Querying the IMDb Database

In this project, two functions were defined to tell a story about the IMDb database.

In [1]:

```
#Imported necessary packages for visual and querying analysis
import sqlalchemy as sa
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

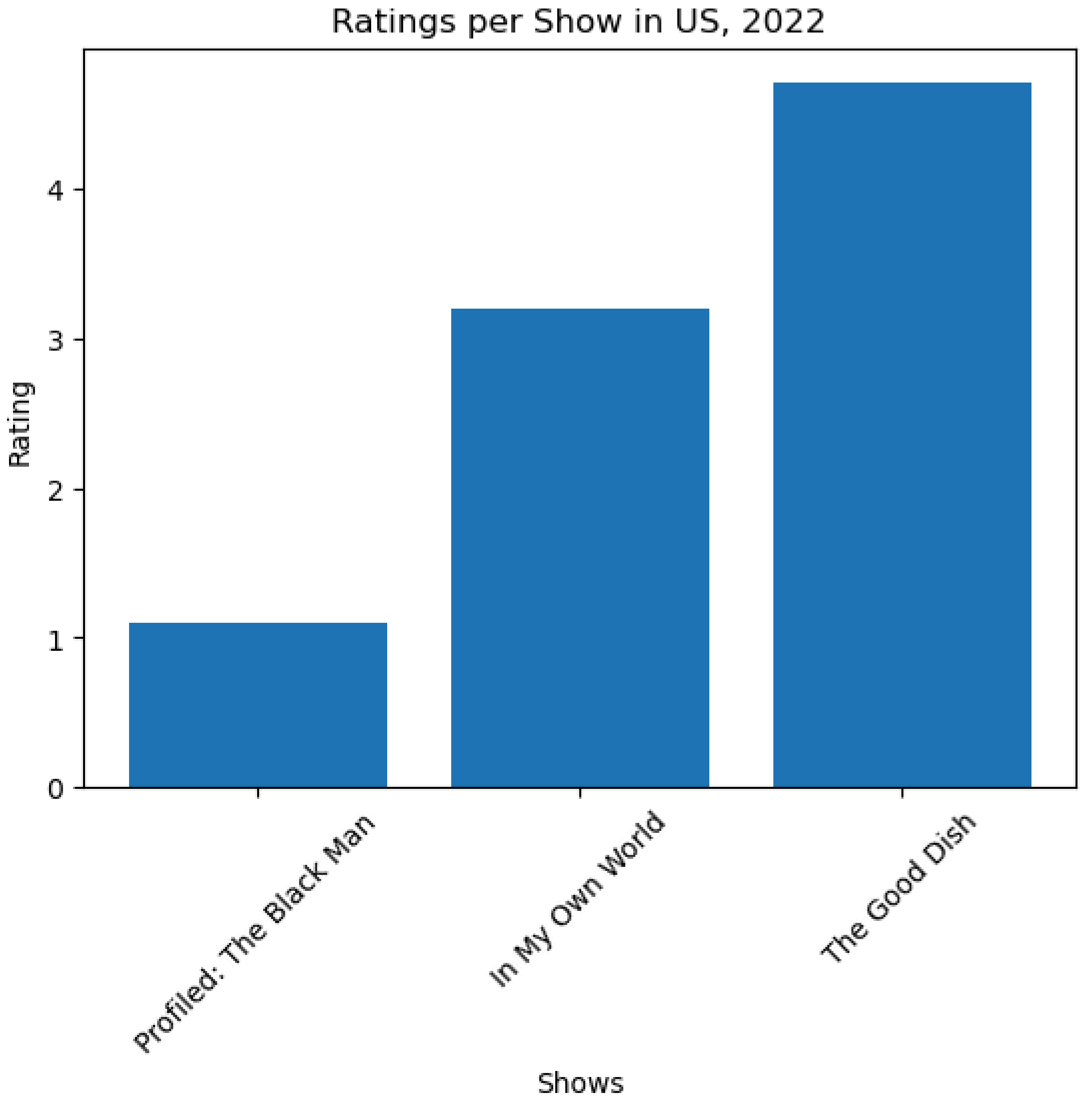
```
#Create a connection string, engine, and connection for definintions later
cstring = 'sqlite+pysqlite:///imdb.db'
engine = sa.create_engine(cstring)
connection = engine.connect()
```

In [3]:

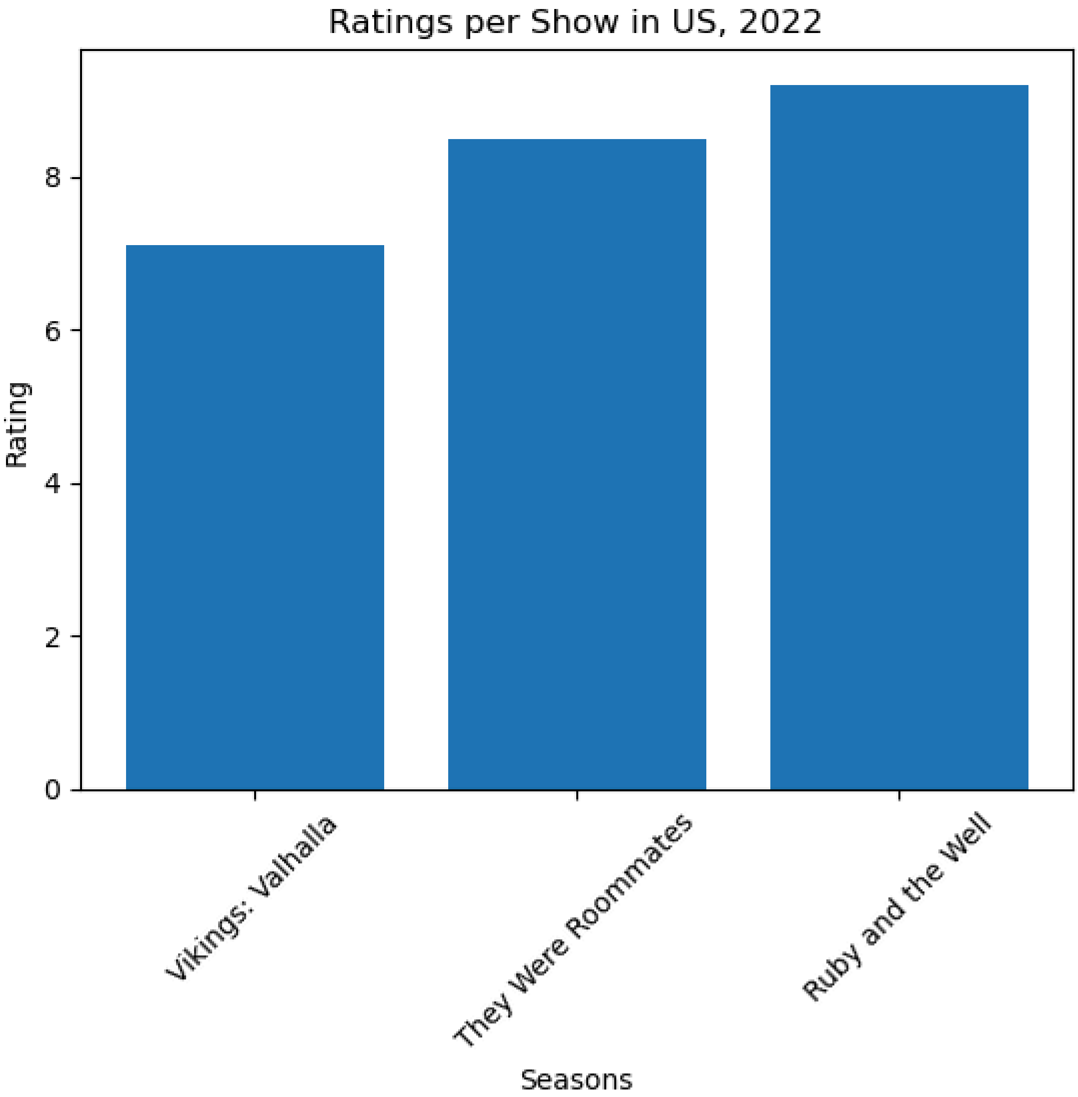
```
print('This function aims to find tv series that premiered in the same year and



def tvSeriesRatings(dbcon, rankHigh, region_name, year):
    if rankHigh == True:
        # Define SQL query where region and tv show are binded methods for high
        #ranking TV shows premiering at the users year
        pyquery = """
        SELECT DISTINCT t.primary_title AS Show, e.season_number AS Season, MAX
        FROM episodes AS e INNER JOIN titles AS t ON (e.show_title_id =t.title_
        WHERE t.type='tvSeries' AND  a.region= :region_in AND t.premiered = :st
        GROUP BY e.season_number
        ORDER BY max_rating
        LIMIT 3
        """
        # Bind query
        prepare_stmt = sa.sql.text(pyquery)
        bound_stmt = prepare_stmt.bindparams(region_in = region_name, startYea

        # Create dataframe from sql query
        df = pd.read_sql_query(bound_stmt, dbcon)

        # Create and display plot
        plt.bar(df.Show,df.max_rating)
        plt.title("Ratings per Show in " + str(region_name) + ", " + str(year)
        plt.xlabel("Seasons")
        plt.ylabel("Rating")
        plt.xticks(rotation=45)


    else:
        # Define SQL query where region and tv show are binded methods for lowe
        #ranking TV shows premiering at the users year
        pyquery = """
        SELECT DISTINCT t.primary_title AS Show, e.season_number AS Season, MIN
        FROM episodes AS e INNER JOIN titles AS t ON (e.show_title_id =t.title_
        WHERE t.type='tvSeries' AND  a.region= :region_in AND t.premiered = :st
        GROUP BY e.season_number
        ORDER BY min_rating
        LIMIT 3 """

        # Bind query
        prepare_stmt = sa.sql.text(pyquery)
        bound_stmt = prepare_stmt.bindparams(region_in = region_name, startYea

        # Create dataframe from sql query
        df = pd.read_sql_query(bound_stmt, dbcon)

        # Create and display plot
        plt.bar(df.Show,df.min_rating)
        plt.title("Ratings per Show in " + str(region_name) + ", " + str(year)
        plt.xlabel("Shows")
        plt.ylabel("Rating")
        plt.xticks(rotation=45)
    plt.show()
```

This function aims to find tv series that premiered in the same year and find their overall ratings.

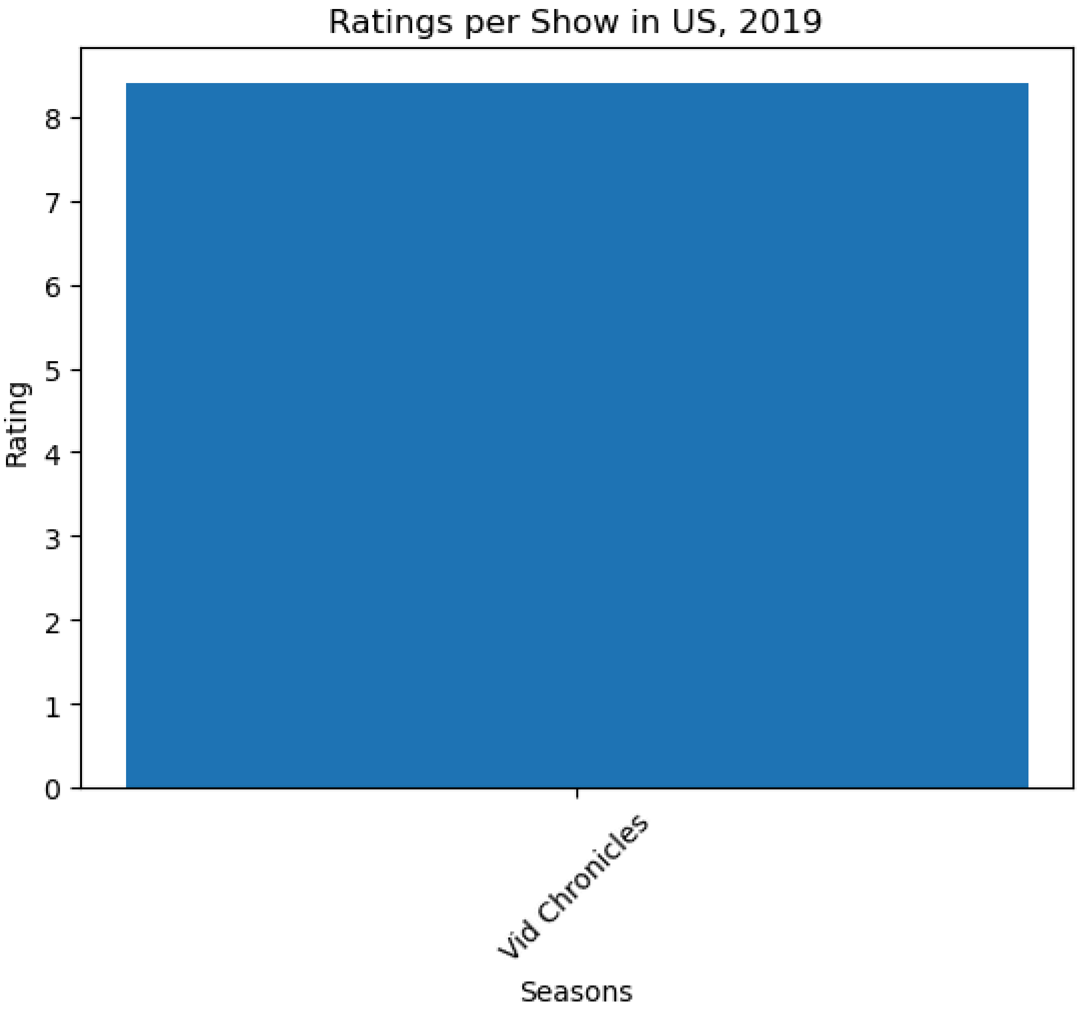
In [4]:
```
tvSeriesRatings(connection, False, 'US', 2022)
```



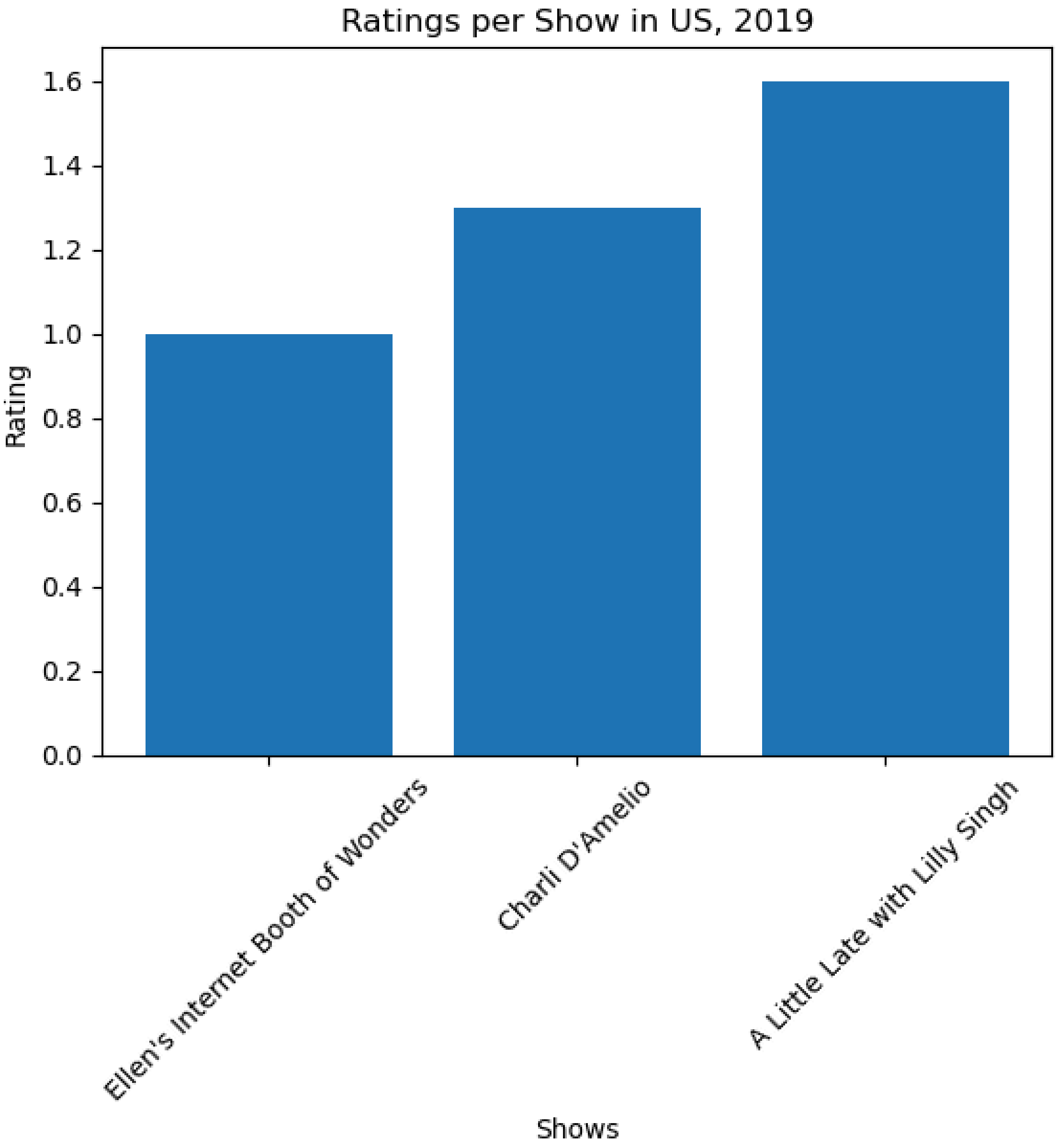
In [5]:
```
tvSeriesRatings(connection, True, 'US', 2022)
```



```
In [6]: tvSeriesRatings(connection, True, 'US', 2019)
```



```
In [7]: tvSeriesRatings(connection, False, 'US', 2019)
```



In 2022 in the US, the maximum lowest rated tv shows and the lowest highest rated tv series were around the same values. With Vikings: Valhalla and the Good Dish being only two ratings apart from each other. Compare this to the shows in 2019, there is a huge difference between the lowest and highest rated tv shows. Also, there was only one show in 2019 that had the highest rating, which could mean that people focused on watching tv series that were already airing. Both Ruby and the Well and Vid Chronicles are family-friendly tv series. As it is child friendly, there might be more mass appeal to the sow then if the show was rated for higher ages. More years would be needed to determine if family-friendly tv series rate the highest. All of the tv series in the year 2019 were influencer or celebrity based with **A Little Late with Lilly Singh** being a talk-show and **Ellen's Internet**, **Charli D'Amelio**, and **Vid Chronicles** all being available to view on Youtube. While in 2022, there was a mix of internet and traditional streaming networks. Only **They Were Roommates** being found on the internet to watch. With such low ratings in 2019, content shifts might need to have been changed that were remedied in 2022. Maybe more traditional content was in between the ratings.

In [17]:

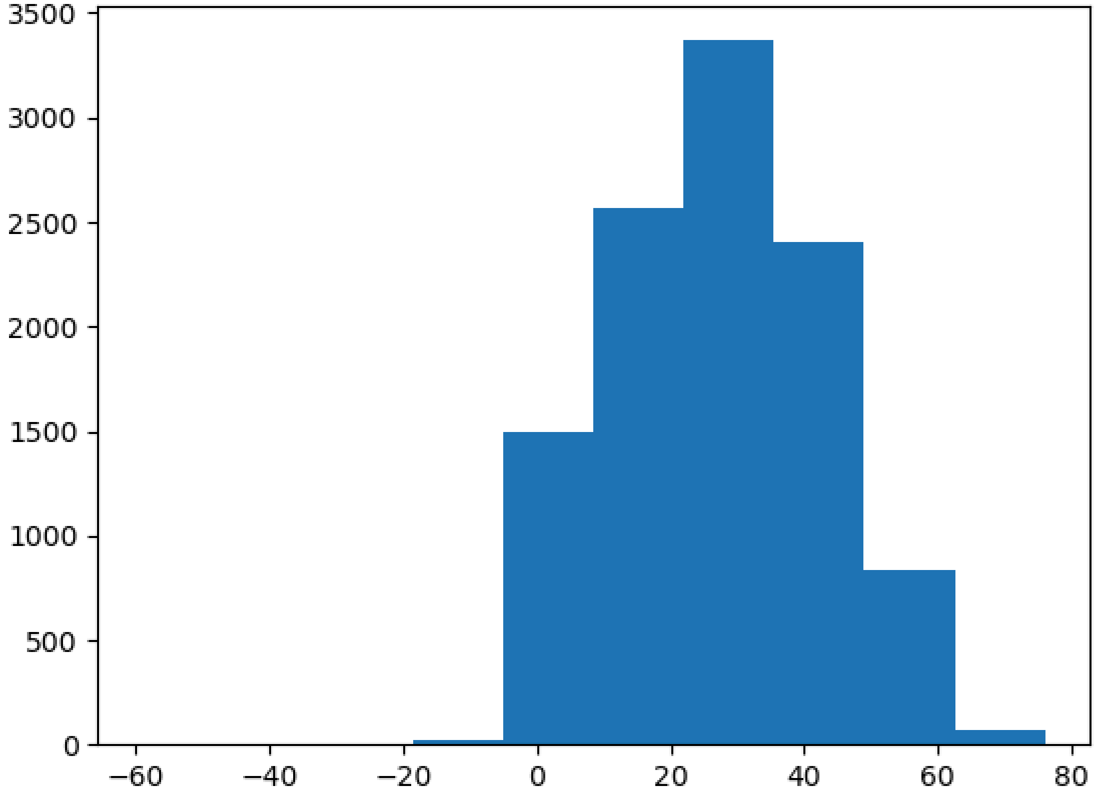
```
print('This function aims to find how long it took from the highest rated movi




def howLong(dbcon, category, formatType):

    # Define SQL query where region and tv show are binded methods
    pyquery = """
    SELECT p.name, p.died ,  MAX(r.rating) AS highest_rating, t.primary_title
    FROM people AS p INNER JOIN crew AS c USING (person_id) INNER JOIN titles
    WHERE c.category = :category_name AND  t.type= :type_name AND p.died IS NO
    GROUP BY p.name
    """

    # Bind query
    prepare_stmt = sa.sql.text(pyquery)
    bound_stmt = prepare_stmt.bindparams(category_name = category, type_name =

    # Create dataframe from sql query
    df = pd.read_sql_query(bound_stmt, dbcon)
    print('The category type is: ', formatType)

    print()

    # Find the difference between how long their highest rated movie was from
    df['timeDur'] = df['died']-df['premiered']
    print()

    # Summary statistics
    print('The max duration is ', df['timeDur'].max())
    print('The minimum duration is ', df['timeDur'].min())
    print('The average is', df['timeDur'].mean())
    print()

    #Histogram visualization
    print('Here is the distribution: ')

    plt.hist(df.timeDur, bins = 10)

    #Show historgram
    plt.show()

    print()
    print("=================")
    print()
    print('This histogram displays something interesting. There are some people
    print('highest rated ' +str(formatType)+ ' premiered. Should we explore mo
    print()
    print("=================")
    print()

    #Create a dataframe of when the time duration is negative
    negative = df.timeDur <0
    negative_df = df[negative]

    #Create a new column and set all rows to temp
    negative_df['ratingValue'] = 'temp'
    #Replace temp with values
    negative_df.loc[negative_df.highest_rating <=5, 'ratingValue'] = 'Low'
    negative_df.loc[(negative_df.highest_rating >5) & (negative_df.highest_rat
    negative_df.loc[negative_df.highest_rating >8, 'ratingValue'] = 'Excellent
    print()

    #Aggregate on low, good, and excellent
    low_row = negative_df[negative_df['ratingValue']=='Low']
    low_table = low_row.agg({'highest_rating': ['median', 'mean']})
    print('Here is the median and mean of the low rows:')
    print()
    print(low_table)
    print()
    print()

    good_row = negative_df[negative_df['ratingValue']=='Good']
    good_table = good_row.agg({'highest_rating': ['median', 'mean']})
    print('Here is the median and mean of the good rows:')
    print()
    print(good_table)
    print()
    print()

    excellent_row = negative_df[negative_df['ratingValue']=='Excellent']
    excellent_table = excellent_row.agg({'highest_rating': ['median', 'mean']}
    print('Here is the median and mean of the excellent rows:')
    print()
    print(excellent_table)
```

This function aims to find how long it took from the highest rated movie out of a certain category to their death

In [18]:

```
howLong(connection, 'director', 'movie')
```

```
The category type is:  movie


The max duration is  76
The minimum duration is  -59
The average is 26.88804307463795

Here is the distribution:
```



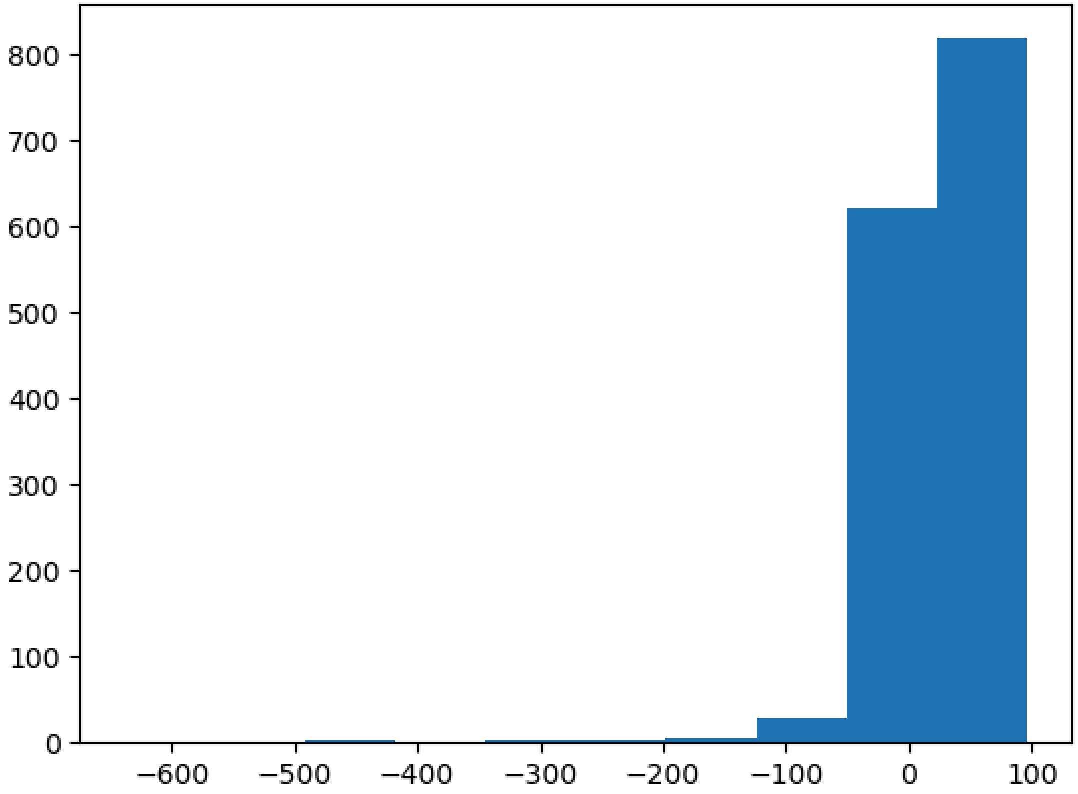
```
=================

This histogram displays something interesting. There are some people who died
before their
highest rated movie premiered. Should we explore more?

=================


Here is the median and mean of the low rows:

        highest_rating
median            4.10
mean              3.88


Here is the median and mean of the good rows:

        highest_rating
median        7.100000
mean          6.888889


Here is the median and mean of the excellent rows:

        highest_rating
median        8.500000
mean          8.523529
```

```
/var/folders/l9/7llt8nnx63vdf6xh_b37d3sm0000gn/T/ipykernel_40696/1574408725.p
y:58: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  negative_df['ratingValue'] = 'temp'
```

Compared to the composer distribution, the director distribution looks more normal. There is not such a huge trailing amount of time since the composer died to their highest rated form of entertainment. Very few directors, in comparison, died before their film was released. Based on the summary statistics, on average, directors were 27 years from their highest rated movie to their death. The mean and median of the excellent rows were very similar, very few directors can get higher than an 8.5 rating.

```
In [19]: howLong(connection, 'composer', 'short')
```

```
The category type is:  short


The max duration is  96
The minimum duration is  -638
The average is 20.568059299191376

Here is the distribution:
```



```
=================

This histogram displays something interesting. There are some people who died
before their
highest rated short premiered. Should we explore more?

=================


Here is the median and mean of the low rows:

        highest_rating
median        4.850000
mean          4.533333


Here is the median and mean of the good rows:

        highest_rating
median        7.000000
mean          6.835484


Here is the median and mean of the excellent rows:

        highest_rating
median        8.700000
mean          8.686667
```

```
/var/folders/l9/7llt8nnx63vdf6xh_b37d3sm0000gn/T/ipykernel_40696/1574408725.p
y:58: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  negative_df['ratingValue'] = 'temp'
```

There is such a difference between composers and directors because there is no way that a studio would hang on to a movie for that long because it would get out dated and not be relevant for the consumer of that generation. However, that speaks to the power that music has that something that was created 600 years before can still be in modern entertainment. While the mean and median was high for the lowest ratings, on the excellent ratings it was on the lower end. There also seems to be a pretty even distribution between people who had died before their short premiered and those who were still alive.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```