Amy Tan

BillboardTop10.db

Chris Brogly

CS 338

University of Waterloo

Overview

BillboardTop10.db is a database made for students learning how to use SQLite. This database contains various relations between the yearly top 10 Billboard songs from 2006 to 2019, song details and song artist. Gender, country origins and regions were added as a feature for users to observe and study if they were interested.

I chose to make a database of the Billboard Yearly Top 10 Songs because music is an interesting topic to explore and it has so many possible relations to chart down into a database. From the list of Billboard top 10 songs, we only see the songs and their rankings, this database consists of more information about each song including information about the singers and the song BPM. Those interested in studying the patterns of mainstream music might find it interesting to play around with the database to see the different relations. My intentions in choosing this topic was to exploit the diversity in song artists and the similarities in song details. The perfect balance of diversity and similarity provides users the potential to manipulate the database into something more personal. Since SQL support SELECT statements, users can use SQL to study the information about the top 10 Billboard songs. Since the database contains years of release dates and years of making it onto the Billboard, users can use GROUP BY queries to study yearly patterns of the data. Additionally, VIEWS can be used to form playlists of songs with similar musical keys or beat. There are many possible ways to utilize this database to acquire information about mainstream music using SQL.

Development Process

BillboardTop10 started as an excel spreadsheet with all the relations and attributes on one table. I planned to utilize Spotify api to obtain song information such as BPM, runtime, … etc. However, I found tunebat.com, a website that generates the key, duration and BPM of songs being searched. This helps mitigate the difficulties when the song is misspelled or has various names if I were to use the api. Other details on the artists came from the Wikipedia pages of the artists and the Billboard information came straight from billboard.com. After writing down all the attributes onto excel I filled in the information and added in the codes later. Excel made it easier to check for duplicates when forming codes such as artist codes and country origin codes. After filling in the spread sheet I manually entered the data into the BillboardTop10 database using INSERT statements.

User Manuel: On SQL

BillboardTop10 consists of 7 tables. Details of each table, columns and data types are all listed below. [PK] stands for Primary Key and [FK] stands for Foreign Key.

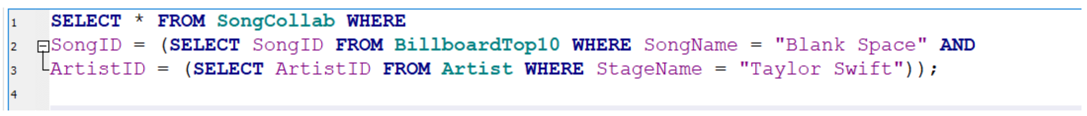
Tables	Table Description	Columns and Details
Billboard Top 10 (BillboardTop10)	This table focuses on the Billboard related achievements of each song. This information can be found on the Billboard Yearly Top 10 charts. https://www.billboard.com/charts/year-end/2006/hot-100-songs	1. Song ID **[PK]** (INTEGER) : SongID 2. Song Name (TEXT) : SongName 3. Song Artist ID **[FK]** (INTEGER) : ArtistID 4. Highest Rank (INTEGER) : HighestRank 5. First Year Charted (INTEGER): FirstYearTop10Charted 6. Total Years Charted (INTEGER): TotalYearsTop10Charted
Song Details (SongDetails)	This table focuses on song details given the SongID. This information can be found at tunebat.com. Runtime is calculated in minutes.	1. Song ID **[PK]** (INTEGER) : SongID 2. Bpm (INTEGER) : Bpm 3. Key/scale (TEXT) : KeyScale 4. Year Released (INTEGER) : YearReleased 5. Album (TEXT): Album 6. Genres (TEXT): Genre 7. Runtime (TEXT) : Runtime
Song Collaborations (SongCollab)	This table focuses on the organizational relationships involved in creating the song given SongID.	1. Song ID **[PK]** (INTEGER) : SongID 2. Collabs/Featured Artists (BLOB): FeatArtists1, FeatArtists2, FeatArtists3… 3. Song Writer ID **[FK]** (BLOB): SongWriter1, SongWriter2, SongWriter3 … 4. Producer (BLOB): Producer1, Producer2 … 5. Label (BLOB)
Top 10 Song Artists (Artists)	This table focuses on the details about the artists of songs that hit the charts between the years 2006 and 2019 given the ArtistID. This information can be found on the wiki pages on the artists. Age is based in the year 2020.	1. Song Artist ID **[PK]** (INTEGER) : ArtistID 2. Stage Name (TEXT) : StageName 3. First Name1 (TEXT) : FirstName 4. Last Name1 (TEXT) : LastName 5. First Name2 (TEXT) : FirstName2 6. Last Name2 (TEXT) : LastName2 7. Type(TEXT): Type 8. First Album (TEXT): FirstAlbum 9. First Album Release Year (INTEGER): YearOfFirstAlbum 10. Gender ID **[FK]** (INTEGER): GenderID 11. Age (Integer): Age 12. Country of Origin ID **[FK]** (INTEGER): CountryOriginID 13. Weeks on billboard top artists (INTEGER): WeeksOnTopArtist 14. Peaks on billboard top artists (INTEGER): PeakOnTopArtist
Gender Code (GenderCode)	Assigns GenderID so that Gender can be coded as INTEGERS	1. Gender ID **[PK]** (INTEGER) : GenderID 2. Gender (TEXT): Gender
Country of Origin Code (CountryOriginCode)	Assigns CountryOfOriginID so that Country of Origin can be coded as INTEGERS	1. Country of Origin **[PK]** (INTEGER): CountryOriginID 2. Country of Origin (TEXT): CountryOrigin 3. Region **[FK]** [TEXT] : Region
Region Code	Shows the relation of Country Origins.	1. Region ID**[PK]** (TEXT) : RegionID 2. Region (TEXT) : RegionName

Figure 1: Tables and attributes found in the database

Searching for Song Collab Details by Song Name and Artist Stage Name

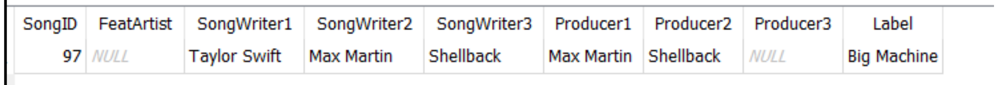
Given a song name and it's artist's stage name, subqueries can be used to find song collab information. Let us find collab information on the song "*Blank Space*" by *Taylor Swift.*

Input:



```sql
SELECT * FROM SongCollab WHERE
SongID = (SELECT SongID FROM BillboardTop10 WHERE SongName = "Blank Space" AND
ArtistID = (SELECT ArtistID FROM Artist WHERE StageName = "Taylor Swift"));
```

Output:



SongID	FeatArtist	SongWriter1	SongWriter2	SongWriter3	Producer1	Producer2	Producer3	Label
97	NULL	Taylor Swift	Max Martin	Shellback	Max Martin	Shellback	NULL	Big Machine

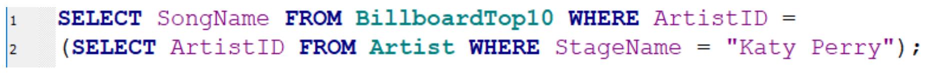
Searching for Songs by Artist Stage Name

Given an artist's stage name, subqueries or joins can be used to find the songs they released that made it onto the yearly Billboard top 10 list. Let us select songs that hit the charts by *Katy Perry*.

Start by using the SELECT statement on the BillboardTop10 table

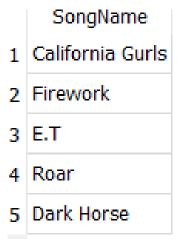
SELECT SongName	for just the song names
SELECT *	for Billboard details about the song (Only when you use subqueries)

Using subqueries Input:



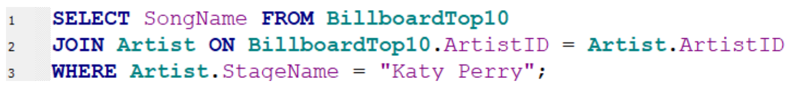
```sql
SELECT SongName FROM BillboardTop10 WHERE ArtistID =
(SELECT ArtistID FROM Artist WHERE StageName = "Katy Perry");
```

Output:



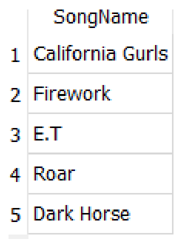
	SongName
1	California Gurls
2	Firework
3	E.T
4	Roar
5	Dark Horse

Using joins Input:



```sql
SELECT SongName FROM BillboardTop10
JOIN Artist ON BillboardTop10.ArtistID = Artist.ArtistID
WHERE Artist.StageName = "Katy Perry";
```

Output:

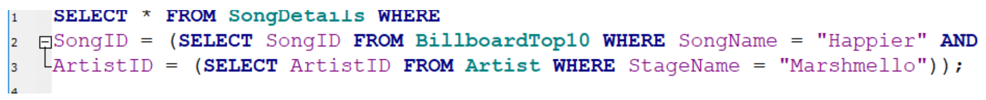


	SongName
1	California Gurls
2	Firework
3	E.T
4	Roar
5	Dark Horse

Searching for Music Details of the Song by Song Name and Artist Stage Name

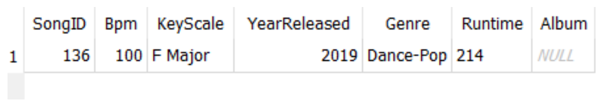
Given the song name, subqueries can be used to search for music information of the song. Let us find the music information of the song “*Happier*” by *Marshmello*.

Input:



```sql
SELECT * FROM SongDetails WHERE
SongID = (SELECT SongID FROM BillboardTop10 WHERE SongName = "Happier" AND
ArtistID = (SELECT ArtistID FROM Artist WHERE StageName = "Marshmello"));
```

Output:

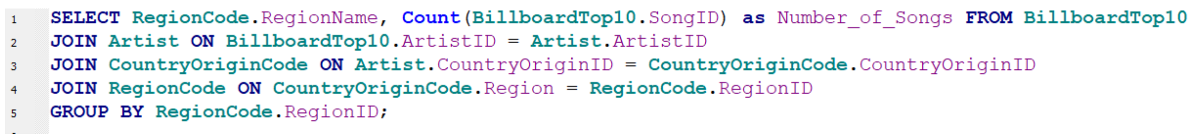


	SongID	Bpm	KeyScale	YearReleased	Genre	Runtime	Album
1	136	100	F Major	2019	Dance-Pop	214	NULL

Grouping Songs by Regions

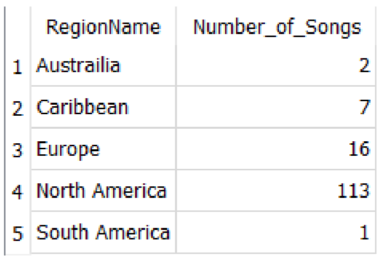
Users can do a lot with the regions of each artist. Functions such as Count, Max, or Min can be used to compare songs from regions to regions. Regions can be explored using joins. Region names can be found on the table RegionCode, which is used to sort the countries of origin of each artist. To group songs by regions, we need to join the tables BillboardTop10, Artist, CountryOrginCode and RegionCode. Lets count the number of songs on the yearly Billboard Top 10 of each regions. We will be using JOINS and GROUP BY statements.

Input:



```sql
SELECT RegionCode.RegionName, Count(BillboardTop10.SongID) as Number_of_Songs FROM BillboardTop10
JOIN Artist ON BillboardTop10.ArtistID = Artist.ArtistID
JOIN CountryOriginCode ON Artist.CountryOriginID = CountryOriginCode.CountryOriginID
JOIN RegionCode ON CountryOriginCode.Region = RegionCode.RegionID
GROUP BY RegionCode.RegionID;
```

Output:



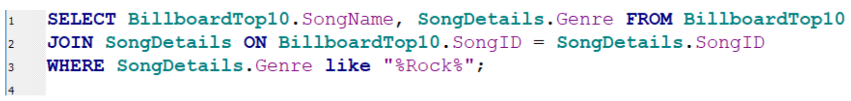
	RegionName	Number_of_Songs
1	Austrailia	2
2	Caribbean	7
3	Europe	16
4	North America	113
5	South America	1

Searching Songs by Genre

Song genres are listed into the database under the attribute genre. This allows users to search for songs by genres as keywords. By using the LIKE function in SQL we can gather a list of songs that are categorized under similar genres. “Pop” songs can be “Dance-Pop” or “Synth-Pop” Etc. To make it uniform, genres in this database has the first letter of each partitioned word capitalized. And by partitioned I mean words that have a non-letter symbol in-between them. Symbols such as a space or a hyphen.

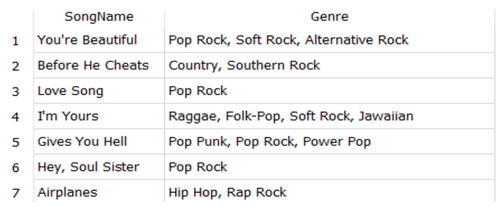
Lets find a list of songs that fall under the “rock” category. To do this we need to join the tables BillboardTop10 and SongDetails and use the LIKE function to search the keyword “rock.

Input:



```sql
SELECT BillboardTop10.SongName, SongDetails.Genre FROM BillboardTop10
JOIN SongDetails ON BillboardTop10.SongID = SongDetails.SongID
WHERE SongDetails.Genre like "%Rock%";
```

Output:



	SongName	Genre
1	You're Beautiful	Pop Rock, Soft Rock, Alternative Rock
2	Before He Cheats	Country, Southern Rock
3	Love Song	Pop Rock
4	I'm Yours	Raggae, Folk-Pop, Soft Rock, Jawaiian
5	Gives You Hell	Pop Punk, Pop Rock, Power Pop
6	Hey, Soul Sister	Pop Rock
7	Airplanes	Hip Hop, Rap Rock

Database Restraints and Additional Features

As shown in Figure 1, the database consists of Billboard information about the songs that made it to the top 10 Billboard list at the end of each year. With the year being listed as integers, users can order the songs by year and observe potential patterns in a timeline. Additionally, various attributes such as song key/scale and genre are listed as text to provide the flexibility for users to group the songs and to create playlists. Runtime is listed in minutes and genre, as explained in the "Searching Songs by Genre" section has the first letter of each partitioned word capitalized.

Song names are listed as text in the database, although some songs are numbers or symbols such as "+" and "22", it is important that users recognize that they are working with texts.

Although most songs are by solo artists, there are several songs by duos and bands. The information about whether an artist is solo or from a group can be found under "type" from the table "Artist". The types include: Solo, Duo, Trio, Band, and Group. For the database, the data only includes up to 2 members of the group. The names are listed under FirstName, LastName, FirstName2, LastName2 from the table "Artist". The data only includes up to 3 song writers and song producers. Their names are listed under SongWriter1, SongWriter2, SongWriter3, Producer1, Producer2, and Producer3 under the table "SongCollab". If there are more writers and producers, it would not be listed anywhere in this database. The table "SongCollab" also contains information about which label the song was released under, the labels are also written as a list and can be searched using keywords and the LIKE function.

Normalization

In this section we will be examining whether my relations in my database satisfies 3NF Normal Forms. This means that there should be no transitive dependencies on the keys of each relation.

Artist:

In this table we have the primary key, ArtistID, has been uniquely created to represent the individual artists in most cases. However when the artist is not "Solo", the normal form breaks down.

When there is more than one artist:

R = (ArtistID, StageName, FirstName, LastName, FirstAlbum, YearOfFirstAlbum, GenderID, Age, CountryOriginID, WeeksOnTopArtist, PeakOnTopArtist, FirstName2, LastName2, Type)

K = ArtistID

Fd1= FirstName, LastName -> Age

Fd2= FirstName, LastName -> GenderID

Fd3 = FirstName, LastName -> CountryOrigin

Fd4= FirstName2, LastName2 -> Age

Fd5= FirstName2, LastName2 -> GenderID

Fd6 = FirstName2, LastName2 -> CountryOrigin

We can in fact combine the FirstName and FirstName2, and LastName and LastName2 to normalize this table

To normalize this table, we need to split the table, R, further into two tables with the following attributes:

R1=(ArtistID, StageName, FirstName, LastName, FirstAlbum, YearOfFirstAlbum, WeeksOnTopArtist, PeakOnTopArtist, FirstName2, LastName2, Type)

R2=(FirstName, LastName, GenderID, Age, CountryOriginID)

BillboardTop10:

This table is in 3NF Normal form as all the attributes in the table depend the key SongID. SongID is the unique integer identification used to represent each unique song in the database. This unique identification can tell us the song name, artist identification, highest rank of the song, year it hit the chart and number of years it hit the chart. Thus, there is no transitive dependencies on this key of the table.

CountryOriginCode:

This table is in 3NF Normal form as all the attributes in the table depend on the key CountryOriginID.

GenderCode:

This table is in 3NF Normal form as all the attributes in the table depend on the key GenderID.

RegionCode:

This table is in 3NF Normal form as all the attributes in the table depend on the key RegionID.

SongCollab:

This table is in 3NF Normal form as all the attributes in the table depend on the key SongID. The song identification tells us about the song being observed. Information such as label, features, writers, and producers can be found by looking up the song id. Thus, there is no transitive dependencies on this key of the table.

SongDetails:

This table is in 3NF Normal form as all the attributes in the table depend on the key SongID. Music details such as BPM, key/scale, year released, genre, runtime and album are all things you can find once you identify the song you are interested in looking up. Thus, there is no transitive dependencies on this key of the table.

Reference

Music information used in the table SongDetails: Tunebat

https://tunebat.com/

Billboard information used in the table BillboardTop10: Billboard

https://www.billboard.com/

Artist Information used throughout the database: Wikipedia

https://www.wikipedia.org/