CSC 540(001) Database management concepts and Systems

WOLFMEDIA

A MEDIA STREAMING SERVICE

Team - G

- Arun Srinivasan Parthasarathy, apartha4
- Kiron Jayesh, kjayesh
- Adittya Soukarjya Saha, asaha4
- Arun Kumar Ramesh, arames25

Project Report - 2

Assumptions:

1. Users' payments can be done only for the entire month. For instance, if a user registers/subscribes at the end of May, the user will have to pay the subscription for the entire month of May.
2. There can only be one main artist for one song.
3. An artist can belong to only one record label.
4. All songs in the album will be made by the same artist.
5. Royalty payment to songs will be made on the first of every month.
6. The podcast payment is made in lump sum to the podcast hosts. The responsibility of splitting the payment lies among the podcast hosts.
7. There cannot be two songs released by the same artist on the same date.
8. No two albums can have the same name if they have the same artist.
9. Podcasts cannot have the same name if they are hosted by the same set of hosts.
10. PodcastHost payments are done for only one host and they can distribute it among themselves.

1. Global Relational Database Schema:

Users(<u>UserID</u>, firstName, lastName, phone, registrationDate, statusOfSubscription, email, monthlySubscriptionFee)

UserID → UserID, firstName, lastName, phone, registrationDate, statusOfSubscription, email, monthlySubscriptionFee

The relation is in 3NF because the LHS of the relation is a superkey. Each user is assigned a unique UserID which can identify all the other attributes. We could have considered firstName, lastName and registrationDate to uniquely identify the other attributes, but multiple Users can have the same firstName, lastName and registrationDate. Similarly, we could have considered Phone and/or Email, but these attributes are subject to change. For instance, a User can change their Phone or Email. Therefore any combination of these and other attributes will not be able to identify the other attributes.

PayTo(<u>UserID</u>, <u>PaymentID</u>)

UserID, PaymentId → UserID, PaymentID

The relation is in 3NF because the two attributes are a superkey as they can be used to derive themselves. This is a trivial Functional Dependency.

Finance(<u>PaymentID</u>, Type, Amount)

PaymentID → PaymentID, Type, Amount

Finance stores all the transactions and each transaction or payment is given an unique primary key which can identity the Type and Amoumnt. Since, PaymentID can uniquely identify the other attributes, making it a superkey. Therefore, the relation is in 3NF.

UserPayment(PaymentID, PaymentCycleDate, TotalEarnings)

PaymentID → PaymentID, PaymentCycleDate, TotalEarnings

‘UserPayment’ is inherited from Finance. We are using an E/R approach for inheritance. Each user transaction has a Payment ID that helps to identify Payment CycleDate and TotalEarnings. The relation is in 3NF, because PaymentID is a superkey (foreign key from Finance) and it can uniquely identify the other payment attributes.

SongPayment(PaymentID, MonthlyPayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate)

PaymentID → PaymentID, MonthlyPayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate

‘SongPayment’ is inherited from Finance. We are using an E/R approach for inheritance. The relation is in 3NF, because PaymentID is a superkey (foreign key from Finance) and it can uniquely identify the other payment attributes. No other attribute(s) can uniquely identify all the other attributes.

PaidTo(Title, ReleaseDate, ArtistID, RLID, PaymentID)

Title, ReleaseDate, ArtistID, RLID, PamentID → Title, ReleaseDate, ArtistID, RLID, PaymentID

The relation is in 3NF because the attributes in the LHS are a superkey as they can derive themselves. This is a Trivial Functional Dependency.

Songs(Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, PlayCount)

Title, ReleaseDate, ArtistID → Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, PlayCount

‘Songs’ is a weak entity, which requires us to include the primary attributes of its connected strong entities. All the attributes of a song can be uniquely identified by Title, ReleaseDate and ArtistID. Thus, the LHS is a superkey, therefore the relation is in 3NF. No other attribute(s) can uniquely identify all the other attributes.

Artists(<u>ArtistID</u>, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners)

ArtistID → ArtistID, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners

ArtistID can uniquely identify the other attributes, making it a superkey. Therefore, the relation is in '3NF. The other viable option that could uniquely identify all the other attributes is the ‘Name’. But we can rule it out, since two artists can have the same name.

Album(<u>Name</u>, <u>Title</u>, <u>ReleaseDate</u>, <u>ArtistID</u>, ReleaseYear, Edition, TrackNumber)

Name, Title, ReleaseDate, ArtistID → Name, Title, ReleaseDate, ArtistID, ReleaseYear, Edition, TrackNumber

‘Album’ is a weak entity, which requires us to include the primary attributes of its connected entities. Thus along with ‘Name’ from the Album, we will also need to include the primary key of ‘Songs’. All the attributes of an album can be uniquely identified by Name, Title, ReleaseDate and ArtistID. Thus, the LHS is a superkey, therefore the relation is in 3NF. No other combination of attributes can uniquely identify the other attributes.

RecordLabel(<u>RLID</u>, Name)

RLID → RLID, Name

RLID can uniquely identify the other attributes, making it a superkey. Therefore, the relation is in 3NF. This is a trivial Functional Dependency. Since two Record Labels can have the same name, RLID was introduced.

CollaboratedBy(ArtistID, Title, ReleaseDate, MainArtistID)

ArtistID, Title, ReleaseDate, MainArtistID → ArtistID, Title, ReleaseDate, MainArtistID

The relation is in 3NF because the attributes in the LHS are a superkey as they can derive themselves. This is a trivial Functional Dependency.

CollaboratedHosts(HostID, Name, MainHostID)

HostID, Name, MainHostID → HostID, Name, MainHostID

The relation is in 3NF because the attributes in the LHS are a superkey as they can derive themselves. This is a trivial Functional Dependency.

ContractedBy(ArtistID, RLID)

ArtistID, RLID → ArtistID, RLID

The relation is in 3NF because the attributes in the LHS are a superkey as they can derive themselves. This is a trivial Functional Dependency.

PodcastPayment(PaymentID, FeePerEpisode, TotalPaymentForEpisode, AdvertisementRevenue)

PaymentID → PaymentID, FeePerEpisode, TotalPaymentForEpisode, AdvertisementRevenue

‘PodcastPayment’ is inherited from Finance. We are using an E/R approach for inheritance. Each Podcast transaction has a Payment ID that helps to identify Payment CycleDate and TotalEarnings. We have used the PaymentID as the primary key, therefore the attributes of the podcast payment can be uniquely identified by the PaymentID. Thus, the relation is in 3NF. No other combination of attribute(s) can uniquely identify all the other attributes.

PaidToPodcast(PaymentID, Name, EpisodeNumber, HostID)

PaymentID, Name, EpisodeNumber, HostID → PaymentID, Name, EpisodeNumber, HostID

The relation is in 3NF because the attributes in the LHS are a superkey as they can derive themselves. This is a trivial Functional Dependency.

Podcast(HostID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors)

HostID, Name → HostID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors

Each podcast that is being created has an unique combination of HostID and Name, thereby making them the primary key. We could identify all the other attributes of the Podcast from them. Since, the LHS is a superkey, the relation is in 3NF. No other combination of attribute(s) can uniquely identify all the other attributes.

PodcastHosts(HostID, FirstName, LastName, Phone, Email, City)

HostID → HostID, FirstName, LastName, Phone, Email, City

Each Host of a podcast(s) is given an HostID which can uniquely identify the Host and the other attributes. GIven an HostID, we could derive all the other attributes from it. This makes HostID a primary key. Since, the LHS is a superkey, the relation is in 3NF. We could have considered First Name and Last Name to uniquely identify the other attributes, but multiple Hosts can have the same FirstName and LastName. Similarly, we could have considered Phone, Email and/or City, but these attributes are subject to change. For instance, a Host can change their Phone, Email or City. Therefore any combination of the other attributes will not be able to identity the other attributes.

Episodes(HostID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount)

HostID, EpisodeNumber, PodcastName → HostID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount

‘Episode’ is a weak entity, which requires us to include the primary attributes of its connected entities. The attributes of an Episode can all be uniquely identified given the HostId, EpisodeNumer and PodcastName. The LHS of the relation is a superkey and therefore the relation is in 3NF. Multiple episodes can possibly have the same Episode Name, and all the remaining attribute(s) are not relevant in uniquely identifying the other attributes.

SpecialGuests(<u>HostID</u>, <u>SplGuestID,EpisodeNumber</u>, <u>PodcastName</u>, Name)

HostID, SplGuestID, EpisodeNumber, PodcastName → HostID, SplGuestID,EpisodeNumber, PodcastName, Name

‘SpecialGuests’ is a weak entity, which requires us to include the primary attributes of its connected entities. The attributes of an SpecialGuests can all be uniquely identified given the HostId, SplGuestID, EpisodeNumer and PodcastName. The LHS of the relation is a superkey and therefore the relation is in 3NF. Multiple Special Guest can possibly have the same Name, therefore it is not relevant for finding the attributes.

2. Design for Global Schema

All the entity sets in our ER diagram are converted into relations, with the sane respective attributes. We are using an E/R based approach for inheritance. The entities that aren't inherited have keys that are represented by entity name. For instance, ‘User’ entity has ‘UserID’ as its primary key.

We have created an entity, “Finance” that has a record of all the payments that is coming in and out of the Wolfmedia Streaming service. This will make it easier for auditing purposes.

Users(<u>UserID</u>, firstName, lastName, phone, registrationDate, statusOfSubscription, email, monthlySubscriptionFee)

UserID is a primary key. Therefore UserID cannot be NULL. firstName, lastName, registrationDate, statusOfSubscription, email, and monthlySubscriptionFee are all fields that are required to either create an account or keep track of subscriptions. Hence, they cannot be NULL. The monthlySubscriptionFee’s default value is 0.

The phone number of the user cannot be NULL as well, because we need to reach out to the users if they cannot be reached via email.

There are no foreign keys that are being used in this relation.

PayTo(<u>UserID</u>, <u>PaymentID</u>)

Both UserID and PaymentID are both foreign keys from Users and Finance respectively and hence they cannot be NULL.

UserPayment(<u>PaymentID</u>, PaymentCycleDate, TotalEarnings)

PaymentID is a foreign key from Finance and hence cannot be NULL. PaymentCycleDate and TotalEarnings cannot be NULL since at the end of each payment cycle there will surely be a payment made from all the users which gets populated in the total earnings. Hence, at the end of each cycle there would surely be a paymentID, cycleDate and the totalEarnings of all the users.

Finance(PaymentID, Type, Amount)

PaymentID is a primary key of the entity Finance and hence cannot be NULL. The other attributes such as Type and Amount can also be not NULL and the Amount will be defaulted to 0.

SongPayment(PaymentID, MonthlyPlayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate)

PaymentID is a foreign key from Finance and hence cannot be NULL. MonthlyPlayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate cannot be NULL as well since attributes of any payment can only take values when a payment is done.

PaidTo(Title, ReleaseDate, ArtistID, RLID, PaymentID)

Title, ReleaseDate, ArtistID, RLID, PaymentID are all foreign keys from Songs, Artists, Record Label, and Finance and hence cannot be NULL.

Songs(Title, ReleaseDate, ArtistID, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, PlayCount)

Title, ReleaseDate, ArtistID are all primary keys of entity Songs and therefore cannot be NULL. ArtistID is the foreign key of the entity Artists. The other attributes, Genres, RoyaltyRate, RoyaltyPaid, ReleaseCountry, Language, Duration, and PlayCount cannot take NULL value. Every song should have some value for the attributes since they are characteristics of the song. RoyaltyPaid and PlayCount are defaulted to 0.

Artists(ArtistID, Name, PrimaryGenre, Country, Type, Status, MonthlyListeners)

ArtitsID is a primary key of the entity Artists and hence cannot be NULL. Name and Country cannot be NULL because they are part of the artist's identity. Type and Status cannot be NULL since they have to be one of the choices given. MonthlyListeners is defaulted to 0. The PrimaryGenre can take NULL value if the artist has no one major genre of songs that they specialise in.

Album(Name, Title, ReleaseDate, ArtistID, ReleaseYear, Edition, TrackNumber)

Name is a primary key of the entity and hence cannot be NULL. Title, ReleaseDate, and ArtistID are foreign keys from other entities and hence they cannot be NULL too. ReleaseYear and TrackNumber cannot be NULL since they are assigned once a song is added to the album. Edition cannot be NULL since it has to take one of three choices, special, limited, collector's edition.

RecordLabel(RLID, Name)

RLID is the primary key and hence cannot be NULL. The name of the record label would be present if a record label exists and hence cannot be NULL.

CollaboratedBy(ArtistID, Title, ReleaseDate, MainArtistID)

ArtistID is a foreign key and is referenced from the Artist's relation. Title, MainArtistID and Release Date are also foreign keys referenced from the song's relation. All these are prime attributes hence cannot exist as NULL values.

ContractedBy(ArtistID, RLID)

ArtistID and RLID are foreign keys of the entities Artists and Record Label and hence cannot be NULL.

PodcastPayment(PaymentID, FeePerEpisode, TotalPaymentForEpisode, AdvertisementRevenue)

PaymentID is a foreign key of the entity Finance since PodcastPayment has a ISA hierarchy with Finance. This implies that PaymentID cannot take a NULL value. FeePerEpisode and AdvertisementRevenue cannot be NULL since its value is fixed when the episode is released. TotalPaymentforEpisode is the summation of FeePerEpisode and AdvertisementRevenue and hence that cannot be NULL too.

PaidToPodcast(PaymentID, Name, HostID)

PaymentID, Name are foreign keys from the entities Finance and Podcast. HostID is a foreign key from Podcast which has it as a foreign key from Podcast Hosts. Hence none of the attributes can be NULL.

Podcast(HostID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors)

HostID(foreign Key) is referenced from the Host's relation and hence cannot be NULL. Name is the primary key of the podcast and hence cannot be NULL. Rating can have a NULL value since 0 rating means it is bad. Hence we keep it NULL until someone actually rates it. Rest all attributes cannot have NULL values. TotalSubscribers, episodeCount, Sponsors are all set to 0 by default.

PodcastHosts(HostID, FirstName, LastName, Phone, Email, City)

HostID is a primary key and cannot take NULL values. FirstName, LastName, Phone, Email, City cannot also take NULL values. Over here Phone cannot take null values as the host is present as an entity in the Media Streaming service and the Phone Number of the host is a point of contact of the service.

Episodes(HostID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount)

HostID and PodcastName are referenced from a podcast(which is referenced from Host). The above two attributes are foreign keys and hence cannot have null values. EpisodeNumber is a primary key and hence cannot be NULL. EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount cannot take NULL values. AdvertisementCount and ListeningCount are defaulted to 0.

SpecialGuests(HostID, SplGuestID,EpisodeNumber, PodcastName, Name)

HostID(referenced from podcast, which is referenced from Host), EpisodeNumber(referenced from episode) and PodcastName(referenced from podcast) are foreign keys and cannot be NULL.

SplGuestID is a primary key and also cannot be NULL.

The name of the special guest would surely be present as it is a podcast and hence it also cannot be NULL.

3. Base Relations:

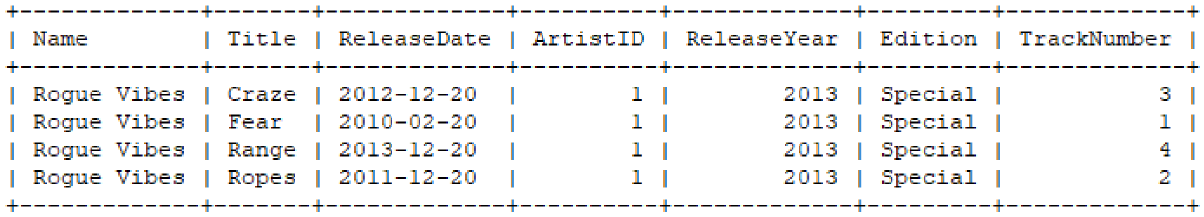
```
CREATE TABLE Users (
    UserID INT NOT NULL,
    firstName VARCHAR(128) NOT NULL,
    lastName  VARCHAR(128) NOT NULL,
    phone VARCHAR(16) NOT NULL,
    registrationDate DATE NOT NULL,
    statusOfSubscription BOOLEAN NOT NULL,
    email VARCHAR(128) NOT NULL,
    monthlySubscriptionFee INT NOT NULL DEFAULT 0,
    PRIMARY KEY(UserID)
    );


CREATE TABLE PayTo(
        UserID INT NOT NULL,
        PaymentID INT NOT NULL,
        PRIMARY KEY(UserID, PaymentID),
        FOREIGN KEY(UserID) REFERENCES Users(UserID)
        ON UPDATE CASCADE ON DELETE CASCADE,
        FOREIGN KEY(PaymentID) REFERENCES Finance(PaymentID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE UserPayment(
        PaymentID INT NOT NULL,
        PaymentCycleDate DATE NOT NULL,
        TotalEarnings DECIMAL(9,2) NOT NULL,
        PRIMARY KEY(PaymentID),
        FOREIGN KEY(PaymentID) REFERENCES Finance(PaymentID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Finance(
        PaymentID INT NOT NULL,
        Type VARCHAR(128) NOT NULL,
        Amount DECIMAL(9,2) NOT NULL DEFAULT 0,
        PRIMARY KEY(PaymentID)
);

CREATE TABLE SongPayment(
    PaymentID INT NOT NULL,
     StartDate DATE NOT NULL,
     MonthlyPlayCount INT NOT NULL,
     TotalPaymentForSong DECIMAL(9,2) NOT NULL,
    RecordLabelPayment DECIMAL(9,2) NOT NULL,
    ArtistPayment DECIMAL(9,2) NOT NULL,
    EndDate DATE NOT NULL,
    PRIMARY KEY(PaymentID),
    FOREIGN KEY(PaymentID) REFERENCES Finance(PaymentID)
    ON UPDATE CASCADE ON DELETE CASCADE
    );

CREATE TABLE PaidTo(
        Title VARCHAR(128) NOT NULL,
        ReleaseDate DATE NOT NULL,
        ArtistID INT NOT NULL,
        RLID INT NOT NULL,
        PaymentID INT NOT NULL,
        PRIMARY KEY(Title, ReleaseDate, ArtistID, RLID, PaymentID),
        FOREIGN KEY(Title, ReleaseDate, ArtistID) REFERENCES Songs(Title, ReleaseDate, ArtistID)
        ON UPDATE CASCADE ON DELETE CASCADE,
        FOREIGN KEY(RLID) REFERENCES RecordLabel(RLID)
        ON UPDATE CASCADE ON DELETE CASCADE,
        FOREIGN KEY(PaymentID) REFERENCES Finance(PaymentID)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Songs(
    Title VARCHAR(128) NOT NULL,
    ReleaseDate DATE NOT NULL,
    ArtistID INT NOT NULL,
    Genres VARCHAR(128) NOT NULL,
    RoyaltyRate DECIMAL(9,2) NOT NULL,
    RoyaltyPaid BOOLEAN NOT NULL DEFAULT 0.0,
    ReleaseCountry VARCHAR(128) NOT NULL,
    Language VARCHAR(128) NOT NULL,
    Duration DECIMAL(9,2) NOT NULL,
    PlayCount INT NOT NULL DEFAULT 0,
    PRIMARY KEY (Title, ReleaseDate, ArtistID),
    FOREIGN KEY(ArtistID) REFERENCES Artists(ArtistID)
    ON UPDATE CASCADE ON DELETE CASCADE
    );


CREATE TABLE Artists(
    ArtistID INT NOT NULL,
    Name VARCHAR(128) NOT NULL,
    PrimaryGenre VARCHAR(128),
    Country VARCHAR(128) NOT NULL,
    Type VARCHAR(128) NOT NULL,
    Status VARCHAR(128) NOT NULL,
    MonthlyListeners INT NOT NULL DEFAULT 0,
    PRIMARY KEY (ArtistID)
    );


CREATE TABLE RecordLabel(
        RLID INT NOT NULL,
        Name VARCHAR(128) NOT NULL,
        PRIMARY KEY(RLID)
);

CREATE TABLE PodcastHosts(
        HostID INT NOT NULL,
        FirstName VARCHAR(128) NOT NULL,
        LastName  VARCHAR(128) NOT NULL,
        Phone VARCHAR(16) NOT NULL,
        Email VARCHAR(128) NOT NULL,
        City VARCHAR(128) NOT NULL,
        PRIMARY KEY(HostID)
);

CREATE TABLE Album(
        Name VARCHAR(128) NOT NULL,
        Title VARCHAR(128) NOT NULL,
        ReleaseDate DATE NOT NULL,
        ArtistID INT NOT NULL,
        ReleaseYear INT NOT NULL,
        Edition VARCHAR(128) NOT NULL,
        TrackNumber INT NOT NULL,
        PRIMARY KEY (Name, Title, ReleaseDate, ArtistID),
        FOREIGN KEY(Title, ReleaseDate, ArtistID) REFERENCES Songs(Title, ReleaseDate, ArtistID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE CollaboratedBy(
ArtistID INT NOT NULL,
Title VARCHAR(128) NOT NULL,
ReleaseDate DATE NOT NULL,
MainArtistID INT NOT NULL,
PRIMARY KEY (ArtistID, Title, ReleaseDate, MainArtistID),
FOREIGN KEY(ArtistID) REFERENCES Artists(ArtistID)
        ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(Title, ReleaseDate, MainArtistID ) REFERENCES Songs(Title, ReleaseDate, ArtistID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE ContractedBy(
ArtistID INT NOT NULL,
RLID INT NOT NULL,
PRIMARY KEY (ArtistID, RLID),
FOREIGN KEY(ArtistID) REFERENCES Artists(ArtistID)
        ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(RLID) REFERENCES RecordLabel(RLID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE PodcastPayment (
PaymentID INT NOT NULL,
FeePerEpisode DECIMAL(9,2) NOT NULL,
TotalPaymentForEpisode DECIMAL(9,2) NOT NULL,
AdvertisementRevenue DECIMAL(9,2) NOT NULL,
PRIMARY KEY (PaymentID),
FOREIGN KEY(PaymentID) REFERENCES Finance(PaymentID)
        ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE PaidToPodcast (
  PaymentID INT NOT NULL,
  HostID INT NOT NULL,
  Name VARCHAR(128) NOT NULL,
  EpisodeNumber INT NOT NULL,
  PRIMARY KEY (PaymentID, Name, HostID, EpisodeNumber),
   FOREIGN KEY (PaymentID) REFERENCES Finance (PaymentID) ON UPDATE
CASCADE ON DELETE CASCADE,

        FOREIGN KEY (Name, HostID, EpisodeNumber) REFERENCES
Episodes(PodcastName,HostID, EpisodeNumber) ON UPDATE CASCADE ON
DELETE CASCADE

);


CREATE TABLE Podcast(

        HostID INT NOT NULL,

        Name VARCHAR(128) NOT NULL,

    Genres VARCHAR(128) NOT NULL,

        Rating DECIMAL(9,2),

        TotalSubscribers INT NOT NULL DEFAULT 0,

        Language VARCHAR(128) NOT NULL,

        EpisodeCount INT NOT NULL DEFAULT 0,

        Sponsors INT NOT NULL DEFAULT 0,

    PRIMARY KEY (HostID, Name),

        FOREIGN KEY(HostID) REFERENCES PodcastHosts(HostID)

        ON UPDATE CASCADE ON DELETE CASCADE

);


CREATE TABLE Episodes(

        HostID INT NOT NULL,

        EpisodeNumber INT NOT NULL,

        PodcastName VARCHAR(128) NOT NULL,

        EpisodeTitle VARCHAR(128) NOT NULL,
	Duration DECIMAL(9,2) NOT NULL,

	ListeningCount INT NOT NULL DEFAULT 0,

	ReleaseDate DATE NOT NULL,

	AdvertisementCount INT NOT NULL DEFAULT 0,

	PRIMARY KEY (PodcastName, HostID, EpisodeNumber),

	FOREIGN KEY(HostID, PodcastName) REFERENCES Podcast(HostID, Name)

	ON UPDATE CASCADE ON DELETE CASCADE

);


CREATE TABLE SpecialGuests(

	HostID INT NOT NULL,

	PodcastName VARCHAR(128) NOT NULL,

	EpisodeNumber INT NOT NULL,

	SplGuestID INT NOT NULL,

	Name VARCHAR(128) NOT NULL,

	PRIMARY KEY (HostID, PodcastName, EpisodeNumber, SplGuestID),

	FOREIGN KEY(PodcastName, HostID, EpisodeNumber) REFERENCES Episodes(PodcastName,HostID, EpisodeNumber)

	ON UPDATE CASCADE ON DELETE CASCADE);
```

Screenshots:

Album:

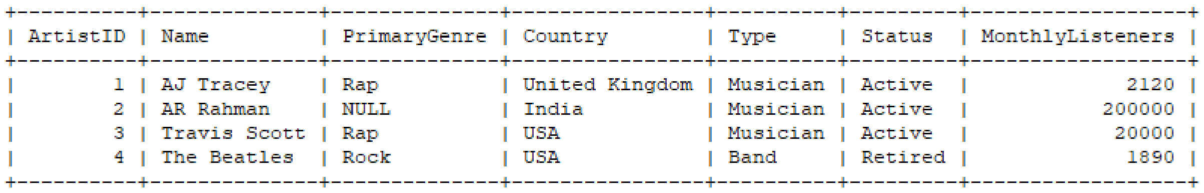
```
SELECT * FROM Album;
```



```
+-------------+-------+-------------+----------+-------------+---------+-------------+
| Name        | Title | ReleaseDate | ArtistID | ReleaseYear | Edition | TrackNumber |
+-------------+-------+-------------+----------+-------------+---------+-------------+
| Rogue Vibes | Craze | 2012-12-20  |        1 |        2013 | Special |           3 |
| Rogue Vibes | Fear  | 2010-02-20  |        1 |        2013 | Special |           1 |
| Rogue Vibes | Range | 2013-12-20  |        1 |        2013 | Special |           4 |
| Rogue Vibes | Ropes | 2011-12-20  |        1 |        2013 | Special |           2 |
+-------------+-------+-------------+----------+-------------+---------+-------------+
```

Artists:

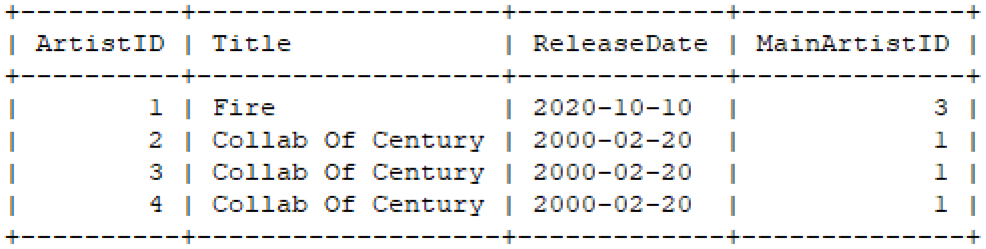
SELECT * FROM Artists;



```
+----------+--------------+--------------+----------------+----------+---------+------------------+
| ArtistID | Name         | PrimaryGenre | Country        | Type     | Status  | MonthlyListeners |
+----------+--------------+--------------+----------------+----------+---------+------------------+
|        1 | AJ Tracey    | Rap          | United Kingdom | Musician | Active  |             2120 |
|        2 | AR Rahman    | NULL         | India          | Musician | Active  |           200000 |
|        3 | Travis Scott | Rap          | USA            | Musician | Active  |            20000 |
|        4 | The Beatles  | Rock         | USA            | Band     | Retired |             1890 |
+----------+--------------+--------------+----------------+----------+---------+------------------+
```

CollaboratedBy:

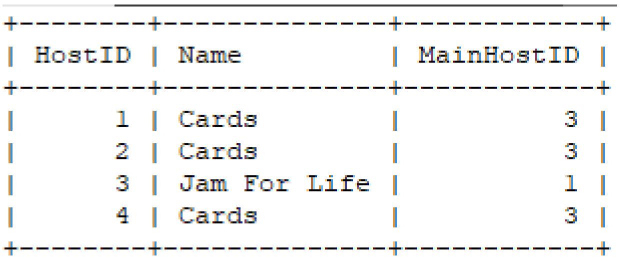
SELECT * FROM CollaboratedBy;



```
+----------+-------------------+-------------+--------------+
| ArtistID | Title             | ReleaseDate | MainArtistID |
+----------+-------------------+-------------+--------------+
|        1 | Fire              | 2020-10-10  |            3 |
|        2 | Collab Of Century | 2000-02-20  |            1 |
|        3 | Collab Of Century | 2000-02-20  |            1 |
|        4 | Collab Of Century | 2000-02-20  |            1 |
+----------+-------------------+-------------+--------------+
```

CollaboratedHosts:

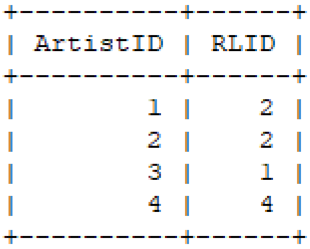
SELECT * FROM CollaboratedHosts;



```
+--------+--------------+------------+
| HostID | Name         | MainHostID |
+--------+--------------+------------+
|      1 | Cards        |          3 |
|      2 | Cards        |          3 |
|      3 | Jam For Life |          1 |
|      4 | Cards        |          3 |
+--------+--------------+------------+
```

ContractedBy:

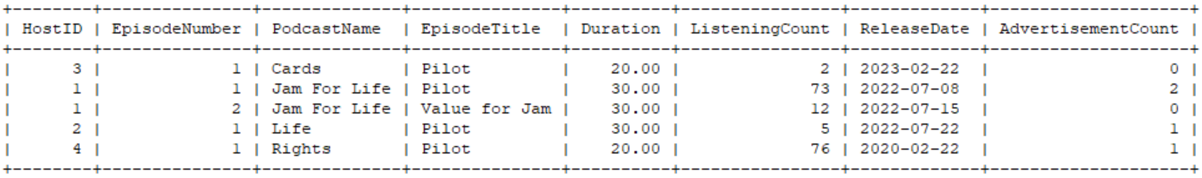
SELECT * FROM ContractedBy;



```
+----------+------+
| ArtistID | RLID |
+----------+------+
|        1 |    2 |
|        2 |    2 |
|        3 |    1 |
|        4 |    4 |
+----------+------+
```

Episodes:

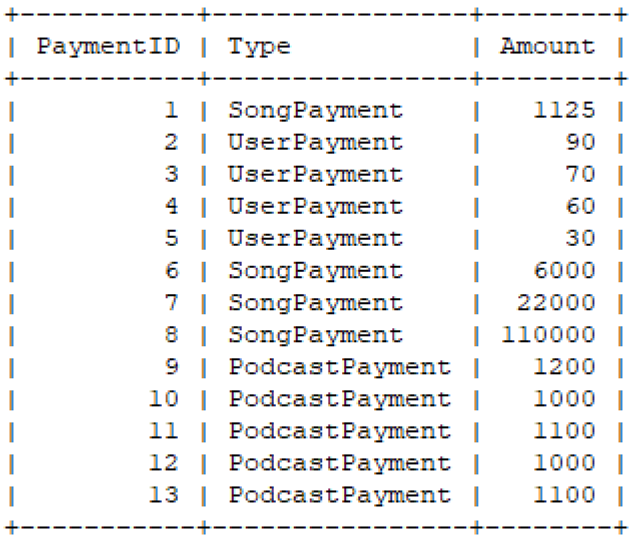
SELECT * FROM Episodes;



```
+--------+---------------+--------------+---------------+----------+----------------+-------------+--------------------+
| HostID | EpisodeNumber | PodcastName  | EpisodeTitle  | Duration | ListeningCount | ReleaseDate | AdvertisementCount |
+--------+---------------+--------------+---------------+----------+----------------+-------------+--------------------+
|      3 |             1 | Cards        | Pilot         |    20.00 |              2 | 2023-02-22  |                  0 |
|      1 |             1 | Jam For Life | Pilot         |    30.00 |             73 | 2022-07-08  |                  2 |
|      1 |             2 | Jam For Life | Value for Jam |    30.00 |             12 | 2022-07-15  |                  0 |
|      2 |             1 | Life         | Pilot         |    30.00 |              5 | 2022-07-22  |                  1 |
|      4 |             1 | Rights       | Pilot         |    20.00 |             76 | 2020-02-22  |                  1 |
+--------+---------------+--------------+---------------+----------+----------------+-------------+--------------------+
```

Finance:

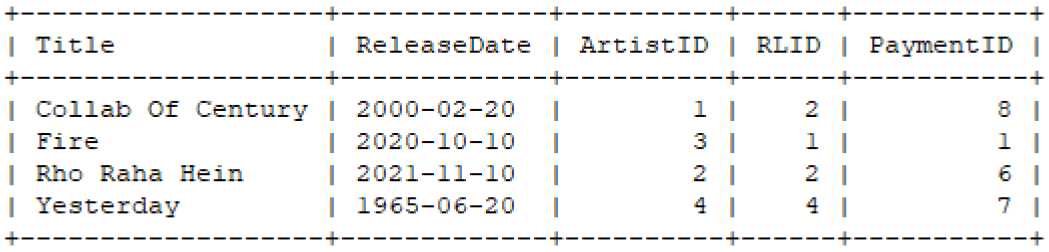
SELECT * FROM Finance;



```
+-----------+----------------+--------+
| PaymentID | Type           | Amount |
+-----------+----------------+--------+
|         1 | SongPayment    |   1125 |
|         2 | UserPayment    |     90 |
|         3 | UserPayment    |     70 |
|         4 | UserPayment    |     60 |
|         5 | UserPayment    |     30 |
|         6 | SongPayment    |   6000 |
|         7 | SongPayment    |  22000 |
|         8 | SongPayment    | 110000 |
|         9 | PodcastPayment |   1200 |
|        10 | PodcastPayment |   1000 |
|        11 | PodcastPayment |   1100 |
|        12 | PodcastPayment |   1000 |
|        13 | PodcastPayment |   1100 |
+-----------+----------------+--------+
```

PaidTo:

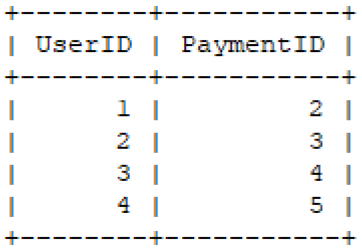
SELECT * FROM PaidTo;



```
+--------------------+-------------+----------+------+-----------+
| Title              | ReleaseDate | ArtistID | RLID | PaymentID |
+--------------------+-------------+----------+------+-----------+
| Collab Of Century  | 2000-02-20  |        1 |    2 |         8 |
| Fire               | 2020-10-10  |        3 |    1 |         1 |
| Rho Raha Hein      | 2021-11-10  |        2 |    2 |         6 |
| Yesterday          | 1965-06-20  |        4 |    4 |         7 |
+--------------------+-------------+----------+------+-----------+
```

PayTo:

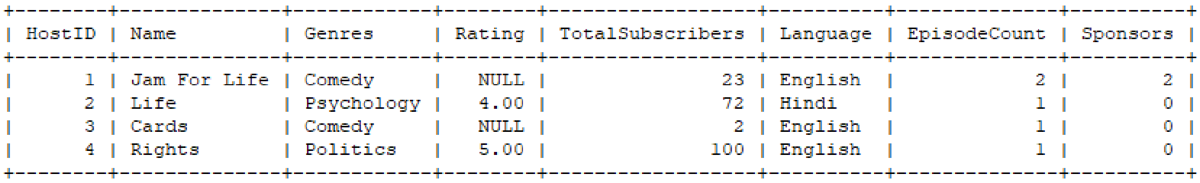
SELECT * FROM PayTo;



```
+--------+-----------+
| UserID | PaymentID |
+--------+-----------+
|      1 |         2 |
|      2 |         3 |
|      3 |         4 |
|      4 |         5 |
+--------+-----------+
```

Podcast:

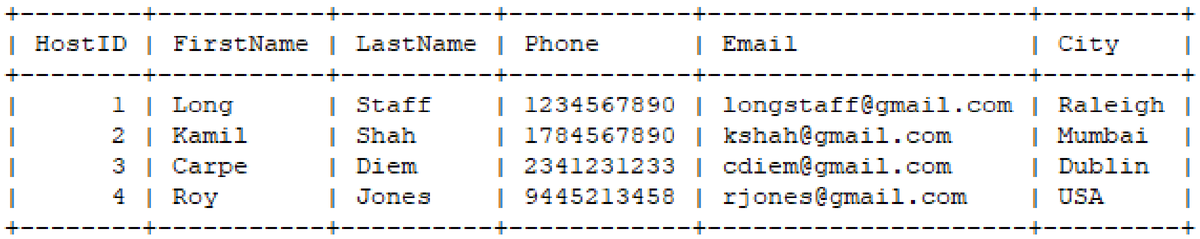
SELECT * FROM Podcast;



```
+--------+--------------+------------+--------+------------------+----------+--------------+----------+
| HostID | Name         | Genres     | Rating | TotalSubscribers | Language | EpisodeCount | Sponsors |
+--------+--------------+------------+--------+------------------+----------+--------------+----------+
|      1 | Jam For Life | Comedy     |   NULL |               23 | English  |            2 |        2 |
|      2 | Life         | Psychology |   4.00 |               72 | Hindi    |            1 |        0 |
|      3 | Cards        | Comedy     |   NULL |                2 | English  |            1 |        0 |
|      4 | Rights       | Politics   |   5.00 |              100 | English  |            1 |        0 |
+--------+--------------+------------+--------+------------------+----------+--------------+----------+
```

PodcastHosts:

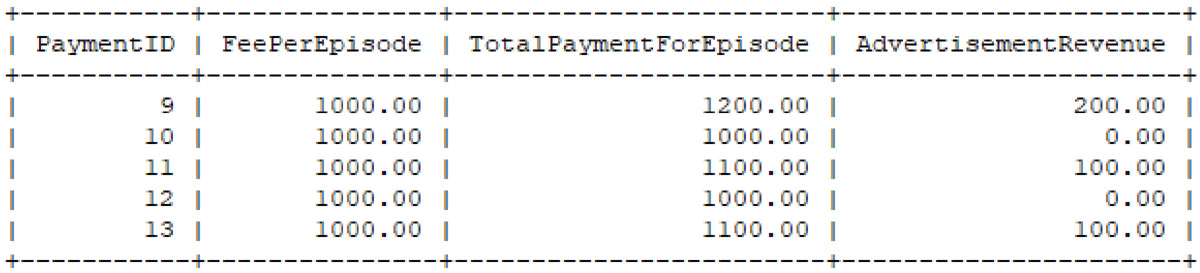
SELECT * FROM PodcastHosts;



```
+--------+-----------+----------+------------+---------------------+---------+
| HostID | FirstName | LastName | Phone      | Email               | City    |
+--------+-----------+----------+------------+---------------------+---------+
|      1 | Long      | Staff    | 1234567890 | longstaff@gmail.com | Raleigh |
|      2 | Kamil     | Shah     | 1784567890 | kshah@gmail.com     | Mumbai  |
|      3 | Carpe     | Diem     | 2341231233 | cdiem@gmail.com     | Dublin  |
|      4 | Roy       | Jones    | 9445213458 | rjones@gmail.com    | USA     |
+--------+-----------+----------+------------+---------------------+---------+
```

PodcastPayment:

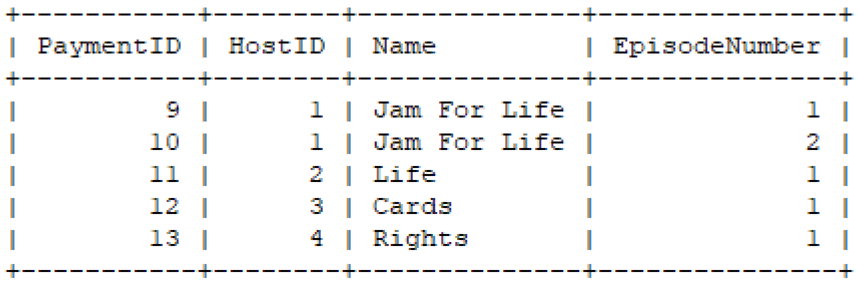
SELECT * FROM PodcastPayment;



```
+-----------+---------------+------------------------+----------------------+
| PaymentID | FeePerEpisode | TotalPaymentForEpisode | AdvertisementRevenue |
+-----------+---------------+------------------------+----------------------+
|         9 |       1000.00 |                1200.00 |               200.00 |
|        10 |       1000.00 |                1000.00 |                 0.00 |
|        11 |       1000.00 |                1100.00 |               100.00 |
|        12 |       1000.00 |                1000.00 |                 0.00 |
|        13 |       1000.00 |                1100.00 |               100.00 |
+-----------+---------------+------------------------+----------------------+
```

PaidToPodcast:

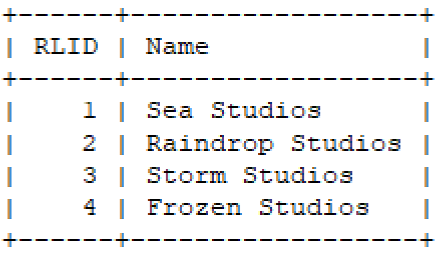
SELECT * FROM PaidToPodcast;



```
+-----------+--------+--------------+---------------+
| PaymentID | HostID | Name         | EpisodeNumber |
+-----------+--------+--------------+---------------+
|         9 |      1 | Jam For Life |             1 |
|        10 |      1 | Jam For Life |             2 |
|        11 |      2 | Life         |             1 |
|        12 |      3 | Cards        |             1 |
|        13 |      4 | Rights       |             1 |
+-----------+--------+--------------+---------------+
```

RecordLabel:

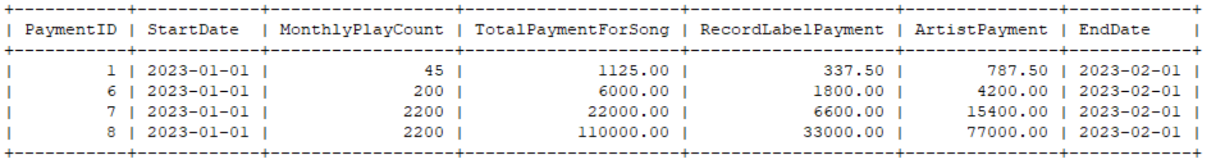
SELECT * FROM RecordLabel;



```
+------+------------------+
| RLID | Name             |
+------+------------------+
|    1 | Sea Studios      |
|    2 | Raindrop Studios |
|    3 | Storm Studios    |
|    4 | Frozen Studios   |
+------+------------------+
```

SongPayment:

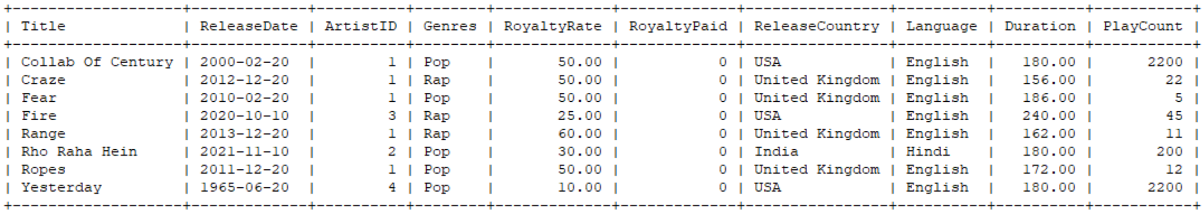
SELECT * FROM SongPayment;



```
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
| PaymentID | StartDate  | MonthlyPlayCount | TotalPaymentForSong | RecordLabelPayment | ArtistPayment | EndDate    |
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
|         1 | 2023-01-01 |               45 |             1125.00 |             337.50 |        787.50 | 2023-02-01 |
|         6 | 2023-01-01 |              200 |             6000.00 |            1800.00 |       4200.00 | 2023-02-01 |
|         7 | 2023-01-01 |             2200 |            22000.00 |            6600.00 |      15400.00 | 2023-02-01 |
|         8 | 2023-01-01 |             2200 |           110000.00 |           33000.00 |      77000.00 | 2023-02-01 |
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
```

Songs:

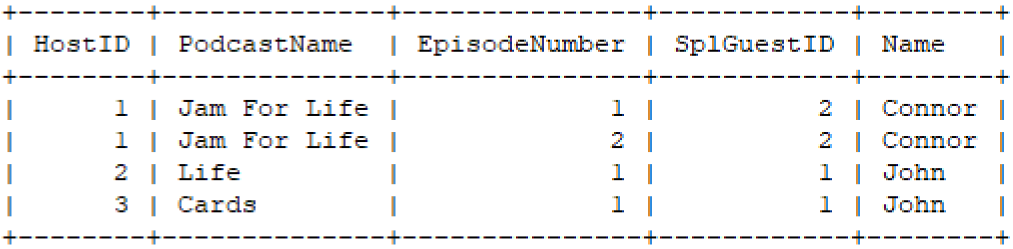
SELECT * FROM Songs;



```
+-------------------+-------------+----------+--------+-------------+-------------+----------------+----------+----------+-----------+
| Title             | ReleaseDate | ArtistID | Genres | RoyaltyRate | RoyaltyPaid | ReleaseCountry | Language | Duration | PlayCount |
+-------------------+-------------+----------+--------+-------------+-------------+----------------+----------+----------+-----------+
| Collab Of Century | 2000-02-20  |        1 | Pop    |       50.00 |           0 | USA            | English  |   180.00 |      2200 |
| Craze             | 2012-12-20  |        1 | Rap    |       50.00 |           0 | United Kingdom | English  |   156.00 |        22 |
| Fear              | 2010-02-20  |        1 | Pop    |       50.00 |           0 | United Kingdom | English  |   186.00 |         5 |
| Fire              | 2020-10-10  |        3 | Rap    |       25.00 |           0 | USA            | English  |   240.00 |        45 |
| Range             | 2013-12-20  |        1 | Rap    |       60.00 |           0 | United Kingdom | English  |   162.00 |        11 |
| Rho Raha Hein     | 2021-11-10  |        2 | Pop    |       30.00 |           0 | India          | Hindi    |   180.00 |       200 |
| Ropes             | 2011-12-20  |        1 | Pop    |       50.00 |           0 | United Kingdom | English  |   172.00 |        12 |
| Yesterday         | 1965-06-20  |        4 | Pop    |       10.00 |           0 | USA            | English  |   180.00 |      2200 |
+-------------------+-------------+----------+--------+-------------+-------------+----------------+----------+----------+-----------+
```

SpecialGuests:

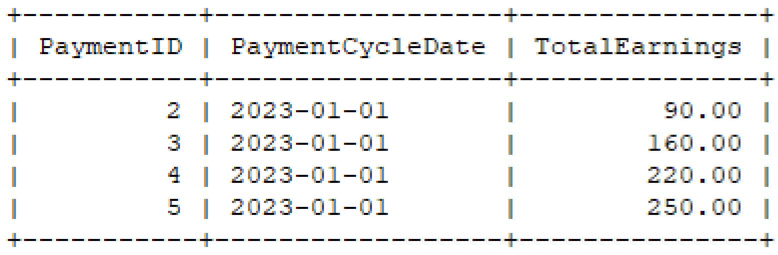
SELECT * FROM SpecialGuests;



```
+--------+--------------+---------------+------------+--------+
| HostID | PodcastName  | EpisodeNumber | SplGuestID | Name   |
+--------+--------------+---------------+------------+--------+
|      1 | Jam For Life |             1 |          2 | Connor |
|      1 | Jam For Life |             2 |          2 | Connor |
|      2 | Life         |             1 |          1 | John   |
|      3 | Cards        |             1 |          1 | John   |
+--------+--------------+---------------+------------+--------+
```

UserPayment:

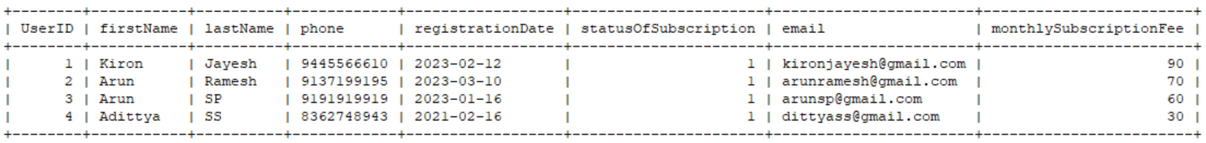
SELECT * FROM UserPayment;



```
+-----------+------------------+---------------+
| PaymentID | PaymentCycleDate | TotalEarnings |
+-----------+------------------+---------------+
|         2 | 2023-01-01       |         90.00 |
|         3 | 2023-01-01       |        160.00 |
|         4 | 2023-01-01       |        220.00 |
|         5 | 2023-01-01       |        250.00 |
+-----------+------------------+---------------+
```

Users:

SELECT * FROM Users;



```
+--------+-----------+----------+------------+------------------+----------------------+----------------------+-----------------------+
| UserID | firstName | lastName | phone      | registrationDate | statusOfSubscription | email                | monthlySubscriptionFee |
+--------+-----------+----------+------------+------------------+----------------------+----------------------+-----------------------+
|      1 | Kiron     | Jayesh   | 9445566610 | 2023-02-12       |                    1 | kironjayesh@gmail.com |                    90 |
|      2 | Arun      | Ramesh   | 9137199195 | 2023-03-10       |                    1 | arunramesh@gmail.com |                    70 |
|      3 | Arun      | SP       | 9191919919 | 2023-01-16       |                    1 | arunsp@gmail.com     |                    60 |
|      4 | Adittya   | SS       | 8362748943 | 2021-02-16       |                    1 | dittyass@gmail.com   |                    30 |
+--------+-----------+----------+------------+------------------+----------------------+----------------------+-----------------------+
```

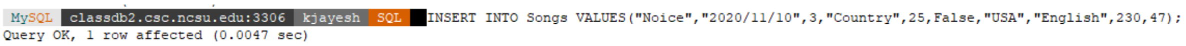
4. SQL Queries:

4.1 Tasks and Operations

4.1.1 Information Processing: Enter/update/delete basic information about songs, artists, podcast hosts, and podcast episodes. Assign songs and artists to albums. Assign artists to record labels. Assign podcast episodes and podcast hosts to podcasts.

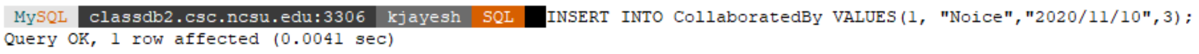
Enter Song Info:

INSERT INTO Songs VALUES("Noice","2020/11/10",3,"Country",25,False,"USA","English",230,47);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO Songs VALUES("Noice","2020/11/10",3,"Country",25,False,"USA","English",230,47);
Query OK, 1 row affected (0.0047 sec)
```

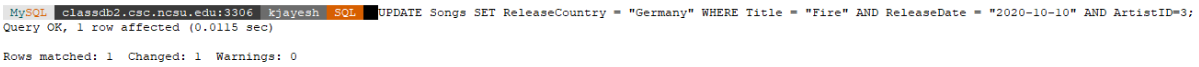
INSERT INTO CollaboratedBy VALUES(1, "Noice","2020/11/10",3);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO CollaboratedBy VALUES(1, "Noice","2020/11/10",3);
Query OK, 1 row affected (0.0041 sec)
```

Update Song Info:

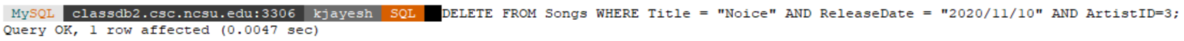
UPDATE Songs SET ReleaseCountry = "Germany" WHERE Title = "Fire" AND ReleaseDate = "2020-10-10" AND ArtistID=3;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Songs SET ReleaseCountry = "Germany" WHERE Title = "Fire" AND ReleaseDate = "2020-10-10" AND ArtistID=3;
Query OK, 1 row affected (0.0115 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Song Info:

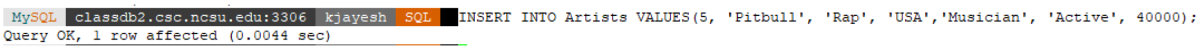
DELETE FROM Songs WHERE Title = "Noice" AND ReleaseDate = "2020/11/10" AND ArtistID=3;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  DELETE FROM Songs WHERE Title = "Noice" AND ReleaseDate = "2020/11/10" AND ArtistID=3;
Query OK, 1 row affected (0.0047 sec)
```

Enter Artists Info:

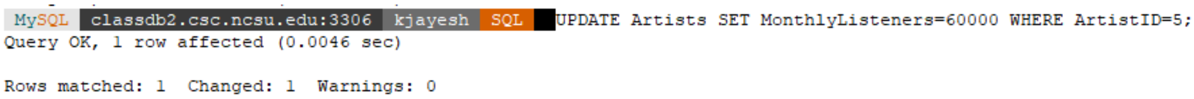
INSERT INTO Artists VALUES(5, 'Pitbull', 'Rap', 'USA','Musician', 'Active', 40000);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO Artists VALUES(5, 'Pitbull', 'Rap', 'USA','Musician', 'Active', 40000);
Query OK, 1 row affected (0.0044 sec)
```

Update Artists Info:

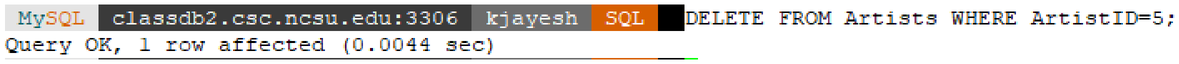
UPDATE Artists SET MonthlyListeners=60000 WHERE ArtistID=5;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Artists SET MonthlyListeners=60000 WHERE ArtistID=5;
Query OK, 1 row affected (0.0046 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Artists Info:

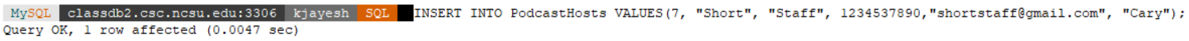
DELETE FROM Artists WHERE ArtistID=5;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  DELETE FROM Artists WHERE ArtistID=5;
Query OK, 1 row affected (0.0044 sec)
```

Enter PodcastHosts Info:

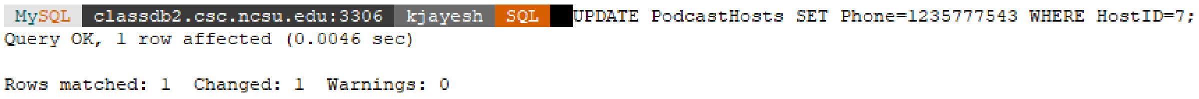
INSERT INTO PodcastHosts VALUES(7, "Short", "Staff", 1234537890,"shortstaff@gmail.com", "Cary");



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO PodcastHosts VALUES(7, "Short", "Staff", 1234537890,"shortstaff@gmail.com", "Cary");
Query OK, 1 row affected (0.0047 sec)
```

Update PodcastHosts Info:

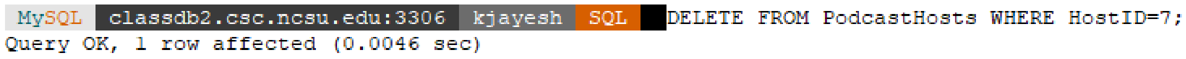
UPDATE PodcastHosts SET Phone=1235777543 WHERE HostID=7;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE PodcastHosts SET Phone=1235777543 WHERE HostID=7;
Query OK, 1 row affected (0.0046 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Delete PodcastHosts Info:

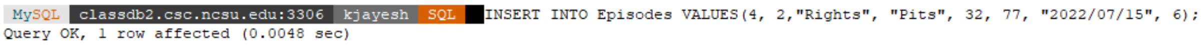
DELETE FROM PodcastHosts WHERE HostID=7;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  DELETE FROM PodcastHosts WHERE HostID=7;
Query OK, 1 row affected (0.0046 sec)
```

Enter Episodes Info/Assign Episode to Podcasts:

INSERT INTO Episodes VALUES(4, 2,"Rights", "Pits", 32, 77, "2022/07/15", 6);



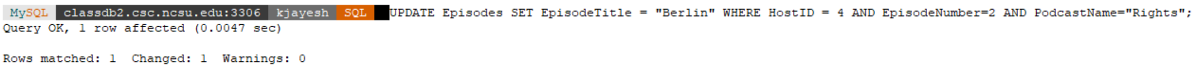
```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO Episodes VALUES(4, 2,"Rights", "Pits", 32, 77, "2022/07/15", 6);
Query OK, 1 row affected (0.0048 sec)
```

Update Episodes Info:

UPDATE Episodes SET EpisodeTitle = "Berlin" WHERE HostID = 4 AND EpisodeNumber=2 AND PodcastName="Rights"

Delete Episodes Info:

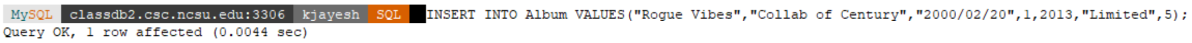
DELETE FROM Episodes WHERE HostID = 4 AND EpisodeNumber=2 AND PodcastName="Rights";



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Episodes SET EpisodeTitle = "Berlin" WHERE HostID = 4 AND EpisodeNumber=2 AND PodcastName="Rights";
Query OK, 1 row affected (0.0047 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Assign Song and Artist to Album:

INSERT INTO Album VALUES("Rogue Vibes","Collab of Century","2000/02/20",1,2013,"Limited",5);

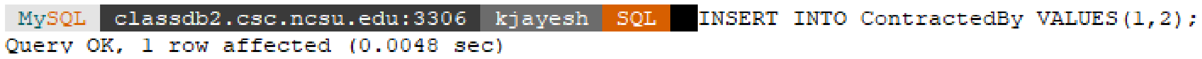


```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO Album VALUES("Rogue Vibes","Collab of Century","2000/02/20",1,2013,"Limited",5);
Query OK, 1 row affected (0.0044 sec)
```

Assign Artist to RecordLabel:

#Delete this value first from ContractedBy and then execute the Insert statement:

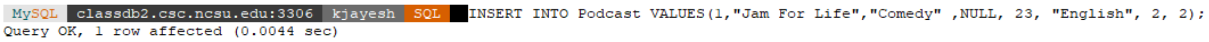
INSERT INTO ContractedBy VALUES(1,2);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO ContractedBy VALUES(1,2);
Query OK, 1 row affected (0.0048 sec)
```

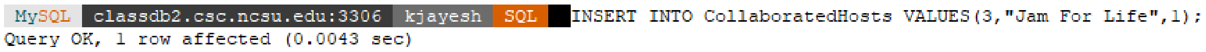
Assign PodcastHost to Podcasts:

INSERT INTO Podcast VALUES(1,"Jam For Life","Comedy" ,NULL, 23, "English", 2, 2);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO Podcast VALUES(1,"Jam For Life","Comedy" ,NULL, 23, "English", 2, 2);
Query OK, 1 row affected (0.0044 sec)
```

INSERT INTO CollaboratedHosts VALUES(3,"Jam For Life",1);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  INSERT INTO CollaboratedHosts VALUES(3,"Jam For Life",1);
Query OK, 1 row affected (0.0043 sec)
```

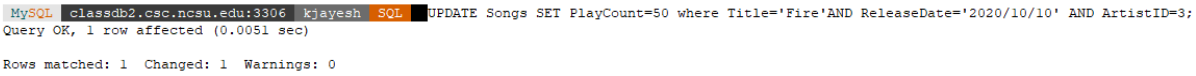
4.1.2. Maintaining metadata and records: Enter/update play count for songs. Enter/update the count of monthly listeners for artists. Enter/update the total count of subscribers and ratings for podcasts. Enter/Update the listening count for podcast episodes. Find songs and podcast episodes given by artist, album, and/or podcast.

Enter/Update PlayCount Info To Songs:

UPDATE Songs SET PlayCount=50 where Title='Fire'AND ReleaseDate='2020/10/10' AND ArtistID=3;

Enter/Update MonthlyListeners Info To Artists:

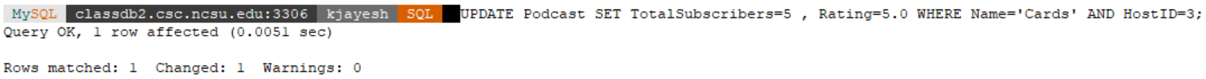
UPDATE Artists SET MonthlyListeners=15000 WHERE ArtistID=1;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Songs SET PlayCount=50 where Title='Fire'AND ReleaseDate='2020/10/10' AND ArtistID=3;
Query OK, 1 row affected (0.0051 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Enter/Update TotalSubscribers and Ratings Info To Podcasts:

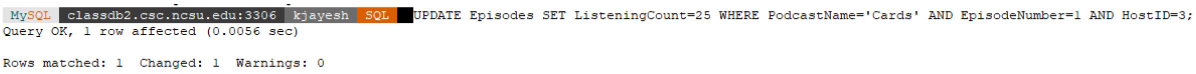
UPDATE Podcast SET TotalSubscribers=5 , Rating=5.0 WHERE Name='Cards' AND HostID=3;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Podcast SET TotalSubscribers=5 , Rating=5.0 WHERE Name='Cards' AND HostID=3;
Query OK, 1 row affected (0.0051 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Enter/Update ListeningCount Info To Episodes:

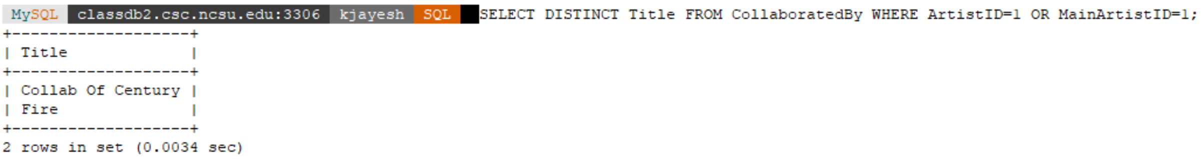
UPDATE Episodes SET ListeningCount=25 WHERE PodcastName='Cards', EpisodeNumber=1, HostID=3;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  UPDATE Episodes SET ListeningCount=25 WHERE PodcastName='Cards' AND EpisodeNumber=1 AND HostID=3;
Query OK, 1 row affected (0.0056 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Find Songs given Artist

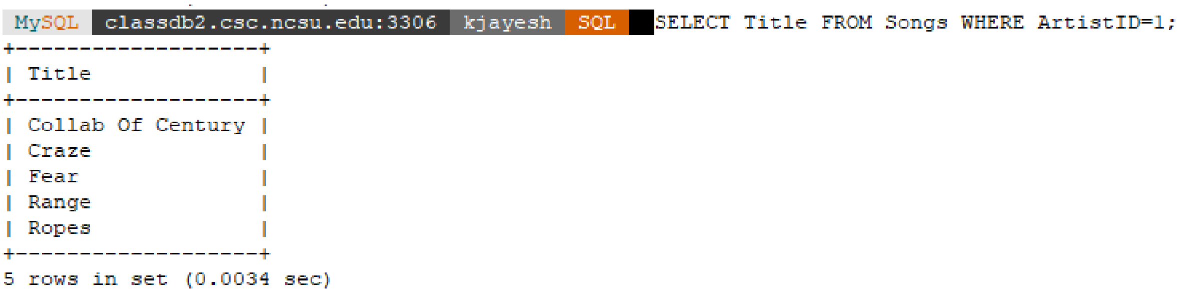
SELECT DISTINCT Title FROM CollaboratedBy WHERE ArtistID=1 OR MainArtistID=1;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  SELECT DISTINCT Title FROM CollaboratedBy WHERE ArtistID=1 OR MainArtistID=1;
+--------------------+
| Title              |
+--------------------+
| Collab Of Century |
| Fire               |
+--------------------+
2 rows in set (0.0034 sec)
```

This query is used to retrieve those songs made by an artist if they are a collaborator or the main artist of the song.

SELECT Title FROM Songs WHERE ArtistID=1;

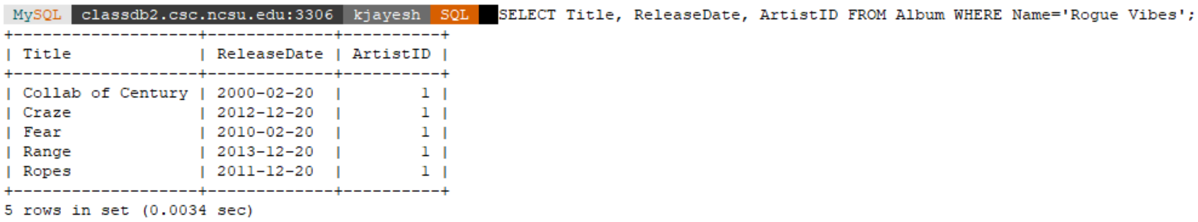


```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  SELECT Title FROM Songs WHERE ArtistID=1;
+--------------------+
| Title              |
+--------------------+
| Collab Of Century  |
| Craze              |
| Fear               |
| Range              |
| Ropes              |
+--------------------+
5 rows in set (0.0034 sec)
```

This query is used to retrieve only those songs whose main artist is the given artist.

Find Songs given Album

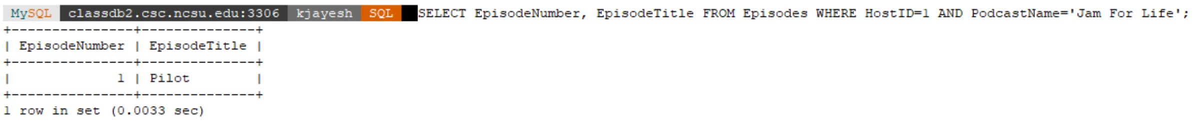
SELECT Title, ReleaseDate, ArtistID FROM Album WHERE Name=’Rogue Vibes’;



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  SELECT Title, ReleaseDate, ArtistID FROM Album WHERE Name='Rogue Vibes';
+-------------------+-------------+----------+
| Title             | ReleaseDate | ArtistID |
+-------------------+-------------+----------+
| Collab of Century | 2000-02-20  |        1 |
| Craze             | 2012-12-20  |        1 |
| Fear              | 2010-02-20  |        1 |
| Range             | 2013-12-20  |        1 |
| Ropes             | 2011-12-20  |        1 |
+-------------------+-------------+----------+
5 rows in set (0.0034 sec)
```

Find Podcast Episodes given Podcast

SELECT EpisodeNumber, EpisodeTitle FROM Episodes WHERE HostID=1 AND PodcastName=’Jam For Life’;



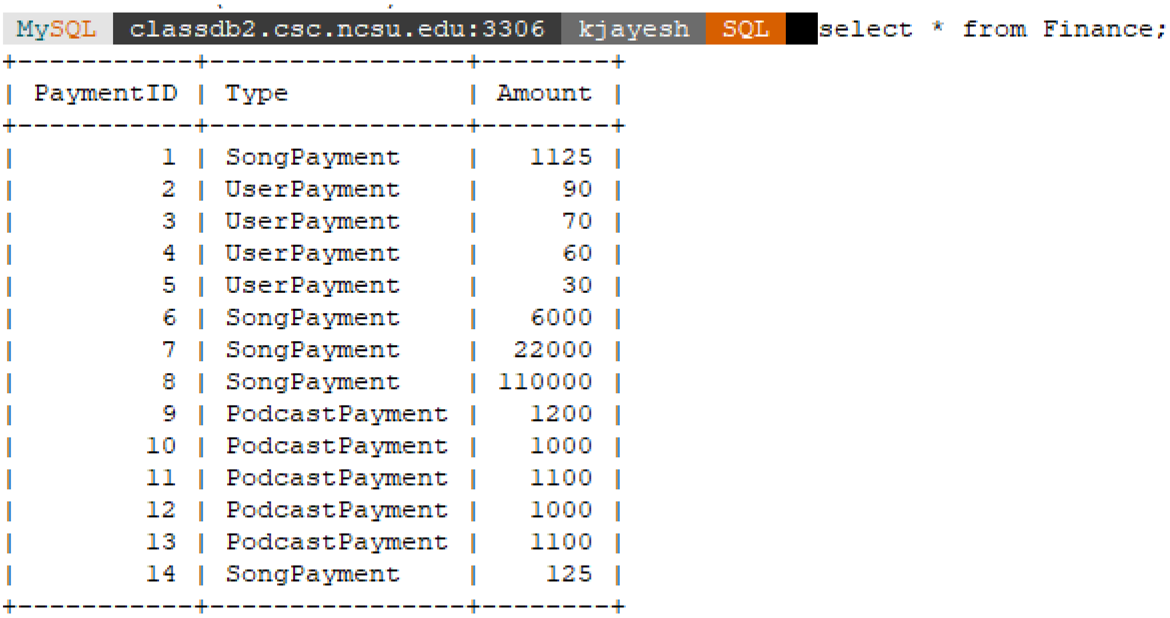
```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  SELECT EpisodeNumber, EpisodeTitle FROM Episodes WHERE HostID=1 AND PodcastName='Jam For Life';
+---------------+--------------+
| EpisodeNumber | EpisodeTitle |
+---------------+--------------+
|             1 | Pilot        |
+---------------+--------------+
1 row in set (0.0033 sec)
```

4.1.3. Maintaining payments: Make royalty payments for a given song. Monthly royalties are generated based on a royalty rate for each song times the total play count per month. 30% of the collected royalties are paid to the record label and the remainder is distributed evenly among all participating artists. Make payment to podcast hosts. Podcast hosts are paid a single flat fee per released episode and an additional bonus based on total advertisements per podcast episode. Receive payment from subscribers.

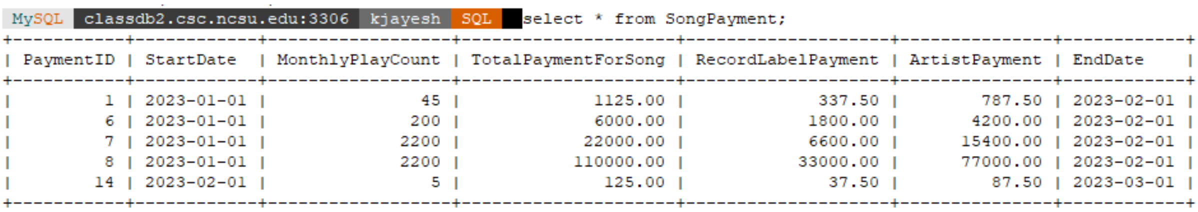
Make royalty payments for a given song:

SET @royaltyRate=(SELECT RoyaltyRate FROM Songs WHERE Title='Fire' AND ReleaseDate='2020/10/10' AND ArtistID=3);

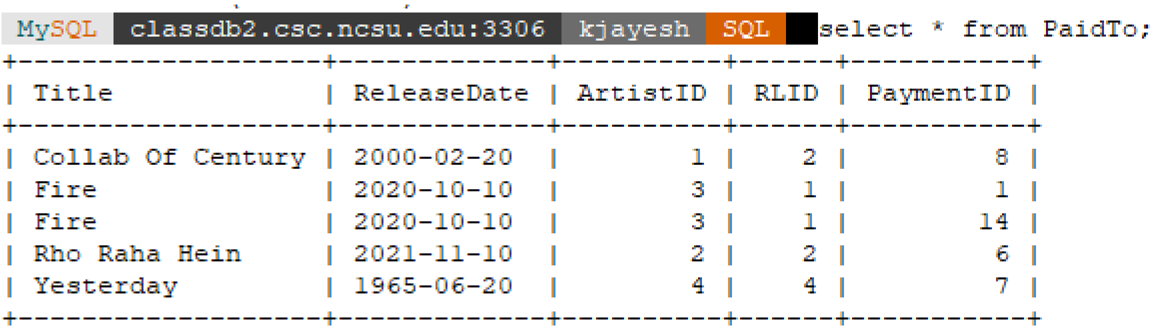
SET @playCount=(SELECT PlayCount FROM Songs WHERE Title='Fire' AND ReleaseDate='2020/10/10' AND ArtistID=3);
SET @sumMonthlyPlayCount=(SELECT SUM(MonthlyPlayCount) FROM (SELECT * FROM SongPayment NATURAL JOIN PaidTo)sub WHERE Title='Fire' AND ReleaseDate='2020/10/10' AND ArtistID=3);
SET @currentPlayCount= @playCount - @sumMonthlyPlayCount;
SET @royaltyPayment = @royaltyRate*@currentPlayCount;
INSERT INTO Finance VALUES (14, 'SongPayment', @royaltyPayment);
INSERT INTO SongPayment VALUES (14, '2023/02/01', @currentPlayCount, @royaltyPayment, 0.3*@royaltyPayment, 0.7*@royaltyPayment, '2023/03/01');
INSERT INTO PaidTo VALUES ('Fire', '2020/10/10', 3, 1, 14);



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from Finance;
+-----------+----------------+--------+
| PaymentID | Type           | Amount |
+-----------+----------------+--------+
|         1 | SongPayment    |   1125 |
|         2 | UserPayment    |     90 |
|         3 | UserPayment    |     70 |
|         4 | UserPayment    |     60 |
|         5 | UserPayment    |     30 |
|         6 | SongPayment    |   6000 |
|         7 | SongPayment    |  22000 |
|         8 | SongPayment    | 110000 |
|         9 | PodcastPayment |   1200 |
|        10 | PodcastPayment |   1000 |
|        11 | PodcastPayment |   1100 |
|        12 | PodcastPayment |   1000 |
|        13 | PodcastPayment |   1100 |
|        14 | SongPayment    |    125 |
+-----------+----------------+--------+
```



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from SongPayment;
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
| PaymentID | StartDate  | MonthlyPlayCount | TotalPaymentForSong | RecordLabelPayment | ArtistPayment | EndDate    |
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
|         1 | 2023-01-01 |               45 |             1125.00 |             337.50 |        787.50 | 2023-02-01 |
|         6 | 2023-01-01 |              200 |             6000.00 |            1800.00 |       4200.00 | 2023-02-01 |
|         7 | 2023-01-01 |             2200 |            22000.00 |            6600.00 |      15400.00 | 2023-02-01 |
|         8 | 2023-01-01 |             2200 |           110000.00 |           33000.00 |      77000.00 | 2023-02-01 |
|        14 | 2023-02-01 |                5 |              125.00 |              37.50 |         87.50 | 2023-03-01 |
+-----------+------------+------------------+---------------------+--------------------+---------------+------------+
```

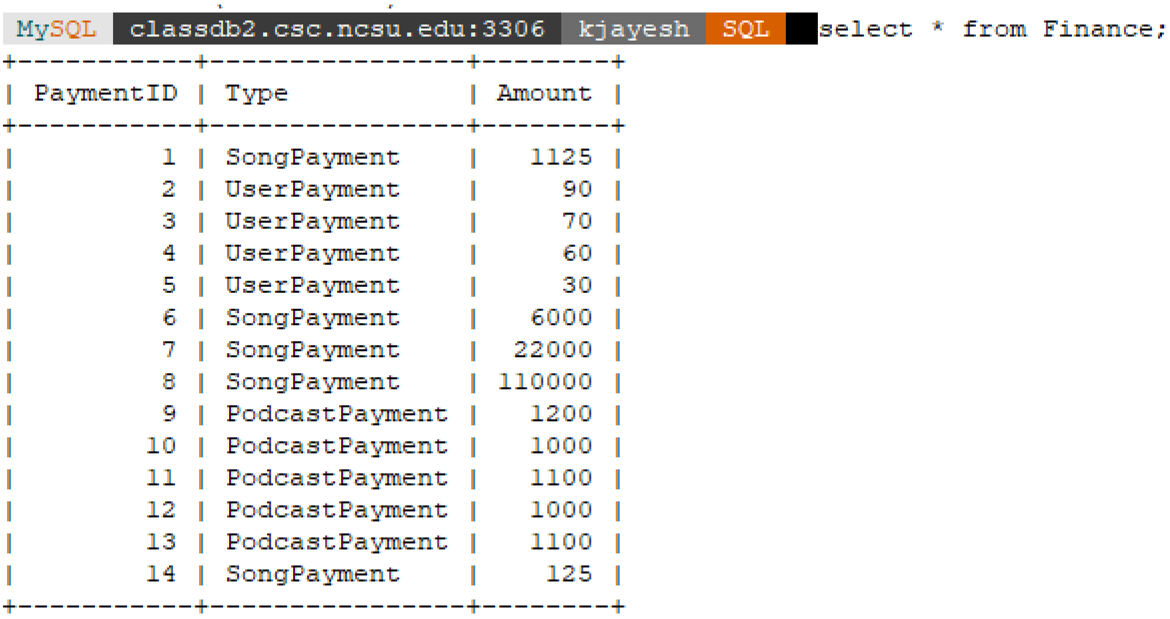


```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from PaidTo;
+-------------------+-------------+----------+------+-----------+
| Title             | ReleaseDate | ArtistID | RLID | PaymentID |
+-------------------+-------------+----------+------+-----------+
| Collab Of Century | 2000-02-20  |        1 |    2 |         8 |
| Fire              | 2020-10-10  |        3 |    1 |         1 |
| Fire              | 2020-10-10  |        3 |    1 |        14 |
| Rho Raha Hein     | 2021-11-10  |        2 |    2 |         6 |
| Yesterday         | 1965-06-20  |        4 |    4 |         7 |
+-------------------+-------------+----------+------+-----------+
```

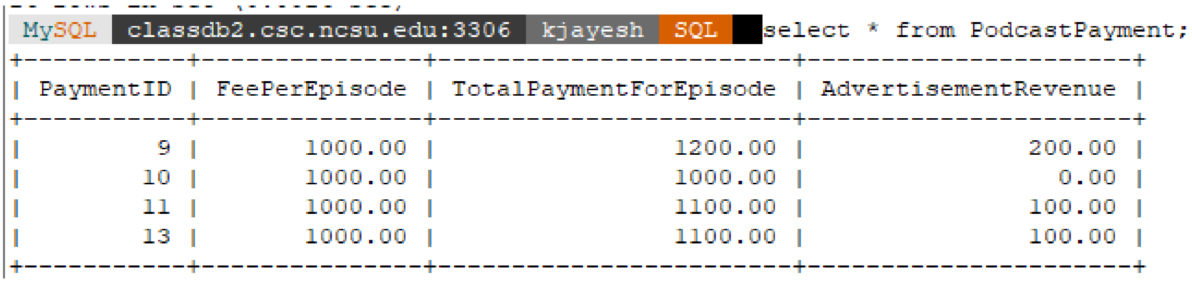
In the tables of Finance, SongPayment, and PaidTo, payment has been made for the song ‘Fire’ for the time period 2023/02/01 - 2023/03/01

Make payment to podcast hosts:

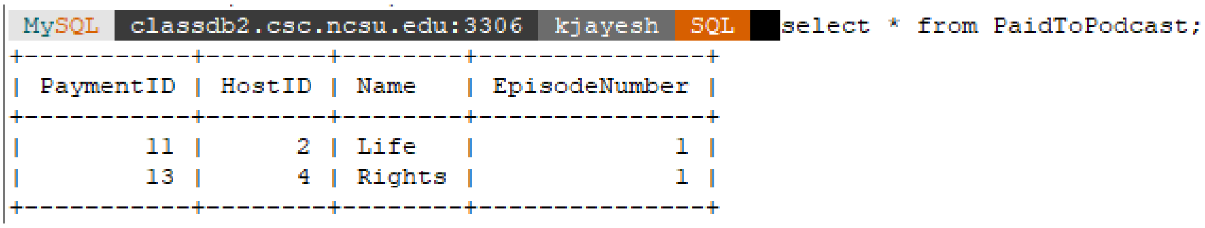
SET @adCount = (SELECT AdvertisementCount FROM Episodes WHERE HostID=2 AND EpisodeNumber=1 AND PodcastName='Life');
SET @episodePayment = 1000 + @adCount*100;
INSERT INTO Finance VALUES (11, 'PodcastPayment', @episodePayment);
INSERT INTO PodcastPayment VALUES(11, 1000, @episodePayment, @adCount*100);
INSERT INTO PaidToPodcast VALUES (11, 2, 'Life',1);



```
 MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from Finance;
+-----------+----------------+--------+
| PaymentID | Type           | Amount |
+-----------+----------------+--------+
|         1 | SongPayment    |   1125 |
|         2 | UserPayment    |     90 |
|         3 | UserPayment    |     70 |
|         4 | UserPayment    |     60 |
|         5 | UserPayment    |     30 |
|         6 | SongPayment    |   6000 |
|         7 | SongPayment    |  22000 |
|         8 | SongPayment    | 110000 |
|         9 | PodcastPayment |   1200 |
|        10 | PodcastPayment |   1000 |
|        11 | PodcastPayment |   1100 |
|        12 | PodcastPayment |   1000 |
|        13 | PodcastPayment |   1100 |
|        14 | SongPayment    |    125 |
+-----------+----------------+--------+
```



```
 MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from PodcastPayment;
+-----------+---------------+------------------------+----------------------+
| PaymentID | FeePerEpisode | TotalPaymentForEpisode | AdvertisementRevenue |
+-----------+---------------+------------------------+----------------------+
|         9 |       1000.00 |                1200.00 |               200.00 |
|        10 |       1000.00 |                1000.00 |                 0.00 |
|        11 |       1000.00 |                1100.00 |               100.00 |
|        13 |       1000.00 |                1100.00 |               100.00 |
+-----------+---------------+------------------------+----------------------+
```



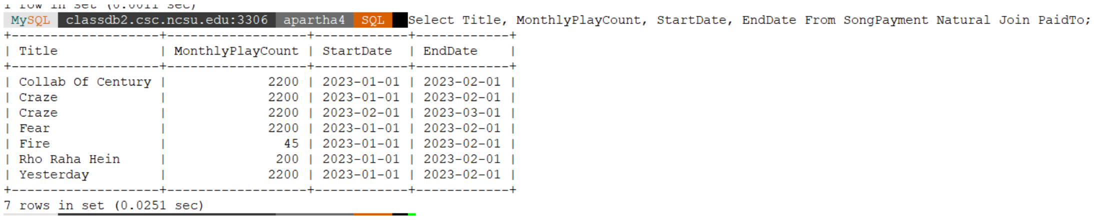
```
 MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from PaidToPodcast;
+-----------+--------+--------+---------------+
| PaymentID | HostID | Name   | EpisodeNumber |
+-----------+--------+--------+---------------+
|        11 |      2 | Life   |             1 |
|        13 |      4 | Rights |             1 |
+-----------+--------+--------+---------------+
```

From the tables, Finance, PodcastPayment, PaidToPodcast, we can see that payment for the first episode of the podcast ‘Life’ has been made. The flat fee is kept as a constant (1000) and revenue per advertisement is kept at 100. This episode has 1 advertisement and hence 1100 is being paid.

4.1.4. Reports: Generate all the following reports: Monthly play count per song/album/artist. Calculate total payments made out to host/artist/record labels per a given time period. Total revenue of the streaming service per month, per year. Report all songs/podcast episodes given an artist, album, and/or podcast.

Generate Monthly Play Count Per Song:

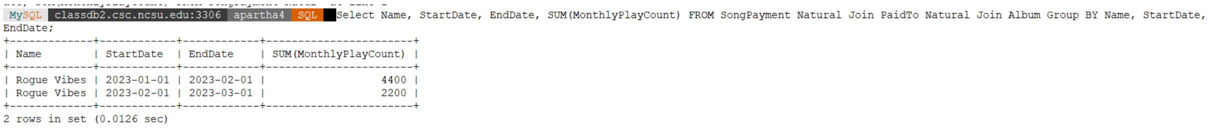
Query: Select Title, MonthlyPlayCount, StartDate, EndDate From SongPayment Natural Join PaidTo;



```
1 row in set (0.0011 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select Title, MonthlyPlayCount, StartDate, EndDate From SongPayment Natural Join PaidTo;
+-------------------+------------------+------------+------------+
| Title             | MonthlyPlayCount | StartDate  | EndDate    |
+-------------------+------------------+------------+------------+
| Collab Of Century |             2200 | 2023-01-01 | 2023-02-01 |
| Craze             |             2200 | 2023-01-01 | 2023-02-01 |
| Craze             |             2200 | 2023-02-01 | 2023-03-01 |
| Fear              |             2200 | 2023-01-01 | 2023-02-01 |
| Fire              |               45 | 2023-01-01 | 2023-02-01 |
| Rho Raha Hein     |              200 | 2023-01-01 | 2023-02-01 |
| Yesterday         |             2200 | 2023-01-01 | 2023-02-01 |
+-------------------+------------------+------------+------------+
7 rows in set (0.0251 sec)
```

Generate Monthly Play Count Per Album:

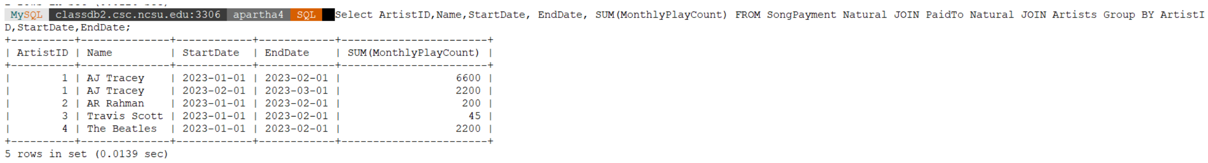
Query: Select Name, StartDate, EndDate, SUM(MonthlyPlayCount) FROM SongPayment Natural Join PaidTo Natural Join Album Group BY Name, StartDate, EndDate;



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select Name, StartDate, EndDate, SUM(MonthlyPlayCount) FROM SongPayment Natural Join PaidTo Natural Join Album Group BY Name, StartDate,
EndDate;
+-------------+------------+------------+-----------------------+
| Name        | StartDate  | EndDate    | SUM(MonthlyPlayCount) |
+-------------+------------+------------+-----------------------+
| Rogue Vibes | 2023-01-01 | 2023-02-01 |                  4400 |
| Rogue Vibes | 2023-02-01 | 2023-03-01 |                  2200 |
+-------------+------------+------------+-----------------------+
2 rows in set (0.0126 sec)
```

Generate Monthly Play Count Per Artist:

Query: Select ArtistID,Name,StartDate, EndDate, SUM(MonthlyPlayCount) FROM SongPayment Natural JOIN PaidTo Natural JOIN Artists Group BY ArtistID,StartDate,EndDate;

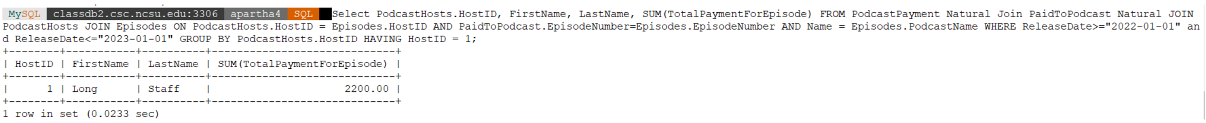


```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select ArtistID,Name,StartDate, EndDate, SUM(MonthlyPlayCount) FROM SongPayment Natural JOIN PaidTo Natural JOIN Artists Group BY ArtistI
D,StartDate,EndDate;
+----------+--------------+------------+------------+-----------------------+
| ArtistID | Name         | StartDate  | EndDate    | SUM(MonthlyPlayCount) |
+----------+--------------+------------+------------+-----------------------+
|        1 | AJ Tracey    | 2023-01-01 | 2023-02-01 |                  6600 |
|        1 | AJ Tracey    | 2023-02-01 | 2023-03-01 |                  2200 |
|        2 | AR Rahman    | 2023-01-01 | 2023-02-01 |                   200 |
|        3 | Travis Scott | 2023-01-01 | 2023-02-01 |                    45 |
|        4 | The Beatles  | 2023-01-01 | 2023-02-01 |                  2200 |
+----------+--------------+------------+------------+-----------------------+
5 rows in set (0.0139 sec)
```

Payments Made to Podcast Host for a specific Period of time using the Release Date of episode:

Query:

PodcastHost: Select PodcastHosts.HostID, FirstName, LastName, SUM(TotalPaymentForEpisode) FROM PodcastPayment Natural Join PaidToPodcast Natural JOIN PodcastHosts JOIN Episodes ON PodcastHosts.HostID = Episodes.HostID AND PaidToPodcast.EpisodeNumber=Episodes.EpisodeNumber AND Name = Episodes.PodcastName WHERE ReleaseDate>="2022-01-01" and ReleaseDate<="2023-01-01" GROUP BY PodcastHosts.HostID HAVING HostID = 1;



```
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select PodcastHosts.HostID, FirstName, LastName, SUM(TotalPaymentForEpisode) FROM PodcastPayment Natural Join PaidToPodcast Natural JOIN
PodcastHosts JOIN Episodes ON PodcastHosts.HostID = Episodes.HostID AND PaidToPodcast.EpisodeNumber=Episodes.EpisodeNumber AND Name = Episodes.PodcastName WHERE ReleaseDate>="2022-01-01" an
d ReleaseDate<="2023-01-01" GROUP BY PodcastHosts.HostID HAVING HostID = 1;
+--------+-----------+----------+----------------------------+
| HostID | FirstName | LastName | SUM(TotalPaymentForEpisode) |
+--------+-----------+----------+----------------------------+
|      1 | Long      | Staff    |                    2200.00 |
+--------+-----------+----------+----------------------------+
1 row in set (0.0233 sec)
```

Payments Made to an Artist For a given Period of Time:

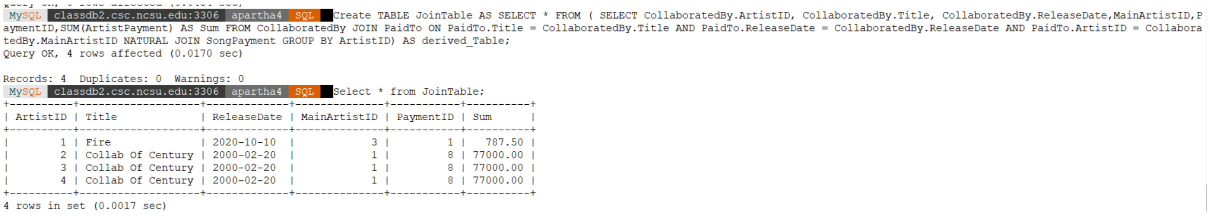
This requires the use of multiple queries to get to the final answer, hence please execute the following steps in Order to get the Answer:

First we need to calculate the values for an Artist if he was a collaborator with other songs:

For that we need to:

Query 1:

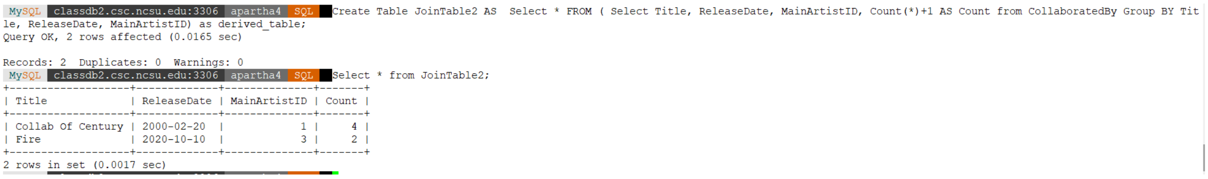
Create TABLE JoinTable AS SELECT * FROM (SELECT CollaboratedBy.ArtistID, CollaboratedBy.Title, CollaboratedBy.ReleaseDate,MainArtistID,PaymentID,SUM(ArtistPayment) AS Sum FROM CollaboratedBy JOIN PaidTo ON PaidTo.Title = CollaboratedBy.Title AND PaidTo.ReleaseDate = CollaboratedBy.ReleaseDate AND PaidTo.ArtistID = CollaboratedBy.MainArtistID NATURAL JOIN SongPayment GROUP BY ArtistID) AS derived_Table;



```
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create TABLE JoinTable AS SELECT * FROM ( SELECT CollaboratedBy.ArtistID, CollaboratedBy.Title, CollaboratedBy.ReleaseDate,MainArtistID,P
aymentID,SUM(ArtistPayment) AS Sum FROM CollaboratedBy JOIN PaidTo ON PaidTo.Title = CollaboratedBy.Title AND PaidTo.ReleaseDate = CollaboratedBy.ReleaseDate AND PaidTo.ArtistID = Collabora
tedBy.MainArtistID NATURAL JOIN SongPayment GROUP BY ArtistID) AS derived_Table;
Query OK, 4 rows affected (0.0170 sec)

Records: 4  Duplicates: 0  Warnings: 0
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * from JoinTable;
+----------+-------------------+-------------+--------------+-----------+----------+
| ArtistID | Title             | ReleaseDate | MainArtistID | PaymentID | Sum      |
+----------+-------------------+-------------+--------------+-----------+----------+
|        1 | Fire              | 2020-10-10  |            3 |         1 |   787.50 |
|        2 | Collab Of Century | 2000-02-20  |            1 |         8 | 77000.00 |
|        3 | Collab Of Century | 2000-02-20  |            1 |         8 | 77000.00 |
|        4 | Collab Of Century | 2000-02-20  |            1 |         8 | 77000.00 |
+----------+-------------------+-------------+--------------+-----------+----------+
4 rows in set (0.0017 sec)
```

Query 2:

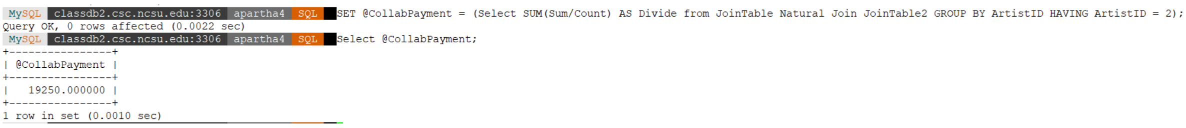
Create Table JoinTable2 AS Select * FROM (Select Title, ReleaseDate, MainArtistID, Count(*)+1 AS Count from CollaboratedBy Group BY Title, ReleaseDate, MainArtistID) as derived_table;



```
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create Table JoinTable2 AS  Select * FROM ( Select Title, ReleaseDate, MainArtistID, Count(*)+1 AS Count from CollaboratedBy Group BY Tit
le, ReleaseDate, MainArtistID) as derived_table;
Query OK, 2 rows affected (0.0165 sec)

Records: 2  Duplicates: 0  Warnings: 0
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * from JoinTable2;
+-------------------+-------------+--------------+-------+
| Title             | ReleaseDate | MainArtistID | Count |
+-------------------+-------------+--------------+-------+
| Collab Of Century | 2000-02-20  |            1 |     4 |
| Fire              | 2020-10-10  |            3 |     2 |
+-------------------+-------------+--------------+-------+
2 rows in set (0.0017 sec)
```

Query 3:

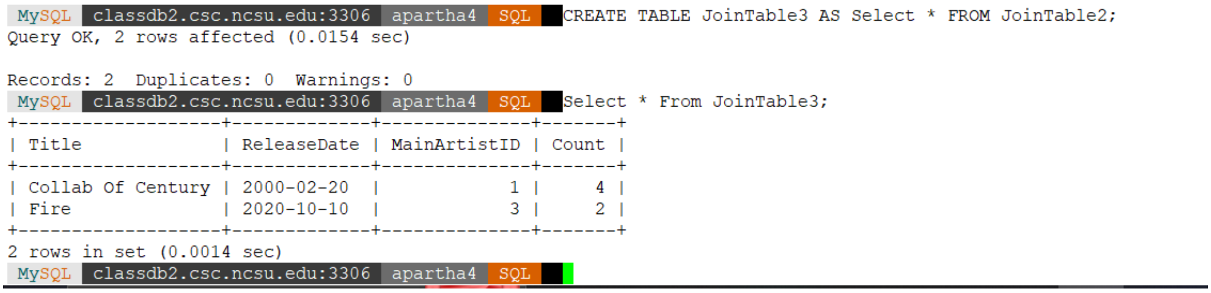
SET @CollabPayment = (Select SUM(Sum/Count) AS Divide from JoinTable Natural Join JoinTable2 GROUP BY ArtistID HAVING ArtistID = 2);



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  SET @CollabPayment = (Select SUM(Sum/Count) AS Divide from JoinTable Natural Join JoinTable2 GROUP BY ArtistID HAVING ArtistID = 2);
Query OK, 0 rows affected (0.0022 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select @CollabPayment;
+----------------+
| @CollabPayment |
+----------------+
|   19250.000000 |
+----------------+
1 row in set (0.0010 sec)
```

When the Artist is a Main Artist:

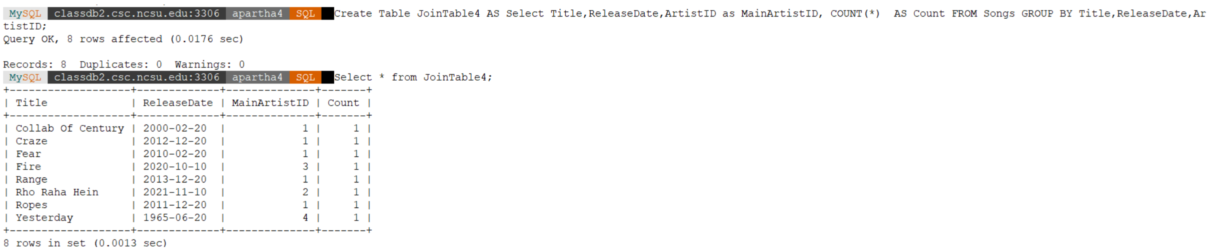
Query1: CREATE TABLE JoinTable3 AS Select * FROM JoinTable2;



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  CREATE TABLE JoinTable3 AS Select * FROM JoinTable2;
Query OK, 2 rows affected (0.0154 sec)

Records: 2  Duplicates: 0  Warnings: 0
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * From JoinTable3;
+-------------------+-------------+--------------+-------+
| Title             | ReleaseDate | MainArtistID | Count |
+-------------------+-------------+--------------+-------+
| Collab Of Century | 2000-02-20  |            1 |     4 |
| Fire              | 2020-10-10  |            3 |     2 |
+-------------------+-------------+--------------+-------+
2 rows in set (0.0014 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL 
```

Query2:

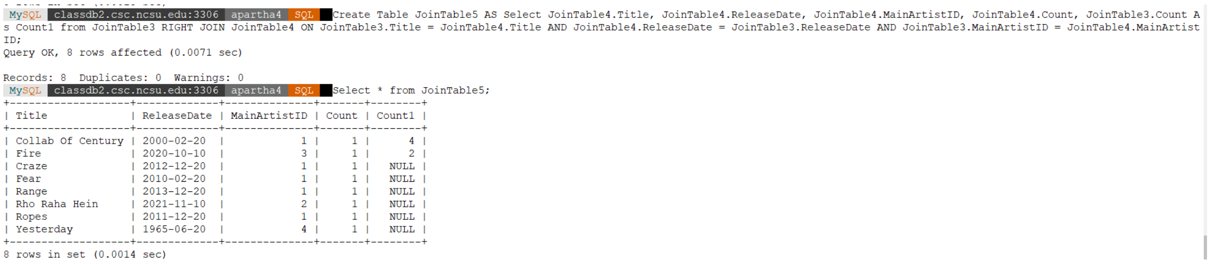
Create Table JoinTable4 AS Select Title,ReleaseDate,ArtistID as MainArtistID, COUNT(*) AS Count FROM Songs GROUP BY Title,ReleaseDate,ArtistID;



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create Table JoinTable4 AS Select Title,ReleaseDate,ArtistID as MainArtistID, COUNT(*)  AS Count FROM Songs GROUP BY Title,ReleaseDate,Ar
tistID;
Query OK, 8 rows affected (0.0176 sec)

Records: 8  Duplicates: 0  Warnings: 0
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * from JoinTable4;
+-------------------+-------------+--------------+-------+
| Title             | ReleaseDate | MainArtistID | Count |
+-------------------+-------------+--------------+-------+
| Collab Of Century | 2000-02-20  |            1 |     1 |
| Craze             | 2012-12-20  |            1 |     1 |
| Fear              | 2010-02-20  |            1 |     1 |
| Fire              | 2020-10-10  |            3 |     1 |
| Range             | 2013-12-20  |            1 |     1 |
| Rho Raha Hein     | 2021-11-10  |            2 |     1 |
| Ropes             | 2011-12-20  |            1 |     1 |
| Yesterday         | 1965-06-20  |            4 |     1 |
+-------------------+-------------+--------------+-------+
8 rows in set (0.0013 sec)
```

Query3:

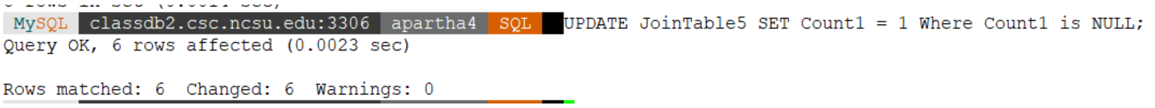
Create Table JoinTable5 AS Select JoinTable4.Title, JoinTable4.ReleaseDate, JoinTable4.MainArtistID, JoinTable4.Count, JoinTable3.Count As Count1 from JoinTable3 RIGHT JOIN JoinTable4 ON JoinTable3.Title = JoinTable4.Title AND JoinTable4.ReleaseDate = JoinTable3.ReleaseDate AND JoinTable3.MainArtistID = JoinTable4.MainArtistID;



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create Table JoinTable5 AS Select JoinTable4.Title, JoinTable4.ReleaseDate, JoinTable4.MainArtistID, JoinTable4.Count, JoinTable3.Count A
s Count1 from JoinTable3 RIGHT JOIN JoinTable4 ON JoinTable3.Title = JoinTable4.Title AND JoinTable4.ReleaseDate = JoinTable3.ReleaseDate AND JoinTable3.MainArtistID = JoinTable4.MainArtist
ID;
Query OK, 8 rows affected (0.0071 sec)

Records: 8  Duplicates: 0  Warnings: 0
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * from JoinTable5;
+-------------------+-------------+--------------+-------+--------+
| Title             | ReleaseDate | MainArtistID | Count | Count1 |
+-------------------+-------------+--------------+-------+--------+
| Collab Of Century | 2000-02-20  |            1 |     1 |      4 |
| Fire              | 2020-10-10  |            3 |     1 |      2 |
| Craze             | 2012-12-20  |            1 |     1 |   NULL |
| Fear              | 2010-02-20  |            1 |     1 |   NULL |
| Range             | 2013-12-20  |            1 |     1 |   NULL |
| Rho Raha Hein     | 2021-11-10  |            2 |     1 |   NULL |
| Ropes             | 2011-12-20  |            1 |     1 |   NULL |
| Yesterday         | 1965-06-20  |            4 |     1 |   NULL |
+-------------------+-------------+--------------+-------+--------+
8 rows in set (0.0014 sec)
```

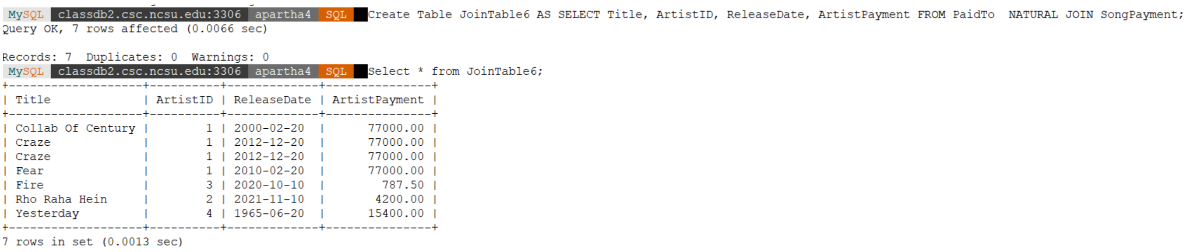
Query 4:

UPDATE JoinTable5 SET Count1 = 1 Where Count1 is NULL;



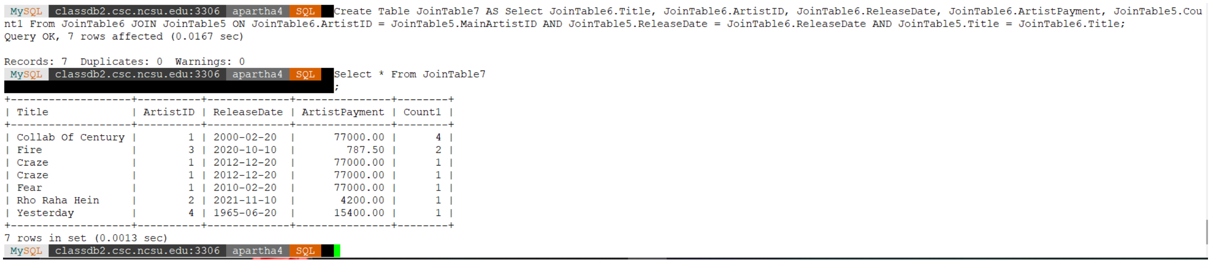
```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  UPDATE JoinTable5 SET Count1 = 1 Where Count1 is NULL;
Query OK, 6 rows affected (0.0023 sec)

Rows matched: 6  Changed: 6  Warnings: 0
```

Query 5:
Create Table JoinTable6 AS SELECT Title, ArtistID, ReleaseDate, ArtistPayment FROM PaidTo NATURAL JOIN SongPayment;



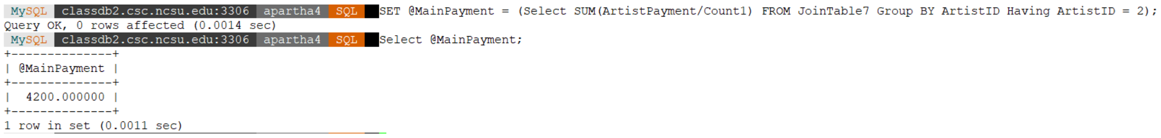
```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create Table JoinTable6 AS SELECT Title, ArtistID, ReleaseDate, ArtistPayment FROM PaidTo  NATURAL JOIN SongPayment;
Query OK, 7 rows affected (0.0066 sec)

Records: 7  Duplicates: 0  Warnings: 0
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * from JoinTable6;
+-------------------+----------+-------------+---------------+
| Title             | ArtistID | ReleaseDate | ArtistPayment |
+-------------------+----------+-------------+---------------+
| Collab Of Century |        1 | 2000-02-20  |      77000.00 |
| Craze             |        1 | 2012-12-20  |      77000.00 |
| Craze             |        1 | 2012-12-20  |      77000.00 |
| Fear              |        1 | 2010-02-20  |      77000.00 |
| Fire              |        3 | 2020-10-10  |        787.50 |
| Rho Raha Hein     |        2 | 2021-11-10  |       4200.00 |
| Yesterday         |        4 | 1965-06-20  |      15400.00 |
+-------------------+----------+-------------+---------------+
7 rows in set (0.0013 sec)
```

Query 6:
Create Table JoinTable7 AS Select JoinTable6.Title, JoinTable6.ArtistID, JoinTable6.ReleaseDate, JoinTable6.ArtistPayment, JoinTable5.Count1 From JoinTable6 JOIN JoinTable5 ON JoinTable6.ArtistID = JoinTable5.MainArtistID AND JoinTable5.ReleaseDate = JoinTable6.ReleaseDate AND JoinTable5.Title = JoinTable6.Title;



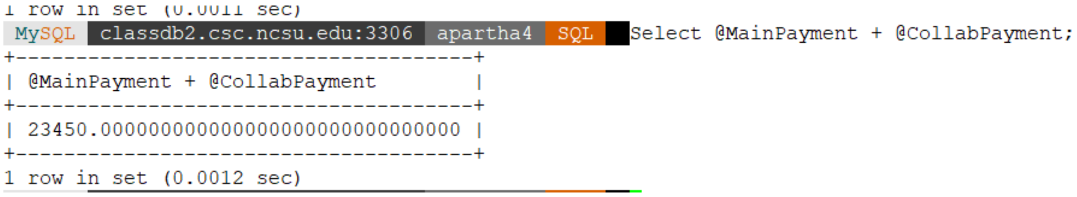
```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Create Table JoinTable7 AS Select JoinTable6.Title, JoinTable6.ArtistID, JoinTable6.ReleaseDate, JoinTable6.ArtistPayment, JoinTable5.Cou
nt1 From JoinTable6 JOIN JoinTable5 ON JoinTable6.ArtistID = JoinTable5.MainArtistID AND JoinTable5.ReleaseDate = JoinTable6.ReleaseDate AND JoinTable5.Title = JoinTable6.Title;
Query OK, 7 rows affected (0.0167 sec)

Records: 7  Duplicates: 0  Warnings: 0
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select * From JoinTable7
                                                  ;
+-------------------+----------+-------------+---------------+--------+
| Title             | ArtistID | ReleaseDate | ArtistPayment | Count1 |
+-------------------+----------+-------------+---------------+--------+
| Collab Of Century |        1 | 2000-02-20  |      77000.00 |      4 |
| Fire              |        3 | 2020-10-10  |        787.50 |      2 |
| Craze             |        1 | 2012-12-20  |      77000.00 |      1 |
| Craze             |        1 | 2012-12-20  |      77000.00 |      1 |
| Fear              |        1 | 2010-02-20  |      77000.00 |      1 |
| Rho Raha Hein     |        2 | 2021-11-10  |       4200.00 |      1 |
| Yesterday         |        4 | 1965-06-20  |      15400.00 |      1 |
+-------------------+----------+-------------+---------------+--------+
7 rows in set (0.0013 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL 
```

Query 7:
SET @MainPayment = (Select SUM(ArtistPayment/Count1) FROM JoinTable7 Group BY ArtistID Having ArtistID = 1);



```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  SET @MainPayment = (Select SUM(ArtistPayment/Count1) FROM JoinTable7 Group BY ArtistID Having ArtistID = 2);
Query OK, 0 rows affected (0.0014 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select @MainPayment;
+--------------+
| @MainPayment |
+--------------+
|  4200.000000 |
+--------------+
1 row in set (0.0011 sec)
```

Query 8:
Select @MainPayment + @CollabPayment

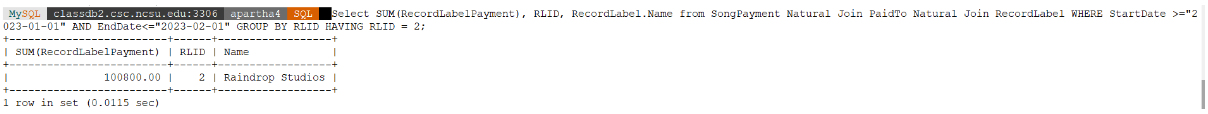


```
1 row in set (0.0011 sec)
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select @MainPayment + @CollabPayment;
+---------------------------------------+
| @MainPayment + @CollabPayment         |
+---------------------------------------+
| 23450.000000000000000000000000000000 |
+---------------------------------------+
1 row in set (0.0012 sec)
```

Payments Made To a Record Label for a given Period of time:

Query:

Select SUM(RecordLabelPayment), RLID, RecordLabel.Name from SongPayment Natural Join PaidTo Natural Join RecordLabel WHERE StartDate >="2023-01-01" AND EndDate<="2023-02-01" GROUP BY RLID HAVING RLID = 2;

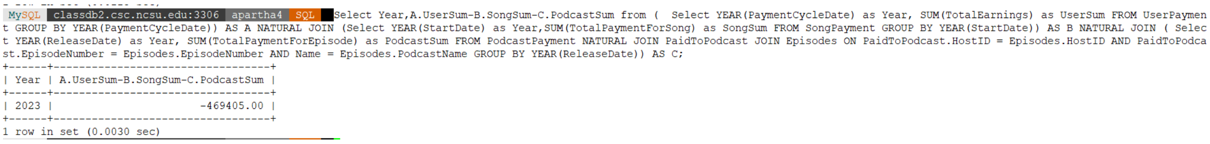


```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select SUM(RecordLabelPayment), RLID, RecordLabel.Name from SongPayment Natural Join PaidTo Natural Join RecordLabel WHERE StartDate >="2
023-01-01" AND EndDate<="2023-02-01" GROUP BY RLID HAVING RLID = 2;
+-------------------------+------+------------------+
| SUM(RecordLabelPayment) | RLID | Name             |
+-------------------------+------+------------------+
|               100800.00 |    2 | Raindrop Studios |
+-------------------------+------+------------------+
1 row in set (0.0115 sec)
```

Generate Report Per Year:

Query:

Select Year,A.UserSum-B.SongSum-C.PodcastSum from (Select YEAR(PaymentCycleDate) as Year, SUM(TotalEarnings) as UserSum FROM UserPayment GROUP BY YEAR(PaymentCycleDate)) AS A NATURAL JOIN (Select YEAR(StartDate) as Year,SUM(TotalPaymentForSong) as SongSum FROM SongPayment GROUP BY YEAR(StartDate)) AS B NATURAL JOIN (Select YEAR(ReleaseDate) as Year, SUM(TotalPaymentForEpisode) as PodcastSum FROM PodcastPayment NATURAL JOIN PaidToPodcast JOIN Episodes ON PaidToPodcast.HostID = Episodes.HostID AND PaidToPodcast.EpisodeNumber = Episodes.EpisodeNumber AND Name = Episodes.PodcastName GROUP BY YEAR(ReleaseDate)) AS C;

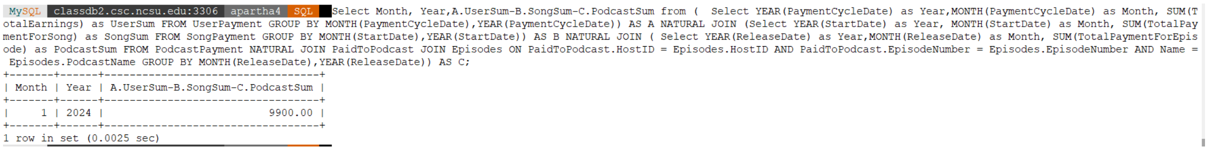


```
 MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select Year,A.UserSum-B.SongSum-C.PodcastSum from (  Select YEAR(PaymentCycleDate) as Year, SUM(TotalEarnings) as UserSum FROM UserPaymen
t GROUP BY YEAR(PaymentCycleDate)) AS A NATURAL JOIN (Select YEAR(StartDate) as Year,SUM(TotalPaymentForSong) as SongSum FROM SongPayment GROUP BY YEAR(StartDate)) AS B NATURAL JOIN ( Selec
t YEAR(ReleaseDate) as Year, SUM(TotalPaymentForEpisode) as PodcastSum FROM PodcastPayment NATURAL JOIN PaidToPodcast JOIN Episodes ON PaidToPodcast.HostID = Episodes.HostID AND PaidToPodca
st.EpisodeNumber = Episodes.EpisodeNumber AND Name = Episodes.PodcastName GROUP BY YEAR(ReleaseDate)) AS C;
+------+----------------------------------+
| Year | A.UserSum-B.SongSum-C.PodcastSum |
+------+----------------------------------+
| 2023 |                       -469405.00 |
+------+----------------------------------+
1 row in set (0.0030 sec)
```

Generate Report Per Month:

Query:

Select Month, Year,A.UserSum-B.SongSum-C.PodcastSum from (Select YEAR(PaymentCycleDate) as Year,MONTH(PaymentCycleDate) as Month, SUM(TotalEarnings) as UserSum FROM UserPayment GROUP BY MONTH(PaymentCycleDate),YEAR(PaymentCycleDate)) AS A NATURAL JOIN (Select YEAR(StartDate) as Year, MONTH(StartDate) as Month, SUM(TotalPaymentForSong) as SongSum FROM SongPayment GROUP BY MONTH(StartDate),YEAR(StartDate)) AS B NATURAL JOIN (Select YEAR(ReleaseDate) as Year,MONTH(ReleaseDate) as Month, SUM(TotalPaymentForEpisode) as PodcastSum FROM PodcastPayment NATURAL JOIN PaidToPodcast JOIN Episodes ON PaidToPodcast.HostID = Episodes.HostID AND PaidToPodcast.EpisodeNumber = Episodes.EpisodeNumber AND Name = Episodes.PodcastName GROUP BY MONTH(ReleaseDate),YEAR(ReleaseDate)) AS C;

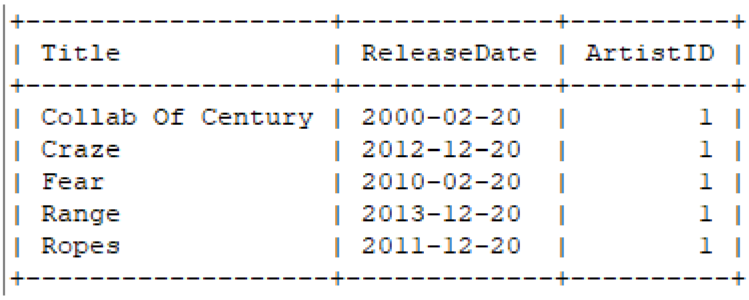


```
MySQL  classdb2.csc.ncsu.edu:3306  apartha4  SQL  Select Month, Year,A.UserSum-B.SongSum-C.PodcastSum from (  Select YEAR(PaymentCycleDate) as Year,MONTH(PaymentCycleDate) as Month, SUM(T
otalEarnings) as UserSum FROM UserPayment GROUP BY MONTH(PaymentCycleDate),YEAR(PaymentCycleDate)) AS A NATURAL JOIN (Select YEAR(StartDate) as Year, MONTH(StartDate) as Month, SUM(TotalPay
mentForSong) as SongSum FROM SongPayment GROUP BY MONTH(StartDate),YEAR(StartDate)) AS B NATURAL JOIN ( Select YEAR(ReleaseDate) as Year,MONTH(ReleaseDate) as Month, SUM(TotalPaymentForEpis
ode) as PodcastSum FROM PodcastPayment NATURAL JOIN PaidToPodcast JOIN Episodes ON PaidToPodcast.HostID = Episodes.HostID AND PaidToPodcast.EpisodeNumber = Episodes.EpisodeNumber AND Name =
 Episodes.PodcastName GROUP BY MONTH(ReleaseDate),YEAR(ReleaseDate)) AS C;
+-------+------+---------------------------------+
| Month | Year | A.UserSum-B.SongSum-C.PodcastSum |
+-------+------+---------------------------------+
|     1 | 2024 |                         9900.00 |
+-------+------+---------------------------------+
1 row in set (0.0025 sec)
```

Generate all songs given an artist

Query:

SELECT Title, ReleaseDate, ArtistID FROM Songs WHERE ArtistID=1;

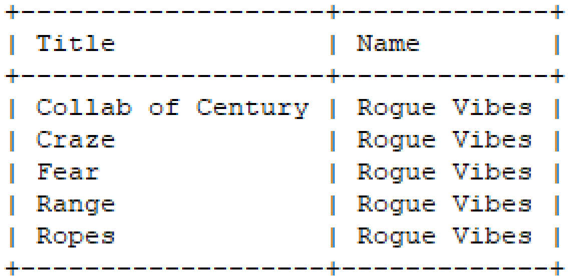


```
+--------------------+-------------+----------+
| Title              | ReleaseDate | ArtistID |
+--------------------+-------------+----------+
| Collab Of Century  | 2000-02-20  |        1 |
| Craze              | 2012-12-20  |        1 |
| Fear               | 2010-02-20  |        1 |
| Range              | 2013-12-20  |        1 |
| Ropes              | 2011-12-20  |        1 |
+--------------------+-------------+----------+
```

Generate all songs given an album

Query:

SELECT Title, Name FROM Album WHERE Name="Rogue Vibes";

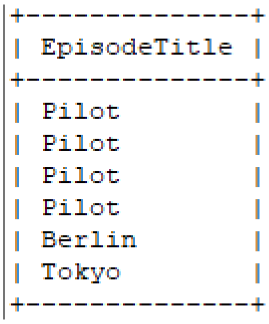


```
+--------------------+-------------+
| Title              | Name        |
+--------------------+-------------+
| Collab of Century  | Rogue Vibes |
| Craze              | Rogue Vibes |
| Fear               | Rogue Vibes |
| Range              | Rogue Vibes |
| Ropes              | Rogue Vibes |
+--------------------+-------------+
```

<u>All podcast episodes given a podcast</u>

Query:

Select EpisodeTitle From Episodes Natural Join Podcast;



```
+---------------+
| EpisodeTitle  |
+---------------+
| Pilot         |
| Pilot         |
| Pilot         |
| Pilot         |
| Berlin        |
| Tokyo         |
+---------------+
```

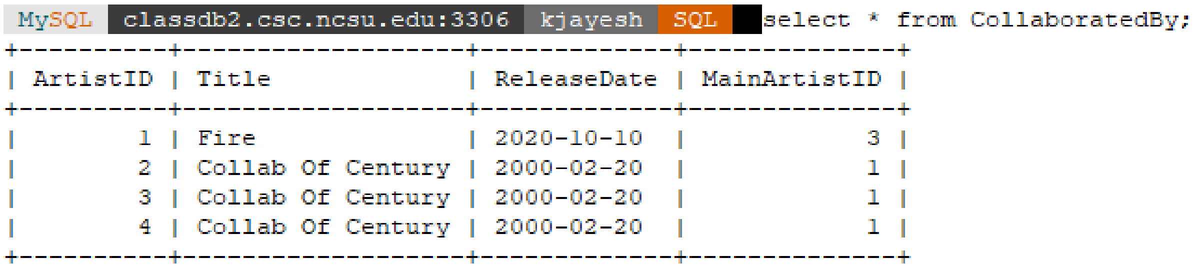
4.2 **EXPLAIN Queries**

The EXPLAIN statement in MariaDB is typically used on SELECT statements they are generally the most complex statements to execute due to the elaborate execution of the query to retrieve the data that matches the query criteria in the best way possible.

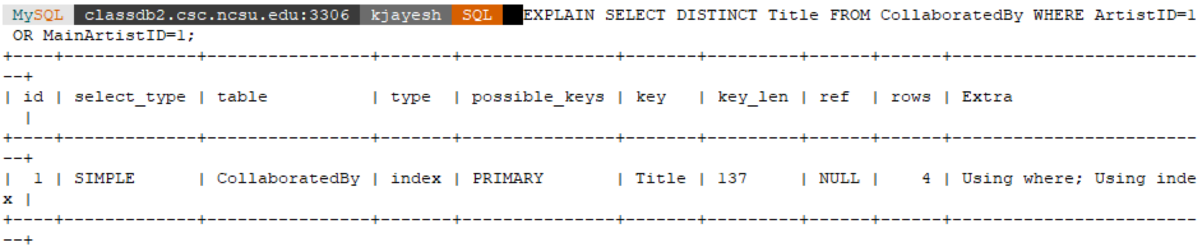
Even in our case, using EXPLAIN on the queries in section 4.1 which are not selection statements and later creating index for a column in the tables make very little sense. Looking at the selection queries in 4.1, we see that all the queries either use the IDs of the different tables or use attributes which are foreign keys.

The IDs of all tables are already indexed by default and creating another index will not make any change to the execution plan of EXPLAIN. Also, the primary keys and foreign keys are indexed as well, and hence there will not be any change in the execution plan of EXPLAIN on those statements either.

For example,
In the query SELECT DISTINCT Title FROM CollaboratedBy WHERE ArtistID=1 OR MainArtistID=1;



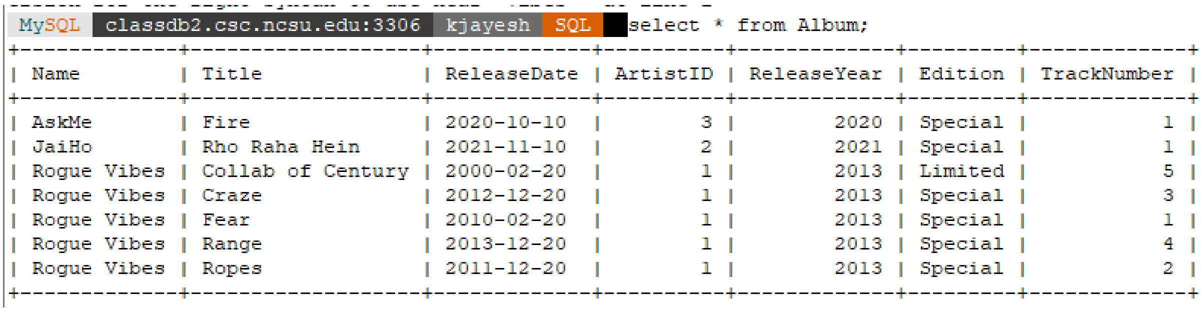
```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from CollaboratedBy;
+----------+-------------------+-------------+--------------+
| ArtistID | Title             | ReleaseDate | MainArtistID |
+----------+-------------------+-------------+--------------+
|        1 | Fire              | 2020-10-10  |            3 |
|        2 | Collab Of Century | 2000-02-20  |            1 |
|        3 | Collab Of Century | 2000-02-20  |            1 |
|        4 | Collab Of Century | 2000-02-20  |            1 |
+----------+-------------------+-------------+--------------+
```



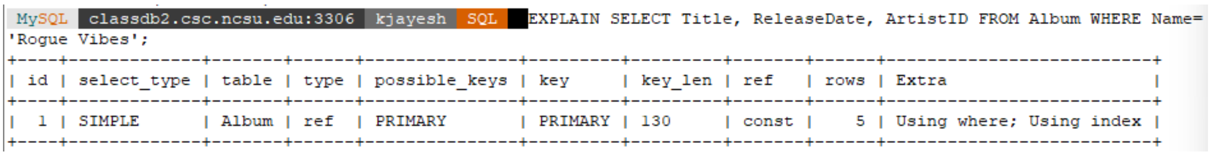
```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  EXPLAIN SELECT DISTINCT Title FROM CollaboratedBy WHERE ArtistID=1
 OR MainArtistID=1;
+----+-------------+----------------+-------+---------------+-------+---------+------+------+-----------------------
--+
| id | select_type | table          | type  | possible_keys | key   | key_len | ref  | rows | Extra
  |
+----+-------------+----------------+-------+---------------+-------+---------+------+------+-----------------------
--+
|  1 | SIMPLE      | CollaboratedBy | index | PRIMARY       | Title | 137     | NULL |    4 | Using where; Using inde
x |
+----+-------------+----------------+-------+---------------+-------+---------+------+------+-----------------------
--+
```

Since the WHERE clause contains IDs which are already indexed, the SELECT statement retrieves the data in the best possible way.

in the query SELECT Title, ReleaseDate, ArtistID FROM Album WHERE Name='Rogue Vibes';



```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  select * from Album;
+-------------+-------------------+-------------+----------+-------------+---------+-------------+
| Name        | Title             | ReleaseDate | ArtistID | ReleaseYear | Edition | TrackNumber |
+-------------+-------------------+-------------+----------+-------------+---------+-------------+
| AskMe       | Fire              | 2020-10-10  |        3 |        2020 | Special |           1 |
| JaiHo       | Rho Raha Hein     | 2021-11-10  |        2 |        2021 | Special |           1 |
| Rogue Vibes | Collab of Century | 2000-02-20  |        1 |        2013 | Limited |           5 |
| Rogue Vibes | Craze             | 2012-12-20  |        1 |        2013 | Special |           3 |
| Rogue Vibes | Fear              | 2010-02-20  |        1 |        2013 | Special |           1 |
| Rogue Vibes | Range             | 2013-12-20  |        1 |        2013 | Special |           4 |
| Rogue Vibes | Ropes             | 2011-12-20  |        1 |        2013 | Special |           2 |
+-------------+-------------------+-------------+----------+-------------+---------+-------------+
```

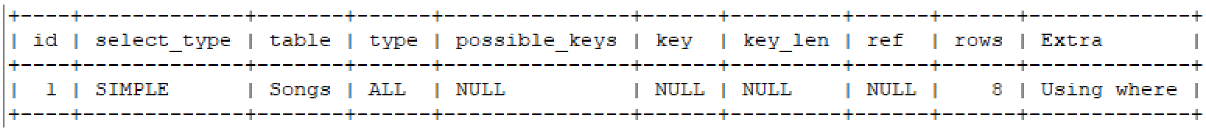


```
MySQL  classdb2.csc.ncsu.edu:3306  kjayesh  SQL  EXPLAIN SELECT Title, ReleaseDate, ArtistID FROM Album WHERE Name=
'Rogue Vibes';
+----+-------------+-------+------+---------------+---------+---------+-------+------+--------------------------+
| id | select_type | table | type | possible_keys | key     | key_len | ref   | rows | Extra                    |
+----+-------------+-------+------+---------------+---------+---------+-------+------+--------------------------+
|  1 | SIMPLE      | Album | ref  | PRIMARY       | PRIMARY | 130     | const |    5 | Using where; Using index |
+----+-------------+-------+------+---------------+---------+---------+-------+------+--------------------------+
```

There are 8 rows in Album, but since Name is a primary key of Album, it is indexed by default and hence it retrieves the data in the most efficient way.

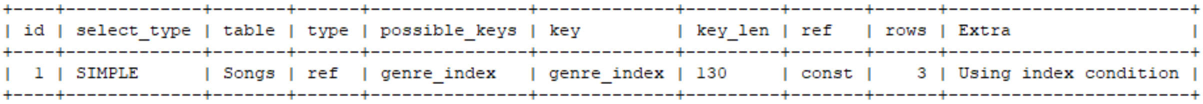
But, we can use the EXPLAIN statement on other queries that are relevant to a media streaming service.

For example, to search for Songs and its language for a particular genre, say ‘Rap’:
(1) SQL Query: SELECT Title, Language FROM Songs WHERE Genres='Rap';
(2) EXPLAIN output: (full table scan since Songs have 8 entries only)



```
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | Songs | ALL  | NULL          | NULL | NULL    | NULL |    8 | Using where |
+----+-------------+-------+------+---------------+------+---------+------+------+-------------+
```

(3) Index creation statement: CREATE INDEX genre_index ON Songs(Genres);
(4) EXPLAIN output after creation of index:



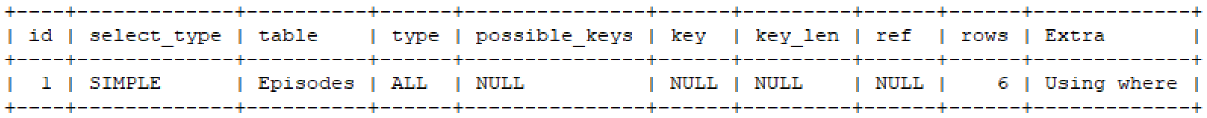
```
+----+-------------+-------+------+---------------+-------------+---------+-------+------+-----------------------+
| id | select_type | table | type | possible_keys | key         | key_len | ref   | rows | Extra                 |
+----+-------------+-------+------+---------------+-------------+---------+-------+------+-----------------------+
|  1 | SIMPLE      | Songs | ref  | genre_index   | genre_index | 130     | const |    3 | Using index condition |
+----+-------------+-------+------+---------------+-------------+---------+-------+------+-----------------------+
```

Now the query looks into only 3 entries while it previously looked into 8 entries before index creation.

Similarly, if we want to retrieve those episodes whose duration is more than 30 minutes:

(1) SQL Query: SELECT EpisodeNumber, EpisodeTitle, Duration FROM Episodes WHERE Duration>30;

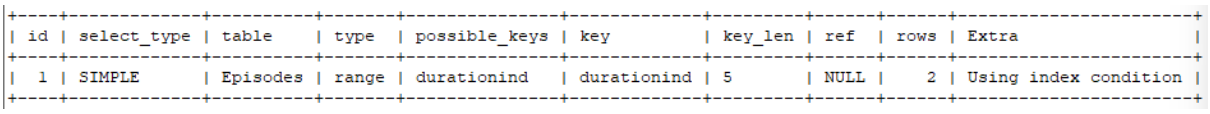
(2) EXPLAIN output: (full table scan since Episodes have 6 entries only)



```
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | Episodes | ALL  | NULL          | NULL | NULL    | NULL |    6 | Using where |
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
```

(3) Index creation statement: CREATE INDEX durationind ON Episodes(Duration);

(4) EXPLAIN output after creation of index:



```
+----+-------------+----------+-------+---------------+-------------+---------+------+------+-----------------------+
| id | select_type | table    | type  | possible_keys | key         | key_len | ref  | rows | Extra                 |
+----+-------------+----------+-------+---------------+-------------+---------+------+------+-----------------------+
|  1 | SIMPLE      | Episodes | range | durationind   | durationind | 5       | NULL |    2 | Using index condition |
+----+-------------+----------+-------+---------------+-------------+---------+------+------+-----------------------+
```

Now the query looks into only 2 entries while it previously looked into 6 entries before index creation.

4.3 Query Correctness:

1. **Query: Find the total earnings for all the songs according to ‘Rap’ genre**

   SELECT Sum(TotalPaymentForSong) As TotalEarnings From SongPayment Natural Join PaidTo Natural Join Songs Where Genres=“Rap”;

   RELATIONAL ALGEBRA:

   $\pi_{\text{TotalEarnings}} (\gamma_{\text{SUM(TotalPaymentForSong)-> TotalEarnings}} (\sigma_{\text{Genres = ‘Rap’}}$ (SongPayment ⋈ PaidTo ⋈ Songs)))

   EXPLANATION:

   The relation SongPayment contains all of the attributes PaymentID, MonthlyPayCount, StartDate, TotalPaymentForSong, RecordLabelPayment, ArtistPayment, EndDate. The relation PaidTo contains all of the attributes Title, ReleaseDate, ArtistID, RLID, PaymentID. Therefore, the natural join of these two relations is based on the common attribute PaymentID. Now, to find the total earnings of songs which have the genre as “Rap”, we have to again join the previous two relations with the relation Songs based on the common attributes Title, ReleaseDate, ArtistID and then we have to select all of the rows which we got from the above join where Genres=“Rap”. After that, the aggregation operation sums all the values in the TotalPaymentForSong column and returns the total earnings of all the songs for which the genre is Rap.

2. **Query: Find the total duration for all podcast episodes according to ‘comedy’ genre**

   SELECT Sum(Duration) As TotalDuration From Episodes Natural Join Podcast Where Genres=“Comedy”;

   RELATIONAL ALGEBRA:

   $\pi_{\text{TotalDuration}} (\gamma_{\text{SUM(Duration)->TotalDuration}}(\sigma_{\text{Genres = ‘Comedy’}} (\text{Episodes} \bowtie \text{Podcast})))$

   EXPLANATION:

   The relation Episodes contains all of the attributes HostID, EpisodeNumber, PodcastName, EpisodeTitle, Duration, ListeningCount, ReleaseDate, AdvertisementCount. The relation Podcast contains all of the attributes HostID, Name, Genres, Rating, TotalSubscribers, Language, Country, EpisodeCount, Sponsors. Therefore, the natural join of these two relations is based on the common attribute HostID and Podcast Name. Then we have to select all of the rows which we got from the above join where Genres=“Comedy”. After that, the aggregation operation sums all the values in the Duration column and returns the total duration of all the songs for which the genre is Comedy.