1. Entities and attributes – see tables and ER Diagram

A lot of relationships would exist between the entities. For example, a one-to-many relationship exists between artists and albums – one artist can have made many albums. Another relationship exists between publishing companies and artists – a many-to-many relationship. A one-to-one relationship exists between artists and tours, because artists can only have one tour. – a one-to-one relationship.

The database also has its share of constraints. The database is only limited to popular music, so any music prior to the 1950s or so would be difficult to classify as “popular music,” so classical music is disregarded entirely. This also means that older pieces of music are left out from the early 20th century. Furthermore, some albums have been deleted from respective publishing companies' catalog, which makes it impossible to catalog some albums, since they no longer technically exist. The database is also restricted to only having entries from artist that are signed to a record label, whether it is a major or independent one. Non-published artists and their respective albums, songs, et al. will not be included in the database.

1. See ER diagram
   1. F
   2. Artist Table:

idArtist -> Name

idArtist -> Years Active

idArtist -> Genre(s)

It is in 3rd Normal Form. There are no partial or transitive dependencies, and every column is atomic and associated with one key. So the artist table is in 3NF.

Album Table:

idAlbum, Name -> Length

idAlbum, Name -> Total Sales

idAlbum, Name -> Release Year

This table is in 3NF. Both primary keys uniquely identify each of the attributes in the album table, and there are no transitive or partial dependencies. Thus, the Album Table is in 3ND.

Song Table:

idSong, Album -> Name

idSong, Album -> Length

idSong, Album -> Artist

idSong, Album -> Year Released

idSong, Album -> Songwriter

idSong, Album -> Single

The database is in 3NF, because all of the relational schema are in 3NF. Thus, all the tables in the database meet the criteria for 3NF, and the database itself must be in 3NF.

Format Table:

idForm -> CD

idForm -> Digital

idForm -> Vinyl

This is in 3NF for reasons similar to other tables with one key.

CriticalReception Table:

idCriticalReception -> Publication

idCriticalReception -> Album Name

idCriticalReception -> Rating

This is in 3NF because all values are atomic and there is no partial dependency.

RetailStore Table:

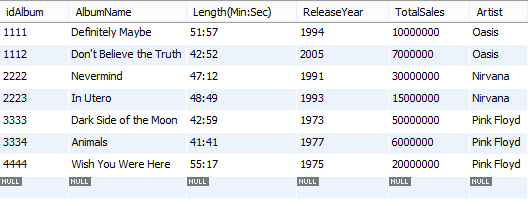
idRetailStore, ArtistName -> StoreName

idRetailStore, ArtistName -> AmountInStock

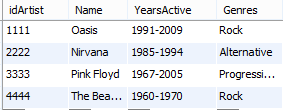
This is in 3NF because all values are atomic and there is no partial dependency.

Overall, the database is in 3NF because all the relations fall into those categories.

4.



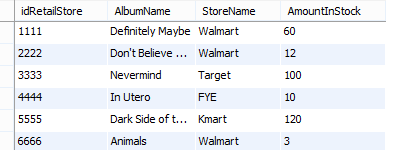
*Default Album Table*

**

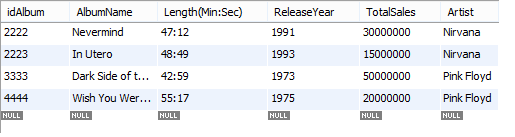
*Default Artist Table*

**

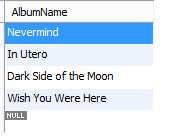
*Default Critical Reception Table*

**

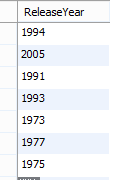
*Default RetailStore Table*



*select \* from album where TotalSales > 10000000;*

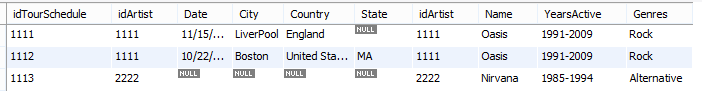


*select AlbumName from album where TotalSales > 10000000;*

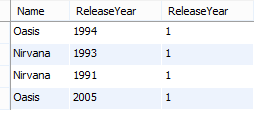


nested select:

*select ReleaseYear from album where Artist in (select Artist from artist);*

**

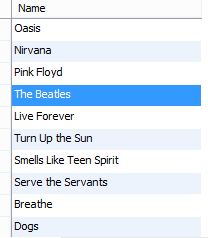
*select \* from tourschedule,artist where tourschedule.idArtist = artist.idArtist;*



*select artist.Name, album.ReleaseYear, COUNT(album.ReleaseYear) AS ReleaseYear*

*from (album INNER JOIN artist on album.Artist = artist.Name)*

*group by TotalSales HAVING COUNT(album.ReleaseYear) > 0;*



Select Union:

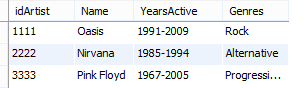
*select artist.Name from artist union select song.SongName from song;*



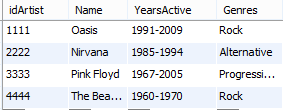
Create View:

*create view Album\_and\_Song as select AlbumName, SongName from song where YearReleased > 1990;*

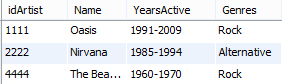
*select \* from Album\_and\_Song;*

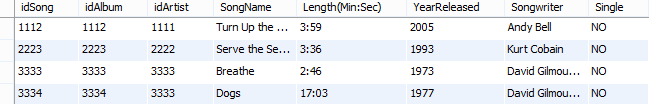
**

*delete from artist where artist.idArtist = 4444;*

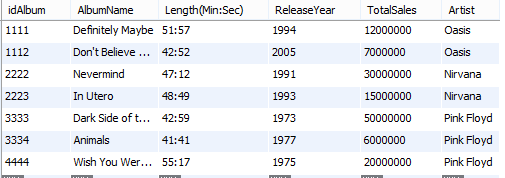
**

*insert into artist values(4444, 'The Beatles', '1960-1970', 'Rock');*

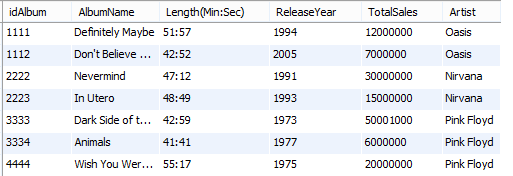
**

**

*delete from artist, song USING artist, song where artist.idArtist = 3333 and song.single='YES';*

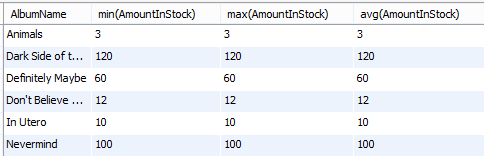
**

*update album set TotalSales = 12000000 where album.idAlbum = 1111;*

**

**

*update album, criticalreception set album.TotalSales = album.TotalSales+1000, criticalreception.Rating = '4/5' where album.idAlbum = '3333' AND criticalreception.idCriticalReception = 5555;*

**

*select AlbumName, min(AmountInStock), max(AmountInStock), avg(AmountInStock)*

*from retailstore group by AlbumName;*

B. Triggers:

For table Album:

*USE `music`;*

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` TRIGGER `album\_ADEL` AFTER DELETE ON `album`*

*FOR EACH ROW*

*begin*

*delete from song where AlbumName = old.AlbumName;*

*end*

*USE `music`;*

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` TRIGGER `album\_BINS` BEFORE INSERT ON `album`*

*FOR EACH ROW*

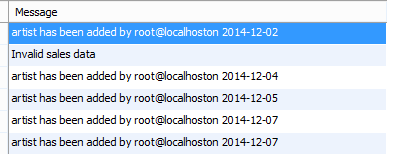
*begin*

*if(new.TotalSales < 0) then*

*insert into log values('Invalid sales data');*

*end if;*

*end*

**

*Log Table*

Artist Trigger:

*USE `music`;*

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` TRIGGER `artist\_AINS` AFTER INSERT ON `artist` FOR EACH ROW*

*begin*

*insert into Log values(concat('artist has been added by ' ,current\_user(),*

*'on ',current\_date()));*

*End*

Retail Store:

*USE `music`;*

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` TRIGGER `retailstore\_AINS` AFTER INSERT ON `retailstore`*

*FOR EACH ROW*

*begin*

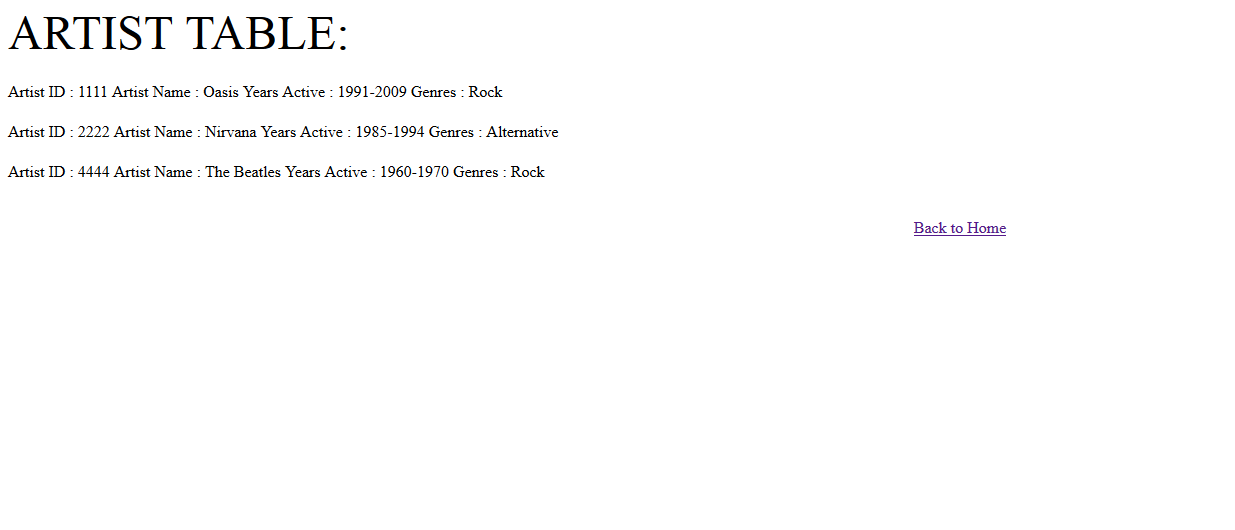
*delete from AmountInStockSummary;*

*insert AmountInStockSummary*

*select AlbumName, min(AmountInStock), max(AmountInStock), avg(AmountInStock)*

*from retailstore group by AlbumName;*

*end*

**

