Сборный проект

Вы работаете в интернет-магазине «Стримчик», который продаёт по всему миру компьютерные игры. Из открытых источников доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation). Вам нужно выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

Перед вами данные до 2016 года. Представим, что сейчас декабрь 2016 г., и вы планируете кампанию на 2017-й. Нужно отработать принцип работы с данными. Неважно, прогнозируете ли вы продажи на 2017 год по данным 2016-го или же 2027-й — по данным 2026 года.

В наборе данных попадается аббревиатура ESRB (Entertainment Software Rating Board) — это ассоциация, определяющая возрастной рейтинг компьютерных игр. ESRB оценивает игровой контент и присваивает ему подходящую возрастную категорию, например, «Для взрослых», «Для детей младшего возраста» или «Для подростков».

Подготовка к работе

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('/datasets/games.csv')
```

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

Out[3]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Oth
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16713 non-null  object 
 1   Platform         16715 non-null  object 
 2   Year_of_Release  16446 non-null  float64
 3   Genre            16713 non-null  object 
 4   NA_sales         16715 non-null  float64
 5   EU_sales         16715 non-null  float64
 6   JP_sales         16715 non-null  float64
 7   Other_sales      16715 non-null  float64
 8   Critic_Score     8137 non-null   float64
 9   User_Score       10014 non-null  object 
 10  Rating           9949 non-null   object 
dtypes: float64(6), object(5)
memory usage: 1.4+ MB
```

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

Out[5]:

	Year_of_Release	NA_sales	EU_sales	JP_sales	Other_sales	Critic_
count	16446.000000	16715.000000	16715.000000	16715.000000	16715.000000	8137.0
mean	2006.484616	0.263377	0.145060	0.077617	0.047342	68.9
std	5.877050	0.813604	0.503339	0.308853	0.186731	13.9
min	1980.000000	0.000000	0.000000	0.000000	0.000000	13.0
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	60.0
50%	2007.000000	0.080000	0.020000	0.000000	0.010000	71.0
75%	2010.000000	0.240000	0.110000	0.040000	0.030000	79.0
max	2016.000000	41.360000	28.960000	10.220000	10.570000	98.0

Предобработка данных

In [6]:
```
# Сначала разберемся с null значениями
df = df.dropna(subset=['Year_of_Release', 'Genre'])
```

In [7]:
```
df['Critic_Score'] = df['Critic_Score'].fillna(-1)
df['User_Score'] = df['User_Score'].fillna(-1)
df['Rating'] = df['Rating'].fillna('unknown')
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16444 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16444 non-null  object 
 1   Platform         16444 non-null  object 
 2   Year_of_Release  16444 non-null  float64
 3   Genre            16444 non-null  object 
 4   NA_sales         16444 non-null  float64
 5   EU_sales         16444 non-null  float64
 6   JP_sales         16444 non-null  float64
 7   Other_sales      16444 non-null  float64
 8   Critic_Score     16444 non-null  float64
 9   User_Score       16444 non-null  object 
 10  Rating           16444 non-null  object 
dtypes: float64(6), object(5)
memory usage: 1.5+ MB
```

In [9]:
```
# Теперь приведем названия столбцов и текстовые поля к нижнему регистру
df.columns = df.columns.str.lower()
```

In [10]:
```
df['name'] = df['name'].str.lower()
df['platform'] = df['platform'].str.lower()
df['genre'] = df['genre'].str.lower()
```

In [11]:
```
# Преобразование типов
df['year_of_release'] = df['year_of_release'].astype(int)
```

In [12]:
```
df['user_score'].unique()
```

Out[12]:
```
array(['8', -1, '8.3', '8.5', '6.6', '8.4', '8.6', '7.7', '6.3', '7.4',
       '8.2', '9', '7.9', '8.1', '8.7', '7.1', '3.4', '5.3', '4.8', '3.2'
,
       '8.9', '6.4', '7.8', '7.5', '2.6', '7.2', '9.2', '7', '7.3', '4.3'
,
       '7.6', '5.7', '5', '9.1', '6.5', 'tbd', '8.8', '6.9', '9.4', '6.8'
,
       '6.1', '6.7', '5.4', '4', '4.9', '4.5', '9.3', '6.2', '4.2', '6',
       '3.7', '4.1', '5.8', '5.6', '5.5', '4.4', '4.6', '5.9', '3.9',
       '3.1', '2.9', '5.2', '3.3', '4.7', '5.1', '3.5', '2.5', '1.9', '3'
,
       '2.7', '2.2', '2', '9.5', '2.1', '3.6', '2.8', '1.8', '3.8', '0',
       '1.6', '9.6', '2.4', '1.7', '1.1', '0.3', '1.5', '0.7', '1.2',
       '2.3', '0.5', '1.3', '0.2', '0.6', '1.4', '0.9', '1', '9.7'],
      dtype=object)
```

In [13]:
```
# tbd - оставим как есть
```

In [14]:
```
# Обработаем дубликаты
df.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
df['total_sales'] = = df[['na_sales','eu_sales','jp_sales', 'other_sales'
```

In [16]:
```
df.head()
```

Out[16]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_
0	wii sports	wii	2006	sports	41.36	28.96	3.77	
1	super mario bros.	nes	1985	platform	29.08	3.58	6.81	
2	mario kart wii	wii	2008	racing	15.68	12.76	3.79	
3	wii sports resort	wii	2009	sports	15.61	10.93	3.28	
4	pokemon red/pokemon blue	gb	1996	role-playing	11.27	8.89	10.22	

In [17]:
```
df.loc[df['user_score']  == "tbd" , 'user_score'] = -1
df['user_score'] = df['user_score'].astype(float)
```

Вывод по предобработке:

- Названия столбцов и записи в текстовых столбцах приведены к нижнему регистру
- Строк с пропусками в столбцах 'year_of_release', 'genre' мало, поэтому они были удалены
- Пропуски в столбцах critic_score, user_score заполнены значением -1
- Пропуски в столбце rating заполнены значением 'unknown'
- Явные дубликаты не найдены
- Добавлен столбец total_sales, который содержит сумму продаж по всем регионам
- Столбец year_of_release приведен к типу int, тк. изначально содержал значения с плавающей точкой без дробной части
- Значение tbd заменено на -1, столбец user_score приведен к типу float

In [18]:
```
# Комментарий ревьюера
# Посмотрим, что у нас осталось
temp = df.copy()
list_c = ['name', 'platform', 'year_of_release', 'genre', 'critic_score',
print(temp.info())
for col_l in list_c:
  print('-'* 25)
  print(col_l, temp[col_l].unique())
  print(col_l,': кол-во NaN',temp[col_l].isna().sum(),
        ', процент NaN', round(temp[col_l].isna().sum()/len(temp)*100, 2)
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 16444 entries, 0 to 16714
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16444 non-null  object 
 1   platform         16444 non-null  object 
 2   year_of_release  16444 non-null  int64  
 3   genre            16444 non-null  object 
 4   na_sales         16444 non-null  float64
 5   eu_sales         16444 non-null  float64
 6   jp_sales         16444 non-null  float64
 7   other_sales      16444 non-null  float64
 8   critic_score     16444 non-null  float64
 9   user_score       16444 non-null  float64
 10  rating           16444 non-null  object 
 11  total_sales      16444 non-null  float64
dtypes: float64(7), int64(1), object(4)
memory usage: 1.6+ MB
None
-------------------------
name ['wii sports' 'super mario bros.' 'mario kart wii' ...
 'woody woodpecker in crazy castle 5' 'lma manager 2007'
 'haitaka no psychedelica']
name : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
platform ['wii' 'nes' 'gb' 'ds' 'x360' 'ps3' 'ps2' 'snes' 'gba' 'ps4' '3d
s' 'n64'
 'ps' 'xb' 'pc' '2600' 'psp' 'xone' 'wiiu' 'gc' 'gen' 'dc' 'psv' 'sat'
 'scd' 'ws' 'ng' 'tg16' '3do' 'gg' 'pcfx']
platform : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
year_of_release [2006 1985 2008 2009 1996 1989 1984 2005 1999 2007 2010 2
013 2004 1990
 1988 2002 2001 2011 1998 2015 2012 2014 1992 1997 1993 1994 1982 2016
 2003 1986 2000 1995 1991 1981 1987 1980 1983]
year_of_release : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
genre ['sports' 'platform' 'racing' 'role-playing' 'puzzle' 'misc' 'shoot
er'
 'simulation' 'action' 'fighting' 'adventure' 'strategy']
genre : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
critic_score [76. -1. 82. 80. 89. 58. 87. 91. 61. 97. 95. 77. 88. 83. 94.
93. 85. 86.
 98. 96. 90. 84. 73. 74. 78. 92. 71. 72. 68. 62. 49. 67. 81. 66. 56. 79.
 70. 59. 64. 75. 60. 63. 69. 50. 25. 42. 44. 55. 48. 57. 29. 47. 65. 54.
 20. 53. 37. 38. 33. 52. 30. 32. 43. 45. 51. 40. 46. 39. 34. 41. 36. 31.
 27. 35. 26. 19. 28. 23. 24. 21. 17. 13.]
critic_score : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
user_score [ 8.  -1.   8.3  8.5  6.6  8.4  8.6  7.7  6.3  7.4  8.2  9.
7.9  8.1
  8.7  7.1  3.4  5.3  4.8  3.2  8.9  6.4  7.8  7.5  2.6  7.2  9.2  7.
  7.3  4.3  7.6  5.7  5.   9.1  6.5  8.8  6.9  9.4  6.8  6.1  6.7  5.4
  4.   4.9  4.5  9.3  6.2  4.2  6.   3.7  4.1  5.8  5.6  5.5  4.4  4.6
  5.9  3.9  3.1  2.9  5.2  3.3  4.7  5.1  3.5  2.5  1.9  3.   2.7  2.2
  2.   9.5  2.1  3.6  2.8  1.8  3.8  0.   1.6  9.6  2.4  1.7  1.1  0.3
```

```
  1.5  0.7  1.2  2.3  0.5  1.3  0.2  0.6  1.4  0.9  1.   9.7]
user_score : кол-во NaN 0 , процент NaN 0.0 %
-------------------------
rating ['E' 'unknown' 'M' 'T' 'E10+' 'K-A' 'AO' 'EC' 'RP']
rating : кол-во NaN 0 , процент NaN 0.0 %
```

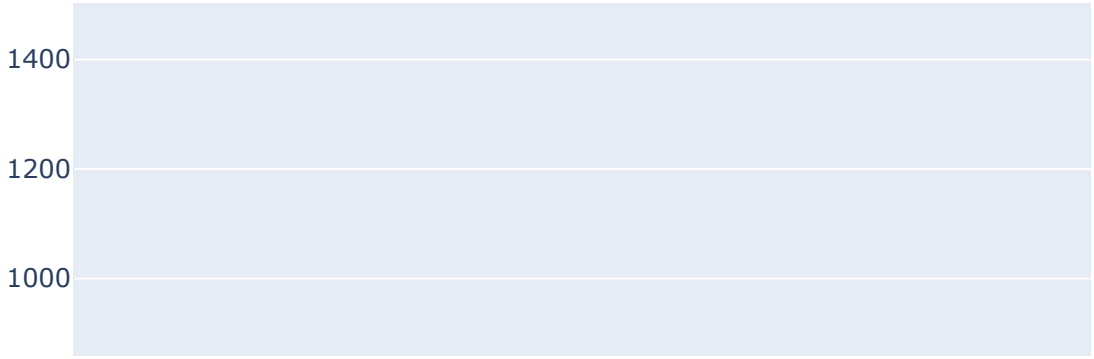
Исследовательский анализ данных

In [19]:
```
# Найдем продажи всех игр в разные годы
ax = df.groupby('year_of_release').agg(func='count')['name']
ax
```

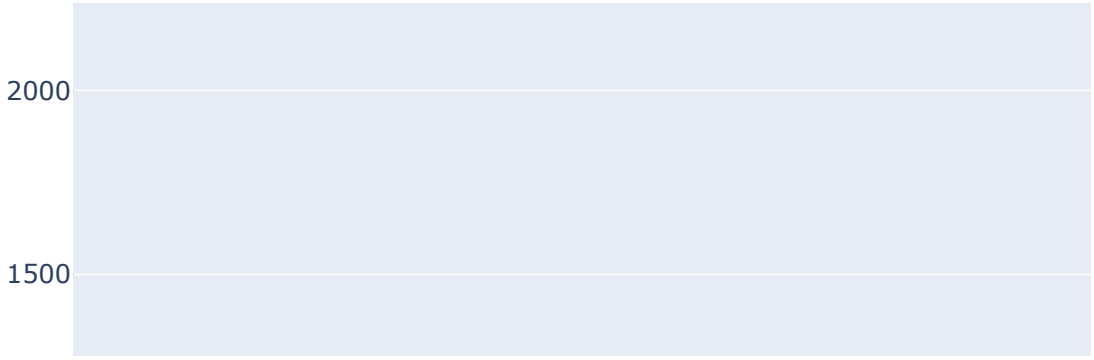
Out[19]:
```
year_of_release
1980       9
1981      46
1982      36
1983      17
1984      14
1985      14
1986      21
1987      16
1988      15
1989      17
1990      16
1991      41
1992      43
1993      60
1994     121
1995     219
1996     263
1997     289
1998     379
1999     338
2000     350
2001     482
2002     829
2003     775
2004     762
2005     939
2006    1006
2007    1197
2008    1427
2009    1426
2010    1255
2011    1136
2012     653
2013     544
2014     581
2015     606
2016     502
Name: name, dtype: int64
```

In [20]:
```
import plotly.express as plx
```

In [21]:
```
plx.bar(ax,x=ax.index, y='name', color='name')
```



In [22]:
```
ax = df.groupby('platform').agg(func='count')['name'].sort_values()
plx.bar(ax, x=ax.index, y='name', color='name')
```



In [23]:
```
year_platforms = df.pivot_table(index=['year_of_release', 'platform'], va
year_platforms.head(20)
```

Out[23]:

year_of_release	platform	name
1980	2600	9
1981	2600	46
1982	2600	36
1983	2600	11
	nes	6
1984	2600	1
	nes	13
1985	2600	1
	ds	1
	nes	11
	pc	1
1986	2600	2
	nes	19
1987	2600	6
	nes	10
1988	2600	2
	gb	1
	nes	11
	pc	1
1989	2600	2

In [24]:
```
year_platforms.tail(20)
```

Out[24]:

year_of_release	platform	name
2014	xone	61
2015	3ds	86
	pc	50
	ps3	73
	ps4	137
	psp	3
	psv	110
	wii	4
	wiu	28
	x360	35
	xone	80
2016	3ds	46
	pc	54
	ps3	38
	ps4	164
	psv	85
	wii	1
	wiu	14
	x360	13
	xone	87

In [25]:
```
# 10 самых популярных платформ
top_platforms = df.pivot_table(index=['platform'], values='name', aggfunc
    sort_values(ascending=False).head(10)
top_platforms
```

Out[25]:
```
platform
ps2     2127
ds      2121
ps3     1306
wii     1286
x360    1232
psp     1193
ps      1190
pc       957
gba      811
xb       803
Name: name, dtype: int64
```

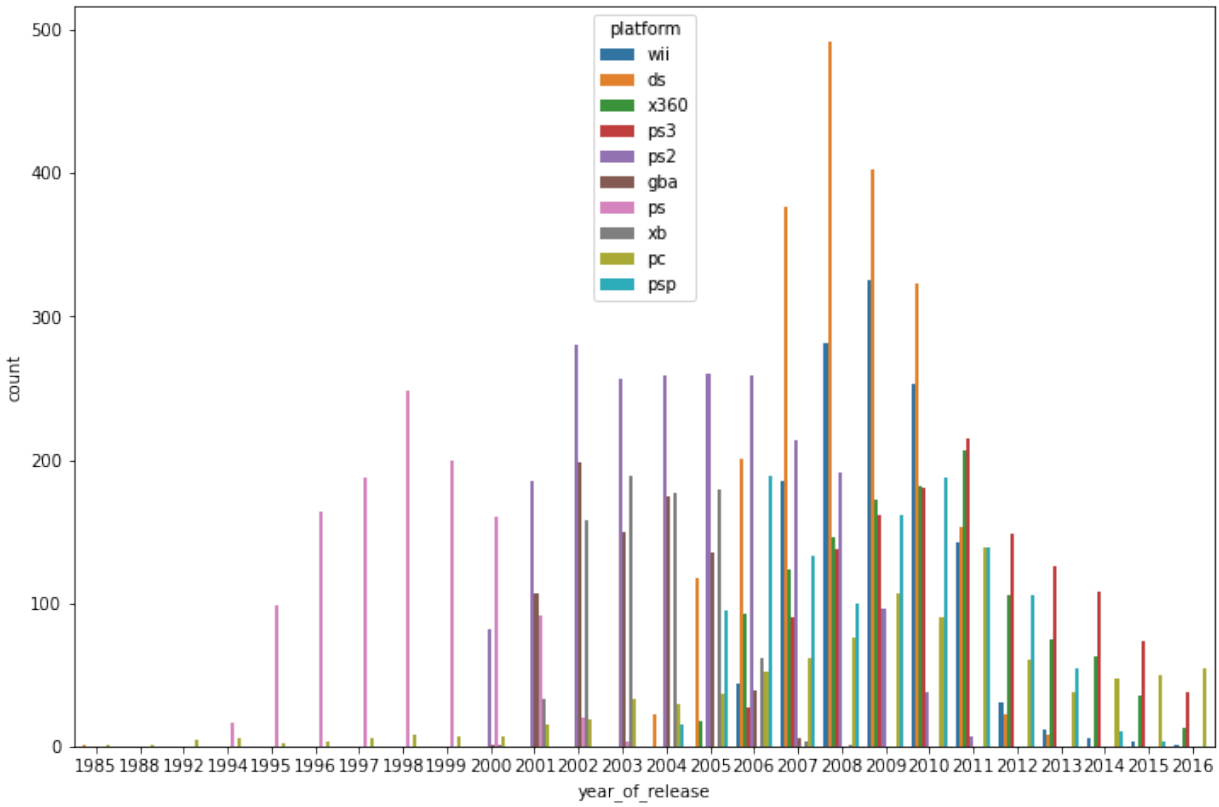
In [26]:
```
top_platforms_list = top_platforms.index
```

In [27]:
```
top_platforms_years = df.query('platform in @top_platforms_list')
```

In [28]:
```
top_platforms_years = top_platforms_years[['platform', 'year_of_release']]
```

In [29]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [30]:
```
# dfm = df.melt(id_vars=['Month', 'Code'], var_name='Cols')
# sns.catplot(kind='bar', data=dfm, col='Code', x='Month', y='value', row
#             col_order=sorted(df.Code.unique()), estimator=sum, ci=None,
plt.figure(figsize=(12, 8))
ax = sns.countplot(top_platforms_years.year_of_release, hue=top_platforms
```



Анализ графика:

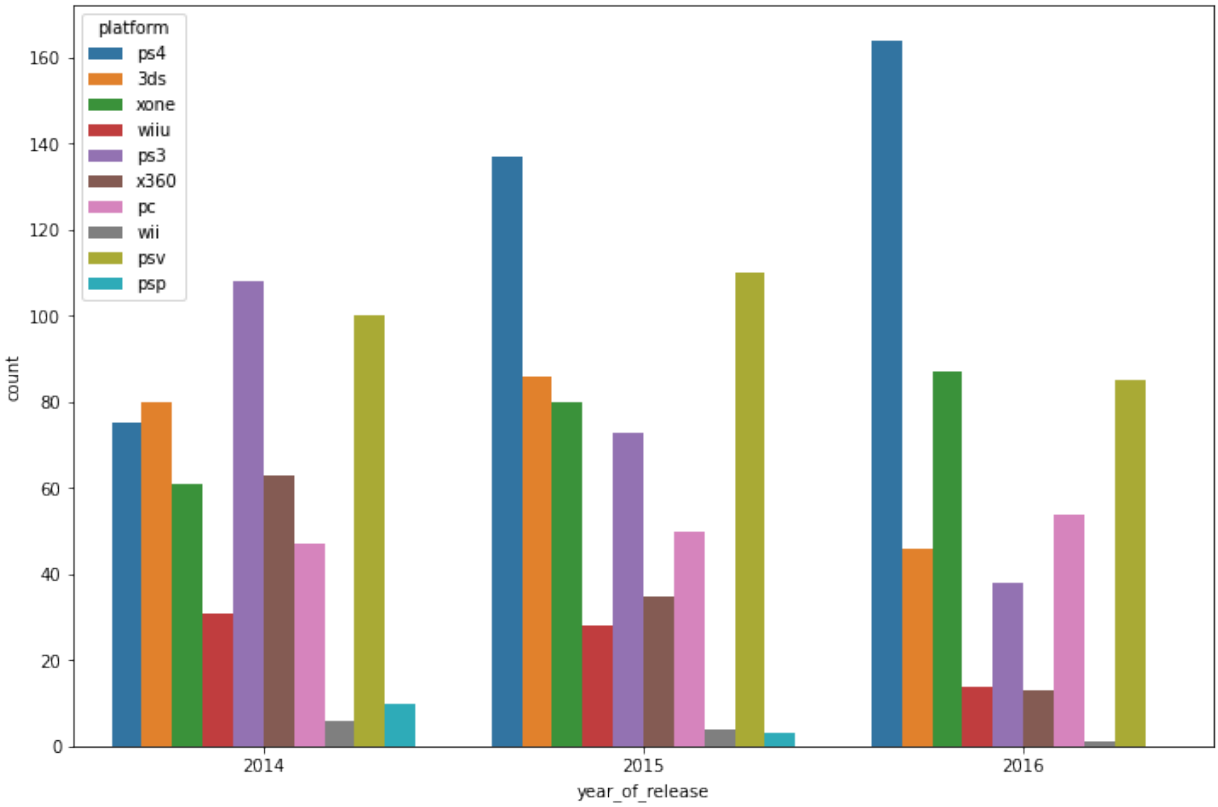
На примере ps, ds, можно видеть, что когда новая платформа появляется, наблюдается рост числа выходящих на нее игр, затем достигается максимум, затем снижение. Подобный цикл занимает 8 лет. Подобное поведение характерно для большинства платформ. Но есть и исключения, например ps2 достигает планки в 2002 году и удерживает уровень в течение 5 лет. Схожая картина и для xb. pc не теряет актуальности много лет, но игр на него выходит не так много.

В дальнейшем анализе будем считать период 3 года актуальным

In [31]:
```
df = df.query('year_of_release >= 2014')
```

In [32]:
```
platforms_actual = df[['platform', 'year_of_release', 'total_sales', 'cri
```

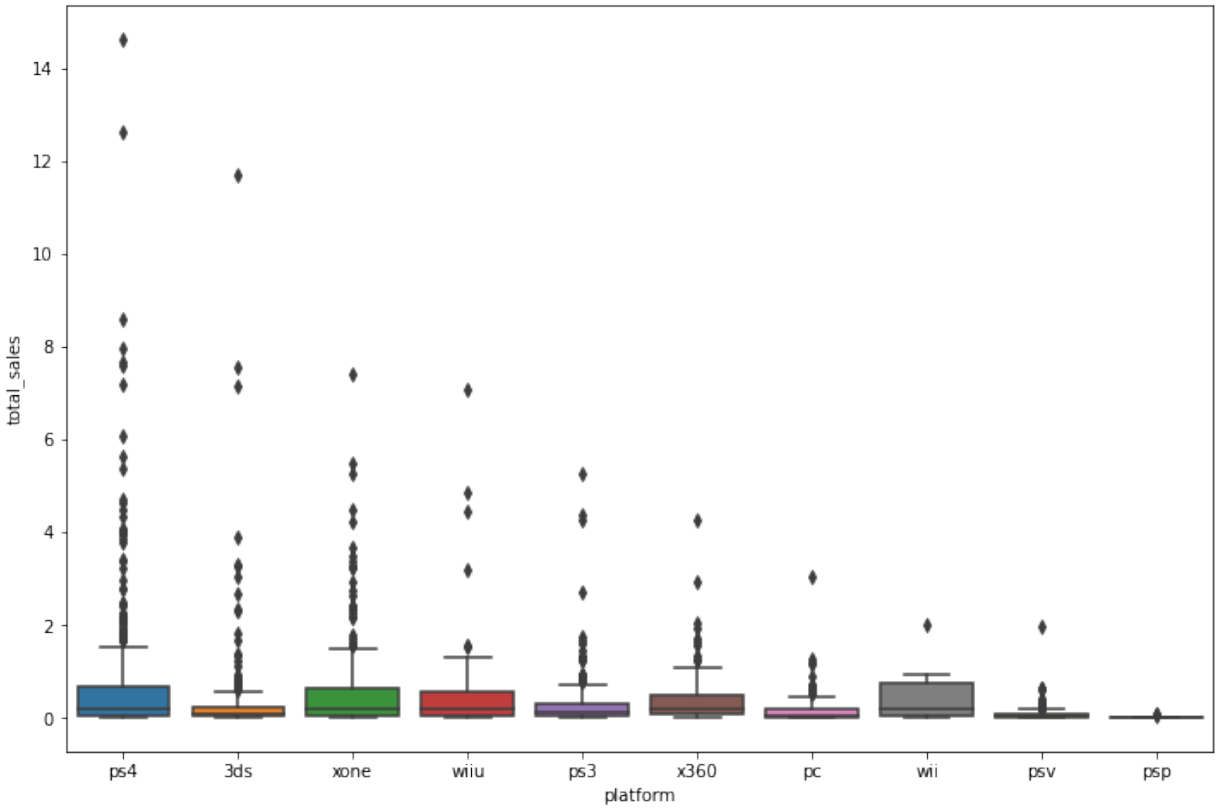
In [33]:
```
plt.figure(figsize=(12, 8))
ax = sns.countplot(platforms_actual.year_of_release, hue=platforms_actual
```



Анализ графика

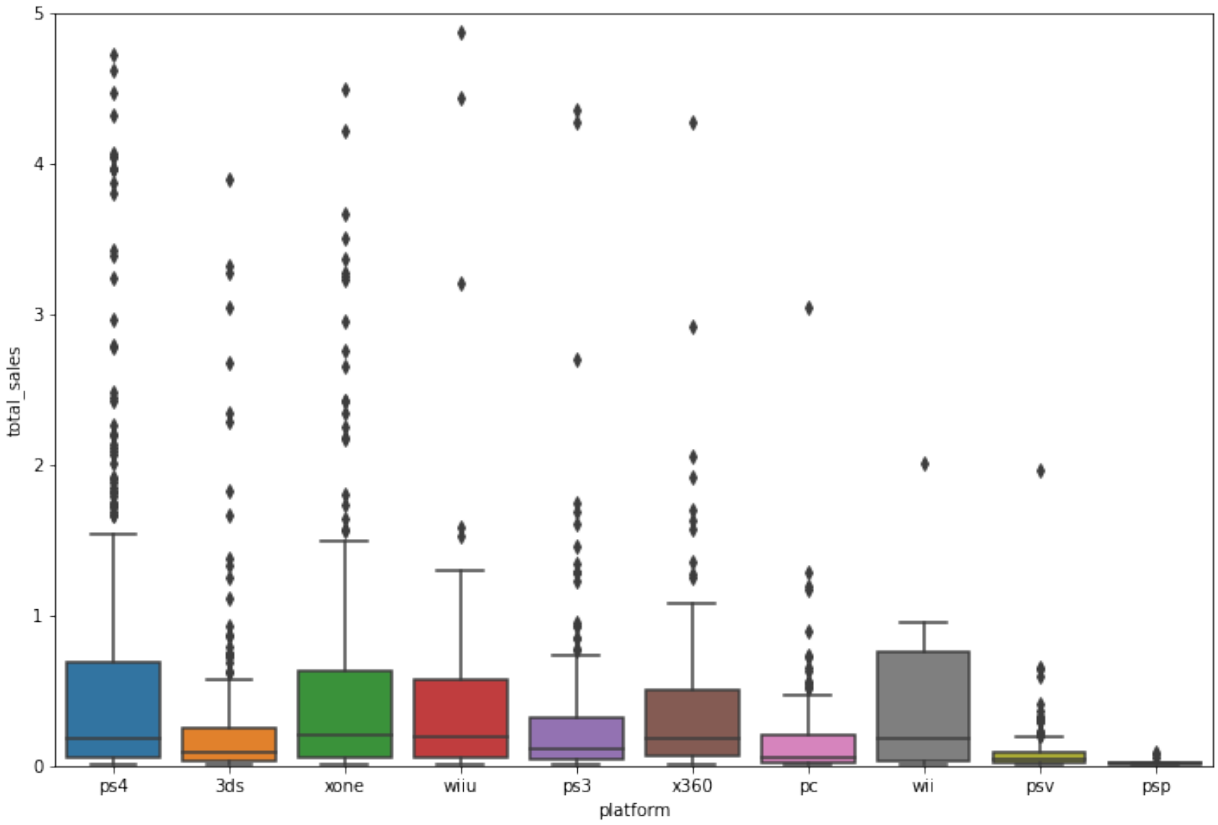
- Устаревающие платформы: ds, wii, psp, ps2, ps3
- Все еще актуальные платформы: pc, psv, pc
- Набирающие популярность платформы: ps4, xone, wiiu

In [34]:
```
plt.figure(figsize=(12, 8))
ax = sns.boxplot(data=platforms_actual, x='platform', y='total_sales')
```



In [35]:
```
plt.figure(figsize=(12, 8))
ax = sns.boxplot(data=platforms_actual, x='platform', y='total_sales')
ax.set_ylim([0, 5])
```

Out[35]:
```
(0.0, 5.0)
```



Анализ графиков

В основном продажи сосредоточены в интервале 0-1, имеется ощутимое число выбросов вверх для большинства платформ

Посмотрим, как влияют отзывы пользователей и критиков на продажи внутри популярной платформы

In [36]:
```
platforms_actual_users = platforms_actual.query('user_score != -1')
platforms_actual_critics = platforms_actual.query('critic_score != -1')
```

In [37]:
```
# Комментарий ревьюера
print(len(df.query("user_score != 'tbd' & user_score != -1")))
print(len(df.query("critic_score != -1")))
print(len(df.query("user_score != 'tbd' & critic_score != -1 & user_score
```

```
888
718
704
```

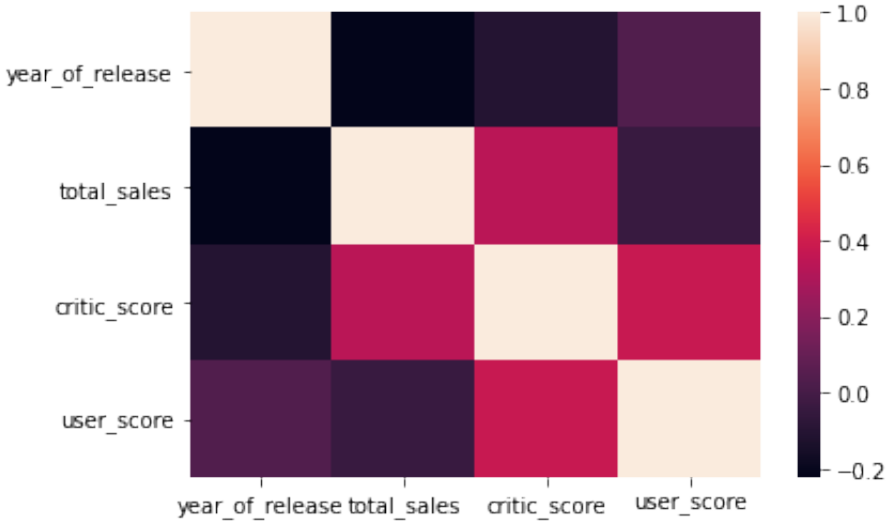
In [38]:
```
platforms_actual_users.query('platform == "ps4"').corr()
```

Out[38]:

	year_of_release	total_sales	critic_score	user_score
year_of_release	1.000000	-0.222551	-0.103807	0.035502
total_sales	-0.222551	1.000000	0.339828	-0.040132
critic_score	-0.103807	0.339828	1.000000	0.374716
user_score	0.035502	-0.040132	0.374716	1.000000

In [39]:
```
sns.heatmap(platforms_actual_users.query('platform == "ps4"').corr())
```

Out[39]:
```
<AxesSubplot:>
```



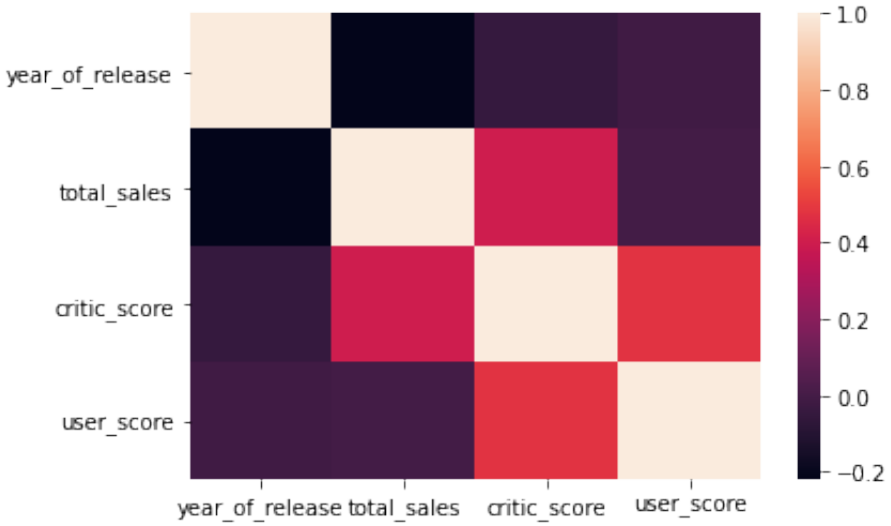
In [40]:
```
platforms_actual_critics.query('platform == "ps4"').corr()
```

Out[40]:

	year_of_release	total_sales	critic_score	user_score
year_of_release	1.000000	-0.219468	-0.044226	-0.012823
total_sales	-0.219468	1.000000	0.402661	-0.005041
critic_score	-0.044226	0.402661	1.000000	0.478431
user_score	-0.012823	-0.005041	0.478431	1.000000

In [41]:
```
sns.heatmap(platforms_actual_critics.query('platform == "ps4"').corr())
```

Out[41]:
```
<AxesSubplot:>
```



In [42]:
```
# Теперь рассмотрим одну из старых платформ
platforms_actual_users.query('platform == "ps3"').corr()
```

Out[42]:

	year_of_release	total_sales	critic_score	user_score
year_of_release	1.000000	-0.164926	-0.476033	-0.258182
total_sales	-0.164926	1.000000	-0.157873	-0.166761
critic_score	-0.476033	-0.157873	1.000000	0.217215
user_score	-0.258182	-0.166761	0.217215	1.000000

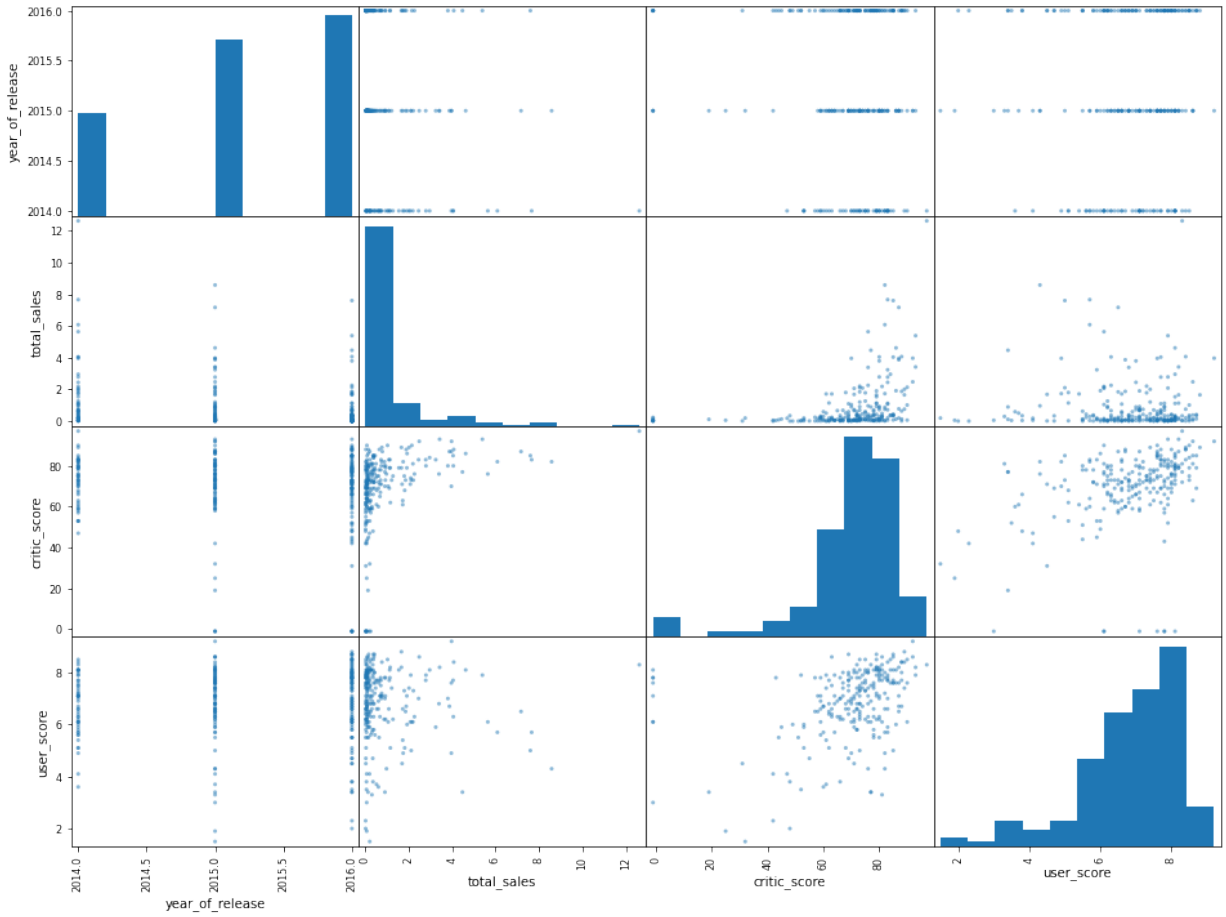
In [43]:
```
platforms_actual_critics.query('platform == "ps3"').corr()
```

Out[43]:

	year_of_release	total_sales	critic_score	user_score
year_of_release	1.000000	-0.146313	-0.044960	-0.013439
total_sales	-0.146313	1.000000	0.446575	0.156595
critic_score	-0.044960	0.446575	1.000000	0.693445
user_score	-0.013439	0.156595	0.693445	1.000000

In [44]:
```
pd.plotting.scatter_matrix(platforms_actual_users.query('platform == "ps4
```

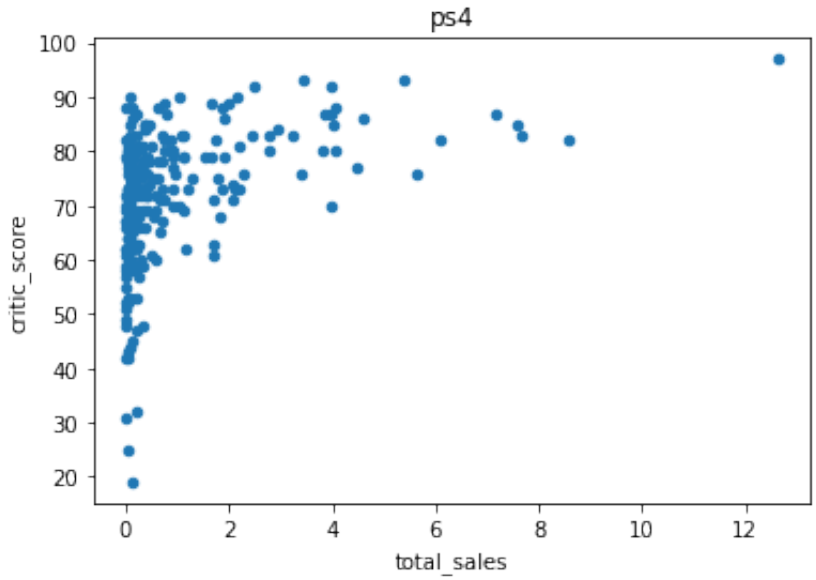
Out[44]:
```
array([[<AxesSubplot:xlabel='year_of_release', ylabel='year_of_release'>,
        <AxesSubplot:xlabel='total_sales', ylabel='year_of_release'>,
        <AxesSubplot:xlabel='critic_score', ylabel='year_of_release'>,
        <AxesSubplot:xlabel='user_score', ylabel='year_of_release'>],
       [<AxesSubplot:xlabel='year_of_release', ylabel='total_sales'>,
        <AxesSubplot:xlabel='total_sales', ylabel='total_sales'>,
        <AxesSubplot:xlabel='critic_score', ylabel='total_sales'>,
        <AxesSubplot:xlabel='user_score', ylabel='total_sales'>],
       [<AxesSubplot:xlabel='year_of_release', ylabel='critic_score'>,
        <AxesSubplot:xlabel='total_sales', ylabel='critic_score'>,
        <AxesSubplot:xlabel='critic_score', ylabel='critic_score'>,
        <AxesSubplot:xlabel='user_score', ylabel='critic_score'>],
       [<AxesSubplot:xlabel='year_of_release', ylabel='user_score'>,
        <AxesSubplot:xlabel='total_sales', ylabel='user_score'>,
        <AxesSubplot:xlabel='critic_score', ylabel='user_score'>,
        <AxesSubplot:xlabel='user_score', ylabel='user_score'>]],
      dtype=object)
```

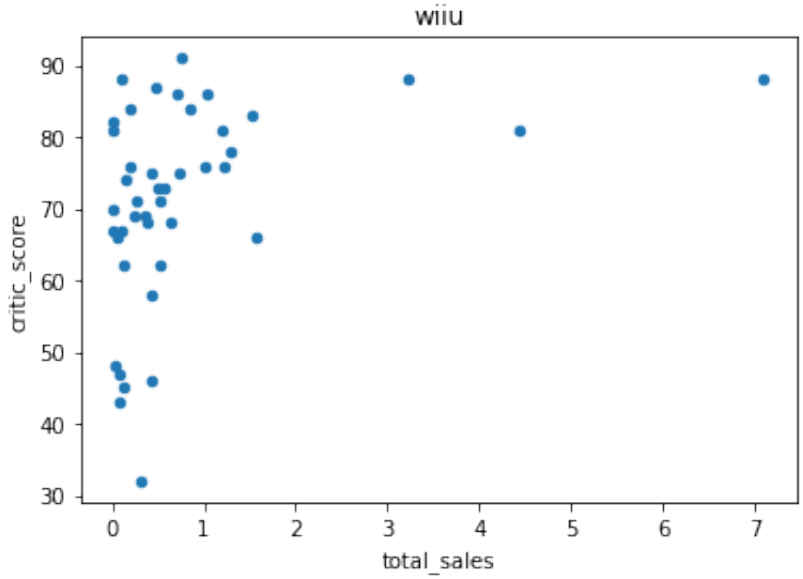


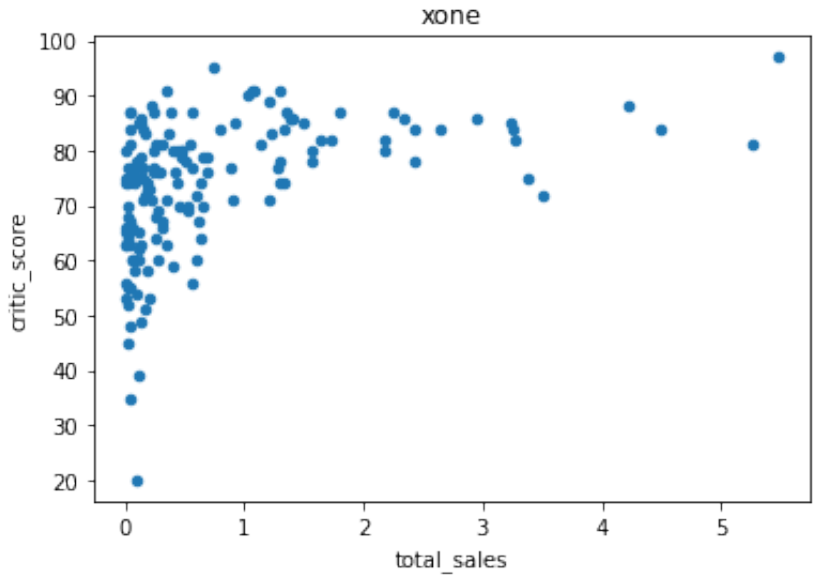
Проверим как связаны параметры total_sales и critics_score для различных платформ

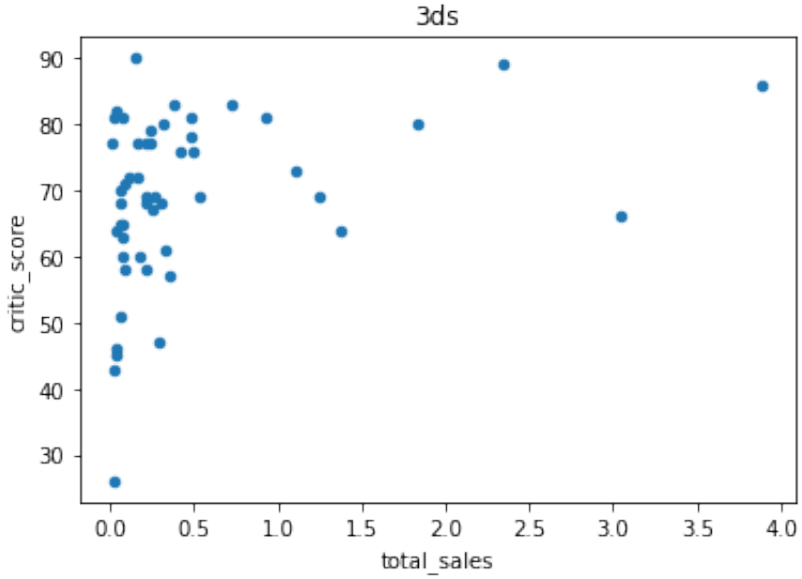
In [45]:
```
def platforms_relations_critics(data):
    for platform in data.query('critic_score != -1')['platform'].unique()
        ax = data.query('platform == @platform').query('critic_score != -
            .plot(kind='scatter', x='total_sales', y='critic_score')
        ax.set_title(platform)
```

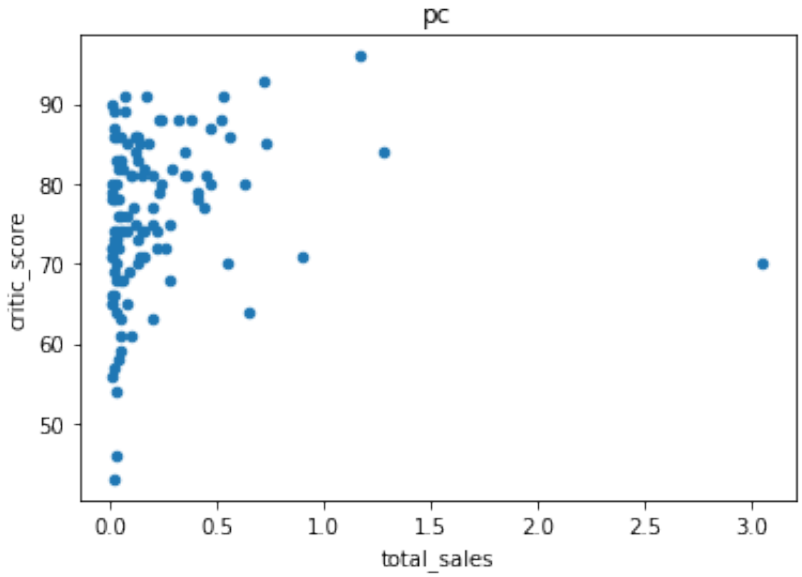
In [46]:
```
platforms_relations_critics(df)
```

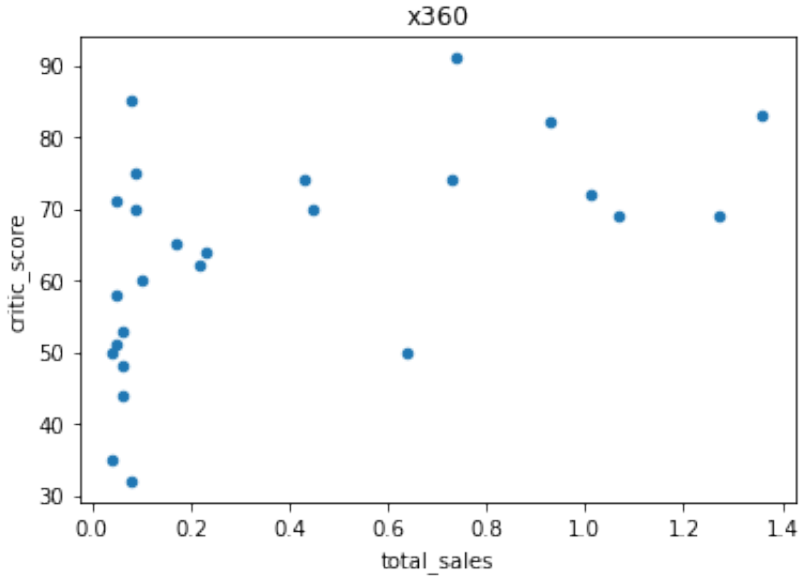


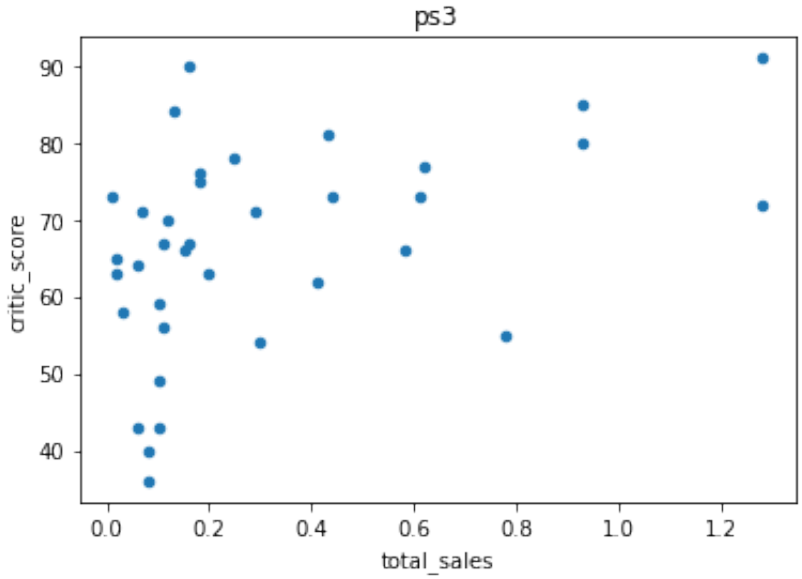


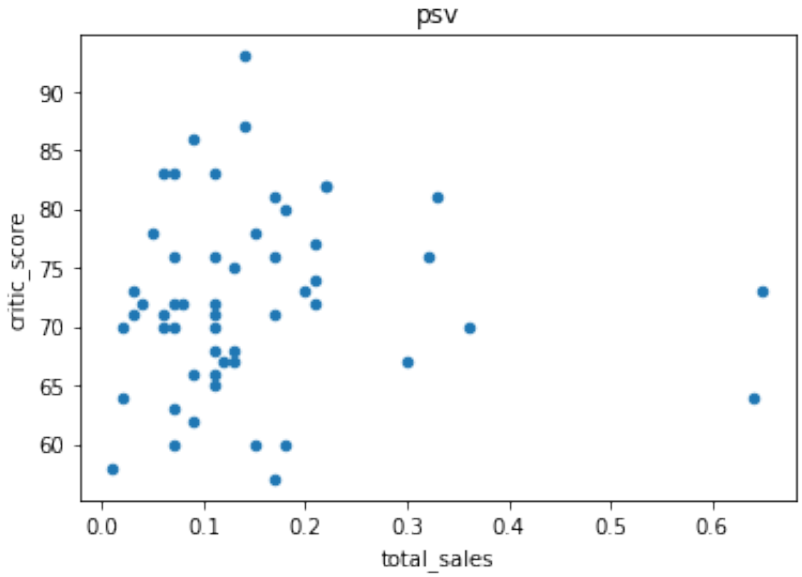



3ds
critic_score
total_sales
90
80
70
60
50
40
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


pc
critic_score
total_sales
90
80
70
60
50
0.0
0.5
1.0
1.5
2.0
2.5
3.0

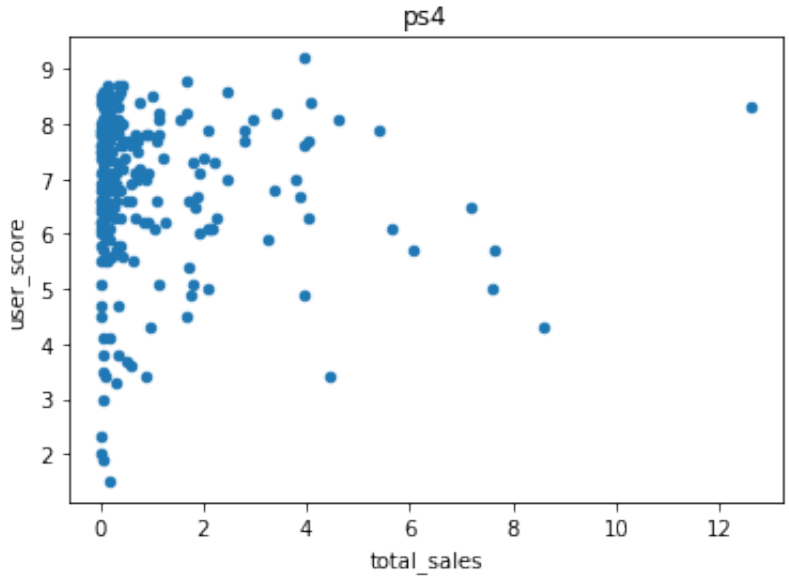

x360
critic_score
total_sales
90
80
70
60
50
40
30
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

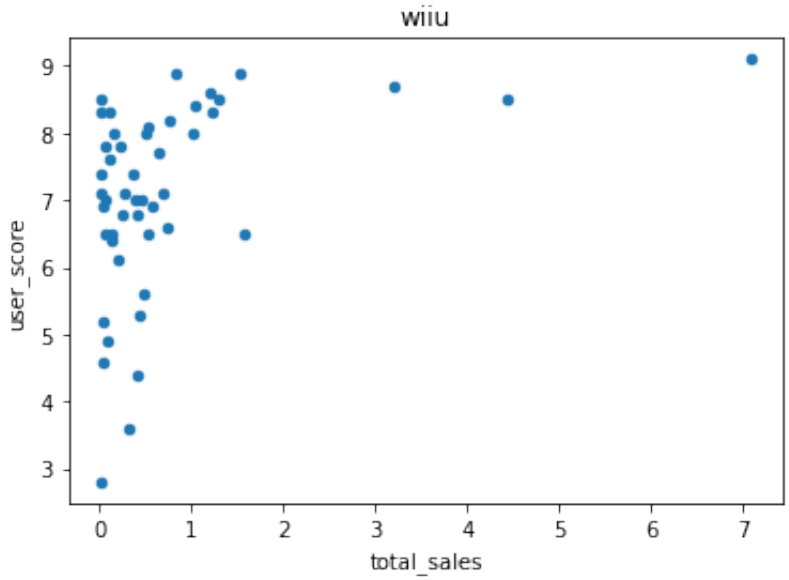


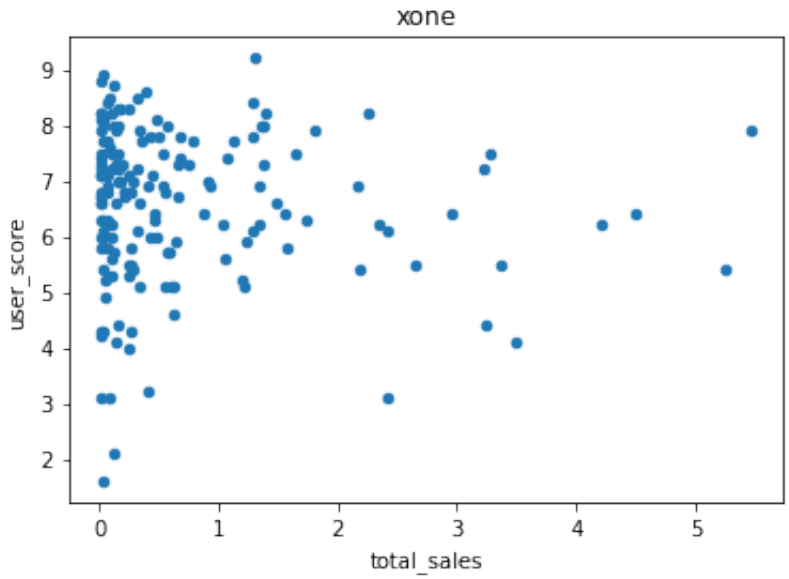


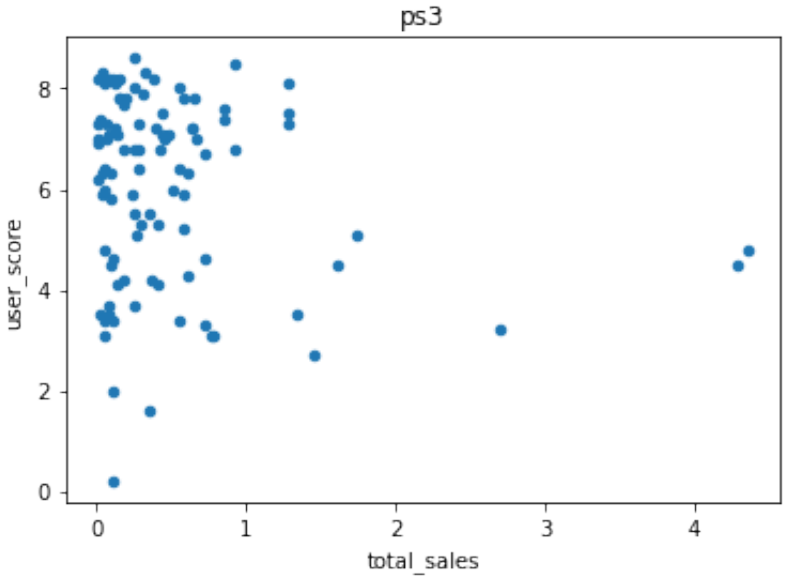
In [47]:
```
def platforms_relations_users(data):
    for platform in data.query('user_score != -1')['platform'].unique():
        ax = data.query('platform == @platform').query('user_score != -1'
            .plot(kind='scatter', x='total_sales', y='user_score')
        ax.set_title(platform)
```

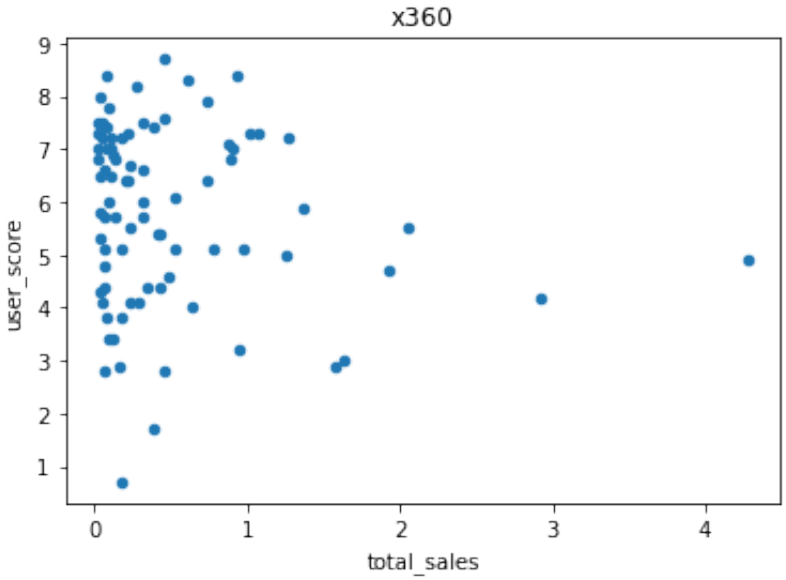
In [48]:
```
platforms_relations_users(df)
```

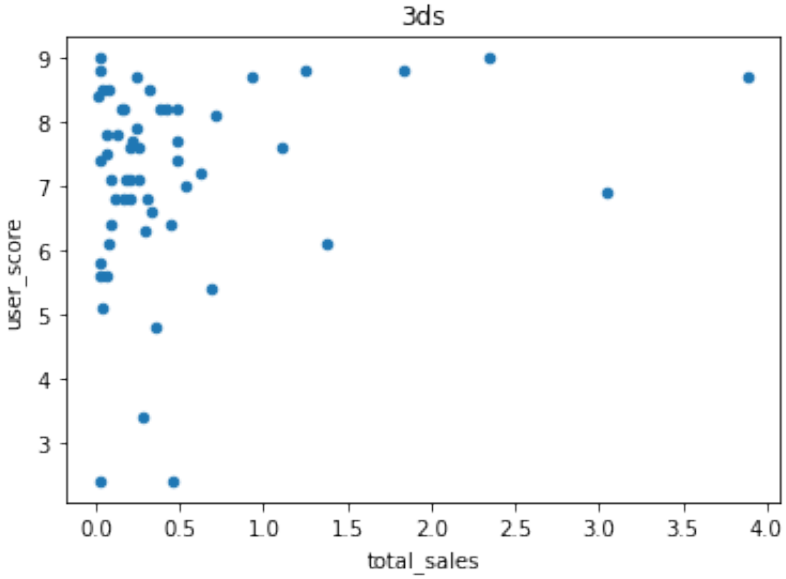

ps4
user_score
total_sales

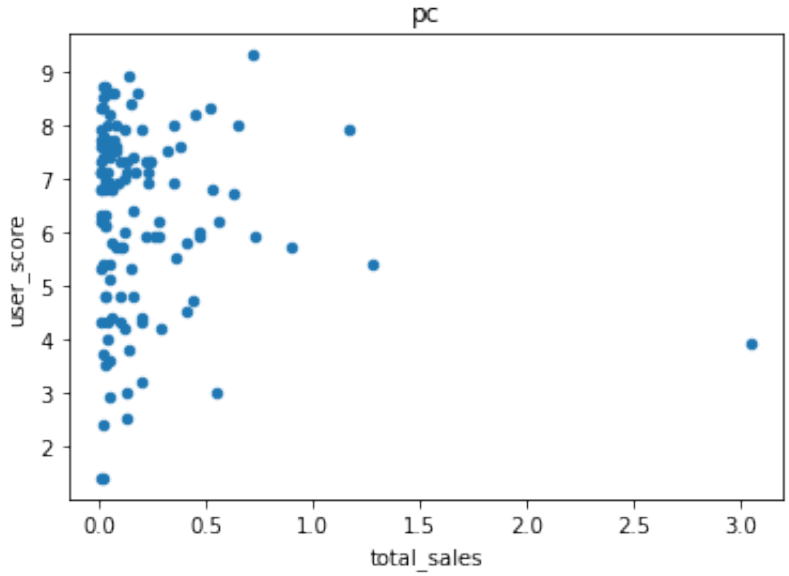

wiiu
user_score
total_sales

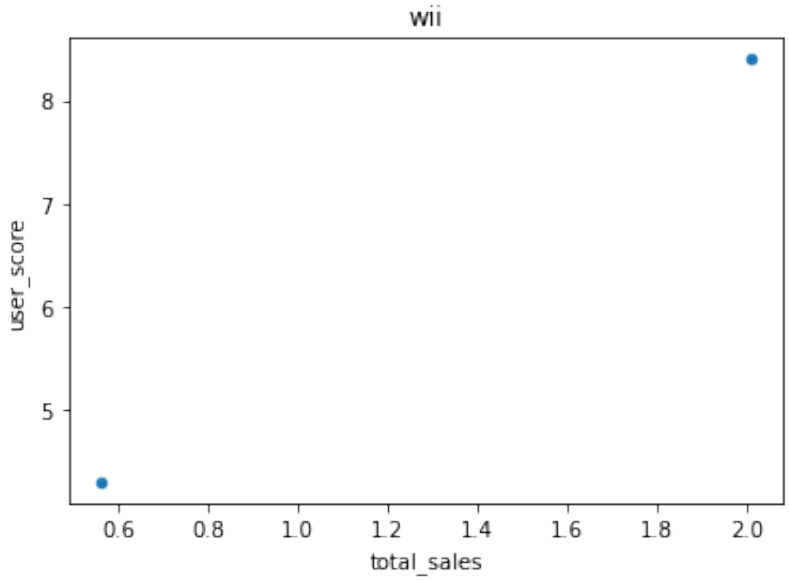

xone
user_score
total_sales

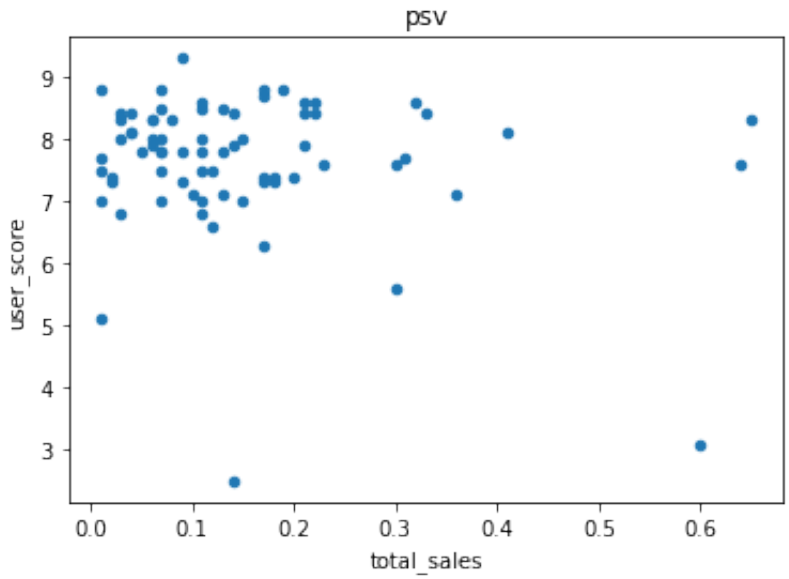

ps3
user_score
total_sales
0
2
4
6
8
0
1
2
3
4


x360
user_score
total_sales
1
2
3
4
5
6
7
8
9
0
1
2
3
4


3ds
user_score
total_sales
3
4
5
6
7
8
9
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


pc
user_score
total_sales
0.0
0.5
1.0
1.5
2.0
2.5
3.0
2
3
4
5
6
7
8
9


wii
user_score
total_sales
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
5
6
7
8

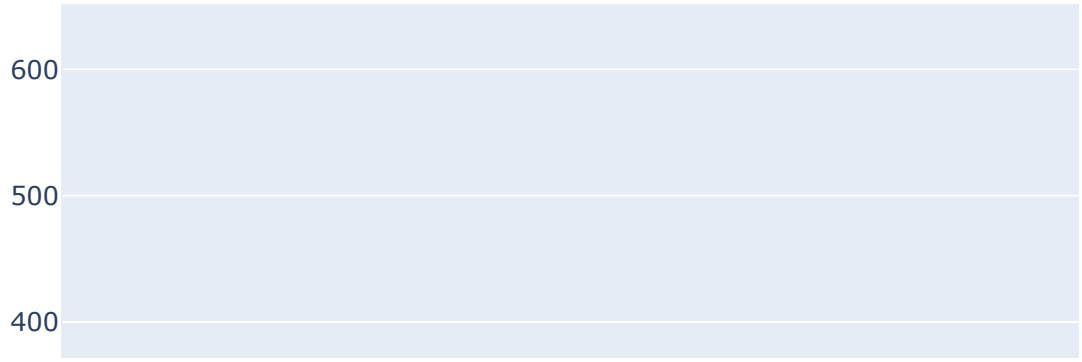

psv
user_score
total_sales
0.0
0.1
0.2
0.3
0.4
0.5
0.6
3
4
5
6
7
8
9

Заметим, что в основном результаты сконцентрированы в верхней левой части графика. При этом игры с оценкой критиков меньше 70 плохо продаются, а остальные могут продаваться хорошо

Существует небольшая корреляция между оценками критиков и общими продажами

Посмотрим на общее распределение игр по жанрам, определим наиболее и наименее прибыльные жанры

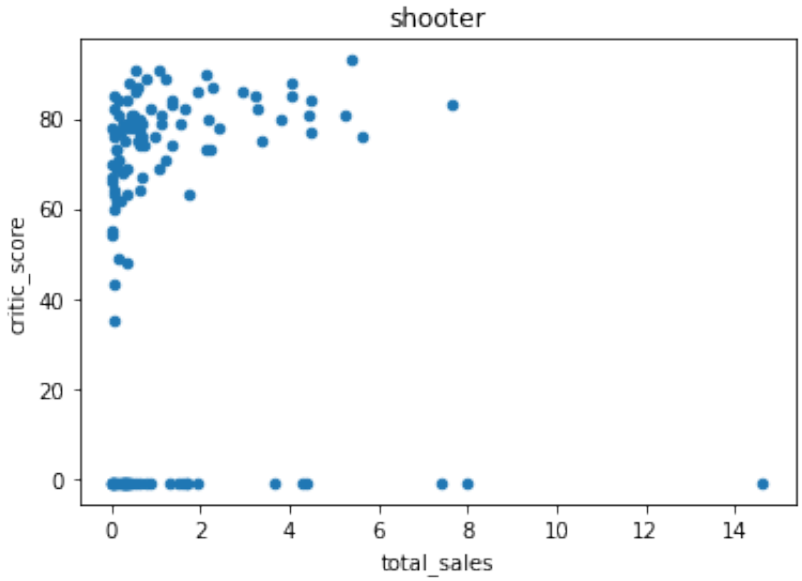
In [49]:
```
ax = df.groupby('genre').agg(func='count')['name'].sort_values()
plx.bar(ax, x=ax.index, y='name', color='name')
```

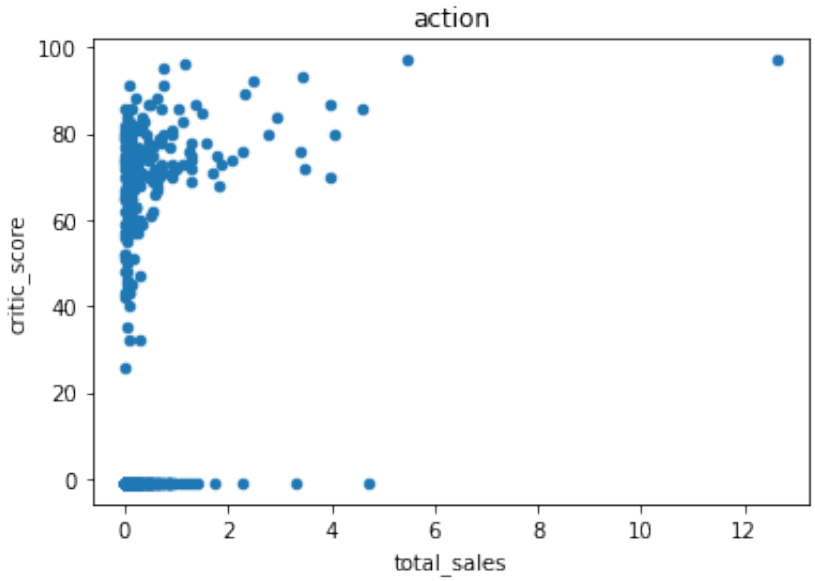


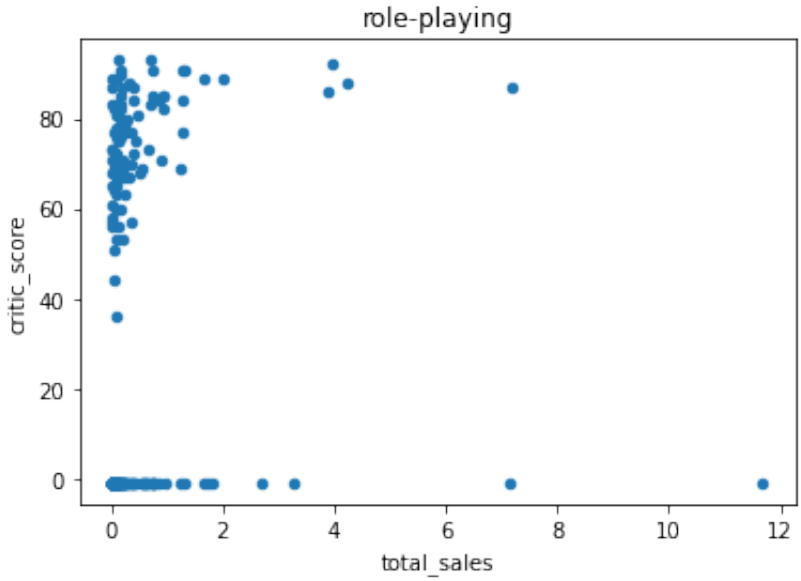
Распределение игр по жанрам

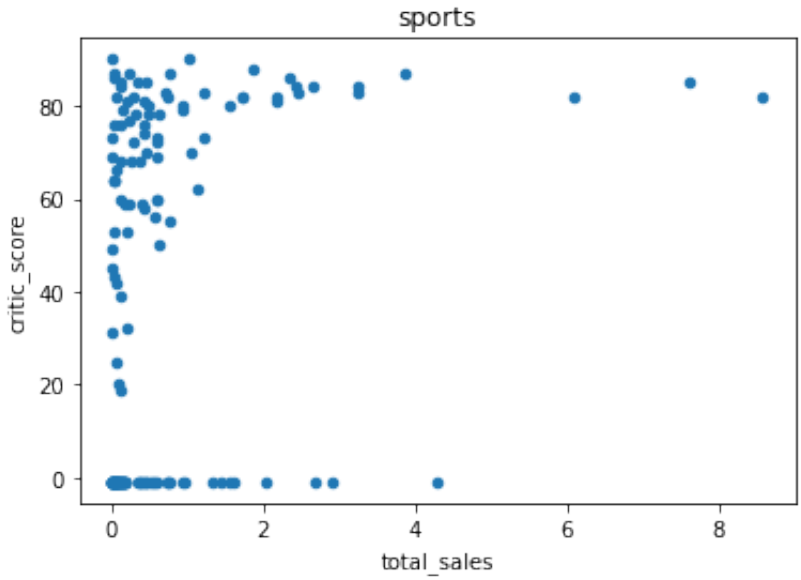
In [50]:
```
def genre_sales(data):
    for genre in data['genre'].unique():
        ax = data.query('genre == @genre')\
            .plot(kind='scatter', x='total_sales', y='critic_score')
        ax.set_title(genre)
    # print(f'Средние продажи для жанра {genre}: {data.query("genre == @g
```

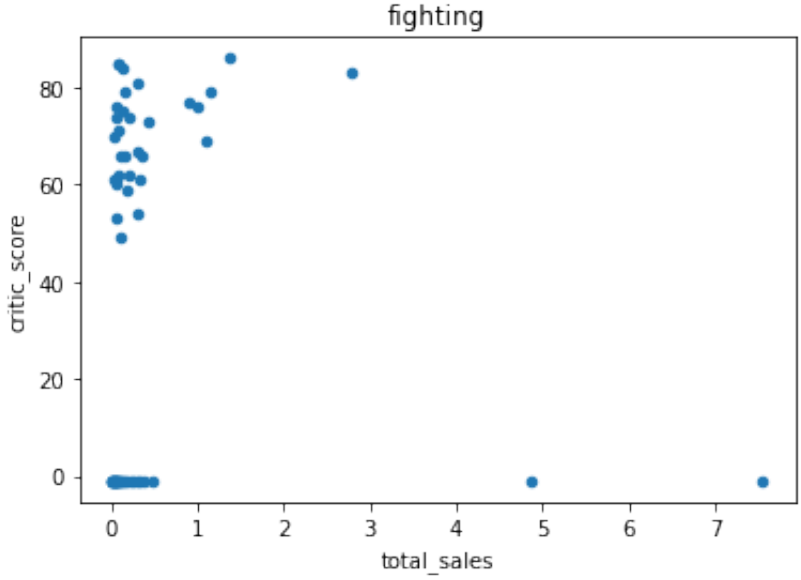
In [51]:
```
genre_sales(df)
```

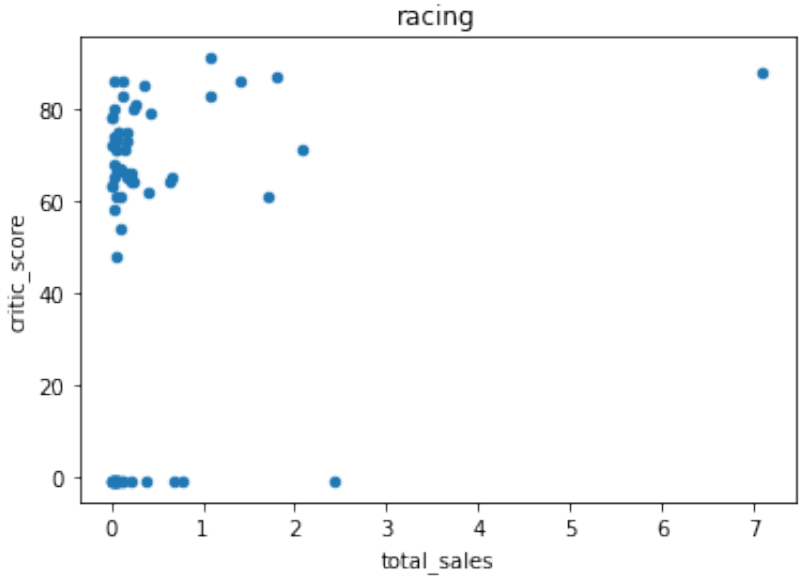


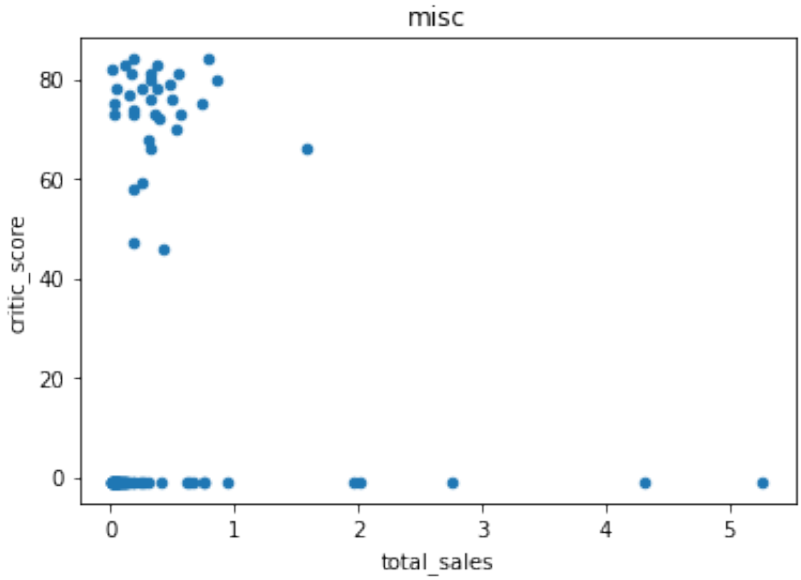


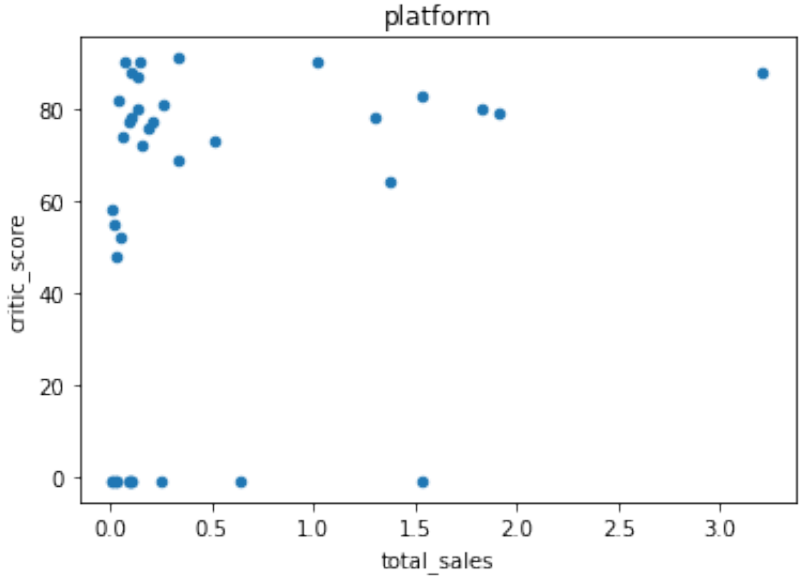

role-playing
critic_score
total_sales
0
20
40
60
80
0
2
4
6
8
10
12

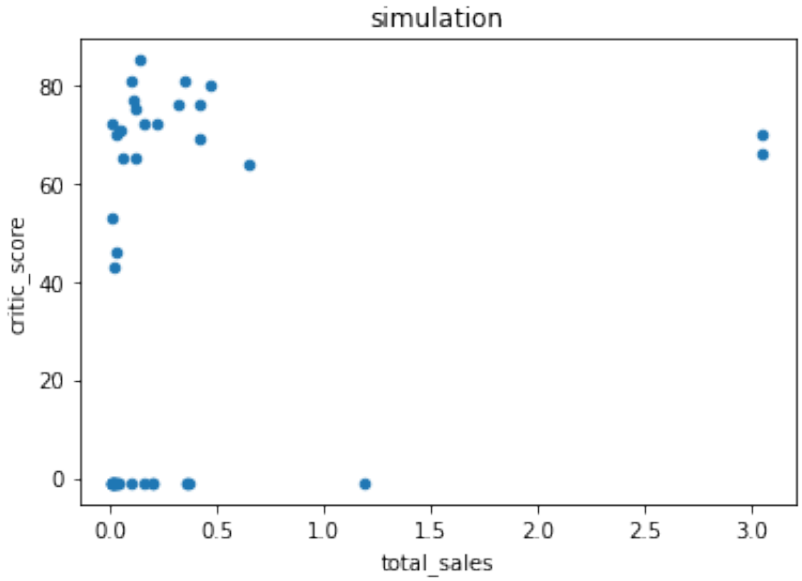

sports
critic_score
total_sales
0
20
40
60
80
0
2
4
6
8

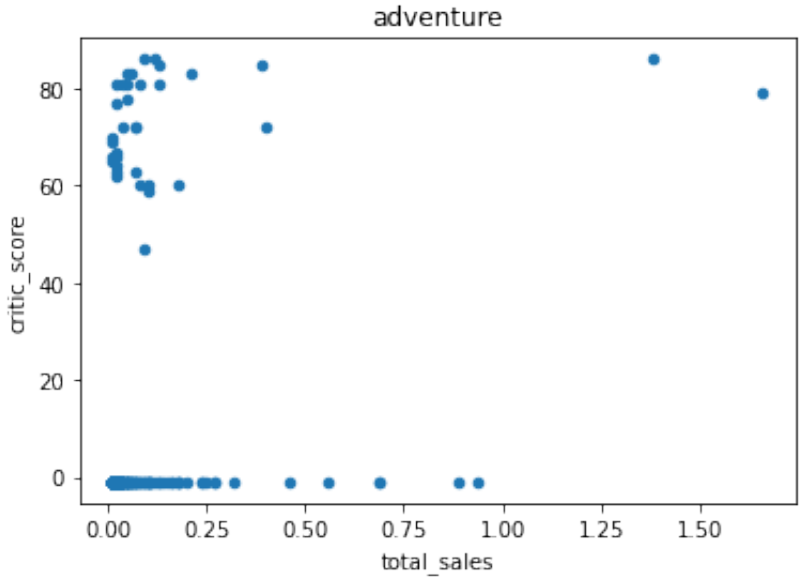

fighting
critic_score
total_sales
0
20
40
60
80
0
1
2
3
4
5
6
7

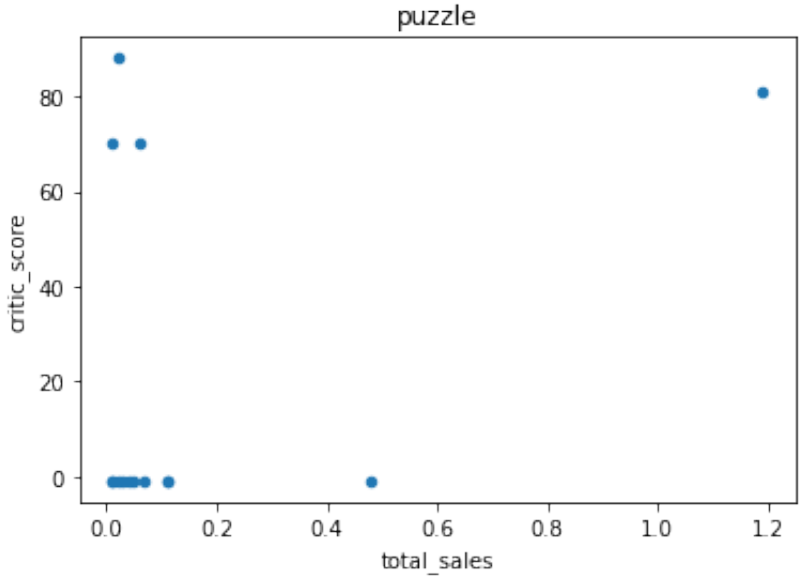

racing
critic_score
total_sales
0
20
40
60
80
0
1
2
3
4
5
6
7

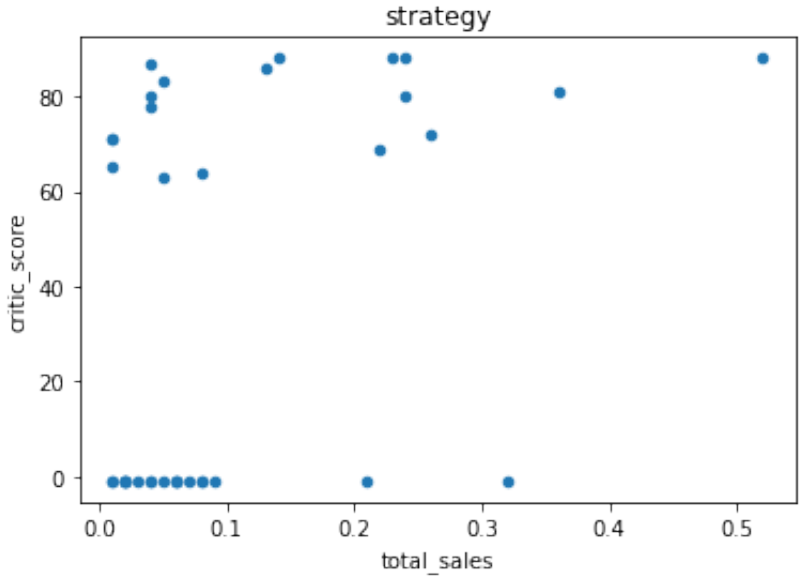

misc
critic_score
total_sales
0
20
40
60
80
0
1
2
3
4
5


platform
critic_score
total_sales
0
20
40
60
80
0.0
0.5
1.0
1.5
2.0
2.5
3.0


simulation
critic_score
total_sales
0
20
40
60
80
0.0
0.5
1.0
1.5
2.0
2.5
3.0


adventure
critic_score
total_sales
0
20
40
60
80
0.00
0.25
0.50
0.75
1.00
1.25
1.50


puzzle
critic_score
total_sales
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
1.2

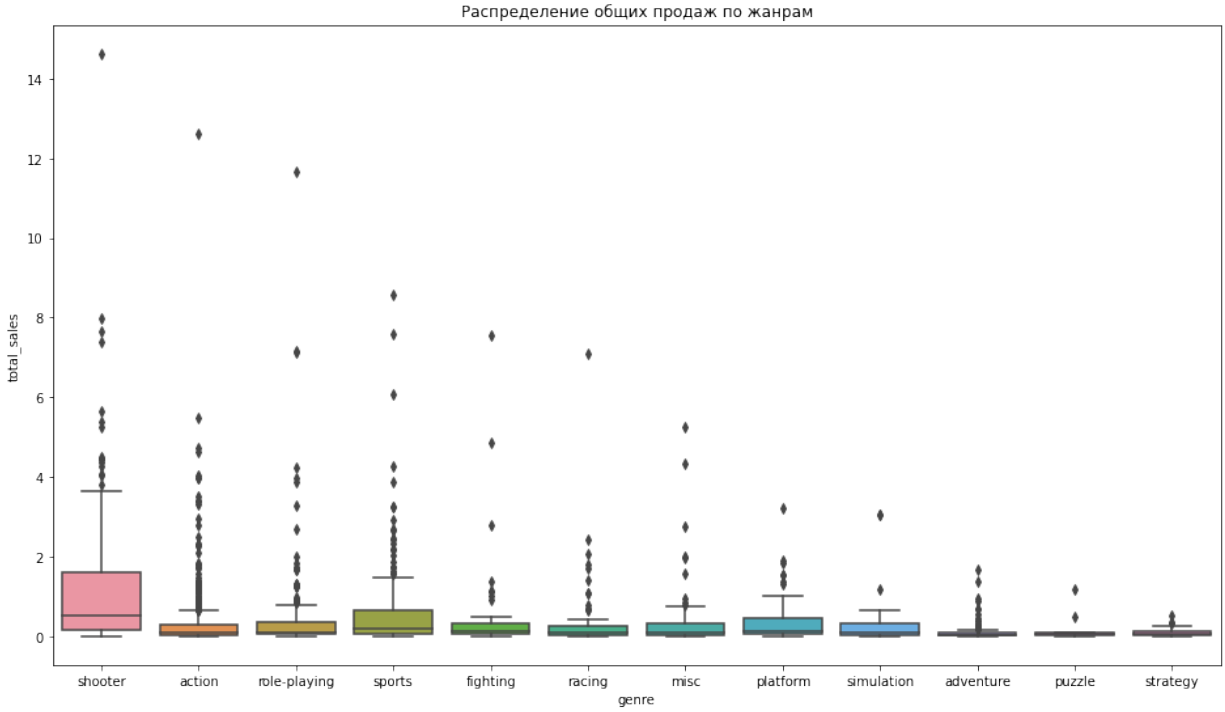


In [52]:
```
# Жанры, сгруппированные по медиане total_sales
print(df.pivot_table(index='genre', values='total_sales', aggfunc='median
          .sort_values(by='total_sales',ascending=False))
```

genre	total_sales
shooter	0.515
sports	0.180
platform	0.140
fighting	0.125
role-playing	0.110
simulation	0.100
action	0.090
misc	0.090
racing	0.090
strategy	0.060
puzzle	0.045
adventure	0.030

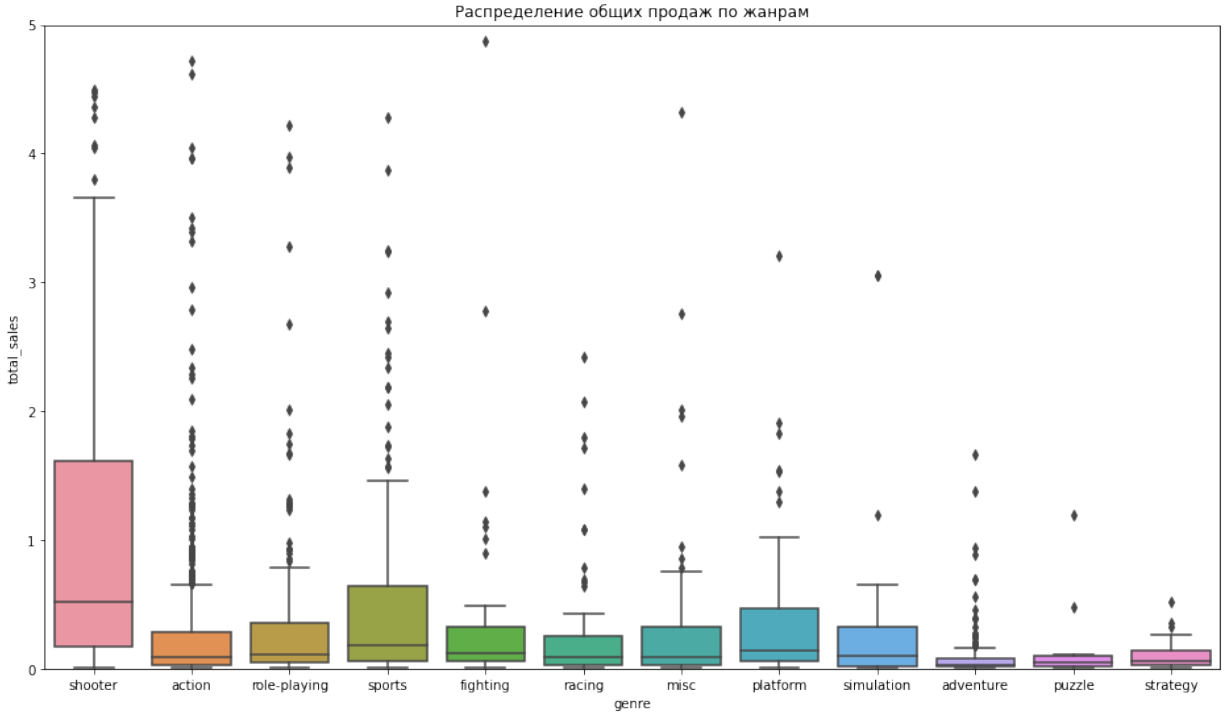
In [53]:
```
plt.figure(figsize=(16,9))
ax = sns.boxplot(data=df, x='genre', y='total_sales')
plt.title('Распределение общих продаж по жанрам')
```

Out[53]:
```
Text(0.5, 1.0, 'Распределение общих продаж по жанрам')
```



In [54]:
```
plt.figure(figsize=(16,9))
ax = sns.boxplot(data=df, x='genre', y='total_sales')
plt.ylim(0,5)
plt.title('Распределение общих продаж по жанрам')
```

Out[54]:
```
Text(0.5, 1.0, 'Распределение общих продаж по жанрам')
```



Вывод по анализу данных:

"Время жизни" платформы в среднем составляет 8 лет. Перспективные платформы в 2017 году: ps4, xone, wiiu. Существует положительная линейная зависимость между оценками критиков и продажами.

Самые прибыльные жанры: shooter, sports, platform
Для них характерно наличие длинного верхнего "уса", а также большого размаха между медианой и 3 квантилем, что говорит о том, что лучшая половина игр может продаваться в больших объемах по сравнению с играми других жанров.

Выделяются жанры с низкими и высокими продажами:

- С высокими: shooter, sports, platform
- С низкими: adventure, puzzle, strategy

Составим портрет пользователя для разных регионов

In [55]:
```
# Я отфильтровал данные с годом релиза < 2014 методом query,
# так что df теперь содержит актуальные данные
df['year_of_release'].min()
```

Out[55]:
```
2014
```

In [56]:
```
# Чтобы найти самые популярные платформы в регионе NA возьмем топ-5 платф
NA_platforms = df.pivot_table(values=['na_sales'], index=['platform'], ag

NA_platforms.head(5)
```

Out[56]:

platform	na_sales
ps4	98.61
xone	81.27
x360	28.30
3ds	22.64
ps3	22.05

Самые популярные платформы в NA регионе: x360, ps3, wii, ds, ps4

In [57]:
```
# Чтобы найти самые популярные платформы в регионе EU возьмем топ-5 платф
EU_platforms = df.pivot_table(values=['eu_sales'], index=['platform'], ag

EU_platforms.head(5)
```

Out[57]:

platform	eu_sales
ps4	130.04
xone	46.25
ps3	25.54
pc	17.97
3ds	16.12

Самые популярные платформы в EU регионе: ps3, x360, wii, ps4, pc

In [58]:
```
# Чтобы найти самые популярные платформы в регионе JP возьмем топ-5 платф
JP_platforms = df.pivot_table(values=['jp_sales'], index=['platform'], ag

JP_platforms.head(5)
```

Out[58]:

platform	jp_sales
3ds	44.24
ps4	15.02
psv	14.54
ps3	11.22
wiiu	7.31

Самые популярные платформы в JP регионе: 3ds, ds, ps3, psp, wii

Самые популярные жанры

In [59]:
```
# Напишем функцию для замены жанров не из топ-5 на other
def genre_replace(data, subset):
    for i in range(len(data)):
        if data.iloc[i, 3] not in subset:
            data.iloc[i, 3] = 'other'
    return data
```

In [60]:
```
# Получим сводные таблицы с топ-5 жанров для всех регионов
def pivots(data):
    region_pivots = []
    for region_sales in ['na_sales', 'eu_sales', 'jp_sales']:
        t = data.pivot_table(values=region_sales, index=['genre'], aggfun

        df_copy = genre_replace(data.copy(), list(t.head(6).index))
        region_pivots.append(df_copy.pivot_table(values=region_sales, ind

    return region_pivots
```

In [61]:
```
NA_genres, EU_genres, JP_genres = pivots(df)
```

In [62]:
```
NA_genres
```

Out[62]:

genre	na_sales
shooter	79.02
action	72.53
sports	46.13
role-playing	33.47
other	25.43
misc	15.05
fighting	12.43

Самые популярные жанры в NA регионе: action, shooter, sports, misc, role-playing

In [63]:
```
EU_genres
```

Out[63]:

genre	eu_sales
action	74.68
shooter	65.52
sports	45.73
other	29.59
role-playing	28.17
racing	14.13
misc	12.86

Самые популярные жанры в EU регионе: action, shooter, sports, misc, role-playing

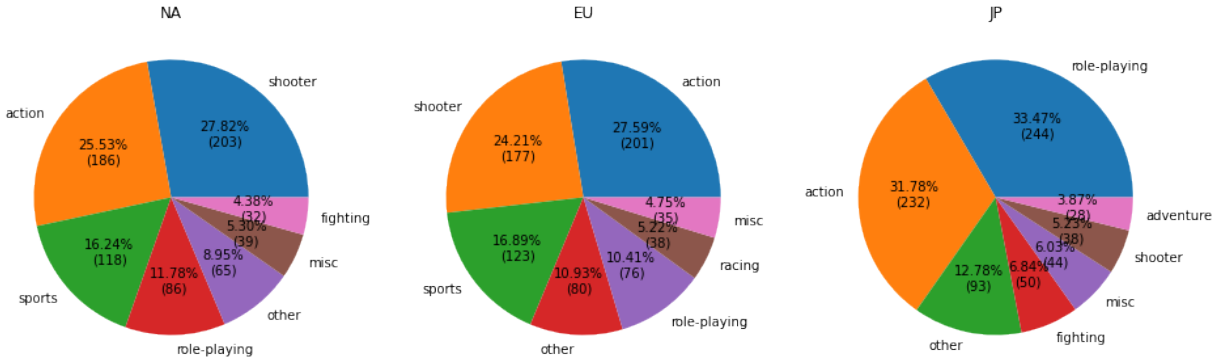
In [64]:
```
JP_genres
```

Out[64]:

genre	jp_sales
role-playing	31.16
action	29.58
other	11.90
fighting	6.37
misc	5.61
shooter	4.87
adventure	3.60

Самые популярные жанры в JP регионе: role-playing, action, misc, sports, adventure

In [65]:
```
total = np.sum(df['total_sales'])
def val_per(x):
    return '{:.2f}%\n({:.0f})'.format(x, total*x/100)


plt.figure(figsize=(16, 9))
plt.subplot(1,3,1)
plt.pie(NA_genres['na_sales'], labels=NA_genres.index, autopct=val_per)
plt.gca().set(title='NA')


plt.subplot(1,3,2)
plt.pie(EU_genres['eu_sales'], labels=EU_genres.index, autopct=val_per)

plt.gca().set(title='EU')

plt.subplot(1,3,3)
plt.pie(JP_genres['jp_sales'], labels=JP_genres.index, autopct=val_per)
plt.gca().set(title='JP')


plt.show()
```



Распределение наиболее популярных жанров в регионах

Оценим влияние ESRB на продажи

In [66]:
```
ESRB_sales_na = df.pivot_table(index='rating', values='na_sales', aggfunc

ESRB_sales_na
```

Out[66]:

rating	na_sales
M	96.42
unknown	64.72
E	50.74
T	38.95
E10+	33.23

In [67]:
```
ESRB_sales_eu = df.pivot_table(index='rating', values='eu_sales', aggfunc

ESRB_sales_eu
```

Out[67]:

rating	eu_sales
M	93.44
unknown	58.95
E	58.06
T	34.07
E10+	26.16

In [68]:
```
ESRB_sales_jp = df.pivot_table(index='rating', values='jp_sales', aggfunc

ESRB_sales_jp
```

Out[68]:

rating	jp_sales
unknown	56.90
T	14.78
E	8.94
M	8.01
E10+	4.46

Вывод по продажам: Жанры E, M, T лидируют во всех регионах, меняясь местами в зависимости от региона.

Вывод по портрету пользователя:

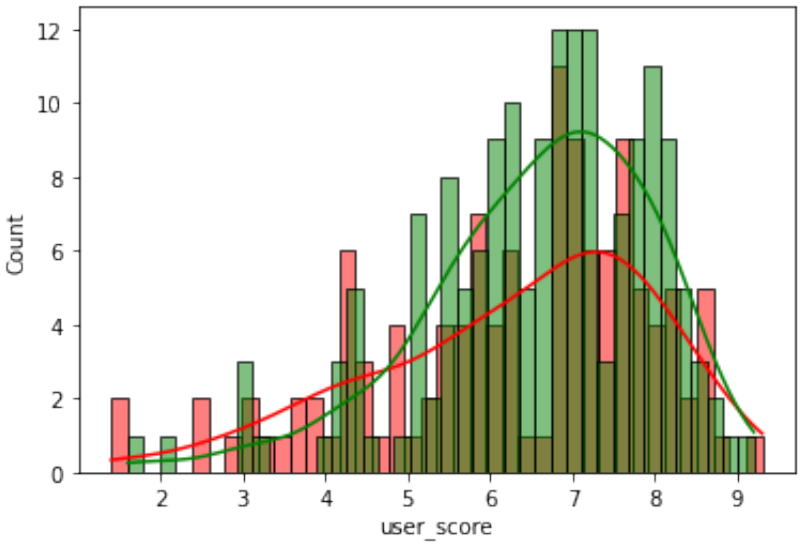
- Жанры action, sports, misc, role-playing популярны во всех регионах
- Игры в жанрах E, M, T продаются лучше
- Платформы значительно отличаются в зависимости от региона

Проверка статистических гипотез

In [69]:
```
from scipy import stats as sps
```

In [70]:
```
# Средние пользовательские рейтинги платформ Xbox One и PC одинаковые?
sample_pc = df.query('platform == "pc"').query('user_score != "tbd"')\
    .query('user_score != -1')['user_score'].astype(float)
sample_xone = df.query('platform == "xone"').query('user_score != "tbd"')
    .query('user_score != -1')['user_score'].astype(float)
sns.histplot(sample_pc, bins=40, kde=True, color='r')
sns.histplot(sample_xone, bins=40, kde=True, color='g')
```

Out[70]:
```
<AxesSubplot:xlabel='user_score', ylabel='Count'>
```



Сформулируем гипотезы на основе вида распределения
Заметим, что мода распределения оценок на xone смещена вправо относительно pc, поэтому рассмотрим одностороннюю гипотезу
H0: xone_mean == pc_mean
H1: xone_mean != pc_mean

In [71]:
```
sps.ttest_ind(sample_pc, sample_xone, equal_var=False)
#ttest_ind применен, поскольку проверяем равенство средних двух генеральн
#equal_var=False, тк нет оснований считать что дисперсии равны
```

Out[71]:
```
Ttest_indResult(statistic=-1.577760647447497, pvalue=0.11601398086668832)
```

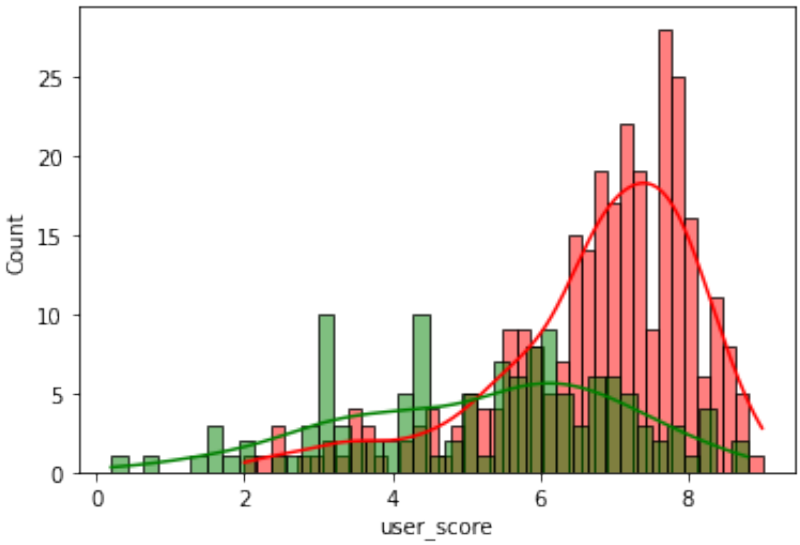
In [72]:
```
pvalue=0.11601398086668832
alpha = .05
if pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не отвергаем нулевую гипотезу')
```

```
Не отвергаем нулевую гипотезу
```

Вывод: На уровне значимости 0.05 не можем отвергнуть нулевую гипотезу

In [73]:
```
# Средние пользовательские рейтинги жанров Action (англ. «действие», экше
sample_action = df.query('genre == "action"').query('user_score != "tbd"'
    .query('user_score != -1')['user_score'].astype(float)
sample_sports = df.query('genre == "sports"').query('user_score != "tbd"'
    .query('user_score != -1')['user_score'].astype(float)
sns.histplot(sample_action, bins=40, kde=True, color='r')
sns.histplot(sample_sports, bins=40, kde=True, color='g')
```

Out[73]:
```
<AxesSubplot:xlabel='user_score', ylabel='Count'>
```



H0: sports_mean == action_mean

H1: sports_mean != action_mean

In [74]:
```
sps.ttest_ind(sample_pc, sample_sports, equal_var=False)
```

Out[74]:
```
Ttest_indResult(statistic=4.755709648522259, pvalue=3.367256530792755e-06
)
```

In [75]:
```
pvalue=3.367256530792755e-06
alpha = .05
if pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не отвергаем нулевую гипотезу')
```

```
Отвергаем нулевую гипотезу
```

Вывод: На уровне значимости 0.05 можем отвергнуть нулевую гипотезу

Общий вывод

Предобработка:

- Названия столбцов и записи в текстовых столбцах приведены к нижнему регистру
- Строк с пропусками в столбцах 'year_of_release', 'genre' мало, поэтому они были удалены
- Пропуски в столбцах critic_score, user_score заполнены значением -1
- Пропуски в столбце rating заполнены значением 'unknown'
- Явные дубликаты не найдены
- Добавлен столбец total_sales, который содержит сумму продаж по всем регионам
- Столбец year_of_release приведен к типу int, тк. изначально содержал

значения с плавающей точкой без дробной части
- Значение tbd заменено на -1, столбец user_score приведен к типу float

Исследовательский анализ данных:

- "Время жизни" платформы в среднем составляет 8 лет.
- Перспективные платформы в 2017 году: ps4, xone, wiiu.
- Существует положительная линейная зависимость между оценками критиков и продажами.

Составление портрета пользователя:

- Жанры action, sports, misc, role-playing популярны во всех регионах
- Игры в жанрах E, M, T продаются лучше
- Платформы значительно отличаются в зависимости от региона

Статистический анализ данных:

На p=0.05 уровне значимости:

- гипотезу о равенстве средних оценок пользователей на платформах pc, xone не получилось отвергнуть
- гипотезу о равенстве средних оценок пользователей для жанров action, sports получилось отвергнуть. Средние оценки не равны

Рекомендация по составлению рекламной компании:

Лучше всего направить усилия на платформы: ps4, xone, wiiu. Хорошо продаются игры в жанрах action, sports, misc, role-playing с рейтингом ESRB - E, M, T. Также стоит учесть, что в разных регионах популярные платформы отличаются, а состав топ-5 жанров хоть и остается без изменений, но позиции меняются. Игры с высокими отзывами критиков продаются лучше. Отзывы пользователей не влияют на продажи.