**Capstone III**

# Exploración sobre la base de datos de películas

Luis de la Ossa
Máster en Ciencia de Datos e Ingeniería de Datos en la Nube
Universidad de Castilla-La Mancha

Marta Bellón Castro
Curso 2022-2023

En este proyecto se parte de un conjunto de datos sobre películas y valoraciones sobre las mismas disponibles en *Internet Movie Database (IMDB)*. Esta información se ha enriquecido con datos adicionales obtenidos mediante consultas a la API de otro sitio, *The Movie Database (TMDB)*. Como resultado de este proceso, se han generado tres conjuntos de datos: `df_movies` , con información sobre películas, géneros, y valoraciones; `df_people` , con información sobre las personas; y `df_credits` con información sobre las personas que participan en cada película (director y tres actores principales). A partir de éstos, se llevará a cabo un análisis exploratorio.

El modo en que se desarrolla el análisis depende en gran medida *de los datos disponibles* y de unos *objetivos* que pueden ser generales, o la respuesta a cuestiones concretas, y que pueden variar o ampliarse en función de los resultados intermedios que vaya arrojando el proceso. En este proyecto se partirá de una situación hipotética inicial en la que se estudian las condiciones para la inversión en una película de *presupuesto relativamente bajo*. Para ello, se analizarán qué factores tienen más incidencia en el éxito de una película, que consideraremos que puede cuantificarse por los ingresos que ha obtenido, o por el número de votos y valoración en *IMDB*. También se estudiarán las diferencias entre géneros con el fin de seleccionar los más adecuados, y se analizará la relevancia de directores, actores, y actrices, y cuales podrían ser los más adecuados. Además, y paralelamente, se obtendrá alguna información adicional que, si bien puede no estár relacionada con el objetivo principal, puede resultar de interés.

Para llevar a cabo la exploración, se propondrán una serie de ejercicios que consisten en la obtención de gráficas o datos concretos. Por último, este proyecto no tiene porqué ceñirse a la resolución de los ejercicios, sino que cabe la posibilidad de dar respuesta a cuestiones distintas a las propuestas en la libreta.

ℹ Una parte esencial de la exploración es la validación y limpieza de los datos. Como esta parte es más "árida", en este proyecto nos centraremos en la exploración propiamente dicha. En el Capstone V veremos un caso en el que la preparación de los datos es algo más compleja.

# Índice



---

In [1]:
```
from IPython.display import display, HTML
display(HTML("<style>.container { width:98% !important; }</style>"))

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

%config InlineBackend.figure_format = 'retina'
%matplotlib inline
```

---

# 1. Primera inspección

El primer paso en el análisis exploratorio consiste en obtener una primera descripción del conjunto de datos como tal, y que incluye aspectos como: archivos; características, relaciones entre ellas, significado y tipos de datos con que se representan; existencia o frecuencia de valores perdidos, etc. A continuación se lee y se muestra la cabecera de cada uno de los tres *DataFrame*, y se inspecciona su estructura.

**`df_movies`**

In [2]:
```
df_movies = pd.read_csv('data/df_movies.csv', index_col='movie_id', sep=';', parse_dates=['release_date'])
display(df_movies.head(2))
df_movies.info()
```

| movie_id | title | genres | avg_rating | num_votes | budget | revenue | release_date | original_language | popularity | keywords |
|---|---|---|---|---|---|---|---|---|---|---|
| **tt0010323** | The Cabinet of Dr. Caligari | Horror,Mystery,Thriller | 8.0 | 64516 | 18000 | 8811 | 1920-02-27 | de | 14.790 | insane asylum,black and white,cult film,silent... |
| **tt0012349** | The Kid | Comedy,Drama,Family | 8.3 | 126789 | 250000 | 2500000 | 1921-01-21 | en | 14.697 | angel,suicide attempt,fistfight,slapstick come... |

```
<class 'pandas.core.frame.DataFrame'>
Index: 3824 entries, tt0010323 to tt9893250
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              3824 non-null   object
 1   genres             3824 non-null   object
 2   avg_rating         3824 non-null   float64
 3   num_votes          3824 non-null   int64
 4   budget             3824 non-null   int64
 5   revenue            3824 non-null   int64
 6   release_date       3824 non-null   datetime64[ns]
 7   original_language  3824 non-null   object
 8   popularity         3824 non-null   float64
 9   keywords           3809 non-null   object
dtypes: datetime64[ns](1), float64(2), int64(3), object(4)
memory usage: 328.6+ KB
```

Puede apreciarse que todos los tipos de datos son los esperados, y que aparentemente no hay datos perdidos salvo en el caso de las palabras clave (que no se utilizarán en este proyecto).

**df_people**

In [3]:
```
df_people = pd.read_csv('data/df_people.csv', index_col='people_id', sep=';')
display(df_people.head(2))
df_people.info()
```

| people_id | name | popularity | imdb_id | gender |
|---|---|---|---|---|
| **2991** | Robert Wiene | 1.886 | nm0927468 | 2 |
| **13848** | Charlie Chaplin | 15.496 | nm0000122 | 2 |

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5791 entries, 2991 to 52775
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        5791 non-null   object
 1   popularity  5791 non-null   float64
 2   imdb_id     5748 non-null   object
 3   gender      5791 non-null   int64
dtypes: float64(1), int64(1), object(2)
memory usage: 226.2+ KB
```

Los datos corresponden también a lo esperado, y que existen valores perdidos para la columna `imdb_id`. Se sobreentiende que algunos de los directores/actores no tienen identificador en la plataforma `IMDb`.

**df_credits**

In [4]:
```
df_credits = pd.read_csv('data/df_credits.csv', sep=';')
display(df_credits.head(2))
df_credits.info()
```

| | people_id | movie_id | rol |
|---|---|---|---|
| **0** | 2991 | tt0010323 | director |
| **1** | 13848 | tt0012349 | director |

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15606 entries, 0 to 15605
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   people_id  15606 non-null  int64
 1   movie_id   15606 non-null  object
 2   rol        15606 non-null  object
dtypes: int64(1), object(2)
memory usage: 365.9+ KB
```

Puede observarse que tampoco existen valores perdidos en este conjunto, y que la codificación de los datos es correcta.

Como consecuencia de esta primera aproximación, se concluye que no es necesario hacer ningún tipo de preprocesamiento, ni adaptar el proceso a la limpieza de datos, por lo que la exploración se centrará en la información.

## 1.1 Exploración inicial de `df_movies`

El *DataFrame* `df_movies` es el núcleo del conjunto de datos, y entorno al cual debe desarrollarse la exploración. Contiene información identificativa, cualitativa, y cuantitativa sobre cada película. En primer lugar se hará una inspección de los datos para comprobar si existen valores extremos.

### Ejercicio 1

Obtener una descripción de las columnas numéricas del *DataFrame* `df_movies` . Con el fin de observar si existen valores extremos y en qué proporción, incorporar a los resultados los cuantiles `[0.25, 0.5, 0.75, 0.95, 0.99, 0.995]` .

In [5]:
```
# Describo las columnas numéricas de df_movies incorporando los cuartiles [0.25, 0.5, 0.75, 0.95, 0.99, 0.995]
df_movies_quant = df_movies.describe(percentiles=[0.25, 0.5, 0.75, 0.95, 0.99, 0.995])
df_movies_quant
```

Out[5]:

| | avg_rating | num_votes | budget | revenue | popularity |
|---|---|---|---|---|---|
| **count** | 3824.000000 | 3.824000e+03 | 3.824000e+03 | 3.824000e+03 | 3824.000000 |
| **mean** | 6.846810 | 1.948407e+05 | 4.229992e+07 | 1.424842e+08 | 50.848736 |
| **std** | 0.914069 | 2.245797e+05 | 4.997971e+07 | 2.189518e+08 | 122.146243 |
| **min** | 1.000000 | 5.003300e+04 | 0.000000e+00 | 0.000000e+00 | 0.600000 |
| **25%** | 6.300000 | 7.371000e+04 | 7.000000e+06 | 2.073692e+07 | 16.158250 |
| **50%** | 6.900000 | 1.157800e+05 | 2.500000e+07 | 6.827535e+07 | 24.984000 |
| **75%** | 7.500000 | 2.192848e+05 | 6.000000e+07 | 1.707762e+08 | 49.250000 |
| **95%** | 8.200000 | 6.222014e+05 | 1.500000e+08 | 5.438717e+08 | 153.238750 |
| **99%** | 8.500000 | 1.144504e+06 | 2.085400e+08 | 1.047176e+09 | 373.312660 |
| **99.5%** | 8.700000 | 1.393663e+06 | 2.500000e+08 | 1.242341e+09 | 689.647770 |
| **max** | 9.300000 | 2.660316e+06 | 3.800000e+08 | 2.920357e+09 | 4328.431000 |

Algo que se observa es que existen películas para las que el presupuesto y/o la recaudación son cero, lo que realmente corresponde a un valor perdido. Se van a tratar de forma distinta ambas columnas. Puesto que la recaudación es el factor de más interés, se eliminarán las películas para las que no la conozcamos.

Eliminar las películas cuyo valor `revenue` es cero (seleccionar aquellas cuya recaudación es mayor que cero).

In [6]:
```
# Cojo las peliculas con recaudación mayor a cero
df_movies = df_movies.loc[(df_movies['revenue']>0)]

# Obtengo la descripción
df_movies.describe(percentiles=[0.25, 0.5, 0.75, 0.95, 0.99, 0.995])
```

Out[6]:

| | avg_rating | num_votes | budget | revenue | popularity |
|---|---|---|---|---|---|
| **count** | 3552.000000 | 3.552000e+03 | 3.552000e+03 | 3.552000e+03 | 3552.000000 |
| **mean** | 6.838598 | 2.030838e+05 | 4.471695e+07 | 1.533951e+08 | 51.019731 |
| **std** | 0.888174 | 2.305133e+05 | 5.040893e+07 | 2.234678e+08 | 112.899118 |
| **min** | 1.600000 | 5.003300e+04 | 0.000000e+00 | 3.030000e+02 | 0.600000 |
| **25%** | 6.300000 | 7.636675e+04 | 1.000000e+07 | 2.932804e+07 | 16.699500 |
| **50%** | 6.900000 | 1.223510e+05 | 2.800000e+07 | 7.822308e+07 | 25.838000 |
| **75%** | 7.500000 | 2.316610e+05 | 6.000000e+07 | 1.809299e+08 | 50.710000 |
| **95%** | 8.100000 | 6.360912e+05 | 1.568000e+08 | 5.768329e+08 | 153.769150 |
| **99%** | 8.500000 | 1.160129e+06 | 2.124500e+08 | 1.066302e+09 | 371.924420 |
| **99.5%** | 8.624500 | 1.426828e+06 | 2.500000e+08 | 1.266142e+09 | 673.006490 |
| **max** | 9.300000 | 2.660316e+06 | 3.800000e+08 | 2.920357e+09 | 4328.431000 |

Por otra parte, se visualizan las películas en las que los presupuestos que sean igual a cero.

In [7]:
```
# Películas con presupuesto igual a cero
df_movies.loc[(df_movies['budget']==0)].describe(percentiles=[0.25, 0.5, 0.75, 0.95, 0.99, 0.995])
```

Out[7]:

| | avg_rating | num_votes | budget | revenue | popularity |
|---|---|---|---|---|---|
| **count** | 178.000000 | 178.000000 | 178.0 | 1.780000e+02 | 178.000000 |
| **mean** | 7.024719 | 80179.932584 | 0.0 | 2.300329e+07 | 24.632944 |
| **std** | 0.791092 | 33542.553010 | 0.0 | 4.118649e+07 | 24.111305 |
| **min** | 3.300000 | 50033.000000 | 0.0 | 3.030000e+02 | 5.753000 |
| **25%** | 6.600000 | 57796.750000 | 0.0 | 2.965419e+06 | 12.608500 |
| **50%** | 7.100000 | 67562.000000 | 0.0 | 8.312787e+06 | 16.856500 |
| **75%** | 7.675000 | 88636.000000 | 0.0 | 2.940498e+07 | 26.154000 |
| **95%** | 8.100000 | 155648.650000 | 0.0 | 7.755945e+07 | 64.539500 |
| **99%** | 8.400000 | 177673.590000 | 0.0 | 1.919054e+08 | 139.306980 |
| **99.5%** | 8.411500 | 203013.315000 | 0.0 | 2.656269e+08 | 169.011300 |
| **max** | 8.500000 | 266274.000000 | 0.0 | 3.579861e+08 | 171.684000 |

En principio parecen películas que han recibido un gran número de votos, y algunas de ellas han conseguido una gran recaudación, por lo que no se descartarán. Se codificarán como *NaN* para tratarlos adecuadamente llegado el caso.

Asignar el valor *NaN* a los valores que son cero en la columna `df_movies['budget']` .

In [8]:
```
# Sustituyo los valores budget=0 por NaN
df_movies.loc[(df_movies['budget']==0), 'budget'] = np.nan
display(df_movies)

# Compruebo que no quedan 0, y que son valores NaN
print('Cantidad de 0 en df_movies: ' + str(df_movies['budget'].eq(0).sum()))
print('Cantidad de valores NaN en df_movies: ' + str(df_movies['budget'].isnull().sum()))
```

| movie_id | title | genres | avg_rating | num_votes | budget | revenue | release_date | original_language | popularity | keywords |
|---|---|---|---|---|---|---|---|---|---|---|
| **tt0010323** | The Cabinet of Dr. Caligari | Horror,Mystery,Thriller | 8.0 | 64516 | 18000.0 | 8811 | 1920-02-27 | de | 14.790 | insane asylum,black and white,cult film,silent... |
| **tt0012349** | The Kid | Comedy,Drama,Family | 8.3 | 126789 | 250000.0 | 2500000 | 1921-01-21 | en | 14.697 | angel,suicide attempt,fistfight,slapstick come... |
| **tt0013442** | Nosferatu | Fantasy,Horror | 7.9 | 98252 | NaN | 19054 | 1922-02-16 | de | 16.524 | germany,transylvania,loss of loved one,shapesh... |
| **tt0015648** | Battleship Potemkin | Drama,History,Thriller | 7.9 | 58131 | NaN | 45100 | 1925-12-24 | ru | 14.994 | odessa,baby carriage,cossack,panic,slaughter,m... |
| **tt0015864** | The Gold Rush | Adventure,Comedy,Drama | 8.2 | 111760 | 923000.0 | 4000000 | 1925-07-12 | en | 14.983 | river,gold,dance,worker,cabin,gold rush,thanks... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **tt9620292** | Promising Young Woman | Crime,Drama,Mystery | 7.5 | 172013 | NaN | 13868965 | 2020-12-13 | en | 27.732 | ohio,coffee shop,psychopath,horror,sociopath,r... |
| **tt9639470** | Last Night in Soho | Drama,Horror,Mystery | 7.1 | 134957 | 43000000.0 | 22957625 | 2021-10-21 | en | 69.686 | london, england,go-go dancer,nightmare,time tr... |
| **tt9731534** | The Night House | Horror,Mystery,Thriller | 6.5 | 53347 | NaN | 14590542 | 2021-07-15 | en | 24.783 | depression,nightmare,widow,forest,grief,grievi... |
| **tt9770150** | Nomadland | Drama | 7.3 | 160491 | 5000000.0 | 14784114 | 2020-12-04 | en | 24.996 | factory worker,based on novel or book,recessio... |
| **tt9784798** | Judas and the Black Messiah | Biography,Drama,History | 7.4 | 79914 | 26000000.0 | 6416063 | 2021-02-12 | en | 20.523 | assassination,chicago, illinois,black panther ... |

3552 rows × 10 columns

```
Cantidad de 0 en df_movies: 0
Cantidad de valores NaN en df_movies: 178
```

Por otra parte, si bien los rangos de las variables son muy amplios en algunos casos, se aprecia una diferencia importante entre el valor **máximo** de la columna `popularity` y el resto. Es necesario por tanto determinar si se trata de un error, o por el contrario existen valores fuera de rango, pero que deben ser considerados dentro del estudio.

Mostrar las 10 filas de `df_movies` con mayor valor de popularidad.

In [9]:
```
# Muestro las 10 filas de df_movies con mayor valor de popularidad.
df_movies.sort_values(by='popularity', ascending=False).head(10)
```

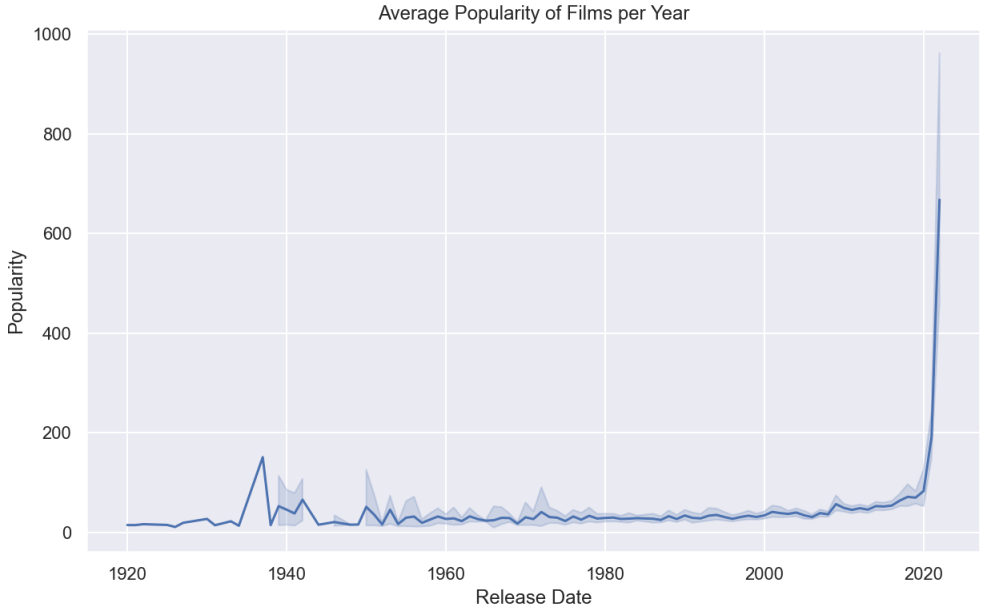
Out[9]:

| movie_id | title | genres | avg_rating | num_votes | budget | revenue | release_date | original_language | popularity | keywords |
|---|---|---|---|---|---|---|---|---|---|---|
| **tt6443346** | Black Adam | Action,Adventure,Fantasy | 7.0 | 109213 | 200000000.0 | 319000000 | 2022-10-19 | en | 4328.431 | lightning,anti hero,superhero,based on comic,d... |
| **tt1745960** | Top Gun: Maverick | Action,Drama | 8.4 | 423820 | 170000000.0 | 1482656000 | 2022-05-24 | en | 1491.786 | fighter pilot,u.s. navy,sequel,nuclear weapons... |
| **tt10648342** | Thor: Love and Thunder | Action,Adventure,Comedy | 6.3 | 305310 | 250000000.0 | 760000000 | 2022-07-06 | en | 1404.107 | ex-girlfriend,hero,greek mythology,sequel,supe... |
| **tt12593682** | Bullet Train | Action,Comedy,Thriller | 7.3 | 171979 | 90000000.0 | 239260044 | 2022-07-03 | en | 1339.013 | japan,assassin,based on novel or book,mission,... |
| **tt8041270** | Jurassic World: Dominion | Action,Adventure,Sci-Fi | 5.7 | 147762 | 165000000.0 | 1001000000 | 2022-06-01 | en | 1267.666 | giant monster,dinosaur,child kidnapping,jurass... |
| **tt10872600** | Spider-Man: No Way Home | Action,Adventure,Fantasy | 8.3 | 742520 | 200000000.0 | 1916050941 | 2021-12-15 | en | 1128.705 | new york city,loss of loved one,showdown,secre... |
| **tt5113044** | Minions: The Rise of Gru | Adventure,Animation,Comedy | 6.6 | 60622 | 85000000.0 | 936000000 | 2022-06-29 | en | 1036.561 | 1970s,duringcreditsstinger,minions,gentleminions |
| **tt1825683** | Black Panther | Action,Adventure,Sci-Fi | 7.3 | 757797 | 200000000.0 | 1347597973 | 2018-02-13 | en | 1006.821 | africa,superhero,based on comic,aftercreditsst... |
| **tt0499549** | Avatar | Action,Adventure,Fantasy | 7.8 | 1243973 | 237000000.0 | 2920357254 | 2009-12-15 | en | 936.245 | culture clash,future,space war,space colony,so... |
| **tt12412888** | Sonic the Hedgehog 2 | Action,Adventure,Comedy | 6.5 | 64453 | 110000000.0 | 401800000 | 2022-03-30 | en | 918.615 | sequel,based on video game,duringcreditsstinge... |

La visualización de los datos permite intuir que la mayoría de las películas para las que el valor de la columna `popularity` es extremadamente alto, son conocidas y de alto presupuesto. Llama la atención también un fenómeno, y es que gran parte de ellas se estrenaron el año pasado o éste. Eso lleva a pensar si el indicador podría estar relacionado con una mayor actividad en el sitio web fruto del uso cada vez más intensivo de las redes sociales; o como en el caso de otras plataformas (*Spotify*, por ejemplo) es un indicador variable que refleja el interés en el momento actual. Esta circunstancia puede comprobarse mostrando la popularidad media de las películas por año.

Existen diversas formas de mostrar la relación entre popularidad y fecha de estreno, una de ellas es mediante un gráfico de barras. Sin embargo, no es la más conveniente debido a que se está contemplando un periodo de casi 100 años y sería más difícil identificar tendencias. Dibujar una gráfica de tipo `sns.lineplot()` que muestre la popularidad media por año.

In [10]:
```
# Dibujo el grafico de la popularidad media por año
plt.figure(figsize=(10,6))
ax = sns.lineplot(data=df_movies, x=df_movies['release_date'].dt.year, y=df_movies['popularity']);
ax.set(xlabel='Release Date', ylabel='Popularity', title='Average Popularity of Films per Year')
plt.show()
```



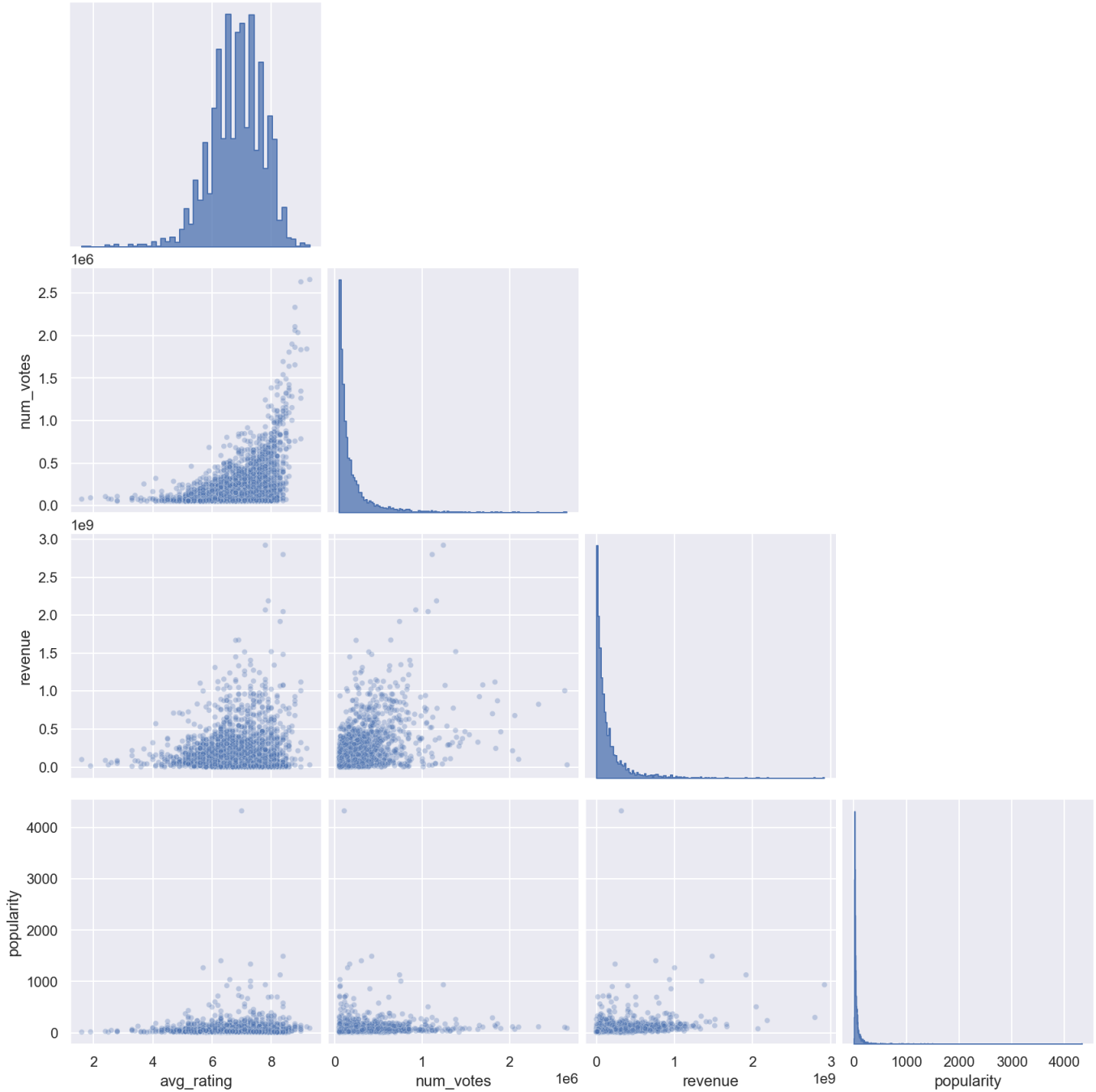

Efectivamente, se muestra que la popularidad de las películas crece con la fecha del estreno, y que se produce un aumento dramático a partir del año 2020, situándose la media del último año entorno a 650. Esta circunstancia es importante ya que se ha de vigilar el tratamiento que se hace con los valores de la columna `popularity`.

Además de `popularity`, el *DataFrame* `df_movies` incluye las variables relacionadas directamente con el ***éxito***, que es el objetivo principal del análisis. Como punto de partida se visualizarán las variables `avg_rating`, `num_votes`, `revenue`, y `popularity`, así como las relaciones entre ellas. Existen varias formas de hacerlo, pero *seaborn* permite mostrar toda la información en un solo gráfico.

### Ejercicio 2

Generar un figura con una matriz de gráficas en las que se muestre, para cada par de variables (de entre las cuatro mencionadas), una gráfica de dispersión; y para cada variable, un histograma. Esta figura se puede hacer de manera sencilla mediante `seaborn.pairplot()` o `seaborn.PairGrid()`. Debido a que el conjunto de datos contiene información sobre casi 4000 películas, es necesario hacer ciertos ajustes sobre las gráficas de dispersión, como por ejemplo utilizar un tamaño reducido o transparencia para las marcas.

In [11]:
```
success_meas = ['avg_rating','num_votes', 'revenue', 'popularity'] # Columnas correspondientes a la valoración


# Creo la figura con la matriz de gráficas

sns.pairplot(
    df_movies,
    vars=success_meas,
    height=3,
    corner=True,
    diag_kind='hist',
    diag_kws=dict(
        element='step',
        fill=True,
        stat="density",
        linewidth=1.0
    ),
    plot_kws=dict(size=20, alpha=0.3), # Tamaño y transparencia de los puntos de dispersión
    grid_kws=dict(diag_sharey=False) # Los histogramas no comparten eje Y
);
```


avg_rating
num_votes
1e6
revenue
1e9
popularity
num_votes
1e6
revenue
1e9
popularity

La gráfica anterior muestra que los valores de algunas variables ( `num_votes` , `revenue` , y `popularity` ) están muy sesgados, de modo que la mayoría se situan en la parte más baja del rango. Aún así, se percibe cierta relación entre `avg_rating` y `num_votes` , ya que parece que la gente tiende a votar más las películas que le gustan. También se percibe una relación similar, aunque menos fuerte, entre `avg_rating` y `revenue` .

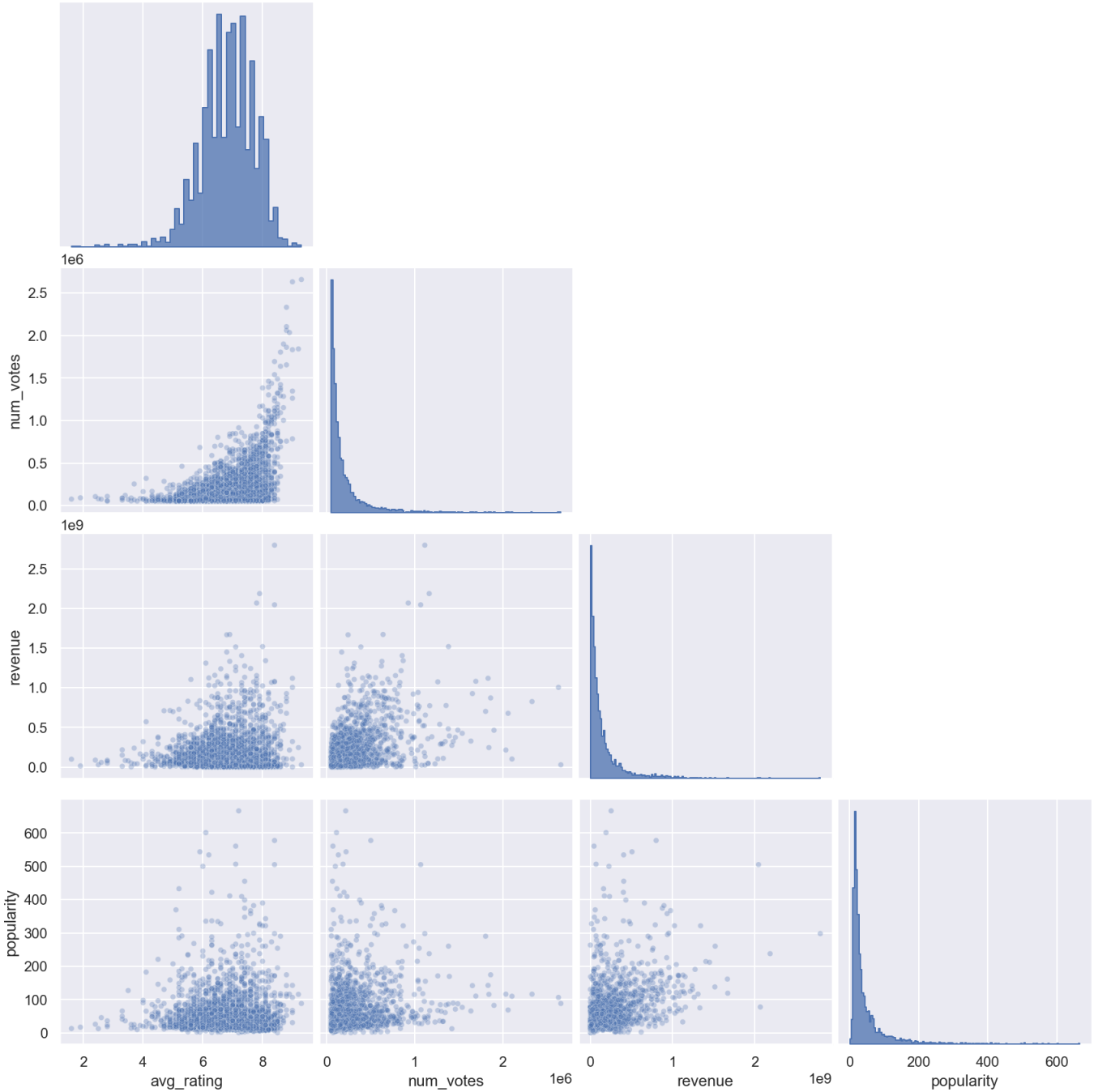
Aunque a priori no parece conveniente eliminar las películas con valores altos en `popularity` , los *outliers* dificultan la visión tanto del histograma correspondiente como de las gráficas de dispersión. Por tanto, se ha de repetir la gráfica, pero descartando las películas en las que `popularity` toma un valor extremadamente alto. Anteriormente se ha visto que, si bien la popularidad crece de forma pronunciada, la media en el último año se situa ligeramente por encima de 650. También que la diferencia crece de manera dramática a partir del cuantil 0.995. Por tanto, se almacenará este valor en la variable `pop_outlier_thresh` , y se utilizará como umbral.

In [12]:
```
# Peliculas con mayor popularidad
pop_outlier_thresh = df_movies['popularity'].quantile(0.995)
pop_outlier_thresh
```

Out[12]:
```
673.0064899999971
```

Repetir la gráfica elaborada en el ejercicio uno, pero descartando las filas correspondientes a valores de `popularity` mayores que `pop_outlier_thresh` (sin eliminarlos del *DataFrame* `df_movies` ).

In [13]:
```
# Realizo la selección condicional de las filas
df_movies_less_popular = df_movies.loc[df_movies['popularity'] < pop_outlier_thresh]

# Creo la figura con la matriz de gráficas

success_meas = ['avg_rating','num_votes', 'revenue', 'popularity'] # Columnas relativas a la valoración

sns.pairplot(
    df_movies_less_popular,
    vars=success_meas,
    height=3,
    corner=True,
    diag_kind='hist',
    diag_kws=dict(
        element='step',
        fill=True,
        stat="density",
        linewidth=1.0
    ),
    plot_kws=dict(size=20, alpha=0.3), # Tamaño y transparencia de los puntos de dispersión
    grid_kws=dict(diag_sharey=False) # Los histogramas no comparten eje Y
);
```


1e6
num_votes
0.0
0.5
1.0
1.5
2.0
2.5
1e9
revenue
popularity
0
100
200
300
400
500
600
2
4
6
8
avg_rating
0
1
2
num_votes
1e6
revenue
1e9
200
400
popularity

El descarte de los *outliers* permite ver ahora todas las gráficas con claridad. Parece que `num_votes`, y en mayor grado `revenue` guardan una ligera relación con `popularity`. Se aprecia que también que `avg_rating` se comporta de manera distinta, y que la relación con `popularity` es prácticamente nula. En cualquier caso, debido al gran número de puntos, y aunque se pueden intuir algunas tendencias, es difícil establecer hacer una valoración clara. En este caso, el coeficiente de correlación entre las variables puede ser proporcionar una visión más precisa.

Obtener el coeficiente de correlación entre los pares de variable. Repetir la operación para los datos con y sin *outliers* (en `popularity`) y comparar.

`DataFrame.corr()` devuelve, a su vez, un *DataFrame*. El método `DataFrame.style.background_gradient()` muestra el conjunto de datos resultante con colores que facilitan su interpretación.

Muestra la correlación incluyendo los *outliers*.

In [14]:
```
# El coeficiente de correlación varía entre 1 y -1.
# Un valor próximo a 1 indica una correlación positiva fuerte, es decir, que cuando aumenta el valor de una variable, también aumenta el valor de la otra.
# Un valor cercano a 0 indica una correlación débil o nula.
# Un valor próximo a -1 indica una correlación negativa fuerte, es decir, que cuando que aumenta el valor de una variable, disminuye el valor de la otra
```

In [15]:
```
# Selecciono las columnas de interés
df_subset = df_movies[success_meas]

# Calculo la matriz de correlación y muestro los coeficientes de correlación
corr_matrix = df_subset.corr()
#display(corr_matrix)

# Muestro el conjunto de datos resultante con colores que facilitan su interpretación
corr_matrix.style.background_gradient()
```

Out[15]:

| | avg_rating | num_votes | revenue | popularity |
|---|---|---|---|---|
| **avg_rating** | 1.000000 | 0.436970 | 0.063270 | 0.015133 |
| **num_votes** | 0.436970 | 1.000000 | 0.522712 | 0.174430 |
| **revenue** | 0.063270 | 0.522712 | 1.000000 | 0.370639 |
| **popularity** | 0.015133 | 0.174430 | 0.370639 | 1.000000 |

Muestra la correlación descartando los *outliers* en función del valor `pop_outlier_thresh`.

In [16]:
```
# Selecciono los datos con outliers en popularity
df_outliers = df_movies.loc[df_movies['popularity'] > pop_outlier_thresh]

# Calculo la correlación para los datos con outliers
corr_matrix_outliers = df_outliers[success_meas].corr()

# Selecciono los datos sin outliers en popularity
df_no_outliers = df_movies.loc[df_movies['popularity'] <= pop_outlier_thresh]

# Calculo la correlación para los datos sin outliers
corr_matrix_no_outliers = df_no_outliers[success_meas].corr()

# Muestro el conjunto de datos resultante con colores que facilitan su interpretación
corr_matrix_no_outliers.style.background_gradient()
```

Out[16]:

| | avg_rating | num_votes | revenue | popularity |
|---|---|---|---|---|
| **avg_rating** | 1.000000 | 0.436785 | 0.055288 | -0.004984 |
| **num_votes** | 0.436785 | 1.000000 | 0.523690 | 0.314606 |
| **revenue** | 0.055288 | 0.523690 | 1.000000 | 0.491084 |
| **popularity** | -0.004984 | 0.314606 | 0.491084 | 1.000000 |

Además, se considera que el presupuesto de las películas ( `budget` ) podría tener una incidencia definitiva en su difusión y, por tanto, en su éxito. Por ello se va a estudiar su relación con algunas de las variables anteriores.

Obtener la correlación de `budget` con las variables `avg_rating` , `num_votes` , `revenue` , y `popularity` .

A menos que se indique explícitamente, se trabajará con el conjunto de datos completos (sin eliminar *outliers*).

In [17]:
```
# Añado 'budget' a la lista success_meas
success_meas.append('budget')

# Selecciono las columnas de interés y calculo la matriz de correlación
df_corr_full = df_movies[success_meas].corr();

# Al ser una serie, lo paso a df
df_corr_full = df_corr_full['budget'].to_frame().sort_values(by="budget", ascending=False)

# Muestro la columna de correlación correspondiente a budget
df_corr_full.style.background_gradient()
```

Out[17]:

| | budget |
|---|---|
| **budget** | 1.000000 |
| **revenue** | 0.706334 |
| **popularity** | 0.318876 |
| **num_votes** | 0.312072 |
| **avg_rating** | -0.146861 |

Efectivamente, el presupuesto tiene una correlación relativamente alta con los ingresos, a pesar de no tenerla (incluso es ligeramente negativa) con las valoraciones. Esto refleja la importancia de la promoción. También que las valoraciones tienden a ser más bajas (muy ligeramente) para películas de alto presupuesto (¿el público exige más?).

☑

Los datos obtenidos muestran algunos factores de interés:

- Cuando se consideran todas las películas, la correlación más alta se da entre el número de votos ( `num_votes` ) y los ingresos de la película ( `revenue` ), lo cual es esperable: cuanta más gente ve una película, más gente tiende a votarla. La correlación entre la valoración de la película ( `avg_rating` ) y los ingresos ( `revenue` ) es nula. Esto hace pensar que la gente va a ver las películas por la promoción y la espectativa que levantan, independientemente de las críticas o las valoraciones.
- Si se eliminan los *outliers* de `popularity` , la relación entre esta variable y las demás cambia ligeramente. En concreto la popularidad ( `popularity` ) parece relacionada de manera algo más fuerte con los ingresos ( `revenue` ). Esto bien podría deberse a que la popularidad de las novedades es mayor, mientras que la recaudación es menor por haber estado menos tiempo en taquilla. Al descartarlas, este fenómeno no afecta tanto.
- La valoración ( `avg_rating` ) tiene cierta relación positiva con el número de votos ( `num_votes` ). Parece que la gente tiende a votar para expresar una valoración positiva.
- El presupuesto ( `budget` ) está muy relacionado con el beneficio ( `revenue` ).

Llegado a este punto, y considerando el objetivo inicial del estudio, nos centraremos principalmente en la variable `revenue` para medir la repercusión de la película (es proporcional a la gente que la ha visto), aunque también se estudiarán en algunos casos la valoración y número de votos.

# 2. Presupuesto

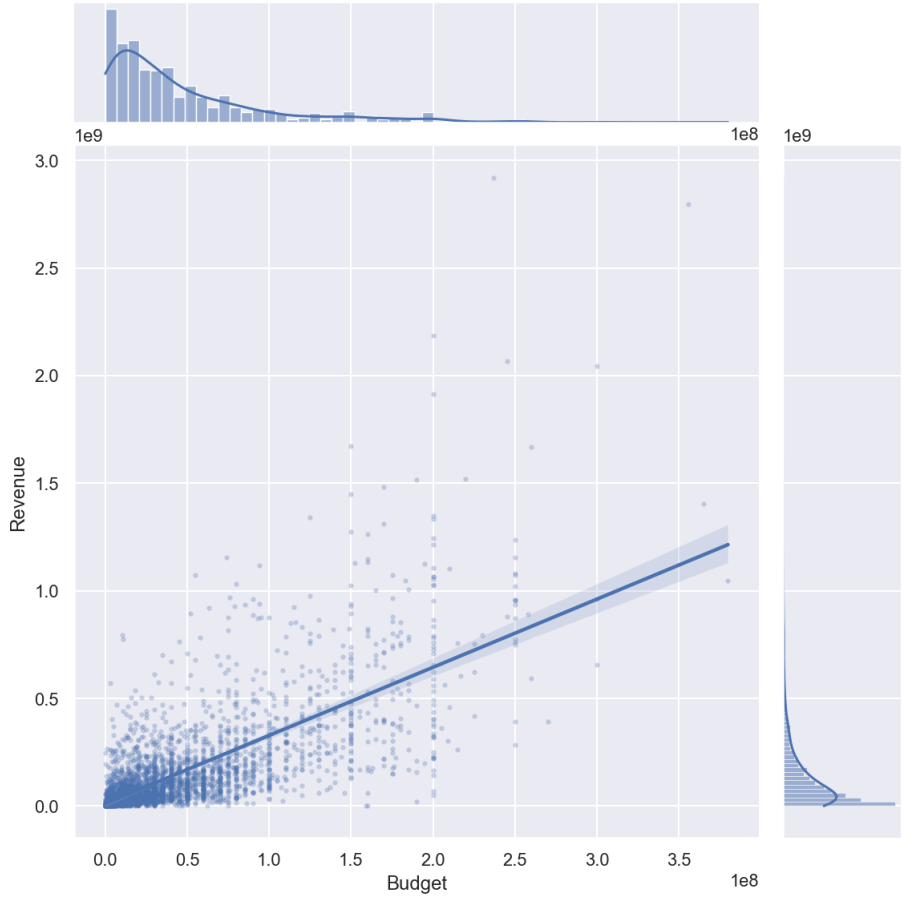
## 2.1 Presupuesto / ingresos

Los datos anteriores mostraban que la correlación entre el presupuesto ( `budget` ) y los ingresos ( `revenue` ) es positiva. Es posible estudiar con más en detalle esta relación. Un recurso interesante en este sentido es un gráfico de tipo `sns.joinplot()` , que dibuja una gráfica de dispersión de las dos variables, la distribución marginal de cada una de ellas y, mediante el parámetro `kind='reg'` puede añadir la recta correspondiente a un modelo de regresión.

**Ejercicio 3**

Construir y analizar una gráfica `sns.joinplot()` con una recta de regresión entre las variables `budget` y `revenue` .

In [18]:
```
# Construyo la gráfica
sns.jointplot(x='budget', y='revenue', data=df_movies, kind='reg', scatter_kws={'s': 5, 'alpha': 0.2}, height=8);

# Modifico los nombres de los ejes
plt.xlabel('Budget')
plt.ylabel('Revenue')
plt.show()
```




Se aprecia cierta relación lineal entre `budget` y `revenue`. Sin embargo, la gráfica no permite determinar si esta relación es significativa.

Estimar un modelo de regresión lineal con `statsmodels` y mostrar los resultados. ¿Qué valor tiene el $R^2$? ¿Qué significa? ¿Se puede afirmar que existe una relación entre el presupuesto y el beneficio? ¿Por qué?

In [19]:
```
#!pip install statsmodels
```

In [20]:
```
import statsmodels.api as sm

# Selecciono las columnas de interés
X = df_movies['budget']
y = df_movies['revenue']

# Añado una constante a la matriz X
X = sm.add_constant(X)

# Creo el modelo de regresión lineal
model = sm.OLS(y, X, missing='drop')

# Estimo el modelo
results = model.fit()

# Muestro los resultados
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.499
Model:                            OLS   Adj. R-squared:                  0.499
Method:                 Least Squares   F-statistic:                     3357.
Date:                Wed, 18 Jan 2023   Prob (F-statistic):               0.00
Time:                        16:44:27   Log-Likelihood:                -68539.
No. Observations:                3374   AIC:                         1.371e+05
Df Residuals:                    3372   BIC:                         1.371e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        1.12e+07   3.78e+06      2.963      0.003    3.79e+06    1.86e+07
budget         3.1668      0.055     57.942      0.000       3.060       3.274
==============================================================================
Omnibus:                     2421.900   Durbin-Watson:                   1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            84694.399
Skew:                           2.990   Prob(JB):                         0.00
Kurtosis:                      26.805   Cond. No.                     9.44e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.44e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Comentar el resultado (contestar a las preguntas) en esta celda .

¿Qué valor tiene el $R2$? Tiene un valor de 0.499

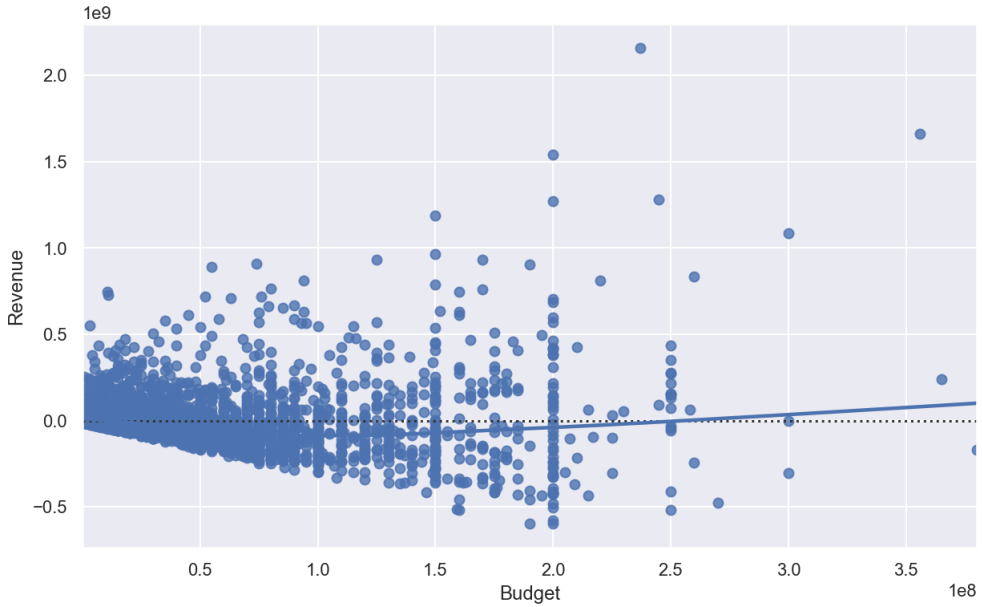
¿Qué significa? En general, un valor de R2 cercano a 1 indica una relación fuerte y positiva entre las variables, mientras que un valor cercano a 0 indica una relación débil o nula. En este caso, un valor de R2 de 0.499 significa que existe una correlación media entre el presupuesto y el beneficio. Sin embargo, el valor de R2 es una medida aproximada de la relación entre las variables y no debe ser utilizado como un indicador absoluto de la calidad del modelo.

¿Se puede afirmar que existe una relación entre el presupuesto y el beneficio? ¿Por qué? Existe cierta relación entre el presupuesto y el beneficio, pero no necesariamente una relación fuerte o directa.

Aunque el modelo establece que existe una relación lineal, se puede apreciar que está condicionada por los valores extremos de `revenue`. Además parece que la varianza en esta variable aumenta cuando lo hace `budget`.

Comprobar esta circunstancia con una gráfica de tipo `sns.residplot()`.

In [21]:
```
# Creo el gráfico
fig = plt.figure(figsize=(10,6))
ax = sns.residplot(x='budget', y='revenue', data=df_movies, lowess=True)
ax.set(xlabel='Budget', ylabel='Revenue')

# Muestro el gráfico
plt.show()
```



Efectivamente, la varianza de `revenue` aumenta conforme aumenta `budget`, es decir, no se cumple la condición de homocedasticidad, por lo que los intervalos de confianza y p-valores asociados al modelo de regresión ***no tienen validez estadística*** aunque sí permiten intuir la relación entre las variables.

Por otra parte, llama la atención que existen varios puntos que rompen claramente las tendencias, ya que existen varias películas que obtuvieron unos ingresos significativamente por encima de las demás, junto con algún fracaso que, pese al alto presupuesto, no generó los ingresos esperados.

Obtener la información (título, presupuesto y fecha de estreno) relativa a las cinco películas con más presupuesto.

In [22]:
```
# Obtengo la información de las cinco películas con más presupuesto
top_5_movies = df_movies.loc[df_movies['budget'].nlargest(5).index]

# Muestro la información de las películas
top_5_movies[['title', 'budget', 'release_date']]
```

Out[22]:

| movie_id | title | budget | release_date |
|---|---|---|---|
| **tt1298650** | Pirates of the Caribbean: On Stranger Tides | 380000000.0 | 2011-05-14 |
| **tt2395427** | Avengers: Age of Ultron | 365000000.0 | 2015-04-22 |
| **tt4154796** | Avengers: Endgame | 356000000.0 | 2019-04-24 |
| **tt0449088** | Pirates of the Caribbean: At World's End | 300000000.0 | 2007-05-19 |
| **tt0974015** | Justice League | 300000000.0 | 2017-11-15 |

Obtener la información (título, ingresos y fecha de estreno) relativa a las diez películas con más ingresos.

In [23]:
```
# Obtengo la información de las cinco películas con más presupuesto
top_5_movies = df_movies.loc[df_movies['revenue'].nlargest(10).index]

# Muestro la información de las películas
top_5_movies[['title', 'revenue', 'release_date']]
```

Out[23]:

| movie_id | title | revenue | release_date |
|---|---|---|---|
| **tt0499549** | Avatar | 2920357254 | 2009-12-15 |
| **tt4154796** | Avengers: Endgame | 2797800564 | 2019-04-24 |
| **tt0120338** | Titanic | 2187463944 | 1997-11-18 |
| **tt2488496** | Star Wars: Episode VII - The Force Awakens | 2068223624 | 2015-12-15 |
| **tt4154756** | Avengers: Infinity War | 2046239637 | 2018-04-25 |
| **tt10872600** | Spider-Man: No Way Home | 1916050941 | 2021-12-15 |
| **tt0369610** | Jurassic World | 1671713208 | 2015-06-06 |
| **tt6105098** | The Lion King | 1667635327 | 2019-07-12 |
| **tt0848228** | The Avengers | 1518815515 | 2012-04-25 |
| **tt2820852** | Furious 7 | 1515047671 | 2015-04-01 |

Hay algunas películas entre las cinco con mayor presupuesto que no aparecen entre las diez con más ingresos.

## 2.2 La fecha de estreno como factor de interés

Aunque la correlación entre presupuesto y los ingresos corresponde con lo que cabe esperarse (más presupuesto implica también más publicidad), hay que considerar que ambos factores dependen fuertemente del año de estreno, ya que se están manejando datos que corresponden a épocas distintas. Para tener una idea de la importancia de este factor, es necesario visualizar como evolucionan ambas cantidades a lo largo de los años.

### Ejercicio 4

Agrupar las películas en función del año de estreno, y calcular la media de presupuesto e ingresos para cada grupo. Para ello, se utilizará la función `DataFrame.groupby()` , y se almacenará el resultado (la media para `budget` y para `revenue` de cada año) en un *DataFrame* que se denominará `df_bud_rev_year` .

In [24]:
```
# Selecciono la columna release_date y extraigo el año
release_year = df_movies['release_date'].dt.year

# Realizo la agrupación utilizando el año como criterio
df_bud_rev_year = df_movies.dropna().groupby(release_year)

# Calculo la media de presupuesto e ingresos para cada grupo
df_bud_rev_year = df_bud_rev_year[['budget', 'revenue']].mean()
df_bud_rev_year
```

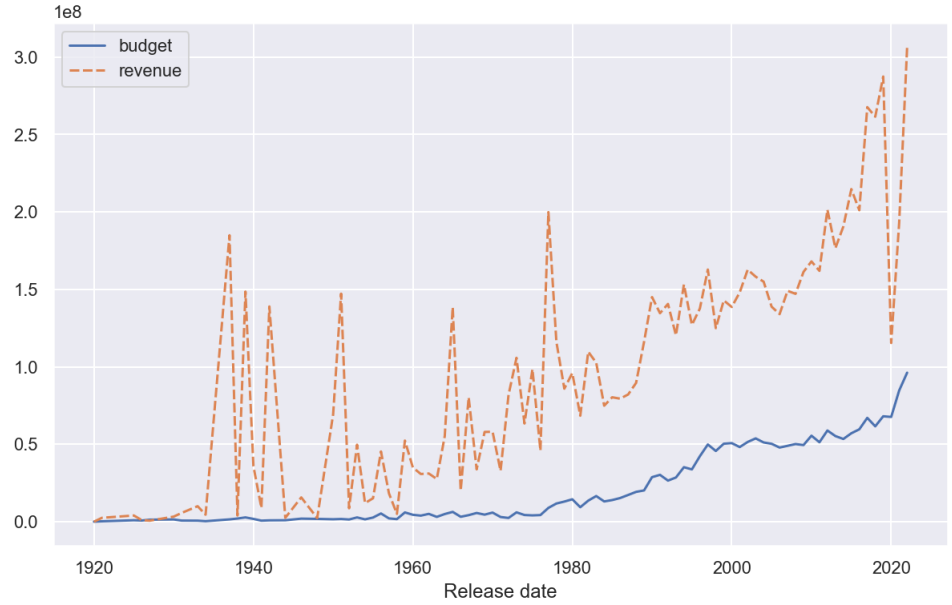
Out[24]:

| release_date | budget | revenue |
|---|---|---|
| **1920** | 1.800000e+04 | 8.811000e+03 |
| **1921** | 2.500000e+05 | 2.500000e+06 |
| **1925** | 9.230000e+05 | 4.000000e+06 |
| **1926** | 7.500000e+05 | 1.000000e+06 |
| **1927** | 1.300000e+06 | 6.504220e+05 |
| **...** | ... | ... |
| **2018** | 6.157286e+07 | 2.614606e+08 |
| **2019** | 6.804592e+07 | 2.874146e+08 |
| **2020** | 6.763889e+07 | 1.153658e+08 |
| **2021** | 8.468246e+07 | 1.930110e+08 |
| **2022** | 9.620772e+07 | 3.066032e+08 |

91 rows × 2 columns

Dibujar la gráfica a partir de la información (dos columnas) que contiene el `DataFrame` `df_bud_rev_year` .

In [25]:
```
# Creo un gráfico de líneas
plt.figure(figsize=(10,6))
ax = sns.lineplot(data=df_bud_rev_year)
ax.set(xlabel='Release date')

# Muestro el gráfico
plt.show()
```




En la gráfica se pueden visualizar varios fenómenos:

- Tanto los presupuestos como los beneficios crecen cada año.
- La línea de beneficios es más irregular que la de presupuesto.
- Los beneficios parecen crecer más que los presupuestos, sobre todo en los últimos años.
- Los beneficios cayeron de forma dramática el primer año de la pandemia.
- Se aprecian seis anomalías importantes anteriores a 1980, en las que se aprecia que los ingresos medios se separan ampliamente de la tendencia.

Obtener los años a los que corresponden estas entradas. Para ello, pueden obtenerse los 6 mayores valores anteriores a 1980, en el *DataFrame* `df_bud_rev_year` .

In [26]:
```
# Obtengo los años con los 6 mayores valores de revenue
top_revenue =  df_bud_rev_year.loc[df_bud_rev_year.index < 1980].nlargest(6, 'revenue').sort_values(by=['release_date'])
top_revenue
```

Out[26]:

| release_date | budget | revenue |
|---|---|---|
| **1937** | 1488423.0 | 1.849255e+08 |
| **1939** | 2759000.0 | 1.485692e+08 |
| **1942** | 868000.0 | 1.389548e+08 |
| **1951** | 1748750.0 | 1.472125e+08 |
| **1965** | 6325000.0 | 1.385671e+08 |
| **1977** | 8900000.0 | 1.998660e+08 |

En 1937, 1939, 1942, 1951, 1965 y 1977 se producen las alteraciones más importantes de la serie.

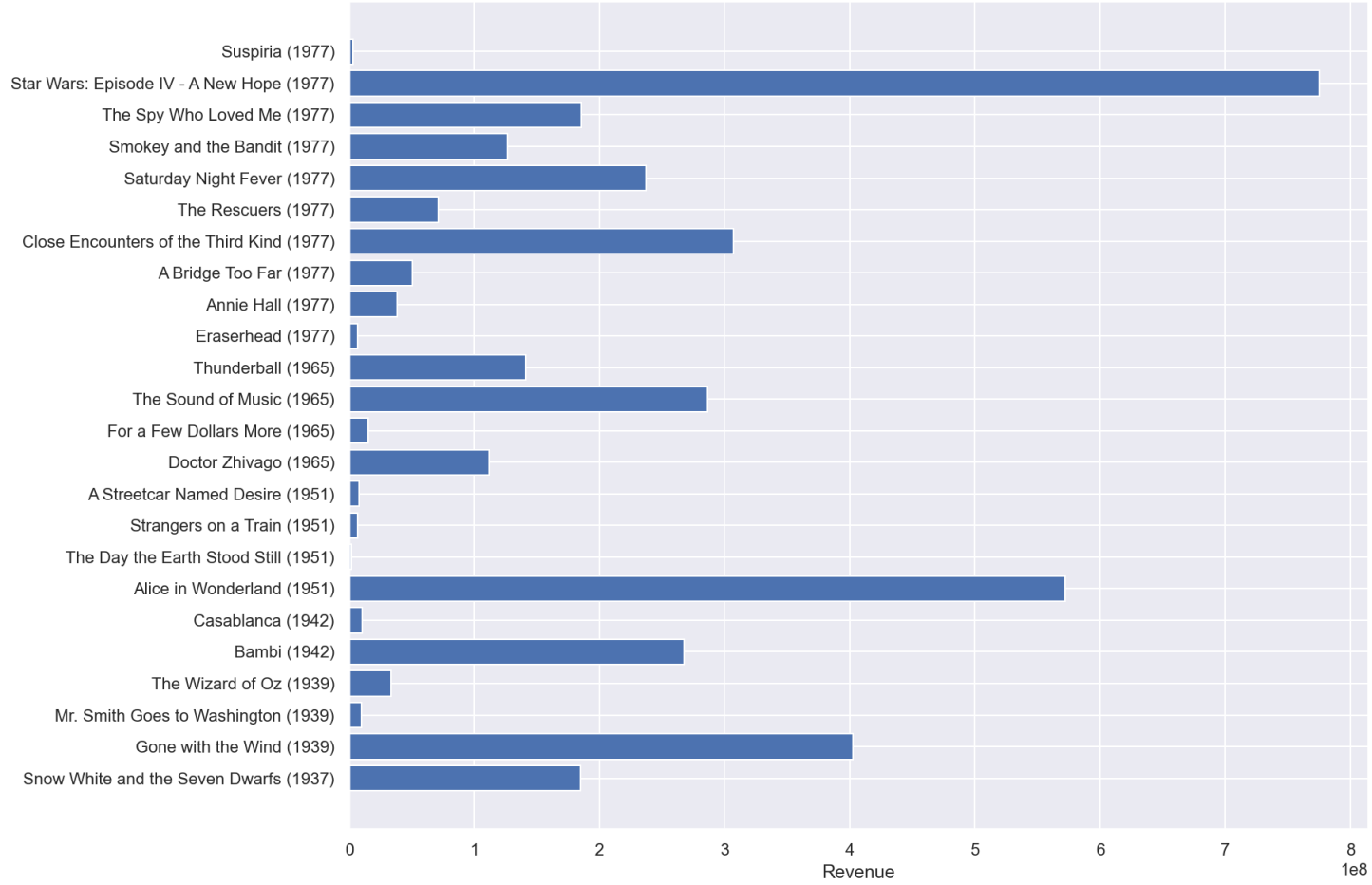
Extraer los datos de las películas correspondientes a estos años y almacenarlos en un *DataFrame* denominado `df_anomalies` . Dibujar un gráfico de barras horizontal con los ingresos de cada película. Las etiquetas en el eje `y` deben contener el título de la películas y, entre paréntesis, el año de estreno.

In [27]:
```
# Obtengo los 6 años con mayor valor en la columna revenue en el DataFrame df_bud_rev_year
highest_revenue_years = top_revenue.index

# Selecciono solo las películas que fueron estrenadas en esos años
df_anomalies = df_movies[(df_movies['release_date'].dt.year).isin(highest_revenue_years)]
df_anomalies.head()
```

Out[27]:

| movie_id | title | genres | avg_rating | num_votes | budget | revenue | release_date | original_language | popularity | keywords |
|---|---|---|---|---|---|---|---|---|---|---|
| **tt0029583** | Snow White and the Seven Dwarfs | Adventure,Animation,Family | 7.6 | 200177 | 1488423.0 | 184925486 | 1937-12-21 | en | 150.919 | poison,dwarf,witch,becoming an adult,sadness,q... |
| **tt0031381** | Gone with the Wind | Drama,Romance,War | 8.2 | 315675 | 4000000.0 | 402352579 | 1939-12-15 | en | 30.056 | southern usa,loss of loved one,based on novel ... |
| **tt0031679** | Mr. Smith Goes to Washington | Comedy,Drama | 8.1 | 116086 | 1500000.0 | 9600000 | 1939-10-19 | en | 14.335 | washington dc, usa,senate,senator,sightseeing,... |
| **tt0032138** | The Wizard of Oz | Adventure,Family,Fantasy | 8.1 | 399394 | 2777000.0 | 33754967 | 1939-08-15 | en | 113.698 | witch,adolescence,based on novel or book,secre... |
| **tt0034492** | Bambi | Adventure,Animation,Drama | 7.3 | 143873 | 858000.0 | 267447150 | 1942-02-11 | en | 107.562 | loss of loved one,forest,deer,coming of age,be... |

In [28]:
```
# Obtengo los valores de ingresos de cada película
revenues = df_anomalies['revenue']

# Obtengo los títulos de las películas y los años de estreno
movie_titles = df_anomalies['title'] + ' (' + (df_anomalies['release_date'].dt.year).astype(str) + ')'

# Dibujo el gráfico de barras horizontal
plt.figure(figsize=(12, 10))
plt.barh(movie_titles, revenues)
plt.xlabel('Revenue')
plt.show()
```



Puede apreciarse que estas anomalías se deben a películas concretas como "*Gone with the wind*" ("Lo que el viento se llevó"), "*Alice in Wonderland*" ("Alicia en el país de las maravillas") y, sobre todo, "*Star Wars*". También hay otras como "*Close Encounters with the Third Kind"* ("Encuentros en la tercera fase"), o algunas otras de animación como "Bambi" y "*Snow White and the Seven Dwarfs*" ("Blancanieves y los siete enanitos").

### Conclusión

Como resumen hasta ahora puede establecerse que, efectivamente, el presupuesto está muy relacionado con los ingresos. Con respecto a la valoración, sin embargo, se había visto en la sección anterior que no parece que haya una relación relevante, lo que constituye el hallazgo que, a priori, podría resultar contrario a nuestra intuición.

Además, hay una variable *latente* que condiciona los ingresos: la fecha de estreno. Para estudiar la repercusión de otros factores se hace necesario tener en cuenta esta circunstancia. Existen varias posibilidades.

- Corregir los ingresos y beneficio en función del año, eliminando el efecto de la tendencia.
- Estudiar distintos periodos de tiempo por separado.
- Hacer el estudio para franjas de prespuesto específicas.
- Calcular el porcentaje o ratio de ingresos con respecto al presupuesto.

Debido esto, y al objetivo del estudio (producción de una película de presupuesto relativamente bajo), se considerarán películas más actuales. Además, desde el año 2000 en adelante se observa que el crecimiento del presupuesto es algo más lento, por lo que se ha decidido centrar el estudio en películas estrenadas a partir de esa fecha. Además, se trabajará también con el ratio recaudación/presupuesto.

> ℹ Cuando se trabaja con datos temporales, se suele eliminar la tendencia o la estacionalidad. En este proyecto no se trabajará a ese nivel, ya que estos conceptos se tratarán en el módulo VII.

In [29]:
```
df_movies = df_movies[df_movies['release_date']>='2000']
```

## 3. Géneros

En la base de datos original, `df_movies`, los géneros de cada película aparecen como una lista separada por comas en la columna `genres`. Para poder estudiar la repercusión de este factor, es necesario llevar a cabo una **transformación** que permita operar de manera ágil. Dado que el número de géneros es limitado y reducido, es posible utilizar variables identificadoras (*dummy*). *Otra posibilidad* consiste en crear un conjunto de datos auxiliar, `df_genres`, en formato *long form*, en el que cada fila corresponda a un par película-género, y hacer las uniones correspondientes con `df_movies`.

ℹ Podéis ir ejecutando cada sentencia por separado y ver el resultado para entender mejor el proceso.

In [30]:
```
# Creo las variables dummy para indicar qué género aparece en cada película.
df_genres = df_movies['genres'].str.get_dummies(',').astype(bool)

# Almaceno los nombres de los  géneros en una lista.
genres = df_genres.columns.to_list()

# Convierto la matriz a formato "long form"
df_genres = (df_genres.reset_index()
                      .melt(id_vars='movie_id', value_vars=genres, var_name='genre', value_name='ind')
                      .set_index('movie_id'))
# Filtro
df_genres = df_genres[df_genres['ind']==True]

# Borro la columna indicador
del df_genres['ind']

# Muestro las entradas correspondientes a la película "Avatar" (Adventure, Action, Fantasy)
df_genres.loc['tt0499549']
```

Out[30]:

| movie_id | genre |
|---|---|
| **tt0499549** | Action |
| **tt0499549** | Adventure |
| **tt0499549** | Fantasy |

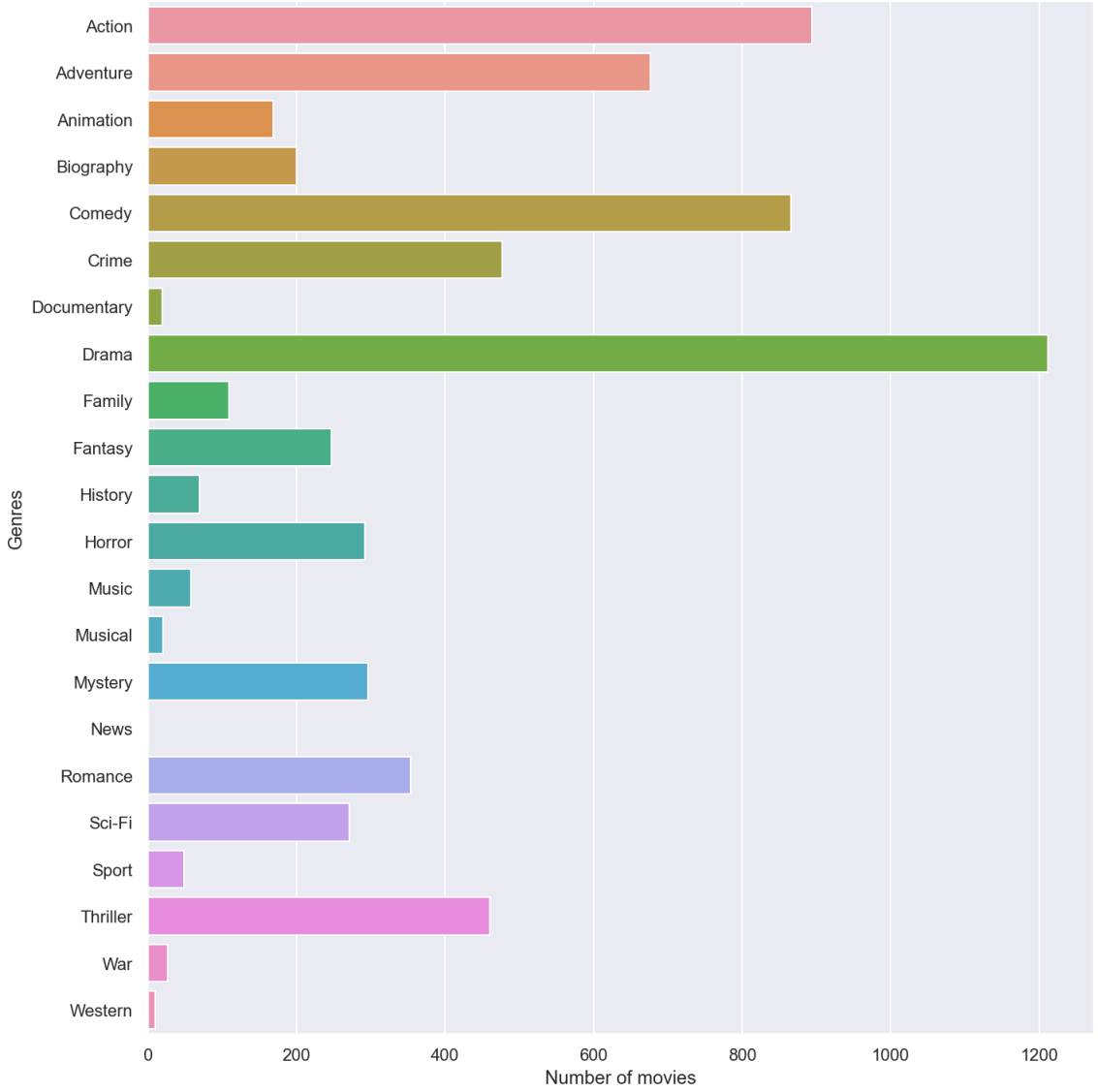
Además, se descartarán las columnas de `df_movies` que no se utilizarán.

In [31]:
```
# Descarto las columnas
df_movies = df_movies[['title','avg_rating', 'num_votes', 'budget', 'revenue', 'popularity', 'release_date']]
```

### Ejercicio 5

A partir del *DataFrame* `df_genres` , es posible visualizar con qué frecuencia aparecen los géneros. Dibujar un gráfico de barras que muestre el número de entradas correspondientes a cada valor de `genre` en el *DataFrame* `df_genres` .

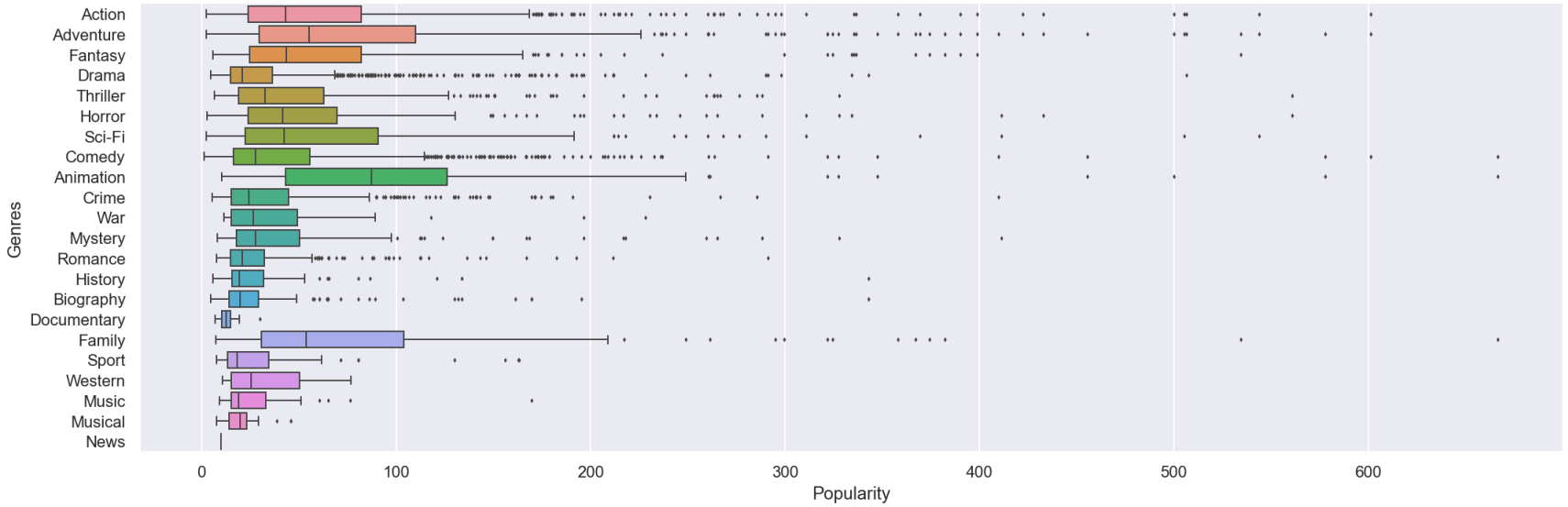
In [32]:
```
# Dibujo un gráfico para visualizar la cantidad de películas que tiene cada género en df_genres
ax = sns.catplot(data=df_genres, y='genre', kind='count', height=10);
ax.set(xlabel='Number of movies', ylabel='Genres')
plt.show();
```




Parece que el drama es, de largo, el género que tratan más películas. Le siguien la acción y la comedia.

Mostrar la distribución de popularidad por género con un gráfico de cajas (horizontal). Esto requiere utilizar información de `df_genres` (género) y de `df_movies` (popularidad).

In [33]:
```
# Selecciono entradas de df_movies que no son outliers
df_movies_filtered = df_movies.loc[df_movies['popularity'] < pop_outlier_thresh]

# Fusiono df_genres con df_movies_filtered
df_merged = df_genres.merge(df_movies_filtered, on='movie_id', how='outer')

# Dibujo un gráfico de cajas horizontal con los datos de df_merged
ax = sns.catplot(data=df_merged, y='genre', x='popularity', kind='box', linewidth=1, fliersize=1, aspect=3);
ax.set(xlabel='Popularity', ylabel='Genres');
plt.show()
```




### Ejercicio 6

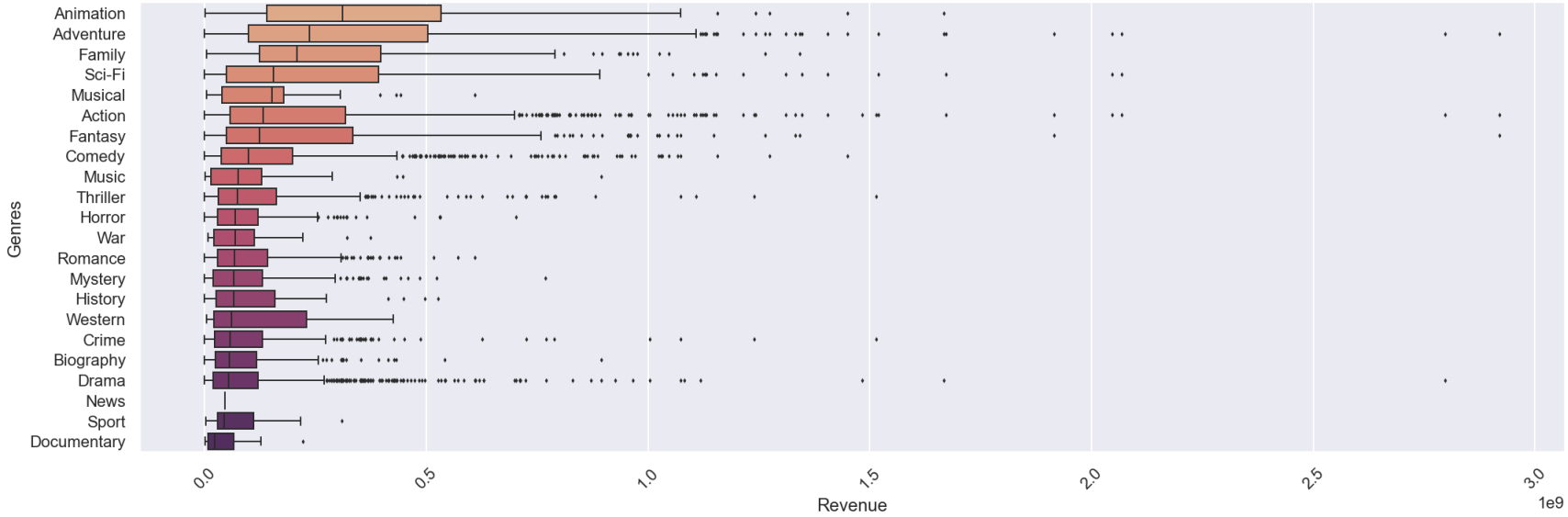
Para explorar la relación entre presupuestos e ingresos en función del género, debido a que las distribuciones están sesgadas, también se utilizarán gráficas del tipo `sns.boxplot()`. Como paso previo, y para facilitar la exploración, se pueden ordenar los géneros en función de la ***mediana*** de los ingresos obtenidos ( `revenue` ) por las películas correspondientes. Este orden se almacenará en la lista `ordered_genres`.

In [34]:
```
# Fusiono df_genres con la columna revenue de df_movies
df_merged = pd.merge(df_genres, df_movies[['revenue']], left_index=True, right_index=True)

# Agrupo df_merged según la columna genre y obtengo la mediana de la columna revenue
ordered_genres = df_merged.groupby('genre')['revenue'].median().sort_values(ascending=False)
ordered_genres
```

Out[34]:
```
genre
Animation      311594032.0
Adventure      237382724.0
Family         209113983.5
Sci-Fi         155010032.0
Musical        152000000.0
Action         132537482.5
Fantasy        124700000.0
Comedy          98268127.5
Music           75759862.0
Thriller        74549911.0
Horror          69493547.0
War             68729358.0
Romance         68279729.5
Mystery         66604309.5
History         65616808.0
Western         61693168.0
Crime           57777106.0
Biography       55168294.0
Drama           54594779.0
News            46243000.0
Sport           44907260.0
Documentary     23164353.0
Name: revenue, dtype: float64
```

Dibujar un gráfico de cajas y bigotes con el beneficio por género, utilizando el orden entre géneros almacenado en `ordered_genres` .

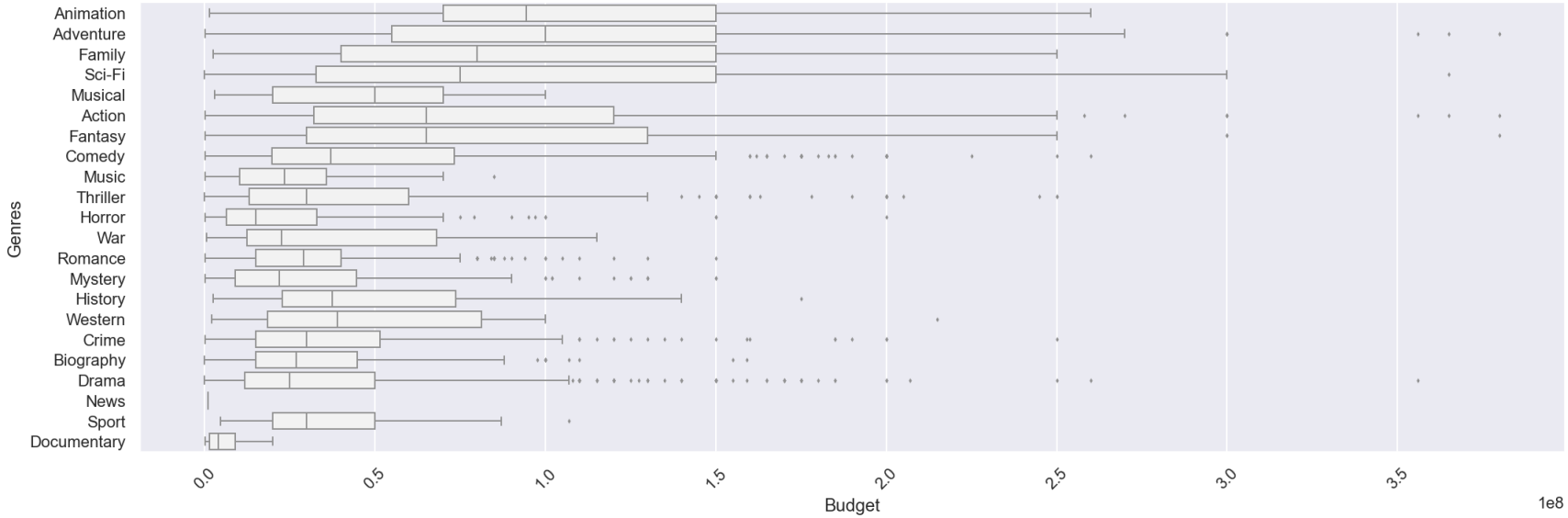
In [35]:
```
# Fusiono df_genres con la columna revenue de df_movies
df_merged2 = pd.merge(df_genres, df_movies[['revenue']], left_index=True, right_index=True)

# Dibujo un gráfico de cajas horizontal con los datos de df_merged2
ax = sns.catplot(data=df_merged2, x='revenue', y='genre', kind='box', order=ordered_genres.index, palette='flare', linewidth=1, fliersize=1, aspect=3);
ax.set(xlabel='Revenue', ylabel='Genres');
plt.xticks(rotation=45);
```




Puede apreciarse que las películas de animación, aventuras, familia y ciencia ficción, son los géneros que más recaudan. Los que menos, las noticias, los deportes y los documentales. Es curioso, aunque puede estar relacionado, que el drama es el género más frecuente y es uno de los que menos recauda.

Anteriormente se comprobó la importancia del presupuesto en la recaudación. Por tanto, es interesante también estudiar si la relación de las recaudaciones por género se refleja también en los presupuestos. Elaborar una gráfica similar a la anterior en la que se muestre el presupuesto por géneros. Para comparar, puede ser de utilidad mantener el mismo orden en la representación utilizado anteriormente (la mediana de los ingresos).

In [36]:
```
# Fusiono df_genres con la columna revenue de df_movies
df_merged3 = pd.merge(df_genres, df_movies[['budget']], left_index=True, right_index=True)

# Dibujo un gráfico de cajas horizontal con los datos de df_merged2
ax = sns.catplot(data=df_merged3, x='budget', y='genre', kind='box', order=ordered_genres.index, color='.95', linewidth=1, fliersize=1, aspect=3);
ax.set(xlabel='Budget', ylabel='Genres');
plt.xticks(rotation=45);
```




Se aprecian algunos cambios menores. No obstante, la tendencia general, es que los géneros que producen más ingresos son también los que más presupuesto reciben.

## 4. Personas

Una vez estudiados los géneros, va a estudiarse si el director y el reparto tienen relación con la rentabilidad de las películas. El *DataFrame* `df_credits` contiene la relación de personas que participan en cada película y en calidad de qué. Por otra parte, el *DataFrame* `df_people` contiene la información relativa a cada persona, incluyendo su nombre.

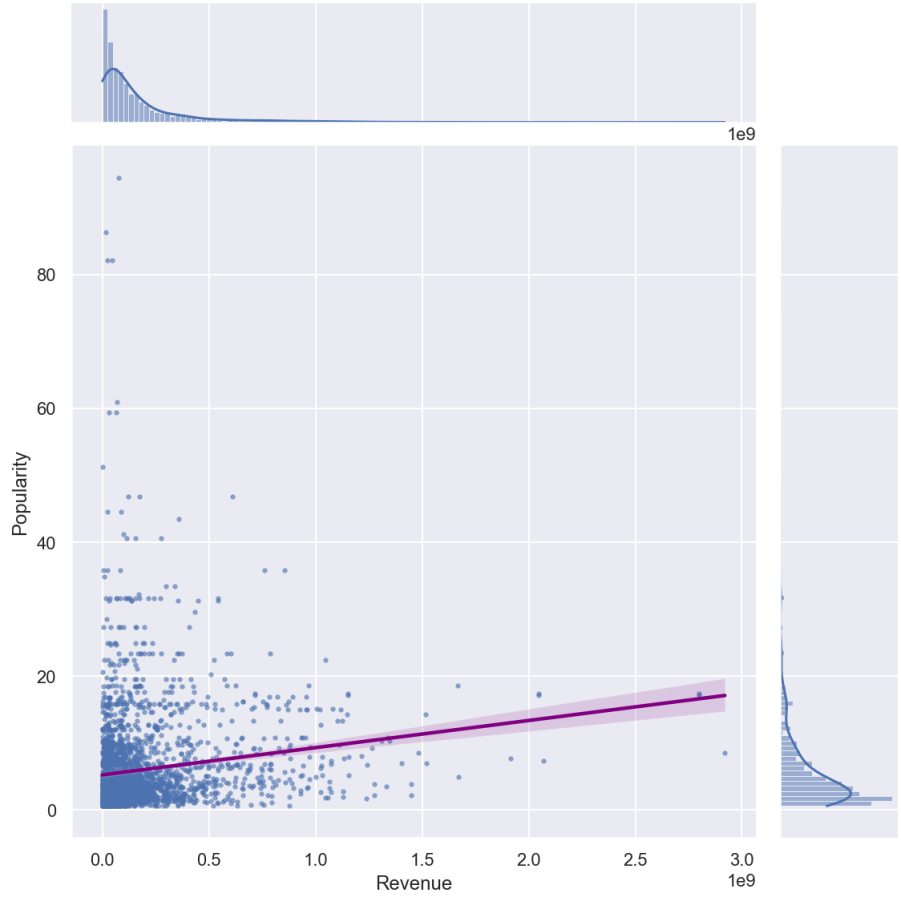
### 4.1 Directores

Interesa obtener información sobre los ingresos que producen los directores, sobre su popularidad y la relación de ambos factores.

### Ejercicio 7

Elaborar una gráfica `sns.jointplot()` para comprobar si existe correlación entre los ingresos de las películas y la popularidad de su director. Esto implica utilizar datos de `df_credits`, de `df_people[popularity]` y de `df_movies['revenue']`) por lo que hay que hacer dos fusiones. El resultado de estas se debe almacenar en un *DataFrame* denominado `df_dir_pop_rev`.

In [37]:
```
# Hago una selección condicional sobre el DataFrame df_credits para seleccionar las filas en las que la columna rol sea director.
df_director = df_credits.query('rol == "director"')

# Fusiono el DataFrame resultante con la columna popularity de df_people utilizando la columna people_id
# con how='inner'se ignoran las filas que contengan valores NaN
df_director_people = pd.merge(df_director, df_people[['popularity']], left_on='people_id', right_index=True, how="inner")

# Fusiono el DataFrame resultante con la columna revenue de df_movies utilizando la columna movie_id
# con how='inner'se ignoran las filas que contengan valores NaN
df_dir_pop_rev = pd.merge(df_director_people, df_movies[['revenue']], left_on='movie_id', right_index=True, how='inner')

# Creo la gráfica de dispersión con un histograma en cada eje
sns.jointplot(x='revenue', y='popularity', data=df_dir_pop_rev, kind='reg', line_kws={'color':'purple'}, scatter_kws={'s':5, 'alpha':0.5}, height=8);

# Modifico los nombres de los ejes
plt.xlabel('Revenue')
plt.ylabel('Popularity')
plt.show()
```




Puede apreciarse que la relación entre popularidad del director y el beneficio es casi nula.

In [38]:
```
df_dir_pop_rev[['popularity','revenue']].corr()
```

Out[38]:

| | popularity | revenue |
|---|---|---|
| **popularity** | 1.0000 | 0.1455 |
| **revenue** | 0.1455 | 1.0000 |

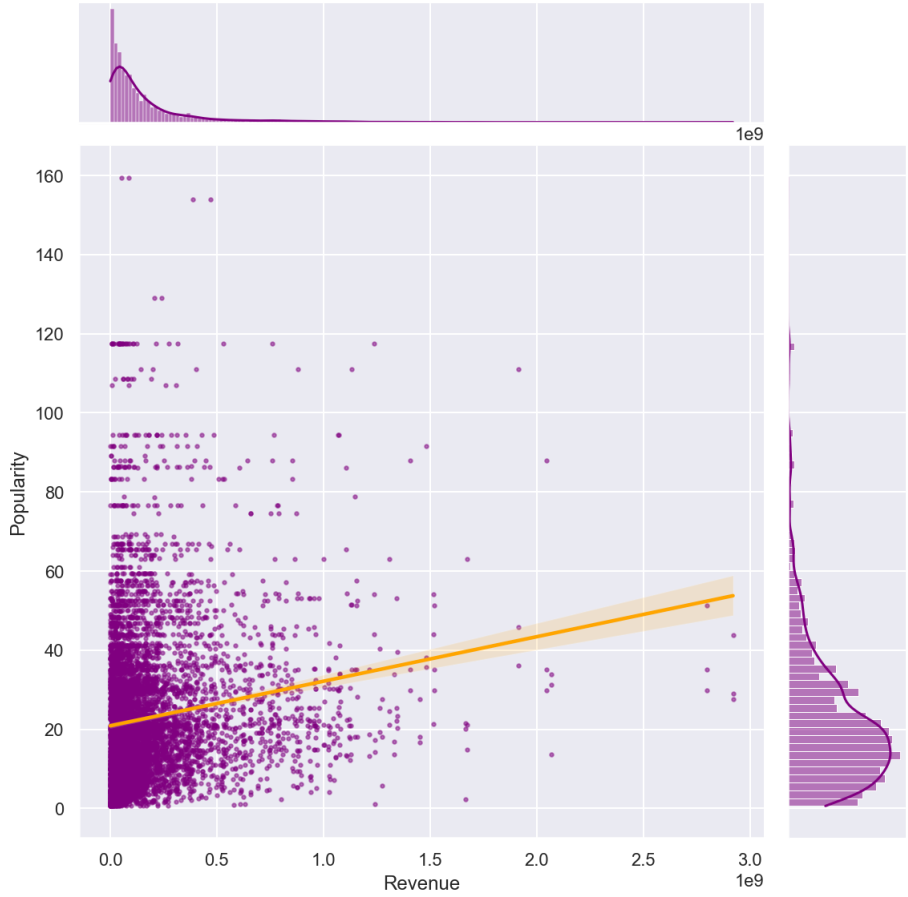
## 4.2 Reparto

Una vez estudiado el papel del director, se procederá de igual manera con el reparto.

### Ejercicio 8

Elaborar una gráfica `sns.jointplot()` para comprobar si existe correlación entre los ingresos de las películas y la popularidad de su reparto. El resultado de estas se debe almacenar en un *DataFrame* denominado `df_cast_pop_rev` .

In [39]:
```
# Hago una selección condicional sobre el DataFrame df_credits para seleccionar las filas en las que la columna rol sea director.
df_cast = df_credits.query('rol == "cast"')

# Fusiono el DataFrame resultante con la columna popularity de df_people utilizando la columna people_id
# con how='inner'se ignoran las filas que contengan valores NaN
df_cast_people = pd.merge(df_cast, df_people[['popularity']], left_on='people_id', right_index=True, how="inner")

# Fusiono el DataFrame resultante con la columna revenue de df_movies utilizando la columna movie_id
# con how='inner'se ignoran las filas que contengan valores NaN
df_cast_pop_rev = pd.merge(df_cast_people, df_movies[['revenue']], left_on='movie_id', right_index=True, how='inner')

# Creo la gráfica de dispersión con un histograma en cada eje
sns.jointplot(x="revenue", y="popularity", data=df_cast_pop_rev, kind='reg', line_kws={'color':'orange'}, scatter_kws={'s':5, 'alpha':0.5}, height=8, color='purp

# Modifico los nombres de los ejes
plt.xlabel('Revenue')
plt.ylabel('Popularity')
plt.show()
```




In [40]:
```
df_cast_pop_rev[['popularity','revenue']].corr()
```

Out[40]:

| | popularity | revenue |
|---|---|---|
| **popularity** | 1.000000 | 0.151211 |
| **revenue** | 0.151211 | 1.000000 |

Se aprecia también que, aunque influye positivamente, no es determinante porque casi es nula.

En definitiva, y según se aprecia en las gráficas, el principal factor en relación a la recaudación es el presupesto, mientras que otros como el director o los actores sean irrelevantes (salvo en casos muy específicos, se intuye).

## 5. Películas de bajo presupuesto

En el objetivo de estudio que se ha establecido inicialmente, se parte de la base de que se dispone de un presupuesto limitado, por lo que se va a estudiar también la relación entre género e ingresos en este supuesto, considerando solamente aquellas películas cuyo presupuesto se encuentra en el 20% más bajo.

A continuación se almacenan las películas en las que se centrará el estudio (posteriores al año 2000 y de bajo presupuesto) en `df_movies_low` .

In [41]:
```
low_budget = df_movies['budget'].quantile(0.20)
df_movies_low = df_movies[df_movies['budget']<low_budget]
len(df_movies_low)
```

Out[41]:
```
472
```

Aunque es algo que se puede gestionar "sobre la marcha", retener solamente las películas de interés en `df_genres` facilita las cosas. En este caso se seleccionan y se almacenan en `df_genres_low` .

In [42]:
```
df_genres_low = df_genres.loc[df_movies_low.index]
df_genres_low
```

Out[42]:

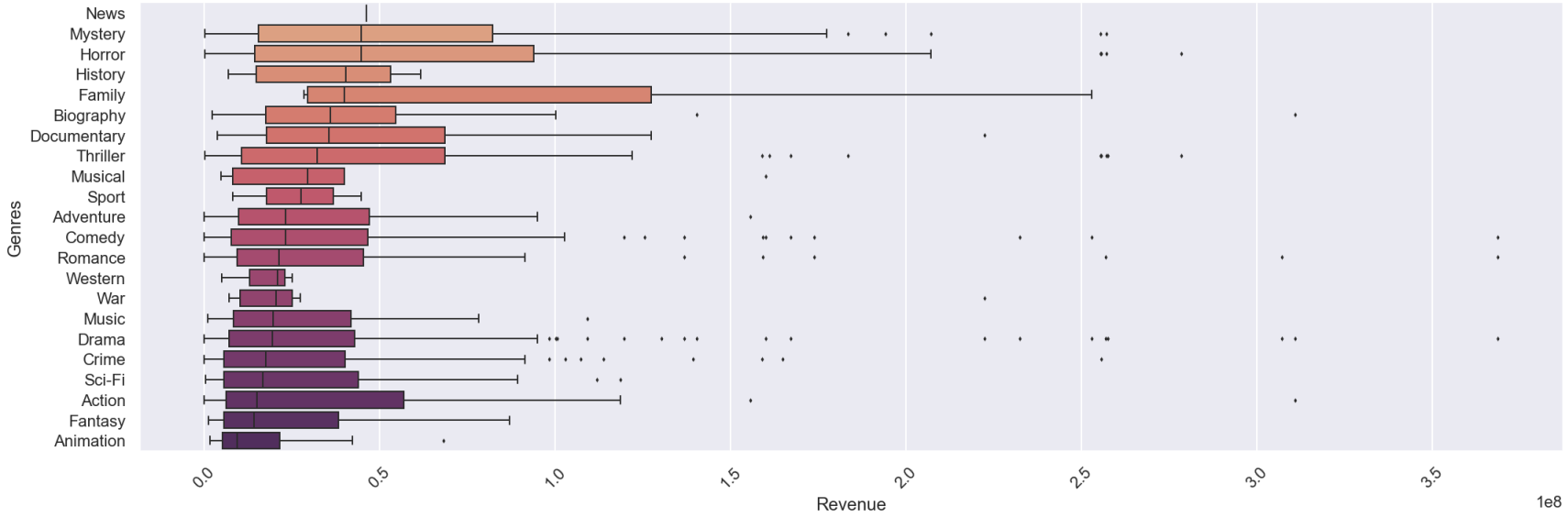
| movie_id | genre |
|---|---|
| **tt0118694** | Drama |
| **tt0118694** | Romance |
| **tt0120679** | Biography |
| **tt0120679** | Drama |
| **tt0120679** | Romance |
| **...** | ... |
| **tt8772262** | Mystery |
| **tt9052870** | Comedy |
| **tt9052870** | Drama |
| **tt9052870** | Romance |
| **tt9770150** | Drama |

1177 rows × 1 columns

### Ejercicio 9

Recalcular el orden de los géneros en función de la mediana los ingresos y construir una gráfica de cajas con los datos almacenados en el *DataFrame* `df_movies_low` .

In [43]:
```
# Fusiono df_genres con la columna revenue de df_movies
df_merged = pd.merge(df_genres_low, df_movies_low[['revenue']], left_index=True, right_index=True)

# Agrupo df_merged según la columna genre y obtengo la mediana de la columna revenue
ordered_genres = df_merged.groupby('genre')['revenue'].median().sort_values(ascending=False)

# Dibujo un gráfico de cajas horizontal con los datos de df_merged2
ax = sns.catplot(data=df_merged, x='revenue', y='genre', kind='box', order=ordered_genres.index, palette='flare', linewidth=1, fliersize=1, aspect=3);
ax.set(xlabel='Revenue', ylabel='Genres');
plt.xticks(rotation=45);
```




Puede apreciarse un cambio claro de tendencia. Cuando los prespuestos son relativamente bajos, géneros como el misterio y el terror pasan a ser más rentables. También algún otro género como las noticias o historia. Llama también la atención que, géneros como la animación o fantasía, que son los más rentables en el caso general, no son rentables cuando los presupuestos son bajos.

Como los géneros más rentables son el terror, y el misterio (si se descartan las noticias, que solo hay una película), se seleccionarán las películas correspondientes a estos, además del *thriller* (esta última por estar relacionada y mostrar también algunos valores altos) para estudiarlas en más profundidad.

Eliminar de `df_movies_low` las películas que no correspondan a los géneros `Horror`, `Thriller` o `Mystery`.

In [44]:
```
# Selecciono las películas de interés en df_genres_low utilizando isin()
movies_of_interest = df_genres_low['genre'].isin(['Horror', 'Thriller', 'Mystery'])

# Obtengo los índices de las películas de interés en df_genres_low
indices_of_interest = movies_of_interest[movies_of_interest].index

# Accedo las filas de df_movies_low correspondientes a las películas de interés
df_movies_low = df_movies_low.loc[indices_of_interest].drop_duplicates()
df_movies_low
```

Out[44]:

| movie_id | title | avg_rating | num_votes | budget | revenue | popularity | release_date |
|---|---|---|---|---|---|---|---|
| **tt0144084** | American Psycho | 7.6 | 611514 | 7000000.0 | 34266564 | 85.933 | 2000-04-13 |
| **tt0181984** | Boiler Room | 7.0 | 53663 | 7000000.0 | 28780255 | 17.792 | 2000-02-18 |
| **tt0209144** | Memento | 8.4 | 1232381 | 9000000.0 | 39723096 | 32.570 | 2000-10-11 |
| **tt0211443** | Jason X | 4.4 | 57313 | 11000000.0 | 16951798 | 47.684 | 2001-11-09 |
| **tt0219699** | The Gift | 6.7 | 69992 | 10000000.0 | 12008642 | 31.863 | 2000-03-16 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **tt8155288** | Happy Death Day 2U | 6.2 | 75209 | 9000000.0 | 25327500 | 35.025 | 2019-02-13 |
| **tt8359848** | Climax | 6.9 | 69075 | 2940457.0 | 1088907 | 17.750 | 2018-09-19 |
| **tt8368406** | Vivarium | 5.8 | 59740 | 3947600.0 | 427399 | 38.603 | 2019-09-07 |
| **tt8663516** | Child's Play | 5.7 | 52524 | 10000000.0 | 44902237 | 52.309 | 2019-06-19 |
| **tt8772262** | Midsommar | 7.1 | 318587 | 9000000.0 | 47980982 | 50.433 | 2019-07-03 |

206 rows × 7 columns

Una vez reducido el conjunto de películas a las que resultan de interés para el contexto planteado, se obtendrán la lista de directores y actores ligados a una mayor recaudación.

### Ejercicio 10

Crear un *DataFrame* denominado `best_directors_low` con los datos ( `people_id`, `revenue`, `name`, `popularity` y `gender` ) de los diez directores para los que la recaudación media de sus películas es mayor.

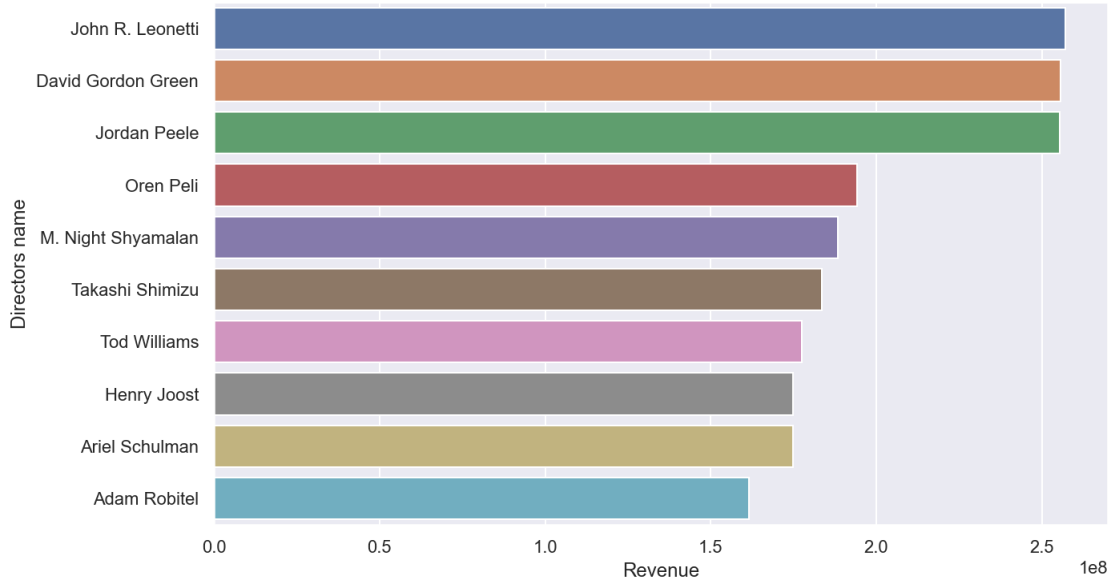
In [45]:
```
# Filtro los créditos correspondientes a directores en df_credits
df_directors = df_credits.loc[df_credits['rol'].isin(['director'])]

# Fusiono el df anterior y df_movies_low utilizando la columna movie_id como clave
df_directors_movies = df_directors.merge(df_movies_low, on="movie_id", how='inner')

# Agrupo el df anterior según el people_id y obtener el valor medio de revenue. Convierto la serie en un df
df_directors_revenue = pd.DataFrame(df_directors_movies.groupby('people_id')['revenue'].mean())

# Fusiono el df anterior y df_people utilizando la columna people_id como clave
df_directors_movies_names = df_directors_revenue.merge(df_people, on="people_id", how='inner')

# Obtengo las 10 filas con mayor valor de revenue en mean_revenue
best_directors_low = df_directors_movies_names[['revenue', 'name', 'popularity', 'gender']].nlargest(10, "revenue")
best_directors_low
```

Out[45]:

| people_id | revenue | name | popularity | gender |
|---|---|---|---|---|
| **26714** | 257047661.0 | John R. Leonetti | 3.716 | 2 |
| **64141** | 255614941.0 | David Gordon Green | 12.051 | 2 |
| **291263** | 255407969.0 | Jordan Peele | 12.006 | 2 |
| **90591** | 194183034.0 | Oren Peli | 1.177 | 2 |
| **11614** | 188452210.0 | M. Night Shyamalan | 12.368 | 2 |
| **20310** | 183474602.0 | Takashi Shimizu | 3.617 | 2 |
| **20133** | 177512032.0 | Tod Williams | 1.658 | 2 |
| **142272** | 174928918.0 | Henry Joost | 1.522 | 2 |
| **142276** | 174928918.0 | Ariel Schulman | 3.156 | 2 |
| **1306608** | 161448094.5 | Adam Robitel | 3.209 | 2 |

Crear una gráfica de barras (horizontal) con los valores `revenue` para cada director en `best_directors_low` .

In [46]:
```
data = best_directors_low[['name', 'revenue']]

# Creo la gráfica de barras
fig = plt.figure(figsize=(10,6));
ax = sns.barplot(x=data['revenue'], y=data['name'], orient='h');
ax.set(xlabel='Revenue', ylabel='Directors name')
plt.show()
```




A continuación se muestra la película del director que ha obtenido más beneficios que, según parece, también es protagonista de la película.

In [47]:
```
# people_id del director
best_director_id = best_directors_low.iloc[0].name

# Créditos de las películas de ese director y que estén en df_movies_low
movies_bd = (df_credits[(df_credits['people_id']==best_director_id) &
                        (df_credits['movie_id'].isin(df_movies_low.index))]
                        ['movie_id'])

# Películas
df_movies.loc[movies_bd]
```

Out[47]:

| movie_id | title | avg_rating | num_votes | budget | revenue | popularity | release_date |
|---|---|---|---|---|---|---|---|
| **tt3322940** | Annabelle | 5.4 | 160448 | 6500000.0 | 257047661 | 85.045 | 2014-10-02 |

Podemos mostrar el cartel de la película que ha catapultado al éxito (al menos económico) a este director.

In [48]:
```
from IPython.display import Image

Image('http://image.tmdb.org/t/p/w185//8ouiyGYcv3E1JPrPrC7pLjxlDhC.jpg')
```

Out[48]:


Un modo quizá más justo de calcular la rentabilidad de una película es el ratio `revenue` / `budget` (para obtener este dato, es necesario considerar solamente las películas con `revenue>0` y `budget>0` , pero en primer apartado se eliminaron las que no cumplían la condición). Obtener los 10 directores que han generado un mayor beneficio según este criterio.

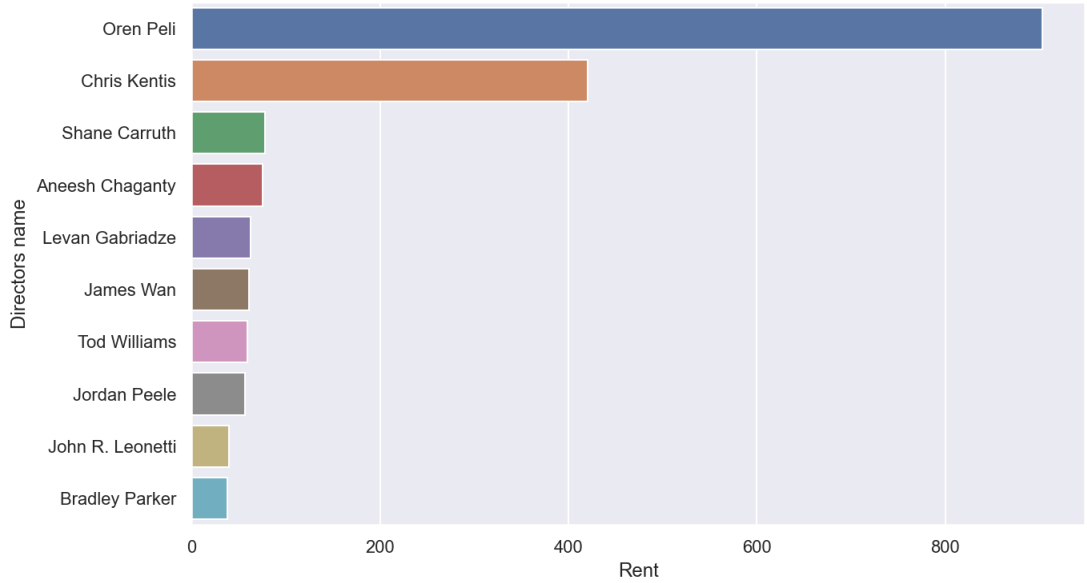
Mostrar una gráfica, similar a la anterior, con esta información. En lugar de utilizar `revenue` , hay que calcular primero una nueva columna, denominada `rent` con la beneficio/presupuesto, y mostrar los datos corrspondientes a esta nueva columna.

In [49]:
```
# Filtro los datos para obtener solo películas con revenue y budget > 0 y creo la nueva columna 'rent' con el ratio entre revenue y budget
df_movies_low=df_movies_low.query("revenue > 0 and budget > 0").assign(rent=df_movies_low["revenue"] / df_movies_low["budget"])

# Fusiono la columna rent del df anterior y df_directors utilizando la columna movie_id como clave
df_directors_rent = df_directors.merge(df_movies_low['rent'], on='movie_id', how='inner')

# Fusiono la columna revenue del df_movies_low y df_directors utilizando la columna movie_id como clave
df_directors_rev = df_directors.merge(df_movies_low['revenue'], on='movie_id', how='inner')

# Elimino los NaN, agrupo por directores, calculo la media de rent, creo una nueva tabla, ordeno por rent y selecciono los 10 valores superiores
df_directors_rent_grouped = df_directors_rent.dropna().groupby(df_directors_rev['people_id'])['rent'].mean().to_frame() .nlargest(10 , 'rent')

# Fusiono el df anterior y df_people utilizando la columna people_id como clave
best_directors_low = df_directors_rent_grouped.merge(df_people, on="people_id", how='inner')
display(best_directors_low)
```

| people_id | rent | name | popularity | imdb_id | gender |
|---|---|---|---|---|---|
| **90591** | 903.176902 | Oren Peli | 1.177 | nm2305431 | 2 |
| **640** | 420.522723 | Chris Kentis | 1.400 | nm0448900 | 2 |
| **76624** | 77.919429 | Shane Carruth | 0.840 | nm1503403 | 2 |
| **1932296** | 75.462037 | Aneesh Chaganty | 0.828 | nm3792134 | 2 |
| **86688** | 62.882090 | Levan Gabriadze | 1.400 | nm0300174 | 2 |
| **2127** | 61.155275 | James Wan | 14.339 | nm1490123 | 2 |
| **20133** | 59.170677 | Tod Williams | 1.658 | nm0931095 | 2 |
| **291263** | 56.757326 | Jordan Peele | 12.006 | nm1443502 | 2 |
| **26714** | 39.545794 | John R. Leonetti | 3.716 | nm0502954 | 2 |
| **999812** | 38.390020 | Bradley Parker | 1.663 | nm0662086 | 2 |

Crear una gráfica de barras (horizontal) con los valores `rent` para cada director en `best_directors_low` .

In [50]:
```
# Creo la gráfica de barras
fig = plt.figure(figsize=(10,6));
ax = sns.barplot(x=best_directors_low.rent, y=best_directors_low.name, orient='h');
ax.set(xlabel='Rent', ylabel='Directors name')
plt.show()
```

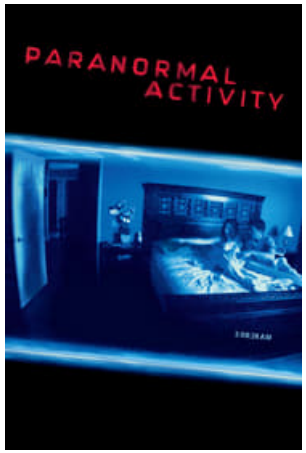



En este sentido, hay un director y película que destaca sobre muy por encima de los demás.

In [51]:
```
# people_id del director
best_director_id = best_directors_low.iloc[0].name
# Créditos de las películas de ese director y que estén en df_movies_low
movies_bd = (df_credits[(df_credits['people_id']==best_director_id) &
                        (df_credits['movie_id'].isin(df_movies_low.index))]
                        ['movie_id'])
# Películas
display(df_movies.loc[movies_bd])

Image('http://image.tmdb.org/t/p/w185/1bjA7de4O0NhMsaOqwvrecophUs.jpg')
```

| movie_id | title | avg_rating | num_votes | budget | revenue | popularity | release_date |
|---|---|---|---|---|---|---|---|
| **tt1179904** | Paranormal Activity | 6.3 | 241286 | 215000.0 | 194183034 | 89.234 | 2007-09-14 |

Out[51]:




Una vez estudiados los directores, procederemos a estudiar el reparto. En este caso, se comenzará qué actores o actrices participan en películas con más ingresos.

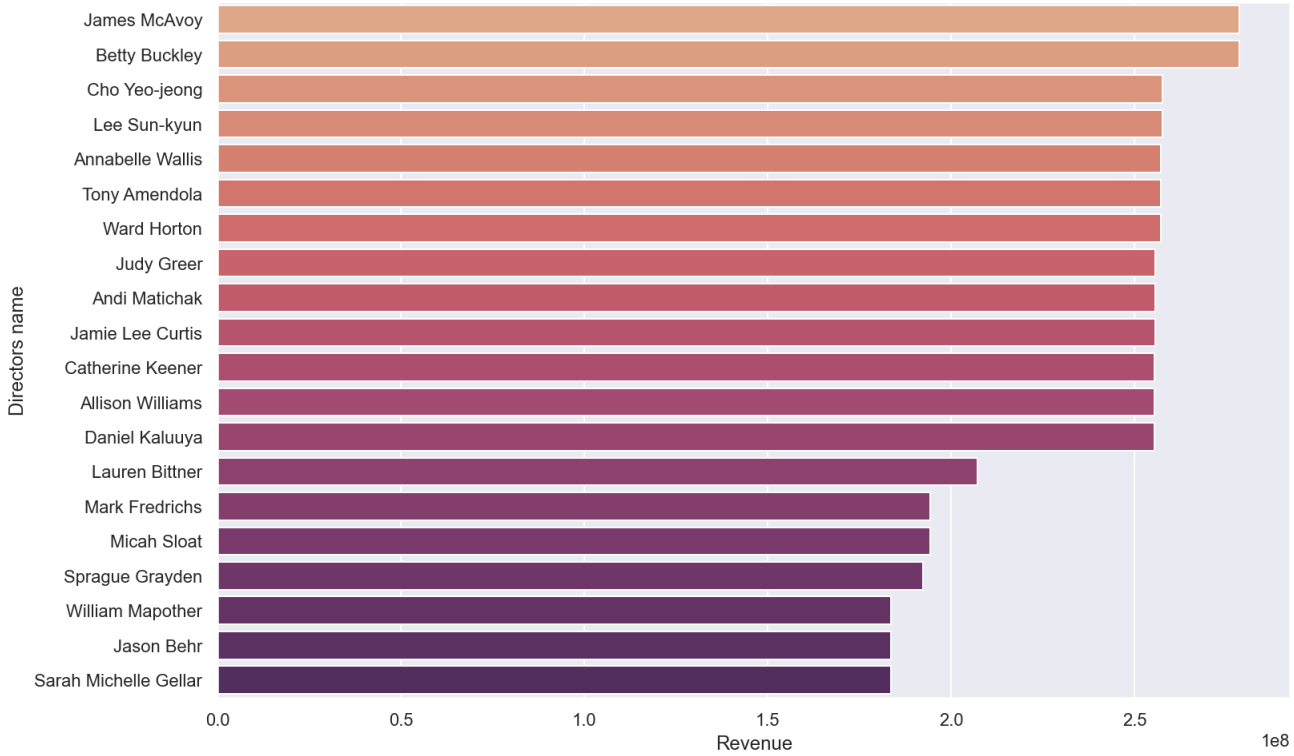
Obtener los nombres de los 20 actores/actrices cuyas películas generan, por término medio, una mayor cantidad de ingresos. Almacenar el resultado en un *DataFrame* denominado `best_cast_low` .

In [52]:
```
# Fusiono df_cast y la columna 'revenue' de df_movies_low, utilizando la columna 'movie_id' como clave
df_cast_rev = df_cast.merge(df_movies_low['revenue'], on='movie_id', how='inner')

# Agrupo el df anterior según el 'people_id' y obtener el valor medio de revenue. Convierto la serie en un df
df_cast_rev_mean = df_cast_rev.dropna().groupby(df_cast_rev['people_id'])['revenue'].mean().to_frame()


# Fusiono el df anterior y df_people utilizando la columna 'people_id' como clave
best_cast_low = df_cast_rev_mean.merge(df_people, on='people_id', how='inner')

# Obtengo las 20 filas con mayor valor de revenue
best_cast_low = best_cast_low.sort_values(by='revenue', ascending=False).head(20)

display(best_cast_low)
```

| people_id | revenue | name | popularity | imdb_id | gender |
|---|---|---|---|---|---|
| **5530** | 278454358.0 | James McAvoy | 36.130 | nm0564215 | 2 |
| **52462** | 278454358.0 | Betty Buckley | 4.241 | nm0000990 | 1 |
| **556435** | 257591776.0 | Cho Yeo-jeong | 14.593 | nm1856097 | 1 |
| **115290** | 257591776.0 | Lee Sun-kyun | 7.180 | nm1310525 | 2 |
| **82809** | 257047661.0 | Annabelle Wallis | 34.983 | nm1834115 | 1 |
| **34842** | 257047661.0 | Tony Amendola | 5.333 | nm0024593 | 2 |
| **94436** | 257047661.0 | Ward Horton | 5.232 | nm1782235 | 2 |
| **20750** | 255614941.0 | Judy Greer | 29.080 | nm0339460 | 1 |
| **1511997** | 255614941.0 | Andi Matichak | 12.880 | nm5506858 | 1 |
| **8944** | 255614941.0 | Jamie Lee Curtis | 21.423 | nm0000130 | 1 |
| **2229** | 255407969.0 | Catherine Keener | 16.190 | nm0001416 | 1 |
| **1255540** | 255407969.0 | Allison Williams | 10.818 | nm4129745 | 1 |
| **206919** | 255407969.0 | Daniel Kaluuya | 14.460 | nm2257207 | 2 |
| **176748** | 207039844.0 | Lauren Bittner | 4.090 | nm1584283 | 1 |
| **90598** | 194183034.0 | Mark Fredrichs | 1.400 | nm2104166 | 2 |
| **90597** | 194183034.0 | Micah Sloat | 1.731 | nm2913790 | 2 |
| **118243** | 192275938.0 | Sprague Grayden | 7.440 | nm0337042 | 1 |
| **15338** | 183474602.0 | William Mapother | 10.038 | nm0544611 | 2 |
| **20386** | 183474602.0 | Jason Behr | 7.468 | nm0004736 | 2 |
| **11863** | 183474602.0 | Sarah Michelle Gellar | 13.653 | nm0001264 | 1 |

Crear una gráfica de barras (horizontal) con los valores `revenue` para cada actor en `best_cast_low` .

In [53]:
```
fig = plt.figure(figsize=(12, 8));
ax = sns.barplot(x=best_cast_low.revenue, y=best_cast_low.name, orient='h', palette='flare');
ax.set(xlabel='Revenue', ylabel='Directors name')
plt.show()
```




En ejercicio anterior se muestra un reparto más o menos equitativo entre actores y actrices para los 10 miembros del reparto que están ligados a mayor beneficio. Sin embargo, se estudiará algo más este factor (género).

### Ejercicio 11

Crear un *DataFrame* denominado `cast_low` , similar al anterior, con el beneficio medio ( `revenue` ) asociado a cada actor, su nombre ( `name` ), popularidad ( `popularity` ) y género ( `gender` ).

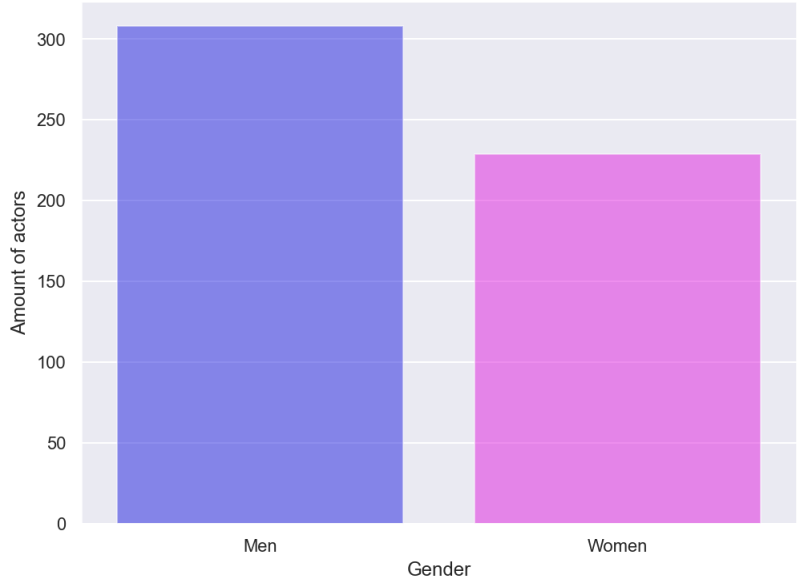
In [54]:
```
# Fusiono df_cast y la columna 'revenue' de df_movies_low, utilizando la columna 'movie_id' como clave
df_cast_rev = df_cast.merge(df_movies_low['revenue'], on='movie_id', how='inner')

# Agrupo el df anterior según el 'people_id' y obtener el valor medio de revenue. Convierto la serie en un df
df_cast_rev_mean = df_cast_rev.dropna().groupby(df_cast_rev['people_id'])['revenue'].mean().to_frame()

# Fusiono el df anterior y df_people utilizando la columna 'people_id' como clave
best_cast_low = df_cast_rev_mean.merge(df_people, on='people_id', how='inner')

# Ordeno las filas segun el valor de revenue
cast_low = best_cast_low.sort_values(by='revenue', ascending=False)

display(cast_low)
```

| people_id | revenue | name | popularity | imdb_id | gender |
|---|---|---|---|---|---|
| **5530** | 278454358.0 | James McAvoy | 36.130 | nm0564215 | 2 |
| **52462** | 278454358.0 | Betty Buckley | 4.241 | nm0000990 | 1 |
| **556435** | 257591776.0 | Cho Yeo-jeong | 14.593 | nm1856097 | 1 |
| **115290** | 257591776.0 | Lee Sun-kyun | 7.180 | nm1310525 | 2 |
| **82809** | 257047661.0 | Annabelle Wallis | 34.983 | nm1834115 | 1 |
| **...** | ... | ... | ... | ... | ... |
| **17287** | 209696.0 | Dominic West | 32.785 | nm0922035 | 2 |
| **11356** | 209696.0 | Imelda Staunton | 40.624 | nm0001767 | 1 |
| **122424** | 110197.0 | Hristos Passalis | 1.429 | nm3217334 | 2 |
| **122428** | 110197.0 | Christos Stergioglou | 2.695 | nm1176651 | 2 |
| **17587** | 110197.0 | Michele Valley | 1.691 | nm0885091 | 1 |

545 rows × 5 columns

A partir del *DataFrame* `cast_low` , obtener un gráfico de barras mostrando el número actores de cada género.

In [55]:
```
# Hay varios valores de género (0, 1, 2 y 3), pero me quedo con mujeres(1) y  hombres(2)
cast_low = cast_low[(cast_low['gender'] == 1) | (cast_low['gender'] == 2)]

# Asigno a los valores  1:'Woman' y 2:'Men'
cast_low['gender'] = cast_low['gender'].replace({1: 'Women', 2: 'Men'})

# Dibujo la gráfica
fig = plt.figure(figsize=(8, 6));
fig = sns.countplot(x='gender', data=cast_low, hue_order=['Women', 'Men'], palette=['blue', 'magenta'], alpha=0.5);
fig.set(xlabel='Gender', ylabel='Amount of actors');
```

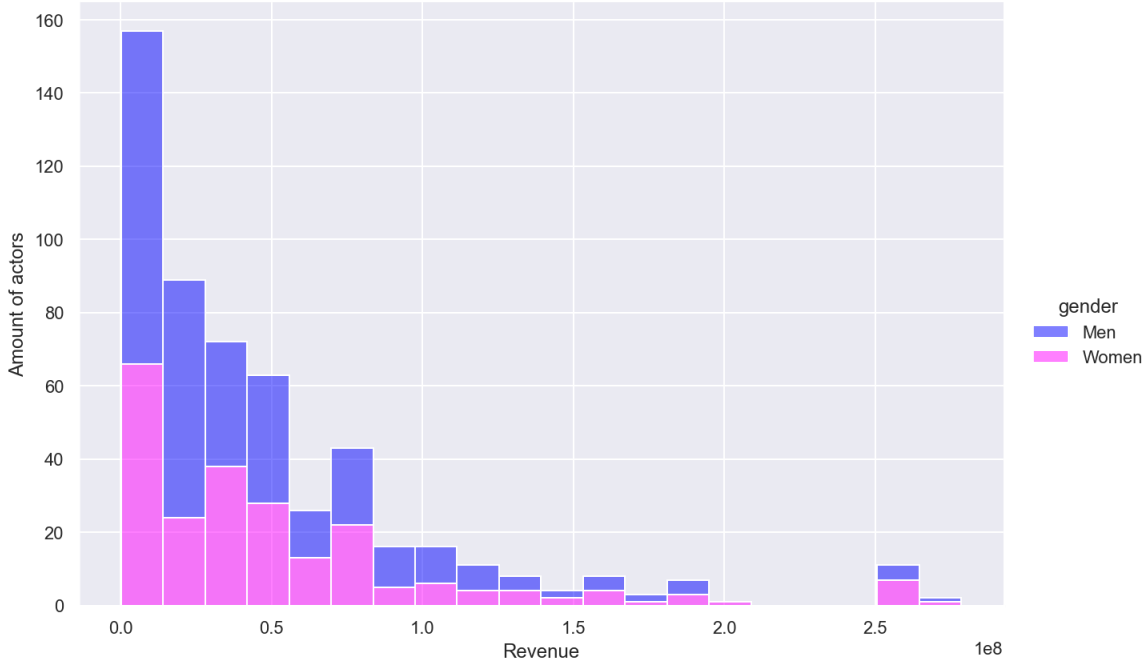



Se aprecia que el número de actores es mayor que el de actrices. Cabe preguntarse si hay algún tipo de diferencia en la recaudación vinculada al género de los actores.

### Ejercicio 12

Mostrar la distribución de `cast_low` para los géneros 1 (mujer) y 2 (hombre).

In [56]:
```
# Dibujo la distribución de cast_low para los géneros 1 (mujer) y 2 (hombre)
ax = sns.displot(data=cast_low, x='revenue', hue='gender', multiple='stack', palette=['blue', 'magenta'], alpha=0.5, height=6, aspect=1.5);
ax.set(xlabel='Revenue', ylabel='Amount of actors')
plt.show()
```




En principio se aprecia alguna diferencia en las distribuciones, pero no está claro que haya una diferencia en la recaudación media. Se puede verificar este hecho mediante un test U de Mann-Whitney. Si fijamos un nivel de significancia del 95% ¿puede afirmarse que son iguales los sueldos medios?

In [57]:
```
from scipy.stats import mannwhitneyu

stat, p = mannwhitneyu(
    x = cast_low[cast_low.gender == 'Men'].revenue,
    y = cast_low[cast_low.gender == 'Women'].revenue
)

print(f'stat={stat:.3f}, p={p:.3f}')
```

```
stat=33293.000, p=0.267
```

Comentar el resultado (contestar a las preguntas) en esta celda .

El valor p es mayor que el nivel de significancia fijado (p > 0.05), así que se considera que las diferencias entre los sueldos medios de hombres y mujeres no son significativas.

☑

Se percibe que, a pesar de que aparecen más actores que actrices, parece que en las películas con los beneficios más altos no hay diferencia o incluso aparecen más actrices que actores. El test refleja que en general no hay diferencias en las recaudaciones de las películas en que participan unos y otros.

## 6. Conclusión

En este trabajo se partía de dos bases de datos de películas y gente, y el objetivo era estudiar qué factores están más relacionados con el éxito de una película, que se mide por los ingresos que genera. Desde el principio, ha quedado claro que el presupuesto es el factor más importante, que los géneros en que más dinero se gasta en la producción son la animación, fantasía, etc., y que éstos son también los que más recaudación obtienen. También se ha comprobado que la información relativa a popularidad de directores y actores **no** parece un factor muy relevante.

No obstante se ha limitado el estudio al periodo actual y al (relativamente bajo) presupuesto disponible. En ese sentido, se ha comprobado que las películas más rentables son aquellas que pertenecen a los géneros de terror, suspense, etc. También se ha descubierto que películas como "Paranormal Activity" generan unos ingresos extraordinariamente altos en proporción con la inversión.