Add your name and PID
Kush Parmar A14032759
Alex Biz A14455796
Jessica Cramer A12606890
Abhik Roy A14421125
Rachel Berge A13422215

Introduction

Overview

Our group analyzed the relationship between profanity in song lyrics and its popularity in general, and within different music genres including metal, hip-hop, pop, and rock. Understanding the popularity of songs with profane lyrics helped us gauge the change in societal conventions through time. Furthermore, we were interested in finding if, and how profanity might influence song popularity. We scrape the popularity data using the spotify API, and compute the profanity data by using a pre-trained SVM. We then do regression analysis to find the correlation.

Research Question

Is there a correlation between the number of profane words in a song and the song's popularity (i.e. its Spotify rating)? Will this change when controlling for genre?

Hypothesis

We hypothesize that the number of profane lyrics will positively correlate with popularity in the genres of rap/hip-hop and pop. These genres often have underlying themes that lend themselves to the presence of profanity, such as violence and rebellion for rap and such as love and recreation for pop. Furthermore, these genres are generally meant for consumption by non-children audiences and thus profanity may be more present and thus more of a correlation with popularity.

Background

In recent years, profanity in popular culture has seemed to be increasing, leading to many analyses and research articles on why we curse and whether or not cursing is good or bad. With the advent of the Internet, the access of progressively younger generations to more profane media has been steadily increasing. But is this just a flawed perception of culture and profanity prevalence that simply comes from the interpreter becoming older? According to a research article by Twenge et al., profanity actually is becoming more prevalent in written media (in particular, books), a correlate for culture [1].

In music, another correlate for culture, the presence of profanity also seems to be increasing, especially since the turn of the millennia. This is backed by data analysis as well, with around 40% of Billboard Hot 100 #1 singles having the label of 'explicit' from 2002-2008 and over 60% of these having said label in 2017 [2]. With this background, some positive correlation between profanity and popularity in music, especially within recent years, seems clear. Can this relationship be generalized further, where the number of profane lyrics correlates with individual song popularity?

Sources:

1. Twenge, J. et al. (2017). The seven words you can never say on television: Increases in the use of swear words in American Books, 1950-2008. *SAGE Open*. DOI: 10.1177/2158244017723689
2. Bannister, M. (2017). The billboard hot 100: Exploring six decades of number one singles. *Github*. URL: https://github.com/mspbannister/dand-p4-billboard/blob/master/Billboard_analysis__100417_.md#the-billboard-hot-100-exploring-six-decades-of-number-one-singles (https://github.com/mspbannister/dand-p4-billboard/blob/master/Billboard_analysis__100417_.md#the-billboard-hot-100-exploring-six-decades-of-number-one-singles)

Dataset Description

link to dataset(s)

- https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics (https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics)
- This dataset is the set of 380,000 songs and their respective lyrics.
- https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-tracks/ (https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-tracks/)
- This dataset is the set of songs that correspond to the songs and lyrics provided by kaggle.
- https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-audiofeatures/ (https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-audiofeatures/) This dataset contains the data that corresponds to the popularity of each song.

Number of observations: 380,000+

Variables included in dataset: Kaggle Dataset: Song Name, Artist, Year, Genre, Lyrics Scraped from Spotify API: Popularity, Audio Features

Source of the data: Kaggle, and scraping different attributes from spotify using the spotify API

Further explanation of dataset: The Kaggle dataset contains 380,000 songs, and their lyrics. Running a simple python script, and using the Spotify API, we get more variables for each song. We will create subsets of the data by genres similarly. The song lyric data will be joined with the data by song name and will allow us to create a dataframe with all the necessary information.

Spotify assigns almost all songs, a popularity rating between 0-100, and several other audio features that can be analysed. The algorithm which generates the popularity is proprietary, therefore it’s not known how exactly it functions, however, it should allow us to do correlation and regression tests on the data in order to test our hypothesis. These are the two datasets that are provided by spotify.

To classify a particular word as profane we will use a python library called profanity-check([https://github.com/vzhou842/profanity-check (https://github.com/vzhou842/profanity-check)](https://github.com/vzhou842/profanity-check)) to calculate the frequency of profane words. This library uses a pre trained SVM to classify strings as being profane.

Data Cleaning and Pre-processing

Setup

We load the kaggle dataset into a pandas Dataframe from the csv file, and use the spotipy API to get the song details based on the song, and artist. This process took about 30 hours for all 380k songs as the API requests act as a bottleneck

In [70]:

```python
# import libraries
import pandas as pd
import spotipy
from profanity_check import predict, predict_prob
import numpy as np
from spotipy.oauth2 import SpotifyClientCredentials
import matplotlib.pyplot as plt
import patsy
import statsmodels.api as sm
import scipy.stats as stats
from numpy.polynomial.polynomial import polyfit
```

In [3]:

```
# load in raw data
raw_data = pd.read_csv("./dataset/with_norm.csv")
# raw_data.index = raw_data["index"]
# raw_data = raw_data.drop(columns=["index"])

# add popularity column
# raw_data["pop"] = np.zeros(len(raw_data))

raw_data
```

152846	152846	you-and-me	2014	damon-albarn	Pop	I met Moko jumbie,\r\nHe walks on stilts throu...
152847	152847	hollow-ponds	2014	damon-albarn	Pop	Chill on the hollow ponds\r\nSet sail by a kid...
152848	152848	the-selfish-giant	2014	damon-albarn	Pop	Celebrate the passing drugs\r\nPut them on the...
152849	152849	hostiles	2014	damon-albarn	Pop	When the serve is done\r\nAnd the parish shuff...
152850	152850	too-good-to-be-true	2012	edens-edge	Country	You walked in shining brighter than a headligh...
152851	152851	skinny-dippin	2012	edens-edge	Country	I warned you that the tank was low\r\nShoulda ...
152852	152852	cherry-pie	2012	edens-edge	Country	To my first pony, Cherry Pie\r\nFrom the littl...
152853	152853	feels-so-real	2012	edens-	Country	It was two years ago and it was

In [3]:

```
# get popularity data from spotify API

# Setup spotify API
# client_manage = SpotifyClientCredentials(client_id="a47de82ee2f24629ba
# spot = spotipy.Spotify(client_credentials_manager=client_manage)
```

In [4]:
```
# iterate over all songs
# for i in range(359999, len(raw_data)):
    # song = raw_data.iloc[i]["song"]
    # artist = raw_data.iloc[i]["artist"]

    # replace hyphen with spaces
    # song = song.replace("-", " ")
    # artist = artist.replace("-", " ")

    # get popularity ratings - maybe other stuff later on
    # result = spot.search("track: " + song + " artist: " + artist, limi
    # result = result['tracks']['items']

    # go through list, and find song
    # if len(result) != 0:
        # name = result[0]["name"]
        # pop = result[0]["popularity"]
        # raw_data.at[i,"pop"] = pop

    # print
    # print("iteration " + str(i) +"/" + str(len(raw_data)), end="\r")

    # save progress for every song, and keep track of progress
    # if (i % 1000) == 0:
        # write to file
        # raw_data.to_csv("./dataset/with_pop.csv")

# write to file
# raw_data.to_csv("./dataset/with_pop.csv")
```

Cleaning

We first drop all columns with missing data. We dropped any songs without popularity ratings, which we assigned as those with returned Spotify ratings of 0.0. We also dropped songs without lyrics data, as this was imperative to our research question. From here, the range of years was reasonable, with no obvious outliers. Any songs without genres listed were not removed from the dataset as a whole, as these could contribute to the general research topic on if there is a relationship between profanity and popularity. For the more specific analysis regarding any relationship between these two variables within the variable of genre, songs without genres were excluded from this analysis. Finally, to clean the dataset further, any carriage returns (\r) were dropped from the lyrics column, and any preexisting index columns were dropped. Line breaks (\n) were left in the lyrics data, as we were ultimately analyzing the degree of profanity within each line of lyrics in each song of the dataset.

In [30]:
```python
# Drop NaN lyrics
# raw_data.dropna(inplace=True, subset=["lyrics"])

# Drop 0 popularity
# raw_data = raw_data.drop(raw_data[raw_data["pop"] == 0].index)

# drop index columns from original data
# raw_data = raw_data.drop(columns=["index", "Unnamed: 0"])

# clean \rs from all lyrics
def clean_lyrics(lyrics):
    lyrics = lyrics.replace("\r", "")
    return lyrics
# raw_data["lyrics"] = raw_data["lyrics"].apply(clean_lyrics)
raw_data
```

152847	hollow-ponds	2014	albarn	Pop	ponds\nSet sail by a kid\n...	33.0	0
152848	the-selfish-giant	2014	damon-albarn	Pop	Celebrate the passing drugs\nPut them on the b...	40.0	0
152849	hostiles	2014	damon-albarn	Pop	When the serve is done\nAnd the parish shuffle...	39.0	0
152850	too-good-to-be-true	2012	edens-edge	Country	You walked in shining brighter than a headligh...	30.0	0
152851	skinny-dippin	2012	edens-edge	Country	I warned you that the tank was low\nShoulda st...	16.0	0
152852	cherry-pie	2012	edens-edge	Country	To my first pony, Cherry Pie\nFrom the little ...	11.0	0
152853	feels-so-real	2012	edens-edge	Country	It was two years ago and it was yesterday\nIt ...	13.0	0
152854	swingin-door	2012	edens-edge	Country	You've got your Texas way of walking,\nYou've ...	25.0	0
	who-am-i-		edens-		I gotta say\nBoy, after only		

To assign a profanity value to each song we split the lyrics into lines, and count the number of profane lines. Furthermore we normalise the profanity rating by the total number of lines.

In [22]:
```python
# add profanity column

def prof(lyrics):
    lyrics = lyrics.split("\n")
    prof = 0
    prof_arr = np.array(predict(lyrics))
    prof = prof_arr.sum()
    print(prof, end="\r")
    return prof

# raw_data["prof"] = raw_data["lyrics"].apply(prof)
```

```
030
```

In [31]:
```
# raw_data.to_csv("./dataset/with_prof_no_norm.csv")
```

In [36]:
```
# add normalised column
# for i in range(len(raw_data)):
    # prof = raw_data.iloc[i]["prof"]
    # lines = len(raw_data.iloc[i]["lyrics"].split("\n"))
    # raw_data.at[i, "norm"] = prof / lines
    # print(i, end="\r")

```

```
152858
```

After we had computed all these values we saved the new dataset to avoid recomputation.

In [38]:
```
# raw_data.to_csv("./dataset/with_norm.csv")
```

Data Analysis

Post cleaning our data gets cut down to about 158k from 380k. We make a few observations when exploring the data

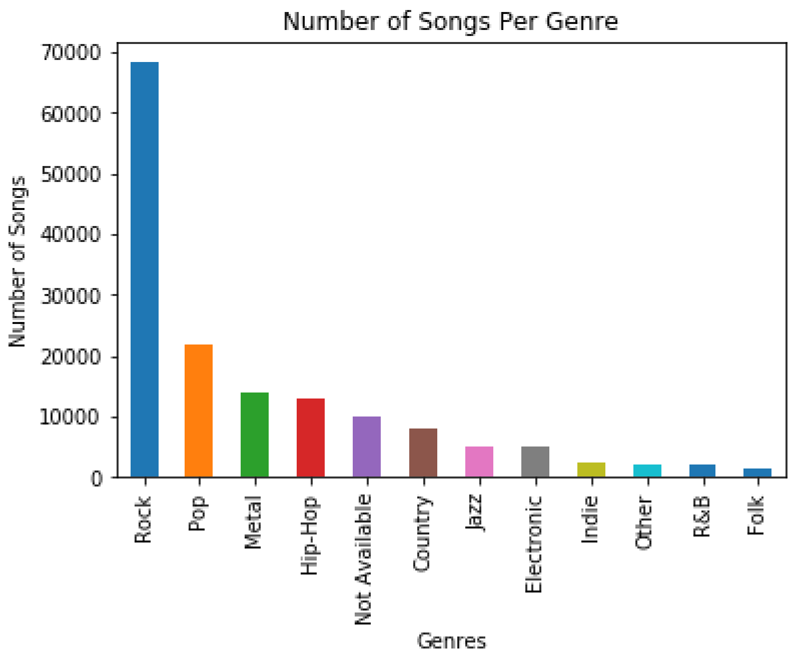
- This dataset is composed primarily of songs classified as rock songs although there are some pop, metal, and hip-hop songs.
- Most of the songs were released between the years 2002 - 2016
- Spotify's popularity rating is biased towards newer songs, and hence the avg popularity ratings are between 1-40 out of a 100
- Most of the songs have 1-15 profane lines

In conclusion the data is not distributed uniformly, and hence any inferences we make cannot be generalised too well. We proceed to analyse the data to find any trends within the dataset, refraining from making any generalising statements.

In [49]:

```python
# summary, and description of data
genres = raw_data["genre"].unique()
print(genres)
print("number of genres: " + str(len(genres)- 1))

ax = raw_data["genre"].value_counts().plot.bar(title="Number of Songs Pe
ax.set_xlabel("Genres")
ax.set_ylabel("Number of Songs")
f1 = plt.gcf()
```

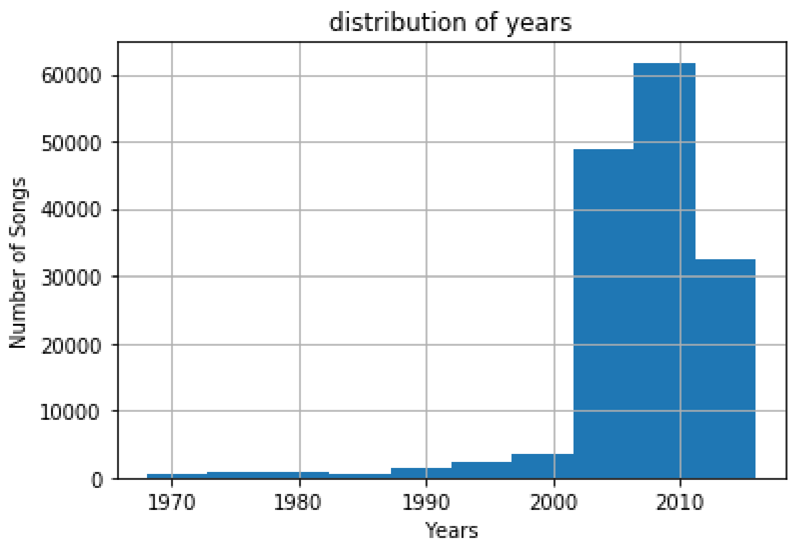
```
['Pop' 'Hip-Hop' 'Rock' 'Metal' 'Not Available' 'Country' 'Jazz'
 'Electronic' 'Folk' 'R&B' 'Indie' 'Other']
number of genres: 11
```



In [52]:

```
# This illustrates that the dataset is primarily composed of data from t
# mid to late 2000's and 2010's up to 2016
years = raw_data["year"].unique()

print(sorted(years))
print("Range : 48 years")

ax = raw_data["year"].hist()
ax.set_title("Distribution of Years")
ax.set_xlabel("Years")
ax.set_ylabel("Number of Songs")
f2 = plt.gcf()

```

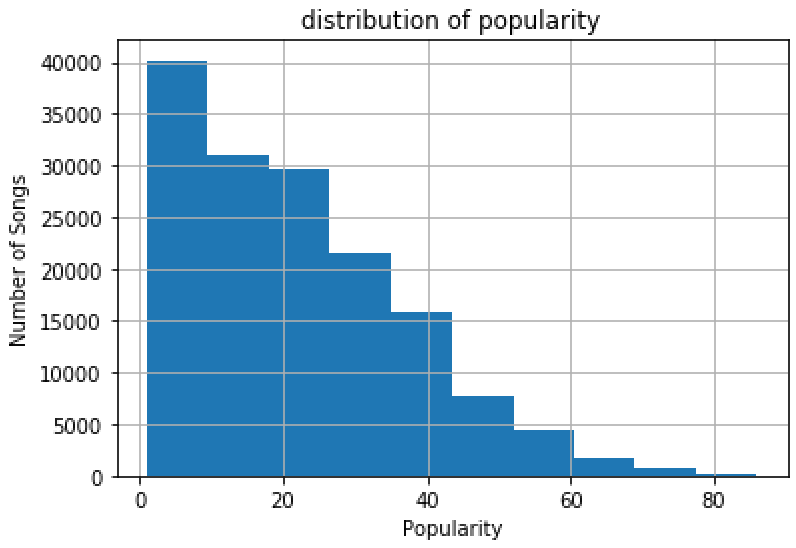
```
[1968, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 19
81, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 200
6, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]
Range : 48 years
```



In [53]:

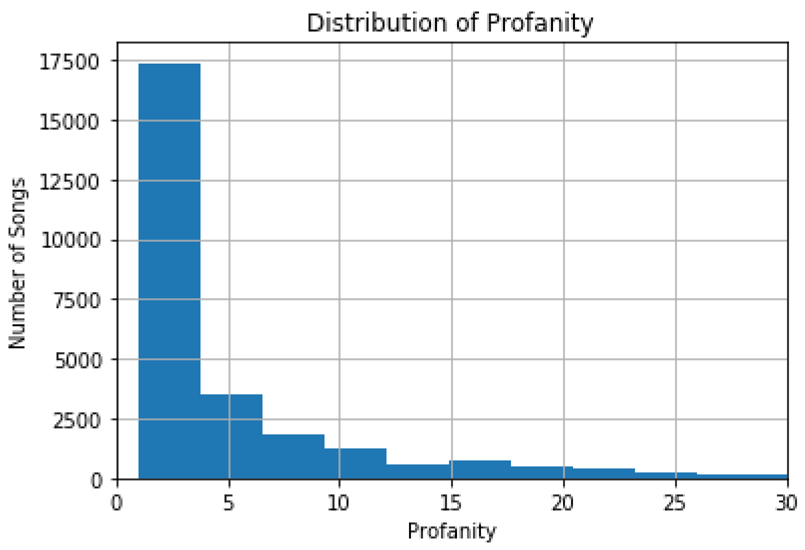
```
# This graph shows that the majority of songs have between a 0-20 popula
# This makes sense considering the popularity algorithm favors newer son
# the dataset is comprised of older music
pop = raw_data["pop"].unique()

print(sorted(pop))
print("Range : 85")

ax = raw_data["pop"].hist()

ax.set_title("Distribution of Popularity")
ax.set_xlabel("Popularity")
ax.set_ylabel("Number of Songs")

f3 = plt.gcf()

```

```
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0,
15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.
0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0,
40.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.
0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0,
65.0, 66.0, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.
0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0]
Range : 85
```



In [54]:
```python
# Our profanity score is based on the number of profane lines present in
# Most songs only have a couple of profane lines

prof = raw_data["prof"].unique()
print(sorted(prof))
print("Range : 140")

with_prof = raw_data[raw_data["prof"] != 0]

ax = with_prof["prof"].hist(bins=50)
plt.xlim(left= 0, right = 30)

ax.set_title("Distribution of Profanity")
ax.set_xlabel("Profanity")
ax.set_ylabel("Number of Songs")
f4 = plt.gcf()

```

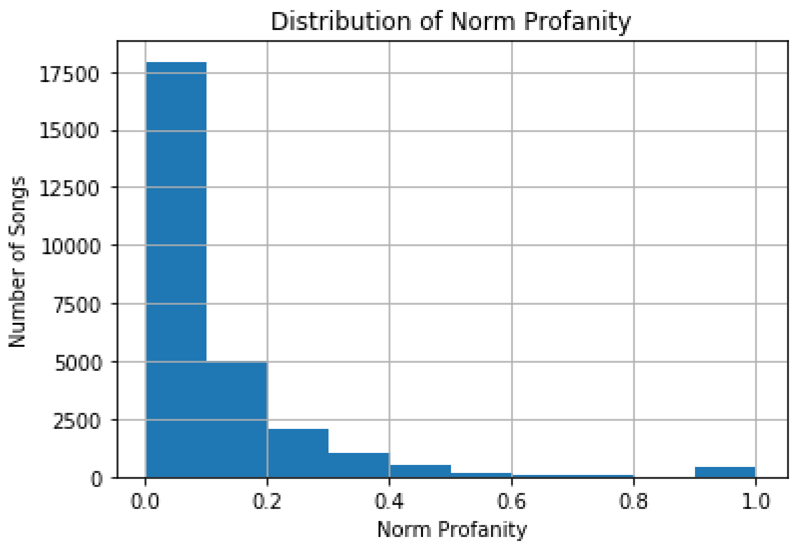
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
59, 61, 62, 63, 64, 67, 68, 76, 80, 83, 87, 92, 93, 99, 140]
Range : 140
```



In [63]:

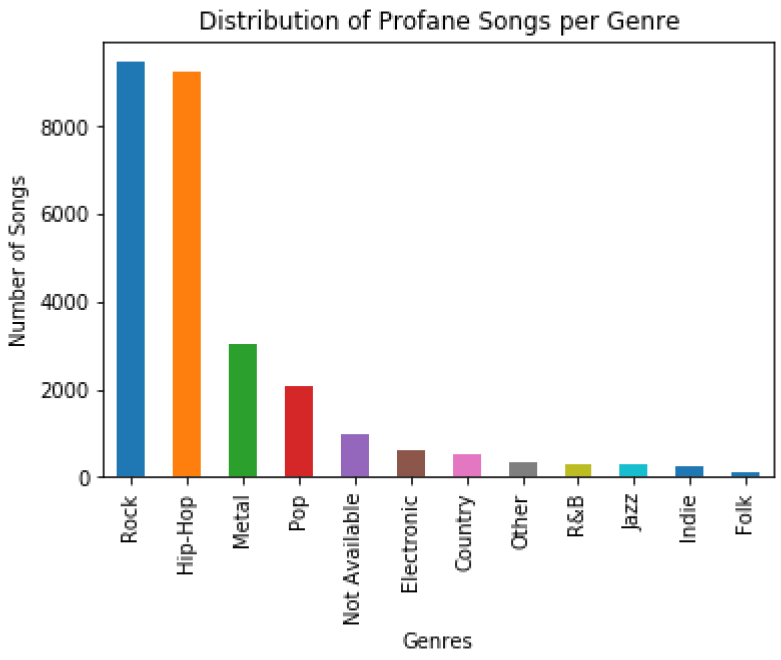
```python
# normalising by the number of total lines seems to only change the dist
# for songs with a lot of profane lines.

norm = with_prof["norm"].unique()
print("no of unique vals " + str(len(norm)))

ax = with_prof["norm"].hist()

ax.set_title("Distribution of Norm Profanity")
ax.set_xlabel("Norm Profanity")
ax.set_ylabel("Number of Songs")
f4 = plt.gcf()
```

```
no of unique vals 1969
```



In [55]:

```python
# if we only consider the profane songs
# we see that Hip hop is disproportionately profane,
# almost matching the over-represented Rock

ax = with_prof["genre"].value_counts().plot.bar()

ax.set_title("Distribution of Profane Songs per Genre")
ax.set_xlabel("Genres")
ax.set_ylabel("Number of Songs")
f5 = plt.gcf()
```



In [12]:

```
# if we normalise by the number of samples we have
# We see that the level of profanity per genre is as below

prof_genre_counts = with_prof["genre"].value_counts()
genre_counts = raw_data["genre"].value_counts()
ratios = {}

for key in prof_genre_counts.keys():
    ratios[key] = prof_genre_counts[key]/genre_counts[key]

sorted(ratios.items(), key=lambda kv: kv[1], reverse=True)
```

Out[12]:

```
[('Hip-Hop', 0.7100627967529484),
 ('Metal', 0.21938408404086918),
 ('Other', 0.16307692307692306),
 ('R&B', 0.15970772442588727),
 ('Rock', 0.13868869748637722),
 ('Electronic', 0.11963309914129586),
 ('Indie', 0.11484716157205241),
 ('Not Available', 0.09770869082466943),
 ('Pop', 0.09433444601947963),
 ('Folk', 0.07148231753197894),
 ('Country', 0.06419967713895443),
 ('Jazz', 0.05714835482008851)]
```

Correlational Analysis

We are interested in observing

- if profane songs are more popular in general
- if profane songs are more popular for a certain genre

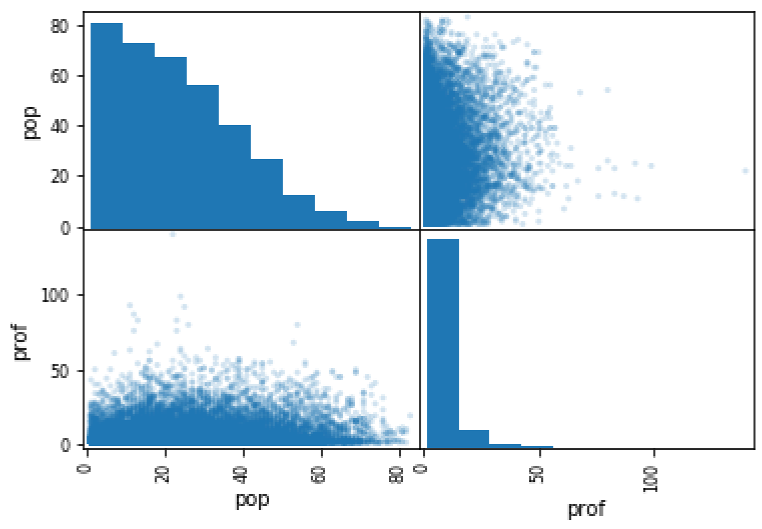
Hence we chose to construct linear models using OLS to analyse these correlations

In [58]:

```
# 1. prof, and pop
f5 = pd.scatter_matrix(with_prof[["pop", "prof"]], alpha=0.2)

outcome_1, predictors_1 = patsy.dmatrices('pop ~ prof', with_prof)
mod_1 = sm.OLS(outcome_1, predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

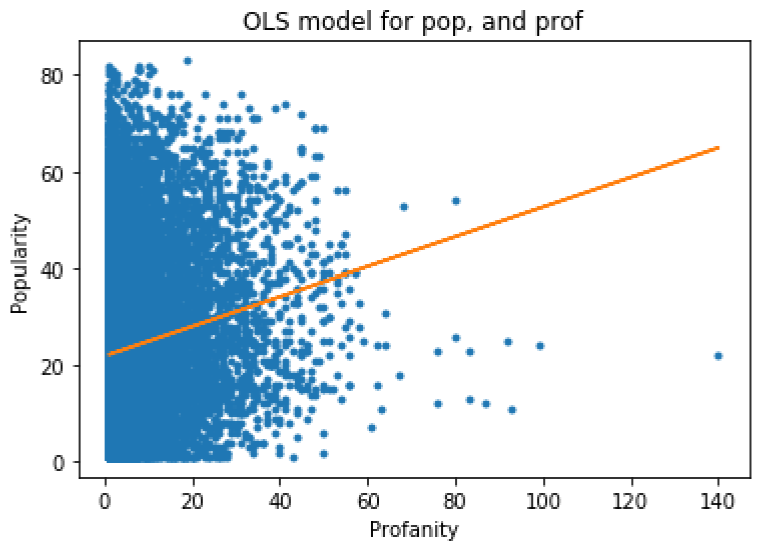
```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: pand
as.scatter_matrix is deprecated, use pandas.plotting.scatter_matrix instead


                            OLS Regression Results
==============================================================================
======
Dep. Variable:                    pop   R-squared:
0.022
Model:                            OLS   Adj. R-squared:
0.022
Method:                 Least Squares   F-statistic:
600.9
Date:                Wed, 12 Jun 2019   Prob (F-statistic):          2.8
4e-131
Time:                        17:54:07   Log-Likelihood:            -1.13
48e+05
No. Observations:               27229   AIC:                         2.2
70e+05
Df Residuals:                   27227   BIC:                         2.2
70e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
Intercept     21.9039      0.115    190.050      0.000      21.678
22.130
prof           0.3071      0.013     24.513      0.000       0.283
0.332
==============================================================================
======
Omnibus:                     1840.640   Durbin-Watson:
0.943
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             22
43.647
Skew:                           0.703   Prob(JB):
0.00
Kurtosis:                       3.009   Cond. No.
11.2
==============================================================================
======

Warnings:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is c
orrectly specified.
```



In [77]:

```python
# We plot the regression line on the scatter plot
x = with_prof["prof"]
y = with_prof["pop"]

b, m = polyfit(x,y,1)

plt.plot(x,y, '.')
plt.title("OLS model for pop, and prof")
plt.xlabel("Profanity")
plt.ylabel("Popularity")
plt.plot(x, b + m*x, '-')
plt.show()
```



Results

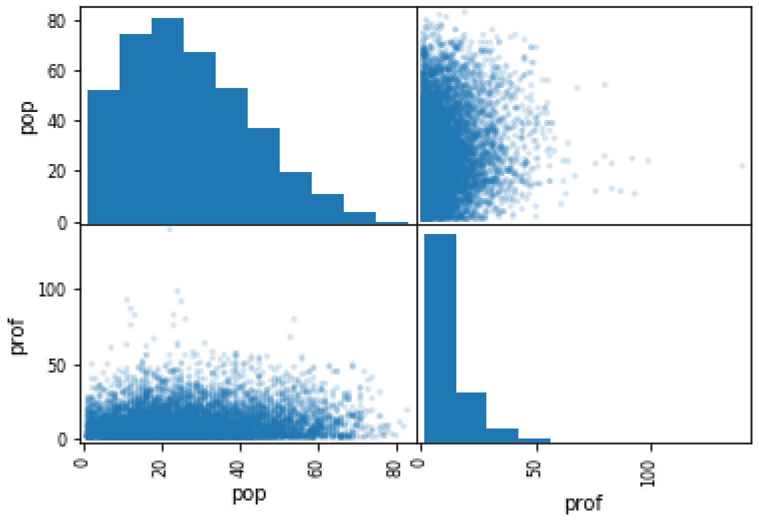
We see a slight positive correlation between profanity, and popularity in general in this dataset.

Next we wanted to see if there is a correlation between prof, and pop within genres. As our profane data is mostly Rock, Hiphop, Metal, and Pop, we limit our analysis to these genres.

In [66]:

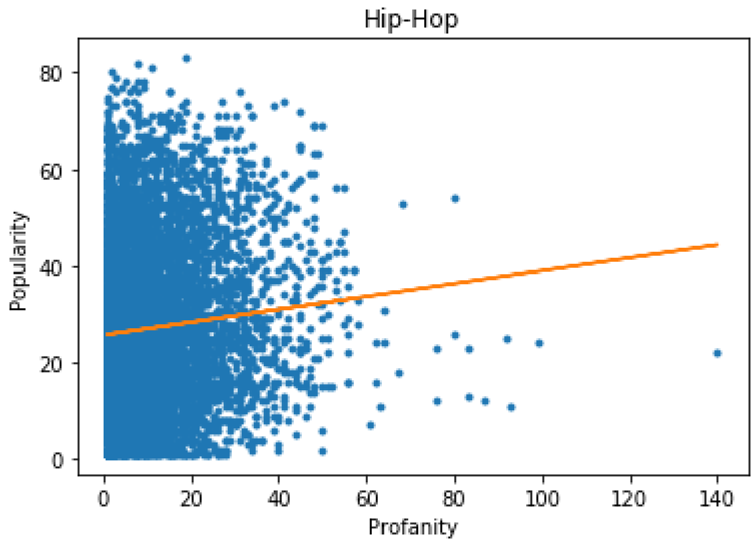
```
# Hip Hop
hiphop = with_prof[with_prof["genre"] == "Hip-Hop"]

f6 = pd.scatter_matrix(hiphop[["pop", "prof"]], alpha=0.2)

outcome_2, predictors_2 = patsy.dmatrices('pop ~ prof',hiphop)
mod_2 = sm.OLS(outcome_2, predictors_2)
res_2 = mod_2.fit()
print(res_2.summary())
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: pand
as.scatter_matrix is deprecated, use pandas.plotting.scatter_matrix instead
  after removing the cwd from sys.path.

                            OLS Regression Results                            
===========================================================================
===
Dep. Variable:                    pop   R-squared:                       0.
007
Model:                            OLS   Adj. R-squared:                  0.
007
Method:                 Least Squares   F-statistic:                     6
8.41
Date:                Wed, 12 Jun 2019   Prob (F-statistic):           1.51e
-16
Time:                        18:52:33   Log-Likelihood:               -387
50.
No. Observations:                9272   AIC:                         7.750e
+04
Df Residuals:                    9270   BIC:                         7.752e
+04
Df Model:                           1                                      
Covariance Type:            nonrobust                                      
===========================================================================
===
                 coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------
---
Intercept     25.7142      0.232    111.036      0.000      25.260      26.
168
prof           0.1329      0.016      8.271      0.000       0.101       0.
164
===========================================================================
===
Omnibus:                      473.123   Durbin-Watson:                   0.
959
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              540.
250
Skew:                           0.583   Prob(JB):                    4.85e-
118
Kurtosis:                       2.807   Cond. No.                        2
0.4
===========================================================================
===
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corr
ectly specified.
```



In [78]:

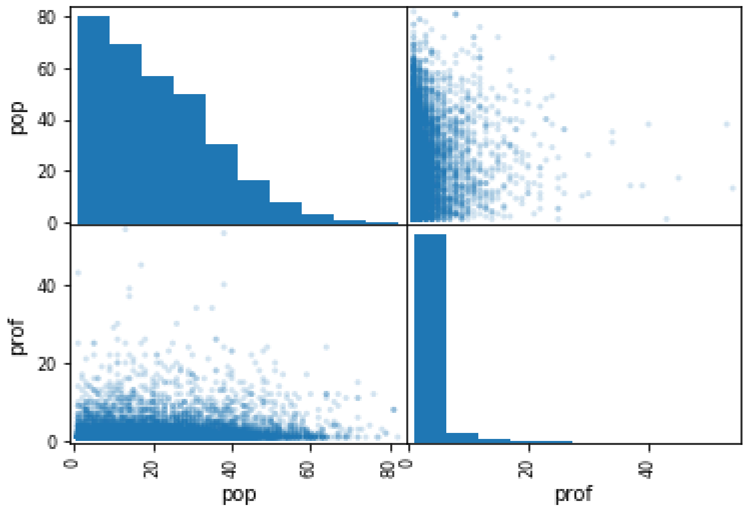
```python
# We plot the regression line on the scatter plot
x = hiphop["prof"]
y = hiphop["pop"]

b, m = polyfit(x,y,1)

plt.plot(x,y, '.')
plt.plot(x, b + m*x, '-')
plt.title("Hip-Hop")
plt.xlabel("Profanity")
plt.ylabel("Popularity")
plt.show()
```



In [67]:

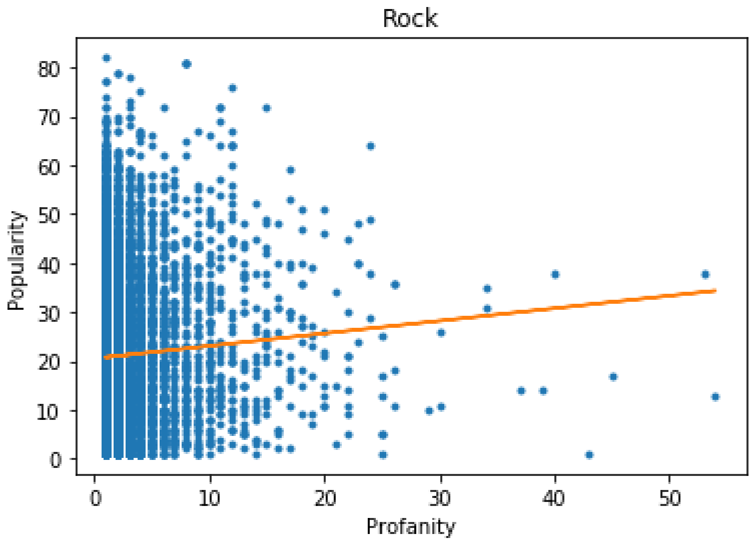
```python
# Rock
rock = with_prof[with_prof["genre"] == "Rock"]

f7 = pd.scatter_matrix(rock[["pop", "prof"]], alpha=0.2)

outcome_3, predictors_3 = patsy.dmatrices('pop ~ prof',rock)
mod_3 = sm.OLS(outcome_3, predictors_3)
res_3 = mod_3.fit()
print(res_3.summary())
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: pand
as.scatter_matrix is deprecated, use pandas.plotting.scatter_matrix instead
  after removing the cwd from sys.path.

                            OLS Regression Results
===========================================================================
===
Dep. Variable:                    pop   R-squared:                       0.
003
Model:                            OLS   Adj. R-squared:                  0.
003
Method:                 Least Squares   F-statistic:                     2
8.41
Date:                Wed, 12 Jun 2019   Prob (F-statistic):           1.00e
-07
Time:                        18:54:20   Log-Likelihood:                -386
80.
No. Observations:                9468   AIC:                         7.736e
+04
Df Residuals:                    9466   BIC:                         7.738e
+04
Df Model:                           1
Covariance Type:            nonrobust
===========================================================================
===
                 coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------
---
Intercept     20.5044      0.189    108.606      0.000      20.134      20.
874
prof           0.2568      0.048      5.330      0.000       0.162       0.
351
===========================================================================
===
Omnibus:                      736.443   Durbin-Watson:                   0.
948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              918.
815
Skew:                           0.758   Prob(JB):                    3.03e-
200
Kurtosis:                       3.179   Cond. No.
5.13
===========================================================================
===
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corr
ectly specified.
```



In [79]:

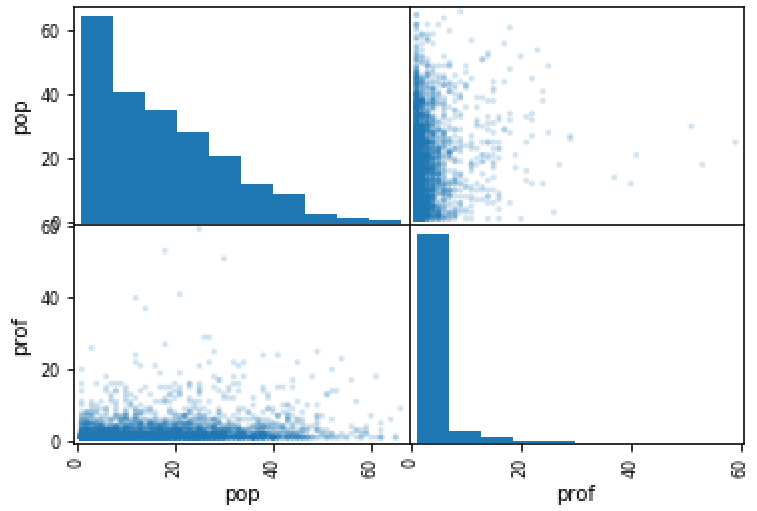
```python
# We plot the regression line on the scatter plot
x = rock["prof"]
y = rock["pop"]

b, m = polyfit(x,y,1)

plt.plot(x,y, '.')
plt.plot(x, b + m*x, '-')
plt.title("Rock")
plt.xlabel("Profanity")
plt.ylabel("Popularity")
plt.show()
```



In [68]:

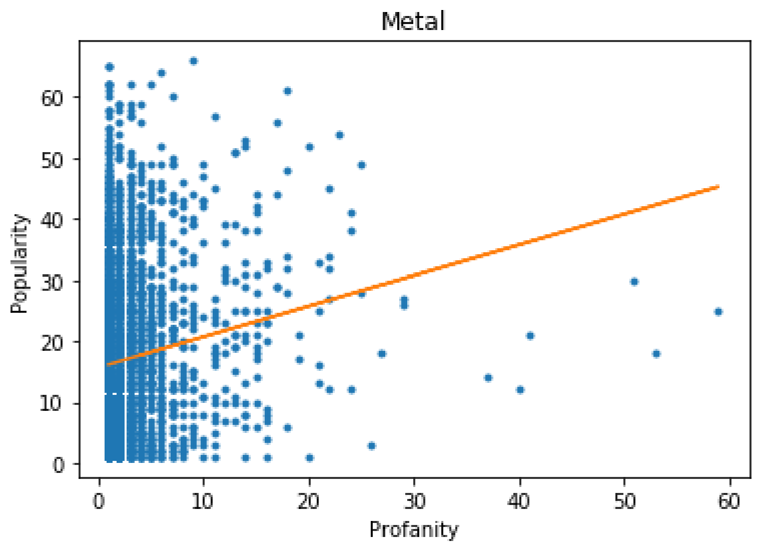
```
# Metal
metal = with_prof[with_prof["genre"] == "Metal"]

f8 = pd.scatter_matrix(metal[["pop", "prof"]], alpha=0.2)

outcome_4, predictors_4 = patsy.dmatrices('pop ~ prof',metal)
mod_4 = sm.OLS(outcome_4, predictors_4)
res_4 = mod_4.fit()
print(res_4.summary())
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: pand
as.scatter_matrix is deprecated, use pandas.plotting.scatter_matrix instead
  after removing the cwd from sys.path.

                            OLS Regression Results
===========================================================================
===
Dep. Variable:                    pop   R-squared:                       0.
021
Model:                            OLS   Adj. R-squared:                  0.
021
Method:                 Least Squares   F-statistic:                     6
6.48
Date:                Wed, 12 Jun 2019   Prob (F-statistic):           5.10e
-16
Time:                        18:55:23   Log-Likelihood:               -121
23.
No. Observations:                3049   AIC:                         2.425e
+04
Df Residuals:                    3047   BIC:                         2.426e
+04
Df Model:                           1
Covariance Type:            nonrobust
===========================================================================
===
                 coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------
---
Intercept     15.6704      0.291     53.916      0.000      15.100      16.
240
prof           0.5018      0.062      8.154      0.000       0.381       0.
622
===========================================================================
===
Omnibus:                      300.548   Durbin-Watson:                   0.
656
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              393.
710
Skew:                           0.869   Prob(JB):                     3.21e
-86
Kurtosis:                       3.274   Cond. No.
5.97
===========================================================================
===
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corr
ectly specified.
```



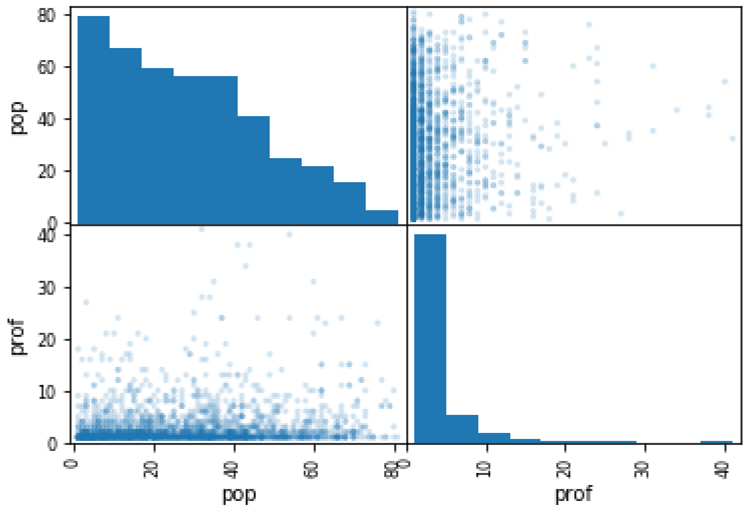
In [80]:

```python
# We plot the regression line on the scatter plot
x = metal["prof"]
y = metal["pop"]

b, m = polyfit(x,y,1)

plt.plot(x,y, '.')
plt.plot(x, b + m*x, '-')
plt.title("Metal")
plt.xlabel("Profanity")
plt.ylabel("Popularity")
plt.show()
```



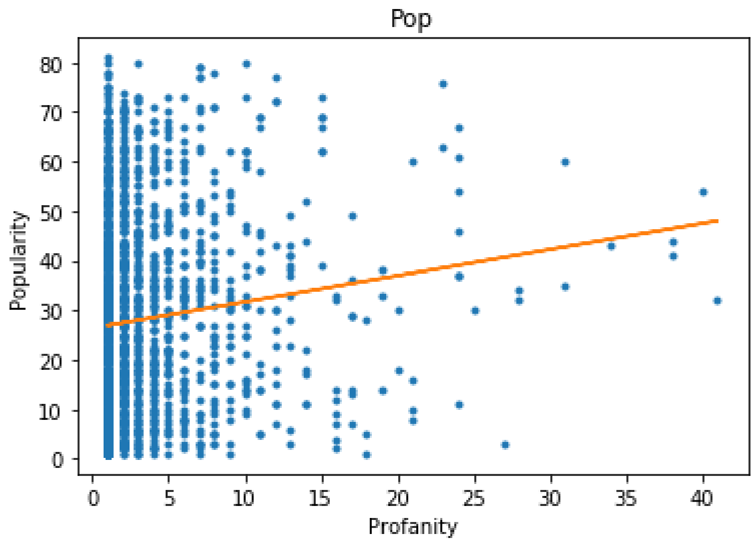
In [69]:
```python
# Pop
pop = with_prof[with_prof["genre"] == "Pop"]

f9 = pd.scatter_matrix(pop[["pop", "prof"]], alpha=0.2)

outcome_5, predictors_5 = patsy.dmatrices('pop ~ prof',pop)
mod_5 = sm.OLS(outcome_5, predictors_5)
res_5 = mod_5.fit()
print(res_5.summary())
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: pand
as.scatter_matrix is deprecated, use pandas.plotting.scatter_matrix instead
  after removing the cwd from sys.path.

                            OLS Regression Results
===========================================================================
===
Dep. Variable:                    pop   R-squared:                       0.
013
Model:                            OLS   Adj. R-squared:                  0.
012
Method:                 Least Squares   F-statistic:                      2
6.55
Date:                Wed, 12 Jun 2019   Prob (F-statistic):           2.81e
-07
Time:                        18:56:16   Log-Likelihood:                -899
6.3
No. Observations:                2063   AIC:                         1.800e
+04
Df Residuals:                    2061   BIC:                         1.801e
+04
Df Model:                           1
Covariance Type:            nonrobust
===========================================================================
===
                 coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------
---
Intercept     26.4433      0.524     50.462      0.000      25.416      27.
471
prof           0.5256      0.102      5.153      0.000       0.326       0.
726
===========================================================================
===
Omnibus:                      140.269   Durbin-Watson:                   0.
932
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              125.
043
Skew:                           0.535   Prob(JB):                     7.04e
-28
Kurtosis:                       2.442   Cond. No.
6.54
===========================================================================
===
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corr
ectly specified.
```



In [81]:

```python
# We plot the regression line on the scatter plot
x = pop["prof"]
y = pop["pop"]

b, m = polyfit(x,y,1)

plt.plot(x,y, '.')
plt.plot(x, b + m*x, '-')
plt.title("Pop")
plt.xlabel("Profanity")
plt.ylabel("Popularity")
plt.show()
```



Results

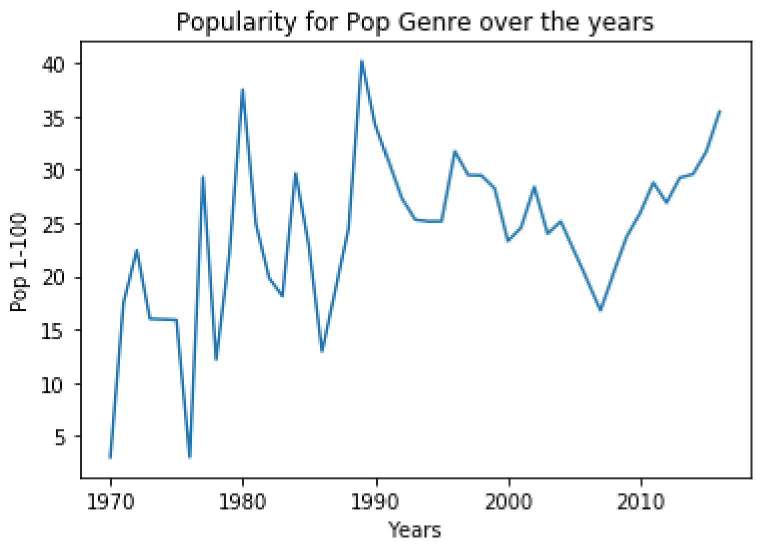
We see maximum positive correlation for Metal, then Pop, Hip-Hop, and the lowest for Rock. This is a counterintuitive result, but can be attributed to profanity being normalised in Hip-Hop leading to a lower effect on popularity.

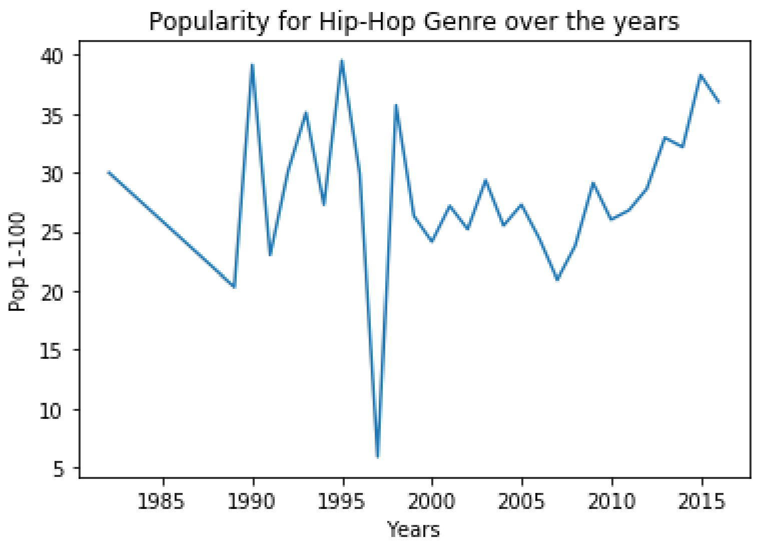
Popular Genres

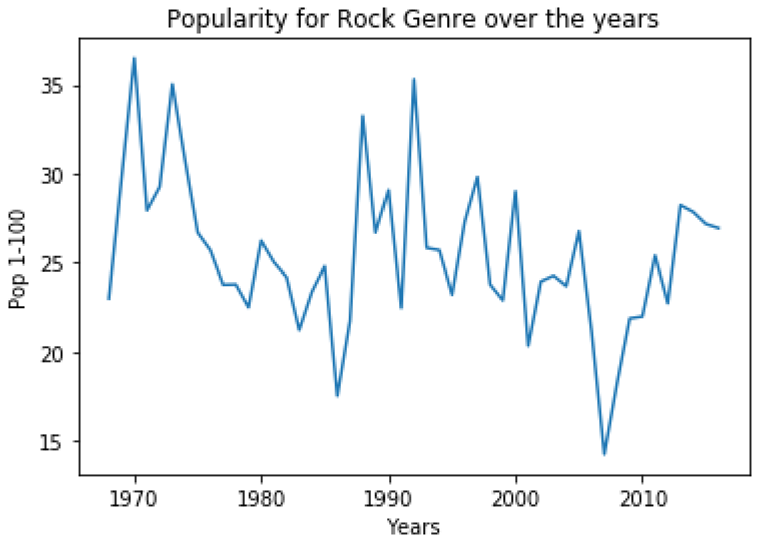
We wanted to see how the popularity of the different genres has changed over the years, so we computed, and graphed the popularity of the songs in the dataset.

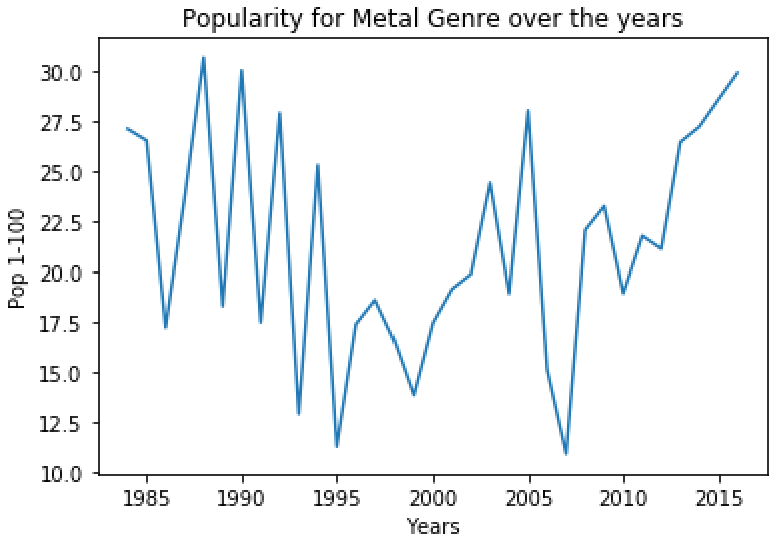
In [65]:

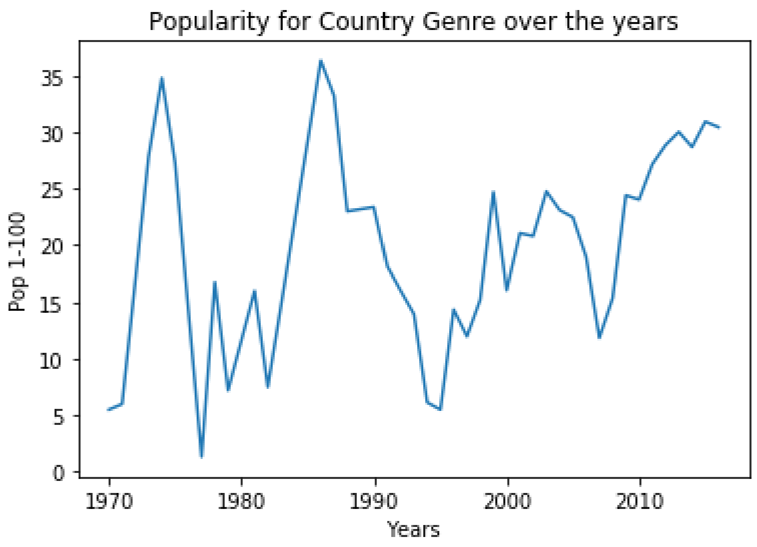
```python
# 2. pop, genre, and year
year_vals = sorted(raw_data["year"].unique())
genre_vals = raw_data["genre"].unique()
avgs = {}

for genre in genre_vals:
    avgs[genre] = {
        "years": [],
        "avgs": []
    }
    for year in year_vals:
        subset = raw_data[raw_data["year"] == year]
        subset = subset[subset["genre"] == genre]["pop"]
        avg = subset.mean()
        if not np.isnan(avg):
            avgs[genre]["avgs"].append(avg)
            avgs[genre]["years"].append(year)

for genre in genre_vals:
    if genre in ["Other", "Not Available"]:
        continue
    plt.plot(avgs[genre]["years"], avgs[genre]["avgs"])
    plt.title("Popularity for " + genre + " Genre over the years")
    plt.ylabel("Pop 1-100")
    plt.xlabel("Years")
    plt.show()


```

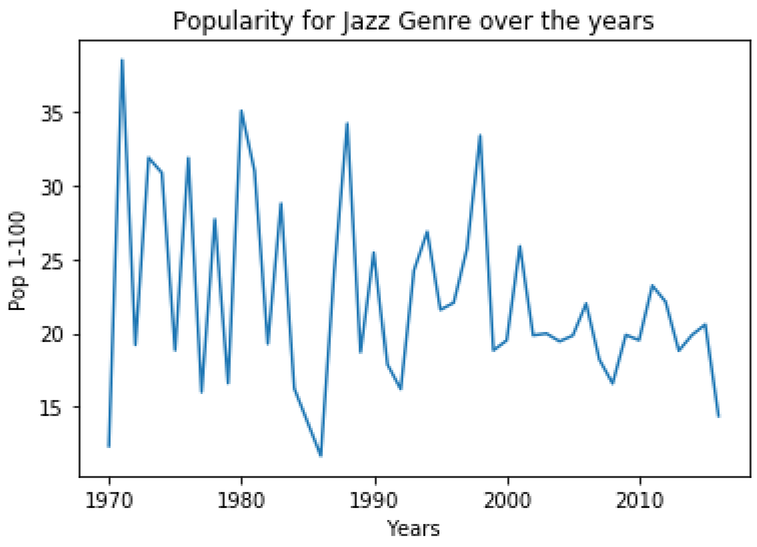


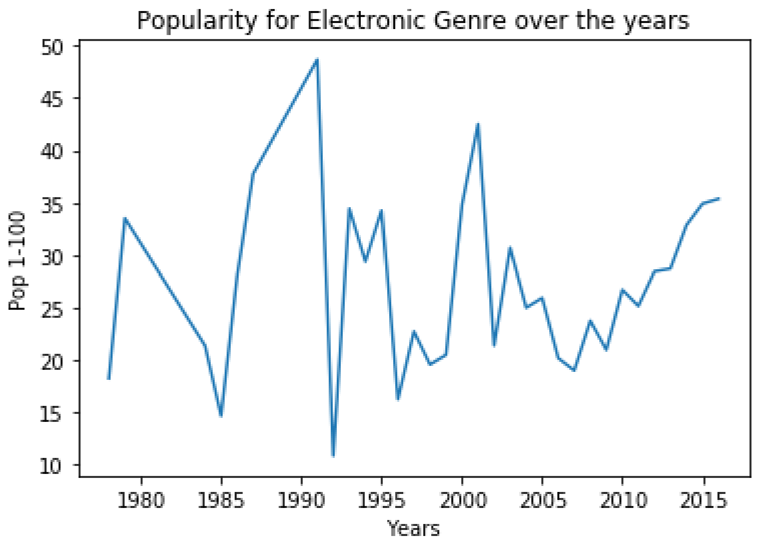

Popularity for Hip-Hop Genre over the years
Pop 1-100
Years
5
10
15
20
25
30
35
40
1985
1990
1995
2000
2005
2010
2015

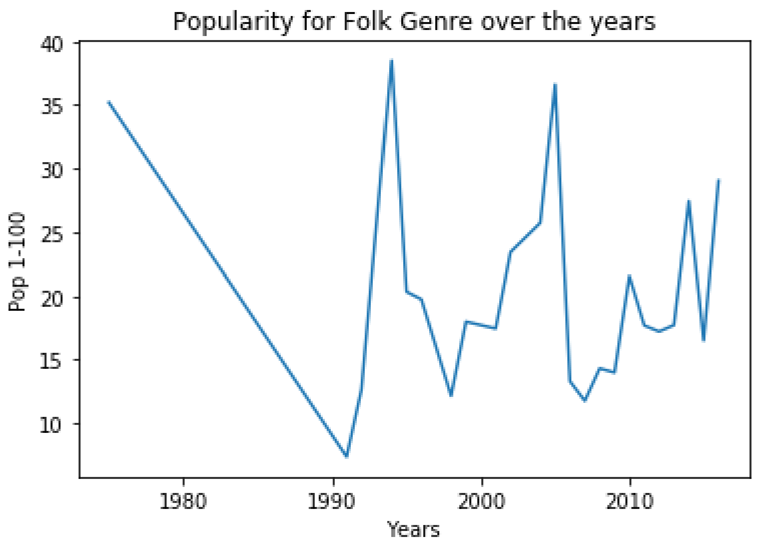

Popularity for Rock Genre over the years
Pop 1-100
Years
15
20
25
30
35
1970
1980
1990
2000
2010

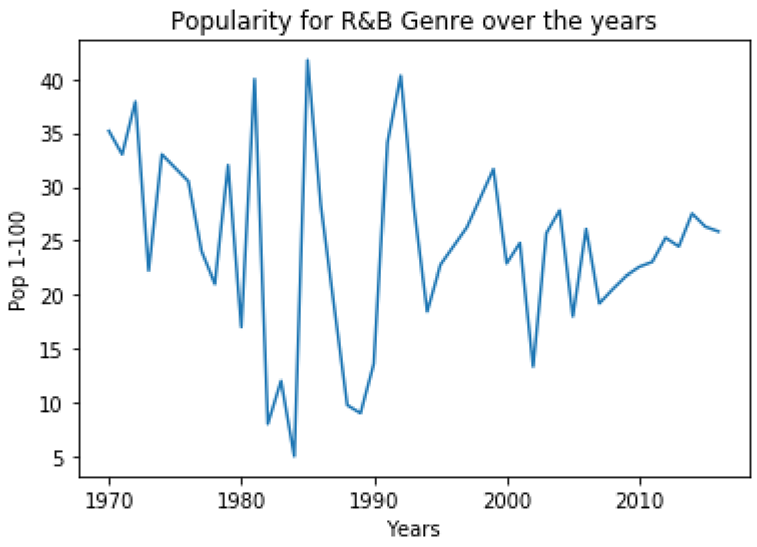

Popularity for Metal Genre over the years
Pop 1-100
Years
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
1985
1990
1995
2000
2005
2010
2015

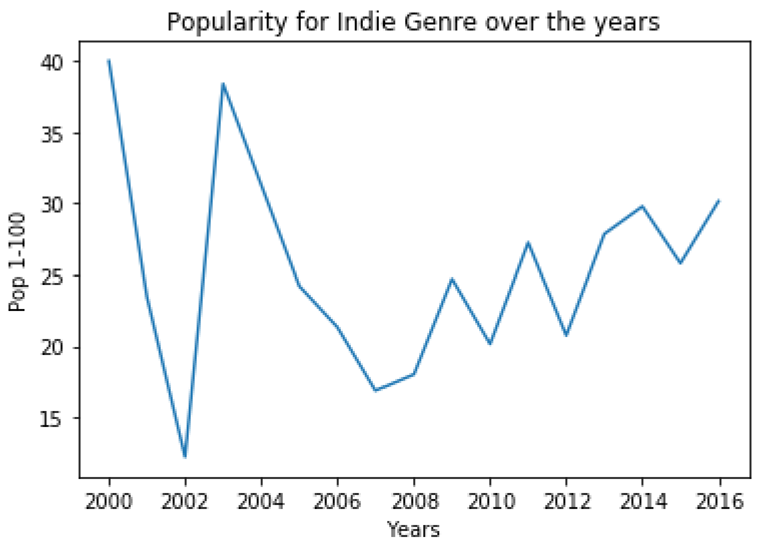

Popularity for Country Genre over the years
Pop 1-100
Years
0
5
10
15
20
25
30
35
1970
1980
1990
2000
2010


Popularity for Jazz Genre over the years
Pop 1-100
Years
15
20
25
30
35
1970
1980
1990
2000
2010


Popularity for Electronic Genre over the years
Pop 1-100
Years
10
15
20
25
30
35
40
45
50
1980
1985
1990
1995
2000
2005
2010
2015







Results

Again, limiting our analysis to our top 4 genres, we see that

- Rock has had a general decline in popularity

- Pop has had a steady rise in popularity
- Metal has had a general decline upto 2015 and then a sudden rise
- Hip-Hop has been very stable, with a gentle rise at the end

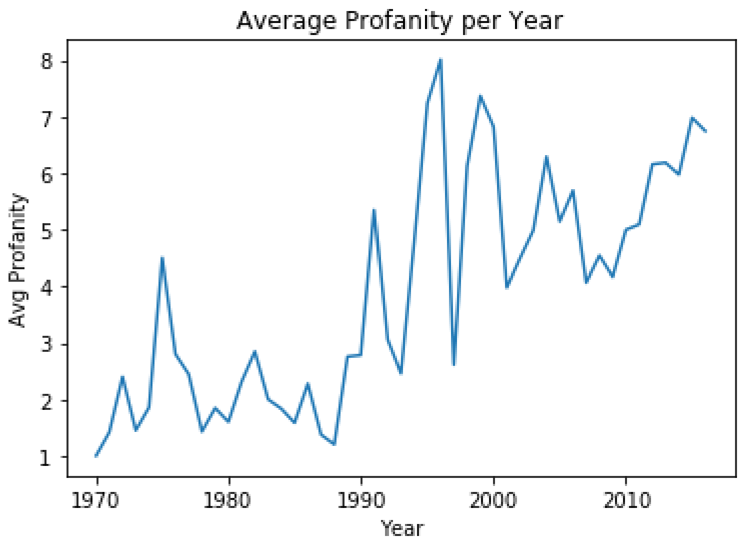
Profanity Over the Years

We wanted to see the trend of profanity in music over the years, so we computed and graphed the average values.

In [64]:

```python
# 3. year and prof
year_vals = with_prof["year"].unique()
year_vals = sorted(year_vals)

avgs = []

for year in year_vals:
    subset = with_prof[with_prof["year"] == year]["prof"]
    avg = subset.mean()
    avgs.append(avg)
    print("Average Profanity for year " + str(year) + " is " + str(avg))

plt.plot(year_vals, avgs)
plt.title("Average Profanity per Year")
plt.xlabel("Year")
plt.ylabel("Avg Profanity")
plt.show()
```

```
Average Profanity for year 1970 is 1.0
Average Profanity for year 1971 is 1.4166666666666667
Average Profanity for year 1972 is 2.4
Average Profanity for year 1973 is 1.45
Average Profanity for year 1974 is 1.8571428571428572
Average Profanity for year 1975 is 4.5
Average Profanity for year 1976 is 2.8
Average Profanity for year 1977 is 2.4444444444444446
Average Profanity for year 1978 is 1.4285714285714286
Average Profanity for year 1979 is 1.8461538461538463
Average Profanity for year 1980 is 1.6
Average Profanity for year 1981 is 2.3125
Average Profanity for year 1982 is 2.8484848484848486
Average Profanity for year 1983 is 2.0
Average Profanity for year 1984 is 1.8333333333333333
Average Profanity for year 1985 is 1.5833333333333333
Average Profanity for year 1986 is 2.2777777777777777
Average Profanity for year 1987 is 1.375
Average Profanity for year 1988 is 1.2
Average Profanity for year 1989 is 2.757575757575758
Average Profanity for year 1990 is 2.783132530120482
Average Profanity for year 1991 is 5.346153846153846
Average Profanity for year 1992 is 3.0579710144927534
Average Profanity for year 1993 is 2.4583333333333335
Average Profanity for year 1994 is 4.775
Average Profanity for year 1995 is 7.240740740740741405
Average Profanity for year 1996 is 8.011904761904763
Average Profanity for year 1997 is 2.6125
Average Profanity for year 1998 is 6.1234567901234565
Average Profanity for year 1999 is 7.367924528301887
Average Profanity for year 2000 is 6.818897637795276
Average Profanity for year 2001 is 3.979591836734694
Average Profanity for year 2002 is 4.493087557603687
Average Profanity for year 2003 is 4.985294117647059
Average Profanity for year 2004 is 6.290909090909091
Average Profanity for year 2005 is 5.143589743589744
Average Profanity for year 2006 is 5.690816735730517
Average Profanity for year 2007 is 4.066305818673883
Average Profanity for year 2008 is 4.540677966101695
Average Profanity for year 2009 is 4.167334669338677
Average Profanity for year 2010 is 4.995387453874539
Average Profanity for year 2011 is 5.090471607314726
Average Profanity for year 2012 is 6.155268022181146
Average Profanity for year 2013 is 6.184753363228699
Average Profanity for year 2014 is 5.9732016925246825
Average Profanity for year 2015 is 6.980769230769231
Average Profanity for year 2016 is 6.745044429254955
```



Results

There's a general rise in profanity in music over the years. This might be as we have more data for more recent years, but in this dataset the trend is very obvious.

Privacy/Ethics Considerations

In terms of ethics and privacy, we did have permission to use this data. All the information, while technically personal, given that it references specific artists, their songs, and the lyrics of those songs, is publically available and is not sensitive information. We concluded that as this information is publically available it does not violate any privacy concerns. However, as we are using a profanity rating to attach some sort of value to our dataset, we acknowledge that as this is not an exact science, in terms of classification, we cannot use any specific datapoint as an example of profanity as we would not want to falsely slander by classifying a clean song as possessing strong elements of profanity. We handle this by not including any one specific point in a publishing of our results.

This also is likely to exclude certain time periods of music as it is only a collection of about 150 thousand songs and this is not inclusive of all songs, as there are obviously more than a hundred and fifty thousand songs. However, we believe that this data is sufficient enough, especially within the time period that we select for to give us enough data points to make reasonable conclusions off of. Therefore, although the profanity classification may be an issue, there should be no other ethical concerns with our data, given that it is non-controversial and publicly available.

Conclusion

Our project question was addressing the relationship of the number of profane lines in a song and its popularity. Our hypothesis was that there would be a positive correlation in the rap, hip hop, and pop genres, and that other genres will show little or no correlation. We also hypothesized that profane lyrics would become more prevalent as time passed. We found that there is a statistically significant effect of profanity on popularity within the subset of data that includes profanity. Rock is the genre with the most profane songs according to our analysis (~68,000 songs), but it is also the most overrepresented genre in our dataset post-cleaning. Hip hop, though our dataset includes far less songs of it as compared to rock (~15,000 songs), has nearly the same number of profane songs.

Plotting the linear regression line on the scatter plots showed us that there is a positive correlation for metal, then pop, then hip-hop, and the lowest for rock. We infer that while this is a counterintuitive result, we can speculate that profanity being normalised in Hip-Hop leads to a lower effect on popularity.

We also found that average profanity has been fluctuating but increasing since 1970, which is what we hypothesized. We are able to see that certain time periods had more profane lyrics, such as in the late 1990s and in more recent years.

Discussion

Limitations: One limitation of our project is that our cleaned dataset does not include all songs and every time period of music. It includes about 150,000 songs; however we believe that this data was sufficient because the initial lyrics dataset contained 382,000 songs, so it should be a decently representative sample from songs in Metrolyrics. Another limitation is that we used a python library to determine profanity and although it uses machine learning to determine profanity and thus is more accurate than a simple dictionary with hard-coded words or phrases to search for, it still cannot necessarily detect underlying themes of profanity within songs. Also, Spotify popularity ratings are biased to newer songs, and we had to trust this algorithm. A problem with this is that even though a song is played many times, it does not mean people listened to it because they enjoyed it, and so this algorithm it might not be indicative solely of song popularity. Popularity is likely determined by a combination of different factors, with profanity being but one. It might be valuable in the future to investigate how strongly each factor of music, including profanity, correlates with a song's popularity.

To do a better analysis, we could scrape our own data, randomizing genres, artists, and time periods to get a more representative sample, and judge popularity based on more rigid statistics like records sold, or number of streams. We also need to find a better way to quantify profanity as the current approach is very susceptible to error.

Impact of work on society: Understanding the popularity of music and its relation to profane lyrics is important because of the impactful nature of easily accessible lyrics. Humans are impressionable creatures who explicitly or implicitly imbibe these lyrics and the presence of profanity could be influential to listeners because genres that include profane lyrics generally have underlying themes of violence or sexual intimacy. Additionally, many songs have recurring themes that send similar underlying messages. This data is also important for the creators of the music because understanding consumer patterns can predict future popularity of created songs. Additionally, being able to look at past popularity patterns can help understand how human listening behavior has changed relative to time, events, artists, and different popular genres. These patterns are indicative of the cultural perception towards profanity, which could help guide future research questions on the openness of human society, particularly in English culture towards more historically taboo themes. These patterns may overall demonstrate the acceptability of profanity in different genres at different points in time, rather than simply popularity.