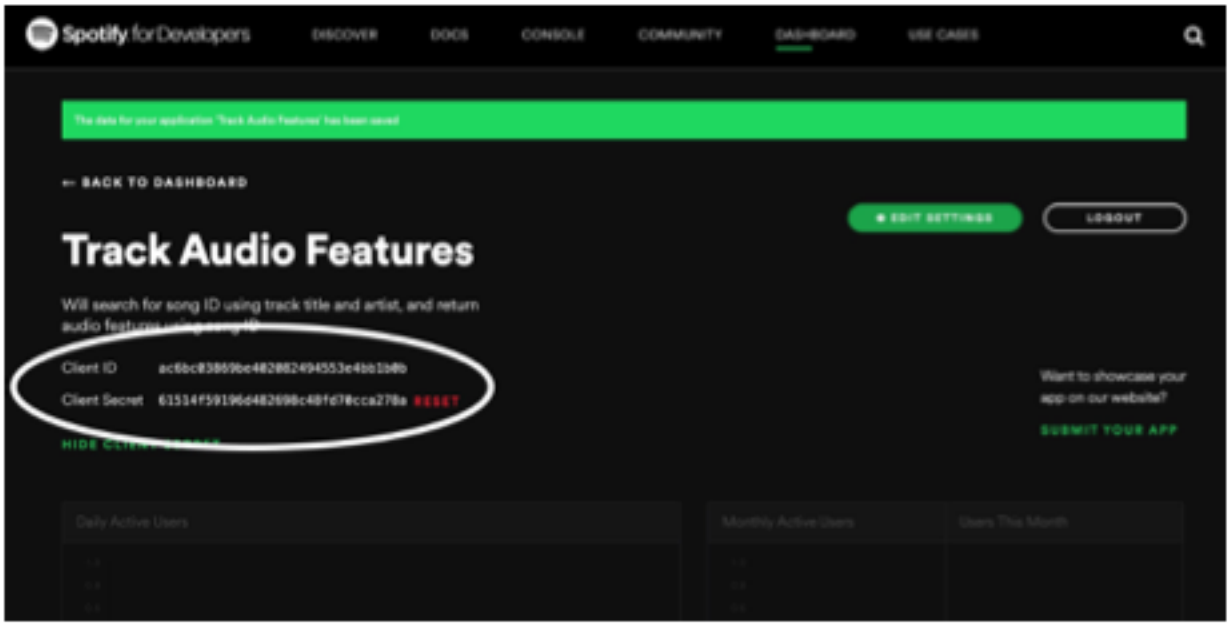
Originally wanted to be able to look at musical tastes/trends w regard to geographical location (w/n US), age, race, job type, income level, etc.

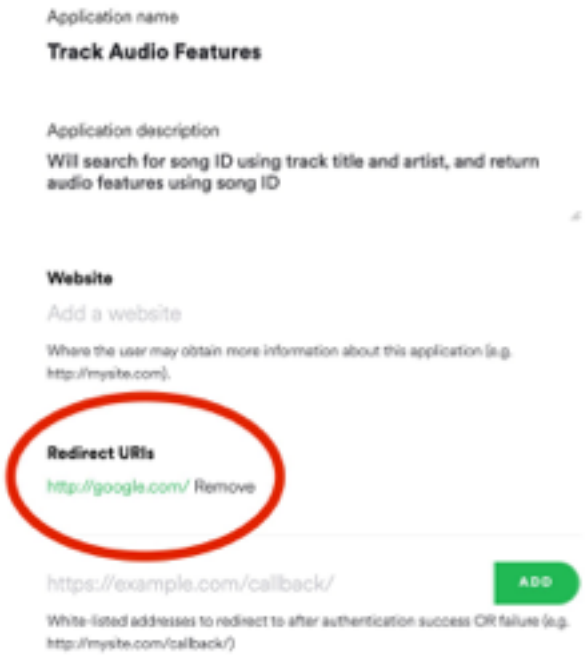
HOWEVER, the Spotify API is quite restrictive and that information would require authorization from every user.

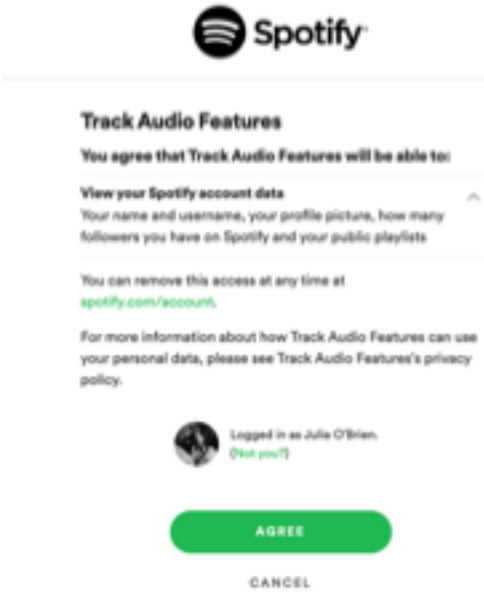
Then we decided to take a look at Spotify's track audio feature data, this also requires authorization and an access token but this can be retrieved w authorization from just one user - so I used myself.

First, created/registered an app on Spotify to collect track audio features.
This created a Client ID and Client Secret that I could use to put together an authorization url.

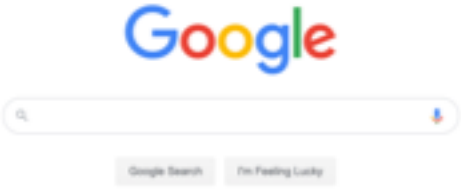


Chose http://google.com/ as my redirect uri. Once have auth url, click it and agree to let app access user account data.





Once I click “Agree” to authorize access, redirected to redirect uri, get code from redirect uri.



Copy and paste code into Jupyter Notebook to be used in the curl command needed to get the access token:

```
In [*]: # Get Spotify Auth and Access Token
client_id = "ac6bc03869be402082494553e4bb1b0b"
client_secret = "61514f59196d482698c48fd70cca278a"
redirect_uri  = "http%3A%2F%2Fgoogle.com%2F"

auth_url = f"https://accounts.spotify.com/authorize?client_id={client_id}&response_type=code&redirect_uri={redirect_uri
print(auth_url)

code_input = input("What is the code retrieved from the redirect uri? ")

# Token endpoint, CURL COMMAND TO GET TOKEN
token_base_url = "https://accounts.spotify.com/api/token"

#Base64 encoded client_id:client_secret
client_id_secret_coded = "YWM2YmMwMzg2OWJlNDAyMDgyNDk0NTUzZTRiYjFiMGI6NjE1MTRmNTkxOTZkNDgyNjk4YzQ4ZmQ3MGNjYTI3OGE="

token_curl = f"curl -H 'Authorization: Basic {client_id_secret_coded}' -d grant_type=authorization_code -d code={code_i
print(token_curl)
```

```
https://accounts.spotify.com/authorize?client_id=ac6bc03869be402082494553e4bb1b0b&response_type=code&redirect_uri=htt
p%3A%2F%2Fgoogle.com%2F

What is the code retrieved from the redirect uri? AQBlWGd8qLxyX2IKp9JaT1LVPRMKBXKzK63-4nne4EpIX4NMzE
```

Outputs curl command, copy and paste into terminal to get access token.

```
# Token endpoint, CURL COMMAND TO GET TOKEN
token_base_url = "https://accounts.spotify.com/api/token"

#Base64 encoded client_id:client_secret
client_id_secret_coded = "YWM2YmMwMzg2OWJlNDAyMDgyNDk0NTUzZTRiYjFiMGI6NjE1MTRmNTkxOTZkNDgyNjk4YzQ4ZmQ3MGNjYTI3OGE="

token_curl = f"curl -H 'Authorization: Basic {client_id_secret_coded}' -d grant_type=authorization_code -d code={code_i
print(token_curl)
```

```
https://accounts.spotify.com/authorize?client_id=ac6bc03869be402082494553e4bb1b0b&response_type=code&redirect_uri=htt
p%3A%2F%2Fgoogle.com%2F
What is the code retrieved from the redirect uri? AQBlWGd8qLxyX2IKp9JaT1LVPRMKBXKzK63-4nne4EpIX4NMzEC-5xNy5qaMUqFLEwS
WzGlzenGGJCECu79Hehm8-J6Kr4tWyff8qMKUGd1hY9_iLc0ujQUKlEJdPDZ2KXmgq3t3vGM_7dbfyrP4hBNrcBRRtpZ3VJrVjqOanWK1gpQgnPdtVg
curl -H 'Authorization: Basic YWM2YmMwMzg2OWJlNDAyMDgyNDk0NTUzZTRiYjFiMGI6NjE1MTRmNTkxOTZkNDgyNjk4YzQ4ZmQ3MGNjYTI3OGE
=' -d grant_type=authorization_code -d code=AQBlWGd8qLxyX2IKp9JaT1LVPRMKBXKzK63-4nne4EpIX4NMzEC-5xNy5qaMUqFLEwSWzGlze
nGGJCECu79Hehm8-J6Kr4tWyff8qMKUGd1hY9_iLc0ujQUKlEJdPDZ2KXmgq3t3vGM_7dbfyrP4hBNrcBRRtpZ3VJrVjqOanWK1gpQgnPdtVg -d redi
rect_uri=http%3A%2F%2Fgoogle.com%2F https://accounts.spotify.com/api/token
```

```
(base) Julias-MacBook-Air:~ jobrien1726$ curl -H 'Authorization: Basic YWM2YmMwM
zg2OWJlNDAyMDgyNDk0NTUzZTRiYjFiMGI6NjE1MTRmNTkxOTZkNDgyNjk4YzQ4ZmQ3MGNjYTI3OGE='
 -d grant_type=authorization_code -d code=AQBlWGd8qLxyX2IKp9JaT1LVPRMKBXKzK63-4n
ne4EpIX4NMzEC-5xNy5qaMUqFLEwSWzGlzenGGJCECu79Hehm8-J6Kr4tWyff8qMKUGd1hY9_iLc0ujQ
UKlEJdPDZ2KXmgq3t3vGM_7dbfyrP4hBNrcBRRtpZ3VJrVjqOanWK1gpQgnPdtVg -d redirect_uri
=http%3A%2F%2Fgoogle.com%2F https://accounts.spotify.com/api/token
{"access_token":"BQCMUUmmdtKp2JwX9M3irsTb_XqIQ-oeGDReqIsBRourjwVW3C3pSEDYePz9Ub0
OPOGSw_Khlz-bON6g_Gs6FMgzWePnjzbgR-QRF8ApcZv-89NBeZKM-RJURV2sttsrwgxlayGsqRS0ChU
pnaGg","token_type":"Bearer","expires_in":3600,"refresh_token":"AQC9SI0cEHrxkIAc
o4tBqFyydl1P3ylhAKrFeaOb36ErjqG7ZfqRRI6-WwcolxnPaDvXxiG8T8vvBQmRfN2yMFbYVLwcMxY_
nD16remRTaBef726ojTo7YVXlVMmauT5zoM","scope":""}(base) Julias-MacBook-Air:~ jobr
```

Once have access token, can proceed with requests to Spotify API.
Read in Billboard csv file with Number One Songs from every week for the last 20 years, extract Song Year from Billboard Chart Date and save to separate column.

In [*]:
```
# Run token curl in terminal for access token
access_token = input("What is the access code retrieved? ")
```

```
What is the access code retrieved? BQCMUUmmdtKp2JwX9M3irsTb_XqIQ-oeGDReqIsBRourjwVW3C
```

In [ ]:
```
# Read in CSV file with Top 100 song hits for the past 20 years
top_songs_df = pd.read_csv("Resources/top_songs.csv")
top_songs_df.head()
```

In [ ]:
```
#Remove unnecessary columns
top_songs_df = top_songs_df.drop(columns=["Unnamed: 0"])
top_songs_df.head()
```

In [ ]:
```
#Extract Year from Chart Date and add to own column to be used in Spotify API request
top_songs_df["Year"] = pd.DatetimeIndex(top_songs_df["Chart Date"]).year
top_songs_df.head()
```

Iterate through Billboard DataFrame, use Song Title, Year and Access Token to put together Spotify search item url, make request to Spotify and extract Song ID and Spotify Popularity Rating for each song.

In [ ]:
```
# Now that we have authorization tokens, and top hits for the last 20 years, use Spotify's search item API
#to extract song info

# Print a statement showing API Calls have begun
print("Beginning Data Retrieval")
print("-----------------------------------------")

for index, row in top_songs_df.iterrows():
    title = row["Title"]
    #artist = row["Artist"]
    year = row["Year"]

    #search_url = ("https://api.spotify.com/v1/search?q={0}%20artist:{1}%20year:{2}&type=track&market=US").format(song_
    search_url = f"https://api.spotify.com/v1/search?q={title}%20year:{year}&type=track&market=US"

    headers = {
        "Accept": "application/json",
        "Content-Type": "application/json",
        "Authorization": "Bearer " + access_token
    }

    song_response=requests.get(search_url, headers=headers).json()
    #print(json.dumps(song_response, indent=4))

    try:
        print("Processing Chart from week " + str(row["Chart Date"]))

        top_songs_df.loc[index, "Song ID"] = song_response["tracks"]["items"][0]["id"]

        top_songs_df.loc[index, "Popularity"] = song_response["tracks"]["items"][0]["popularity"]

    except (KeyError, IndexError):
        print("Track not found. Skipping this track...")
        top_songs_df.loc[index, "Song ID"] = "NaN"
        top_songs_df.loc[index, "Popularity"] = "NaN"

print("-----------------------------------------")
print("Data Retrieval Complete")
print("-----------------------------------------")

display(top_songs_df)
```

Once we have Song ID and Popularity Rating added to DataFrame, clean up DataFrame by extracting only rows that have a Song ID available (this is what we will need to get audio features).

Rename and Reorganize columns, save DataFrame to csv file

In [ ]:
```
#Drop rows lacking a Song ID
top_songs_detail = top_songs_df.loc[top_songs_df["Song ID"] != "NaN", :]
display(top_songs_detail)
```

In [ ]:
```
#Reorganize and Rename Columns
top_songs_detail = top_songs_detail[["Chart Date", "Title", "Artist", "Year", "Song ID", "Popularity",
                                     "Number of Weeks In Top 100"]].rename(columns={"Chart Date": "Billboard Chart Date
                                                                                    "Song ID": "Spotify Song ID",
                                                                                    "Popularity": "Spotify Popularity I

top_songs_detail.head()
```

In [ ]:
```
#Save DataFrame to CSV file
top_songs_detail.to_csv("Resources/top_songs_detail.csv")
```

Iterate through new dataframe, use Song ID to create audio feature search url, make requests to Spotify API and extract all audio features, save each to own column in dataframe.

In [ ]:
```
# Use Song ID to search for Audio Features in Spotify API

# Print a statement showing API Calls have begun
print("Beginning Data Retrieval")
print("-----------------------------------------")

for index, row in top_songs_detail.iterrows():
    song_id = row["Spotify Song ID"]

    audio_url = f"https://api.spotify.com/v1/audio-features/{song_id}"

    headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
    "Authorization": "Bearer " + access_token
}

    audio_response = requests.get(audio_url, headers=headers).json()
    #print(json.dumps(audio_response, indent=4))
```


```
{
    "danceability": 0.695,
    "energy": 0.762,
    "key": 0,
    "loudness": -3.497,
    "mode": 1,
    "speechiness": 0.0395,
    "acousticness": 0.192,
    "instrumentalness": 0.00244,
    "liveness": 0.0863,
    "valence": 0.553,
    "tempo": 120.042,
    "type": "audio_features",
    "id": "21jGcNKet2qwijlDFuPiPb",
    "uri": "spotify:track:21jGcNKet2qwijlDFuPiPb",
    "track_href": "https://api.spotify.com/v1/tracks/21jGcNKet2qwijlDFuPiPb",
    "analysis_url": "https://api.spotify.com/v1/audio-analysis/21jGcNKet2qwijlDFuPiPb",
    "duration_ms": 215280,
    "time_signature": 4
}
```

```
    try:
        print("Fetching Audio Features for " + str(row["Title"]))
        top_songs_detail.loc[index, "Danceability"] = audio_response["danceability"]
        top_songs_detail.loc[index, "Energy"] = audio_response["energy"]
        top_songs_detail.loc[index, "Key"] = audio_response["key"]
        top_songs_detail.loc[index, "Loudness"] = audio_response["loudness"]
        top_songs_detail.loc[index, "Mode"] = audio_response["mode"]
        top_songs_detail.loc[index, "Speechiness"] = audio_response["speechiness"]
        top_songs_detail.loc[index, "Acousticness"] = audio_response["acousticness"]
        top_songs_detail.loc[index, "Instrumentalness"] = audio_response["instrumentalness"]
        top_songs_detail.loc[index, "Liveness"] = audio_response["liveness"]
        top_songs_detail.loc[index, "Valence"] = audio_response["valence"]
        top_songs_detail.loc[index, "Tempo"] = audio_response["tempo"]
        top_songs_detail.loc[index, "Duration (ms)"] = audio_response["duration_ms"]
        top_songs_detail.loc[index, "Time Signature"] = audio_response["time_signature"]

    except (KeyError, IndexError):
        print("Audio features for this Song ID not found. Skipping...")
        top_songs_detail.loc[index, "Danceability"] = "NaN"
        top_songs_detail.loc[index, "Energy"] = "NaN"
        top_songs_detail.loc[index, "Key"] = "NaN"
        top_songs_detail.loc[index, "Loudness"] = "NaN"
        top_songs_detail.loc[index, "Mode"] = "NaN"
        song_detail_sub.loc[index, "Speechiness"] = "NaN"
        top_songs_detail.loc[index, "Acousticness"] = "NaN"
        top_songs_detail.loc[index, "Instrumentalness"] = "NaN"
        top_songs_detail.loc[index, "Liveness"] = "NaN"
        top_songs_detail.loc[index, "Valence"] = "NaN"
        top_songs_detail.loc[index, "Tempo"] = "NaN"
        top_songs_detail.loc[index, "Duration (ms)"] = "NaN"
        top_songs_detail.loc[index, "Time Signature"] = "NaN"

print("-----------------------------------------")
print("Data Retrieval Complete")
print("-----------------------------------------")

display(top_songs_detail)
```

Change Duration column from ms to mins, change name of DataFrame and save to csv file.

In [ ]:
```
#Change name of DataFrame
audio_feature_data = top_songs_detail
audio_feature_data
```

In [ ]:
```
#Convert Duration column from ms to mins
audio_feature_data["Duration (ms)"]= audio_feature_data["Duration (ms)"] / 60000

audio_feature_data = audio_feature_data.rename(columns={"Duration (ms)": "Duration (mins)"})

audio_feature_data.head()
```

In [ ]:
```
# Save our dataframe to CSV
audio_feature_data.to_csv("Resources/audio_feature_data.csv")
```

Realized no access to genre or explicitness data via Spotify API so used iTunes API for that information.

In [ ]:
```
#Use iTunes API to collect genre and explicitness data

# Print a statement showing API Calls have begun
print("Beginning Data Retrieval")
print("-----------------------------------------")

for index, row in audio_data_df.iterrows():
    title = row["Title"]
    artist = row["Artist"]
    year = row["Year"]

    itunes_url = f"https://itunes.apple.com/search?term={title}+{artist}&year={year}&entity=musicTrack&media=music&cour

    itunes_response = requests.get(itunes_url).json()
    #print(json.dumps(itunes_response, indent=4))

    try:
        print("Processing Track Details for " + str(row["Title"]))

        audio_data_df.loc[index, "Genre"] = itunes_response["results"][0]["primaryGenreName"]

        audio_data_df.loc[index, "Explicitness"] = itunes_response["results"][0]["trackExplicitness"]

    except (KeyError, IndexError):
        print("Track Info not found. Skipping this track...")
        audio_data_df.loc[index, "Genre"] = "NaN"
        audio_data_df.loc[index, "Explicitness"] = "NaN"

    sleep(1)

print("-----------------------------------------")
print("Data Retrieval Complete")
print("-----------------------------------------")

display(audio_data_df)
```

Reorganize Columns, save completed/ FINAL DataFrame to csv file.

In [ ]:
```
#Reorganize Columns
track_data_complete = audio_data_df[["Billboard Chart Date", "Title", "Artist", "Genre", "Year",
                                     "Spotify Song ID", "Spotify Popularity Rating",
                                     "Number of Weeks In Top 100", "Danceability", "Energy",
                                     "Key", "Loudness", "Mode", "Acousticness",
                                     "Instrumentalness", "Liveness", "Valence", "Tempo",
                                     "Explicitness", "Duration (mins)", "Time Signature"]]
track_data_complete.head()
```

In [ ]:
```
# Save to CSV
track_data_complete.to_csv("Resources/track_data_complete.csv")
```

	Billboard Chart Date	Year	Month	Title	Artist	Genre	Spotify Song ID	Spotify Popularity Rating	Number of Weeks In Top 100	Danceability	Energy	Key	Loudness	Mode
0	2019-12-14	2019	12	Heartless	The Weeknd	R&B/Soul	57vxBYX0HMM6H1aD09V7PU	94.0	2	0.531	0.750	10.0	-5.831	0.0
1	2019-12-07	2019	12	Circles	Post Malone	Hip-Hop/Rap	21jGcNKet2qwijlDFuPiPb	99.0	13	0.695	0.762	0.0	-3.497	1.0
2	2019-11-30	2019	11	Circles	Post Malone	Hip-Hop/Rap	21jGcNKet2qwijlDFuPiPb	99.0	12	0.695	0.762	0.0	-3.497	1.0
3	2019-11-23	2019	11	Someone You Loved	Lewis Capaldi	Alternative	7qEHsqek33rTcFNT9PFqLf	96.0	27	0.501	0.405	1.0	-5.679	1.0
4	2019-11-16	2019	11	Someone You Loved	Lewis Capaldi	Alternative	7qEHsqek33rTcFNT9PFqLf	96.0	26	0.501	0.405	1.0	-5.679	1.0
5	2019-11-09	2019	11	Lose You To Love Me	Selena Gomez	Pop	1HfMVBKM75vxSfsQ5VeGZ5	98.0	2	0.505	0.340	4.0	-9.005	1.0
6	2019-11-02	2019	11	Someone You Loved	Lewis Capaldi	Alternative	7qEHsqek33rTcFNT9PFqLf	96.0	24	0.501	0.405	1.0	-5.679	1.0
7	2019-10-26	2019	10	Truth Hurts	Lizzo	Pop	5qmqI61DAAOUaWBAUo8xKhh	91.0	24	0.715	0.624	4.0	-3.046	0.0
8	2019-10-19	2019	10	HIGHEST IN THE ROOM	Travis Scott	Hip-Hop/Rap	3eekarcy7kvN4yt5ZFzltW	97.0	1	0.598	0.427	7.0	-8.764	0.0
9	2019-10-12	2019	10	Truth Hurts	Lizzo	Pop	5qmqI61DAAOUaWBAUo8xKhh	91.0	22	0.715	0.624	4.0	-3.046	0.0
10	2019-10-05	2019	10	Truth Hurts	Lizzo	Pop	5qmqI61DAAOUaWBAUo8xKhh	91.0	21	0.715	0.624	4.0	-3.046	0.0

Speechiness	Acousticness	Instrumentalness	Liveness	Valence	Tempo	Explicitness	Duration (mins)	Time Signature
0.1110	0.00632	0.000076	0.1170	0.1980	169.954	1.0	3.334667	4.0
0.0395	0.19200	0.002440	0.0863	0.5530	120.042	0.0	3.588000	4.0
0.0395	0.19200	0.002440	0.0863	0.5530	120.042	0.0	3.588000	4.0
0.0319	0.75100	0.000000	0.1050	0.4460	109.891	0.0	3.036017	4.0
0.0319	0.75100	0.000000	0.1050	0.4460	109.891	0.0	3.036017	4.0
0.0438	0.57600	0.000000	0.2100	0.0916	101.993	0.0	3.440983	4.0
0.0319	0.75100	0.000000	0.1050	0.4460	109.891	0.0	3.036017	4.0
0.1140	0.11000	0.000000	0.1230	0.4120	158.087	1.0	2.888750	4.0
0.0317	0.05460	0.000006	0.2100	0.0605	76.469	1.0	2.928683	4.0
0.1140	0.11000	0.000000	0.1230	0.4120	158.087	1.0	2.888750	4.0
0.1140	0.11000	0.000000	0.1230	0.4120	158.087	1.0	2.888750	4.0