Проект: "Самые популярные компьютерные игры"

Вводные данные.

Клиент: интернет-магазин «Стримчик», который продаёт по всему миру компьютерные игры.

Из открытых источников доступны исторические данные до 2016 года о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation). Данные за 2016 год могут быть неполными.

Оценку игрового контента производит **ESRB**, он же присваивает игре подходящую возрастную категорию.

Цель: для планирования рекламные кампании в 2017 году нужно выявить определяющие успешность игры закономерности, чтобы сделать ставку на потенциально популярный продукт.

План работы:

- загрузка и настройка библиотек

---

1. Загрузка файла с данными
2. Изучение общей информации и выполнение предобработки данных:
    - проверка названий столбцов;
    - преобразование данных в нужные типы;
    - Обработка пропусков при необходимости с пояснением выбранной методики и возможных причин пропусков;
    - Подсчет суммарных продаж во всех регионах, фиксация результата в отдельный столбец.
3. Проведение исследовательского анализа данных:
    - сколько игр выпускалось в разные годы. Важны ли данные за все периоды;
    - как менялись продажи по платформам;
    - Выберем платформы с наибольшими суммарными продажам и построим распределение по годам. За какой характерный срок появляются новые и исчезают старые платформы?
    - Возьмем данные за соответствующий актуальный период , определенный самостоятельно в результате исследования предыдущих вопросов. Основной фактор — эти данные помогут построить прогноз на 2017 год. Данные за предыдущие годы не учитывать в работе.
    - выяснить какие платформы лидируют по продажам, растут или падают. Выбрать несколько потенциально прибыльных платформ.
    - Построить график «ящик с усами» по глобальным продажам игр в разбивке по платформам. Описать результат.
    - Посмотреть как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Построить диаграмму рассеяния и посчитать корреляцию между отзывами и продажами. Сформулировать выводы.
    - Соотнести выводы с продажами игр на других платформах.
    - Посмотреть на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?
4. Составление портрета пользователя каждого региона NA, EU, JP. Для этого определить для пользователя региона:
    - Самые популярные платформы (топ-5). Описать различия в долях продаж.
    - Самые популярные жанры (топ-5). Пояснить разницу.
    - Влияет ли рейтинг ESRB на продажи в отдельном регионе?
5. Проверить гипотезы. Для этого задать пороговое значение alpha, пояснить как вы сформулирована нулевая и альтернативная гипотезы, какой критерий применен для проверки гипотез и почему.
   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.
6. Общий вывод

Загрузка и настройка библиотек.

In [1]:

```
# Датасеты
import pandas as pd
from pandas.api.types import CategoricalDtype
# Математика
import numpy as np

# Статистика
from scipy import stats as st
from scipy.stats import binom

# Графики
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import rcParams, rcParamsDefault

import seaborn as sns

# Обязательно для приемлемого отображения графиков plt
rcParams['figure.figsize'] = 10, 6
%config InlineBackend.figure_format = 'svg'
# доп. для декорирования графиков
factor = .8
default_dpi = rcParamsDefault['figure.dpi']
rcParams['figure.dpi'] = default_dpi * factor
# Копирование значений, а не ссылок, через b = copy.deepcopy(a)
import copy
```

1 Загрузка файла с данными

1.1 Загрузка занных.

Загрузим данные из cvs-файла в датафрейм с помощью библиотеки pandas. Помимо пути директории укажем ссылку на исх.данные.

In [2]:

```
try:
    games = pd.read_csv('/datasets/games.csv', sep=',')
except:
    games = pd.read_csv('https://code.s3.yandex.net/datasets/games.csv')
print('\nПосмотрим на получившийся датафрейм. Выведем три случайные строки:\n')
games.sample(3)
```

Посмотрим на получившийся датафрейм. Выведем три случайные строки:

Out[2]:

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
7150	Skies of Arcadia Legends	GC	2002.0	Role-Playing	0.17	0.04	0.0	0.01	84.0	9.2	T
4171	Epic Mickey 2: The Power of Two	PS3	2012.0	Action	0.21	0.18	0.0	0.08	59.0	4.3	E
15946	Gears of War	PC	2007.0	Shooter	0.00	0.01	0.0	0.00	87.0	7.8	M

In [3]:

```
print('Датафрейм содержит', len(games), 'строк', len(games.columns), 'столбцов.')
```

Датафрейм содержит 16715 строк 11 столбцов.

In [4]:

```
print('Датафрем соержит колонки:\n')
list(games.columns)
```

Датафрем соержит колонки:

Out[4]:

['Name',
 'Platform',
 'Year_of_Release',
 'Genre',
 'NA_sales',
 'EU_sales',
 'JP_sales',
 'Other_sales',
 'Critic_Score',
 'User_Score',
 'Rating']

1.2 Итоги раздела

Загружены данные в датафрейм **games**.
В выборку попали 16715 строк 11 столбцов.
Данные содержат информацию о названии игры, платформе размещения, год релиза, жанр. данные о продажах (миллионы) в странах NA, EU, JP и других; оценку критиков, оценку пользователей, рейтинг от организации ESRB.
Колонки в датафрейме и описание данных:

- Name — название игры
- Platform — платформа
- Year_of_Release — год выпуска
- Genre — жанр игры
- NA_sales — продажи в Северной Америке (миллионы проданных копий)
- EU_sales — продажи в Европе (миллионы проданных копий)
- JP_sales — продажи в Японии (миллионы проданных копий)
- Other_sales — продажи в других странах (миллионы проданных копий)
- Critic_Score — оценка критиков (максимум 100)
- User_Score — оценка пользователей (максимум 10)
- Rating — рейтинг от организации ESRB.

Предварительно можно утверждать, что предоставленного объема данных достаточно для исследования и проверки гипотез.
Для дальнейшего анализа проведем предобработку данных.

2 Изучение общей информации и выполнение предобработки данных:

- проверка названий столбцов;
- преобразование данных в нужные типы;
- Обработка пропусков при необходимости с пояснением выбранной методики и возможных причин пропусков;
- Подсчет суммарных продаж во всех регионах, фиксация результата в отдельный столбец.

2.1 Проверка названий столбцов

Для преобразования наименований столбцов в нижний регистр используем метод str.lower()

In [5]:

```
print(' Название столбцов ДО преобразования регистра на нижний:\n')
```

```
list(games.columns)
```

Название столбцов ДО преобразования регистра на нижний:

Out[5]:

['Name',
 'Platform',
 'Year_of_Release',
 'Genre',
 'NA_sales',
 'EU_sales',
 'JP_sales',
 'Other_sales',
 'Critic_Score',
 'User_Score',
 'Rating']

In [6]:

```
games.columns = games.columns.str.lower()
print(' Название столбцов ПОСЛЕ преобразования регистра на нижний:\n')
list(games.columns)
```

Название столбцов ПОСЛЕ преобразования регистра на нижний:

Out[6]:

['name',
 'platform',
 'year_of_release',
 'genre',
 'na_sales',
 'eu_sales',
 'jp_sales',
 'other_sales',
 'critic_score',
 'user_score',
 'rating']

2.2 Преобразование данных в нужные типы и обработка пропусков

2.2.1 Посмотрим на тип данных, наличие пропусков и дубликатов во всем датафрейме.

In [7]:

```
print('Сводная информация по датафрейму games:\n ')
print(games.info())
print()
print(
    'Датафрейм games содержит', \
    len(games), \
    'строк;\nколичество явных дубликатов в games =', \
    games.duplicated().sum(), 'шт.;'\
    '\nколичество пропусков в games = ',\
    games.isnull().values.sum(), '.'
     )
```

Сводная информация по датафрейму games:

<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
None
```

Датафрейм games содержит 16715 строк;
количество явных дубликатов в games = 0 шт.;
количество пропусков в games = 22318 .

Проверим датафрейм на "неявные" дубликаты. Поскольку в данных часто встречаются разного рода ошибки, полученные, например, при сборе из разных БД, использовании внешних данных, техническом сбое при логировании. Поэтому следует сделать более тщательную проверку.
Сравним данные,используя дополнительный параметр subset()) по столбцам name, platform, year_of_release.

In [8]:

```
print(games[games.duplicated(['name', 'platform', 'year_of_release'])].count())
games[games.duplicated(['name', 'platform', 'year_of_release'])]
```

```
name             1
platform         2
year_of_release    2
genre            1
na_sales         2
eu_sales         2
jp_sales         2
other_sales      2
critic_score     1
user_score       1
rating           1
dtype: int64
```

Out[8]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
14244	NaN	GEN	1993.0	NaN	0.0	0.00	0.03	0.0	NaN	NaN	NaN
16230	Madden NFL 13	PS3	2012.0	Sports	0.0	0.01	0.00	0.0	83.0	5.5	E

Поиск обнаружил две строки с неявными дубликатами. Удалим их.

In [9]:

```
games = games.drop_duplicates(['name', 'platform', 'year_of_release'])
```

```
if (len(games[games.duplicated(['name', 'platform', 'year_of_release'])])) != 0:
   print('Ahtung! Что-то пошло не так!')
else:
   print('Неявные дубликаты удалены успешно!')
```
Неявные дубликаты удалены успешно!

Дубликатов теперь в датафрейме нет. А вот пропусков достаточно.
Изучим каждую колонку отдельно. При необходимости произведем преобразование данных в нужные типы, обработаем пропуски, по возможности их заполним, с пояснением выбранной методики и возможных причин пропусков.

2.2.2 Колонка name

In [10]:
```
print('Пропусков в столбце name', \
     games['name'].isnull().sum(), \
     'шт. или', \
     round(games['name'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк.'
    )
```
Пропусков в столбце name 1 шт. или 0.006 % от общего числа строк.

Строк с пропусками мало, можно посмотреть на них:

In [11]:
```
games[games['name'].isna()]
```
Out[11]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
659	NaN	GEN	1993.0	NaN	1.78	0.53	0.0	0.08	NaN	NaN	NaN

Год релиза 1993 и платформа GEN. Помимо пропусков в наименовании есть еще пропуски в других колонках: с жанром, с оценками критиков и пользователей, не указан рейтинг ESRB.
Пропуски в названии игры могут быть вызваны технологическими причинами, например повреждение данных при переносе, либо человеческим фактором - при занесении данных этот стобец был случайно пропущен.
Название игры важный параметр, заполнить его какими-то средними значениями не имеет смысла - нет названия объекта. Малое количество пропусков - 0,006% - позволяет нам **удалить строку с пропусками в столбце name** без сожаления и поисков решения их устранения.

In [12]:
```
games = games.dropna(subset=['name'])
if games['name'].isnull().sum() == 0:
   print('Пропуски в столбце name удалены успешно!')
else:
   print('Ahtung!!! Пропуски не удалены')
```
Пропуски в столбце name удалены успешно!

In [13]:
```
print('Всего строк с названиями игр:', len(games['name']), 'шт.\n')
print('Всего уникальных названий игр:', len(games['name'].value_counts()), 'шт.')
print('Повторов названий игр:', (len(games['name'])) -
(len(games['name'].value_counts())), 'шт.')
```
Всего строк с названиями игр: 16712 шт.

Всего уникальных названий игр: 11559 шт.
Повторов названий игр: 5153 шт.

Получается больше 5 тысяч игр повторно указаны в колонке name. Проведем эксперимент: отсортируем названия по частоте встреч, затем посмотрим самое популярное название игры для первых двух-трех названий, поищем в чем дело.

In [14]:

```
games['name'].value_counts()
```

Out[14]:

```
Need for Speed: Most Wanted                12
Madden NFL 07                               9
LEGO Marvel Super Heroes                    9
FIFA 14                                     9
Ratatouille                                 9
                                           ..
Guitar Freaks 3rdMIX & DrumMania 2ndMIX     1
Kanzen Chuuki Pro Yakyuu Greatest Nine      1
Shin Sangoku Musou: Multi Raid 2            1
Battle Hunter                               1
Chocobo's Dungeon 2                         1
Name: name, Length: 11559, dtype: int64
```

In [15]:

```
result_1 = games.query('name == "Need for Speed: Most Wanted"')
result_1.head(12)
```

Out[15]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
253	Need for Speed: Most Wanted	PS2	2005.0	Racing	2.03	1.79	0.08	0.47	82.0	9.1	T
523	Need for Speed: Most Wanted	PS3	2012.0	Racing	0.71	1.46	0.06	0.58	NaN	NaN	NaN
1190	Need for Speed: Most Wanted	X360	2012.0	Racing	0.62	0.78	0.01	0.15	83.0	8.5	T
1591	Need for Speed: Most Wanted	X360	2005.0	Racing	1.00	0.13	0.02	0.10	83.0	8.5	T
1998	Need for Speed: Most Wanted	XB	2005.0	Racing	0.53	0.46	0.00	0.05	83.0	8.8	T
2048	Need for Speed: Most Wanted	PSV	2012.0	Racing	0.33	0.45	0.01	0.22	NaN	NaN	NaN
3581	Need for Speed: Most Wanted	GC	2005.0	Racing	0.43	0.11	0.00	0.02	80.0	9.1	T
5972	Need for Speed: Most Wanted	PC	2005.0	Racing	0.02	0.23	0.00	0.04	82.0	8.5	T
6273	Need for Speed: Most Wanted	WiiU	2013.0	Racing	0.13	0.12	0.00	0.02	NaN	NaN	NaN
6410	Need for Speed: Most Wanted	DS	2005.0	Racing	0.24	0.01	0.00	0.02	45.0	6.1	E
6473	Need for Speed: Most Wanted	GBA	2005.0	Racing	0.19	0.07	0.00	0.00	NaN	8.3	E
11715	Need for Speed: Most Wanted	PC	2012.0	Racing	0.00	0.06	0.00	0.02	82.0	8.5	T

In [16]:

```
result_2 = games.query('name == "Madden NFL 07"')
result_2.head(9)
```

Out[16]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
241	Madden NFL 07	PS2	2006.0	Sports	3.63	0.24	0.01	0.61	84.0	8.2	E

972	Madden NFL 07	X360	2006.0	Sports	1.66	0.00	0.01	0.13	80.0	6	E
2035	Madden NFL 07	XB	2006.0	Sports	0.97	0.03	0.00	0.03	83.0	8.7	E
2479	Madden NFL 07	PSP	NaN	Sports	0.77	0.03	0.00	0.04	78.0	6.6	E
3229	Madden NFL 07	GC	2006.0	Sports	0.48	0.13	0.00	0.02	82.0	9.2	E
3838	Madden NFL 07	PS3	2006.0	Sports	0.47	0.00	0.01	0.04	76.0	4.2	E
4006	Madden NFL 07	Wii	2006.0	Sports	0.46	0.00	0.00	0.04	81.0	8	E
7372	Madden NFL 07	DS	2006.0	Sports	0.20	0.00	0.00	0.02	70.0	6.5	E
14889	Madden NFL 07	GBA	2006.0	Sports	0.02	0.01	0.00	0.00	68.0	9.3	E

Видим, что название у игры одно, а вот платформы, где эта игра размещена - разные. Поэтому такие повторы среди названий игры не вызывают вопросов.

Случайная находка по пропуску года в игре Madden NFL 07. Видим, что игра запущена в 2006 на всех остальных платформах. Однако и в этом случае заполнить пропуск нельзя. Ведь игра могла впревые выйти как раз на платформе PSP и произойти это могло и раньше 2006 года.

Посмотрим теперь тип данных по значениям колонки name

In [17]:

```
print('Тип данных в колонке name:', games['name'].dtype)
```

Тип данных в колонке name: object

Тип данных object в колонке name соответствует содержимому колонки. Пропуски удалены. Дубликатов нет.

Смотрим другую колонку.

2.2.3 Колонка platform

In [18]:

```
print(' Список платформ игр (колонка platform):\n', \
      list(games['platform'].sort_values().unique()))
if games['platform'].isnull().sum() == 0:
   print('\nОбработка пропусков не требуется, так как их в столбце platform нет!')
else:
   print('Пропусков в столбце platform', \
     games['platform'].isnull().sum(), \
     'шт. или', \
     round(games['platform'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Треубется обработка пропусков.'
        )
print('\nТип данных в колонке platform:', games['platform'].dtype)
```

Список платформ игр (колонка platform):
['2600', '3DO', '3DS', 'DC', 'DS', 'GB', 'GBA', 'GC', 'GEN', 'GG', 'N64', 'NES', 'NG', 'PC', 'PCFX', 'PS', 'PS2', 'PS3', 'PS4', 'PSP', 'PSV', 'SAT', 'SCD', 'SNES', 'TG16', 'WS', 'Wii', 'WiiU', 'X360', 'XB', 'XOne']

Обработка пропусков не требуется, так как их в столбце platform нет!

Тип данных в колонке platform: object

Претензий к значениям в колонке platform - не обнаружено. Тип данных object в колонке platform соответствует содержимому колонки. Пропусков нет. Смотрим другую колонку.

2.2.4 Колонка year_of_release

In [19]:

```
if games['year_of_release'].isnull().sum() == 0:
   print('\nОбработка пропусков не требуется, так как их в столбце year_of_release
нет!')

print('Пропусков в столбце year_of_release', \
     games['year_of_release'].isnull().sum(), \
     'шт. или', \
```

```
    round(games['year_of_release'].isnull().sum() / len(games) * 100, 3), \
    '% от общего числа строк.\n\
Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке
year_of_release'
    )
```

Пропусков в столбце year_of_release 269 шт. или 1.61 % от общего числа строк.
Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке year_of_release

In [20]:

```
games[games['year_of_release'].isna()].sample(10)
```

Out[20]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
4220	Circus Atari	2600	NaN	Action	0.43	0.03	0.00	0.00	NaN	NaN	NaN
1840	Rock Band	PS2	NaN	Misc	0.71	0.06	0.00	0.35	82.0	6.8	T
6356	Rock Revolution	PS3	NaN	Misc	0.25	0.00	0.00	0.02	42.0	6.6	T
7848	Pet Zombies	3DS	NaN	Simulation	0.18	0.00	0.00	0.01	NaN	tbd	T
16522	Virtua Quest	GC	NaN	Role-Playing	0.01	0.00	0.00	0.00	55.0	5.5	T
10317	Happy Feet Two	PS3	NaN	Action	0.09	0.00	0.00	0.01	NaN	NaN	NaN
7724	Dead Space 3	PC	NaN	Action	0.02	0.16	0.00	0.02	78.0	6	M
12784	Tom Clancy's Rainbow Six: Critical Hour	XB	NaN	Shooter	0.04	0.01	0.00	0.00	54.0	3.6	M
13317	Monster Hunter Frontier Online	PS3	NaN	Role-Playing	0.00	0.00	0.05	0.00	NaN	NaN	NaN
3233	Test Drive Unlimited 2	PS3	NaN	Racing	0.16	0.34	0.01	0.12	70.0	6.1	T

In [21]:

```
print('Список уникальных значений колонки year_of_release:\n', \
     games['year_of_release'].sort_values().unique()
    )
```

Список уникальных значений колонки year_of_release:
[1980. 1981. 1982. 1983. 1984. 1985. 1986. 1987. 1988. 1989. 1990. 1991.
1992. 1993. 1994. 1995. 1996. 1997. 1998. 1999. 2000. 2001. 2002. 2003.
2004. 2005. 2006. 2007. 2008. 2009. 2010. 2011. 2012. 2013. 2014. 2015.
2016. nan]

Какой-то закономерности по пропускам не обнаружено. Причина пропусков - возможно техническая - ошибка при переносе/сохранении данных. Возможно не вся информация о годе релиза была указана в открытых источниках. Человеческий фактор также не исключен, например по ошибке или невнимательности данные не были занесены.
Год выпуска важен для исследования, так как будет анализироваться информация в том числе по годам. Заполнить пропуски возможно только вручную, используя данные из открытых источников. Конечно же это нецелесообразно, учитывая временные затраты, а также отсутствие гарантии успешности поисков - не вся информация может оказаться в открытых источниках.
Заполнить пропуски каким-то общим значением в данном случае не получится. Данные даные не за один год, а за множество лет. Информация об играх собрана аж с 1980 года по 2016 год включительно.
Так как пропусков мало - менее 2%, то удалим пропуски в столбце year_of_release.

In [22]:

```
games = games.dropna(subset=['year_of_release'])
if games['year_of_release'].isnull().sum() == 0:
   print('Пропуски в столбце year_of_release удалены успешно!')
else:
```

```
   print('Ahtung!!! Пропуски не удалены')
```
Пропуски в столбце year_of_release удалены успешно!

In [23]:
```
games['year_of_release'].dtype
```
Out[23]:

dtype('float64')

В колонке year_of_release - текущий тип данных float64 необходимо заменить на целочисленный.

В колонке содержится информация о годе. Года измеряются в целых числах, а не вещественных. Помимо эстетической составляющей, целые числа - это основной нестроковый тип данных, который используется для представления категориальных данных.

В нашем случае, колонку с годом релиза в дальнейшем можно будет представить как категориальные данные. При необходимости экономит памяти и увеличивает скорость обработки, а значит это удобнее для хранения.

In [24]:
```
games['year_of_release'] = games['year_of_release'].astype('Int64')
print('После преобразования, тип данных в колонке year_of_release:
"',games['year_of_release'].dtypes, '".')
```
После преобразования, тип данных в колонке year_of_release: " Int64 ".

2.2.5 Колонка genre

In [25]:
```
print(' Список уникальных значений жанра игр(колонка genre):\n',
list(games['genre'].sort_values().unique()))
```
Список уникальных значений жанра игр(колонка genre):
['Action', 'Adventure', 'Fighting', 'Misc', 'Platform', 'Puzzle', 'Racing', 'Role-Playing', 'Shooter', 'Simulation', 'Sports', 'Strategy']

In [26]:
```
if games['genre'].isnull().sum() == 0:
   print('\nОбработка пропусков не требуется, так как их в столбце genre нет!')
else:
   print('Пропусков в столбце genre', \
     games['genre'].isnull().sum(), \
     'шт. или', \
     round(games['genre'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
print('\nТип данных в колонке platform:', games['genre'].dtype)
```
Обработка пропусков не требуется, так как их в столбце genre нет!

Тип данных в колонке platform: object

Никаких претензий к уникальным значениям в колонке с названием жанра игры - не обнаружено. Тип данных соответствует содержимому, пропусков нет.

2.2.6 Колонка rating.

In [27]:
```
print('Уникальных значений в столбце rating:', len(games['rating'].unique()),
'шт.\n')
print(games['rating'].sort_values().unique())
print('\nТип данных в столбце rating:', games['rating'].dtypes)
```
Уникальных значений в столбце rating: 9 шт.

['AO' 'E' 'E10+' 'EC' 'K-A' 'M' 'RP' 'T' nan]

Тип данных в столбце rating: object

Текущий тип данных в столбце rating это object. В принципе, этот тип данных не противоречит содержимому. Для расшифровки содержимого обратимся к открытым источникам , описывающим подробнее значения из столбца.
Данные из столбца **rating** содержат оценку, которая присваивается **ESRB** (Entertainment Software Rating Board). Это неправительственная организация, которая занимается регулированием компьютерных игр и их рекламных кампаний. У организации есть собственная система возрастных рейтингов.
Рейтинг игр ESRB основан на их содержании, аналогично рейтинговым системам кино, и состоит из двух частей - знака рейтинга и краткого описания содержимого. Маркировка рейтинга игр выглядит так:

- EC (Early childhood) — Для детей младшего возраст».
- E (Everyone) — Для всех. Первоначально K-A (Kids to Adults)
- E10+ (Everyone 10 and older) — Для всех от 10 лет и старше.
- T (Teen) — Подросткам.
- M (Matur) — Для взрослых.
- AO (Adults Only 18+) — Только для взрослых.
- RP (Rating Pending) — Рейтинг ожидается.

Мы получили из колонки уникальные значения маркировки рейтинга, вот они (за исключением пропусков nan):

- 'AO', 'E', 'E10+', 'EC', 'K-A', 'M', 'RP', 'T.

Маркировка **K-A**, как видим из описания из открытых источников, ранее использовалась, а теперь ее нет среди маркировки, только упоминание рядом с маркировкой "E". Они идентичины, ранее маркировка **K-A** означала Kids to Adults. Сейчас для этого используется маркировка **E**. Маркировка игры как **K-A** по факту является неявным дубликатом маркировки **E**. Данные собраны из открытых источников за разные годы, поэтому этот тип маркировки попал в таблицу (предположительно).
Значения столбца **rating** содержат строковая переменные, которые состоят всего из нескольких разных значений. Если заменить тип данных в столбце **rating** с object на category это позволит сэкономить некоторое количество памяти. Еще можно задать порядок в категориях сортировка и min / max будут использовать логический порядок вместо лексического.
Пока не будем преобразовывать этот столбец в тип данных категориальный, так как не все ясно с пропусками, да и значений в датасете не так много, чтобы беспокоиться о времени обработки. Ведь преобразование в категориальные данные помимо преимущества, дает и вероятность в дальнейшем потерять часть категориальных данных. Например если будет необходимость внести изменения и не отследить на этой стадии, что по внесенным изменениям присвоено место в категории.
Произведем замену неявного дубликата: значение K-A в стобце rating заменим на E. Используем для этого pandas.str.replace() function используется для замены строки другой строкой в переменной или столбце данных.

In [28]:

```
games['rating'] = games['rating'].str.replace('K-A', 'E')
print('Уникальные значения стобце rating после замены K-A на E:')
games['rating'].unique()
```

Уникальные значения стобце rating после замены K-A на E:

Out[28]:

array(['E', nan, 'M', 'T', 'E10+', 'AO', 'EC', 'RP'], dtype=object)

In [29]:

```
print('Пропусков в столбце rating', \
      games['rating'].isnull().sum(), \
      'шт. или', \
      round(games['rating'].isnull().sum() / len(games) * 100, 3), \
      '% от общего числа строк.\n\
Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке rating:'
```

```
    )
```

Пропусков в столбце rating 6676 шт. или 40.601 % от общего числа строк.

Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке rating:

In [30]:

```
games[games['rating'].isna()].sample(10)
```

Out[30]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
368	Rugrats in Paris: The Movie	PS	2000	Action	1.96	1.33	0.00	0.23	NaN	NaN	NaN
16713	Spirits & Spells	GBA	2003	Platform	0.01	0.00	0.00	0.00	NaN	NaN	NaN
2615	Yars' Revenge	2600	1982	Shooter	0.73	0.04	0.00	0.01	NaN	NaN	NaN
639	Need for Speed (2015)	PS4	2015	Racing	0.50	1.50	0.05	0.37	NaN	NaN	NaN
11873	Harukanaru Toki no Naka de 4	PS2	2008	Adventure	0.00	0.00	0.07	0.00	NaN	NaN	NaN
789	Scooby-Doo! Night of 100 Frights	PS2	2002	Platform	1.17	0.72	0.00	0.22	NaN	NaN	NaN
10436	Jikkyou Powerful Pro Yakyuu 10	GC	2003	Sports	0.00	0.00	0.10	0.00	NaN	NaN	NaN
3432	Diablo	PS	1997	Role-Playing	0.29	0.19	0.07	0.04	NaN	NaN	NaN
189	Super Mario Land 3: Wario Land	GB	1994	Platform	2.49	0.98	1.57	0.15	NaN	NaN	NaN
13962	Adidas Power Soccer	PS	1996	Sports	0.02	0.01	0.00	0.00	NaN	NaN	NaN

Пропусков столбце rating очень много - 40 %. Причина пропусков может быть или в сбое при обработке данных, либо отсуттсвие такой информации в открытых источниках изначально. Случайные строки показали, что год релиза, платформа, жанр у игр разный. Из особенностей - данные по оценке критиков и оценке пользователей - также не заполнены. Возможно такая оценка и вовсе не проводилась.

Заполнить самостоятельно эти пропуски можно только вручную, что не целесообразно. Учитывая, что этот рейтинг промаркирован текстом, для дальнейшего удобства восприятия информации заполним все пропуски значением unknown.

In [31]:

```
games['rating'] = games['rating'].fillna('unknown')
if  games['rating'].isnull().sum() == 0:
   print('В столбце rating пропуски успешно заполнены значением "unknown"!')
else:
   print('Ahtung! Что-то пошло не так. Требуется проверка!')
```

В столбце rating пропуски успешно заполнены значением "unknown"!

In [32]:

```
games['rating'].value_counts()
```

Out[32]:

```
unknown    6676
E          3923
T          2905
M          1536
E10+       1393
EC            8
RP            1
AO            1
Name: rating, dtype: int64
```

Итог обработки столбца rating: тип данных object соответствует содержимому колонки. Никаких вопросов к уникальным значениям в колонке rating - не осталось: маркировка идентифицирована, а пропуски заменены значением unknown.

Смотрим другую колонку.

2.2.7 Столбец user_score

In [33]:

```
print('Уникальных значений в столбце user_score:', len(games['user_score'].unique()),
'шт.\n')
print(games['user_score'].sort_values().unique(), '\n')

if games['user_score'].isnull().sum() == 0:
   print('\nОбработка пропусков не требуется, так как их в столбце user_score нет!')
else:
   print('Пропусков в столбце user_score', \
     games['user_score'].isnull().sum(), \
     'шт. или', \
     round(games['user_score'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
```

Уникальных значений в столбце user_score: 97 шт.

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

Пропусков в столбце user_score 6605 шт. или 40.169 % от общего числа строк. Требуется обработка пропусков.

Видим, что помимо огромного числа пропусков nan таблица содержит данные с аббревиатурой tbd.
Из открытых источников найдена расшифровка tbd - To Be Determined, то есть "Будет определено". Очень часто подобное сокращение можно встретить в анонсах компьютерных игр в графе дата выхода. В нашем случае такая аббревиатура применена к данным об оценке пользователя. Это значит что есть обещание провести такую оценку, но по факту данных о такой оценке нет. Можно смело приравнять такое обещание к пропуску. Ведь заменить это значение на какое-то число или ноль будет некорректно. У нас нет информации об оценке игроков и взять ее негде.
Принято решение - заменить tbd на пропуск nan.
Произведем замену и посчитаем получившиеся пропуски.

In [34]:

```
games = games.replace('tbd', np.nan)
print('Пропусков в столбце user_score после замены tbd на nan:',
games['user_score'].isnull().sum(), 'шт.или', \
     round(games['user_score'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк.'
        )
```

Пропусков в столбце user_score после замены tbd на nan: 8981 шт.или 54.619 % от общего числа строк.

Видим, что пропусков стало еще больше. Учитывая, что в датасете присутствуют игры за десятки лет, наличие пропусков не удивляет. Скорее всего такая информация стала собираться не сразу и причин пропусков вероятно несколько. Во-первых, не для всех игр из открытых источников была собрана

информация о рейтинге пользователей. Во-вторых, в принципе не для всех игр такой рейтинг составлялся. Да и в целом такая оценка очень субъективная неясна методика оценки: что сравнивают с чем, какие критерии.
Необходимо принять решение о том, что же делать с таким количеством пропусков, ведь удалить строки без влияния на весь набор данных не получится. Замена на ноль некорректна, так как среди уникальных значений есть значение оценки 0. Да и если столбец будет участвовать в расчетах, например среднего, то замененное значение повлияет на результат, даже если заменим на 0. Метод заполнения с помощью последнего непропущенного значения в прямом и обратном порядке также не подходит - оценка игры никак не зависит от соседних значений сверху или снизу.
Принято решение оставить пропуски в столбце user_score. Отсутствие информации - тоже информация в данном случае. Значение nan не будет мешать расчету показателей.
Разберемся теперь с типом данных.

In [35]:

```
print('\nТип данных в столбце user_score:', games['user_score'].dtypes)
```

Тип данных в столбце user_score: object

Текущий тип данных в столбце user_score это object. Он не подходит для значений, которые хранятся в столбце. Оценка пользователя это вещественное число. Преобразуем тип данных в вещественное число.

In [36]:

```
games['user_score'] =  pd.to_numeric(games['user_score'], errors='coerce')
print('После преобразования, тип данных в столбце user_score:',
games['user_score'].dtype)
```

После преобразования, тип данных в столбце user_score: float64

In [37]:

```
#games['user_score'] = games['user_score'].astype(float)
#print('После преобразования, тип данных в столбце user_score:',
games['user_score'].dtype)
```

Итог обработки столбца user_score: тип данных заменен на float64. Значение в столбце tbd заменено на пропуск, а все попуски в столбце оставлены. Сами пропуски будут информацией.
Смотрим другую колонку.

2.2.8 Колонка critic_score

Оценка критиков - значение максимум 100

In [38]:

```
print('Пропусков в столбце critic_score', \
      games['critic_score'].isnull().sum(), \
      'шт. или', \
      round(games['critic_score'].isnull().sum() / len(games) * 100, 3), \
      '% от общего числа строк.\n\
Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке
critic_score:'
     )
```

Пропусков в столбце critic_score 8461 шт. или 51.457 % от общего числа строк.
Выведем случайные 10 строк датафрейма с фильтром на пропуски по колонке critic_score:

In [39]:

```
games[games['critic_score'].isna()].sample(10)
```

Out[39]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
14500	Far East of Eden Shinden	NG	1995	Fighting	0.00	0.00	0.03	0.00	NaN	NaN	unknown

7327	Yoostar2	X360	2011	Misc	0.11	0.09	0.00	0.02	NaN	NaN	unknown
1234	F1 Race	NES	1984	Racing	0.00	0.00	1.52	0.00	NaN	NaN	unknown
6509	Yakuza: Ishin	PS3	2014	Action	0.00	0.00	0.26	0.00	NaN	NaN	unknown
16047	Accel World: Kasoku no Chouten	PS3	2013	Adventure	0.00	0.00	0.02	0.00	NaN	NaN	unknown
13104	Expendable	PS	1999	Action	0.03	0.02	0.00	0.00	NaN	NaN	unknown
6385	Just Dance Kids 2	Wii	2011	Misc	0.25	0.00	0.00	0.02	NaN	NaN	E
3770	Virtual Soccer	SNES	1993	Sports	0.00	0.00	0.53	0.00	NaN	NaN	unknown
16350	Ultraman Fighting Evolution 0	PSP	2006	Fighting	0.00	0.00	0.01	0.00	NaN	NaN	unknown
10372	Carnage Heart	PS	1995	Strategy	0.01	0.01	0.09	0.01	NaN	NaN	unknown

Пропусков в столбце critic_score также много, как и в столбце user_score. Причины таких пропусков, предположительно, такие же как и у значений оценки пользователей: не для всех игр такая оценка проводилась, т.е. отсутствие такой оценки в принципе, также не идеальный сбор данных. Не исключено, что для игр на одной платформе оценка есть, а для такой же игры на другой платформе такая оценка не проводилась.
Решение по пропускам в колонке critic_score - оставляем. В этом случае отсутствие данных - это тоже данные.

In [40]:

```
print('Уникальных значений в столбце critic_score:',
len(games['critic_score'].unique()), 'шт.\n')
print(games['critic_score'].sort_values().unique())
print('\nТип данных в столбце critic_score:', games['critic_score'].dtypes)
```

```
Уникальных значений в столбце critic_score: 82 шт.

[13. 17. 19. 20. 21. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53.
 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71.
 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89.
 90. 91. 92. 93. 94. 95. 96. 97. 98. nan]

Тип данных в столбце critic_score: float64
```

Оценка критиков дается в целых числах - в баллах, где 100 это максимальный балл. Текущий тип данных - вещественный. Переведем этот тип данных в целочисленный.
Столбец содержит nan, а поскольку NaN является числом с плавающей запятой, столбец целых чисел даже с одним пропущенным значением преобразуется в dtype с плавающей запятой. В качестве альтернативы используем строковый псевдоним dtype='Int64' (обратите внимание на заглавную букву "I"). Pandas может представлять целочисленные данные с возможными отсутствующими значениями, используя arrays.IntegerArray . Это extension type , реализованный в pandas. Источник
Применим pandas.array().Этот метод используется для создания массива из последовательности данных желаемого типа.

In [41]:

```
games['critic_score'] = pd.array(games['critic_score'], dtype=pd.Int64Dtype())
print(
    'Тип значений в столбце после применения метода pandas.array():', \
      games['critic_score'].dtype, '\nСписок уникальных значения в столбце
critic_score:')
games['critic_score'].sort_values().unique()
```

Тип значений в столбце после применения метода pandas.array(): Int64
Список уникальных значения в столбце critic_score:

Out[41]:

```
<IntegerArray>
[ 13,  17,  19,  20,  21,  23,  24,  25,  26,  27,  28,  29,  30,
  31,  32,  33,  34,  35,  36,  37,  38,  39,  40,  41,  42,  43,
  44,  45,  46,  47,  48,  49,  50,  51,  52,  53,  54,  55,  56,
  57,  58,  59,  60,  61,  62,  63,  64,  65,  66,  67,  68,  69,
  70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  80,  81,  82,
  83,  84,  85,  86,  87,  88,  89,  90,  91,  92,  93,  94,  95,
  96,  97,  98, <NA>]
Length: 82, dtype: Int64
```

Итог обработки столбца critic_score: тип данных заменен на Int64. Все пропуски в столбце оставлены.
Смотрим другую колонку.

2.2.9 Колонка na_sales

Колонка содержит данные продажах в Северной Америке (миллионы проданных копий).

In [42]:

```
print('Уникальных значений в столбце na_sales:', len(games['na_sales'].unique()),
'шт.')

if games['na_sales'].isnull().sum() == 0:
   print('Обработка пропусков не требуется, так как их в столбце na_sales нет!')
else:
   print('Пропусков в столбце na_sales', \
     games['na_sales'].isnull().sum(), \
     'шт. или', \
     round(games['na_sales'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
print('Тип значений в столбце na_sales:', games['na_sales'].dtype)
```

Уникальных значений в столбце na_sales: 401 шт.
Обработка пропусков не требуется, так как их в столбце na_sales нет!
Тип значений в столбце na_sales: float64

Столбец na_sales не требует обработки. Тип данных float64 соответствует содержимому колонки в которой содержатся данные о продажах (миллионы проданных копий) как числа с дробной частью. Смотрим другую колонку.

2.2.10 Колонка eu_sales

Колонка содержит данные о продажах в Европе (миллионы проданных копий)

In [43]:

```
print('Уникальных значений в столбце eu_sales:', len(games['eu_sales'].unique()),
'шт.')

if games['eu_sales'].isnull().sum() == 0:
   print('Обработка пропусков не требуется, так как их в столбце na_sales нет!')
else:
   print('Пропусков в столбце eu_sales', \
     games['eu_sales'].isnull().sum(), \
     'шт. или', \
     round(games['eu_sales'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
```

```
print('Тип значений в столбце eu_sales:', games['eu_sales'].dtype)
```

Уникальных значений в столбце eu_sales: 307 шт.
Обработка пропусков не требуется, так как их в столбце na_sales нет!
Тип значений в столбце eu_sales: float64

Столбец eu_sales не требует обработки. Тип данных float64 соответствует содержимому колонки в которой содержатся данные о продажах (миллионы проданных копий) как числа с дробной частью. Смотрим другую колонку.

2.2.11 Колонка jp_sales

Колонка содержит данные о продажах в Японии (миллионы проданных копий).

In [44]:

```
print('Уникальных значений в столбце jp_sales:', len(games['jp_sales'].unique()),
'шт.')

if games['jp_sales'].isnull().sum() == 0:
   print('Обработка пропусков не требуется, так как их в столбце jp_sales нет!')
else:
   print('Пропусков в столбце jp_sales', \
     games['jp_sales'].isnull().sum(), \
     'шт. или', \
     round(games['jp_sales'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
print('Тип значений в столбце jp_sales:', games['jp_sales'].dtype)
```

Уникальных значений в столбце jp_sales: 244 шт.
Обработка пропусков не требуется, так как их в столбце jp_sales нет!
Тип значений в столбце jp_sales: float64

Столбец jp_sales не требует обработки. Тип данных float64 соответствует содержимому колонки в которой содержатся данные о продажах (миллионы проданных копий) как числа с дробной частью.

2.2.12 Колонка other_sales

Колонка содержит данные о продажах в других странах (миллионы проданных копий).

In [45]:

```
print('Уникальных значений в столбце other_sales:',
len(games['other_sales'].unique()), 'шт.')

if games['other_sales'].isnull().sum() == 0:
   print('Обработка пропусков не требуется, так как их в столбце other_sales нет!')
else:
   print('Пропусков в столбце other_sales', \
     games['other_sales'].isnull().sum(), \
     'шт. или', \
     round(games['other_sales'].isnull().sum() / len(games) * 100, 3), \
     '% от общего числа строк. Требуется обработка пропусков.'
        )
print('Тип значений в столбце other_sales:', games['other_sales'].dtype)
```

Уникальных значений в столбце other_sales: 155 шт.
Обработка пропусков не требуется, так как их в столбце other_sales нет!
Тип значений в столбце other_sales: float64

Столбец other sales не требует обработки. Тип данных float64 соответствует содержимому колонки в которой содержатся данные о продажах (миллионы проданных копий) как числа с дробной частью. Все колонки обработаны

2.3 Подсчет суммарных продаж млн. копий игр во всех регионах.

Посчитаем суммарные продажи проданных млн. копий игр во всех регионах и запишем их в отдельный столбец total_sales. Для этого сложим все продажи.

Сперва выведем название колонок. Затем добавим новый столбец total_sales.

In [46]:

```
games.columns
```

Out[46]:

```
Index(['name', 'platform', 'year_of_release', 'genre', 'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'critic_score', 'user_score', 'rating'],
      dtype='object')
```

In [47]:

```
games['total_sales'] =
games[['na_sales','eu_sales','jp_sales','other_sales']].sum(axis = 1)
print('\nПосмотрим на случайные 3 строки датафрейма games с новым столбцом
total_sales:')
games.sample(3)
```

Посмотрим на случайные 3 строки датафрейма games с новым столбцом total_sales:

Out[47]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	total_sales
14046	Nobunaga's Ambition: Sphere of Influence - Sen...	PS3	2016	Misc	0.00	0.00	0.04	0.00	<NA>	NaN	unknown	0.04
2293	MLB 2005	PS2	2004	Sports	0.44	0.35	0.00	0.12	78	7.6	E	0.91
2977	Mortal Kombat Mythologies: Sub-Zero	PS	1997	Fighting	0.38	0.26	0.00	0.04	<NA>	NaN	unknown	0.68

Колонку total_sales переместим в другое место датафрейма, между other_sales и critic_score. Используем такое решение

In [48]:

```
games = pd.DataFrame(games,columns=['name', 'platform', 'year_of_release', 'genre',
'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'total_sales', 'critic_score', 'user_score',
'rating'])
print('\nПосмотрим на случайные 3 строки датафрейма games после смещения стобца
total_sales:')
games.sample(3)
```

Посмотрим на случайные 3 строки датафрейма games после смещения стобца total_sales:

Out[48]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	total_sales	critic_score	user_score	rating
12012	IA/VT Colorful	PSV	2015	Misc	0.00	0.0	0.07	0.0	0.07	<NA>	NaN	unknown
8162	Super Robot Taisen A Portable	PSP	2008	Strategy	0.00	0.0	0.18	0.0	0.18	<NA>	NaN	unknown
13405	SpongeBob SquarePants: Plankton's Robotic Revenge	X360	2013	Shooter	0.04	0.0	0.00	0.0	0.04	<NA>	6.1	E10+

2.4 Итоги раздела

Изучили общую информацию и выполнили предобработку данных.

Произведена замена названий столбцов — приведение в нижний регистр.

Явных дубликатов не было.

Удалено 2 строки неявных дубликатов.

Удалены строки с пропусками в столбцах name (1 строка) и year_of_release (269 строк).
Пропуски в столбце rating заменены на строковое значение — unknown.
Было выяснено, что значение tbd в столбце user_score можно считать как пропуск, заменили это значение на пропуск, а сами пропуски оставили, так как адекватной замены для заполнения пропуска нет.
Аналогично с пропусками поступили в столбцах critic_score — пропуски оставлены. Устаревшую маркировку рейтинга K-A заменили на современное обозначение E.
В следующих колонках тип данных заменен на соответствующий значениям, которые в нем содержатся:

- year_of_release - текущий тип данных float64 заменен на целочисленный — Int64.
- user_score - тип данных заменен на float64.
- critic_score - тип данных заменен на Int64.

Добавили столбец в датафрейм total_sales, в который поместили результаты суммирования продаж во всех регионах.
Можно приступать к исследовательскому анализу.

3 Проведение исследовательского анализа данных:

3.1 Сколько игр выпускалось в разные годы. Важны ли данные за все периоды.

Данные о годе выпуска содержатся в колонке year_of_release. Тут собраны данные за много лет.

In [49]:

```
print('Уникальных значений в столбце year_of_release:',
len(games['year_of_release'].unique()), 'шт.')
```

Уникальных значений в столбце year_of_release: 37 шт.

Данные аж за 37 лет.Конечно удобнее будет посмотреть на столбчатую диаграмму. Она хорошо подходит для дискретных и категориальных величин. В нашем случае посмотрим количества выпускаемых игр по годам.

In [50]:

```
games.pivot_table(
   index='year_of_release', values='name',
   aggfunc='count').plot.bar(
   figsize=(10,5), alpha=0.4, color='yellow', ec='black', linewidth=1, grid=True);

plt.title('Количество выпускаемых игр по годам')
plt.xlabel('Год')
plt.ylabel('Количество игр')
plt.show()
```

Пик количества выпускаемых игр приходится на 2008 и 2009 годы.

В 2010 и 2011 годах количество игр плавно снижается, а затем в 2012 году произошло резкое снижение количество выпускаемых игр.

Все остальные годы с 2013 по 2016 уровень выпускаемых игр колеблется от 500 до 600 в год.

С 1980 по 1990 год количество игр было выпущено минимально. Это не удивительно. ПК в то время могли себе позволить не все.

С 1991 года пошел рост выпуска игр. И дальше по мере развития компьютеров, программирования, выпуска игровых приставок рост продолжился. С того моменты было выпущено много новых игровых консолей, какие-то покинули рынок и больше не поддерживаются.

Стоит учитывать и экономический кризис, который обвалил не только индустрию игр.

Есть более менее стабильные 2012-2016 годы после резкого падения количества выпуска игр.

Считаю, что данные за периоды старше 5 лет нет смысла брать. Следует посмотреть и другие показатели, как они меняются по годам.

3.2 Как менялись продажи по платформам.

Построим сводную таблицу platform в которую агрегируем данные по платформе и общей продаже.

In [51]:

```
platform = games.pivot_table(
   index='platform', values='total_sales',
   aggfunc='sum').sort_values(by='total_sales', ascending=True)
platform
```

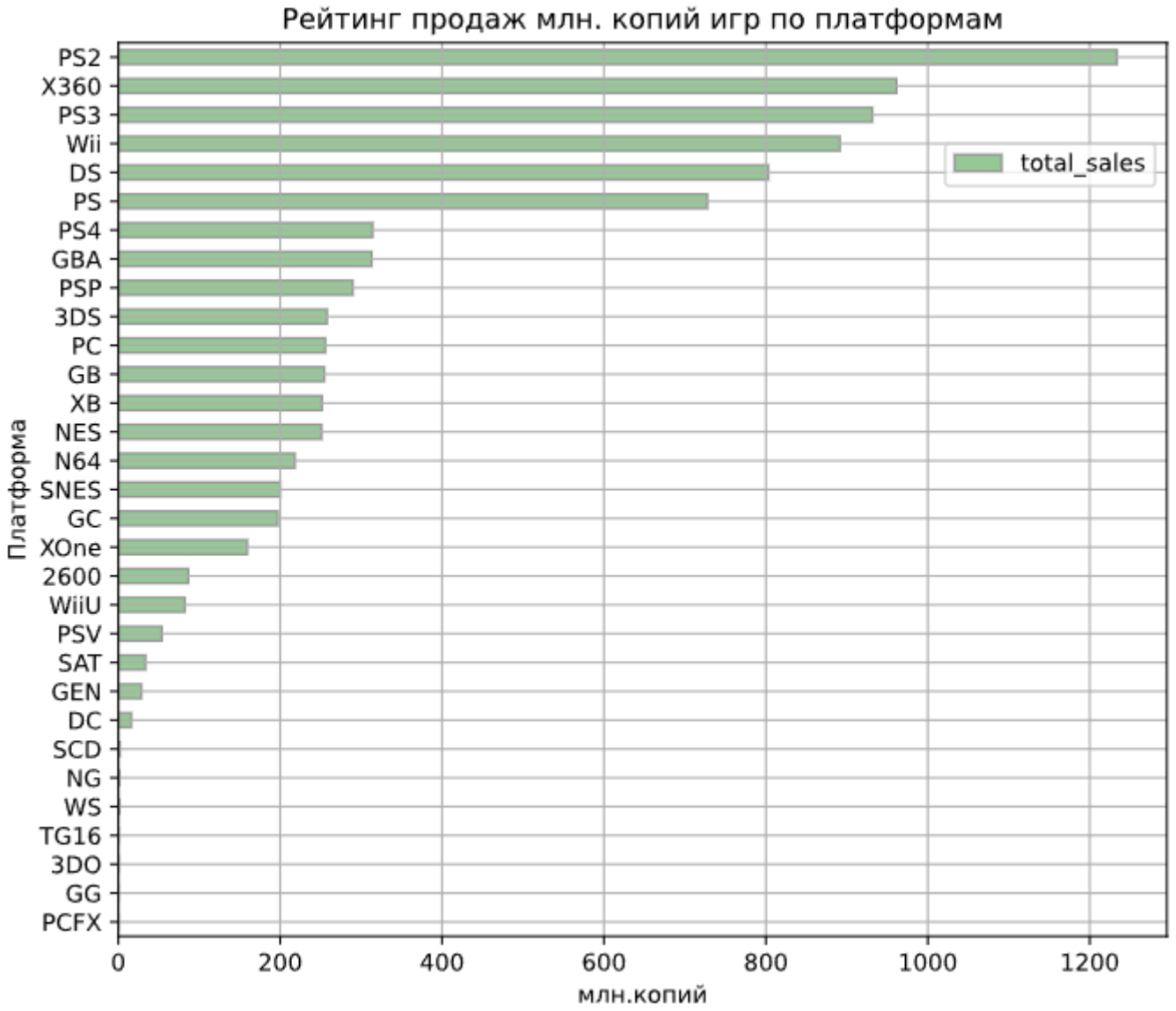
Out[51]:

	total_sales
platform	
PCFX	0.03
GG	0.04
3DO	0.10
TG16	0.16
WS	1.42
NG	1.44
SCD	1.86
DC	15.95
GEN	28.35

SAT	33.59
PSV	53.81
WiiU	82.19
2600	86.48
XOne	159.32
GC	196.73
SNES	200.04
N64	218.01
NES	251.05
XB	251.57
GB	254.43
PC	255.76
3DS	257.81
PSP	289.53
GBA	312.88
PS4	314.14
PS	727.58
DS	802.78
Wii	891.18
PS3	931.33
X360	961.24
PS2	1233.56

In [52]:

```
platform.plot.barh(
   figsize=(8,7), alpha=0.4, color='green', ec='black', linewidth=1, grid=True);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Рейтинг продаж млн. копий игр по платформам')
plt.xlabel('млн.копий')
plt.ylabel('Платформа')
plt.show()
```



По диаграмме видно, что в топе по продажам - 6 платформ. Объявим переменную top_platform в которую сохраним список этих платформ. Затем построим распределение по годам для платформ из этого списка, чтобы посмотреть какой характерный срок появления новых и исчезания старых платформ.

In [53]:

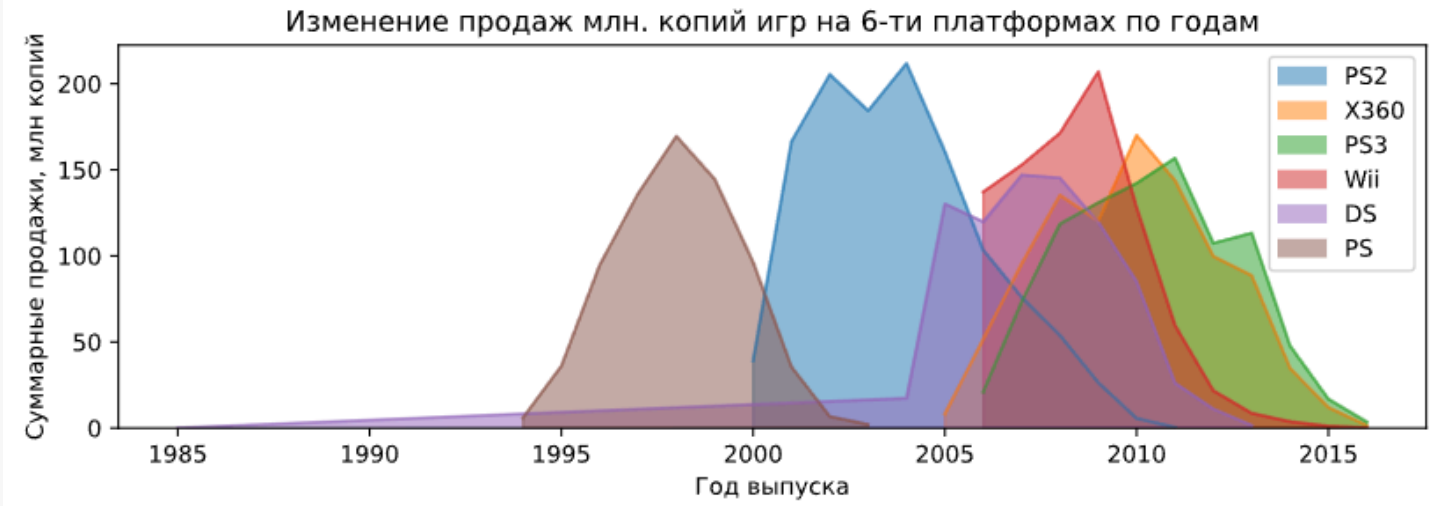
```
top_platform = games.pivot_table(
    index='platform',
    values='total_sales',
    aggfunc='sum').sort_values(
    by='total_sales', ascending=False).reset_index().head(6)['platform'].tolist()
print('Наибольшие суммарные продажи млн. копий у этих 6 платформ:\n', top_platform)
```

Наибольшие суммарные продажи млн. копий у этих 6 платформ:
['PS2', 'X360', 'PS3', 'Wii', 'DS', 'PS']

In [54]:

```
for name in top_platform:
    (
    games.query('platform == @name')
    .pivot_table(index = 'year_of_release', values = 'total_sales', aggfunc = 'sum')
    .sort_values('year_of_release', ascending = False)['total_sales']
    .plot.area(figsize = (10, 3), stacked=False, label=name)
    )
    plt.title('Изменение продаж млн. копий игр на 6-ти платформах по годам')
    plt.xlabel('Год выпуска')
```

```
    plt.ylabel('Суммарные продажи, млн копий')
    plt.legend()
```



На графике виден выброс у платформы DS. Из открытых источников выяснено, что эта игровая платформа появилась в 2004 году. Видимо появление в датасете этой платформы с выпущенными играми в 1985 году - это ошибка ввода. Построим срез по этой платформе за года до 2004.

In [55]:

```
print(
    'Количество строк с ошибочными данными платформы DS:', \
      len(games.query('platform == "DS" & year_of_release < 2004')), \
    '\nПосмотрим на эти строки')
games.query('platform == "DS" & year_of_release < 2004')
```

Количество строк с ошибочными данными платформы DS: 1
Посмотрим на эти строки

Out[55]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	total_sales	critic_score	user_score	rating
15957	Strongest Tokyo University Shogi DS	DS	1985	Action	0.0	0.0	0.02	0.0	0.02	<NA>	NaN	unknown

Удаляем строку без сожаления. Индекс строки Эта платформа начала выпускаться в 2004 году. Явная ошибка в данных, которая искажает график. После удаления выброса построим график снова.

In [56]:

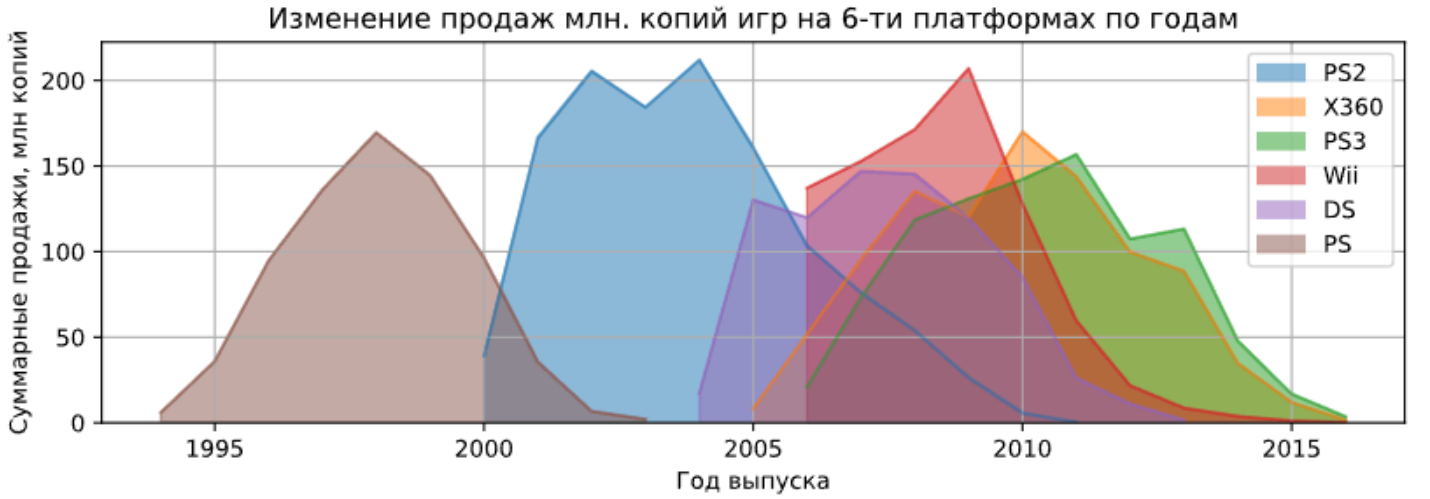
```
games = games.drop(index= 15957)
if (len(games.query('platform == "DS" & year_of_release < 2004')) == 0):
    print('Удаление прошло успешно!')
else:
    ('Ahtung! Что-то пошло не так!')
```

Удаление прошло успешно!

In [57]:

```
for name in top_platform:
    (
    games.query('platform == @name')
    .pivot_table(index = 'year_of_release', values = 'total_sales', aggfunc = 'sum')
    .sort_values('year_of_release', ascending = False)['total_sales']
    .plot.area(figsize = (10, 3), stacked=False, label=name, grid=True)
    )
    plt.title('Изменение продаж млн. копий игр на 6-ти платформах по годам')
    plt.xlabel('Год выпуска')
    plt.ylabel('Суммарные продажи, млн копий')
```

```
plt.legend()
```



Проанализируем данные на графике по платформам и периоду продаж игр на них за период до 2016 года. Сразу отметим, что все продажи в 2015 году есть только у трех платформ и каждая их них уже на своих минимальных значениях продаж. В 2016 году продажи минимальны у всех трех платформ.

- **платформа PS** - продажи плавно нарастали начиная с 1994 года. Закончились продажи в 2003 году, пик продаж пришелся на 1998 год, затем пошел плавный спад. Весь цикл продлился **9 лет с 1994 по 2003 годы**;
- **платформа PS2** - продажи начались в 2000 году достаточно резко выросли за год - более чем в 3 раза с даты релиза. Закончились продажи игр на этой платформе в 2011 году. Видим два пика продаж 2002 и 2004 годах затем резкий спад продаж с 2005 года, и далее продажи продолжались снижаться вплоть до 2011 года. Весь цикл продлился **11 лет с 2000 по 2011 годы**;
- **платформа DS** - продажи начались в 2004 году, также за год выросли более чем в 2 раза, пик продаж в 2008 -2009 году и затем падение вплоть до 2013 года. Весь цикл занял **9 лет с 2004 по 2013 год**;
- **платформа X360** - продажи начались с 2005 года и стабильно росли до 2008 года, в 2009 году продажи немного просели, после чего достигли максимума в 2010 году и затем стабильно снижались вплоть до 2016 года.
- **платформа Wii** - продажи начались в 2006 году причем это самый высокий рост продаж в первый год, далее продажи росли и достигли максимума в 2009 году, после чего произошло резкое падение продаж вплоть до 2016 года. Весь цикл занял **10 лет с 2006 по 2016 год**;
- **платформа PS3** - в 2006 году начались продажи игр на этой платформе и стабильно росли с 2006 по 2008 год. Плавный рост сохранился дальше вплоть до 2011 год, где был максимум, в 2012 продажи упали и сохранялись на одинаковом уровне до 2013 годах, после чего резко пошло снижение продаж. Из всех платформ в 2015 году у этой больше всех продаж, правда не намного. Весь цикл занял **10 лет с 2006 по 2016 год**.

Если в период 1994 - 2000 год лидером по продаже была всего одна платформа PS. В 2000 году появилось уже две лидирующие по продажам платформы PS - с падением продаж, а PS2 с огромным ростом продаж. пока в2004 годах не появилась DS, что повлияло на текущего в этот год лидера - у него продажи упали, а у новинки взлетели, в 2005 году появилась еще одна новинка X360, а в 2006 году аж две Wii и PS3. Одни игровые платформы сменяли другие. В 2005 году 3 платформы, в 2006 году уже 5 платформ игровых, после 2011 года их снова стало 4, затем 3, пока продажи на этих платформах в 2016 году практически не прекратились совсем. Жизненный цикл у всех рассмотренных платформ 9 -11 лет. Платформа PS по своему плавному росту продаж не показательна, так как условия технические поменялись с того времени, а также сам рынок покупателей изменился. По всем остальным платформам видно, что рост продаж стремительный первый, второй год. Поэтому для дальнейшего анализа платформ нужно брать те платформы, дата релиза которых не больше 2-3 лет и которые показали большой рост продаж на 2-3 год. Ведь для анализа нам важно найти перспективные платформы с ростом продаж, а не те, которые находятся на закате жизненного цикла.

3.3 Определим актуальный период.

В результате исследования по количества выпускаемых игр мы определили, что период для анализа лучше взять за последние 5 лет. По результатам исследования жизненного цикла платформ этот срок мы сократили до двух трех лет из-за особенностей жизненного срока платформ.

Нам известно, что за 2016 год данные могут быть не полные.
Так как нам необходимо построить прогноз на 2017 год, то принято решение, что **актуальный период для анализа ограничен годами с2014 по 2016 включительно.**

3.4 Определим какие платформы лидируют по продажам, растут или падают.

Сформируем датафрейм, в который войдут игры, выпущенные с 2014 года и назовем его actual_period

In [58]:

```
actual_period = games.query('year_of_release > 2013')
print('Рейтинг платформ по количеству проданных млн. копий за период 2014-2016 год:')
actual_period.pivot_table(
   index=['platform'],
   values = 'total_sales',
   aggfunc='sum').sort_values(
   by='total_sales', ascending = False)
```

Рейтинг платформ по количеству проданных млн. копий за период 2014-2016 год:

Out[58]:

	total_sales
platform	
PS4	288.15
XOne	140.36
3DS	86.68
PS3	68.18
X360	48.22
WiiU	42.98
PC	27.05
PSV	22.40
Wii	5.07
PSP	0.36

Выберем несколько потенциально прибыльных платформ. Посмотрим начало жизненного цикла этих платформ.

In [59]:

```
top_platform_actual = games.query('year_of_release > 2013').pivot_table(
   index='platform',
   values='total_sales',
   aggfunc='sum').sort_values(
   by='total_sales', ascending=False).reset_index()['platform'].tolist()
top_platform_actual
```

Out[59]:

['PS4', 'XOne', '3DS', 'PS3', 'X360', 'WiiU', 'PC', 'PSV', 'Wii', 'PSP']

In [60]:

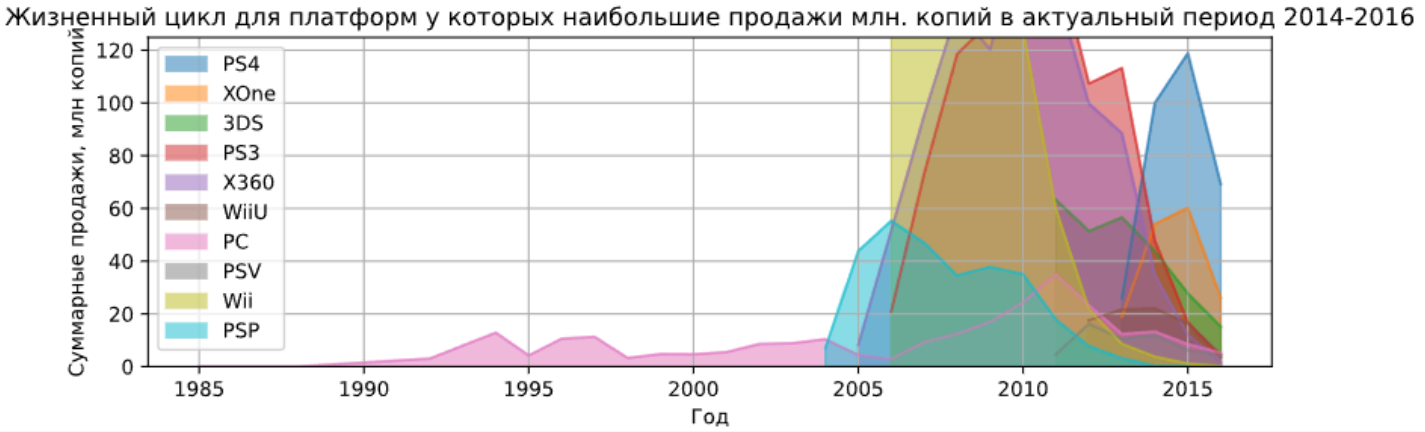
```
print('Наибольшие суммарные продажи млн. копий в 2014-2016 годах у этих платформ:\n',
top_platform_actual)
```

Наибольшие суммарные продажи млн. копий в 2014-2016 годах у этих платформ:
['PS4', 'XOne', '3DS', 'PS3', 'X360', 'WiiU', 'PC', 'PSV', 'Wii', 'PSP']

In [61]:

```
for t in top_platform_actual:
   (
   games.query('platform == @t')
   .pivot_table(index = 'year_of_release', values = 'total_sales', aggfunc = 'sum')
   .sort_values('year_of_release', ascending = False)['total_sales']
```

```
    .plot.area(figsize = (10, 3), stacked=False, label=t, grid=True)
    )
    plt.title('Жизненный цикл для платформ у которых наибольшие продажи млн. копий в актуальный период 2014-2016')
    plt.xlabel('Год')
    plt.ylabel('Суммарные продажи, млн копий')
    plt.legend()
```



Для такого количества платформ уже тяжелее разобраться на графике, однако видно лидеров, у которых жизненный цикл находится на пике, а нам только и нужна эта информация. Итак выберем те платформы, которые будут лидерами продаж, а также с подходящим под наши критерии жизненным циклом, а именно - не угасающим, растущим:

- PS4. Выбор обусловлен тем,что платформа лидер у нее самый высокий результат по продажам в 2015 году и подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. падение в 2016 году не показательно. помним, что данные могут быть не полными. Однако и при таких данных в 2016 году это самые высокие показатели.
- XOne. Выбор обусловлен тем, что у платформы второй результат по количеству продаж как в 2015 году, так и в 2016 году. А также подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. Опять же падение продаж в 2016 году и второе место - не показательно. помним, что данные могут быть неполными и место по продажам может быть выше.
- WiiU. На графике эта платформа не показывает каких-то грандиозных выдающихся продаж, в 2015 году есть небольшой спад. Но все же она по сроку жизненного цикла и уровню продаж в 2014 году может занимать третье место.
- PC. Выбор обусловлен уникальным жизненным циклом. Платформа жизнеспособна на протяжении с всего изучаемого периода с 1985. Пик был в 2011 году. Она то становится более популярное, то менее. Однако до сих пор актуальна. Хотя конечно показатели по продажам за все предыдущие года не такой высокий как у других платформ. Зато стабильный. Оценим теперь каждую из 10 лидирующих платформ отдельно: как менялось количество выпускаемых игр и общая прибыль за актуальный период. Подтверждаются ли выводы о выборе PS4, XOne и PC.

Для этого построим графики. Ранее мы ввели переменную top_platform_actual, в которую записали список из названий платформ в актуальном периоде. Используем ее а также сводную таблицу actual_period в которой находится информация по годам продаж игр с 2014 года.

В модуле plotly.graph_objects есть специальные объекты под названием graph objects(графические объекты) Импортируем этот модуль и построим график.

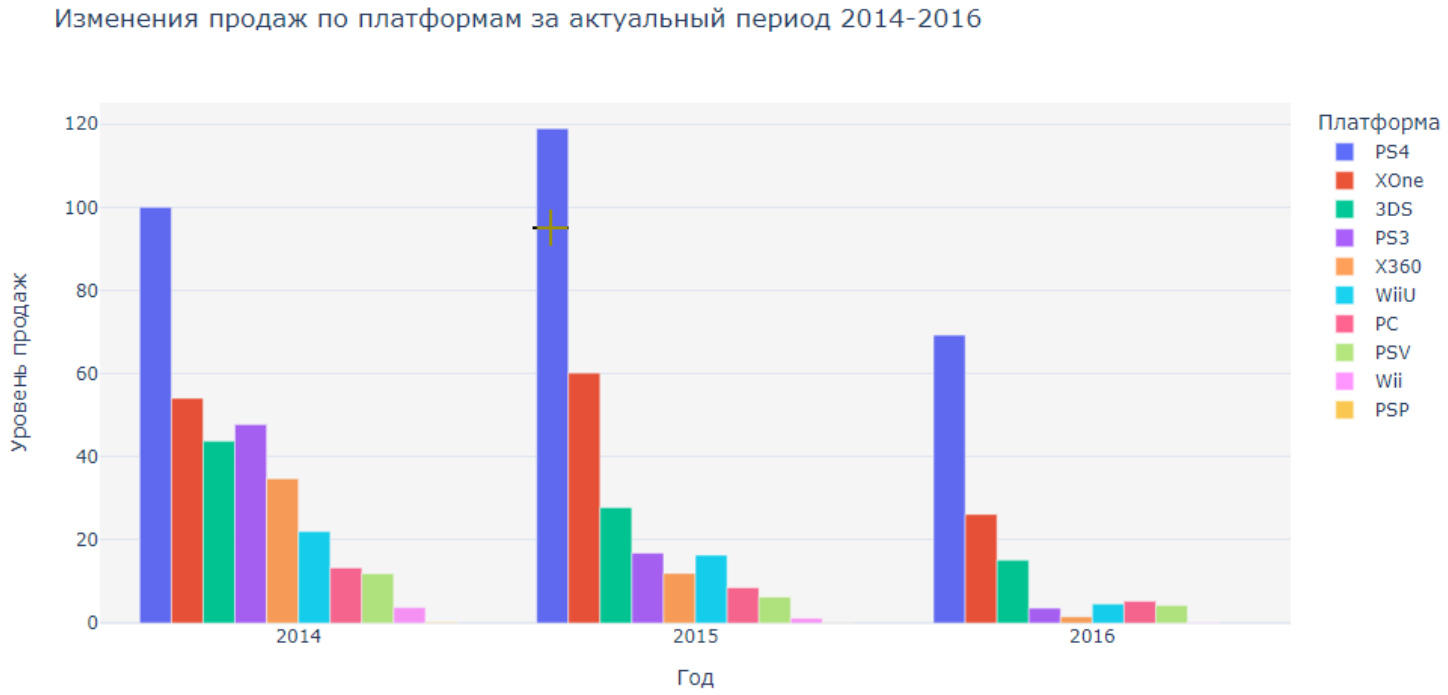
In [62]:

```
import plotly.graph_objects as go
bar_actual = []

for index in top_platform_actual:
    bar_actual.append(go.Bar(
        x=actual_period[actual_period.platform == index].groupby('year_of_release')['total_sales'].sum().index,
        y=actual_period[actual_period.platform == index].groupby('year_of_release')['total_sales'].sum(), name=index)
```

```
            )

layout = {'title': 'Изменения продаж по платформам за актуальный период 2014-2016'}
fig = go.Figure(data=bar_actual, layout=layout)
fig.layout.template = 'plotly_white'
fig.update_layout(legend_title_text = "Платформа")
fig.update_xaxes(title_text="Год")
fig.update_yaxes(title_text="Уровень продаж")
fig.show()
```



Проанализируем 2014 и 2015 год. Помним, что данных за 2016 год могут быть неполными. И фактического падения продаж может и не быть. Однако общие тенденции и по имеющимся данным можно посмотреть.

Полученные ранее данные о двух лидерах подтверждаются и на этой диаграмме. безусловный лидер это PS4. Второй по счету это XOne. PS3 в 2015 году на небольшое количество опередил по продажам WiiU, но при этом и по уровню падения продаж тоже опередил, а учитывая что жизненный цикл этой платформы на закате, а WiiU это свежая по дате выпуска платформа, то остановим выбор на WiiU.

Остальные показали падение, а не рост.

Платформа PC показывает падение уровня продаж по сравнению с 2014 годом, что было видно и на предыдущем графике, где мы смотрели жизненный цикл топ 10 платформ. При этом была выделена ее живучесть и относительная стабильность. У этой платформы есть свой покупатель. И это неплохо, учитывая такой срок жизни.

Подтвердили выбор четырех платформ для детального изучения. Это PS4, XOne, WiiU и PC. Создадим список top_four с наименованиями этих платформ.

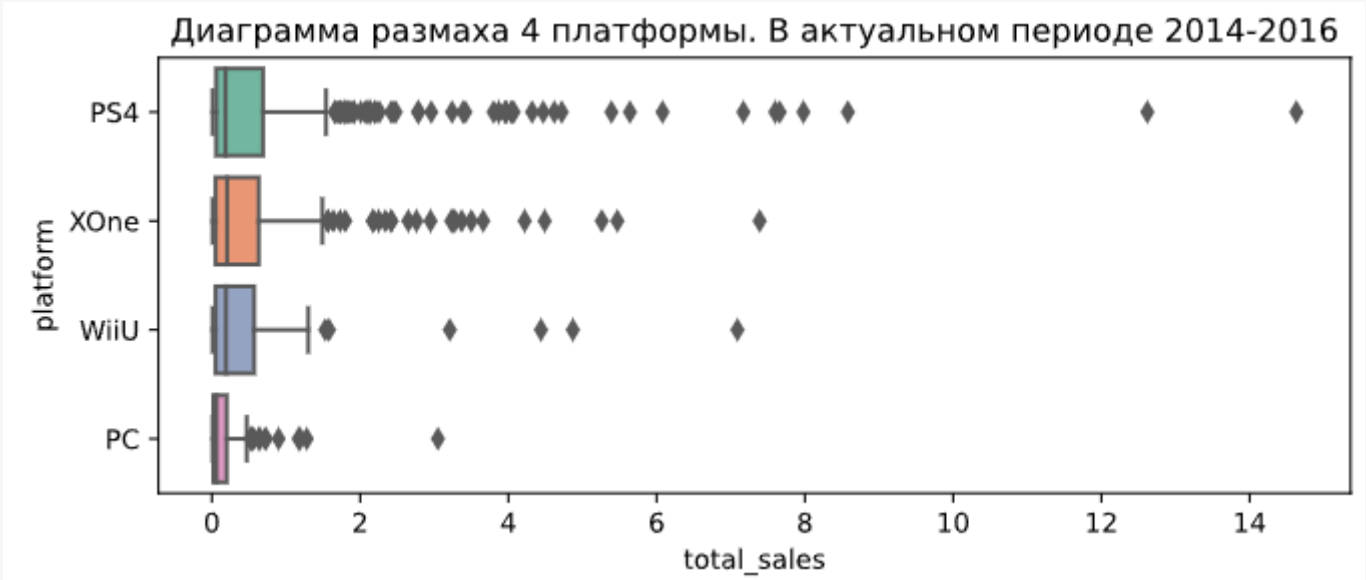
Построим график «ящик с усами» по глобальным продажам для игр в разбивке по этим четырем платформам. Опишем результат. используем ранее созданный датафрейм actual_period с данными о продажах за период 2014-2016 год.

In [63]:

```
top_four = actual_period[(actual_period['platform'] == 'PS4') \
                         | (actual_period['platform'] == 'XOne') \
                         | (actual_period['platform'] == 'WiiU')\
                         | (actual_period['platform'] == 'PC')
                        ]

plt.figure(figsize=(8,3))
sns.boxplot(x='total_sales', y='platform', data=top_four, palette='Set2')
```

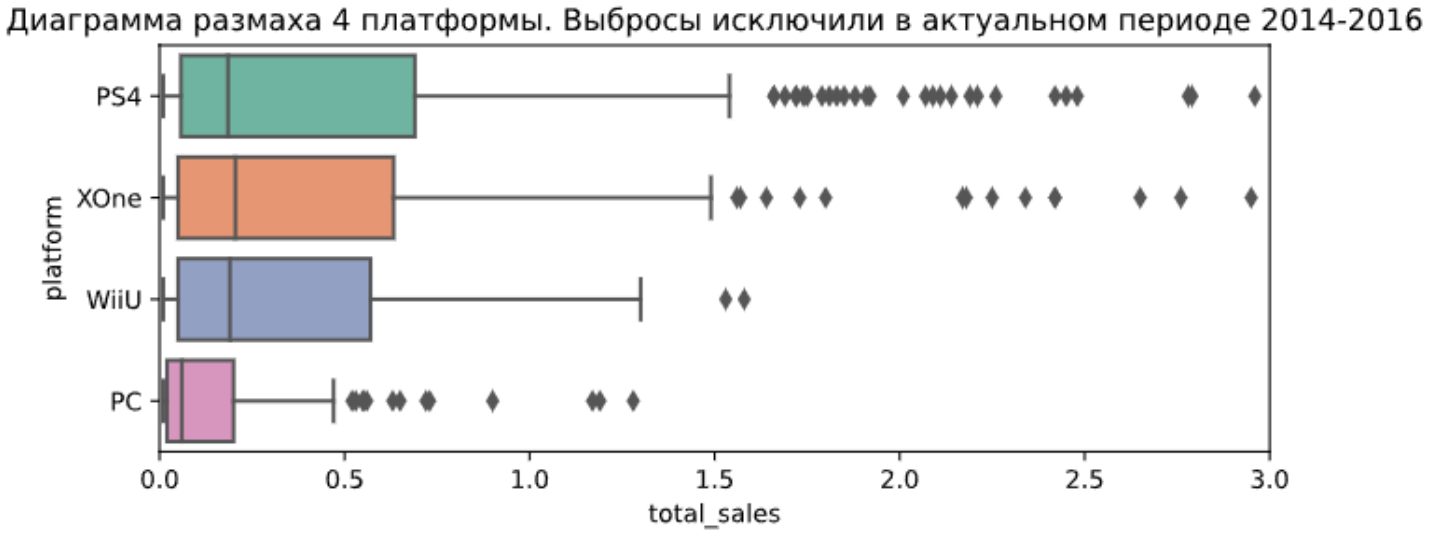
```
plt.title('Диаграмма размаха 4 платформы. В актуальном периоде 2014-2016');
```



Видим очень много выбросов по всем платформам. Зададим ограничения для удобства визуализации данных по такому графику.

In [64]:

```
plt.figure(figsize=(8,3))
sns.boxplot(x='total_sales', y='platform', data=top_four, palette='Set2')
plt.title('Диаграмма размаха 4 платформы. Выбросы исключили в актуальном периоде
2014-2016');
plt.xlim(0, 3)
plt.show()
```



Остальные платформы показали, что нижний квартиль у всех платформ кроме визуально находится на одном уровне, более того медиана также приблизительно на одном уровне, у XOne она слегка больше остальных. У PS4 больше всех верхний квартиль и максимум.

Диаграмма размаха подтвердила ранее полученные выводы.

- PS4 -самая продаваемая платформа - распределение смещено в сторону максимальных значений.
- XOne - вторая по популярности платформа.
- WiiU - занимает третье место.

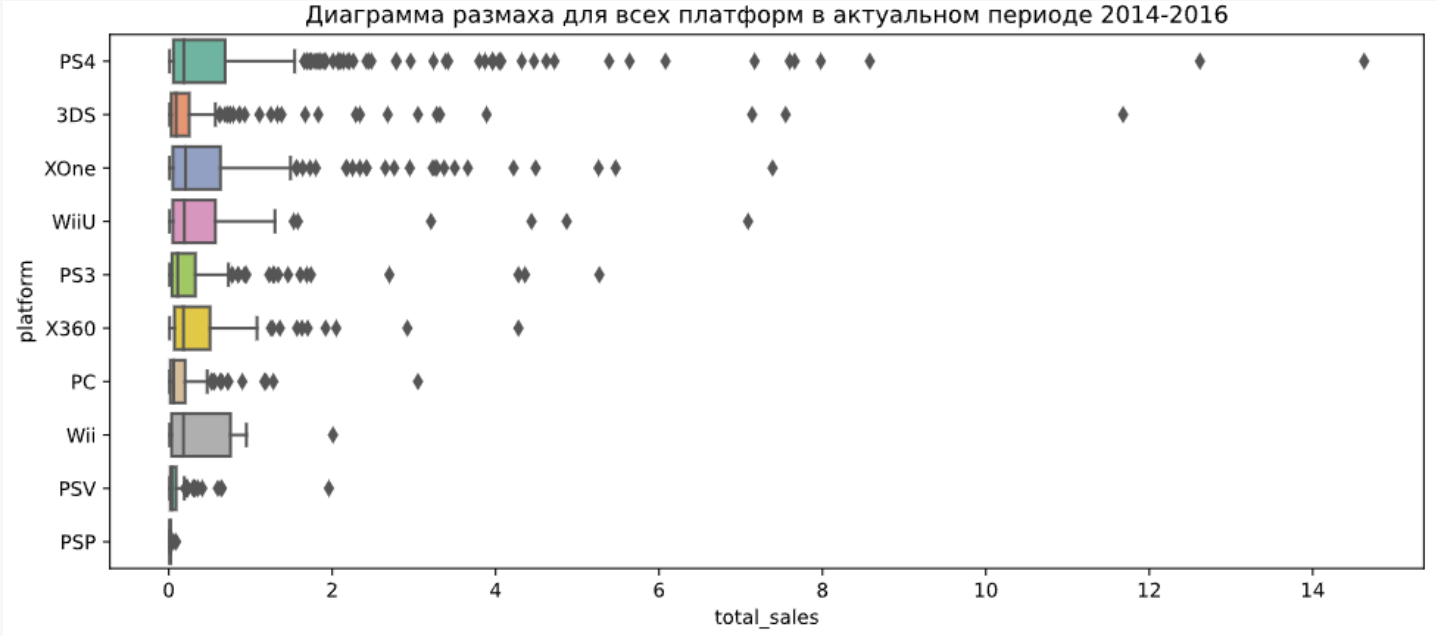
Платформа PC показала что уровень ее медианы лежит по границе нижнего квартиля всех трех остальных платформ, а верхний квартиль платформы PC лежит в границах медиан остальных трех платформ. Максимальное значение не дотягивает до границы верхнего квартиля платформы которая на третьем месте WiiU.

Никаких чудес по PC- это было видно и по предыдущим графикам.

Построим диаграмму размаха для всех платформ за актуальный период 2014-2016 по глобальным продажам.

In [65]:

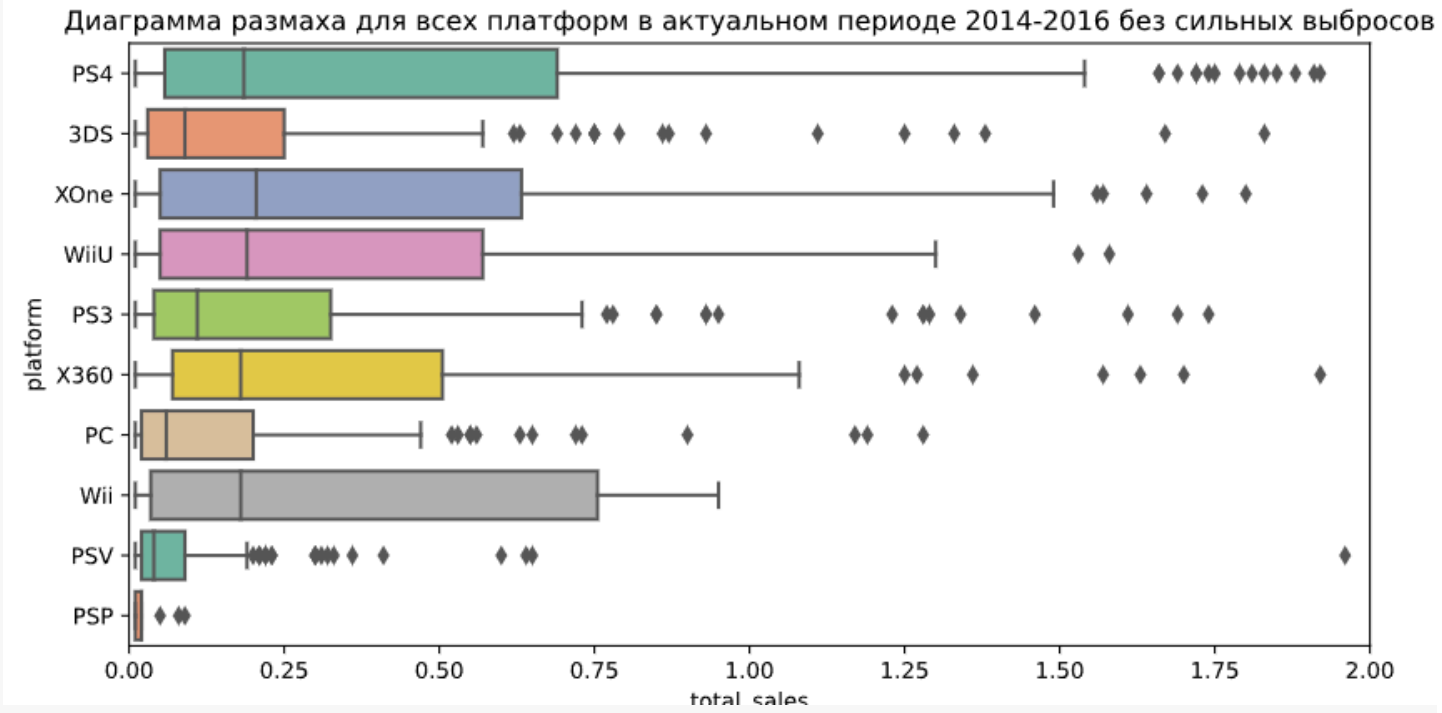
```
plt.figure(figsize=(12,5))
sns.boxplot(x='total_sales', y='platform', data=actual_period, palette='Set2')
plt.title('Диаграмма размаха для всех платформ в актуальном периоде 2014-2016');
plt.show()
```



Присутствует очень много выбросов и они характерны для всех платформ. Уберем эти выбросы и еще раз посмотрим на диаграмму размаха.

In [66]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x='total_sales', y='platform', data=actual_period, palette='Set2')
plt.title('Диаграмма размаха для всех платформ в актуальном периоде 2014-2016 без
сильных выбросов');
plt.xlim(0, 2)
plt.show()
```



Границы этих платформ высокие, но при всем при этом медиана у всех платформ, кроме PS и PS3, не выходит за рамки 0.25 млн продаж копий.

3.5 Исследуем платформу PS4

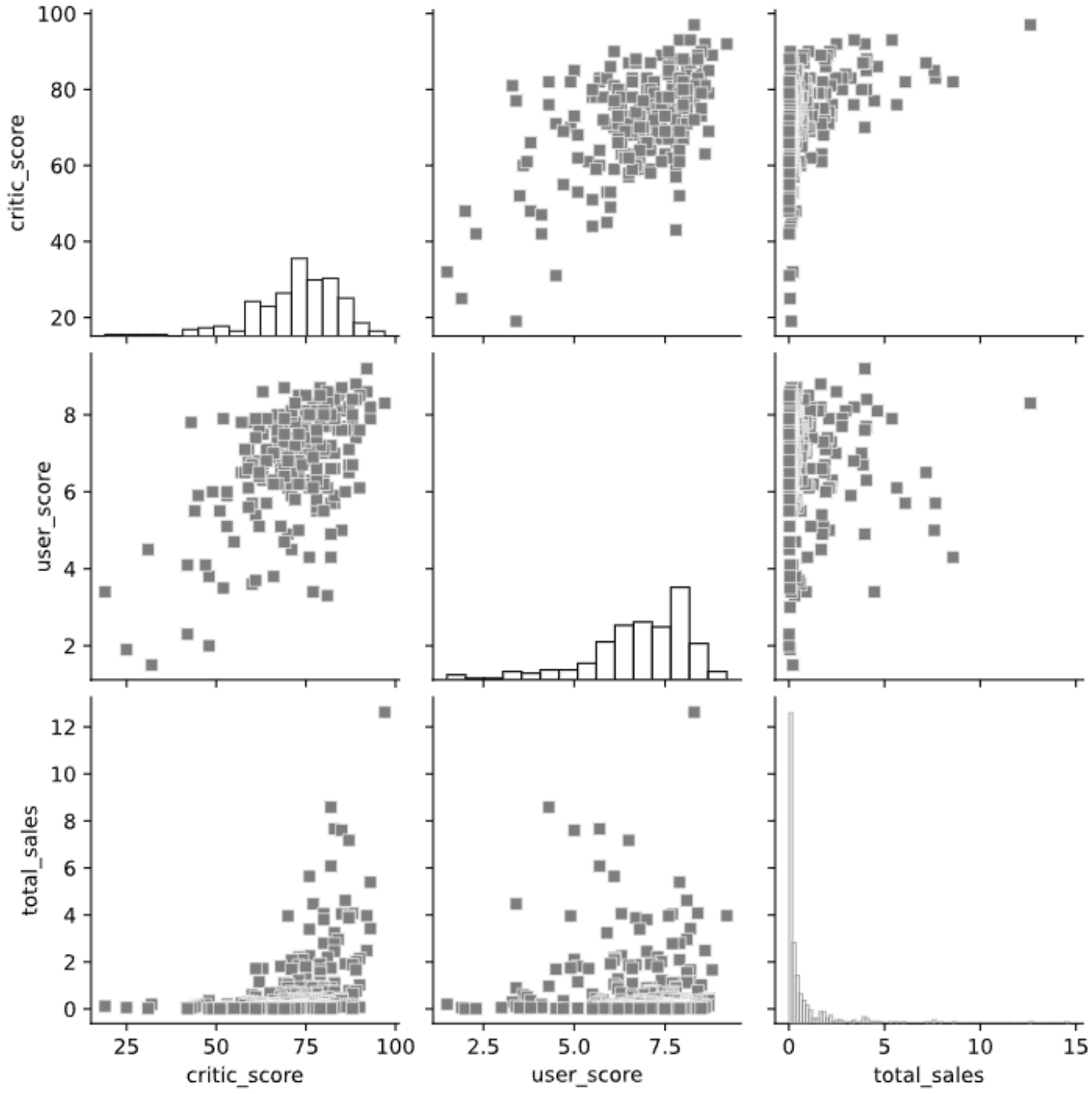
Посмотрим как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Платформу выбрали самую популярную. Построим диаграмму рассеяния и посчитать корреляцию между отзывами и продажами. Сформулировать выводы. Используем ранее созданный датфрейм actual_period с данными о продажах за период 2014-2016 год.

In [67]:

```
print('Влияние отзывов пользователей и критиков на продажи игр на платформе
PS4.\nГрафик корреляции и Диаграмма рассеяния')
sns.pairplot(actual_period[actual_period.platform == "PS4"]\
             [['critic_score', 'user_score', 'total_sales']], \
             plot_kws={'color': 'grey', 'marker': 's'}, \
             diag_kws = {'color': 'white'});
plt.show()
actual_period[actual_period['platform'] == 'PS4'][['total_sales', 'critic_score',
'user_score']].corr(method='spearman')
```

Влияние отзывов пользователей и критиков на продажи игр на платформе PS4.

```
График корреляции и Диаграмма рассеяния
```



Out[67]:

	total_sales	critic_score	user_score
total_sales	1.000000	0.503512	-0.005280
critic_score	0.503512	1.000000	0.442835
user_score	-0.005280	0.442835	1.000000

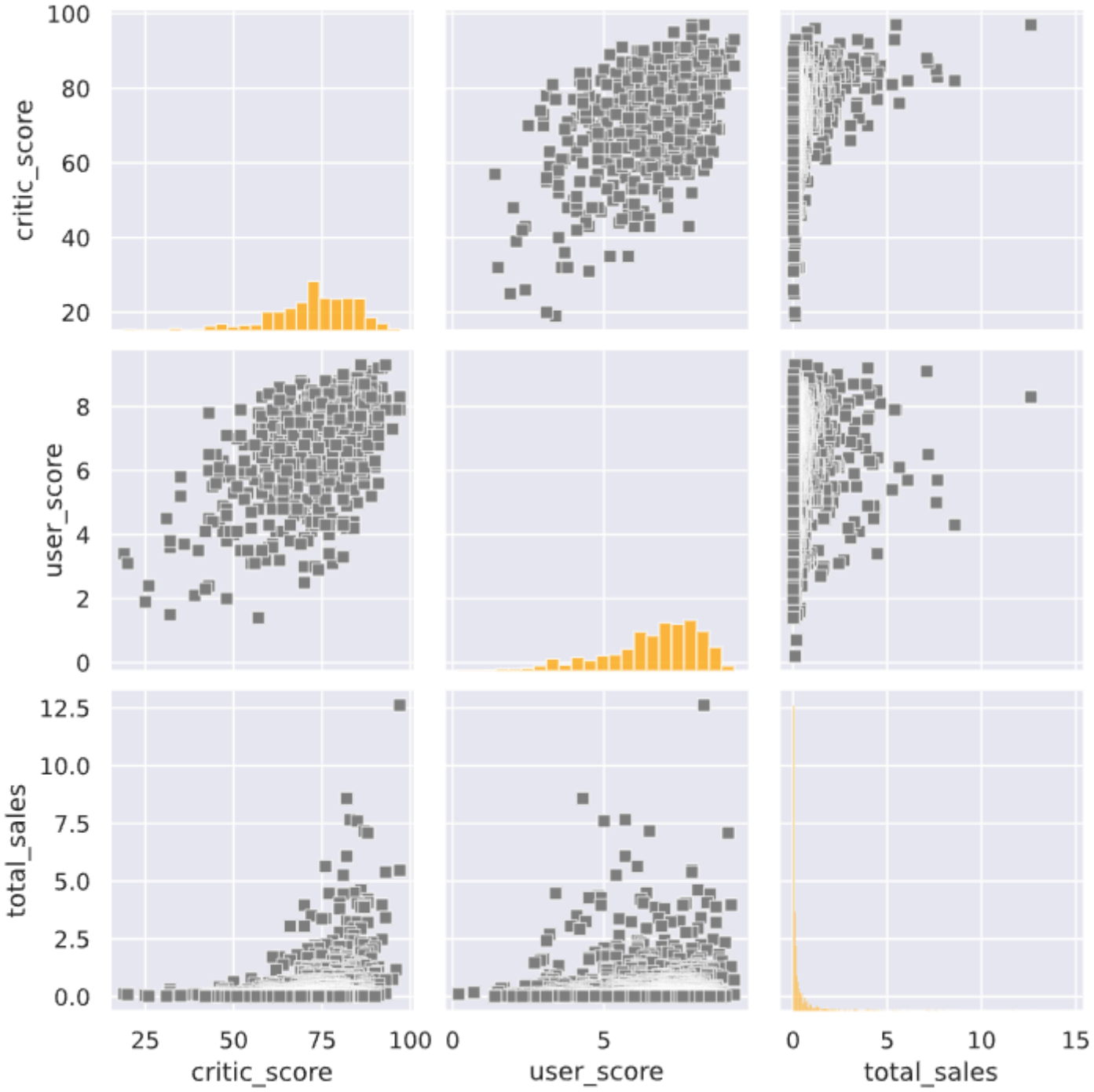
Диаграмма рассеивания и график корреляции показывает слабую корреляцию оценки критиков и оценки пользователей - в чем-то мнения совпадают. Общие продажи и есть слабая корреляция с оценкой критиков, а оценка пользователей совсем слабо коррелирует с общими продажами. Как итог, можем заключить, что особого прямого влияния на продажи ни оценка критиков, ни пользователей не оказывает, хотя к критикам прислушиваются видимо больше.

Соотнесем выводы с продажами игр на других платформах.

In [68]:

```
print('Влияние отзывов пользователей и критиков на продажи игр на всех платформах \
за актуальный период.\nГрафик корреляции и Диаграмма рассеяния')
sns.pairplot(actual_period[['platform', 'critic_score', 'user_score',
'total_sales']],
             plot_kws = {'color': 'grey', 'marker': 's'},
             diag_kws = {'color': 'orange'});
plt.show()
actual_period[['platform', 'total_sales', 'critic_score',
'user_score']].corr(method='spearman')
```

Влияние отзывов пользователей и критиков на продажи игр на всех платформах за актуальный период.
График корреляции и Диаграмма рассеяния



Out[68]:

	total_sales	critic_score	user_score
total_sales	1.00000	0.408280	-0.028250
critic_score	0.40828	1.000000	0.431267
user_score	-0.02825	0.431267	1.000000

3.6 Исследуем влияния отзывов пользователей и критиков на других платформах.

Построим отдельные матрицы корреляции для популярных платформ. Используем датафрейм top_four в котором толкьо 4 платформы 'PS4','XOne','PC','WiiU', предварительно удалим пропуски в столбцах с оценкой пользователей и критиков. Удобнее будет работать.

In [69]:

```
top_four = top_four.dropna()
```

In [70]:

```
platforms = ['PS4','XOne','PC','WiiU']

rows = 2
```

```
cols = 2

fig, axes = plt.subplots(rows, cols, figsize=(8,5))

count = 0
for row in range(rows):
   for col in range(cols):
       # перебираем по индексу
       index = platforms[count]
       df = top_four[(top_four['platform'] == index)]

       df_f = df[['total_sales' ,'critic_score', 'user_score']]

       sns.set(font_scale=1.0)
       ax = sns.heatmap(df_f.corr()[['total_sales']].sort_values(by='total_sales',
ascending=False),
                        cmap="crest", annot=True, annot_kws={'size':15},
ax=axes[row,col])
       ax.set_title(index, fontsize=20)
       ax.set_yticklabels(ax.get_yticklabels(), rotation=0)
       plt.tight_layout(pad=5)
       count +=1
```

Резюмируем: Есть слабая корреляция глобальных продаж с оценкой критиков по всем платформам, а также такая же слабая корреляция между оценкой пользователей и оценкой критиков.

Выводы, сделанные по продажам по платформе PS4 соотносятся и на другие платформы.

Влияние оценок критиков и пользователей на продажи нет. Тут большее влияние скорее всего оказывают третьи факторы: финансовый кризис, мода на тот или очередной гаджет в моменте, технический прогресс, появление новинок, предпочтения пользователей, иные факторы, влияющие системно.

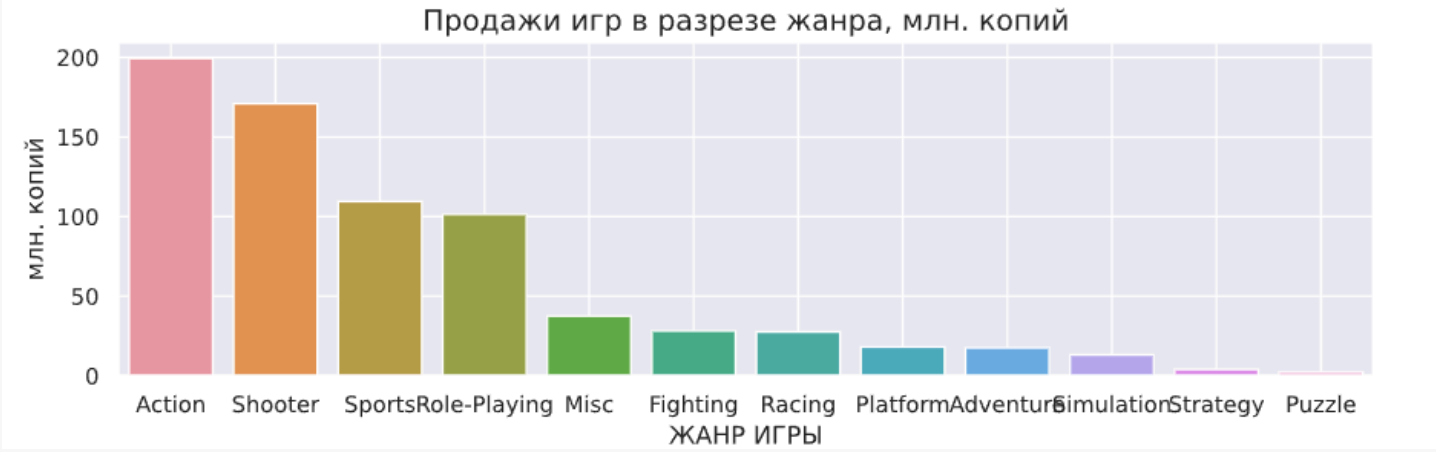
3.7 Посмотрим как жанр игры влияет на продажи.

Посмотрим на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах. Какой жанр игр самый предпочтительный для пользователей всех стран. Есть ли жанры, которые особо выделяются как в самые популярные так и самые непопулярные по количеству проданных копий.

Используем ранее созданный датафрейм actual_period с данными о продажах за период 2014-2016 год.

In [71]:

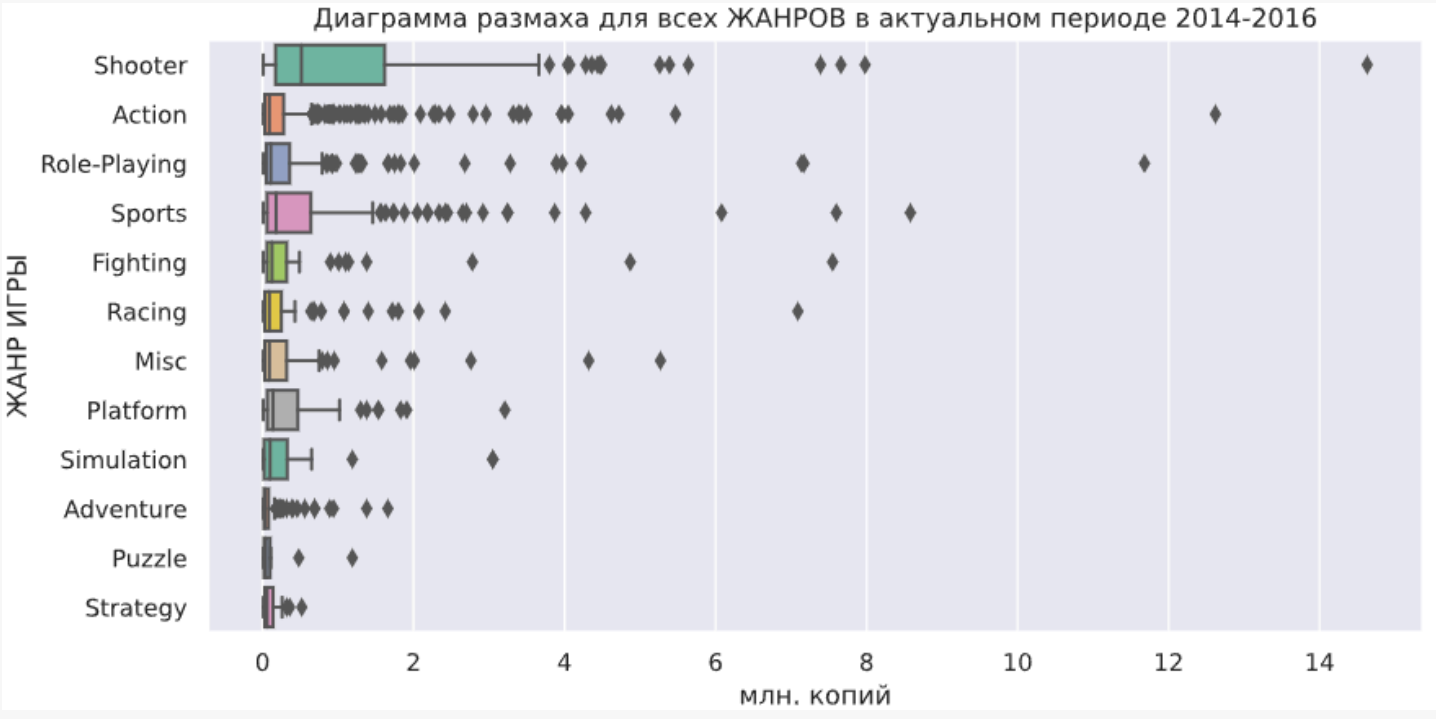
```
genre_sales = actual_period.pivot_table(
   index='genre',
   values='total_sales',
   aggfunc='sum').sort_values(by='total_sales', ascending=False).reset_index()
plt.figure(figsize=(11, 3))
plt.title('Продажи игр в разрезе жанра, млн. копий', fontsize=14)
sns.barplot(x='genre', y='total_sales', data=genre_sales)
plt.ylabel('млн. копий')
plt.xlabel('ЖАНР ИГРЫ')
plt.grid(True);
```



Так как за высокими показателями общих продаж может скрываться множество мелких игр с низкими продажами или какие-то единичные успешные игры с множеством мелких неудач. Рассмотрим и сравним медианные продажи по всем жанрам. Так как их много, то быстро удобно будет сранить данные на диаграмме размаха. Построим ее для всего датафрейма и попробуем найти жанр, где игры стабильно приносят высокий доход.

In [72]:

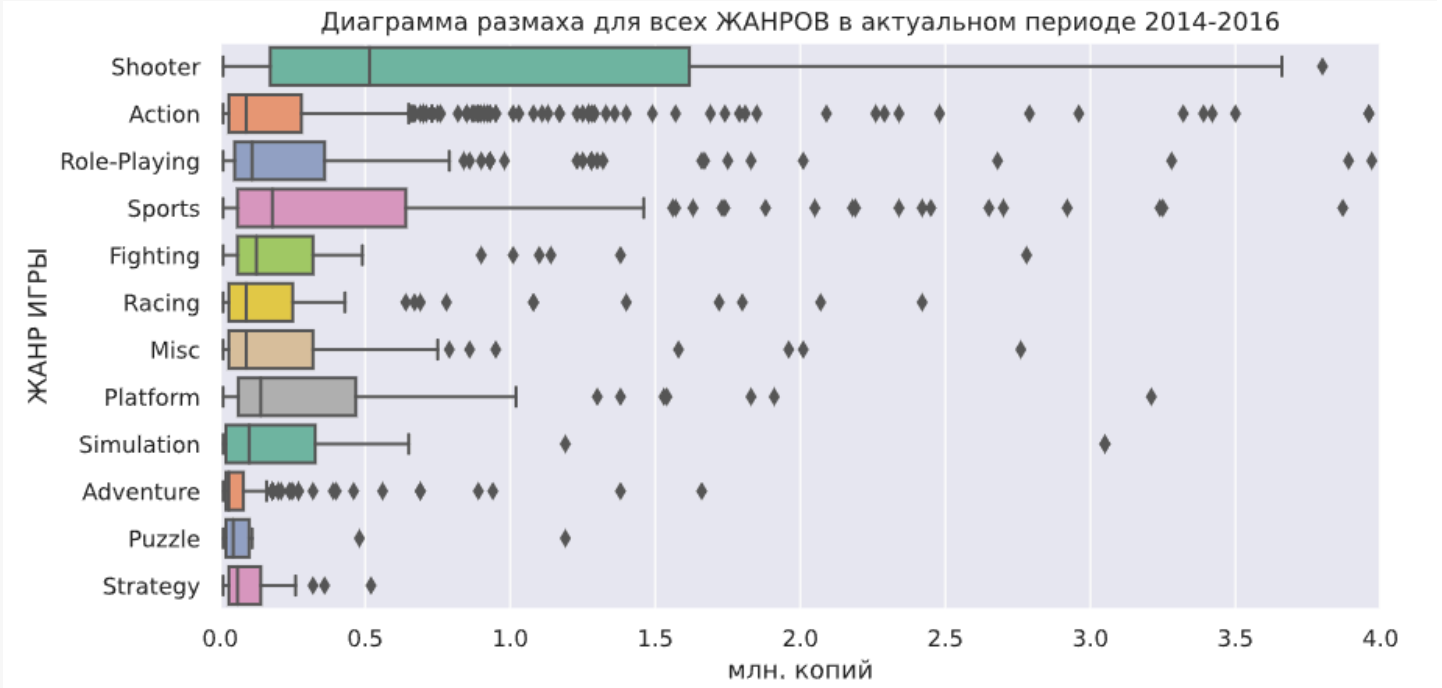
```
plt.figure(figsize=(10,5))
sns.boxplot(x='total_sales', y='genre', data=actual_period, palette='Set2')
plt.xlabel('млн. копий')
plt.ylabel('ЖАНР ИГРЫ')
plt.title('Диаграмма размаха для всех ЖАНРОВ в актуальном периоде 2014-2016');
plt.show()
```



И опять выбросы, но даже сейчас видно, что рейтинг по медиане отличается от построенного графика основанного на общем числе продаж. Уберем выбросы. По жанру Shooter верхний квартиль приближен к 4 млн копий.

In [73]:

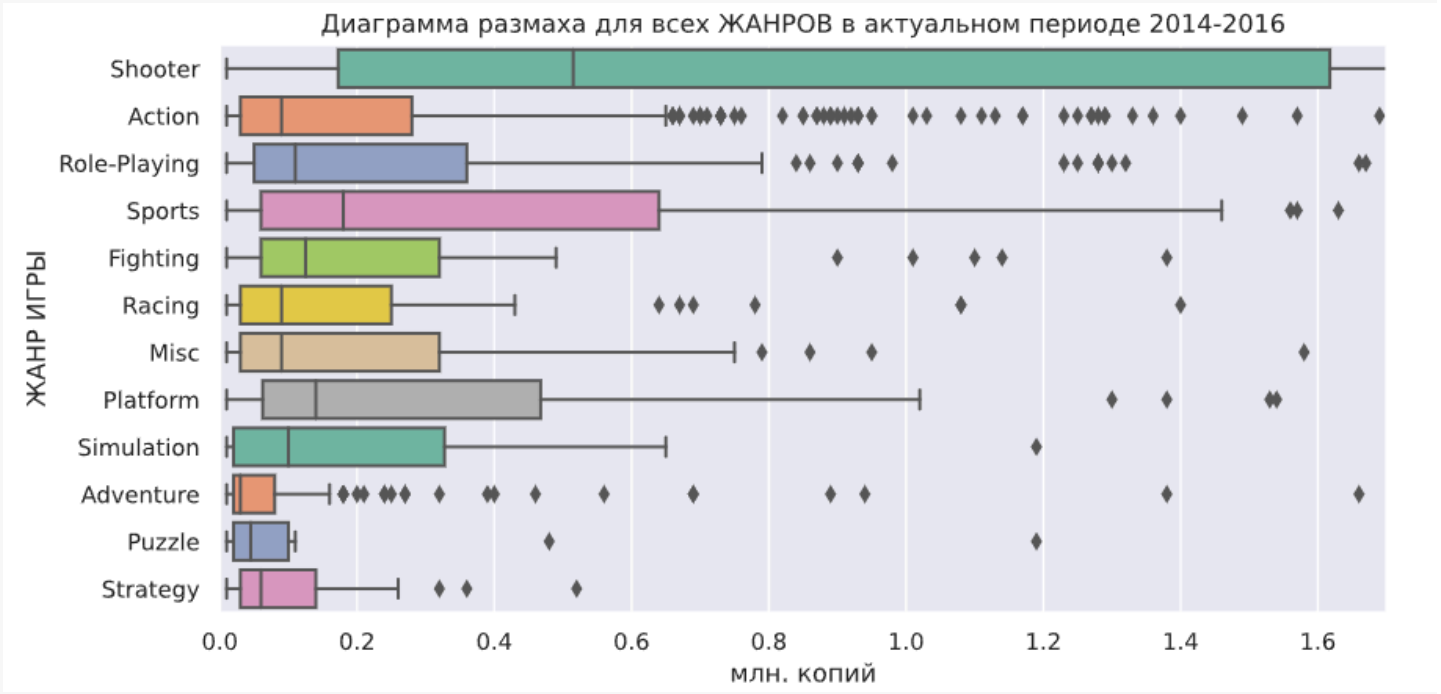
```
plt.figure(figsize=(10,5))
sns.boxplot(x='total_sales', y='genre', data=actual_period, palette='Set2')
plt.title('Диаграмма размаха для всех ЖАНРОВ в актуальном периоде 2014-2016')
plt.xlabel('млн. копий')
plt.ylabel('ЖАНР ИГРЫ')
plt.xlim(0, 4)
plt.show()
```



Shooter верхний квартиль в районе 3,6-3,7 млн копий. Еще больше отсечем выбросов, чтобы лучше разглядеть и сравнить медианы по жанрам.

In [74]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x='total_sales', y='genre', data=actual_period, palette='Set2')
plt.title('Диаграмма размаха для всех ЖАНРОВ в актуальном периоде 2014-2016')
plt.xlabel('млн. копий')
plt.ylabel('ЖАНР ИГРЫ')
plt.xlim(0, 1.7)
plt.show()
```



Как видим, видимо много выбросов характерно для компьютерных игр. Будем считать это некой особенностью индустрии. Что замечательно, у самых непопулярных жанров таких выбросов очень мало.
Adventure показал самые скромные результаты по медиане.
Явно выделяется и по ширине размаха и по медиане жанр Shooter.
Второе место занял жанр Sport.
Посмотрим сводную таблицу с медианными значениями продаж.

In [75]:

```
print('Медиана по продажам игр по жанрам, млн. копий')
actual_period_pivot = actual_period.pivot_table(index='genre', values='total_sales',
aggfunc='median')\
   .sort_values('total_sales', ascending=False)
actual_period_pivot.rename(columns = {'total_sales':'median_sales'})
```

Медиана по продажам игр по жанрам, млн. копий

Out[75]:

	median_sales
genre	
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

Безусловные лидер продаж - это игры в жанре **Shooter**. На момент зарождения жанра за рубежом укрепилось слово «шутер», как вариант описания игрового процесса. Если брать общие продажи, то их меньше, чем у жанра Action.
Второе место с большим отрывом от стрелялок у жанра **Sport**. Это видео игра, имитирующая занятия спортом. Большинство видов спорта были воссозданы с помощью видеоигр, включая командные виды спорта, легкую атлетику, экстремальные виды спорта и спортивные единоборства. **Третье место** занимает

жанр **Platform**. Жанр компьютерных игр, в которых основу игрового процесса составляют прыжки по платформам, лазанье по лестницам, сбор предметов, необходимых для победы над врагами или завершения уровня. Многие игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Персонажами таких игр часто бывают вымышленные существа (к примеру, драконы, гоблины) или антропоморфные животные.

Что ж, на самых топовых по популярности местах игры в которых делается упор на эксплуатацию физических возможностей игрока, таких как зрительно-моторная координация и скорость реакции.

Самый прибыльный по общим продажам жанр игры Action по медиане сравнялся с Misc и Racing и разделили 7 место по популярности на троих. 8,9 и 10 места заняли жанры Strategy, Puzzle, Adventure.

Последнее место по количеству проданных миллионов копий занимает жанр **Adventure**. Это один из основных жанров компьютерных игр, представляющий собой интерактивную историю с главным героем, управляемым игроком. Важнейшими элементами игры в жанре квеста являются собственно повествование и исследование мира, а ключевую роль в игровом процессе играет решение головоломок и задач, требующих от игрока умственных усилий.

Puzzle - на девятом месте. Этот жанр компьютерных игр, главной целью в которых является решение различных логических задач, которые требуют от игрока наличия интуиции, стратегии и в некоторых случаях удачи. ... Прародителями данного жанра можно считать механические и настольные логические игры: кроссворды, кубик Рубика и так далее. А вот компьютерным прототипом данного жанра можно смело считать известную игру Тетрис.

Strategy на восьмом месте.Strategy - один из основных жанров компьютерных игр, в котором игроку для победы необходимо применять стратегическое мышление. В популярных играх такого жанра игроку часто предлагается играть не за конкретного персонажа, а за их условные массы, к примеру, руководить строительством города или командовать целыми армиями в военных кампаниях.

Как видим, наименьшая популярность у игр, в которых необходимо размышлять, решать логические и стратегические головоломки и задачки.

3.8 Итоги раздела.

Провели исследовательский анализ данных и посмотрели количество игр, выпускаемых за разные годы. Пришли к выводу, что рассматривать данные за все года нецелесообразно из-за того, что за весь период обстоятельства, влияющие как на само количество игр, так и на их продажи не сопоставимыми. Также вторым фактором, сужающим актуальный срок анализа - это жизненный цикл платформы игры. Характерный срок появления платформ это 9-11 лет. Появляется платформа, идет рост продаж. затем плавный или резкий спад и платформа исчезает.

Поэтому рассматривать данные дольше трех лет нет смысла для построения прогнозов на 2017 год. Принято решение, что актуальный период для анализа это 2014,2015 и 2016 год. учтено. что данные за 2016 год могут быть неполными.

За актуальный период 2014-2016(3 года) игры продавались на десяти платформах.

Оценили по графику какие платформы лидируют по продажам, растут или падают. Выделили 4 платформы претендента на лидеров продаж и с подходящим жизненным циклом, а именно - не угасающим, растущим. Это:

- **PS4**. Выбор обусловлен тем,что платформа лидер у нее самый высокий результат по продажам в 2015 году и подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. падение в 2016 году не показательно. помним, что данные могут быть неполными. Однако и при таких данных в 2016 году это самые высокие показатели.
- **XOne**. Выбор обусловлен тем, что у платформы второй результат по количеству продаж как в 2015 году, так и в 2016 году. А также подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. Опять же падение продаж в 2016 году и второе место - не показательно. помним, что данные могут быть неполными и место по продажам может быть выше.
- **WiiU**. На графике эта платформа не показывает каких-то грандиозных выдающихся продаж, в 2015 году есть небольшой спад. Но все же она по сроку жизненного цикла и уровню продаж в 2014 году может занимать третье место.
- **PC**. Выбор обусловлен уникальным жизненным циклом. Платформа жизнеспособна на протяжении с всего изучаемого периода с 1985. Пик был в 2011 году. Она то становится более популярное, то менее. Однако до сих пор актуальна. Хотя конечно показатели по продажам за все предыдущие года не такой высокий как у других платформ. Зато стабильный.

Построили график «ящик с усами» по глобальным продажам игр в разбивке по платформам. Границы этих платформ высокие, но при всем при этом медиана у всех платформ, кроме PS и PS3, не выходит за рамки 0.25 млн продаж копий.
Ящик с усами для потенциально прибыльных платформ PS4, XOne, WiiU подтвердил данные с графика. нижний квартиль у всех платформ кроме визуально находится на одном уровне, более того медиана также приблизительно на одном уровне, у XOne она слегка больше остальных. У PS4 больше всех верхний квартиль и максимум.
Диаграмма размаха подтвердила ранее полученные выводы.

- PS4 -самая продаваемая платформа - распределение смещено в сторону максимальных значений.
- XOne - вторая по популярности платформа.
- WiiU - занимает третье место.

Платформа PC показала что уровень ее медианы лежит по границе нижнего квартиля всех трех остальных платформ, а верхний квартиль платформы PC лежит в границах медиан остальных трех платформ. Максимальное значение не дотягивает до границы верхнего квартиля платформы которая на третьем месте WiiU. Это было видно и по предыдущим графикам. Зато это платформа стабильная. Стоит это иметь ввиду.
Диаграмма рассеяния и расчет корреляции между отзывами и продажами не показал какой-то явной взаимосвязи. Да, рейтинг критиков и пользователей показывает не сильную корреляцию на глобальные продажи. Это справедливо как для лидера продаж по платформе, так и по всем платформам.
Общее распределение игр по игр по жанрам показало, что люди сильно предпочитают активные адреналиновые игры, чем стратегические, логические, также никаких удивлений это не вызывает.
Самыми популярными жанрами являются Shooter и Sport, Platform. Action лидирует в числе общих продаж видимо за счет каких-то звездных продуктов. Медианные значения из лидера привели только на 7-е место.
Не перспективными в плане продаж являются игры в жанре Adventure, Puzzle и Strategy.
Как итог: портрет потенциально прибыльной игры:

- это игра в жанрах Shooter и Sport и Platform
- на платформах - XOne, PS4, WiiU,
- при этом рейтинг пользователя или критика не важны, так как они не имеют сильного влияния на продажи.

Отдельно стоит упомянуть платформу PC - это неумираемый пока вид платформы. Если XOne, PS4, WiiU свой жизненный цикл отживут, после появятся новые платформы, то высока доля вероятности, что эти новые платформы встретит бессмертный PC. считаю, что не стоит сбрасывать со счетов этуу платформу. Она показывает устойчивость, за счет которой можно получать прибыль без необходимости разрабатывать новую платформу
Теперь определим портрет пользователя для каждого региона.

4 Портрет игрока каждого региона

Составим портрета пользователя каждого региона NA, EU, JP в котором для пользователя региона определим:

- Самые популярные платформы (топ-5). Описать различия в долях продаж.
- Самые популярные жанры (топ-5). Пояснить разницу.
- Влияет ли рейтинг ESRB на продажи в отдельном регионе?

Используем датафрейм actual_period с данными за 2014-2016 год.

4.1 Самые популярные платформы (топ-5) игроков по регионам

Самые популярные платформы (топ-5) игрока региона NA - Северная Америка

In [76]:

```
na_sales_top = actual_period.pivot_table(
    index='platform',
    values='na_sales',
    aggfunc='sum').sort_values(by='na_sales', ascending = False).head(5)
print('Продажи по самым популярным платформам (топ-5) игрока региона NA:\n ',
na_sales_top)
```

Продажи по самым популярным платформам (топ-5) игрока региона NA:

platform	na_sales

PS4 98.61
XOne 81.27
X360 28.30
3DS 22.64
PS3 22.05

In [77]:

```
na_sales_top.plot.barh(
    figsize=(7,2), alpha=0.4, color='green', ec='black', linewidth=1, grid=False);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Рейтинг продаж млн. копий игр по платформам в Северной Америке')
plt.xlabel('млн.копий')
plt.ylabel('Платформа')
plt.show()
```

Самые популярные платформы (топ-5) игрока региона EU - Европа

In [78]:

```
eu_sales_top = actual_period.pivot_table(
    index='platform',
    values='eu_sales',
    aggfunc='sum').sort_values(by='eu_sales', ascending = False).head(5)
print('Продажи по самым популярным платформам (топ-5) игрока региона eu:\n ',
eu_sales_top)
```

Продажи по самым популярным платформам (топ-5) игрока региона eu:

platform	eu_sales
PS4	130.04
XOne	46.25
PS3	25.54
PC	17.97
3DS	16.12

In [79]:

```
eu_sales_top.plot.barh(
    figsize=(7,2), alpha=0.4, color='yellow', ec='black', linewidth=1, grid=False);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Рейтинг продаж млн. копий игр по платформам в Европе')
plt.xlabel('млн.копий')
plt.ylabel('Платформа')
plt.show()
```

Самые популярные платформы (топ-5) игрока региона JP - Япония

In [80]:

```
jp_sales_top = actual_period.pivot_table(
   index='platform',
   values='jp_sales',
   aggfunc='sum').sort_values(by='jp_sales', ascending = False).head(5)
print('Продажи по самым популярным платформам (топ-5) игрока региона jp:\n ',
jp_sales_top)
```

Продажи по самым популярным платформам (топ-5) игрока региона jp:

platform	jp_sales
3DS	44.24
PS4	15.02
PSV	14.54
PS3	11.22
WiiU	7.31

In [81]:

```
jp_sales_top.plot.barh(
   figsize=(7,2), alpha=0.3, color='red', ec='black', linewidth=1, grid=False);
plt.legend(bbox_to_anchor=(1, 0.9))
plt.title('Рейтинг продаж млн. копий игр по платформам в Японии')
plt.xlabel('млн.копий')
plt.ylabel('Платформа')
plt.show()
```

4.2 Объединим данные о популярных платформах.

Объединим данные по самым популярным платформам (топ-5) среди игроков разных регионов. Сравним все платформы которые есть в продажах в разных регионах. Добавим колонку total_sales_top

In [82]:

```
region_top = pd.merge(na_sales_top, eu_sales_top, left_index=True, right_index=True,
how='outer')
region_top = pd.merge(region_top, jp_sales_top, left_index=True, right_index=True,
how='outer')
region_top['total_sales_top'] = region_top[['na_sales', 'eu_sales',
'jp_sales']].sum(axis = 1)
region_top['na_part'] = round((region_top['na_sales'] /
region_top['total_sales_top']), 2)
region_top['eu_part'] = round((region_top['eu_sales'] /
region_top['total_sales_top']), 2)
region_top['jp_part'] = round((region_top['jp_sales'] /
region_top['total_sales_top']), 2)
region_top
```

Out[82]:

platform	na_sales	eu_sales	jp_sales	total_sales_top	na_part	eu_part	jp_part
3DS	22.64	16.12	44.24	83.00	0.27	0.19	0.53
PC	NaN	17.97	NaN	17.97	NaN	1.00	NaN
PS3	22.05	25.54	11.22	58.81	0.37	0.43	0.19
PS4	98.61	130.04	15.02	243.67	0.40	0.53	0.06
PSV	NaN	NaN	14.54	14.54	NaN	NaN	1.00
WiiU	NaN	NaN	7.31	7.31	NaN	NaN	1.00
X360	28.30	NaN	NaN	28.30	1.00	NaN	NaN
XOne	81.27	46.25	NaN	127.52	0.64	0.36	NaN

Япония сильнее всего отличается от остальных стран по платформам. Две платформы находятся только в топ-5 в этой стране, а именно PSV и WiiU, половина всех продаж на платформе 3DS также принадлежит этой стране, а доля платформ PS3 и PS4 среди глобальных продаж в японии 0,19 и 0,06 соответственно.

Рынки стран Северной Америки и Европы больше похожи. Но и там есть различия. Для европейского рынка больше характерны продажи на PS4 - это 0,53 часть всех продаж и PS3 - это 0,43 доля всех продаж этой платформы, на третьем месте XOne - с долей 0,36, самая малая доля выпала на платформу 3DS - 0,19. PC - 100% находится в топ-5 среди глобальных продаж только в европе.

Рынок Северной Америки похож на европейский прежде всего тем, что там также популярны платформы PS4 - это 0,40 часть всех продаж и PS3 - это 0,37. Это немного меньше европейского рынка, но все равно ощутимо. Зато платформа XOne почти в два раза популярнее, чем в Европе, доля продаж игр на этой платформе составила аж 0,64. Отличает Северную Америку от Европы и Японии - платформа X360 среди топ-5 самых популярных по продаваемости в стране. Ни в Японии, ни в Европе эти платформы не находятся в топ-5.

Наибольшие точки пересечений среди всех трех регионов среди платформ PS3, PS4, 3DS.

По глобальным продажам только PS4 является фаворитом - 243,67 млн проданных копий и охватывает все три региона. XOne тоже занимает лидирующее место по продажам 127.52 млн копий, но в топ-5 в Японии его нет. 3DS занимает третье место среди глобальных продаж, при этом большая часть продаж ориентирована на Японию, а остальные продажи делят между собой европа и Северная Америка.

4.3 Самые популярные жанры (топ-5) игроков по регионам.

Самые популярные жанры (топ-5) игрока региона NA - Северная Америка

In [83]:

```
na_genre_top = actual_period.pivot_table(
   index='genre',
   values='na_sales',
   aggfunc='sum').sort_values(by='na_sales', ascending = False).head(5)
print('Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона NA:\n ',
na_genre_top)
```

Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона NA:

	na_sales
genre	
Shooter	79.02
Action	72.53
Sports	46.13
Role-Playing	33.47
Misc	15.05

Самые популярные жанры (топ-5) игрока региона EU - Европа

In [84]:

```
eu_genre_top = actual_period.pivot_table(
   index='genre',
   values='eu_sales',
   aggfunc='sum').sort_values(by='eu_sales', ascending = False).head(5)
print('Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона EU:\n ',
eu_genre_top)
```

Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона EU:

	eu_sales
genre	
Action	74.68
Shooter	65.52
Sports	45.73
Role-Playing	28.17
Racing	14.13

Самые популярные жанры (топ-5) игрока региона JP - Япония

In [85]:

```
jp_genre_top = actual_period.pivot_table(
   index='genre',
   values='jp_sales',
   aggfunc='sum').sort_values(by='jp_sales', ascending = False).head(5)
print('Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона JP:\n ',
jp_genre_top)
```

Продажи по самым популярным ЖАНРАМ (топ-5) игрока региона JP:

	jp_sales
genre	
Role-Playing	31.16
Action	29.58
Fighting	6.37
Misc	5.61
Shooter	4.87

4.4 Объединим данные о популярных жанрах среди регионов.

Объединим данные по самым популярным платформам (топ-5) среди игроков разных регионов. Сравним все платформы которые есть в продажах в разных регионах. Добавим колонку total_sales_top и колонки с долями продаж регионов по жанрам.

In [86]:

```
region_top_genre= pd.merge(na_genre_top, eu_genre_top, left_index=True,
right_index=True, how='outer')
region_top_genre = pd.merge(region_top_genre, jp_genre_top, left_index=True,
right_index=True, how='outer')
region_top_genre['total_sales_top'] = region_top_genre[['na_sales', 'eu_sales',
'jp_sales']].sum(axis = 1)
```

```
region_top_genre['na_part'] = round((region_top_genre['na_sales'] /
region_top_genre['total_sales_top']), 2)
region_top_genre['eu_part'] = round((region_top_genre['eu_sales'] /
region_top_genre['total_sales_top']), 2)
region_top_genre['jp_part'] = round((region_top_genre['jp_sales'] /
region_top_genre['total_sales_top']), 2)
region_top_genre
```

Out[86]:

genre	na_sales	eu_sales	jp_sales	total_sales_top	na_part	eu_part	jp_part
Action	72.53	74.68	29.58	176.79	0.41	0.42	0.17
Fighting	NaN	NaN	6.37	6.37	NaN	NaN	1.00
Misc	15.05	NaN	5.61	20.66	0.73	NaN	0.27
Racing	NaN	14.13	NaN	14.13	NaN	1.00	NaN
Role-Playing	33.47	28.17	31.16	92.80	0.36	0.30	0.34
Shooter	79.02	65.52	4.87	149.41	0.53	0.44	0.03
Sports	46.13	45.73	NaN	91.86	0.50	0.50	NaN

И снова Япония сильнее всего отличается от остальных стран по жанрам. Самые популярные жанры в Европе и Северной америке это Action и Shooter - это самые прибыльные жанры, а в Японии доля Action всего 0,17 от общего числа, тоже самое и по жанру Shooter - доля всего рынка 0,03, да и среди всех ТОП-5 по Японии занимает последнее место. Зато жанр Fighting среди топ-5 есть только в Японии. Жанр Role-Playing - лидер среди топ-5 жанров в Японии и остальные регионы с Японией солидарны. Доля этого жанра почти ровно разделена на все регионы. В топ-5 Японии также вошел жанр Misc не очень популярный среди остальных стран, в Европе его нет среди лидеров, а в Северной Америке это последнее место в топ-5.
Игроки Японии точно отличаются от игроков остальных регионов. Это не вызывает удивления, так как Япония сама по себе сильно отличается по менталитету от других стран. Это сказывается и на предпочтениях.
Что до Северной Америки и Европы, то как было отмечено выше - Action, Shooter и Role-Playing это то, что объединяет эти страны, причем Role-Playing еще и с Японией! Shooter составляет 0,53 доли в Северной Америке и 0,44 в Европе, а Action по 0,41 и 0,42 доли соответственно. Также Sports одинаково любят как в Европе, так и в Северной Америки доли по 0,5.
Европейцы и Североамериканцы предпочитают динамичные игры!
Резюмируем: Action - вот что подходит для всех трех регионов среди самых продаваемых жанров.Shooter второе место по популярности среди глобального рынка, но в Японии его доля среди топ 5 самая минимальная. Среди жанров с одинаковой популярностью среди трех регионов это Role-Playing - его число глобальных продаж на третьем месте.

4.4.1 Влияние рейтинга ESRB на продажи в регионе JP

In [87]:

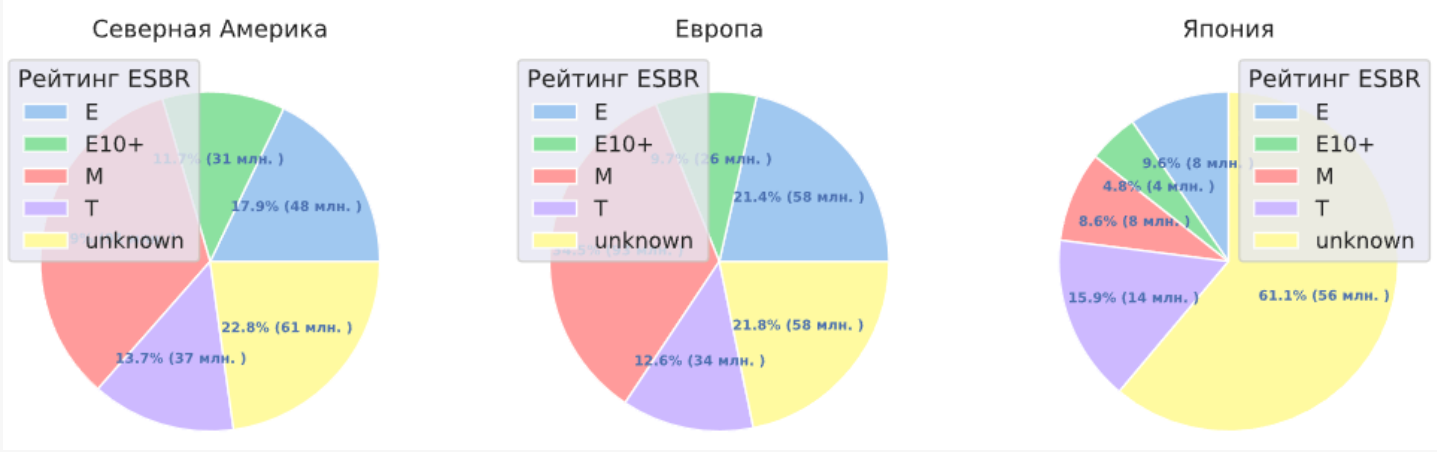
```
na_rating =
actual_period.groupby('rating')['na_sales'].agg(na_sales='sum').reset_index()
eu_rating =
actual_period.groupby('rating')['eu_sales'].agg(eu_sales='sum').reset_index()
jp_rating =
actual_period.groupby('rating')['jp_sales'].agg(jp_sales='sum').reset_index()
fig, ax = plt.subplots(1, 3, figsize=(13, 5))
fig.suptitle('Продажи по рейтингам ESRB в разрезе регионов')
data1, categories1 = na_rating['na_sales'], na_rating['rating']
data2, categories2 = eu_rating['eu_sales'], eu_rating['rating']
data3, categories3 = jp_rating['jp_sales'], jp_rating['rating']
colors = ['#a1c9f4', '#8de5a1', '#ff9f9b', '#d0bbff', '#fffea3']

def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}% ({:d} млн. )".format(pct, absolute)
```

```
wedges, texts, autotexts = ax[0].pie(
   data1, autopct=lambda pct: func(pct, data2), textprops=dict(color ='b'),
colors=colors)
ax[0].set_title('Северная Америка')
ax[0].legend(wedges, categories1, title='Рейтинг ESBR', loc="upper left")
plt.setp(autotexts, size=7, weight ='bold')

wedges, texts, autotexts = ax[1].pie(
   data2, autopct=lambda pct: func(pct, data2), textprops=dict(color ='b'),
colors=colors)
ax[1].set_title('Европа')
ax[1].legend(wedges, categories2, title='Рейтинг ESBR', loc='upper left')
plt.setp(autotexts, size=7, weight ='bold')

wedges, texts, autotexts = ax[2].pie(
   data3, autopct=lambda pct: func(pct, data3), startangle = 90, textprops=dict(color
='b'), colors=colors)
ax[2].set_title('Япония')
ax[2].legend(wedges, categories3, title='Рейтинг ESBR', loc='upper right')
plt.setp(autotexts, size=7, weight ='bold')
plt.show()
```



Рейтинг ESBR в Японии в 61 % случаях не известен. В европе и Северной Америке такой рейтинг установлен почти у 80% игр. а доля неизвестных игр 21-22% почти одинаковый у обеих стран. Их можно сравнить между собой. Японию Можно описать обособленно, неизвестная доля слишком велика, чтобы сравнивать с другими странами - среди известных маркировок 16 % составляет доля с маркировкой T - подросткам, причем как ни странно, в Европе и Северной Америки такие игры тоже приближаются к таким показателям 14 % и 13 % соответственно. В Японии же больше популярны те игры, которые не проходили оценку в ESBR.

В Европе и Северное Америке большая часть продаж приходится на игры с маркировкой M (для взрослых) - 34-35% на втором месте с маркировкой E - для всех 18-21%. Самая малая часть по популярности приходится на игры E10+ - игры для всех старше 10 лет.

Напомним маркировку ESBR:

- EC (Early childhood) — Для детей младшего возраст».
- E (Everyone) — Для всех. Первоначально K-A (Kids to Adults)
- E10+ (Everyone 10 and older) — Для всех от 10 лет и старше.
- T (Teen) — Подросткам.
- M (Matur) — Для взрослых.
- AO (Adults Only 18+) — Только для взрослых.
- RP (Rating Pending) — Рейтинг ожидается.

Как видим, отсутствие рейтинга у большей половины проданных игр в Японии никак не повлияли на продажи. Одна пятая игр, проданных в Северной Америке и Европе также никак не промаркирована. Однако

видно разделение рынков Европы и Северной америки с рынком Японии. если у Европы и Северной Америки маркировка игр. а также наличие такой маркировки очень похожи, то Япония отличается.
Резюмируя: Рейтинг ESRB не влияет на продажи в отдельном регионе.

4.5 Выводы раздела.

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

- Самые популярные платформы (топ-5) и их различия в долях продаж.

Наибольшие точки пересечений среди всех трех регионов среди платформ PS3, PS4, 3DS.
По глобальным продажам только PS4 является фаворитом - 243,67 млн. проданных копий и охватывает все три региона. XOne тоже занимает лидирующее место по продажам 127.52 млн копий, но в топ-5 в Японии его нет. 3DS занимает третье место среди глобальных продаж, при этом большая часть продаж ориентирована на Японию, а остальные продажи делят между собой Европа и Северная Америка.

- Самые популярные жанры (топ-5).

Action - вот что подходит для всех трех регионов среди самых продаваемых жанров. Shooter занимает второе место по популярности среди глобального рынка, но в Японии его доля среди топ 5 самая минимальная. Среди жанров с одинаковой популярностью среди трех регионов это Role-Playing - его число глобальных продаж на третьем месте. Европейцы и Североамериканцы предпочитают динамичные игры!
В Японии есть жанр, которого нет в топ-5других стран и он там очень популярен, это жанр Misc. Это игра, сочетающая геймплей шутера с видом от первого лица с большим количеством игроков, подключенных через Интернет.

- Рейтинг ESRB не влияет на продажи в отдельном регионе.

5 Проверка гипотез

5.1 Гипотеза о рейтинге платформ платформ Xbox One и PC

Проверим гипотезу: Средние пользовательские рейтинги платформ Xbox One и PC равны.
Нулевую гипотезу (H_0) принято формулировать так, чтобы использовать знак равенства и уже исходя из формулировки нулевой гипотезы формулируем альтернативную. В нашем случае требуется на уровне значимости проверить гипотезу о равенстве генеральных средних против одной из конкурирующих гипотез. Так как изначально утверждается, что обе средние генеральных совокупностей равны, то строится двусторонняя критическая область, и альтернативная гипотеза утверждает, что обе средние генеральных совокупностей не равны. Она предполагает возможное отклонение ив большую и в меньшую сторону. Альтернативная гипотеза принимается, когда отбрасывается нулевая гипотеза.
Сформулируем гипотезы.
H_0: Средние пользовательские рейтинги платформы Xbox One равны средним пользовательским рейтингам платформы PC.
H_1: Средние пользовательские рейтинги платформ Xbox One не равны средним пользовательским рейтингам платформы PC.
Уровень статистической значимости регулирует то, насколько далеко должно оказаться наблюдаемое значение от предполагаемого в нулевой гипотезе, чтобы отвергнуть ее. Конвенциональные значения 1% и 5%. Пороговое значение сделаю равное 5%, как общепринятое.
Алгоритм проверки гипотезы: Логика проверки гипотезы основана на сравнении вероятности получения наблюдаемого значения при условии, что нулевая гипотеза верна.

- сначала сравним дисперсии выборок
- вычислим значение (p-value) получить наблюдаемое на выборке значение, при условии, что H_0 верна. Если значение будет большим (сравним с пороговым значением), то нулевую гипотезу не отвергаем.
- выборки разного размера и есть сомнения, что дисперсии у совокупностей одинаковые, поэтому параметр equal_var укажем False/

Используем датафрейм actual_period с данными за 2014-2016 год. Отберем в xone из actual_period все данные по платформе Xbox One, удалим все NaN из получившейся таблицы. Тоже самое сделаем и для платформы PC. отберем в pc все данные по платформе PC и дропнем все наны. Выведем на экран получившиеся значения, чтобы убедиться, что все сработало.

In [88]:

```
xone = actual_period.query('platform == "XOne"')
xone = xone.dropna()
xone.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 151 entries, 165 to 16660
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   name            151 non-null    object 
 1   platform        151 non-null    object 
 2   year_of_release  151 non-null    Int64  
 3   genre           151 non-null    object 
 4   na_sales        151 non-null    float64
 5   eu_sales        151 non-null    float64
 6   jp_sales        151 non-null    float64
 7   other_sales     151 non-null    float64
 8   total_sales     151 non-null    float64
 9   critic_score    151 non-null    Int64  
 10  user_score      151 non-null    float64
 11  rating          151 non-null    object 
dtypes: Int64(2), float64(6), object(4)
memory usage: 15.6+ KB
```

In [89]:

```
pc = actual_period.query('platform == "PC"')
pc = pc.dropna()
pc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115 entries, 458 to 16692
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   name            115 non-null    object 
 1   platform        115 non-null    object 
 2   year_of_release  115 non-null    Int64  
 3   genre           115 non-null    object 
 4   na_sales        115 non-null    float64
 5   eu_sales        115 non-null    float64
 6   jp_sales        115 non-null    float64
 7   other_sales     115 non-null    float64
 8   total_sales     115 non-null    float64
 9   critic_score    115 non-null    Int64  
 10  user_score      115 non-null    float64
 11  rating          115 non-null    object 
dtypes: Int64(2), float64(6), object(4)
memory usage: 11.9+ KB
```

In [90]:

```
alpha = .05

results = st.ttest_ind(xone['user_score'], pc['user_score'], alternative='two-sided', equal_var=False)

print('p-значение:', results.pvalue)

if results.pvalue < alpha:
```

```
    print('Отвергаем нулевую гипотезу.\
    \nПолученное значение p-value меньше заданного уровня значимости.')
else:
    print('Не получилось отвергнуть нулевую гипотезу')
```

p-значение: 0.1403785186826978
Не получилось отвергнуть нулевую гипотезу

Так как не получилось отвергнуть гипотезу Средние пользовательские рейтинги платформ Xbox One и PC равны то оснований утверждать, что средние пользовательские рейтинги платформ Xbox One и PC значимо различаются у нас нет.

5.2 Гипотеза Средние пользовательские рейтинги жанров Action и Sports разные.

Проверим гипотезу: Средние пользовательские рейтинги жанров Action и Sports разные.
Нулевую гипотезу (H_0) принято формулировать так, чтобы использовать знак равенства и уже исходя из формулировки нулевой гипотезы формулируем альтернативную. В нашем случае требуется на уровне значимости проверить гипотезу о неравенстве генеральных средних против одной из конкурирующих гипотез.
Изначально утверждается, что обе средние генеральных совокупностей не равны. При этом не указано в какую сторону они не равны- что больше. а что меньше. Поэтому нулевая гипотеза будет утверждать они равны, а альтернативная, что не равны. Если удастся принять альтернативную гипотезу, то это будет значить, что Средние пользовательские рейтинги жанров Action и Sports разные. Альтернативная гипотеза принимается, когда отбрасывается нулевая гипотеза.
Сформулируем гипотезы.
H_0: Средние пользовательские рейтинги жанра Action равны средним пользовательским рейтингам жанра Sports.
H_1: Средние пользовательские рейтинги жанра Action равны средним пользовательским рейтингам жанра Sports
Уровень статистической значимости регулирует то, насколько далеко должно оказаться наблюдаемое значение от предполагаемого в нулевой гипотезе, чтобы отвергнуть ее. Конвенциональные значения 1% и 5%. Пороговое значение сделаю равное 5%, как общепринятое.
Алгоритм проверки гипотезы: Логика проверки гипотезы основана на сравнении вероятности получения наблюдаемого значения при условии, что нулевая гипотеза верна.

- сначала сравним дисперсии выборок
- вычислим значение (p-value) получить наблюдаемое на выборке значение, при условии, что H_0 верна. Если значение будет большим (сравним с пороговым значением), то нулевую гипотезу не отвергаем.
- выборки разного размера и есть сомнения, что дисперсии у совокупностей одинаковые, поэтому параметр equal_var укажем False

Используем датафрейм actual_period с данными за 2014-2016 год.
Отберем в action из actual_period все данные по жанру Action, удалим все NaN из получившейся таблицы. Тоже самое сделаем и для жанра Sports. Отберем в sports все данные по жанру Sports и удалим все пропуски. Выведем на экран получившиеся значения, чтобы убедиться, что все сработало.

In [91]:

```
action = actual_period.query('genre == "Action"')
action = action.dropna()
action.info()
```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 227 entries, 42 to 16692
Data columns (total 12 columns):

#	Column	Non-Null Count	Dtype
0	name	227 non-null	object
1	platform	227 non-null	object

```
 2   year_of_release  227 non-null    Int64
 3   genre            227 non-null    object
 4   na_sales         227 non-null    float64
 5   eu_sales         227 non-null    float64
 6   jp_sales         227 non-null    float64
 7   other_sales      227 non-null    float64
 8   total_sales      227 non-null    float64
 9   critic_score     227 non-null    Int64
 10  user_score       227 non-null    float64
 11  rating           227 non-null    object
dtypes: Int64(2), float64(6), object(4)
memory usage: 23.5+ KB
```

In [92]:

```
sports = actual_period.query('genre == "Sports"')
sports = sports.dropna()
sports.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83 entries, 77 to 16146
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             83 non-null     object
 1   platform         83 non-null     object
 2   year_of_release  83 non-null     Int64
 3   genre            83 non-null     object
 4   na_sales         83 non-null     float64
 5   eu_sales         83 non-null     float64
 6   jp_sales         83 non-null     float64
 7   other_sales      83 non-null     float64
 8   total_sales      83 non-null     float64
 9   critic_score     83 non-null     Int64
 10  user_score       83 non-null     float64
 11  rating           83 non-null     object
dtypes: Int64(2), float64(6), object(4)
memory usage: 8.6+ KB
```

In [93]:

```
alpha = .05

results_1 = st.ttest_ind(action['user_score'], sports['user_score'],
alternative='two-sided', equal_var=False)

print('p-значение:', results_1.pvalue)

if results_1.pvalue < alpha:
    print('Отвергаем нулевую гипотезу.\
    \nПолученное значение p-value меньше заданного уровня значимости.')
else:
    print('Не получилось отвергнуть нулевую гипотезу')
```

```
p-значение: 8.327612976032047e-09
Отвергаем нулевую гипотезу.
```

Полученное значение p-value меньше заданного уровня значимости.

Так как нулевая гипотеза сформирована на то, что оценки пользователей обоих жанров равны и она была отвергнута, то можно утверждать обратное что Средние пользовательские рейтинги жанров Action и Sports разные.

5.3 Вывод по разделу

Проверка гипотез показала что при пороговом значении в 5%

- можно утверждать, что средние пользовательские рейтинги платформ Xbox One и PC равны;
- можно утверждать, что средние пользовательские рейтинги жанров Action и Sports разные.

6 Общий вывод

Проведено исследование, целью которого было найти закономерности, определяющие успешность игры и выявить потенциально популярный продукт.

Данные были за период до 2016 года включительно с 1985 год. Было учтено, что за 2016 год данные могли быть не полными.

Провели загрузку и подготовили данные к анализу.

Произведена замена названий столбцов — приведение в нижний регистр.
Явных дубликатов не было. Удалено 2 строки неявных дубликатов.
В ходе подготовки данных было удалено некоторое количество данных с критическими пропусками, которые могли помешать анализу, а именно с данными по годам и данными по названию игры.Объем удаленных строк менее 2%, а именно: строки с пропусками в столбцах name (1 строки) и year_of_release (269 строк).
Было выяснено, что значение tbd в столбце user_score можно считать как пропуск, так как это только “обещание” что когда-то будет произведена такая оценка. Произведена замена этого значение на пропуск, а сами пропуски оставили, так как адекватной замены для заполнения пропуска нет.
По колонкам с оценкой критиков и маркировкой игры ESRB было много пропусков. Причинами скорее всего является то, что не для всех игр такие оценки и маркировка проводились. Это не является обязательным для выпуска игры, как например информация о названии игры или платформы под которую ее выпустили. Такие пропуски оставили. Не маркированные ESRB игры пометили как unknown. Устаревшую маркировку рейтинга К-А заменили на современное обозначение Е.

В следующих колонках тип данных заменен на соответствующий значениям, которые в нем содержатся:

- year_of_release - текущий тип данных float64 заменен на целочисленный — Int64.
- user_score - тип данных заменен на float64.
- critic_score - тип данных заменен на Int64.

Добавлен столбец в датафрейм total_sales, в который поместили результаты суммирования глобальных продаж миллионов копий во всех регионах.
В ходе исследовательского анализа данных пришли к выводу, что рассматривать данные за все года нецелесообразно. За весь период представленных данных, обстоятельства, влияющие как на само количество игр, так и на их продажи были не сопоставимыми. Также вторым фактором, сужающим срок анализа - это жизненный цикл платформы игры. В ходе исследования выяснилось, что характерный срок появления и ухода платформ это 9-11 лет. Появляется платформа, идет рост продаж, затем плавный или резкий спад и платформа исчезает.
Поэтому лучше рассматривать данные за период три года для построения прогноза.
Принято решение, что актуальный период для анализа это 2014,2015 и 2016 год. учтено, что данные за 2016 год могут быть неполными.

За актуальный период 2014-2016 (3 года) игры продавались на десяти платформах.
Оценили по графику какие платформы лидируют по продажам, растут или падают. Выделили 4 платформы претендента на лидеров продаж и с подходящим жизненным циклом, а именно - не угасающим, растущим.
Это:

- PS4. Выбор обусловлен тем,что платформа лидер у нее самый высокий результат по продажам в 2015 году и подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. падение в 2016 году не показательно. помним, что данные могут быть неполными. Однако и при таких данных в 2016 году это самые высокие показатели.
- XOne. Выбор обусловлен тем, что у платформы второй результат по количеству продаж как в 2015 году, так и в 2016 году. А также подходящий под наши критерии жизненный цикл - старт продаж в 2014 году. Опять же падение продаж в 2016 году и второе место - не показательно. помним, что данные могут быть неполными и место по продажам может быть выше.
- WiiU. На графике эта платформа не показывает каких-то грандиозных выдающихся продаж, в 2015 году есть небольшой спад. Но всеже она по сроку жизненного цикла и уровню продаж в 2014 году может занимать третье место.
- PC. Выбор обусловлен уникальным жизненным циклом. Платформа жизнеспособна на протяжении с всего изучаемого периода с 1985. Пик был в 2011 году. Она то становится более популярное, то менее. Однако до сих пор актуальна. Хотя конечно показатели по продажам за все предыдущие года не такой высокий как у других платформ. Зато стабильный.

Построили график «ящик с усами» по глобальным продажам игр в разбивке по платформам. Границы этих платформ высокие, но при всем при этом медиана у всех платформ, кроме PS и PS3, не выходит за рамки 0.25 млн продаж копий.

Диаграмма размаха подтвердила ранее полученные выводы:
- PS4 -самая продаваемая платформа - распределение смещено в сторону максимальных значений.
- XOne - вторая по популярности платформа.
- WiiU - занимает третье место.
- Платформа PC показала что уровень ее медианы лежит по границе нижнего квартиля всех трех остальных платформ, а верхний квартиль платформы PC лежит в границах медиан остальных трех платформ. Максимальное значение не дотягивает до границы верхнего квартиля платформы которая на третьем месте WiiU. Это было видно и по предыдущим графикам. Зато это платформа стабильная. Стоит это иметь ввиду.

Рейтинг критиков и пользователей показывает не сильную корреляцию на глобальные продажи. Это справедливо как для лидера продаж по платформе PS4, так и по всем платформам.
Общее распределение игр по игр по жанрам показало, что люди сильно предпочитают активные адреналиновые игры, чем стратегические, логические, также никаких удивлений это не вызывает.
Самыми популярными жанрами являются Shooter и Sport, Platform. Action лидирует в числе общих продаж видимо за счет каких-то звездных продуктов. Медианные значения из лидера привели только на 7-е место.
Не перспективными в плане продаж являются игры в жанре Adventure, Puzzle и Strategy.
Как итог: портрет потенциально прибыльной игры:
- это игра в жанрах Shooter и Sport и Platform
- на платформах - XOne, PS4, WiiU,
- при этом рейтинг пользователя или критика не важны, так как они не имеют сильного влияния на продажи.

Отдельно стоит упомянуть платформу PC - это неумираемый пока вид платформы. Если XOne, PS4, WiiU свой жизненный цикл отживут, после появятся новые платформы, то высока доля вероятности, что эти новые платформы встретит бессмертный PC. считаю, что не стоит сбрасывать со счетов этуу платформу.

Она показывает устойчивость, за счет которой можно получать прибыль без необходимости разрабатывать новую платформу.

Провели исследование портрета пользователя каждого региона (Северная Америка, Европа, Япония) на предмет предпочтения жанра и платформ, а также влияния рейтинга ESRB и выяснили следующее.
- Самые популярные платформы (топ-5) и их различия в долях продаж.
- Наибольшие точки пересечений среди всех трех регионов среди платформ PS3, PS4, 3DS.

- По глобальным продажам только PS4 является фаворитом - 243,67 млн. проданных копий и охватывает все три региона. XOne тоже занимает лидирующее место по продажам 127.52 млн копий, но в топ-5 в Японии его нет. 3DS занимает третье место среди глобальных продаж, при этом большая часть продаж ориентирована на Японию, а остальные продажи делят между собой Европа и Северная Америка.

Самые популярные жанры (топ-5).

Action - вот что подходит для всех трех регионов среди самых продаваемых жанров. Shooter занимает второе место по популярности среди глобального рынка, но в Японии его доля среди топ 5 самая минимальная. Среди жанров с одинаковой популярностью среди трех регионов это Role-Playing - его число глобальных продаж на третьем месте. Европейцы и Североамериканцы предпочитают динамичные игры!
В Японии есть жанр, которого нет в топ-5 других стран и он там очень популярен, это жанр Misc. Это игра, сочетающая геймплей шутера с видом от первого лица с большим количеством игроков, подключенных через Интернет.
Рейтинг ESRB не влияет на продажи в отдельном регионе.
Проверили гипотезы и выяснили, что при пороговом значении в 5%:

- можно утверждать, что средние пользовательские рейтинги платформ Xbox One и PC равны;
- можно утверждать, что средние пользовательские рейтинги жанров Action и Sports разные.

Резюмируем. При планировании рекламных акций компьютерных игр можно придерживаться двух направлений:

- Либо учитывать различия по предпочтениям пользователей из разных регионов
- Либо не учитывать предпочтения пользователей из разных регионов

Так как магазин глобальный и продает компьютерные игры по всему миру, то стратегия, в которой учитываются предпочтения пользователей из разных регионов является предпочтительней. Шире охват. Однако и расходов на расширение ассортимента будет больше, так как рекламная кампания вероятно должна будет направлена на несколько продуктов.

Игры, ассортимент которых должен быть представлен как основной такие:

- на платформах PS4, XOne, 3DS и для японского рынка WiiU
- жанр игры Shooter, Sports, Platform и Role-Playing, Action, а для японского рынка Misc
- можно не считать главным критерием оценку критиков или пользователей, а также рейтинг ESRB. И учитывать эти данные при подборе игр только при прочих равных критериях отбора. Явных преимуществ эти оценки при выборе пользователей не дают. Однако нельзя не учесть наличие некого удобства при наличии таких оценок, как дополнительного фильтра при подборе.

Учитывая проведенный анализ, можно предположить, что потенциально популярный продукт в 2017 году, по которому можно планировать рекламные кампании- это игра на платформе PS4 в жанре Shooter.