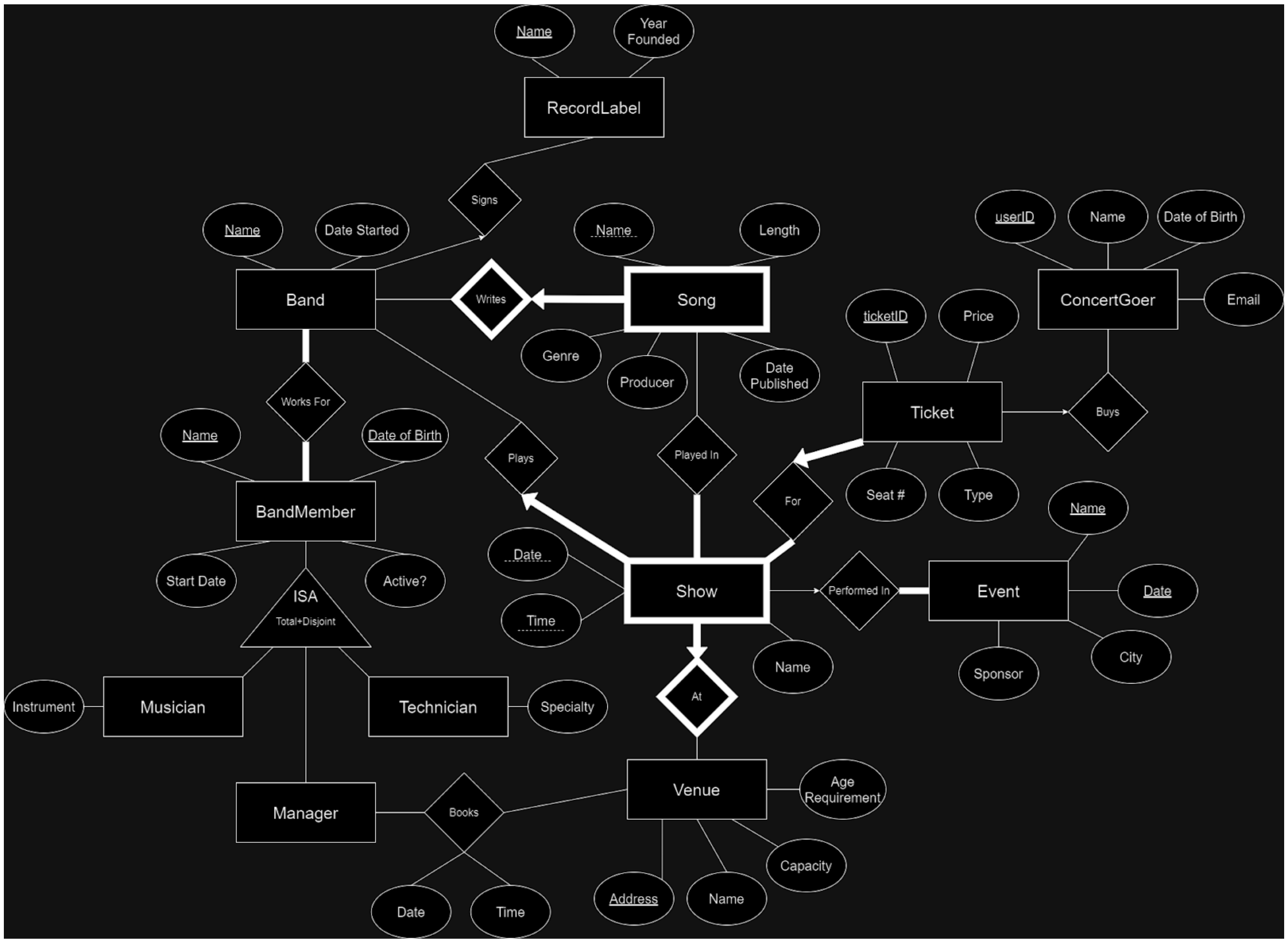
Summary

Our project aims to be a way for people interested in seeing bands perform to find performances that they would like to buy tickets for. For example, if someone is a fan of some band, they can use this application to look up upcoming shows by the band in their city, including information about venue, setlist, band members, etc. It also allows for potential performers to publish upcoming shows for concert goers to buy tickets for.

ER Diagram

Please zoom in if you need to!



Major Changes

- BandMember ISA has been overhauled. All types of musicians fall into one entity set, and a BandMember now need not be a musician. The ISA is now total + disjoint rather than overlap.
- Show now only has one parent entity, Venue. Show's PK is now (venueAddress, showDate, showTime).
- Removed RecordLabel DISTRIBUTES Song to simplify our project by removing redundancy, as RecordLabel can still be determined from the Song's Band.
- Venue has a new relationship, BOOKS, containing information about a venue booking that a manager of a band would make.
- Show PERFORMED IN Event is now a many-to-one relationship.

Minor Changes

- eventCity is no longer a part of Event's key.
- Tickets now contain information about the type of seat they are sold for (e.g. floor, lower-level, etc.).
- BandMembers may store the date they joined the band.
- WorksFor has a boolean field called active? that specifies if the BandMember in the relationship is active in its corresponding Band.
- Songs may have a producer listed.
- Shows now must have a name listed.
- ConcertGoer now must have an email

Schemas

Underlined attributes are Primary Keys

Bold attributes are Foreign Keys

Entities

Band(bandName, dateStarted, **recordLabelName**)

- recordLabelName references RecordLabel (may be null)
- dateStarted is DATE type
- All other attributes are strings

Song(songName, **bandName,** length, genre, producer, date)

- bandName references Band (not null - implied by being part of PK)
- length is not null
- date is DATE type
- All other attributes are strings

Show(**venueAddress**, showDate, showTime, showName, **bandName**, **eventName, eventDate**)

- bandName references Band (not null - participation constraint)
- venueAddress references Venue (not null - implied by being part of PK)
- eventName, eventDate references Event (may be null)
- showDate, eventDate are DATE types
- All other attributes are strings
    - showTime is length (00:00)

RecordLabel(recordLabelName, yearFounded)

- recordLabelName is string
- yearFounded is int > 0

BandMember(memberName, memberDOB, startDate)

- memberDOB, startDate is DATE type
- memberName is string

Musician(**memberName, memberDOB,** instrument)

- memberName, memberDOB references BandMember (not null - implied by being part of PK
- instrument is string, not null (e.g. "guitar")

Manager(**memberName, memberDOB**)

- memberName, memberDOB references BandMember (not null - implied by being part of PK

Technician(**memberName, memberDOB,** specialty)

- memberName, memberDOB references BandMember (not null - implied by being part of PK
- specialty is string, not null (e.g. "audio")

Event(eventName, eventDate, city, sponsor)

- eventDate is DATE type
- city is not null
- All other attributes are strings

Venue(venueAddress, venueName, capacity, ageReq)

- venueName is not null
- Capacity is not null
- ageReq may be null (null implies no age requirement)
- venueAddress, venueName are strings
- capacity, ageReq are ints, both > 0

ConcertGoer(userID, goerName, email, dob)

- goerName is not null
- email is unique and not null
- dob is optional to include (including allows user access to age restricted shows)
- userID is int > 0
- goerName is string
- dob is DATE type

Ticket(ticketID, seatNum, price, type, **userID**, **venueAddress**, **showDate, showTime**)

- userID references ConcertGoer (may be null)
- venueAddress, showDate, showTime references Show (not null)
- seatNum, price, type are not null
- seatNum is a code (string) in the form "type + row + number"
    - E.g. a lower-level seat row 3 number 25 might be "L0325"
- ticketID, userID are strings

- price is floating point (two decimal places) >= 0.00
- showDate is DATE type
- All other attributes are strings
    - showTime is length 5 ("00:00")

Relations

WorksFor(**memberName**, **memberDOB**, **bandName**, active?)

- memberName, memberDOB references BandMember (not null - implied by being part of PK)
- bandName references Band (not null - implied by being part of PK)
- active? is one string character ("y"/"n")
- All other attributes are strings

Books(**memberName**, **memberDOB**, **venueAddress**, bookingDate, bookingTime)

- memberName, memberDOB references Manager (not null - implied by being part of PK)
- venueAddress references Venue (not null - implied by being part of PK)
- (venueAddress bookingDate, bookingTime) must be unique
    - Prevents overlapping bookings from different managers at the same venue
- All attributes are strings, except for memberDOB, bookingDate which is DATE

PlayedIn(**songName, venueAddress, showDate, showTime**)

- songName references Song (not null - implied by being part of PK)
- venueAddress, showDate, showTime references Show (not null - implied by being part of PK)
- All attributes are strings, except for showDate which is DATE

Functional Dependencies

Band

bandName → dateStarted

bandName → recordLabelName

Song

songName, bandName → length

songName, bandName → genre

songName, bandName → producer

songName, bandName → date

Show

venueAddress, showDate, showTime → showName

venueAddress, showDate, showTime → bandName

venueAddress, showDate, showTime → eventName

venueAddress, showDate, showTime → eventDate

RecordLabel

recordLabelName → yearFounded

BandMember

memberName, memberDOB → startDate

Musician

memberName, memberDOB → instrument

Technician

memberName, memberDOB → specialty

Event

eventName, eventDate → sponsor

eventName, eventDate → city

Venue

venueAddress → venueName

venueAddress → capacity

venueAddress → ageReq

ConcertGoer

userID → goerName

userID → email

userID → dob

Ticket

ticketID → seatNum

ticketID → price

ticketID → type

ticketID → userID

ticketID → venueAddress

ticketID → showDate

ticketID → showTime

seatNum, venueAddress, showDate, showTime → price

seatNum → type

WorksFor

memberName, memberDOB, bandName → active?

Books

memberName, memberDOB, venueAddress → bookingDate

memberName, memberDOB, venueAddress → bookingTime

Normalization

Our project has two FDs that violate BCNF and 3NF, found in the Ticket schema.

FDs:

- ticketID → seatNum, price, type, userID, venueAddress, showDate, showTime
- seatNum, venueAddress, showDate, showTime → price, type
- seatNum → type

Closures:

- [ticketID]$^+$ = {ticketID, seatNum, price, type, userID, venueAddress, showDate, showTime}
- [seatNum]+ = {seatNum, type}
- [seatNum, venueAddress, showDate, showTime]$^+$ = {seatNum, venueAddress, showDate, showTime, price}

Key is ticketID.

The FD [seatNum, venueAddress, showDate, showTime → price] violates BCNF, since the LHS is not a superkey. It also violates 3NF since price is not a part of the key (ticketID).

We decompose into two BCNF tables:

- TicketPrice(<u>seatNum</u>, <u>**venueAddress,**</u> <u>**showDate**</u>, <u>**showTime**</u>, price)
  - No violations
  - price is not null
  - venueAddress, showDate, showTime references Show (not null)
- TicketID(<u>ticketID</u>, seatNum, type, **userID**, **venueAddress**, **showDate, showTime**)
  - seatNum is not null
  - type is not null
  - userID references ConcertGoer (may be null)
  - venueAddress, showDate, showTime references Show (not null)

The FD [seatNum → type] violates BCNF in TicketID, so we will revise TicketID and add TicketType through another decomposition.

- TicketID(<u>ticketID</u>, seatNum, **userID**, **venueAddress**, **showDate, showTime**)

    - No violations
    - seatNum is not null
    - userID references ConcertGoer (may be null)
    - venueAddress, showDate, showTime references Show (not null)
- TicketType(<u>seatNum</u>, type)
    - No violations
    - type is not null

Final decomposition:

- TicketID(<u>ticketID</u>, seatNum, **userID**, **venueAddress**, **showDate, showTime**)
- TicketType(<u>seatNum</u>, type)
- TicketPrice(<u>seatNum</u>, <u>**venueAddress,**</u> <u>**showDate**</u>, <u>**showTime**</u>, price)

All other FDs are in BCNF, since all LHS are primary keys.

SQL DDL

Create Tables

```
CREATE TABLE Band (
        bandName            VARCHAR     PRIMARY KEY,
        dateStarted         DATE,
        recordLabelName     VARCHAR,
        FOREIGN KEY (recordLabelName) REFERENCES RecordLabel
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE Song (
        songName            VARCHAR,
        bandName            VARCHAR,
        length              VARCHAR     NOT NULL,
        genre               VARCHAR,
        producer            VARCHAR,
        date                DATE,
        PRIMARY KEY (songName, bandName),
        FOREIGN KEY (bandName) REFERENCES Band
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Show (
        venueAddress        VARCHAR,
        showDate            DATE,
        showTime            CHAR(5),
        showName            VARCHAR,
        bandName            VARCHAR     NOT NULL,
        eventName               VARCHAR,
        eventDate               DATE,
        PRIMARY KEY (venueAddress, showDate, showTime),
        FOREIGN KEY (venueAddress) REFERENCES Venue
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (bandName) REFERENCES Band
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (eventName, eventDate) REFERENCES Event
                ON DELETE SET NULL
                ON UPDATE CASCADE
);

CREATE TABLE RecordLabel (
        recordLabelName                 VARCHAR     PRIMARY KEY,
        yearFounded                     INTEGER,
);

CREATE TABLED BandMember (
        memberName                      VARCHAR,
        memberDOB                       DATE,
        startDate                       DATE,
        PRIMARY KEY (memberName, memberDOB)
);

CREATE TABLE Musician (
        memberName                      VARCHAR,
        memberDOB                       DATE,
        instrument                      VARCHAR     NOT NULL,
        PRIMARY KEY (memberName, memberDOB),
        FOREIGN KEY (memberName, memberDOB) REFERENCES BandMember
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Manager (
        memberName                VARCHAR,
        memberDOB                 DATE,
        PRIMARY KEY (memberName, memberDOB)
        FOREIGN KEY (memberName, memberDOB) REFERENCES BandMember
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Technician (
        memberName                VARCHAR,
        memberDOB                 DATE,
        specialty                 VARCHAR    NOT NULL,
        PRIMARY KEY (memberName, memberDOB)
        FOREIGN KEY (memberName, memberDOB) REFERENCES BandMember
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Event (
        eventName           VARCHAR,
        eventDate           DATE,
        city                VARCHAR    NOT NULL,
        sponsor             VARCHAR,
        PRIMARY KEY (eventName, eventDate)
);
```

```
CREATE TABLE Venue (
        venueAddress        VARCHAR     PRIMARY KEY,
        venueName           VARCHAR     NOT NULL,
        capacity            INTEGER     NOT NULL,
        ageReq              INTEGER,
);

CREATE TABLE ConcertGoer (
        userID              INTEGER     PRIMARY KEY,
        goerName            VARCHAR     NOT NULL,
        email               VARCHAR     UNIQUE      NOT NULL,
        dob                 DATE,
);

CREATE TABLE TicketID (
        ticketID            INTEGER     PRIMARY KEY,
        seatNum             VARCHAR     NOT NULL,
        userID              INTEGER,
        venueAddress        VARCHAR     NOT NULL,
        showDate            DATE        NOT NULL,
        showTime            CHAR(5)     NOT NULL,
        FOREIGN KEY (userID) REFERENCES ConcertGoer
                ON DELETE SET NULL
                ON UPDATE CASCADE,
        UNIQUE (seatNum, venueAddress, showDate, showTime)
);

CREATE TABLE TicketType (
        seatNum             CHAR(5)     PRIMARY KEY,
        type                VARCHAR     NOT NULL
```

```
);

CREATE TABLE TicketPrice (
        seatNum             CHAR(5),
        venueAddress        VARCHAR,
        showDate            DATE,
        showTime            CHAR(5),
        price               REAL         NOT NULL,
        PRIMARY KEY (seatNum, venueAddress, showDate, showTime),
        FOREIGN KEY (venueAddress, showDate, showTime) REFERENCES Show
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE WorksFor (
        memberName          VARCHAR,
        memberDOB           DATE,
        bandName            VARCHAR,
        active?             CHAR(1)      NOT NULL,
        PRIMARY KEY (memberName, memberDOB, bandName),
        FOREIGN KEY (memberName, memberDOB) REFERENCES BandMember
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY (bandName) REFERENCES Band
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Books (
        memberName          VARCHAR,
        memberDOB           DATE,
        venueAddress        VARCHAR,
        bookingDate         DATE            NOT NULL,
        bookingTime         CHAR(5)         NOT NULL,
        UNIQUE(venueAddress, bookingDate, bookingTime),
        PRIMARY KEY (memberName, memberDOB, venueAddress),
        FOREIGN KEY (memberName, memberDOB) REFERENCES Manager
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY (venueAddress) REFERENCES Venue
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE PlayedIn (
        songName            VARCHAR,
        bandName            VARCHAR,
        venueAddress        VARCHAR,
        showDate            DATE,
        showTime            CHAR(5),
        PRIMARY KEY (songName, bandName, venueAddress, showDate, showTime),
        FOREIGN KEY (songName, bandName) REFERENCES Song
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (venueAddress, showDate, showTime) REFERENCES Show
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Insert

```
INSERT INTO RecordLabel
VALUES ("RCA Records", 1900),
        ("Universal Music", 1940),
        ("Warner Records", 1950),
        ("Sony Music Entertainment", 1920),
        ("Atlantic Records", 2011)

INSERT INTO Band
VALUES ("Arctic Monkeys", 2006-02-10, "RCA Records"),
        ("Radiohead", 1996-02-11, "RCA Records"),
        ("Joji", 2005-07-02, "Universal Music"),
        ("Kiss", 1975-01-01, "Warner Records"),
        ("Super Cool Indie Band", 1920-04-05, NULL)

INSERT INTO Song
VALUES ("Do I Wanna Know?", "Arctic Monkeys", "4:34", "Indie Rock", "James Ford",
2013-06-19),
        ("505", "Arctic Monkeys", "4:13", "Indie Rock", NULL, 2007-04-23),
        ("Glimpse of Us", "Joji", "3:53", "Lo-fi", "Connor McDonough", 2022-06-10),
        ("Weird Fishes/Arpeggi", "Radiohead", "5:19", "Alternative Rock", "Nigel Godrich",
2007-10-10),
        ("Super Epic Song", "Super Cool Indie Band", "7:59", "Jazz", "Steve Steves",
1977-01-10)

INSERT INTO BandMember
VALUES ("Alex Turner", 1991-02-03, 2010-03-03),
        ("Matt Helders", 1971-04-03, 1993-01-03),
        ("Jamie Cook", 1952-03-03, 1981-02-03),
        ("Nick O'Malley", 1933-02-03, 1961-03-03),
        ("George Kusunoki", 1994-01-03, 2021-02-03).
        ("Thom Yorke", 1955-03-03, 1980-10-03),
        ("Jonny Greenwood", 1937-02-03, 1971-05-03),
        ("Colin Greenwood", 1991-01-03, NULL),
        ("Ed O'Brien", 1990-02-03, 2023-04-03),
        ("Philip Selway", 1970-04-03, 2022-02-03),
        ("John Manager", 1950-03-03, 1981-02-03),
        ("Tom Manager", 1930-02-03, 1991-02-03),
        ("Jim Manager", 1999-04-33, NULL),
        ("Tim Manager", 1888-12-19, NULL),
        ("Jeff Manager", 2003-02-01, NULL),
        ("Super Man", 1990-01-03, NULL),
        ("John Technician", 1971-03-02, 1991-02-03),
        ("John Technician", 1972-04-03, NULL),
        ("Jim Technician", 1998-11-11, NULL),
        ("Tom Technician", 1997-12-31, NULL),
        ("Tim Technician", 2001-09-11, 2018-02-02)

INSERT INTO Musician
VALUES ("Alex Turner", 1991-02-03, "Guitar"),
        ("Jonny Greenwood", 1937-02-03, "Keyboard"),
        ("Nick O'Malley", 1933-02-03, "Bass Guitar"),
        ("George Kusunoki", 1994-01-03, "Vocals"),
        ("Matt Helders", 1971-04-03, "Drums"),
        ("Super Man", 1990-01-03, "Alto Saxophone")

INSERT INTO Manager
VALUES("John Manager", 1950-03-03),
        ("Tom Manager", 1930-02-03),
        ("Jim Manager", 1999-04-33),
        ("Tim Manager", 1888-12-19),
        ("Jeff Manager", 2003-02-01)

INSERT INTO Technician
VALUES ("John Technician", 1971-03-02, "Lighting"),
        ("John Technician", 1972-04-03, "Audio"),
        ("Jim Technician", 1998-11-11, "Audio"),
        ("Tom Technician", 1997-12-31, "Camera"),
        ("Tim Technician", 2001-09-11, "Audio")

INSERT INTO Event
VALUES("Coachella", 2005-01-12, "Indio", "Coca-Cola"),
        ("Calgary Stampede", 2024-07-5, "Calgary", "Honda"),
        ("Lollapalooza", 2010-10-01, "Chicago", NULL),
        ("Winnipeg Folk Festival", 2015-12-04, "Winnipeg", "Red Bull"),
        ("Big Dog Festival", 2043-10-03, "Gotham City", "AWS")

INSERT INTO Venue
VALUES("231 Oak Rd, Vancouver, BC, Canada", "Smooth Grooves", 600, 21),
        ("5054 Rat Ave, New York City, NY, USA", "Rat Jam", 100, 1),
        ("12 Slop St, Calgary, AB, Canada", "The Big Cheese", 200,  NULL),
        ("44 44th St, Toronto, ON, Canada", "Black Hole Guys", 60, 18),
        ("101 Real Rd, Chicago, IL, USA", "Real Ones", 40, 21)

INSERT INTO ConcertGoer
VALUES(1, "Kahn Sert", "iloveconcerts@gmail.com", 1990-01-03),
        (2, "Micheal Joaquin", "mikejoaq@gmail.com", NULL),
        (3, "Micheal Jackson", "123mjalive@gmail.com", 1995-04-30),
        (4, "Kai Fakelastname", "kaikaifakelastname@gmail.com", NULL),
        (5, "Name Five", "name5@gmail.com", 2000-10-03)

INSERT INTO TicketID
```

```
VALUES(1, "F1399", 1, "231 Oak Rd, Vancouver, BC, Canada", 2011-12-30, "18:00")
        (2, "B1298", 1, "5054 Rat Ave, New York City, NY, USA", 2011-12-30, "18:30")
        (3, "U3044", 2, "231 Oak Rd, Vancouver, BC, Canada", 2015-02-13, "20:00")
        (4, "L3401", 3, "44 44th St, Toronto, ON, Canada", 2018-03-31, "19:30")
        (5, "L1328", 4, "101 Real Rd, Chicago, IL, USA", 2023-12-30, "00:00")

INSERT INTO TicketType
VALUES("F1399", "Floor"),
        ("B1298", "Balcony"),
        ("U3044", "Upper"),
        ("L3401", "Lower"),
        ("L1328", "Lower")

INSERT INTO TicketPrice
VALUES ("F1399", "231 Oak Rd, Vancouver, BC, Canada", 2011-12-30, "18:00", 100.00),
        ("B1298", 1, "5054 Rat Ave, New York City, NY, USA", 2011-12-30, "18:30", 50.99),
        ("U3044", 2, "231 Oak Rd, Vancouver, BC, Canada", 2015-02-13, "20:00", 1200.53),
        ("L3401", 3, "44 44th St, Toronto, ON, Canada", 2018-03-31, "19:30", 7.98),
        ("L1328", 4, "101 Real Rd, Chicago, IL, USA", 2023-12-30, "00:00", 154.22)

INSERT INTO Show
VALUES("231 Oak Rd, Vancouver, BC, Canada", 2011-12-30, "18:00", NULL, "Arctic
Monkeys", "Coachella", 2005-01-12),
        ("5054 Rat Ave, New York City, NY, USA", 2011-12-30, "18:30", "The Rat Show",
"Super Cool Indie Band", NULL, NULL),
        ("231 Oak Rd, Vancouver, BC, Canada", 2015-02-13, "20:00", NULL, "Radiohead",
NULL, NULL),
        ("44 44th St, Toronto, ON, Canada", 2018-03-31, "19:30", NULL, "Joji", NULL, NULL),
        ("101 Real Rd, Chicago, IL, USA", 2023-12-30, "00:00", "Midnight Music", "Arctic
Monkeys", NULL, NULL)
```

```
INSERT INTO WorksFor
VALUES ("Alex Turner", 1991-02-03, "Arctic Monkeys", "y"),
        ("Matt Helders", 1971-04-03, "Arctic Monkeys", "y"),
        ("Jamie Cook", 1952-03-03, "Arctic Monkeys", "y"),
        ("Nick O'Malley", 1933-02-03, "Arctic Monkeys", "y"),
        ("George Kusunoki", 1994-01-03, "Joji", "y").
        ("Thom Yorke", 1955-03-03, "Radiohead", "y"),
        ("Jonny Greenwood", 1937-02-03, "Radiohead", "y"),
        ("Colin Greenwood", 1991-01-03, "Radiohead", "y"),
        ("Ed O'Brien", 1990-02-03, "Radiohead", "y"),
        ("Philip Selway", 1970-04-03, "Radiohead", "y"),
        ("John Manager", 1950-03-03, "Kiss", "n"),
        ("John Manager", 1950-03-03, "Arctic Monkeys", "y"),
        ("John Manager", 1950-03-03, "Joji", "y"),
        ("Tom Manager", 1930-02-03, "Radiohead", "y"),
        ("Jim Manager", 1999-04-33, "Super Cool Indie Band", "y"),
        ("Tim Manager", 1888-12-19, "Super Cool Indie Band", "y"),
        ("Jeff Manager", 2003-02-01, "Super Cool Indie Band", "y"),
        ("Super Man", 1990-01-03, "Super Cool Indie Band", "y"),
        ("John Technician", 1971-03-02, "Radiohead", "y"),
        ("John Technician", 1972-04-03, "Radiohead", "n"),
        ("Jim Technician", 1998-11-11, "Joji", "n"),
        ("Tom Technician", 1997-12-31, "Joji", "y"),
        ("Tim Technician", 2001-09-11, "Radiohead", "y")

INSERT INTO Books
VALUES ("John Manager", 1950-03-03, "231 Oak Rd, Vancouver, BC, Canada", 2011-12-30,
"18:00"),
        ("Jim Manager", 1999-04-33, "5054 Rat Ave, New York City, NY, USA", 2011-12-30,
"18:30"),
        ("Tom Manager", 1930-02-03, "231 Oak Rd, Vancouver, BC, Canada", 2015-02-13,
"20:00"),
        ("John Manager", 1950-03-03, "44 44th St, Toronto, ON, Canada", 2018-03-31, "19:30"),
        ("John Manager", 1950-03-03, "101 Real Rd, Chicago, IL, USA", 2023-12-30, "00:00")

INSERT INTO PlayedIn
VALUES ("Do I Wanna Know?", "Arctic Monkeys", "231 Oak Rd, Vancouver, BC, Canada",
2011-12-30, "18:00"),
        ("Do I Wanna Know?", "Arctic Monkeys", "101 Real Rd, Chicago, IL, USA",
2023-12-30, "00:00"),
        ("Weird Fishes/Arpeggi", "Radiohead", "231 Oak Rd, Vancouver, BC, Canada",
2015-02-13, "20:00"),
        ("Super Epic Song", "Super Cool Indie Band", "5054 Rat Ave, New York City, NY,
USA", 2011-12-30, "18:30"),
        ("Glimpse of Us", "Joji", "44 44th St, Toronto, ON, Canada", 2018-03-31, "19:30")
```