1. Token Acquisition Notebook

Intro

When we first got assigned as a group for this final project, we knew exactly that we wanted to use Spotify. Both of us listen to a lot of music and wanted to see the trends of music from mid 1940s till present day. Spotify is very generous in that they give you information all about the specific song that you ask for. So for this project we are taking this information that they are allowing us to acess and see if there are any trends in music and if there are specific notes are used by all hits notes. We believe that this will be different then any other Spotify project because no one will go into the depth that we are going into the music and hope you enjoy our project!

Functions

The first think needed in our steps of obtain and then further analyzing our Spotify data had to meant working with the Spotify API to access their database of songs and their features to measure trends. To do this we used a handful of functions over and over again to perform operations on the API, saving responses and saving the data for ourselves

In [12]:
```
%run functions.py # Run functions.py to access most functions for use
```

In [13]:
```
def addcreds(dic, key):
    """
    The following function takes a dictionary and key input. What is doe
s is opens the creds.json uses the key
    as a key in the creds.json dictionary and appends the input dictiona
ry into the creds["key"] spot. This is
    helpful as a function in saving info that comes along with the OAuth
 process.
    """
    with open("creds.json", "r+") as file: # Open json file
        creds = json.load(file) # Define creds as JSON file
        creds[key] = dic # Create new key with value as dic
        file.seek(0)  # Rewind
        json.dump(creds, file) # Dump creds file
        file.truncate() # Turncate
```

OAuth Authentication/Authorization

Before constructing the URL to give to the User/Resource Owner, we had to create an application in Spotify in order to get a clientID and client secret. You have to do this, so that Spotify can accept your application. After completing this task of creating an application, we moved onto constucting the URL in order to move closer to get authorization from Spotify.

Construct the URL to give to the User/Resource Owner

Below we have our parameters: protocol, location, resource. By pluggin in the specific values for each and run it through our BuildURL function that we have created in our 'functions.py' file we have the ability to ability to create a URL to put into our HTTP get function. After running through the HTTPGet function, we will now get a the url that we should click on.

Although this sounded really easy, we had some problems getting to this point. At first we created the url, we were not using the correct parameters for the protocol, location, and the resource. Once, we got this fixed it was printing out the correct URL, although it was not letting us get to the screen to give authorization to our application. After consulting with Professor, we decided to restart our kernal. Once we did this, our authorization url worked.

In [14]:
```
creds = opencreds() # Define creds from open creds function
protocol = 'https' # Define protocol
location = 'accounts.spotify.com' # Define location
resource = 'authorize' # Define resource
url = buildURL(protocol, location, resource) #Creates the URL with the s
pecific protocol, location, and resource given above.
HTTPGet(url, creds["Spotify Authorization"], "Y") #Uses the HTTPGet func
tion, which we created in the functions.py, to give the authorization en
dpoints, in order to create a specific authorization URL.
```

```
https://accounts.spotify.com/authorize?client_id=c2d716c696dd4831abf543
c799e6b764&client_secret=ad83971bb3314ccfbb3b07b679cfe398&redirect_uri=
https%3A%2F%2Flocalhost%2Fcallback%2F&response_type=code
```

After clicking this URL, we were directed into signining into our spotify account. By doing this, we have given our application that we have created, the ability to acesses the information into our personal spotify account.

Obtain Authorization Code

Now that we have been redericted to the redirect URL that we have made in our spotify app, we were given our authorization code inbedded into this URL. In order to get this code out, we decided to use a regular expression to get the authroization code. After retrieving the authorization code, we inserted it into a dictionary, which we then used our 'addcreds' function from our 'functions.py' file in order to expand our creds file.

In [19]:
```
authorization_link = "https://localhost/callback/?code=AQAg7lSawKi5QOGUr
-xjOJY_B33YpN2WqMejWvxtoQsBwXS4PpFmluGEUyuCX6LLsbk4m_oQ5ZrUZey9jbokikrFC
an3OF6qgKFhGwWzhCSyrAWffIzdrQalUq1de5tepCf_rp_xYHA_EbrQO49eYmjOpKdbMpjWu
Rn_pDrCY-C4rKzYqyGf--2mQAMv0us7og" # Paste link here
authorization_code = re.search(r'(?<=code=).*', authorization_link).grou
p() # Use regular expressions to parse code
authorization_code
```

Out[19]:
```
'AQAg7lSawKi5QOGUr-xjOJY_B33YpN2WqMejWvxtoQsBwXS4PpFmluGEUyuCX6LLsbk4m_
oQ5ZrUZey9jbokikrFCan3OF6qgKFhGwWzhCSyrAWffIzdrQalUq1de5tepCf_rp_xYHA_E
brQO49eYmjOpKdbMpjWuRn_pDrCY-C4rKzYqyGf--2mQAMv0us7og'
```

To always be consistent in saving where we left off in the OAuth process, each time we went through different steps in the OAuth dance, we would document our responses in our creds.json file. The following code is appending a dictionary in creds with the authorization code just retrieved in the above code.

In [16]:
```
dic = {"code": authorization_code,
       'grant_type': "authorization_code",
       'redirect_uri': creds["Spotify Authorization"]['redirect_uri'],
       'client_id':creds["Spotify Authorization"]['client_id'],
       'client_secret': creds["Spotify Authorization"]['client_secret']}
 # The authorization_code is created in the above code chunk and is adde
d to the creds.json file
addcreds(dic, "Spotify Token") # Add dic to creds file
```

Obtain Access Token and Refresh Token

Now that we have our authorization code in our creds file we again can get into creating the specific protocol, location, and reource we need to get the access token and refresh token. In order to do get this token, Spotify wants us to use a post, and then we set the specific parameters needed to get this token.

While getting this token, we found out that we were not having the corrrect parameters set out by Spotify to get a token. The main issue we ran into while obtaining the access and refresh token had to do with the auth parameter in the requests.post. Spotify required in this parameter that your client id and client secret are encoded in a certain Base 64 type. After making adjustments we were able to get a token, which comes via a JSON file token response.

In [17]:
```
creds = opencreds() # Define creds from open creds function
protocol = 'https' # Define protocol
location = 'accounts.spotify.com' # Define location
resource = 'api/token' # Define resource
url = buildURL(protocol, location, resource) # Define url using buildURL
 function
resp = requests.post(url, auth=HTTPBasicAuth(creds['Spotify HTTPBasicAut
h']['client_id'], creds['Spotify HTTPBasicAuth']['client_secret']), data
=creds["Spotify Token"]) # Post function to get refresh token
token_response = resp.json() # Get JSON response
token_response
```

Out[17]:
```
{'access_token': 'BQCnpcvmb1Sogfx7-tVOX0Fb6XzONT5ohHxQ4urbGzd9yamYIUsSD
WeVyj0vlU6C_7mH8f63saUkIIwm0reX5L--UvH3Boj7hIgpGR_6ong2_fT9oT--GID9WSb3
qa9USXAkC0m4XnUKPYlGzDIqlk_9XlvYxspaEw',
 'expires_in': 3600,
 'refresh_token': 'AQAskizjsvTy4FrLHdci4VW67mMPSMpQN1l2r4qa_8Zms-MIzymD
SBmRtOQqpRpTJ2AuHTbbwGJAYNXRo6OAUwjsPeD5jsDWmpejeUtoFsP5j941qQcuMesHo3K
5FzEPY8A',
 'scope': '',
 'token_type': 'Bearer'}
```

After getting this, we wanted to get our refresh token in our addscred function, so that we can have access to our application throughout this whole project. The function of the refresh token is vital to working on our project over time. Performing a post request with the refresh token at anytime will refresh the access token. So we can write a function using this refresh token to automatically refresh each time we want to make a request.

In [18]:
```
dic = {"refresh_token": token_response["refresh_token"],
      "grant_type": "refresh_token"} # Define dic of refresh token info
addcreds(dic, "Spotify Refresh") # Add refresh info to creds
```

Below, the 'getToken' function, helps get a refresh token everytime we run this, so that we can have access to our app and able to use it through the whole project. We wrote this at the end of this notebook so that we can add it to our functions.py file to get the proper authorization for each request from the Spotify API.

In [20]:
```
def getToken():
    '''
    The following functions does not take any inputs. It opens our creds
 file and uses info to refresh the
    access token of the Spotify API. It then returns the access token. I
t is helpful to have a function that
    refreshes the token fro requests rather than doing it over and over
 again. This function will be added to
    creds.json for further use.
    '''
    creds = opencreds() # Use opencreds to get creds
    protocol = 'https' # Define protocol
    location = 'accounts.spotify.com' # Define location
    resource = 'api/token' # Define resource
    url = buildURL(protocol, location, resource) # use the buildURL to d
efine url
    auth = HTTPBasicAuth(creds['Spotify HTTPBasicAuth']['client_id'], cr
eds['Spotify HTTPBasicAuth']['client_secret']) # Create auth with HTTPBa
sicAuth
    data = creds["Spotify Refresh"] # define data as creds["Spotify Refr
esh"]
    resp = requests.post(url, auth=auth, data=data) # Post requests to g
et new refrshed token
    code = resp.json()["access_token"] # Parse token
    return code # Return token
```

We're off to a pretty great start! Following the OAuth Dance of the Spotify API was pretty straight forward. In summary, the difficulties that we faced mostly had to do with getting the data, header and auth parameters for our requests to the Spotify API. What made things very simple in going through this was adding obtained new responses from Spotify into our creds.json file. This allowed the program to handle the capturing of the string responses needed for further analysis.

With the getToken function that we have written at the end of this notebook, we are using a summation of going through each step of the OAuth dance to get proper authorization to access the Spotify API. We can now use this function in the furture to provide proper authorization to actual data requests from the Spotify database.