Проект: Анализ мирового рынка видеоигр с 1980 по 2016 годы (с прогнозами на 2017)

План исследования:

1) **Импортировать подобранные данные, и провести их первичный осмотр**

2) **Произвести предобработку, обработав и исправив:**

- Названия столбцов
- Пропуски в данныхв
- Дубликаты
- Типы данных

А также путем вычеслений преобразовав новые данные

3) **Осуществить исследовательский анализ, а именно:**

- Выявить акьтвность выпуска видеоигр по годам
- Определить наиболее успешные компании выпуска видеоигр во временном промежутке (по критерием выпускаемости и продоваемости видеоигр)
- Отследить, как менялись продажи по платформам. Определить средний период популярности отдельной платформы
- Определить потенциальных лидеров на 2017 год
- Выявить, как влияют на продажи внутри отдельной платформы отзывы пользователей и критиков. Сравнить состояние по выборке из платформ
- Отобразить распределение игр по жанрам (по тем же критериям: выпускаемость и продоваемость)

4) **Составить портрет пользователя каждого региона и сравнить их**

- Самые популярные платформы (топ-5)
- Самые популярные жанры (топ-5)
- Влияние рейтинга ESRB на продажи в отдельном регионе

5) **Осуществить проверку статистических гипотез**

- 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) **Подвести итоги работы**

Цель исследования:

Выявить тенденции пользовательского интереса к конкретным игровым платформам во временном разрезе, для передачи информации разработчикам видеоигр о перспективных игровых платформах

Набор данных

Поле	Описание
airports_nearest	Расстояние до ближайшего аэропорта в метрах (м)
balcony	Количество балконов
ceiling_height	Высота потолков в метрах (м)
cityCenters_nearest	Расстояние до центра города в метрах (м)
days_exposition	Сколько дней было размещено объявление (от публикации до снятия)
first_day_exposition	Дата публикации
floor	Этаж
floors_total	Всего этажей в доме
is_apartment	Апартаменты (булев тип)
kitchen_area	Площадь кухни в квадратных метрах ($м^2$)
last_price	Цена на момент снятия с публикации
living_area	Жилая площадь в квадратных метрах ($м^2$)
locality_name	Название населенного пункта
open_plan	Свободная планировка (булев тип)
parks_around3000	Количество парков в радиусе 3 км
parks_nearest	Расстояние до ближайшего парка в метрах (м)
ponds_around3000	Количество водоемов в радиусе 3 км
ponds_nearest	Расстояние до ближайшего водоема в метрах (м)
rooms	Количество комнат
studio	Квартира-студия (булев тип)
total_area	Общая площадь квартиры в квадратных метрах ($м^2$)
total_images	Количество фотографий квартиры в объявлении

Импорт библиотек

In [1]:
```
#Импортируем необходимые библиотеки
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from matplotlib import cm
from scipy import stats as st
```

Знакомство с данными

In [2]:
```
#Отрываем датафрейм и выводим первые строки

filename = 'games.csv'

try:
    # Попытка прочитать данные из файла CSV
    games = pd.read_csv(filename)
```

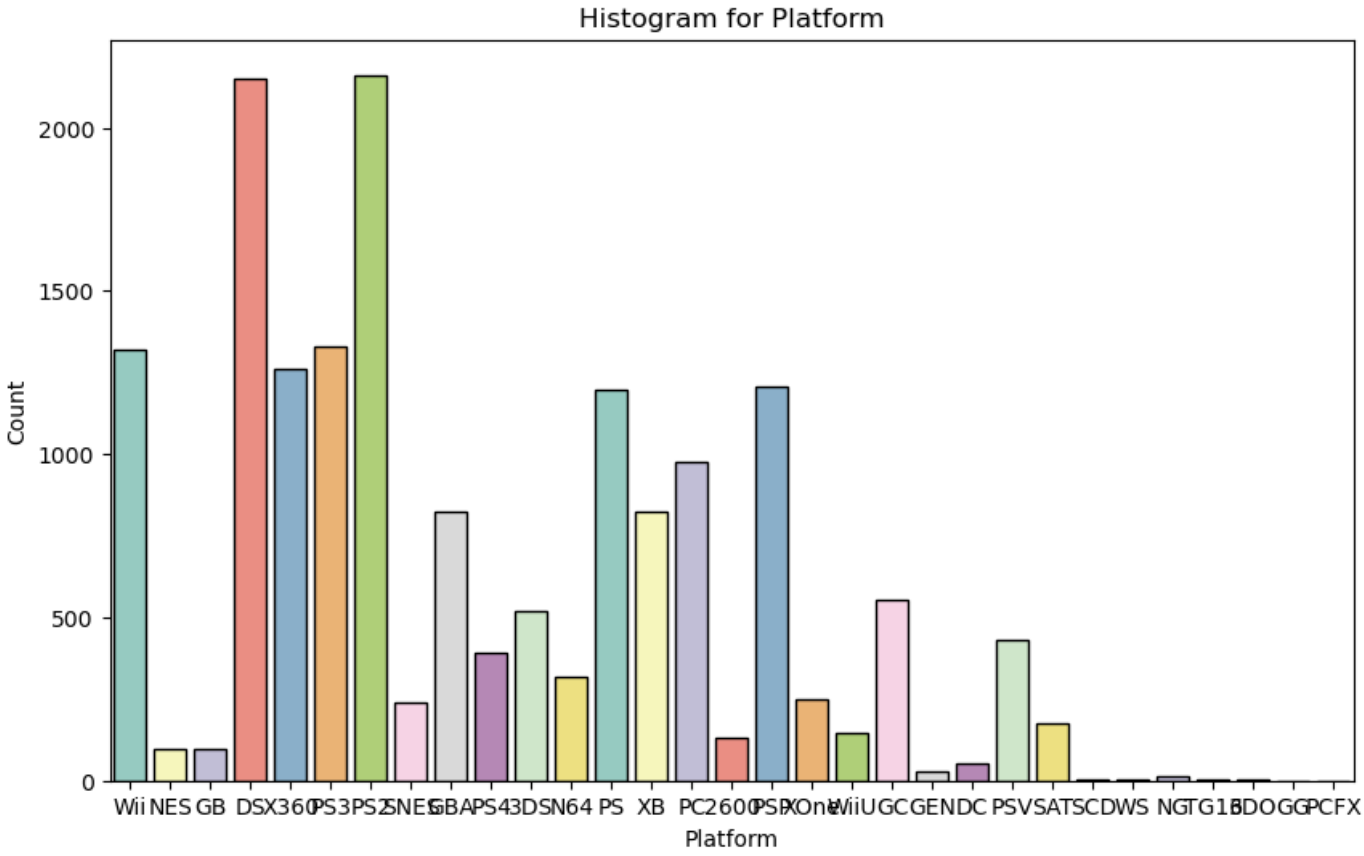
```
    print("Файл успешно прочитан:")
    display(data.head())
except FileNotFoundError:
    # Обработка случая, когда файл не найден
    print(f"Файл '{filename}' не найден. Пожалуйста, проверьте путь к файлу.")
except Exception as e:
    # Обработка всех остальных типов исключений
    print(f"Произошла ошибка при чтении файла '{filename}': {e}")
```

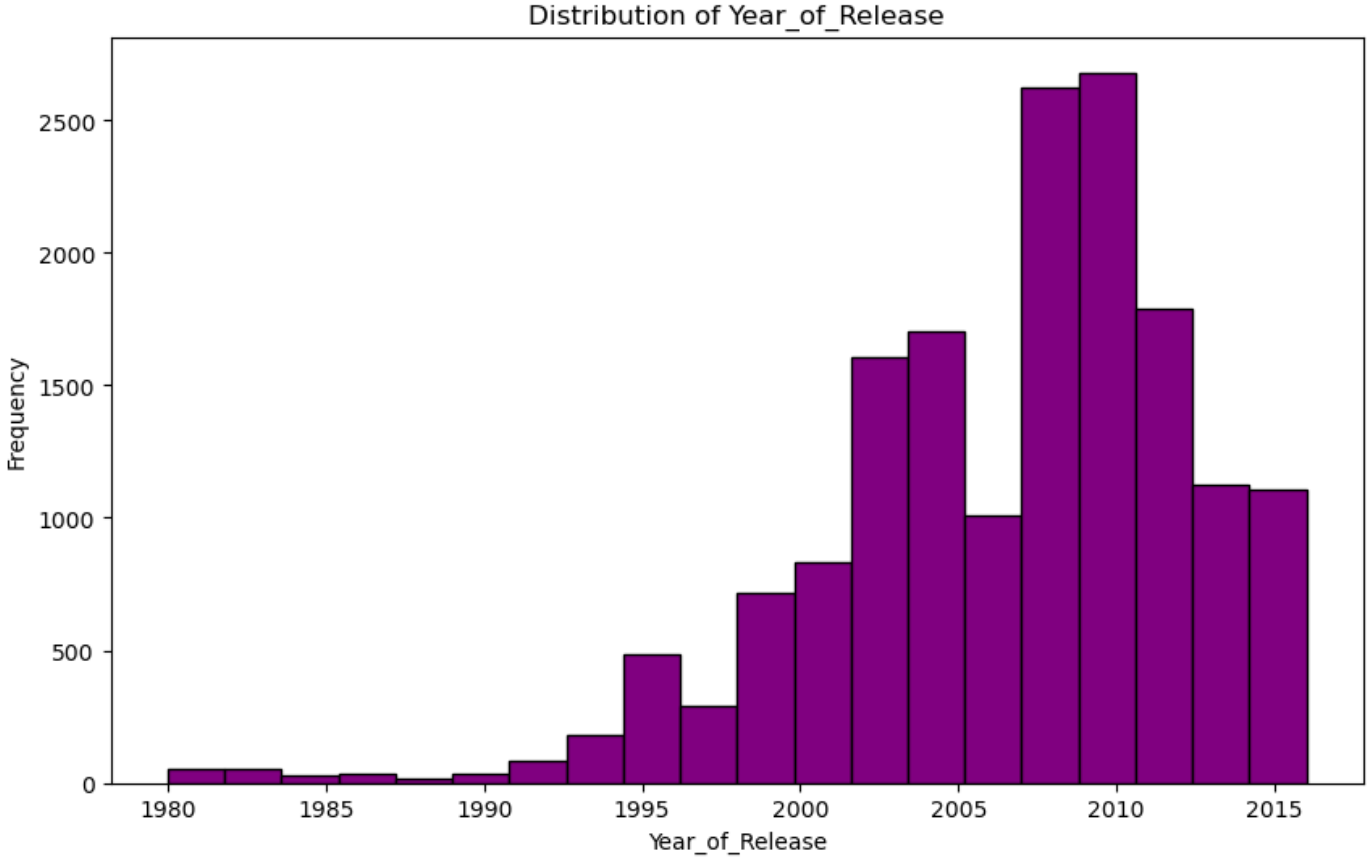
```
Файл успешно прочитан:
Произошла ошибка при чтении файла 'games.csv': name 'data' is not defined
```

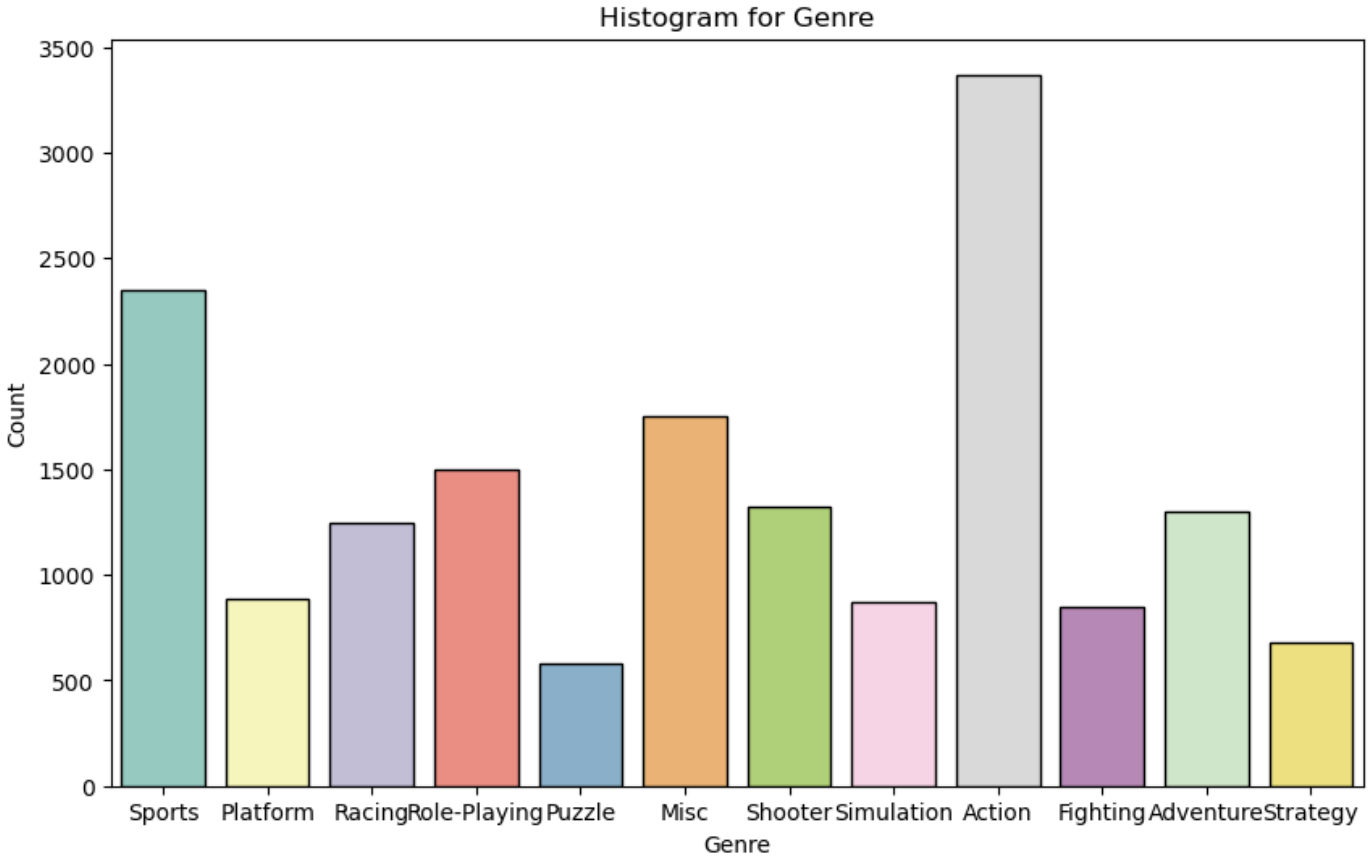
In [3]:
```
#Смотрим общую информацию о данных
games.info()
```

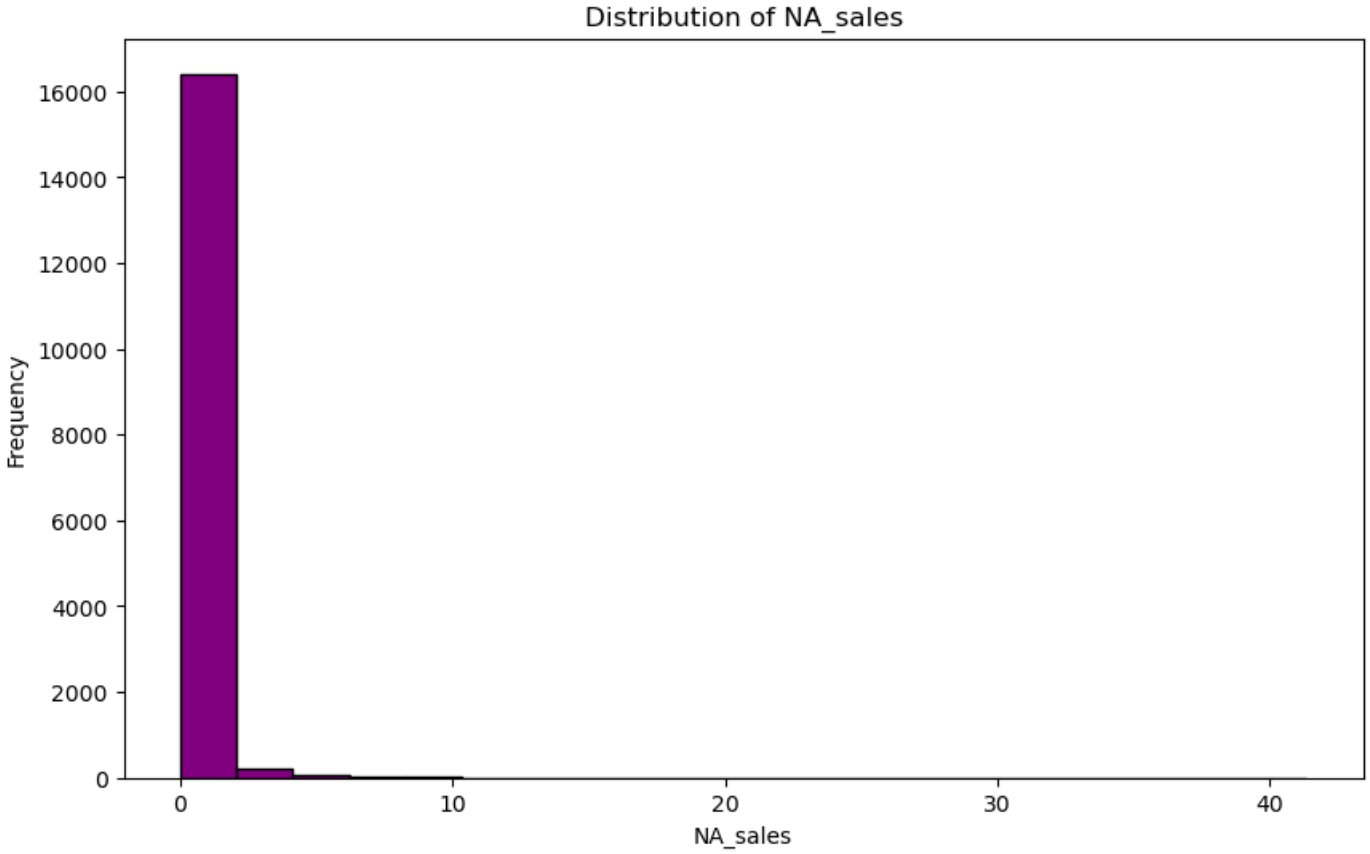
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16713 non-null  object 
 1   Platform         16715 non-null  object 
 2 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

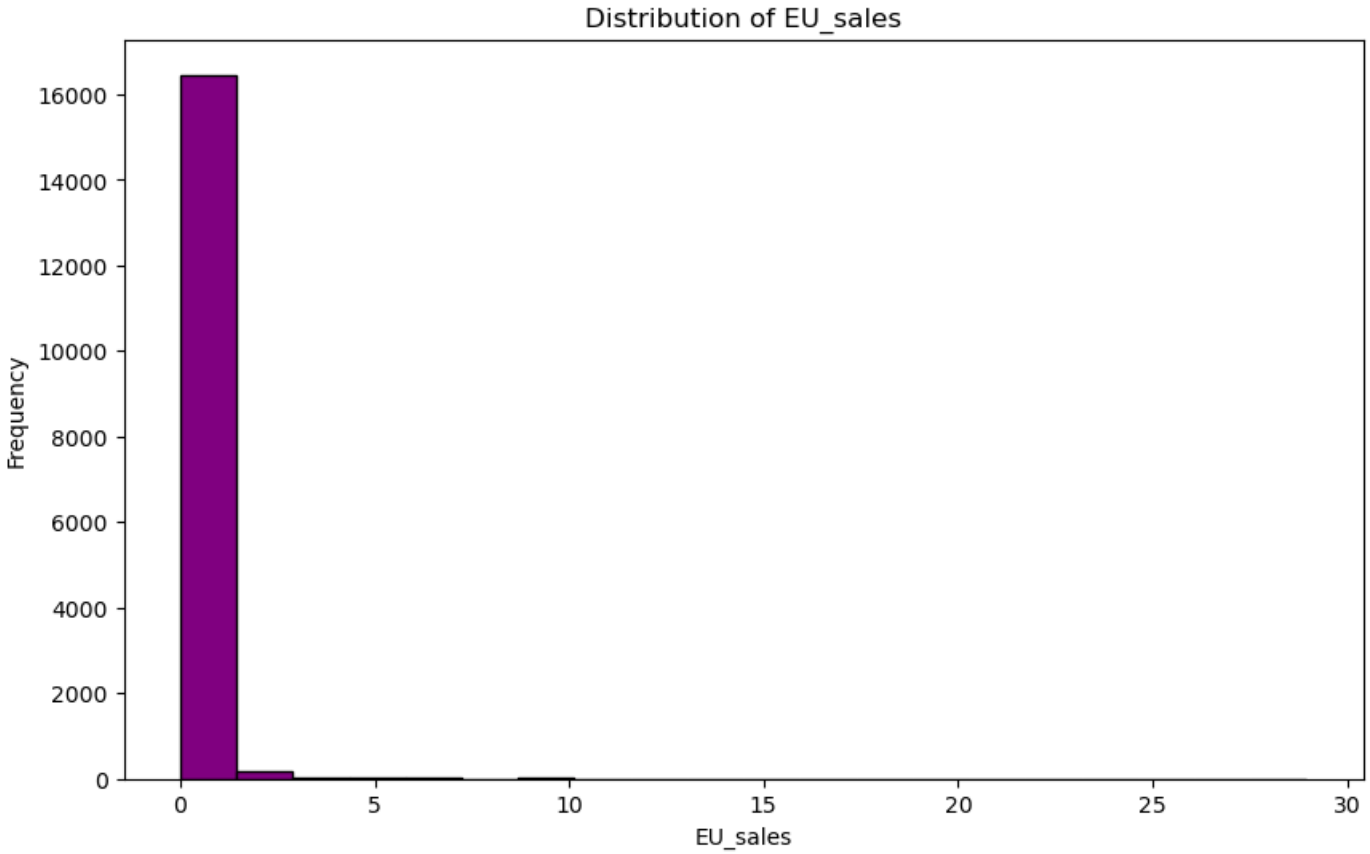
In [4]:
```
# Предполагается, что data - это ваш DataFrame
for column in games.columns:
    if column != 'Name':  # Пропускаем качественные данные

        plt.figure(figsize=(10, 6))
        if games[column].dtype == 'object':  # Для категориальных данных
            sns.countplot(x=column, data=games, palette='Set3', edgecolor='black')
            plt.title(f'Histogram for {column}')
            plt.xlabel(column)
            plt.ylabel('Count')
        elif games[column].dtype == 'int64' or games[column].dtype == 'float64':  # Для
            if games[column].nunique() < 30:  # Проверяем, является ли переменная дискре
                sns.countplot(x=column, data=games, palette='Set3', edgecolor='black')
                plt.title(f'Distribution of {column} (Discrete)')
                plt.xlabel(column)
                plt.ylabel('Count')
            else:
                plt.hist(games[column], bins=20, color='purple', edgecolor='black')
                plt.title(f'Distribution of {column}')
                plt.xlabel(column)
                plt.ylabel('Frequency')
        plt.show()
```

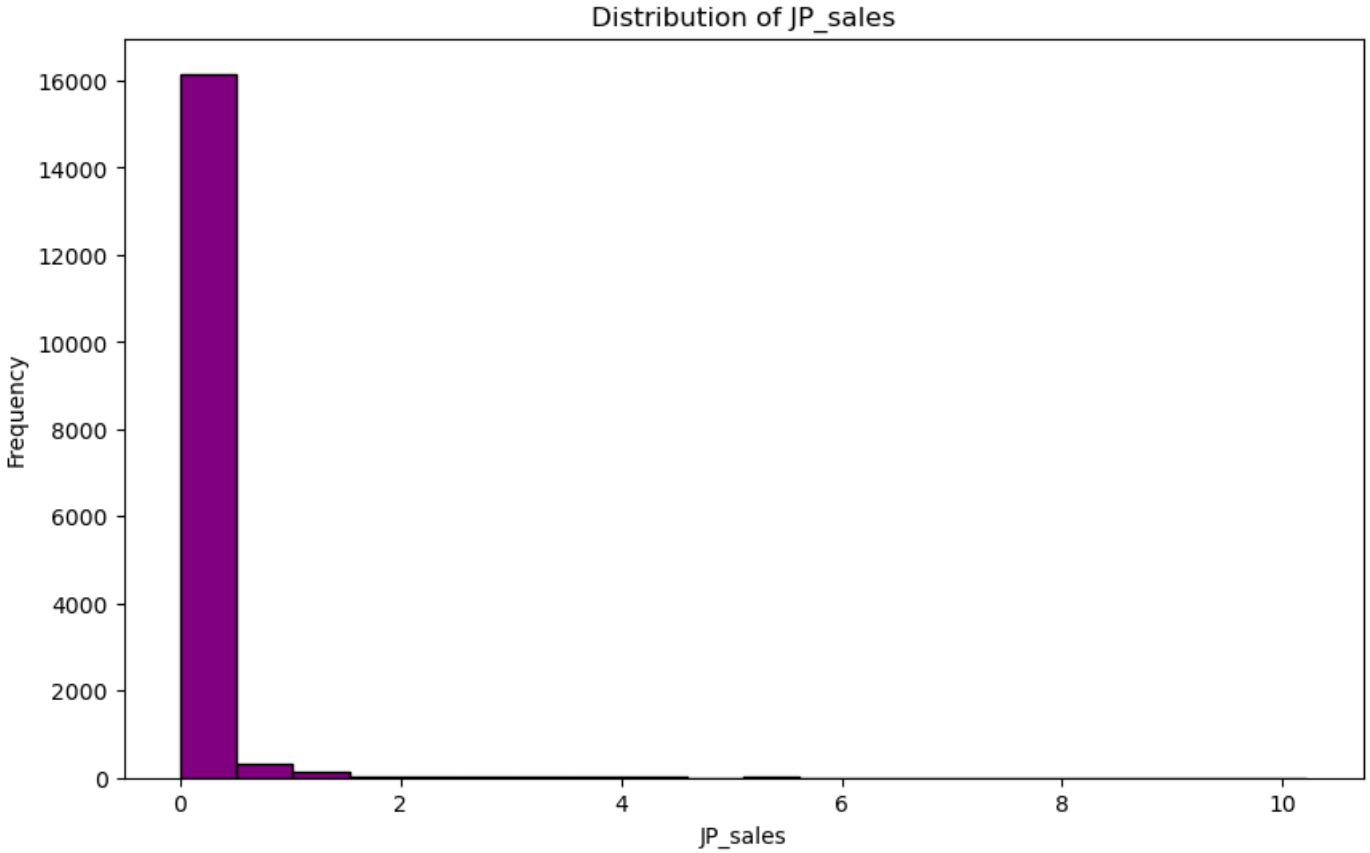

Histogram for Platform
Count
Platform
0
500
1000
1500
2000
Wii
NES
GB
DS
X360
PS3
PS2
SNES
GBA
PS4
3DS
N64
PS
XB
PC
2600
PSP
XOne
WiiU
GC
GEN
DC
PSV
SAT
SCD
WS
NG
TG16
3DO
GG
PCFX

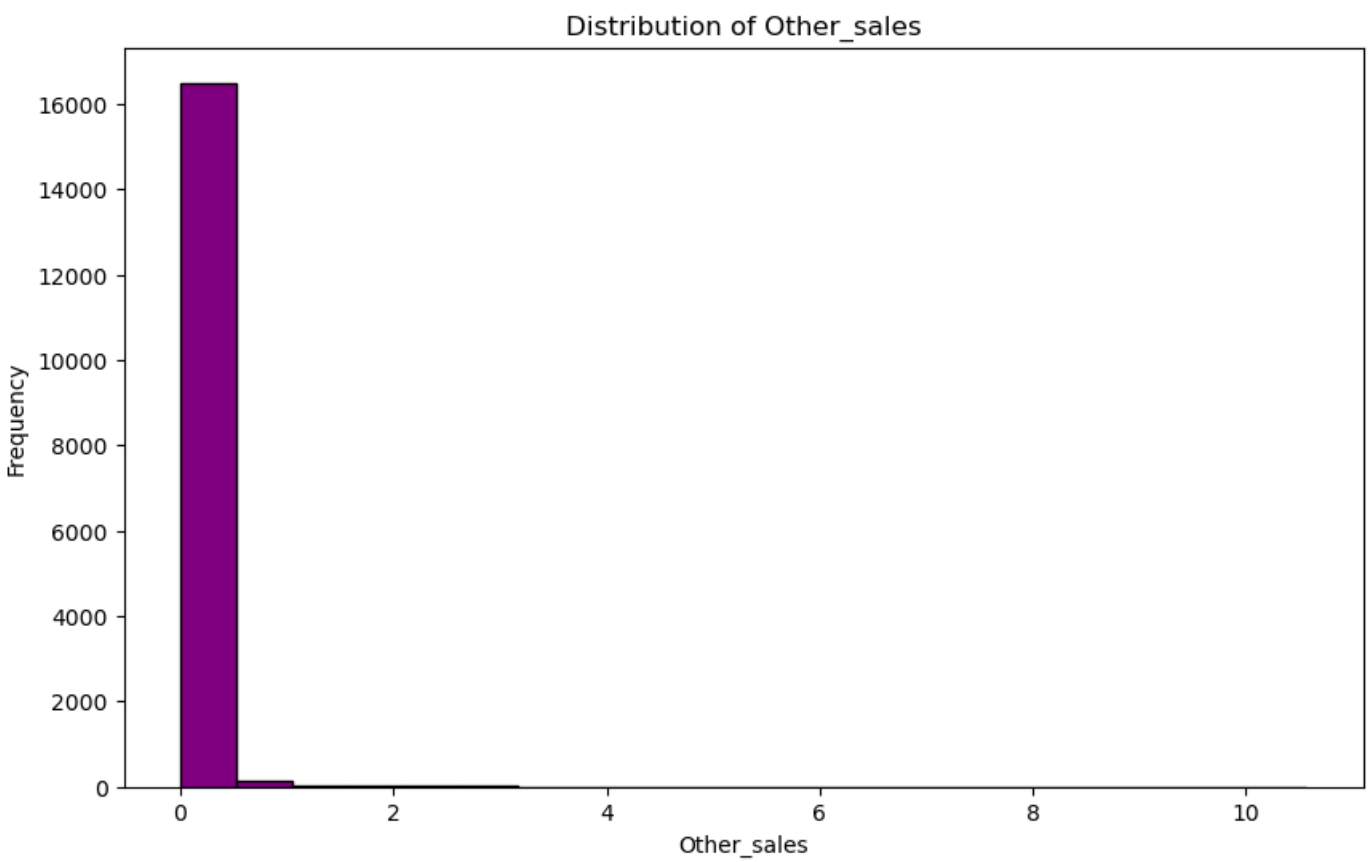

Distribution of Year_of_Release
Frequency
Year_of_Release
0
500
1000
1500
2000
2500
1980
1985
1990
1995
2000
2005
2010
2015

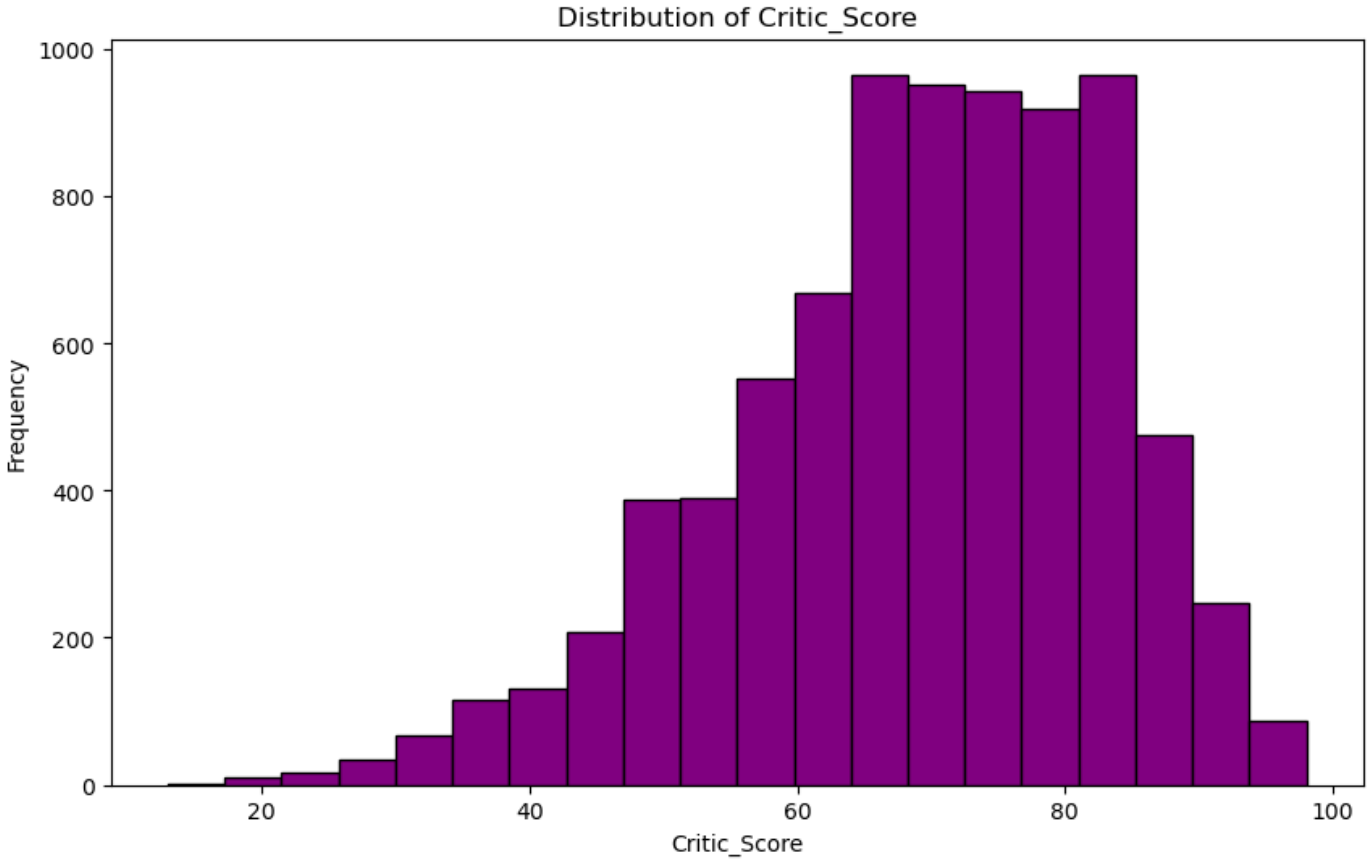

Histogram for Genre
Count
Genre
0
500
1000
1500
2000
2500
3000
3500
Sports
Platform
Racing
Role-Playing
Puzzle
Misc
Shooter
Simulation
Action
Fighting
Adventure
Strategy

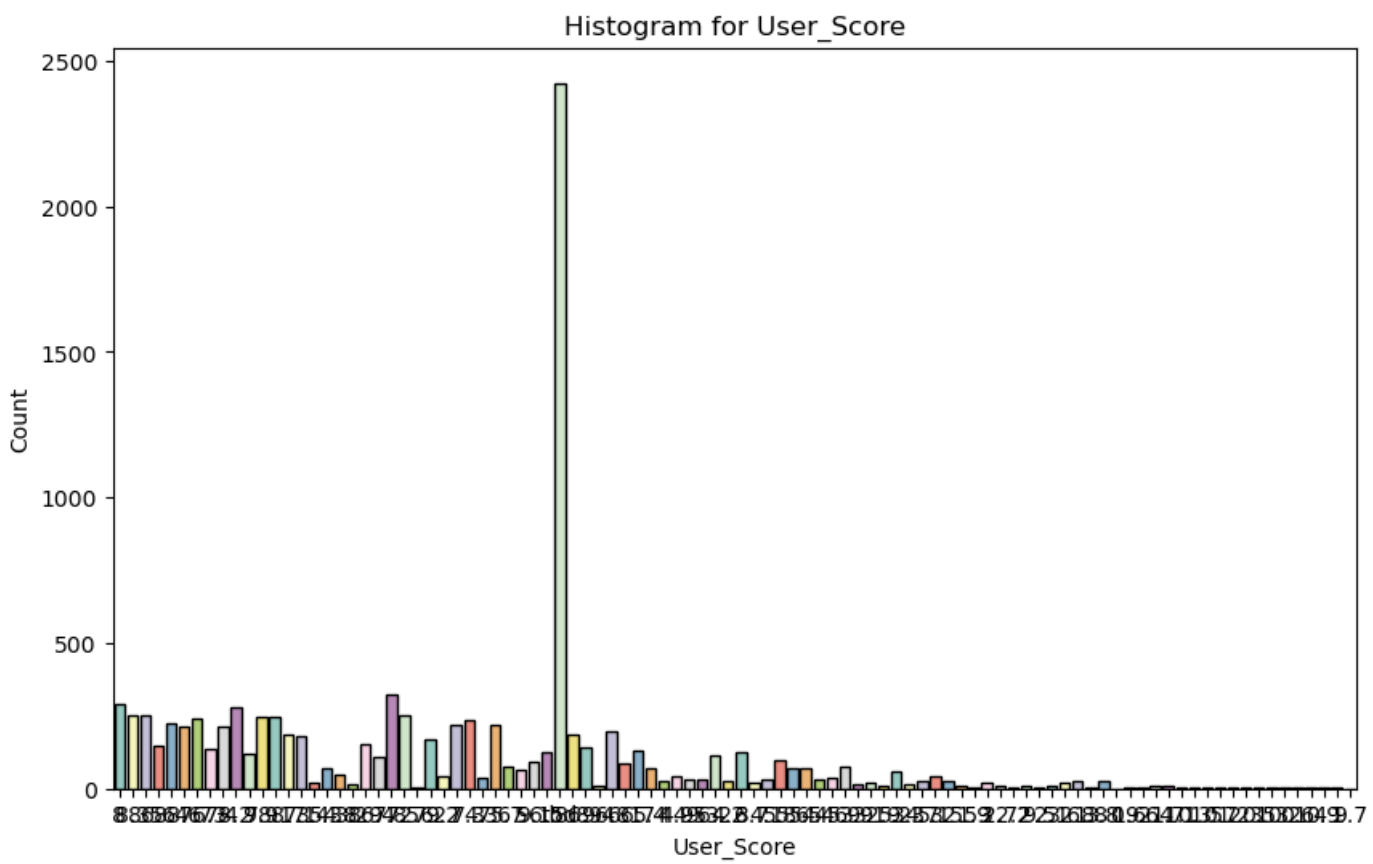

Distribution of NA_sales
Frequency
NA_sales
0
2000
4000
6000
8000
10000
12000
14000
16000
0
10
20
30
40

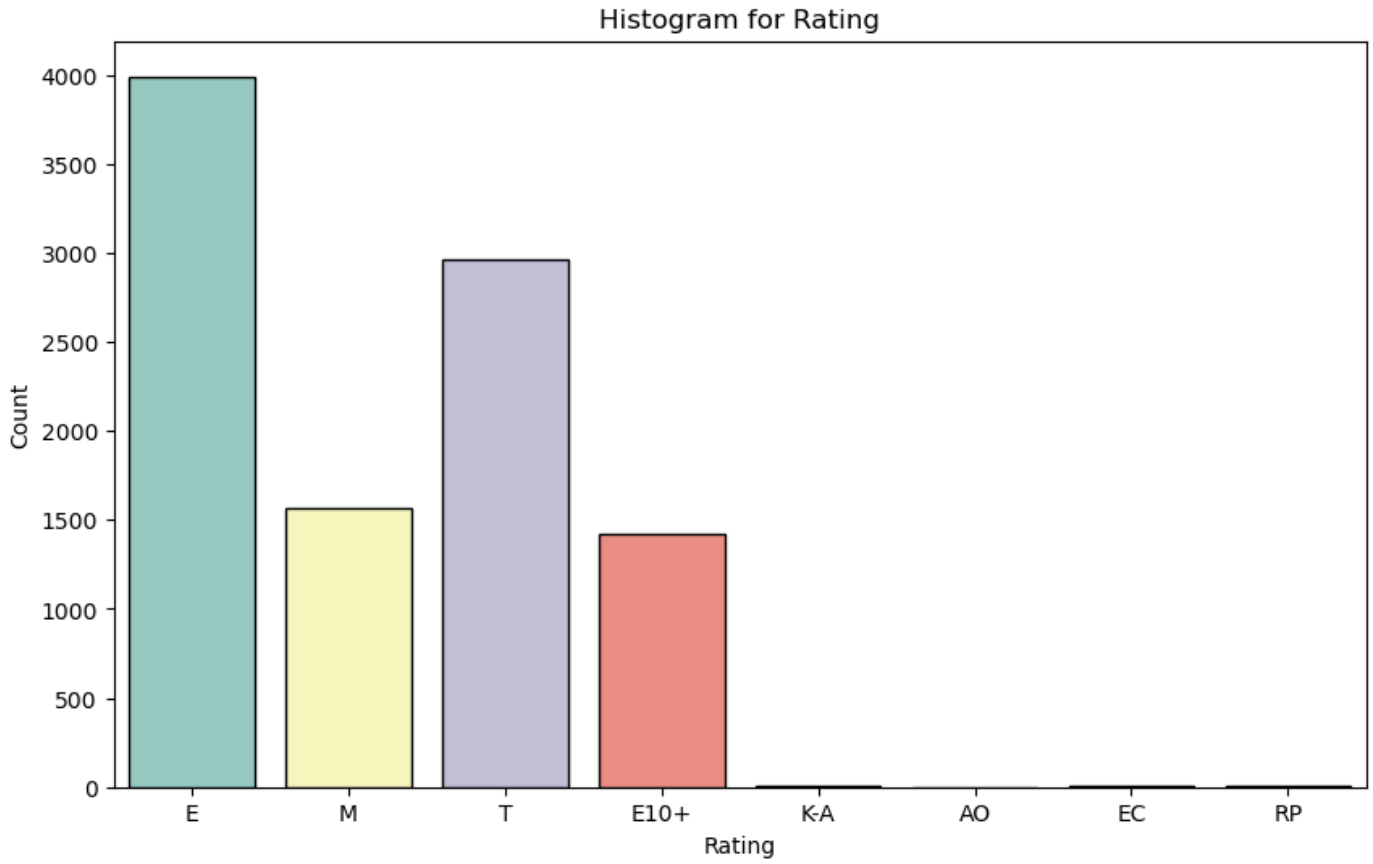

Distribution of EU_sales
Frequency
EU_sales
0
2000
4000
6000
8000
10000
12000
14000
16000
0
5
10
15
20
25
30


Distribution of JP_sales
Frequency
JP_sales
0
2000
4000
6000
8000
10000
12000
14000
16000
0
2
4
6
8
10


Distribution of Other_sales
Frequency
16000
14000
12000
10000
8000
6000
4000
2000
0
0
2
4
6
8
10
Other_sales


Distribution of Critic_Score
Frequency
1000
800
600
400
200
0
20
40
60
80
100
Critic_Score





В целом, аномальных отличий и распределений на гистограммах не наблюдается, все количественные переменные представлены в стандартном виде. *Общее количество наблюдений: 16713*.

Имеются пропуски и некорректные форматы данных. Перейдем к предобработке

Предобработка данных

Работа с названиями переменных

In [5]:
```
#Изменяем стиль названий переменных
games = games.rename(columns={'Name':'name', 'Platform': 'platform', 'Year_of_Release':'

games.columns
```

Out[5]:
```
Index(['name', 'platform', 'year_of_release', 'genre', 'na_sales', 'eu_sales',
       'jp_sales', 'other_sales', 'critic_score', 'user_score', 'rating'],
      dtype='object')
```

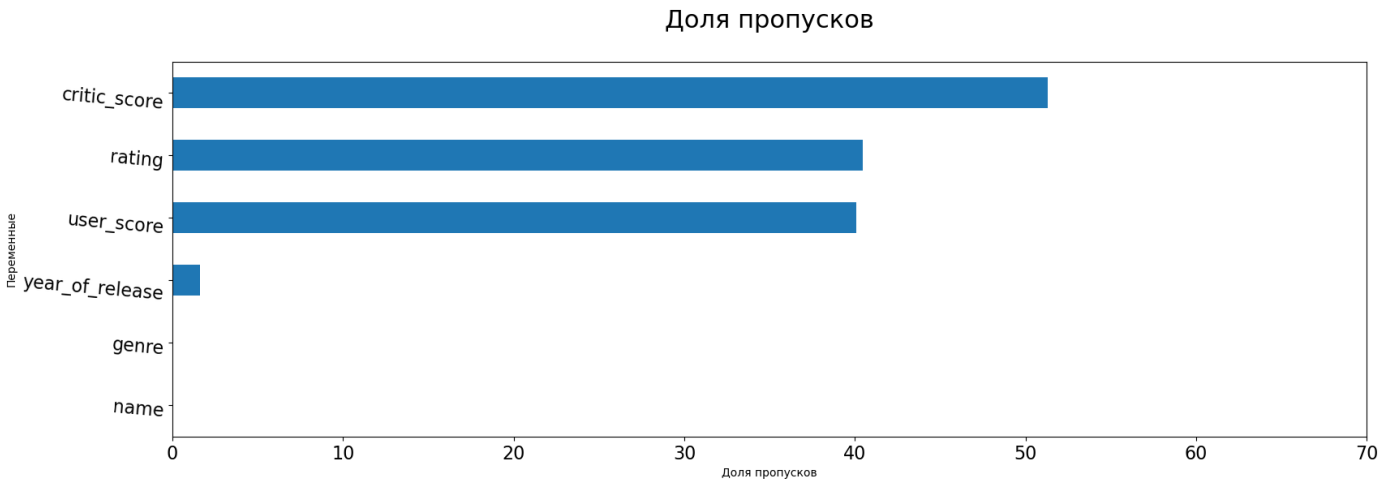
Преобразование новых данных

In [6]:
```
#Создаем переменную с суммарными продажами по всем регионам
games["sum_sales"] = games["na_sales"] + games["eu_sales"] + games["jp_sales"] + games["
```

Обработка пропусков

In [7]:
```
# пропущенные значения бары

def pass_value_barh(df):
    try:
        (
            (df.isna().mean()*100)
            .to_frame()
            .rename(columns = {0:'space'})
            .query('space > 0')
            .sort_values(by = 'space', ascending = True)
            .plot(kind = 'barh', figsize = (19,6), rot = -5, legend = False, fontsize =
            .set_title('Доля пропусков' + "\n", fontsize = 22)
        );
        plt.xlim(0, 70)
        plt.xlabel("Доля пропусков")
        plt.ylabel("Переменные")
    except:
        print('пропусков не осталось :) или произошла ошибка в первой части функции ')
```

In [8]:
```
pass_value_barh(games)
```



Так, мы наблюдаем что отсутствуют значения "year_of_release", "critic_score", "user_score", "rating". В годах выпуска отсутствует небольшое количество значений. Больше всего пропусков в оценках и рейтингах. Пропусвки в них занимают существенную долю.

Перейдем к работе с пропускам по выделенным переменным

year_of_release

Начнем с года выпуска, так как это переменная находится в отдельной категории и имеет меньше всего пропусков

In [9]:
```
#Смотрим нулевые значения по годам
games.loc[games[games['year_of_release'].isnull() == True].index]["name"].value_counts()
```

Out[9]:
```
name
LEGO Harry Potter: Years 5-7          7
Happy Feet Two                        4
Rock Band                             4
Test Drive Unlimited 2                3
Bejeweled 3                           3
WRC: FIA World Rally Championship     3
Singularity                           2
NBA Live 2003                         2
Silent Hill: Homecoming               2
Rock Revolution                       2
Name: count, dtype: int64
```

In [10]:
```
# check
# Подсчитываем жизненный срок платформы

years_of_life = games.pivot_table(index='platform',values='year_of_release',aggfunc=['mi
years_of_life.columns=['min','max']
years_of_life['years_of_life']=years_of_life['max']-years_of_life['min']
years_of_life.head(15)

# years_of_life.sort_values(by = 'years_of_life', ascending = False)
```

Out[10]:

platform	min	max	years_of_life
2600	1980.0	1989.0	9.0
3DO	1994.0	1995.0	1.0
3DS	2011.0	2016.0	5.0
DC	1998.0	2008.0	10.0
DS	1985.0	2013.0	28.0
GB	1988.0	2001.0	13.0
GBA	2000.0	2007.0	7.0
GC	2001.0	2007.0	6.0
GEN	1990.0	1994.0	4.0
GG	1992.0	1992.0	0.0
N64	1996.0	2002.0	6.0
NES	1983.0	1994.0	11.0
NG	1993.0	1996.0	3.0
PC	1985.0	2016.0	31.0
PCFX	1996.0	1996.0	0.0

In [87]:
```
#Смотрим сколько пропусков
print('Доля пропусков в годе выпуска:')
games["year_of_release"].isna().sum()
```

```
Доля пропусков в годе выпуска:
0
```

Out[87]:

In [12]:
```
#Открываем срез с пропусками
games[games['year_of_release'].isnull() == True].head()
```

Out[12]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_sc
183	Madden NFL 2004	PS2	NaN	Sports	4.26	0.26	0.01	0.71	94.0	
377	FIFA Soccer 2004	PS2	NaN	Sports	0.59	2.36	0.04	0.51	84.0	
456	LEGO Batman: The Videogame	Wii	NaN	Action	1.80	0.97	0.00	0.29	74.0	
475	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	1.57	1.02	0.00	0.41	NaN	N
609	Space Invaders	2600	NaN	Shooter	2.36	0.14	0.00	0.03	NaN	N

Интересный факт, что в названиях некоторых игр находится пропущенная дата релиза. Хоть это и совершенно незначительная часть, сделаем эту замену.

Заменим пропуски на нули

In [88]:
```
#Замена
games.loc[games["year_of_release"].isnull() == True, "year_of_release"] = 0

pd.options.mode.chained_assignment = None
year = []
years = []

#Напишем код, для замены пропусков в дате релиза, в тех местах, где дата расположена в н
for i in games[games['year_of_release'] == 0].index:
    for j in games["name"][i]:
        if j in ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]:
            year.append(j)
    if len(year) > 3:
        games['year_of_release'][i] = int("".join(year))
    year = []

#Убедимся, что все заменилось
games["year_of_release"].unique()
```

Out[88]:
```
array([2006, 1985, 2008, 2009, 1996, 1989, 1984, 2005, 1999, 2007, 2010,
       2013, 2004, 1990, 1988, 2002, 2001, 2011, 1998, 2015, 2012, 2014,
       1992, 1997, 1993, 1994, 1982, 2016, 2003, 1986, 2000, 1995, 1991,
       1981, 1987, 1980, 1983])
```

In [89]:
```
#Смотрим сколько пропусков
games.loc[games["year_of_release"] == 0]["name"].count()
```

Out[89]:
```
0
```

In [15]:
```
#Какой процент занимают данные без года от общего массива
print(games.loc[games["year_of_release"] == 0, "year_of_release"].count() / games["name"
```

```
print(games.loc[games["year_of_release"] == 0, "sum_sales"].sort_values())

#Таких данных около процента, а продажи у таких игр не превышают 3 млн.
#Следовательно их удаление не отразится на исследовании популярных платформ. Удалим эти
games = games.loc[games['year_of_release'] != 0]
```

```
0.015018249267037636
16522    0.01
15816    0.01
15966    0.01
16017    0.01
16059    0.01
         ...
678      2.33
657      2.40
627      2.47
609      2.53
456      3.06
Name: sum_sales, Length: 251, dtype: float64
```

In [16]:
```
#Смотрим сколько пропусков
games.loc[games["year_of_release"] == 0]["name"].count()
```

Out[16]:
```
0
```

name & genre

In [17]:
```
#Отследим в каких наблюдениях пропуски совпадают
games.loc[games["name"].isna() & games["genre"].isna()]

#Пропуски по жанру и по имени присутствуют в одних и тех же наблюдениях в малом количест
#Соответственно их можно и даже нужно удалить

#Исключим значения с пропуском в жанре и в имени
games.dropna(axis = 'index', subset = ['name'], inplace = True)
games.info()

games.loc[games["name"].isna()].index
```

Out[17]:
```
<class 'pandas.core.frame.DataFrame'>
Index: 16462 entries, 0 to 16714
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16462 non-null  object
 1   platform         16462 non-null  object
 2   year_of_release  16462 non-null  float64
 3   genre            16462 non-null  object
 4   na_sales         16462 non-null  float64
 5   eu_sales         16462 non-null  float64
 6   jp_sales         16462 non-null  float64
 7   other_sales      16462 non-null  float64
 8   critic_score     7996 non-null   float64
 9   user_score       9852 non-null   object
 10  rating           9780 non-null   object
 11  sum_sales        16462 non-null  float64
dtypes: float64(7), object(5)
memory usage: 1.6+ MB
Index([], dtype='int64')
```

critic_score & user_score & rating

Только в трех переменных отсутствуют значения в большом количестве: **в переменных с пользовательским и экспертным рейтингами, а также с рейтингом от организаций (ESRB).**

In [18]:
```
#Рассмотрим уникальные значения
games["user_score"].unique()
```

Out[18]:
```
array(['8', nan, '8.3', '8.5', '6.6', '8.4', '8.6', '7.7', '6.3', '7.4',
       '8.2', '9', '7.9', '8.1', '8.7', '7.1', '3.4', '5.3', '4.8', '3.2',
       '8.9', '6.4', '7.8', '7.5', '2.6', '7.2', '9.2', '7', '7.3', '4.3',
       '7.6', '5.7', '5', '9.1', '6.5', 'tbd', '8.8', '6.9', '9.4', '6.8',
       '6.1', '6.7', '5.4', '4', '4.9', '4.5', '9.3', '6.2', '4.2', '6',
       '3.7', '4.1', '5.8', '5.6', '5.5', '4.4', '4.6', '5.9', '3.9',
       '3.1', '2.9', '5.2', '3.3', '4.7', '5.1', '3.5', '2.5', '1.9', '3',
       '2.7', '2.2', '2', '9.5', '2.1', '3.6', '2.8', '1.8', '3.8', '0',
       '1.6', '9.6', '2.4', '1.7', '1.1', '0.3', '1.5', '0.7', '1.2',
       '2.3', '0.5', '1.3', '0.2', '0.6', '1.4', '0.9', '1', '9.7'],
      dtype=object)
```

В значениях присутствует такой элемент, как tbd что означает - будет оценено. Поскольку оценка на данный момент отсутствует, заменим на пропуски, чтобы можно было перемести переменную в числовой формат

In [19]:
```
#Заменяем на пропуски
games.loc[games["user_score"] == "tbd", "user_score"] = np.nan
#Проверяем
games.loc[games["user_score"] == "tbd"]
```

Out[19]:

name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating

In [20]:
```
games[['critic_score', 'user_score', 'rating']].isnull().corr()
```

Out[20]:

	critic_score	user_score	rating
critic_score	1.000000	0.799410	0.783011
user_score	0.799410	1.000000	0.732882
rating	0.783011	0.732882	1.000000

Итак, пропущенные значения очень активно коррелируют друг с другом, причем положительно, что означает, что пропуски в одной переменной относительно соответствуют пропускам в других двух.

Соответственно, есть вероятность, что во всех трех переменных в одних и тех же наблюдениях отсутствуют значения. Проверим это

In [21]:
```
games.loc[games["user_score"].isna() & games["rating"].isna() & games["critic_score"].is
```

Out[21]:
```
6585
```

Целых 6000 тысяч наблюдений с пропусками в трех переменных - достаточно весомое количество. Вместо удаления заменю их на нули, промаркеровав единицами. В случае необходимости работы с рейтингом, отброшу промаркерованные значения.

In [22]:
```
#создаем столбец для маркеров
games["no_scored"] = 0

#маркеруем пропуски единицами
games.loc[games["user_score"].isna() & games["rating"].isna() & games["critic_score"].is
```

In [23]:
```
#Заменяем пропуски на
games.loc[games["no_scored"] == 1, "user_score"] = 0
```

```
games.loc[games["no_scored"] == 1, "rating"] = 0
games.loc[games["no_scored"] == 1, "critic_score"] = 0
```

In [24]:
```
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16462 entries, 0 to 16714
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16462 non-null  object 
 1   platform         16462 non-null  object 
 2   year_of_release  16462 non-null  float64
 3   genre            16462 non-null  object 
 4   na_sales         16462 non-null  float64
 5   eu_sales         16462 non-null  float64
 6   jp_sales         16462 non-null  float64
 7   other_sales      16462 non-null  float64
 8   critic_score     14581 non-null  float64
 9   user_score       14059 non-null  object 
 10  rating           16365 non-null  object 
 11  sum_sales        16462 non-null  float64
 12  no_scored        16462 non-null  int64  
dtypes: float64(7), int64(1), object(5)
memory usage: 1.8+ MB
```

Изменение типов данных

Года измеряются в целых числах (в данных не указаны месяца, только года)

In [25]:
```
#Меняем год релиза на целочисленный тип

games["year_of_release"] = games["year_of_release"].astype("int")
```

Там, где пользовательский рейтинг имеется, он измеряется в числах с плавующей точкой, (то есть бал и дробная доля), так что заменим тип на флот

In [26]:
```
#Меняем рейтинг на числовой формат
games["user_score"] = games["user_score"].astype("float64")
```

In [27]:
```
#Проверим, что все форматы заменились
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16462 entries, 0 to 16714
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16462 non-null  object 
 1   platform         16462 non-null  object 
 2   year_of_release  16462 non-null  int32  
 3   genre            16462 non-null  object 
 4   na_sales         16462 non-null  float64
 5   eu_sales         16462 non-null  float64
 6   jp_sales         16462 non-null  float64
 7   other_sales      16462 non-null  float64
 8   critic_score     14581 non-null  float64
 9   user_score       14059 non-null  float64
 10  rating           16365 non-null  object 
 11  sum_sales        16462 non-null  float64
 12  no_scored        16462 non-null  int64  
dtypes: float64(7), int32(1), int64(1), object(4)
memory usage: 1.7+ MB
```

Проверка дубликатов

In [28]:
```
#Проверяем на наличие явных дубликатов
games.duplicated().sum()
```

Out[28]:
```
0
```

Также проверим дубликаты по ключевым столбцам

In [29]:
```
games.shape
```

Out[29]:
```
(16462, 13)
```

In [30]:
```
#Создадим копию
games_check = games

#приведем значения к нижнему регистру
games_check["name"].str.lower()
games_check["platform"].str.lower()

#Найдем ининдексы неполных дубликатов
drop = games_check[games_check[['name', 'platform', 'year_of_release']].duplicated()].in

#удалим неполные дубликаты
games = games.drop(drop)
```

Явные дубликаты отсутствуют, неполные - удалены, двигаемся дальше

In [31]:
```
games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16461 entries, 0 to 16714
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16461 non-null  object 
 1   platform         16461 non-null  object 
 2   year_of_release  16461 non-null  int32  
 3   genre            16461 non-null  object 
 4   na_sales         16461 non-null  float64
 5   eu_sales         16461 non-null  float64
 6   jp_sales         16461 non-null  float64
 7   other_sales      16461 non-null  float64
 8   critic_score     14580 non-null  float64
 9   user_score       14058 non-null  float64
 10  rating           16364 non-null  object 
 11  sum_sales        16461 non-null  float64
 12  no_scored        16461 non-null  int64  
dtypes: float64(7), int32(1), int64(1), object(4)
memory usage: 1.7+ MB
```

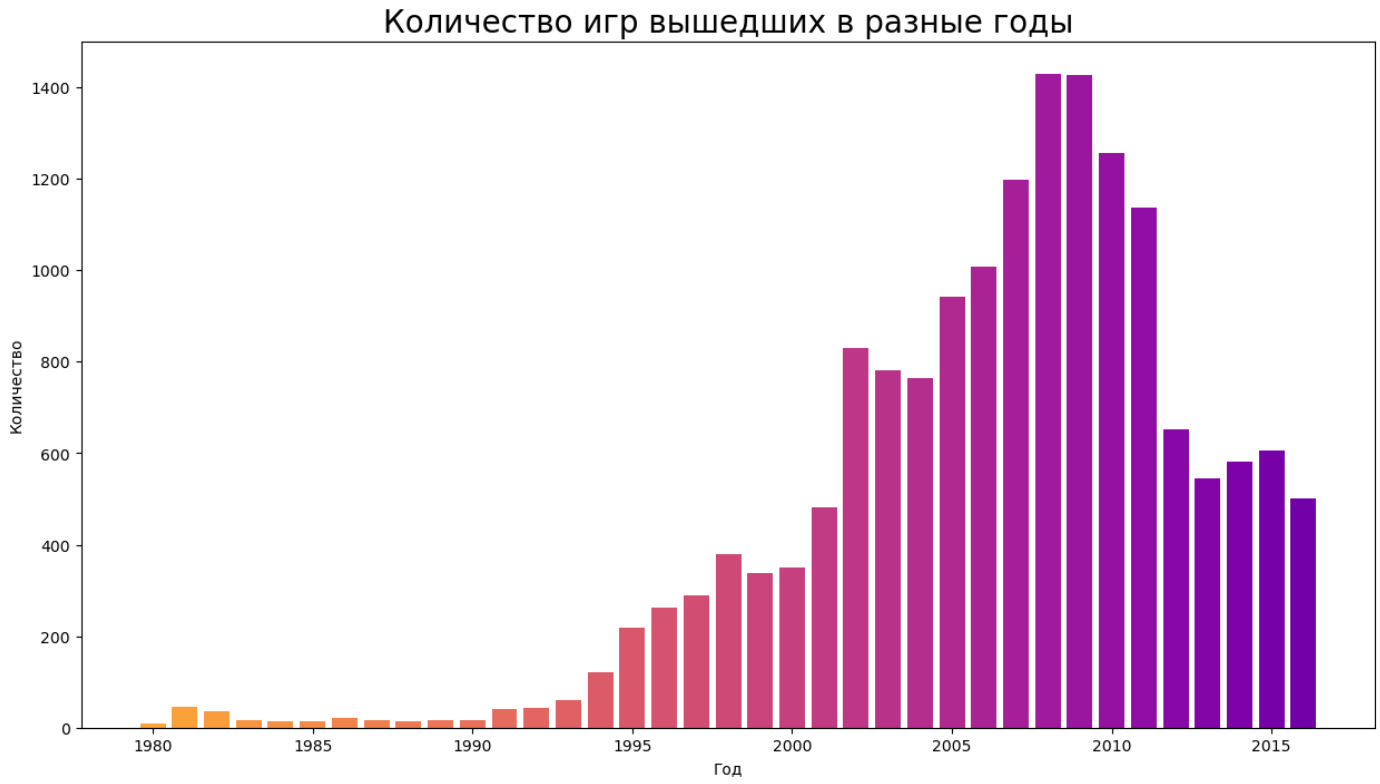
Исследовательский анализ

В рамках ИА блока выполним установленные в плане исследовательские задачи. Каждая задача находится в отдельном блоке

Анализ активности выпуска игр по годам

In [32]:
```
#Построем содную таблицу с подсчетом количества игр, выпущенных за каждый год
year_freq = games.groupby("year_of_release").agg({"name":"count"}).reset_index()
```

```
year_freq.columns = ["year_of_release", "amount"]

#По таблице построим график распределения выпуска игр по годам
fig = plt.figure(figsize=(15, 8))
ax = fig.add_subplot()
color = cm.plasma(np.linspace(.8, .20, 38))

ax.bar("year_of_release", "amount", color=color, data=year_freq)
ax.set_ylabel('Количество')
ax.set_xlabel('Год')
ax.set_title('Количество игр вышедших в разные годы')
ax.title.set_size(20)

plt.show()
```



Так, по диаграмме можно заметить, что основное количество игр начало выпускаться примерно с 1995-2000х годов, до этого рынок видеоигр был куда менее наполнен, индустрия только зарождалась. **Пока что не стану отбрасывать ранние года, поскольку там может находиться уникальная информация, но учту данную статистику, и в зависимости от задачи отброшу или оставлю года до 1995 (или дальше).**

Анализ выпусков и суммарных продаж видеоигр по платформам

In [33]:
```
#Составим рейтинг платформ по количеству выпущенных игр

platforms_amount = (
    games
    .groupby("platform").agg({"name":"count"})
    .reset_index()
    .sort_values(by="name", ascending = False))
platforms_amount.columns = ["platform", "amount"]
platforms_amount.head()
```

Out[33]:

	platform	amount
16	PS2	2135
4	DS	2122

17	PS3	1305
26	Wii	1286
28	X360	1234

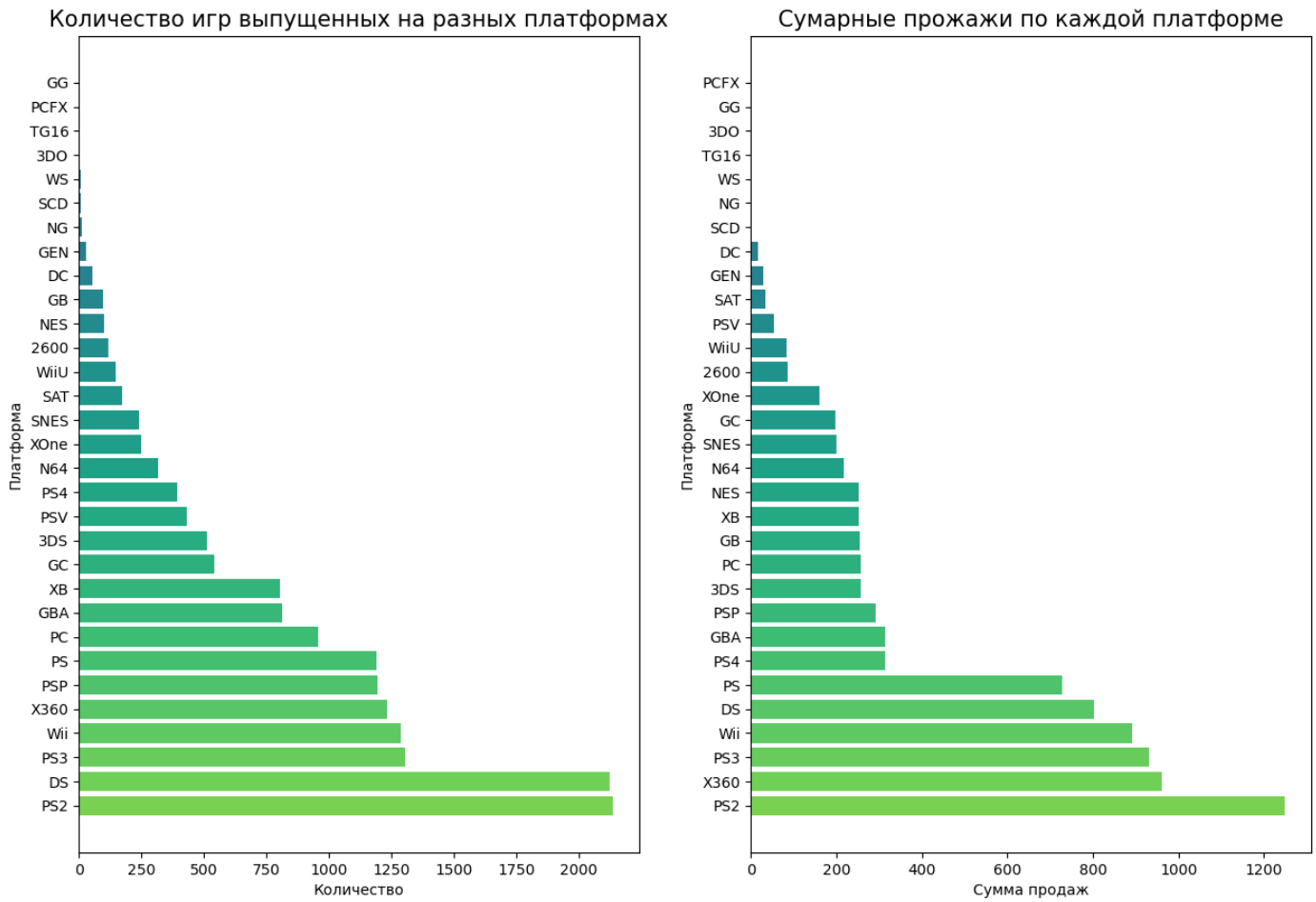
In [34]:
```
#А теперь по суммарным продажам
platforms_sales = (
    games
    .groupby("platform").agg({"sum_sales":"sum"})
    .reset_index()
    .sort_values(by="sum_sales", ascending = False))

platforms_sales.head()
```

Out[34]:

	platform	sum_sales
16	PS2	1248.12
28	X360	961.30
17	PS3	931.33
26	Wii	891.18
4	DS	802.79

In [35]:
```
#Построим графики успешности платформ по выпуску игр и по продажам
fig = plt.figure(figsize=(15, 10))
color = cm.viridis(np.linspace(.8, .20, 38))

#График по выпуску
ax1 = plt.subplot(121)
ax1.barh("platform", "amount", color=color, data=platforms_amount)
ax1.set_ylabel('Платформа')
ax1.set_xlabel('Количество')
ax1.set_title('Количество игр выпущенных на разных платформах')
ax1.title.set_size(15)

#График по продажам
ax2 = plt.subplot(122)
ax2.barh("platform", "sum_sales", color=color, data=platforms_sales)
ax2.set_ylabel('Платформа')
ax2.set_xlabel('Сумма продаж')
ax2.set_title('Сумарные прожажи по каждой платформе')
ax2.title.set_size(15)


plt.show
```

Out[35]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



В целом, в топе по обоим критериям фигурируют одни и те же платформы. Для точности составим смешенный рейтинг, приведя распределения по обеим переменным к стандартному виду.

In [36]:
```
#Стандартизируем распределения количества выпущенных игр
platforms_amount_st = platforms_amount
platforms_amount_st_x = platforms_amount_st["amount"]
platforms_amount_st["amount"] = (platforms_amount_st_x-platforms_amount_st_x.mean ())/pl

#Стандартизируем распределение суммы продаж
platforms_sales_st = platforms_sales
platforms_sales_st_x = platforms_sales_st["sum_sales"]
platforms_sales_st["sum_sales"] = (platforms_sales_st_x-platforms_sales_st_x.mean ())/pl
```

In [37]:
```
#Соеденим два датафрейма и посчитами общий идекс, суммировав стандартизированные значени
platforms = platforms_amount_st.merge(platforms_sales_st)
platforms["index"] = platforms["amount"] + platforms["sum_sales"]
platforms = platforms.query("index > 0").sort_values(by="index", ascending=False)
platforms
```

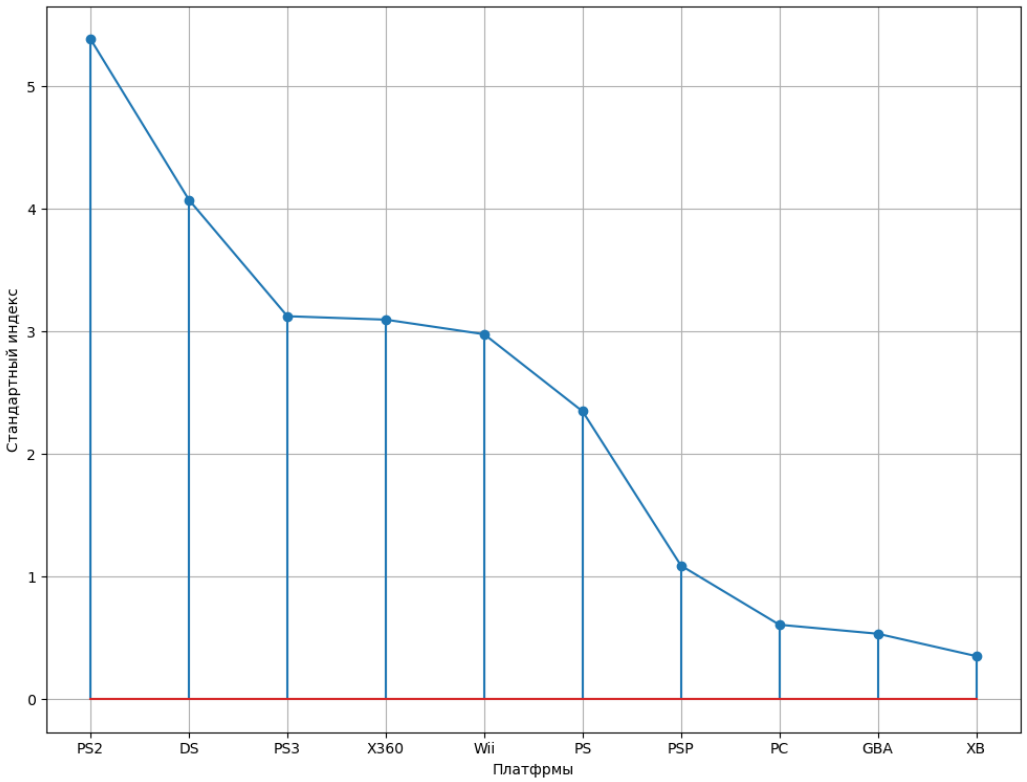
Out[37]:

	platform	amount	sum_sales	index
0	PS2	2.594136	2.789566	5.383702
1	DS	2.573112	1.499868	4.072980
2	PS3	1.251784	1.872126	3.123910
4	X360	1.136956	1.958921	3.095877
3	Wii	1.221056	1.755850	2.976906
6	PS	1.065795	1.282056	2.347852
5	PSP	1.073882	0.015788	1.089670

7	PC	0.690584	-0.083344	0.607240
8	GBA	0.452842	0.081065	0.533907
9	XB	0.444755	-0.093335	0.351420

In [38]:
```
#Посторем круговую диаграмму с отображением плотформ, попавших в топ по суммарному крите
fig = plt.figure(figsize=(12, 9))
ax = fig.add_subplot()
color = cm.plasma(np.linspace(.8, .20, 38))

ax.stem("platform", "index", data=platforms)
ax.plot("platform", "index", data=platforms)
ax.set_ylabel('Стандартный индекс')
ax.set_xlabel('Платфрмы')
ax.grid(True)
ax.set_title('Смешенный рейтинг платформ по количеству выпущенных игр и суммарным продаж
ax.title.set_size(20)
```

Смешенный рейтинг платформ по количеству выпущенных игр и суммарным продажам



Наиболее популярными игровыми платформами являются DS, PS и X360.

Итак, мы отобрали наиболее популярные игровые платформы. В список попали: **'DS', 'PS3', 'Wii', 'X360', 'PSP', 'PC', 'PS2', '3DS', 'PSV', 'PS4'.**

Далее расмотрим, как менялись их продажи и выпуски по годам

In [39]:
```
#Отбираем в массивы с выпусками и продажами те данные, которые касаются выборочных платф
platforms_amount = platforms_amount.loc[platforms_amount["platform"].isin(list(platforms
platforms_sales = platforms_sales.loc[platforms_sales["platform"].isin(list(platforms["p

#Собираем в одну таблицу
platforms_new = platforms_amount.merge(platforms_sales)
```

Отберем в сводную таблицу 1) платформы из полученного списка 2) данные после 1995 года (т.к. на первом графике мы увидели, что до 1995 года рынок был ппрактически не активен)

In [40]:
```
#Производим отбор по указанным критериям
games_slice = games.loc[(games["platform"].isin(list(platforms["platform"]))) & (games["

#Строим сводную таблицу с сумаррными продажами по годам
top_years = pd.pivot_table(games_slice, index="platform", columns="year_of_release", val
#Заполняем пропуски нулями для отображения на графике
top_years = top_years.fillna(0)
top_years
```

Out[40]:

year_of_release	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	...	2007	2008	20
platform														
DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	17.27	...	146.94	145.32	119.
GBA	0.00	0.00	0.00	0.00	0.00	0.07	61.53	74.16	56.67	77.91	...	3.40	0.00	0.
PC	4.22	10.58	11.27	3.26	4.74	4.66	5.47	8.57	8.84	10.39	...	9.28	12.42	16.
PS	35.96	94.70	136.17	169.49	144.53	96.37	35.59	6.67	2.07	0.00	...	0.00	0.00	0.
PS2	0.00	0.00	0.00	0.00	0.00	39.17	166.43	205.38	186.77	220.55	...	75.99	53.90	26.
PS3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	73.19	118.52	130.
PSP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.06	...	46.93	34.56	38.
Wii	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	152.77	171.32	206.
X360	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	95.42	135.26	120.
XB	0.00	0.00	0.00	0.00	0.00	0.99	22.26	48.59	55.40	65.42	...	0.55	0.18	0.

10 rows × 22 columns

In [41]:
```
top_years
```

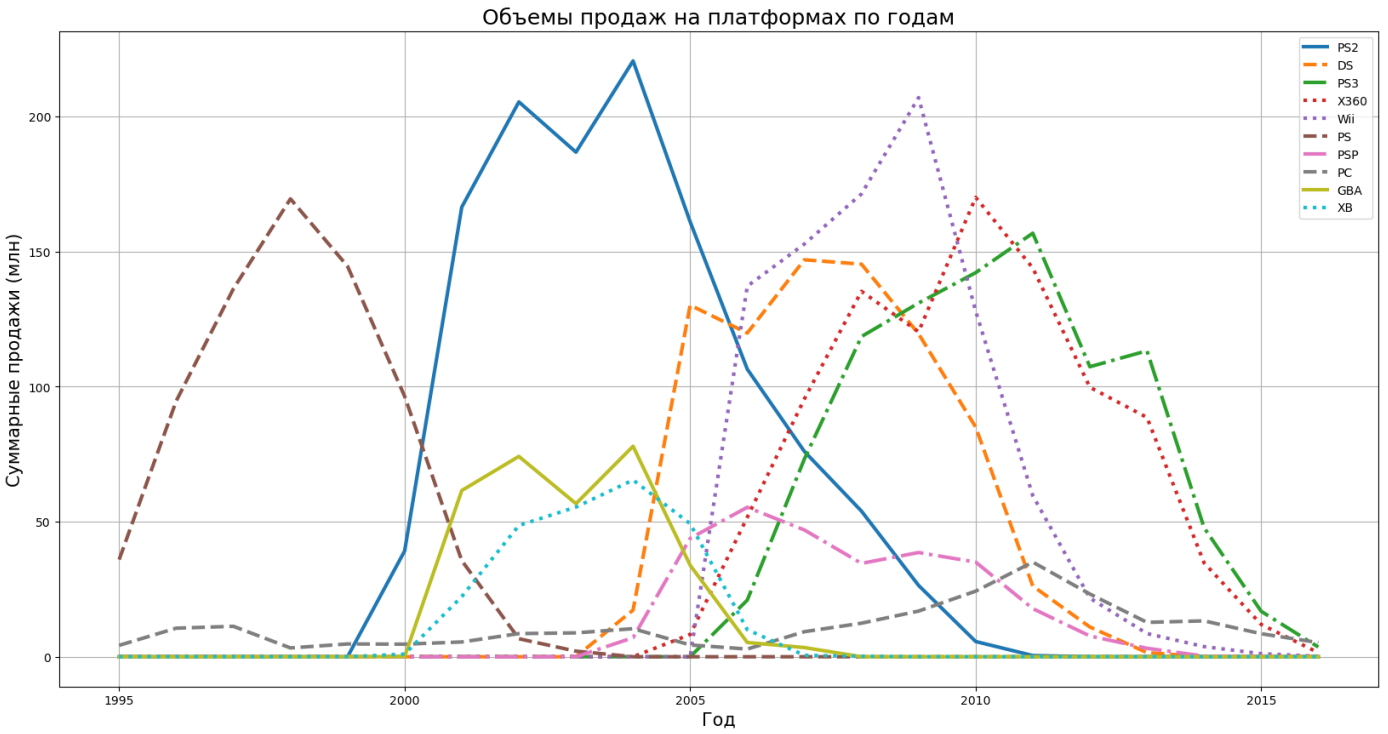
Out[41]:

year_of_release	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	...	2007	2008	20
platform														
DS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	17.27	...	146.94	145.32	119.
GBA	0.00	0.00	0.00	0.00	0.00	0.07	61.53	74.16	56.67	77.91	...	3.40	0.00	0.
PC	4.22	10.58	11.27	3.26	4.74	4.66	5.47	8.57	8.84	10.39	...	9.28	12.42	16.
PS	35.96	94.70	136.17	169.49	144.53	96.37	35.59	6.67	2.07	0.00	...	0.00	0.00	0.
PS2	0.00	0.00	0.00	0.00	0.00	39.17	166.43	205.38	186.77	220.55	...	75.99	53.90	26.
PS3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	73.19	118.52	130.
PSP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.06	...	46.93	34.56	38.
Wii	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	152.77	171.32	206.
X360	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	95.42	135.26	120.
XB	0.00	0.00	0.00	0.00	0.00	0.99	22.26	48.59	55.40	65.42	...	0.55	0.18	0.

10 rows × 22 columns

In [42]:
```
#Посторим график объема продаж по каждой из выбранных  платформ
styles = ["-", "--", "-.", ":", "dotted", "dashed", "dashdot", "--", "solid", ":"]

#Создаем поле для графика
```

```
fig, ax = plt.subplots(figsize=(20, 10))

#Строим графики по каждой из платформ за выделенный период
for style, plat in enumerate(list(platforms["platform"])):
    ax = top_years.loc[plat]
    plt.plot(top_years.columns, ax, label=plat, linestyle=styles[style], linewidth=3)
    plt.title('Объемы продаж на платформах по годам', fontsize=18)
    plt.xlabel('Год', fontsize=15)
    plt.ylabel('Суммарные продажи (млн)', fontsize=15)
    plt.legend()
    plt.grid(True)
```



Итак, по построенному графику мы видим, что объем продаж по платформам имеет парабалическую форму: т.е. объем продаж каждой платформы взлетает на определенное время, а затем платформа устаревает, и продажи падают. По графику видно, что примерный период роста и спада составляет **10 лет.**

Также по диаграмме мы видем, что устаревшие платформы потеряли свою актуальность примерно к 2014 году. Приблизительно в это же время должен начаться период зарожддения новых платформ.

Для актуальнова анализа используется период около 2 лет, при этом период жизни платформы - 10 лет, в связи с чем, возьмем актуальный период в 4-5 лет. Учитывая, что 2016 год не полный, возьмем данные с 2012 года.

In [43]:
```
#Отбрасываем данные до 2007
games_actual = games.loc[games["year_of_release"] >= 2014]
```

Далее повторим раннее прописанный блок кода, только уже на новых данных.

In [44]:
```
#Составим рейтинг платформ по количеству выпущенных игр

platforms_amount = (
    games_actual
    .groupby("platform").agg({"name":"count"})
    .reset_index()
    .sort_values(by="name", ascending = False))
platforms_amount.columns = ["platform", "amount"]
```

```
#А теперь по суммарным продажам
platforms_sales = (
    games_actual
    .groupby("platform").agg({"sum_sales":"sum"})
    .reset_index()
    .sort_values(by="sum_sales", ascending = False))

#Стандартизируем распределения количества выпущенных игр
platforms_amount_st = platforms_amount
platforms_amount_st_x = platforms_amount_st["amount"]
platforms_amount_st["amount"] = (platforms_amount_st_x-platforms_amount_st_x.mean ())/pl

#Стандартизируем распределение суммы продаж
platforms_sales_st = platforms_sales
platforms_sales_st_x = platforms_sales_st["sum_sales"]
platforms_sales_st["sum_sales"] = (platforms_sales_st_x-platforms_sales_st_x.mean ())/pl

#Соеденим два датафрейма и посчитами общий идекс, суммировав стандартизированные значени
platforms_new = platforms_amount_st.merge(platforms_sales_st)
platforms_new["index"] = platforms_new["amount"] + platforms_new["sum_sales"]
platforms_new = platforms_new.loc[:10]
platforms_new
```

Out[44]:

	platform	amount	sum_sales	index
0	PS4	1.731461	2.492085	4.223546
1	PSV	1.054260	-0.585314	0.468946
2	XOne	0.494106	0.780669	1.274775
3	PS3	0.418861	-0.055179	0.363682
4	3DS	0.360338	0.159052	0.519390
5	PC	-0.149653	-0.531466	-0.681120
6	X360	-0.484073	-0.286317	-0.770390
7	WiiU	-0.801773	-0.346996	-1.148769
8	PSP	-1.303403	-0.840538	-2.143941
9	Wii	-1.320124	-0.785996	-2.106120

После удаления 19 лет можно заметить, что полученный актуальный рейтинг значительно изменился, некоторые платформы ушли, а новые - появились. Теперь список состоит из следующих представителей: **'PS4', 'PSV', 'XOne', 'PS3', '3DS', 'PC', 'X360', 'WiiU', 'PSP', 'Wii'**

Рассмотрим успехи данных платформ в производстве и продаже видеоигр на диаграмме размаха

In [45]:
```
#Изменим количество знаков после запятой
pd.set_option("display.precision", 2)

#Производим отбор по указанным критериям
games_slice2 = games_actual.loc[games_actual["platform"].isin(list(platforms_new["platfo

top_years2 = games_slice2[["platform", "sum_sales", "year_of_release"]].set_index("platf
games_slice2
```

Out[45]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score
31	Call of Duty: Black Ops 3	PS4	2015	Shooter	6.03	5.86	0.36	2.38	0.0

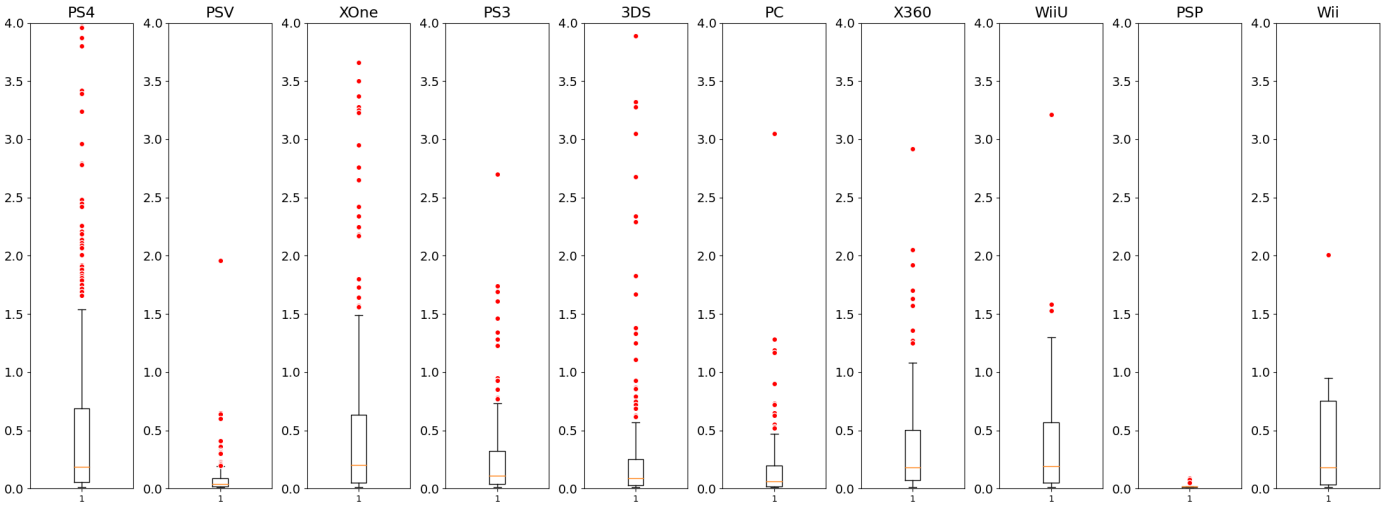
42	Grand Theft Auto V	PS4	2014	Action	3.96	6.31	0.38	1.97	97.0
47	Pokemon Omega Ruby/Pokemon Alpha Sapphire	3DS	2014	Role-Playing	4.35	3.49	3.10	0.74	0.0
77	FIFA 16	PS4	2015	Sports	1.12	6.12	0.06	1.28	82.0
87	Star Wars Battlefront (2015)	PS4	2015	Shooter	2.99	3.49	0.22	1.28	0.0
...	...	...	...	...	...	...	...	...	...
16703	Strawberry Nauts	PSV	2016	Adventure	0.00	0.00	0.01	0.00	0.0
16707	Aiyoku no Eustia	PSV	2014	Misc	0.00	0.00	0.01	0.00	0.0
16710	Samurai Warriors: Sanada Maru	PS3	2016	Action	0.00	0.00	0.01	0.00	0.0
16712	Haitaka no Psychedelica	PSV	2016	Adventure	0.00	0.00	0.01	0.00	0.0
16714	Winning Post 8 2016	PSV	2016	Simulation	0.00	0.00	0.01	0.00	0.0

1689 rows × 13 columns

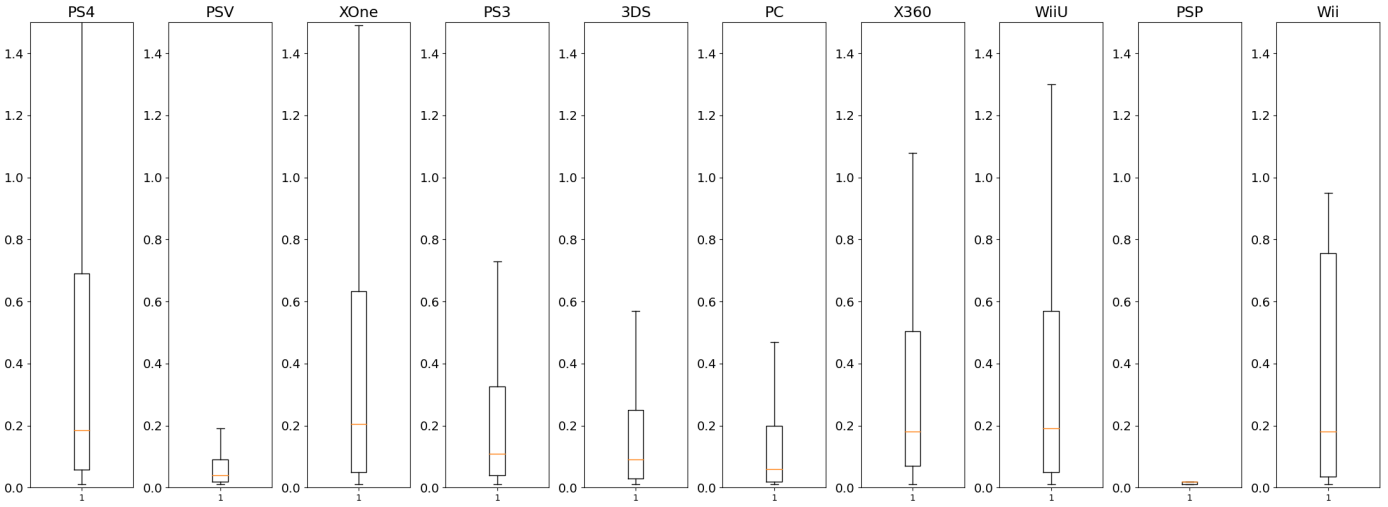
In [46]:
```
#Устанавливаем цвет и маркер для обозначения выбросов
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')

plat = list(platforms_new["platform"])

#Создаем поле для графика
fig, axs = plt.subplots(1, len(list(platforms_new["platform"])), figsize=(22, 9))
fig.suptitle('Диаграммы размаха по продоваемости платформ  с учетом аномалий' + '\n', fo
top_years3 = top_years2['sum_sales']

#Строим графики по каждой из платформ за выделенный период
for i, ax in enumerate(axs.flat):
    ax.boxplot(top_years3.loc[plat[i]], flierprops = red_circle)
    ax.set_title(plat[i], fontsize=17)
    ax.tick_params(axis="y", labelsize=14)
    ax.set_ylim([0, 4])



plt.tight_layout()


#Устанавливаем цвет и маркер для обозначения выбросов
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')

plat = list(platforms_new["platform"])

#Создаем поле для графика
fig, axs = plt.subplots(1, len(list(platforms_new["platform"])), figsize=(22, 9))
fig.suptitle('Диаграммы размаха по продоваемости платформ без учета аномалий' + '\n', fo
top_years3 = top_years2['sum_sales']
```

```
#Строим графики по каждой из платформ за выделенный период
for i, ax in enumerate(axs.flat):
    ax.boxplot(top_years3.loc[plat[i]], flierprops = red_circle, showfliers=False)
    ax.set_title(plat[i], fontsize=17)
    ax.tick_params(axis="y", labelsize=14)
    ax.set_ylim([0, 1.5])



plt.tight_layout()
```

Диаграммы размаха по продоваемости платформ с учетом аномалий

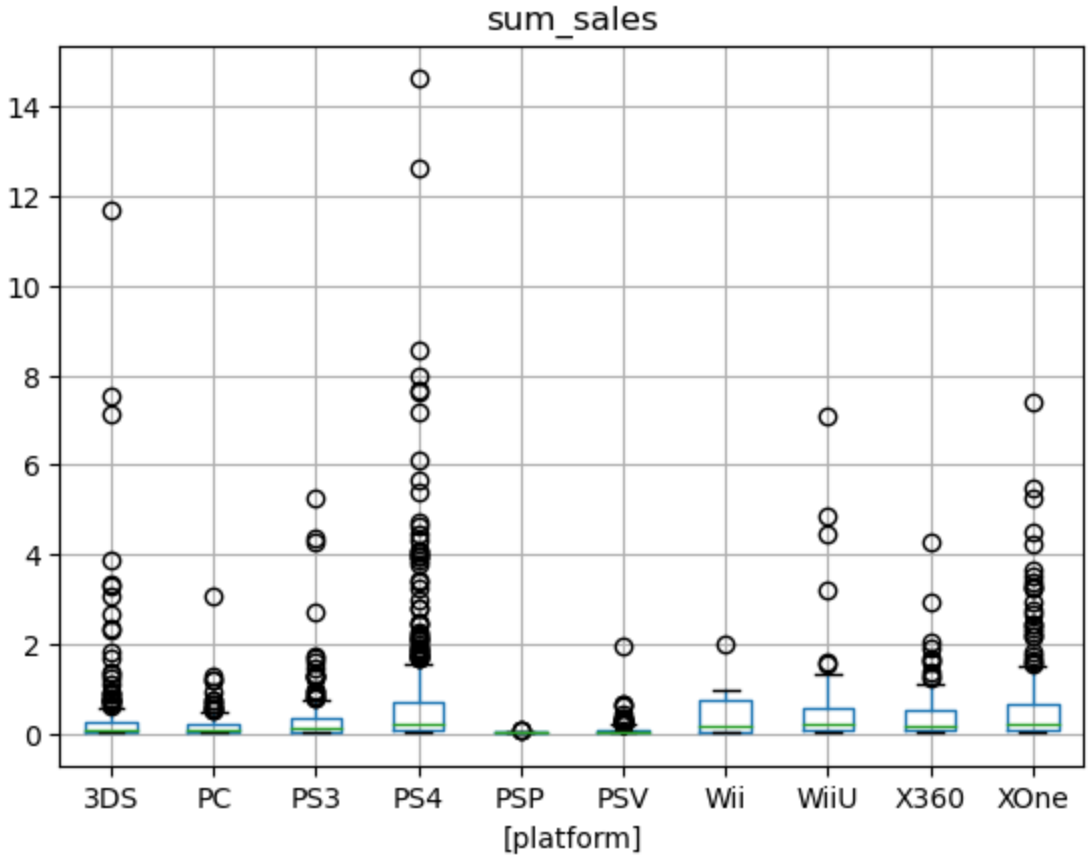


Диаграммы размаха по продоваемости платформ без учета аномалий



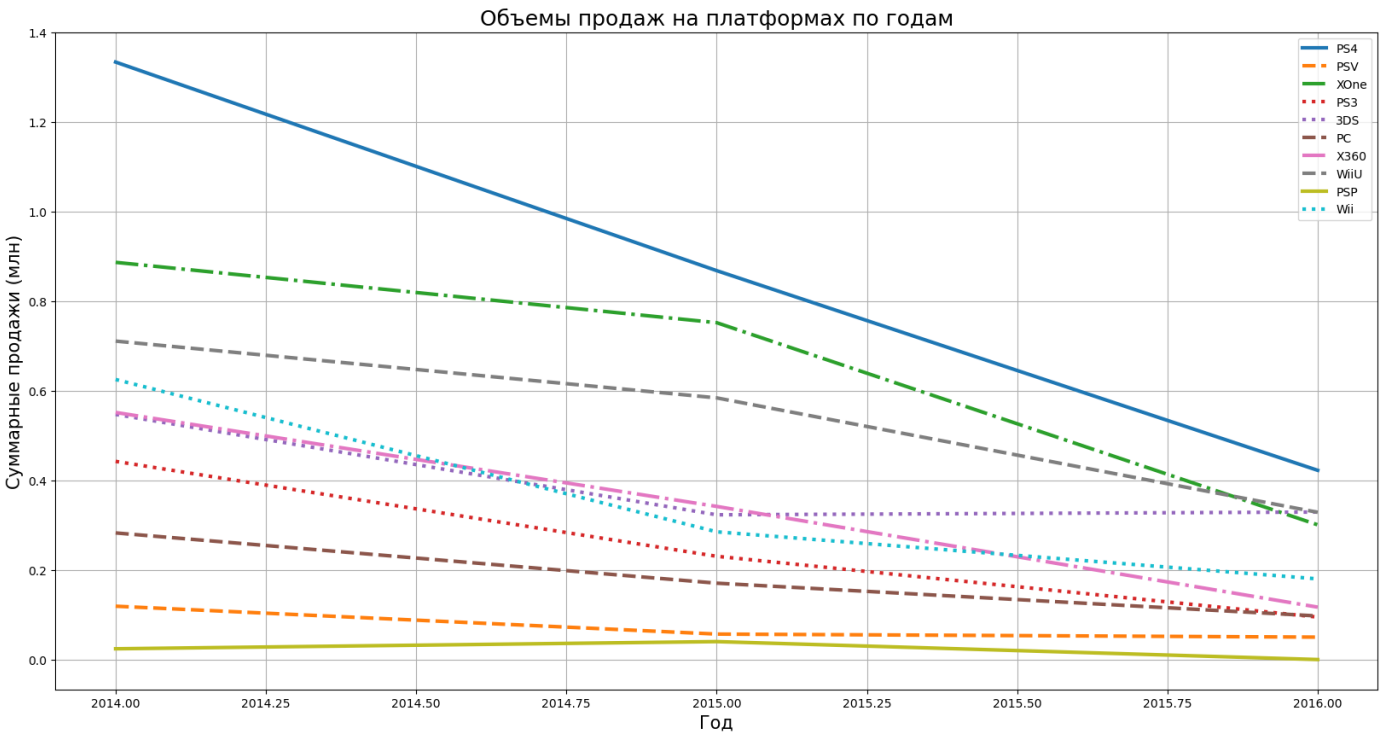
Диаграмм размаха недостаточно для того, чтобы сделать вывод о прогнозе, потому что не известно к какому году относятся наблюдаемые аномалии (например у WiiU). В связи с чем ниже еще раз отобразим линейный график, построенный на новых данных

In [47]:
```
# check
games_actual[['sum_sales','platform']].boxplot(by='platform')
plt.suptitle("");
```



In [48]:
```
top_plat = pd.pivot_table(games_slice2, index="platform", columns="year_of_release", val
#Заполняем пропуски нулями для отображения на графике
top_plat = top_plat.fillna(0)
```

In [49]:
```
styles = ["-", "--", "-.", ":", "dotted", "dashed", "dashdot", "--", "solid", ":", "dott

#Создаем поле для графика
fig, ax = plt.subplots(figsize=(20, 10))

#Строим графики по каждой из платформ за выделенный период
for style, plat in enumerate(list(platforms_new["platform"])):
    ax = top_plat.loc[plat]
    plt.plot(top_plat.columns, ax, label=plat, linestyle=styles[style], linewidth=3)
    plt.title('Объемы продаж на платформах по годам', fontsize=18)
    plt.xlabel('Год', fontsize=15)
    plt.ylabel('Суммарные продажи (млн)', fontsize=15)
    plt.legend()
    plt.grid(True)
```



Таким образом мы видим, что:

1) PC демонстрирует хоть и не самые высокие, но самые стабильные продажи, что тоже позволяет считать его перспективной платформой (так как компьютер не является консолью, которая постоянно выходит в новых версиях)

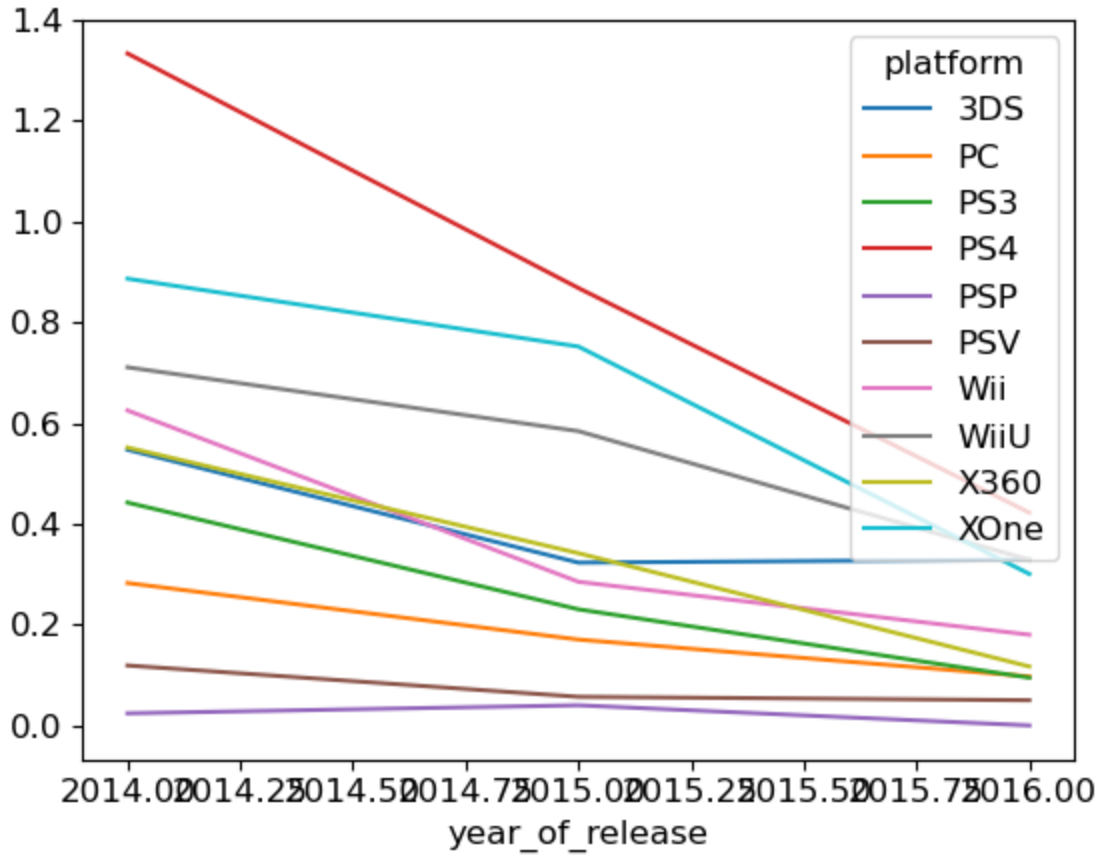
2) WiiU также держиться в середине рейтинга по продажам, мене стабильно чем ПК, но более прибыльно, что делает ее перспективной на 2017 год.

3) Платформа 3DS хоть и демонстрирует явный спад, еще не до конца потеряла свою актуальность, всвязи с чем е тоже можно считать прибыльной на 2017 год.

4) Что касается платформ, рост которых начался недавно, по диаграмме размаха они имеют довольно высокую медиану, и достаточно большой размах вариации, что говорит о недавнем начале роста. К их числу мы можем отнести PS4 и XOne - они потенциально являются самыми перспективными на грядущий год

In [102…
```
top_plat.T.plot(kind='line')
```

Out[102]:
```
<Axes: xlabel='year_of_release'>
```



Анализ влияния отзывов на продажи видеоигр

Рассмотрим далее как влияют отзывы пользователей и критиков на продажи видеоигр.
Исследовательская гипотеза: рейтинги пользователей и критиков оказывают вдияние на объем продаж

Сделаем оценку на примере одной платформы. В качестве таковой возьмем PC, как наиболее стабильную.

In [51]:
```
#Сделаем срез данных по выбранной платформе
games_pc = games.loc[(games["platform"] == "PC") & (games["no_scored"] == 0)]
games_pc.head()
```

Out[51]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_sco
85	The Sims 3	PC	2009	Simulation	0.99	6.42	0.00	0.60	86.0	7
138	World of Warcraft	PC	2004	Role-Playing	0.08	6.21	0.00	0.00	93.0	7
192	Diablo III	PC	2012	Role-Playing	2.44	2.16	0.00	0.54	88.0	4
218	StarCraft II: Wings of Liberty	PC	2010	Strategy	2.57	1.68	0.00	0.58	93.0	8
284	Half-Life	PC	1997	Shooter	4.03	0.00	0.09	0.00	96.0	9

In [52]:
```
#Строим диаграмму рассеяния
fig, ax = plt.subplots(figsize=(20, 7))
```

```
ax.scatter("user_score", "sum_sales", data = games_pc)
plt.title('Зависимость объемы продаж от рейтинга поьльзователей', fontsize=18)
plt.xlabel('Рейтинг', fontsize=15)
plt.ylabel('Суммарные продажи (млн)', fontsize=15)
```

Out[52]:
```
Text(0, 0.5, 'Суммарные продажи (млн)')
```

In [53]:
```
#Определяем корреляцию между текущеми переменными
round(games_pc['user_score'].corr(games_pc['sum_sales']), 2)
```

Out[53]:
```
0.01
```

Корреляция слабая отсутствует: значение близко к нулю. Рассмотрим, что касается оценок критиков

In [54]:
```
#Строим диаграмму рассеяния
fig, ax = plt.subplots(figsize=(20, 7))

ax.scatter("critic_score", "sum_sales", data = games_pc)
plt.title('Зависимость объемы продаж от рейтинга критиков', fontsize=18)
plt.xlabel('Рейтинг', fontsize=15)
plt.ylabel('Суммарные продажи (млн)', fontsize=15)
```

Out[54]:
```
Text(0, 0.5, 'Суммарные продажи (млн)')
```

In [55]:
```
#Определяем корреляцию между текущеми переменными
round(games_pc['critic_score'].corr(games_pc['sum_sales']), 2)
```

Out[55]:
```
0.26
```

Корреляция продаж с оценками критиков выше. Ее можно назвать умеренной. В данном случае, касаемо ПК игр, оценки критиков имеют значение для продаж, но не однозначное.

Сравним как коррелируют данный показатели на других платформах

In [56]:
```
#Выведем топы платформ по выпускам игр и продажам
amount = games.loc[games["platform"].isin(list(platforms_new["platform"]))].groupby("pla
print(amount)

sales = games.loc[games["platform"].isin(list(platforms_new["platform"]))].groupby("plat
print(sales)
```

```
platform
PS3     1305
Wii     1286
X360    1234
PSP     1195
PC       958
3DS      512
PSV      429
PS4      392
XOne     247
WiiU     147
Name: name, dtype: int64
platform
X360    961.30
PS3     931.33
Wii     891.18
PS4     314.14
PSP     290.34
3DS     257.81
PC      256.11
XOne    159.32
WiiU     82.19
PSV      53.81
Name: sum_sales, dtype: float64
```

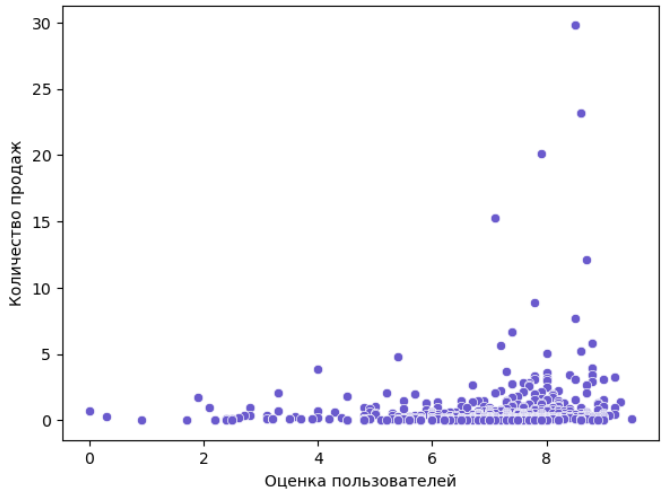
Отберем платформы, находящтеся в топе по обоим значениям. Берем в выборку следующие платформы: DS, PS3, Wii, X360, PS4, 3DS.

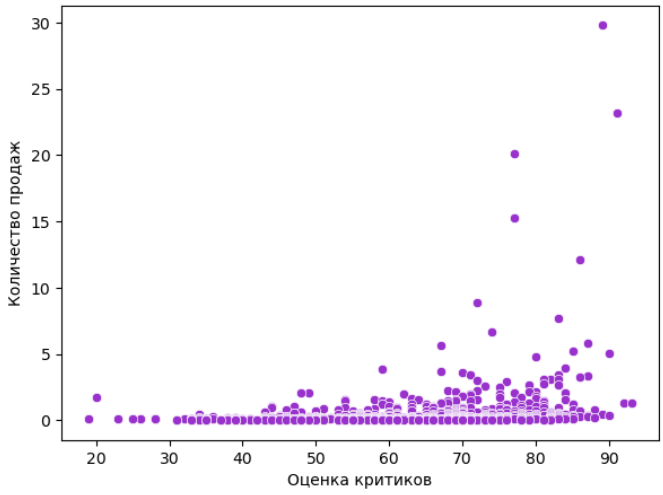
In [57]:
```
#Напишем функции для:

#Вывода двух диаграмм рассеяния для взаимосвязи пользовательского рейтинга и рейтинга кр
def platform_rating_plot(platform_name):
    platform = games.loc[(games['platform']==platform_name) & (games["no_scored"] == 0)]
    fig, ax = plt.subplots(1 ,2, figsize=(15,5))
    sns.scatterplot(x='user_score', y='sum_sales', data=platform, ax=ax[0], color = "sla
    sns.scatterplot(x='critic_score', y='sum_sales', data=platform, ax=ax[1], color = "d
    fig.suptitle(platform_name, fontsize=15)
    ax[0].set(xlabel='Оценка пользователей')
    ax[1].set(xlabel='Оценка критиков')
    ax[0].set(ylabel='Количество продаж')
    ax[1].set(ylabel='Количество продаж')
    plt.show()

#Вывода и сохранения корреляции между пользовательским рейтингом и продажами
def check_correlation_users(platform_name, flag):
    platform = games.loc[(games['platform']==platform_name) & (games["no_scored"] == 0)]
    if flag == 1:
        print("Корреляция продаж с пользовательским рейтингом:", round(platform["user_sc
    if flag == 0:
        return platform["user_score"].corr(platform['sum_sales'])

#Вывода и сохранения корреляции между рейтингом критиков и продажами
```

```
def check_correlation_critics(platform_name, flag):
    platform = games.loc[(games['platform']==platform_name) & (games["no_scored"] == 0)]
    if flag == 1:
        print("Корреляция продаж с рейтингом критиков:", round(platform["critic_score"].
    if flag == 0:
        return platform["critic_score"].corr(platform['sum_sales'])
```

In [58]:
```
#Создадим списки для сохранения корреляций
user_corr = []
critic_corr = []

#Создадим переменную с выборочными элементами
score = ["DS", "PS3", "Wii", "X360", "PS4", "3DS"]

#Для каждого элемента построим графики, выведем и сохраним корреляцию
for element in score:
    platform_rating_plot(element)
    check_correlation_users(element, 1)
    check_correlation_critics(element, 1)

    user_corr.append(check_correlation_users(element, 0))
    critic_corr.append(check_correlation_critics(element, 0))
```

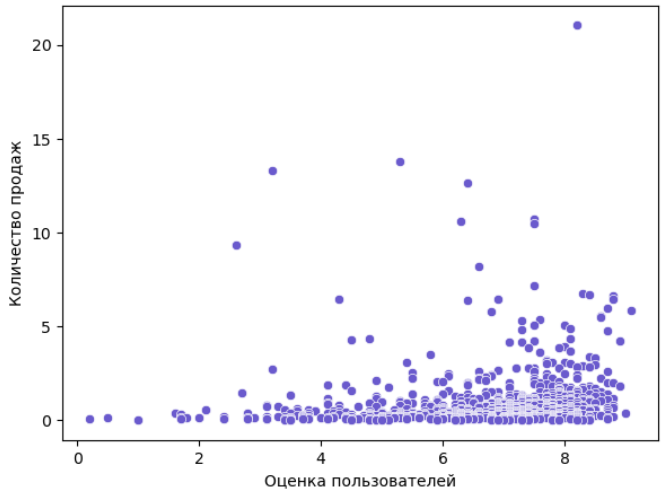
DS

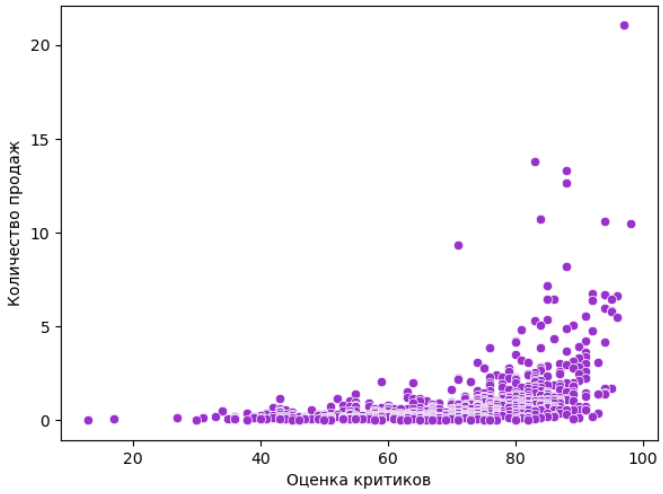





```
Корреляция продаж с пользовательским рейтингом: 0.13
Корреляция продаж с рейтингом критиков: 0.24
```

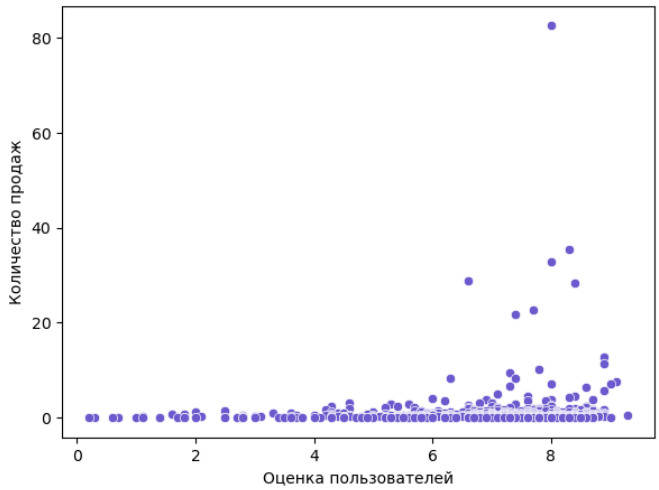
PS3

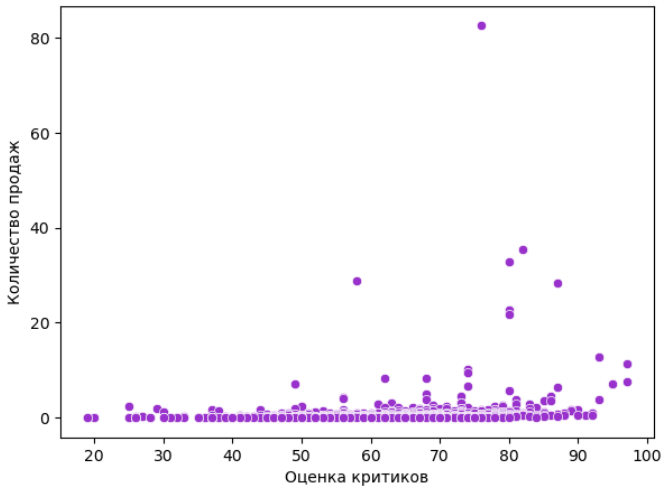





```
Корреляция продаж с пользовательским рейтингом: 0.13
Корреляция продаж с рейтингом критиков: 0.43
```

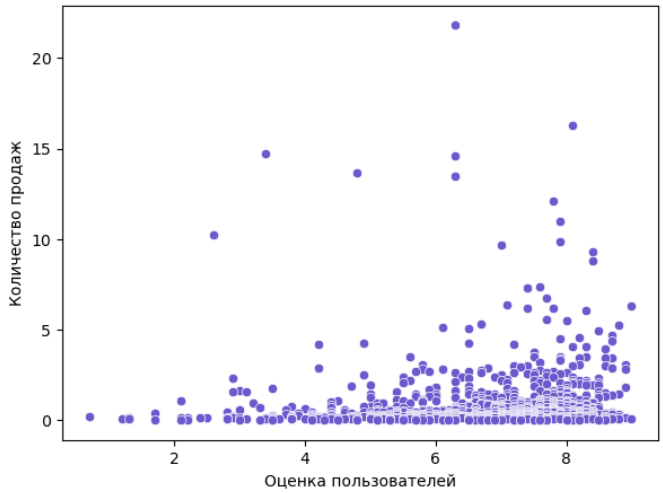
Wii

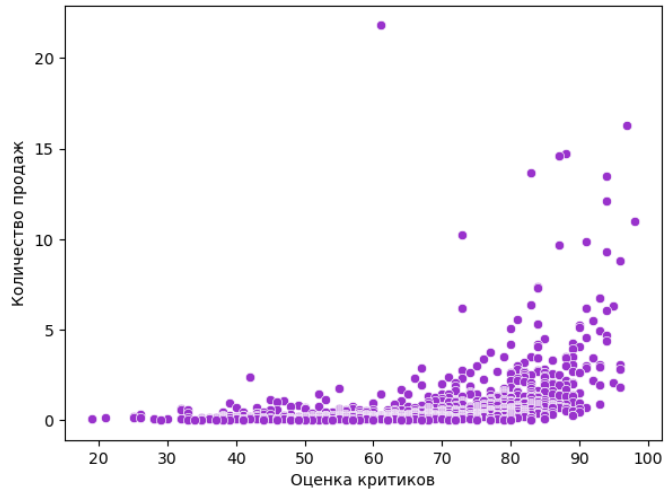




Корреляция продаж с пользовательским рейтингом: 0.11
Корреляция продаж с рейтингом критиков: 0.18

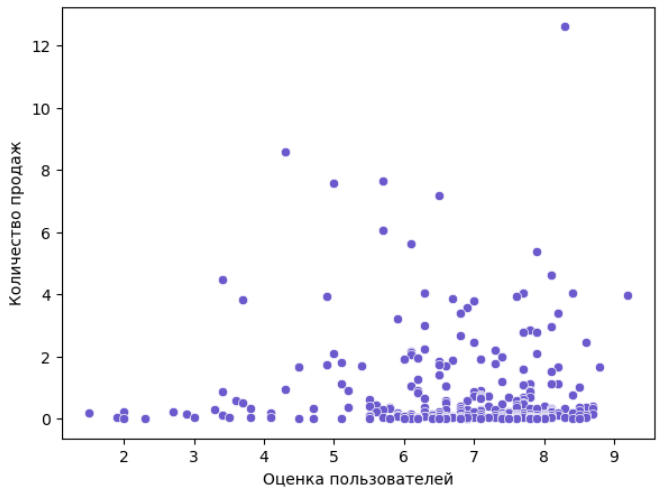
X360

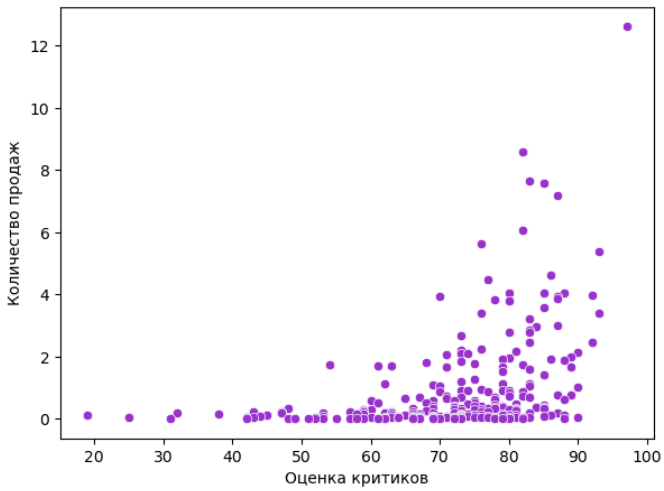




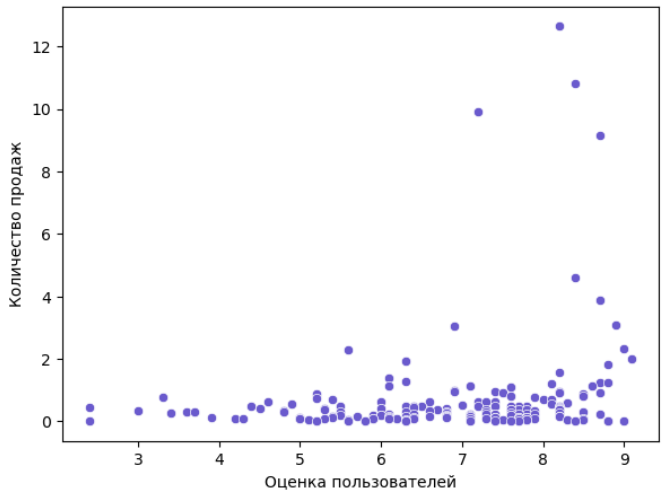
Корреляция продаж с пользовательским рейтингом: 0.11
Корреляция продаж с рейтингом критиков: 0.39

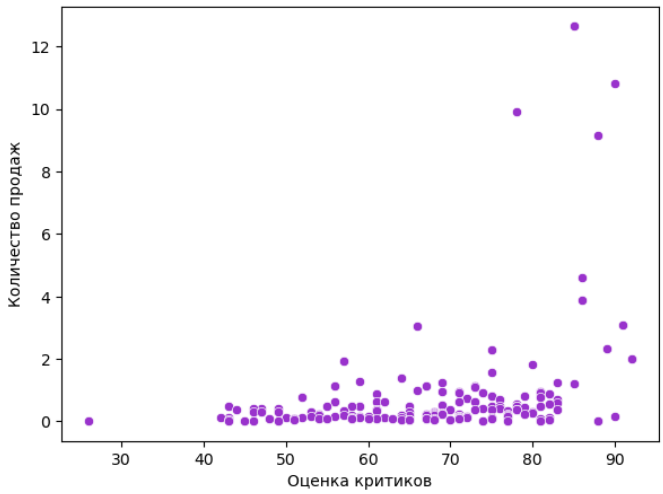
PS4





Корреляция продаж с пользовательским рейтингом: -0.03
Корреляция продаж с рейтингом критиков: 0.41



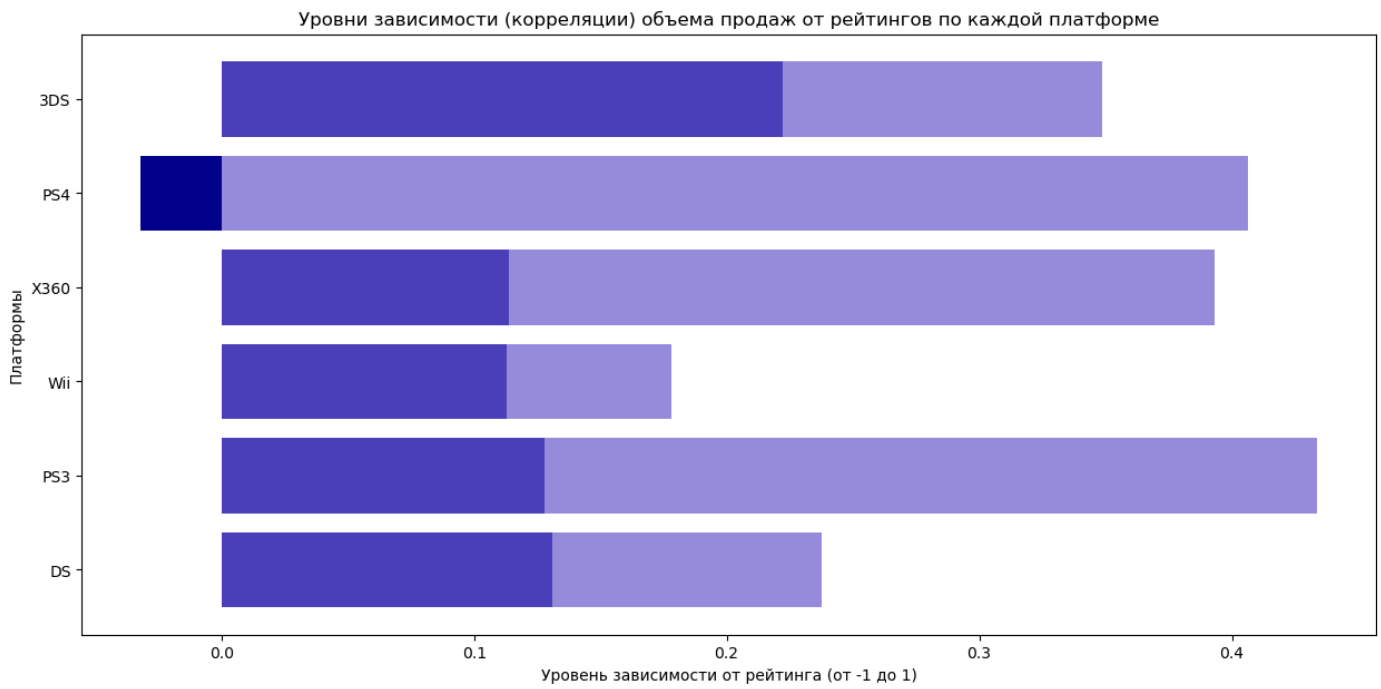



```
Корреляция продаж с пользовательским рейтингом: 0.22
Корреляция продаж с рейтингом критиков: 0.35
```

In [59]:
```
#Выведем на столбчатой диаграмме значения корреляций по каждому элементу для обеих завис

fig, ax = plt.subplots(figsize=(15, 7))
plt.barh(score, user_corr, color="darkblue")
plt.barh(score, critic_corr, alpha = 0.7, color="slateblue")

#А также выведем среднее и стандартное отклонение между корреляциями в группе отдельной
print("Среднее влияние пользовательского рейтинга на продажи:", pd.Series(user_corr).mea
print("Стандартное отклонение влияния пользовательского рейтинга на продажи:", pd.Series
print()
print("Среднее влияние рейтинга критиков на продажи:", pd.Series(critic_corr).mean())
print("Стандартное отклонение влияния рейтинга критиков на продажи:", pd.Series(critic_c

ax.set_ylabel('Платформы')
ax.set_xlabel('Уровень зависимости от рейтинга (от -1 до 1)')
ax.set_title('Уровни зависимости (корреляции) объема продаж от рейтингов по каждой платф
```

```
Среднее влияние пользовательского рейтинга на продажи: 0.11259467950039768
Стандартное отклонение влияния пользовательского рейтинга на продажи: 0.0817957357094608
6

Среднее влияние рейтинга критиков на продажи: 0.3329371514138369
Стандартное отклонение влияния рейтинга критиков на продажи: 0.10255359242893396
```

Out[59]:
```
Text(0.5, 1.0, 'Уровни зависимости (корреляции) объема продаж от рейтингов по каждой пла
тформе')
```



Вывод: В среднем по платформам действительно наблюдается определенная корреляция по одоим рейтингам. Однако все же, рассматривая первую платформу (ПК) мы обнаружили отрицательную корреляцию между пользовательским рейтингом и объемом продаж, что совершенно противоречило гипотезе. Далее же, в среднем корреляция оказалась положительной, и все же значения 0.18 недостаточно, чтобы говорить о сильной взаимосвязи. Вероятно, она есть, но не рекомендую ориентироваться на данный критерий.

Что касается рейтинга пользователей, в среднем также наблюдается умеренная корреляция, однако практически по каждой платформе она выше. В связи с чем можно сказать, что рейтинг критиков все таки в определенной мере влияет на продоваемость видеоигр.

Прибыльность и популярность жанров

Далее составим рейтинг наиболее выпускаемых и наиболее продоваемых жанров видеоигр

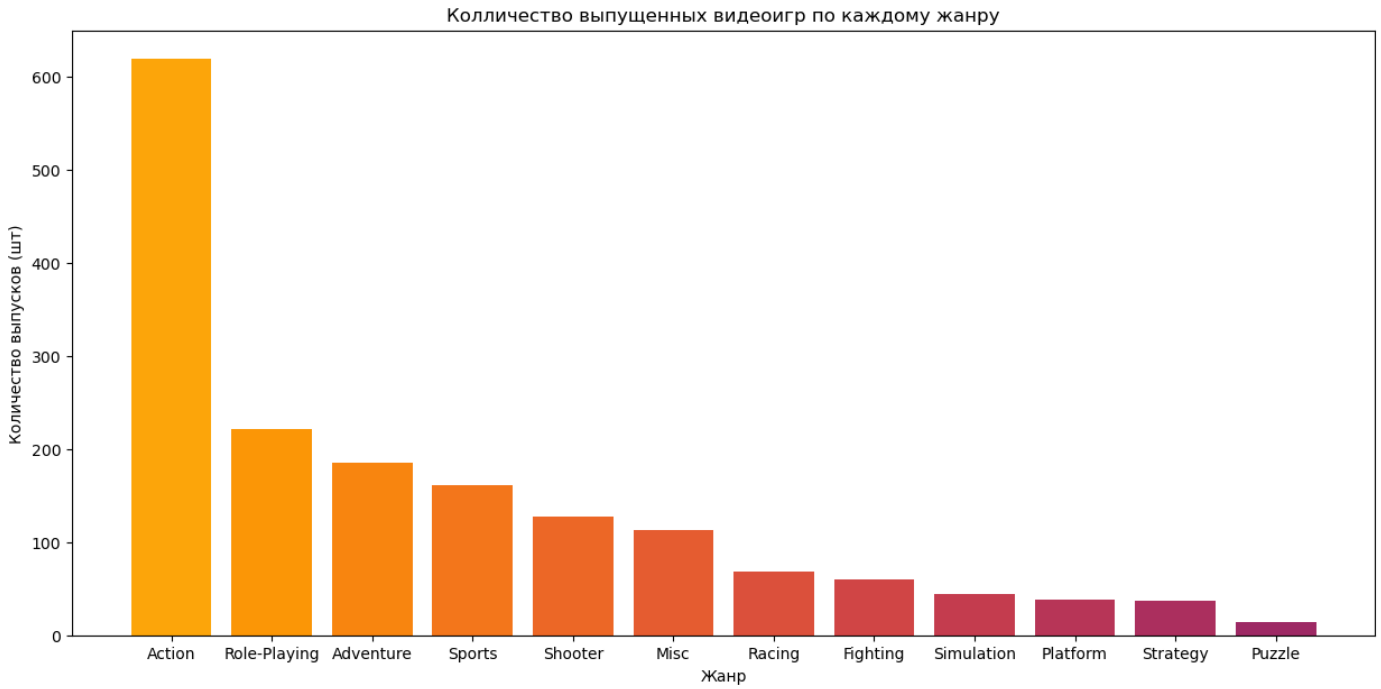
In [60]:
```
#Составим свожную таблицу, содержащую информацию о количестве релизов внутри каждого жан
top_release_genres = (
    pd.pivot_table(games_actual, index = "genre", aggfunc="count")
    .reset_index()
    .sort_values(by="name", ascending=False)
)

top_release_genres.loc[:, ["genre", "name"]]
```

Out[60]:

	genre	name
0	Action	619
7	Role-Playing	221
1	Adventure	185
10	Sports	161
8	Shooter	128
3	Misc	113
6	Racing	69

2	Fighting	60
9	Simulation	44
4	Platform	38
11	Strategy	37
5	Puzzle	14

In [61]:
```
#По предыдущей таблице выведем столбчатую диаграмму
fig, ax = plt.subplots(figsize=(15, 7))
color = cm.inferno(np.linspace(.8, .15, 20))

plt.bar("genre", "name", data=top_release_genres, color=color)
plt.xlabel("Жанр")
plt.ylabel("Количество выпусков (шт)")
plt.title("Колличество выпущенных видеоигр по каждому жанру")

plt.show ()
```



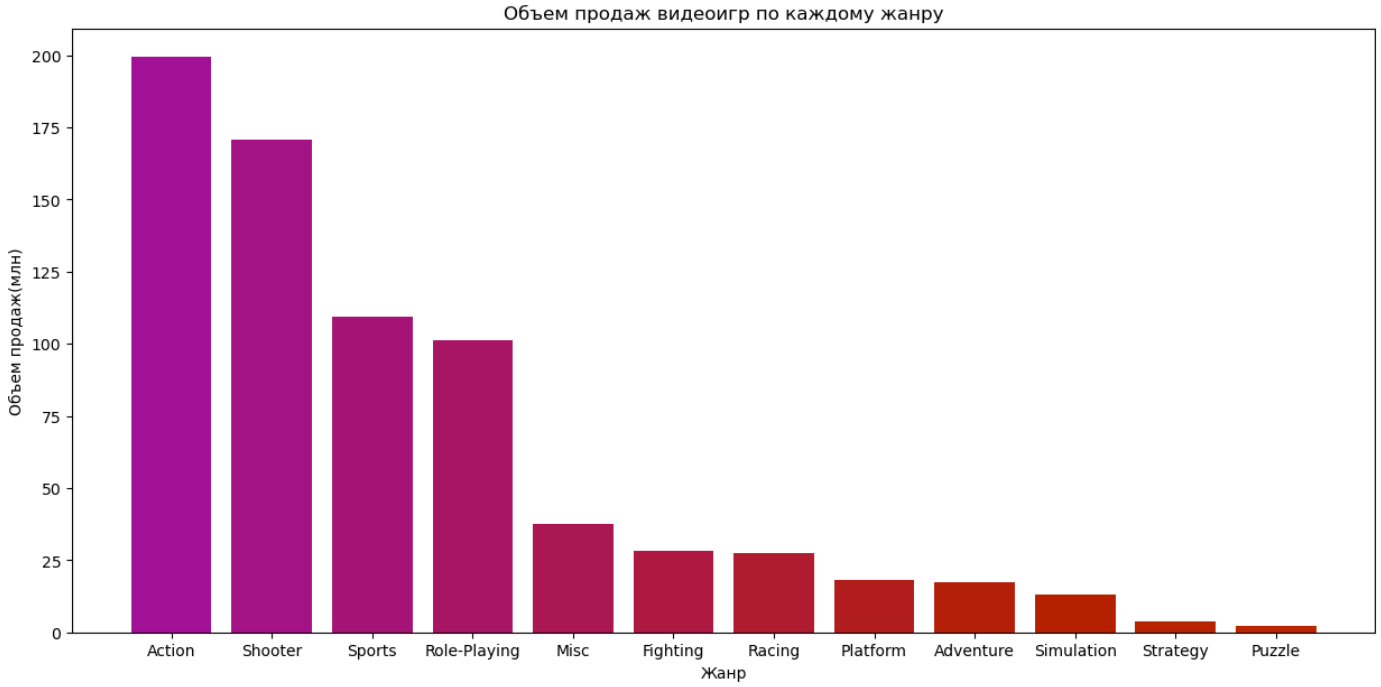
In [62]:
```
#Аналогично составим сводную таблицу с продоваемостью внутри каждого жанра
top_sales_genres = (
    pd.pivot_table(games_actual, index = "genre", values = "sum_sales", aggfunc="sum")
    .reset_index()
    .sort_values(by="sum_sales", ascending=False)
)

top_sales_genres
```

Out[62]:

	genre	sum_sales
0	Action	199.36
8	Shooter	170.94
10	Sports	109.48
7	Role-Playing	101.44
3	Misc	37.55
2	Fighting	28.22

6	Racing	27.52
4	Platform	18.09
1	Adventure	17.55
9	Simulation	13.13
11	Strategy	3.96
5	Puzzle	2.21

In [63]:
```
#И выведем столбчатую диаграмму по данной таблице
fig, ax = plt.subplots(figsize=(15, 7))
color = cm.gnuplot(np.linspace(.4, .8, 35))

plt.bar("genre", "sum_sales", data=top_sales_genres, color=color)
plt.xlabel("Жанр")
plt.ylabel("Объем продаж(млн)")
plt.title("Объем продаж видеоигр по каждому жанру")

plt.show()
```

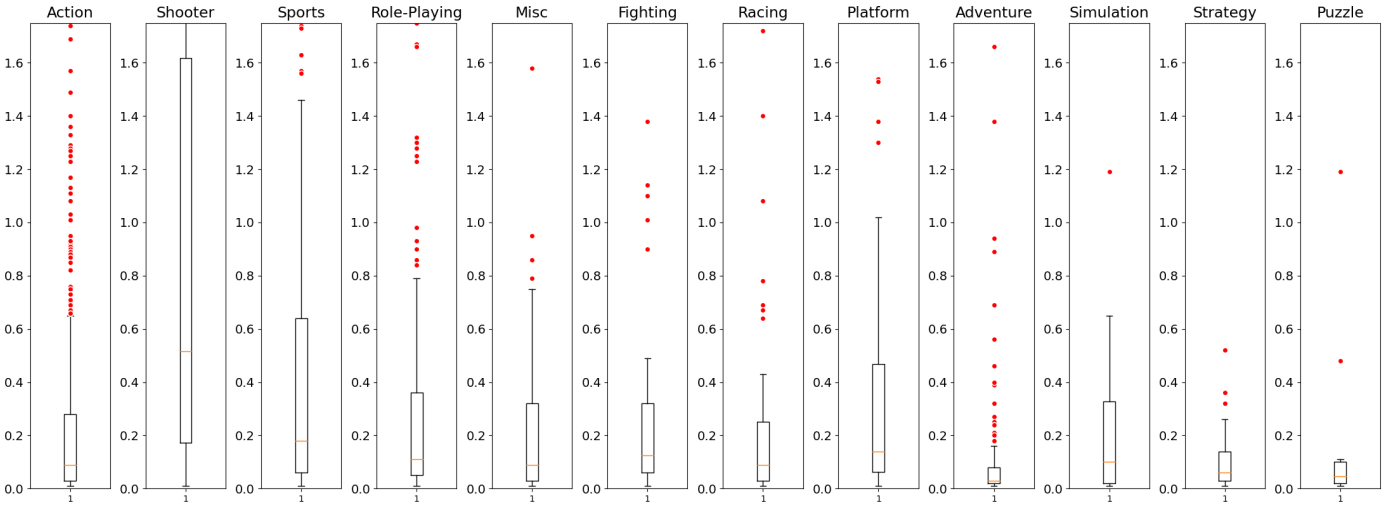


In [64]:
```
#Составим табллицу с данными по продажам для построения диаграммы размаха
box_sales = games_actual.loc[games_actual["genre"].isin(list(top_sales_genres["genre"]))
box_sales = box_sales[["genre", "sum_sales"]]
box_sales = box_sales.set_index('genre')
```

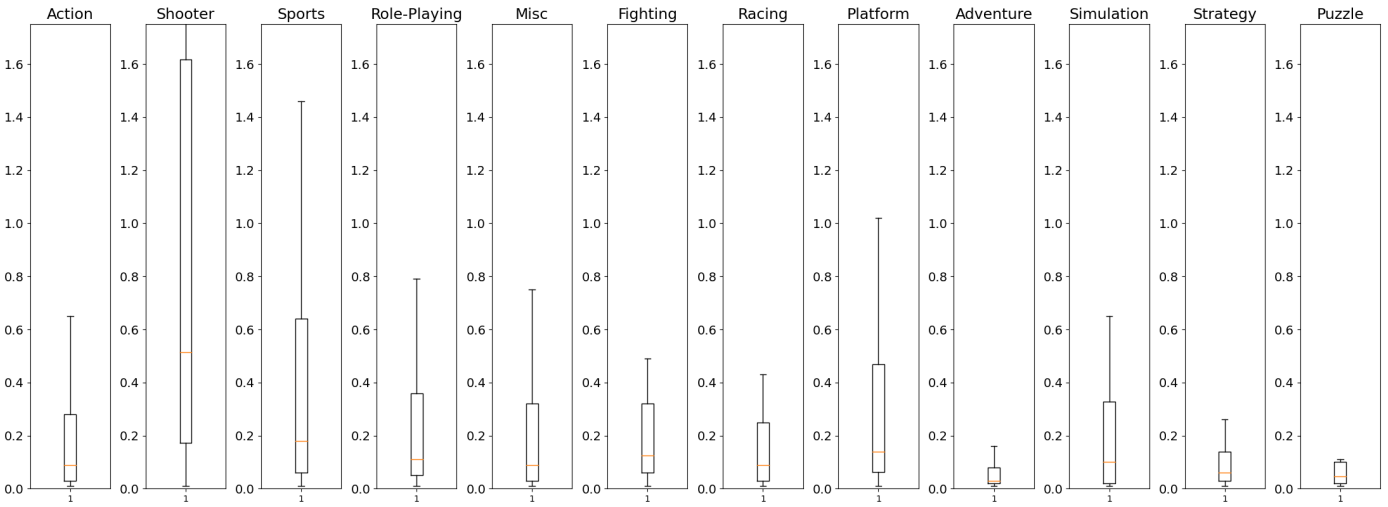
In [65]:
```
#Устанавливаем цвет и маркер для обозначения выбросов
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')


plat = list(top_sales_genres["genre"])

#Создаем поле для графика
fig, axs = plt.subplots(1, len(list(top_sales_genres["genre"])), figsize=(22, 9))
fig.suptitle('Диаграммы размаха по продоваемости жанров с учетом аномалий' + '\n', fonts

#Строим графики по каждой из платформ за выделенный период
for i, ax in enumerate(axs.flat):
    ax.boxplot(box_sales.loc[plat[i]], flierprops = red_circle, data=box_sales)
    ax.set_title(plat[i], fontsize=17)
```

```
    ax.tick_params(axis="y", labelsize=14)
    ax.set_ylim([0, 1.75])

plt.tight_layout()



fig, axs = plt.subplots(1, len(list(top_sales_genres["genre"])), figsize=(22, 9))
fig.suptitle('Диаграммы размаха по продоваемости жанров без учета аномалий' + '\n', font

#Строим графики по каждой из платформ за выделенный период
for i, ax in enumerate(axs.flat):
    ax.boxplot(box_sales.loc[plat[i]], flierprops = red_circle, data=box_sales, showflie
    ax.set_title(plat[i], fontsize=17)
    ax.tick_params(axis="y", labelsize=14)
    ax.set_ylim([0, 1.75])


plt.tight_layout()
```

Диаграммы размаха по продоваемости жанров с учетом аномалий



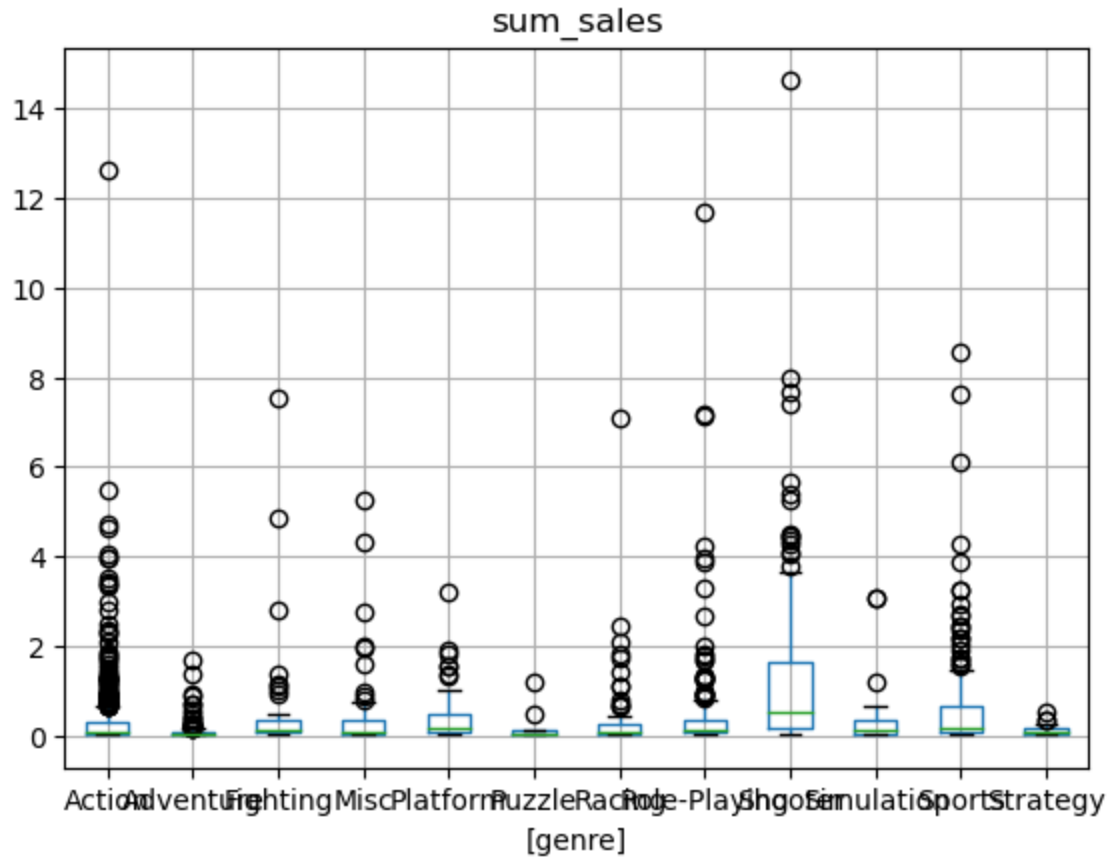
Диаграммы размаха по продоваемости жанров без учета аномалий



Итог: Таким образом, среди жанров по обоим критериям побеждают шутеры - они будут самыми выпускаемыми и продоваемыми жанрами. Также, достаточную актуальность (по медианным продажам) имеют Спортивные игры. Достаточно востребованы (но судя по аномалиям, менее стабильны) Экшены.

In [66]:
```
# check
# график без оформления ...
```

```
games_actual[['sum_sales','genre']].boxplot(by='genre')
plt.suptitle("");
```



Таким образом в рамках данного блока удалось составить рейтинг самых выпускаемых и продоваемых жанров, в которых верхушку занимают шутеры и аркады, будучи самыми прибыльными, на второй ступени спортивные и ролевые игры. Также выпускаются миски и платформы, но их продоваемость на рынке куда менее стабильна и высока.

Портрет среднестатистического пользователя видеоигр по регионам

Топ платформ по продажам в трех регионах

In [67]:
```
north_america = (
    games_actual.groupby("platform")
    .agg({"na_sales":"sum"})
    .sort_values(by="na_sales", ascending=False)
    .reset_index(0))

display(north_america)

#А  также сгруппируем таблтцу, прревратив рейтинг в топ-5
north_america.loc[north_america.index > 4, 'platform'] = "other"
display(north_america)
north_america = pd.pivot_table(north_america, index = "platform").reset_index(0)
display(north_america)
```

	platform	na_sales
0	PS4	98.61
1	XOne	81.27
2	X360	28.30

3	3DS	22.64
4	PS3	22.05
5	WiiU	19.36
6	PC	7.23
7	PSV	2.52
8	Wii	2.08
9	PSP	0.00

	platform	na_sales
0	PS4	98.61
1	XOne	81.27
2	X360	28.30
3	3DS	22.64
4	PS3	22.05
5	other	19.36
6	other	7.23
7	other	2.52
8	other	2.08
9	other	0.00

	platform	na_sales
0	3DS	22.64
1	PS3	22.05
2	PS4	98.61
3	X360	28.30
4	XOne	81.27
5	other	6.24

In [68]:
```
#Сделаем срезы данных из основного массива, которые будут содержать информацию только по
#по каждой платформе
north_america = (
    games_actual.groupby("platform")
    .agg({"na_sales":"sum"})
    .sort_values(by="na_sales", ascending=False)
    .reset_index(0))

#А  также сгруппируем таблтцу, прревратив рейтинг в топ-5
north_america.loc[north_america.index > 4, 'platform'] = "other"
north_america = pd.pivot_table(north_america, index = "platform", aggfunc="sum").reset_i

#Далее аналогично

europe = (
    games_actual.groupby("platform")
    .agg({"eu_sales":"sum"})
    .sort_values(by="eu_sales", ascending=False)
    .reset_index(0))
```

```
europe.loc[europe.index > 4, 'platform'] = "other"
europe = pd.pivot_table(europe, index = "platform", aggfunc="sum").reset_index(0)

japan = (
    games_actual
    .groupby("platform")
    .agg({"jp_sales":"sum"})
    .sort_values(by="jp_sales", ascending=False)
    .reset_index(0))

japan.loc[japan.index > 4, 'platform'] = "other"
japan = pd.pivot_table(japan, index = "platform", aggfunc="sum").reset_index(0)
```

In [69]:

```
#Выведим три графика распределения продаж между платформами по каждому региону

#Задаем цвета и массивы
regions = [north_america, europe, japan]
reg_data = ["na_sales", "eu_sales", "jp_sales"]
colours = ("darkblue", "crimson", "indigo", "violet", "darkorchid", "slateblue")
names = ["Северная Америка", "Европа", "Япония"]


fig, axs = plt.subplots(1, 3, figsize=(25, 7))
fig.suptitle('Портрет пользователей регионов по продажам на наиболее популярных платформ

for i, ax in enumerate(axs.flat):
    ax.pie(reg_data[i], labels=list(regions[i]["platform"]), autopct='%1.1f%%', colors =
    ax.set_ylabel('Продажи')
    ax.set_xlabel('Платформа')
    ax.set_title(names[i])
    ax.title.set_size(15)

plt.show()
```

- В целом, продажи в Северной Америке выше, чем в Европе. А в Европе выше чем в Японии
- Наиболее популярные (по продажам) платформы:

```
- в Америке:  PS4 - 35%, X360 - 29%, Xone - 10%
- в Европе: PS4 - 53%, XOne - 19%, PS3 - 11%,
- в Японии 3DS - 48%, PS4 - 16%, PSV - 15%
```

Так, можно заметить, что списки Европы и Америки похожи, поскольку это регионы со схожей культурой и, соответственно, со схожим рынком. Япония как страна азиатского происхождения имеет свой рынок, на котороп PS и Xbox в первую очередь заменяет 3DS

Топ жанров по продажам в трех регионах

In [70]:
```
#Сделаем срезы данных из основного массива, которые будут содержать информацию только по
#по каждому жанру
north_america_genre = (
    games_actual.groupby("genre")
    .agg({"na_sales":"sum"})
    .sort_values(by="na_sales", ascending=False)
    .reset_index(0))

north_america_genre.loc[north_america_genre.index > 4, 'genre'] = "other"
north_america_genre = pd.pivot_table(north_america_genre, index = "genre", aggfunc="sum"

europe_genre = (
    games_actual.groupby("genre")
    .agg({"eu_sales":"sum"})
    .sort_values(by="eu_sales", ascending=False)
    .reset_index(0))

europe_genre.loc[europe_genre.index > 4, 'genre'] = "other"
europe_genre = pd.pivot_table(europe_genre, index = "genre", aggfunc="sum").reset_index(

japan_genre = (
    games_actual
    .groupby("genre")
    .agg({"jp_sales":"sum"})
    .sort_values(by="jp_sales", ascending=False)
    .reset_index(0))

japan_genre.loc[japan_genre.index > 4, 'genre'] = "other"
japan_genre = pd.pivot_table(japan_genre, index = "genre", aggfunc="sum").reset_index(0)
```

In [71]:
```
#Выведим три графика распределения продаж между платформами по каждому региону

#Задаем цвета и массивы
regions_genre = [north_america_genre, europe_genre, japan_genre]
reg_data = ["na_sales", "eu_sales", "jp_sales"]
names = ["Северная Америка", "Европа", "Япония"]
pie_chart_colors = ('lemonchiffon', 'greenyellow', 'lime', "gold", "palegreen", "navajow


fig, axs = plt.subplots(1, 3, figsize=(25, 7))
fig.suptitle('Портрет пользователей регионов по продажам в наиболее популярных жанрах' +

for i, ax in enumerate(axs.flat):
    ax.pie(reg_data[i], labels=list(regions_genre[i]["genre"]), autopct='%1.1f%%', color
    ax.set_ylabel('Продажи')
    ax.set_xlabel('Жанр')
    ax.set_title(names[i])
    ax.title.set_size(15)

plt.show()
```

Япония снова отличилась :)

В целом, уже ранее бфло выявлено, что экшены являются самым популярным жанром, в Японии при это в первую очередь предпочитают Ролевые жанры (33%), что совершенно логично и ожидаемо для Японской культуры. А вот шутеры Японцы в отличии от Европейцев и Амереканцев, (у которых их популярность составляет примерно 16-27% рынка) не любят. Спортивные игры в Японии также не популярны.

Влияние ESRB на продажи по регионам

In [72]:
```
#Еще раз выведем уникальные значения рейтинга
games_actual["rating"].unique()
```

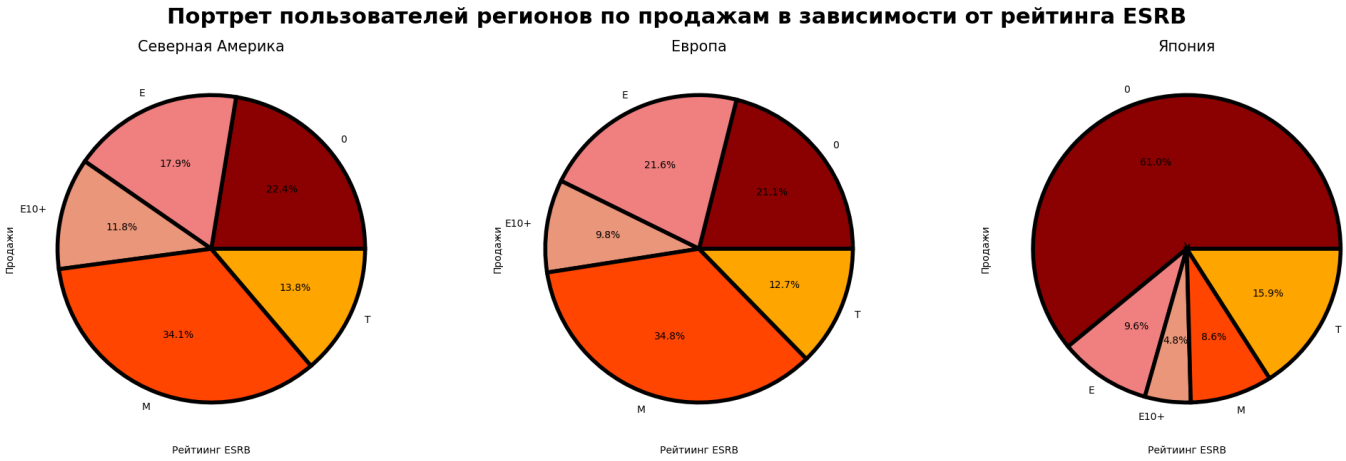
Out[72]:
```
array([0, 'M', 'E', 'T', 'E10+', nan], dtype=object)
```

In [73]:
```
#Заменим отсутствие рейтинга ESRB на No, чтобы видеть продажи без рейтинга на графике
games_actual.loc[(games_actual["rating"] == np.nan) & (games_actual["no_scored"] == 1),
```

In [74]:
```
#Сделаем срезы данных из основного массива, которые будут содержать информацию только по
#по рейтингуу ESRB
north_america_rating = (
    games_actual
    .groupby("rating")
    .agg({"na_sales":"sum"})
    .sort_values(by="na_sales", ascending=False)
    .reset_index(0))

north_america_rating.loc[north_america_rating.index > 4, 'rating'] = "other"
north_america_rating = pd.pivot_table(north_america_rating, index = "rating", aggfunc="s

europe_rating = (
    games_actual
    .groupby("rating")
    .agg({"eu_sales":"sum"})
    .sort_values(by="eu_sales", ascending=False)
    .reset_index(0))

europe_rating.loc[europe_rating.index > 4, 'rating'] = "other"
europe_rating = pd.pivot_table(europe_rating, index = "rating", aggfunc="sum").reset_ind

japan_rating = (
    games_actual
    .groupby("rating")
    .agg({"jp_sales":"sum"})
    .sort_values(by="jp_sales", ascending=False)
    .reset_index(0))
```

```
japan_rating.loc[japan_rating.index > 4, 'rating'] = "other"
japan_rating = pd.pivot_table(japan_rating, index = "rating",  aggfunc="sum").reset_inde
```

In [75]:
```
#Выведим три графика распределения продаж между платформами по каждому региону

#Задаем цвета и массивы
regions_rating = [north_america_rating, europe_rating, japan_rating]
reg_data = ["na_sales", "eu_sales", "jp_sales"]
colors = ["darkred", "lightcoral", "darksalmon", "orangered", "orange"]
names = ["Северная Америка", "Европа", "Япония"]
pie_chart_colors = ('lemonchiffon', 'greenyellow', 'lime')


fig, axs = plt.subplots(1, 3, figsize=(25, 7))
fig.suptitle('Портрет пользователей регионов по продажам в зависимости от рейтинга ESRB'

for i, ax in enumerate(axs.flat):
    ax.pie(reg_data[i], labels=list(regions_rating[i]["rating"]), autopct='%1.1f%%', col
    ax.set_ylabel('Продажи')
    ax.set_xlabel('Рейтиинг ESRB')
    ax.set_title(names[i])
    ax.title.set_size(15)

plt.show()
```



Таким образом в Европе и в Америке наиболее продоваемы игры в жанре M(17+) и E(для всех). В Японии этот рейтинг распространен куда меньше, и большенство продаваемых игр не оценено. А вот в Европе и Америке игры без данного рейтинга продаются реже и занимают лишь 17% рынка. Кстати, при этом во всех трех странах приблизительно одинаковой популярностью пользуются игры с рейтингом T(13+) (12-15%).

Итак, получается, что портреты Американского и Европейского пользователей очень схожи: они используют кансоли преймущественно заподного производства (PS/Xbox) и ориентируются на возростной рейтинг ESRB. В то время как Японские пользователи чаще используют консоли собственного производства (DS) и очень слабо используют данный рейтинг (видемо имея свой внутренний).

Проверка статистических гипотез

Далее выдвенем и проверим две статистические гипотезы относительно пользовательских рейтингов:

- Средние пользовательские рейтинги платформ Xbox One и PC одинаковые;

**- Средние пользовательские рейтинги жанров Action (англ. «действие», экшен-игры) и Sports (англ. «спортивные соревнования») разные. **

В основы проверки гипотез ляжет двувыборочный статистический тест: t-test Стьюдента. Который используется для порверки гипотез о равенсве или различии средних между выборочными совокупностями

рейтинги платформ Xbox One и PC

Иток, выдвенем нулевую статистическую гипотезу (H0), которая будет звучать следующим образом: **Средние пользовательские рейтинги платформ Xbox One и PC равны.** Соответственно, альтернативная гипотеза - двусторонняя: **Средние пользовательские рейтинги платформ Xbox One и PC не равны.**

In [76]:
```
games_test = games.loc[games["no_scored"] == 0]
games_test = games_test.dropna()
games_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6835 entries, 0 to 16702
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             6835 non-null   object 
 1   platform         6835 non-null   object 
 2   year_of_release  6835 non-null   int32  
 3   genre            6835 non-null   object 
 4   na_sales         6835 non-null   float64
 5   eu_sales         6835 non-null   float64
 6   jp_sales         6835 non-null   float64
 7   other_sales      6835 non-null   float64
 8   critic_score     6835 non-null   float64
 9   user_score       6835 non-null   float64
 10  rating           6835 non-null   object 
 11  sum_sales        6835 non-null   float64
 12  no_scored        6835 non-null   int64  
dtypes: float64(7), int32(1), int64(1), object(4)
memory usage: 720.9+ KB
```

In [77]:
```
#Выведем годы начал производств видеоигр на рассматриваемых платформах
print("XOne:", games_test.query("platform == 'XOne'")["year_of_release"].min(),
      "PC:", games_test.query("platform == 'PC'")["year_of_release"].min())
```

```
XOne: 2013 PC: 1985
```

Так, игргы на ПК стали выпускаться гораздо раньше. Для сравнения выборок, возьмем значение в одинаковый временной промежуток

In [78]:
```
games_test_1 = games_test.loc[games_test["year_of_release"] >= 2013]
```

In [79]:
```
#Для проверки данной гипотезы необходимо сформировать два массива: Рейтинги Xbox и рейти

xbox_ratings = games_test_1.query("platform == 'XOne'")["user_score"]

pc_ratings = games_test_1.query("platform == 'PC'")["user_score"]

print("Среднее значениие по Xbox:", xbox_ratings.mean())
print("Среднее значениие по PC:", pc_ratings.mean())
```

```
Среднее значениие по Xbox: 6.49308176100629
Среднее значениие по PC: 6.235820895522389
```

Уровень статистической значимости alpha будет равняться 0.05 (5%), что является одним из унифицированных значений. Выбор в пользу 5% вместо 1% основан на том, что выборочные данные не большие и получить меньше 1% будет тяжело, поэтому 5ти будет достаточно.

In [80]:
```
#Зададим уровень статистической значимости
alpha = 0.05

#Осуществим проверку двувыборочным статистическим t-test'ом для двусторонней гипотезы (в
test_result = st.ttest_ind(xbox_ratings, pc_ratings, equal_var=False)

print("p_значение:", test_result.pvalue)

if test_result.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Нулевая гипотеза не отвергается")
```

```
p_значение: 0.16694185960105132
Нулевая гипотеза не отвергается
```

Нулевая гипотеза не отвергается, что означает, что наблюдаемые данные статистически не противоречат выдвинутой гипотезе, а значит - средние пользовательский рейтинг платформ Xbox One и PC равны.

P_значение равняется 0.16 или 16%, что можно интерпретировать таким образом, что с вероятностью 16% наблюдаемое (или большее) различие получается случайно, чего не достаточно (согласно выбранному доверительному интервалу) для того, чтобы предположить верность альтернативной гипотезы об отличии

Вывод - Нулевая гипотеза не была отвергнута - рейтинг платформ Xbox One и PC равны.

In [81]:
```
# Приведены два датасета: сумма покупок, совершённых за месяц посетителями ...

sample_1 = [3071, 3636, 3454, 3151, 2185, 3259, 1727, 2263, 2015,
2582, 4815, 633, 3186, 887, 2028, 3589, 2564, 1422, 1785,
3180, 1770, 2716, 2546, 1848, 4644, 3134, 475, 2686,
1838, 3352]
sample_2 = [1211, 1228, 2157, 3699, 600, 1898, 1688, 1420, 5048, 3007,
509, 3777, 5583, 3949, 121, 1674, 4300, 1338, 3066,
3562, 1010, 2311, 462, 863, 2021, 528, 1849, 255,
1740, 2596]
alpha = .05 # критический уровень статистической значимости
# если p-value окажется меньше него - отвергнем гипотезу
results = st.ttest_ind(
sample_1,
sample_2)
print('p-значение:', results.pvalue)
if (results.pvalue < alpha):
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
p-значение: 0.1912450522572209
Не получилось отвергнуть нулевую гипотезу
```

рейтинги жанров Action и Sports

Далее необходимо проверить различность рейтингов двух жанров. **Средние пользовательские рейтинги платформ жанорв Action и Sports не равны.** Однако нулевую гипотезу также выдвенем

через сходство средних, а не через различие. Смысл проверки от этого не должен поменяться.

Поэтому выдвегаем нулевую статистическую гипотезу (H0), которая будет звучать следующим образом: **Средние пользовательские рейтинги игр в жанре Action и Sports равны.** Соответственно, альтернативная гипотеза - двусторонняя: **Средние пользовательские рейтинги игр в жанре Action и Sports не равны.**

In [82]:
```
print("XOne:", games_test.query("genre == 'Action'")["year_of_release"].min(),
      "PC:", games_test.query("genre == 'Sports'")["year_of_release"].min())
```

```
XOne: 1996 PC: 1998
```

Временные промежутки выборок примерно равны, оставим в исходном виде

In [83]:
```
#Для проверки данной гипотезы необходимо сформировать два массива: Рейтинги Экшена и рей
action_rating = games_test.query("genre == 'Action'")["user_score"]

sports_rating = games_test.query("genre == 'Sports'")["user_score"]

print("Среднее значениие по Action:", action_rating.mean())
print("Среднее значениие по Sports:", sports_rating.mean())
```

```
Среднее значениие по Action: 7.096689147762109
Среднее значениие по Sports: 7.1216842105263165
```

In [84]:
```
#Здададим уровень статистической значимости
alpha = 0.05

#Осуществим проверку двувыборочным статистическим t-test'ом для двусторонней гипотезы
test_result = st.ttest_ind(action_rating, sports_rating, equal_var=False)

print("p_значение:", test_result.pvalue)

if test_result.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Нулевая гипотеза не отвергается")
```

```
p_значение: 0.6686293034269601
Нулевая гипотеза не отвергается
```

Нулевая гипотеза не отвергается, что означает, что наблюдаемые данные статистически не противоречат выдвинутой гипотезе, а значит - средние пользовательский рейтинг жанорв Action и Sports равны.

P_значение равняется 0.66 или 66%, что можно интерпретировать таким образом, что с вероятностью 66% наблюдаемое (или большее) различие между рейтингами получается случайно, чего не достаточно (согласно выбранному доверительному интервалу) для того, чтобы предположить верность альтернативной гипотезы об отличии

Вывод: Соответственно, данных для отвержения нулевой гипотезы недостаточно - рейтинги жанорв Action и Sports равны.

Общие выводы

Итак, проектная работа по анализу рынка видеоигр позволила провести глубокое исследование данных, выявить основные тренды и закономерности, которые характеризуют данную индустрию. Анализ продаж, платформ, жанров, региональных предпочтений и влияния отзывов на успех игр

позволил получить необходимую информацию для разработки стратегий продвижения и маркетинга магазиина видеоигр.

В ходе данной проектной работы были осущетслвены:

1) Импорт и первичный осмотр данных:

- Импортированный массив состоял из **16713 наблюдений**.
- Первичный осмотр данных позволил оценить общую структуру набора информации, которая не вызвала особых вопросов и затруднений

2) Предобработка данных:

- Названия столбцов были исправлены для удобства работы с данными.
- Пропуски в данных были обработаны, дубликаты проверены. В результате работы с пропусками осталосб **16462** наблюдений. Две переменные остались примерно с двумя тысячами пропусков
- Типы данных были скорректированы для дальнейшего анализа в соответствии с необходимостью.
- В качестве новых данных было добавлено агрегированное значение **sum_sales, для работы с общим числом продаж**.

3) Исследовательский анализ данных:

- Активность выпуска видеоигр по годам была выявлена. **Рынок начал функционировать примерно с 1995 года, однако в итоге оказалось, что актуальный для анализа период начинается только с 2014**
- Наиболее успешные компании в выпуске видеоигр к 2017 году были определены. К их числу отнеслись **'3DS', 'PS4', 'PC', 'XOne', 'WiiU'**.
- Изучено изменение продаж по платформам и средний период популярности каждой платформы. *Продажи изменяются в парабалической форме т.е. идут в верх, достигают пика и спадают*. **Средний период популярности отдельной платформы - 10 лет.**
- Потенциальные лидеры на 2017 год были выявлены. К их чеслу были отнесены **PS4' и 'XOne'**, также к активно действующим платформам были отнесены **PC, PS3, WiiU, X360, 3DS**
- Влияние отзывов пользователей и критиков на продажи было проанализировано. Коэффициент корреляции показал, что **покупатели слабо ориентируются на отзывы пользователей об игре. На отзывы критиков - более активно, но также не первостепенно, умеренно. Средний уровень взаимосвязи - 0,33 (измеряется до 1)**.
- Распределение игр по жанрам было исследовано. Самые перспективные жанры: **Шутеры, Спортивные игры и Экшены**

4) Составление портрета пользователя каждого региона:

- Самые популярные платформы и жанры в каждом регионе были определены.
- Влияние рейтинга ESRB на продажи в отдельном регионе было проанализировано.

  портреты Американского и Европейского пользователей очень схожи: они используют кансоли преймущественно заподного производства (PS/Xbox) и ориентируются на возростной рейтинг ESRB. В то время как Японские пользователи чаще используют консоли собственного производства (DS) и очень слабо используют данный рейтинг

5) Проверка статистических гипотез:

- Была проверена гипотеза о равенстве средних пользовательских рейтингов платформ Xbox One и PC. **Отвергнуть нулевую гипотезу о равенстве не удалось - рейтинг платформ Xbox One и PC не различаются.**

- Была проверена гипотеза о различии средних пользовательских рейтингов жанров Action и Sports. **Нулевую гипотезу отвергуть не удалось, а значит вышепоставленная альтернативная гипотеза не принимается - рейтинги жанорв Action и Sports одинаковы.**

Итоги работы:

В ходе анализа данных были успешно выполнены поставленные задачи, получены ценные выводы о рынке видеоигр, предпочтениях пользователей и влиянии различных факторов на продажи. Результаты анализа помогут принять обоснованные решения в сфере выбора траектории продажи видеоигр.