Project: Investigating TMDB Dataset

Table of Contents

Introduction

> This dataset comes from IMDB and contains information about 10,000 movies, short films and tv series collected from The Movie Database (TMDb), including user ratings, revenue, runtime and budget.
>
> In this project, i'll be answering the following questions:
>
> - What month is considered "best" for releasing a films/shows?
> - What is the relationship between runtime and vote average?
> - What genres are associated with films/shows that have high revenues?
> - What percentage do the top 5 genres make up?

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

Data Wrangling

Explore General Properties of the Dataset

In [2]:
```
# Read file and explore column names
tmdb_df = pd.read_csv('tmdb-movies.csv')
tmdb_df.head(1)
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.juras

1 rows × 21 columns

Data Cleaning - Drop Unecessary Columns

> Remove columns that are not useful for answering questions (Budget, Revenue, Homepage, Tagline, Keywords and Overview)

In [3]:
```
tmdb_df.drop(['budget','revenue','homepage','tagline','keywords','overview','rele
tmdb_df.columns
```

Out[3]:
```
Index(['id', 'imdb_id', 'popularity', 'original_title', 'runtime', 'genres',
       'production_companies', 'release_date', 'vote_count', 'vote_average',
       'budget_adj', 'revenue_adj'],
      dtype='object')
```

In [4]:
```
# Also remove production_companies
tmdb_df.drop(['production_companies'], axis=1, inplace=True)
tmdb_df.columns
```

Out[4]:
```
Index(['id', 'imdb_id', 'popularity', 'original_title', 'runtime', 'genres',
       'release_date', 'vote_count', 'vote_average', 'budget_adj',
       'revenue_adj'],
      dtype='object')
```

In [5]:
```
# Explore dataset
tmdb_df.describe()
```

Out[5]:

	id	popularity	runtime	vote_count	vote_average	budget_adj	rev
count	10866.000000	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.086
mean	66064.177434	0.646441	102.070863	217.389748	5.974922	1.755104e+07	5.136
std	92130.136561	1.000185	31.381405	575.619058	0.935142	3.430616e+07	1.446
min	5.000000	0.000065	0.000000	10.000000	1.500000	0.000000e+00	0.000
25%	10596.250000	0.207583	90.000000	17.000000	5.400000	0.000000e+00	0.000
50%	20669.000000	0.383856	99.000000	38.000000	6.000000	0.000000e+00	0.000
75%	75610.000000	0.713817	111.000000	145.750000	6.600000	2.085325e+07	3.369
max	417859.000000	32.985763	900.000000	9767.000000	9.200000	4.250000e+08	2.827

Data Cleaning - Filling 0 Values

> According to the data above, budget_adj, revenue_adj and runtime all contain values of 0. Fill these in with the average of each column.

In [5]:
```
# Get average of budget_adj
print(tmdb_df['budget_adj'].mean())
```

```
17551039.822886847
```

In [6]:
```
# Replace 0 values with mean.
tmdb_df['budget_adj'] = tmdb_df['budget_adj'].replace(0, 17551039.822886847)
```

In [7]:
```
# Get average of revenue_adj
print(tmdb_df['revenue_adj'].mean())
```

```
51364363.25325093
```

In [8]:
```
# Replace 0 values with mean
tmdb_df['revenue_adj'] = tmdb_df['revenue_adj'].replace(0, 51364363.25325093)
```

In [9]:
```
# Get average of runtime
print(tmdb_df['runtime'].mean())
```

```
102.07086324314375
```

In [10]:
```
# Replace 0 values with mean
tmdb_df['runtime'] = tmdb_df['runtime'].replace(0, 102.07086324314375)
tmdb_df.describe()
```

Out[10]:

	id	popularity	runtime	vote_count	vote_average	budget_adj	rev
count	10866.000000	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.08
mean	66064.177434	0.646441	102.362065	217.389748	5.974922	1.755104e+07	7.98
std	92130.136561	1.000185	30.902781	575.619058	0.935142	3.430616e+07	1.36
min	5.000000	0.000065	2.000000	10.000000	1.500000	0.000000e+00	2.37
25%	10596.250000	0.207583	90.000000	17.000000	5.400000	0.000000e+00	5.13
50%	20669.000000	0.383856	99.000000	38.000000	6.000000	0.000000e+00	5.13
75%	75610.000000	0.713817	111.000000	145.750000	6.600000	2.085325e+07	5.13
max	417859.000000	32.985763	900.000000	9767.000000	9.200000	4.250000e+08	2.82

Data Cleaning - Cleaning Duplicates

Find and remove duplicate rows

In [11]:
```
# Find out if there are any duplicate rows
sum(tmdb_df.duplicated())
```

Out[11]: 1

In [12]:
```
# Remove the duplicated rows
tmdb_df.drop_duplicates(inplace=True)
```

Data Cleaning - Changing Datatypes

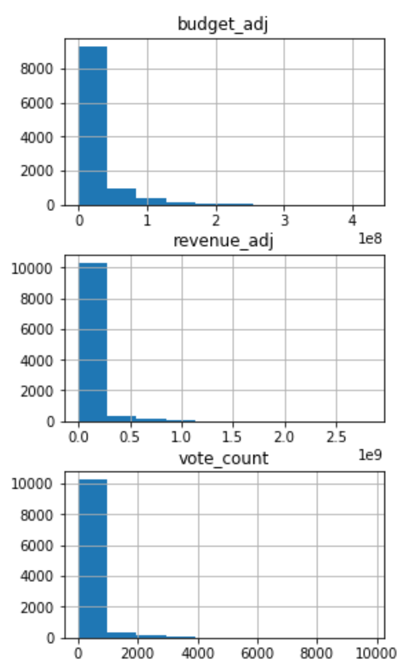
Change datatypes of columns to appropriate kinds. Ex. 'release_date' needs to be datetime.

In [13]:
```
tmdb_df['release_date'] =  pd.to_datetime(tmdb_df['release_date'])
tmdb_df.dtypes
```

Out[13]:
```
id                         int64
imdb_id                   object
popularity               float64
original_title            object
runtime                  float64
genres                    object
release_date      datetime64[ns]
vote_count                 int64
vote_average             float64
budget_adj               float64
revenue_adj              float64
dtype: object
```

Exploratory Data Analysis

In [14]:
```
# Explore what the histogram of the data looks like
tmdb_df.hist(figsize=(15,8));
```



What month is considered "best" for releasing a film/show?

> "Best" is a label that defines movies that have the top revenues. So i'll explore what month(s) have the highest revenues.

In [15]:
```
#  First, i'll create a new column called "month".
#  Then i'll extract the month from release_date.

tmdb_df['month'] = tmdb_df['release_date'].apply(lambda x: x.month)
```

In [16]:
```
tmdb_df.head(3)
```

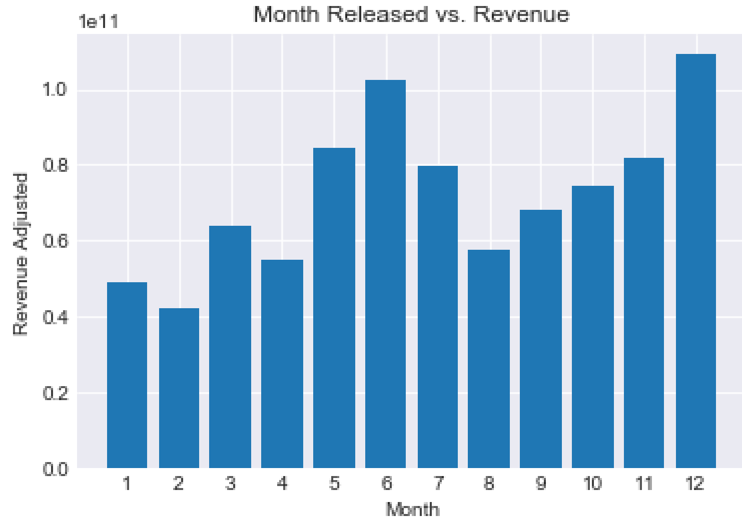
Out[16]:

	id	imdb_id	popularity	original_title	runtime	genres	release_date	vote_
0	135397	tt0369610	32.985763	Jurassic World	124.0	Action|Adventure|Science Fiction|Thriller	2015-06-09	
1	76341	tt1392190	28.419936	Mad Max: Fury Road	120.0	Action|Adventure|Science Fiction|Thriller	2015-05-13	
2	262500	tt2908446	13.112507	Insurgent	119.0	Adventure|Science Fiction|Thriller	2015-03-18	

In [17]:
```
# Group by month and sum the revenues.
month_revenue = tmdb_df.groupby('month')['revenue_adj'].sum()
month_revenue
```

Out[17]:
```
month
1     4.910687e+10
2     4.235442e+10
3     6.385725e+10
4     5.487055e+10
5     8.423232e+10
6     1.021322e+11
7     7.987658e+10
8     5.757434e+10
9     6.804293e+10
10    7.424615e+10
11    8.171477e+10
12    1.091239e+11
Name: revenue_adj, dtype: float64
```

In [18]:
```
# Now plot these in a bar chart.
sns.set_style('darkgrid')
plt.bar([1,2,3,4,5,6,7,8,9,10,11,12], month_revenue, tick_label = [1,2,3,4,5,6,7,
plt.title('Month Released vs. Revenue')
plt.ylabel('Revenue Adjusted')
plt.xlabel('Month');
```



From this chart, we can see that June and December have the highest revenue for movie releases. However, to make this conclusive, I must check the number of movie releases each month, because there could be a few high-earning movies that skew the data.

In [19]:
```
tmdb_df['month'].value_counts()
```

Out[19]:
```
9     1331
10    1153
12     985
1      919
8      918
6      827
3      822
11     814
5      809
7      799
4      797
2      691
Name: month, dtype: int64
```

In [20]:
```
tmdb_df['month'].value_counts().mean()
```

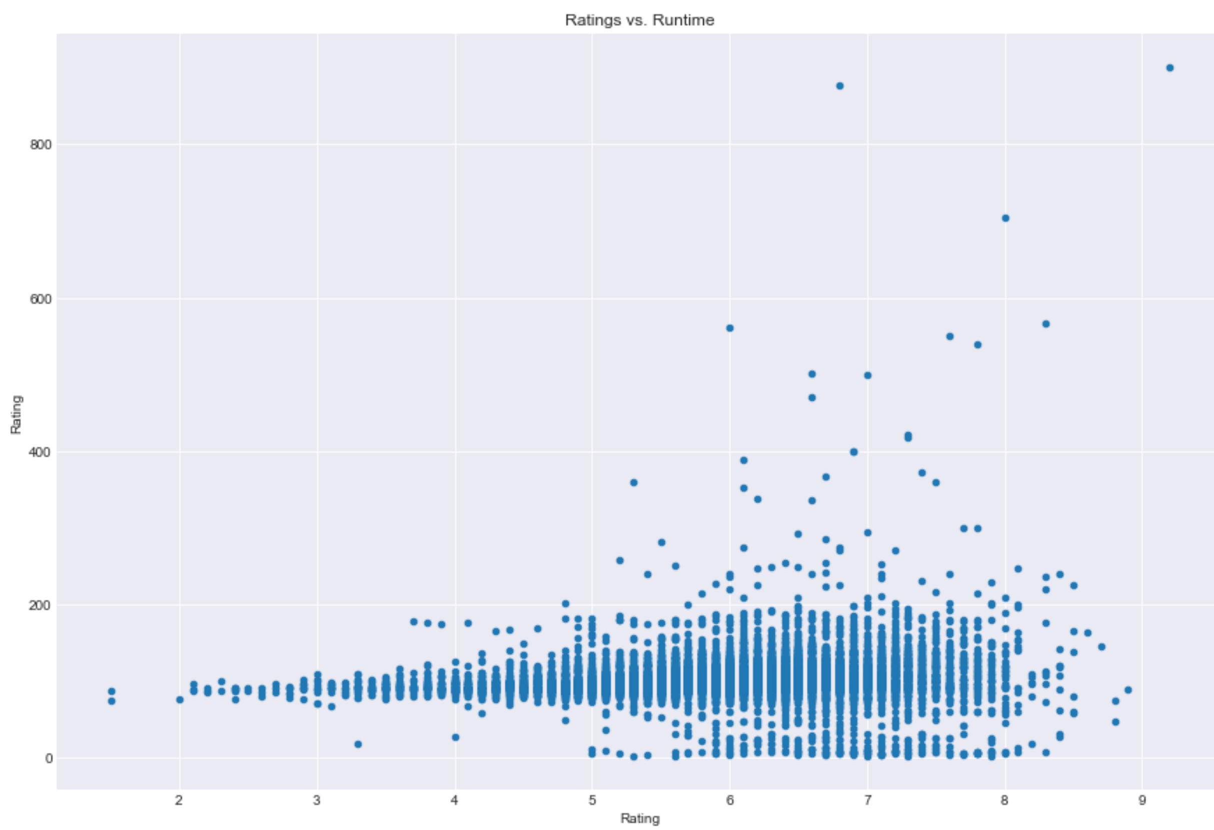
Out[20]:
```
905.4166666666666
```

Although the number of movie releases ranges from 691-1331, June and December are not far from the average number of 905.5 releases/month. This means there's no significant data to conclude there were a few high-earning movies that skewed the data, or that there were more movie releases during those months.

Therefore, we can still conclude June and December are "better" months to release movies in, as they'll most likely produce the highest revenue.

What is the relationship between runtime and vote average?

In [21]:
```
# Plot scatter plot of these two columns
tmdb_df.plot(x='vote_average', y='runtime', kind='scatter', figsize=(15,10))
plt.title('Ratings vs. Runtime')
plt.xlabel('Rating')
plt.ylabel('Rating');
```



From this scatter plot, we can draw several conclusions:

If it's a short film, it's likely to have a mid-to-high rating.

The ratings of films with a runtime of around 100 minutes are unpredictable, as they can run from low to high.

Films/shows with a runtime above or below 100 minues tend to have mid-to-high ratings.

Tv series (or movies with long runtimes) consistently get higher-than-average ratings.

What genres are associated with films/shows that have high revenues?

Most films/shows have more than one genre, so I need to make it so every row only has one genre.

In [22]:
```
tmdb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 12 columns):
id                  10865 non-null int64
imdb_id             10855 non-null object
popularity          10865 non-null float64
original_title      10865 non-null object
runtime             10865 non-null float64
genres              10842 non-null object
release_date        10865 non-null datetime64[ns]
vote_count          10865 non-null int64
vote_average        10865 non-null float64
budget_adj          10865 non-null float64
revenue_adj         10865 non-null float64
month               10865 non-null int64
dtypes: datetime64[ns](1), float64(5), int64(3), object(3)
memory usage: 1.1+ MB
```

In [23]:
```
# First remove Null values from genres column
tmdb_df = tmdb_df.dropna(subset=['genres'], axis=0)
tmdb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10842 entries, 0 to 10865
Data columns (total 12 columns):
id                10842 non-null int64
imdb_id           10834 non-null object
popularity        10842 non-null float64
original_title    10842 non-null object
runtime           10842 non-null float64
genres            10842 non-null object
release_date      10842 non-null datetime64[ns]
vote_count        10842 non-null int64
vote_average      10842 non-null float64
budget_adj        10842 non-null float64
revenue_adj       10842 non-null float64
month             10842 non-null int64
dtypes: datetime64[ns](1), float64(5), int64(3), object(3)
memory usage: 1.1+ MB
```

In [24]:
```
# Then split the genres column by the "|" character and make each genre a new col
genres = tmdb_df['genres'].str.split('|', expand=True).rename(columns = lambda x:
```

In [25]:
```
# Remove original genres column
tmdb_df.drop('genres', axis=1, inplace=True)
```

In [26]:
```
# Now merge new_tmdb and genres
tmdb_df = pd.merge(tmdb_df, genres, left_index=True, right_index=True, how='inner
```

In [27]:
```
# Now i'll pull the top 10 revenue-earning films/shows and create a new df.
top_rev = tmdb_df.nlargest(10, 'revenue_adj')
top_rev
```

Out[27]:

	id	imdb_id	popularity	original_title	runtime	release_date	vote_count	vote_avera
1386	19995	tt0499549	9.432768	Avatar	162.0	2009-12-10	8458	
1329	11	tt0076759	12.037933	Star Wars	121.0	1977-03-20	4428	
5231	597	tt0120338	4.355219	Titanic	194.0	1997-11-18	4654	
10594	9552	tt0070047	2.010733	The Exorcist	122.0	1973-12-26	1113	
9806	578	tt0073195	2.563191	Jaws	124.0	1975-06-18	1415	
3	140607	tt2488496	11.173104	Star Wars: The Force Awakens	136.0	2015-12-15	5292	
8889	601	tt0083866	2.900556	E.T. the Extra-Terrestrial	115.0	1982-04-03	1830	

In [28]:
```
# Now I want to group the genres by type and count the number of times they occur
# First, I'll create a copy of the df and delete all unrelated columns, then use
copy_df = top_rev.copy()
copy_df.drop(['id','imdb_id' ,'popularity','original_title','runtime','release_da
df1 = copy_df.melt()
```

In [29]:
```
# Then I'll use crosstab to group the genres, and assign it to a new df.
df2 = pd.crosstab(index=df1['value'], columns=df1['variable'])
df2
```

Out[29]:

variable	genre1	genre2	genre3	genre4	genre5
value					
Action	2	2	0	0	1
Adventure	2	3	2	0	0
Animation	0	1	0	0	0
Comedy	0	0	1	0	0
Crime	1	0	0	0	0
Drama	2	1	0	0	0
Family	0	0	1	1	0
Fantasy	0	0	1	2	0
Horror	1	1	0	0	0
Mystery	0	0	1	0	0
Romance	0	1	0	0	0
Science Fiction	2	0	2	1	0
Thriller	0	1	2	1	0

In [31]:
```
# Create a new column with the totals for each genre title
df2['totals'] = df2['genre1'] + df2['genre2'] + df2['genre3'] + df2['genre4'] + d
df2
```

Out[31]:

variable	genre1	genre2	genre3	genre4	genre5	totals
value						
Action	2	2	0	0	1	5
Adventure	2	3	2	0	0	7
Animation	0	1	0	0	0	1
Comedy	0	0	1	0	0	1
Crime	1	0	0	0	0	1
Drama	2	1	0	0	0	3
Family	0	0	1	1	0	2
Fantasy	0	0	1	2	0	3
Horror	1	1	0	0	0	2
Mystery	0	0	1	0	0	1
Romance	0	1	0	0	0	1
Science Fiction	2	0	2	1	0	5
Thriller	0	1	2	1	0	4

In [32]:
```
# Plot in a bar chart
df2['totals'].plot(kind="bar", figsize=(8,5), fontsize=12)
plt.xlabel('Genre', fontsize = 14)
plt.ylabel('Frequency', fontsize = 14)
plt.title('Genres of the Highest Earning Films/Shows', fontsize = 14);
```

From this bar chart, we can see that of the films/shows that have earned the highest revenues, the three most frequently occuring genres are Adventure (7), Action (5) and Science Fiction (5).

If you are a movie production company, this information would be useful for determining which genres land on the "highest revenue earning" list more often than others.

What percentage do the top 5 genres make up?

In [33]:
```
# I'll use the same technique from above to count all of the genres, but from the
copy_df = tmdb_df.copy()
copy_df.drop(['id','imdb_id' ,'popularity','original_title','runtime','release_da
df3 = copy_df.melt()
```

In [34]:
```
# Then I'll use crosstab to group the genres, and assign it to a new df.
df4 = pd.crosstab(index=df3['value'], columns=df3['variable'])
```

In [42]:
```
# Create a new column with the totals for each genre title, and orderby the top 3
df4['totals'] = df4['genre1'] + df4['genre2'] + df4['genre3'] + df4['genre4'] + d
```

In [43]:
```
# Identify the top 3 genres.
top5 = df4.nlargest(5, 'totals')
top5
```

Out[43]:

variable	genre1	genre2	genre3	genre4	genre5	totals
value						
Horror	915	489	186	36	11	1637
Adventure	586	626	183	62	14	1471
Crime	380	449	350	152	23	1354
Family	144	448	401	178	60	1231
Science Fiction	214	330	401	216	68	1229

In [192]:
```
# Drop the top 5 genres from df
df4.drop(['Drama','Comedy','Thriller','Action','Romance'], inplace=True)
```

In [193]:
```
# Sum the remaining genres - this will be the "other" category - and turn it into
df4['totals'].sum()
```

Out[193]: 11399

In [241]:
```
# Add "other" group as a row to the top 5 dataframe.
count = top5.append({'totals':'11399'}, ignore_index=True)
count
```

Out[241]:

variable	genre1	genre2	genre3	genre4	genre5	totals
0	2453.0	1618.0	546.0	124.0	19.0	4760
1	2319.0	990.0	388.0	81.0	15.0	3793
2	491.0	961.0	886.0	449.0	120.0	2907
3	1590.0	544.0	198.0	42.0	10.0	2384
4	186.0	704.0	583.0	194.0	45.0	1712
5	NaN	NaN	NaN	NaN	NaN	11399

In [244]:
```
# Relabel index
count.index=['Drama','Comedy','Thriller','Action','Romance','Other']
count
```

Out[244]:

variable	genre1	genre2	genre3	genre4	genre5	totals
Drama	2453.0	1618.0	546.0	124.0	19.0	4760
Comedy	2319.0	990.0	388.0	81.0	15.0	3793
Thriller	491.0	961.0	886.0	449.0	120.0	2907
Action	1590.0	544.0	198.0	42.0	10.0	2384
Romance	186.0	704.0	583.0	194.0	45.0	1712
Other	NaN	NaN	NaN	NaN	NaN	11399

In [281]:
```
# Get grand total of number of genres
genre_total = count['totals'].sum()
genre_total
```

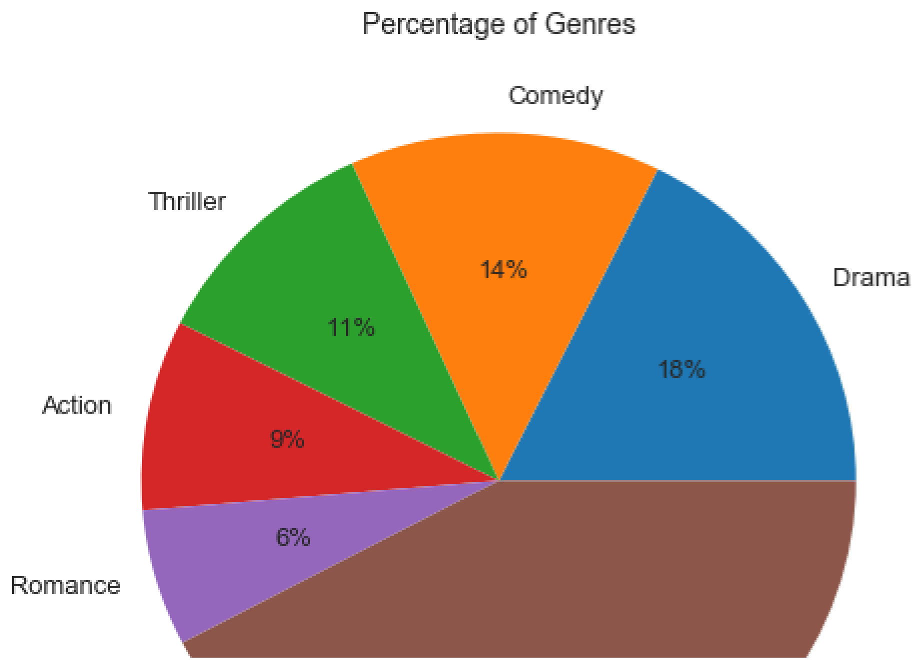
Out[281]: 26955

In [288]:
```
# Calculate the percentage of the total for each genre and add to new column.
count['percentage'] = count.loc[:,'totals'] / 26955 * 100
count
```

Out[288]:

variable	genre1	genre2	genre3	genre4	genre5	totals	percentage
Drama	2453.0	1618.0	546.0	124.0	19.0	4760	17.659061
Comedy	2319.0	990.0	388.0	81.0	15.0	3793	14.071601
Thriller	491.0	961.0	886.0	449.0	120.0	2907	10.784641
Action	1590.0	544.0	198.0	42.0	10.0	2384	8.844370
Romance	186.0	704.0	583.0	194.0	45.0	1712	6.351326
Other	NaN	NaN	NaN	NaN	NaN	11399	42.289000

In [300]:
```
# Plot in a pie chart
count['percentage'].plot(kind="pie", figsize=(8,8), fontsize=13, autopct='%1.0f%%
plt.title('Percentage of Genres', fontsize = 14)
plt.ylabel('');
```



From this pie chart, we can see that out of the top 5 genres, Drama is the most frequently made. This means that close to 1 out of every 5 films/shows is in the Drama category.

However we can see that these top 5 genres only made up just over half of the total number of films/shows produced - we still have several other less produced genres that when combined, make up a good portion of the whole.

Also, we can see that just because a genre produces a larger revenue than others, doesn't necessarily mean that it's going to be one of the most frequently produced genres as well.

Conclusions

Throughout this data analysis, I posed questions that Production Companies might find useful, and I've come to several conclusions:

It is best to release a movie/show in June or December, because I can conclusively say that those movies are more popular and tend to bring in the most revenue. This could be due to the fact that in the Summer and Winter, families are looking for things to do together.

The conclusions I've come to in analyzing the relationship between ratings and runtime are that short films (less than 10 minutes) are likely to have a mid-to-high rating, and TV series (greater than 300 minutes) consistently get higher-than-average ratings. The ratings of films/shows with a runtime of around 100 minutes are unpredictable, as they can run from low to high, and films with a runtime above or below 100 minues tend to have mid-to-high ratings. Just at first glance of the scatterplot, users are more friendly - as in they tend to give mostly mid-to-high ratings overall - so production companies will want to make sure their film/show is reviewed on TMDB.

If you're a production company and you want to know what genres earn the highest revenues, my bar chart above concluded that out of the top earning films/shows, Adventure, Action and Science Fiction were the most frequent genres on that list. You can conclude that you are more likely to earn a higher revenue if you produce those genres.

Finally, when I calculated the percentages of each genre, I noticed that only of the highet earning genres is in the top 5 most frequently produced genres (Action). Perhaps it is because Adventure and Science Fiction movies are more expensive to produce so they are more rarely made, or perhaps production companies want to focus on genres that are more popular with people, not necessarily the genres that produce the highest revenues. No matter the cause, I can conlcude that just because a genre produces a larger revenue than others, doesn't necessarily mean that it's going to be one of the most frequently produced genres.

A few notes about my data cleaning are that in the runtime, budget_adj and revenue_adj I filled all of the 0 values with their means. This possibly could've been more accurate if I used regression to find like-properties to fill the 0 values instead of the mean.

Resources I used in my analysis:

https://stackoverflow.com/questions/47517831/how-to-copy-column-with-the-pandas-and-change-the-name (https://stackoverflow.com/questions/47517831/how-to-copy-column-with-the-pandas-and-change-the-name) https://stackoverflow.com/questions/25146121/extracting-just-month-and-year-from-pandas-datetime-column-python (https://stackoverflow.com/questions/25146121/extracting-just-month-and-year-from-pandas-datetime-column-python) https://stackoverflow.com/questions/30405413/python-pandas-extract-year-from-datetime-dfyear-dfdate-year-is-not (https://stackoverflow.com/questions/30405413/python-pandas-extract-year-from-datetime-dfyear-dfdate-year-is-not) https://stackoverflow.com/questions/48733618/how-to-drop-rows-from-a-