Student Name: Yousef Al-ghanim

Student ID: 2122123716

Exploratory Data Analysis

Perform exploratory analysis to find some initial insights on the following data sets:

- movies.json (https://raw.githubusercontent.com/vega/vega-datasets/gh-pages/data/movies.json)

Remember, that you are approaching the data with no specific question, only to get some general insights on it, so you can be able to ask the right questions in future analysis.

Be sure to perform the following steps:

1. Identify the variables in the data set and prepare a table describing what each variable represents. See table markdown (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#tables) to see how to write create markdown tables in your report. Description should include:
   - Variable definition
   - Data type
   - Missing data report
   - Report on the distribution of the data
   - level of analysis
2. Include table or list of all transformed variables/aggregations that were used in the study and include:
   - Variable description
   - Steps in transformation
   - Distribution if applicable
   - level of analysis
3. Start exploring relationships and groups to identify insights. Under every graph, write the main insights derived from the graph, and then compile a list of insights at the top of the report
4. Prepare a list of possible questions that come to mind after discovering these insights, and explain whether the question can be answered with the current data, or will require more data?

Note: Include responses to these 4 items in the top 4 cells of the report using mardown. the analysis should be at the bottom of the report in a section labeled **Analysis**

Results:

1- Variable identification in the data set:

Variable	Variable definition	Data type	Missing data report	Report on the distribution of the data	level of analysis
Creative_Type	The creative type of the movie	Categorical(string)	446 missing data	The column consist out of 9 unique values and we can use Pie Chart to show the distribution	Movie
Director	This column will show the director of the movie	Categorical(string)	1331 missing data	The column consist out of 550 unique values, where Steven Spielberg is the most frequent director in the data set	Directors
Distributor	The Distributor of the movie	Categorical(string)	232 missing data	The column consist out of 174 unique values where Warner Bros. distributed most of the movies	Distributors
IMDB_Rating	IMDB rating out of 10	Continues(float)	211 missing data	After ploting the values of this column using histogram we can say that it's skewed to the left	Movie
IMDB_Votes	How many IMDB users voted on the movie	Continues(float)	211 missing data	The maximum value of the column is 519541 and the minimum value is 18	Movie
MPAA_Rating	The Motion Picture Association of America film rating of the movie	Categorical(string)	602 missing data	The column consist out of 7 unique values and we can use Pie Chart to show the distribution	Movie
Major_Genre	The genre of the movie	Categorical(string)	275 missing data	The column consist out of 10 unique values and we can use Pie Chart to show the distribution	Genres
Production_Budget	The production budget of the movie	Continues(float)	1 missing data	The maximum value of the column is 300000000 and the minimum value is 218	Movie

Variable	Variable definition	Data type	Missing data report	Report on the distribution of the data	level of analysis
Release_Date	The releasing date of the movie	Continues(Datetime64)	7 missing data	The first movie was released on 1920-09-17 and the last movie was released on 2016-12-24	Movie
Rotten_Tomatoes_Rating	Rotten Tomatoes rating out of 100	Since the values of the variable is only from 0 to 100 and does not containt any decimal places then we can say that it's Discrete(float)	880 missing data	The highest rating is 100 and the lowest rating is 1	Movie
Running_Time_min	The running time of the movie in minutes	Continues(float)	1992 missing data	After ploting the values of this column using histogram we can say that it's skewed to the right	Movie
Source	The source of the movie wether it's original or based on books, remake and etc..	Categorical(string)	365 missing data	The column consist out of 18 unique values and we can use Bar Chart to show the distribution	Sources
Title	The title of the movie	Categorical(string)	1 missing data	This column consist out of 3177 unique values	Movie
US_DVD_Sales	The total DVD sales in the US	Continues(float)	2637 missing data	The maximum amount of DVD sales in the US is 352582053 and the minimum is 618454	Movie
US_Gross	The Gross sales of the movie in the US	Continues(float)	7 missing data	The minimum US Gross sales is 0.0 and maximum is 760167650.0.	Movie
Worldwide_Gross	The Gross sales of the movie in the Worldwide	Continues(float)	7 missing data	The minimum Worldwide Gross sales is 0.0 and maximum is 2767891499.0.	Movie

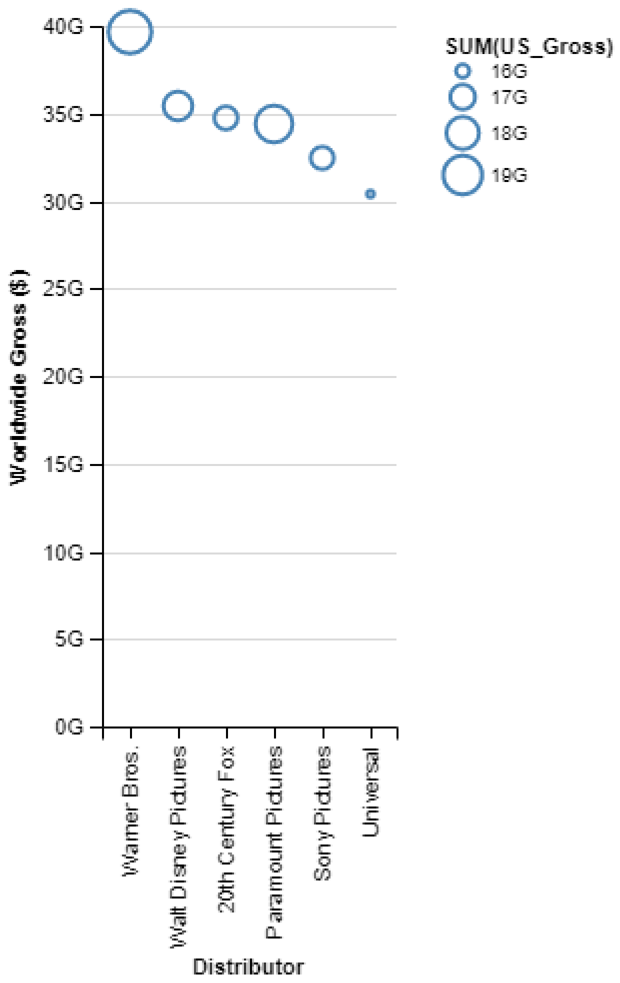
2- Transformed variables:

Variable	Variable description	Steps in transformation	Distribution	level of analysis

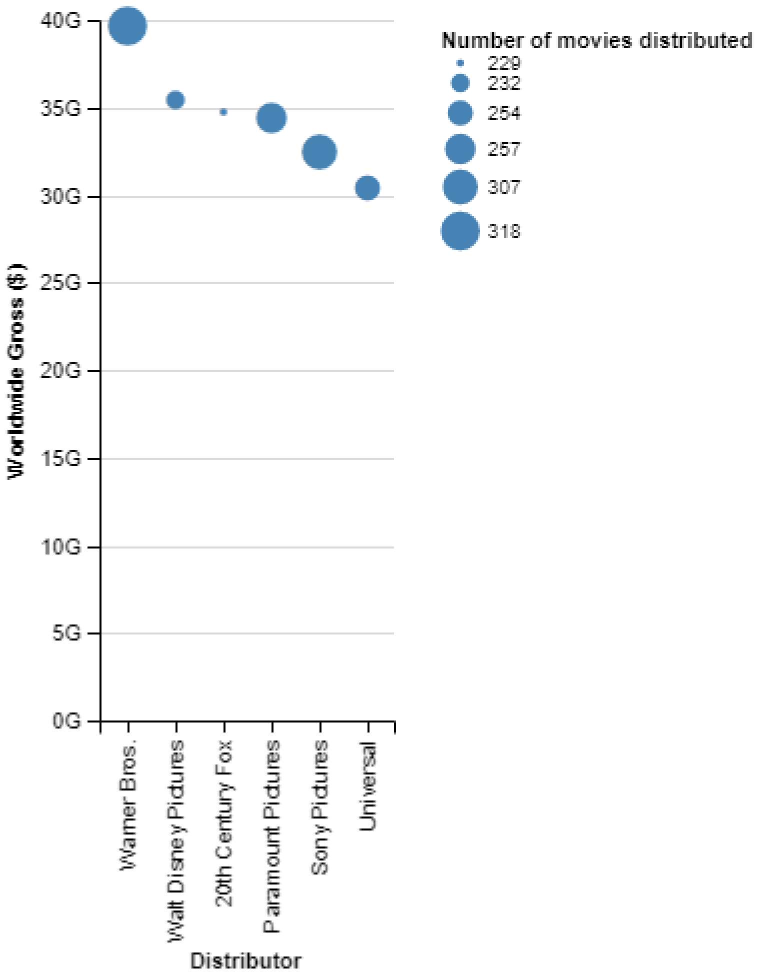
Variable	Variable description	Steps in transformation	Distribution	level of analysis
Worldwide_Gross_Excluding_US	The gross sales for the movie in the worldwide excluding the US	We can do that by deducting the US gross from Worldwide gross	The minimum Worldwide Gross Excluding US is 0.0 and maximum is 2007723849.0.	Movie
Mean_IMDB_Rating	The average of the IMDB rating across the data set	By using the mean method in the dataframe and assigning the result to new column	Not applicable	Movie
Major_Genre_Worldwide_Gross	The total gross of each of the major genre	Using the groupby method in the dataframe on the Major_Genre column then transform method has been used to sum the Worldwide_Gross in each column	The minimum value of this column is 153622009 (Concert/Performance) and the maximum value of this column is 66080959632 (Adventure)	Genres
Log_Production_Budget	The log of the column Production_Budget	We used log function in numpy package and applied it to the column	Skewed to the left	Movie
Log_Worldwide_Gross	The log of the column Worldwide_Gross	We used log function in numpy package and applied it to the column	Skewed to the left	Movie

3- Insights:

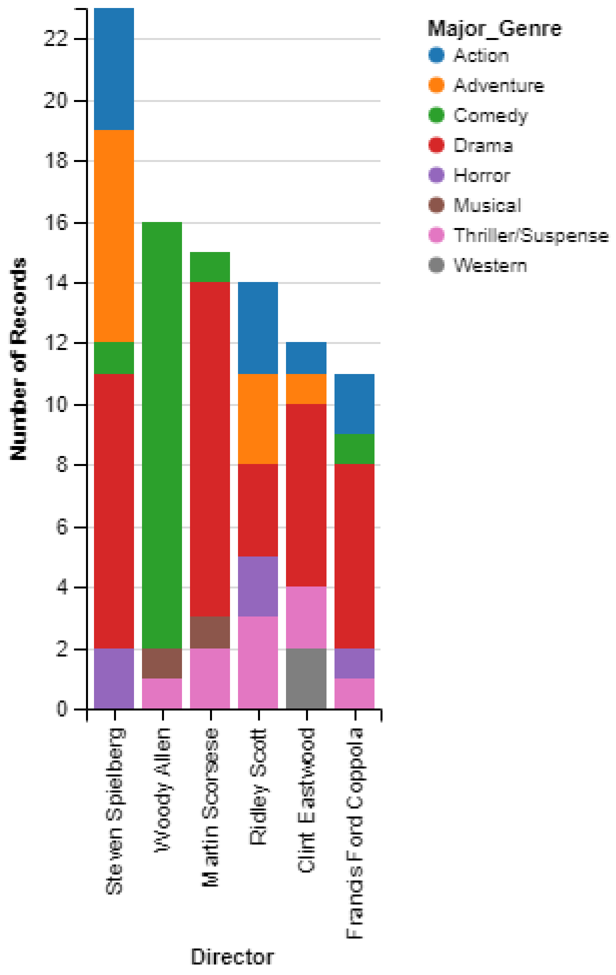
- Warner Bros. make most of their profit in the US and Universal Make most of their profit out side the US:



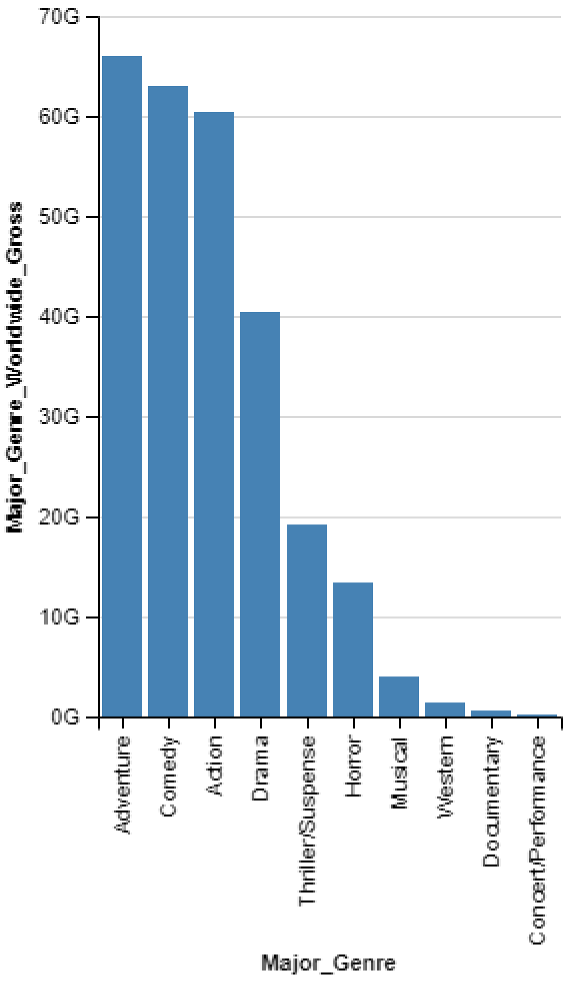
- Walt Disney Pictures and 20th Century Fox distributed less movies than Sony Pictures, Paramount Pictures, and Universal yet thier movies made more profit. :



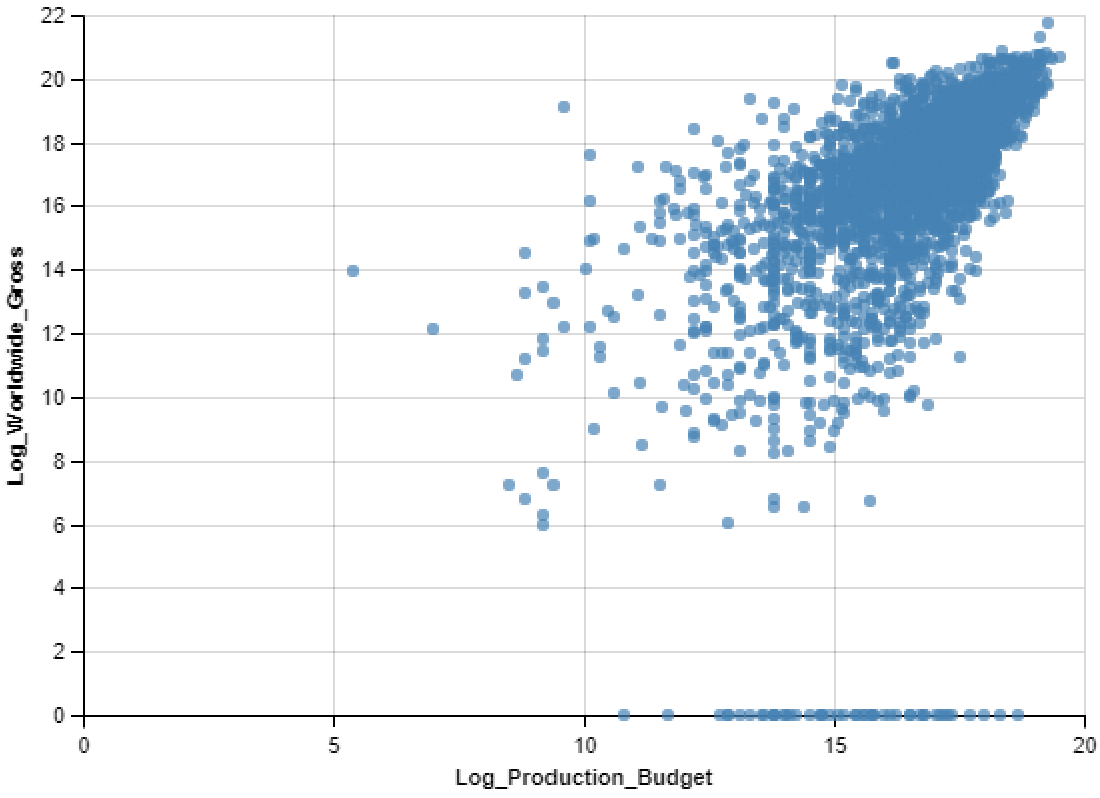
- 80% of the movies that director Woody Allen has directed is a comdey movie:



- The genre that made the most worldwide gross is the adventure genre:

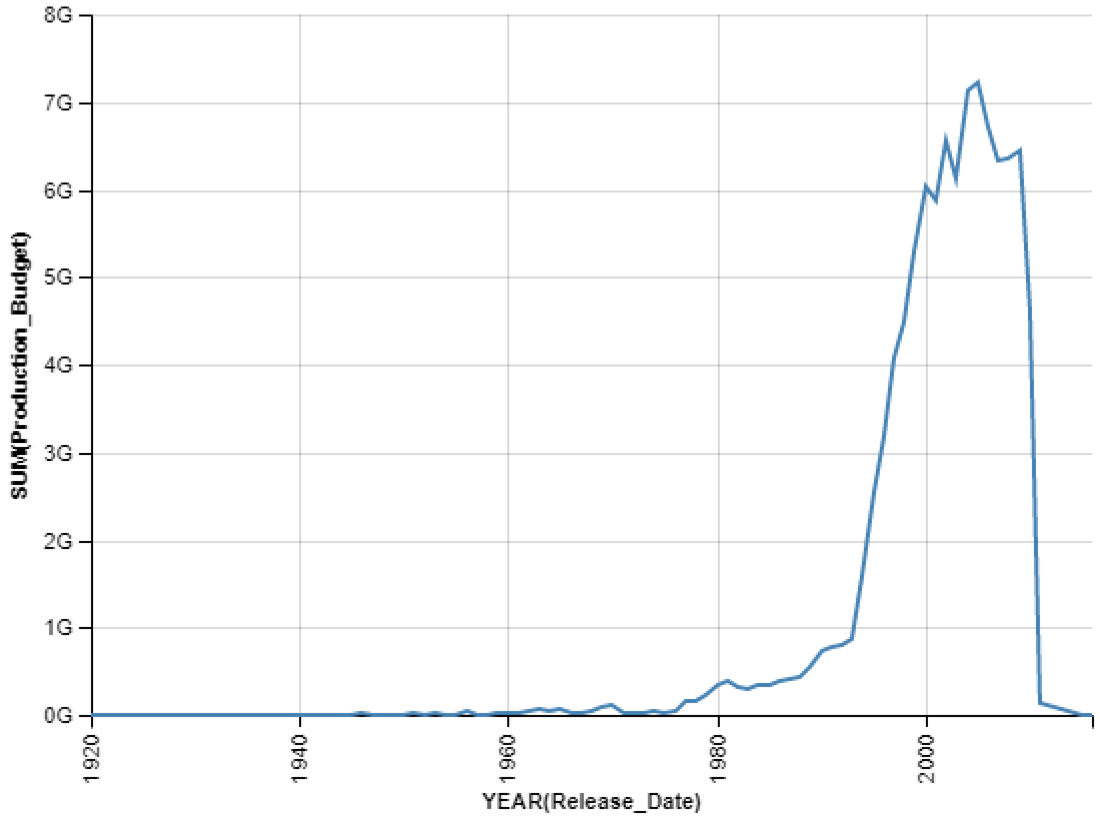


- There is <u>positive relationship</u> between production budget and worldwide gross:



4- Questions:

1. why the production budget increased dramatically in the 1990's? (maybe if we know more about the number of people involved in the movie (cast) and technology used we can answer that question)



2. why there is a difference in IMDB rating and rotten tomatoes rating? (if we had the reviews or comments from the each platform we might be able to perform a sentiment analysis to see which type of users do each platform serve)
3. Is there any correlation between the actors that act in the movies and the rating of the movie? (list of actors need to perform that analysis (SNA?))

Analysis

In [195]:
```
# importing the necessary packages:
import pandas as pd
import altair as alt
from imdbpie import Imdb
import numpy as np
import re
%matplotlib inline
# initializing imdb client. ref:https://github.com/richardasaurus/imdb-pie
imdb = Imdb()
imdb = Imdb(anonymize=True)
#loading json file into dataframe:
url='https://raw.githubusercontent.com/vega/vega-datasets/gh-pages/data/movies.js
movies_df = pd.read_json(url)
movies_df.head()
```

Out[195]:

	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre	Pro
0	None	None	Gramercy	6.1	1071.0	R	None	
1	None	None	Strand	6.9	207.0	R	Drama	
2	None	None	Lionsgate	6.8	865.0	None	Comedy	
3	None	None	Fine Line	NaN	NaN	None	Comedy	
4	Contemporary Fiction	None	Trimark	3.4	165.0	R	Drama	

In [196]:
```
# displaying al columns in the dataframe
movies_df.columns
```

Out[196]:
```
Index(['Creative_Type', 'Director', 'Distributor', 'IMDB_Rating', 'IMDB_Votes',
       'MPAA_Rating', 'Major_Genre', 'Production_Budget', 'Release_Date',
       'Rotten_Tomatoes_Rating', 'Running_Time_min', 'Source', 'Title',
       'US_DVD_Sales', 'US_Gross', 'Worldwide_Gross'],
      dtype='object')
```

In [197]:
```
# displaying the data type of the columns
movies_df.dtypes
```

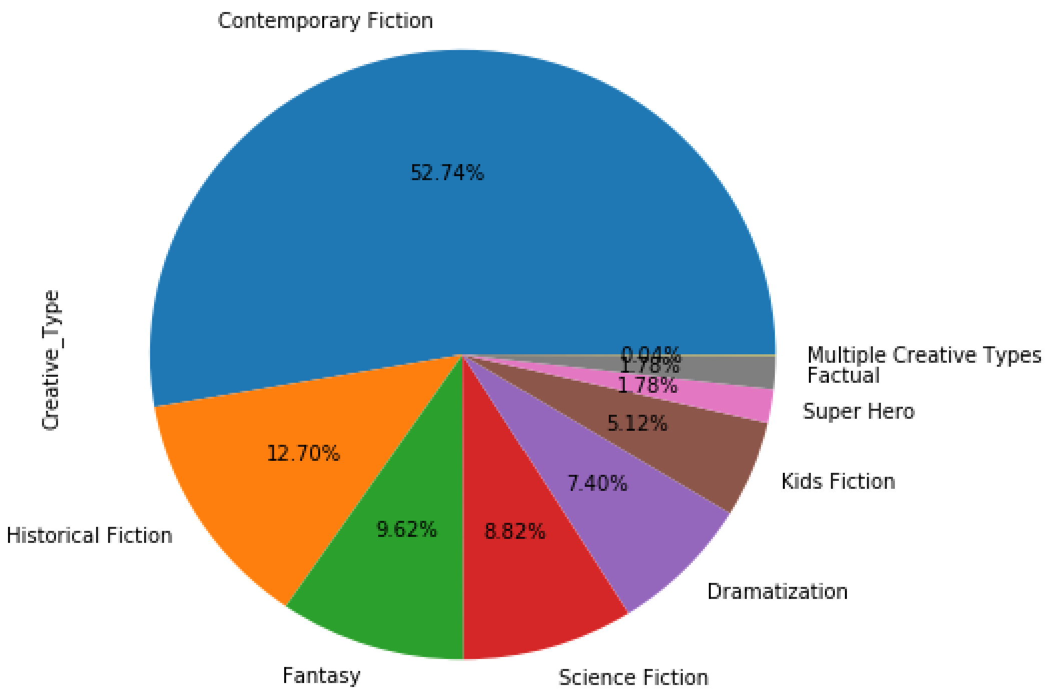
Out[197]:
```
Creative_Type              object
Director                   object
Distributor                object
IMDB_Rating               float64
IMDB_Votes                float64
MPAA_Rating                object
Major_Genre                object
Production_Budget         float64
Release_Date               object
Rotten_Tomatoes_Rating    float64
Running_Time_min          float64
Source                     object
Title                      object
US_DVD_Sales              float64
US_Gross                  float64
Worldwide_Gross           float64
dtype: object
```

In [198]:
```
# Checking for missing data in Creative_Type column:
movies_df.Creative_Type.isnull().sum()
```

Out[198]:
```
446
```

Back to top

```
In [199]: # checking for the distribution of the data of Creative_Type column:
          print(movies_df.Creative_Type.unique())
          movies_df.Creative_Type.value_counts().plot(kind='pie',autopct='%1.2f%%',figsize=
          # Contemporary Fiction is the most frequent creative type in the data set

          [None 'Contemporary Fiction' 'Science Fiction' 'Historical Fiction'
           'Fantasy' 'Dramatization' 'Factual' 'Super Hero' 'Multiple Creative Types'
           'Kids Fiction']

Out[199]: <matplotlib.axes._subplots.AxesSubplot at 0x263d9b3e8d0>
```



Back to top

```
In [200]: # Checking missing data in Director column
          print('There is {} missing data in Director column and {} unique values'.format(
              movies_df.Director.isnull().sum(),len(movies_df.Director.unique())-1))
          movies_df.Director.value_counts().head()
          # Steven Spielberg directed most of the movies in the data set

          There is 1331 missing data in Director column and 550 unique values

Out[200]: Steven Spielberg    23
          Woody Allen         16
          Martin Scorsese     15
          Spike Lee           15
          Ridley Scott        14
          Name: Director, dtype: int64
```

Back to top

In [201]:
```
# Checking for missing data in Distributor column
print('There is {} missing data in Distributor column and {} unique values'.forma
    movies_df.Distributor.isnull().sum(),len(movies_df.Distributor.unique())-1))
movies_df.Distributor.value_counts().head()
# Warner Bros.distributed most of the movies
```

```
There is 232 missing data in Distributor column and 174 unique values
```

Out[201]:
```
Warner Bros.            318
Sony Pictures           307
Paramount Pictures      257
Universal               254
Walt Disney Pictures    232
Name: Distributor, dtype: int64
```

In [202]:
```
# Checking for missing data in IMDB_Rating column (part 1)
print('There is {} missing data in IMDB_Rating column'.format(
    movies_df.IMDB_Rating.isnull().sum()))
```

```
There is 213 missing data in IMDB_Rating column
```

In [203]:
```
# I was trying to fetch imdb raings using imdbpie package,
# but I couldn't since there was a problem with the-
# Release_Date column where there are different formats
# for the date, so I need to fix it before fetching the data
```

In [204]:
```
# changing the data type of Release_data column to DateTime
movies_df['Release_Date'] = pd.to_datetime(movies_df['Release_Date'])
movies_df.dtypes
```

Out[204]:
```
Creative_Type                    object
Director                         object
Distributor                      object
IMDB_Rating                     float64
IMDB_Votes                      float64
MPAA_Rating                      object
Major_Genre                      object
Production_Budget               float64
Release_Date             datetime64[ns]
Rotten_Tomatoes_Rating          float64
Running_Time_min                float64
Source                           object
Title                            object
US_DVD_Sales                    float64
US_Gross                        float64
Worldwide_Gross                 float64
dtype: object
```

In [205]:
```
movies_df.Release_Date.head(10)
# after checking the data after converting the release date column to
# datetime, there is an issue with the conversion where date 1946 is converted to
# 2046 (check the last row).
```

Out[205]:
```
0   1998-06-12
1   1998-08-07
2   1998-08-28
3   1998-09-11
4   1998-10-09
5   1999-01-15
6   1999-04-04
7   1999-04-09
8   1986-07-01
9   2046-12-31
Name: Release_Date, dtype: datetime64[ns]
```

In [206]:
```
# reloading the dataframe
movies_df = pd.read_json(url)
```

In [207]:
```
# converting the release_date column using another method:
def change_date_format(date):
    '''This function will take a date as a string and
           change the format of that date'''
    if re.match('(\d{1,2})[-](\w{3})[-](\d{2})$', str(date)):
        day_month_year = date.split('-')
        if int(day_month_year[-1]) >= 19 :
            day_month_year[-1] = '19' + day_month_year[-1]
        else:
            day_month_year[-1] = '20' + day_month_year[-1]
        new_date = '-'.join(day_month_year)
        return new_date
    else:
        return date
movies_df.Release_Date = movies_df.Release_Date.apply(change_date_format)

# chaning the data type of Release_Date column
movies_df.Release_Date = pd.to_datetime(movies_df.Release_Date)
movies_df.Release_Date.head()
```

Out[207]:
```
0   1998-06-12
1   1998-08-07
2   1998-08-28
3   1998-09-11
4   1998-10-09
Name: Release_Date, dtype: datetime64[ns]
```

Back to top

In [208]:
```
# After fixing the format issue now we can fetch the data

# filling IMDB_Rating column missing data using imdb client
def get_imdb_rating(movie_title):
    '''this function will take the title of a movie and search in
            imdb client to find the rating of that movie base on
                    the title then the year'''
     # first of all we need to find the id of the movie by passing-
    # the title to search_for_title method
    movies = imdb.search_for_title(movie_title)
    movie_date = movies_df.loc[movies_df.Title == movie_title,'Release_Date'].to_
    regex = r"[?|$|.|!|,| |-|:|;|#|*|_|%|@|'|]"
    for movie_info in movies:
        movie_release_date = imdb.get_title_by_id(movie_info['imdb_id']).release_
        # to ensure accurucy we need to crossmatch the title of the movie and the
        if re.sub(regex,r'',movie_info['title'].lower()) == re.sub(regex,r'',movi
            if movie_release_date == movie_date:
                movie_rating = imdb.get_title_by_id(movie_info['imdb_id']).rating
                return movie_rating
            else:
                return np.nan
```

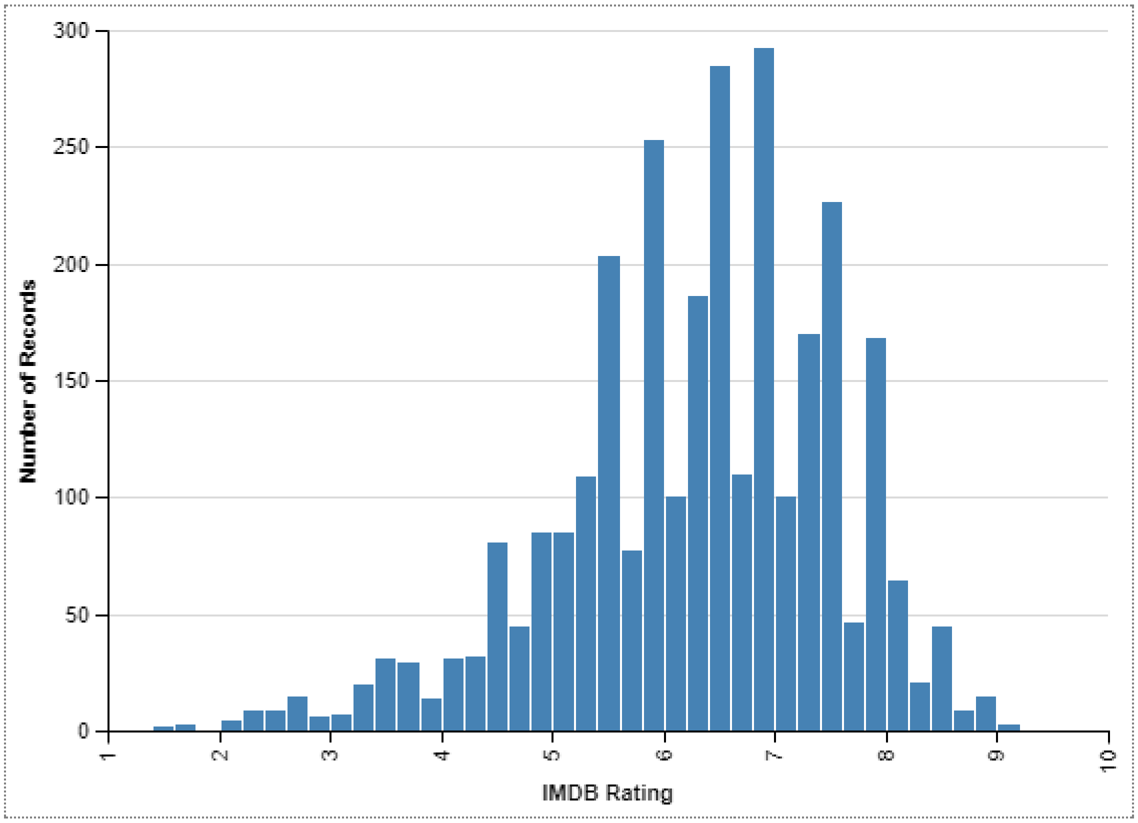
In [209]:
```
print('\n','''Before filling some of the missing values in the IMDB_Rating column
        now there is {} missing data in IMDB_Rating column.'''.format(
    movies_df.IMDB_Rating.isnull().sum()))
print(movies_df.IMDB_Rating.head())

# applying the function for the first 10 records(rows) with missing values becaus
# it will talk along time to apply it to all the records(rows) in the columns
movies_rating_df = movies_df.loc[(movies_df.IMDB_Rating.isnull() == True)
                &
                (movies_df.Release_Date.isnull() == False),'Title'][:10].apply(ge

movies_df.loc[movies_df.IMDB_Rating.isnull() == True
                     &
                (movies_df.Release_Date.isnull() == False), 'IMDB_Rating'] = movi
# checking if the function works
print('\n','''After filling some of the missing values in the IMDB_Rating column,
        now there is {} missing data in IMDB_Rating column.'''.format(
    movies_df.IMDB_Rating.isnull().sum()))
print(movies_df.IMDB_Rating.head())
# check row number 3 (3.7)
```

```
 Before filling some of the missing values in the IMDB_Rating column,
        now there is 213 missing data in IMDB_Rating column.
0    6.1
1    6.9
2    6.8
3    NaN
4    3.4
Name: IMDB_Rating, dtype: float64

 After filling some of the missing values in the IMDB_Rating column,
        now there is 211 missing data in IMDB_Rating column.
0    6.1
1    6.9
2    6.8
3    3.7
4    3.4
Name: IMDB_Rating, dtype: float64
```

In [210]:
```
# You must be asking why there is only 2 less missing data and we searched for 10
# movies with missing data,thats because there is some issues with movie title-
# and the accuracy of release date in the data set.
```

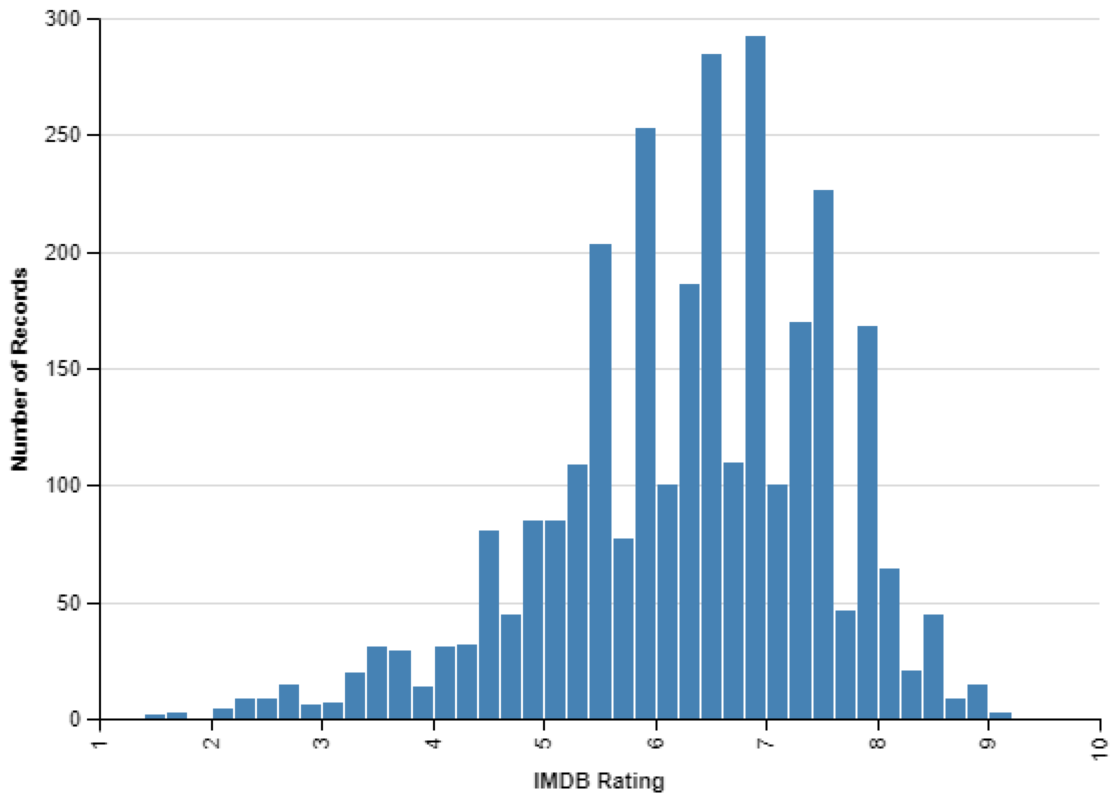
Back to top

```
In [211]: # Checking the distribution of the data in IMDB_Rating column:
          n_bins = int(np.sqrt(len(movies_df[movies_df.IMDB_Rating.notnull()==True])))
          alt.Chart(movies_df).mark_bar().encode(
                  x=alt.X('IMDB_Rating:Q',bin=alt.Bin(maxbins=n_bins),title='IMDB Rating'),
                  y='count(*):Q').display()
          movies_df.IMDB_Rating.describe()
```



Export as PNG View Source Open in Vega Editor

```
Out[211]: count    2990.000000
          mean        6.282809
          std         1.253096
          min         1.400000
          25%         5.600000
          50%         6.400000
          75%         7.200000
          max         9.200000
          Name: IMDB_Rating, dtype: float64
```



Back to top

In [212]:
```
#filling IMDB_Votes column missing data using imdb client
def get_imdb_votes(movie_title):
    '''this function will take the title of a movie and search in
            imdb client to find the votes of that movie base on
                    the title and the year'''
    # first of all we need to find the id of the movie by passing-
    # the title to search_for_title method
    movies = imdb.search_for_title(movie_title)
    movie_date = movies_df.loc[movies_df.Title == movie_title,'Release_Date'].to_
    regex = r"[?|$|.|!|,| |-|:|;|#|*|_|%|@|'|]"
    for movie_info in movies:
        movie_release_date = imdb.get_title_by_id(movie_info['imdb_id']).release_
        # to ensure accurucy we need to crossmatch the title of the movie and the
        if re.sub(regex,r'',movie_info['title'].lower()) == re.sub(regex,r'',movi
            if movie_release_date == movie_date:
                movie_votes = imdb.get_title_by_id(movie_info['imdb_id']).votes
                return movie_votes
            else:
                return np.nan
```

In [213]:
```
print('\n','''Before filling some of the missing values in the IMDB_Votes column,
        now there is {} missing data in IMDB_Votes column.'''.format(
    movies_df.IMDB_Votes.isnull().sum()))

print(movies_df.IMDB_Votes.head())
# applying the function for the first 10 records(rows) with missing values becaus
# it will talk along time to apply it to all the records(rows) in the columns

movies_votes_df = movies_df.loc[(movies_df.IMDB_Votes.isnull() == True)
                &
                (movies_df.Release_Date.isnull() == False),'Title'][:10].apply(ge
movies_df.loc[movies_df.IMDB_Votes.isnull() == True
                     &
                (movies_df.Release_Date.isnull() == False), 'IMDB_Votes'] = movie
# checking if the function works
print('\n','''After filling some of the missing values in the IMDB_Votes column,
        now there is {} missing data in IMDB_Votes column.'''.format(
    movies_df.IMDB_Votes.isnull().sum()))
print(movies_df.IMDB_Votes.head())
# check row number 3 (268)
```

```
 Before filling some of the missing values in the IMDB_Votes column,
        now there is 213 missing data in IMDB_Votes column.
0    1071.0
1     207.0
2     865.0
3       NaN
4     165.0
Name: IMDB_Votes, dtype: float64

 After filling some of the missing values in the IMDB_Votes column,
        now there is 211 missing data in IMDB_Votes column.
0    1071.0
1     207.0
2     865.0
3     268.0
4     165.0
Name: IMDB_Votes, dtype: float64
```

Back to top

In [214]:
```
# Checking the distribution of the data in IMDB_Rating column:
print('The minimum number of votes in a movie is {} and maximum is {}.'.format(
                        movies_df.IMDB_Votes.min(),movies_df.IMDB_Votes.max()))
print(movies_df.IMDB_Votes.describe())
```

```
The minimum number of votes in a movie is 5.0 and maximum is 519541.0.
count      2990.000000
mean      29899.216388
std       44924.988724
min           5.000000
25%        4836.250000
50%       15106.000000
75%       35784.250000
max      519541.000000
Name: IMDB_Votes, dtype: float64
```

In [215]:
```
#filling MPAA_Rating column missing data using imdb client
def get_mpaa_rating(movie_title):
    '''this function will take the title of a movie and search in
            imdb client to find the mpaa rating of that movie base on
                    the title and the year'''
     # first of all we need to find the id of the movie by passing-
    # the title to search_for_title method
    movies = imdb.search_for_title(movie_title)
    # then we need to get the release date of the movie that we want to lookup
    movie_date = movies_df.loc[movies_df.Title == movie_title,'Release_Date'].to_
    regex = r"[?|$|.|!|,| |-|:|;|#|*|_|%|@|'|]"
    for movie_info in movies:
        movie_release_date = imdb.get_title_by_id(movie_info['imdb_id']).release_
        # to ensure accurucy we need to crossmatch the title of the movie and the
        if re.sub(regex,r'',movie_info['title'].lower()) == re.sub(regex,r'',movi
            if movie_release_date == movie_date:
                movie_mpaa_rating = imdb.get_title_by_id(movie_info['imdb_id']).c
                return movie_mpaa_rating
            else:
                return np.nan
```

In [216]:
```
print('\n','''Before filling some of the missing values in the IMDB_Votes column,
        now there is {} missing data in MPAA_Rating column.'''.format(
    movies_df.MPAA_Rating.isnull().sum()))

print(movies_df.MPAA_Rating.head())

# applying the function for the first 10 records(rows) with missing values becaus
# it will talk along time to apply it to all the records(rows) in the columns
movies_mpaa_rating_df = movies_df.loc[(movies_df.MPAA_Rating.isnull() == True)
                &
                (movies_df.Release_Date.isnull() == False),'Title'][:10].apply(ge
movies_df.loc[movies_df.MPAA_Rating.isnull() == True
                     &
                (movies_df.Release_Date.isnull() == False), 'MPAA_Rating'] = movi
# checking if the function works
print('\n','''After filling some of the missing values in the IMDB_Votes column,
        now there is {} missing data in MPAA_Rating column.'''.format(
    movies_df.MPAA_Rating.isnull().sum()))
print(movies_df.MPAA_Rating.head())
# check row number 2 and 3 of column MPAA_Rating
```

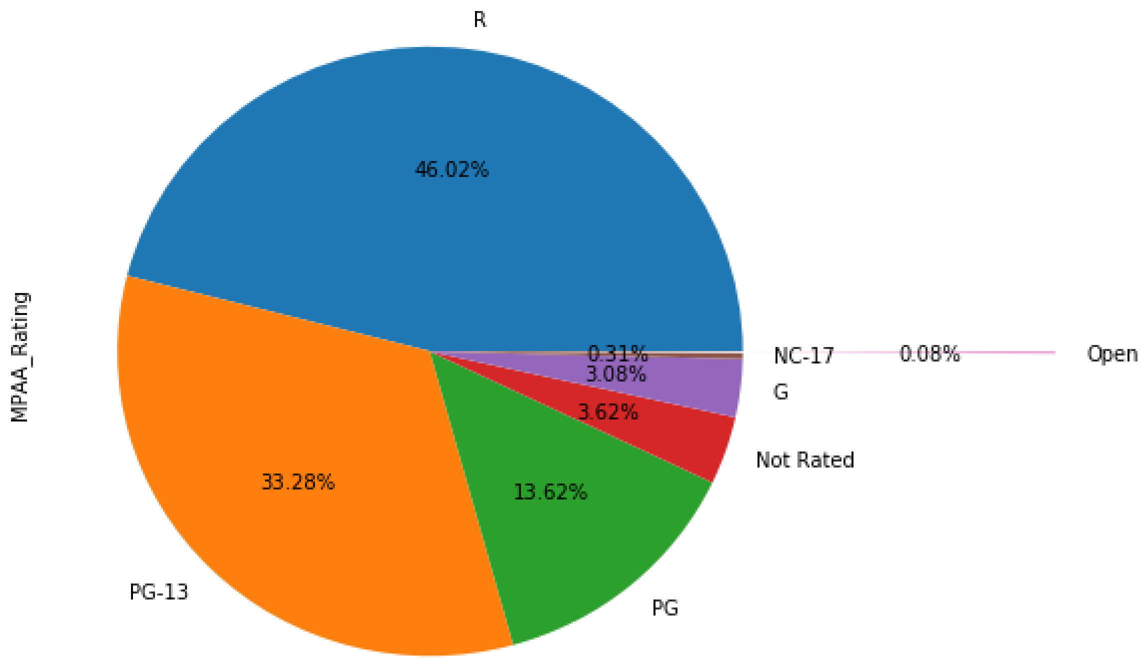
```
 Before filling some of the missing values in the IMDB_Votes column,
        now there is 605 missing data in MPAA_Rating column.
0       R
1       R
2    None
3    None
4       R
Name: MPAA_Rating, dtype: object

 After filling some of the missing values in the IMDB_Votes column,
        now there is 602 missing data in MPAA_Rating column.
0    R
1    R
2    R
3    R
4    R
Name: MPAA_Rating, dtype: object
```

Back to top

In [217]:
```
# checking for the distribution of the data of MPAA_Rating column:
print(movies_df.MPAA_Rating.unique())
movies_df.MPAA_Rating.value_counts().plot(kind='pie',autopct='%1.2f%%',figsize=[7
# I used the explode argument in plot to make sure that NC-17 and open labels don
# R rating is the most frequent MPAA rating in the data set
```

```
['R' nan 'G' None 'PG' 'Not Rated' 'PG-13' 'NC-17' 'Open']
```

Out[217]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x263d973a0f0>
```

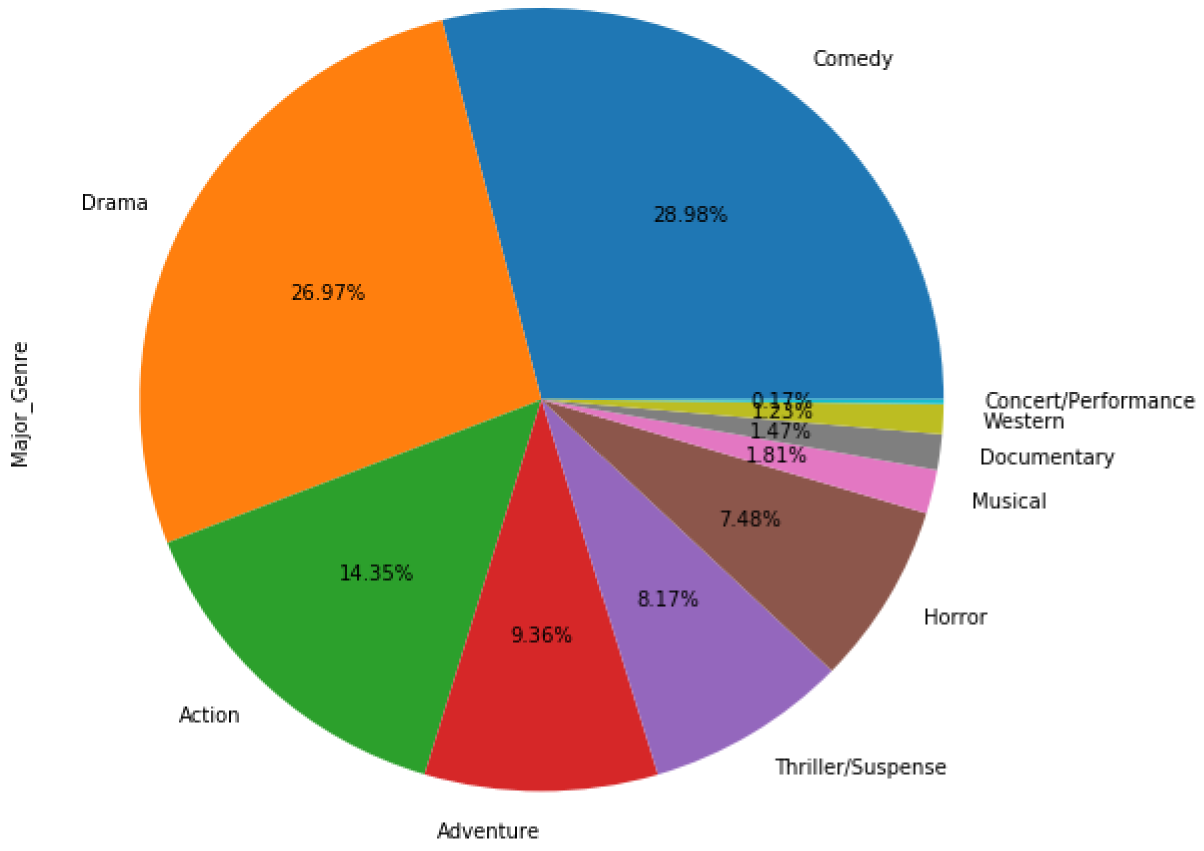


In [218]:
```
# checking for missing data in Major_Genre column:
print(movies_df.Major_Genre.isnull().sum())
```

```
275
```

In [219]:
```
# checking for the distribution of Major_Genre column:
# Since there is 3 kinds of comedy, why don't we add them together for simplicity
movies_df.Major_Genre = movies_df[movies_df.Major_Genre.notnull()].Major_Genre.ap
movies_df.Major_Genre.value_counts().plot(kind='pie',autopct='%1.2f%%', figsize=[
# Drama is the most frequent genre in the data set
print(movies_df.Major_Genre.unique())
```

```
[nan 'Drama' 'Comedy' 'Musical' 'Thriller/Suspense' 'Adventure' 'Action'
 'Horror' 'Western' 'Documentary' 'Concert/Performance']
```



Back to top

In [220]:
```
# checking for missing data in Production_Budget column:
print(movies_df.Production_Budget.isnull().sum())
```

```
1
```

In [221]:
```
# Checking the distribution of the data in Production_Budget column:

print('The minimum production budget in a movie is {}$ and maximum is {}$.'.forma
                        movies_df.Production_Budget.min(),movies_df.Production_Bu
# 218$ budget for a movie wow.
```

```
The minimum production budget in a movie is 218.0$ and maximum is 300000000.0$.
```

In [222]:
```
# checking for missing data in Release_Date column:
print(movies_df.Release_Date.isnull().sum())
```

```
7
```

In [223]:
```
# Checking the distribution of the data in Release_Date column:

print('The first movie was released on {} and last movie was released on {}.'.for
                        movies_df.Release_Date.min(),movies_df.Release_Date.max()
```

```
The first movie was released on 1920-09-17 00:00:00 and last movie was released
on 2016-12-24 00:00:00.
```

In [224]:
```
# checking for missing data in Rotten_Tomatoes_Rating column:
print(movies_df.Rotten_Tomatoes_Rating.isnull().sum())
```

```
880
```

Back to top

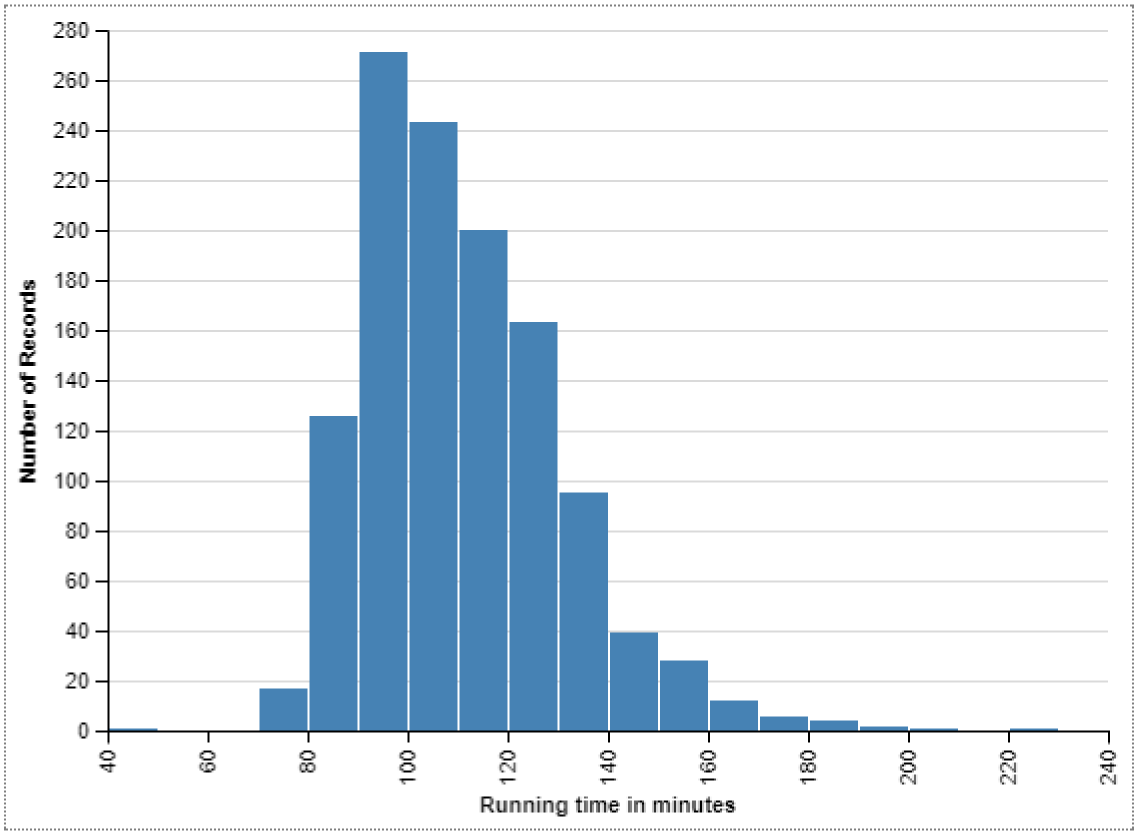
In [225]:
```
# Checking the distribution of the data in Rotten_Tomatoes_Rating column:

print('The movie with highest rotten tomatoes rating got {} and the movie with lo
                        movies_df.Rotten_Tomatoes_Rating.max(),movies_df.Rotten_T
```

```
The movie with highest rotten tomatoes rating got 100.0 and the movie with lowe
st rotten tomatoes rating got 1.0.
```

In [226]:
```
# checking for missing data in Running_Time_min column:
print(movies_df.Running_Time_min.isnull().sum())
```

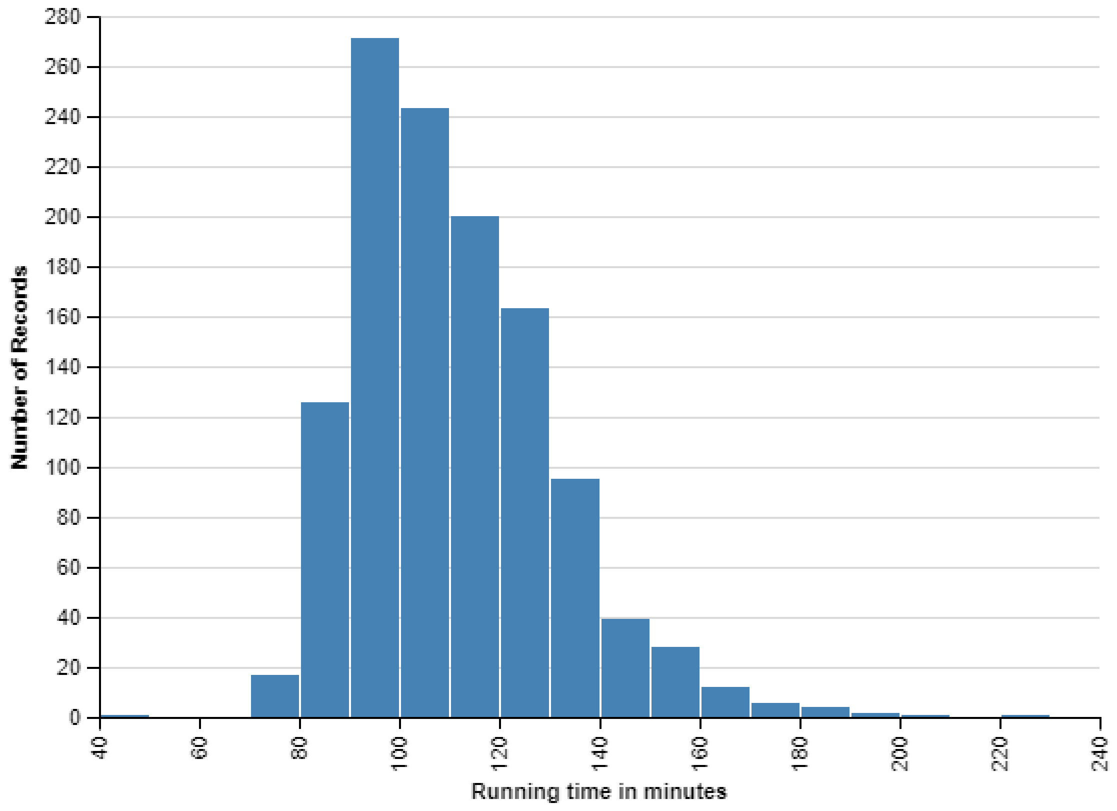
```
1992
```

```
In [227]: # Checking the distribution of the data in Running_Time_min column:
          n_bins = int(np.sqrt(len(movies_df[movies_df.Running_Time_min.notnull()==True])))
          alt.Chart(movies_df).mark_bar().encode(
                      x=alt.X('Running_Time_min:Q',bin=alt.Bin(maxbins=n_bins),title='Runni
          movies_df.Running_Time_min.describe()
```



Export as PNG View Source Open in Vega Editor

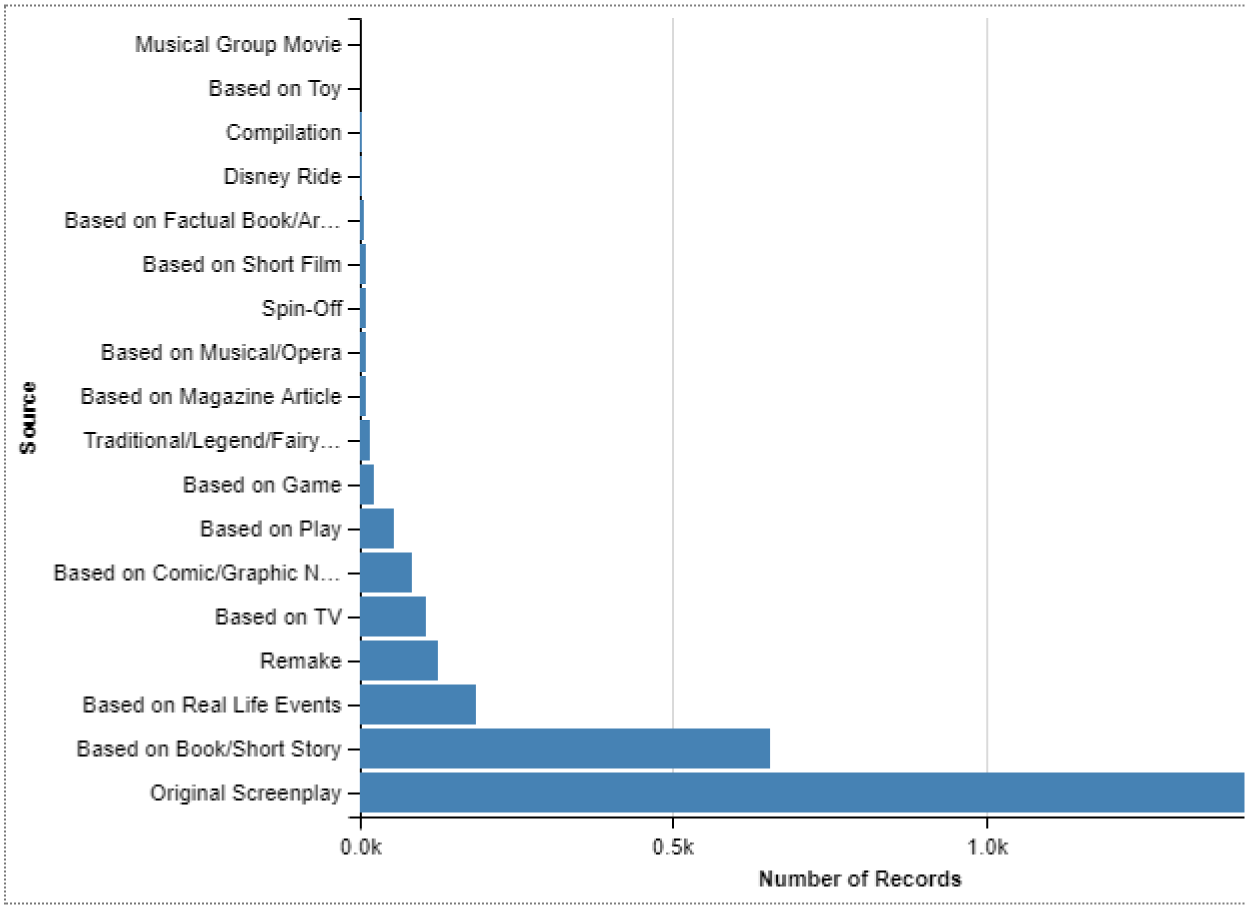
```
Out[227]: count    1209.000000
          mean      110.193548
          std        20.171014
          min        46.000000
          25%        95.000000
          50%       107.000000
          75%       121.000000
          max       222.000000
          Name: Running_Time_min, dtype: float64
```



In [228]:
```
# checking for missing data in Source column:
print(movies_df.Source.isnull().sum())
```

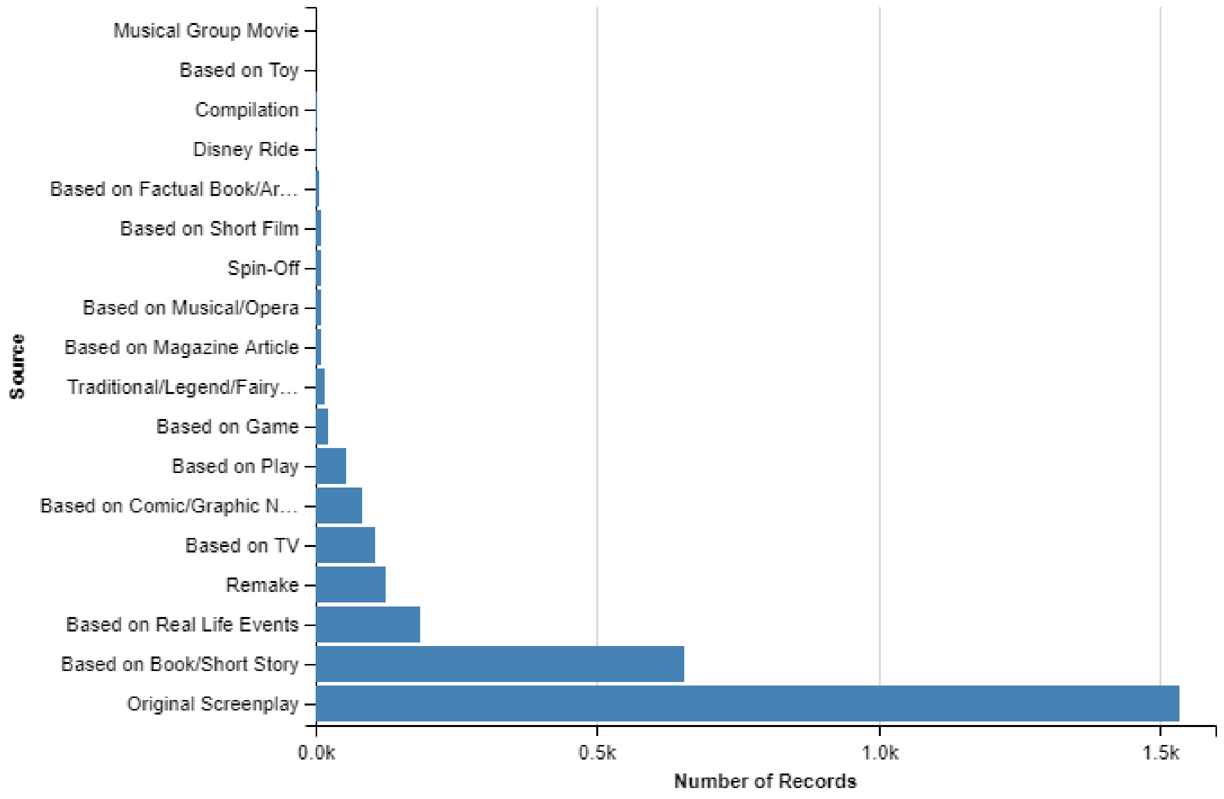
```
365
```

In [229]:
```
# checking for the distribution of Source column:
alt.Chart(movies_df[movies_df.Source.isnull()==False]).mark_bar().encode(
             y=alt.Y('Source:N',sort=alt.SortField(field='Source', op='count',orde
# Original Screenplay is the most frequent source in the data set
movies_df.Source.unique()
```



Export as PNG View Source Open in Vega Editor

Out[229]:
```
array([None, 'Original Screenplay', 'Based on Short Film', 'Based on Play',
       'Based on Book/Short Story', 'Remake',
       'Based on Comic/Graphic Novel', 'Based on Real Life Events',
       'Traditional/Legend/Fairytale', 'Based on TV', 'Compilation',
       'Based on Musical/Opera', 'Based on Game', 'Spin-Off',
       'Based on Factual Book/Article', 'Based on Magazine Article',
       'Disney Ride', 'Based on Toy', 'Musical Group Movie'], dtype=object)
```



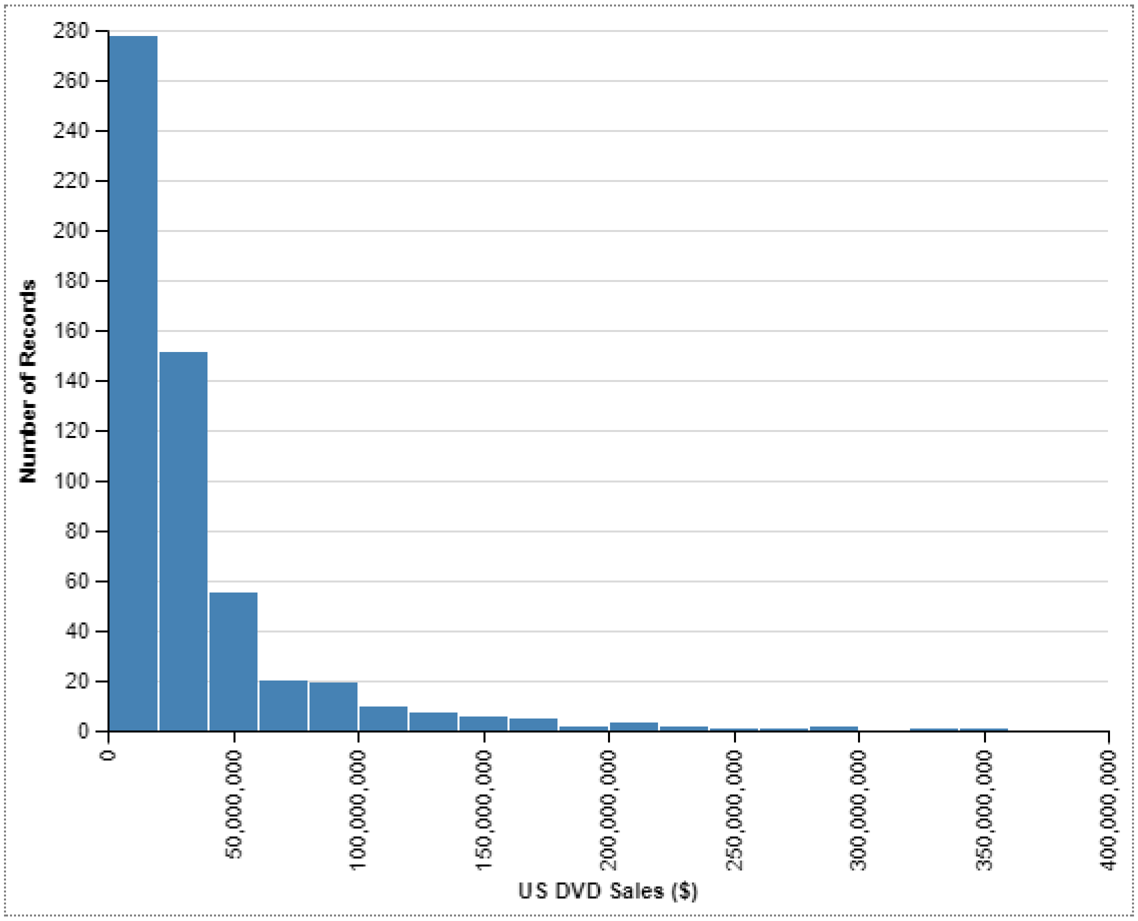
```
In [230]: # checking for missing data in Title column:
          print(movies_df.Title.isnull().sum())

          1
```

```
In [231]: # checking for the distribution of Source column:
          len(movies_df.Title.unique())

Out[231]: 3177
```

```
In [232]: # checking for missing data in Title column:
          print(movies_df.US_DVD_Sales.isnull().sum())

          2637
```

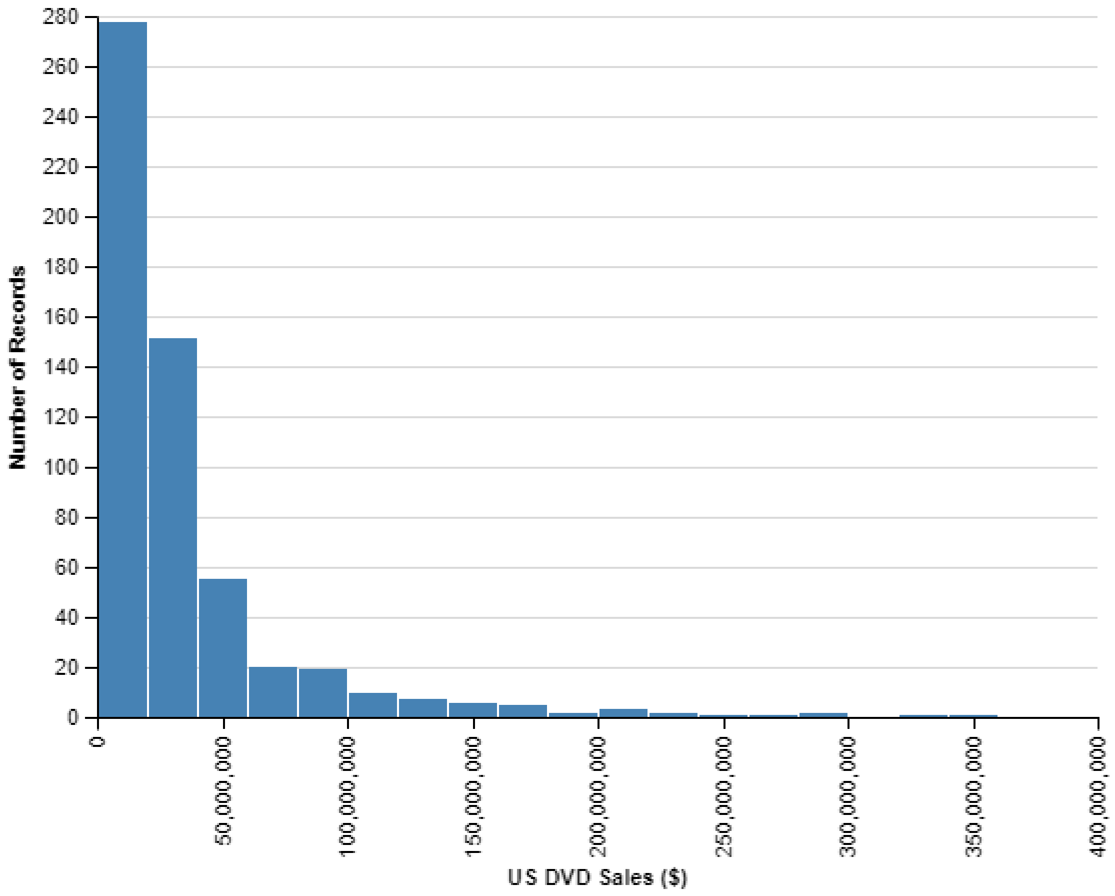
Back to top

In [233]:
```
# Checking the distribution of the data in US_DVD_Sales column:
n_bins = int(np.sqrt(len(movies_df[movies_df.US_DVD_Sales.notnull()==True])))
alt.Chart(movies_df).mark_bar().encode(
            x=alt.X('US_DVD_Sales:Q',bin=alt.Bin(maxbins=n_bins),title='US DVD Sa
movies_df.US_DVD_Sales.describe()
```



Export as PNG View Source Open in Vega Editor

Out[233]:
```
count    5.640000e+02
mean     3.490155e+07
std      4.589512e+07
min      6.184540e+05
25%      9.906211e+06
50%      2.033156e+07
75%      3.779422e+07
max      3.525821e+08
Name: US_DVD_Sales, dtype: float64
```



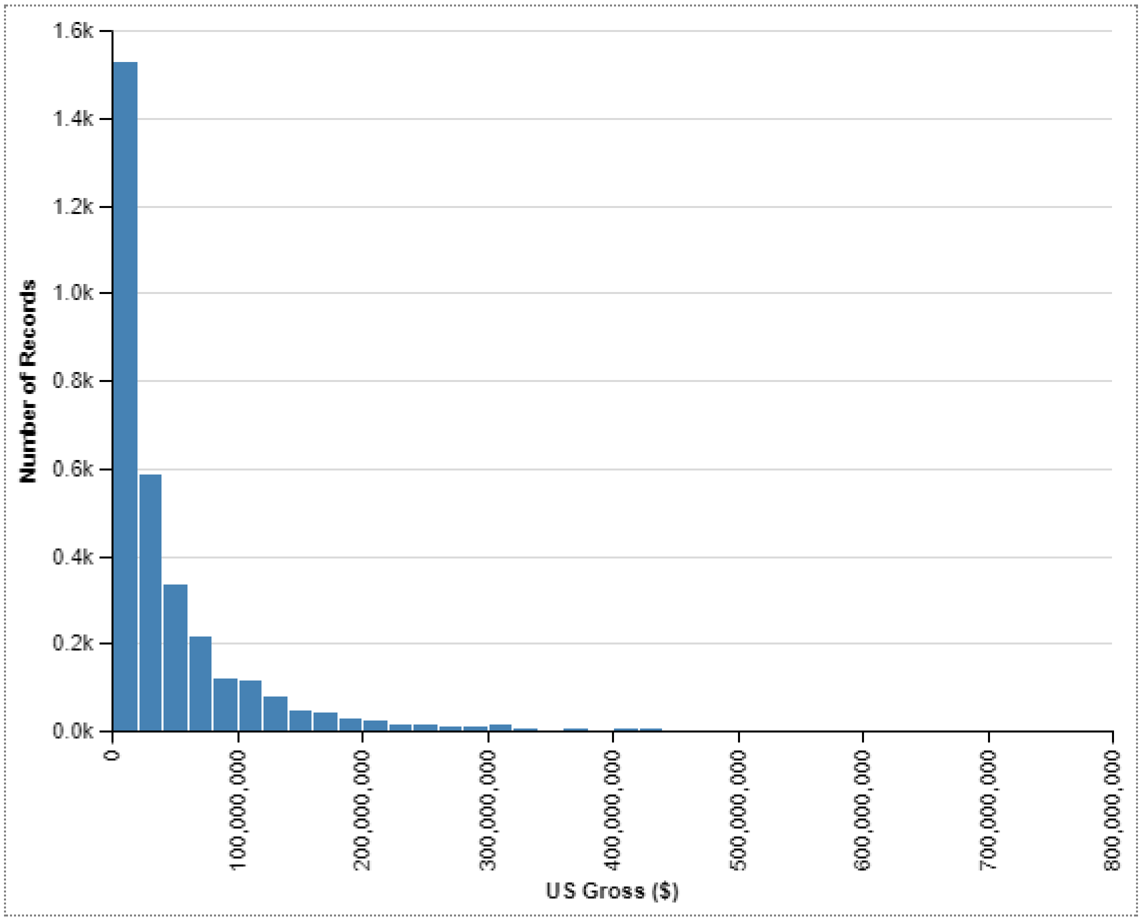
```
In [234]: print('The minimum US DVD sales is {}$ and maximum is {}$.'.format(
                                    movies_df.US_DVD_Sales.min(),movies_df.US_DVD_Sales.max()
```

```
The minimum US DVD sales is 618454.0$ and maximum is 352582053.0$.
```

```
In [235]: # checking for missing data in US_Gross column:
          print(movies_df.US_Gross.isnull().sum())
```

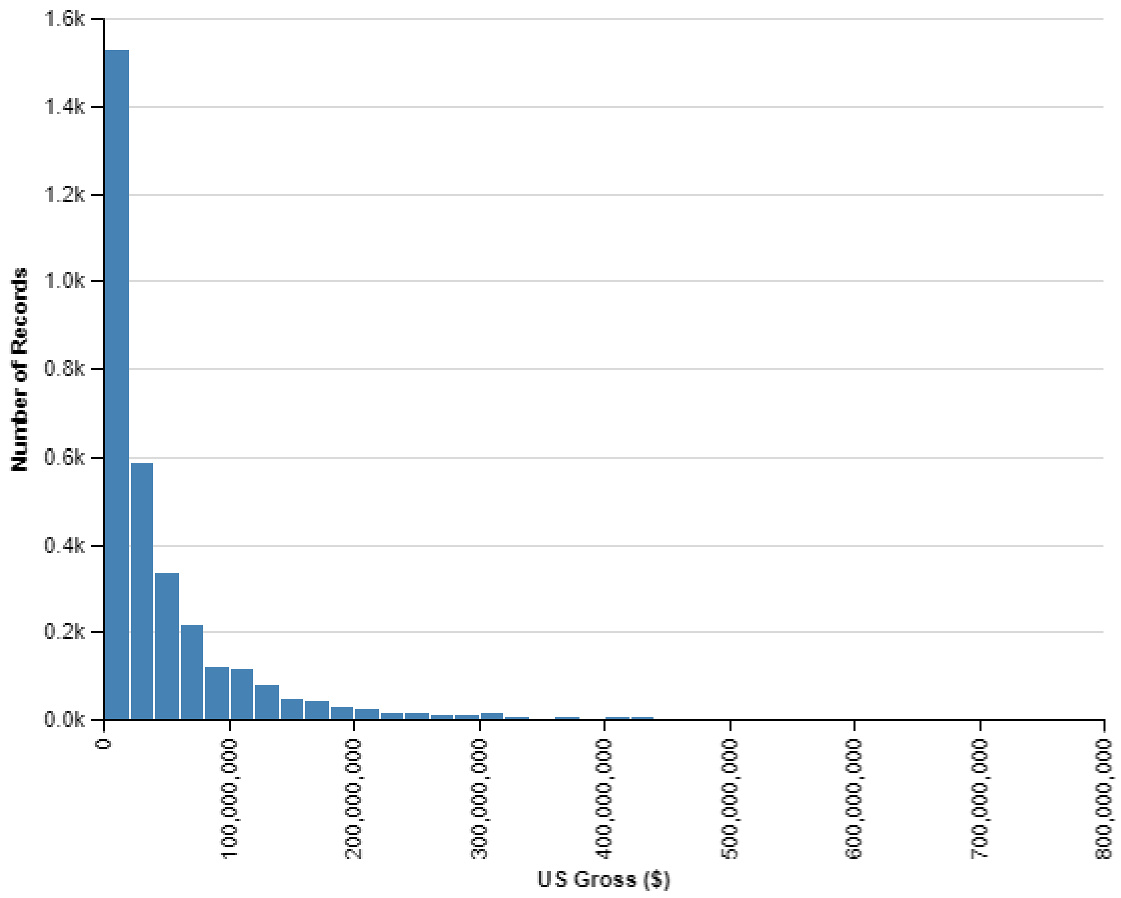
```
7
```

In [236]:
```
# Checking the distribution of the data in US_Gross column:
n_bins = int(np.sqrt(len(movies_df[movies_df.US_Gross.notnull()==True])))
alt.Chart(movies_df).mark_bar().encode(
            x=alt.X('US_Gross:Q',bin=alt.Bin(maxbins=n_bins),title='US Gross ($)'
movies_df.US_Gross.describe()
```



Export as PNG View Source Open in Vega Editor

Out[236]:
```
count    3.194000e+03
mean     4.400209e+07
std      6.255531e+07
min      0.000000e+00
25%      5.493221e+06
50%      2.201947e+07
75%      5.609176e+07
max      7.601676e+08
Name: US_Gross, dtype: float64
```



```
In [237]: print('The minimum US Gross sales is {}$ and maximum is {}$.'.format(
                                movies_df.US_Gross.min(),movies_df.US_Gross.max()))
```

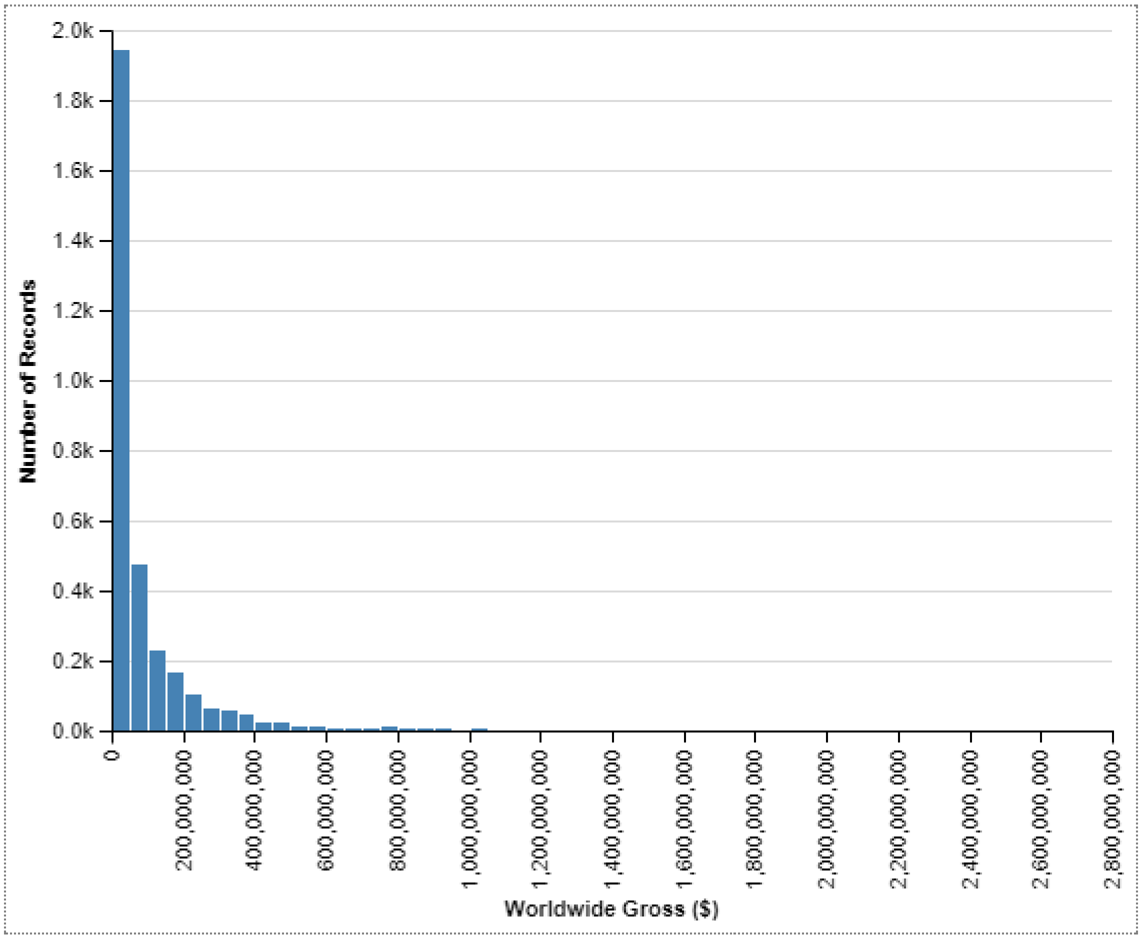
```
The minimum US Gross sales is 0.0$ and maximum is 760167650.0$.
```

Back to top

```
In [238]: # checking for missing data in Worldwide_Gross column:
          print(movies_df.Worldwide_Gross.isnull().sum())
```

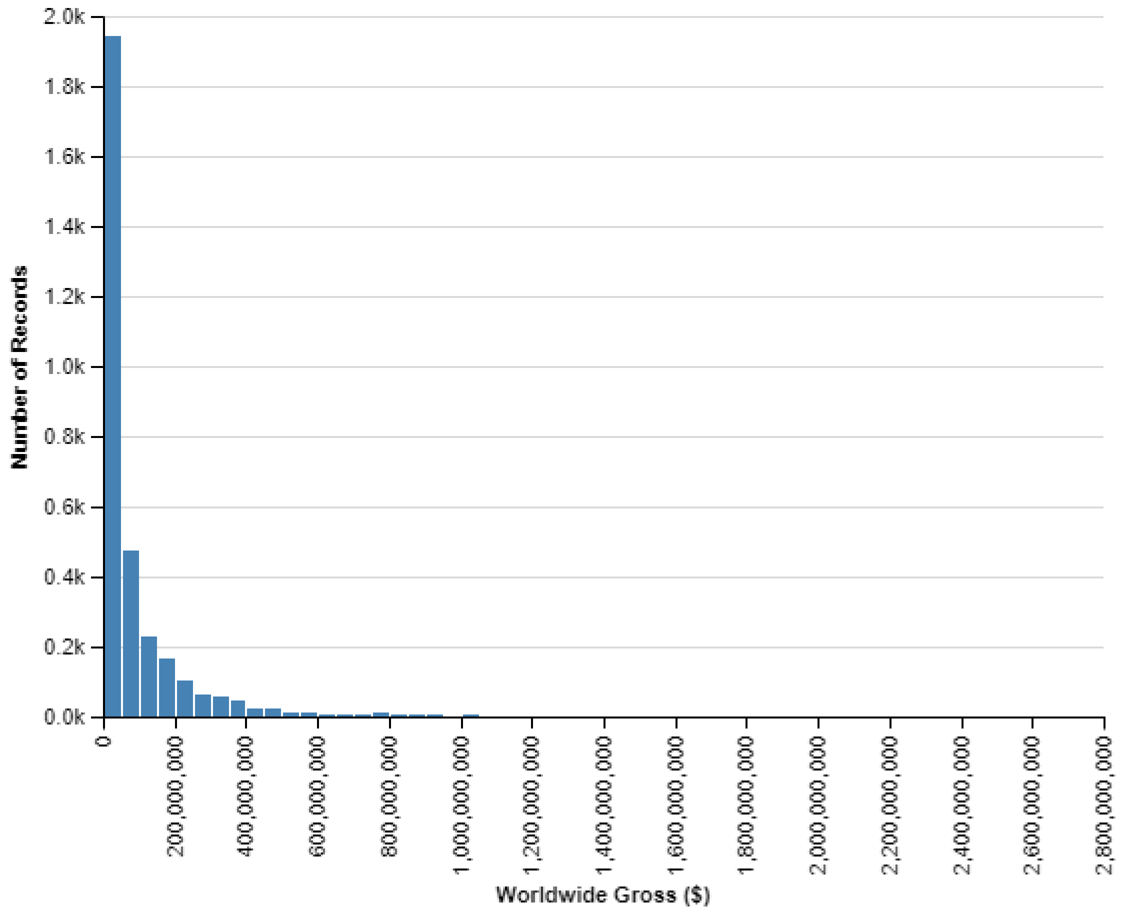
```
7
```

In [239]:
```
# Checking the distribution of the data in Worldwide_Gross column:
n_bins = int(np.sqrt(len(movies_df[movies_df.Worldwide_Gross.notnull()==True])))
alt.Chart(movies_df).mark_bar().encode(
            x=alt.X('Worldwide_Gross:Q',bin=alt.Bin(maxbins=n_bins),title='Worldw
movies_df.Worldwide_Gross.describe()
```



Export as PNG View Source Open in Vega Editor

Out[239]:
```
count    3.194000e+03
mean     8.534340e+07
std      1.499473e+08
min      0.000000e+00
25%      8.031285e+06
50%      3.116893e+07
75%      9.728380e+07
max      2.767891e+09
Name: Worldwide_Gross, dtype: float64
```

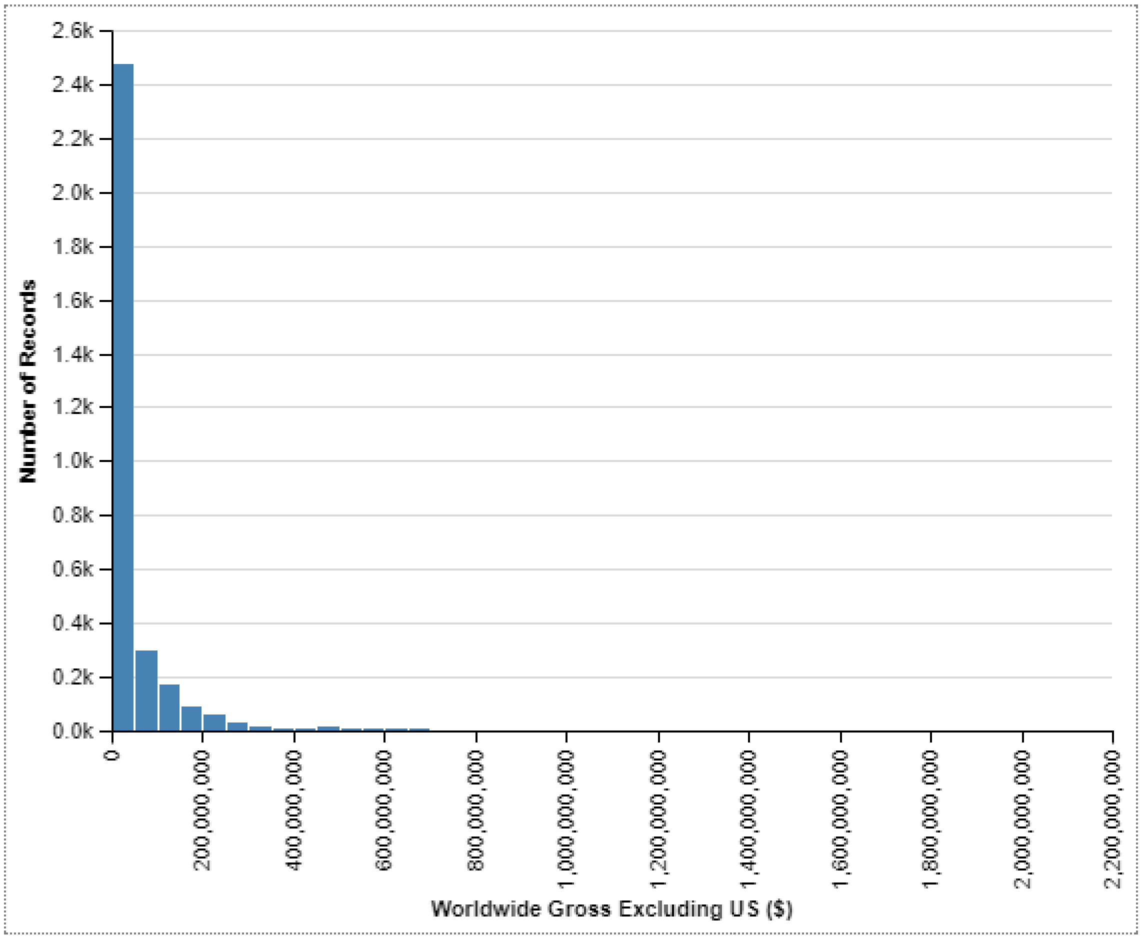


In [240]:
```
print('The minimum Worldwide Gross sales is {}$ and maximum is {}$.'.format(
                        movies_df.Worldwide_Gross.min(),movies_df.Worldwide_Gross
```

```
The minimum Worldwide Gross sales is 0.0$ and maximum is 2767891499.0$.
```

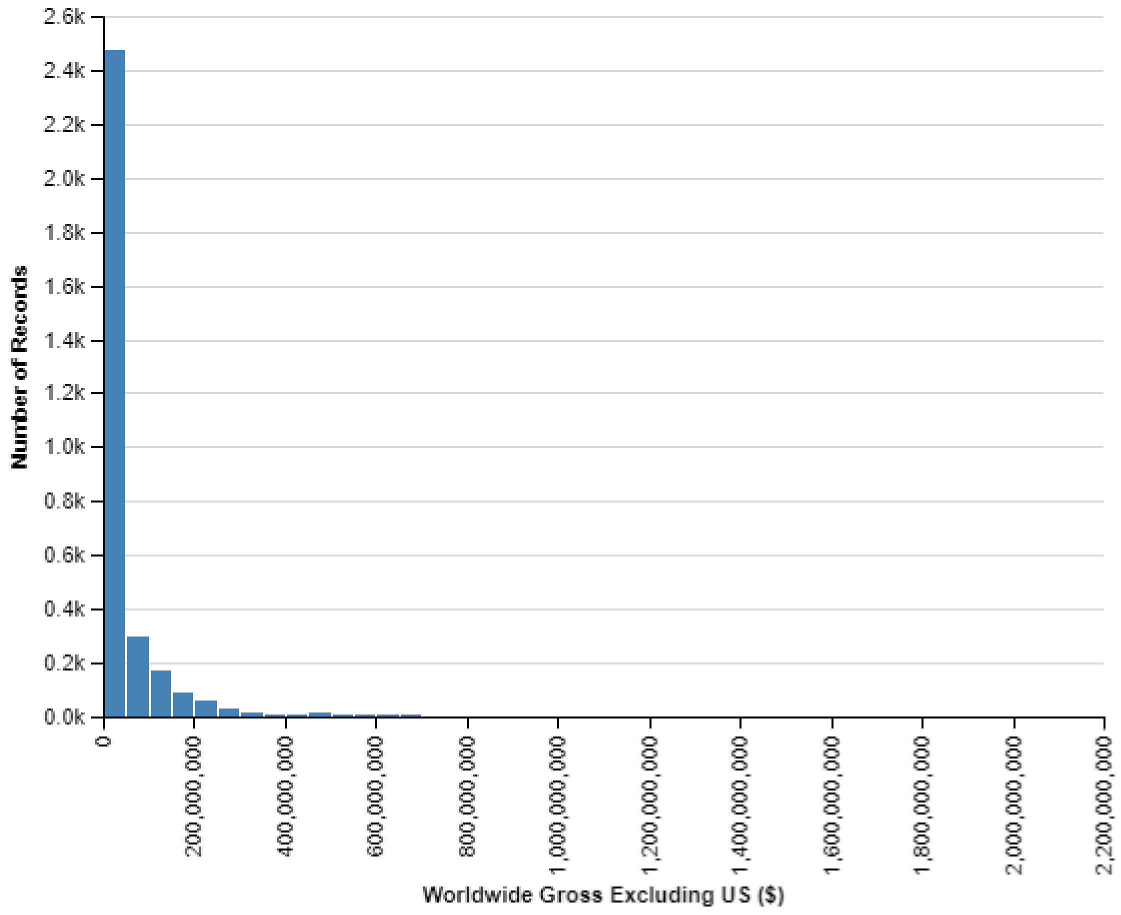
In [241]:
```
movies_df['Worldwide_Gross_Excluding_US'] = movies_df.Worldwide_Gross - movies_df
```

```
In [242]: # Checking the distribution of the data in Worldwide_Gross_Excluding_US column:
          n_bins = int(np.sqrt(len(movies_df[movies_df.Worldwide_Gross_Excluding_US.notnull
          alt.Chart(movies_df).mark_bar().encode(
                       x=alt.X('Worldwide_Gross_Excluding_US:Q',bin=alt.Bin(maxbins=n_bins),
          movies_df.Worldwide_Gross_Excluding_US.describe()
```



Export as PNG View Source Open in Vega Editor

```
Out[242]: count    3.194000e+03
          mean     4.134131e+07
          std      9.363655e+07
          min      0.000000e+00
          25%      0.000000e+00
          50%      2.689910e+06
          75%      4.269355e+07
          max      2.007724e+09
          Name: Worldwide_Gross_Excluding_US, dtype: float64
```



In [243]:
```
print('The minimum Worldwide Gross Excluding US is {}$ and maximum is {}$.'.forma
                        movies_df.Worldwide_Gross_Excluding_US.min(),movies_df.Wo
```

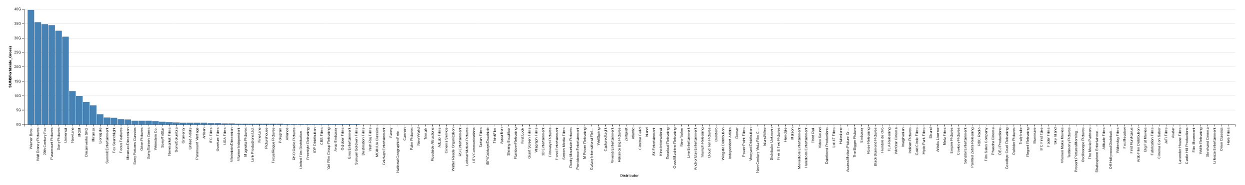
```
The minimum Worldwide Gross Excluding US is 0.0$ and maximum is 2007723849.0$.
```

In [244]:
```
movies_df.head(10)
```

Out[244]:

	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre	F
0	None	None	Gramercy	6.1	1071.0	R	NaN	
1	None	None	Strand	6.9	207.0	R	Drama	
2	None	None	Lionsgate	6.8	865.0	R	Comedy	
3	None	None	Fine Line	3.7	268.0	R	Comedy	
4	Contemporary Fiction	None	Trimark	3.4	165.0	R	Drama	
5	None	None	MGM	NaN	NaN	NaN	NaN	
6	None	Christopher Nolan	Zeitgeist	7.7	15133.0	R	NaN	
7	Contemporary Fiction	None	Artisan	3.8	353.0	R	Comedy	
8	None	Roman Polanski	None	5.8	3275.0	R	NaN	
9	None	None	None	7.0	2906.0	NaN	NaN	

In [245]:
```
alt.Chart(movies_df[movies_df.Distributor.isnull()==False]).mark_bar().encode(x=a
```

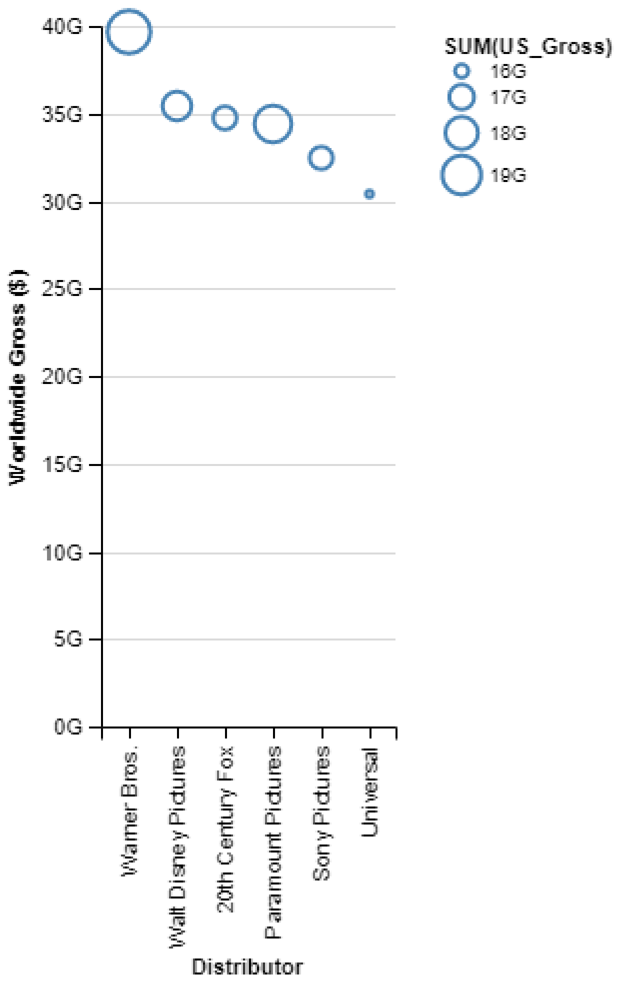


In [246]:
```
top_6_distributors_df = movies_df[movies_df.Distributor.isin([
    'Warner Bros.',
    'Walt Disney Pictures',
    '20th Century Fox',
    'Paramount Pictures',
    'Sony Pictures',
    'Universal'])]
top_6_distributors_df.head()
```

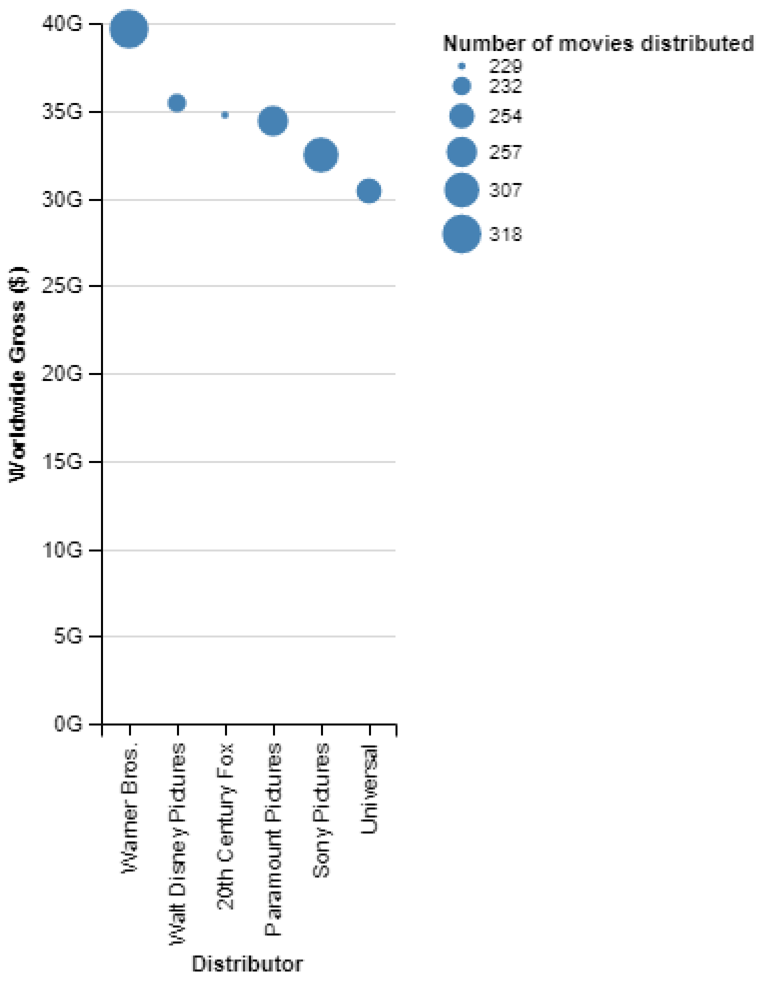
Out[246]:

	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre	Pr
11	None	None	Sony Pictures	7.5	9111.0	NaN	Musical	
12	None	None	Universal	8.4	82786.0	NaN	NaN	
20	Science Fiction	Terry Gilliam	Universal	8.1	169858.0	R	Drama	
22	Historical Fiction	Steven Spielberg	Universal	5.6	13364.0	NaN	Comedy	
25	None	Richard Fleischer	Walt Disney Pictures	NaN	NaN	NaN	Adventure	

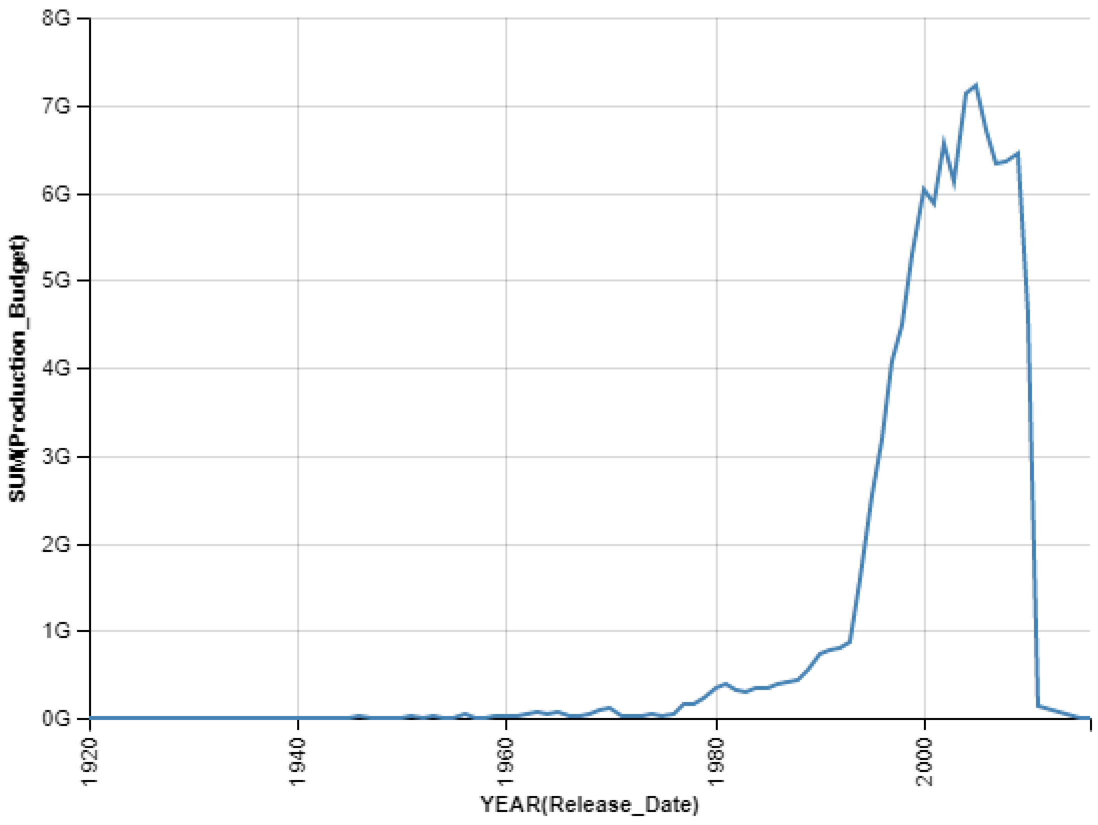
```
In [247]: alt.Chart(top_6_distributors_df).mark_point().encode(
              x=alt.X('Distributor',sort=alt.SortField(field='Worldwide_Gross', op='sum',or
              y=alt.Y('sum(Worldwide_Gross)',title='Worldwide Gross ($)'),size='sum(US_Gros
```



```
In [248]: alt.Chart(top_6_distributors_df).mark_circle().encode(
              x=alt.X('Distributor',sort=alt.SortField(field='Worldwide_Gross', op='sum',or
              y=alt.Y('sum(Worldwide_Gross)',title='Worldwide Gross ($)'),size=alt.Size('co
```



```
In [249]: alt.Chart(movies_df).mark_line().encode(x=alt.X('Release_Date:T',timeUnit='year')
```



```
In [250]: movies_df['Mean_IMDB_Rating'] = movies_df.IMDB_Rating.mean()
```

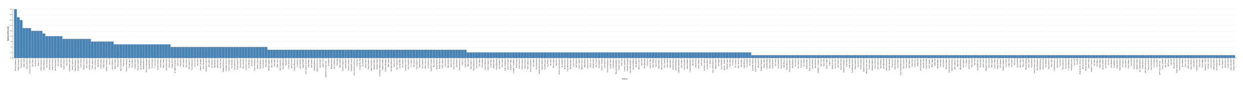
```
In [251]: movies_df.head()
```

Out[251]:

	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre	Pro
0	None	None	Gramercy	6.1	1071.0	R	NaN	
1	None	None	Strand	6.9	207.0	R	Drama	
2	None	None	Lionsgate	6.8	865.0	R	Comedy	
3	None	None	Fine Line	3.7	268.0	R	Comedy	
4	Contemporary Fiction	None	Trimark	3.4	165.0	R	Drama	

```
In [252]: movies_rating_above_mean = movies_df[(movies_df.IMDB_Rating > movies_df.Mean_IMDB_
```

```
In [253]: alt.Chart(movies_rating_above_mean).mark_bar().encode(
                                    x = alt.X('Director',sort=alt.SortField(field='Direct
```



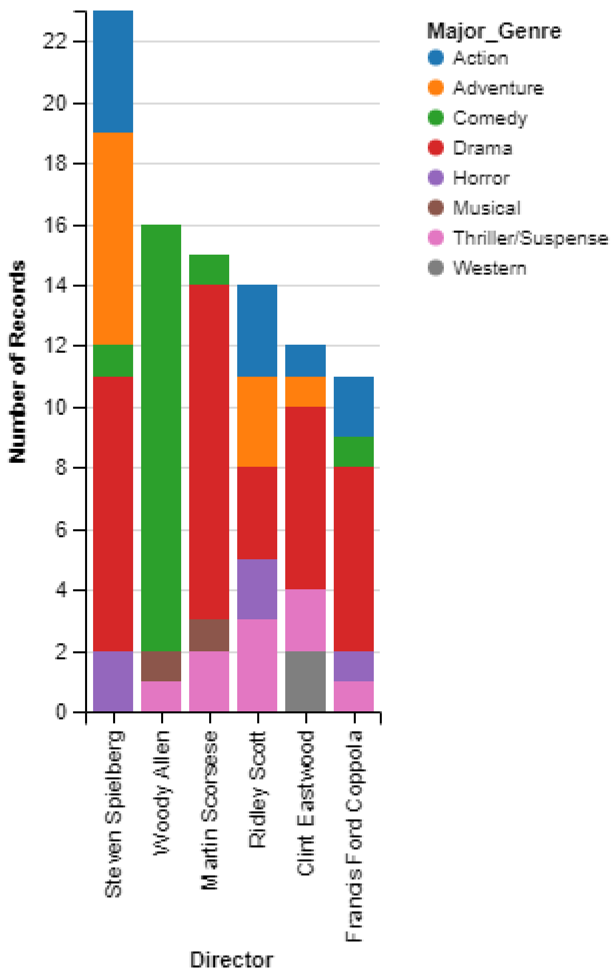
```
In [254]: top_6_directors_df = movies_df[movies_df.Director.isin(['Steven Spielberg',
                                                    'Martin Scorsese',
                                                    'Woody Allen',
                                                    'Ridley Scott',
                                                    'Clint Eastwood',
                                                    'Francis Ford Coppola'])]
```

```
In [255]: top_6_directors_df.head()
```

Out[255]:

	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre
22	Historical Fiction	Steven Spielberg	Universal	5.6	13364.0	NaN	Comedy
57	None	Woody Allen	MGM	8.2	65406.0	NaN	Comedy
61	Historical Fiction	Francis Ford Coppola	MGM	8.6	173141.0	R	Action
109	Science Fiction	Ridley Scott	Warner Bros.	8.3	185546.0	R	Thriller/Suspense
118	None	Woody Allen	MGM	7.1	12415.0	PG-13	Comedy

```
In [256]: alt.Chart(top_6_directors_df[top_6_directors_df.Major_Genre.notnull()]).mark_bar(
                           x=alt.X('Director',sort=alt.SortField(field='Director',op='count'
                           y='count(*)',color='Major_Genre')
```

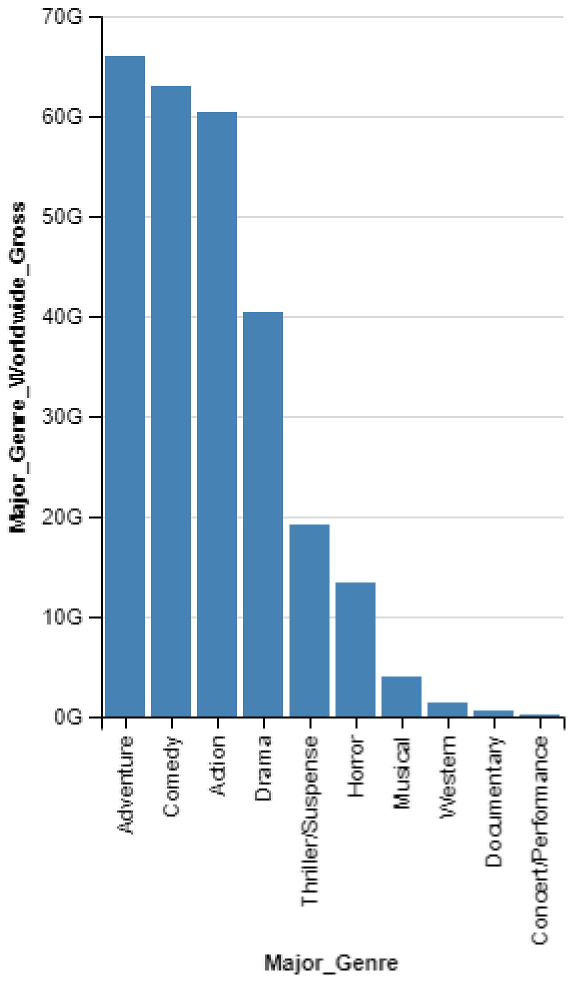


```
In [257]: movies_df['Major_Genre_Worldwide_Gross']=movies_df.groupby('Major_Genre')['Worldw
```

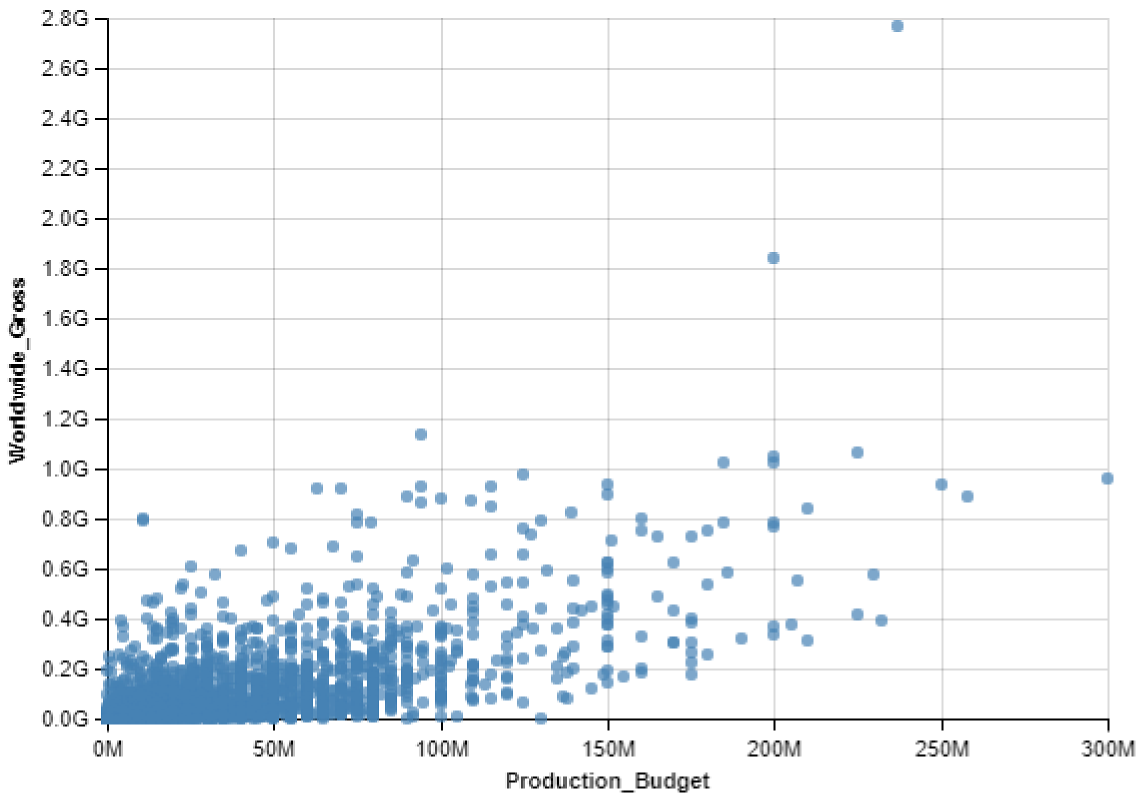
```
In [258]: print('The minimum Worldwide Gross sales is {}$ and maximum is {}$.'.format(
                                     movies_df.Major_Genre_Worldwide_Gross.min(),movies_df.Maj
```

```
The minimum Worldwide Gross sales is 153622009.0$ and maximum is 66080959632.0
$.
```

In [259]:
```
alt.Chart(movies_df).mark_bar().encode(
x= alt.X('Major_Genre',sort=alt.SortField(field='Major_Genre_Worldwide_Gross',op=
y='Major_Genre_Worldwide_Gross')
```



In [260]:
```
alt.Chart(movies_df).mark_circle().encode(x='Production_Budget',
                                          y='Worldwide_Gross')
#we can use log to spread out the clustering
```



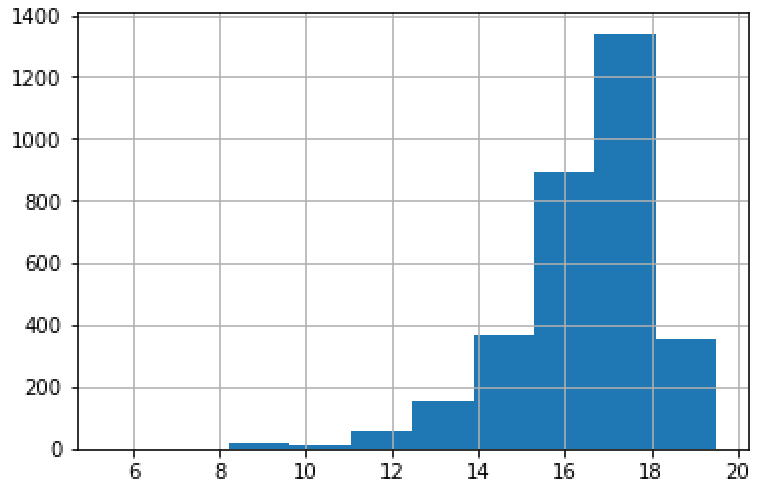
In [261]:
```
movies_df['Production_Budget_1'] = movies_df.Production_Budget + 1
movies_df['Worldwide_Gross_1']= movies_df.Worldwide_Gross + 1
movies_df['Log_Production_Budget'] = movies_df.Production_Budget_1.apply(np.log)
movies_df['Log_Worldwide_Gross'] = movies_df.Worldwide_Gross_1.apply(np.log)
movies_df.head()
```

Out[261]:

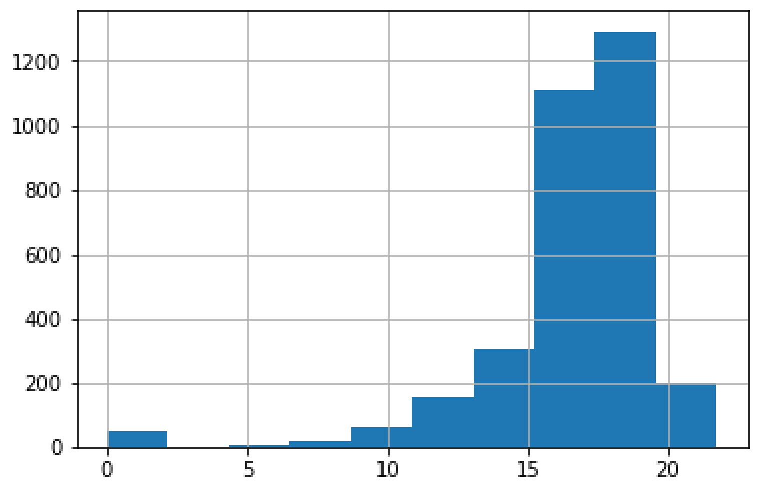
	Creative_Type	Director	Distributor	IMDB_Rating	IMDB_Votes	MPAA_Rating	Major_Genre	Pro
0	None	None	Gramercy	6.1	1071.0	R	NaN	
1	None	None	Strand	6.9	207.0	R	Drama	
2	None	None	Lionsgate	6.8	865.0	R	Comedy	
3	None	None	Fine Line	3.7	268.0	R	Comedy	
4	Contemporary Fiction	None	Trimark	3.4	165.0	R	Drama	

5 rows × 23 columns

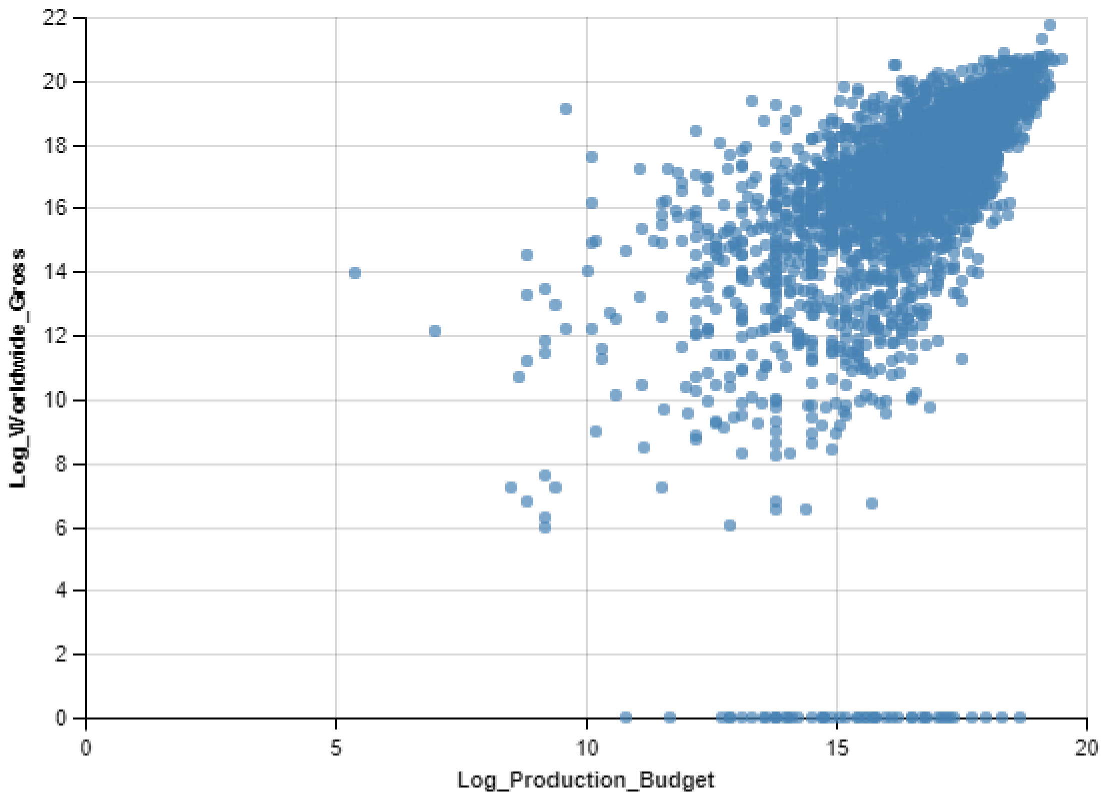
```
In [262]: #Checking the distribution of Log_Production_Budget column
          movies_df.Log_Production_Budget.hist()

Out[262]: <matplotlib.axes._subplots.AxesSubplot at 0x263db2a1c50>
```



```
In [263]: #Checking the distribution of Log_Worldwide_Gross column
          movies_df.Log_Worldwide_Gross.hist()

Out[263]: <matplotlib.axes._subplots.AxesSubplot at 0x263db2b5d30>
```



In [264]:
```
alt.Chart(movies_df).mark_circle().encode(x='Log_Production_Budget',
                                          y='Log_Worldwide_Gross')
```



In [265]:
```
movies_df.Production_Budget.corr(movies_df.Worldwide_Gross)
#Positive relationship between production budget and worldwide gross
```

Out[265]:
```
0.66577953300611326
```

In [ ]: