Исследование данных о продажах игр

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».

Цель исследования:

```
1) Изучить изменения продаж игр;
2) Составить портер пользователя;
3) Определить и проверить гипотезы.
```

Ход исследования: 1) Для начала ознакомимся с данными, проверим их содержание и качество; 2) Подготовим данные к работе обработав пропуски, дубликаты, тип данных и пр.; 3) Проведём исследовательский анализ данных, посмотрим как коррелируют признаки с продажами; 4) Составим портер пользователя для каждого региона; 5) Проверим гипотезы с помощью т-теста, предварительно проверив выборки на схожесть; 6) Подготовим вывод и выведем рекомендации.

Ознакомление с данными

Импорт библиотек

In [1]:
```
import pandas as pd
import datetime as dt
import numpy as np
import scipy.stats as st
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
```

```
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

Открытие данных

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

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

```
<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
```

Out[3]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sa
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1

Описание данных

- Name — название игры
- Platform — платформа
- Year_of_Release — год выпуска
- Genre — жанр игры
- NA_sales — продажи в Северной Америке

- EU_sales — продажи в Европе
- JP_sales — продажи в Японии
- Other_sales — продажи в других странах
- Critic_Score — оценка критиков
- User_Score — оценка пользователей
- Rating — рейтинг от организации ESRB (англ. Entertainment Software Rating Board). Эта ассоциация определяет рейтинг компьютерных игр и присваивает им подходящую возрастную категорию.

По итогам общего анализа:

- Названия столбцов начинаются с верхнего регистра
- Поля с пропущенными данными: name, year_of_release, genre, critic-score, user_score, rating
- Тип данных неверен в: year_of_release, user_score

Подготовка данных

- ☑ Замените названия столбцов (приведите к нижнему регистру);
- ☑ Преобразуйте данные в нужные типы. Опишите, в каких столбцах заменили тип данных и почему;
- ☑ Обработайте пропуски при необходимости:
- ☑ Объясните, почему заполнили пропуски определённым образом или почему не стали это делать;
- ☑ Опишите причины, которые могли привести к пропускам;
- ☑ Обратите внимание на аббревиатуру 'tbd' в столбце с оценкой пользователей. Отдельно разберите это значение и опишите, как его обработать;
- ☑ Посчитайте суммарные продажи во всех регионах и запишите их в отдельный столбец.

Корректируем названия столбцов

In [4]:
```
df.columns = df.columns.str.lower()
#df.columns
```

Займёмся обработкой пропусков и преобразованием типов данных.

Предположим, что пропуски возникли из-за некачественного сбора данных или из-за их отсутствия.

Проверка на дубликаты

In [5]:
```
print("Кол-во повторяющихся строк в датафрейме:", df.duplicated().sum())
```

```
Кол-во повторяющихся строк в датафрейме: 0
```

In [6]:
```
duplicated = df[df.duplicated(subset=['platform', 'name', 'genre'])]
duplicated
```

Out[6]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sa
1591	Need for Speed: Most Wanted	X360	2005.0	Racing	1.0	0.13	0.02	0
4127	Sonic the Hedgehog	PS3	NaN	Platform	0.0	0.48	0.00	0
11715	Need for Speed: Most Wanted	PC	2012.0	Racing	0.0	0.06	0.00	0
14244	NaN	GEN	1993.0	NaN	0.0	0.00	0.03	0
16230	Madden NFL 13	PS3	2012.0	Sports	0.0	0.01	0.00	0

In [7]:
```
df[df['name'] == 'Need for Speed: Most Wanted']
```

Out[7]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales
253	Need for Speed: Most Wanted	PS2	2005.0	Racing	2.03	1.79	0.08	0.47
523	Need for Speed: Most Wanted	PS3	2012.0	Racing	0.71	1.46	0.06	0.58
1190	Need for Speed: Most Wanted	X360	2012.0	Racing	0.62	0.78	0.01	0.15
1591	Need for Speed: Most Wanted	X360	2005.0	Racing	1.00	0.13	0.02	0.10
1998	Need for Speed: Most Wanted	XB	2005.0	Racing	0.53	0.46	0.00	0.05
2048	Need for Speed: Most Wanted	PSV	2012.0	Racing	0.33	0.45	0.01	0.22
3581	Need for Speed: Most Wanted	GC	2005.0	Racing	0.43	0.11	0.00	0.02
5972	Need for Speed: Most Wanted	PC	2005.0	Racing	0.02	0.23	0.00	0.04
6273	Need for Speed: Most Wanted	WiiU	2013.0	Racing	0.13	0.12	0.00	0.02
6410	Need for	DS	2005.0	Racing	0.24	0.01	0.00	0.02

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales
	Speed: Most Wanted							
6473	Need for Speed: Most Wanted	GBA	2005.0	Racing	0.19	0.07	0.00	0.00
11715	Need for Speed: Most Wanted	PC	2012.0	Racing	0.00	0.06	0.00	0.02

Дублика игры Need for Speed: Most Wanted по платформам PC и X360 обоснован тем, что это две разные версии игры с одним наименованием.

In [8]:
```
df[df['name'] == 'Sonic the Hedgehog']
```

Out[8]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sal
257	Sonic the Hedgehog	GEN	1991.0	Platform	3.03	0.91	0.26	0.1
1745	Sonic the Hedgehog	PS3	2006.0	Platform	0.41	0.06	0.04	0.6
1996	Sonic the Hedgehog	X360	2006.0	Platform	0.44	0.48	0.00	0.1
4127	Sonic the Hedgehog	PS3	NaN	Platform	0.00	0.48	0.00	0.0

Дублика игры Sonic the Hedgehog по платформе PS3 можно обосновать ошибкой в сборе данных, стоит оставить только первый варант, т.к. он более полон данными.

In [9]:
```
df[df['name'] == 'Madden NFL 13']
```

Out[9]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales
507	Madden NFL 13	X360	2012.0	Sports	2.53	0.15	0.0	0.17
604	Madden NFL 13	PS3	2012.0	Sports	2.11	0.22	0.0	0.23
3986	Madden NFL 13	Wii	2012.0	Sports	0.47	0.00	0.0	0.03
5887	Madden NFL 13	PSV	2012.0	Sports	0.28	0.00	0.0	0.02
7066	Madden NFL 13	WiiU	2012.0	Sports	0.21	0.00	0.0	0.02
16230	Madden NFL 13	PS3	2012.0	Sports	0.00	0.01	0.0	0.00

С хокеем стоит поступить также как и с Соником.

In [10]:
```
df = df.drop_duplicates(subset=['platform', 'name', 'genre', 'year_of_release'])
df
```

Out[10]:

	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	
...	...	...	...	...	...	...	...	
16710	Samurai Warriors: Sanada Maru	PS3	2016.0	Action	0.00	0.00	0.01	
16711	LMA Manager 2007	X360	2006.0	Sports	0.00	0.01	0.00	
16712	Haitaka no Psychedelica	PSV	2016.0	Adventure	0.00	0.00	0.01	
16713	Spirits & Spells	GBA	2003.0	Platform	0.01	0.00	0.00	
16714	Winning Post 8 2016	PSV	2016.0	Simulation	0.00	0.00	0.01	

16713 rows × 11 columns

Обработка пропусков

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
name                1
platform            0
year_of_release     269
genre               1
na_sales            0
eu_sales            0
jp_sales            0
other_sales         0
critic_score        8577
user_score          6700
rating              6765
dtype: int64
```

В столбце "year_of_release"

Преобразуем тип данных в столбце year_of_release с float на int, заполнив пропуски 0.

In [12]:
```
df['year_of_release'] = df['year_of_release'].fillna(0).astype('int')

#Посмотрим на игры с 0 датой выпуска
df[df['year_of_release'] == 0].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 269 entries, 183 to 16522
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             269 non-null    object 
 1   platform         269 non-null    object 
 2   year_of_release  269 non-null    int64  
 3   genre            269 non-null    object 
 4   na_sales         269 non-null    float64
 5   eu_sales         269 non-null    float64
 6   jp_sales         269 non-null    float64
 7   other_sales      269 non-null    float64
 8   critic_score     154 non-null    float64
 9   user_score       175 non-null    object 
 10  rating           181 non-null    object 
dtypes: float64(5), int64(1), object(5)
memory usage: 25.2+ KB
```

Остаётся 269 строк с нулевым годом выпуска, данные строки можно попробовать заполнить, удалить или просто не учитывать в анализе.

In [13]:
```
df[df['year_of_release'] == 0].head()
```

Out[13]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sale
183	Madden NFL 2004	PS2	0	Sports	4.26	0.26	0.01	0.7
377	FIFA Soccer 2004	PS2	0	Sports	0.59	2.36	0.04	0.5
456	LEGO Batman: The Videogame	Wii	0	Action	1.80	0.97	0.00	0.2
475	wwe Smackdown vs. Raw 2006	PS2	0	Fighting	1.57	1.02	0.00	0.4
609	Space Invaders	2600	0	Shooter	2.36	0.14	0.00	0.0

Посмотрим на год выпуска одной из игр на других платформах

In [14]:
```
df[df['name'] == 'Madden NFL 2004'].head()
```

Out[14]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales
183	Madden NFL 2004	PS2	0	Sports	4.26	0.26	0.01	0.71
1881	Madden NFL 2004	XB	2003	Sports	1.02	0.02	0.00	0.05
3889	Madden NFL 2004	GC	2003	Sports	0.40	0.10	0.00	0.01
5708	Madden NFL 2004	GBA	2003	Sports	0.22	0.08	0.00	0.01

Заполним нулевые даты, датами с других платформ

In [15]:
```
def year_ (x):
    year = x['year_of_release']
    if year == 0:
        year = b.loc[x['name'],'year_of_release']
        return year
    else:
        return x['year_of_release']

b = df.pivot_table(index='name', values='year_of_release', aggfunc='max')
df['year_of_release'] = df.apply(year_, axis=1).astype('int64')
```

In [16]:
```
print('Кол-во игр с неизвестными датами выпуска', len(df[df['year_of_release'] == 0
```

```
Кол-во игр с неизвестными датами выпуска 133
```

Строк с неизвестными датами выпуска 133 - удалаяем.

In [17]:
```
print('Доля удаляемых значений', ((len(df[df['year_of_release'] == 0]['name'].uniqu
```

```
Доля удаляемых значений 0.7957877101657393 %
```

In [18]:
```
df = df[df['year_of_release'] != 0]
print('Кол-во игр с неизвестными датами выпуска', len(df[df['year_of_release'] == 0
```

```
Кол-во игр с неизвестными датами выпуска 0
```

В столбце "rating"

Проверим содержание столбца rating

In [19]:
```
print('Кол-во пропущенных значений в rating:', df['rating'].isnull().sum())
df['rating'].value_counts()
```

```
Кол-во пропущенных значений в rating: 6700
```

Out[19]:
```
E       3957
T       2930
M       1554
E10+    1412
EC         8
K-A        3
RP         2
AO         1
Name: rating, dtype: int64
```

Согласно открытым источникам:

- To be determined or To be decided (TBD) TBD is an acronym that means to be determined or to be decided.

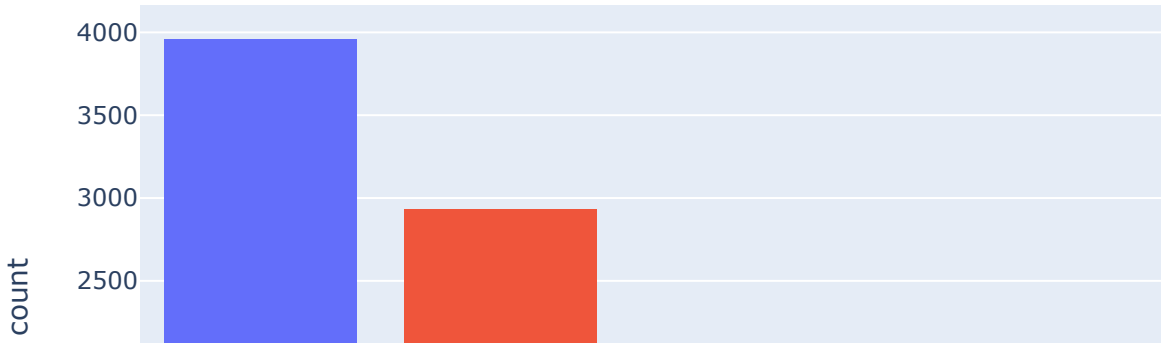
Проверим кол-во каждого рейтинга.

In [20]:
```
rating_value = df.groupby('rating')['na_sales'].count().reset_index()

rating_value.columns = ['rating', 'count']

rating_value = rating_value.sort_values('count', ascending = False)

rating_value
```

Out[20]:

	rating	count
1	E	3957
7	T	2930
5	M	1554
2	E10+	1412
3	EC	8
4	K-A	3
6	RP	2
0	AO	1

In [21]:
```
fig = px.histogram(rating_value, x='rating', y = 'count', color = 'rating',
             title = 'Кол-во значений рейтинга игр до обработки пропусков',
             labels={
                    "y": "Кол-во",
                    "x": "Рейтинг"})

fig.show()
```



Проверим кол-во каждого рейтинга, заменив пропуски на TDB.

In [22]:
```
df['rating'] = df['rating'].fillna('tbd')

rating_value = df.groupby('rating')['na_sales'].count().reset_index()

rating_value.columns = ['rating', 'count']

rating_value = rating_value.sort_values('count', ascending = False)

fig = px.histogram(rating_value, x='rating', y = 'count', color = 'rating',
             title = 'Кол-во значений рейтинга игр после обработки пропусков',
             labels={
                    "y": "Кол-во",
                    "x": "Рейтинг"})

fig.show()
```

В столбцах "critic_score" и "user_score"

In [23]:
```
critic_user_score_value = df.groupby(['critic_score', 'user_score'])['na_sales'].co

critic_user_score_value.columns = ['critic_score', 'user_score', 'count']

critic_user_score_value = critic_user_score_value.sort_values('critic_score', ascen

critic_user_score_value.head()
```

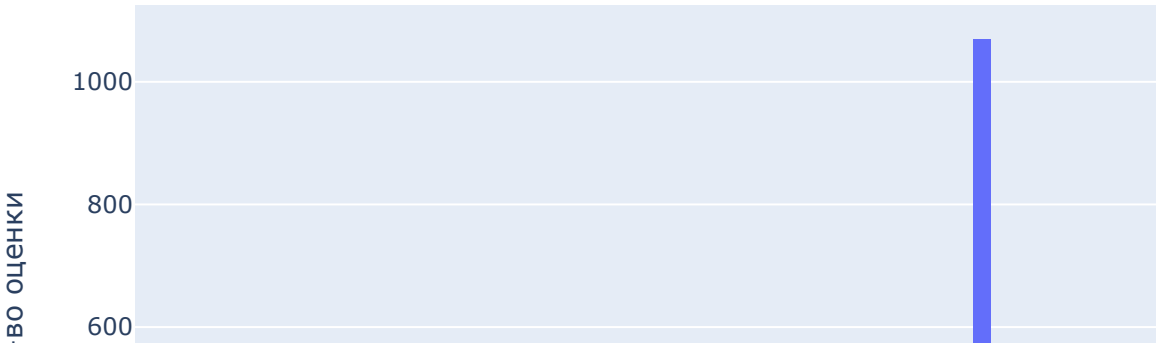
Out[23]:

	critic_score	user_score	count
2377	98.0	8.8	1
2376	98.0	7.9	1
2375	98.0	7.7	1
2374	98.0	7.5	1
2367	97.0	8.2	1

In [24]:
```
fig = px.histogram(critic_user_score_value, x='critic_score', y = 'count',
             title = 'Кол-во оценок критиков до обработки пропусков',
             labels={
                    "critic_score": "Оценка критиков",
                    "count": "кол-во оценки"})

fig.show()
```

Оценки критиков распределены нормально, значения NaN заменять не будем, чтобы не искажать результаты исследований.

In [25]:
```
fig = px.histogram(critic_user_score_value, x='user_score', y = 'count',
             title = 'Кол-во оценок пользователей до обработки пропусков',
             labels={
                    "user_score": "Пользовательская оценка",
                    "count": "кол-во оценки"})

fig.show()
```



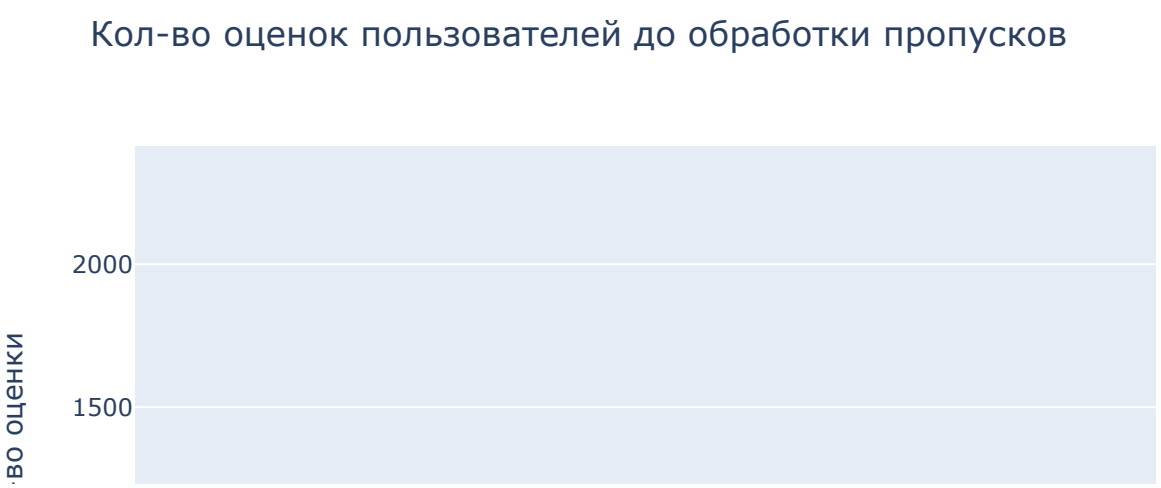
Снова видим TDB в user_score, заменим его на 0 и преобразуем тип данных, но не будем использовать в анализе, т.к. сильно исказит результат. Также поступим с пропусками в critic_score

In [26]:
```
df.loc[df['user_score'] == 'tbd', 'user_score'] = 0

df['user_score'].fillna(0, inplace=True)

df['user_score'] = pd.to_numeric(df['user_score'])

df.loc[df['user_score'] == 0, 'user_score'] = None
```

Преобразовали тип данных в столбце 'user_score' и вернули значения Nan, чтобы минимизировать искажения при анализе данных. Посмотрим как распределы значения пользовательской оценки.

In [27]:
```
user_score_value = df.groupby(['user_score'])['na_sales'].count().reset_index()

user_score_value.columns = ['user_score', 'count']

user_score_value = user_score_value.sort_values('user_score', ascending = False)
```

In [28]:
```
fig = px.histogram(user_score_value, x='user_score', y = 'count',
             title = 'Кол-во оценок пользователей до обработки пропусков',
             labels={
                    "user_score": "Пользовательская оценка",
                    "count": "кол-во оценки"})

fig.show()
```



В столбцах "name" и "genre"

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16567 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16566 non-null  object 
 1   platform         16567 non-null  object 
 2   year_of_release  16567 non-null  int64  
 3   genre            16566 non-null  object 
 4   na_sales         16567 non-null  float64
 5   eu_sales         16567 non-null  float64
 6   jp_sales         16567 non-null  float64
 7   other_sales      16567 non-null  float64
 8   critic_score     8074 non-null   float64
 9   user_score       7538 non-null   float64
 10  rating           16567 non-null  object 
dtypes: float64(6), int64(1), object(4)
memory usage: 1.5+ MB
```

В столбцах 'name' и 'genre' всего по два пропущенных значения, можем спокойно удалить строки с пропусками.

In [30]:
```
df = df.dropna(subset=['name', 'genre'])
#df.info()
```

Финальная сверка

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

Out[31]:
```
name                  0
platform              0
year_of_release       0
genre                 0
na_sales              0
eu_sales              0
jp_sales              0
other_sales           0
critic_score       8492
user_score         9028
rating                0
dtype: int64
```

Запишем суммарные продажи во всех регионах в новый столбец

In [32]:
```
df['summary_sale'] = df['na_sales'] + df['eu_sales'] + df['jp_sales'] + df['other_s
#df.head()
```

Все пропуски обработанны:

- Строки с пропуском имени были удаленны, так же как и строки с пропуском жанра.
- Пропуски в столбце rating были замененны на tbd, что означает что рейтинга нет или он ожидается

- Пропуски в столбцах critic_score и user_score были заменены на нулевые, что означает, что нет оценки.
- Добавлен столбец с сумарными продажами по всем регионам.

Проведём исследовательский анализ данных

- [x] Посмотрите, сколько игр выпускалось в разные годы. Важны ли данные за все периоды?
- [x] Посмотрите, как менялись продажи по платформам. Выберите платформы с наибольшими суммарными продажами и постройте распределение по годам. За какой характерный срок появляются новые и исчезают старые платформы?
- [x] Возьмите данные за соответствующий актуальный период. Актуальный период определите самостоятельно в результате исследования предыдущих вопросов. Основной фактор — эти данные помогут построить прогноз на 2017 год.
- [x] Не учитывайте в работе данные за предыдущие годы.
- [x] Какие платформы лидируют по продажам, растут или падают? Выберите несколько потенциально прибыльных платформ.
- [x] Постройте график «ящик с усами» по глобальным продажам игр в разбивке по платформам. Опишите результат.
- [x] Посмотрите, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Постройте диаграмму рассеяния и посчитайте корреляцию между отзывами и продажами. Сформулируйте выводы.
- [x] Соотнесите выводы с продажами игр на других платформах.
- [x] Посмотрите на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?

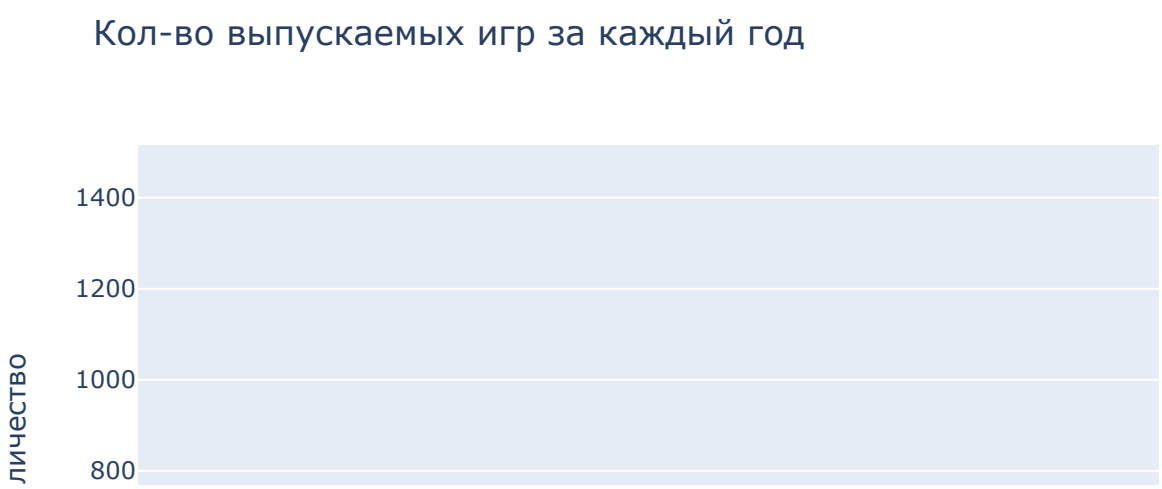
In [33]:
```
df.head()
```

Out[33]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sale
0	Wii Sports	Wii	2006	Sports	41.36	28.96	3.77	8.4
1	Super Mario Bros.	NES	1985	Platform	29.08	3.58	6.81	0.7
2	Mario Kart Wii	Wii	2008	Racing	15.68	12.76	3.79	3.2
3	Wii Sports Resort	Wii	2009	Sports	15.61	10.93	3.28	2.9
4	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	11.27	8.89	10.22	1.0

Сколько игр выпускалось в разные годы?

In [34]:
```
df_name_year = df.pivot_table(index = ['year_of_release'], values = 'name', aggfunc

fig = px.histogram(df_name_year, x="year_of_release", y = 'name', nbins= 38, color
             title = 'Кол-во выпускаемых игр за каждый год',
             labels={
                    "year_of_release": "Год выпуска",
                    "name": "Количество"})

fig.show()
```



После очистки данных, видим, что подходящий период для анализа начинается с минимум 2000 года.

Как менялись продажи по платформам?

In [35]:
```
#Поссчитаем продажи по платформам за каждый год
df_plat_year = df.pivot_table(index = ['year_of_release', 'platform'], values = 'su
df_plat_year.head()
```

Out[35]:

	year_of_release	platform	summary_sale
0	1980	2600	11.38
1	1981	2600	35.68
2	1982	2600	28.88
3	1983	2600	5.84
4	1983	NES	10.96

In [36]:
```
#выделим топ 5 платформ
df_plat_year_top = df_plat_year.groupby(['platform'])['summary_sale'].sum().reset_i
df_plat_year_top
```

Out[36]:

	platform	summary_sale
16	PS2	1247.16
28	X360	966.61
17	PS3	935.18
26	Wii	903.31
4	DS	804.28

Напишем функцию, которая вернёт либо название платформы, которая входит в топ-5, либо значение "others", если платформа не входит в топ-5

In [37]:
```
def top_5_sales_usd_clm (x):
  if x in df_plat_year_top['platform'].unique():
    return x
  return 'others'
```

In [38]:
```
df_plat_year['top5'] = df_plat_year['platform'].apply(top_5_sales_usd_clm)
df_plat_year.head()
```

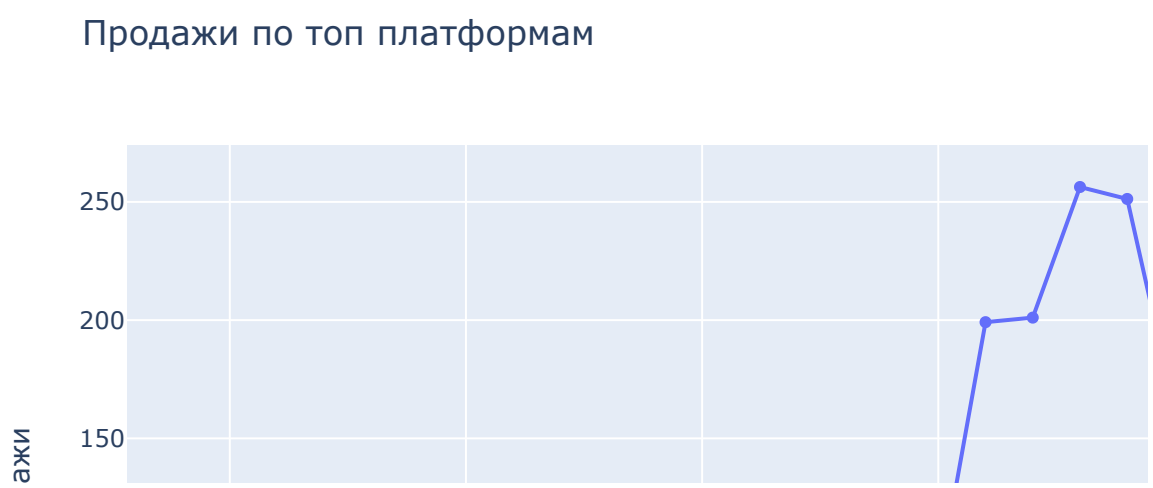
Out[38]:

	year_of_release	platform	summary_sale	top5
0	1980	2600	11.38	others
1	1981	2600	35.68	others
2	1982	2600	28.88	others
3	1983	2600	5.84	others
4	1983	NES	10.96	others

Перегруппируем данные опираясь на топ5 плафторм и построим график.

In [39]:
```
df_plat_year_top5 = df_plat_year.pivot_table(index = ['year_of_release', 'top5'], v
```

In [40]:
```
fig = px.line(df_plat_year_top5, x="year_of_release", y='summary_sale',  color='top
              title = 'Продажи по топ платформам',
              labels={"year_of_release": "год", "summary_sale": "продажи"})
fig.update_layout(legend=dict(
    title="Платформа"))
fig.show()
```



Данный график даёт нам понимание того, что платформы обладают своим циклом жизни.

Далее посмотрим на максимальную и минимальную дату релизов на платформах, а также посчитаем время жизни платформ.

In [41]:
```
df_platform_lifetime = df_plat_year.pivot_table(index = 'platform', values = 'year_
df_platform_lifetime['lifetime'] = df_platform_lifetime[('max','year_of_release')]
df_platform_lifetime
```

Out[41]:

platform	min year_of_release	max year_of_release	lifetime
2600	1980	2002	22
NES	1983	1994	11
DS	1985	2013	28
PC	1985	2016	31
GB	1988	2001	13
SNES	1990	1999	9
GEN	1990	1994	4
GG	1992	1992	0
SCD	1993	1994	1
NG	1993	1996	3
SAT	1994	1999	5
PS	1994	2003	9
3DO	1994	1995	1
TG16	1995	1995	0
PCFX	1996	1996	0
N64	1996	2004	8
DC	1998	2008	10
WS	1999	2001	2
GBA	2000	2007	7
XB	2000	2008	8
PS2	2000	2011	11
GC	2001	2007	6
PSP	2004	2015	11
PS3	2005	2016	11
X360	2005	2016	11
Wii	2006	2016	10
3DS	2010	2016	6
PSV	2011	2016	5

	min	max	lifetime
	year_of_release	year_of_release	
platform			
WiiU	2012	2016	4
PS4	2013	2016	3
XOne	2013	2016	3

Видим, что компьютеры бессмертны, поэтому для расчёта средней продолжительности жизни уберём их из датафрейма.

In [42]:
```
df_platform_lifetime = df_platform_lifetime[df_platform_lifetime['lifetime'] < 30]
```

In [43]:
```
print('Среднее время жизни платформы:', df_platform_lifetime['lifetime'].median())
```

```
Среднее время жизни платформы: 6.5
```

Исходя из полученной информации, для построения прогноза на 2017 год нам нужны данные актуальных платформ, таким образом лучшей минимальной датой, которая отсечёт неактивные приставки и лучше отразит тенденции, будет 2011 год.

Посмотрим на продажи за актуальный период.

In [44]:
```
df_p_y_2011 = df.pivot_table(index = ['year_of_release', 'platform'], values = 'sum
df_p_y_2011 = df_p_y_2011[df_p_y_2011['year_of_release'] >= 2011]
df_p_y_2011.head()
```

Out[44]:

	year_of_release	platform	summary_sale
184	2011	3DS	63.20
185	2011	DS	26.33
186	2011	PC	35.16
187	2011	PS2	0.45
188	2011	PS3	157.98

Перепроверим, остались ли "погибшие" платформы.

In [45]:
```
df_p_y_2011.pivot_table(index = 'platform', values = 'year_of_release', aggfunc = (
```

Out[45]:

platform	min year_of_release	max year_of_release
PS2	2011	2011
DS	2011	2013
PSP	2011	2015
3DS	2011	2016
PC	2011	2016
PS3	2011	2016
PS4	2013	2016
PSV	2011	2016
Wii	2011	2016
WiiU	2012	2016
X360	2011	2016
XOne	2013	2016

Уберём "погибшие" платформы из выборки

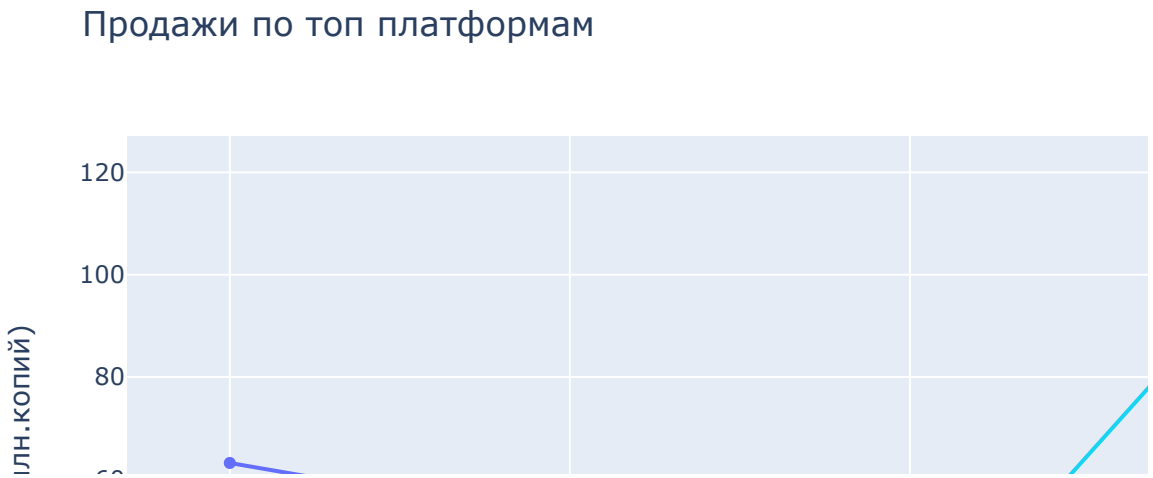
In [46]:
```
dead = ['DS', 'Wii', 'PSP', 'X360', 'PS3']
df_p_y_2011 = df_p_y_2011.query('platform != @dead')
df_p_y_2011.pivot_table(index = 'platform', values = 'year_of_release', aggfunc = (
```

Out[46]:

platform	min year_of_release	max year_of_release
PS2	2011	2011
3DS	2011	2016
PC	2011	2016
PS4	2013	2016
PSV	2011	2016
WiiU	2012	2016
XOne	2013	2016

In [47]:
```
fig = px.line(df_p_y_2011, x="year_of_release", y='summary_sale',  color='platform'
              title = 'Продажи по топ платформам',
```

```
              labels={"year_of_release": "год", "summary_sale": "продажи (млн.копий
fig.update_layout(legend=dict(
    title="Платформа"))
fig.show()
```

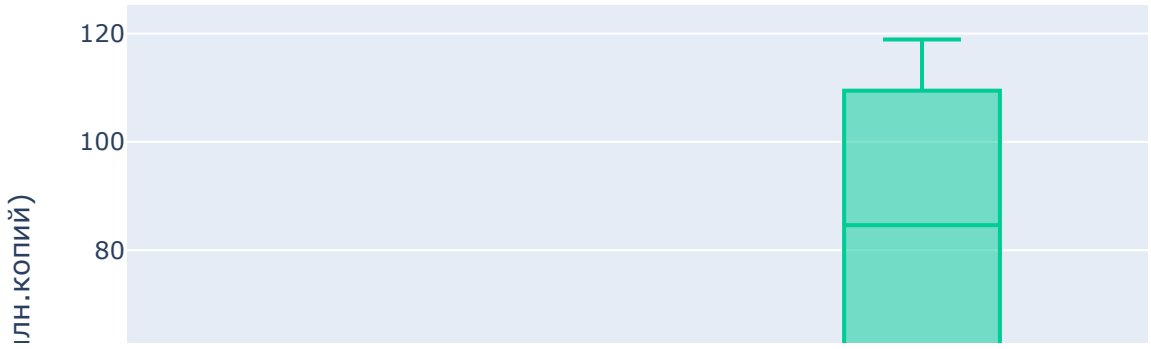


Для большей сбалансированности данных, предлагаю сократить период анализа, повысив нижнюю границу до 2013 года.

In [48]:
```
df_p_y_2013 = df_p_y_2011[df_p_y_2011['year_of_release'] >= 2013]
```

Помня о том, что данные за 2016 год могут быть неполными, стоит отметить две приставки с положительной тенденцией: PS4 и XOne. У остальных приставок отрицательная тенденция, скачков продаж или стабильного прироста не наблюдается.

In [49]:
```
fig = px.box(df_p_y_2013, x="platform", y="summary_sale", color = 'platform',
             title = 'Продажи игр по актуальным платформам',
             labels={"summary_sale": "продажи (млн.копий)", "platform": "платформа

fig.show()
```



Подтверждаем, что PS4 с 69.125 млн.копий и XOne с 34.14 млн.копий лидируют, PC и PSV довольно стабильны в своём результате.

Далее посмотрим на влияние отзывов критиков и юзеров на продажи.

Как влияют на продажи отзывы пользователей и критиков?

In [50]:
```
df_2013_emotions = df.pivot_table(index = ['name', 'year_of_release', 'platform', '
df_2013_emotions = df_2013_emotions[df_2013_emotions['year_of_release'] >= 2013]
```

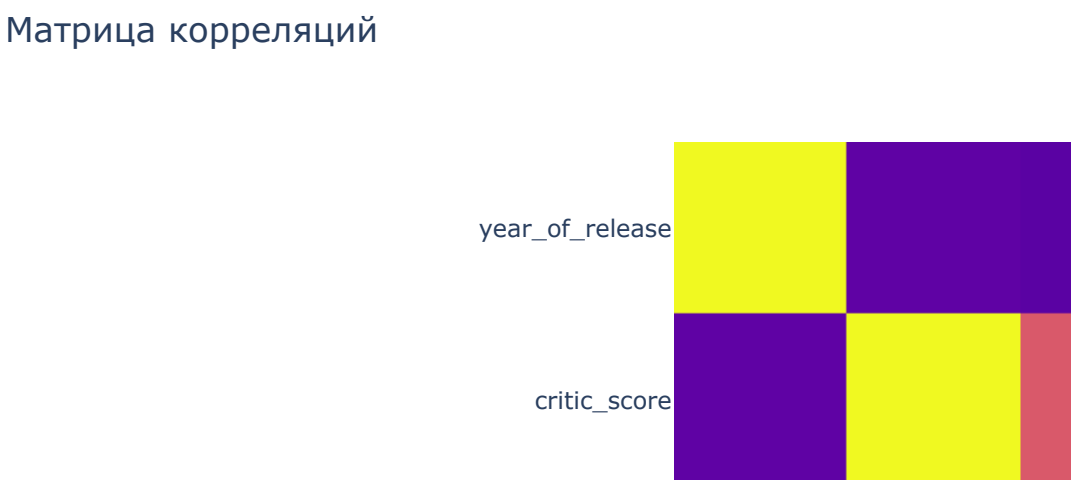
Построим матрицу корреляций.

In [51]:
```
cm = df_2013_emotions.corr()
cm
```

Out[51]:

	year_of_release	critic_score	user_score	summary_sale
year_of_release	1.000000	0.062660	0.050183	-0.134477
critic_score	0.062660	1.000000	0.502702	0.311230
user_score	0.050183	0.502702	1.000000	-0.003964
summary_sale	-0.134477	0.311230	-0.003964	1.000000

In [52]:
```
fig = px.imshow(cm, title = 'Матрица корреляций')
fig.show()
```

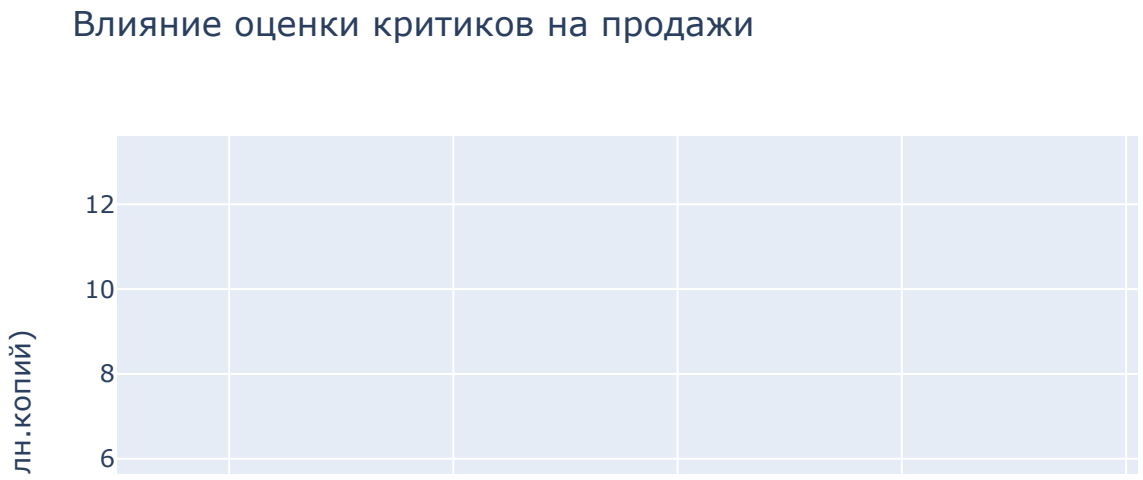


Наблюдаем умеренный уровень корреляции между выручкой и оценкой критиков = 0.31

Далее посмотрим более детально в разрезе двух платформ: PS4 и XOne. Сначала на влияние оценок критиков.

In [53]:
```
df_2013_emotions_X_PS = df_2013_emotions.query('platform == ["PS4", "XOne"]').sort_
fig = px.scatter(df_2013_emotions_X_PS, x="critic_score", y="summary_sale", color =
                 title = 'Влияние оценки критиков на продажи',
```

```
              labels={"critic_score": "оценка критиков", "summary_sale": "продажи(м
fig.update_layout(legend=dict(
    title="Платформа"))
fig.update_xaxes(range=[15,100])
fig.show()
```

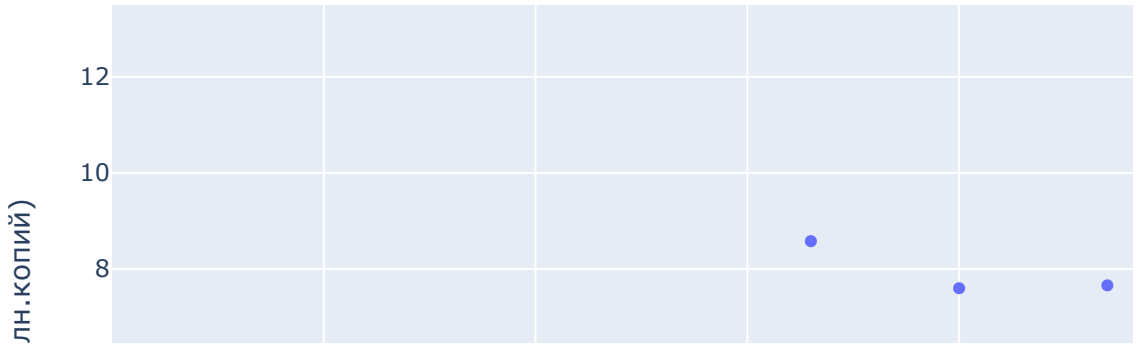


In [54]:
```
df_2013_emotions_X_PS['critic_score'].corr(df_2013_emotions_X_PS['summary_sale'])
```

Out[54]:
```
0.39211550748782226
```

Оценка критиков, действительно, положительно влияет на продажи.

In [55]:
```
fig = px.scatter(df_2013_emotions_X_PS, x="user_score", y="summary_sale", color = '
              title = 'Влияние оценки юзеров на продажи',
              labels={"user_score": "оценка юзеров", "summary_sale": "продажи(млн.к
fig.update_layout(legend=dict(
    title="Платформа"))
fig.update_xaxes(range=[1,10])
fig.show()
```



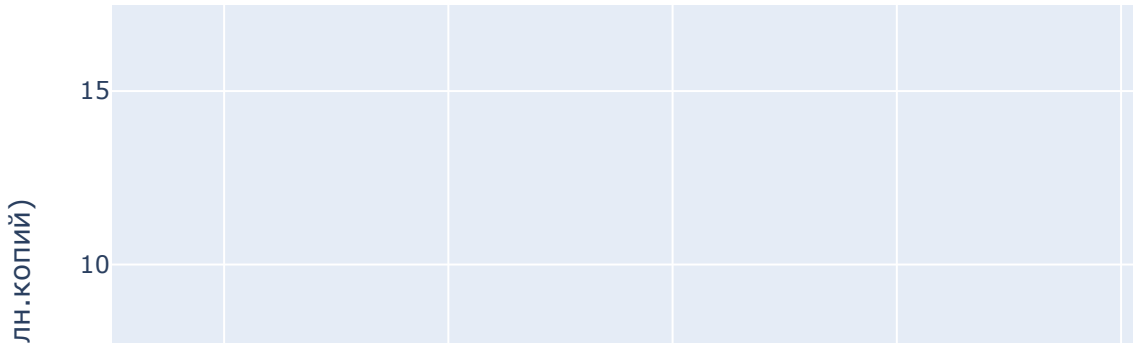
In [56]:
```
df_2013_emotions_X_PS['user_score'].corr(df_2013_emotions_X_PS['summary_sale'])
```

Out[56]:
```
-0.04465510490000265
```

Как ни странно, люди больше доверяют критикам, чем себе подобным.

Дальше посмотрим на продажи по жанрам за актуальный период.

In [57]:
```
df_2013_emotions_all = df_2013_emotions.query('platform == ["WiiU", "X360", "3DS"]'
fig = px.scatter(df_2013_emotions_all, x="critic_score", y="summary_sale", color =
              title = 'Влияние оценки критиков на продажи',
              labels={"critic_score": "оценка критиков", "summary_sale": "продажи(м
fig.update_layout(legend=dict(
    title="Платформа"))
fig.update_xaxes(range=[15,100])
fig.show()
```



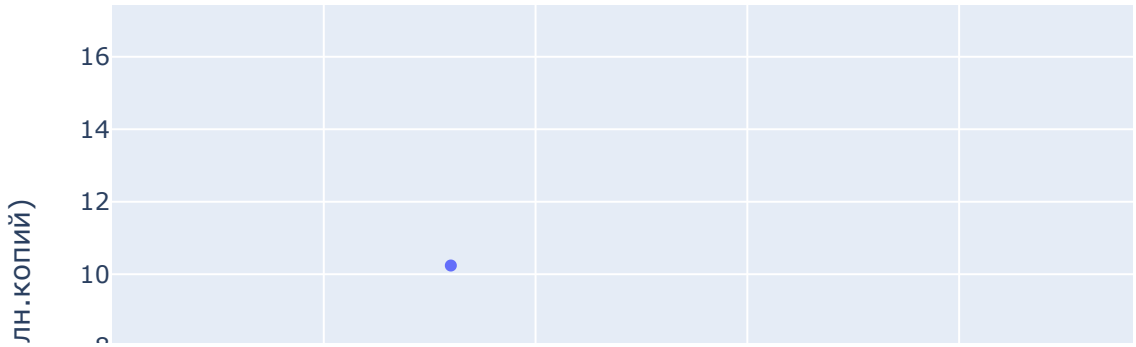
In [58]:
```
df_2013_emotions_all['critic_score'].corr(df_2013_emotions_all['summary_sale'])
```

Out[58]:
```
0.33517324405093885
```

На графике можно выбрать определнную платформу и отобразить по ней результаты.

Наблюдения относительно корреляции оценки критиков и продаж приведенные по платформам PS4 и XOne также применимы к другим платформам.

In [59]:
```
fig = px.scatter(df_2013_emotions_all, x="user_score", y="summary_sale", color = 'p
              title = 'Влияние оценки юзеров на продажи',
              labels={"user_score": "оценка юзеров", "summary_sale": "продажи(млн.к
fig.update_layout(legend=dict(
    title="Платформа"))
fig.update_xaxes(range=[1,10])
fig.show()
```



In [60]:
```
df_2013_emotions_all['user_score'].corr(df_2013_emotions_all['summary_sale'])
```

Out[60]:
```
0.09187670908654896
```

На графике можно выбрать определнную платформу и отобразить по ней результаты.

Таким образом, стоит отметить, что есть платформы с положительной корреляцией: WiiU и 3DS

Общее распределение игр по жанрам

In [61]:
```
df_2013 = df.query('year_of_release >= 2013')
df_genre_2013 = df_2013.groupby('genre').agg({'summary_sale' : "sum", 'name' : 'cou
df_genre_2013['summary_sale'] = round(df_genre_2013['summary_sale'], 2)
df_genre_2013.sort_values( by = 'summary_sale', ascending = False).head()
```

Out[61]:

	genre	summary_sale	name
0	Action	322.50	769
8	Shooter	232.98	187
10	Sports	150.65	214
7	Role-Playing	145.89	292
3	Misc	63.06	156

In [62]:

```
fig = px.histogram(df_genre_2013, x="genre", y = 'summary_sale', color = 'name',
                   title = 'Кол-во проданных игр по жанру',
                   labels={
                          "summary_sale": "Кол-во продаж(млн.копий)",
                          "genre": "Жанр"})
fig.update_layout(legend=dict(
    title="Кол-во наименований"))

fig.show()
```

В топ-3 жанров попадают:

1. Action - 232.25 млн. продаж и 312 выпущенных игр
2. Shooter - 170.99 млн. продаж и 132 выпущенных игр
3. Sports - 115.06 млн. продаж и 110 выпущенных игр

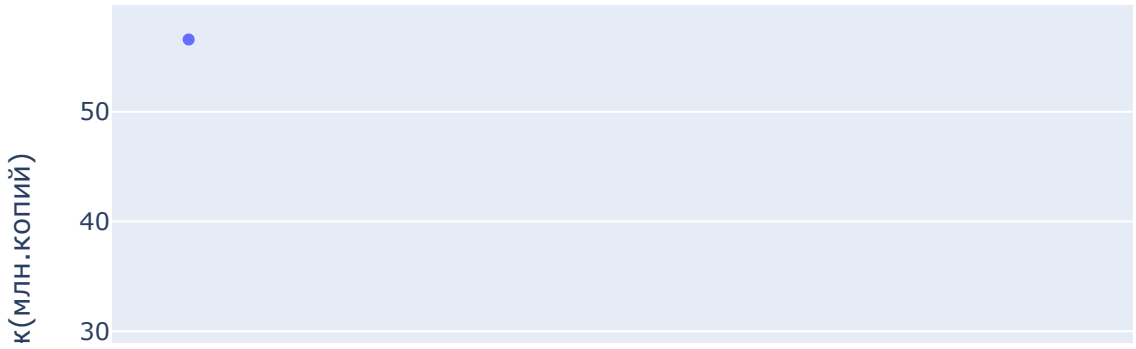
Проверим наличие выбросов в продаж по жанрам.

In [63]:
```
df_genre_2013 = (df_2013.groupby(['genre', 'name']).agg({'summary_sale' : "sum"})
                 .reset_index().sort_values(by = 'summary_sale', ascending = False))

df_genre_2013.head()
```

Out[63]:

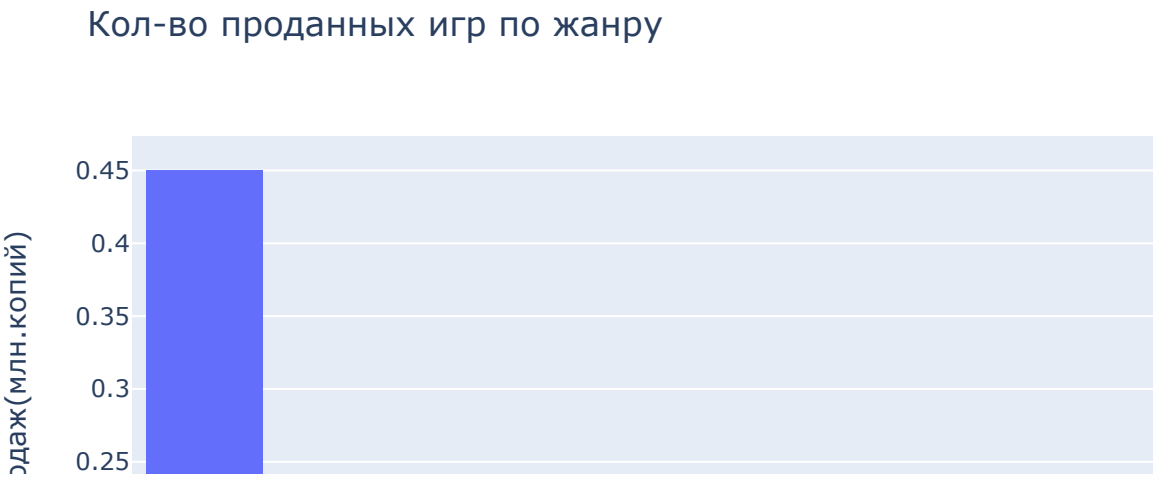
	genre	name	summary_sale
142	Action	Grand Theft Auto V	56.58
1040	Shooter	Call of Duty: Ghosts	27.39
1039	Shooter	Call of Duty: Black Ops 3	25.67
683	Misc	Minecraft	24.16
1038	Shooter	Call of Duty: Advanced Warfare	21.97

In [64]:
```
fig = px.box(df_genre_2013, x="genre", y="summary_sale", color = 'genre',
             title = 'Кол-во проданных игр по жанру',
             labels={
                    "summary_sale": "Кол-во продаж(млн.копий)",
                    "genre": "Жанр"})

fig.show()
```



Жанры, которые раннее попали в топ по продажам обладают множеством выбросов. Посмотри как распределятся медианные значения продаж, чтобы выявить более стабильные жанры.

In [65]:
```
df_genre_2013_mean = df_2013.groupby('genre').agg({'summary_sale' : "median", 'name
df_genre_2013_mean['summary_sale'] = round(df_genre_2013_mean['summary_sale'], 2)
df_genre_2013_mean_top5 = df_genre_2013_mean.sort_values( by = 'summary_sale', asce

fig = px.histogram(df_genre_2013_mean, x="genre", y = 'summary_sale', color = 'genr
              title = 'Кол-во проданных игр по жанру',
              labels={
                     "summary_sale": "Кол-во продаж(млн.копий)",
                     "genre": "Жанр"})
fig.update_layout(legend=dict(
    title="Кол-во наименований"))

fig.show()
```



В топ-5 жанров попадают:

1. Shooter - 0.58 млн. продаж на игру
2. Sports - 0.54 млн. продаж на игру
3. Misc - 0.34 млн. продаж на игру
4. Platform - 0.26 млн. продаж на игру
5. Action - 0.22 млн. продаж на игру

- ☑ Посмотрите, сколько игр выпускалось в разные годы.

С 2000-ых кол-во выпускаемых игр резко росло и достигло пика на 2008 год. После чего кол-во выпускаемых игр стало снижаться.

- ☑ Посмотрите, как менялись продажи по платформам. Выберите платформы с наибольшими суммарными продажами и постройте распределение по годам. За какой характерный срок появляются новые и исчезают старые платформы?

Посмотрев на продажи по платформам, мы увидели, что они обладают своим жизненным циклом. Мы вычислили средний lifetime платформы - 6 лет.

- ☑ Возьмите данные за соответствующий актуальный период. Актуальный период определите самостоятельно в результате исследования предыдущих вопросов. Основной фактор — эти данные помогут построить прогноз на 2017 год.

Актуальный период с 2013 по 2016 год, т.к. в этот период "отсеклись" все "погибшие" платформы, а все остальные на 2013 год уже существовали.

- ☑ Какие платформы лидируют по продажам, растут или падают? Выберите несколько потенциально прибыльных платформ.

Было выделено две платформы, которые появились в 2013 году: PS4 и XOne. Они "свежи" и полны возможностей, когда устаревшие обладают стабильно вниз идущим трендом. Самые потенциально прибыльные платформы - следующие версии существующих.

- ☑ Посмотрите, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Постройте диаграмму рассеяния и посчитайте корреляцию между отзывами и продажами.

Оценка критиков, действительно, умеренно положительно влияет на продажи, когда пользовательская оценка, оказывает очень слабое влияние.

- ☑ Посмотрите на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?

В топ-3 жанров попадают по общей сумме продаж:

Action - 232.25 млн. продаж и 312 выпущенных игр, Shooter - 170.99 млн. продаж и 132 выпущенных игр, Sports - 115.06 млн. продаж и 110 выпущенных игр.

В топ-5 жанров попадают по медианной сумме продаж:

1. Shooter - 0.58 млн. продаж на игру.
2. Sports - 0.54 млн. продаж на игру.
3. Misc - 0.34 млн. продаж на игру.
4. Platform - 0.26 млн. продаж на игру.
5. Action - 0.22 млн. продаж на игру.

Составим портрет пользователя каждого региона

Определите для пользователя каждого региона (NA, EU, JP):

- ☑ Самые популярные платформы (топ-5). Опишите различия в долях продаж.
- ☑ Самые популярные жанры (топ-5). Поясните разницу.

- ☑ Влияет ли рейтинг ESRB на продажи в отдельном регионе?

In [66]:
```
df.head()
```

Out[66]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sale
0	Wii Sports	Wii	2006	Sports	41.36	28.96	3.77	8.4
1	Super Mario Bros.	NES	1985	Platform	29.08	3.58	6.81	0.7
2	Mario Kart Wii	Wii	2008	Racing	15.68	12.76	3.79	3.2
3	Wii Sports Resort	Wii	2009	Sports	15.61	10.93	3.28	2.9
4	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	11.27	8.89	10.22	1.0

Самые популярные платформы

In [67]:
```
platform = list(df_p_y_2013['platform'].unique())
platform
```

Out[67]:
```
['3DS', 'PC', 'PS4', 'PSV', 'WiiU', 'XOne']
```

Доли продаж по региону за актуальный период

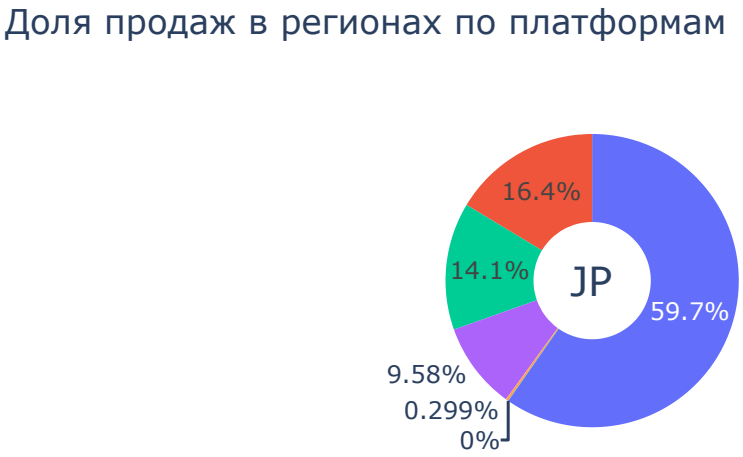
In [68]:
```
df_dolya_platform = (
            df.query('year_of_release > 2012 and platform == @platform').groupby('p
            .agg({'na_sales' : 'sum', 'eu_sales' : 'sum', 'jp_sales' : 'sum', 'summ
           )
df_dolya_platform['dolya_na'] =  df_dolya_platform['na_sales']/df_dolya_platform['s
df_dolya_platform['dolya_eu'] =  df_dolya_platform['eu_sales']/df_dolya_platform['s
df_dolya_platform['dolya_jp'] =  df_dolya_platform['jp_sales']/df_dolya_platform['s

numeric_columns = df_dolya_platform.drop(columns=['platform']).columns

df_dolya_platform.sort_values(by = 'summary_sale', ascending = False).style.format(
    .highlight_max(color='yellowgreen', subset=numeric_columns) \
    .highlight_min(color='coral', subset=numeric_columns)
```

Out[68]:

	platform	na_sales	eu_sales	jp_sales	summary_sale	dolya_na	dolya_eu	dolya_jp
2	PS4	108.74	141.09	15.96	314.14	0.14	0.19	0.02
5	XOne	93.12	51.59	0.34	159.32	0.12	0.07	0.00
0	3DS	38.20	30.96	67.81	143.25	0.05	0.04	0.09
4	WiiU	29.21	19.85	10.88	64.63	0.04	0.03	0.01
1	PC	11.19	25.84	0.00	40.06	0.01	0.03	0.00
3	PSV	5.04	6.10	18.59	32.99	0.01	0.01	0.02

In [69]:

```
labels = list(df_dolya_platform['platform'].unique())
values_jp = list(df_dolya_platform['dolya_jp'].unique())
values_eu = list(df_dolya_platform['dolya_eu'].unique())
values_na = list(df_dolya_platform['dolya_na'].unique())

# Create subplots: use 'domain' type for Pie subplot
specs = [[{'type':'domain'}, {'type':'domain'}], [{'type':'domain'}, {'type':'domai
fig = make_subplots(rows=2, cols=2, specs=specs)

fig.add_trace(go.Pie(labels=labels, values=values_jp, name="JP"),
              1, 1)
fig.add_trace(go.Pie(labels=labels, values=values_eu, name="EU"),
              1, 2)
fig.add_trace(go.Pie(labels=labels, values=values_na, name="NA"),
              2, 1)

fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Доля продаж в регионах по платформам",
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='JP', x=0.21, y=0.83, font_size=20, showarrow=False),
                 dict(text='EU', x=0.795, y=0.83, font_size=20, showarrow=False),
                 dict(text='NA', x=0.205, y=0.17, font_size=20, showarrow=False)])
fig.show()
```



1. EU: PS4, XOne, PC, WiiU, 3DS - Европа
2. JP: 3DS, PS4, WiiU, PSV, XOne - Япония
3. NA: PS4, XOne, WiiU, 3DS, PC - США

Самые популярные жанры

Доли продаж по жанру

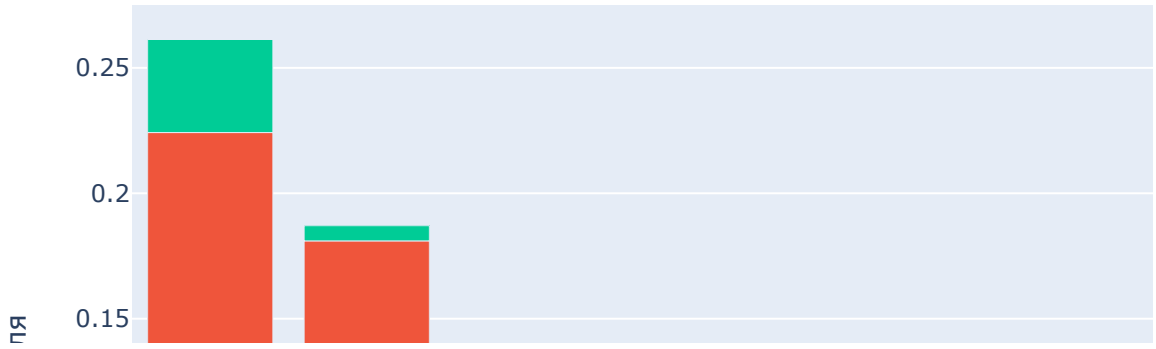
In [70]:
```
df_dolya_genre = df.query('year_of_release > 2012').groupby('genre').agg({'na_sales
df_dolya_genre['dolya_na'] =  df_dolya_genre['na_sales']/df_dolya_genre['summary_sa
df_dolya_genre['dolya_eu'] =  df_dolya_genre['eu_sales']/df_dolya_genre['summary_sa
df_dolya_genre['dolya_jp'] =  df_dolya_genre['jp_sales']/df_dolya_genre['summary_sa

numeric_columns = df_dolya_genre.drop(columns=['genre']).columns

df_dolya_genre.sort_values(by = 'summary_sale', ascending = False).style.format({i:
    .highlight_max(color='yellowgreen', subset=numeric_columns) \
    .highlight_min(color='coral', subset=numeric_columns)
```

Out[70]:

	genre	na_sales	eu_sales	jp_sales	summary_sale	dolya_na	dolya_eu	dolya_jp
0	Action	126.13	118.61	40.49	322.50	0.12	0.11	0.04
8	Shooter	109.74	87.86	6.61	232.98	0.10	0.08	0.01
10	Sports	65.27	60.52	5.41	150.65	0.06	0.06	0.00
7	Role-Playing	46.40	36.97	51.04	145.89	0.04	0.03	0.05
3	Misc	27.49	20.04	9.44	63.06	0.03	0.02	0.01
4	Platform	18.14	15.58	4.79	42.63	0.02	0.01	0.00
6	Racing	12.96	20.19	2.30	39.89	0.01	0.02	0.00
2	Fighting	15.55	8.55	7.65	35.31	0.01	0.01	0.01
1	Adventure	7.14	8.25	5.82	23.64	0.01	0.01	0.01
9	Simulation	4.86	10.92	4.52	21.76	0.00	0.01	0.00
11	Strategy	3.28	4.22	1.77	10.08	0.00	0.00	0.00
5	Puzzle	0.83	1.00	1.18	3.17	0.00	0.00	0.00

In [71]:

```
df_dolya_genre = df_dolya_genre.sort_values(by = 'summary_sale', ascending = False)
fig = px.bar(df_dolya_genre, x="genre", y=['dolya_na', 'dolya_eu', 'dolya_jp'],
             title = 'Доля продаж игр в регионе по жанру',
             labels={
                    "value": "Доля",
                    "genre": "Жанр"})
fig.update_layout(legend=dict(
    title="Регион"))

fig.show()
```



EU: Action, Shooter, Sports, Role-Playing, Racing - Европа
JP: Action, Role-Playing, Shooter, Platform, Misc - Япония
NA: Action, Shooter, Sports, Role-Playing, Platform - США

Влияет ли рейтинг ESRB на продажи в отдельном регионе?

Рейтинг по регионам

In [72]:
```
df_dolya_rating = df.query('year_of_release > 2012').groupby('rating').agg({'na_sal
df_dolya_rating['dolya_na'] =  df_dolya_rating['na_sales']/df_dolya_rating['summary
df_dolya_rating['dolya_eu'] =  df_dolya_rating['eu_sales']/df_dolya_rating['summary
df_dolya_rating['dolya_jp'] =  df_dolya_rating['jp_sales']/df_dolya_rating['summary

numeric_columns = df_dolya_rating.drop(columns=['rating']).columns

df_dolya_rating.sort_values(by = 'summary_sale', ascending = False).style.format({i
    .highlight_max(color='yellowgreen', subset=numeric_columns) \
    .highlight_min(color='coral', subset=numeric_columns)
```

Out[72]:

	rating	na_sales	eu_sales	jp_sales	summary_sale	dolya_na	dolya_eu	dolya_jp
2	M	165.29	145.80	14.11	372.31	0.15	0.13	0.01
4	tbd	89.42	78.91	85.29	277.08	0.08	0.07	0.08
0	E	79.05	83.36	15.14	200.16	0.07	0.08	0.01
3	T	49.79	41.95	20.59	126.62	0.05	0.04	0.02
1	E10+	54.24	42.69	5.89	115.39	0.05	0.04	0.01

In [73]:
```
df_dolya_rating = df_dolya_rating.sort_values(by = 'summary_sale', ascending = Fals
fig = px.bar(df_dolya_rating, x="rating", y=['dolya_na', 'dolya_eu', 'dolya_jp'],
             title = 'Доля продаж игр в регионе по рейтингу',
             labels={
                    "value": "Доля",
                    "rating": "рейтинг"})
fig.update_layout(legend=dict(
    title="Регион"))

fig.show()
```

Влияет ли рейтинг ESRB на продажи в отдельном регионе?

Да, влияние есть:

1. Рейтинг "M" («Mature») — «Для взрослых»: Материалы игры не подходят для лиц младше 17 лет. Является самым популярным в регионах EU и NA.
2. «T» («Teen») — «Подросткам»: Игра подходит для лиц от 13 лет. Является самым популярным в регионе JP.
3. Наименее популярный рейтинг в регионах JP, EU и NA - «E10+» («Everyone 10 and older») — «Для всех от 10 лет и старше».

Пользователи региона EU:

1. PS4, XOne, PC
2. Action, Shooter, Sports
3. Любимый рейтинг - "M" («Mature») — «Для взрослых»: Материалы игры не подходят для лиц младше 17 лет.

Пользователи региона NA:

1. PS4, XOne, WiiU
2. Action, Shooter, Sports
3. Любимый рейтинг - "M" («Mature») — «Для взрослых»: Материалы игры не подходят для лиц младше 17 лет.

Пользователи региона JP:

1. 3DS, PS4, WiiU
2. Action, Role-Playing, Shooter
3. «T» («Teen») — «Подросткам»: Игра подходит для лиц от 13 лет. Является самым популярным в регионе JP.

Итог:

Мы получаем схожую картинку между пользователями регионов EU и NA по всем направлениям. Пользователи региона JP отличаются любовью к приставке 3DS, играм жанра Role-Playing и возрастным рейтингом «T».

Проверим гипотезы

- ☑ Средние пользовательские рейтинги платформ Xbox One и PC одинаковые;
- ☑ Средние пользовательские рейтинги жанров Action (англ. «действие», экшен-игры) и Sports (англ. «спортивные соревнования») разные.

Средние пользовательские рейтинги платформ Xbox One и PC одинаковые

H0 - Статистически значимых различий в среднем пользовательском рейтинге платформ Xbox One и PC нет.

H1 - Различие в среднем пользовательском рейтинге платформ Xbox One и PC статистически значимо.

alpha = 0.05

In [74]:
```
df = df.dropna()
xone = list(df[(df['year_of_release'] > 2012) & (df['platform'] == 'XOne')]['user_s
pc = list(df[(df['year_of_release'] > 2012) & (df['platform'] == 'PC')]['user_score

res = stats.bartlett(xone, pc)
print('степень различия двух групп:', res.statistic)
```

```
степень различия двух групп: 9.025826010108553
```

Группы сильно различны, поэтому в тест передаём equal_var = False

In [75]:
```
alpha = .05 # критический уровень статистической значимости
            # если p-value окажется меньше него - отвергнем гипотезу

results = st.ttest_ind(xone, pc, equal_var = False)

print('p-значение: {:.5f}'.format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу, рейтинги платформ Xbox One и PC разные")
else:
    print("Не получилось отвергнуть нулевую гипотезу, рейтинги платформ Xbox One и
```

```
p-значение: 0.19483
Не получилось отвергнуть нулевую гипотезу, рейтинги платформ Xbox One и PC одинаковы
е
```

Средние пользовательские рейтинги жанров Action и Sports разные.

H0 - Статистически значимых различий в среднем пользовательском рейтинге жанров Action и Sports нет.

H1 - Различие в среднем пользовательском рейтинге жанров Action и Sports статистически значимо.

alpha = 0.05

In [76]:
```
action = list(df[(df['year_of_release'] > 2012) & (df['genre'] == 'Action')]['user_
sports = list(df[(df['year_of_release'] > 2012) & (df['genre'] == 'Sports')]['user_

res = stats.bartlett(action, sports)
print('степень различия двух групп:', res.statistic)
```

```
степень различия двух групп: 2.8263801001119457
```

Группы сильно различны, поэтому в тест передаём equal_var = False

In [77]:
```
alpha = .05 # критический уровень статистической значимости
            # если p-value окажется меньше него - отвергнем гипотезу

results = st.ttest_ind(action, sports, equal_var = False)

print('p-значение: {:.20f}'.format(results.pvalue))

if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу, средние пользовательские рейтинги жанров Act
else:
    print("Не получилось отвергнуть нулевую гипотезу, средние пользовательские рейт
```

```
p-значение: 0.00000000000010460522
Отвергаем нулевую гипотезу, средние пользовательские рейтинги жанров Action и Sports
разные
```

При проведении статистических тестов получили, что:

1. Средние рейтинги платформ Xbox One и PC одинаковые.
2. Средние пользовательские рейтинги жанров Action и Sports различны.

Вывод

В данной работе был выполнен следующий ряд задач, который привёл к следующим результатам:

1. Ознакомление с данными:

   - Названия столбцов начинаются с верхнего регистра
   - Поля с пропущенными данными: name, year_of_release, genre, critic-score, user_score, rating
   - Тип данных неверен в: year_of_release, user_score
2. Подготовка данных к анализу:

   - Строки с пропуском имени были удаленны, так же как и строки с пропуском жанра.
   - Пропуски в столбце rating были замененны на "tbd", что так или иначе означает - рейтинга нет.
   В дальнейшем анализе строки со значением "tdb" в столбце "rating" не использовались, чтобы избежать искажения
   результатов.
   - Пропуски в столбцах "critic_score" и "user_score" были заменены на нулевые, что означает, что нет оценки.
   В дальнейшем анализе строки с нулевыми значениями в данных столбцах не использовались, чтобы избежать искажения
   результатов.
   - Суммарные продажи по регионам посчитаны и занесены в столбец "summary_sale".
3. Исследовательский анализ:

- Пик выпуска игр пришёлся на 2008 год.
- Средний срок жизни платформы - 6 лет, поэтому дальнейший период анализа определен с 2013 по 2016 год.
- Актуальными плаформами, на которые стоит обратить внимание являются PS4 и XOne.
- В топ жанров попали по общим продажам: Action, Shooter, Sports.
- В топ жанров попали по медианным продажам: Shooter, Sports, Misc, Platform, Action. Они отличаются стабильным выпуском приносящих деньги игр.
- Отзывы критиков обладают влиянием на продажи. Данные показатели коррелируют с умеренным значением - "0,31". Чего нельзя отметить о пользовательских оценках, обладающих слабым, отрицательным коэф-ом корреляции - "-0.003964".

4. Портреты пользователей:

- Мы получаем схожую картинку между пользователями регионов EU и NA по всем направлениям:
    а) Любимые приставки: PS4 и XOne.
    б) Любимые жанры: Action, Shooter, Sports.
    в) Любимый возратсной рейтинг игр: «M» («Mature»).
- Пользователи региона JP отличаются любовью к приставке 3DS, играм жанра Role-Playing и возрастным рейтингом «T» (Teen).
- Вторым любимым рейтингом для всех регионов является «E» («Everyone») — «Для всех».

5. Проверка гипотез:

- Подтвердили, что средние пользовательские рейтинги платформ Xbox One и PC одинаковые.
- Опровергли, что средние пользовательские рейтинги жанров Action и Sports одинаковые.

Подведя итог получаем, что:

- Стоит поработать над сбором данных, скорее всего именно он является причиной отсутствия множества данных.
- Планировать выпуск игры стоит на первые 3 года жизни платформы.
- Платформа, жанр и рейтинг игры зависят от региона. Для EU и NA стандрты одни, а для JP совсем другие.
- Покупка мнения "критиков" может оправдать себя.