```
/*
Project 2 Group 5

Bala Gaurav Reddy Pasam
Khushbu Singh
Raja Krishna Srivastav Arra
Atlaf Khan

*/
-- Schema for the Stockton Symphony Database

-- Drop the database if it exists and create a new one
DROP DATABASE IF EXISTS StocktonSymphonyDB;
GO

CREATE DATABASE StocktonSymphonyDB;
GO

USE StocktonSymphonyDB;
GO

-- Create the Concerts table
-- This table stores information about all concerts, including the name, date,
  time, and venue.
CREATE TABLE Concerts (
    ConcertID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each concert.
    Name VARCHAR(255) NOT NULL CHECK (LEN(Name) > 0), -- Name of the concert.
    Date DATE NOT NULL, -- Date when the concert will take place.
    Time TIME NOT NULL, -- Time of the concert.
    Day VARCHAR(20) CHECK (Day IN ('Monday', 'Tuesday', 'Wednesday', 'Thursday',
      'Friday', 'Saturday', 'Sunday')), -- Day of the week.
    SeriesType VARCHAR(50) CHECK (SeriesType IN ('Pops', 'Classical', 'Jazz',
      'Other')), -- Series type classification (e.g., Pops or Classical).
    SeriesNumber INT CHECK (SeriesNumber > 0), -- Sequence number within a series.
    VenueName VARCHAR(255) NOT NULL CHECK (LEN(VenueName) > 0) -- Name of the venue
       where the concert is held.
);
GO

-- Create the StocktonCustomers table
-- This table stores customer details, such as their name, age, and subscription
  details.
CREATE TABLE StocktonCustomers (
    CustomerID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each customer.
    Name VARCHAR(255) NOT NULL CHECK (LEN(Name) > 0), -- Full name of the customer.
    Age INT CHECK (Age > 0), -- Age of the customer.
    SubscriptionDetails VARCHAR(50) DEFAULT 'Unsubscribed' CHECK
      (SubscriptionDetails IN ('Gold', 'Silver', 'Unsubscribed')), -- Subscription
      tier.
```

```sql
    LoyaltyScore INT DEFAULT 0 CHECK (LoyaltyScore >= 0), -- Numeric score
      representing customer loyalty.
    PastAttendance INT DEFAULT 0 CHECK (PastAttendance >= 0) -- Number of concerts
      attended in the past.
);
GO

-- Create the StocktonTickets table
-- This table stores information about tickets purchased for concerts.
CREATE TABLE StocktonTickets (
    TicketID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each ticket.
    ConcertID INT NOT NULL, -- References the concert the ticket is for.
    CustomerID INT NOT NULL, -- References the customer who purchased the ticket.
    TicketType VARCHAR(50) NOT NULL CHECK (TicketType IN ('Single',
      'Subscription')), -- Type of ticket (single or subscription).
    FOREIGN KEY (ConcertID) REFERENCES Concerts(ConcertID), -- Foreign key linking
      to the Concerts table.
    FOREIGN KEY (CustomerID) REFERENCES StocktonCustomers(CustomerID) -- Foreign
      key linking to the StocktonCustomers table.
);
GO

-- Create the StocktonRevenue table
-- This table records revenue details for concerts, including ticket sales and
  revenue.
CREATE TABLE StocktonRevenue (
    RevenueID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each revenue record.
    ConcertID INT NOT NULL, -- References the concert for which the revenue is
      recorded.
    SingleTicketsSold INT CHECK (SingleTicketsSold >= 0), -- Number of single
      tickets sold.
    SingleTicketRevenue DECIMAL(10, 2) CHECK (SingleTicketRevenue >= 0), -- Revenue
       from single tickets.
    SubTicketsSold INT CHECK (SubTicketsSold >= 0), -- Number of subscription
      tickets sold.
    SubRevenue DECIMAL(10, 2) CHECK (SubRevenue >= 0), -- Revenue from subscription
       tickets.
    TotalRevenue AS (SingleTicketRevenue + SubRevenue), -- Automatically calculated
       total revenue.
    FOREIGN KEY (ConcertID) REFERENCES Concerts(ConcertID) -- Foreign key linking
      to the Concerts table.
);
GO

-- Create the Composers table
-- This table stores information about composers whose pieces are performed in
  concerts.
CREATE TABLE Composers (
    ComposerID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each composer.
    Name VARCHAR(255) NOT NULL CHECK (LEN(Name) > 0), -- Full name of the composer.
    Country VARCHAR(100) NOT NULL CHECK (LEN(Country) > 0) -- Country of origin of
      the composer.
);
GO

-- Create the PerformedPieces table
-- This table stores details about the musical pieces performed in concerts.
CREATE TABLE PerformedPieces (
    PieceID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each performed piece.
    ConcertID INT NOT NULL, -- References the concert in which the piece was
      performed.
    ComposerID INT NOT NULL, -- References the composer of the piece.
    PieceName VARCHAR(255) NOT NULL CHECK (LEN(PieceName) > 0), -- Name of the
      musical piece.
    PerformanceDate DATE NOT NULL, -- Date when the piece was performed.
    FOREIGN KEY (ConcertID) REFERENCES Concerts(ConcertID), -- Foreign key linking
      to the Concerts table.
    FOREIGN KEY (ComposerID) REFERENCES Composers(ComposerID) -- Foreign key
      linking to the Composers table.
);
GO

-- Create the StocktonFeedback table
-- This table stores customer feedback on concerts they attended.
CREATE TABLE StocktonFeedback (
    FeedbackID INT IDENTITY(1,1) PRIMARY KEY, -- Unique ID for each feedback entry.
    CustomerID INT NOT NULL, -- References the customer providing the feedback.
    TicketID INT NOT NULL, -- References the ticket for the concert.
    ConcertName VARCHAR(255) NOT NULL CHECK (LEN(ConcertName) > 0), -- Name of the
      concert the feedback is about.
    FavouritePiece VARCHAR(255), -- Name of the customer's favorite piece from the
      concert.
    Comments TEXT, -- Additional comments or feedback from the customer.
    FOREIGN KEY (CustomerID) REFERENCES StocktonCustomers(CustomerID), -- Foreign
      key linking to the StocktonCustomers table.
    FOREIGN KEY (TicketID) REFERENCES StocktonTickets(TicketID) -- Foreign key
      linking to the StocktonTickets table.
);
GO
```