For this assignment you will perform an exploratory data analysis (EDA) for historic data from Billboard Hot 100 Chart. The Billboard Hot 100 is the music industry standard record chart in the United States for songs, published weekly by Billboard magazine. Chart rankings are based on sales, radio play, and online streaming in the United States. Every week, Billboard releases The Hot 100 chart of songs that were trending on sales and airplay for that week. The dataset you will analyse is a collection of all The Hot 100 charts released since its inception in 1958. The song with rank 1 is the most popular song that week. Marks will be awarded for complete and correct answers to all 10 questions. For full marks, you must show clearly all steps and computations in your answers. You must submit your completed Jupyter notebook in .ipynb and PDF format. To create a PDF of your notebook select File → Download as → PDF via LaTeX (.pdf).

In [15]:

```
import pandas as pd
import datetime
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
%matplotlib inline
```

1. **Load the charts.csv dataset into Python as a pandas DataFrame. Include the arguments parse_dates=True in the read_csv function to convert the date strings into datetime objects. Describe the data. Comment on the size of the dataset and the type of data in each column.**

In [3]:

```
df = pd.read_csv('charts.csv', parse_dates=True)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 328487 entries, 0 to 328486
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   rank            328487 non-null  int64  
 1   song            328487 non-null  object 
 2   artist          328487 non-null  object 
 3   last-week       296422 non-null  float64
 4   peak-rank       328487 non-null  int64  
 5   weeks-on-board  328487 non-null  int64  
 6   date            328487 non-null  object 
dtypes: float64(1), int64(3), object(3)
memory usage: 17.5+ MB
None
```

1. **Write code to answer the following questions:**
   - **How many different dates are included in the data set? How many different artists featured in the charts since its inception in 1958?**
   - **Determine the top 10 artists that feature in the charts the most amount of times?**
   - **What is the maximum length of time a song featured in the charts? What was this song and who was the artist?**

In [4]:

```
print("Unique Dates in the Dataset: ", df['date'].nunique())
print("Different artists to have appeared in charts: ", df['artist'].nunique())
n = 10
top_10 = df['artist'].value_counts()[:n].index.tolist()
print('Top 10 Artists to have appeared in the Charts: ')
for i in range(n):
  print(i+1, top_10[i])

most_appeared_count = df['song'].value_counts(dropna=False)[0]
```

```
most_appeared_song = df['song'].value_counts().index.tolist()[0]
most_appeared_song_artist = df.loc[df['song'] == most_appeared_song, 'artist'].iloc[0]
print(f'The most appearances by a song is {most_appeared_count} by {most_appeared_song} s
ang by {most_appeared_song_artist}')
```

```
Unique Dates in the Dataset:  3285
Different artists to have appeared in charts:  10100
Top 10 Artists to have appeared in the Charts:
1 Taylor Swift
2 Elton John
3 Madonna
4 Kenny Chesney
5 Drake
6 Tim McGraw
7 Keith Urban
8 Stevie Wonder
9 Rod Stewart
10 Mariah Carey
The most appearances by a song is 208 by Stay sang by BTS
```

1. **Determing how many missing values there are in each column of the dataset. Can you think of a reason why why these values are missing? Replace any missing values in the last-week column with the integer 101. Is this a reasonable value to replace the missing values with?**

In [5]:

```
print("Columns and the number of missing values: ")
print(df.isna().sum())
print("..................................................")
df['last-week'] = df['last-week'].fillna(101)
```

```
Columns and the number of missing values:
rank                  0
song                  0
artist                0
last-week         32065
peak-rank             0
weeks-on-board        0
date                  0
dtype: int64
..................................................
```

Replacing missing values with 101 is not a reasonable way to handle the missing values since 101 is not a true representation of lack of a value. 0 would be a better option

1. **For each entry compute the difference between its rank last week and its current rank. Add the change in ranking as an additional column to the DataFrame. Create a numerical summary of change in ranking (mean, standard deviation, minimum, maximum, etc.) and interpret your results.**

Note: If the song is higher up the charts this week, the change in ranking should be positive. While if the song falls down the charts, the change in ranking should be negative.

In [6]:

```
df['change-in-ranking'] = df['last-week'] - df['rank']
print(df.head())
print("Mean change in Rank: ", df['change-in-ranking'].mean())
print("Standard Deviation Change in Rank: ", df['change-in-ranking'].std())
print("Minimum change in Rank: ", df['change-in-ranking'].min())
print("Maximum change in Rank: ", df['change-in-ranking'].max())
```

```
   rank                              song  ...        date  change-in-ranking
0     1                            Butter  ...  2021-07-17                0.0
1     2                         Good 4 U   ...  2021-07-17                0.0
2     3                        Levitating  ...  2021-07-17                1.0
3     4                     Kiss Me More   ...  2021-07-17               -1.0
4     5  Montero (Call Me By Your Name)    ...  2021-07-17                3.0

[5 rows x 8 columns]
```

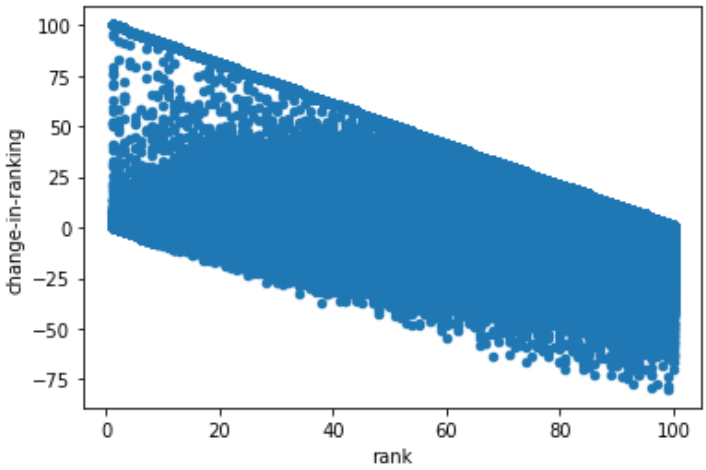
```
[5 rows x 6 columns]
Mean change in Rank:  2.312746014301936
Standard Deviation Change in Rank:  12.803090919085838
Minimum change in Rank:  -80.0
Maximum change in Rank:  100.0
```

1. **Create a scatter plot of the current rank versus the change in ranking from the previous week. Describe and interpret the plot. Compute the correlation coefficient between the current rank and the change in ranking. Does is agree with your what you found in your plot?**

In [7]:

```
df.plot.scatter(x = 'rank', y = 'change-in-ranking')
print('Correlation Coefficient: ', df['rank'].corr(df['change-in-ranking']))
```

```
Correlation Coefficient:  -0.048228471168613465
```



From the scatter plot, there is a general trend that an increase in the change in ranking leads to an better rank of the song. The correlation coefficient of -0.04 shows that ther is no meaningful relationship between the two columns

1. **Songs will typically appear in the charts for a number of weeks, as such, there are a number of duplicate entries. Delete all duplicate entries, keeping only the most recent entries of a particular song by a particular artist in the charts. How many unique songs (with unique artists) have appeared in the charts since its inception in 1958? Repeat Q2C for the new DataFrame, with unique entries only. Are the top 10 entries the same? If not, discuss why they may have changed. You may find the pandas method drop_duplicates useful. See the pandas documentation for details on how to apply this method to a DataFrame.**

In [10]:

```
uniquedf = df.drop_duplicates(subset=['song', 'artist'], inplace=False, keep='first')
print(len(uniquedf), "unique songs with unique artists")
top_10 = uniquedf['artist'].value_counts()[:n].index.tolist()
print('Top 10 Artists to have appeared in the Charts: ')
for i in range(n):
  print(i+1, top_10[i])
```

```
29467 unique songs with unique artists
Top 10 Artists to have appeared in the Charts:
1 Glee Cast
2 Taylor Swift
3 Drake
4 The Beatles
5 Aretha Franklin
6 Elton John
7 The Rolling Stones
8 The Beach Boys
9 Stevie Wonder
10 Madonna
```

The top 10 artists with most appearances has changed with the new dataset. This new list consist of artists with

The top 10 artists with most appearances has changed with the new dataset. This new list consist of artists with the highest number of unique songs in the billboard charts as compared to the previous one where most appearances included repeat appearances for the same song

1. **Recompute the summary statistics for the change in ranking from the previous week and recreate the scatter plot of the current ratings versus the change in ranking for the new DataFrame. Discuss and interpret any changes in these statistics and the plot.**

In [11]:

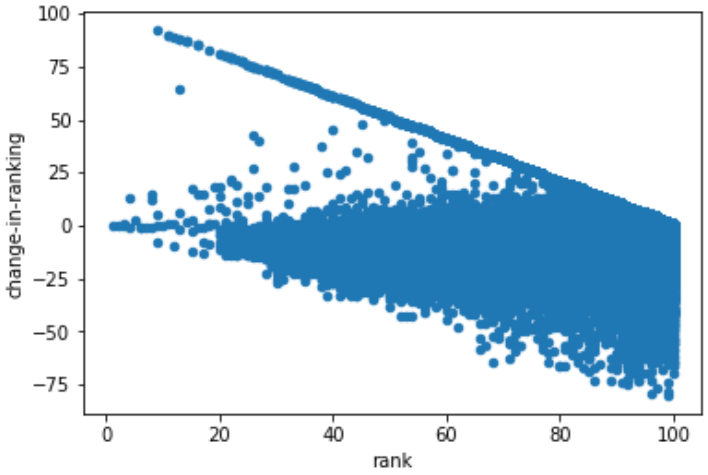
```
uniquedf['change-in-ranking'] = uniquedf['last-week'] - uniquedf['rank']
print("Mean change in Rank: ", uniquedf['change-in-ranking'].mean())
print("Standard Deviation Change in Rank: ", uniquedf['change-in-ranking'].std())
print("Minimum change in Rank: ", uniquedf['change-in-ranking'].min())
print("Maximum change in Rank: ", uniquedf['change-in-ranking'].max())

uniquedf.plot.scatter(x = 'rank', y = 'change-in-ranking')
print('Correlation Coefficient: ', uniquedf['rank'].corr(uniquedf['change-in-ranking']))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
Mean change in Rank:  -7.5868938134184
Standard Deviation Change in Rank:  15.88071740116989
Minimum change in Rank:  -80.0
Maximum change in Rank:  92.0
Correlation Coefficient:  -0.15500198550902927
```



Mean change in rank has changed from ~2.3 to ~7.58 Standard deviation has changed from ~12.8 to ~15.88 Maximum change in rankhas dropped to 92 from 100 The correlation coefficient has changed from 0.04 to 1.15, indicating a weak relationship between current week's rank and change in rank

1. **Using the new DataFrame, with unique entries only, create a histogram for the length of time each entry spent in the charts. Describe and interpret your plot. Compute the summary statistics for the length of time a song spends in the charts. Discuss these statistics in relation to your histogram.**

In [18]:

```
ax = uniquedf.hist(column='weeks-on-board', bins=100, grid=False, figsize=(12,8), color=
'#86bf91', zorder=2, rwidth=0.9)

ax = ax[0]
for x in ax:

    # Despine
    x.spines['right'].set_visible(False)
```

```
    x.spines['top'].set_visible(False)
    x.spines['left'].set_visible(False)

    # Switch off ticks
    x.tick_params(axis="both", which="both", bottom="off", top="off", labelbottom="on",
left="off", right="off", labelleft="on")

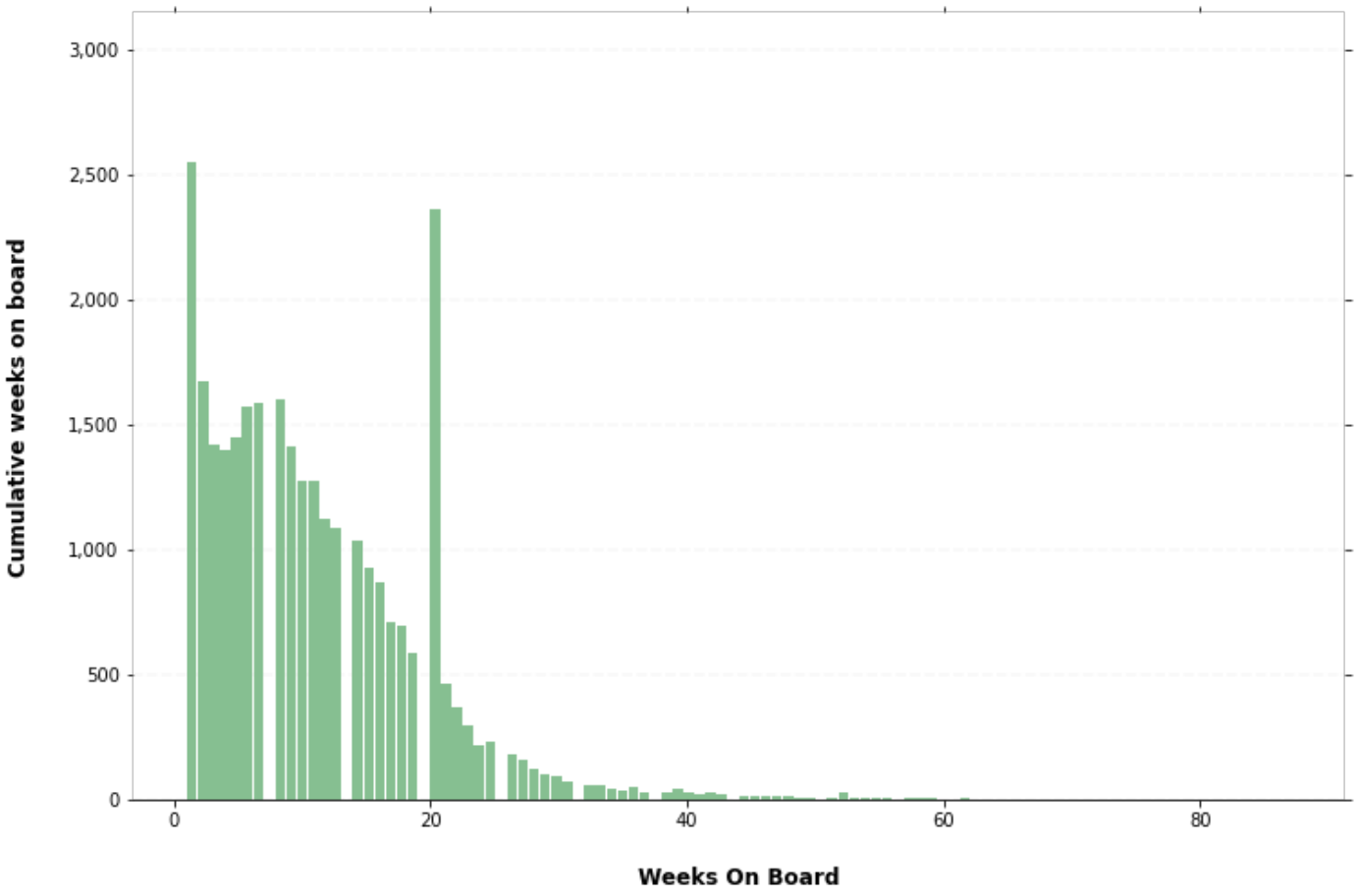
    # Draw horizontal axis lines
    vals = x.get_yticks()
    for tick in vals:
        x.axhline(y=tick, linestyle='dashed', alpha=0.4, color='#eeeeee', zorder=1)

    # Remove title
    x.set_title("")

    # Set x-axis label
    x.set_xlabel("Weeks On Board", labelpad=20, weight='bold', size=12)

    # Set y-axis label
    x.set_ylabel("Cumulative weeks on board", labelpad=20, weight='bold', size=12)

    # Format y-axis label
    x.yaxis.set_major_formatter(StrMethodFormatter('{x:,g}'))
```



Majority of songs spent less than 40 weeks on chart, with very few songs spending more than 60 weeks on chart

In [19]:

```
print("Mean time on the charts: ", uniquedf['weeks-on-board'].mean())
print("Standard Deviation Weeks on board: ", uniquedf['weeks-on-board'].std())
print("Minimum weeks on Board: ", uniquedf['weeks-on-board'].min())
print("Maximum weeks on Board: ", uniquedf['weeks-on-board'].max())
```

```
Mean time on the charts:  11.149082023959005
Standard Deviation Weeks on board:  8.287710754543093
Minimum weeks on Board:  1
Maximum weeks on Board:  87
```

Mean time on charts is 11 weeks, with a deviation of 8, coinciding with the histogram visualization that majority of the songs spend between 1 and 20 weeks on the board

1. **Using the new DataFrame, with unique entries only, create a histogram for the peak rank. Describe and interpret your plot. Compute the summary statistics for peak rank. Discuss these statistics in relation to your histogram.**

In [21]:

```
ax = uniquedf.hist(column='peak-rank', bins=100, grid=False, figsize=(12,8), color='#86b
f91', zorder=2, rwidth=0.9)

ax = ax[0]
for x in ax:

    # Despine
    x.spines['right'].set_visible(False)
    x.spines['top'].set_visible(False)
    x.spines['left'].set_visible(False)

    # Switch off ticks
    x.tick_params(axis="both", which="both", bottom="off", top="off", labelbottom="on",
left="off", right="off", labelleft="on")

    # Draw horizontal axis lines
    vals = x.get_yticks()
    for tick in vals:
        x.axhline(y=tick, linestyle='dashed', alpha=0.4, color='#eeeeee', zorder=1)

    # Remove title
    x.set_title("")

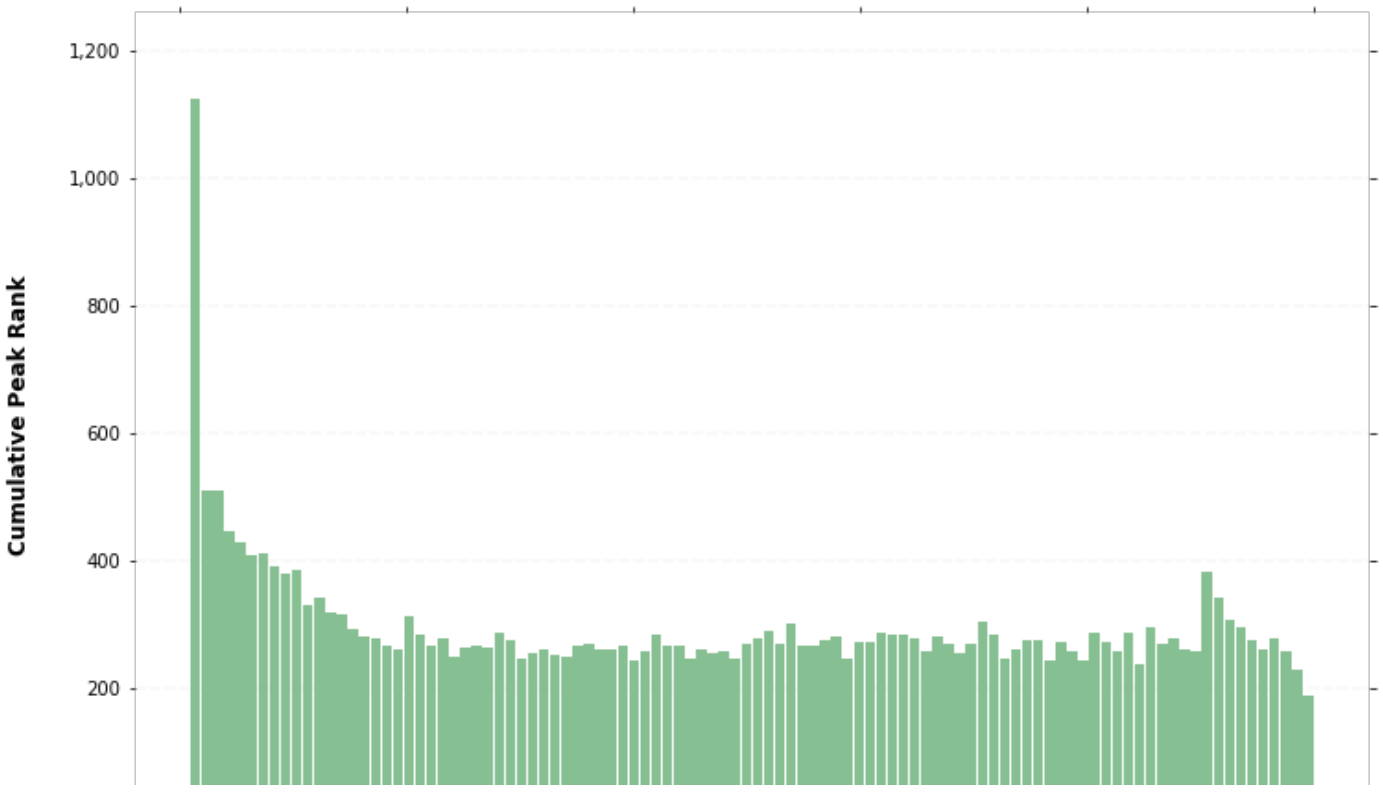
    # Set x-axis label
    x.set_xlabel("Peak Rank", labelpad=20, weight='bold', size=12)

    # Set y-axis label
    x.set_ylabel("Cumulative Peak Rank", labelpad=20, weight='bold', size=12)

    # Format y-axis label
    x.yaxis.set_major_formatter(StrMethodFormatter('{x:,g}'))

print("Mean peak rank: ", uniquedf['peak-rank'].mean())
print("Standard Deviation Peak Rank: ", uniquedf['peak-rank'].std())
```

```
Mean peak rank:  46.67930227033631
Standard Deviation Peak Rank:  30.541818343055507
```





The histogram portrays a generally uniform graph. The graph shows us that majority of the songs have a peak rank of 1 and the lowestpeak rank is 100. The charts have a peak rank of 46 with a standard deviation of 30.54. This mean and the large deviation lead to the generally uniform shape of the histogram, showing that majority of the songs are within 30 peak ranks of the mean peak rank of 46.

1. **Write a short summary on what you have learnt about the Billboard Hot 100 Chart from this analysis. (5-10 sentences)**

There are 29,467 unique songs in the billboard charts, which are far fewer than he total of 328,487. Glee cast has the highest number of unique songs on the billboard charts. The overall artist with the most appearances is Taylor Swift, indicating that she may be top due to the same songs staying on the chars for long periods of time. This ultimately means that the Billboard Hot 100 Chart often contains duplicate entries for different weeks.