

Spotify Personal Music Database Analysis

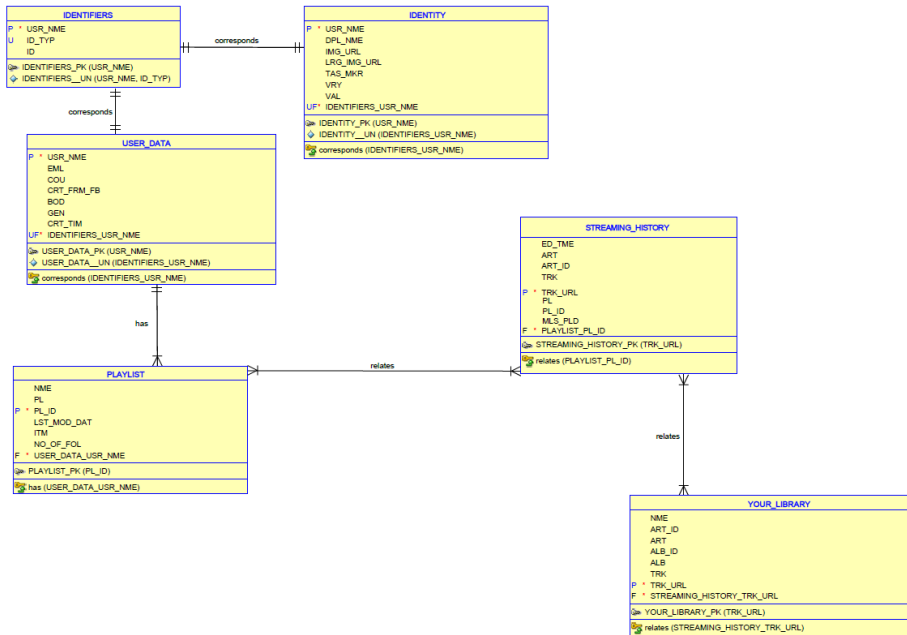
SQL Project

Madeline Lin

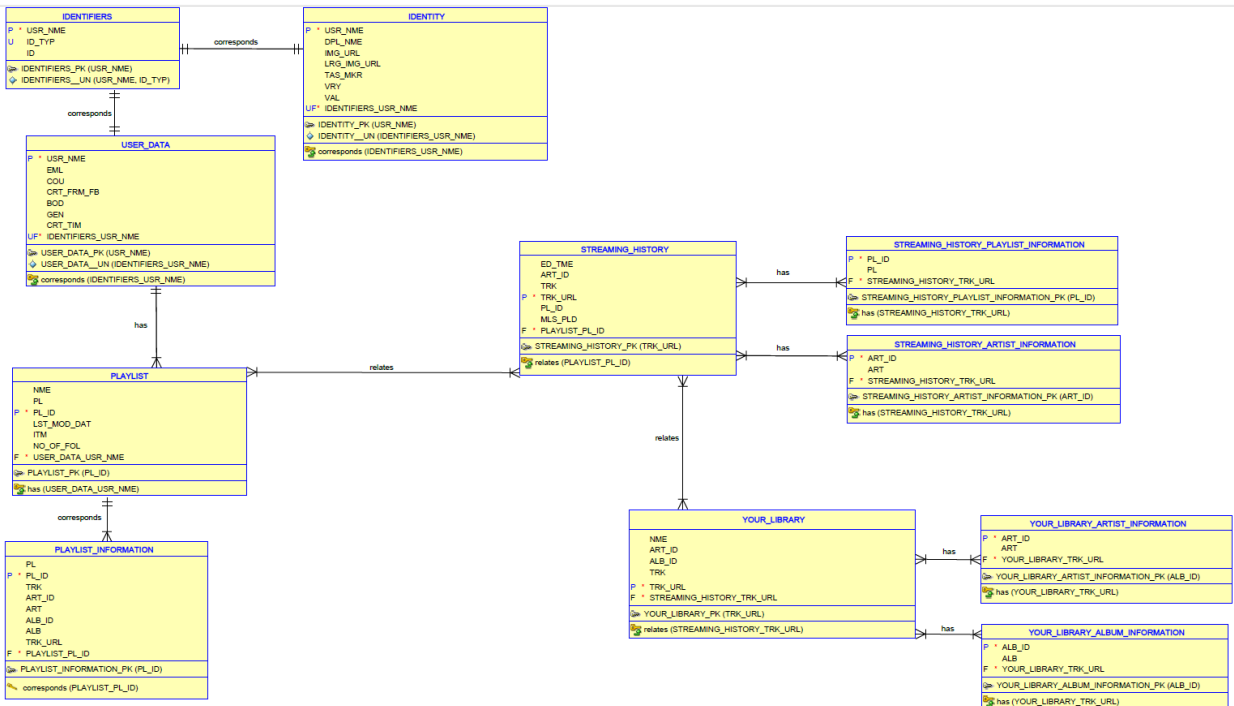
August 3rd, 2022

a) ERD Diagram

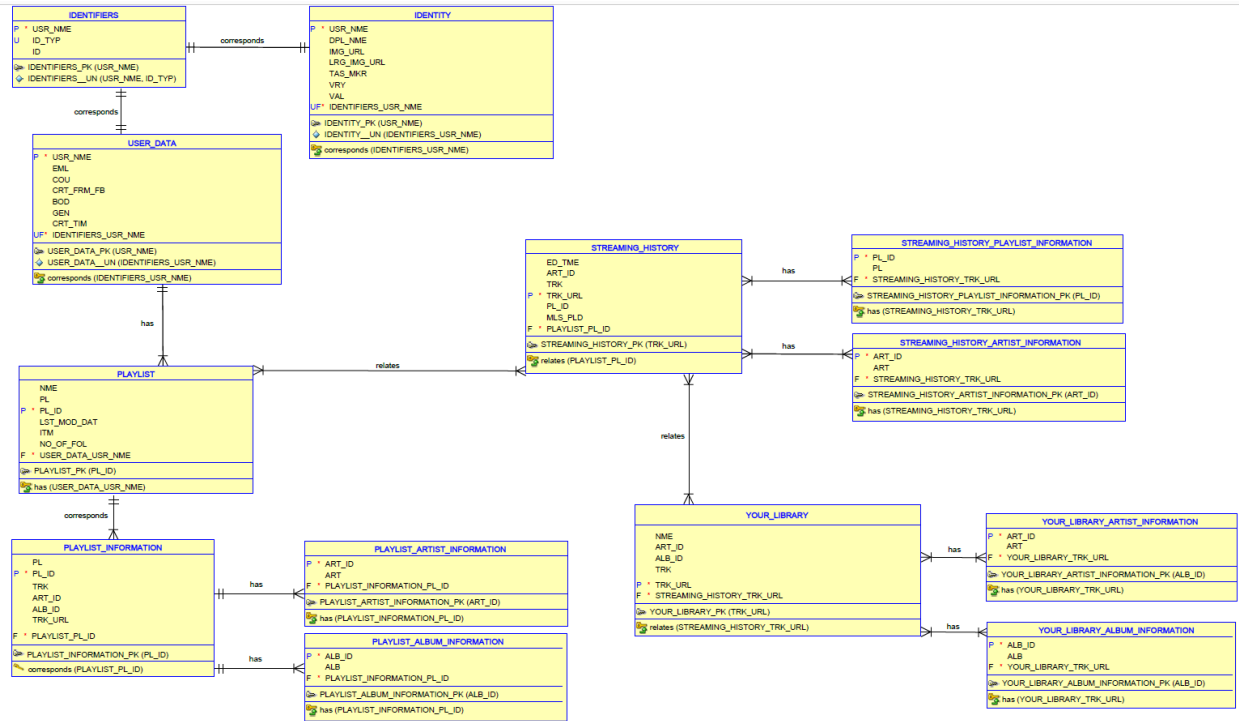
1NF



2NF



3NF



b) Data Model Explanation of Progression From 1NF to 2NF to 3NF

1NF:

Our goal is to 1) Identify the primary key (PK); 2) Make sure that there are no sub-columns, that is, each column must contain atomic values; 3) Eliminate the repeating columns.

Basically, I pulled 6 sheets which are related to my analysis. They are: Identifiers, Identity, User Data, Playlist, Streaming History and Your Library.

I reviewed each sheet and do not think they have any repeating groups.

I identified the PK for each table.

In the IDENTIFIERS Table, the PK should be `USR_NME`. Once a person becomes a user of Spotify, he or she will have the information of other attributes including ID and ID type. Therefore, I think the user name should be the PK.

In the IDENTITY Table, the PK should be `USR_NME` as well.

In the `USER_DATA` Table, the PK should also be `USR_NME`.

In the `PLAYLIST` Table, the PK should be the `PL_ID` as the playlist ID uniquely specifies the playlist and other attributes depend on playlist ID.

In the `STREAMING HISTORY` Table, the PK should be the `TRK_URL`. The main element of this table is the track. The Track URL uniquely specifies the track and other attributes depend on the Track URL.

In the `YOUR_LIBRARY` Table, the PK should be the `TRK_URL` as well. Reason is the same as it for the `STREAMING HISTORY` Table.

2NF:

Our goal is to eliminate partial dependencies. This means that each field that is not the primary key must be dependent on the primary key.

In order to eliminate the partial dependencies, in terms of the `PLAYLIST` Table, the playlist usually embeds a bunch of tracks, and each track corresponds to an artist, belongs to an album, etc. Therefore, I created the sub table of Playlist, which is `PLAYLIST INFORMATION`. In this table, it includes the information of playlist, artist, album and track.

The same for the `STREAMING HISTORY` Table. I created two sub tables named `STREAMING HISTORY PLAYLIST INFORMATION` and `STREAMING HISTORY ARTIST INFORMATION` respectively. With respect to `STREAMING HISTORY PLAYLIST INFORMATION` Table, I determined `PL_ID` as its PK. With respect to `STREAMING HISTORY ARTIST INFORMATION` Table, I determined `ART_ID` as its PK.

Same method for `YOUR_LIBRARY` Table. I created two sub tables named `YOUR_LIBRARY ARTIST INFORMATION` and `YOUR_LIBRARY ALBUM INFORMATION` respectively. With regard to `YOUR_LIBRARY ARTIST INFORMATION` Table, I determined `ART ID` as its PK. With regard to `YOUR_LIBRARY ALBUM INFORMATION` Table, I determined `ALB_ID` as its PK.

3NF:

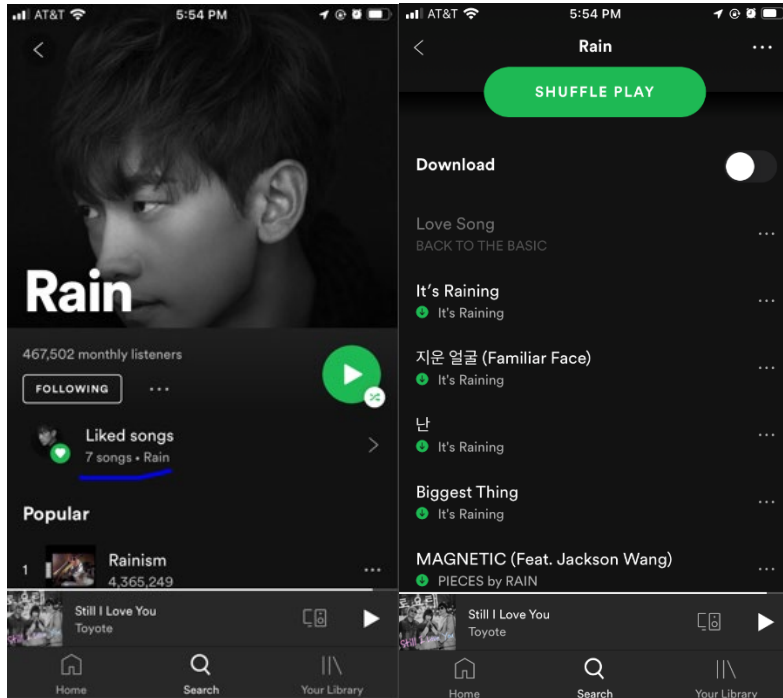
Our goal is to eliminate transitive dependencies. Transitive dependencies mean that an attribute is dependent on another attribute that is not part of the primary key.

In my opinion, it looks pretty good till 2NF. However, in order to eliminate the transitive dependencies, in terms of the PLAYLIST INFORMATION Table, I created two sub tables, i.e. PLAYLIST ARTIST INFORMATION Table and PLAYLIST ALBUM INFORMATION Table.

c) SQLs

1. Extract the information of the number of songs I have for an artist I love

i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

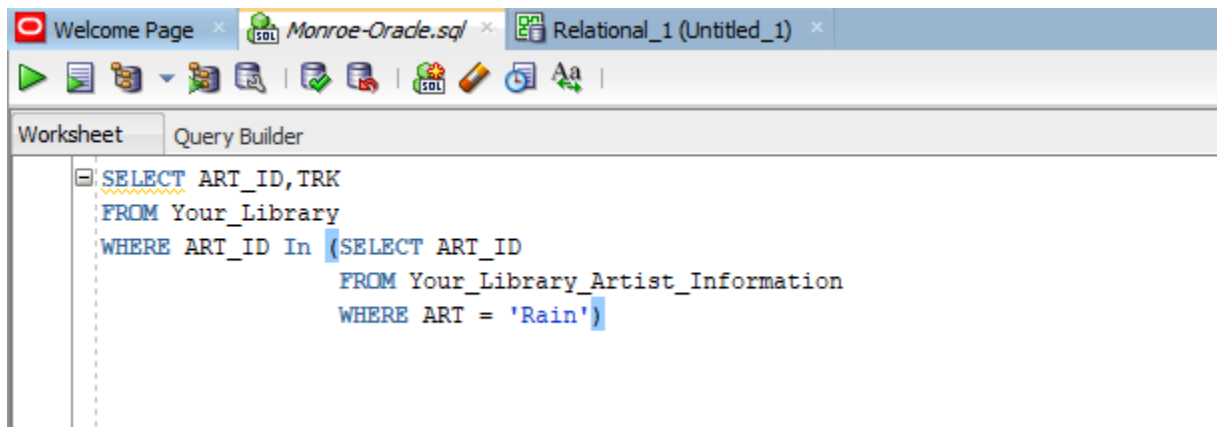
I love the artist “Rain”, who is a Korean singer, dancer and actor. I would like to see how many tracks from Rain I have in my library.

In Your Library, I selected ART_ID and TRK. As the artist information (i.e. ART) is in Your Library Artist Information, I used Subqueries. In the Your Library Artist Information, I selected ART_ID where the ART is Rain.

SQL gives me the result of a table which contains the information of the Rain’s tracks that I have in my library.

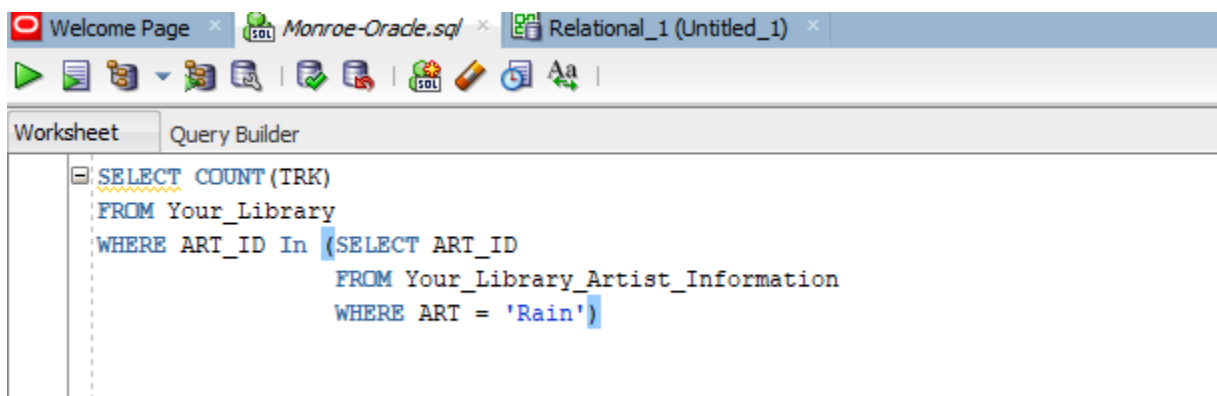
After that, I used COUNT function to count the number of the tracks, which is 8 based on the returned result.

iii) SQL to Reproduce the Data



The screenshot shows the SQL Developer interface with the 'Query Builder' tab selected. The query editor contains the following SQL statement:

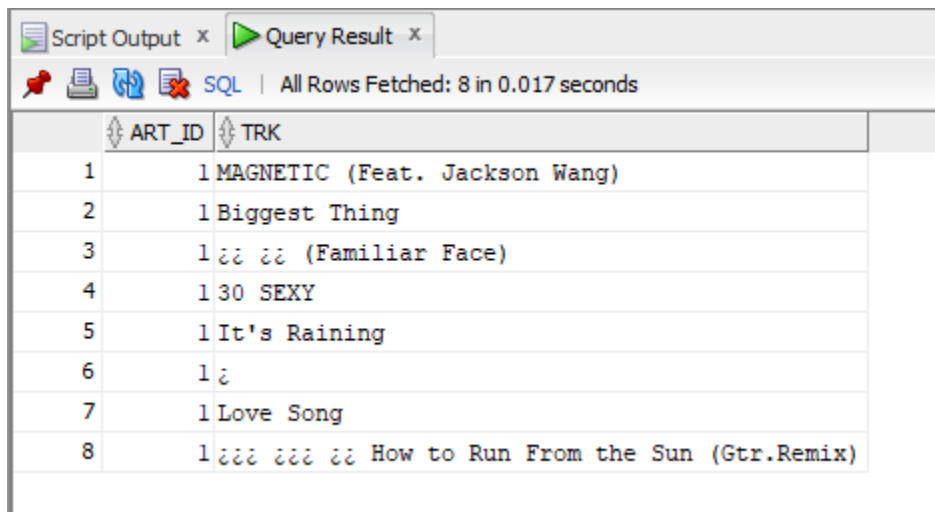
```
SELECT ART_ID, TRK
FROM Your_Library
WHERE ART_ID In (SELECT ART_ID
                  FROM Your_Library_Artist_Information
                  WHERE ART = 'Rain')
```



The screenshot shows the SQL Developer interface with the 'Query Builder' tab selected. The query editor contains the following SQL statement:





```
SELECT COUNT(TRK)
FROM Your_Library
WHERE ART_ID In (SELECT ART_ID
                  FROM Your_Library_Artist_Information
                  WHERE ART = 'Rain')
```

iv) Results of the SQL



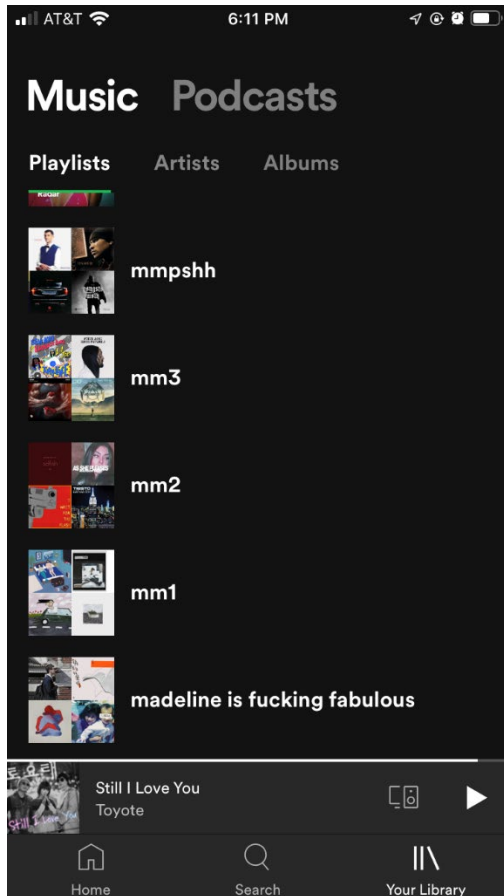
The screenshot shows the SQL Developer interface with the 'Query Result' tab selected. The results of the SQL query are displayed in a table with 8 rows. The table has two columns: ART_ID and TRK.

ART_ID	TRK
1	1 MAGNETIC (Feat. Jackson Wang)
2	1 Biggest Thing
3	1 22 22 (Familiar Face)
4	1 30 SEXY
5	1 It's Raining
6	1 2
7	1 Love Song
8	1 222 222 22 How to Run From the Sun (Gtr.Remix)

Script Output x		Query Result x	
			 SQL All Rows Fetched: 1 in 0.01
		COUNT(TRK)	
1		8	

2. My Favorite Playlist

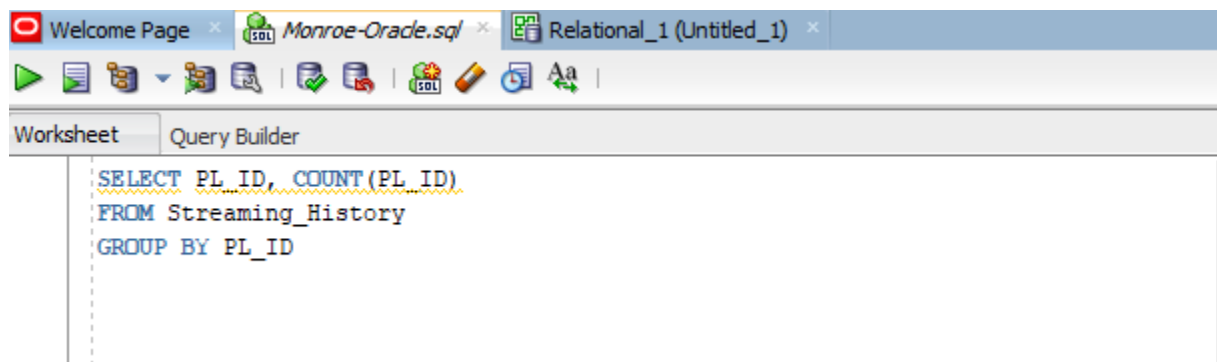
i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

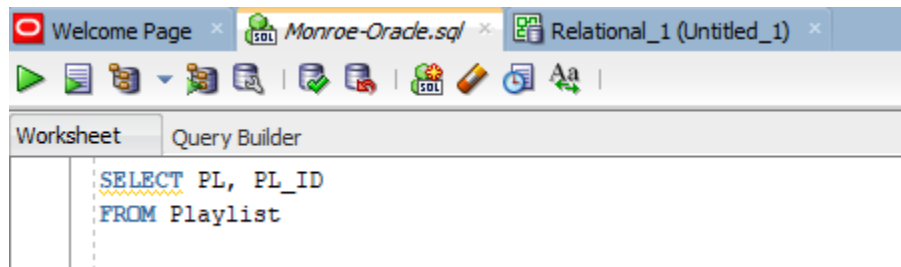
Based on the Streaming History, I selected PL_ID and used Count function to count the number of each PL_ID, grouped by PL_ID. Therefore, I am able to extract the information of how many songs from the playlist that I have streamed. Because in the Streaming History, there is no playlist name included, I selected playlist and playlist ID from the Playlist so that I will be able to know which playlist ID corresponds to which playlist.

iii) SQL to Reproduce the Data



The screenshot shows the SQL Developer interface with the 'Query Builder' tab selected. The query editor contains the following SQL statement:

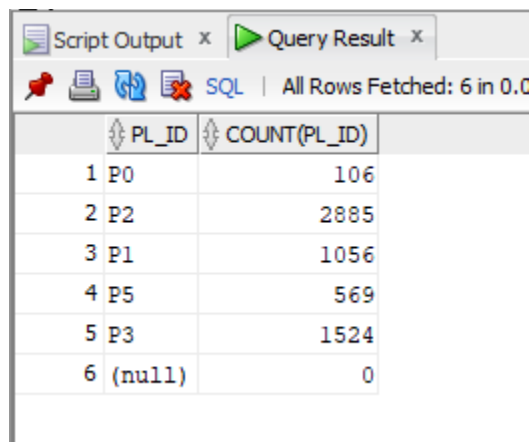
```
SELECT PL_ID, COUNT(PL_ID)
FROM Streaming_History
GROUP BY PL_ID
```



The screenshot shows the SQL Developer interface with the 'Query Builder' tab selected. The query editor contains the following SQL statement:

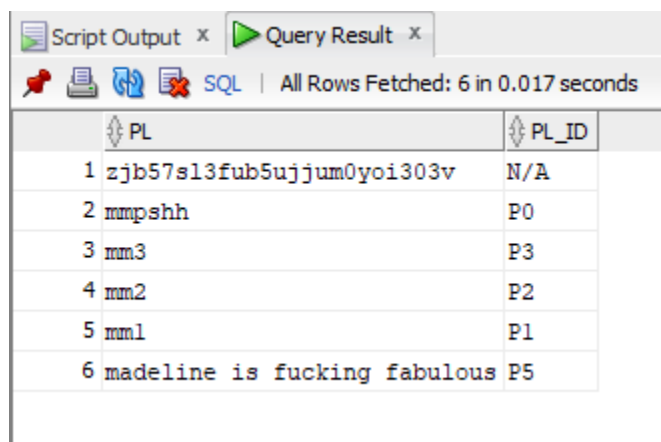
```
SELECT PL, PL_ID
FROM Playlist
```

iv) Results of the SQL



The screenshot shows the 'Query Result' tab in SQL Developer. The status bar indicates 'All Rows Fetched: 6 in 0.0...'. The results are displayed in a table with two columns: PL_ID and COUNT(PL_ID).

PL_ID	COUNT(PL_ID)
1 P0	106
2 P2	2885
3 P1	1056
4 P5	569
5 P3	1524
6 (null)	0

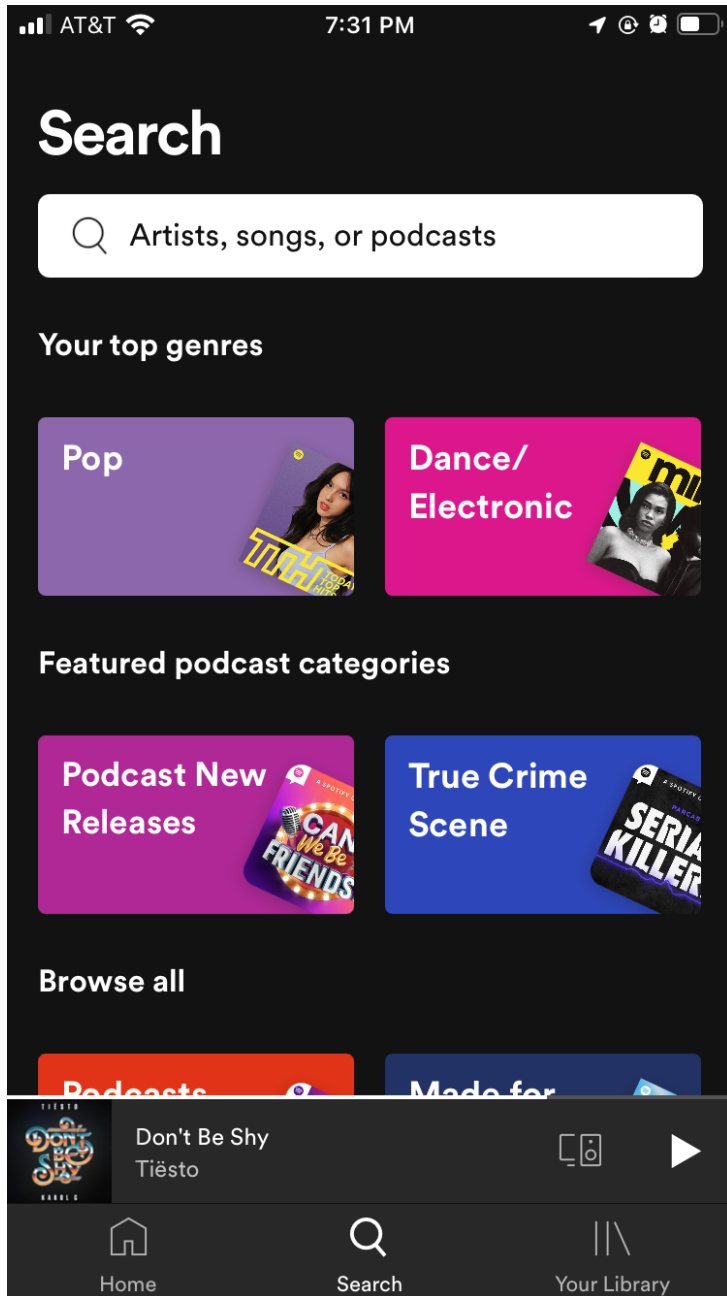


The screenshot shows the 'Query Result' tab in SQL Developer. The status bar indicates 'All Rows Fetched: 6 in 0.017 seconds'. The results are displayed in a table with two columns: PL and PL_ID.

PL	PL_ID
1 zjb57s13fub5ujjum0yoi303v	N/A
2 mmpshh	P0
3 mm3	P3
4 mm2	P2
5 mm1	P1
6 madeline is fucking fabulous	P5

3. My Favorite type of Genre

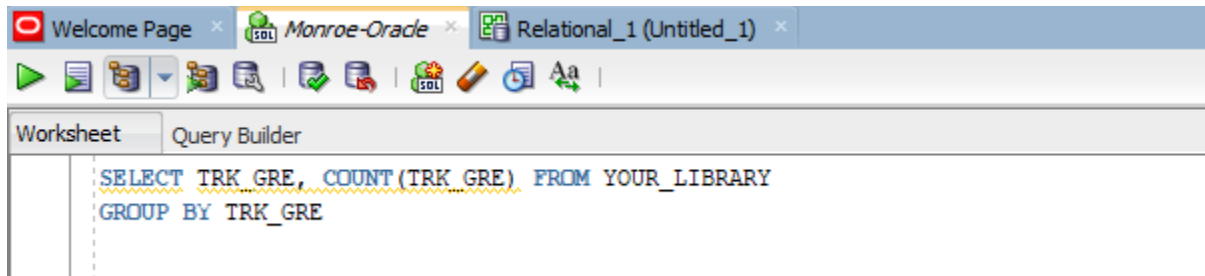
i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

In Your Library, I selected the TRK_GRE and used Count function to count the number of TRK_GRE, grouped by TRK_GRE. It will give me the result of different genres of music that I listen to as well as the number of songs of each genre.

iii) SQL to Reproduce the Data



iv) Results of the SQL

The screenshot shows a SQL query result window with the following tabs: 'Script Output', 'Query Result', and 'Query Result 1'. The 'Query Result' tab is active, displaying the results of the query. The status bar indicates 'All Rows Fetched: 9 in 0.014 seconds'.

TRK_GRE	COUNT(TRK_GRE)
1 K-Indie	22
2 J-pop	1
3 R&B/Soul	13
4 Undefined	5
5 K-pop	64
6 M-pop	49
7 Electronic/Dance	58
8 Pop	227
9 Hip Hop/Rap	1

4. My OnRepeat Track

i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

I would like to see my most repeated track based on the streaming history table. I would like to extract the one with the max streaming times.

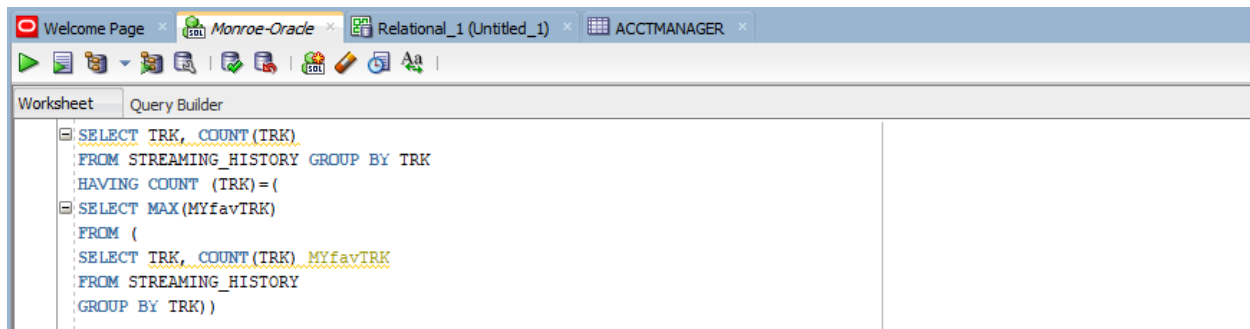
This SQL contains three subparts. First, from the Streaming History, I selected TRK and used Count function to count TRK (the number of times a track is played). I determined the new variable to be MYfavTRK (i.e. my favorite track). I also used Group by TRK. This part will return all tracks with the corresponding number of times played.

Based on the above part, I used the MAX function to determine the TRK which has the max number of times played and named it MYfavTRK.

Note that Having clause serves as the Where clause for grouped data.

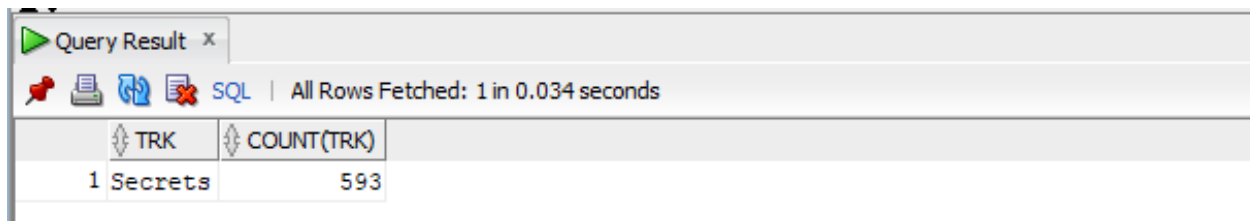
Finally, I used Select statement and Count function as shown the beginning part to extract the my onrepeat track name with its count.

iii) SQL to Reproduce the Data



```
SELECT TRK, COUNT(TRK)
FROM STREAMING_HISTORY GROUP BY TRK
HAVING COUNT (TRK)= (
SELECT MAX(MYfavTRK)
FROM (
SELECT TRK, COUNT(TRK) MYfavTRK
FROM STREAMING_HISTORY
GROUP BY TRK) )
```

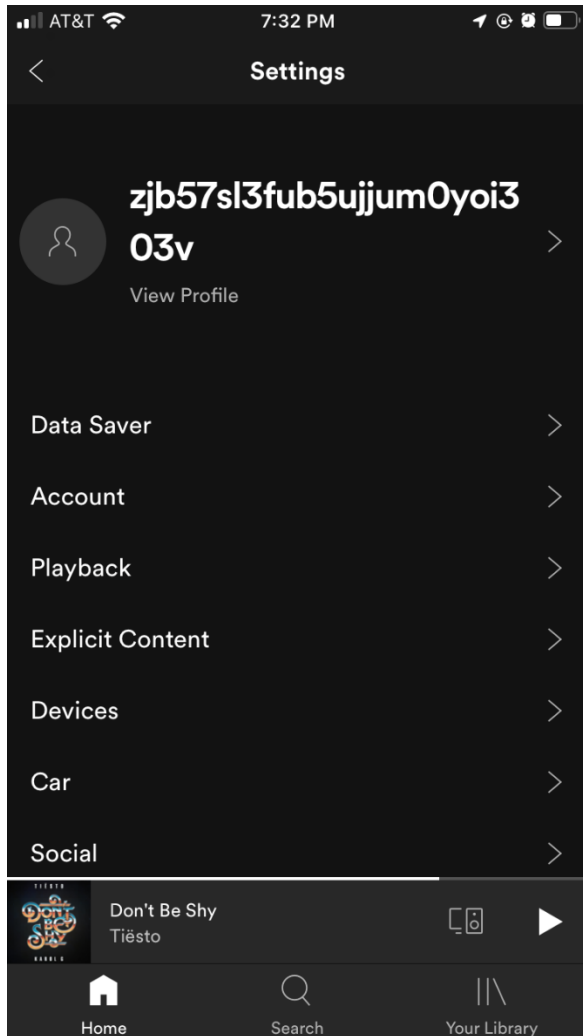
iv) Results of the SQL



TRK	COUNT(TRK)
1 Secrets	593

5. Validate my username

i) Snapshot of the Screen That the SQL is Providing Data For



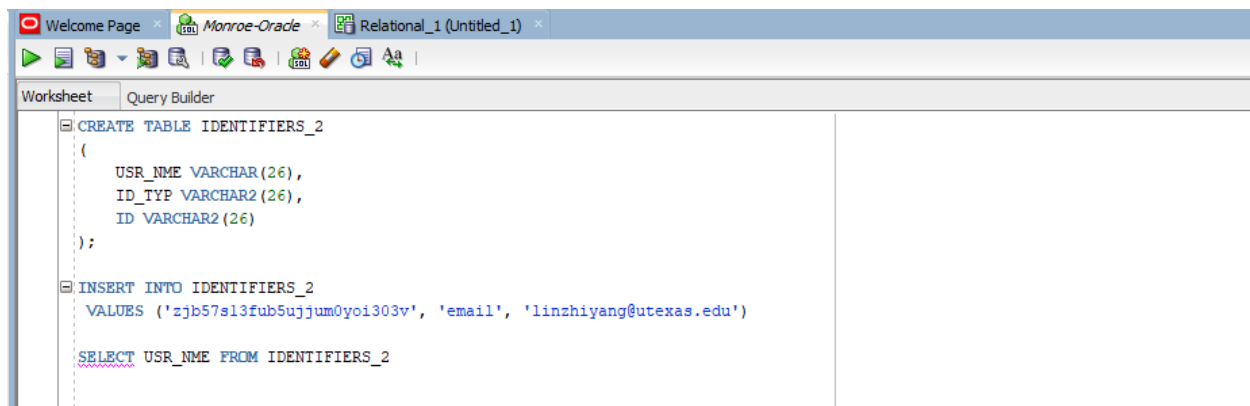
ii) Explanation of How SQL Works

For this case, based on the professor's feedbacks about the project guideline, I created tables, constraints, populated tables, and provided metadata for the database.

I selected a spreadsheet with less information and try to create the table manually. As indicated below, I used CREATE TABLE command and named the IDENTIFIERS_2 as the original IDENTIFIERS Table has been imported. Then, I filled the attribute name and constraint information. I also used INSERT statements to fill the information.

It gives me the result which can be validated from the i) screenshot. My user name is zjb57sl3fub5ujjum0yoi303v.

iii) SQL to Reproduce the Data



The screenshot shows the SQL Developer interface with the 'Query Builder' tab active. The SQL script in the editor consists of three statements: a CREATE TABLE statement for IDENTIFIERS_2, an INSERT INTO statement, and a SELECT statement. The table has columns USR_NME, ID_TYP, and ID. The insert statement adds a row with a long alphanumeric string, 'email', and an email address. The select statement retrieves the USR_NME column.

```
CREATE TABLE IDENTIFIERS_2
(
  USR_NME VARCHAR(26),
  ID_TYP VARCHAR2(26),
  ID VARCHAR2(26)
);

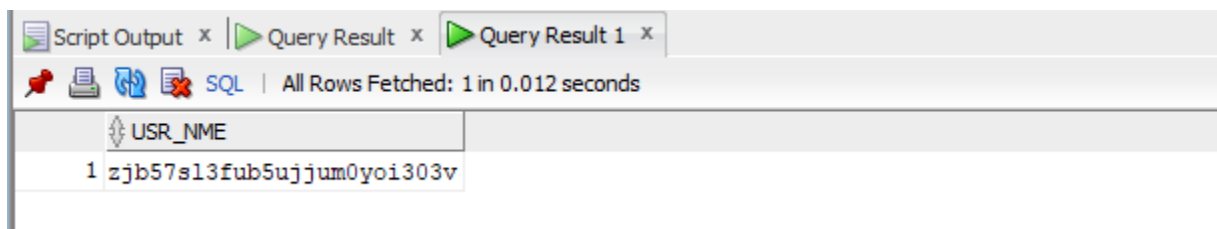
INSERT INTO IDENTIFIERS_2
VALUES ('zjb57s13fub5ujjum0yoi303v', 'email', 'linzhiyang@utexas.edu')

SELECT USR_NME FROM IDENTIFIERS_2
```

iv) Results of the SQL

Table IDENTIFIERS_2 created.

1 row inserted.



The screenshot shows the 'Query Result' tab in SQL Developer. It displays a single row of data from the SELECT statement. The column is labeled 'USR_NME' and the value is '1 zjb57s13fub5ujjum0yoi303v'. The status bar indicates 'All Rows Fetched: 1 in 0.012 seconds'.

USR_NME
1 zjb57s13fub5ujjum0yoi303v