MUSIC COLLECTION DATABASE

By: Katelyn Boylan

Database Systems

Spring 2017

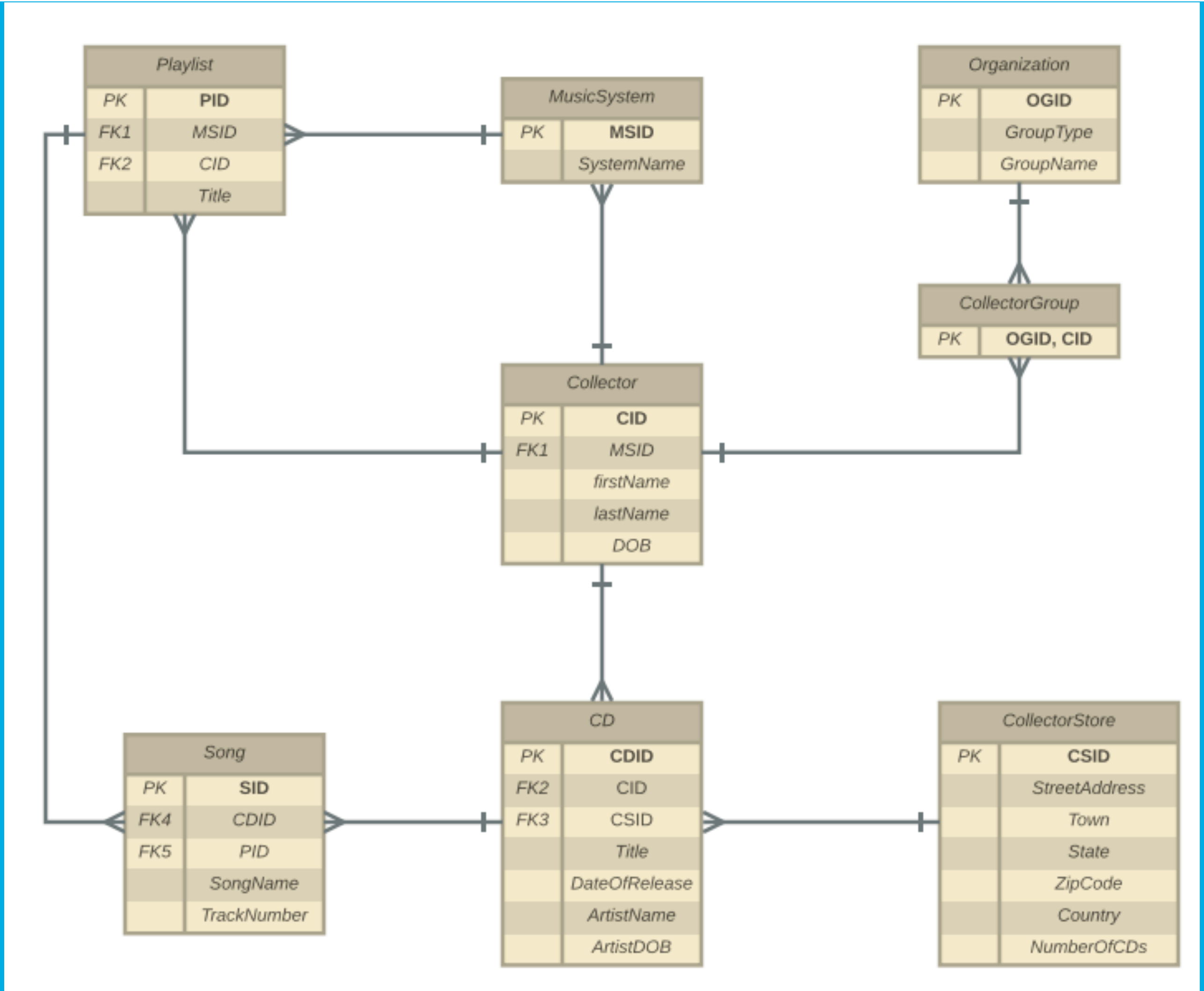
TABLE OF CONTENTS

EXECUTIVE SUMMARY

This ERD database was designed to show the process of sharing music between different groups of people. Specifically, it shows how groups of people download CD's from Collector stores, download music, and create playlists to various Music systems such as iTunes, Spotify, Pandora, and Apple Music. Designed in Postgres, the tables in the database show six collectors who are a part of different organizations that create playlists on different music systems.

The create statements, functional dependencies, and sample data are followed by views, stored procedures, and sample reports. The views and reports manipulate the data in the database while the stored procedure serves as a function for the user. Security and future enhancements are also considered at the end of the project after Implementation notes.

MUSIC COLLECTION ER DIAGRAM



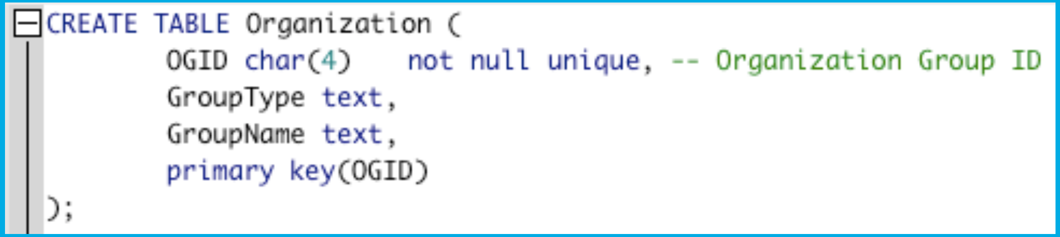
CREATE TABLES

ORGANIZATION TABLE

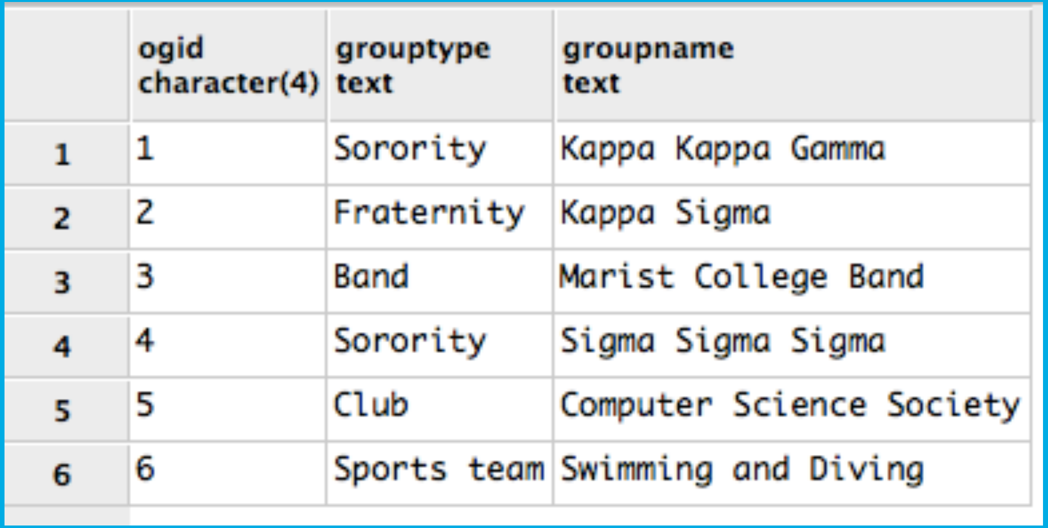
Keeps track of any group that may or may not be a Collector Group. The primary key is "OGID", which will show the user what type of group the Organization is as well as its Group Name.

<u>Functional Dependencies:</u>

OGID $\longrightarrow$ GroupType, GroupName



```
CREATE TABLE Organization (
        OGID char(4)    not null unique, -- Organization Group ID
        GroupType text,
        GroupName text,
        primary key(OGID)
);
```

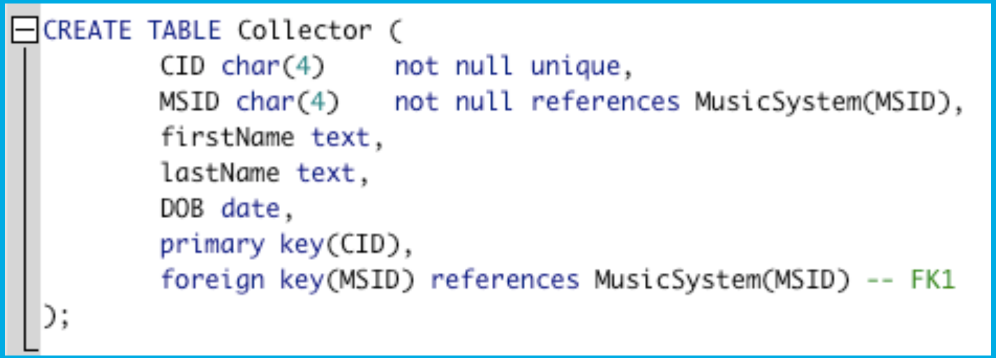


	ogid character(4)	grouptype text	groupname text
1	1	Sorority	Kappa Kappa Gamma
2	2	Fraternity	Kappa Sigma
3	3	Band	Marist College Band
4	4	Sorority	Sigma Sigma Sigma
5	5	Club	Computer Science Society
6	6	Sports team	Swimming and Diving

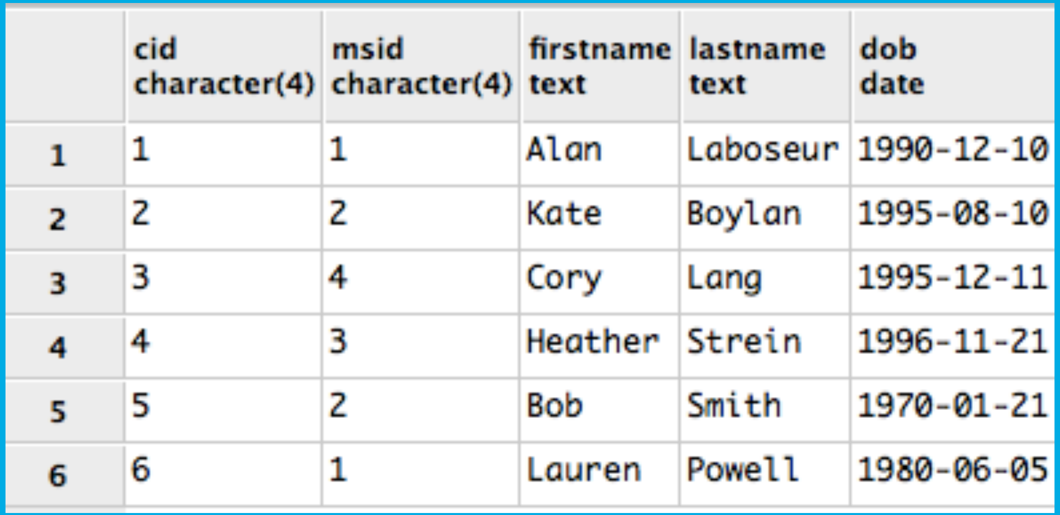
COLLECTOR TABLE

Collectors are people that own one or more music systems and add playlists made up of songs from CDS acquired from Collector stores. Collectors make up one or more CollectorGroups and are known from their "CID".

CID $\rightarrow$ MSID (FK1), firstName, lastName, DOB (Date of Birth)



```
CREATE TABLE Collector (
        CID char(4)     not null unique,
        MSID char(4)    not null references MusicSystem(MSID),
        firstName text,
        lastName text,
        DOB date,
        primary key(CID),
        foreign key(MSID) references MusicSystem(MSID) -- FK1
);
```

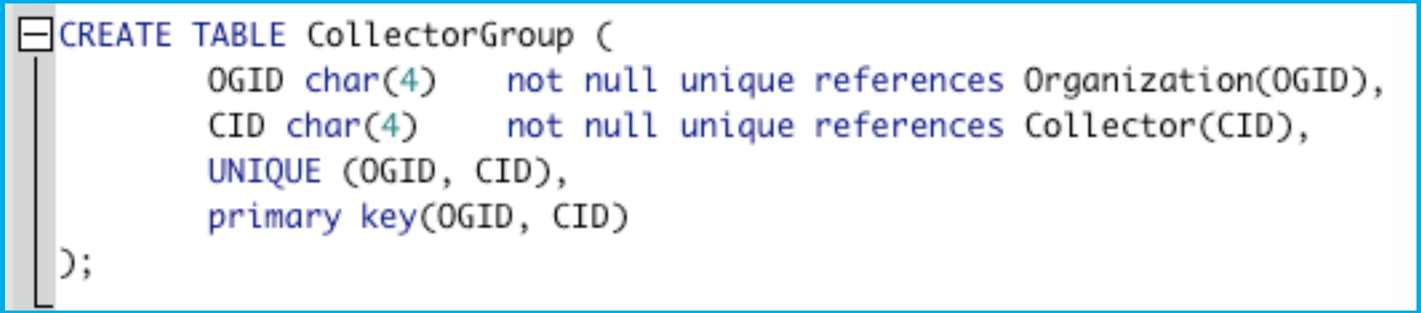


	cid character(4)	msid character(4)	firstname text	lastname text	dob date
1	1	1	Alan	Laboseur	1990-12-10
2	2	2	Kate	Boylan	1995-08-10
3	3	4	Cory	Lang	1995-12-11
4	4	3	Heather	Strein	1996-11-21
5	5	2	Bob	Smith	1970-01-21
6	6	1	Lauren	Powell	1980-06-05

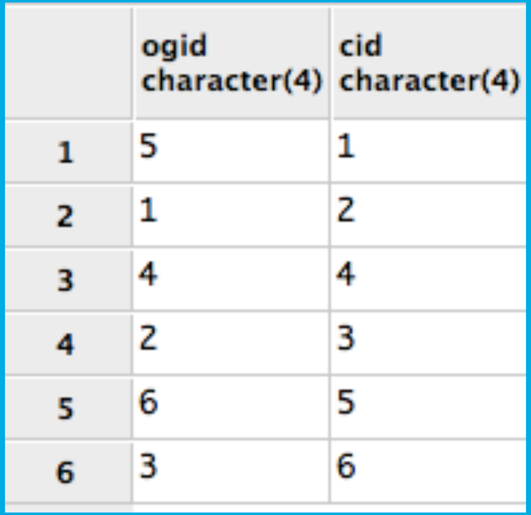
COLLECTOR GROUP TABLE

Collector Group: Groups of people from the Organization table with a name. The primary key is a composite of the two tables.

OGID, CID $\rightarrow$ OGID, CID



```
CREATE TABLE CollectorGroup (
        OGID char(4)    not null unique references Organization(OGID),
        CID char(4)     not null unique references Collector(CID),
        UNIQUE (OGID, CID),
        primary key(OGID, CID)
);
```

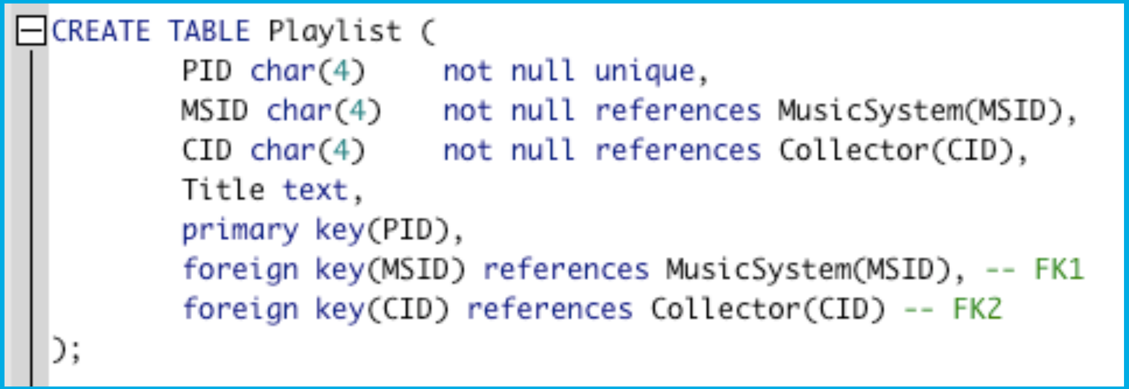


	ogid character(4)	cid character(4)
1	5	1
2	1	2
3	4	4
4	2	3
5	6	5
6	3	6

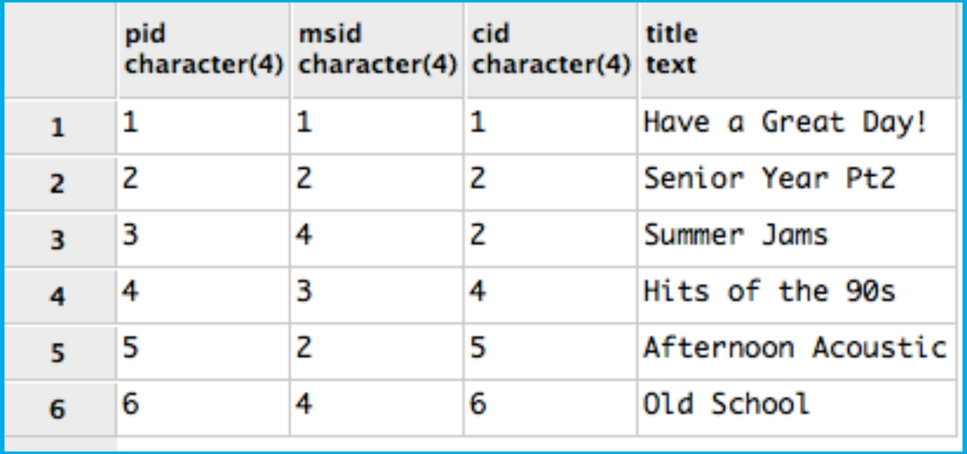
PLAYLIST TABLE

Playlists are created by Collectors on a Music System. Playlists are made up of many songs and have their own title, chosen by the Collector. Playlists are unique by their "PID"

PID → MSID (FK1), CID (FK2), Title



```
CREATE TABLE Playlist (
        PID char(4)     not null unique,
        MSID char(4)    not null references MusicSystem(MSID),
        CID char(4)     not null references Collector(CID),
        Title text,
        primary key(PID),
        foreign key(MSID) references MusicSystem(MSID), -- FK1
        foreign key(CID) references Collector(CID) -- FK2
);
```

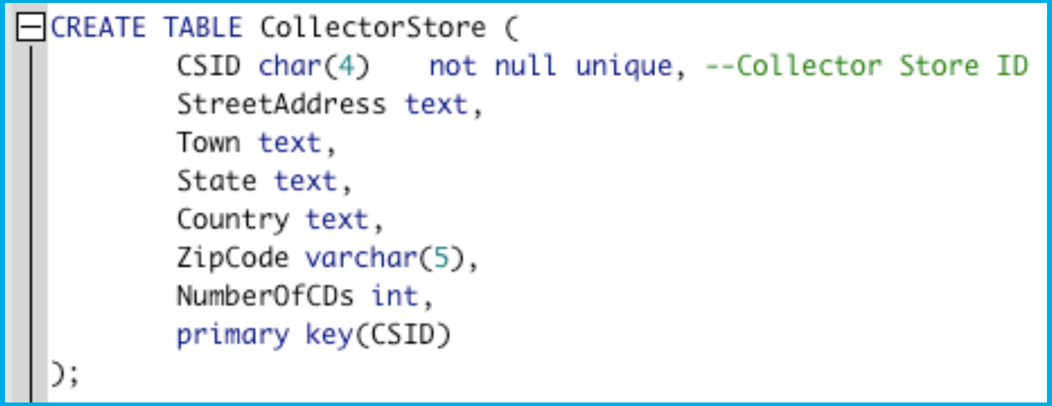


	pid character(4)	msid character(4)	cid character(4)	title text
1	1	1	1	Have a Great Day!
2	2	2	2	Senior Year Pt2
3	3	4	2	Summer Jams
4	4	3	4	Hits of the 90s
5	5	2	5	Afternoon Acoustic
6	6	4	6	Old School

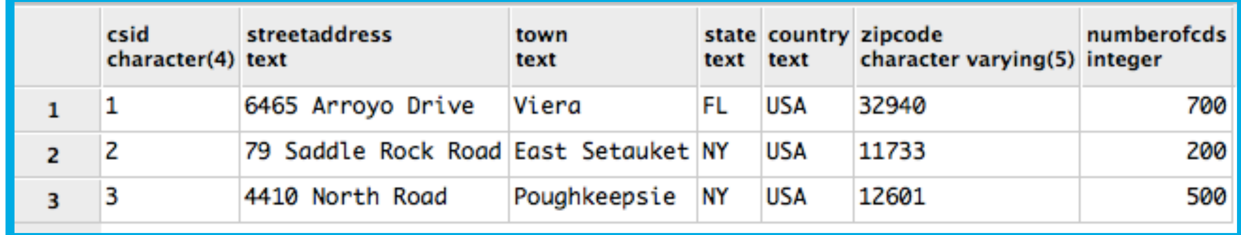
COLLECTOR STORE TABLE

Collector Stores hold CDs that Collectors can buy in order to download songs onto their playlists on their Music System. Collector Stores have a certain amount of CDs in inventory. Collector Stores are unique from their "CSID."

CSID $\rightarrow$ StreetAddress, State, Town, ZipCode, Country, NumberOfCDs



```
CREATE TABLE CollectorStore (
        CSID char(4)    not null unique, --Collector Store ID
        StreetAddress text,
        Town text,
        State text,
        Country text,
        ZipCode varchar(5),
        NumberOfCDs int,
        primary key(CSID)
);
```

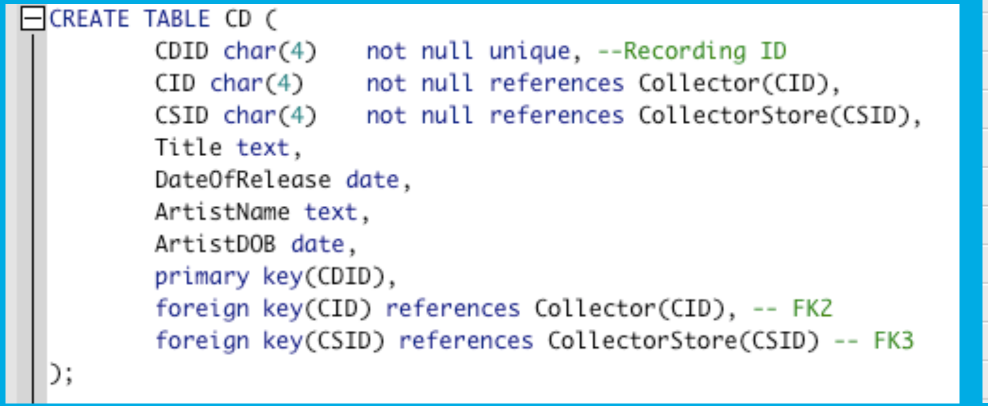


	csid character(4)	streetaddress text	town text	state text	country text	zipcode character varying(5)	numberofcds integer
1	1	6465 Arroyo Drive	Viera	FL	USA	32940	700
2	2	79 Saddle Rock Road	East Setauket	NY	USA	11733	200
3	3	4410 North Road	Poughkeepsie	NY	USA	12601	500

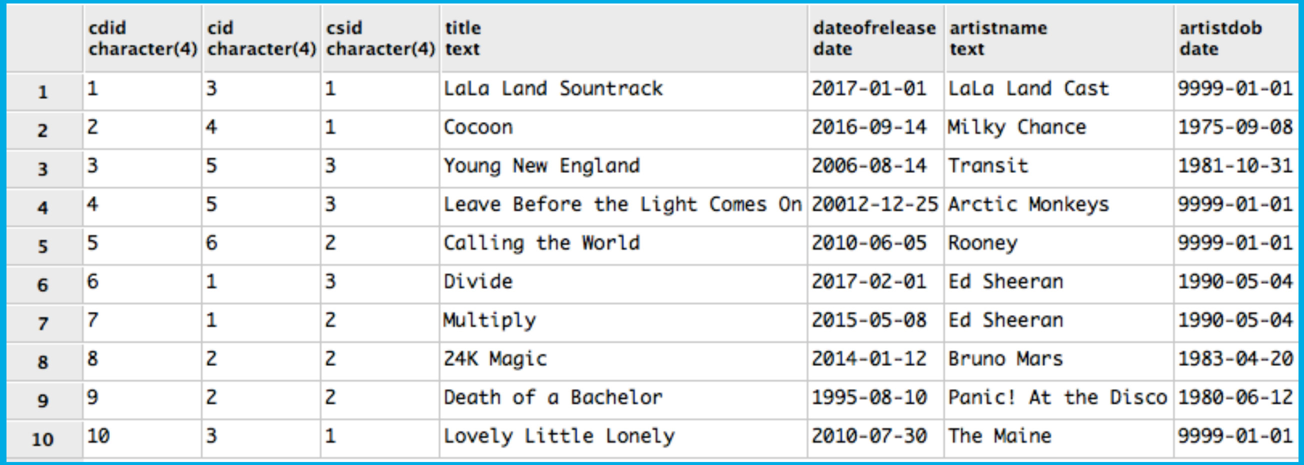
CD TABLE

CDs are bought and owned by Collectors from Collector Stores. CDs have a title, a Date of Release, an Artist name, and the Artists have a Date of Birth. CDs are made up of many songs that are ultimately put on a playlist. CDs are unique by their “CDID”.

CDID $\rightarrow$ CID (FK2), CSID (FK3), Title, DateOfRelease, ArtistName, ArtistDOB



```
CREATE TABLE CD (
        CDID char(4)    not null unique, --Recording ID
        CID char(4)     not null references Collector(CID),
        CSID char(4)    not null references CollectorStore(CSID),
        Title text,
        DateOfRelease date,
        ArtistName text,
        ArtistDOB date,
        primary key(CDID),
        foreign key(CID) references Collector(CID), -- FK2
        foreign key(CSID) references CollectorStore(CSID) -- FK3
);
```

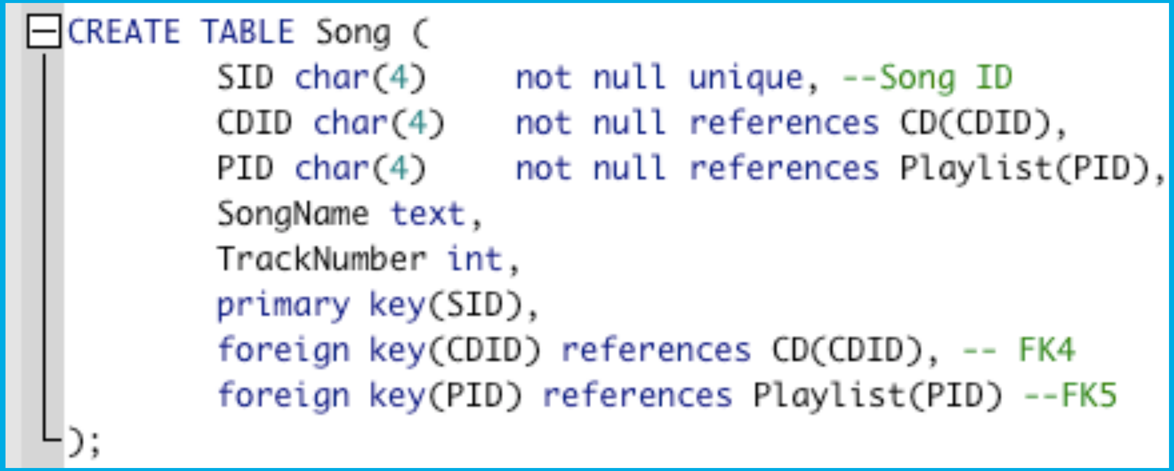


	cdid character(4)	cid character(4)	csid character(4)	title text	dateofrelease date	artistname text	artistdob date
1	1	3	1	LaLa Land Sountrack	2017-01-01	LaLa Land Cast	9999-01-01
2	2	4	1	Cocoon	2016-09-14	Milky Chance	1975-09-08
3	3	5	3	Young New England	2006-08-14	Transit	1981-10-31
4	4	5	3	Leave Before the Light Comes On	20012-12-25	Arctic Monkeys	9999-01-01
5	5	6	2	Calling the World	2010-06-05	Rooney	9999-01-01
6	6	1	3	Divide	2017-02-01	Ed Sheeran	1990-05-04
7	7	1	2	Multiply	2015-05-08	Ed Sheeran	1990-05-04
8	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
9	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
10	10	3	1	Lovely Little Lonely	2010-07-30	The Maine	9999-01-01

SONG TABLE

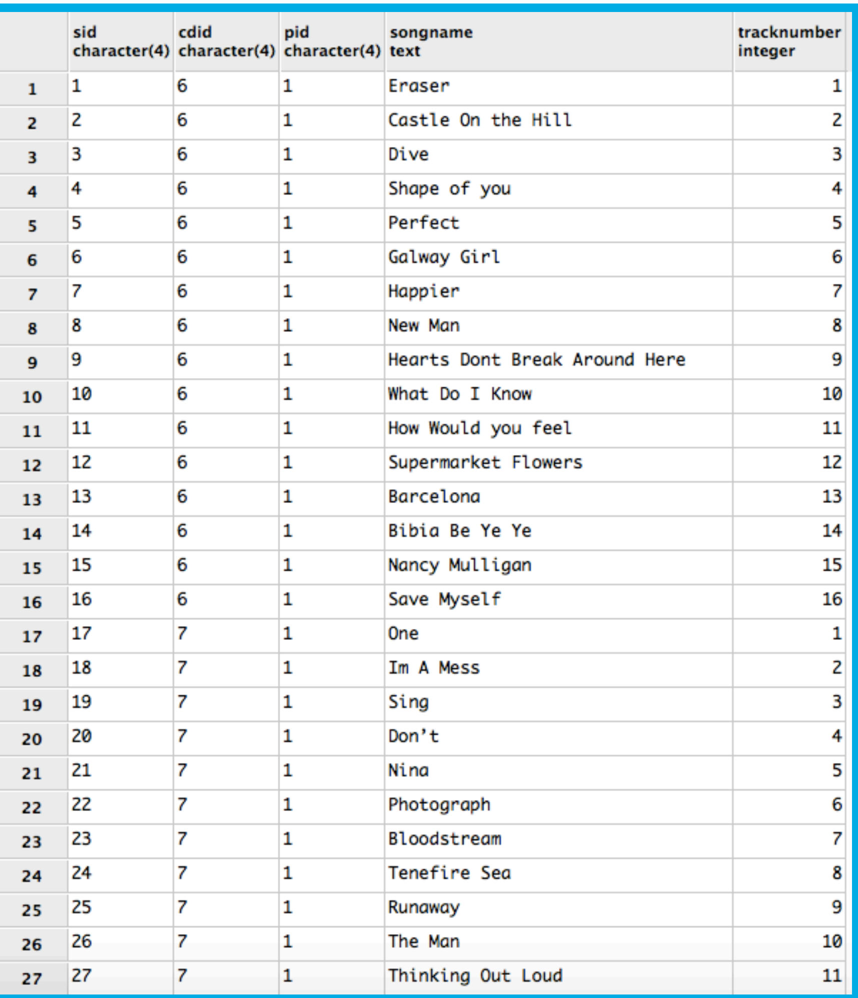
Songs are downloaded by collectors from CDs that are bought in Collector Stores. Many songs make up one playlist and one CD. Songs have a SongName and TrackNumber on its CD and they are unique by their "SID".

SID $\longrightarrow$ CDID (FK4), PID(FK5), SongName, TrackNumber

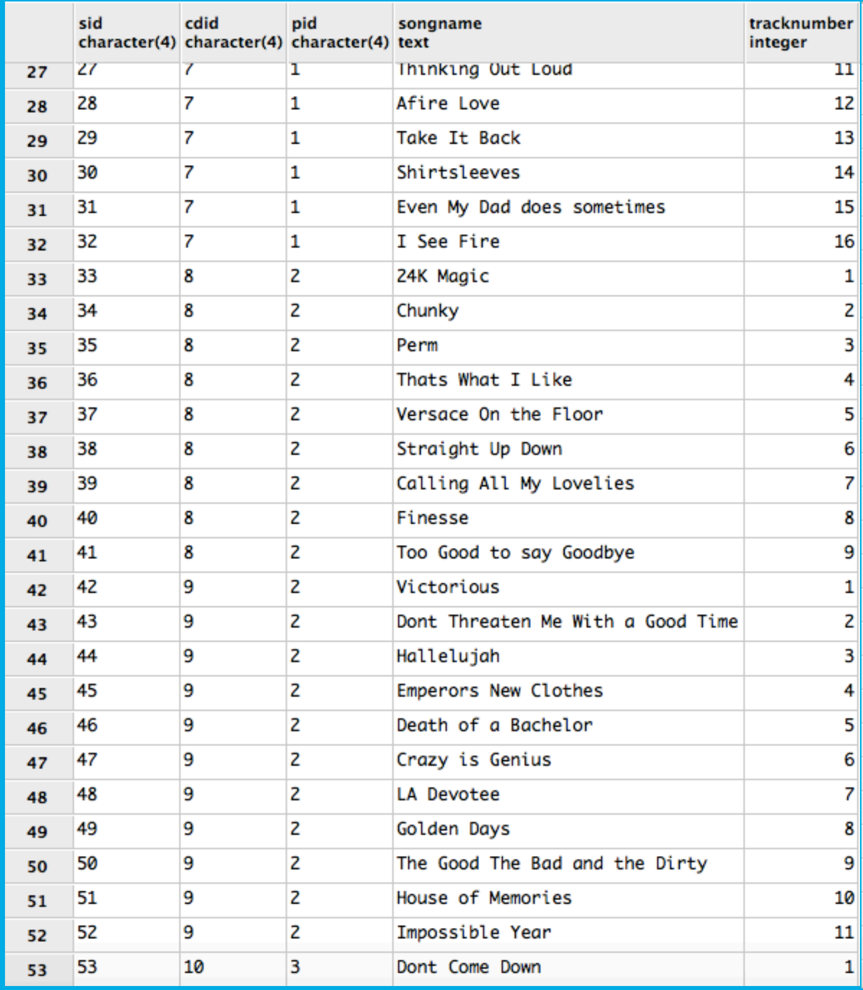


```
CREATE TABLE Song (
        SID char(4)     not null unique, --Song ID
        CDID char(4)    not null references CD(CDID),
        PID char(4)     not null references Playlist(PID),
        SongName text,
        TrackNumber int,
        primary key(SID),
        foreign key(CDID) references CD(CDID), -- FK4
        foreign key(PID) references Playlist(PID) --FK5
);
```

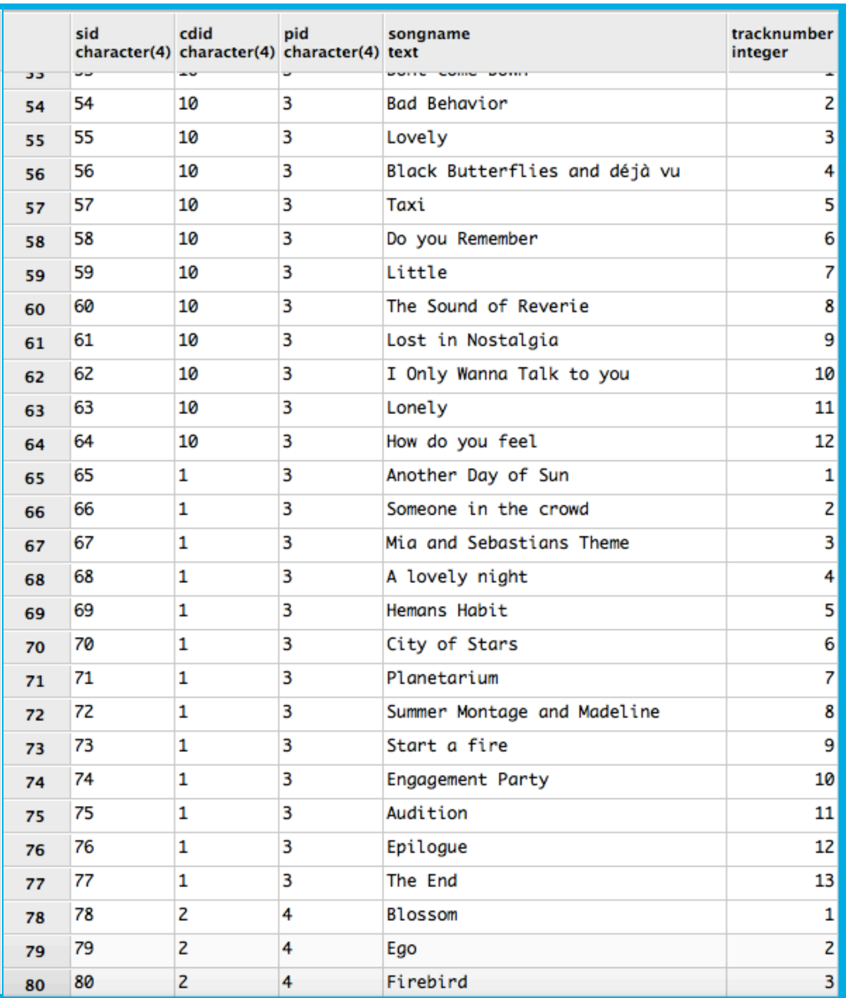
SONG TABLE EXAMPLE



	sid character(4)	cdid character(4)	pid character(4)	songname text	tracknumber integer
1	1	6	1	Eraser	1
2	2	6	1	Castle On the Hill	2
3	3	6	1	Dive	3
4	4	6	1	Shape of you	4
5	5	6	1	Perfect	5
6	6	6	1	Galway Girl	6
7	7	6	1	Happier	7
8	8	6	1	New Man	8
9	9	6	1	Hearts Dont Break Around Here	9
10	10	6	1	What Do I Know	10
11	11	6	1	How Would you feel	11
12	12	6	1	Supermarket Flowers	12
13	13	6	1	Barcelona	13
14	14	6	1	Bibia Be Ye Ye	14
15	15	6	1	Nancy Mulligan	15
16	16	6	1	Save Myself	16
17	17	7	1	One	1
18	18	7	1	Im A Mess	2
19	19	7	1	Sing	3
20	20	7	1	Don't	4
21	21	7	1	Nina	5
22	22	7	1	Photograph	6
23	23	7	1	Bloodstream	7
24	24	7	1	Tenefire Sea	8
25	25	7	1	Runaway	9
26	26	7	1	The Man	10
27	27	7	1	Thinking Out Loud	11

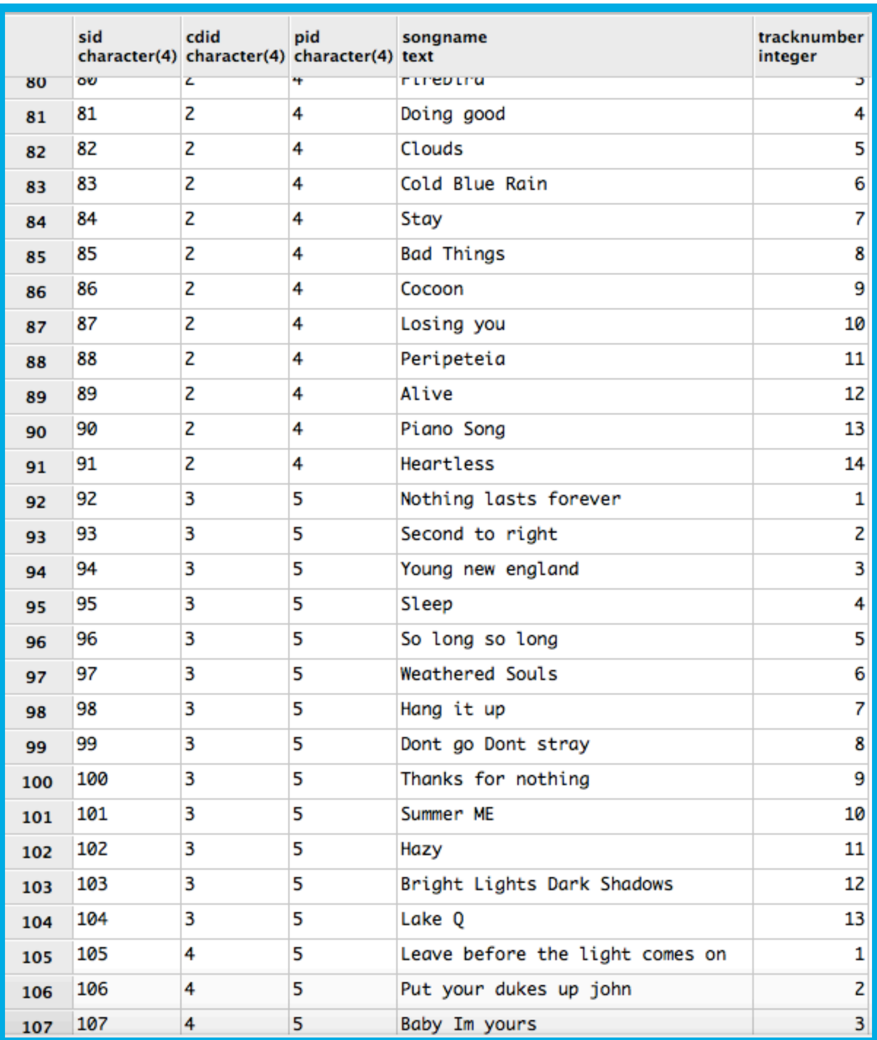


	sid character(4)	cdid character(4)	pid character(4)	songname text	tracknumber integer
27	27	7	1	Thinking Out Loud	11
28	28	7	1	Afire Love	12
29	29	7	1	Take It Back	13
30	30	7	1	Shirtsleeves	14
31	31	7	1	Even My Dad does sometimes	15
32	32	7	1	I See Fire	16
33	33	8	2	24K Magic	1
34	34	8	2	Chunky	2
35	35	8	2	Perm	3
36	36	8	2	Thats What I Like	4
37	37	8	2	Versace On the Floor	5
38	38	8	2	Straight Up Down	6
39	39	8	2	Calling All My Lovelies	7
40	40	8	2	Finesse	8
41	41	8	2	Too Good to say Goodbye	9
42	42	9	2	Victorious	1
43	43	9	2	Dont Threaten Me With a Good Time	2
44	44	9	2	Hallelujah	3
45	45	9	2	Emperors New Clothes	4
46	46	9	2	Death of a Bachelor	5
47	47	9	2	Crazy is Genius	6
48	48	9	2	LA Devotee	7
49	49	9	2	Golden Days	8
50	50	9	2	The Good The Bad and the Dirty	9
51	51	9	2	House of Memories	10
52	52	9	2	Impossible Year	11
53	53	10	3	Dont Come Down	1

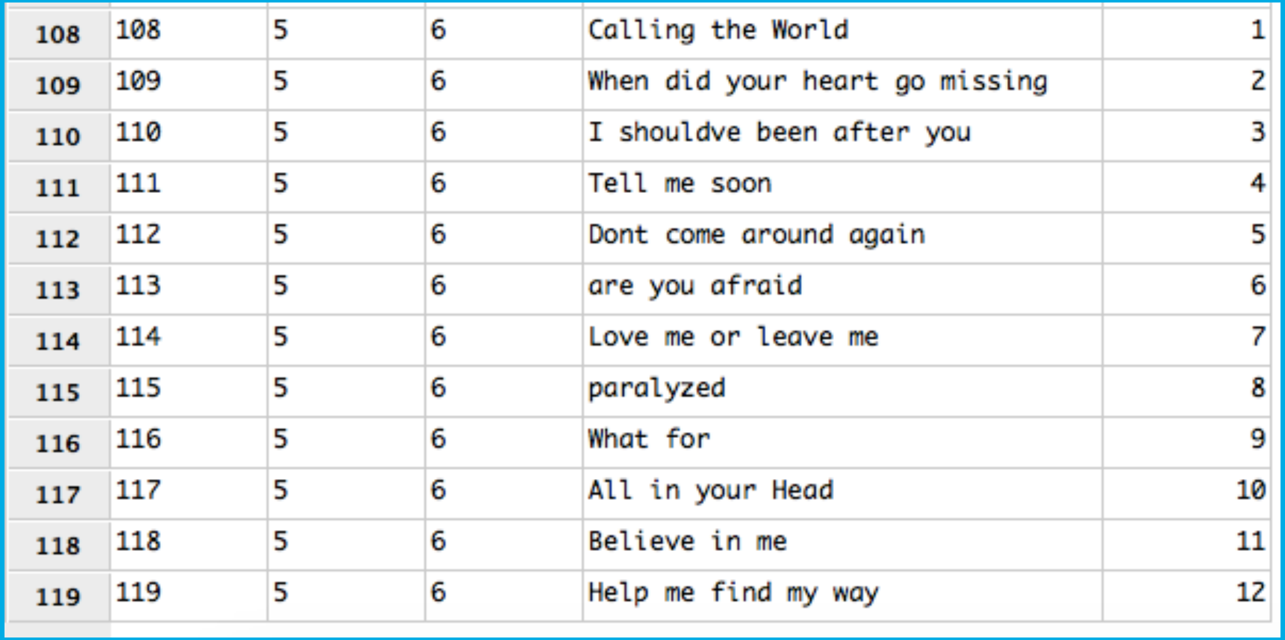


	sid character(4)	cdid character(4)	pid character(4)	songname text	tracknumber integer
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
54	54	10	3	Bad Behavior	2
55	55	10	3	Lovely	3
56	56	10	3	Black Butterflies and déjà vu	4
57	57	10	3	Taxi	5
58	58	10	3	Do you Remember	6
59	59	10	3	Little	7
60	60	10	3	The Sound of Reverie	8
61	61	10	3	Lost in Nostalgia	9
62	62	10	3	I Only Wanna Talk to you	10
63	63	10	3	Lonely	11
64	64	10	3	How do you feel	12
65	65	1	3	Another Day of Sun	1
66	66	1	3	Someone in the crowd	2
67	67	1	3	Mia and Sebastians Theme	3
68	68	1	3	A lovely night	4
69	69	1	3	Hemans Habit	5
70	70	1	3	City of Stars	6
71	71	1	3	Planetarium	7
72	72	1	3	Summer Montage and Madeline	8
73	73	1	3	Start a fire	9
74	74	1	3	Engagement Party	10
75	75	1	3	Audition	11
76	76	1	3	Epilogue	12
77	77	1	3	The End	13
78	78	2	4	Blossom	1
79	79	2	4	Ego	2
80	80	2	4	Firebird	3

SONG TABLE EXAMPLE CONTINUED



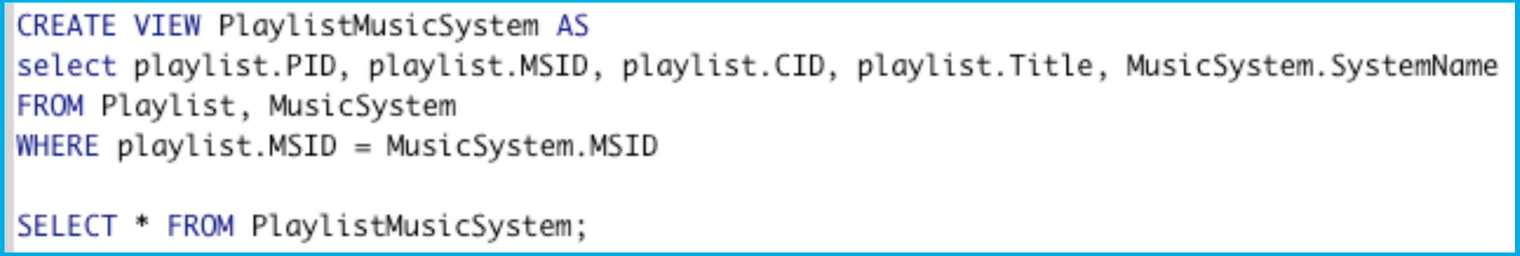
	sid character(4)	cdid character(4)	pid character(4)	songname text	tracknumber integer
80	80	2	4	Firebird	3
81	81	2	4	Doing good	4
82	82	2	4	Clouds	5
83	83	2	4	Cold Blue Rain	6
84	84	2	4	Stay	7
85	85	2	4	Bad Things	8
86	86	2	4	Cocoon	9
87	87	2	4	Losing you	10
88	88	2	4	Peripeteia	11
89	89	2	4	Alive	12
90	90	2	4	Piano Song	13
91	91	2	4	Heartless	14
92	92	3	5	Nothing lasts forever	1
93	93	3	5	Second to right	2
94	94	3	5	Young new england	3
95	95	3	5	Sleep	4
96	96	3	5	So long so long	5
97	97	3	5	Weathered Souls	6
98	98	3	5	Hang it up	7
99	99	3	5	Dont go Dont stray	8
100	100	3	5	Thanks for nothing	9
101	101	3	5	Summer ME	10
102	102	3	5	Hazy	11
103	103	3	5	Bright Lights Dark Shadows	12
104	104	3	5	Lake Q	13
105	105	4	5	Leave before the light comes on	1
106	106	4	5	Put your dukes up john	2
107	107	4	5	Baby Im yours	3



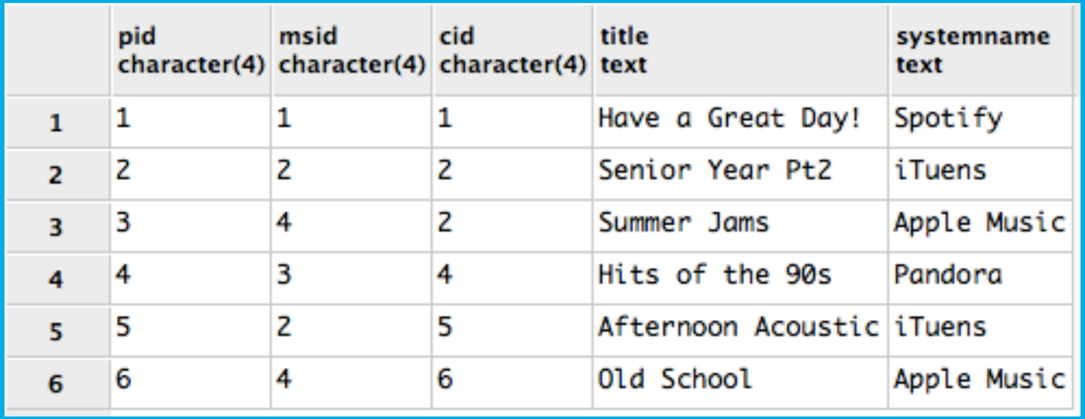
108	108	5	6	Calling the World	1
109	109	5	6	When did your heart go missing	2
110	110	5	6	I shouldve been after you	3
111	111	5	6	Tell me soon	4
112	112	5	6	Dont come around again	5
113	113	5	6	are you afraid	6
114	114	5	6	Love me or leave me	7
115	115	5	6	paralyzed	8
116	116	5	6	What for	9
117	117	5	6	All in your Head	10
118	118	5	6	Believe in me	11
119	119	5	6	Help me find my way	12

VIEWS

1. PlaylistMusicSystem View connects MSID to Music System Name for each playlist.



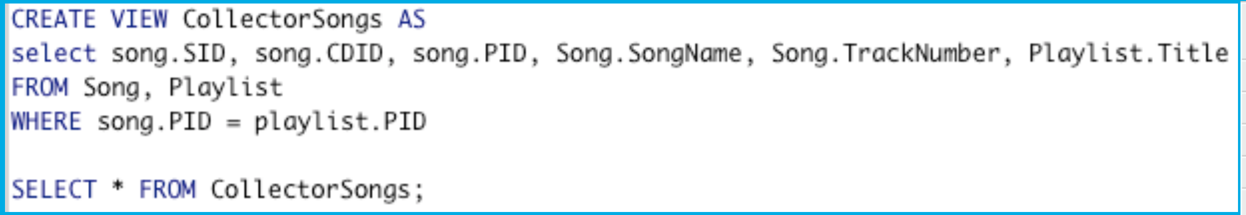
```
CREATE VIEW PlaylistMusicSystem AS
select playlist.PID, playlist.MSID, playlist.CID, playlist.Title, MusicSystem.SystemName
FROM Playlist, MusicSystem
WHERE playlist.MSID = MusicSystem.MSID

SELECT * FROM PlaylistMusicSystem;
```



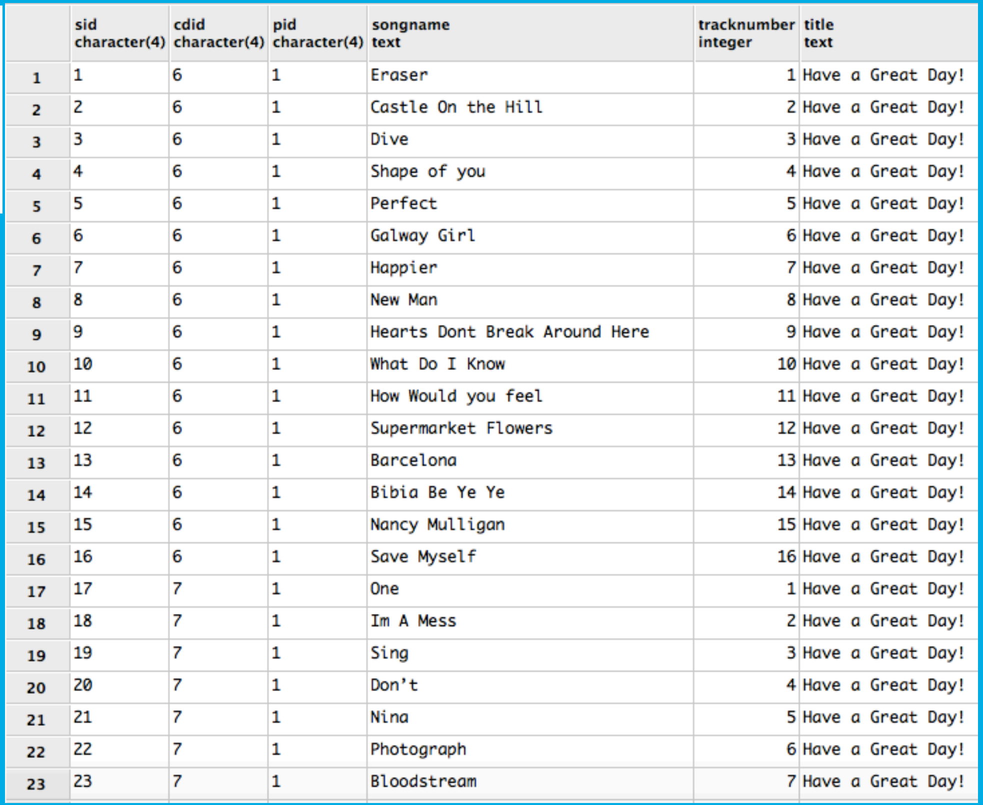
	pid character(4)	msid character(4)	cid character(4)	title text	systemname text
1	1	1	1	Have a Great Day!	Spotify
2	2	2	2	Senior Year Pt2	iTuens
3	3	4	2	Summer Jams	Apple Music
4	4	3	4	Hits of the 90s	Pandora
5	5	2	5	Afternoon Acoustic	iTuens
6	6	4	6	Old School	Apple Music

VIEWS

CollectorSongs View connects each song to its playlist while showing the playlist name.



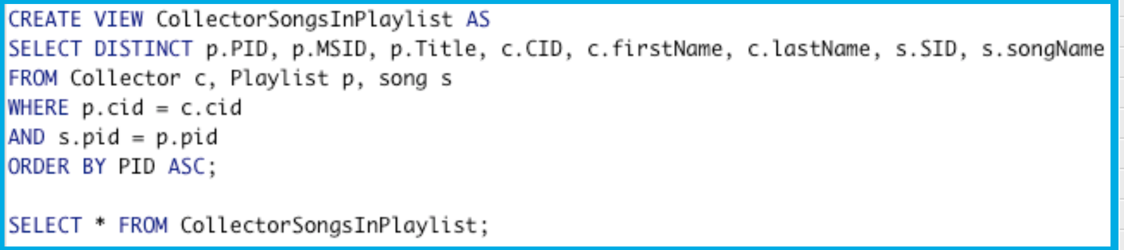
```sql
CREATE VIEW CollectorSongs AS
select song.SID, song.CDID, song.PID, Song.SongName, Song.TrackNumber, Playlist.Title
FROM Song, Playlist
WHERE song.PID = playlist.PID

SELECT * FROM CollectorSongs;
```



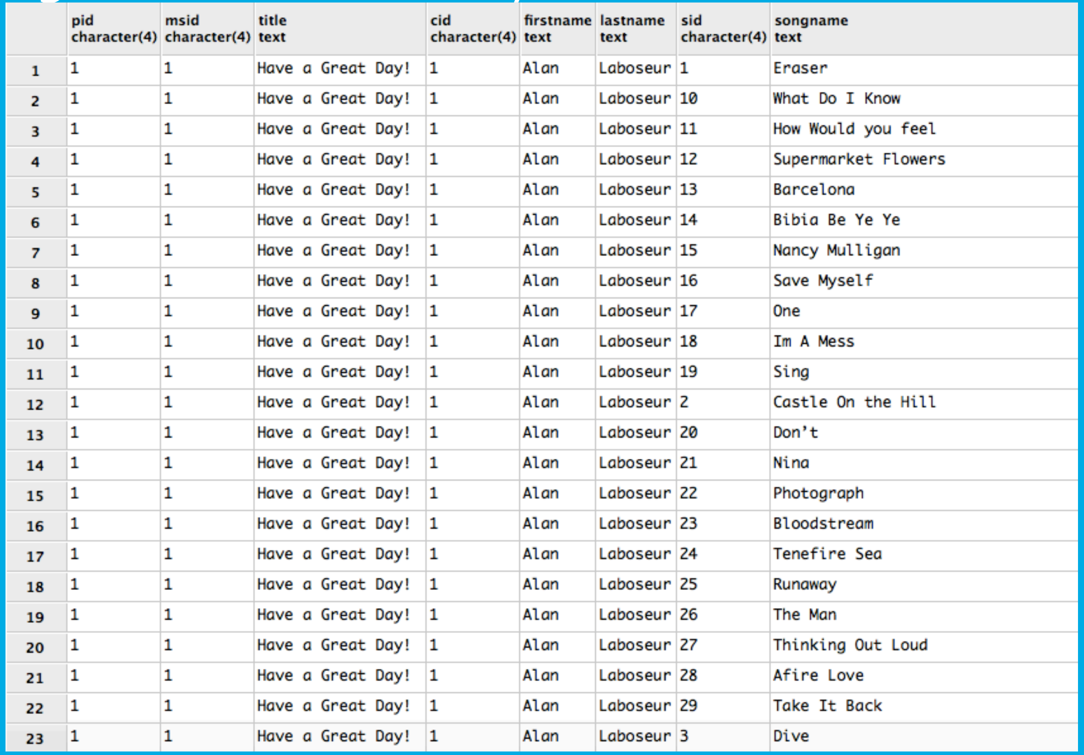
	sid	cdid	pid	songname	tracknumber	title
	character(4)	character(4)	character(4)	text	integer	text
1	1	6	1	Eraser	1	Have a Great Day!
2	2	6	1	Castle On the Hill	2	Have a Great Day!
3	3	6	1	Dive	3	Have a Great Day!
4	4	6	1	Shape of you	4	Have a Great Day!
5	5	6	1	Perfect	5	Have a Great Day!
6	6	6	1	Galway Girl	6	Have a Great Day!
7	7	6	1	Happier	7	Have a Great Day!
8	8	6	1	New Man	8	Have a Great Day!
9	9	6	1	Hearts Dont Break Around Here	9	Have a Great Day!
10	10	6	1	What Do I Know	10	Have a Great Day!
11	11	6	1	How Would you feel	11	Have a Great Day!
12	12	6	1	Supermarket Flowers	12	Have a Great Day!
13	13	6	1	Barcelona	13	Have a Great Day!
14	14	6	1	Bibia Be Ye Ye	14	Have a Great Day!
15	15	6	1	Nancy Mulligan	15	Have a Great Day!
16	16	6	1	Save Myself	16	Have a Great Day!
17	17	7	1	One	1	Have a Great Day!
18	18	7	1	Im A Mess	2	Have a Great Day!
19	19	7	1	Sing	3	Have a Great Day!
20	20	7	1	Don't	4	Have a Great Day!
21	21	7	1	Nina	5	Have a Great Day!
22	22	7	1	Photograph	6	Have a Great Day!
23	23	7	1	Bloodstream	7	Have a Great Day!

VIEWS

CollectorSongsInPlaylist View: Combines the Playlists, Collector, and Song tables. The Collector name is pulled in order for the user to easily visualize who (first and last name) made which Playlists with what songs and what the Playlists are called.



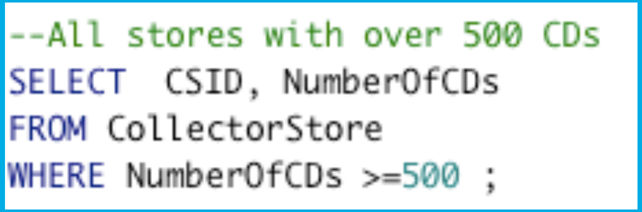
```
CREATE VIEW CollectorSongsInPlaylist AS
SELECT DISTINCT p.PID, p.MSID, p.Title, c.CID, c.firstName, c.lastName, s.SID, s.songName
FROM Collector c, Playlist p, song s
WHERE p.cid = c.cid
AND s.pid = p.pid
ORDER BY PID ASC;

SELECT * FROM CollectorSongsInPlaylist;
```



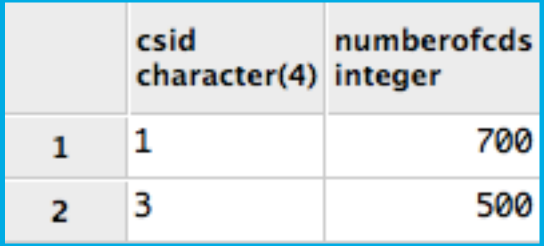
	pid character(4)	msid character(4)	title	cid character(4)	firstname text	lastname text	sid character(4)	songname text
1	1	1	Have a Great Day!	1	Alan	Laboseur	1	Eraser
2	1	1	Have a Great Day!	1	Alan	Laboseur	10	What Do I Know
3	1	1	Have a Great Day!	1	Alan	Laboseur	11	How Would you feel
4	1	1	Have a Great Day!	1	Alan	Laboseur	12	Supermarket Flowers
5	1	1	Have a Great Day!	1	Alan	Laboseur	13	Barcelona
6	1	1	Have a Great Day!	1	Alan	Laboseur	14	Bibia Be Ye Ye
7	1	1	Have a Great Day!	1	Alan	Laboseur	15	Nancy Mulligan
8	1	1	Have a Great Day!	1	Alan	Laboseur	16	Save Myself
9	1	1	Have a Great Day!	1	Alan	Laboseur	17	One
10	1	1	Have a Great Day!	1	Alan	Laboseur	18	Im A Mess
11	1	1	Have a Great Day!	1	Alan	Laboseur	19	Sing
12	1	1	Have a Great Day!	1	Alan	Laboseur	2	Castle On the Hill
13	1	1	Have a Great Day!	1	Alan	Laboseur	20	Don't
14	1	1	Have a Great Day!	1	Alan	Laboseur	21	Nina
15	1	1	Have a Great Day!	1	Alan	Laboseur	22	Photograph
16	1	1	Have a Great Day!	1	Alan	Laboseur	23	Bloodstream
17	1	1	Have a Great Day!	1	Alan	Laboseur	24	Tenefire Sea
18	1	1	Have a Great Day!	1	Alan	Laboseur	25	Runaway
19	1	1	Have a Great Day!	1	Alan	Laboseur	26	The Man
20	1	1	Have a Great Day!	1	Alan	Laboseur	27	Thinking Out Loud
21	1	1	Have a Great Day!	1	Alan	Laboseur	28	Afire Love
22	1	1	Have a Great Day!	1	Alan	Laboseur	29	Take It Back
23	1	1	Have a Great Day!	1	Alan	Laboseur	3	Dive

REPORTS

1.

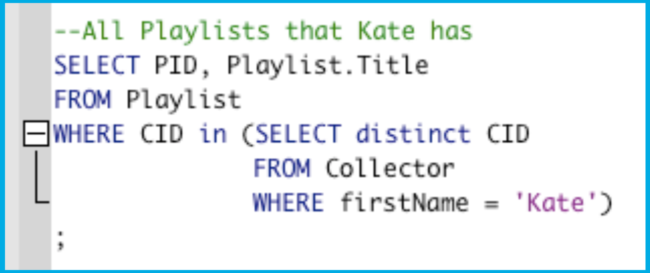


```
--All stores with over 500 CDs
SELECT  CSID, NumberOfCDs
FROM CollectorStore
WHERE NumberOfCDs >=500 ;
```

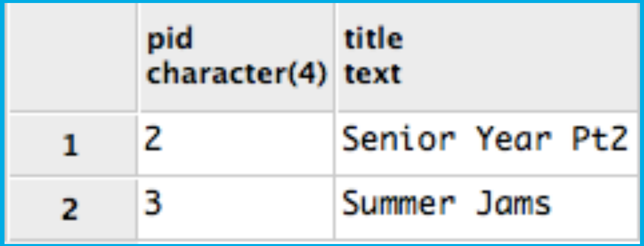


	csid character(4)	numberofcds integer
1	1	700
2	3	500

2.

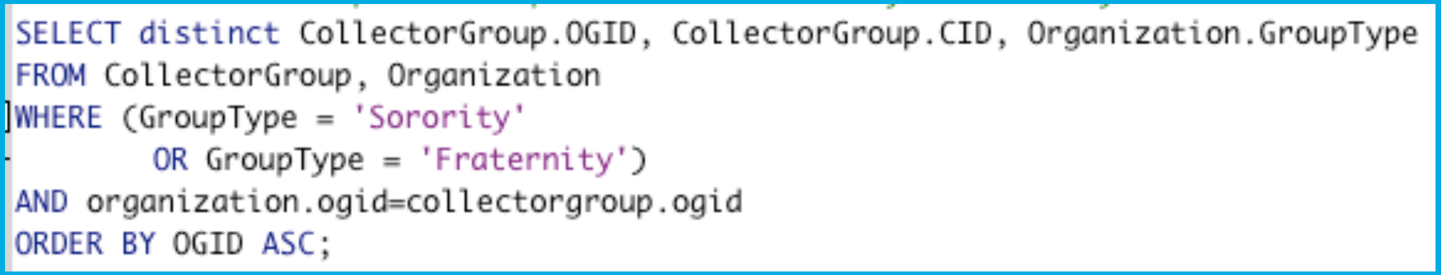


```
--All Playlists that Kate has
SELECT PID, Playlist.Title
FROM Playlist
WHERE CID in (SELECT distinct CID
              FROM Collector
              WHERE firstName = 'Kate')
;
```

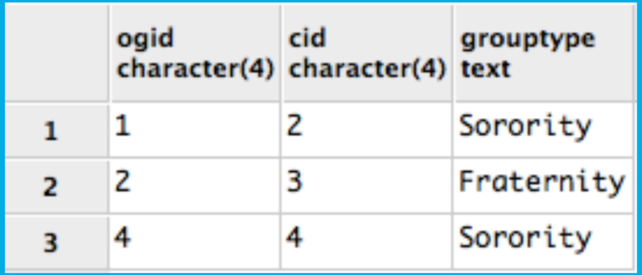


	pid character(4)	title text
1	2	Senior Year Pt2
2	3	Summer Jams

3.



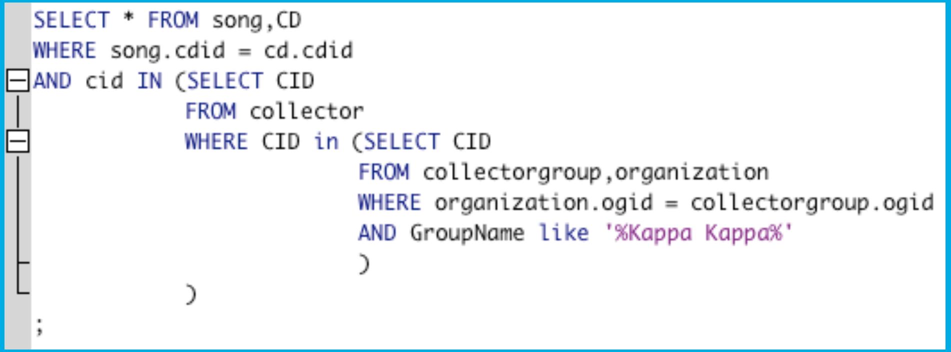
```
SELECT distinct CollectorGroup.OGID, CollectorGroup.CID, Organization.GroupType
FROM CollectorGroup, Organization
WHERE (GroupType = 'Sorority'
       OR GroupType = 'Fraternity')
AND organization.ogid=collectorgroup.ogid
ORDER BY OGID ASC;
```



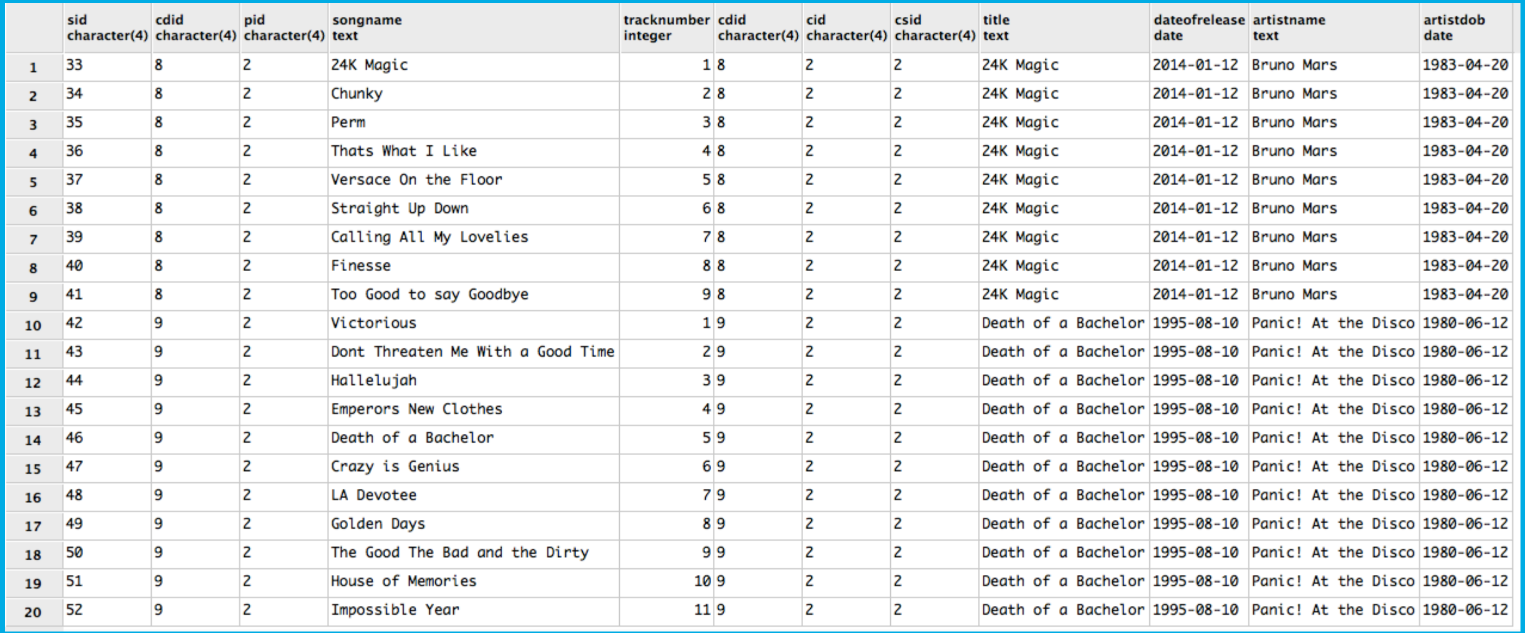
	ogid character(4)	cid character(4)	grouptype text
1	1	2	Sorority
2	2	3	Fraternity
3	4	4	Sorority

REPORTS

4.



```sql
SELECT * FROM song,CD
WHERE song.cdid = cd.cdid
AND cid IN (SELECT CID
            FROM collector
            WHERE CID in (SELECT CID
                          FROM collectorgroup,organization
                          WHERE organization.ogid = collectorgroup.ogid
                          AND GroupName like '%Kappa Kappa%'
                          )
            )
;
```



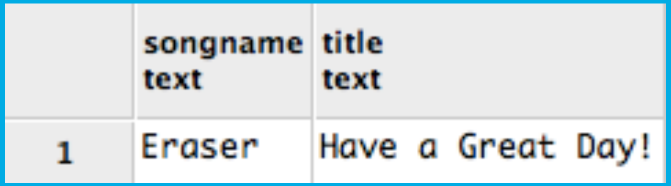
	sid character(4)	cdid character(4)	pid character(4)	songname text	tracknumber integer	cdid character(4)	cid character(4)	csid character(4)	title text	dateofrelease date	artistname text	artistdob date
1	33	8	2	24K Magic	1	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
2	34	8	2	Chunky	2	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
3	35	8	2	Perm	3	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
4	36	8	2	Thats What I Like	4	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
5	37	8	2	Versace On the Floor	5	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
6	38	8	2	Straight Up Down	6	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
7	39	8	2	Calling All My Lovelies	7	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
8	40	8	2	Finesse	8	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
9	41	8	2	Too Good to say Goodbye	9	8	2	2	24K Magic	2014-01-12	Bruno Mars	1983-04-20
10	42	9	2	Victorious	1	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
11	43	9	2	Dont Threaten Me With a Good Time	2	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
12	44	9	2	Hallelujah	3	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
13	45	9	2	Emperors New Clothes	4	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
14	46	9	2	Death of a Bachelor	5	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
15	47	9	2	Crazy is Genius	6	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
16	48	9	2	LA Devotee	7	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
17	49	9	2	Golden Days	8	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
18	50	9	2	The Good The Bad and the Dirty	9	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
19	51	9	2	House of Memories	10	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12
20	52	9	2	Impossible Year	11	9	2	2	Death of a Bachelor	1995-08-10	Panic! At the Disco	1980-06-12

STORED PROCEDURE

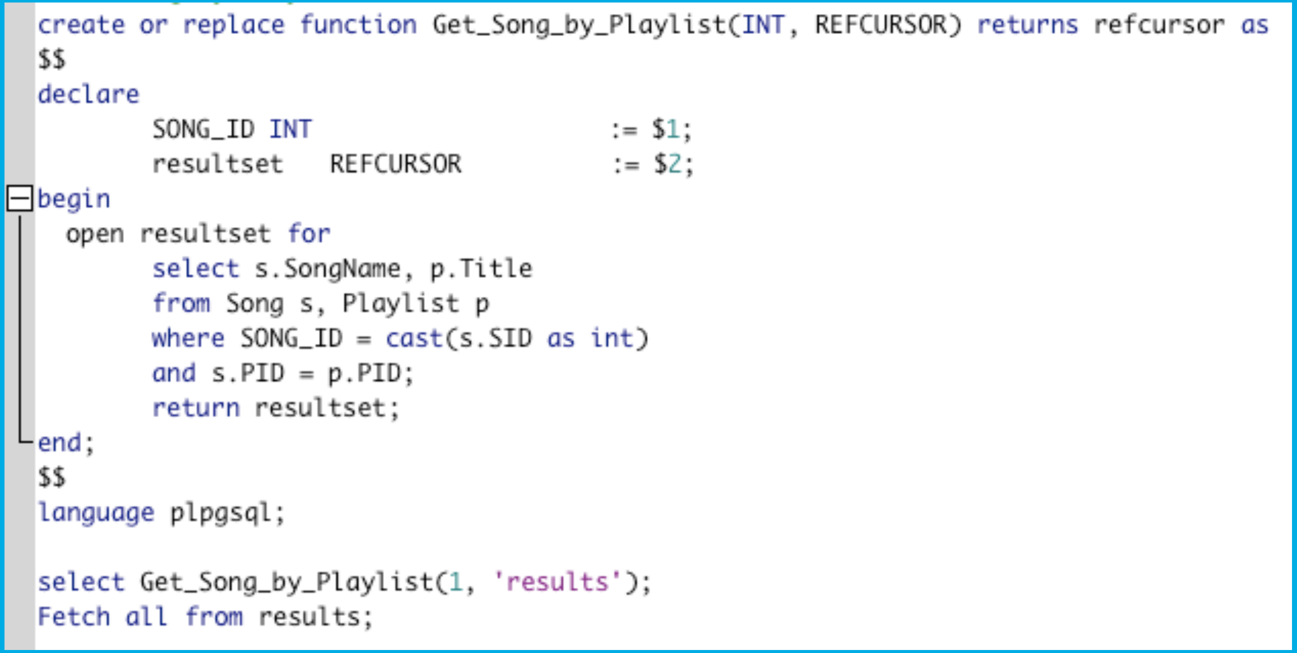
Get_Song_By_Playlist: By creating this Stored Procedure, a user can easily find a song by playlist.

By putting in any SID in the stored procedure, the user can easily see which Playlist that song is on.

Output:



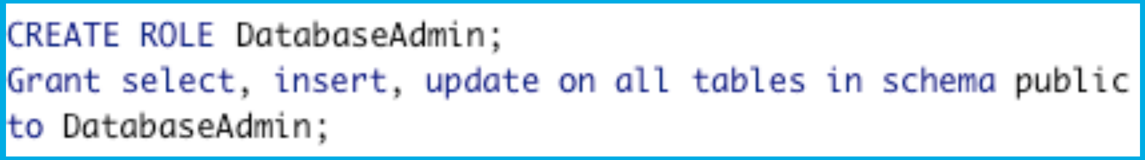
	songname text	title text
1	Eraser	Have a Great Day!



```
create or replace function Get_Song_by_Playlist(INT, REFCURSOR) returns refcursor as
$$
declare
        SONG_ID INT                     := $1;
        resultset    REFCURSOR          := $2;
begin
  open resultset for
        select s.SongName, p.Title
        from Song s, Playlist p
        where SONG_ID = cast(s.SID as int)
        and s.PID = p.PID;
        return resultset;
end;
$$
language plpgsql;

select Get_Song_by_Playlist(1, 'results');
Fetch all from results;
```

SECURITY

Database Administrator: Access in case of overwriting or fixing any data in the entire database

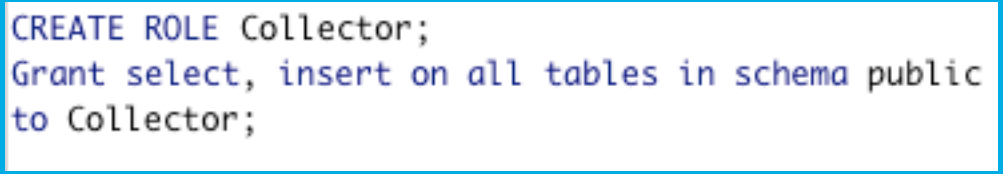


```
CREATE ROLE DatabaseAdmin;
Grant select, insert, update on all tables in schema public
to DatabaseAdmin;
```

Organization: New and old organizations should have access to update or delete their group from the database in collecting music

```
CREATE ROLE Organization;
Grant select, insert, update on all tables in schema public
to Organization;
```

Collector: Should have access to adding new CDs and music to the database and their playlists



```
CREATE ROLE Collector;
Grant select, insert on all tables in schema public
to Collector;
```

IMPLEMENTATION NOTES

This database was implemented on PostgresSql Version 9.3. At first, the database seemed simple, but after inputting the data, I began to find more problems with corresponding foreign keys and inputting the correct data. Implementation became the longest part, while the Views and queries came easier as I was more familiar with my database. The most tedious part was inputting each song from the 10 CDs listed in the tables. Toward the end of inputting the data, I deleted the separation of "Music Possessions" into Records or CDs due to repetitive field names and trouble with the Songs table on Postgres.

Although the database may seem simple at first glance, constructing it and making it perfect was complex. In this database, I am assuming that each collector is buying a CD from a store instead of online. It may be best for an updated version of this database to have the option to buy a song online on iTunes, Spotify, Pandora, or Apple Music. Many future enhancements were considered toward the end of the project and would lead to a more efficient and complex Music Collection database.

KNOWN PROBLEMS WITH DATABASE

In implementing the database, I noticed a few key features that should be fixed and replaced in the future by the Database Administrators:

- Make the CDs have no Artist DOB because bands don’t have a Date of Birth, they have multiple members with multiple different birthdays
- Create a connection from the Collector to the Collector Store by adding a Street Address to the collectors fields in order to chose which CSID that Collector would buy CDs.
- Song duration should be added as a field on the Song table
- Add a date to when a Playlist was created
- Add different types for different genres of music
- Input songs from different CDs to different playlists owned by different people
- Create more tables in connecting which Collectors share music with each other (More complex than this database)