Spotify - Deskriptive Analyse

In [1]:
```
pip install pyspark
```

```
Requirement already satisfied: pyspark in ./lib/python3.8
/site-packages (3.1.1)
Requirement already satisfied: py4j==0.10.9 in ./lib/pyth
on3.8/site-packages (from pyspark) (0.10.9)
Note: you may need to restart the kernel to use updated p
ackages.
```

In [2]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in ./lib/python3.8/
site-packages (1.2.4)
Requirement already satisfied: numpy>=1.16.5 in ./lib/pyt
hon3.8/site-packages (from pandas) (1.20.2)
Requirement already satisfied: pytz>=2017.3 in ./lib/pyth
on3.8/site-packages (from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
./lib/python3.8/site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in ./lib/python3.
8/site-packages (from python-dateutil>=2.7.3->pandas) (1.
16.0)
Note: you may need to restart the kernel to use updated p
ackages.
```

In [3]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in ./lib/python3.8
/site-packages (0.11.1)
Requirement already satisfied: pandas>=0.23 in ./lib/pyth
on3.8/site-packages (from seaborn) (1.2.4)
Requirement already satisfied: numpy>=1.15 in ./lib/pytho
n3.8/site-packages (from seaborn) (1.20.2)
Requirement already satisfied: matplotlib>=2.2 in ./lib/p
ython3.8/site-packages (from seaborn) (3.4.2)
Requirement already satisfied: scipy>=1.0 in ./lib/python
3.8/site-packages (from seaborn) (1.6.3)
Requirement already satisfied: kiwisolver>=1.0.1 in ./lib
/python3.8/site-packages (from matplotlib>=2.2->seaborn)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in ./lib/pyth
on3.8/site-packages (from matplotlib>=2.2->seaborn) (0.10
.0)
Requirement already satisfied: python-dateutil>=2.7 in ./
lib/python3.8/site-packages (from matplotlib>=2.2->seabor
n) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in ./lib/pyt
hon3.8/site-packages (from matplotlib>=2.2->seaborn) (8.2
.0)
Requirement already satisfied: pyparsing>=2.2.1 in ./lib/
python3.8/site-packages (from matplotlib>=2.2->seaborn) (
2.4.7)
Requirement already satisfied: six in ./lib/python3.8/sit
e-packages (from cycler>=0.10->matplotlib>=2.2->seaborn)
(1.16.0)
Requirement already satisfied: pytz>=2017.3 in ./lib/pyth
on3.8/site-packages (from pandas>=0.23->seaborn) (2021.1)
Note: you may need to restart the kernel to use updated p
ackages.
```

In [4]:

```
pip install pyarrow
```

```
Requirement already satisfied: pyarrow in ./lib/python3.8
/site-packages (4.0.0)
Requirement already satisfied: numpy>=1.16.6 in ./lib/pyt
hon3.8/site-packages (from pyarrow) (1.20.2)
Note: you may need to restart the kernel to use updated p
ackages.
```

In [5]:

```
pip install pyspark
```

```
Requirement already satisfied: pyspark in ./lib/python3.8
/site-packages (3.1.1)
Requirement already satisfied: py4j==0.10.9 in ./lib/pyth
on3.8/site-packages (from pyspark) (0.10.9)
Note: you may need to restart the kernel to use updated p
ackages.
```

In [6]:

```
#Import notwendiger Libraries
from pyspark.sql import SQLContext, SparkSession
from pyspark.sql.types import StructType, DateType, Strin
from pyspark.sql.types import *
import numpy as np
import pandas as pd
#from tqdm.notebook import tqdm
import ast
from pyspark.sql.functions import isnan, when, count, col
from pyspark.sql.functions import min, max
from pyspark.sql.functions import split, explode
from pyspark.sql.functions import udf, concat, col, lit
import re
from spark_plot import mpl
from collections import Counter
import pyarrow as pa
from pyspark.sql.functions import col, skewness, kurtosis
import matplotlib.pyplot as plt
import seaborn as sb
from spark_plot import mpl
```

In [7]:

```
#Erstellung einer Spark-Session
spark = SparkSession.builder.appName('bigdata_spark').get
sc = spark.sparkContext
sqlContext = SQLContext(spark.sparkContext)
```

In [8]:

```
#Import des Spotify-Datensatzes aus Hadoop
spotify = sc.textFile('hdfs://localhost:9000/input/Spotif
```

In [9]:

```
#Erstellung von Dataframes und dazugehörigen Datentypen
spotify_schema = StructType()
spotify_schema.add("Rank",IntegerType(),True)
spotify_schema.add("Track",StringType(),True)
spotify_schema.add("Artist",StringType(),True)
spotify_schema.add("Streams",IntegerType(),True)
spotify_schema.add("Link",StringType(),True)
spotify_schema.add("Week",DateType(),True)
spotify_schema.add("Album_Name",StringType(),True)
spotify_schema.add("Duration_MS",IntegerType(),True)
spotify_schema.add("Explicit",StringType(),True)
spotify_schema.add("Track_Number_on_Album",IntegerType(),
spotify_schema.add("Artist_Followers",IntegerType(),True)
spotify_schema.add("Artist_Genres",StringType(),True)

df_spotify = sqlContext.read.options(delimiter=',').schem
```

In [10]:

```
#Splitten des Attributs "Woche" in Year, Month, Day
df_spotify_split_week = df_spotify.withColumn('Year', spl
       .withColumn('Month', split(df_spotify['Week'], '-'
       .withColumn('Day', split(df_spotify['Week'], '-').
df_spotify_split_week.printSchema()
```

```
root
 |-- Rank: integer (nullable = true)
 |-- Track: string (nullable = true)
 |-- Artist: string (nullable = true)
 |-- Streams: integer (nullable = true)
 |-- Link: string (nullable = true)
 |-- Week: date (nullable = true)
 |-- Album_Name: string (nullable = true)
 |-- Duration_MS: integer (nullable = true)
 |-- Explicit: string (nullable = true)
 |-- Track_Number_on_Album: integer (nullable = true)
 |-- Artist_Followers: integer (nullable = true)
 |-- Artist_Genres: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Day: string (nullable = true)
```

In [11]:

```
#Anzahl der Datensätze
print('Anzahl der Datensätze:')
print(df_spotify.count(),'\n')
```

```
Anzahl der Datensätze:
44200
```

In [12]:

```
#Anzahl der Spalten
print('Anzahl der Spalten:')
print(len(df_spotify.columns),'\n')
```

```
Anzahl der Spalten:
12
```

In [13]:

```
#Nullwerte rausfiltern
df_spotify_null = df_spotify_split_week.select([count(whe
print("Anzahl von Nullwerten pro Attribut:",df_spotify_nu
```

```
Anzahl von Nullwerten pro Attribut:
0
Rank                       0
Track                      5
Artist                     5
Streams                    1
Link                       0
Week                       1
Album_Name                 5
Duration_MS               69
Explicit                   0
Track_Number_on_Album     68
Artist_Followers           1
Artist_Genres              0
Year                       1
Month                      1
Day                        1
```

In [14]:

```
#Entfernen der Nullwerte
df_spotify_Nullwerte_entfernt = df_spotify_split_week.na.
df_spotify_ohne_Nullwerte = df_spotify_Nullwerte_entfernt
print("Datensatz ohne Nullwerte",df_spotify_ohne_Nullwert
```

```
Datensatz ohne Nullwerte                          0
Rank                       0
Track                      0
Artist                     0
Streams                    0
Link                       0
Week                       0
Album_Name                 0
Duration_MS                0
Explicit                   0
Track_Number_on_Album      0
Artist_Followers           0
Artist_Genres              0
Year                       0
Month                      0
Day                        0
```

Datenaufbereitung

In [15]:

```
#Duplikate untersuchen
num = df_spotify_Nullwerte_entfernt.count()
uniq = df_spotify_Nullwerte_entfernt.distinct().count()

if num > uniq:
    print('Datensatz hat Duplikate')
else:
    print('Keine Duplikate')
```

```
Keine Duplikate
```

Statistische Analyse

In [17]:

```
#Statistische Analyse
num_cols = ['Rank','Streams','Duration_MS','Artist_Follow
df_spotify_Nullwerte_entfernt.select(num_cols).describe()

def describe_pd(df_spotify_Nullwerte_entfernt, columns, d

    if deciles:
        percentiles = np.array(range(0, 110, 10))
    else:
        percentiles = [25, 50, 75]

    percs = np.transpose([np.percentile(df_spotify_Nullwe
    percs = pd.DataFrame(percs, columns=columns)
    percs['summary'] = [str(p) + '%' for p in percentiles

    spark_describe = df_spotify_Nullwerte_entfernt.descri
    new_df = pd.concat([spark_describe, percs],ignore_ind
    new_df = new_df.round(2)
    return new_df[['summary'] + columns]

print(describe_pd(df_spotify_Nullwerte_entfernt,num_cols)
```

```
+-------+------------------+-----------------+-----------
------+--------------------+
|summary|              Rank|          Streams|      Durat
ion_MS|    Artist_Followers|
+-------+------------------+-----------------+-----------
------+--------------------+
|  count|             44126|            44126|
44126|               44126|
|   mean|100.49048180211213|8353916.343584281|206805.7020
804061| 1.7254916143860 76E7|
| stddev| 57.74704152236701| 5827782.80023996|40591.47245
869757|1.8033383848867502E7|
|    min|                 1|          2525159|
30133|                   9|
|    max|               200|         80764045|
577199|            78967872|
+-------+------------------+-----------------+-----------
------+--------------------+
```

```
  summary                 Rank             Streams        D
uration_MS  \
0   count                44126               44126
44126
1    mean  100.49048180211213  8353916.343584281  206805.
7020804061
2  stddev   57.74704152236701   5827782.80023996  40591.4
7245869757
3     min                   1            2525159
30133
4     max                 200           80764045
577199
5     25%                50.0          4923302.0
181890.0
6     50%               100.0          6292136.5
204346.0
7     75%               151.0          9616605.5
227360.0

       Artist_Followers
0                 44126
1   1.7254916143860 76E7
2  1.8033383848867502E7
3                     9
4              78967872
5             3963838.0
6            10653835.0
7            26792113.0
```

In [18]:

```
#Skewness & Kurtosis
var = 'Rank'
df_spotify_Nullwerte_entfernt.select(skewness(var), kurto

var_1 = 'Streams'
df_spotify_Nullwerte_entfernt.select(skewness(var_1), kur

var_2 = 'Duration_MS'
df_spotify_Nullwerte_entfernt.select(skewness(var_2), kur

var_3 = 'Artist_Followers'
df_spotify_Nullwerte_entfernt.select(skewness(var_3), kur
```

```
+--------------------+-------------------+
|      skewness(Rank)|     kurtosis(Rank)|
+--------------------+-------------------+
|1.291998872474615...|-1.2005666566413042|
+--------------------+-------------------+

+------------------+------------------+
| skewness(Streams)| kurtosis(Streams)|
+------------------+------------------+
|2.9584505289401375|13.308330353045903|
+------------------+------------------+

+---------------------+---------------------+
|skewness(Duration_MS)|kurtosis(Duration_MS)|
+---------------------+---------------------+
|   0.7801356121678907|     2.72880533982092|
+---------------------+---------------------+

+-------------------------+-------------------------+
|skewness(Artist_Followers)|kurtosis(Artist_Followers)|
+-------------------------+-------------------------+
|        1.5857442638682082|         2.464059906891813|
+-------------------------+-------------------------+
```

In [87]:

```
print('Deskriptive Analyse\n')
print('Ziel: In welche Künstler/-innen lohnt es sich aus

#Erstellung einer Tabelle für den späteren Einsatz von Sp
df_spotify_Nullwerte_entfernt.registerTempTable('Spotify'
sqlContext = SQLContext(spark)
```

```
Deskriptive Analyse

Ziel: In welche Künstler/-innen lohnt es sich aus Sicht v
on Musikprodzent/-innen, Konzertveranstalter/-innen und a
nderern Akteur/-innen der Musikindustrie zu investieren?
```

In [42]:
```
print('Frage 1: Aus wie vielen Künstler/-innen, Songs und
df1 = sqlContext.sql('SELECT COUNT (DISTINCT Artist) AS K
print(df1)
#df1.to_csv('/home/bigdata/Dokumente/anzahl_kuenstler_lie
```

```
Frage 1: Aus wie vielen Künstler/-innen, Songs und Musikg
attungen besteht der Datensatz?

   Kuenstler  Songs  Musikgattungen
0        822   3509             661
```

In [43]:
```
print('\nFrage 2: Wie viele Künstler/-innen, Songs und Mu
df2 = sqlContext.sql('SELECT Year AS Jahr, COUNT (DISTINC
print(df2)
#df2.to_csv('/home/bigdata/Dokumente/anzahl_kuenstler_lie
```

```
Frage 2: Wie viele Künstler/-innen, Songs und Musikgattun
gen waren pro Jahr in den Charts?

   Jahr  Kuenstler  Songs  Musikgattungen
0  2017        332    870             295
1  2018        380   1122             328
2  2019        394   1046             334
3  2020        392   1143             328
4  2021        227    418             195
```

In [44]:
```
print('\nFrage 3: Welche Künstler/-innen kommen, gemessen
df3 = sqlContext.sql('SELECT Artist, COUNT (Artist) AS Ha
print(df3.head(21))
#df3.to_csv('/home/bigdata/Dokumente/anzahl_kuenstler_cha
```

```
Frage 3: Welche Künstler/-innen kommen, gemessen an der A
nzahl der Wochen, am häufigsten in den Top 200 Charts vor
?
                Artist  Haeufigkeit
0          Ed Sheeran         1320
1         Post Malone         1296
2        XXXTENTACION          948
3               Drake          910
4       Billie Eilish          907
5       Ariana Grande          718
6    The Chainsmokers          657
7          Juice WRLD          651
8        Travis Scott          597
9            Bad Bunny          560
10           J Balvin          551
11    Imagine Dragons          516
12         The Weeknd          488
13              Ozuna          467
14       Shawn Mendes          466
15              Khalid          462
16           Dua Lipa          452
17              Queen          377
18         Marshmello          364
19       Taylor Swift          357
20          Sam Smith          347
```

In [47]:

```
print('\nFrage 4: Welche Künstler/-innen hatten zwischen
df4 = sqlContext.sql('SELECT Artist AS Kuenstler, MAX(Art
print(df4.head(21))
#df4.to_csv('/home/bigdata/Dokumente/anzahl_follower_pro_
```

```
Frage 4: Welche Künstler/-innen hatten zwischen 2017 und
2021 die meisten Follower?
          Kuenstler  Follower
0        Ed Sheeran  78967872
1     Ariana Grande  61390454
2             Drake  54405324
3     Justin Bieber  44672226
4            Eminem  43728461
5           Rihanna  42275572
6     Billie Eilish  41852475
7      Taylor Swift  38902703
8   Imagine Dragons  33692509
9             Queen  33527593
10     Shawn Mendes  32473361
11        Bad Bunny  32304467
12      Post Malone  32192178
13              BTS  31673242
14       The Weeknd  31348348
15         Maroon 5  30323494
16       Marshmello  30270515
17       Bruno Mars  29942000
18         Coldplay  29776593
19      Alan Walker  28057563
20            Ozuna  27797552
```

In [96]:

```
print('\nFrage 5: Welche Künstler/-innen erzielten die me
df56 = sqlContext.sql('SELECT Artist AS Kuenstler, MAX(St
print(df56.head(21))
#df56.to_csv('/home/bigdata/Dokumente/anzahl_streams_pro_
```

```
Frage 5: Welche Künstler/-innen erzielten die meisten Str
eams pro Track bezogen auf die einzelnen Jahre?
         Kuenstler   Streams                                
Song  Jahr
0   Olivia Rodrigo  80764045                                dr
ivers license  2021
1    Ariana Grande  71467874                                
7 rings  2019
2            Drake  67499798                                 I
n My Feelings  2018
3     Shawn Mendes  67237638                                
Señorita  2019
4       Ed Sheeran  64275251                                
Shape of You  2017
5    Ariana Grande  59975503                                
thank u, next  2018
6       Ed Sheeran  58370367          I Don't Care (with J
ustin Bieber)  2019
7        Lil Nas X  55582612             MONTERO (Call Me
By Your Name)  2021
8            Drake  54891573                                
God's Plan  2018
9       Luis Fonsi  54848635                            Desp
acito - Remix  2017
10   Ariana Grande  54707620  break up with your girlfrie
nd, i'm bored  2019
11    Mariah Carey  53401383           All I Want for Chr
istmas Is You  2020
12       Bad Bunny  53344093                                
DÁKITI  2020
13      The Weeknd  52375259                                Bl
inding Lights  2020
14   Justin Bieber  52357127    Peaches (feat. Daniel Cae
sar & Giveon)  2021
15     Tones And I  52055226                                
Dance Monkey  2020
16   Billie Eilish  50342324                                
bad guy  2019
17     Roddy Ricch  48937592                                
The Box  2020
18      THE SCOTTS  48430814                                
THE SCOTTS  2020
19    Travis Scott  48161108                            HIGHES
T IN THE ROOM  2019
20           Wham!  48011162                                 L
ast Christmas  2020
```

In [93]:

```
print('\nFrage 6: Welche Künstler/-innen waren am häufigs
df7 = sqlContext.sql('SELECT Artist AS Kuenstler, COUNT(A
print(df7.head(21))
#df7.to_csv('/home/bigdata/Dokumente/kuenstler_am_laengst
```

```
Frage 6: Welche Künstler/-innen waren am häufigsten, geme
ssen an der Anzahl der Wochen, auf den ersten 10 Plätzen
der Charts?
          Kuenstler  Wochen  Rank
0       Post Malone      26     1
1        The Weeknd      23     3
2             Drake      22     1
3     Ariana Grande      20     1
4        Ed Sheeran      20     1
5       Post Malone      18     2
6       Post Malone      18     5
7       Tones And I      17     1
8          Dua Lipa      17     4
9             Drake      16     2
10      Post Malone      16     4
11       The Weeknd      15     1
12      Post Malone      15     3
13     Shawn Mendes      14     1
14   Camila Cabello      12     2
15       Luis Fonsi      12     1
16       The Weeknd      12     6
17       Ed Sheeran      12     5
18       Marshmello      11     9
19      Post Malone      11     6
20    Ariana Grande      11     8
```

In [95]:

```
print('\nFrage 7: Welche Songs hatten die meisten Streams
df5 = sqlContext.sql('SELECT Track AS Song, MAX(Streams)
print(df5.head(21))
#df5.to_csv('/home/bigdata/Dokumente/streams_songs.csv')
```

```
Frage 7: Welche Songs hatten die meisten Streams?
                                                Song   Streams
0                                   drivers license  80764045
1                                             7 rings  71467874
2                                    In My Feelings  67499798
3                                          Señorita  67237638
4                                      Shape of You  64275251
5                                    thank u, next  59975503
6          I Don't Care (with Justin Bieber)  58370367
7             MONTERO (Call Me By Your Name)  55582612
8                                       God's Plan  54891573
9                                Despacito - Remix  54848635
10  break up with your girlfriend, i'm bored  54707620
11           All I Want for Christmas Is You  53401383
12                                        DÁKITI  53344093
13                                Blinding Lights  52375259
14    Peaches (feat. Daniel Caesar & Giveon)  52357127
15                                   Dance Monkey  52055226
16                                        bad guy  50342324
17                                        The Box  48937592
18                                    THE SCOTTS  48430814
19                           HIGHEST IN THE ROOM  48161108
20                                Last Christmas  48011162
```

In [97]:

```
print('\nFrage 8: Welche Musikgattung hatte die meisten S
df12 = sqlContext.sql('SELECT distinct Artist_Genres AS M
print(df12.head(21))
#df12.to_csv('/home/bigdata/Dokumente/streams_genre.csv')
```

```
Frage 8: Welche Musikgattung hatte die meisten Streams?
                                          Musikgattung   S
treams
0                              ['pop', 'post-teen pop']  80
764045
1                              ['pop', 'post-teen pop']  71
467874
2   ['canadian hip hop', 'canadian pop', 'hip hop'...  67
499798
3   ['canadian pop', 'dance pop', 'pop', 'post-tee...  67
237638
4   ['canadian pop', 'dance pop', 'pop', 'post-tee...  66
933317
5                              ['pop', 'post-teen pop']  65
873080
6   ['canadian hip hop', 'canadian pop', 'hip hop'...  65
825491
7   ['canadian pop', 'dance pop', 'pop', 'post-tee...  64
942021
8                              ['pop', 'post-teen pop']  64
681075
9                                    ['pop', 'uk pop']  64
275251
10                             ['pop', 'post-teen pop']  63
197614
11  ['canadian pop', 'dance pop', 'pop', 'post-tee...  63
045599
12  ['canadian pop', 'dance pop', 'pop', 'post-tee...  61
224745
13  ['canadian hip hop', 'canadian pop', 'hip hop'...  60
285459
14                             ['pop', 'post-teen pop']  59
975503
15                             ['pop', 'post-teen pop']  59
324475
16                                   ['pop', 'uk pop']  58
370367
17  ['canadian pop', 'dance pop', 'pop', 'post-tee...  57
580873
18                                   ['pop', 'uk pop']  57
006531
19  ['country rap', 'lgbtq+ hip hop', 'pop rap', '...  55
582612
20  ['canadian hip hop', 'canadian pop', 'hip hop'...  54
891573
```

In [98]:

```
print('\nFrage 9: Wie lange dauerten Songs an, die in den
df13 = df_spotify_Nullwerte_entfernt.groupby('Rank','Trac
print(df13.head(21))
#df13.to_csv('/home/bigdata/Dokumente/durchschnittliche_l
```

```
Frage 9: Wie lange dauerten Songs an, die in den Top 10 C
harts waren?
    Rank                                    Track  avg(Dur
ation_MS)
0      1                                 bad guy
194087.0
1      1                     Lose You To Love Me
206458.0
2      1                                  SAD!
166605.0
3      1                          Nice For What
210925.0
4      1                             THE SCOTTS
165977.0
5      1                           Shape of You
233712.0
6      1                        This Is America
225773.0
7      1          Lucky You (feat. Joyner Lucas)
244679.0
8      1                                Nonstop
238613.0
9      1            ROCKSTAR (feat. Roddy Ricch)
181733.0
10     1                                The Box
196652.0
11     1          MONTERO (Call Me By Your Name)
137875.0
12     1               Havana (feat. Young Thug)
217306.0
13     1                              cardigan
239560.0
14     1         WAP (feat. Megan Thee Stallion)
187541.0
15     1                             positions
172324.0
16     1  Peaches (feat. Daniel Caesar & Giveon)
198081.0
17     1                           God's Plan
198960.0
18     1                           Better Now
231266.0
19     1                          thank u, next
207333.0
20     1                Look What You Made Me Do
211859.0
```

In [84]:

```
print('\nFrage 10: Welche Musikgattungen wurden pro Jahre
df9 = sqlContext.sql('SELECT Artist_Genres AS Musikgattun
df10 = df9[['Musikgattung','Monat']].value_counts()
print(df10.head(21))
#df10.to_csv('/home/bigdata/Dokumente/genre_jahreszeiten.
```

```
Frage 9: Welche Musikgattungen wurden pro Jahrezeit am me
isten gehört?
Musikgattung
Monat
['pop', 'uk pop']
03       190
['pop', 'post-teen pop']
12       165

01       154

08       153

11       150
['dfw rap', 'melodic rap', 'rap']
05       149
['pop', 'uk pop']
01       148
['dance pop', 'pop', 'post-teen pop']
03       146
['pop', 'uk pop']
04       145

08       144
['dance pop', 'pop', 'post-teen pop']
01       142
['pop', 'post-teen pop']
03       138
['pop', 'uk pop']
07       137

05       135
['pop', 'post-teen pop']
02       134
['latin', 'reggaeton', 'trap latino']
03       128
['canadian hip hop', 'canadian pop', 'hip hop', 'pop rap'
, 'rap', 'toronto rap']  07       126
['pop', 'uk pop']
02       124
['dfw rap', 'melodic rap', 'rap']
06       122
['dance pop', 'pop', 'post-teen pop']
06       121
['pop', 'uk pop']
09       120
dtype: int64
```

In [30]:

```
print('\nFrage 11: Wieviel Lieder gab es pro Musikgattung
df8 = sqlContext.sql('SELECT Artist_Genres AS Musikgattun
print(df8.head(21))
#df8.to_csv('/home/bigdata/Dokumente/anzahl_lieder_genre.
```

```
Frage 8: Wieviel Lieder gibt es pro Genre?
                                          Musikgattung  So
ng
0                             ['pop', 'post-teen pop']   1
63
1                ['dance pop', 'pop', 'post-teen pop']   1
03
2   ['canadian hip hop', 'canadian pop', 'hip hop'...
80
3                ['latin', 'reggaeton', 'trap latino']
64
4                                    ['german hip hop']
64
5                         ['k-pop', 'k-pop boy group']
62
6                ['detroit hip hop', 'hip hop', 'rap']
51
7         ['melodic rap', 'philly rap', 'rap', 'trap']
50
8                                     ['pop', 'uk pop']
48
9                         ['emo rap', 'miami hip hop']
46
10  ['latin', 'reggaeton', 'reggaeton colombiano',...
46
11                       ['chicago rap', 'melodic rap']
44
12                    ['dfw rap', 'melodic rap', 'rap']
43
13            ['canadian pop', 'pop', 'post-teen pop']
38
14  ['canadian contemporary r&b', 'canadian pop', ...
37
15                       ['atl hip hop', 'rap', 'trap']
36
16  ['dance pop', 'edm', 'electropop', 'pop', 'pop...
35
17    ['dance pop', 'pop', 'post-teen pop', 'uk pop']
35
18  ['conscious hip hop', 'dmv rap', 'hip hop', 'p...
34
19  ['latin', 'latin hip hop', 'reggaeton', 'trap ...
34
20                              ['modern rock', 'rock']
33
```

In [99]:

```
print('\nFrage 12: Welche Künstler/-innen erzielten die m
df6 = sqlContext.sql('SELECT Artist AS Kuenstler, MAX(Str
print(df6.head(21))
#df6.to_csv('/home/bigdata/Dokumente/streams_kuenstler.cs
```

```
Frage 12: Welche Künstler/-innen erzielten die meisten St
reams?
          Kuenstler   Streams
0    Olivia Rodrigo  80764045
1     Ariana Grande  71467874
2             Drake  67499798
3      Shawn Mendes  67237638
4        Ed Sheeran  64275251
5         Lil Nas X  55582612
6        Luis Fonsi  54848635
7      Mariah Carey  53401383
8         Bad Bunny  53344093
9        The Weeknd  52375259
10    Justin Bieber  52357127
11      Tones And I  52055226
12    Billie Eilish  50342324
13      Roddy Ricch  48937592
14       THE SCOTTS  48430814
15     Travis Scott  48161108
16             Wham!  48011162
17     Selena Gomez  47227738
18      Post Malone  46995997
19          Cardi B  45977242
20     Taylor Swift  44480002
```

In [100…

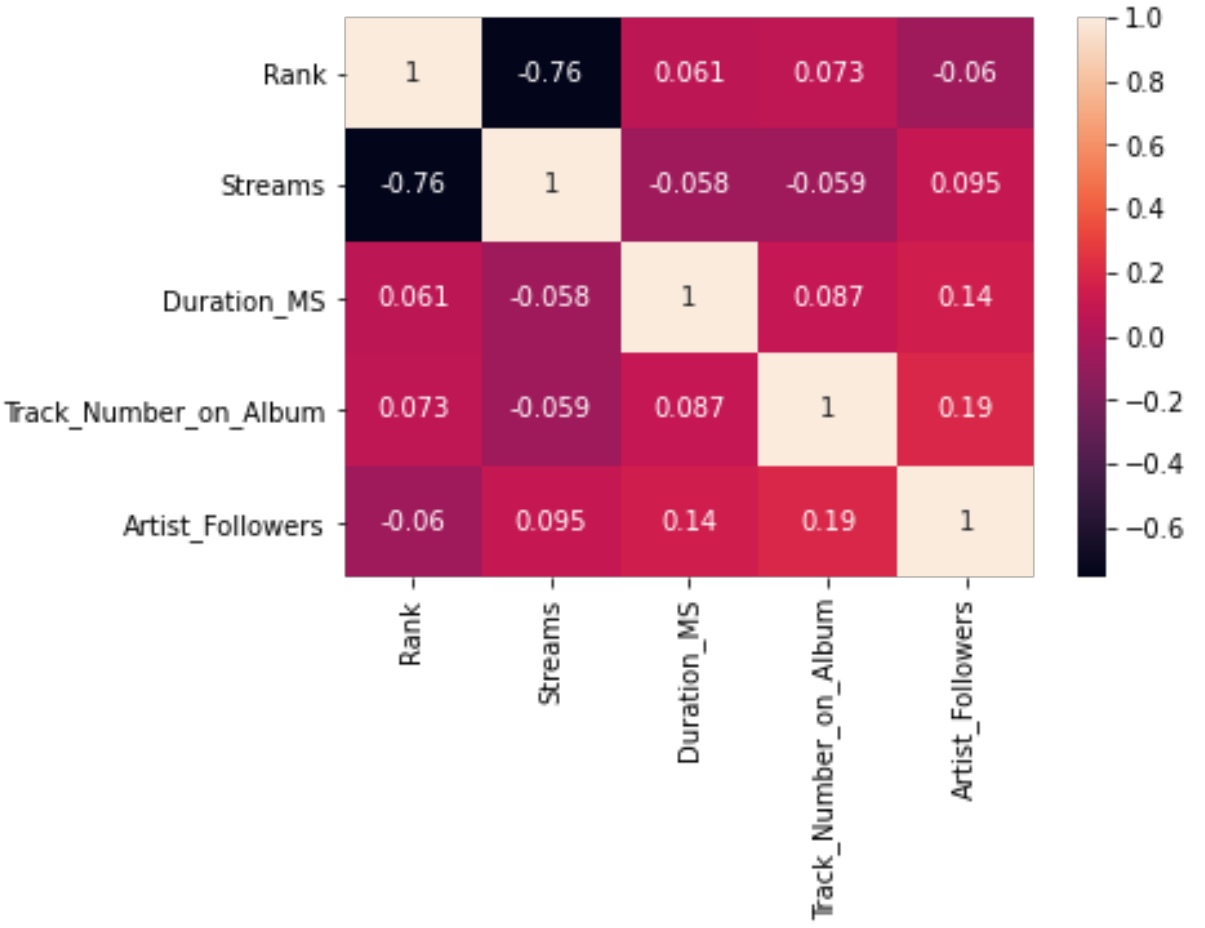
```
pandasDF = df_spotify_Nullwerte_entfernt.toPandas()
print('\nFrage 13: Welche Künstler/-innen sind der Musikg
l= pandasDF[pandasDF['Artist_Genres']=="['pop', 'post-tee
print(l)
#l.to_csv('/home/bigdata/Dokumente/genre_pop_post-teen_po
```

```
Frage 13: Welche Künstler/-innen sind der Musikgattung "P
oP, Pop-Teen pop" zuzuordnen?
Ariana Grande      718
Taylor Swift       357
Harry Styles       329
Louis Tomlinson     32
Olivia Rodrigo      27
Name: Artist, dtype: int64
```

In [38]:

```
#Korrelationsmatrix:
corr_matrix = pandasDF.corr()
corr_matrix
sb.heatmap(corr_matrix,annot=True)
```

Out[38]: <AxesSubplot:>

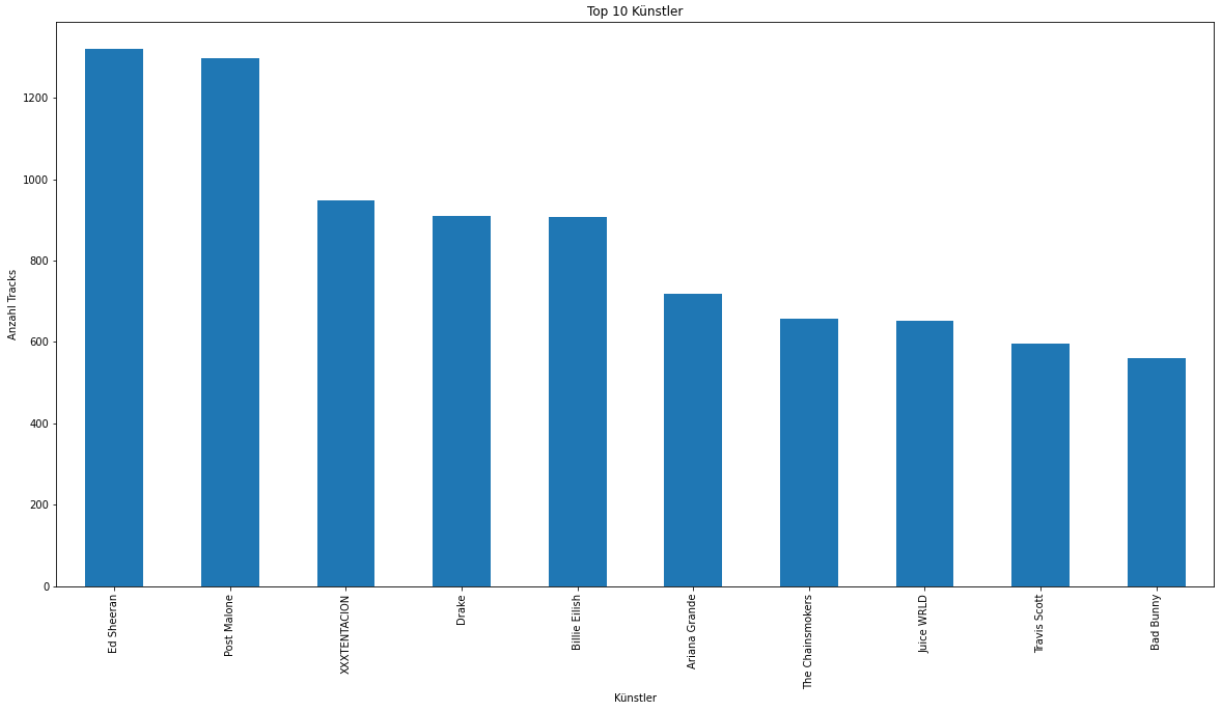


In [39]:
```
df20 = df_spotify_Nullwerte_entfernt.toPandas()
df20.to_csv('/home/bigdata/Dokumente/komplette_dateien.cs
```

In [ ]:
```
```

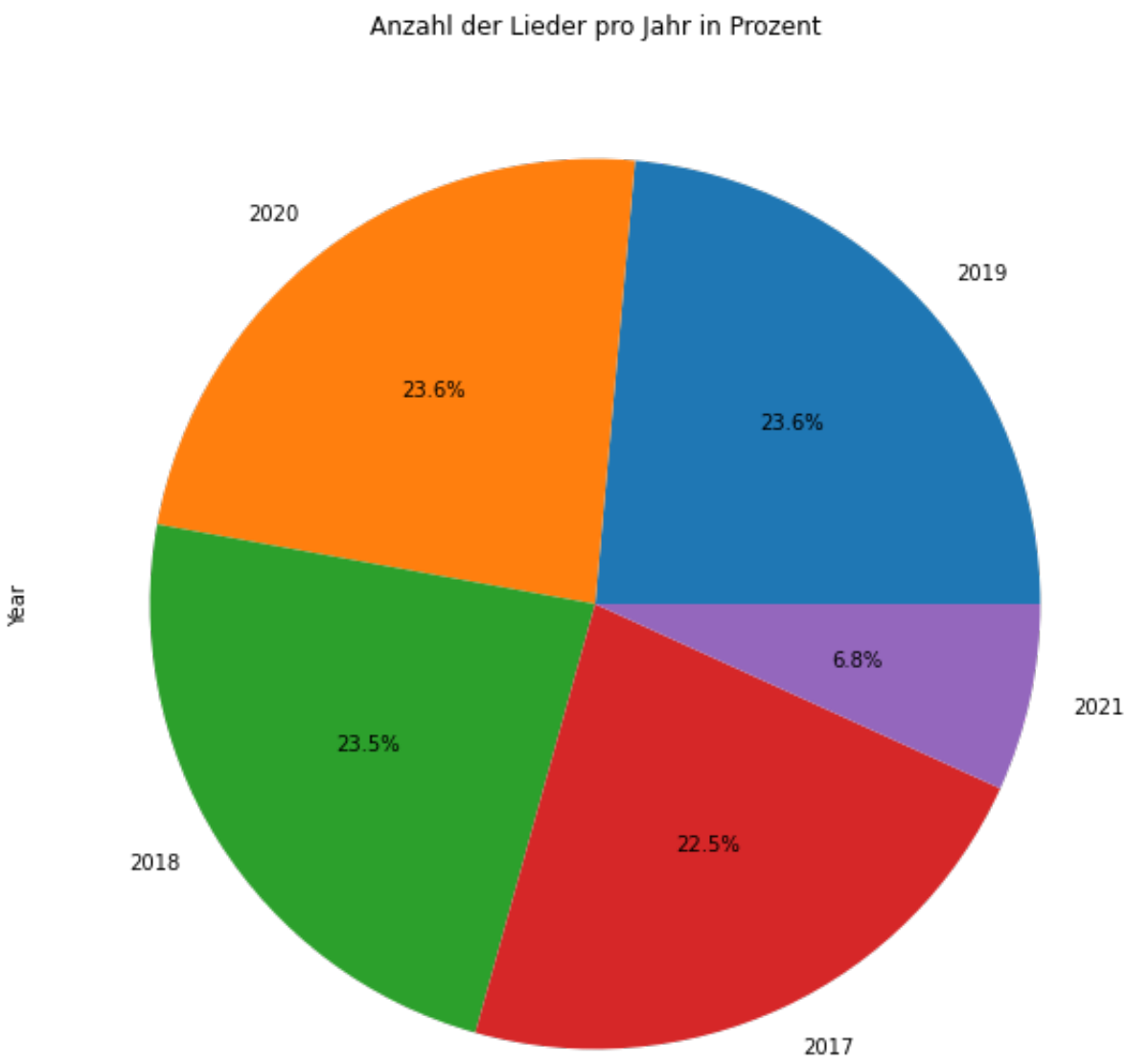
In [24]:
```
#Graph 3:
pandasDF['Artist'].value_counts().head(10).plot.bar(figsi
plt.xlabel('Künstler')
plt.ylabel('Anzahl Tracks')
plt.title('Top 10 Künstler')
```

Out[24]: Text(0.5, 1.0, 'Top 10 Künstler')



In [22]:

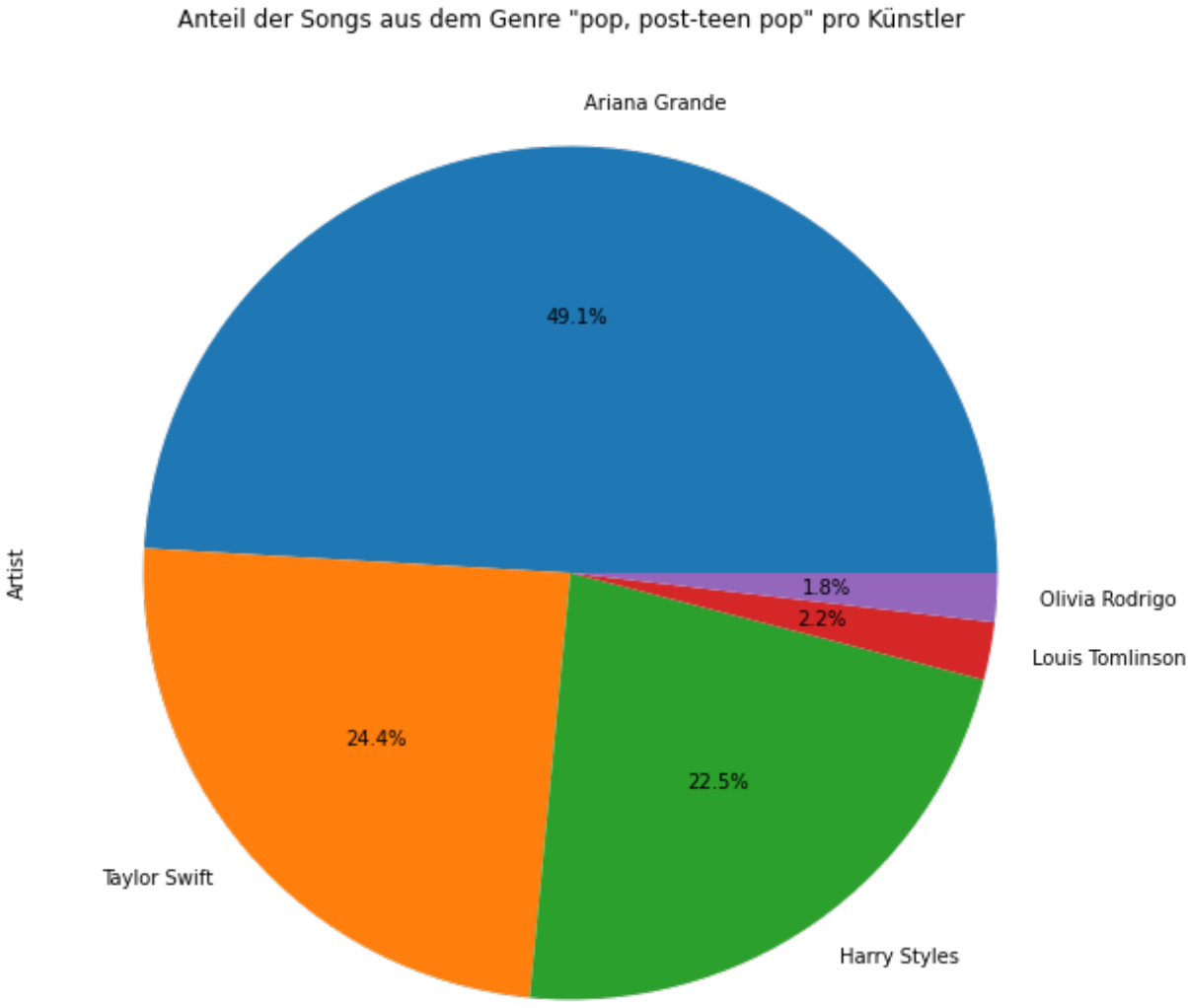
```
#Graph 1:
#print('\nAnzahl der Lieder pro Jahr in Prozent')
pandasDF = df_spotify_Nullwerte_entfernt.toPandas()
pandasDF['Year'].value_counts().plot.pie(figsize=(10,10),
plt.title('Anzahl der Lieder pro Jahr in Prozent')
plt.show()
```



In [36]:

```
#Graph 2:
pandasDF[pandasDF['Artist_Genres']=="['pop', 'post-teen p
plt.title('Anteil der Songs aus dem Genre "pop, post-teen
```

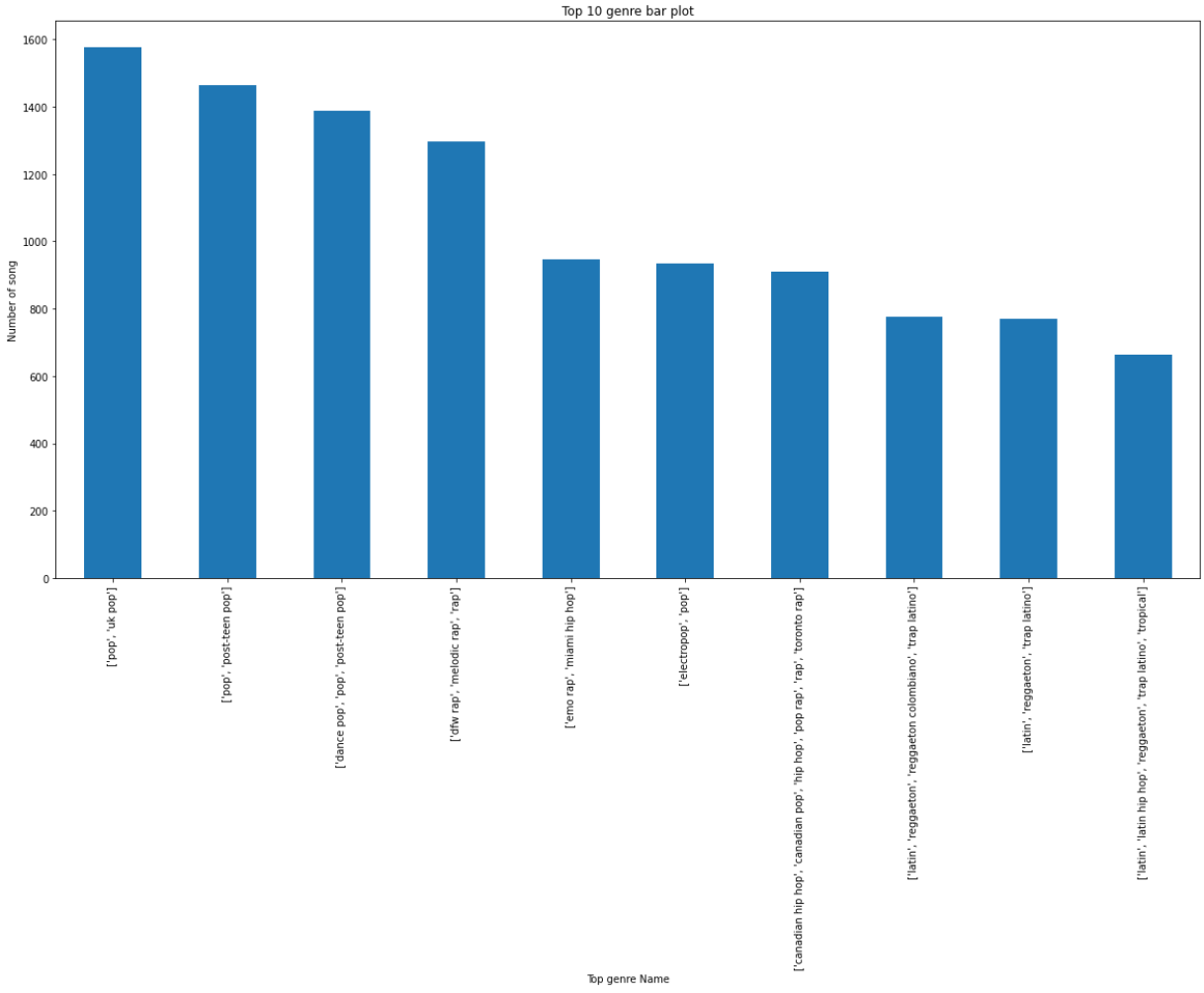
Out[36]: Text(0.5, 1.0, 'Anteil der Songs aus dem Genre "pop, post-teen pop" pro Künstler')



In [31]:

```
#Graph 5:
pandasDF = df_spotify_Nullwerte_entfernt.toPandas()
pandasDF['Artist_Genres'].value_counts().head(10).plot.ba
plt.xlabel('Top genre Name')
plt.ylabel('Number of song')
plt.title('Top 10 genre bar plot')
```

Out[31]: Text(0.5, 1.0, 'Top 10 genre bar plot')



In [29]:

```
#Graph 4:
pandasDF['Artist'].value_counts().head(10).plot.pie(figsi
plt.title('Top 10 Künstler basierend auf der Anzahl an So
```

Out[29]: Text(0.5, 1.0, 'Top 10 Künstler basierend auf der Anzahl an Songs in Prozent')