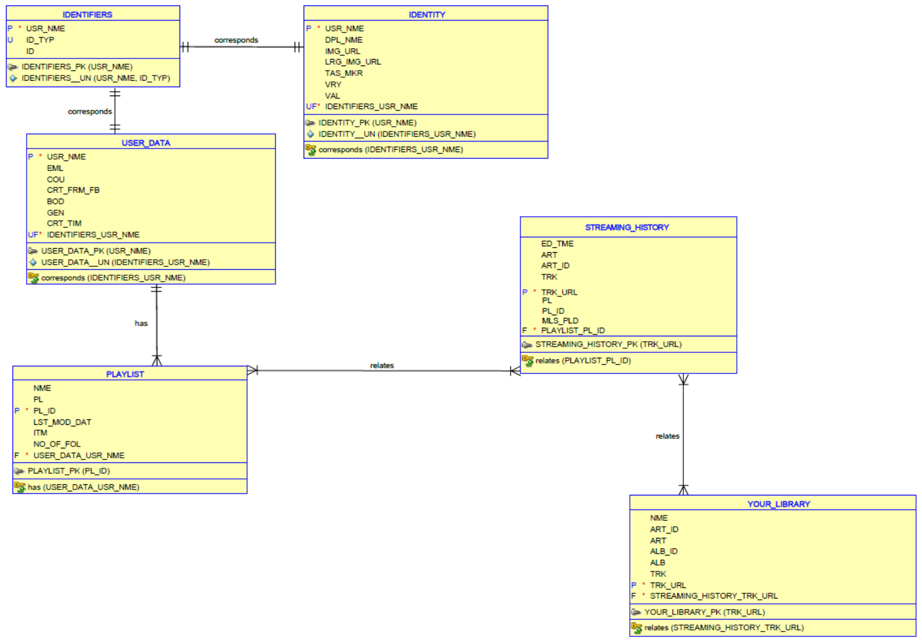
Spotify Personal Music Database Analysis

Madeline Lin

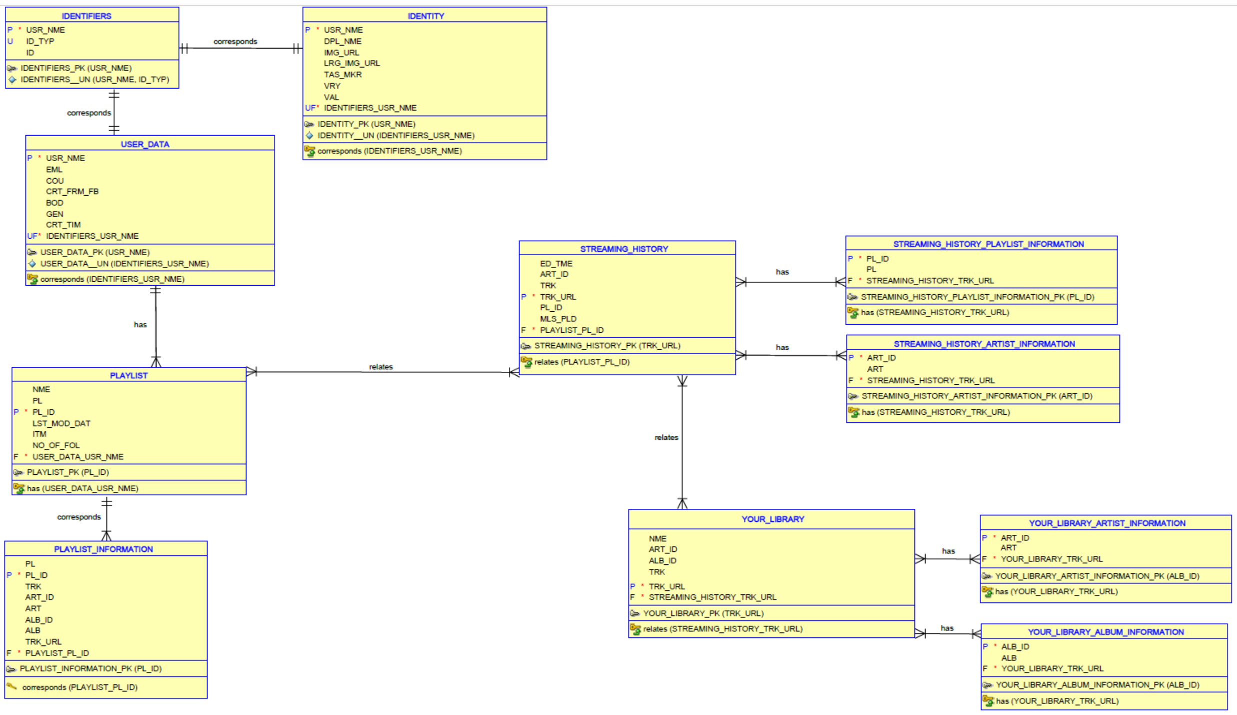
August 3rd, 2022

a) ERD Diagram

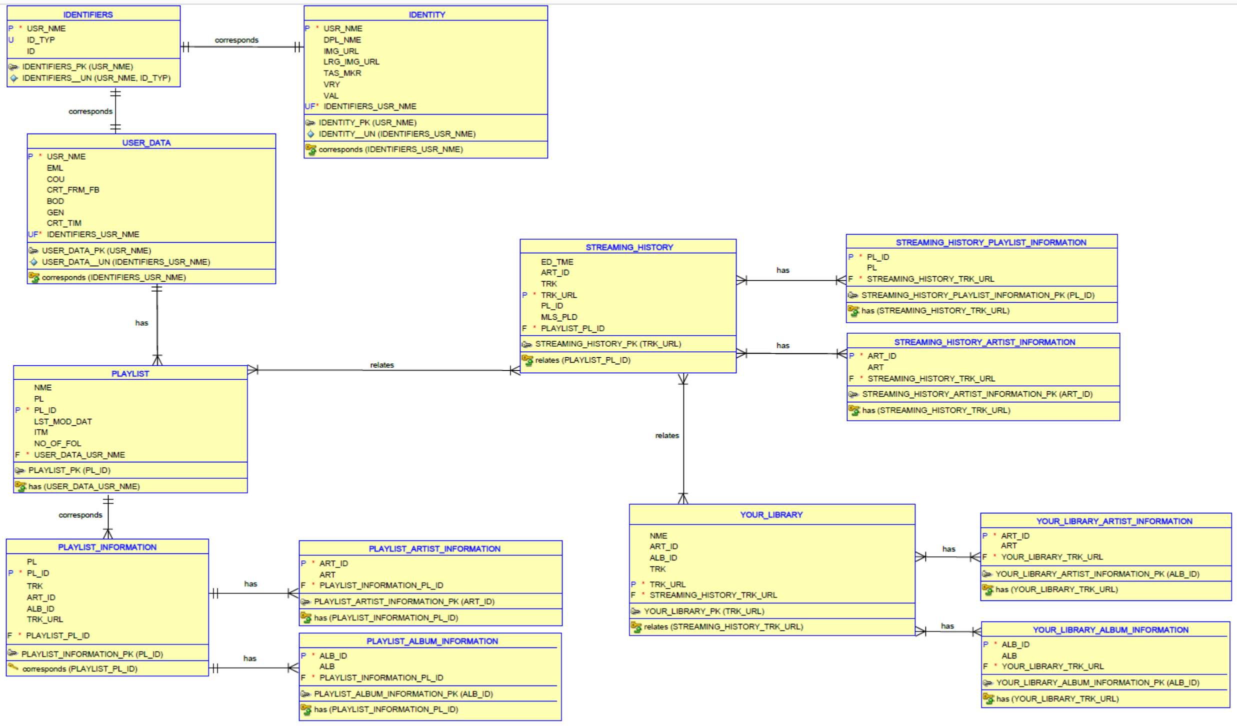
1NF



2NF


IDENTIFIERS
P * USR_NME
U ID_TYP
ID
IDENTIFIERS_PK (USR_NME)
IDENTIFIERS__UN (USR_NME, ID_TYP)
corresponds
IDENTITY
P * USR_NME
DPL_NME
IMG_URL
LRG_IMG_URL
TAS_MKR
VRY
VAL
UF* IDENTIFIERS_USR_NME
IDENTITY_PK (USR_NME)
IDENTITY__UN (IDENTIFIERS_USR_NME)
corresponds (IDENTIFIERS_USR_NME)
corresponds
USER_DATA
P * USR_NME
EML
COU
CRT_FRM_FB
BOD
GEN
CRT_TIM
UF* IDENTIFIERS_USR_NME
USER_DATA_PK (USR_NME)
USER_DATA__UN (IDENTIFIERS_USR_NME)
corresponds (IDENTIFIERS_USR_NME)
has
PLAYLIST
NME
PL
P * PL_ID
LST_MOD_DAT
ITM
NO_OF_FOL
F * USER_DATA_USR_NME
PLAYLIST_PK (PL_ID)
has (USER_DATA_USR_NME)
corresponds
PLAYLIST_INFORMATION
PL
P * PL_ID
TRK
ART_ID
ART
ALB_ID
ALB
TRK_URL
F * PLAYLIST_PL_ID
PLAYLIST_INFORMATION_PK (PL_ID)
corresponds (PLAYLIST_PL_ID)
relates
STREAMING_HISTORY
ED_TME
ART_ID
TRK
P * TRK_URL
PL_ID
MLS_PLD
F * PLAYLIST_PL_ID
STREAMING_HISTORY_PK (TRK_URL)
relates (PLAYLIST_PL_ID)
has
STREAMING_HISTORY_PLAYLIST_INFORMATION
P * PL_ID
PL
F * STREAMING_HISTORY_TRK_URL
STREAMING_HISTORY_PLAYLIST_INFORMATION_PK (PL_ID)
has (STREAMING_HISTORY_TRK_URL)
has
STREAMING_HISTORY_ARTIST_INFORMATION
P * ART_ID
ART
F * STREAMING_HISTORY_TRK_URL
STREAMING_HISTORY_ARTIST_INFORMATION_PK (ART_ID)
has (STREAMING_HISTORY_TRK_URL)
relates
YOUR_LIBRARY
NME
ART_ID
ALB_ID
TRK
P * TRK_URL
F * STREAMING_HISTORY_TRK_URL
YOUR_LIBRARY_PK (TRK_URL)
relates (STREAMING_HISTORY_TRK_URL)
has
YOUR_LIBRARY_ARTIST_INFORMATION
P * ART_ID
ART
F * YOUR_LIBRARY_TRK_URL
YOUR_LIBRARY_ARTIST_INFORMATION_PK (ALB_ID)
has (YOUR_LIBRARY_TRK_URL)
has
YOUR_LIBRARY_ALBUM_INFORMATION
P * ALB_ID
ALB
F * YOUR_LIBRARY_TRK_URL
YOUR_LIBRARY_ALBUM_INFORMATION_PK (ALB_ID)
has (YOUR_LIBRARY_TRK_URL)

3NF


IDENTIFIERS
P * USR_NME
U ID_TYP
ID
IDENTIFIERS_PK (USR_NME)
IDENTIFIERS__UN (USR_NME, ID_TYP)
corresponds
IDENTITY
P * USR_NME
DPL_NME
IMG_URL
LRG_IMG_URL
TAS_MKR
VRY
VAL
UF* IDENTIFIERS_USR_NME
IDENTITY_PK (USR_NME)
IDENTITY__UN (IDENTIFIERS_USR_NME)
corresponds (IDENTIFIERS_USR_NME)
corresponds
USER_DATA
P * USR_NME
EML
COU
CRT_FRM_FB
BOD
GEN
CRT_TIM
UF* IDENTIFIERS_USR_NME
USER_DATA_PK (USR_NME)
USER_DATA__UN (IDENTIFIERS_USR_NME)
corresponds (IDENTIFIERS_USR_NME)
has
PLAYLIST
NME
PL
P * PL_ID
LST_MOD_DAT
ITM
NO_OF_FOL
F * USER_DATA_USR_NME
PLAYLIST_PK (PL_ID)
has (USER_DATA_USR_NME)
relates
STREAMING_HISTORY
ED_TME
ART_ID
TRK
P * TRK_URL
PL_ID
MLS_PLD
F * PLAYLIST_PL_ID
STREAMING_HISTORY_PK (TRK_URL)
relates (PLAYLIST_PL_ID)
has
STREAMING_HISTORY_PLAYLIST_INFORMATION
P * PL_ID
PL
F * STREAMING_HISTORY_TRK_URL
STREAMING_HISTORY_PLAYLIST_INFORMATION_PK (PL_ID)
has (STREAMING_HISTORY_TRK_URL)
has
STREAMING_HISTORY_ARTIST_INFORMATION
P * ART_ID
ART
F * STREAMING_HISTORY_TRK_URL
STREAMING_HISTORY_ARTIST_INFORMATION_PK (ART_ID)
has (STREAMING_HISTORY_TRK_URL)
relates
corresponds
PLAYLIST_INFORMATION
PL
P * PL_ID
TRK
ART_ID
ALB_ID
TRK_URL
F * PLAYLIST_PL_ID
PLAYLIST_INFORMATION_PK (PL_ID)
corresponds (PLAYLIST_PL_ID)
has
PLAYLIST_ARTIST_INFORMATION
P * ART_ID
ART
F * PLAYLIST_INFORMATION_PL_ID
PLAYLIST_ARTIST_INFORMATION_PK (ART_ID)
has (PLAYLIST_INFORMATION_PL_ID)
has
PLAYLIST_ALBUM_INFORMATION
P * ALB_ID
ALB
F * PLAYLIST_INFORMATION_PL_ID
PLAYLIST_ALBUM_INFORMATION_PK (ALB_ID)
has (PLAYLIST_INFORMATION_PL_ID)
YOUR_LIBRARY
NME
ART_ID
ALB_ID
TRK
P * TRK_URL
F * STREAMING_HISTORY_TRK_URL
YOUR_LIBRARY_PK (TRK_URL)
relates (STREAMING_HISTORY_TRK_URL)
has
YOUR_LIBRARY_ARTIST_INFORMATION
P * ART_ID
ART
F * YOUR_LIBRARY_TRK_URL
YOUR_LIBRARY_ARTIST_INFORMATION_PK (ALB_ID)
has (YOUR_LIBRARY_TRK_URL)
has
YOUR_LIBRARY_ALBUM_INFORMATION
P * ALB_ID
ALB
F * YOUR_LIBRARY_TRK_URL
YOUR_LIBRARY_ALBUM_INFORMATION_PK (ALB_ID)
has (YOUR_LIBRARY_TRK_URL)

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 STREMING 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:

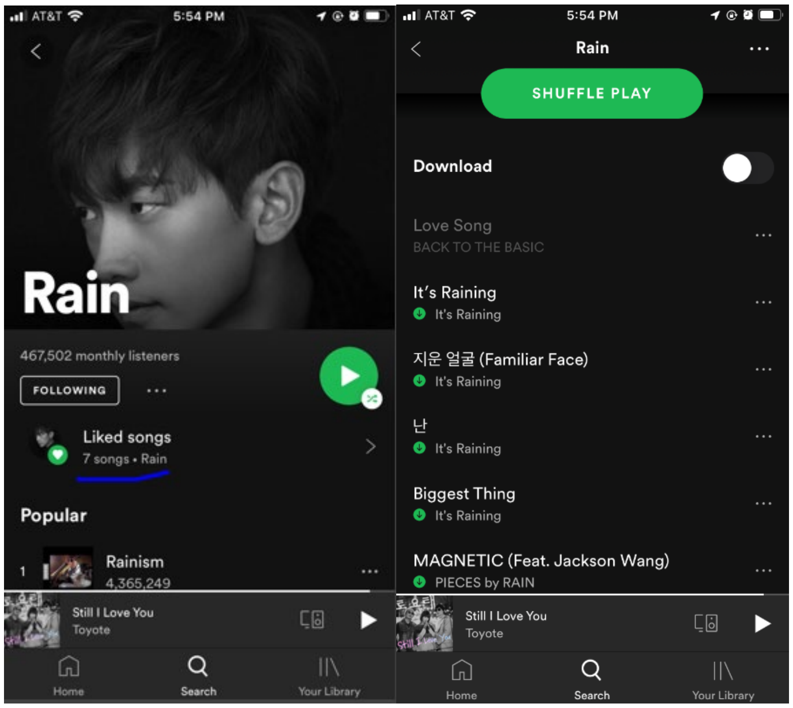
Our goal is to eliminate transitive dependencies. Transitive dependencies mean that an attribute is dependent on another attribute that is not part of the primary key.

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

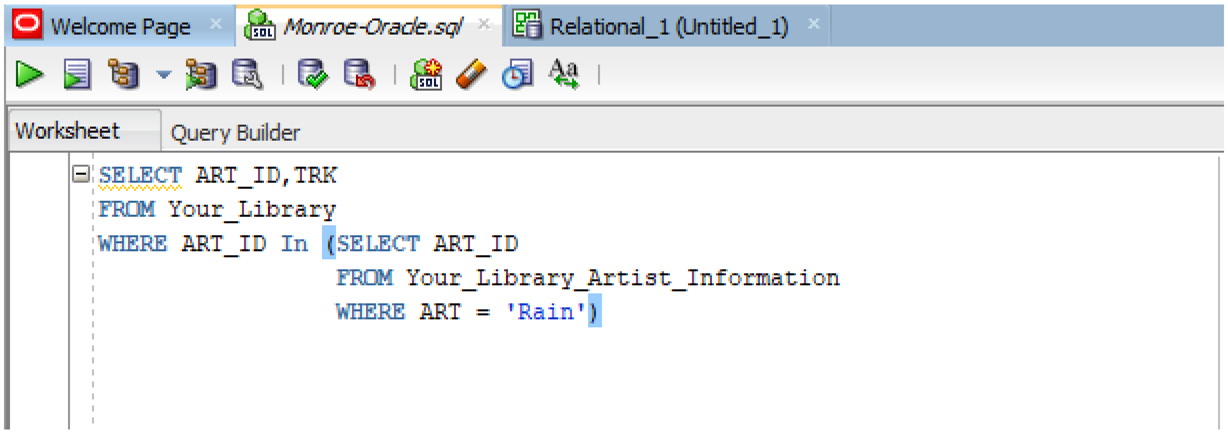
I love the artist "Rain", who is a Korean singer, dancer and actor. I would like to see how many tracks from Rain I have in my library.

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.

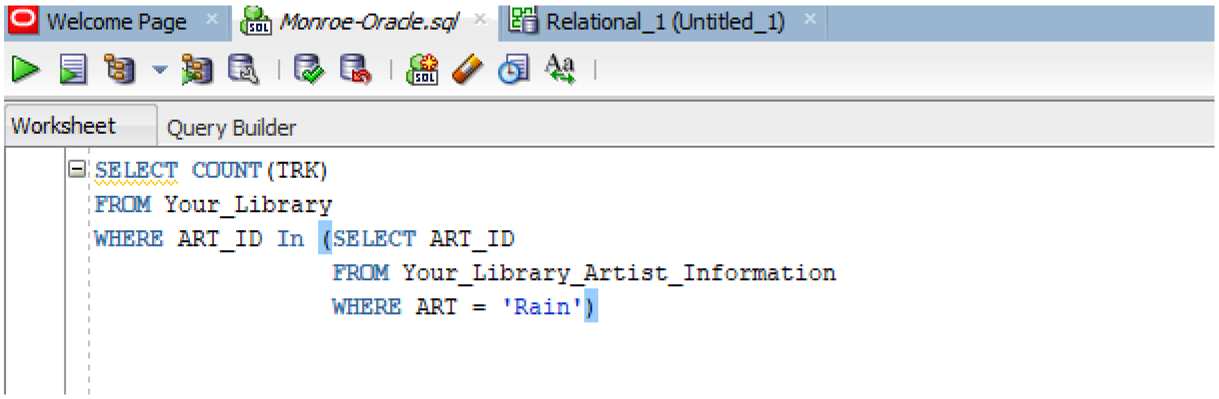
SQL gives me the result of a table which contains the information of the Rain's tracks that I have in my library.

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

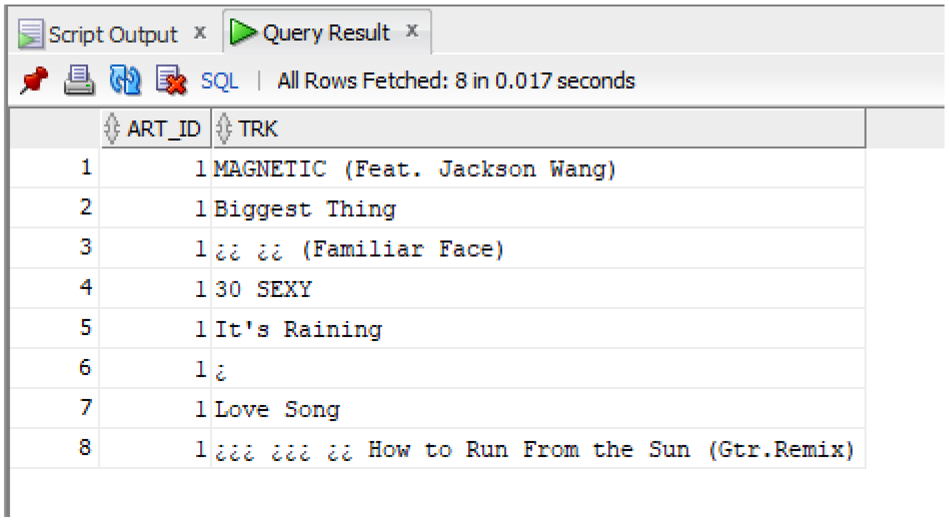


```
SELECT ART_ID,TRK
FROM Your_Library
WHERE ART_ID In (SELECT ART_ID
                 FROM Your_Library_Artist_Information
                 WHERE ART = 'Rain')
```



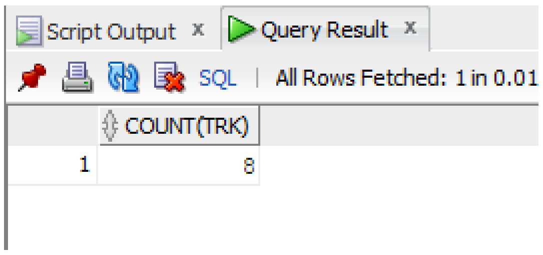
```
SELECT COUNT(TRK)
FROM Your_Library
WHERE ART_ID In (SELECT ART_ID
                 FROM Your_Library_Artist_Information
                 WHERE ART = 'Rain')
```

iv) Results of the SQL



All Rows Fetched: 8 in 0.017 seconds

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

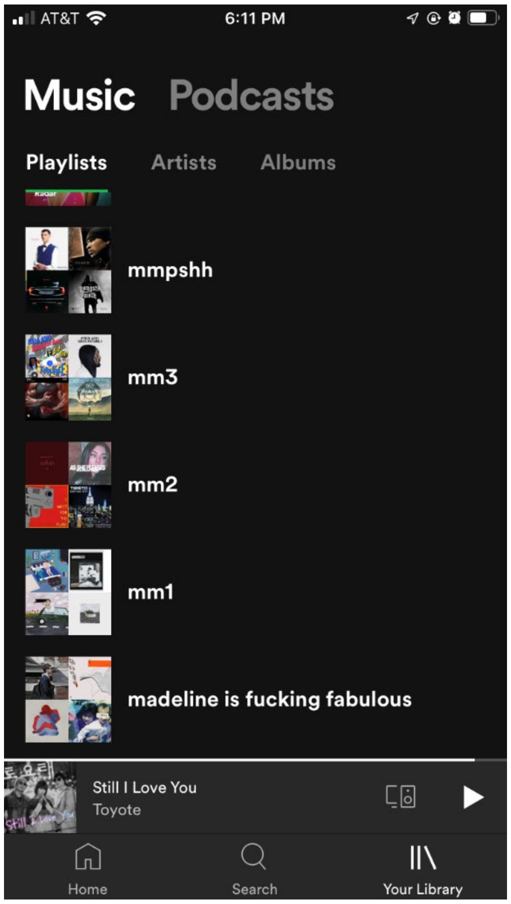


All Rows Fetched: 1 in 0.01

	COUNT(TRK)
1	8

2. My Favorite Playlist

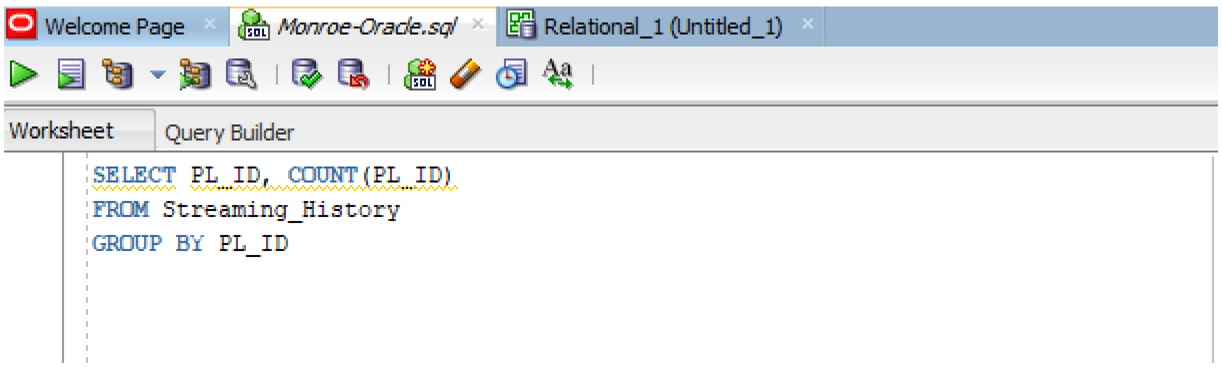
i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

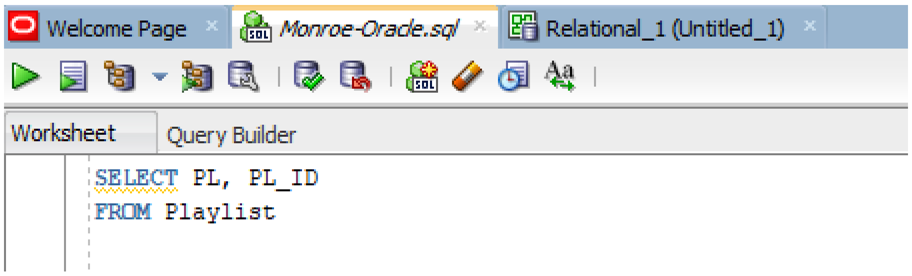
Based on the Streaming History, I selected PL_ID and used Count function to count the number of each PL_ID, grouped by PL_ID. Therefore, I am able to extract the information of how many songs from the playlist that I have streamed. Because in the Streaming History, there is no playlist name included,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


Welcome Page | Monroe-Oracle.sql | Relational_1 (Untitled_1)

Worksheet | Query Builder

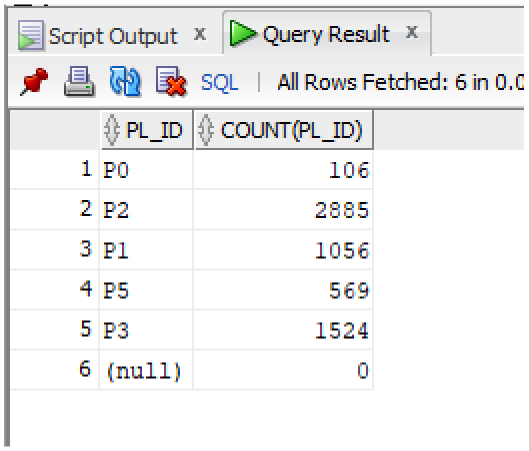
```
SELECT PL_ID, COUNT(PL_ID)
FROM Streaming_History
GROUP BY PL_ID
```


Welcome Page | Monroe-Oracle.sql | Relational_1 (Untitled_1)

Worksheet | Query Builder

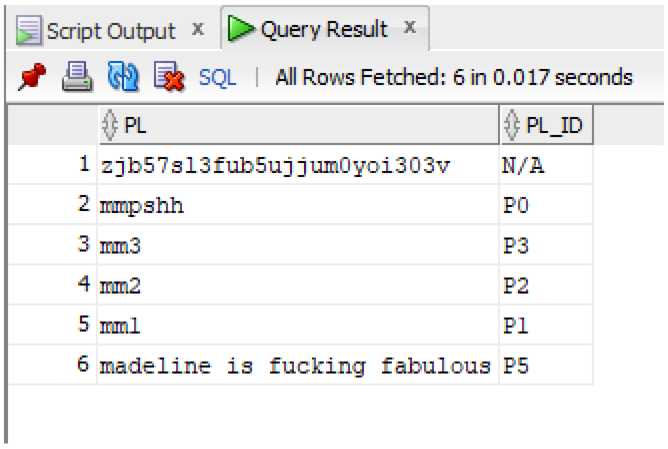
```
SELECT PL, PL_ID
FROM Playlist
```

iv) Results of the SQL


Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.0

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

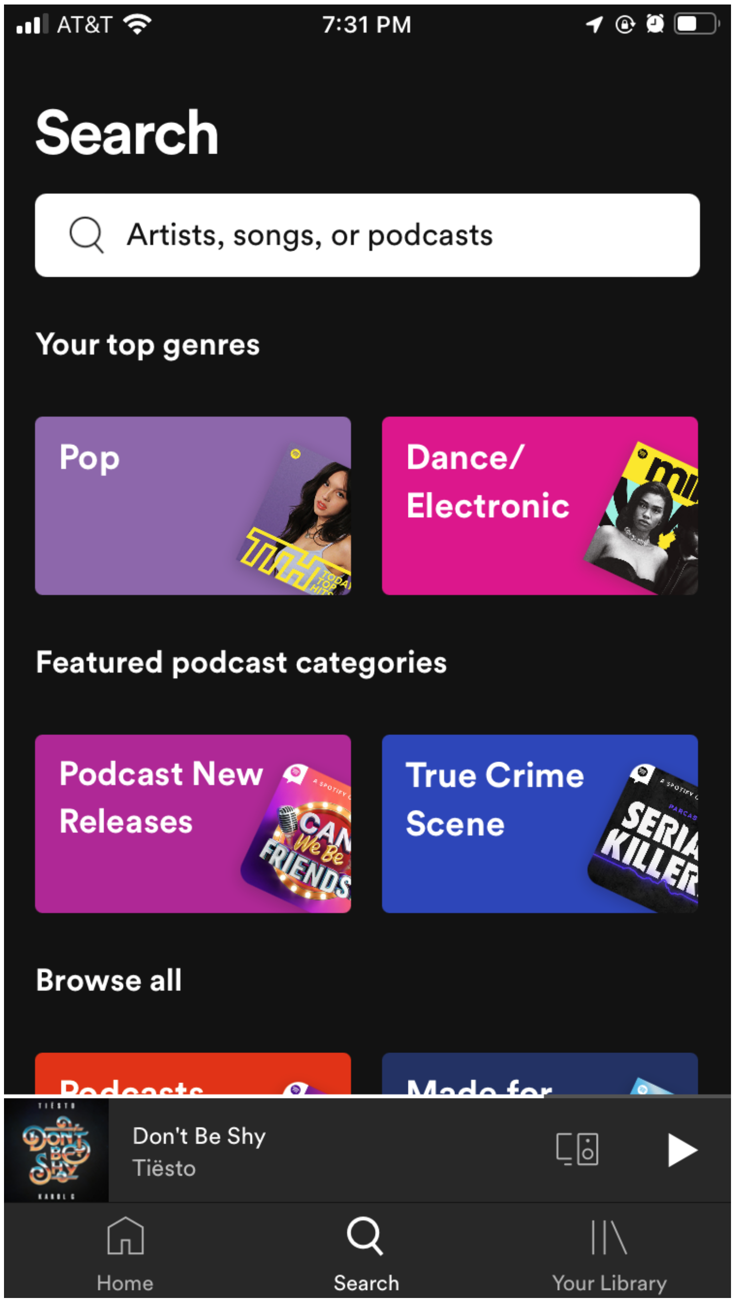

Script Output | Query Result

SQL | All Rows Fetched: 6 in 0.017 seconds

	PL	PL_ID
1	zjb57sl3fub5ujjum0yoi303v	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

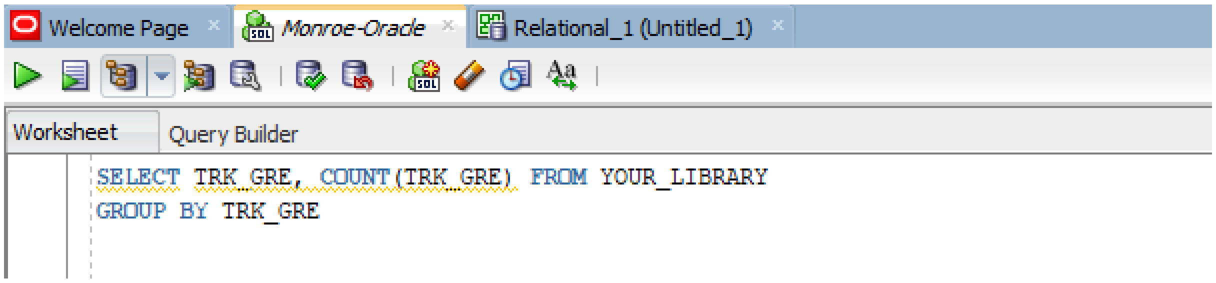
i) Snapshot of the Screen That the SQL is Providing Data For



ii) Explanation of How SQL Works

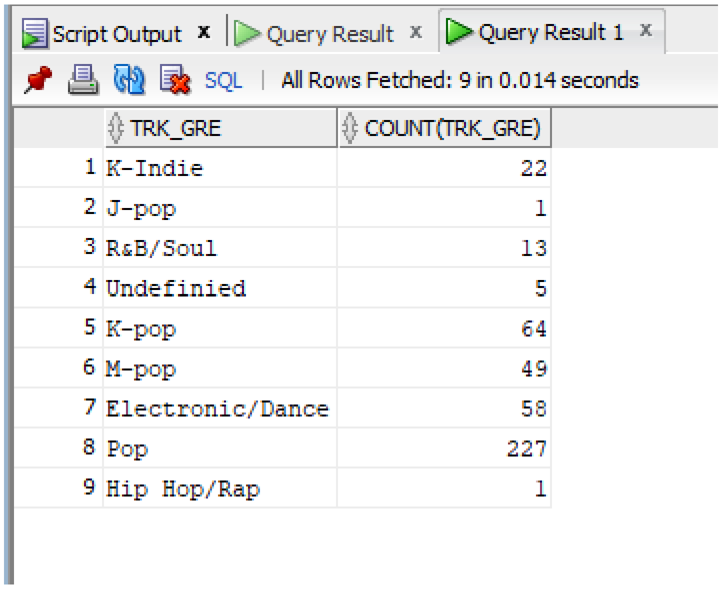
In Your Library, I selected the TRK_GRE and used Count function to count the number of TRK_GRE, grouped by TRK_GRE. It will give me the result of different genres of music that I listen to as well as the number of songs of each genre.

iii) SQL to Reproduce the Data



```
SELECT TRK_GRE, COUNT(TRK_GRE) FROM YOUR_LIBRARY
GROUP BY TRK_GRE
```

iv) Results of the SQL

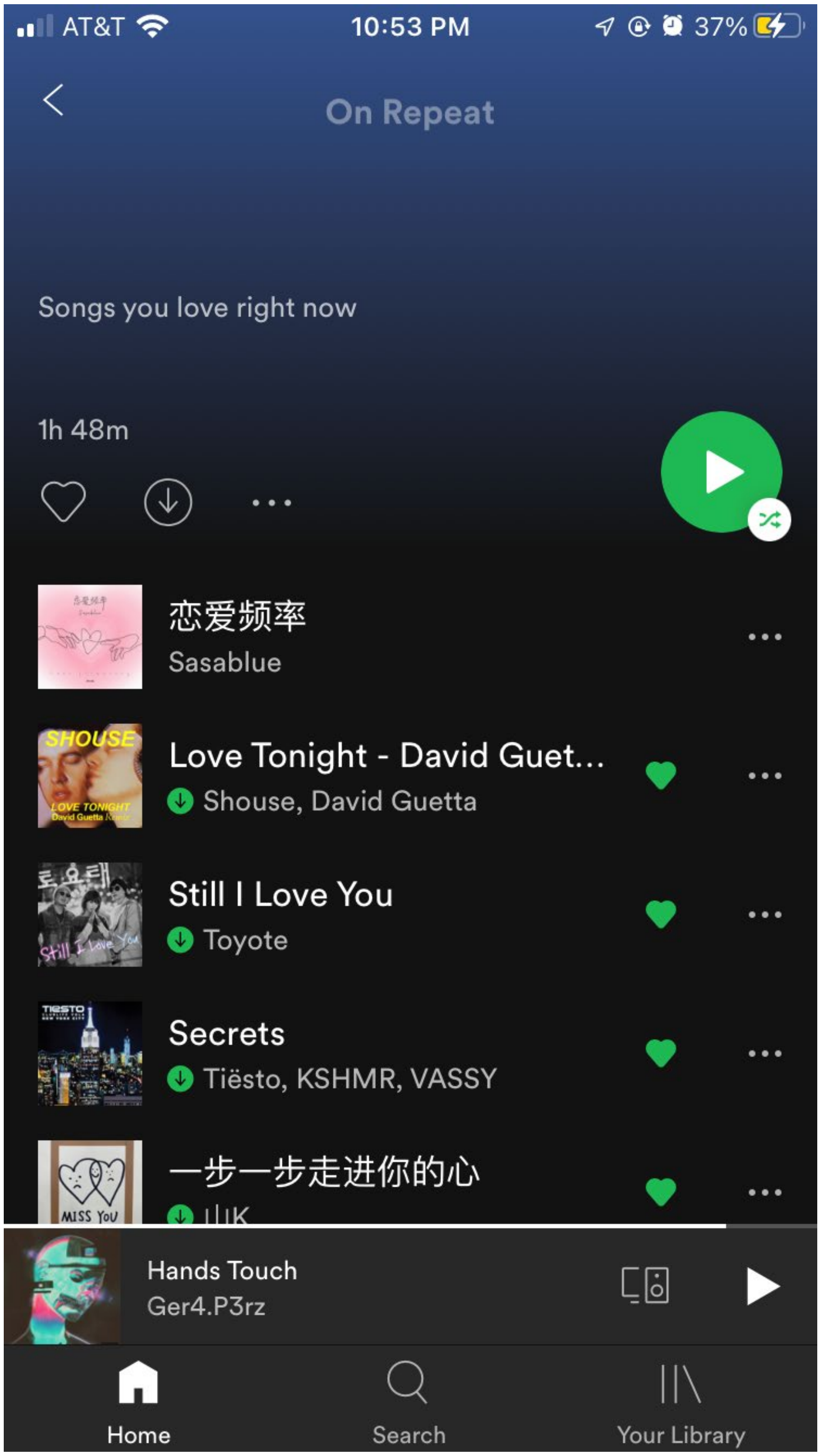


All Rows Fetched: 9 in 0.014 seconds

	TRK_GRE	COUNT(TRK_GRE)
1	K-Indie	22
2	J-pop	1
3	R&B/Soul	13
4	Undefinied	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



Songs you love right now

1h 48m

恋爱频率
Sasablue

Love Tonight - David Guet...
Shouse, David Guetta

Still I Love You
Toyote

Secrets
Tiësto, KSHMR, VASSY

一步一步走进你的心
山K

Hands Touch
Ger4.P3rz

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.

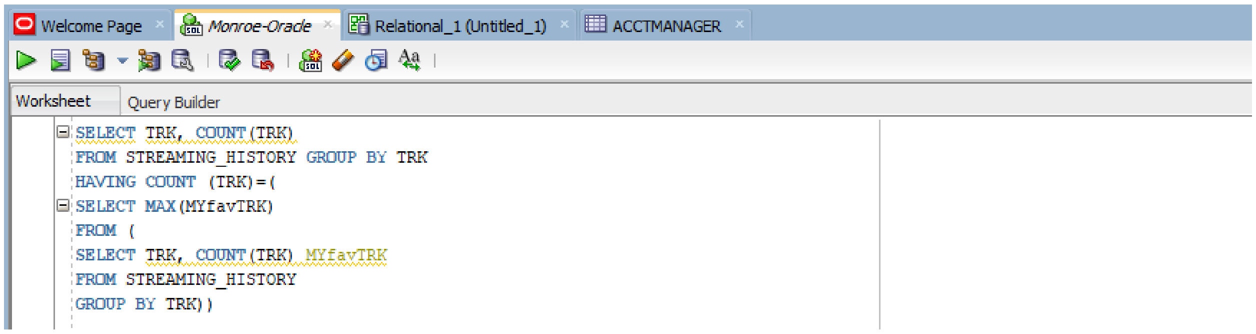
This SQL contains three subparts. First, from the Streaming History, I selected TRK and used Count function to count TRK (the number of times a track is played). I determined the new variable to be MYfavTRK (i.e. my favorite track). I also used Group by TRK. This part will return all tracks with the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

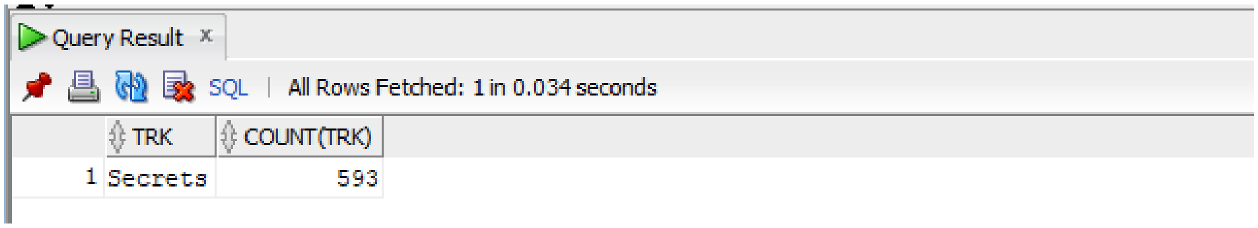
Finally, I used Select statement and Count function as shown the beginning part to extract the my onrepeat track name with its count.

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

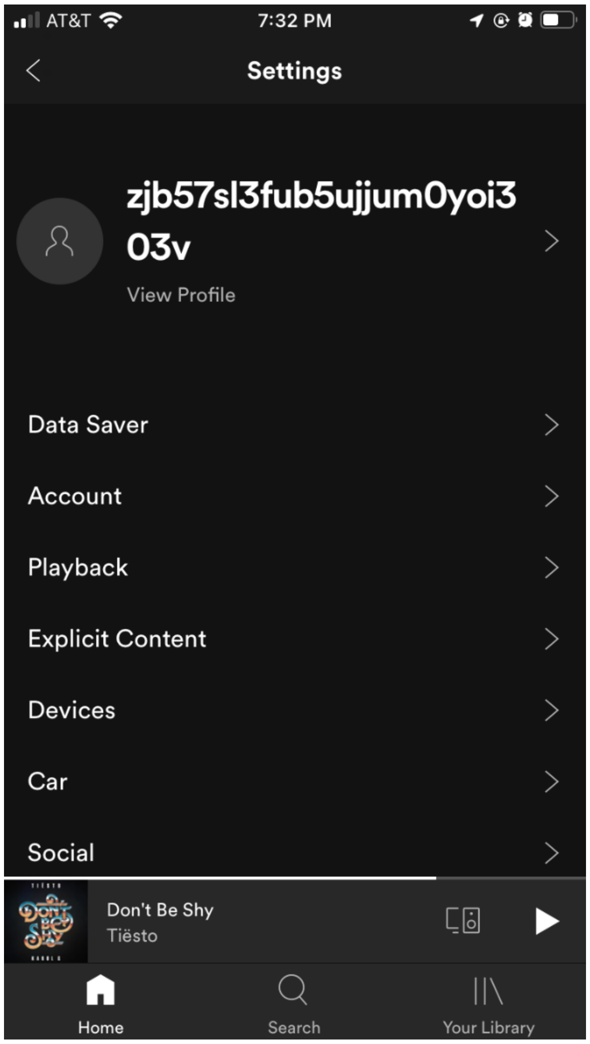


All Rows Fetched: 1 in 0.034 seconds

	TRK	COUNT(TRK)
1	Secrets	593

5. Validate my username

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



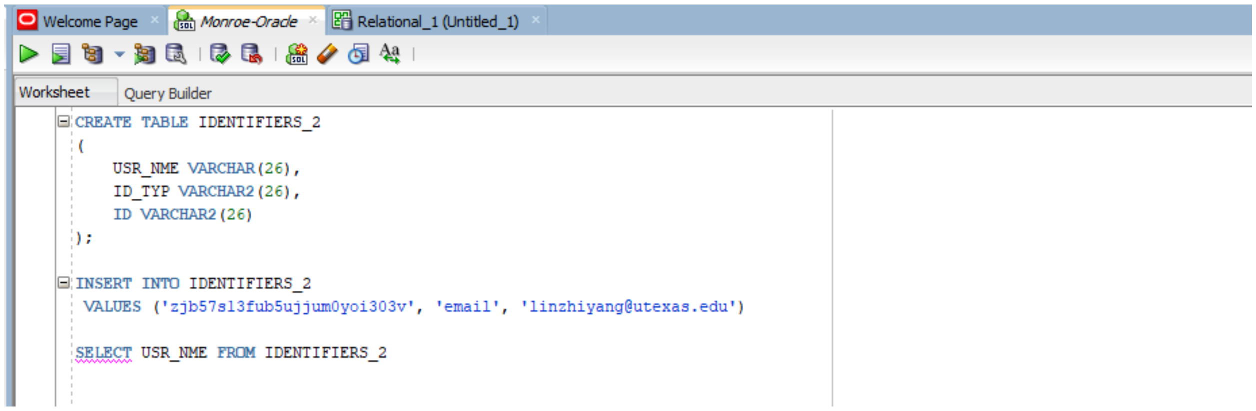
ii) Explanation of How SQL Works

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 zjb57s13fub5ujjum0yoi303v.

iii) SQL to Reproduce the Data

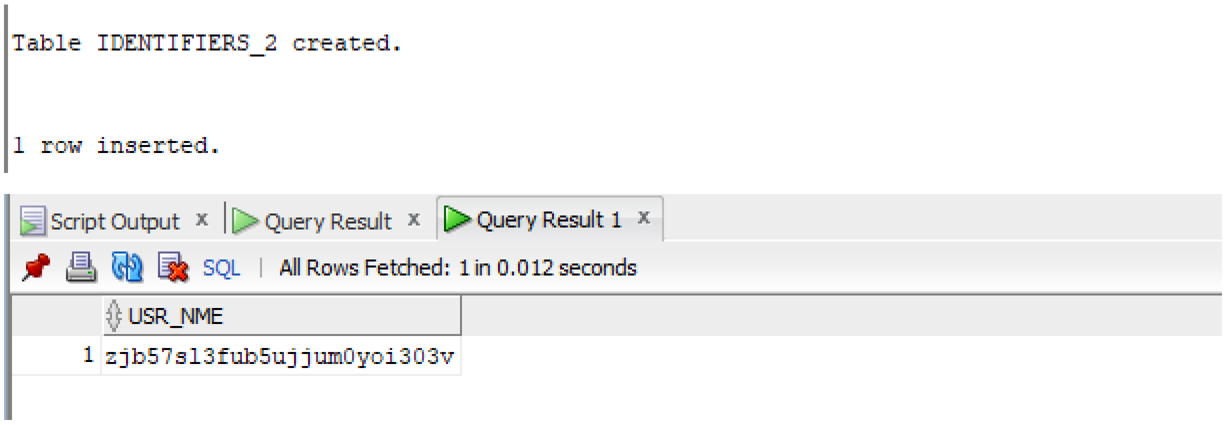


```
CREATE TABLE IDENTIFIERS_2
(
    USR_NME VARCHAR(26),
    ID_TYP VARCHAR2(26),
    ID VARCHAR2(26)
);

INSERT INTO IDENTIFIERS_2
 VALUES ('zjb57sl3fub5ujjum0yoi303v', 'email', 'linzhiyang@utexas.edu')

SELECT USR_NME FROM IDENTIFIERS_2
```

iv) Results of the SQL



```
Table IDENTIFIERS_2 created.


1 row inserted.
```

All Rows Fetched: 1 in 0.012 seconds

	USR_NME
1	zjb57sl3fub5ujjum0yoi303v