JJenkins_HW#3

October 1, 2018

1 Question 1

PART 1: (1 point)

Load the MovieData.csv dataset into a pandas DataFrame as described in this week's lesson, and use it to find the following values:

a. What is the median profit of movies with budgets of over $50M?

b. How many movies were released by each film distributor? Output the results to a csv file.

PART 2: (6 points)

c. What are the mean and median movie profits by decade? Which decade was the most profitable? (Hint: Answering this question requires several steps: grouping the movies by decade, computing the mean and median profits for each decade, and combining the results back together.)

```
In [1]: import pandas as pd
        import numpy as np
        import datetime as dt
        %matplotlib inline
```

```
In [2]: def make_date(date_str):
            '''
            Turn a MM/DD/YY string into a datetime object
            '''
            m, d, y = date_str.split("/")
            m = int(m)
            d = int(d)
            y = int(y)
            if y > 13:
                y += 1900
            else:
                y += 2000
            return dt.datetime(y, m, d)
```

```
In [3]: movies = pd.read_csv("MovieData.csv", sep='\t', na_values=["Unknown", "Unkno"],
                               parse_dates=[0], date_parser=make_date)
```

```
In [4]: movies.head()
```

```
Out[4]:   Release_Date                                    Movie  Distributor  \
        0   2012-03-09                              John Carter          NaN
        1   2007-05-25  Pirates of the Caribbean: At World's End  Buena Vista
        2   2013-12-13          The Hobbit: There and Back Again     New Line
        3   2012-12-14         The Hobbit: An Unexpected Journey     New Line
        4   2010-11-24                                  Tangled  Buena Vista

              Budget     US Gross  Worldwide Gross
        0  300000000   66439100.0      254439100.0
        1  300000000  309420425.0      960996492.0
        2  270000000          NaN              NaN
        3  270000000          NaN              NaN
        4  260000000  200821936.0      586581936.0
```

```
In [5]: # Replace missing values with zeros
        movies.fillna(0, inplace=True)
```

```
In [6]: movies.head()
```

```
Out[6]:   Release_Date                                    Movie  Distributor  \
        0   2012-03-09                              John Carter            0
        1   2007-05-25  Pirates of the Caribbean: At World's End  Buena Vista
        2   2013-12-13          The Hobbit: There and Back Again     New Line
        3   2012-12-14         The Hobbit: An Unexpected Journey     New Line
        4   2010-11-24                                  Tangled  Buena Vista

              Budget     US Gross  Worldwide Gross
        0  300000000   66439100.0      254439100.0
        1  300000000  309420425.0      960996492.0
        2  270000000          0.0              0.0
        3  270000000          0.0              0.0
        4  260000000  200821936.0      586581936.0
```

```
In [7]: print("The date of the oldest movie in the dataset is %r." % min(movies["Release_Date"]
        print("The date of the newest movie in the dataset is %r." % max(movies["Release_Date"]
```

```
The date of the oldest movie in the dataset is Timestamp('1915-02-08 00:00:00').
The date of the newest movie in the dataset is Timestamp('2013-12-13 00:00:00').
```

```
In [8]: # Fill in Worldwide Gross when it is zero
        movies["Worldwide Gross"][movies["Worldwide Gross"]==0] = movies["US Gross"]
        #movies["US Gross"][movies["US Gross"]==0] = movies["Worldwide Gross"]
```

```
C:\Users\jenkij\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:2: Se
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [9]: movies["Profits"] = movies["Worldwide Gross"] - movies["Budget"]
        movies.head()
```

Out[9]:

	Release_Date	Movie	Distributor	\
0	2012-03-09	John Carter	0	
1	2007-05-25	Pirates of the Caribbean: At World's End	Buena Vista	
2	2013-12-13	The Hobbit: There and Back Again	New Line	
3	2012-12-14	The Hobbit: An Unexpected Journey	New Line	
4	2010-11-24	Tangled	Buena Vista	

	Budget	US Gross	Worldwide Gross	Profits
0	300000000	66439100.0	254439100.0	-45560900.0
1	300000000	309420425.0	960996492.0	660996492.0
2	270000000	0.0	0.0	-270000000.0
3	270000000	0.0	0.0	-270000000.0
4	260000000	200821936.0	586581936.0	326581936.0

```
In [10]: #movies[(movies.Budget > 50000000) & (movies["US Gross"].notnull())]
         bigger_budget = movies[movies.Budget > 50000000]

In [11]: #movies[(movies.Budget > 50000000)].median()
         bigger_budget.Profits.median()
```

Out[11]: 89246220.0

1.0.1 Answer to 1 a.

a. What is the median profit of movies with budgets of over 50M?

```
In [12]: print("The median profit of movies with budgets of over $50M is %d." %
              bigger_budget.Profits.median())

The median profit of movies with budgets of over $50M is 89246220.
```

```
In [13]: distributors = movies.groupby("Distributor").aggregate(len)
         distributor_count = distributors["Movie"]
         distributor_count
```

Out[13]:

```
Distributor
0                                659
20th Century Fox                 230
3D Entertainment                   1
8 X Entertainment                  1
ART                                1
Access                             1
```

```
Alliance                              4
American International Pictures       1
Anchor Bay                            4
Apparition                            4
Artisan                              23
Artistic License                      1
Atlantic                              1
Attitude Films                        1
Avatar                                1
Avco Embassy                          5
Barking Cow                           1
Big Pictures                          1
Bigger Picture                        1
Black Diamond Pictures                1
Buena Vista                         227
CBS Films                             3
CFP                                   1
CHRIST                                1
Cannon                                4
Cinema Service                        1
Cinema con Sabor                      1
Cloud Ten Pictures                    1
Columbia                             26
Consolidated Pictures Group           1
                                    ...
Third Rail                            2
TriStar Pictures                      4
Trimark                               9
Triumph                               1
Truly Indie                           1
USA Films                            15
United Artists                       23
United Film Distribution              2
Universal                           261
Universal/Arenas Entertainment        1
Universal/Rogue                       1
Videos                                1
Vitagraph Films                       2
Walt Disney Co.                       9
Walt Disney Pictures                  1
Warner Bros.                        311
Warner Independent                    3
Warner Independent Pictures           7
Weinstein                             1
Weinstein Ci.                         1
Weinstein Co.                        33
Weinstein/Dimension                   1
Weintraub                             2
```

```
WellSpring                          2
Wellspring                          1
WinStar                             1
Winstar                             1
Yash Raj                            1
Zeitgeist                           7
Zion                                1
Name: Movie, Length: 209, dtype: int64
```

1.0.2 Answer to Question 1 b.

b. How many movies were released by each film distributor?

1.1 NOTE: See below for outputting of the results to a csv file.

```
In [14]: print("The number of movies released by each film distributor is shown below:")
         print(distributor_count)
```

```
The number of movies released by each film distributor is shown below:
Distributor
0                                 659
20th Century Fox                  230
3D Entertainment                    1
8 X Entertainment                   1
ART                                 1
Access                              1
Alliance                            4
American International Pictures     1
Anchor Bay                          4
Apparition                          4
Artisan                            23
Artistic License                    1
Atlantic                            1
Attitude Films                      1
Avatar                              1
Avco Embassy                        5
Barking Cow                         1
Big Pictures                        1
Bigger Picture                      1
Black Diamond Pictures              1
Buena Vista                       227
CBS Films                           3
CFP                                 1
CHRIST                              1
Cannon                              4
Cinema Service                      1
Cinema con Sabor                    1
Cloud Ten Pictures                  1
```

```
Columbia                              26
Consolidated Pictures Group            1
                                     ...
Third Rail                             2
TriStar Pictures                       4
Trimark                                9
Triumph                                1
Truly Indie                            1
USA Films                             15
United Artists                        23
United Film Distribution               2
Universal                            261
Universal/Arenas Entertainment         1
Universal/Rogue                        1
Videos                                 1
Vitagraph Films                        2
Walt Disney Co.                        9
Walt Disney Pictures                   1
Warner Bros.                         311
Warner Independent                     3
Warner Independent Pictures            7
Weinstein                              1
Weinstein Ci.                          1
Weinstein Co.                         33
Weinstein/Dimension                    1
Weintraub                              2
WellSpring                             2
Wellspring                             1
WinStar                                1
Winstar                                1
Yash Raj                               1
Zeitgeist                              7
Zion                                   1
Name: Movie, Length: 209, dtype: int64
```

```
In [15]: type(distributor_count)
```

```
Out[15]: pandas.core.series.Series
```

```
In [16]: data = pd.DataFrame(distributor_count)
         data.head()
```

```
Out[16]:                    Movie
         Distributor
         0                    659
         20th Century Fox     230
         3D Entertainment       1
```

```
8 X Entertainment      1
ART                    1
```

In [17]:
```
data.index
```

Out[17]:
```
Index([                            0,                '20th Century Fox',
                '3D Entertainment',                '8 X Entertainment',
                             'ART',                            'Access',
                       'Alliance',  'American International Pictures',
                     'Anchor Bay',                       'Apparition',
       ...
                   'Weinstein Co.',              'Weinstein/Dimension',
                       'Weintraub',                       'WellSpring',
                     'Wellspring',                          'WinStar',
                        'Winstar',                        'Yash Raj',
                       'Zeitgeist',                             'Zion'],
      dtype='object', name='Distributor', length=209)
```

In [18]:
```
data.columns = ["Counts"]
data
```

Out[18]:

Distributor	Counts
0	659
20th Century Fox	230
3D Entertainment	1
8 X Entertainment	1
ART	1
Access	1
Alliance	4
American International Pictures	1
Anchor Bay	4
Apparition	4
Artisan	23
Artistic License	1
Atlantic	1
Attitude Films	1
Avatar	1
Avco Embassy	5
Barking Cow	1
Big Pictures	1
Bigger Picture	1
Black Diamond Pictures	1
Buena Vista	227
CBS Films	3
CFP	1
CHRIST	1
Cannon	4
Cinema Service	1

```
Cinema con Sabor                      1
Cloud Ten Pictures                    1
Columbia                             26
Consolidated Pictures Group           1
...                                 ...
Third Rail                            2
TriStar Pictures                      4
Trimark                               9
Triumph                               1
Truly Indie                           1
USA Films                            15
United Artists                       23
United Film Distribution              2
Universal                           261
Universal/Arenas Entertainment        1
Universal/Rogue                       1
Videos                                1
Vitagraph Films                       2
Walt Disney Co.                       9
Walt Disney Pictures                  1
Warner Bros.                        311
Warner Independent                    3
Warner Independent Pictures           7
Weinstein                             1
Weinstein Ci.                         1
Weinstein Co.                        33
Weinstein/Dimension                   1
Weintraub                             2
WellSpring                            2
Wellspring                            1
WinStar                               1
Winstar                               1
Yash Raj                              1
Zeitgeist                             7
Zion                                  1

[209 rows x 1 columns]
```

1.1.1 Outputting the results to a csv file.

```
In [19]: data.to_csv('hw3_1b.csv', sep=',') # Outputs the results to a csv file.
```

What are the mean and median movie profits by decade? Which decade was the most profitable? (Hint: Answering this question requires several steps: grouping the movies by decade, computing the mean and median profits for each decade, and combining the results back together.)

```
In [20]: movies.Release_Date[19].year
```

```
Out[20]: 2005

In [21]: movies["Year"] = movies.Release_Date.apply(lambda x: x.year)
         movies.head()

Out[21]:   Release_Date                                    Movie  Distributor  \
         0   2012-03-09                              John Carter            0
         1   2007-05-25  Pirates of the Caribbean: At World's End  Buena Vista
         2   2013-12-13          The Hobbit: There and Back Again     New Line
         3   2012-12-14         The Hobbit: An Unexpected Journey     New Line
         4   2010-11-24                                  Tangled  Buena Vista

               Budget     US Gross  Worldwide Gross      Profits  Year
         0  300000000   66439100.0      254439100.0  -45560900.0  2012
         1  300000000  309420425.0      960996492.0  660996492.0  2007
         2  270000000          0.0              0.0 -270000000.0  2013
         3  270000000          0.0              0.0 -270000000.0  2012
         4  260000000  200821936.0      586581936.0  326581936.0  2010

In [22]: print(movies.Year.min())
         print(movies.Year.max())

1915
2013


In [23]: by_year = movies.groupby("Year")
         print(by_year.groups.keys())

dict_keys([1915, 1916, 1920, 1925, 1927, 1929, 1930, 1931, 1933, 1934, 1935, 1936, 1937, 1938,


In [24]: by_year.sum().head()

Out[24]:         Budget    US Gross  Worldwide Gross     Profits
         Year
         1915   110000  10000000.0       11000000.0  10890000.0
         1916   585907   8000000.0        8000000.0   7414093.0
         1920   100000   3000000.0        3000000.0   2900000.0
         1925  4145000  20000000.0       31000000.0  26855000.0
         1927  2000000         0.0              0.0  -2000000.0

In [25]: # Grouping the movies by decade
         movies["Decade"] = movies.Year.apply(lambda x: (x //10 * 10))
         movies["Decade"].unique()

Out[25]: array([2010, 2000, 1990, 1980, 1970, 1960, 1950, 1940, 1930, 1920, 1910],
               dtype=int64)

In [26]: by_decade = movies.groupby("Decade")
         by_decade.head()
```

Out[26]:

	Release_Date	Movie	\
0	2012-03-09	John Carter	
1	2007-05-25	Pirates of the Caribbean: At World's End	
2	2013-12-13	The Hobbit: There and Back Again	
3	2012-12-14	The Hobbit: An Unexpected Journey	
4	2010-11-24	Tangled	
5	2007-05-04	Spider-Man 3	
6	2009-07-15	Harry Potter and the Half-Blood Prince	
7	2011-05-20	Pirates of the Caribbean: On Stranger Tides	
9	2009-12-18	Avatar	
10	2006-06-28	Superman Returns	
23	1997-12-19	Titanic	
42	1995-07-28	Waterworld	
44	1999-06-30	Wild Wild West	
87	1999-06-16	Tarzan	
92	1998-07-01	Armageddon	
459	1988-06-22	Who Framed Roger Rabbit?	
471	1989-08-09	The Abyss	
617	1988-05-25	Rambo III	
641	1989-12-22	Tango & Cash	
644	1978-12-15	Superman	
682	1981-06-19	Superman II	
886	1963-06-12	Cleopatra	
1126	1979-12-07	Star Trek: The Motion Picture	
1202	1979-12-14	1941	
1206	1979-08-15	Apocalypse Now	
1209	1979-06-29	Moonraker	
1551	1969-12-16	Hello, Dolly	
1674	1969-01-01	Sweet Charity	
1677	1965-02-15	The Greatest Story Ever Told	
1679	1969-10-15	Paint Your Wagon	
2030	1959-11-18	Ben-Hur	
2173	1956-10-05	The Ten Commandments	
2570	1951-02-23	Quo Vadis?	
2766	1956-10-17	Around the World in 80 Days	
2772	1946-12-31	Duel in the Sun	
2777	1956-01-01	War and Peace	
2821	1944-08-01	Wilson	
2968	1930-01-01	Hell's Angels	
2996	1939-12-15	Gone with the Wind	
2997	1925-12-30	Ben-Hur	
3006	1948-01-01	The Pirate	
3054	1946-01-01	It's a Wonderful Life	
3075	1948-01-01	Red River	
3129	1939-01-01	The Wizard of Oz	
3193	1927-08-12	Wings	
3210	1938-01-01	Alexander's Ragtime Band	
3268	1938-01-01	You Can't Take It With You	

						\
3501	1916-09-05				Intolerance	
3502	1929-06-06				The Broadway Melody	
3539	1925-01-01				The Big Parade	
3558	1916-12-24				20,000 Leagues Under the Sea	
3574	1915-02-08				The Birth of a Nation	
3580	1920-09-17				Over the Hill to the Poorhouse	

	Distributor	Budget	US Gross	Worldwide Gross	\
0	0	300000000	66439100.0	2.544391e+08	
1	Buena Vista	300000000	309420425.0	9.609965e+08	
2	New Line	270000000	0.0	0.000000e+00	
3	New Line	270000000	0.0	0.000000e+00	
4	Buena Vista	260000000	200821936.0	5.865819e+08	
5	Sony	258000000	336530303.0	8.908753e+08	
6	Warner Bros.	250000000	301959197.0	9.344165e+08	
7	Buena Vista	250000000	241063875.0	1.043664e+09	
9	20th Century Fox	237000000	760507625.0	2.783919e+09	
10	Warner Bros.	232000000	200120000.0	3.908740e+08	
23	20th Century Fox	200000000	600788188.0	1.842880e+09	
42	Universal	175000000	88246220.0	2.642462e+08	
44	Warner Bros.	175000000	113805681.0	2.212293e+08	
87	Buena Vista	145000000	171091819.0	4.481918e+08	
92	Buena Vista	140000000	201578182.0	5.546000e+08	
459	Buena Vista	70000000	154112492.0	3.515000e+08	
471	20th Century Fox	70000000	54243125.0	5.424312e+07	
617	TriStar Pictures	58000000	53715611.0	1.887156e+08	
641	Warner Bros.	55000000	63408614.0	6.340861e+07	
644	Warner Bros.	55000000	134218018.0	3.002000e+08	
682	Warner Bros.	54000000	108185706.0	1.081857e+08	
886	0	44000000	48000000.0	6.200000e+07	
1126	Paramount Pictures	35000000	82258456.0	1.390000e+08	
1202	Universal	32000000	34175000.0	9.487500e+07	
1206	MGM/UA	31500000	78800000.0	7.880000e+07	
1209	MGM/UA	31000000	70300000.0	2.103000e+08	
1551	20th Century Fox	24000000	33208099.0	3.320810e+07	
1674	0	20000000	8000000.0	8.000000e+06	
1677	MGM/UA	20000000	15473333.0	1.547333e+07	
1679	Paramount Pictures	20000000	31678778.0	3.167878e+07	
2030	0	15000000	73000000.0	7.300000e+07	
2173	0	13500000	80000000.0	8.000000e+07	
2570	0	8250000	30000000.0	3.000000e+07	
2766	United Artists	6000000	42000000.0	4.200000e+07	
2772	0	6000000	20400000.0	2.040000e+07	
2777	0	6000000	12500000.0	1.250000e+07	
2821	0	5200000	2000000.0	2.000000e+06	
2968	0	4000000	0.0	0.000000e+00	
2996	MGM/UA	3900000	198680470.0	3.905252e+08	
2997	0	3900000	9000000.0	9.000000e+06	

3006	0	3700000	2956000.0	2.956000e+06
3054	0	3180000	6600000.0	6.600000e+06
3075	0	3000000	9012000.0	9.012000e+06
3129	0	2777000	28202232.0	2.820223e+07
3193	Paramount Pictures	2000000	0.0	0.000000e+00
3210	0	2000000	4000000.0	4.000000e+06
3268	0	1644000	4000000.0	4.000000e+06
3501	0	385907	0.0	0.000000e+00
3502	0	379000	2800000.0	4.358000e+06
3539	0	245000	11000000.0	2.200000e+07
3558	0	200000	8000000.0	8.000000e+06
3574	0	110000	10000000.0	1.100000e+07
3580	0	100000	3000000.0	3.000000e+06

	Profits	Year	Decade
0	-4.556090e+07	2012	2010
1	6.609965e+08	2007	2000
2	-2.700000e+08	2013	2010
3	-2.700000e+08	2012	2010
4	3.265819e+08	2010	2010
5	6.328753e+08	2007	2000
6	6.844165e+08	2009	2000
7	7.936639e+08	2011	2010
9	2.546919e+09	2009	2000
10	1.588740e+08	2006	2000
23	1.642880e+09	1997	1990
42	8.924622e+07	1995	1990
44	4.622934e+07	1999	1990
87	3.031918e+08	1999	1990
92	4.146000e+08	1998	1990
459	2.815000e+08	1988	1980
471	-1.575688e+07	1989	1980
617	1.307156e+08	1988	1980
641	8.408614e+06	1989	1980
644	2.452000e+08	1978	1970
682	5.418571e+07	1981	1980
886	1.800000e+07	1963	1960
1126	1.040000e+08	1979	1970
1202	6.287500e+07	1979	1970
1206	4.730000e+07	1979	1970
1209	1.793000e+08	1979	1970
1551	9.208099e+06	1969	1960
1674	-1.200000e+07	1969	1960
1677	-4.526667e+06	1965	1960
1679	1.167878e+07	1969	1960
2030	5.800000e+07	1959	1950
2173	6.650000e+07	1956	1950
2570	2.175000e+07	1951	1950

```
2766  3.600000e+07  1956    1950
2772  1.440000e+07  1946    1940
2777  6.500000e+06  1956    1950
2821 -3.200000e+06  1944    1940
2968 -4.000000e+06  1930    1930
2996  3.866252e+08  1939    1930
2997  5.100000e+06  1925    1920
3006 -7.440000e+05  1948    1940
3054  3.420000e+06  1946    1940
3075  6.012000e+06  1948    1940
3129  2.542523e+07  1939    1930
3193 -2.000000e+06  1927    1920
3210  2.000000e+06  1938    1930
3268  2.356000e+06  1938    1930
3501 -3.859070e+05  1916    1910
3502  3.979000e+06  1929    1920
3539  2.175500e+07  1925    1920
3558  7.800000e+06  1916    1910
3574  1.089000e+07  1915    1910
3580  2.900000e+06  1920    1920
```

```
In [27]: # Computing the mean and median profits for each decade
         by_decade_mean = by_decade.Profits.mean()
         by_decade_median = by_decade.Profits.median()
```

1.1.2 Answers to Question 1 c.

c. What are the mean and median movie profits by decade? Which decade was the most profitable? (Hint: Answering this question requires several steps: grouping the movies by decade, computing the mean and median profits for each decade, and combining the results back together.)

```
In [28]: print("The mean movie profits by decade were:")
         by_decade_mean
```

```
The mean movie profits by decade were:


Out[28]: Decade
         1910    6.101364e+06
         1920    6.346800e+06
         1930    3.892876e+07
         1940    1.025301e+07
         1950    1.816625e+07
         1960    2.845890e+07
         1970    6.358547e+07
         1980    5.114162e+07
         1990    5.751548e+07
         2000    5.318013e+07
```

```
2010    6.331232e+07
Name: Profits, dtype: float64
```

```
In [29]: print("The median movie profits by decade were:")
         by_decade_median
```

```
The median movie profits by decade were:
```

```
Out[29]: Decade
         1910     7800000.0
         1920     3979000.0
         1930     2265500.0
         1940     6012000.0
         1950     8690000.0
         1960    10564923.0
         1970    19533200.0
         1980    16168359.0
         1990     9133087.0
         2000     8762690.0
         2010     8626300.0
         Name: Profits, dtype: float64
```

```
In [30]: print("The most profitable decade for movie profits was the 1970s with %f in profits."
               by_decade_mean.max())
```

```
The most profitable decade for movie profits was the 1970s with 63585471.388350 in profits.
```

2 Question 2

2.0.1 PART 1: (1 point)

Load the earthquake data in QuakeData.csv into a DataFrame, and use it to answer the following questions:

a. What is the median earthquake magnitude?

b. What is the correlation between magnitude and depth?

```
In [31]: #earthquakes = pd.read_csv("QuakeData.csv", sep=',')
         earthquakes = pd.read_csv("QuakeData.csv", parse_dates=[0])
```

```
In [32]: earthquakes.head()
```

```
Out[32]:                 DateTime  Latitude  Longitude  Depth  Magnitude MagType  \
         0 2012-01-01 00:30:08.770    12.008    143.487   35.0        5.1      mb
         1 2012-01-01 00:43:42.770    12.014    143.536   35.0        4.4      mb
         2 2012-01-01 00:50:08.040   -11.366    166.218   67.5        5.3      mb
         3 2012-01-01 01:22:07.660    -6.747    130.008  145.0        4.2      mb
```

```
4 2012-01-01 02:35:21.110    23.472     91.834   27.8        4.6      mb

   NbStations  Gap  Distance   RMS Source                  EventID  \
0         178   45       NaN  1.20    pde   pde20120101003008770_35
1          29  121       NaN  0.98    pde   pde20120101004342770_35
2         143   43       NaN  0.82    pde   pde20120101005008040_67
3          14  112       NaN  1.16    pde  pde20120101012207660_145
4          74   77       NaN  0.65    pde   pde20120101023521110_27

         Version
0  1363392487731
1  1363392488431
2  1363392488479
3  1363392488594
4  1363392488611
```

```
In [33]: earthquakes.dtypes
```

```
Out[33]: DateTime      datetime64[ns]
         Latitude             float64
         Longitude            float64
         Depth                float64
         Magnitude            float64
         MagType               object
         NbStations             int64
         Gap                    int64
         Distance             float64
         RMS                  float64
         Source                object
         EventID               object
         Version                int64
         dtype: object
```

```
In [34]: earthquakes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12684 entries, 0 to 12683
Data columns (total 13 columns):
DateTime      12684 non-null datetime64[ns]
Latitude      12684 non-null float64
Longitude     12684 non-null float64
Depth         12676 non-null float64
Magnitude     12684 non-null float64
MagType       12684 non-null object
NbStations    12684 non-null int64
Gap           12684 non-null int64
Distance      1 non-null float64
RMS           10544 non-null float64
Source        12684 non-null object
```

```
EventID       12684 non-null object
Version       12684 non-null int64
dtypes: datetime64[ns](1), float64(6), int64(3), object(3)
memory usage: 1.3+ MB
```

```
In [35]: earthquakes.Magnitude.describe()
```

```
Out[35]: count    12684.000000
         mean         4.558483
         std          0.418082
         min          4.000000
         25%          4.300000
         50%          4.500000
         75%          4.800000
         max          8.600000
         Name: Magnitude, dtype: float64
```

2.0.2 Answer to 2 a.

2.0.3 a. What is the median earthquake magnitude?

```
In [36]: print("The median earthquake magnitude for the datset is %f." % earthquakes.Magnitude
```

```
The median earthquake magnitude for the datset is 4.500000.
```

2.0.4 Answer to 2 b.

2.0.5 b. What is the correlation between magnitude and depth?

```
In [37]: earthquakes_subset = earthquakes[['Magnitude','Depth']]
```

```
In [38]: earthquakes_subset.corr()
```

```
Out[38]:            Magnitude     Depth
         Magnitude   1.000000  0.029175
         Depth       0.029175  1.000000
```

```
In [39]: earthquakes.Depth.corr(earthquakes.Magnitude)
```

```
Out[39]: 0.02917515915997664
```

```
In [40]: print("The correlation between magnitude and depth is %f." %
               earthquakes.Depth.corr(earthquakes.Magnitude))
```

```
The correlation between magnitude and depth is 0.029175.
```

3 Question 2

3.0.1 PART 2: (7 points)

c. What fraction (not count) of earthquakes happen each month, across all years (i.e. all earthquakes occurring in January as a proportion of the grand total, all earthquakes in February as a proportion of the grand total, etc.)?

d. Is there correlation between the number of movies released monthly (i.e. Jan-1990, Feb-1990...) , and the number of earthquakes in that month?

In [41]:
```
earthquakes.head()
```

Out[41]:

	DateTime	Latitude	Longitude	Depth	Magnitude	MagType	NbStations	Gap	Distance	RMS	Source	EventID	Version
0	2012-01-01 00:30:08.770	12.008	143.487	35.0	5.1	mb	178	45	NaN	1.20	pde	pde20120101003008770_35	1363392487731
1	2012-01-01 00:43:42.770	12.014	143.536	35.0	4.4	mb	29	121	NaN	0.98	pde	pde20120101004342770_35	1363392488431
2	2012-01-01 00:50:08.040	-11.366	166.218	67.5	5.3	mb	143	43	NaN	0.82	pde	pde20120101005008040_67	1363392488479
3	2012-01-01 01:22:07.660	-6.747	130.008	145.0	4.2	mb	14	112	NaN	1.16	pde	pde20120101012207660_145	1363392488594
4	2012-01-01 02:35:21.110	23.472	91.834	27.8	4.6	mb	74	77	NaN	0.65	pde	pde20120101023521110_27	1363392488611

In [42]:
```
earthquakes["Month"] = earthquakes.DateTime.apply(lambda x: x.month)
earthquakes.columns
```

Out[42]:
```
Index(['DateTime', 'Latitude', 'Longitude', 'Depth', 'Magnitude', 'MagType',
       'NbStations', 'Gap', 'Distance', 'RMS', 'Source', 'EventID', 'Version',
       'Month'],
      dtype='object')
```

In [43]:
```
#by_month = earthquakes.groupby("Month")
#by_month.head()
```

In [44]:
```
quakes = earthquakes[["Month", "DateTime"]]
```

In [45]:
```
by_month = quakes.groupby("Month")
by_month.head()
```

Out[45]:

	Month	DateTime
0	1	2012-01-01 00:30:08.770
1	1	2012-01-01 00:43:42.770
2	1	2012-01-01 00:50:08.040
3	1	2012-01-01 01:22:07.660
4	1	2012-01-01 02:35:21.110
1005	2	2012-02-01 01:29:24.860
1006	2	2012-02-01 01:54:58.660
1007	2	2012-02-01 02:43:19.000
1008	2	2012-02-01 04:26:14.450
1009	2	2012-02-01 04:30:47.110
2086	3	2012-03-01 00:16:01.060
2087	3	2012-03-01 00:43:29.140
2088	3	2012-03-01 00:59:59.990
2089	3	2012-03-01 01:41:11.220
2090	3	2012-03-01 01:41:40.500
3231	4	2012-04-01 00:06:36.820
3232	4	2012-04-01 03:18:23.330
3233	4	2012-04-01 04:06:10.970
3234	4	2012-04-01 04:24:40.450
3235	4	2012-04-01 04:30:56.750
4624	5	2012-05-01 00:09:12.960
4625	5	2012-05-01 01:43:37.790
4626	5	2012-05-01 02:32:28.070
4627	5	2012-05-01 02:43:34.000
4628	5	2012-05-01 04:53:10.580
5682	6	2012-06-01 01:49:48.160
5683	6	2012-06-01 02:51:45.700
5684	6	2012-06-01 04:14:47.500
5685	6	2012-06-01 05:07:01.980
5686	6	2012-06-01 06:18:50.120
6582	7	2012-07-01 00:44:31.000
6583	7	2012-07-01 01:43:26.400
6584	7	2012-07-01 02:04:37.200
6585	7	2012-07-01 02:49:46.000
6586	7	2012-07-01 03:25:20.800
7464	8	2012-08-01 00:03:47.220
7465	8	2012-08-01 00:20:37.990
7466	8	2012-08-01 00:32:47.540
7467	8	2012-08-01 01:21:31.990
7468	8	2012-08-01 07:44:58.260
8486	9	2012-09-01 00:01:57.890
8487	9	2012-09-01 00:25:02.670
8488	9	2012-09-01 00:36:05.680
8489	9	2012-09-01 00:50:00.340
8490	9	2012-09-01 00:52:03.730
9618	10	2012-10-01 01:00:42.650
9619	10	2012-10-01 01:32:33.000

```
9620      10 2012-10-01 01:36:27.000
9621      10 2012-10-01 02:58:19.220
9622      10 2012-10-01 04:53:57.750
10669     11 2012-11-01 01:23:22.130
10670     11 2012-11-01 01:24:13.290
10671     11 2012-11-01 01:37:30.170
10672     11 2012-11-01 03:16:01.500
10673     11 2012-11-01 03:27:13.420
11665     12 2012-12-01 00:37:39.410
11666     12 2012-12-01 02:22:22.610
11667     12 2012-12-01 03:34:02.930
11668     12 2012-12-01 05:55:53.960
11669     12 2012-12-01 08:00:57.940
```

```
In [46]: total_quakes_per_month = by_month.aggregate(len)
         total_quakes_per_month.columns = ["Number of Earthquakes that Month"]
         total_quakes_per_month
```

Out[46]:

Month	Number of Earthquakes that Month
1	1024
2	1081
3	1145
4	1393
5	1058
6	900
7	882
8	1022
9	1132
10	1051
11	996
12	1000

3.0.2 Answer to Question 2 c.

c. What fraction (not count) of earthquakes happen each month, across all years (i.e. all earthquakes occurring in January as a proportion of the grand total, all earthquakes in February as a proportion of the grand total, etc.)?

```
In [47]: fraction = total_quakes_per_month / total_quakes_per_month["Number of Earthquakes that
         fraction.columns = ["Fraction of Total Earthquakes"]
         print("The fractions of earthquakes that happen each month as a proportion " +
               "of the grand total of all earthquakes are shown below: ")
         fraction
```

```
The fractions of earthquakes that happen each month as a proportion of the grand total of all e
```

Out[47]:

Month	Fraction of Total Earthquakes

```
1                          0.080732
2                          0.085225
3                          0.090271
4                          0.109823
5                          0.083412
6                          0.070956
7                          0.069536
8                          0.080574
9                          0.089246
10                         0.082860
11                         0.078524
12                         0.078839
```

In [48]:
```
movies["Month"] = movies.Release_Date.apply(lambda x: x.month)
movies.head()
```

Out[48]:

	Release_Date	Movie	Distributor	Budget	US Gross	Worldwide Gross	Profits	Year	Decade	Month
0	2012-03-09	John Carter	0	300000000	66439100.0	254439100.0	-45560900.0	2012	2010	3
1	2007-05-25	Pirates of the Caribbean: At World's End	Buena Vista	300000000	309420425.0	960996492.0	660996492.0	2007	2000	5
2	2013-12-13	The Hobbit: There and Back Again	New Line	270000000	0.0	0.0	-270000000.0	2013	2010	12
3	2012-12-14	The Hobbit: An Unexpected Journey	New Line	270000000	0.0	0.0	-270000000.0	2012	2010	12
4	2010-11-24	Tangled	Buena Vista	260000000	200821936.0	586581936.0	326581936.0	2010	2010	11

In [49]:
```
movies_by_month = movies.groupby("Month")
movies_by_month
```

Out[49]:
```
<pandas.core.groupby.groupby.DataFrameGroupBy object at 0x0000000008F39DA0>
```

In [50]:
```
total_movies_per_month = movies_by_month.aggregate(len)
total_movies_per_month
```

Out[50]:

Month	Release_Date	Movie	Distributor	Budget	US Gross	Worldwide Gross
1	256	256	256	256	256.0	256.0
2	231	231	231	231	231.0	231.0
3	282	282	282	282	282.0	282.0
4	288	288	288	288	288.0	288.0
5	255	255	255	255	255.0	255.0
6	305	305	305	305	305.0	305.0
7	282	282	282	282	282.0	282.0
8	321	321	321	321	321.0	321.0
9	307	307	307	307	307.0	307.0

```
10              367    367          367     367     367.0           367.0
11              312    312          312     312     312.0           312.0
12              421    421          421     421     421.0           421.0

       Profits  Year  Decade
Month
1        256.0   256     256
2        231.0   231     231
3        282.0   282     282
4        288.0   288     288
5        255.0   255     255
6        305.0   305     305
7        282.0   282     282
8        321.0   321     321
9        307.0   307     307
10       367.0   367     367
11       312.0   312     312
12       421.0   421     421
```

In [51]:
```
total_movies_per_month = total_movies_per_month[["Movie"]]
total_movies_per_month.columns = ["Movies Released Monthly"]
```

In [52]:
```
total_movies_per_month
```

Out[52]:
```
       Movies Released Monthly
Month
1                          256
2                          231
3                          282
4                          288
5                          255
6                          305
7                          282
8                          321
9                          307
10                         367
11                         312
12                         421
```

In [53]:
```
monthly_data = total_movies_per_month.merge(total_quakes_per_month, how="right",
                                            left_index=True, right_on="Month")
monthly_data.columns
```

Out[53]:
```
Index(['Movies Released Monthly', 'Number of Earthquakes that Month'], dtype='object'
```

In [54]:
```
monthly_data
```

Out[54]:
```
       Movies Released Monthly  Number of Earthquakes that Month
Month
```

1	256	1024
2	231	1081
3	282	1145
4	288	1393
5	255	1058
6	305	900
7	282	882
8	321	1022
9	307	1132
10	367	1051
11	312	996
12	421	1000

3.0.3 Answer to Question 2 d.

d. Is there correlation between the number of movies released monthly (i.e. Jan-1990, Feb-1990...) , and the number of earthquakes in that month?

```
In [55]: monthly_data.corr()
```

```
Out[55]:                                  Movies Released Monthly  \
         Movies Released Monthly                         1.000000
         Number of Earthquakes that Month               -0.156923

                                           Number of Earthquakes that Month
         Movies Released Monthly                                  -0.156923
         Number of Earthquakes that Month                          1.000000
```