Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика, искусственный и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по Лабораторной работе №1

«*Разведочный анализ данных.*

Исследование и визуализация данных»

по дисциплине «Технология машинного обучения»

Выполнил:
студент группы ИУ5-61Б
И.А. Абуховский

Проверил:
Ю.Е. Гапанюк

2023 г.

Загрузим необходимые библиотеки для анализа и сам датасет.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import math as mth
import matplotlib.patches as patches
from scipy import stats as st
plt.rcParams.update({'figure.max_open_warning': 0})
import plotly.graph_objects as go
import plotly.express as px
```

In [2]:
```
df = pd.read_csv('GamesData.csv')
```

Обработка датасета

In [3]:
```
df.head()
```

Out[3]:

	App ID	Title	Reviews Total	Reviews Score Fancy	Release Date	Reviews D7	Reviews D30	Reviews D90	Launch Price	Tags	name_slug	
0	730	Counter-Strike: Global Offensive	6580452	86,71%	2012-08-21	453	987	1531	$14,99	FPS, Shooter, Multiplayer, Competitive, Action...	counter_strike_global_offensive	$
1	578080	PUBG: BATTLEGROUNDS	2062357	64,20%	2017-12-21	7274	18958	37059	$29,99	Survival, Shooter, Multiplayer, Battle Royale,...	pubg_battlegrounds	$
2	570	Dota 2	1791358	72,77%	2013-07-09	0	0	0	$29,99	Free to Play, MOBA, Multiplayer, Strategy, e-s...	dota_2	$
3	271590	Grand Theft Auto V	1262540	89,01%	2015-04-13	6912	12337	17615	$29,99	Open-World, Action, Multiplayer, Automobile Si...	grand_theft_auto_v	$
4	359550	Tom Clancy's Rainbow Six® Siege	903537	78,82%	2015-12-01	705	2051	4422	$59,99	FPS, Hero-Shooter, Multiplayer, Tactical, Shoo...	tom_clancys_rainbow_six_siege	$

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54046 entries, 0 to 54045
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   App ID               54046 non-null  int64 
 1   Title                54045 non-null  object
 2   Reviews Total        54046 non-null  int64 
 3   Reviews Score Fancy  54046 non-null  object
 4   Release Date         54046 non-null  object
 5   Reviews D7           54046 non-null  int64 
 6   Reviews D30          54046 non-null  int64 
 7   Reviews D90          54046 non-null  int64 
 8   Launch Price         54046 non-null  object
 9   Tags                 54046 non-null  object
dtypes: int64(5), object(5)
memory usage: 4.1+ MB
```

Пропуски

Явные пропуски

In [6]:
```
def draw_missing_data_table(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)*100
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    return missing_data
```

In [7]:
```
draw_missing_data_table(df)
```

Out[7]:

	Total	Percent
Title	1	0.00185
App ID	0	0.00000
Reviews Total	0	0.00000
Reviews Score Fancy	0	0.00000
Release Date	0	0.00000
Reviews D7	0	0.00000
Reviews D30	0	0.00000
Reviews D90	0	0.00000
Launch Price	0	0.00000
Tags	0	0.00000

Заметим, что явных пропусков нет, кроме одного пропущенного заголовка. Тк, процент пропуска не велик(не достигает даже 1%), то просто удалим пропуски из столбца Title

Неявные пропуски

In [8]:
```
df.dropna(subset=['Title'],inplace = True,axis = 0 )
```

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
App ID	54045.0	1.049972e+06	514108.331134	10.0	629970.0	1033140.0	1466090.0	2111520.0
Reviews Total	54045.0	1.193282e+03	33335.491768	0.0	0.0	15.0	82.0	6580452.0
Reviews D7	54045.0	2.956542e+01	561.544065	0.0	0.0	0.0	0.0	79585.0
Reviews D30	54045.0	5.660224e+01	1179.436362	0.0	0.0	0.0	0.0	151632.0
Reviews D90	54045.0	7.800377e+01	1361.617582	0.0	0.0	0.0	0.0	154636.0

После подробного описания числовых значений видим, что в столбцах Reviews Total, Reviews D7, Reviews D30, Reviews D90 встречаются аномальные значения min, 25%, 50%, 75% - 0.0. Проверим какой процент пропусков будет составлять эти значения заменив их на NaN

In [10]:
```
data = df.copy()
data.replace(0, np.NaN,inplace=True)
```

In [11]:
```
draw_missing_data_table(data).round(1)
```

Out[11]:

	Total	Percent
Reviews D7	45404	84.0
Reviews D30	42718	79.0
Reviews D90	40582	75.1
Reviews Total	22278	41.2
App ID	0	0.0
Title	0	0.0
Reviews Score Fancy	0	0.0
Release Date	0	0.0
Launch Price	0	0.0
Tags	0	0.0

Как мы видим процент достаточно велик это может быть связано с тем, что

Игры, в которых 0 в ячейках D7/D30/D90, могут быть:

- Слишком стары, чтобы исследовать эти данные
- По какой-то причине данные были повреждены
- Буквально есть 0 отзывов

In [12]:
```
df = data.dropna()
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8286 entries, 0 to 31521
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   App ID               8286 non-null   int64  
 1   Title                8286 non-null   object 
 2   Reviews Total        8286 non-null   float64
 3   Reviews Score Fancy  8286 non-null   object 
 4   Release Date         8286 non-null   object 
 5   Reviews D7           8286 non-null   float64
 6   Reviews D30          8286 non-null   float64
 7   Reviews D90          8286 non-null   float64
 8   Launch Price         8286 non-null   object 
 9   Tags                 8286 non-null   object 
dtypes: float64(4), int64(1), object(5)
memory usage: 712.1+ KB
```

Приведение к корректному типу данных

Заметим, что Reviews Score Fancy, Launch Price, Revenue Estimated имеют посторонние знаки в значених создадим функцию, которая удалит эти аномальные знаки.

In [14]:
```
def convert_currency(val):
    """
    Преобразует числовое значение строки в число с плавающей точкой:
     - удаляет $
     - удаляет запятые
     - преобразует в число с плавающей точкой
    """
    new_val = val.replace(',', '.').replace('$', '')
    return float(new_val)
```

In [15]:
```
df['Launch Price']=df['Launch Price'].apply(convert_currency)
```

```
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/3872005823.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df['Launch Price']=df['Launch Price'].apply(convert_currency)
```

In [16]:
```
def convert_currency_per(val):
    """
    Преобразует числовое значение строки в число с плавающей точкой:
     - удаляет %
     - удаляет запятые
     - преобразует в число с плавающей точкой
    """
    new_val = val.replace(',', '.').replace('%', '')
    return float(new_val)
```

In [17]:
```
df['Reviews Score Fancy']=df['Reviews Score Fancy'].apply(convert_currency_per)
```

```
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/4125488754.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df['Reviews Score Fancy']=df['Reviews Score Fancy'].apply(convert_currency_per)
```

In [18]:
```
df['Reviews Score Fancy']=df['Reviews Score Fancy'].apply(lambda x: x / 100)
```

```
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/3854904187.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df['Reviews Score Fancy']=df['Reviews Score Fancy'].apply(lambda x: x / 100)
```

In [19]:
```
df[['Reviews Score Fancy', 'Launch Price']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8286 entries, 0 to 31521
Data columns (total 2 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Reviews Score Fancy  8286 non-null   float64
 1   Launch Price         8286 non-null   float64
dtypes: float64(2)
memory usage: 194.2 KB
```

Обратим внимание на то, что у даты реализации не верный тип данных переведем его в datetime и выделим из данного столбца

год, для дальнейшего анализа

In [20]:
```
df['Release Date']=pd.to_datetime(df['Release Date'])
```

```
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/1373310905.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df['Release Date']=pd.to_datetime(df['Release Date'])
```

In [21]:
```
df['Year']=pd.DatetimeIndex(df['Release Date']).year
```

```
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/3787351047.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df['Year']=pd.DatetimeIndex(df['Release Date']).year
```

In [22]:
```
df['Year'].info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 8286 entries, 0 to 31521
Series name: Year
Non-Null Count  Dtype
--------------  -----
8286 non-null   int64
dtypes: int64(1)
memory usage: 129.5 KB
```

Проверим данные на дубликаты.

Обработка дубликатов

In [23]:
```
df.duplicated().sum()
```

Out[23]:
```
0
```

In [24]:
```
df.columns
```

Out[24]:
```
Index(['App ID', 'Title', 'Reviews Total', 'Reviews Score Fancy',
       'Release Date', 'Reviews D7', 'Reviews D30', 'Reviews D90',
       'Launch Price', 'Tags', 'Year'],
      dtype='object')
```

In [25]:
```
df = df.rename(columns={"App ID": "id"})
```

Обработка датафрейма закончена.

Визуальный анализ

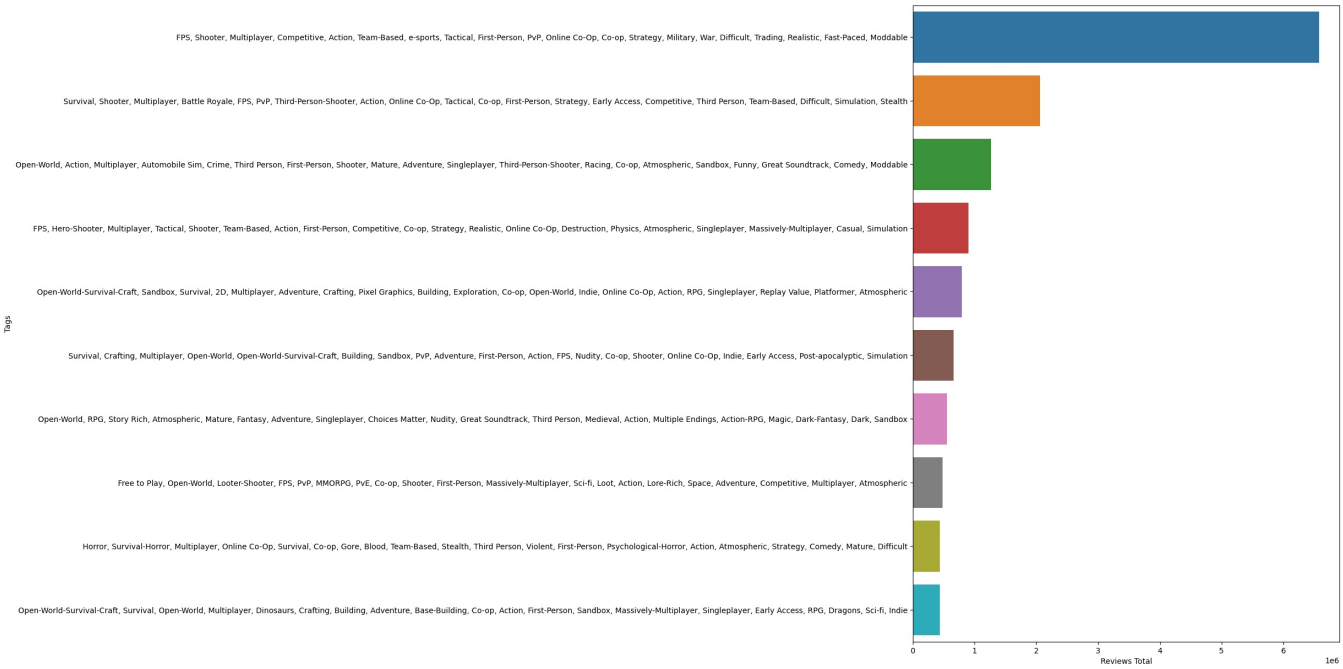
In [26]:
```
top_tags = df.groupby(['Tags']).agg({'Reviews Total': 'sum'}).reset_index()
top_tags.columns = ['Tags', 'Reviews Total']
top_tags = top_tags.sort_values('Reviews Total',ascending=False)
```

In [27]:
```
top10=top_tags.head(10)
```

In [28]:
```
plt.figure(figsize = (10,15))
sns.barplot(x='Reviews Total', y = 'Tags',data = top10);

top10
```

Out[28]:

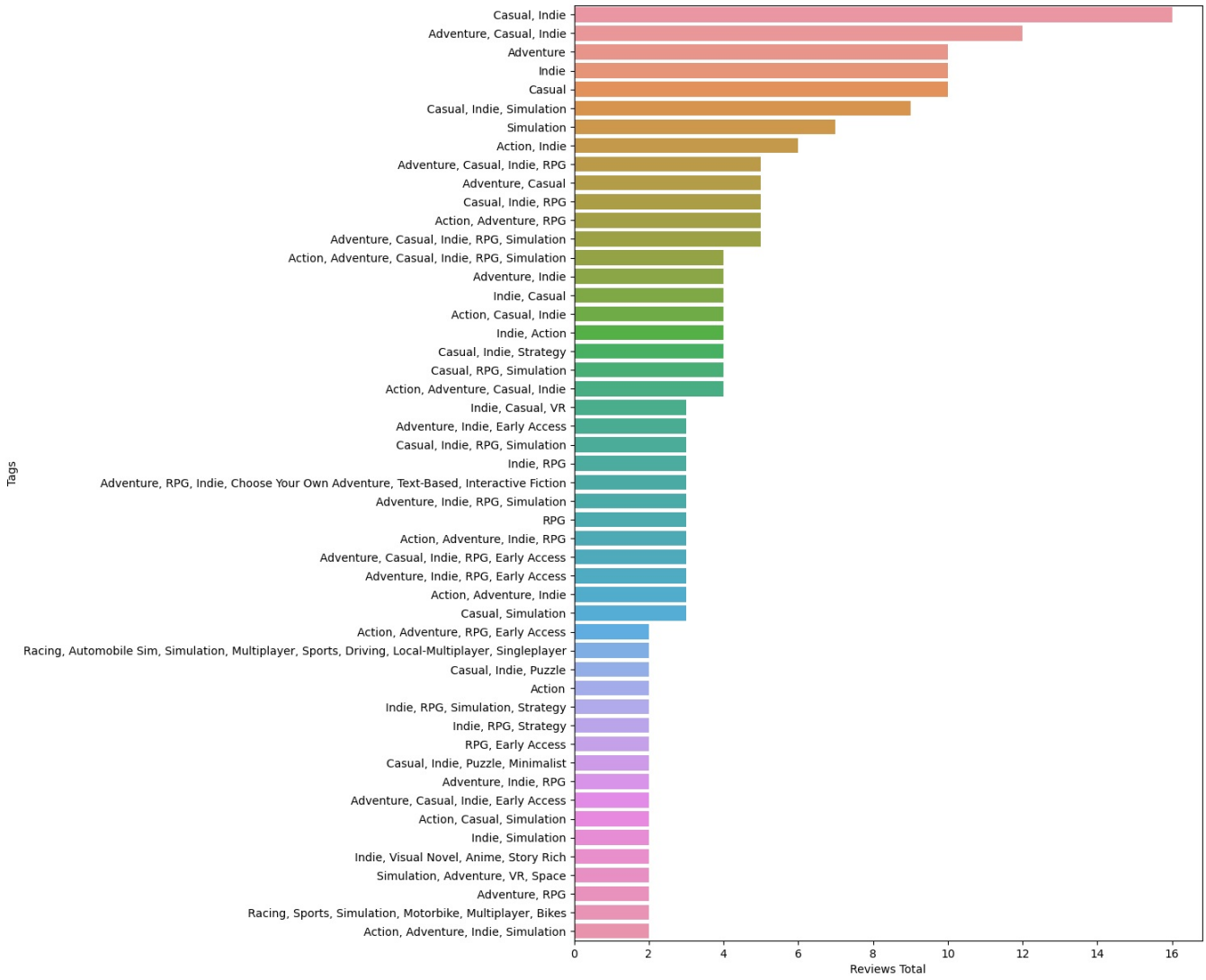
	Tags	Reviews Total
3195	FPS, Shooter, Multiplayer, Competitive, Action...	6580452.0
7603	Survival, Shooter, Multiplayer, Battle Royale,...	2062357.0
4760	Open-World, Action, Multiplayer, Automobile Si...	1262540.0
3163	FPS, Hero-Shooter, Multiplayer, Tactical, Shoo...	903537.0
4832	Open-World-Survival-Craft, Sandbox, Survival, ...	791389.0
7561	Survival, Crafting, Multiplayer, Open-World, O...	663866.0
4792	Open-World, RPG, Story Rich, Atmospheric, Matu...	552734.0
3330	Free to Play, Open-World, Looter-Shooter, FPS,...	483864.0
3600	Horror, Survival-Horror, Multiplayer, Online C...	439048.0
4851	Open-World-Survival-Craft, Survival, Open-Worl...	438642.0



In [29]:
```
top_tag_count = df.groupby(['Tags']).agg({'Reviews Total': 'count'}).reset_index()
top_tag_count.columns = ['Tags', 'Reviews Total']
top_tag_count = top_tag_count.sort_values('Reviews Total',ascending=False)
```

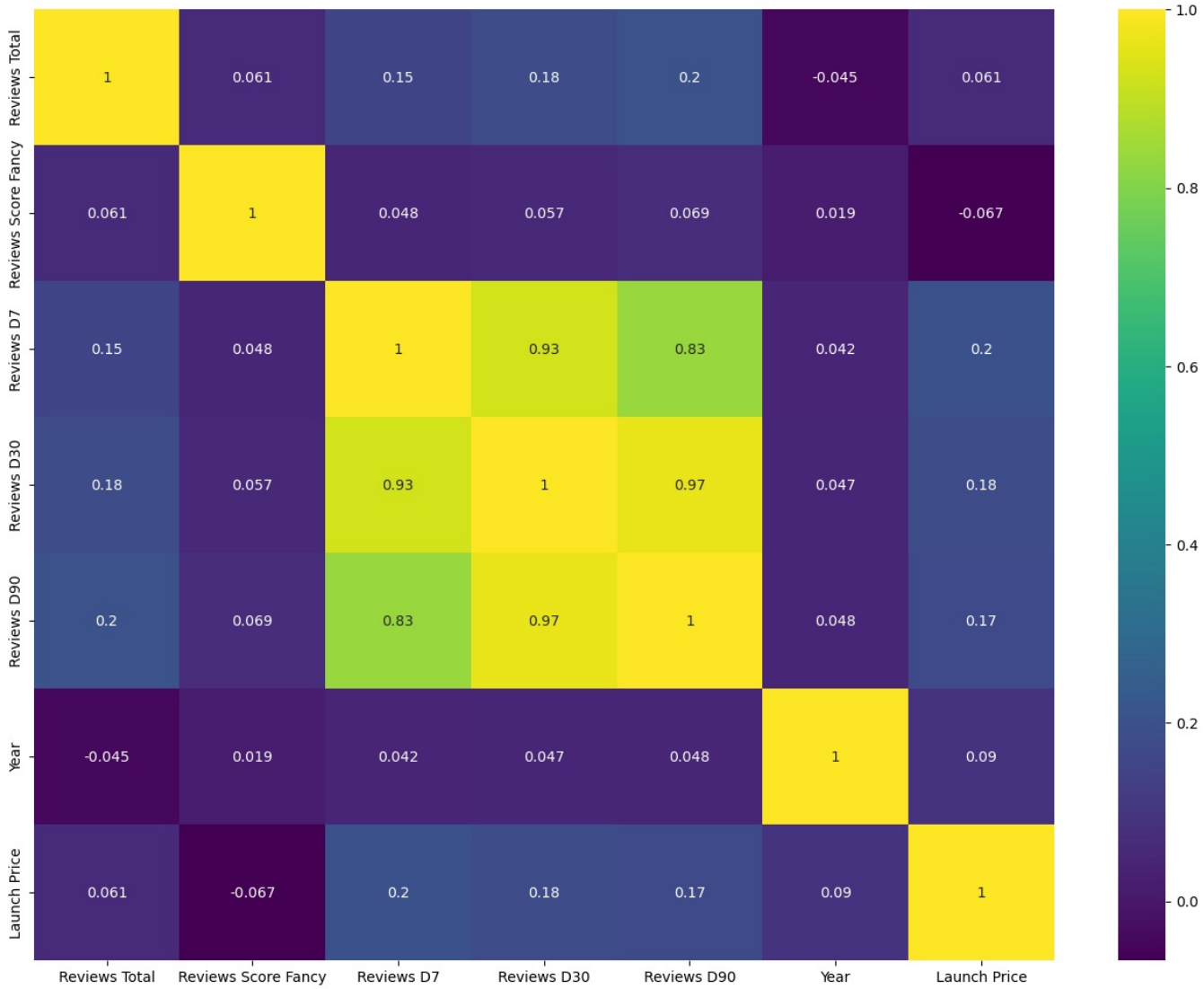
In [30]:
```
top50_count=top_tag_count.head(50)
```

In [31]:
```
plt.figure(figsize = (10,15))
sns.barplot(x='Reviews Total', y = 'Tags',data = top50_count);
```



In [32]:
```
data = df[['Reviews Total', 'Reviews Score Fancy',
       'Release Date', 'Reviews D7', 'Reviews D30', 'Reviews D90',
       'Year', 'Launch Price']]
```

In [33]:
```
plt.figure(figsize=(16,12))
sns.heatmap(data.corr(),annot=True,cmap='viridis')
plt.show()
```



По тепловой диаграмме отлично видно, как взаимодействуют данные. Выделим пару особенностей: 1) Видно, что год, отзывы и стоимость имеют отрицательную или малую связь, те эти столбцы не зависят друг от друга

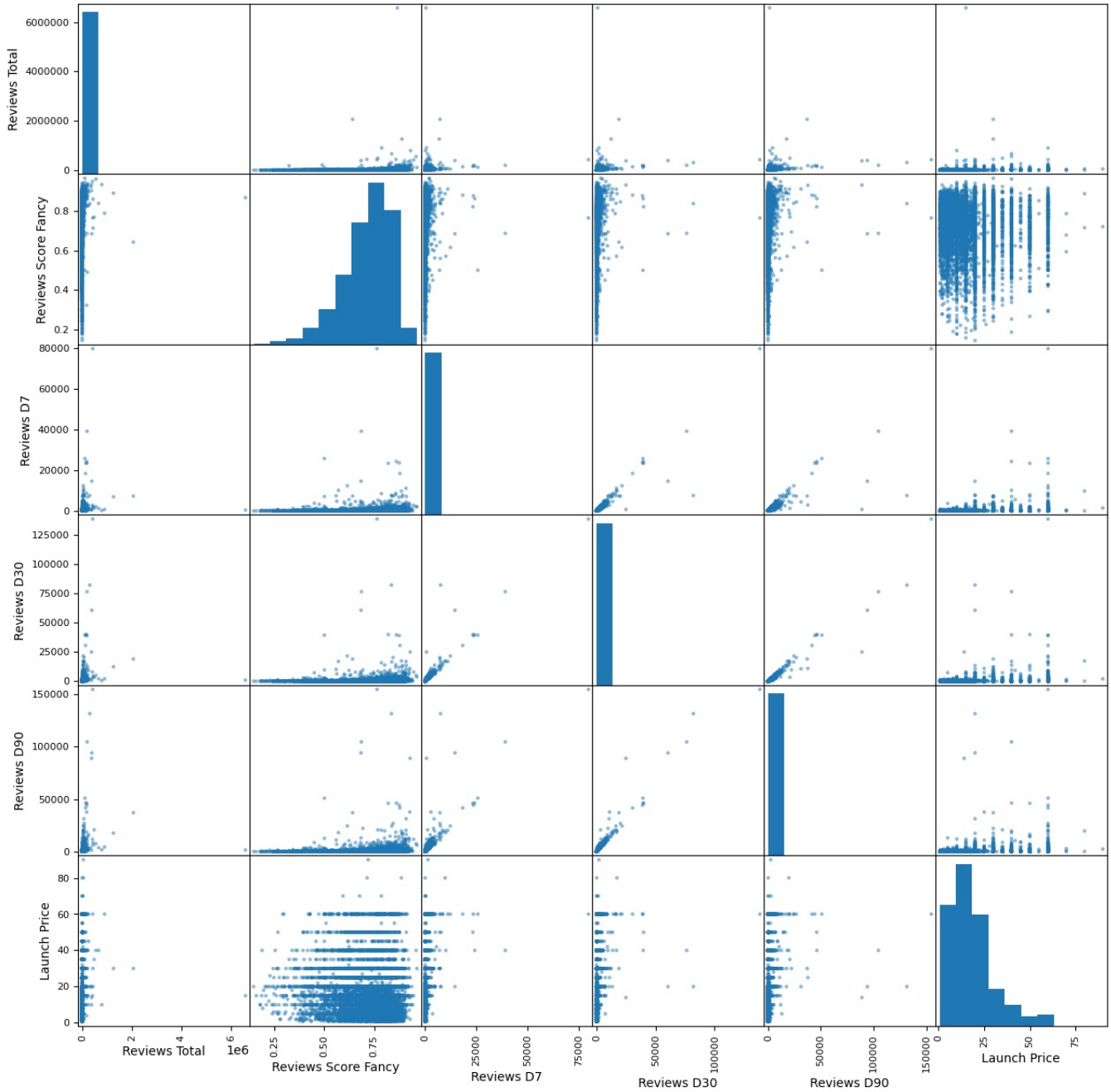
2) А вот 'Reviews D7', 'Reviews D30', 'Reviews D90' положительно влияют друг на друга

3) Цена не зависит от отзывов.

In [34]:
```
data = df.sort_values(by = 'Reviews Total', ascending=False)
```

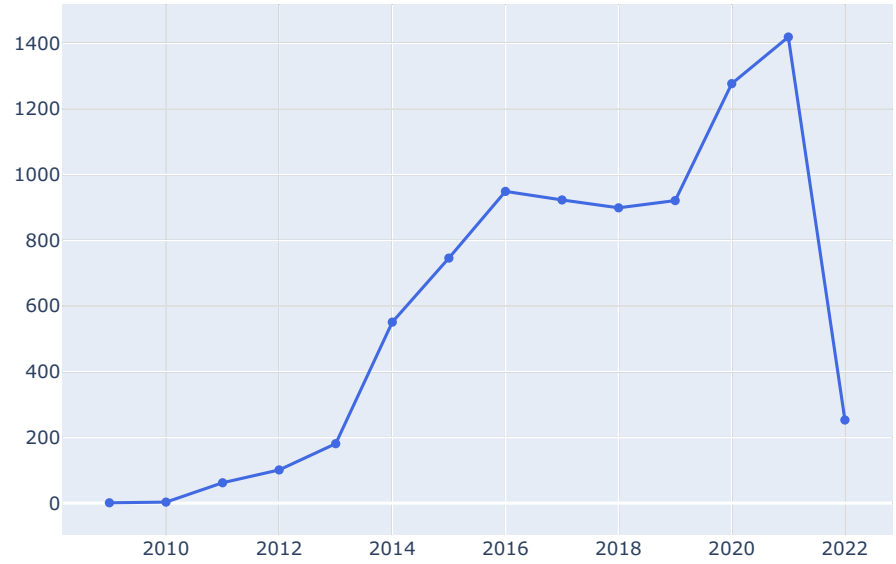
In [35]:
```
print(data[['Reviews Total', 'Reviews Score Fancy',
       'Release Date', 'Reviews D7', 'Reviews D30', 'Reviews D90',
        'Launch Price']].corr())
pd.plotting.scatter_matrix(data[['Reviews Total', 'Reviews Score Fancy',
       'Release Date', 'Reviews D7', 'Reviews D30', 'Reviews D90',
        'Launch Price']],figsize=(15,15));
```

```
                  Reviews Total  Reviews Score Fancy  Reviews D7  \
Reviews Total          1.000000             0.061336    0.151380
Reviews Score Fancy    0.061336             1.000000    0.048261
Reviews D7             0.151380             0.048261    1.000000
Reviews D30            0.175795             0.056823    0.928409
Reviews D90            0.201443             0.068940    0.833482
Launch Price           0.061172            -0.066617    0.196508

                  Reviews D30  Reviews D90  Launch Price
Reviews Total        0.175795     0.201443      0.061172
Reviews Score Fancy  0.056823     0.068940     -0.066617
Reviews D7           0.928409     0.833482      0.196508
Reviews D30          1.000000     0.965162      0.177761
Reviews D90          0.965162     1.000000      0.172378
Launch Price         0.177761     0.172378      1.000000
```



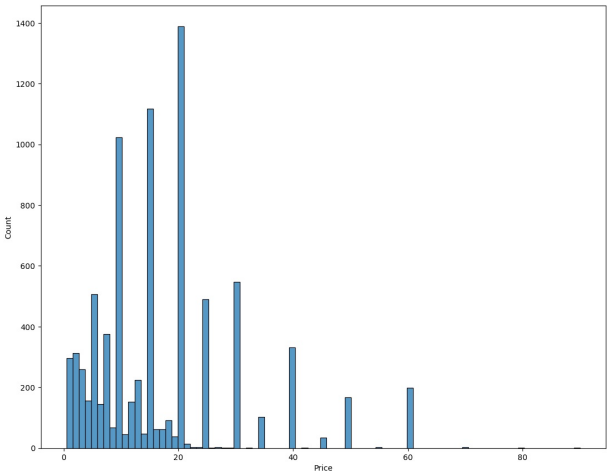
In [36]:
```
years_data = data['Year'].value_counts().reset_index().rename(columns = {'Year' : 'count'}).sort_values('index'

t1 = go.Scatter(x=years_data['index'], y=years_data["count"], name="Games", marker=dict(color="royalblue"))

data2 = [t1]

layout = go.Layout(title="Новые игры, появившиеся за эти годы", legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data2, layout=layout)
fig.show()
```

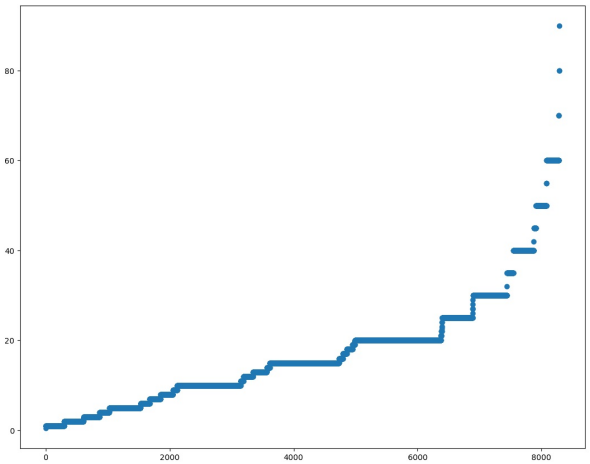
Новые игры, появившиеся за эти годы



In [37]:
```
new_data = data.head()
```

In [38]:
```
df = df.rename(columns={"Launch Price": "Price"})
plt.figure(figsize = (30, 10))

plt.subplot(121)

sns.histplot(df['Price'])

plt.subplot(122)
g1 = plt.scatter(range(df.shape[0]), np.sort(df.Price.values))


plt.subplots_adjust(wspace = 0.3, hspace = 0.5,
                    top = 0.9)
plt.show()
```





Большинство игр имеют стартовую цену менее 20 долларов. Короткий хвост распределения длиннее с правой стороны по сравнению с левой, что показывает, что распределение отрицательно искажено.

Посмотри какие игры имеют "аномальную" стоимость, те тот самый отрицательно искаженный хвост.

In [39]:
```
df = df.round(1)
```

In [40]:
```
top_games = df.groupby(['Title']).agg({'Price': 'sum'}).reset_index()
top_games.columns = ['name', 'price']
top_games = top_games.sort_values('price',ascending=False)
fig = px.bar(top_games[:10], x='name', y='price', color='name', title='Top Games based on total sum of their pr


fig.show()
```

Гипотеза 1

- H0: У игр с ценой 20 и 60 долларов рейтинг одинаковый
- H1: У игр с ценой 20 и 60 долларов рейтинг разный

In [41]:
```
df.describe().T
```

Out[41]:

	count	mean	std	min	25%	50%	75%	max
id	8286.0	778199.992035	439882.135226	620.0	396237.5	672745.0	1124147.50	1889000.0
Reviews Total	8286.0	6332.878590	81396.378466	10.0	150.0	479.0	1966.50	6580452.0
Reviews Score Fancy	8286.0	0.714036	0.127270	0.1	0.6	0.7	0.80	1.0
Reviews D7	8286.0	179.894280	1258.653440	10.0	15.0	29.0	81.00	79585.0
Reviews D30	8286.0	337.029085	2483.323368	10.0	23.0	48.0	138.75	138239.0
Reviews D90	8286.0	489.966329	3432.256145	10.0	32.0	69.0	205.00	154636.0
Price	8286.0	17.252812	12.834177	0.5	9.0	15.0	20.00	90.0
Year	8286.0	2017.940019	2.565713	2009.0	2016.0	2018.0	2020.00	2022.0

In [42]:
```
df['Reviews Total'] = df['Reviews Total'].astype('Int64')
```

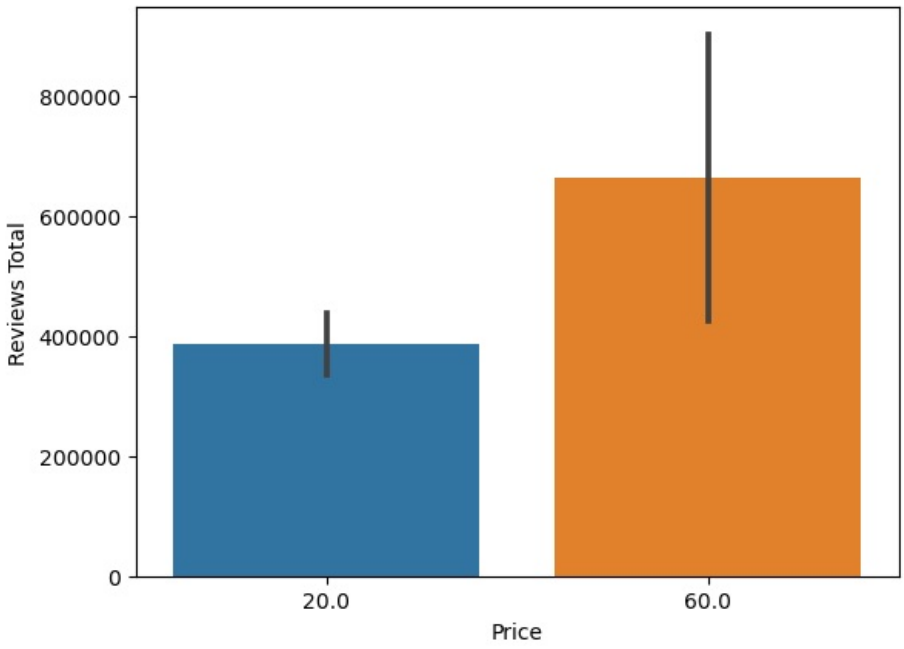
In [43]:
```
alpha = .01
result_first = st.ttest_ind(df.query('Price == 20.00')['Reviews Total'].dropna(),
                            df.query('Price == 60.00')['Reviews Total'].dropna(),
                            equal_var=False)

prob_first = result_first.pvalue
print('p-значение: ',prob_first)
if prob_first < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/1727161654.py in <module>
      1 alpha = .01
----> 2 result_first = st.ttest_ind(df.query('Price == 20.00')['Reviews Total'].dropna(),
      3                            df.query('Price == 60.00')['Reviews Total'].dropna(),
      4                            equal_var=False)
      5

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in ttest_ind(a, b, axis, equal_var, nan_po
licy, permutations, random_state, alternative, trim)
   6582
   6583         if trim == 0:
-> 6584             v1 = _var(a, axis, ddof=1)
   6585             v2 = _var(b, axis, ddof=1)
   6586             m1 = np.mean(a, axis)

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in _var(x, axis, ddof, mean)
   1245 def _var(x, axis=0, ddof=0, mean=None):
   1246     # Calculate variance of sample, warning if precision is lost
-> 1247     var = _moment(x, 2, axis, mean=mean)
   1248     if ddof != 0:
   1249         n = x.shape[axis] if axis is not None else x.size

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in _moment(a, moment, axis, mean)
   1218         a_zero_mean = a - mean
   1219
-> 1220         eps = np.finfo(a_zero_mean.dtype).resolution * 10
   1221         with np.errstate(divide='ignore', invalid='ignore'):
   1222             rel_diff = np.max(np.abs(a_zero_mean), axis=axis,

~/opt/anaconda3/lib/python3.9/site-packages/numpy/core/getlimits.py in __new__(cls, dtype)
    396             dtype = newdtype
    397         if not issubclass(dtype, numeric.inexact):
--> 398             raise ValueError("data type %r not inexact" % (dtype))
    399         obj = cls._finfo_cache.get(dtype, None)
    400         if obj is not None:

ValueError: data type <class 'numpy.object_'> not inexact
```

In [44]:
```
data=df.query('Price == 20.00 | Price == 60.00').head()
```

In [45]:
```
plt.figsize=(16,8)
sns.barplot(x=data["Price"],y=data["Reviews Total"]);
print('Суммарная оценка зависит от стоимости, чем выше цена, тем выше оценка. В данном случае лидируют игры с ц
```

```
Суммарная оценка зависит от стоимости, чем выше цена, тем выше оценка. В данном случае лидируют игры с ценой 60
$
```



Гипотеза 2

Спустя время Reviews Total и Price должны расти, тк выходят обнавления, появляются добавления.

In [46]:
```
df['Title'].value_counts().head()
```

Out[46]:
```
The Good Life                          2
Counter-Strike: Global Offensive       1
MOTHER                                 1
Primal Light                           1
Paddle Up                              1
Name: Title, dtype: int64
```

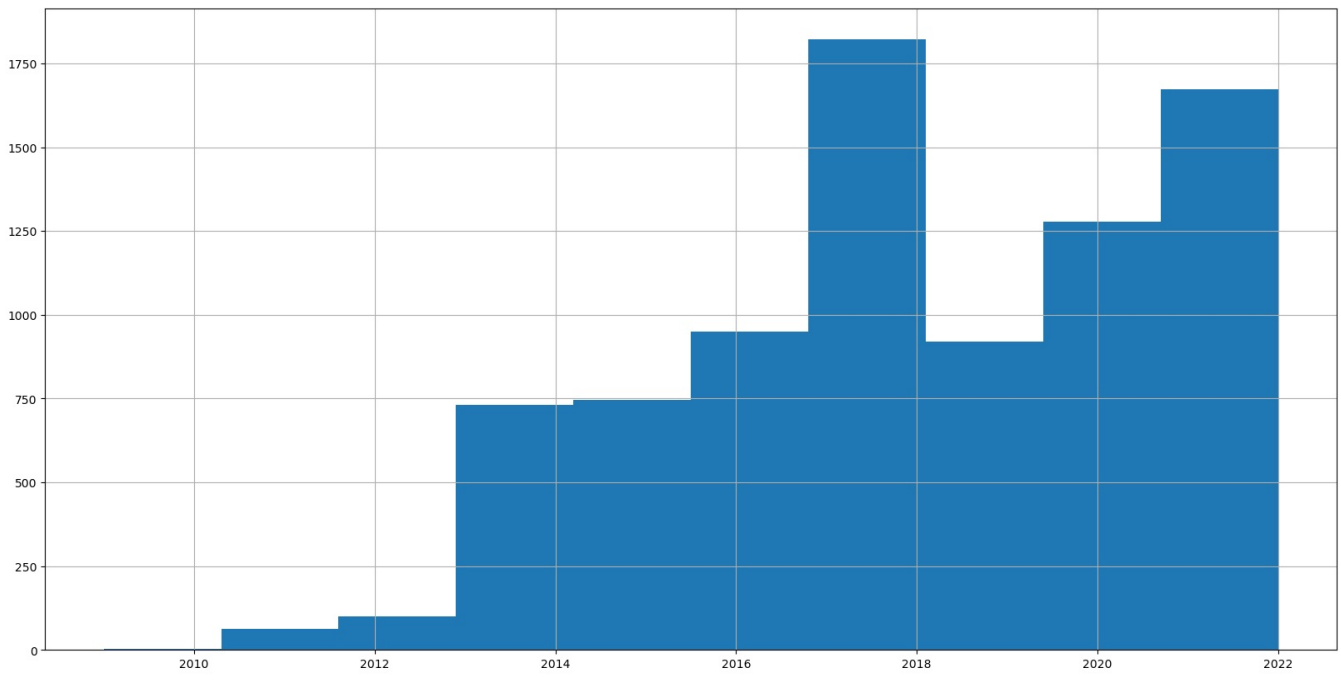
In [47]:
```
data=df.query('Title == "The Good Life"')
```

In [48]:
```
fig = go.Figure()

fig.add_trace(go.Scatter(x=data["Year"], y=data["Price"],name='Изменение стоимости со временем'))
fig.add_trace(go.Scatter(x=data["Year"], y=data["Reviews Total"],name='Изменение оценки со временем'))
fig.show()
```

По графикам видно, что стоимость игры выросла с 2014 на 20$, а рейтинг игры наоборот упал на 24 единицы. Это может быть связано с тем, что затраты на игру и инфлянция были выше, чем раньше, но при этом выпущенные обновления стали менее интересными.

Гипотеза 3

In [49]:
```
df['Year'].hist(figsize=(20,10))
plt.show()
```



Заметим, что пик выпуска игр приходит на 2017-2018, дальше видин резкий спад, а потом новый пик 2020-2022. Проверим, какие в какой промежуток рейтинг был выше.

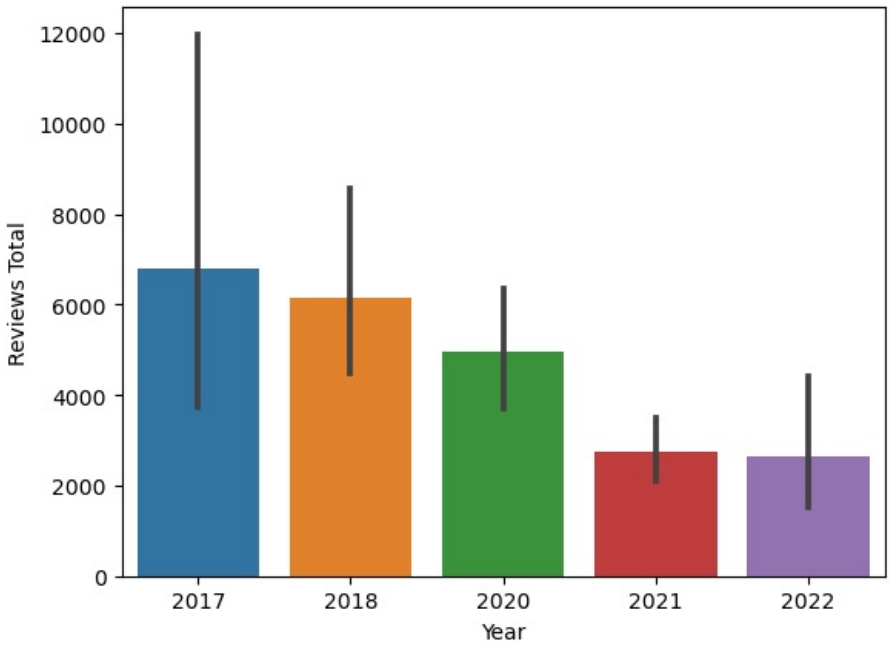
- H0: Средний рейтинг у игр выпуска 2017-2018 и 2020-2022 одинаковый
- H1: Средний рейтинг у игр выпуска 2017-2018 и 2020-2022 разный

In [50]:
```
p1 = (2017,2018)
p2 = (2020,2022)

alpha = .05
result_first = st.ttest_ind(df.query('@p1[0]<=Year<=@p1[1]')['Reviews Total'].dropna(),
                            df.query('@p2[0]<=Year<=@p2[1]')['Reviews Total'].dropna(),
                            equal_var=False)

prob_first = result_first.pvalue
print('p-значение: ',prob_first)
if prob_first < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/wk/x1qc19p10hd1t2ps922n01tr0000gp/T/ipykernel_1562/2956908420.py in <module>
      3
      4 alpha = .05
----> 5 result_first = st.ttest_ind(df.query('@p1[0]<=Year<=@p1[1]')['Reviews Total'].dropna(),
      6                            df.query('@p2[0]<=Year<=@p2[1]')['Reviews Total'].dropna(),
      7                            equal_var=False)

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in ttest_ind(a, b, axis, equal_var, nan_po
licy, permutations, random_state, alternative, trim)
   6582
   6583         if trim == 0:
-> 6584             v1 = _var(a, axis, ddof=1)
   6585             v2 = _var(b, axis, ddof=1)
   6586             m1 = np.mean(a, axis)

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in _var(x, axis, ddof, mean)
   1245 def _var(x, axis=0, ddof=0, mean=None):
   1246     # Calculate variance of sample, warning if precision is lost
-> 1247     var = _moment(x, 2, axis, mean=mean)
   1248     if ddof != 0:
   1249         n = x.shape[axis] if axis is not None else x.size

~/opt/anaconda3/lib/python3.9/site-packages/scipy/stats/_stats_py.py in _moment(a, moment, axis, mean)
   1218         a_zero_mean = a - mean
   1219
-> 1220         eps = np.finfo(a_zero_mean.dtype).resolution * 10
   1221         with np.errstate(divide='ignore', invalid='ignore'):
   1222             rel_diff = np.max(np.abs(a_zero_mean), axis=axis,

~/opt/anaconda3/lib/python3.9/site-packages/numpy/core/getlimits.py in __new__(cls, dtype)
    396             dtype = newdtype
    397         if not issubclass(dtype, numeric.inexact):
--> 398             raise ValueError("data type %r not inexact" % (dtype))
    399         obj = cls._finfo_cache.get(dtype, None)
    400         if obj is not None:

ValueError: data type <class 'numpy.object_'> not inexact
```

In [51]:
```
data = df.query('@p1[0]<=Year<=@p1[1] | @p2[0]<=Year<=@p2[1]')
```

In [52]:
```
plt.figsize=(16,8)
sns.barplot(x=data["Year"],y=data["Reviews Total"]);
```



Как видим, что рейтинг игр, выпушенных в 2021-2021 намного ниже, чем у игр 2017-2018