SPOTIFY DATABASE SYSTEM

*a music, podcast, and video streaming digital service*

ISM 6215 Business Database Systems I

Section# 071A

Fall 2017

Abby Bukovich

Hayley Milam

Anirudh Bhagat

Himanshu Vaishnav

Prathibha Padmanabhan

Tushar Mallya

**About Spotify**

Spotify is a music streaming service that offers standard features available for free with a standard account, and additional special features through their monthly subscription service, Spotify Premium. Standard users may listen to playlists, create stations, follow their friends, and receive customized music recommendations based on their activity. Spotify Premium allows users to enjoy their music ad-free. In addition, the subscription service allows users to download music so that it is available offline.

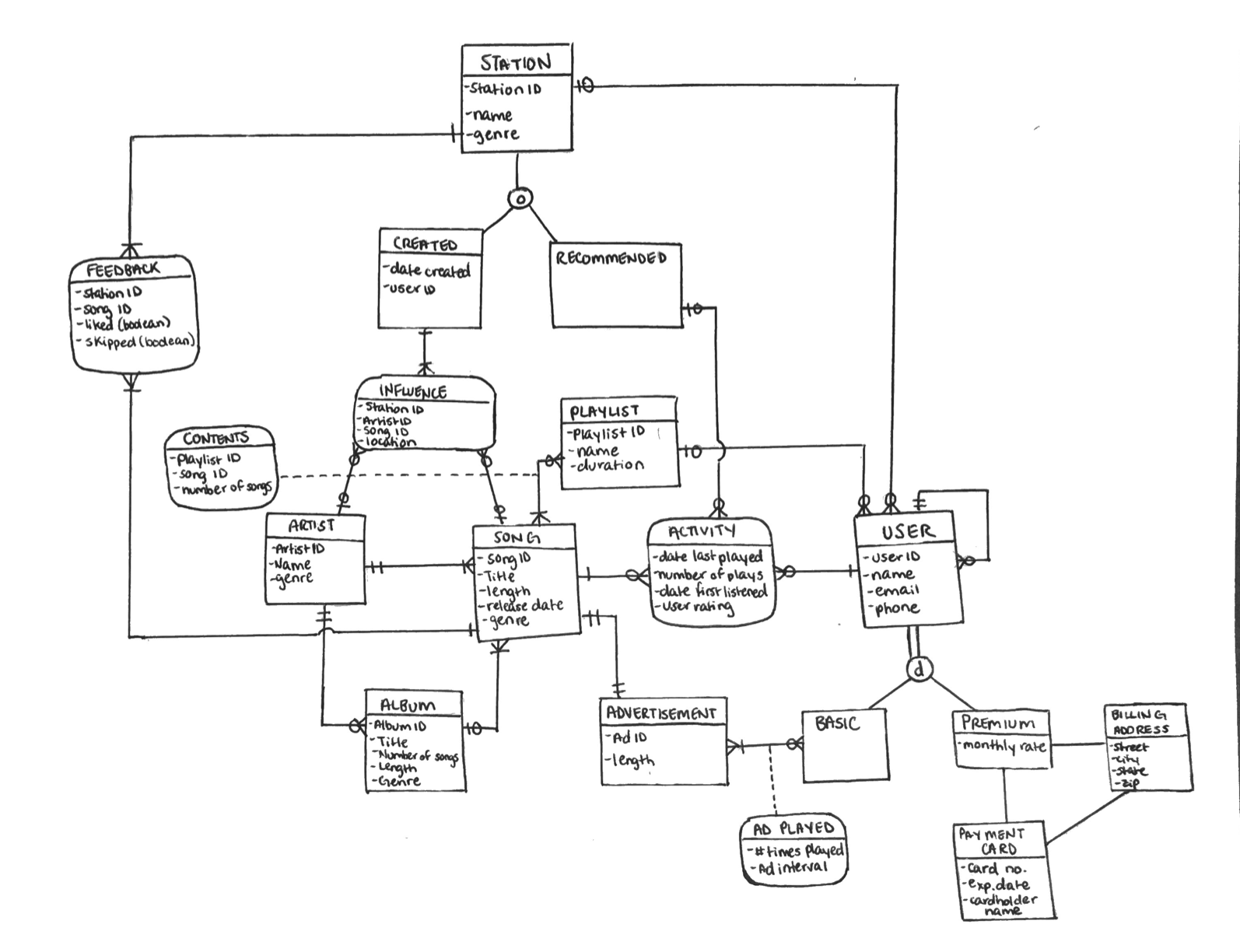
In order to generate customized recommendations for each user, Spotify records data about the songs users listen to, such as how recently and frequently they listen to them. This data is used to recommend stations that feature music with genres that the user might be interested in, based on their activity.

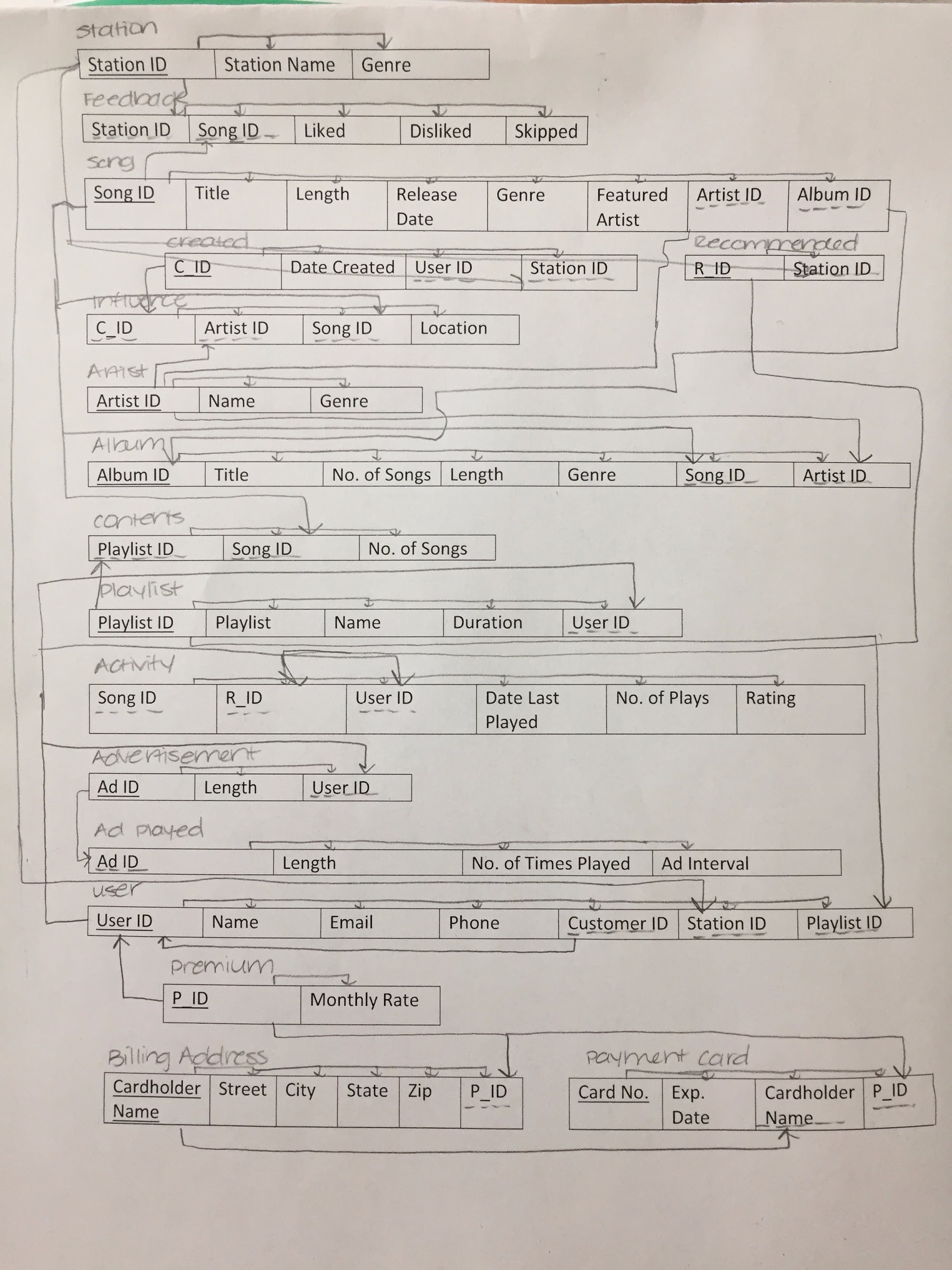
Users can create their own stations by specifying a song or artist and the station will play music within that genre. Users can use more than one artist, song, or a combination of the two to narrow the scope of what the station will play.

**Assumptions**

Because Spotify has such a large and complex database, we made the following assumptions when creating our ER Diagram to fit the scope of this class.

1. Songs don't have to be in an album because they can be released as a single. (optional one-to-many relationship between album and song)
2. The artist of a song and/or album is considered the main promoting artist, and does not include featured artists, if there are any.
3. An artist must have at least 1 song, but does not have to have any albums because they could have only released singles.
4. User can listen to one playlist/station at a time, but many users can listen to that playlist/station simultaneously.
5. Each station receives instances of feedback for songs played, which is used to determine the next song to play.
6. Each station that is created must have at least one Influence. Each Influence is comprised of either one artist or one song.





**Physical Database Design**

1. To determine premium users, we need to create a view which includes User\_Id, Name, Email, Phone no. Monthly rate, and Card no. This view contains more than one table and increases the speed to find premium users.
2. Create a view related to user’s favorite songs which includes, User\_Id, Song\_Id, Song title, Date last played, Genre and Number of plays. This enable us to determine the genre a user listens to.

**Create Table:**

1. CREATE TABLE USER

( USER\_ID INTEGER NOT NULL,

FNAME VARCHAR(35) NOT NULL,

LNAME VARCHAR(35) NOT NULL,

EMAIL VARCHAR(35) NOT NULL,

PHONE\_NUMBER INTEGER

PLAYLIST\_ID INTEGER

CONSTRAINTS USER\_PK PRIMARY KEY (USER\_ID),

CONSTRAINTS USER\_FK FOREIGN KEY (PLAYLIST\_ID) REFERENCES PLAYLIST(PLAYLIST\_ID));

2) CREATE TABLE SONG

( SONG\_ID INTEGER NOT NULL,

TITLE VARCHAR(35) NOT NULL,

LENGTH INTEGER

RELEASE\_DATE INTEGER

GENRE VARCHAR(35)

ARTIST\_ID INTEGER

PLAYLIST\_ID INTEGER

CONSTRAINT SONG\_PK PRIMARY KEY (SONG\_ID),

CONSTRAINT SONG\_FK FOREIGN KEY (PLAYLIST\_ID) REFERENCES PLAYLIST(PLAYLIST\_ID),

CONSTRAINT SONG\_FK FOREIGN KEY (ARTIST\_ID) REFERENCES ARTIST(ARTIST\_ID));

3) CREATE TABLE PLAYLIST

( PLAYLIST\_ID INTEGER NOT NULL,

NAME VARCHAR(35)

NUMBER\_OF\_SONGS INTEGER

DURATION INTEGER

CONSTRAINT PLAYLIST\_PK PRIMARY KEY (PLAYLIST\_ID));

**SQL Queries**

Selecting all records from a table and displaying all columns:

SELECT \* FROM USER;

Selecting all records from a table and displaying a specified column:

SELECT TITLE

FROM PLAYLIST;

Displaying data for a given condition:

SELECT SONG\_ID, TITLE

FROM SONG

WHERE GENRE = 'Pop';

Displaying data for a given condition and sorting the output:

SELECT USER\_ID, NAME

FROM USER

WHERE LNAME = 'Smith'

ORDER BY USER\_ID;

Populate the table with a few rows:

INSERT INTO USER VALUES (2, 'Noah', 'Jobs', 'n.jobs@gmail.com', 3521112345, 20);

INSERT INTO USER VALUES (3, 'Aaron', 'Matts', 'a.matts@hotmail.com', 3521934745, 30);

Correlated subquery:

SELECT p.Name FROM PLAYLIST p

WHERE p.PLAYLIST\_ID = (SELECT u.PLAYLIST\_ID

FROM USER u

WHERE u.PLAYLIST\_ID = p.PLAYLIST\_ID);

Selects all records that match either left or right table records:

SELECT s.TITLE

FROM SONG AS s JOIN ACTIVITY AS a

ON s.SONG\_ID = a.SONG\_ID

WHERE (NUMBER\_OF\_PLAYS) >= 50

LIMIT 10;