Investigate_a_Dataset.py

June 4, 2020

In [1]:
```
#Data Wrangling and Cleaning
```

In [22]:
```
#Importing required Modules

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [9]:
```
#extracting csv (tmdb-movies datasets)
df=pd.read_csv('tmdb-movies.csv')
df.head(10)
```

Out[9]:

	id	imdb_id	popularity	budget	revenue	original_title
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7
5	281957	tt1663202	9.110700	135000000	532950503	The Revenant
6	87101	tt1340138	8.654359	155000000	440603537	Terminator Genisys
7	286217	tt3659388	7.667400	108000000	595380321	The Martian
8	211672	tt2293640	7.404165	74000000	1156730962	Minions
9	150540	tt2096673	6.326804	175000000	853708609	Inside Out

```
                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...
5  Leonardo DiCaprio|Tom Hardy|Will Poulter|Domhn...
6  Arnold Schwarzenegger|Jason Clarke|Emilia Clar...
7  Matt Damon|Jessica Chastain|Kristen Wiig|Jeff ...
8  Sandra Bullock|Jon Hamm|Michael Keaton|Allison...
9  Amy Poehler|Phyllis Smith|Richard Kind|Bill Ha...

                                        homepage  \
0                     http://www.jurassicworld.com/
1                       http://www.madmaxmovie.com/
2     http://www.thedivergentseries.movie/#insurgent
3  http://www.starwars.com/films/star-wars-episod...
4                            http://www.furious7.com/
5       http://www.foxmovies.com/movies/the-revenant
6                    http://www.terminatormovie.com/
7         http://www.foxmovies.com/movies/the-martian
8                      http://www.minionsmovie.com/
9                http://movies.disney.com/inside-out

                          director  \
0                  Colin Trevorrow
1                    George Miller
2                 Robert Schwentke
3                      J.J. Abrams
4                        James Wan
5  Alejandro GonzÃąlez IÃńÃąrritu
6                       Alan Taylor
7                      Ridley Scott
8         Kyle Balda|Pierre Coffin
9                       Pete Docter

                                       tagline      ...       \
0                              The park is open.     ...
1                             What a Lovely Day.     ...
2                       One Choice Can Destroy You   ...
3                    Every generation has a story.   ...
4                              Vengeance Hits Home    ...
5  (n. One who has returned, as if from the dead.)    ...
6                                Reset the future     ...
7                                  Bring Him Home     ...
8     Before Gru, they had a history of bad bosses    ...
9          Meet the little voices inside your head.  ...
```

```
                                            overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137
5  In the 1820s, a frontiersman, Hugh Glass, sets...     156
6  The year is 2029. John Connor, leader of the r...     125
7  During a manned mission to Mars, Astronaut Mar...     141
8  Minions Stuart, Kevin and Bob are recruited by...      91
9  Growing up can be a bumpy road, and it's no ex...      94

                                       genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                     Action|Crime|Thriller
5           Western|Drama|Adventure|Thriller
6  Science Fiction|Action|Thriller|Adventure
7            Drama|Adventure|Science Fiction
8          Family|Animation|Adventure|Comedy
9                   Comedy|Animation|Family

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947
5  Regency Enterprises|Appian Way|CatchPlay|Anony...     12/25/15       3929
6            Paramount Pictures|Skydance Productions      6/23/15       2598
7  Twentieth Century Fox Film Corporation|Scott F...      9/30/15       4572
8      Universal Pictures|Illumination Entertainment      6/17/15       2893
9  Walt Disney Pictures|Pixar Animation Studios|W...       6/9/15       3935

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09
5           7.2          2015  1.241999e+08  4.903142e+08
6           5.8          2015  1.425999e+08  4.053551e+08
7           7.6          2015  9.935996e+07  5.477497e+08
8           6.5          2015  6.807997e+07  1.064192e+09
9           8.0          2015  1.609999e+08  7.854116e+08
```

```
[10 rows x 21 columns]
```

In [8]:
```
df.head(5)
#note: You can ingnore this.
```

Out[8]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...	overview	runtime	genres
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	...	Twenty-two years after the events of Jurassic ...	124	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	...	An apocalyptic story set in the furthest reach...	120	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	...	Beatrice Prior must confront her inner demons ...	119	
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams	Every generation has a story.	...	Thirty years after defeating the Galactic Empi...	136	
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Vengeance Hits Home	...	Deckard Shaw seeks revenge against Dominic Tor...	137	

```
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                      Action|Crime|Thriller

                                production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3          Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

In [10]:
```
#What are the unique column names present? check?

df.columns.unique()
```

Out[10]:
```
Index(['id', 'imdb_id', 'popularity', 'budget', 'revenue', 'original_title',
       'cast', 'homepage', 'director', 'tagline', 'keywords', 'overview',
       'runtime', 'genres', 'production_companies', 'release_date',
       'vote_count', 'vote_average', 'release_year', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

In [11]:
```
#Gathering statistical Data of given dataset
df.describe()
```

Out[11]:

	id	popularity	budget	revenue	runtime
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000

	vote_count	vote_average	release_year	budget_adj	revenue_adj
count	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086600e+04
mean	217.389748	5.974922	2001.322658	1.755104e+07	5.136436e+07

```
std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

In [12]:
```
#data type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

In [13]:
```
#Is there any duplicate rows present?,check

df[df.duplicated()]
```

Out[13]:
```
          id    imdb_id  popularity    budget  revenue original_title  \
2090  42194  tt0411951     0.59643  30000000   967000         TEKKEN

                                                 cast homepage  \
2090  Jon Foo|Kelly Overton|Cary-Hiroyuki Tagawa|Ian...      NaN

                director              tagline     ...       \
```

```
2090  Dwight H. Little  Survival is no game     ...

                                                 overview runtime  \
2090  In the year of 2039, after World Wars destroy ...      92

                                              genres    production_companies  \
2090  Crime|Drama|Action|Thriller|Science Fiction  Namco|Light Song Films

     release_date vote_count  vote_average  release_year  budget_adj  \
2090      3/20/10        110           5.0          2010  30000000.0

      revenue_adj
2090     967000.0

[1 rows x 21 columns]
```

In [14]:
```
#Dropping the Duplicated rows
df.drop_duplicates(inplace=True)
```

In [15]:
```
#there any missing Values in the Dataset
df[df==0].count()
```

Out[15]:
```
id                         0
imdb_id                    0
popularity                 0
budget                  5696
revenue                 6016
original_title             0
cast                       0
homepage                   0
director                   0
tagline                    0
keywords                   0
overview                   0
runtime                   31
genres                     0
production_companies       0
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj              5696
revenue_adj             6016
dtype: int64
```

In [17]:
```
#Replacing the 0 values with Numpy Data type NaN
df['budget'].replace(0, np.NaN, inplace = True)
df['revenue'].replace(0, np.NaN, inplace = True)
df['runtime'].replace(0, np.NaN, inplace = True)
```

```
df['budget_adj'].replace(0, np.NaN, inplace = True)
df['revenue_adj'].replace(0, np.NaN, inplace = True)
```

In [18]:
```
#Now dropping these NA values because any other technique(replacing with mean, last val
df.dropna(axis=0, inplace=True)
```

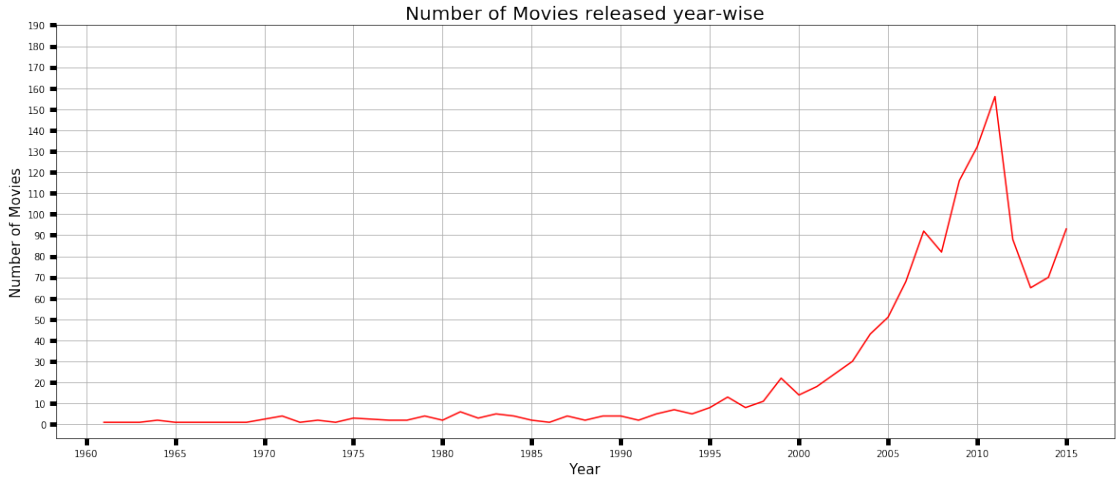
In [19]:
```
#Splitting the Cast, Director and Genres columns to extract the data
df_cast = (df['cast'].str.split('|', expand=True).rename(columns=lambda x: f"cast_{x+1}
df_dir = (df['director'].str.split('|', expand=True).rename(columns=lambda x: f"directo
df_gen = (df['genres'].str.split('|', expand=True).rename(columns=lambda x: f"genres_{x
df_prod = (df['production_companies'].str.split('|', expand=True).rename(columns=lambda
```

In [20]:
```
#Removing the old columns and joining back the new columns
df = df.drop(['cast', 'director', 'genres', 'production_companies'], axis = 1)
df = df.join([df_cast, df_dir, df_gen, df_prod])
```

In [21]:
```
#Changing Release date type to Standard Datetime format
df.release_date = pd.to_datetime(df['release_date'])

#And the float64 types to int64 types for ease of plotting
col = ['popularity', 'budget', 'revenue', 'vote_average']
df[col] = df[col].astype(np.int64)
```

In [ ]:
```
#Exploratory Data Analysis of dataset
```

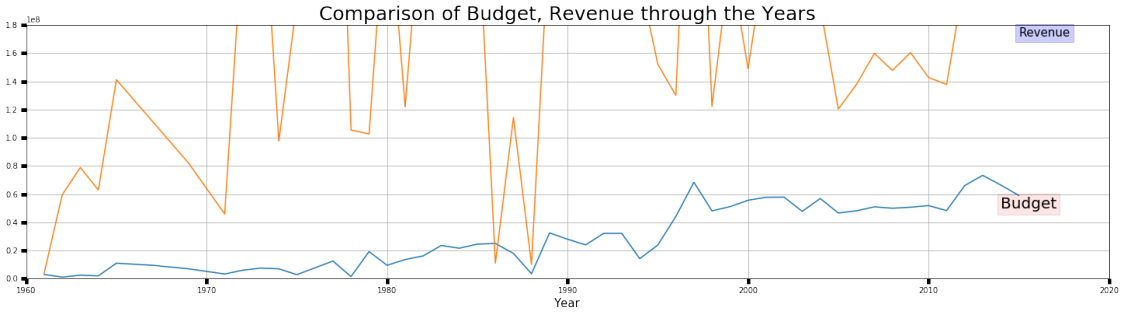
In [23]:
```
#Number of movies released year-wise
data = df.groupby('release_year').count()['original_title']

params = {'figure.figsize': (20, 8), 'axes.labelsize':15, 'axes.titlesize':20}
plt.rcParams.update(params)

data.plot(xticks=np.arange(1960, 2020, 5), yticks=np.arange(0, 200, 10), c='r')

plt.title('Number of Movies released year-wise')
plt.ylabel('Number of Movies')
plt.xlabel('Year')

plt.tick_params(which='both', width=5)
plt.tick_params(which='major', length=7)

plt.grid(True, which='both', axis='both')

plt.show()
```



In [24]:
```
#FROM 1960 to 2015: 2000% growth rate around 65% growth rate Y-O-Y From 1960 -> 1980:
#It has increased about 500% From 1980 -> 2000:
#It increased 1500% From 2000 -> 2015: It increased 95%
```

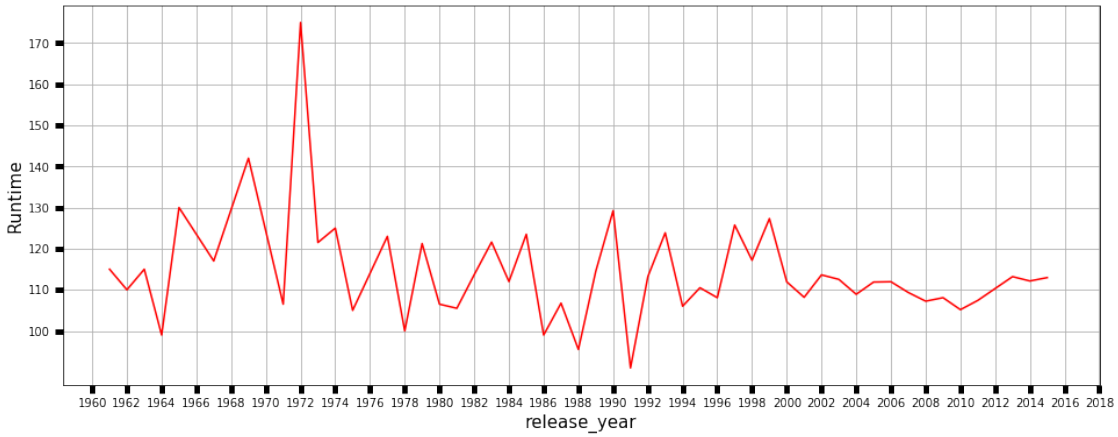
In [26]:
```
#Comparison of Budget, Revenue, Profit through the years
```

In [38]:
```
(df.groupby('release_year')['budget']).mean().plot()
(df.groupby('release_year')['revenue']).mean().plot()

params = {'figure.figsize': (25, 6), 'axes.labelsize':15, 'axes.titlesize':25}
plt.rcParams.update(params)

plt.title('Comparison of Budget, Revenue through the Years')
plt.xlabel('Year')

ax = plt.axis([1960, 2020, 1, 180000000])

plt.text(2015, 170000000, 'Revenue', verticalalignment='bottom', fontsize=15, bbox=dict

plt.text(2014, 50000000, 'Budget', fontsize=20, bbox=dict(facecolor='red', alpha=0.1))

plt.tick_params(which='both', width=5)
plt.tick_params(which='major', length=7)

plt.grid(True, which='both', axis='both')

plt.show()
```



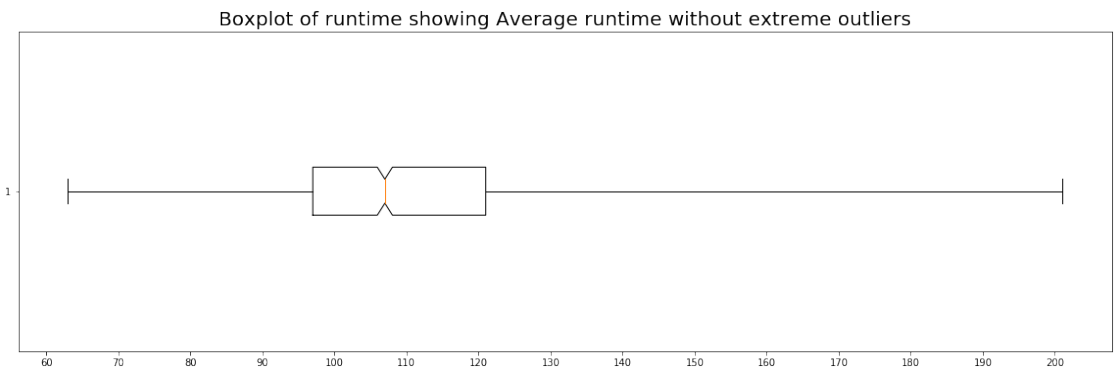
In [39]:
```
# Movie with Highest and Lowest Stats across different Parameters

def maxmin(val):

    min1 = df[val].idxmin()
    max1 = df[val].idxmax()

    max2 = pd.DataFrame(df.loc[max1,:])
    max2.columns = [df['original_title'][max1]]
    min2 = pd.DataFrame(df.loc[min1,:])
    min2.columns = [df['original_title'][min1]]

    print(color.BOLD + color.RED)
    print(''.join(['Movie with Highest ' + val + ' --> ', df['original_title'][max1]]),
    print(df[val][max1])
    print(''.join(['Movie with Lowest ' + val + ' --> ', df['original_title'][min1]]),
    print(df[val][min1])
```

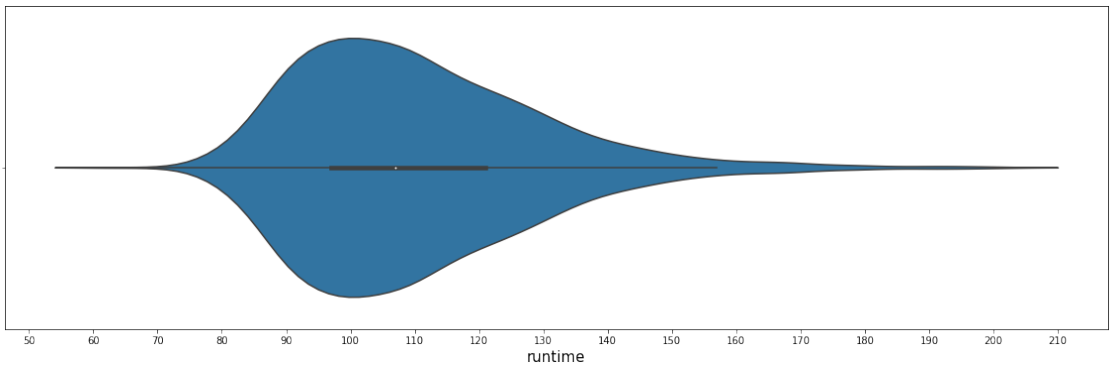
In [41]:
```
#Trend of runtime of movies from 1960-2015

po = df.groupby('release_year')['runtime'].mean()

params = {'figure.figsize': (20, 8), 'axes.labelsize':15, 'axes.titlesize':20}

plt.rcParams.update(params)
plt.title('')
plt.ylabel('Runtime')

po.plot(figsize=(16, 6), xticks=np.arange(1960, 2020, 2), yticks=np.arange(100, 180, 10

plt.tick_params(which='both', width=5)
plt.tick_params(which='major', length=7)

plt.grid(True, which='both', axis='both')

plt.show()
```



In [42]:
```
#Average Runtime of Movies
```

```
params = {'figure.figsize': (20, 6), 'axes.labelsize':15, 'axes.titlesize':20}
plt.rcParams.update(params)

plt.grid(False)

plt.title('Boxplot of runtime showing Average runtime along with outliers')

plt.title('Boxplot of runtime showing Average runtime without extreme outliers')

plt.xticks(np.arange(0, 350, 10))
plt.boxplot(df.runtime, notch=True, vert=False, whis=5, showfliers=False, autorange=Tru
plt.show()

plt.xticks(np.arange(0, 350, 10))

plt.grid(False)
sns.violinplot(df.runtime, notch=True, vert=False, whis=5, showfliers=True, autorange=T
plt.show()
```

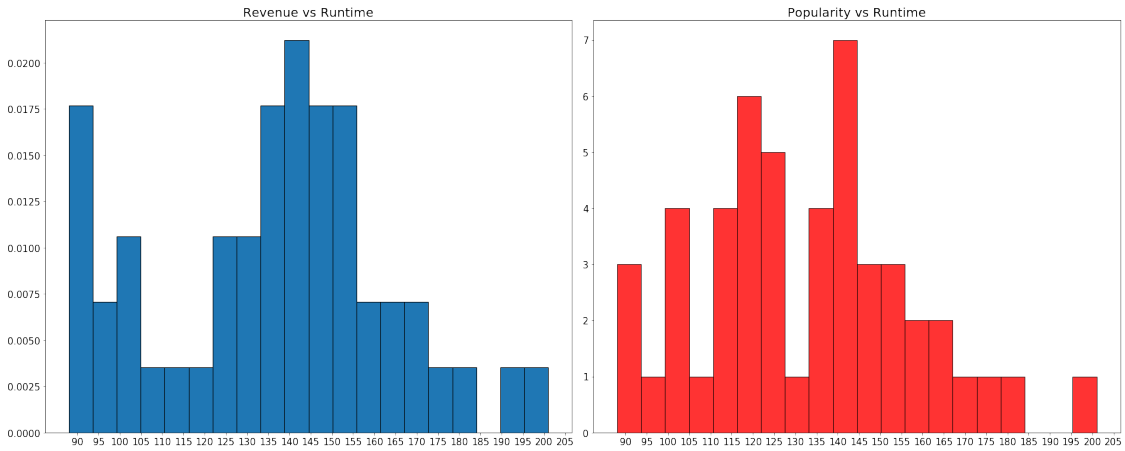




In [44]: *#How does Popularity and Revenue depend on Runtime?#*

```
test5 = df.sort_values(by='popularity', ascending=False).head(50)
test6 = df.sort_values(by='revenue', ascending=False).head(50)

params = {'figure.figsize': (25, 10), 'axes.labelsize':15, 'axes.titlesize':20}
plt.rcParams.update(params)

fig, (ax1, ax2) = plt.subplots(1, 2, sharex=True, sharey=False)

ax1.hist(test6.runtime, bins=20, density=True, edgecolor="k")
ax1.set_title('Revenue vs Runtime')

ax2.hist(test5.runtime, bins=20, color='Red', alpha=0.8, edgecolor="k")
ax2.set_title('Popularity vs Runtime')

plt.sca(ax1)
ax1.grid(False)
plt.yticks(fontsize=15)
plt.xticks(np.arange(90, 210, 5), fontsize=15)

plt.sca(ax2)
ax2.grid(False)
plt.yticks(fontsize=15)
plt.xticks(np.arange(90, 210, 5), fontsize=15)

plt.tight_layout()
```

```
        plt.show()
```



In [45]: *#Comparison of Avg. Runtime and Popularity across different Genres*

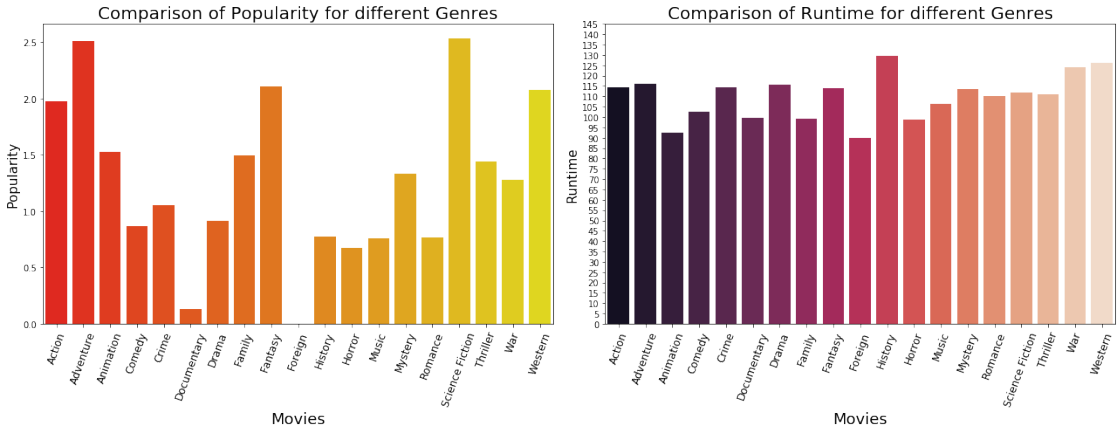
In [46]: *#Here we will create a new dataframe using Melt function to compare Genres Stats'*

```
        gen_cols = ['genres_1','genres_2', 'genres_3', 'genres_4', 'genres_5']
        non_gen_cols = ['popularity', 'ROI', 'budget_US_dollars', 'revenue_US_dollars', 'profit
                        'vote_count', 'vote_average', 'runtime', 'release_year']
        gen_types = df.melt(id_vars=non_gen_cols, value_vars=gen_cols, var_name='gen_type', val
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:1472: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-listlike
  return self._getitem_tuple(key)
```

In [47]:
```
        a = gen_types.groupby('type')['popularity'].mean()
        b = a.index

        c = gen_types.groupby('type')['runtime'].mean()
        d = c.index

        fig, ax = plt.subplots(nrows=1, ncols=2, figsize = [18,7], sharex=True)

        sns.barplot(b, a, ax=ax[0], palette='autumn')
        sns.barplot(d, c, palette='rocket')
```

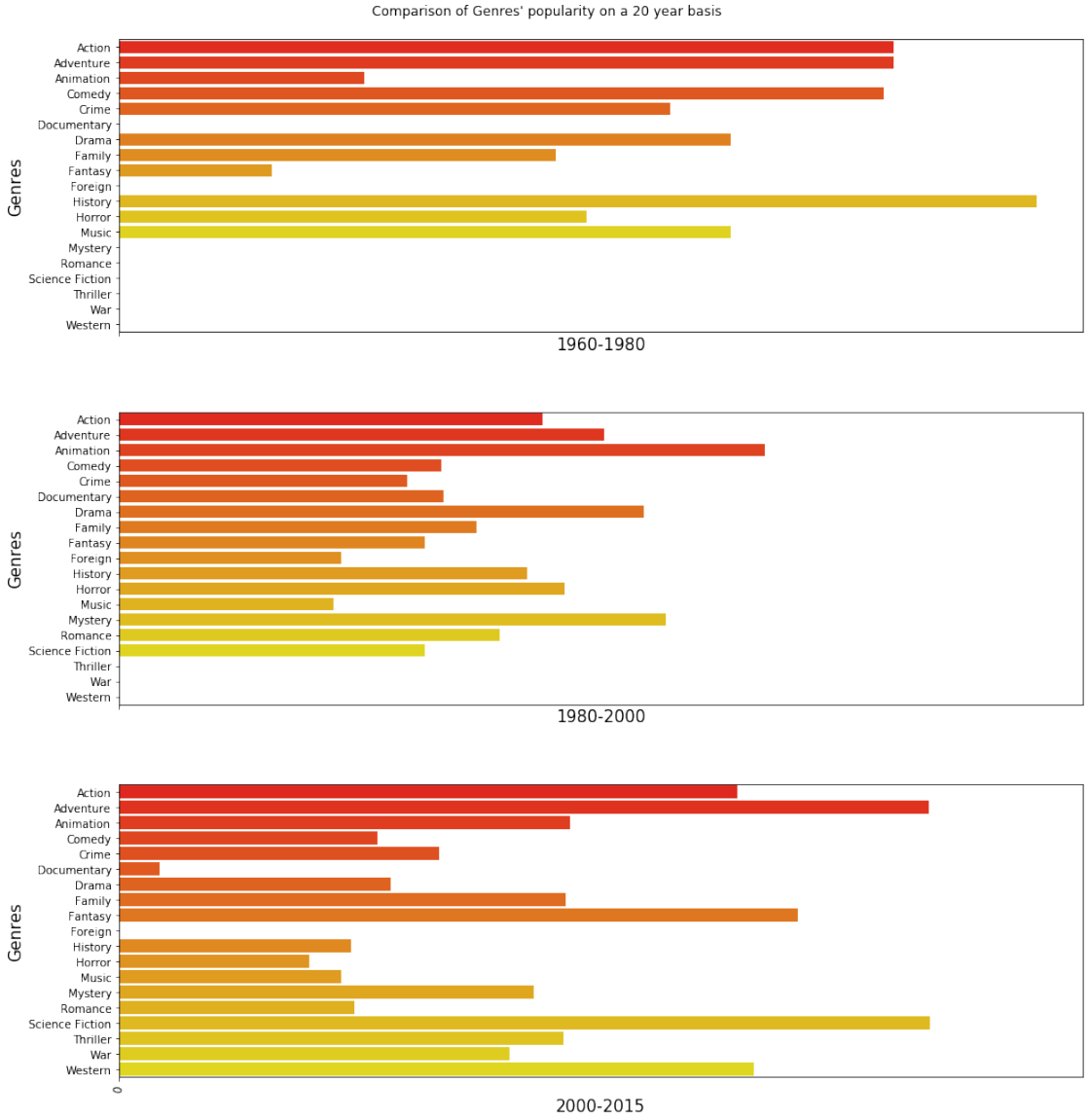
```
plt.sca(ax[0])
plt.xticks(fontsize=12, rotation=70)
plt.title('Comparison of Popularity for different Genres')
plt.xlabel('Movies', fontsize=18)
plt.ylabel('Popularity')

plt.sca(ax[1])
plt.xticks(fontsize=12, rotation=70)
plt.yticks(np.arange(0, 150, 5))
#Smaller Yticks were considered to better compare for trends
plt.title('Comparison of Runtime for different Genres')
plt.xlabel('Movies', fontsize=18)
plt.ylabel('Runtime')

plt.tight_layout()
plt.show()
```



In [52]: *#Which Genres were most popular over the years compared on a year-on-year basis*

In [50]:
```
fig, ax = plt.subplots(nrows=3, ncols=1, figsize=(15, 15), sharex=True, sharey=True)

plt.suptitle("Comparison of Genres' popularity on a 20 year basis")

gen_sub = gen_types.loc[gen_types['release_year'].between(1960, 1980)]
a = gen_sub.groupby('type')['popularity'].mean()
b = a.index
sns.barplot(a, b, ax=ax[0], palette='autumn')

gen_sub = gen_types.loc[gen_types['release_year'].between(1980, 2000)]
a = gen_sub.groupby('type')['popularity'].mean()
b = a.index
```

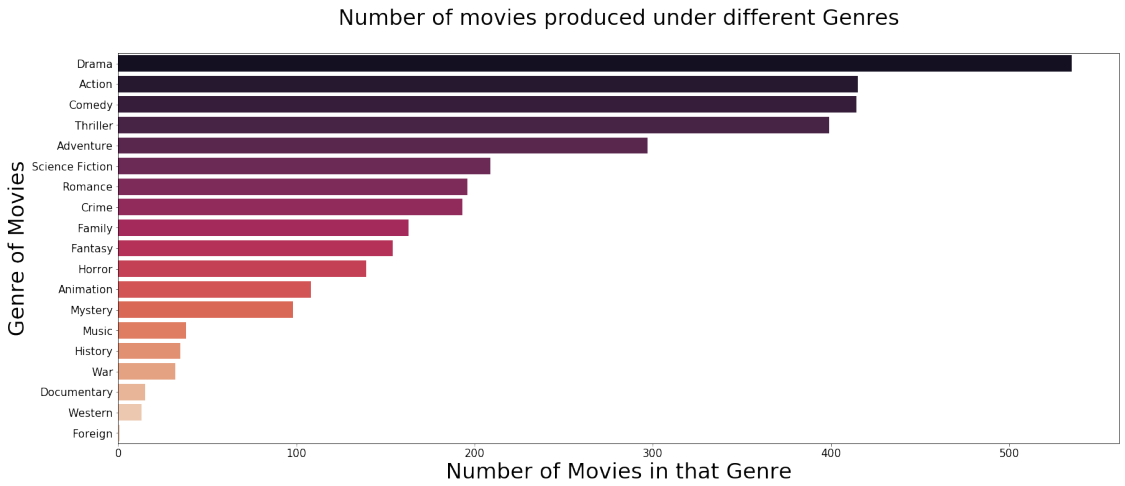
```
sns.barplot(a, b, ax=ax[1], palette='autumn')

gen_sub = gen_types.loc[gen_types['release_year'].between(2000, 2015)]
a = gen_sub.groupby('type')['popularity'].mean()
b = a.index
sns.barplot(a, b, ax=ax[2], palette='autumn')

plt.sca(ax[0])
plt.xticks(rotation=80)
plt.xlabel('1960-1980')
plt.ylabel('Genres')
plt.sca(ax[1])
plt.xticks(rotation=80)
plt.xlabel('1980-2000')
plt.ylabel('Genres')
plt.sca(ax[2])
plt.xticks(rotation=80)
plt.xlabel('2000-2015')
plt.ylabel('Genres')
plt.xticks(np.arange(0, 100, 100))

plt.tight_layout(pad=5)
plt.show()
```



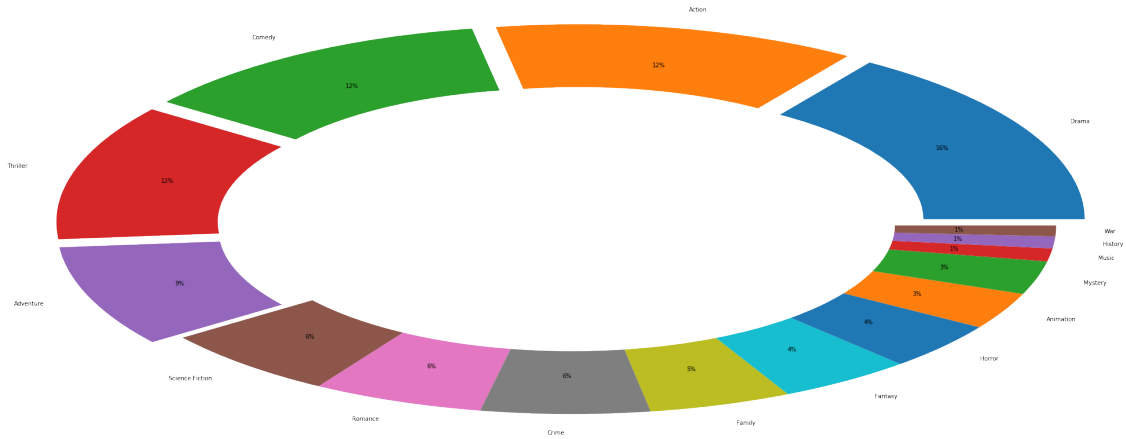
In [53]: *#Distribution of movies across Genres*

In [54]:
```
type_counts = gen_types['type'].value_counts()
type_order = type_counts.index

params = {'figure.figsize': (25, 10), 'axes.labelsize':15, 'axes.titlesize':20}
plt.rcParams.update(params)

sns.countplot(data=gen_types, y='type', order=type_order, palette='rocket')

plt.title('\nNumber of movies produced under different Genres\n', fontsize=30)
```

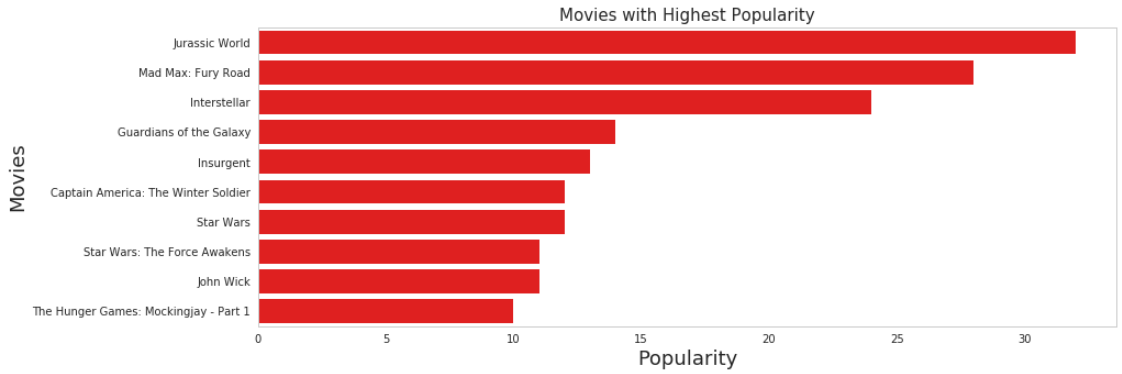
```
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

plt.xlabel('Number of Movies in that Genre', fontsize=30)
plt.ylabel('Genre of Movies', fontsize=30)
plt.show()
```

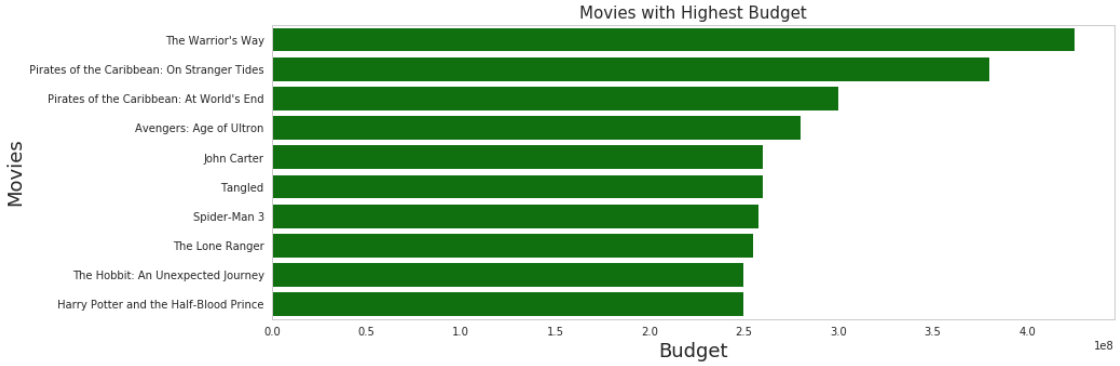


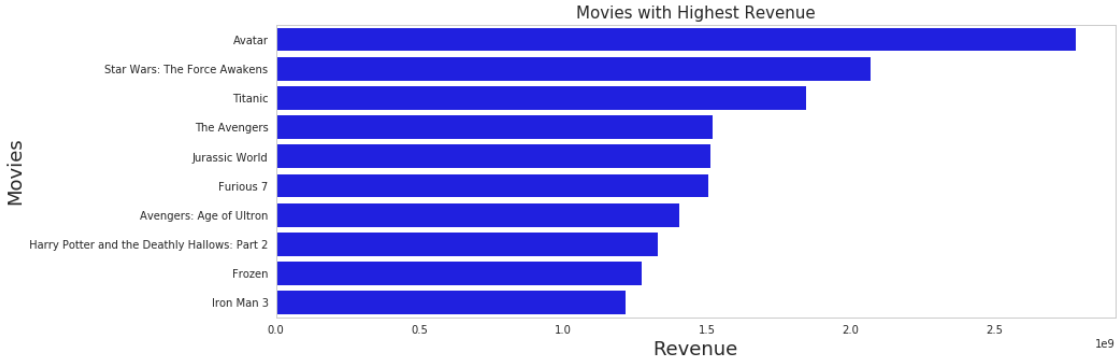
In [69]:
```
labels = gen_types['type'].value_counts()[:16].index
sizes = gen_types['type'].value_counts()[:16]

explode = (0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, radius=1.5, explode=explode, labels=labels, labeldistance = 1.1, pctdist
        wedgeprops=dict(width=0.5), startangle=0, autopct='%1.0f%%', shadow=False)

plt.show()
```

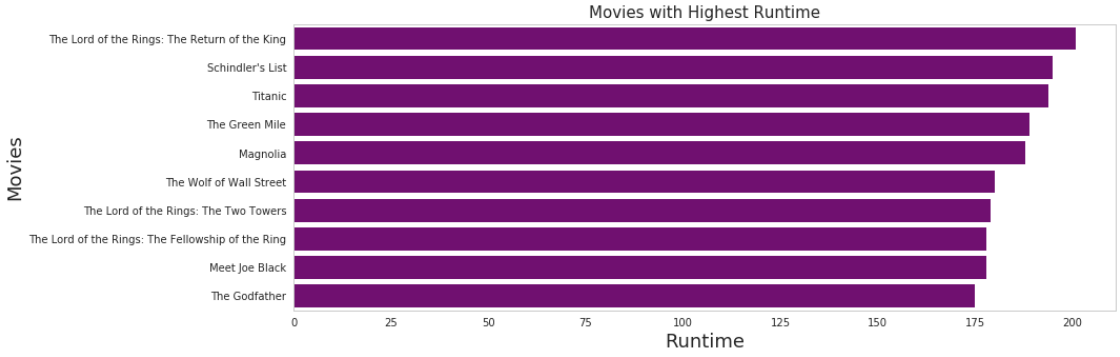


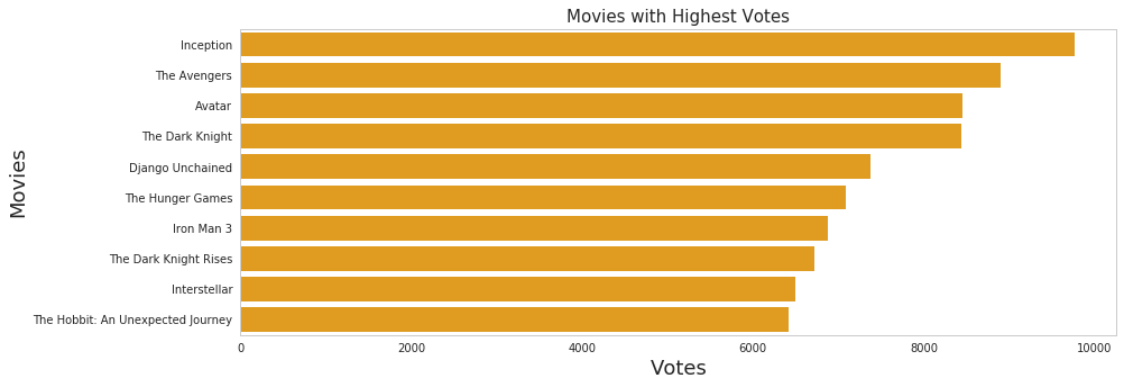
```
In [ ]: #Top 10 movies across different parameters

In [80]: def top10(val):

             info = (df[val].sort_values(ascending=False)).head(10)

             arr = []
             for i in info.index.values:
                 arr.append((df['original_title'][i], info[i]))

             arr = pd.DataFrame(arr)

             choose = {'popularity':'red', 'budget':'green', 'revenue':'blue', \
                       'runtime':'purple', 'vote_count':'orange', 'ROI':'violet'}
             title = {'popularity':"Popularity", 'budget':'Budget', 'revenue':'Revenue', \
                       'runtime':'Runtime', 'vote_count':'Votes'}

             ax = sns.barplot(x=arr[1], y=arr[0], data=arr, color = choose[val])

             sns.set(rc={'figure.figsize':(14, 5)})
             ax.set_title('Movies with Highest ' + title[val], fontsize = 15)
             ax.set_xlabel(title[val], fontsize = 18)
             ax.set_ylabel("Movies", fontsize = 18)
             ax.grid(False)
             sns.set_style("whitegrid")
             plt.show()


In [81]: top10('popularity')
         top10('budget')
         top10('revenue')
         top10('runtime')
         top10('vote_count')
```




Movies with Highest Budget
The Warrior's Way
Pirates of the Caribbean: On Stranger Tides
Pirates of the Caribbean: At World's End
Avengers: Age of Ultron
John Carter
Tangled
Spider-Man 3
The Lone Ranger
The Hobbit: An Unexpected Journey
Harry Potter and the Half-Blood Prince
Movies
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e8
Budget


Movies with Highest Revenue
Avatar
Star Wars: The Force Awakens
Titanic
The Avengers
Jurassic World
Furious 7
Avengers: Age of Ultron
Harry Potter and the Deathly Hallows: Part 2
Frozen
Iron Man 3
Movies
0.0
0.5
1.0
1.5
2.0
2.5
1e9
Revenue

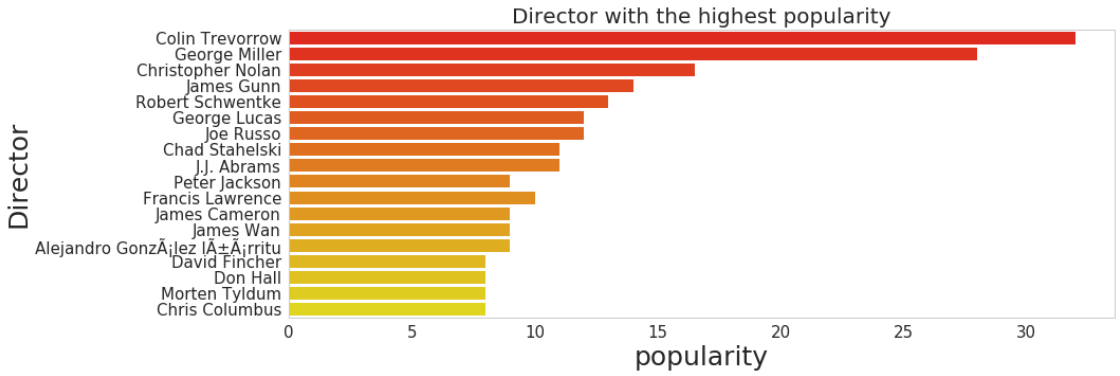

Movies with Highest Runtime
The Lord of the Rings: The Return of the King
Schindler's List
Titanic
The Green Mile
Magnolia
The Wolf of Wall Street
The Lord of the Rings: The Two Towers
The Lord of the Rings: The Fellowship of the Ring
Meet Joe Black
The Godfather
Movies
0
25
50
75
100
125
150
175
200
Runtime

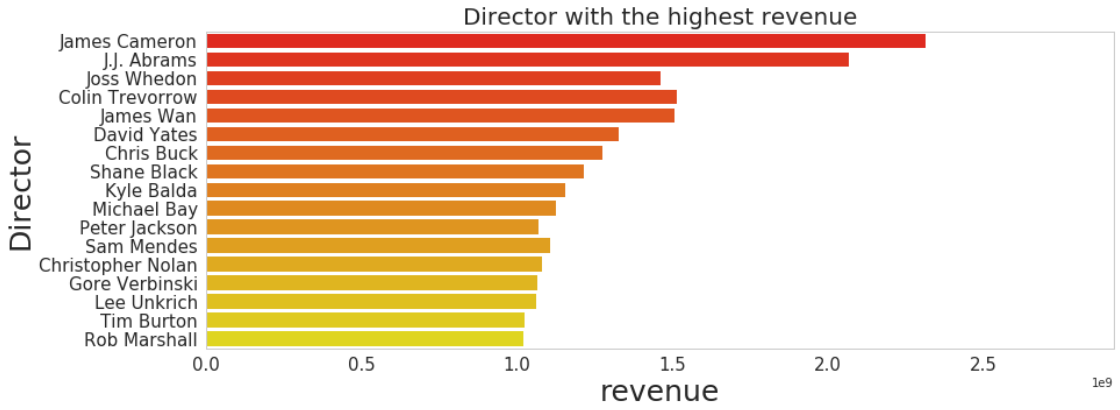


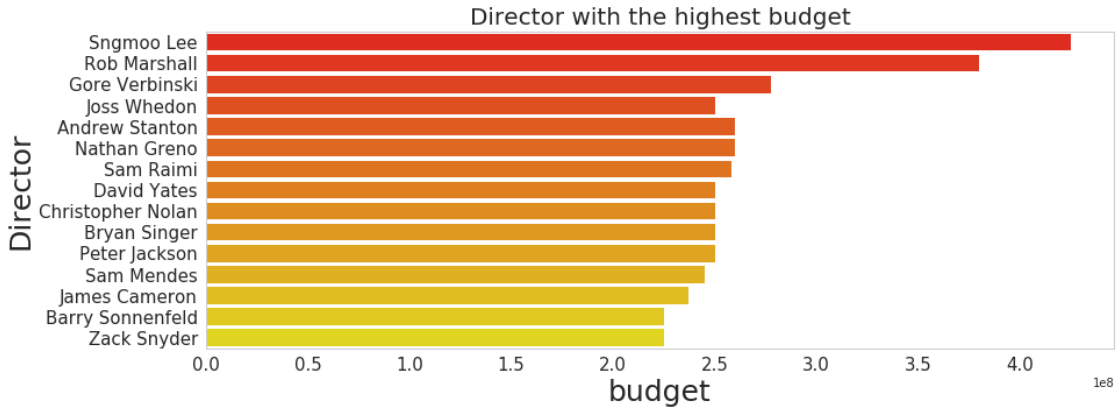
In [ ]: *#TOP DIRECTORS ACROSS DIFFERENT PARAMETERS*

In [82]:
```
def direct(val):

    test5 = df.sort_values(by=val, ascending=False)

    sns.barplot(test5[val].iloc[:20], test5.director_1.iloc[:20], palette='autumn', err

    params = {'figure.figsize': (14, 5), 'axes.labelsize':15, 'axes.titlesize':20}

    plt.rcParams.update(params)

    plt.title("Director with the highest " + val, fontsize=20)
    plt.xlabel(val, fontsize=25)
    plt.yticks(fontsize=15)
    plt.ylabel('Director', fontsize=25)
    plt.xticks(fontsize=15)
    plt.grid(False)


    plt.show()
```

In [84]:
```
#COMMOND
direct('popularity')
direct('revenue')
direct('budget')
```





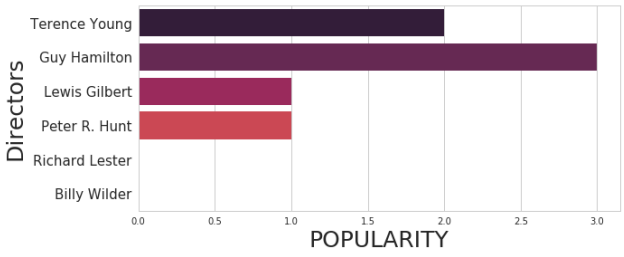


In [85]: *#Top Directors of the decade from 1960 to 2015*

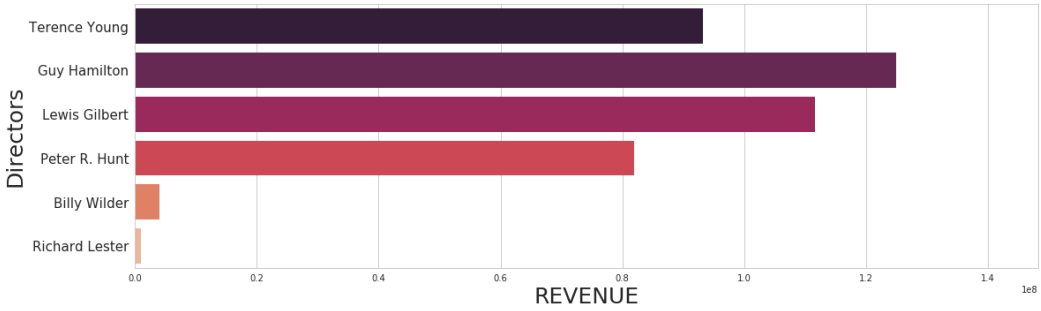
In [86]: def dir_year(val1, val2, val3):

```
    a = df[df.release_year.between(val2, val3, inclusive=True)].sort_values(by=val1, as
    b = df[df.release_year.between(val2, val3, inclusive=True)].sort_values(by=val1, as

    sns.barplot(a, b, palette='rocket', errwidth=False)

    params = {'figure.figsize': (18, 6), 'axes.labelsize':15, 'axes.titlesize':20}

    plt.rcParams.update(params)

    plt.title('Top 20 Directors of every decade starting from '+str(val2)+' to '+str(va
              ' based on ' + val1 + '\n',
              fontsize=30)

    plt.xlabel(val1.upper(), fontsize=25)
    plt.ylabel('Directors', fontsize=25)
    plt.yticks(fontsize=15)

    plt.tight_layout()
    plt.show()
```

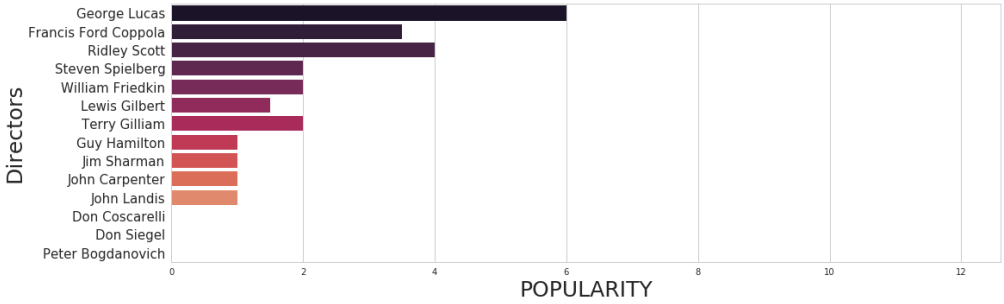
In [87]:
```
#commonds

dir_year('popularity', 1960, 1969)
dir_year('revenue', 1960, 1969)
dir_year('popularity', 1970, 1979)
dir_year('revenue', 1970, 1979)
dir_year('popularity', 1980, 1989)
dir_year('revenue', 1980, 1989)
dir_year('popularity', 1990, 1999)
dir_year('revenue', 1990, 1999)
dir_year('popularity', 2000, 2009)
dir_year('revenue', 2000, 2009)
dir_year('popularity', 2010, 2015)
dir_year('revenue', 2010, 2015)
```



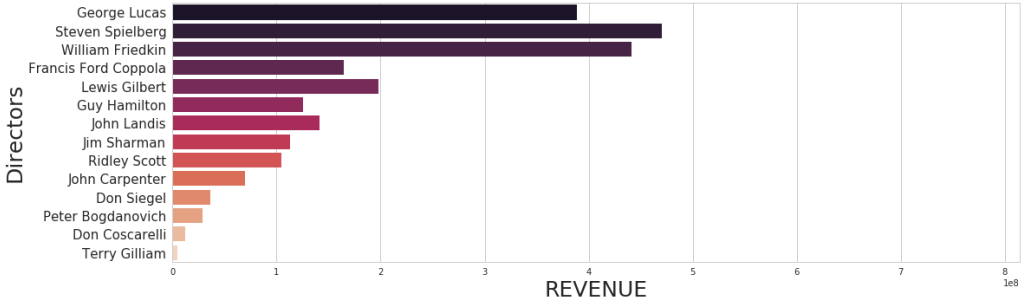
Top 20 Directors of every decade starting from 1960 to 1969 based on revenue



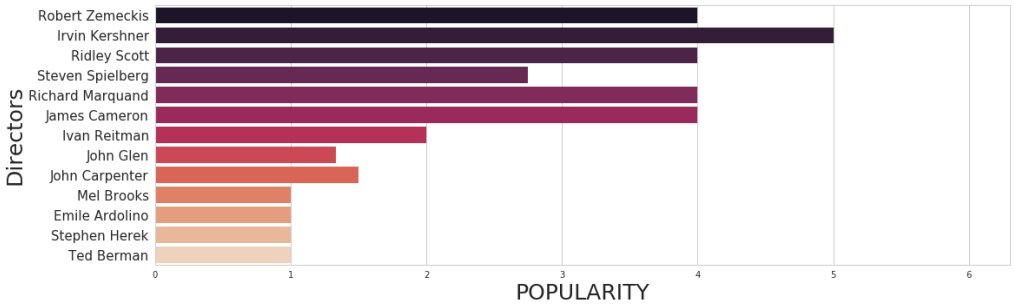
Top 20 Directors of every decade starting from 1970 to 1979 based on popularity



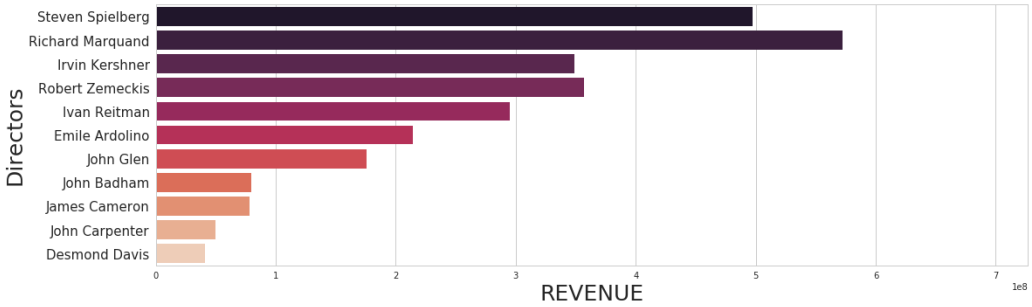
Top 20 Directors of every decade starting from 1970 to 1979 based on revenue



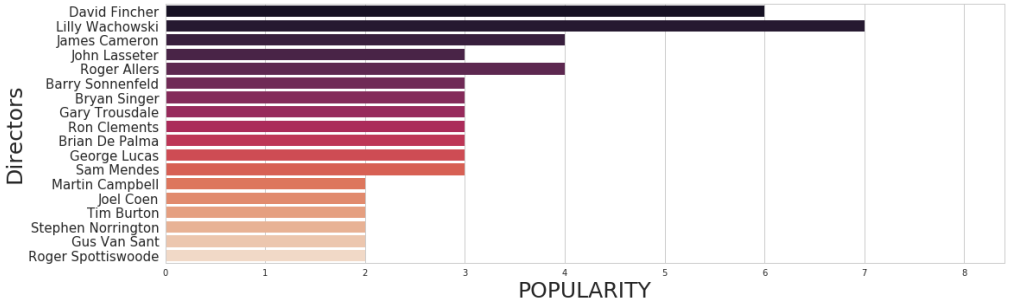
Top 20 Directors of every decade starting from 1980 to 1989 based on popularity



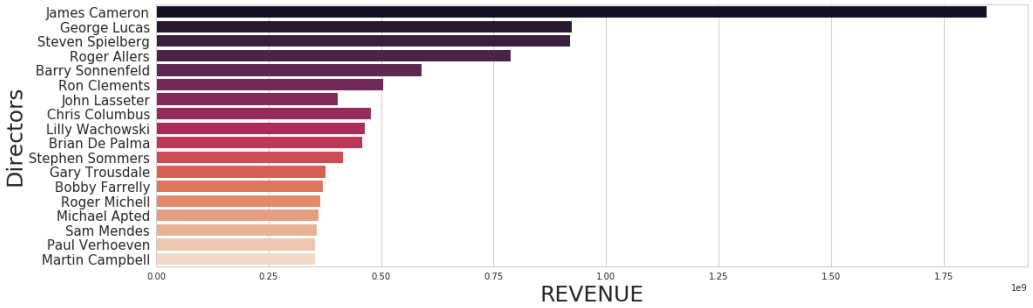
Top 20 Directors of every decade starting from 1980 to 1989 based on revenue



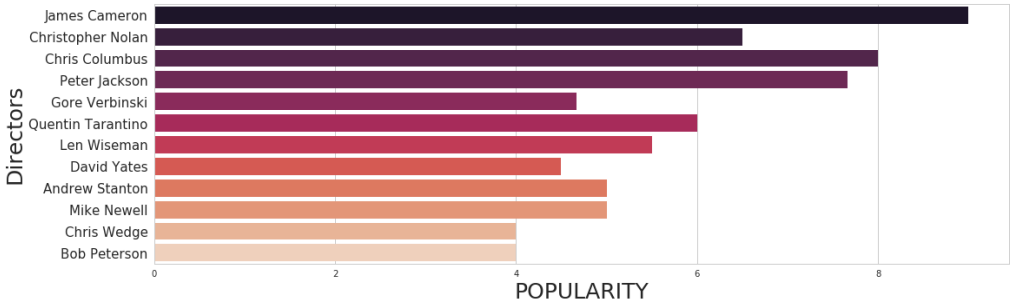
Top 20 Directors of every decade starting from 1990 to 1999 based on popularity



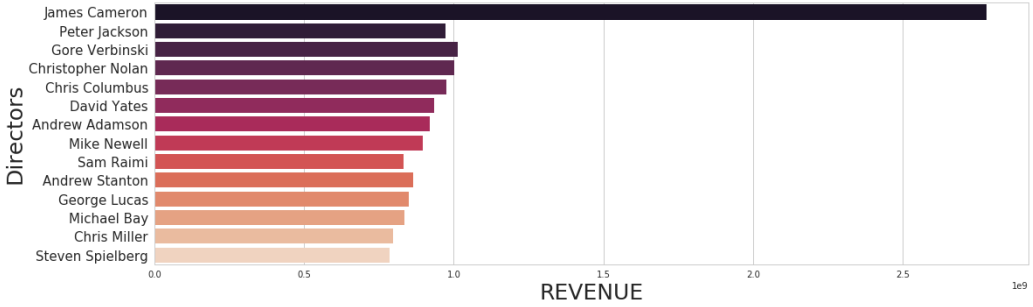
Top 20 Directors of every decade starting from 1990 to 1999 based on revenue

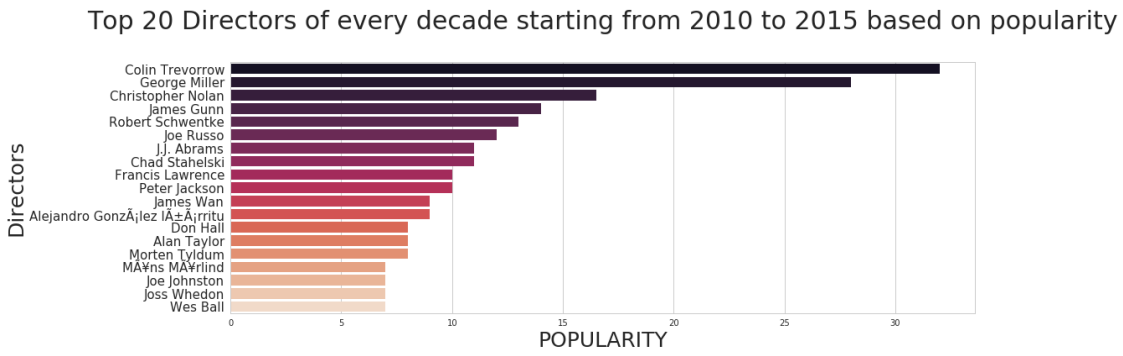


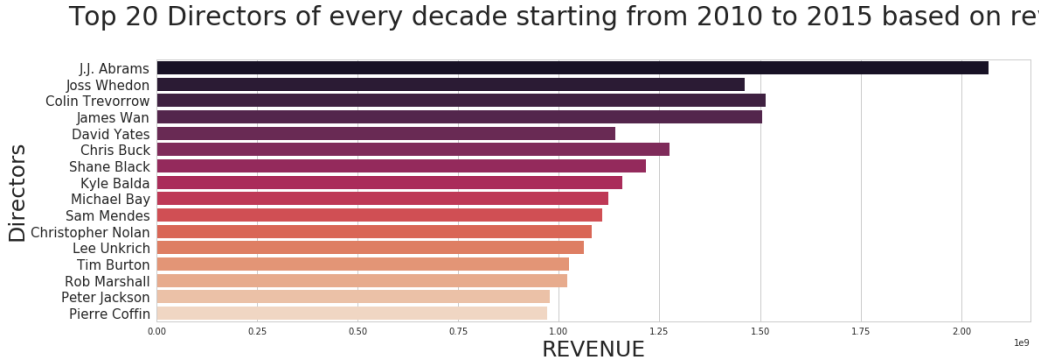
Top 20 Directors of every decade starting from 2000 to 2009 based on popularity



Top 20 Directors of every decade starting from 2000 to 2009 based on revenue



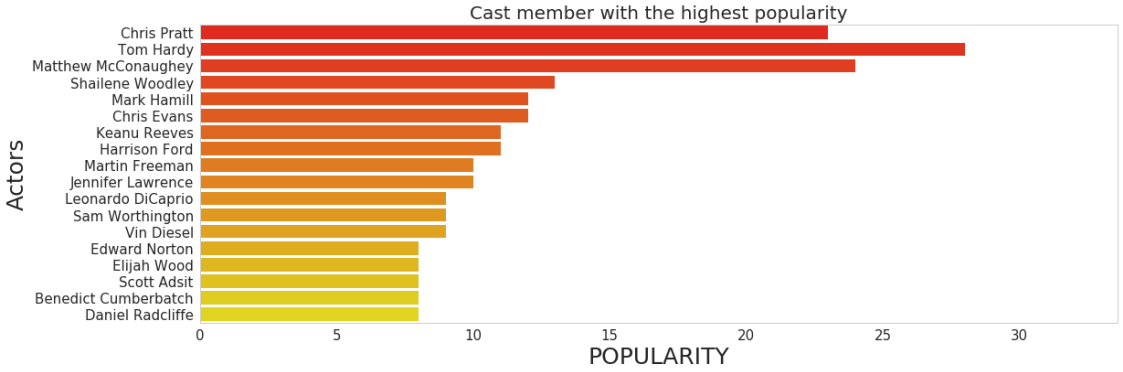


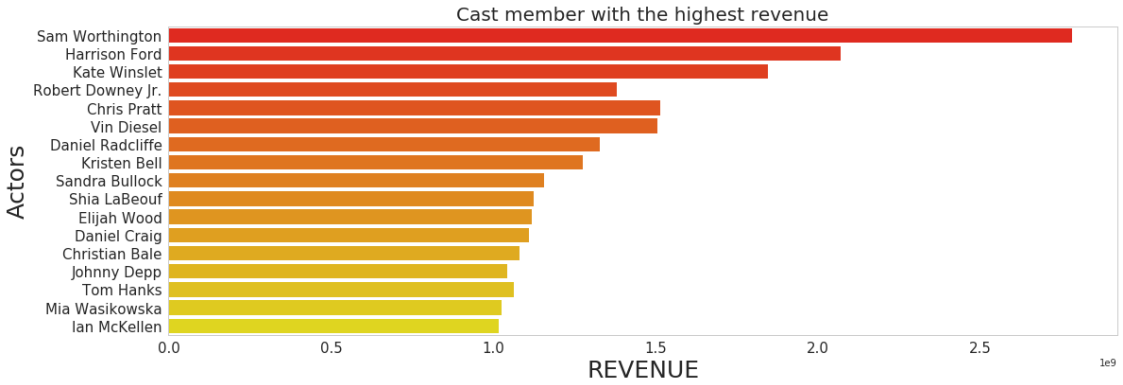


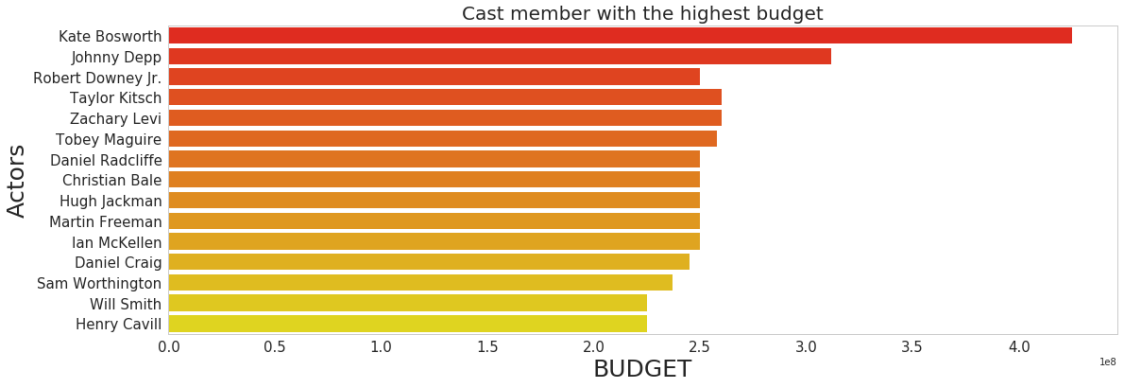
In [88]: *#AND LET'S NOT FORGATE ABOUT THE CASTS*

In [89]:
```
def cast(val):

    test5 = df.sort_values(by=val, ascending=False)


    sns.barplot(test5[val].iloc[:20], test5.cast_1.iloc[:20], palette='autumn', errwidt

    params = {'figure.figsize': (18, 6), 'axes.labelsize':15, 'axes.titlesize':20}

    plt.rcParams.update(params)

    plt.title("Cast member with the highest " + val)
    plt.xlabel(val.upper(), fontsize=25)
    plt.yticks(fontsize=15)
    plt.ylabel('Actors', fontsize=25)
    plt.xticks(fontsize=15)
    plt.grid(False)

    plt.show()
```

```
In [90]: #commnds
         cast('popularity')
         cast('revenue')
         cast('budget')
```



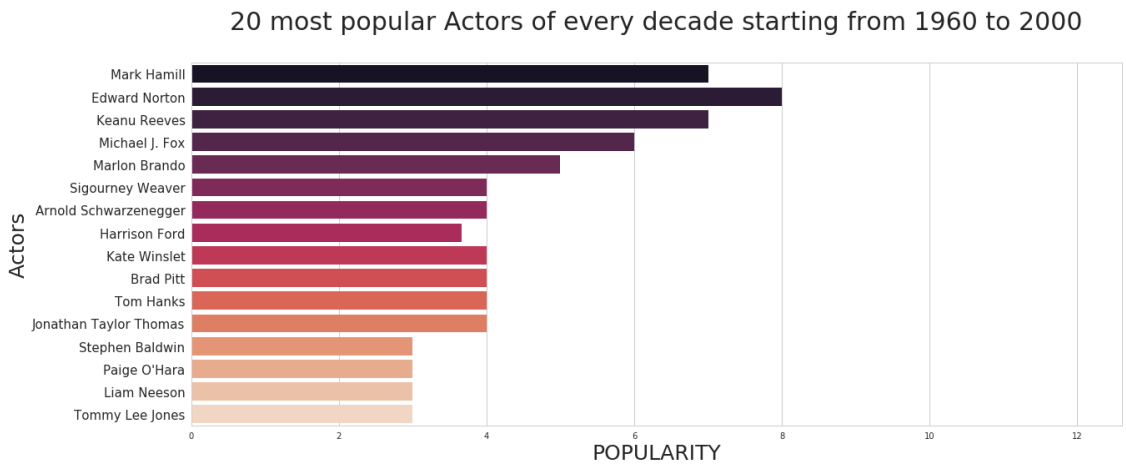




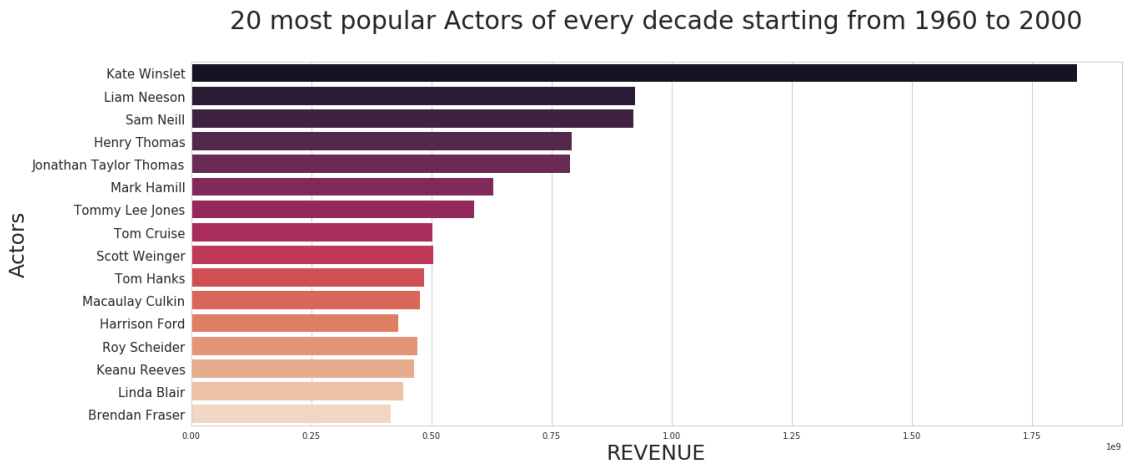
In [91]:
```
#MOST POPULAR ACTORS OF EVERY DECADE FROM 1960 TO 2015
```

In [94]:
```
def cast_year(val1, val2, val3):

    a = df[df.release_year.between(val2, val3, inclusive=True)].sort_values(by=val1, as
    b = df[df.release_year.between(val2, val3, inclusive=True)].sort_values(by=val1, as

    sns.barplot(a, b, palette='rocket', errwidth=False)

    params = {'figure.figsize': (20, 8), 'axes.labelsize':15, 'axes.titlesize':20}

    plt.rcParams.update(params)

    plt.title('20 most popular Actors of every decade starting from '+str(val2)+' to '+
    plt.xlabel(val1.upper(), fontsize=25)
    plt.ylabel('Actors', fontsize=25)
    plt.yticks(fontsize=15)
```

In [100]:
```
#commnds for popularity 1960 to 2000

cast_year('popularity', 1960, 2000)
```

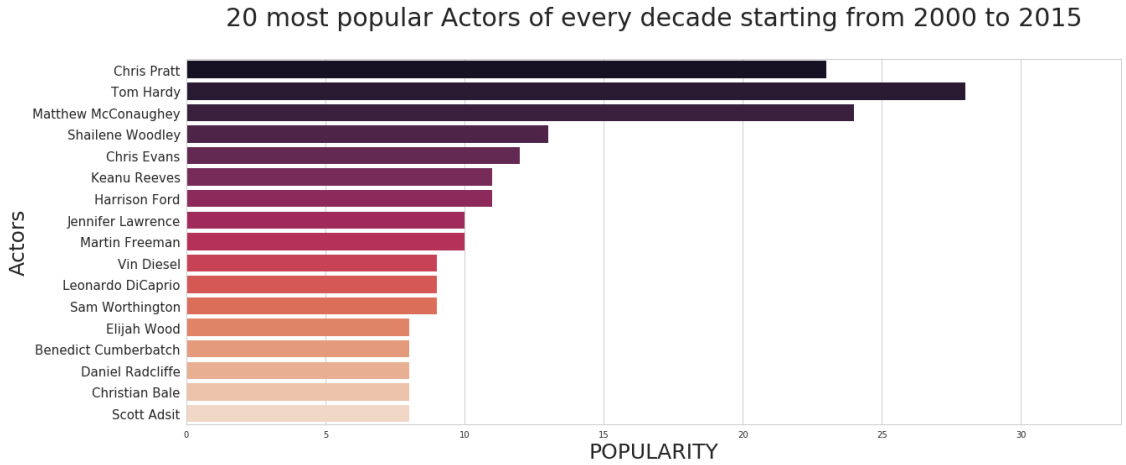


In [99]:
```
#commnds for revenue 1960 to 2000

cast_year('revenue', 1960, 2000)
```



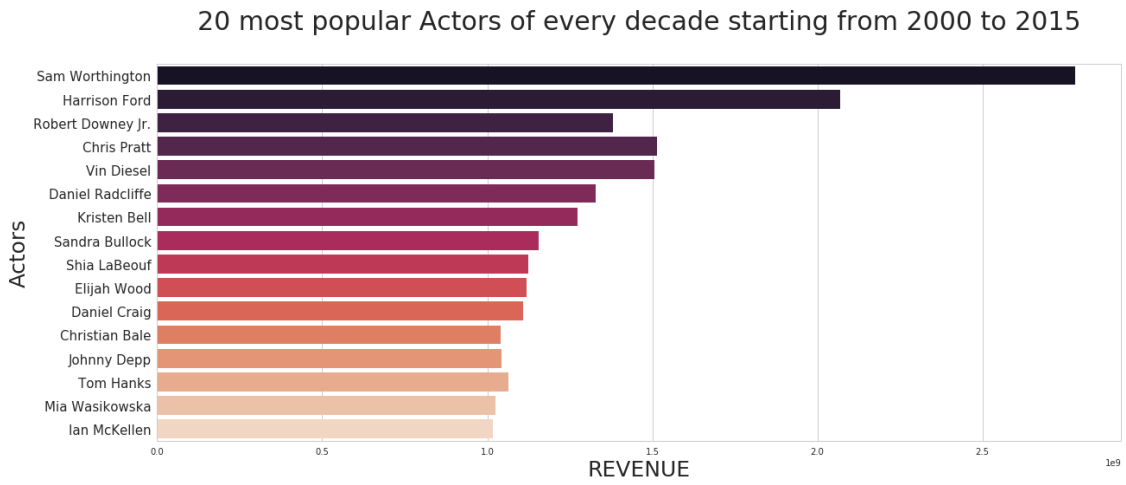
In [101]: *#commnds for popularity 2000 to 2015*

```
cast_year('popularity', 2000, 2015)
```



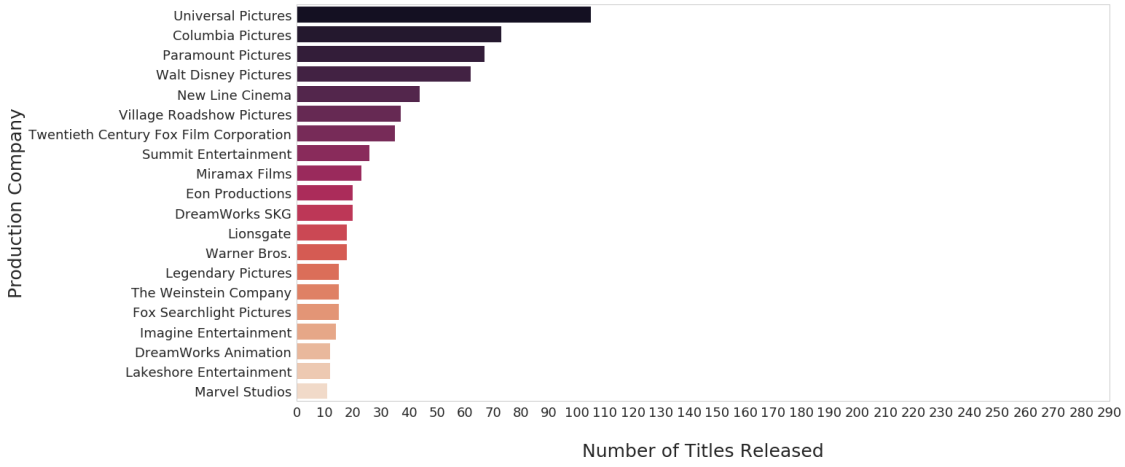
In [102]: *#commnds for revenue 2000 to 2015*

```
cast_year('revenue', 2000, 2015)
```



In [103]:
```
#TOP PRODUCTION COMPANIES & NUMBER OF TITLE RELEASED BY MOVIES
```

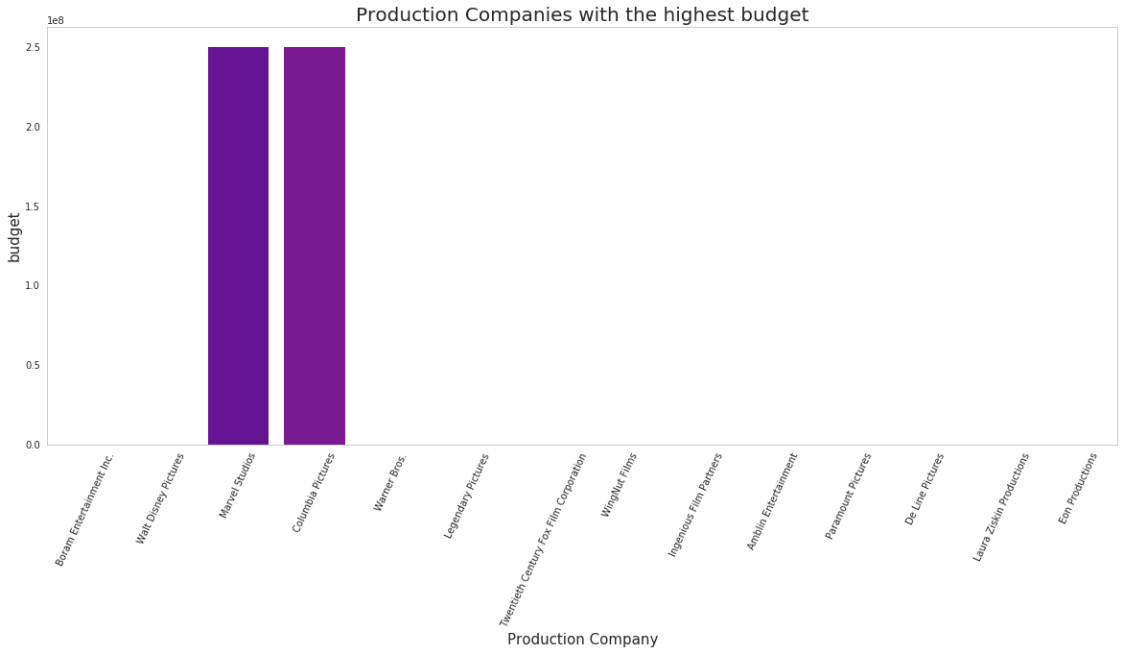
In [104]:
```
prod = pd.DataFrame(df.groupby('production_companies_1', as_index=False)['original_tit

prod.columns = ['Movies', 'Titles_Released']

prod = prod.sort_values(by='Titles_Released', ascending=False)[:20]

prod.reset_index(drop=True, inplace=True)

sns.set(rc={'figure.figsize':(20, 10)})
sns.set_style("whitegrid")

sns.barplot(prod.Titles_Released, prod.Movies, palette='rocket')

plt.xticks(np.arange(0, 300, 10))
plt.xlabel('\nNumber of Titles Released',fontsize=25)
plt.ylabel('Production Company',fontsize=25)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.grid(False)


plt.show()
```

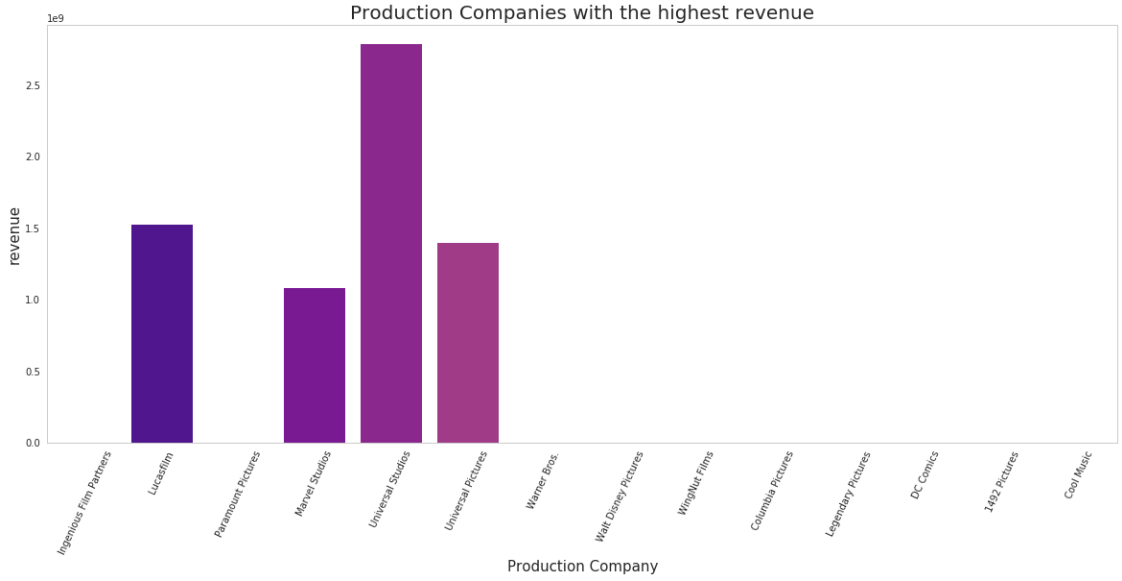


In [106]:
```
#Top Production Companies across different Parameters
```

In [108]:
```
def production(val):

    test5 = df.sort_values(by=val, ascending=False)


    lst = []
    for i in test5[val].iloc[:30]:

        if i > 1000:
            lst.append(i)
        else:
            True

    lst = sorted(lst, reverse=True)

    sns.barplot(test5.production_companies_1.iloc[:len(lst)], lst, palette='plasma', e

    params = {'figure.figsize': (20, 8), 'axes.labelsize':15, 'axes.titlesize':20}

    plt.rcParams.update(params)

    plt.title("Production Companies with the highest " + val)
    plt.ylabel(val)
    plt.xlabel('Production Company')

    plt.xticks(rotation=65)


    plt.grid(False)
    plt.show()
```

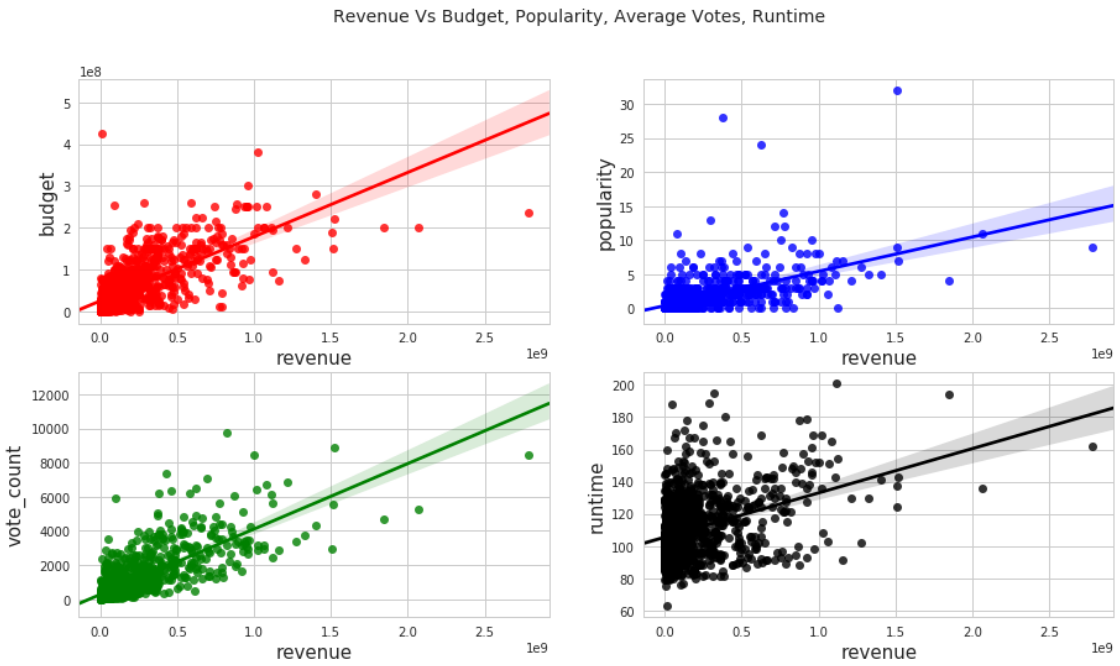
```
In [110]: production('budget')
          production('revenue')
```






```
In [111]: #How is Revenue affected by the different parameters?

In [115]: fig, axes = plt.subplots(2,2,figsize = (15, 8))
```

```
fig.suptitle("Revenue Vs Budget, Popularity, Average Votes, Runtime", fontsize=14)

sns.regplot(x=df['revenue'], y=df['budget'],color='r', ax=axes[0][0])
sns.regplot(x=df['revenue'], y=df['popularity'],color='b', ax=axes[0][1])
sns.regplot(x=df['revenue'], y=df['vote_count'],color='g', ax=axes[1][0])
sns.regplot(x=df['revenue'], y=df['runtime'],color='k', ax=axes[1][1])

sns.set_style("whitegrid")
```



In [ ]: