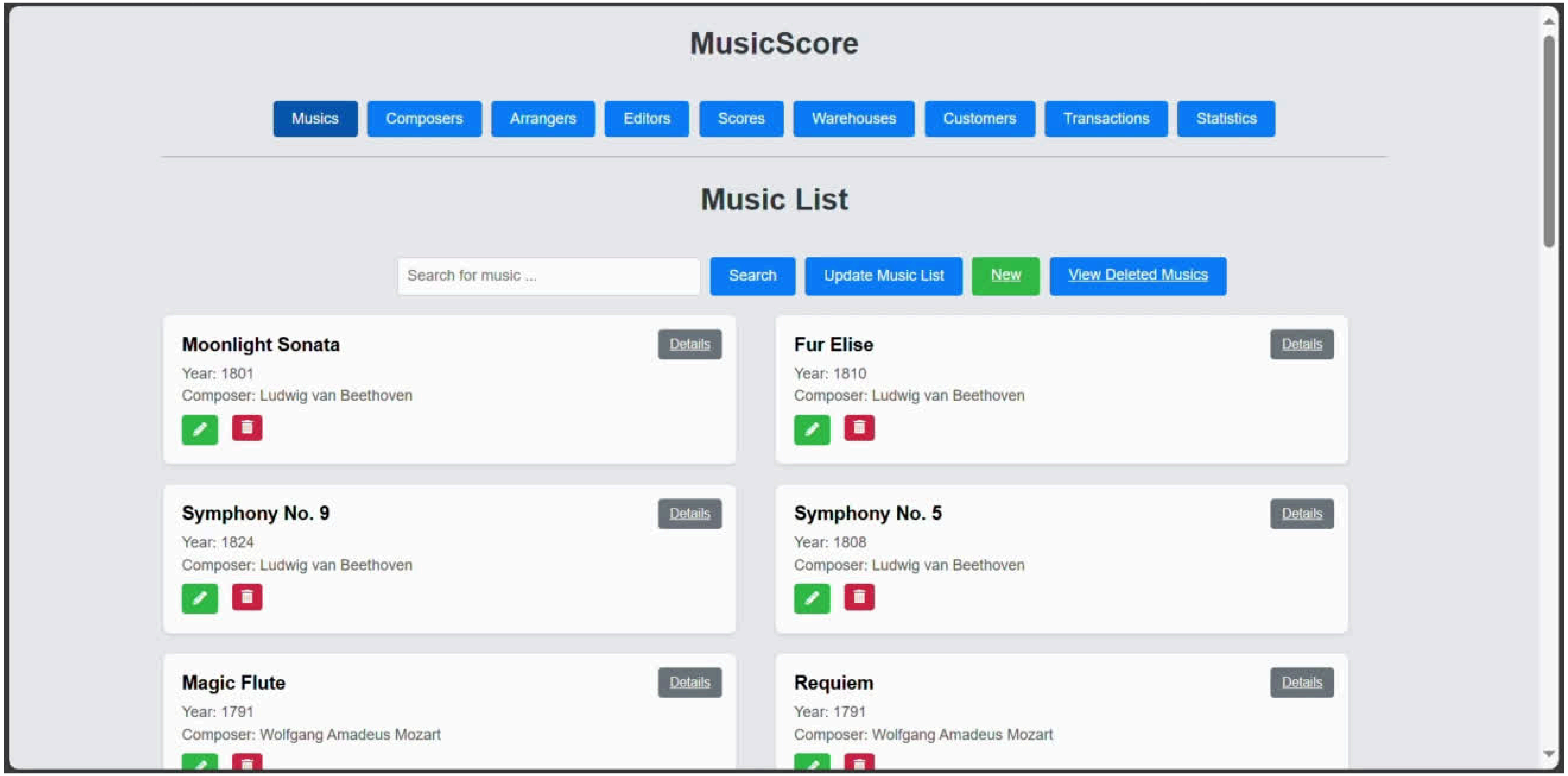
Genre Sales

- Classical (2024): 4 sales
- Romantic (2024): 2 sales
- Baroque (2024): 2 sales

Music Sales by Genre

- Classical - Fur Elise: 2 sales
- Classical - Moonlight Sonata: 1 sales
- Romantic - Swan Lake: 1 sales
- Classical - Symphony No. 5: 1 sales
- Romantic - Symphony No. 9: 1 sales
- Baroque - Canon in D: 1 sales
- Baroque - Four Seasons: 1 sales

Video Demonstration



Music List

Moonlight Sonata
Year: 1801
Composer: Ludwig van Beethoven

Fur Elise
Year: 1810
Composer: Ludwig van Beethoven

Symphony No. 9
Year: 1824
Composer: Ludwig van Beethoven

Symphony No. 5
Year: 1808
Composer: Ludwig van Beethoven

Magic Flute
Year: 1791
Composer: Wolfgang Amadeus Mozart

Requiem
Year: 1791
Composer: Wolfgang Amadeus Mozart