IMAGE CRETAED WITH HTTPS://WWW.CRAIYON.COM/

Music_festival.db

At a vibrant music festival named "Harmony Fest" (Table: Festival, Field: Name), held in a spacious venue with multiple stages (Table: Stages, Fields: Name, Location) in a beautiful outdoor setting, organizers prepare for a weekend filled with music, excitement, and entertainment.

On the main stage, a variety of talented artists from different musical genres (Table: Artists, Field: Genre), ranging from rock to electronic, are scheduled to perform throughout the event. Attendees (Table: Attendees, Fields: Name, Age), thrilled by the diversity of artists, arrive at the festival with their tickets in hand and are greeted by an access control team

that records their entry and assigns seats based on the purchased ticket (Table: Tickets, Fields: SeatNumber).

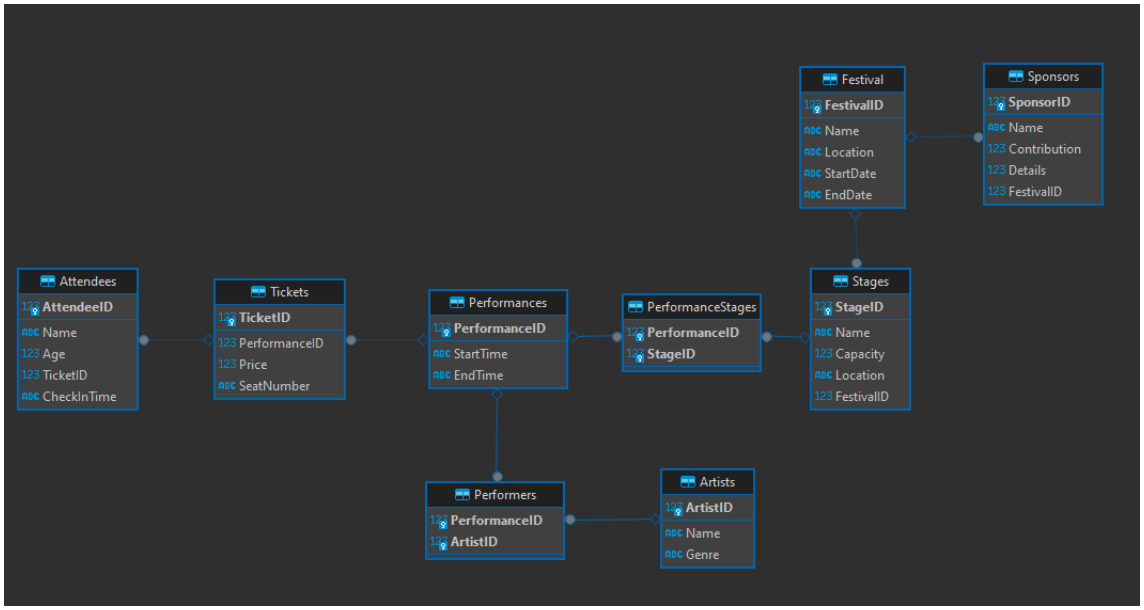
Meanwhile, backstage, festival sponsors (Table: Sponsors, Field: Name), with their generous financial contributions (Table: Sponsors, Field: Contribution), are eager to promote their brands and products through various activations and interactive experiences (Table: Sponsors, Field: Details).

With the start of each performance (Table: Performances, Fields: StartTime), spectators fill the designated areas in front of the stages, while others enjoy the music from the comfort of their assigned seats (Table: Tickets, Fields: Price).

Artists, supported by an expert production team, deliver memorable performances that delight the crowd and create unforgettable moments (Table: Performers). As the festival comes to a close, both attendees and artists leave with lasting memories of an unforgettable musical experience at "Harmony Fest" (Table: Festival, Field: Name).

Contenido

Entity-Relationship Diagram



Artists:

Columns:

- ArtistID: Unique identifier for the artist (INTEGER, PRIMARY KEY).
- Name: Name of the artist (TEXT, cannot be null).
- Genre: Genre of the artist's music (TEXT).

Definition:

```sql
-- Create tables
CREATE TABLE Artists (
    ArtistID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Genre TEXT
);
```

Example:

Artists | Enter a SQL expression to filter results (use Ctrl+Space)

	ArtistID	Name	Genre
1	1	John Doe Band	Rock
2	2	The Groove Masters	Jazz
3	3	Electro Beats	Electronic
4	4	Soulful Sisters	Soul
5	5	The Country Crew	Country

Performances:

Columns:

- PerformanceID: Unique identifier for the performance (INTEGER, PRIMARY KEY).
- StartTime: Start time of the performance (TIMESTAMP).
- EndTime: End time of the performance (TIMESTAMP).

Definition:

```sql
CREATE TABLE Performances (
    PerformanceID INTEGER PRIMARY KEY,
    StartTime TIMESTAMP,
    EndTime TIMESTAMP
);
```

Example:

Performances | Enter a SQL expression to filter results (use Ctrl+Space)

	PerformanceID	StartTime	EndTime
1	1	2024-07-10 18:00:00	2024-07-10 20:00:00
2	2	2024-07-10 20:30:00	2024-07-10 22:30:00
3	3	2024-07-11 19:00:00	2024-07-11 21:00:00
4	4	2024-07-12 20:00:00	2024-07-12 22:00:00
5	5	2024-07-13 21:00:00	2024-07-13 23:00:00

Stages:

Columns:

- StageID: Unique identifier for the stage (INTEGER, PRIMARY KEY).
- Name: Name of the stage (TEXT).
- Capacity: Capacity of the stage (INTEGER).
- Location: Location of the stage (TEXT).
- FestivalID: ID of the festival the stage belongs to, foreign key referencing the Festival table (INTEGER).

Definition:

```sql
CREATE TABLE Stages (
    StageID INTEGER PRIMARY KEY,
    Name TEXT,
    Capacity INTEGER,
    Location TEXT,
    FestivalID INTEGER,
    FOREIGN KEY (FestivalID) REFERENCES Festival(FestivalID)
);
```

Example:

Stages | Enter a SQL expression to filter results (use Ctrl+Space)

	StageID	Name	Capacity	Location	FestivalID
1	1	Main Stage	5.000	Central Park	1
2	2	Jazz Lounge	300	Downtown	1
3	3	EDM Arena	2.000	City Center	1
4	4	Soulful Corner	500	Waterfront	1
5	5	Country Barn	1.000	Outskirts	1

Attendees:

Columns:

- AttendeeID: Unique identifier for the attendee (INTEGER, PRIMARY KEY).
- Name: Name of the attendee (TEXT, cannot be null).
- Age: Age of the attendee (INTEGER).
- TicketID: ID of the ticket associated with the attendee, foreign key referencing the Tickets table (INTEGER).
- CheckInTime: Time when the attendee checked in (TIMESTAMP).

Definition:

```sql
CREATE TABLE Attendees (
    AttendeeID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Age INTEGER,
    TicketID INTEGER,
    CheckInTime TIMESTAMP,
    FOREIGN KEY (TicketID) REFERENCES Tickets(TicketID)
);
```

Example:

Attendees | Enter a SQL expression to filter results (use Ctrl+Space)

	AttendeeID	Name	Age	TicketID	CheckInTime
1	1	Alice Smith	28	1	2024-07-10 17:30:00
2	2	Bob Johnson	35	2	2024-07-10 20:00:00
3	3	Charlie Brown	22	3	2024-07-11 18:30:00
4	4	Diana Ross	45	4	2024-07-12 19:45:00
5	5	Eva Green	30	5	2024-07-13 20:30:00

Tickets:

Columns:

- TicketID: Unique identifier for the ticket (INTEGER, PRIMARY KEY).
- PerformanceID: ID of the performance the ticket is for, foreign key referencing the Performances table (INTEGER).
- Price: Price of the ticket (REAL).
- SeatNumber: Seat number assigned to the ticket (TEXT).

Definition:

```sql
CREATE TABLE Tickets (
    TicketID INTEGER PRIMARY KEY,
    PerformanceID INTEGER,
    Price REAL,
    SeatNumber TEXT,
    FOREIGN KEY (PerformanceID) REFERENCES Performances(PerformanceID)
);
```

Example:

Tickets | Enter a SQL expression to filter results (use Ctrl+Space)

	TicketID	PerformanceID	Price	SeatNumber
1	1	1	50	A101
2	2	2	40	B203
3	3	3	35	C305
4	4	4	55	D102
5	5	5	45	E204

Sponsors:

Columns:

- SponsorID: Unique identifier for the sponsor (INTEGER, PRIMARY KEY).
- Name: Name of the sponsor (TEXT).
- Contribution: Contribution amount from the sponsor (REAL).
- Details: Details about the sponsorship, stored as JSON.
- FestivalID: ID of the festival the sponsor is associated with, foreign key referencing the Festival table (INTEGER).

Definition:

```sql
CREATE TABLE Sponsors (
    SponsorID INTEGER PRIMARY KEY,
    Name TEXT,
    Contribution REAL,
    Details JSON,
    FestivalID INTEGER,
    FOREIGN KEY (FestivalID) REFERENCES Festival(FestivalID)
);
```

Example:

Sponsors | Enter a SQL expression to filter results (use Ctrl+Space)

	SponsorID	Name	Contribution	Details	FestivalID
1	1	BigCorp	10.000	{"Level": "Platinum"}	1
2	2	MegaCorp	7.500	{"Level": "Gold"}	1
3	3	SuperCorp	5.000	{"Level": "Silver"}	1
4	4	AwesomeCorp	2.500	{"Level": "Bronze"}	1
5	5	TechCorp	1.000	{"Level": "Supporter"}	1

Festival:

Columns:

- FestivalID: Unique identifier for the festival (INTEGER, PRIMARY KEY).
- Name: Name of the festival (TEXT).
- Location: Location of the festival (TEXT).
- StartDate: Start date of the festival (TIMESTAMP).
- EndDate: End date of the festival (TIMESTAMP).

Definition:

```sql
CREATE TABLE Festival (
    FestivalID INTEGER PRIMARY KEY,
    Name TEXT,
    Location TEXT,
    StartDate TIMESTAMP,
    EndDate TIMESTAMP
);
```

Example:

Festival | Enter a SQL expression to filter results (use Ctrl+Space)

	FestivalID	Name	Location	StartDate	EndDate
1	1	Summer Music Festival	Citywide	2024-07-10	2024-07-13

Performers:

Columns:

- PerformanceID: ID of the performance, part of the composite primary key, foreign key referencing the Performances table (INTEGER).
- ArtistID: ID of the artist performing, part of the composite primary key, foreign key referencing the Artists table (INTEGER).

Definition:

```sql
CREATE TABLE Performers (
    PerformanceID INTEGER,
    ArtistID INTEGER,
    PRIMARY KEY (PerformanceID, ArtistID),
    FOREIGN KEY (PerformanceID) REFERENCES Performances(PerformanceID),
    FOREIGN KEY (ArtistID) REFERENCES Artists(ArtistID)
);
```

Example:

Performers | Enter a SQL expression to filter results (us

	PerformanceID	ArtistID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

PerformaceStages:

Columns:

- PerformanceID: ID of the performance, part of the composite primary key, foreign key referencing the Performances table (INTEGER).
- StageID: ID of the stage where the performance takes place, part of the composite primary key, foreign key referencing the Stages table (INTEGER).

Definition:

```sql
CREATE TABLE PerformanceStages (
    PerformanceID INTEGER,
    StageID INTEGER,
    PRIMARY KEY (PerformanceID, StageID),
    FOREIGN KEY (PerformanceID) REFERENCES Performances(PerformanceID),
    FOREIGN KEY (StageID) REFERENCES Stages(StageID)
);
```

Example:

PerformanceStages | Enter a SQL expression to filter r

	PerformanceID	StageID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

Query simples

-- Retrieve all performances with their associated artists and stages:

```
SELECT p.PerformanceID, a.Name AS Artist, s.Name AS Stage
FROM Performances p
INNER JOIN Performers pf ON p.PerformanceID = pf.PerformanceID
INNER JOIN Artists a ON pf.ArtistID = a.ArtistID
INNER JOIN PerformanceStages ps ON p.PerformanceID = ps.PerformanceID
INNER JOIN Stages s ON ps.StageID = s.StageID;
```

-- List all attendees who checked in after a specified time:

```
SELECT *
FROM Attendees
WHERE CheckInTime > '2024-07-11 18:00:00';
```

-- Calculate the total contribution from sponsors for the festival:

```
SELECT SUM(Contribution) AS TotalContribution
FROM Sponsors;
```

-- Find out the average ticket price for the performances:

```
SELECT AVG(Price) AS AverageTicketPrice
FROM Tickets;
```

-- Retrieve the name and location of the festival along with the start and end dates:

```
SELECT Name, Location, StartDate, EndDate
FROM Festival;
```

-- query that uses Common Table Expressions (CTEs) and joins to retrieve information about attendees who purchased tickets for performances at specific stages:

```
WITH StagePerformancesCount AS (
  SELECT ps.StageID, COUNT(ps.PerformanceID) AS NumPerformances
  FROM PerformanceStages ps
  GROUP BY ps.StageID
```

```
),
AttendeePerformanceCounts AS (
   SELECT a.AttendeeID, COUNT(t.PerformanceID) AS NumPerformancesAttended
   FROM Attendees a
   LEFT JOIN Tickets t ON a.TicketID = t.TicketID
   GROUP BY a.AttendeeID
)
SELECT a.Name AS AttendeeName, s.Name AS StageName, spc.NumPerformances AS
TotalPerformancesAtStage,
     apc.NumPerformancesAttended AS PerformancesAttended
FROM Attendees a
JOIN Tickets t ON a.TicketID = t.TicketID
JOIN Performances p ON t.PerformanceID = p.PerformanceID
JOIN PerformanceStages ps ON p.PerformanceID = ps.PerformanceID
JOIN Stages s ON ps.StageID = s.StageID
JOIN StagePerformancesCount spc ON s.StageID = spc.StageID
JOIN AttendeePerformanceCounts apc ON a.AttendeeID = apc.AttendeeID
WHERE apc.NumPerformancesAttended > spc.NumPerformances / 2;
```