In [1]:
```
#import packages
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np
```

In [2]:
```
#define webiste url and get content
url = 'https://www.the-numbers.com/market/distributors'
r = requests.get(url)
soup = BeautifulSoup(r.content)
```

In [3]:
```
#clean up the html code
soup.prettify
```

Out[3]:
```
<bound method Tag.prettify of <!DOCTYPE html>
<html xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="http://ogp.me/ns
#">
<head>
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async="" src="https://www.googletagmanager.com/gtag/js?id=UA-1343128-
1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-1343128-1');
</script>
<meta content='(PICS-1.1 "https://www.icra.org/ratingsv02.html" l gen true fo
r "https://www.the-numbers.com/" r (cb 1 lz 1 nz 1 oz 1 vz 1) "https://www.rs
ac.org/ratingsv01.html" l gen true for "https://www.the-numbers.com/" r (n 0
s 0 v 0 l 0))' http-equiv="PICS-Label"/>
<!--<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" >
```

In [4]:
```
#define data as a list
data = []

#find table
results = soup.select('table')[0]

#find all tr elements
rows = results.find_all(['tr'])

#find data in td tags
for row in rows:
    cols = row.find_all('td')
    #get text
    cols = [ele.text.strip() for ele in cols]
    #remove header row
    data.append(cols[1:])

#name columns
cols = ['Distributor', 'Number of Movies', 'Box Office Total', 'Ticket Sales', 'M
#define dataframe
#remove redundant "Rank" column
df = pd.DataFrame(data[1:], columns = cols)

#return head of table
df.head()

df.to_pickle('daves_data.pkl')
```

In [5]:
```
#convert numerical data from objects into integers and floats
df['Number of Movies'] = df['Number of Movies'].astype(int)

df['Box Office Total'] = df['Box Office Total'].str.replace('$', '')
df['Box Office Total'] = df['Box Office Total'].str.replace(',', '')
df['Box Office Total'] = df['Box Office Total'].astype(float)

df['Ticket Sales'] = df['Ticket Sales'].str.replace(',', '')
df['Ticket Sales'] = df['Ticket Sales'].astype(float)

df['Market Share'] = df['Market Share'].str.replace('%', '')
df['Market Share'] = df['Market Share'].astype(float)

#check the datatypes
df.dtypes
```

Out[5]:
```
Distributor          object
Number of Movies      int32
Box Office Total    float64
Ticket Sales        float64
Market Share        float64
dtype: object
```

In [6]:
```
#create a new variable for the rows that have a market share less than 10%
remaining = df[6:884]
remaining
```

Out[6]:

	Distributor	Number of Movies	Box Office Total	Ticket Sales	Market Share
6	Lionsgate	416	9.538425e+09	1.211612e+09	4.07
7	New Line	207	6.194343e+09	1.116306e+09	2.64
8	Dreamworks SKG	77	4.278649e+09	7.604313e+08	1.83
9	Miramax	384	3.835979e+09	7.140996e+08	1.64
10	MGM	238	3.705595e+09	6.640381e+08	1.58
...	...	...	...	...	...
879	levelFILM	1	1.037000e+03	1.270000e+02	0.00
880	Gunnison Galaxy	1	9.030000e+02	1.040000e+02	0.00
881	Cinevolve Studios	2	8.940000e+02	1.180000e+02	0.00
882	Oilrag Productions	1	8.310000e+02	1.260000e+02	0.00
883	Lavender House Films	1	4.010000e+02	5.500000e+01	0.00

878 rows × 5 columns

In [7]:
```
#create variables for the averages of the lower rows
rem_num_movie = remaining['Number of Movies'].mean()
rem_box_office = remaining['Box Office Total'].mean()
rem_tickets = remaining['Ticket Sales'].mean()
rem_market_share = remaining['Market Share'].mean()*100
print(rem_num_movie)
print(rem_box_office)
print(rem_tickets)
print(rem_market_share)
```

```
13.29384965831435
59861410.19020501
9056278.633257404
2.5239179954441906
```

In [8]:
```
#drop the first row because the company owns most of the studios that make the mo
#drop all of the rows with a market share less than 10%
#market share values and remaining average value will be used to create bar graph
df = df[1:6]
df
```

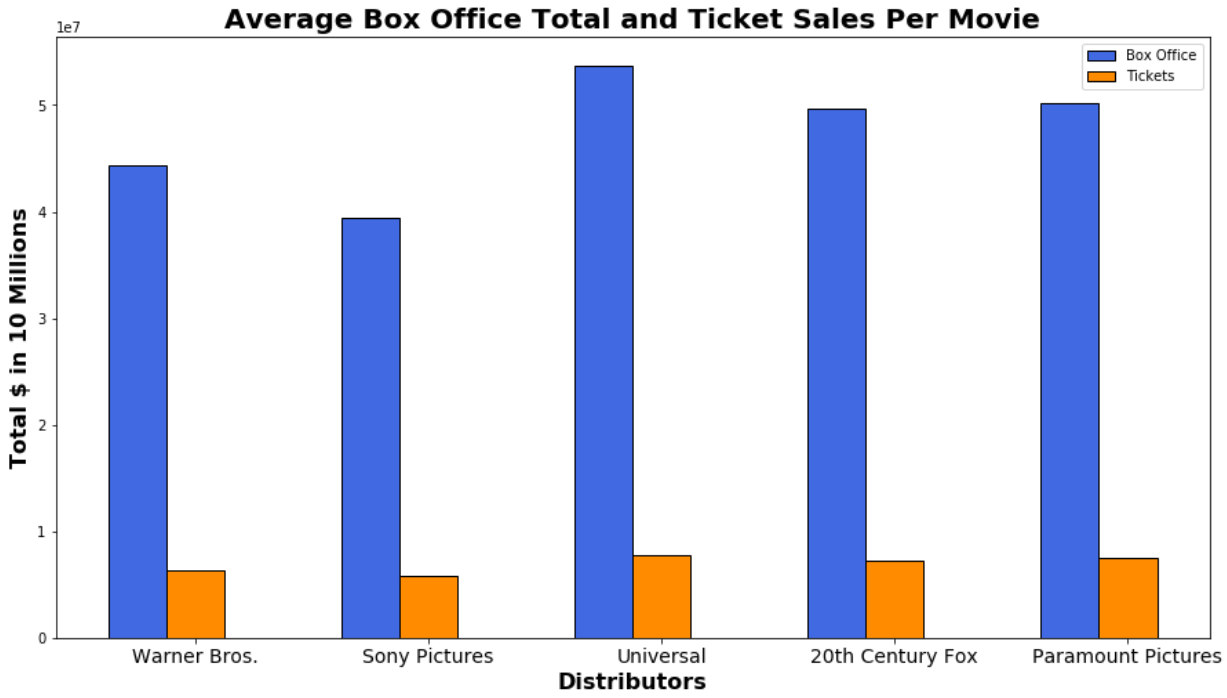
Out[8]:

	Distributor	Number of Movies	Box Office Total	Ticket Sales	Market Share
1	Warner Bros.	803	3.564125e+10	5.133326e+09	15.21
2	Sony Pictures	730	2.877543e+10	4.257946e+09	12.28
3	Universal	511	2.746428e+10	3.938556e+09	11.72
4	20th Century Fox	520	2.585503e+10	3.792393e+09	11.03
5	Paramount Pictures	483	2.423624e+10	3.647709e+09	10.34

In [9]:
```
#create new variablrs to get the average box office total and average ticket sales
box_office_per_movie = df['Box Office Total']/df['Number of Movies']
tickets_per_movie = df['Ticket Sales']/df['Number of Movies']
print(box_office_per_movie)
print(tickets_per_movie)
```

```
1    4.438511e+07
2    3.941840e+07
3    5.374614e+07
4    4.972121e+07
5    5.017856e+07
dtype: float64
1    6.392685e+06
2    5.832802e+06
3    7.707545e+06
4    7.293064e+06
5    7.552194e+06
dtype: float64
```

In [10]:
```
#import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

#determine bar width
barWidth = 0.25

#input values for each bar in double bar graph
bars1 = box_office_per_movie
bars2 = tickets_per_movie

#determine position of each bar
r1 = np.arange(len(bars1))
r2 = [x + barWidth for x in r1]

#size, format and label the data in the graph
plt.figure(figsize=(15,8))
plt.bar(r1, bars1, color='royalblue', width=barWidth, edgecolor='black', label='B
plt.bar(r2, bars2, color='darkorange', width=barWidth, edgecolor='black', label='

#add xticks on the middle of the group bars and label each axis
plt.title('Average Box Office Total and Ticket Sales Per Movie', fontweight='bold
plt.xlabel('Distributors', fontweight='bold', fontsize=16)
plt.xticks([r + barWidth for r in range(len(bars1))], ['Warner Bros.', 'Sony Pict
                                                       '20th Century Fox', 'Param
plt.ylabel('Total $ in 10 Millions', fontweight='bold', fontsize=16)

#create legend and show the graph
plt.legend()
plt.savefig('images/avg_box_office_tickets.png', dpi=300)
plt.show()
```



In [11]:
```
# Function to convert money string to integer
def money_str_int(str):
    number = int(str.strip('$').replace(',',''))
    return number
```

In [12]:
```
# LOAD CAT'S DATA
df_movie_3 = pd.read_pickle('cats_bomojo_data.pkl')
df_movie_3.head(3)
```

Out[12]:

	Rank	Title	Lifetime Gross	Year	URL	Distributor	Bu
0	1	Star Wars: Episode VII - The Force Awakens	936662225	2015	https://www.boxofficemojo.com/title/tt2488496/...	Walt Disney Studios Motion Pictures	24500
1	2	Avengers: Endgame	858373000	2019	https://www.boxofficemojo.com/title/tt4154796/...	Walt Disney Studios Motion Pictures	35600
2	3	Avatar	760507625	2009	https://www.boxofficemojo.com/title/tt0499549/...	Twentieth Century Fox	23700

In [13]:
```
# Get data of home market distributors
url= 'https://www.the-numbers.com/home-market/distributors'
response = requests.get('https://www.the-numbers.com/home-market/distributors')
soup = BeautifulSoup(response.text, 'lxml')
art_body= soup.find_all('table', id = 'page_filling_chart')
for body in art_body:
    print(art_body.text)
```

In [14]:
```
# Confirm webscrape status
response.status_code
```

Out[14]:
```
200
```

In [15]:
```
# Find basic information on chart
df_list = pd.read_html(response.text)
df_home = df_list[0]
df_home.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 3 columns):
Home Market Distributors              614 non-null object
No. of Movies                         614 non-null int64
Total Domestic Home Market Revenue    614 non-null object
dtypes: int64(1), object(2)
memory usage: 14.5+ KB
```

In [16]:
```
# confirm the columns in the chart
df_home.columns
```

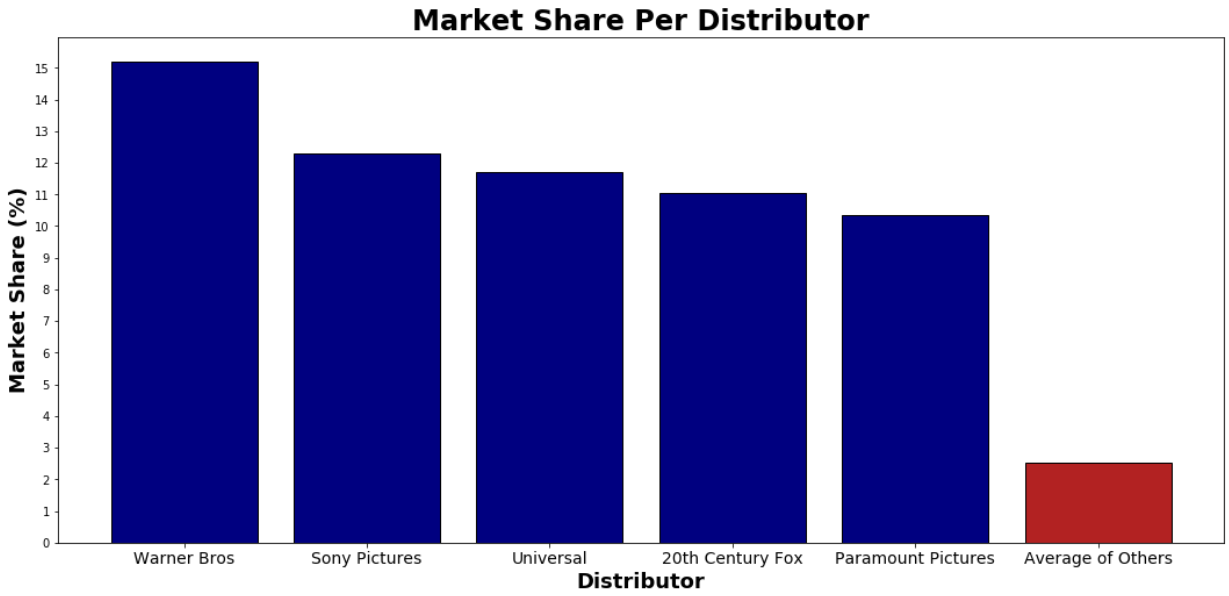
Out[16]:
```
Index(['Home Market Distributors', 'No. of Movies',
       'Total Domestic Home Market Revenue'],
      dtype='object')
```

In [17]:
```
# confirm there isn't missing data
df_home.isna().sum()
```

Out[17]:
```
Home Market Distributors              0
No. of Movies                         0
Total Domestic Home Market Revenue    0
dtype: int64
```

In [ ]:
```

```

In [1]:
```
#import libraries
import numpy as np
import matplotlib.pyplot as plt

#create names for each bar
labels = ['Warner Bros', 'Sony Pictures', 'Universal', '20th Century Fox',
                'Paramount Pictures', 'Average of Others']

#input values
y = [15.21, 12.28, 11.72, 11.03, 10.34, 2.52]

x = [1, 2, 3, 4, 5, 6]

#modify the figure size of the graph
fig = plt.figure(figsize=(18, 8))

#plot it, label it and format it
plt.bar(x, y, color=('navy', 'navy', 'navy', 'navy', 'navy', 'firebrick'), edgeco

plt.xlabel('Distributor', fontweight='bold', fontsize=18)
plt.ylabel('Market Share (%)', fontweight='bold', fontsize=18)
plt.title('Market Share Per Distributor', fontweight='bold', fontsize=24)
plt.xticks(x, labels, fontsize=14)
plt.yticks(np.arange(0, 16, 1))

#show the graph
plt.savefig('images/market_share_per_distributor.png', dpi=300)
plt.show()
```



In [1]:
```
# import libraries
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [2]:
```
# Webscrape needed chart
url= 'https://www.the-numbers.com/home-market/distributors'
response = requests.get('https://www.the-numbers.com/home-market/distributors')
soup = BeautifulSoup(response.text, 'lxml')
art_body= soup.find_all('table', id = 'page_filling_chart')
for body in art_body:
    print(art_body.text)
```

In [3]:
```
# Confirm webscrape status
response.status_code
```

Out[3]:
```
200
```

In [4]:
```
# Find basic information on chart
df_list = pd.read_html(response.text)
df_home = df_list[0]
df_home.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 3 columns):
Home Market Distributors              614 non-null object
No. of Movies                         614 non-null int64
Total Domestic Home Market Revenue    614 non-null object
dtypes: int64(1), object(2)
memory usage: 14.5+ KB
```

In [5]:
```
# pull the first 5 home distributors
df_home.head(15)
```

Out[5]:

	Home Market Distributors	No. of Movies	Total Domestic Home Market Revenue
0	Walt Disney Home Entertainment	388	$10,723,810,487
1	Universal Home Entertainment	1236	$9,480,962,035
2	Warner Home Video	962	$9,289,947,358
3	Fox Home Entertainment	857	$8,862,951,476
4	Sony Pictures Home Entertainment	1300	$7,646,065,095
5	Paramount Home Video	471	$5,987,882,889
6	Lionsgate Home Entertainment	1312	$4,164,608,183
7	Buena Vista Home Entertainment	78	$1,746,787,598
8	New Line Home Video	104	$1,197,796,076
9	Summit Home Video	34	$1,189,957,871
10	Dreamworks Video	43	$691,601,602
11	Weinstein Co./Genius	89	$676,696,151
12	Anchor Bay Home Entertainment	282	$475,782,498
13	Dreamworks Animated Video	14	$424,825,091
14	MGM Video	400	$237,303,214

In [6]:
```
# confirm the columns in the chart
df_home.columns
```

Out[6]:
```
Index(['Home Market Distributors', 'No. of Movies',
       'Total Domestic Home Market Revenue'],
      dtype='object')
```

In [7]:
```
# confirm there isn't missing data
df_home.isna().sum()
```

Out[7]:
```
Home Market Distributors             0
No. of Movies                        0
Total Domestic Home Market Revenue   0
dtype: int64
```

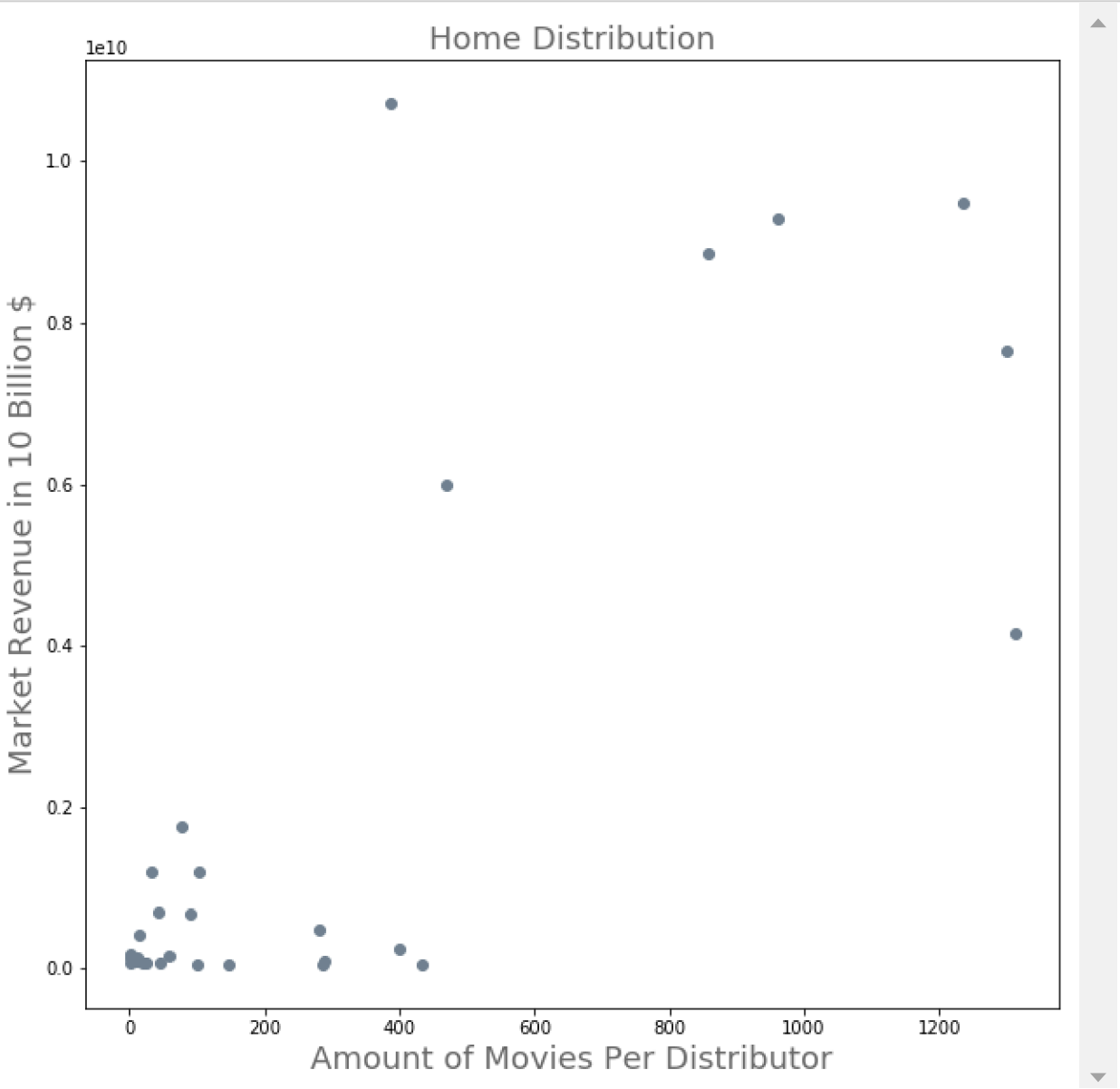
In [8]:
```
# Change column[2] into numbers
def money_str_int(str):
    number = int(str.strip('$').replace(',',''))
    return number
```

In [9]:
```
# Apply changesv
df_home['Total Domestic Home Market Revenue'] = df_home['Total Domestic Home Mark
df_home['Total Domestic Home Market Revenue'] = df_home['Total Domestic Home Mark
df_home['Total Domestic Home Market Revenue'] = df_home['Total Domestic Home Mark
df_home.head()
```

Out[9]:

	Home Market Distributors	No. of Movies	Total Domestic Home Market Revenue
0	Walt Disney Home Entertainment	388	1.072381e+10
1	Universal Home Entertainment	1236	9.480962e+09
2	Warner Home Video	962	9.289947e+09
3	Fox Home Entertainment	857	8.862951e+09
4	Sony Pictures Home Entertainment	1300	7.646065e+09

In [10]:
```
#import libraries
import matplotlib.pyplot as plt
import numpy as np

# apply size restictions
plt.figure(figsize=(10,10))

# create data
y= (df_home.loc[0:30, 'Total Domestic Home Market Revenue'])
x = (df_home.loc[0:30, 'No. of Movies'])



# use the scatter function
plt.scatter( x, y, color= 'slategrey')
plt.title("Home Distribution", fontsize=18, fontweight=0, color='dimgrey')
plt.xlabel("Amount of Movies Per Distributor", fontsize=18, fontweight=0, color='
plt.ylabel("Market Revenue in 10 Billion $", fontsize=18, fontweight=0, color='di
plt.savefig('images/home_distribution_chart.png', dpi=300)
plt.show()
```



In [11]:
```
#define webiste url and get content
url = 'https://www.the-numbers.com/market/distributors'
r = requests.get(url)
soup = BeautifulSoup(r.content)
```

In [12]:
```
#clean up the html code
soup.prettify
```

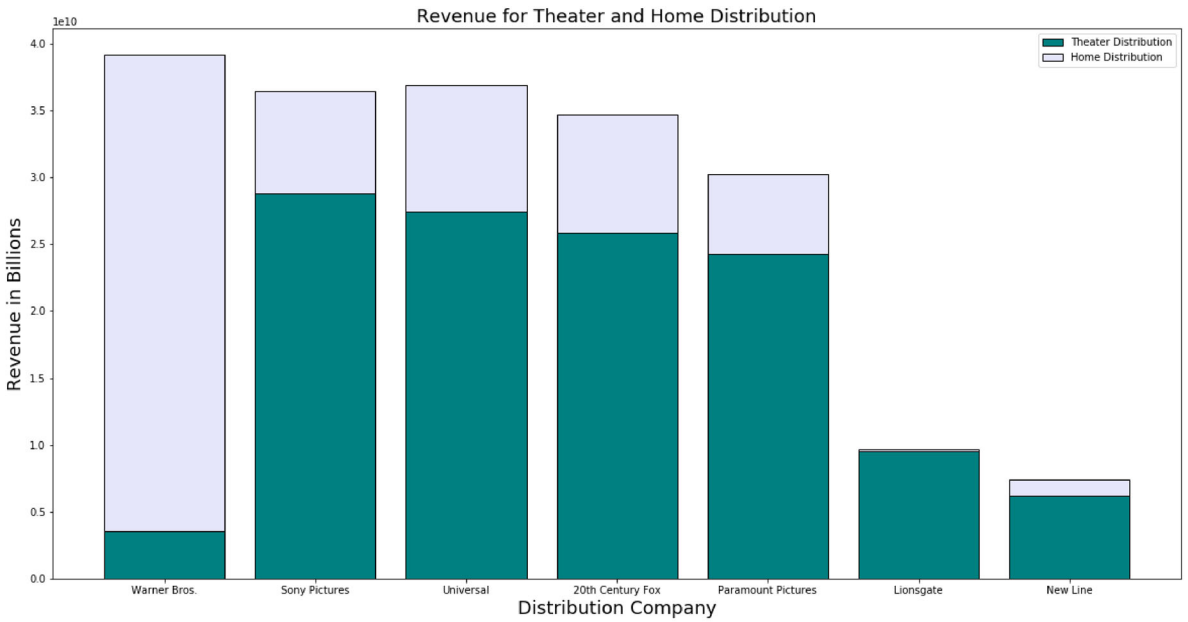
Out[12]:
```
<bound method Tag.prettify of <!DOCTYPE html>
<html xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="http://ogp.me/ns
#">
<head>
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async="" src="https://www.googletagmanager.com/gtag/js?id=UA-1343128-
1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-1343128-1');
</script>
<meta content='(PICS-1.1 "https://www.icra.org/ratingsv02.html" l gen true fo
r "https://www.the-numbers.com/" r (cb 1 lz 1 nz 1 oz 1 vz 1) "https://www.rs
ac.org/ratingsv01.html" l gen true for "https://www.the-numbers.com/" r (n 0
s 0 v 0 l 0))' http-equiv="PICS-Label"/>
<!--<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" >
```

In [13]:
```
#define data as a list
data = []

#find table
results = soup.select('table')[0]

#find all tr elements
rows = results.find_all(['tr'])

#find data in td tags
for row in rows:
    cols = row.find_all('td')
    #get text
    cols = [ele.text.strip() for ele in cols]
    #remove header row
    data.append(cols[1:])

#name columns
cols = ['Distributor', 'Number of Movies', 'Box Office Total', 'Ticket Sales', 'M

#define dataframe
#remove redundant "Rank" column
df = pd.DataFrame(data[1:], columns = cols)

#return head of table
df.head(8)
```

Out[13]:

	Distributor	Number of Movies	Box Office Total	Ticket Sales	Market Share
0	Walt Disney	574	$39,694,407,526	5,668,676,902	16.94%
1	Warner Bros.	803	$35,641,246,799	5,133,326,376	15.21%
2	Sony Pictures	730	$28,775,433,129	4,257,945,527	12.28%
3	Universal	511	$27,464,279,056	3,938,555,708	11.72%
4	20th Century Fox	520	$25,855,030,516	3,792,393,298	11.03%
5	Paramount Pictures	483	$24,236,244,100	3,647,709,479	10.34%
6	Lionsgate	416	$9,538,424,750	1,211,612,233	4.07%
7	New Line	207	$6,194,343,024	1,116,305,898	2.64%

In [14]:
```
# libraries
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import pandas as pd

# apply size restictions
plt.figure(figsize=(20,10))


# Values of each group
bars1 = [3535746799, 28775031704, 27464279056, 25854741898, 24235919096, 95384247
bars2 = [35635746799, 7645687205, 9478437927, 8862787530, 5987291353, 142880327,


# Heights of bars1 + bars2
bars = np.add(bars1, bars2).tolist()

# The position of the bars on the x-axis
r = range(len(bars))

# Names of group and bar width
names = ['Warner Bros.','Sony Pictures','Universal','20th Century Fox','Paramount
barWidth = 1

# Create brown bars
plt.bar(r, bars1, color= 'teal', edgecolor='black', width=0.8, label= 'Theater Di
# Create green bars (middle), on top of the firs ones
plt.bar(r, bars2, bottom=bars1, color='lavender', edgecolor='black', width= 0.8,

# Custom X axis
plt.title("Revenue for Theater and Home Distribution", fontsize=18, fontweight=0,
plt.xticks(r, names)
plt.xlabel("Distribution Company", fontsize=18, fontweight=0)
plt.ylabel("Revenue in Billions", fontsize=18, fontweight=0)
plt.legend()
edgecolor= 'black'

# Show graphic
plt.show()
```


Revenue for Theater and Home Distribution
1e10
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Revenue in Billions
Theater Distribution
Home Distribution
Warner Bros.
Sony Pictures
Universal
20th Century Fox
Paramount Pictures
Lionsgate
New Line
Distribution Company

In [1]:
```
# Generate dataframe for use by other notebooks (takes a while to run!)
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
```

In [2]:
```
# Function to convert money string to integer
def money_str_int(str):
    number = int(str.strip('$').replace(',',''))
    return number
```

In [3]:
```
# Function to get table of top grossing movies from a page

def get_table(soup):
    headers = [col.text.strip() for col in soup.findAll('th')] # column names
    headers.append('URL')
    rows = soup.findAll('table')[0].findAll('tr') # retrieve rows
    data = []
    base_url = 'https://www.boxofficemojo.com'

    # Format table data
    for row in rows[1:]:
        # Get list of text displayed on web page
        cell_data = row.findAll('td')
        cells = [cell.text for cell in cell_data]
        cells[2] = money_str_int(cells[2])

        # Get URL for each movie and append to list
        movie_url = base_url + row.find('a').attrs['href']
        cells.append(movie_url)

        # Add list to data
        data.append(cells)

    return pd.DataFrame(data, columns = headers) #return a data frame
```

In [4]:
```
# This loop will go through all the pages to get a table of 1000 movies
url = 'https://www.boxofficemojo.com/chart/top_lifetime_gross/'
base_url = 'https://www.boxofficemojo.com'
df_movie = pd.DataFrame()

# Get HTML data for top 1000 grossing movies from Box Office Mojo
for i in range(5):
    html_page = requests.get(url)
    soup = BeautifulSoup(html_page.content, 'html.parser')
    fetched_data = get_table(soup)
    if i < 4:
        url = base_url + soup.findAll('li', class_='a-last')[0].find('a').attrs["
    df_movie = df_movie.append(fetched_data, ignore_index = True)

df_movie.head()
```

Out[4]:

	Rank	Title	Lifetime Gross	Year	URL
0	1	Star Wars: Episode VII - The Force Awakens	936662225	2015	https://www.boxofficemojo.com/title/tt2488496/...
1	2	Avengers: Endgame	858373000	2019	https://www.boxofficemojo.com/title/tt4154796/...
2	3	Avatar	760507625	2009	https://www.boxofficemojo.com/title/tt0499549/...
3	4	Black Panther	700426566	2018	https://www.boxofficemojo.com/title/tt1825683/...
4	5	Avengers: Infinity War	678815482	2018	https://www.boxofficemojo.com/title/tt4154756/...

In [5]:
```
# Function to iterate through above dataframe and pull data from each movie's page
def get_movie_data(url):
    movie_page = requests.get(url)
    movie = BeautifulSoup(movie_page.content, 'html.parser')
    divs = movie.findAll('div', class_='a-section a-spacing-none') #first section

    # Variables will come back as 'No Data' if the webpage doesn't have this infor
    distributor = 'No Data'
    budget = 'No Data'
    rating = 'No Data'
    duration = 'No Data'
    genres = 'No Data'

    for div in divs:
        spans = div.findAll('span')
        i=0
        for span in spans:
            if span.text  == 'Domestic Distributor':
                distributor = spans[i+1].text.replace('See full company informati
                i+=1
                break
            if span.text == 'Budget':
                budget = money_str_int(spans[i+1].text) #convert budget string to
                i+=1
                break
            if span.text == 'MPAA':
                rating = spans[i+1].text
                i+=1
                break
            if span.text == 'Running Time':
                dur = spans[i+1].text.split() #imports duration as hours and minu
                if len(dur) == 2:
                    dur.extend(['0','0'])
                duration = round( (float(dur[0]) + float(dur[2])/60) , 2) #conver
                i+=1
                break
            if span.text == 'Genres':
                genres = spans[i+1].text.replace(' ','').replace('\n\n',',') #.st
                i+=1
                break
            else:
                i+=1

    df_movie = pd.DataFrame([distributor, budget, rating, duration, genres]).tran
    df_movie.columns = ['Distributor', 'Budget', 'Rating', 'Running_Time_hrs', 'G

    return df_movie
```

In [6]:
```
# loop to get data from first 500 URLs from prior dataframe (broke this up due to
df_movie_2 = pd.DataFrame(columns = ['Distributor', 'Budget', 'Rating', 'Running_
for url in df_movie['URL'][:500]:
    movie_data = get_movie_data(url)
    df_movie_2 = df_movie_2.append(movie_data, ignore_index=True)
df_movie_2
```

Out[6]:

	Distributor	Budget	Rating	Running_Time_hrs	Gen
0	Walt Disney Studios Motion Pictures	245000000	PG-13	2.3	Action,Adventure,Sc
1	Walt Disney Studios Motion Pictures	356000000	PG-13	3.02	Action,Adventure,Drama,Sc
2	Twentieth Century Fox	237000000	PG-13	2.7	Action,Adventure,Fantasy,Sc
3	Walt Disney Studios Motion Pictures	No Data	PG-13	2.23	Action,Adventure,Sc
4	Walt Disney Studios Motion Pictures	No Data	PG-13	2.48	Action,Adventure,Sc
...	...	...	...	...	
495	Walt Disney Studios Motion Pictures	50000000	PG	2.08	Adventure,Comedy,Drama,Fantasy,Mus
496	United Artists	No Data	No Data	1.52	Drama,S
497	Walt Disney Studios Motion Pictures	85000000	PG	1.78	Animation,Comedy,Family,Fantasy,Musical,Roma
498	Paramount Pictures	150000000	PG-13	2.77	Drama,Fantasy,Roma
499	Warner Bros.	50000000	PG-13	1.78	Action,Comedy,Cr

500 rows × 5 columns

In [7]:
```
# loop to add data from last 500 URLs from prior dataframe
for url in df_movie['URL'][500:]:
    movie_data = get_movie_data(url)
    df_movie_2 = df_movie_2.append(movie_data, ignore_index=True)
```

In [8]:
```
# Join previous dataframes into one and pickle file for use by other notebooks
df_movie_2.columns = ['Distributor', 'Budget', 'Rating', 'Running_Time_hrs', 'Gen
result = pd.concat([df_movie, df_movie_2], axis=1, join='inner')
result.to_pickle('cats_bomojo_data.pkl')
```

In [1]:
```
#import packages
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# Load Cat's data frame
df10 = pd.read_pickle('cats_bomojo_data.pkl')
```

In [3]:
```
# Split Genre column from string to list
temp = [x.split(',') for x in df10['Genres']] # Temp list for list of genres
for i in range(len(df10.index)): # maps list back into dataframe
    df10.at[i, 'Genres'] = temp[i]
df10.head(3)
```

Out[3]:

	Rank	Title	Lifetime Gross	Year	URL	Distributor	Bu
0	1	Star Wars: Episode VII - The Force Awakens	936662225	2015	https://www.boxofficemojo.com/title/tt2488496/...	Walt Disney Studios Motion Pictures	24500
1	2	Avengers: Endgame	858373000	2019	https://www.boxofficemojo.com/title/tt4154796/...	Walt Disney Studios Motion Pictures	35600
2	3	Avatar	760507625	2009	https://www.boxofficemojo.com/title/tt0499549/...	Twentieth Century Fox	23700

In [4]:
```
# create dictionary of genre and counts of each genre for each distributor
genre_counts = {}
for i in range(len(df10.index)):
    distributor = df10['Distributor'][i]
    for genre in df10['Genres'][i]:
        if distributor not in genre_counts.keys():
            genre_counts[distributor] = {genre : 1}
        if genre not in genre_counts[distributor].keys():
            genre_counts[distributor][genre] = 1
        else:
            genre_counts[distributor][genre] += 1

genre_counts['Twentieth Century Fox']
```

Out[4]:
```
{'Action': 62,
 'Adventure': 73,
 'Fantasy': 36,
 'Sci-Fi': 44,
 'Comedy': 59,
 'Family': 39,
 'Thriller': 38,
 'Drama': 31,
 'Romance': 11,
 'Animation': 25,
 'Music': 5,
 'Biography': 7,
 'Crime': 14,
 'History': 3,
 'Western': 2,
 'Musical': 7,
 'Mystery': 7,
 'Sport': 3,
 'War': 2,
 'Horror': 2}
```

In [5]:
```
# Load Dave's data frame and limit to top 10 distributers
df_top10_dists = pd.read_pickle('daves_data.pkl')[:10]
df_top10_dists
```

Out[5]:

	Distributor	Number of Movies	Box Office Total	Ticket Sales	Market Share
0	Walt Disney	573	$39,691,885,638	5,668,400,077	16.94%
1	Warner Bros.	803	$35,638,946,799	5,133,073,907	15.21%
2	Sony Pictures	730	$28,775,313,514	4,257,932,398	12.28%
3	Universal	511	$27,464,279,056	3,938,555,708	11.72%
4	20th Century Fox	520	$25,854,741,898	3,792,361,616	11.03%
5	Paramount Pictures	482	$24,235,980,763	3,647,680,573	10.34%
6	Lionsgate	416	$9,538,424,750	1,211,612,233	4.07%
7	New Line	207	$6,195,281,617	1,116,424,257	2.64%
8	Dreamworks SKG	77	$4,278,649,271	760,431,349	1.83%
9	Miramax	384	$3,835,978,908	714,099,626	1.64%

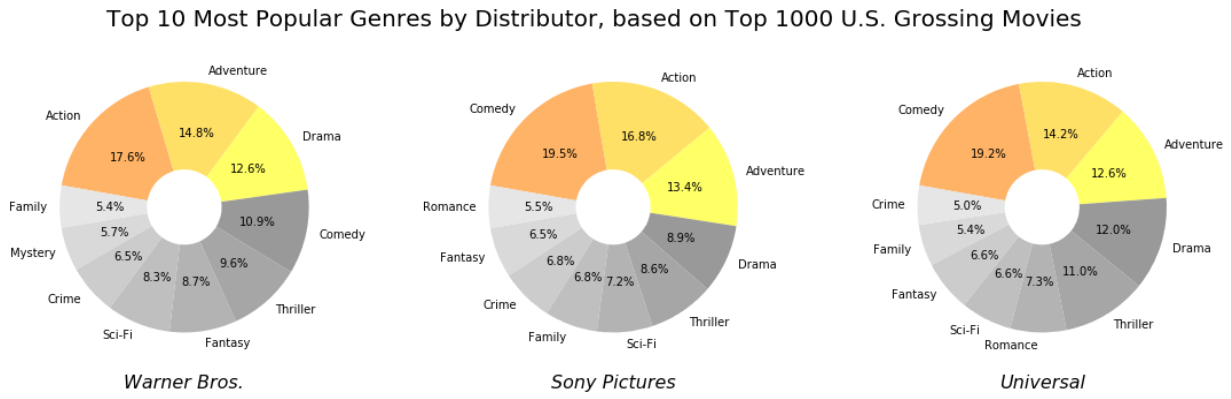
In [6]:
```
# convert dictionary to Data Frame
df_counts = pd.DataFrame.from_dict(genre_counts, orient='columns')
df_counts = df_counts.replace(np.nan, 0).astype(int)
df_counts.reset_index(inplace=True)

# rename columsn to match Dave's data
df_counts = df_counts.rename(columns={'index':'Genre',
                   'Walt Disney Studios Motion Pictures':'Walt Disney',
                   'Twentieth Century Fox':'20th Century Fox',
                    'Sony Pictures Entertainment (SPE)':'Sony Pictures',
                   'Universal Pictures':'Universal',
                   'New Line Cinema':'New Line',
                   'DreamWorks':'Dreamworks SKG'
                  })
df_counts.head(2)
```

Out[6]:

	Genre	Walt Disney	20th Century Fox	Paramount Pictures	Universal	Warner Bros.	DreamWorks Distribution	Lionsgate	Sony Pictures	D
0	Action	54	62	55	45	81	1	12	49	
1	Adventure	107	73	48	40	68	9	10	39	

2 rows × 40 columns

In [7]:
```
# Create subplots of top Genres for each distributer
count = 10 # No. of genres to display
no_dists = 3 # No. of distributers to display
x=0 #initialize counter

fig, axs = plt.subplots(1, no_dists, figsize=(19,5)) # initialize subplot

# loop through top 3 distributers to create plot for each
for dist in df_top10_dists['Distributor'][1:no_dists+1]: # start at 2nd distribut
    top_genres = df_counts.sort_values(by=dist,ascending=False).Genre[:count] # c
    counts = df_counts.sort_values(by=dist,ascending=False)[dist][:count] # creat
    colors = ['#ffb366', '#ffe066', '#ffff66',
              '#999999','#a6a6a6','#b3b3b3','#bfbfbf','#cccccc','#d9d9d9','#e6e6e

    # create pie subplot for current distributer
    axs[x].pie(counts, labels=top_genres,
               counterclock=False, startangle=-190, colors=colors,
               autopct='%.1f%%', wedgeprops=dict(width=.7))
    axs[x].set_title(dist, fontsize=16, y=-0.1, style='italic')
    x+=1

plt.suptitle('Top 10 Most Popular Genres by Distributor, based on Top 1000 U.S. G
plt.savefig('images/genres_by_distributer.png', dpi=300) # save plot to file for
```

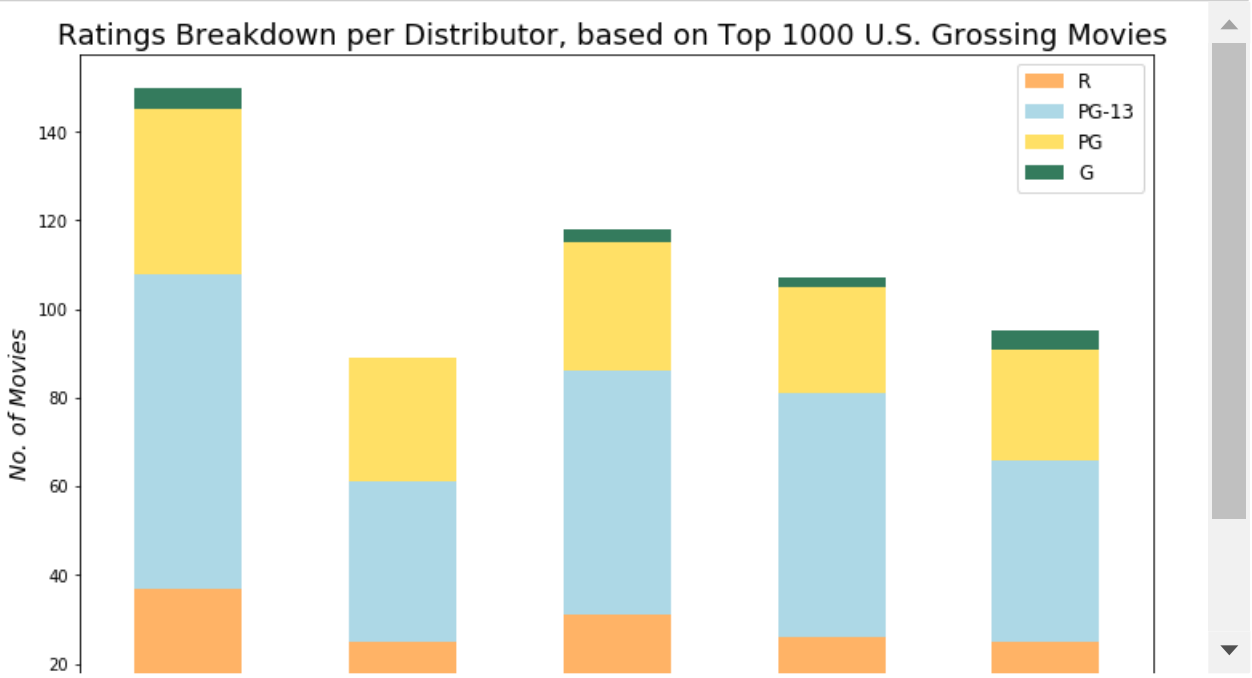


In [8]:
```
# create dictionary of ratings and counts of each rating for each distributor
rating_counts = {}
i=0

# loop through Rating column and count occurences
for rating in df10['Rating']:
    distributor = df10['Distributor'][i]
    if rating=='No Data' or rating=='Approved': # skip no data or outdated rating
        continue
    if distributor not in rating_counts.keys():
        rating_counts[distributor] = {rating : 1}
    if rating not in rating_counts[distributor].keys():
        rating_counts[distributor][rating] = 1
    else:
        rating_counts[distributor][rating] += 1
    i+=1

# Construct and clean dataframe of rating counts
df_rating_counts = pd.DataFrame(rating_counts).transpose().replace(np.nan, 0).asty
df_rating_counts.reset_index(inplace=True)
df_rating_counts = df_rating_counts.rename(columns={'index':'Distributor'})
df_rating_counts = df_rating_counts.replace({'Walt Disney Studios Motion Pictures
                   'Twentieth Century Fox':'20th Century Fox',
                    'Sony Pictures Entertainment (SPE)':'Sony Pictures',
                   'Universal Pictures':'Universal',
                   'New Line Cinema':'New Line',
                   'DreamWorks':'Dreamworks SKG'})

# Join ratings dataframe with top 10 distributers and reorder ratings columns
df_top10_dists_w_ratings = df_top10_dists.set_index('Distributor').join(df_rating_
df_top10_dists_w_ratings = pd.concat([df_top10_dists_w_ratings.iloc[:, 0:4],
                                      df_top10_dists_w_ratings['R'],
                                      df_top10_dists_w_ratings['PG-13'],
                                      df_top10_dists_w_ratings['PG'],
                                      df_top10_dists_w_ratings['G']], axis=1, join='i
df_top10_dists_w_ratings.head(5)
```

Out[8]:

Distributor	Number of Movies	Box Office Total	Ticket Sales	Market Share	R	PG-13	PG	G
Walt Disney	573	$39,691,885,638	5,668,400,077	16.94%	34	77	33	7
Warner Bros.	803	$35,638,946,799	5,133,073,907	15.21%	37	71	37	5
Sony Pictures	730	$28,775,313,514	4,257,932,398	12.28%	25	36	28	0
Universal	511	$27,464,279,056	3,938,555,708	11.72%	31	55	29	3
20th Century Fox	520	$25,854,741,898	3,792,361,616	11.03%	26	55	24	2

In [9]:
```
limit = 5 # no of distributers to display
colors = ['#ffb366', 'lightblue', '#ffe066', '#347B5D']

# create stacked bar graph
ax2 = df_top10_dists_w_ratings[1:limit+1].plot.bar(
                                          y=df_top10_dists_w_ratings.columns,
                                          stacked=True, color=colors, rot=0,
                                          figsize=(12,8))
plt.legend(bbox_to_anchor=(1, 1), fontsize=12)

ax2.set_xlabel('Distributor', fontsize=14, style='italic')
ax2.set_ylabel('No. of Movies', fontsize=14, style='italic')
ax2.set_title('Ratings Breakdown per Distributor, based on Top 1000 U.S. Grossing

plt.savefig('images/ratings_by_distributer.png', dpi=300) # save plot to file for
```



In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
```

In [2]:
```
# LOAD CAT'S DATA
df_movie_3 = pd.read_pickle('cats_bomojo_data.pkl')
df_movie_3.head()
```

Out[2]:

	Rank	Title	Lifetime Gross	Year	URL	Distributor	Bu
0	1	Star Wars: Episode VII - The Force Awakens	936662225	2015	https://www.boxofficemojo.com/title/tt2488496/...	Walt Disney Studios Motion Pictures	24500
1	2	Avengers: Endgame	858373000	2019	https://www.boxofficemojo.com/title/tt4154796/...	Walt Disney Studios Motion Pictures	35600
2	3	Avatar	760507625	2009	https://www.boxofficemojo.com/title/tt0499549/...	Twentieth Century Fox	23700
3	4	Black Panther	700426566	2018	https://www.boxofficemojo.com/title/tt1825683/...	Walt Disney Studios Motion Pictures	No
4	5	Avengers: Infinity War	678815482	2018	https://www.boxofficemojo.com/title/tt4154756/...	Walt Disney Studios Motion Pictures	No

In [3]:
```
# function to get unique values
def unique(list1):

    # intilize a null list
    unique_list = []

    # traverse for all elements
    for x in list1:
        # check if exists in unique_list or not
        if x not in unique_list:
            unique_list.append(x)
    # print list
    for x in unique_list:
        print(x)
#get list of unique distributors
unique(df_movie_3['Distributor'])
```

```
Walt Disney Studios Motion Pictures
Twentieth Century Fox
Paramount Pictures
Universal Pictures
Warner Bros.
DreamWorks Distribution
Lionsgate
Sony Pictures Entertainment (SPE)
DreamWorks
New Line Cinema
Newmarket Films
Summit Entertainment
Columbia Pictures
IFC Films
TriStar Pictures
Metro-Goldwyn-Mayer (MGM)
Orion Pictures
No Data
Miramax
The Weinstein Company
Fox Searchlight Pictures
Revolution Studios
Artisan Entertainment
Sony Pictures Classics
United Artists
Screen Gems
USA Films
STX Entertainment
Dimension Films
AVCO Embassy Pictures
RKO Radio Pictures
United Artists Releasing
FilmDistrict
Focus Features
IMAX
MacGillivray Freeman Films
American International Pictures (AIP)
Relativity Media
Roadside Attractions
```

In [4]:
```
#trim list to top five distributors
dist_data = df_movie_3['Distributor'].isin(['Twentieth Century Fox', 'Sony Pictur
                                            'Universal Pictures', 'Warner Bros.', 'Par
top_distributors = object
top_distributors = df_movie_3[dist_data]
top_distributors
```

Out[4]:

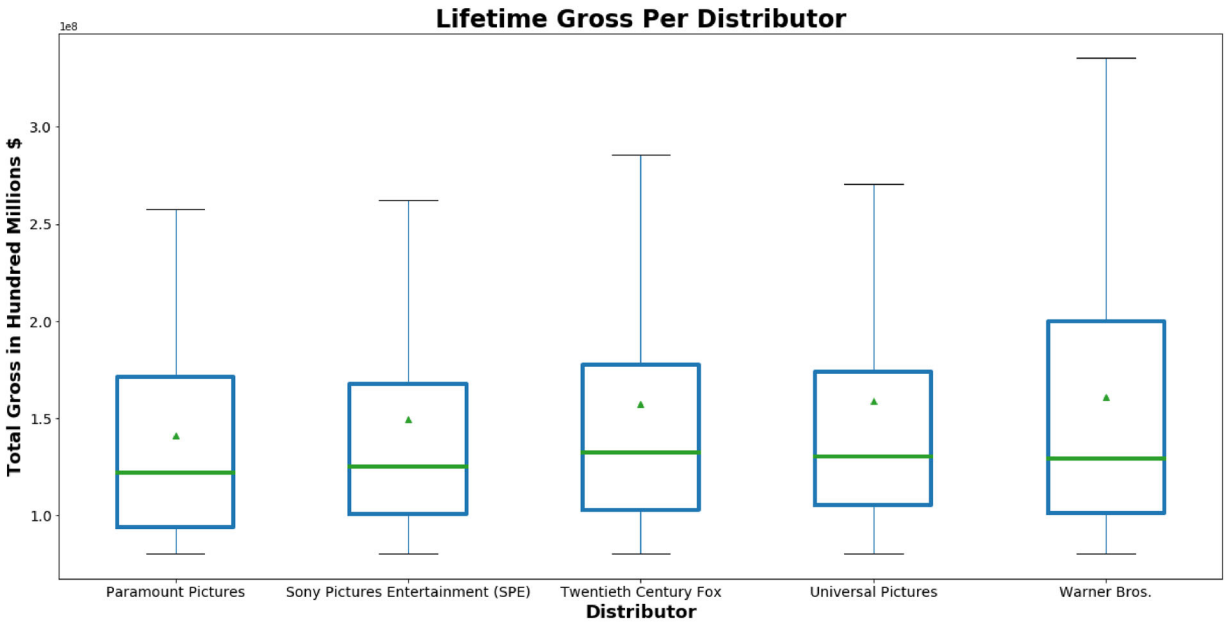
	Rank	Title	Lifetime Gross	Year	URL	Distributor
2	3	Avatar	760507625	2009	https://www.boxofficemojo.com/title/tt0499549/...	Twentieth Century Fox
5	6	Titanic	659363944	1997	https://www.boxofficemojo.com/title/tt0120338/...	Paramount Pictures
6	7	Jurassic World	652270625	2015	https://www.boxofficemojo.com/title/tt0369610/...	Universal Pictures
11	12	The Dark Knight	535234033	2008	https://www.boxofficemojo.com/title/tt0468569/...	Warner Bros.
17	18	Star Wars: Episode I - The Phantom Menace	474544677	1999	https://www.boxofficemojo.com/title/tt0120915/...	Twentieth Century Fox
...	...	...	...	...	...	...
993	994	Jack Reacher	80070736	2012	https://www.boxofficemojo.com/title/tt0790724/...	Paramount Pictures
994	995	Cloverfield	80048433	2008	https://www.boxofficemojo.com/title/tt1060277/...	Paramount Pictures
995	996	Footloose	80035402	1984	https://www.boxofficemojo.com/title/tt0087277/...	Paramount Pictures
997	998	Men in Black: International	80001807	2019	https://www.boxofficemojo.com/title/tt2283336/...	Sony Pictures Entertainment (SPE)
998	999	A Star Is Born	80000000	1976	https://www.boxofficemojo.com/title/tt0075265/...	Warner Bros.

640 rows × 10 columns

In [5]:
```
#import necessary libraries
import matplotlib.pyplot as plt

#format and label the plot
boxprops = dict(linestyle='-', linewidth=4, color='k')
medianprops = dict(linestyle='-', linewidth=4, color='k')

bp = top_distributors.boxplot(by ='Distributor', column =['Lifetime Gross'], grid
                              showfliers=False, showmeans=True, boxprops=boxprops, med
bp.set_xlabel('Distributor', fontweight='bold', fontsize=18)
bp.set_ylabel('Total Gross in Hundred Millions $', fontweight='bold', fontsize=18
bp.set_title('Lifetime Gross Per Distributor', fontweight='bold', fontsize=24)
bp.tick_params(axis='y', labelsize=14)
bp.tick_params(axis='x', labelsize=14)

#show the plot
plt.savefig('images/lifetime_gross_per_distributor.png', dpi=300)
plt.suptitle("")
```

```
C:\Users\cm_fr\anaconda3\envs\learn-env\lib\site-packages\numpy\core\_asarray.p
y:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequenc
es (which is a list-or-tuple of lists-or-tuples-or ndarrays with different leng
ths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=
object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

Out[5]: Text(0.5, 0.98, '')



In [6]:
```
#get mean, median, Q1, Q3 and IQR for each distributor
```

In [7]:
```
paramount_df = top_distributors.loc[top_distributors['Distributor'] == 'Paramount
para_mean = paramount_df['Lifetime Gross'].mean()
para_median = paramount_df['Lifetime Gross'].median()
para_q1 = paramount_df['Lifetime Gross'].quantile(.25)
para_q3 = paramount_df['Lifetime Gross'].quantile(.75)
para_iqr = para_q3 - para_q1
print(para_mean)
print(para_median)
print(para_q1)
print(para_q3)
print(para_iqr)
```

```
141254524.95726496
121697323.0
93617009.0
171243005.0
77625996.0
```

In [8]:
```
sony_df = top_distributors.loc[top_distributors['Distributor'] == 'Sony Pictures
sony_mean = sony_df['Lifetime Gross'].mean()
sony_median = sony_df['Lifetime Gross'].median()
sony_q1 = sony_df['Lifetime Gross'].quantile(.25)
sony_q3 = sony_df['Lifetime Gross'].quantile(.75)
sony_iqr = sony_q3 - sony_q1
print(sony_mean)
print(sony_median)
print(sony_q1)
print(sony_q3)
print(sony_iqr)
```

```
149112273.0
124799506.5
100424808.75
167724618.75
67299810.0
```

In [9]:
```
fox_df = top_distributors.loc[top_distributors['Distributor'] == 'Twentieth Centu
fox_mean = fox_df['Lifetime Gross'].mean()
fox_median = fox_df['Lifetime Gross'].median()
fox_q1 = fox_df['Lifetime Gross'].quantile(.25)
fox_q3 = fox_df['Lifetime Gross'].quantile(.75)
fox_iqr = fox_q3 - fox_q1
print(fox_mean)
print(fox_median)
print(fox_q1)
print(fox_q3)
print(fox_iqr)
```

```
157176453.81538463
132314889.0
102338632.25
177359062.75
75020430.5
```

In [10]:
```
uni_df = top_distributors.loc[top_distributors['Distributor'] == 'Universal Pictu
uni_mean = uni_df['Lifetime Gross'].mean()
uni_median = uni_df['Lifetime Gross'].median()
uni_q1 = uni_df['Lifetime Gross'].quantile(.25)
uni_q3 = uni_df['Lifetime Gross'].quantile(.75)
uni_iqr = uni_q3 - fox_q1
print(uni_mean)
print(uni_median)
print(uni_q1)
print(uni_q3)
print(uni_iqr)
```

```
158504266.1968504
130164645.0
105083630.0
173897434.0
71558801.75
```

In [11]:
```
warner_df = top_distributors.loc[top_distributors['Distributor'] == 'Warner Bros.
warner_mean = warner_df['Lifetime Gross'].mean()
warner_median = warner_df['Lifetime Gross'].median()
warner_q1 = warner_df['Lifetime Gross'].quantile(.25)
warner_q3 = warner_df['Lifetime Gross'].quantile(.75)
warner_iqr = warner_q3 - warner_q1
print(warner_mean)
print(warner_median)
print(warner_q1)
print(warner_q3)
print(warner_iqr)
```

```
160660839.01807228
128880039.5
100947305.0
199560973.25
98613668.25
```