Introduction:

Welcome to the analysis of the dataset, we will be analyzing the movie dataset and will be following steps:
Step 1: Questions- In the first step of the report, we will gather some **Questions** from the given dataset.

> 1. Which genres are most popular from year to year?
> 2. Which genres are most popular all over the Decades?
> 3. Does the movie length has increased or decreased from year to year?
> 4. Does the movies associated with higher revenues are popular?
> 5. Does the movie associated with higher revenues make more profit?

Step 2: Wrangle- In this step we will gather, access and clean our data. We will do the modifications such as filling the missing values, dropping the duplicate values, updating the data types of the data, to ensure the clean analysis of the provided dataset.
Step 3: Explore- In this step we will be exploring the data which involves steps like finding patterns of the data, visualizing relationships in the data and building intuition about what we are working with.
Step 4: Drawing conclusions- We will summaries our findings in this step.
Let's begin!

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling:

In this process we will gather, access and clean our data. We will do the modifications such as filling the missing values, dropping the duplicate values, updating the data types of the data, to ensure the clean analysis of the provided dataset.

Lets gather the Data first:

In [5]:
```
#Gathering the Data using read_csv
df= pd.read_csv('tmdb-movies.csv')
df.head()
```

Out[5]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	ta
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The pa c
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	Wl L
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	Ch De
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	E gener h s
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Venge Hits H

5 rows × 21 columns

In [6]:
```
#Accessing the number of rows and columns in the dataset
df.shape
```

Out[6]: (10866, 21)

In [7]:
```
#Accessing the stats of each columns
df.describe()
```

Out[7]:

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	revenue_a
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086600e+
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658	1.755104e+07	5.136436e+
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941	3.430616e+07	1.446325e+
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.000000e+
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.000000e+
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.000000e+
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000	2.085325e+07	3.369710e+
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.827124e+

Important facts:

- Popularity ranges from (0 to 33) having the average of 7.
- Budget (usd) ranges from approx. 0 - 425 million with the average 17.6 million.
- Revenue (usd) ranges from approx. 0 - 2.8 billion with the average 51.4 million.
- The average runtime of movies is approz 102 mins.
- Release year ranges from 1960 to 2015 and most of the movies released after 1995.

Next we will clean the Data:

In [8]:
```
#checking the info of dataset, checking the missing data, investigating the data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

Data Cleaning:

Here we will modify our dataset. First we will remove the duplicate values and extra data after checking the dataset, then add and replace information to ensure our dataset is clean for analysis.

- First we will drop the columns that we dont need in the analysis.
- As the most concerned columns are the budget and revenue rather than those which are inflated.

In [9]:
```
#checking the missing values in dataset
df.isnull().sum()
```

Out[9]:
```
id                         0
imdb_id                   10
popularity                 0
budget                     0
revenue                    0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

In [12]:
```
#droping the columns which we dont need further in our analysis
df.drop(['imdb_id','cast','homepage','tagline','keywords','overview','production_companies','release_date'], axis=1, inpl
df.head()
```

Out[12]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year	
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015	1.:
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015	1.:
2	262500	13.112507	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	2480	6.3	2015	1.(
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015	1.8
4	168259	9.335014	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	2947	7.3	2015	1.7

Now we will check if the dataset has some missing value or not

In [13]:
```
# Checking  for the duplicate data
sum(df.duplicated())
```

Out[13]: 1

There is only 1 duplicate, so we will drop that particlar row and perform two checks to ensure the duplicates were removed or not.

In [16]:
```
#droping of duplicate values by using drop
df.drop_duplicates(inplace=True)
#rechecking the total sum of duplicates in the dataset
print(sum(df.duplicated()))
print(df.shape)
```

```
0
(10865, 13)
```

There are no longer any duplicates and the dataset now has one less row. Next, we will assess if any rows have missing values.

In [17]:
```
#rechecking if there is any missing values
df.isnull().sum()
```

Out[17]:
```
id                 0
popularity         0
budget             0
revenue            0
original_title     0
director          44
runtime            0
genres            23
vote_count         0
vote_average       0
release_year       0
budget_adj         0
revenue_adj        0
dtype: int64
```

In [18]:
```
df[df.isnull().any(axis=1)].sort_values(['runtime'], ascending=True)
```

Out[18]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year
2370	127717	0.081892	0	0	Freshman Father	Michael Scott	0	NaN	12	5.8	2010
1241	296370	0.135376	0	0	Dance-Off	NaN	0	Romance|Music|Comedy	18	5.7	2014
2315	48373	0.171615	0	0	Listen to Your Heart	NaN	0	Drama|Music|Romance	29	7.3	2010
4890	126909	0.083202	0	0	Cousin Ben Troop Screening	Wes Anderson	2	NaN	14	7.0	2012
5934	200204	0.067433	0	0	Prada: Candy	Wes Anderson|Roman Coppola	3	NaN	27	6.9	2013
...	...	...	...	...	...	...	...	...	...	...	...
2401	45644	0.067753	0	0	Opeth: In Live Concert At The Royal Albert Hall	NaN	163	Music	10	8.6	2010
3224	20313	0.224721	0	0	John Mayer: Where the Light Is Live in Los Ang...	NaN	164	Music	16	8.5	2008
4547	123024	0.520520	0	0	London 2012 Olympic Opening Ceremony: Isles of...	Danny Boyle	220	NaN	12	8.3	2012
4939	168219	0.003183	0	0	The Men Who Built America	NaN	360	Documentary|History	11	5.3	2012
6181	18729	0.000065	0	0	North and South, Book I	NaN	561	Drama|History|Western	17	6.0	1985

65 rows × 13 columns

It seems that these movies are a combination of shorts and features films. Now we can also remove the rows that have no director, and/or genre. After doing this we will check the dataset again to ensure there is no missing information (should return False on checking) and the new dataset info.

In [19]:
```
df.dropna(inplace=True)
print(df.isnull().sum().any())
print(df.info())
```

```
False
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10800 entries, 0 to 10865
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              10800 non-null  int64  
 1   popularity      10800 non-null  float64
 2   budget          10800 non-null  int64  
 3   revenue         10800 non-null  int64  
 4   original_title  10800 non-null  object 
 5   director        10800 non-null  object 
 6   runtime         10800 non-null  int64  
 7   genres          10800 non-null  object 
 8   vote_count      10800 non-null  int64  
 9   vote_average    10800 non-null  float64
 10  release_year    10800 non-null  int64  
 11  budget_adj      10800 non-null  float64
 12  revenue_adj     10800 non-null  float64
dtypes: float64(4), int64(6), object(3)
memory usage: 1.2+ MB
None
```

There are now 10,800 rows and 13 columns. All columns now have the full amount of rows.
If we wanted to replace nulls with mean values: df.fillna(df.mean(), inplace=True)
Additional cleaning checks:

- Do columns need renaming? No
- Do datatypes need converting? No, could convert budget and revenue from float to int but it will not make any difference.
- Let's review popularity, vote count, and vote average. We can also remove any outliers later.

In [21]:
```
df[['original_title','popularity', 'vote_count', 'vote_average']].sort_values('popularity', ascending=False).head(15)
```

Out[21]:

	original_title	popularity	vote_count	vote_average
0	Jurassic World	32.985763	5562	6.5
1	Mad Max: Fury Road	28.419936	6185	7.1
629	Interstellar	24.949134	6498	8.0
630	Guardians of the Galaxy	14.311205	5612	7.9
2	Insurgent	13.112507	2480	6.3
631	Captain America: The Winter Soldier	12.971027	3848	7.6
1329	Star Wars	12.037933	4428	7.9
632	John Wick	11.422751	2712	7.0
3	Star Wars: The Force Awakens	11.173104	5292	7.5
633	The Hunger Games: Mockingjay - Part 1	10.739009	3590	6.6
634	The Hobbit: The Battle of the Five Armies	10.174599	3110	7.1
1386	Avatar	9.432768	8458	7.1
1919	Inception	9.363643	9767	7.9
4	Furious 7	9.335014	2947	7.3
5	The Revenant	9.110700	3929	7.2

In [22]:
```
df[['original_title','popularity', 'vote_count', 'vote_average']].sort_values('popularity', ascending=False).tail()
```

Out[22]:

	original_title	popularity	vote_count	vote_average
7268	Born into Brothels	0.001117	23	6.4
6961	Khosla Ka Ghosla!	0.001115	10	6.8
6551	Mon petit doigt m'a dit...	0.000973	13	5.7
6080	G.B.F.	0.000620	82	6.1
9977	The Hospital	0.000188	10	6.4

- We will keep as is it as it doesn't seem like there are outliers for popularity, just a wide range of values.
- For vote average, some have upwards of 10,000 votes while others have only 10.
- We can also review the vote_count column while exploring the data.

We can add a profit column so we can create a profitability ratio.
Profit = revenue (also known as income) - budget (alsoknown as cost or expense)

In [23]:
```
df['profit'] = df['revenue'] - df['budget']
df.head()
```

Out[23]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year	
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015	1.3
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015	1.3
2	262500	13.112507	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	2480	6.3	2015	1.0
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015	1.8
4	168259	9.335014	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	2947	7.3	2015	1.7

In [24]:
```
# we will make sure there is no negative numbers for profit column
df.loc[df['profit'] < 0, 'profit'] = 0
```

Now we have profit column than we can also create a column for profitability ratio.
Profitability ratio = (profit/revenue) x 100 = percentage

we will adjust for non-zero division by adding .0001 to the denominator, revenue. As we do not want ant non-decimal values ranging from 1-100. So we will convert this column from float to integer.

In [25]:
```
df['profitability_ratio'] = (df['profit'] / (df['revenue'] + .0001)) * 100
df['profitability_ratio'] = df['profitability_ratio'].astype(int)
df.sort_values(['profitability_ratio'], ascending=False).tail()
```

Out[25]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average
5613	210024	0.646924	0	0	An Adventure in Space and Time	Terry McDonough	90	Drama	36	7.4
5612	85889	0.660633	5000000	0	Filth	Jon S. Baird	97	Crime|Drama|Comedy	370	6.6
5611	313106	0.661187	0	0	Doctor Who: The Day of the Doctor	Nick Hurran	77	Science Fiction|Adventure	190	8.0
1766	72962	0.781772	0	0	George & A.J.	Josh Cooley	4	Animation|Adventure|Comedy|Family|Fantasy	15	5.7
10865	22293	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren	74	Horror	15	1.5

In [26]:
```
print(df['profitability_ratio'].nunique())
```

```
100
```

In [27]:
```
df.loc[df['profitability_ratio'] < 0, 'profitability_ratio'] = 0
print(df['profitability_ratio'].nunique())
```

```
100
```

Next we will create a new column, revenue_rating, to join the revenue column into groups: low (under a million), mediun (millions), and high (billions).

In [28]:
```
bin_edges = [0, 1e+06, 1e+09, 2.827124e+09]
bin_names = ['under_million', 'millions', 'billions']
df['revenue_rating'] = pd.cut(df['revenue'], bin_edges, labels=bin_names)
df.head()
```

Out[28]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year	
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015	1.3
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015	1.3
2	262500	13.112507	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	2480	6.3	2015	1.0
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015	1.8
4	168259	9.335014	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	2947	7.3	2015	1.7

In [29]:
```
df['revenue_rating'].value_counts()
```

Out[29]:
```
millions         4300
under_million     526
billions           22
Name: revenue_rating, dtype: int64
```

In [31]:
```
df.isnull().sum()
```

Out[31]:
```
id                      0
popularity              0
budget                  0
revenue                 0
original_title          0
director                0
runtime                 0
genres                  0
vote_count              0
vote_average            0
release_year            0
budget_adj              0
revenue_adj             0
profit                  0
profitability_ratio     0
revenue_rating       5952
dtype: int64
```

Now our task is to clean up the revenue rating, so now let us make all the rows with null values 0 as those rows have no revenue/budget

In [32]:
```
df.revenue_rating.fillna('under_million', inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10800 entries, 0 to 10865
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype   
---  ------               --------------  -----   
 0   id                   10800 non-null  int64   
 1   popularity           10800 non-null  float64 
 2   budget               10800 non-null  int64   
 3   revenue              10800 non-null  int64   
 4   original_title       10800 non-null  object  
 5   director             10800 non-null  object  
 6   runtime              10800 non-null  int64   
 7   genres               10800 non-null  object  
 8   vote_count           10800 non-null  int64   
 9   vote_average         10800 non-null  float64 
 10  release_year         10800 non-null  int64   
 11  budget_adj           10800 non-null  float64 
 12  revenue_adj          10800 non-null  float64 
 13  profit               10800 non-null  int64   
 14  profitability_ratio  10800 non-null  int32   
 15  revenue_rating       10800 non-null  category
dtypes: category(1), float64(4), int32(1), int64(7), object(3)
memory usage: 1.3+ MB
```

Now we will be creating a new column which consist all decades as the release years range from 1960 to 2015.

In [33]:
```
bin_edges = [1959, 1970, 1980, 1990, 2000, 2010, 2015]
bin_names = ['sixties', 'seventies', 'eighties', 'nineties', 'two_thousands', 'two_thousand_tens']
df['decades'] = pd.cut(df['release_year'], bin_edges, labels=bin_names)
df.head()
```

Out[33]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year	
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015	1.
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015	1.
2	262500	13.112507	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	2480	6.3	2015	1.
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J.J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015	1.
4	168259	9.335014	190000000	1506249360	Furious 7	James Wan	137	Action|Crime|Thriller	2947	7.3	2015	1.

In [34]:
```
#Checking that both the newly created columns look good.
df[['release_year', 'decades']].head()
```

Out[34]:

	release_year	decades
0	2015	two_thousand_tens
1	2015	two_thousand_tens
2	2015	two_thousand_tens
3	2015	two_thousand_tens
4	2015	two_thousand_tens

Let us now work with the column that have more than one values per cell, genres. We will be creating separate dataframe for genres in case we want the original dataframe unharmed.

In [35]:
```
df['genres'].str.contains('|')
df['genres'].nunique()
```

Out[35]:
```
2031
```

now we will split up the genres column cells so we can tally each genre individually. In next step we will remove the 'genres' column (with multiple values) and replace it with a 'genre' column (with single values). Then we will make sure that there is a new row for each genre (stacked), so there will be multiple rows with the same original_title.

In [36]:
```
df_split_genre = df.copy()
split_genre = df_split_genre['genres'].str.split('|').apply(pd.Series, 1).stack().reset_index(level=1, drop=True)
split_genre.name = 'genre_split'
df_split_genre = df_split_genre.drop(['genres'], axis=1).join(split_genre)
```

In [37]:
```
#checking that the new genre column contains only single values
df_split_genre['genre_split'].unique()
```

Out[37]:
```
array(['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy',
       'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',
       'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror',
       'Documentary', 'TV Movie', 'Foreign'], dtype=object)
```

In [38]:
```
#checking the duplicates and viewing the info for our new dataset
print(df_split_genre.info())
print(df_split_genre.shape)
print(sum(df_split_genre.duplicated()))
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26864 entries, 0 to 10865
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   26864 non-null  int64
 1   popularity           26864 non-null  float64
 2   budget               26864 non-null  int64
 3   revenue              26864 non-null  int64
 4   original_title       26864 non-null  object
 5   director             26864 non-null  object
 6   runtime              26864 non-null  int64
 7   vote_count           26864 non-null  int64
 8   vote_average         26864 non-null  float64
 9   release_year         26864 non-null  int64
 10  budget_adj           26864 non-null  float64
 11  revenue_adj          26864 non-null  float64
 12  profit               26864 non-null  int64
 13  profitability_ratio  26864 non-null  int32
 14  revenue_rating       26864 non-null  category
 15  decades              26864 non-null  category
 16  genre_split          26864 non-null  object
dtypes: category(2), float64(4), int32(1), int64(7), object(3)
memory usage: 3.2+ MB
None
(26864, 17)
0
```

We now have 26,864 rows (from 10,000) and 13 columns (same), which makes sense, and no duplicate rows.

In [39]:
```
#checking for any null values
df_split_genre.isnull().sum()
```

Out[39]:
```
id                    0
popularity            0
budget                0
revenue               0
original_title        0
director              0
runtime               0
vote_count            0
vote_average          0
release_year          0
budget_adj            0
revenue_adj           0
profit                0
profitability_ratio   0
revenue_rating        0
decades               0
genre_split           0
dtype: int64
```

Looks good!!!

In [40]:
```
df_split_genre.head()
```

Out[40]:

	id	popularity	budget	revenue	original_title	director	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015	1.379999e+08	1.392446e+09
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015	1.379999e+08	1.392446e+09
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015	1.379999e+08	1.392446e+09
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015	1.379999e+08	1.392446e+09
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	6185	7.1	2015	1.379999e+08	3.481613e+08

In [41]:
```
df_split_genre.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26864 entries, 0 to 10865
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype   
---  ------               --------------  -----   
 0   id                   26864 non-null  int64   
 1   popularity           26864 non-null  float64 
 2   budget               26864 non-null  int64   
 3   revenue              26864 non-null  int64   
 4   original_title       26864 non-null  object  
 5   director             26864 non-null  object  
 6   runtime              26864 non-null  int64   
 7   vote_count           26864 non-null  int64   
 8   vote_average         26864 non-null  float64 
 9   release_year         26864 non-null  int64   
 10  budget_adj           26864 non-null  float64 
 11  revenue_adj          26864 non-null  float64 
 12  profit               26864 non-null  int64   
 13  profitability_ratio  26864 non-null  int32   
 14  revenue_rating       26864 non-null  category
 15  decades              26864 non-null  category
 16  genre_split          26864 non-null  object  
dtypes: category(2), float64(4), int32(1), int64(7), object(3)
memory usage: 3.2+ MB
```

In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10800 entries, 0 to 10865
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   10800 non-null  int64
 1   popularity           10800 non-null  float64
 2   budget               10800 non-null  int64
 3   revenue              10800 non-null  int64
 4   original_title       10800 non-null  object
 5   director             10800 non-null  object
 6   runtime              10800 non-null  int64
 7   genres               10800 non-null  object
 8   vote_count           10800 non-null  int64
 9   vote_average         10800 non-null  float64
 10  release_year         10800 non-null  int64
 11  budget_adj           10800 non-null  float64
 12  revenue_adj          10800 non-null  float64
 13  profit               10800 non-null  int64
 14  profitability_ratio  10800 non-null  int32
 15  revenue_rating       10800 non-null  category
 16  decades              10800 non-null  category
dtypes: category(2), float64(4), int32(1), int64(7), object(3)
memory usage: 1.3+ MB
```

We now have 2 clean dataframes: df, df_split_genre. We will save them below.

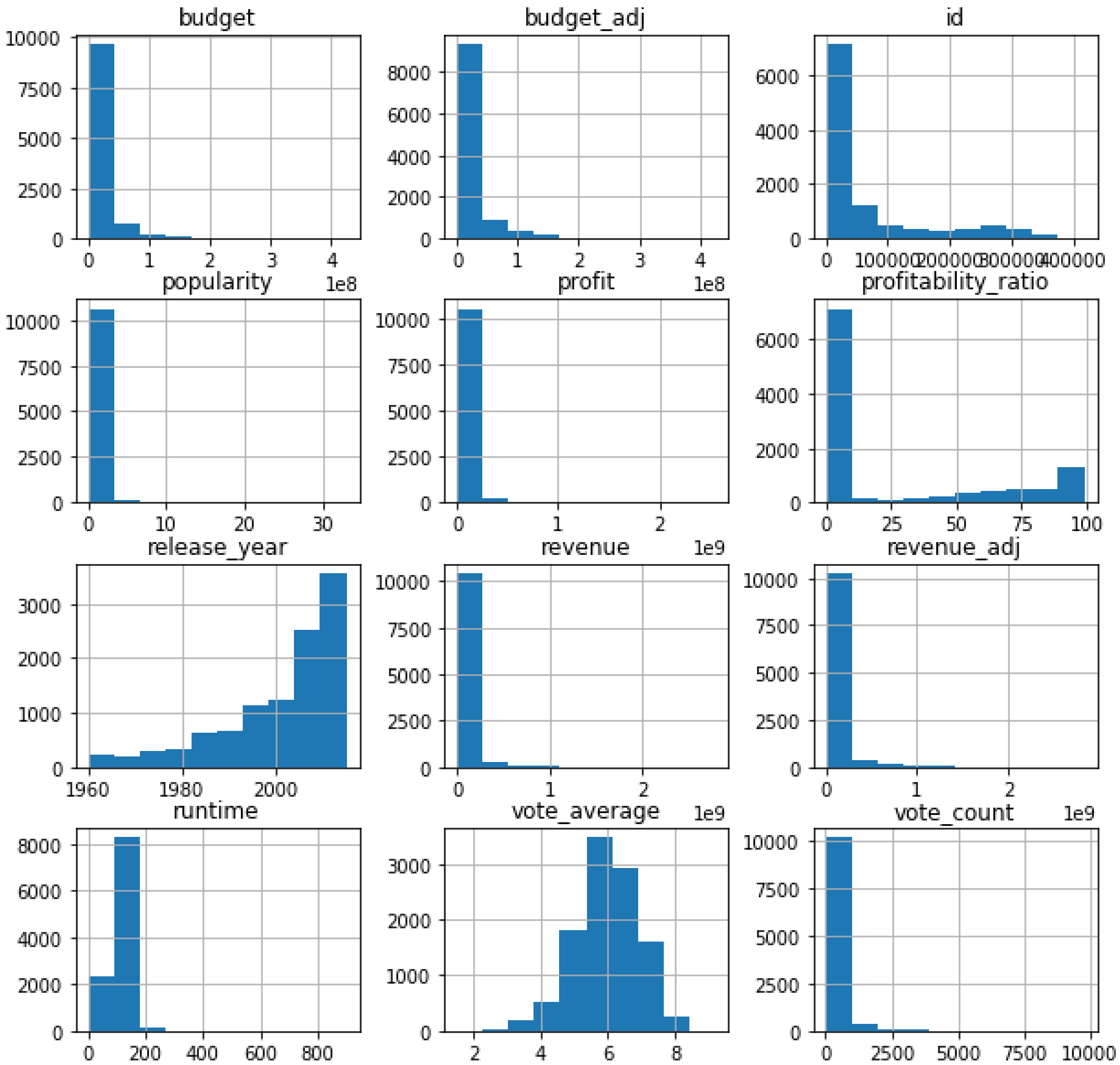
In [43]:
```
df.to_csv('tmbd-movies-clean.csv', index=False)
df_split_genre.to_csv('tmbd-movies-genre.csv', index=False)
```

EDA: Exploring Data Analysis:

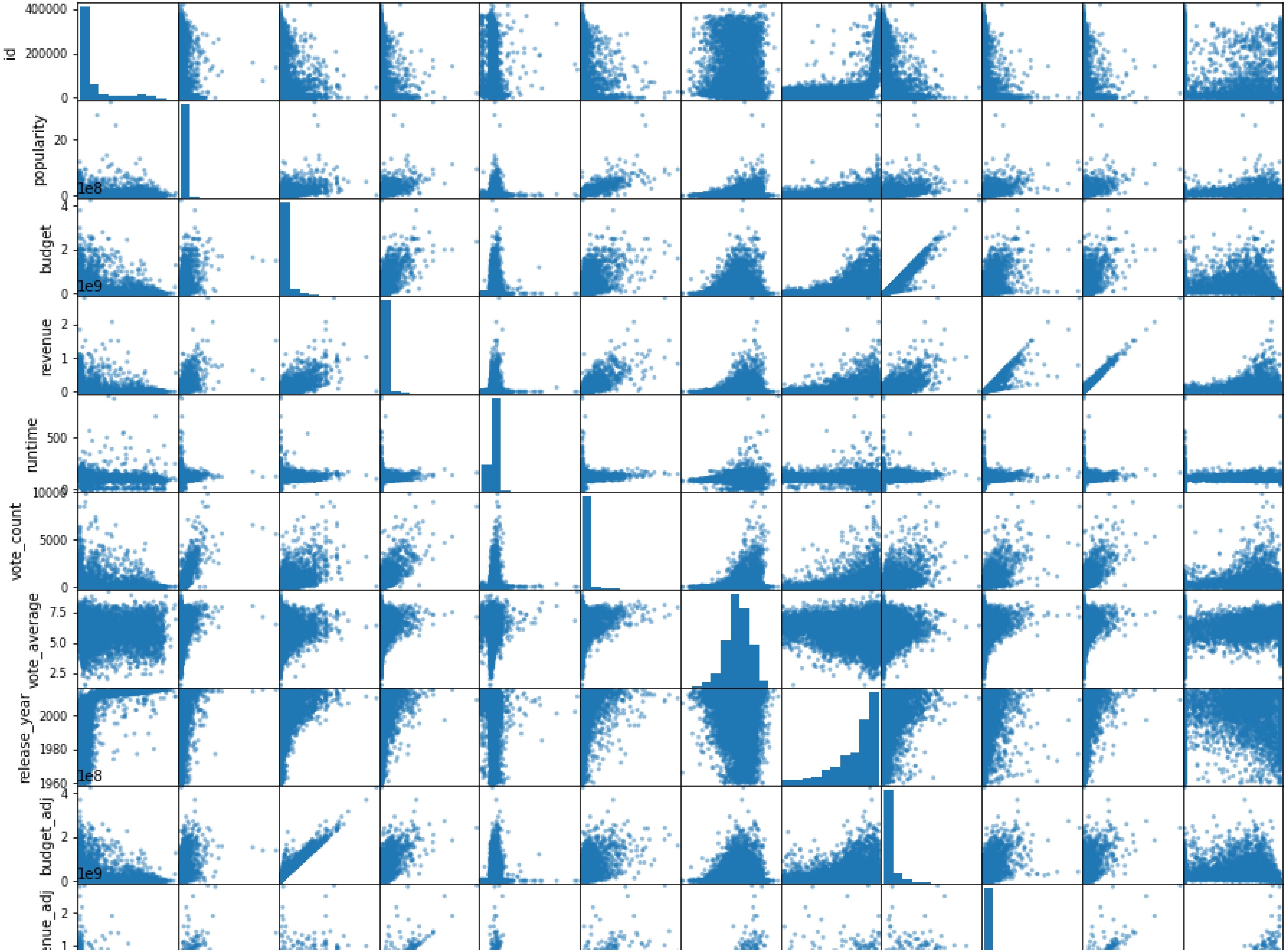
In this step we will be exploring the data which involves steps like finding patterns of the data, visualizing relationships in the data and building intuition about what we are working with. We will compute statistics and create visualizations with to address our questions. Let's move on to exploring and augmenting our data.

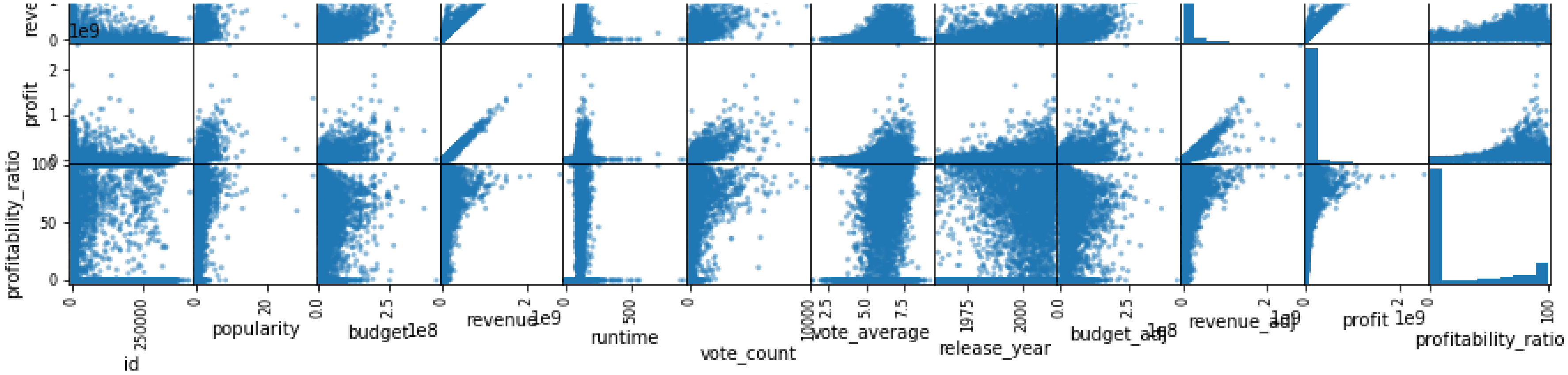
Let's first view all columns with numerical data with a histogram:

In [45]:
```
df.hist(figsize=(10,10));
```



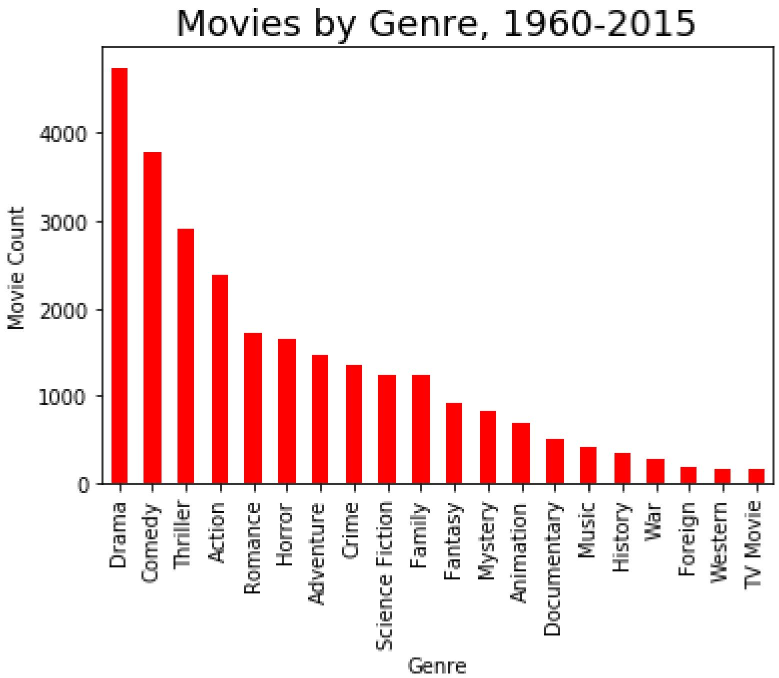
In [46]:
```
pd.plotting.scatter_matrix(df, figsize=(15, 15));
```





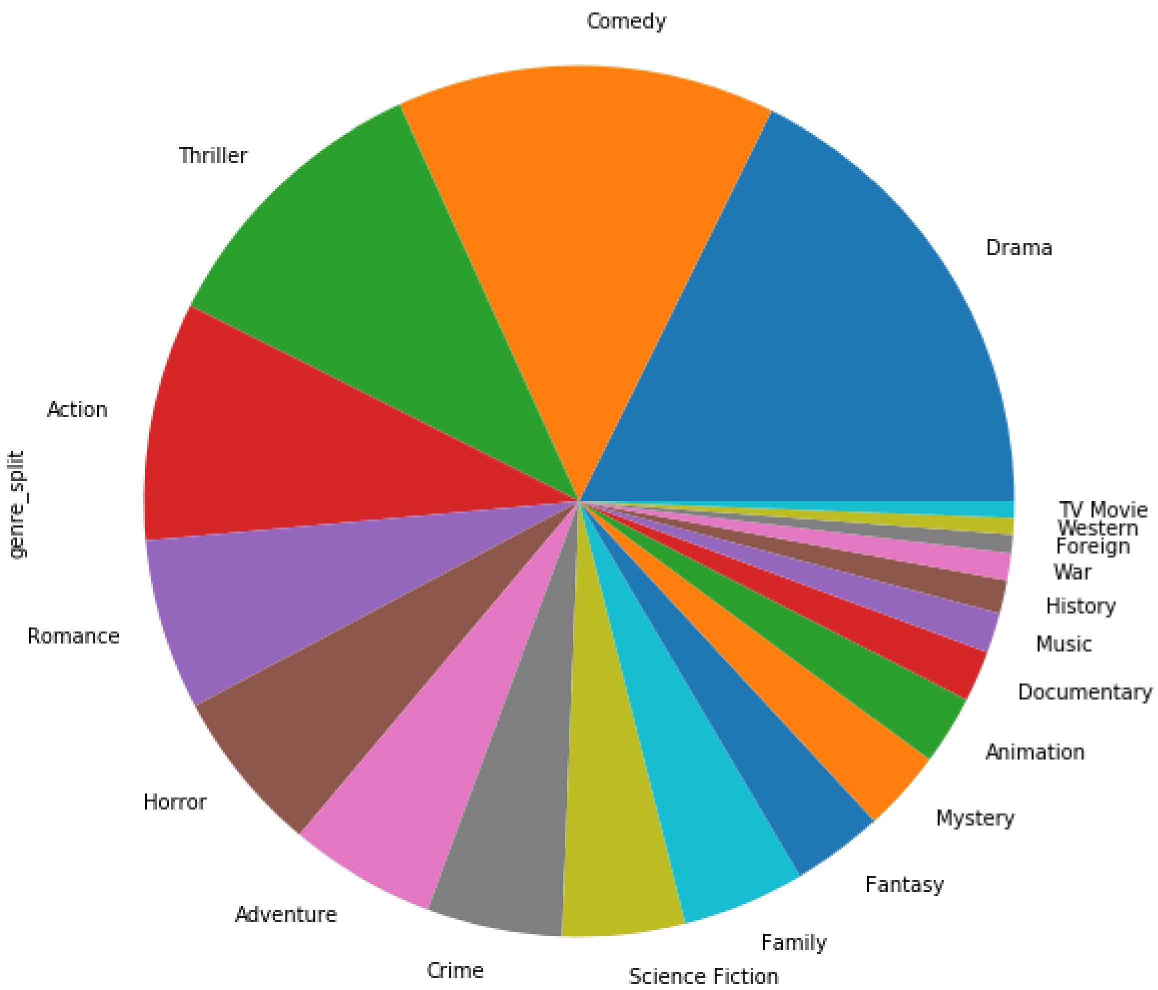
1.Which genres are most popular from year to year?

In [49]:
```
df_split_genre['genre_split'].value_counts().plot(kind='bar', color='r'
);
plt.title('Movies by Genre, 1960-2015', size=18)
plt.xlabel('Genre', size=10)
plt.ylabel('Movie Count', size=10);
```



In [50]:
```
#also a pie chart
df_split_genre['genre_split'].value_counts().plot(kind='pie', figsize=(10,10))
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x28b645790c8>



- Drama, Comedy, Thriller, and Action are the most popular genres in general.
- The pie chart is a better visual since we can assess that these top 4 genres make up about 50% of all movies.
- TV Movies, Westerns, and Foreigns are the least popular genres.

2. Which genres are most popular all over the Decades?

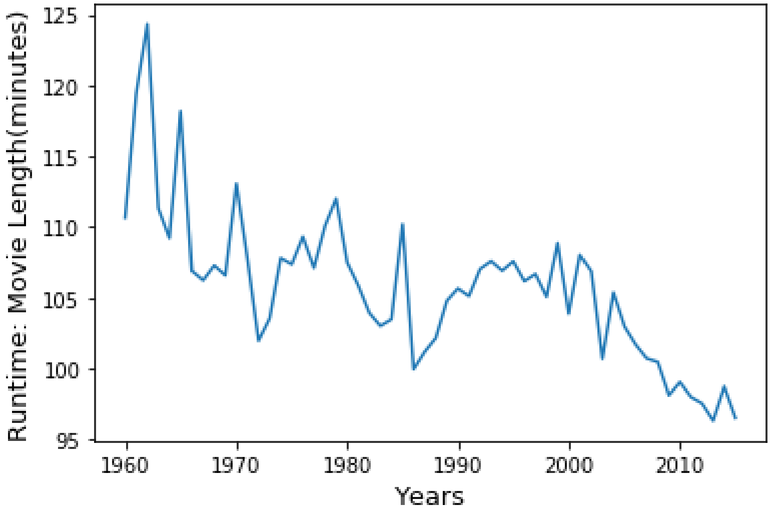
In [51]:
```
genres_decades = df_split_genre.groupby(['decades'])['genre_split'].value_counts()
genres_decades.groupby(level=0).nlargest(3).reset_index(level=0, drop=True)
```

Out[51]:
```
decades            genre_split
sixties            Drama           187
                   Comedy          125
                   Action           89
seventies          Drama           252
                   Thriller        168
                   Action          137
eighties           Comedy          451
                   Drama           450
                   Action          283
nineties           Drama           903
                   Comedy          785
                   Thriller        512
two_thousands      Drama          1719
                   Comedy         1422
                   Thriller       1043
two_thousand_tens  Drama          1243
                   Comedy          864
                   Thriller        831
Name: genre_split, dtype: int64
```

So as we can see that Drama is the most popular genre all over the decades except in the 80s which is taken over by Comedy.

3. Does the movie length has increased or decreased from year to year?

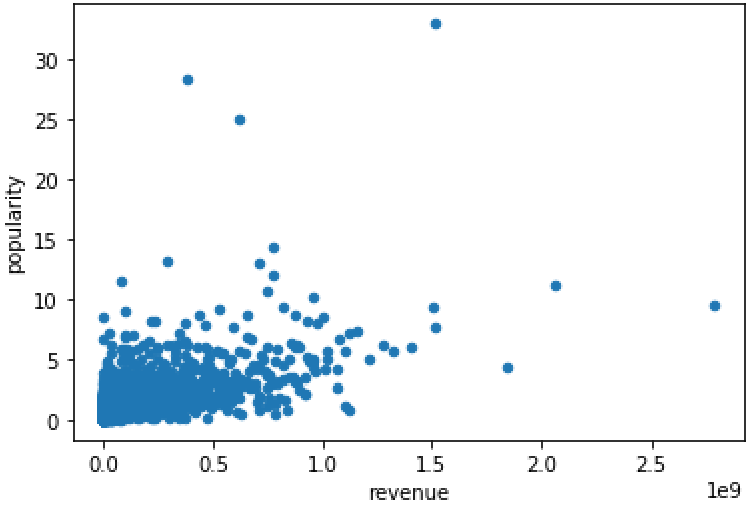
In [56]:
```
runtime = df.groupby('release_year')['runtime'].mean()
plt.plot(runtime)
plt.xlabel('Years', size=13)
plt.ylabel('Runtime: Movie Length(minutes)', size=13);
```



We can observe that there is a drastic change in the Runtime of the movies. It has decreased much from year to year.

4. Does the movies associated with higher revenues are popular?

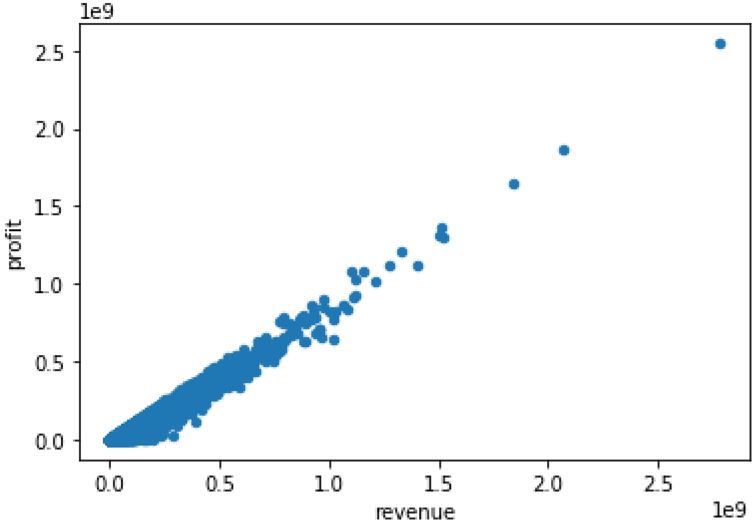
In [57]:
```
#General scatter plot of revenue vs popularity:
df.plot(x='revenue', y='popularity', kind='scatter');
```



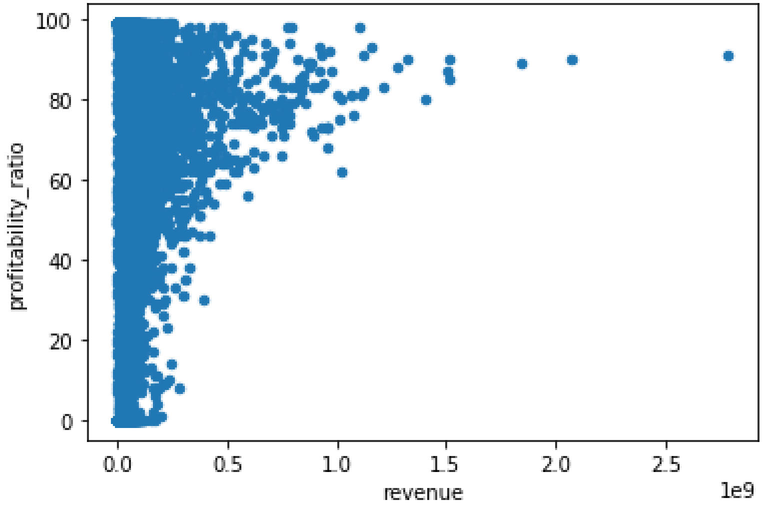
Revenue and popularity have positive interrelationship with each other, movies with higher revenues turn to be more popular.

5. Does the movie associated with higher revenues make more profit?

In [58]:
```
#General scatter plot of revenue vs profit:
df.plot(x='revenue', y='profit', kind='scatter');
```



In [59]:
```
#General scatter plot of revenue vs profitablity ratio:
df.plot(x='revenue', y='profitability_ratio', kind='scatter');
```



- Revenue and profit have a strong and positive interrelationship with each other.
- Revenue and profitability have a weak and positive interrelationship with each other.

Conclusions:

- Drama, Comedy, Thriller, and Action are the most popular genres in general.
- The pie chart is a better visual since we can assess that these top 4 genres make up about 50% of all movies.
- TV Movies, Westerns, and Foreigns are the least popular genres.
- We saw that Drama is the most popular genre all over the decades except in the 80s which is taken over by Comedy.
- We observed that there is a drastic change in the Runtime of the movies.

- Runtime has decreased much from year to year.
- Revenue and popularity have positive interrelationship with each other.
- Movies with higher revenues turn to be more popular.
- Revenue and profit have a strong and positive interrelationship with each other.
- Revenue and profitability have a weak and positive interrelationship with each other.

Limitations:

- The value of 'popularity' and 'votes' is subjective and dependent on those users voting and navigating through the website.
- The budget and revenue does not have currency specified. and the movies were made in different countries, due to which there is a fluctuation in the currency.
- For top grossing movies, I picked the top 100 box office revenue. I could have broken it down differently, such as over a billion and under, or billions / millions / under, but after assessing all I found this to be a good breakdown.
- I first split the genre column in the original df, then realized it affected assessment (such as most popular by title since the title was repeated over multiple rows) so I went back and created a new df with the genre split.