Table of contents

ICE RETAIL

Introduction

We are looking at data from Ice, an online retail store for video games. The data set includes categories based on genre, platform, rating, sales, release year, critic, and user scores.

Purpose

The purpose of this project is to collect historical data, and apply insights to determine whether a game will succeed or not in the future. These insights will be used to optimize capital allocation in regards to advertising potentially big winners. We will be determining total sales, and the distribution of sales across the different platforms. We will illustrate which platforms lead, and lag in sales. Data analysis will determine the most popular platforms and genres. Investigations will determine if the average user ratings of certain platforms are the same, or if the average ratings of certain genres are the same.

Reading and Viewing Data

In [ ]:
```
!pip install --user -U plotly_express
```

```
Requirement already satisfied: plotly_express in c:\users\xix\appdata\roaming\python\python39\site-packages (0.4.1)
Requirement already satisfied: numpy>=1.11 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (1.23.5)
Requirement already satisfied: patsy>=0.5 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (0.5.2)
Requirement already satisfied: scipy>=0.18 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (1.8.1)
Requirement already satisfied: plotly>=4.1.0 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (5.6.0)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (0.13.2)
Requirement already satisfied: pandas>=0.20.0 in c:\users\xix\anaconda3\lib\site-packages (from plotly_express) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\xix\anaconda3\lib\site-packages (from pandas>=0.20.0->plotly_express) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\xix\anaconda3\lib\site-packages (from pandas>=0.20.0->plotly_express) (2.8.2)
Requirement already satisfied: six in c:\users\xix\anaconda3\lib\site-packages (from patsy>=0.5->plotly_express) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\xix\anaconda3\lib\site-packages (from plotly>=4.1.0->plotly_express) (8.0.1)
Requirement already satisfied: packaging>=21.3 in c:\users\xix\anaconda3\lib\site-packages (from statsmodels>=0.9.0->plotly_express) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\xix\anaconda3\lib\site-packages (from packaging>=21.3->statsmodels>=0.9.0->plotly_exp
ress) (3.0.4)
```

In [ ]:
```
# Import useful packages
import pandas as pd
import numpy as np
import math as mt
from scipy import stats as st
import plotly.express as px
import matplotlib.pyplot as plt
import plotly.graph_objects as go
```

In [ ]:
```
# Read the dataframe
df = pd.read_csv('datasets/games.csv')
```

In [ ]:
```
# Visual of the Dataframe
display(df.head())
```

	Name	Platform	Year_of_Release	Genre	NA_sales	EU_sales	JP_sales	Other_sales	Critic_Score	User_Score	Rating
0	Wii Sports	Wii	2006.0	Sports	41.36	28.96	3.77	8.45	76.0	8	E
1	Super Mario Bros.	NES	1985.0	Platform	29.08	3.58	6.81	0.77	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	15.68	12.76	3.79	3.29	82.0	8.3	E
3	Wii Sports Resort	Wii	2009.0	Sports	15.61	10.93	3.28	2.95	80.0	8	E
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	11.27	8.89	10.22	1.00	NaN	NaN	NaN

In [ ]:
```
# We see the columns need to be changed to lowercase
df.columns
```

Out[ ]:
```
Index(['Name', 'Platform', 'Year_of_Release', 'Genre', 'NA_sales', 'EU_sales',
       'JP_sales', 'Other_sales', 'Critic_Score', 'User_Score', 'Rating'],
      dtype='object')
```

In [ ]:
```
# Converting columns to lowercase
df.columns = df.columns.str.lower()
```

In [ ]:
```
# Info of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16715 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   name             16713 non-null  object 
 1   platform         16715 non-null  object 
 2   year_of_release  16446 non-null  float64
 3   genre            16713 non-null  object 
 4   na_sales         16715 non-null  float64
 5   eu_sales         16715 non-null  float64
 6   jp_sales         16715 non-null  float64
 7   other_sales      16715 non-null  float64
 8   critic_score     8137 non-null   float64
 9   user_score       10014 non-null  object 
 10  rating           9949 non-null   object 
dtypes: float64(6), object(5)
memory usage: 1.4+ MB
```

We see some missing values in some columns, especially with the scores and rating. Sales data is all there, and all but 2 game names are present.

In [ ]:
```
# Description of the dataset
df.describe()
```

Out[ ]:

	year_of_release	na_sales	eu_sales	jp_sales	other_sales	critic_score
count	16446.000000	16715.000000	16715.000000	16715.000000	16715.000000	8137.000000
mean	2006.484616	0.263377	0.145060	0.077617	0.047342	68.967679
std	5.877050	0.813604	0.503339	0.308853	0.186731	13.938165
min	1980.000000	0.000000	0.000000	0.000000	0.000000	13.000000
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	60.000000
50%	2007.000000	0.080000	0.020000	0.000000	0.010000	71.000000
75%	2010.000000	0.240000	0.110000	0.040000	0.030000	79.000000
max	2016.000000	41.360000	28.960000	10.220000	10.570000	98.000000

In [ ]:
```
# Unique values of each column
df.nunique()
```

Out[ ]:
```
name               11559
platform              31
year_of_release       37
genre                 12
na_sales             402
eu_sales             307
jp_sales             244
other_sales          155
critic_score          82
user_score            96
rating                 8
dtype: int64
```

We have 31 different platforms spanning a time frame of 37 years. The dataset has 12 different genres, a range of 96 values for user score, and 8 different values for rating. Most games were sold in the North American region, while less games are sold in the Japanese region. The data shows the sales of 4 regions, which we can use to make summary data on total and average sales. There appears to be a large quantity of score and rating data missing.

Cleaning Data

Changing Float Columns to Integers

In [ ]:
```
# Visual of different user score values
df['user_score'].unique()
```

Out[ ]:
```
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)
```

In [ ]:
```
# replacing TBD with a value
df['user_score'] = df['user_score'].replace(to_replace='tbd', value=0)
```

In [ ]:
```
# Need to convert user score from object to integer
df['user_score'] = pd.to_numeric(df['user_score'], errors='coerce')
```

In [ ]:
```
# Change release year to integer, years dont have decimals
df['year_of_release'] = df['year_of_release'].astype('int', errors='ignore')
```

Checking for Duplicates

In [ ]:
```
# Checking for duplicates
df.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
# Checking for duplicates filtering for same name and platform
df[df[['name', 'platform']].duplicated()].value_counts(ascending=False)
```

Out[ ]:
```
name                       platform  year_of_release  genre   na_sales  eu_sales  jp_sales  other_sales  critic_score  user_score  rating
Madden NFL 13              PS3       2012.0           Sports  0.0       0.01      0.00      0.00         83.0          5.5         E         1
Need for Speed: Most Wanted  PC      2012.0           Racing  0.0       0.06      0.00      0.02         82.0          8.5         T         1
                           X360      2005.0           Racing  1.0       0.13      0.02      0.10         83.0          8.5         T         1
dtype: int64
```

In [ ]:
```
# checking for duplicates based on same name, platform, genre
df[df[['name','platform', 'genre']].duplicated()]
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
1591	Need for Speed: Most Wanted	X360	2005.0	Racing	1.0	0.13	0.02	0.10	83.0	8.5	T
4127	Sonic the Hedgehog	PS3	NaN	Platform	0.0	0.48	0.00	0.00	43.0	4.1	E10+
11715	Need for Speed: Most Wanted	PC	2012.0	Racing	0.0	0.06	0.00	0.02	82.0	8.5	T
14244	NaN	GEN	1993.0	NaN	0.0	0.00	0.03	0.00	NaN	NaN	NaN
16230	Madden NFL 13	PS3	2012.0	Sports	0.0	0.01	0.00	0.00	83.0	5.5	E

In [ ]:
```
# looking for games with duplicate platform and year of release as hidden duplicates
df[df[['name','platform', 'year_of_release']].duplicated()]
```

Out[ ]:

	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

In [ ]:
```
# Checking for Sonic
df[df['name']== 'Sonic the Hedgehog']
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
257	Sonic the Hedgehog	GEN	1991.0	Platform	3.03	0.91	0.26	0.13	NaN	NaN	NaN
1745	Sonic the Hedgehog	PS3	2006.0	Platform	0.41	0.06	0.04	0.66	43.0	4.1	E10+
1996	Sonic the Hedgehog	X360	2006.0	Platform	0.44	0.48	0.00	0.11	46.0	4.4	E10+
4127	Sonic the Hedgehog	PS3	NaN	Platform	0.00	0.48	0.00	0.00	43.0	4.1	E10+

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

Out[ ]:

	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 [ ]:
```
# Looking at Madden hidden duplicates
df[df['name']== 'Madden NFL 13']
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
507	Madden NFL 13	X360	2012.0	Sports	2.53	0.15	0.0	0.17	81.0	5.8	E
604	Madden NFL 13	PS3	2012.0	Sports	2.11	0.22	0.0	0.23	83.0	5.5	E
3986	Madden NFL 13	Wii	2012.0	Sports	0.47	0.00	0.0	0.03	NaN	7.3	E
5887	Madden NFL 13	PSV	2012.0	Sports	0.28	0.00	0.0	0.02	63.0	7.3	E
7066	Madden NFL 13	WiiU	2012.0	Sports	0.21	0.00	0.0	0.02	75.0	6.7	E
16230	Madden NFL 13	PS3	2012.0	Sports	0.00	0.01	0.0	0.00	83.0	5.5	E

In [ ]:
```
# Dropping the hidden duplicate
df = df.drop_duplicates(subset=['name','platform', 'year_of_release'], keep='first')
```

In [ ]:
```
# QC check
df[df[['name','platform', 'genre', 'year_of_release']].duplicated()]
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating

In [ ]:
```
# Looking for hidden duplicates
df.groupby(['name', 'platform'])['name'].value_counts(ascending=False)
```

Out[ ]:
```
name                          platform  name
 Beyblade Burst               3DS        Beyblade Burst                1
 Fire Emblem Fates            3DS        Fire Emblem Fates             1
 Frozen: Olaf's Quest         3DS        Frozen: Olaf's Quest          1
                              DS         Frozen: Olaf's Quest          1
 Haikyu!! Cross Team Match!   3DS        Haikyu!! Cross Team Match!    1
                                                                     ..
uDraw Studio                  Wii       uDraw Studio                  1
uDraw Studio: Instant Artist  Wii       uDraw Studio: Instant Artist  1
                              X360      uDraw Studio: Instant Artist  1
wwe Smackdown vs. Raw 2006    PS2       wwe Smackdown vs. Raw 2006    1
¡Shin Chan Flipa en colores!  DS        ¡Shin Chan Flipa en colores!  1
Name: name, Length: 16709, dtype: int64
```

We see Madden 13 has a hidden duplicate among the PS3 platform. We decide to keep the first, and drop the second instance, as the second has minimal data. Sonic and Need for speed do not have duplicate games, released on the same platform, in the same year.

Checking for Missing Values

In [ ]:
```
# Looking for missing values
df.isna().sum()
```

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

Missing Names

In [ ]:
```
# Visual of the rows of missing names
df[df['name'].isna()]
```

Out[ ]:

	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

In [ ]:
```
# Dropping the rows of missing names
df = df.dropna(subset=['name', 'genre'])
```

Missing Year of Release

In [ ]:
```
# Looking for vales with missing release years
df[df['year_of_release'].isna()]
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating
183	Madden NFL 2004	PS2	NaN	Sports	4.26	0.26	0.01	0.71	94.0	8.5	E
377	FIFA Soccer 2004	PS2	NaN	Sports	0.59	2.36	0.04	0.51	84.0	6.4	E
456	LEGO Batman: The Videogame	Wii	NaN	Action	1.80	0.97	0.00	0.29	74.0	7.9	E10+
475	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	1.57	1.02	0.00	0.41	NaN	NaN	NaN
609	Space Invaders	2600	NaN	Shooter	2.36	0.14	0.00	0.03	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
16373	PDC World Championship Darts 2008	PSP	NaN	Sports	0.01	0.00	0.00	0.00	43.0	0.0	E10+
16405	Freaky Flyers	GC	NaN	Racing	0.01	0.00	0.00	0.00	69.0	6.5	T
16448	Inversion	PC	NaN	Shooter	0.01	0.00	0.00	0.00	59.0	6.7	M
16458	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	0.01	0.00	0.00	0.00	NaN	NaN	NaN
16522	Virtua Quest	GC	NaN	Role-Playing	0.01	0.00	0.00	0.00	55.0	5.5	T

269 rows × 11 columns

In [ ]:
```
# Mean release year per platform
df.groupby('platform')['year_of_release'].mean().round()
```

Out[ ]:
```
platform
2600    1982.0
3DO     1995.0
3DS     2013.0
DC      2000.0
DS      2008.0
GB      1996.0
GBA     2003.0
GC      2003.0
GEN     1993.0
GG      1992.0
N64     1999.0
NES     1987.0
NG      1994.0
PC      2009.0
PCFX    1996.0
PS      1998.0
PS2     2005.0
PS3     2011.0
PS4     2015.0
PSP     2009.0
PSV     2014.0
SAT     1996.0
SCD     1994.0
SNES    1994.0
TG16    1995.0
WS      2000.0
Wii     2009.0
WiiU    2014.0
X360    2010.0
XB      2004.0
XOne    2015.0
Name: year_of_release, dtype: float64
```

In [ ]:
```
# Median release year per platform
df.groupby('platform')['year_of_release'].median()
```

Out[ ]:
```
platform
2600    1982.0
3DO     1995.0
3DS     2013.0
DC      2000.0
DS      2008.0
GB      1997.0
GBA     2003.0
GC      2003.0
GEN     1993.0
GG      1992.0
N64     1999.0
NES     1986.5
NG      1994.5
PC      2010.0
PCFX    1996.0
PS      1998.0
PS2     2005.0
PS3     2011.0
PS4     2015.0
PSP     2009.0
PSV     2014.0
SAT     1996.0
SCD     1994.0
SNES    1994.0
TG16    1995.0
WS      2000.0
Wii     2009.0
WiiU    2013.0
X360    2010.0
XB      2004.0
XOne    2015.0
Name: year_of_release, dtype: float64
```

In [ ]:
```
# Percent of data with missing year
(df['year_of_release'].isna().sum() / df['year_of_release'].count()) * 100
```

Out[ ]:
```
1.635954509517728
```

In [ ]:
```
# fill year of release based on median of platform grouping
df = df.dropna(subset=['year_of_release'])
```

In [ ]:
```
# checking removal of missing values
df['year_of_release'].isna().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
# Change release year to integer
df['year_of_release'] = df['year_of_release'].astype('int')
```

In [ ]:
```
# info on columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16443 entries, 0 to 16714
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   name             16443 non-null  object
 1   platform         16443 non-null  object
 2   year_of_release  16443 non-null  int32
 3   genre            16443 non-null  object
 4   na_sales         16443 non-null  float64
 5   eu_sales         16443 non-null  float64
 6   jp_sales         16443 non-null  float64
 7   other_sales      16443 non-null  float64
 8   critic_score     7982 non-null   float64
 9   user_score       9838 non-null   float64
 10  rating           9767 non-null   object
dtypes: float64(6), int32(1), object(4)
memory usage: 1.4+ MB
```

Missing Rating

In [ ]:
```
# Looking for missing ratings
df['rating'].isna().sum()
```

Out[ ]:
```
6676
```

In [ ]:
```
# trying to figure out missing ratings
df.groupby(['name','genre', 'platform'])['rating'].value_counts()
```

Out[ ]:
```
name                          genre         platform  rating
 Tales of Xillia 2            Role-Playing  PS3       T         1
.hack//Infection Part 1       Role-Playing  PS2       T         1
.hack//Mutation Part 2        Role-Playing  PS2       T         1
.hack//Outbreak Part 3        Role-Playing  PS2       T         1
007 Racing                    Racing        PS        T         1
                                                                ..
thinkSMART FAMILY!            Misc          Wii       E         1
thinkSMART: Chess for Kids    Misc          DS        E         1
uDraw Studio                  Misc          Wii       E         1
uDraw Studio: Instant Artist  Misc          Wii       E         1
                                            X360      E         1
Name: rating, Length: 9765, dtype: int64
```

In [ ]:
```
# The ratings of the titles, based on the platform
df.groupby(['name', 'platform'])['rating'].value_counts()
```

Out[ ]:
```
name                          platform  rating
 Tales of Xillia 2            PS3       T         1
.hack//Infection Part 1       PS2       T         1
.hack//Mutation Part 2        PS2       T         1
.hack//Outbreak Part 3        PS2       T         1
007 Racing                    PS        T         1
                                                 ..
thinkSMART FAMILY!            Wii       E         1
thinkSMART: Chess for Kids    DS        E         1
uDraw Studio                  Wii       E         1
uDraw Studio: Instant Artist  Wii       E         1
                              X360      E         1
Name: rating, Length: 9765, dtype: int64
```

In [ ]:
```
# Grouping titles based on rating
df.groupby('name')['rating'].value_counts()
```

Out[ ]:
```
name                          rating
 Tales of Xillia 2            T         1
.hack//Infection Part 1       T         1
.hack//Mutation Part 2        T         1
.hack//Outbreak Part 3        T         1
007 Racing                    T         1
                                       ..
thinkSMART                    E         1
thinkSMART FAMILY!            E         1
thinkSMART: Chess for Kids    E         1
uDraw Studio                  E         1
uDraw Studio: Instant Artist  E         2
Name: rating, Length: 6170, dtype: int64
```

In [ ]:
```
# Rating count per title
df[['name', 'rating']].value_counts()
```

Out[ ]:
```
name                    rating
FIFA Soccer 13          E         8
Ratatouille             E         8
FIFA 15                 E         8
Terraria                T         8
FIFA 14                 E         8
                                 ..
Imagine: Teacher        E         1
Imagine: Sweet 16       E         1
Imagine: Soccer Captain E         1
Imagine: Salon Stylist  E         1
Mega Man Legends 2      E         1
Length: 6170, dtype: int64
```

In [ ]:
```
# counting the different ratings of platforms
platform_rating = df[['platform', 'rating']].value_counts().reset_index()
display(platform_rating)
```

	platform	rating	0
0	DS	E	865
1	PS2	T	567
2	PS2	E	541
3	Wii	E	493
4	GBA	E	419
...	...	...	...
70	GC	EC	1
71	PC	RP	1
72	PS	K-A	1
73	PS2	EC	1
74	DC	M	1

75 rows × 3 columns

In [ ]:
```
# sorting by platform
platform_rating.sort_values('platform')
```

Out[ ]:

	platform	rating	0
60	3DS	M	12
42	3DS	T	47
35	3DS	E10+	74
31	3DS	E	90
74	DC	M	1
...	...	...	...
53	XB	E10+	31
51	XOne	E10+	31
45	XOne	T	40
36	XOne	M	70
43	XOne	E	45

75 rows × 3 columns

In [ ]:
```
# Best option
genre_rating = df[['genre', 'rating']].value_counts()
```

We dropped the two entries with missing names, as they also had other critical data that was missing. We fill in missing year of release based on the median of the year the platform was released. This should have a minimal effect on our data, as we are filling a few hundred missing values. Then, we changed the year of release into an integer type, as we should not have decimals in our years.

We fill TBD user scores with zero, to differentiate them from the other scores. Using a median value would change the results of the data. We believe these values are labeled as TBD, as a filler because data was not collected or missing. Another possibility is the users are still providing data on the scores of the games, or not enough data has been collected.

We looked at grouping the data to determine a median value to replace the missing score and ratings data. The best option was to determine the ratings by genre. However, we decided to keep missing user score, critic score, and rating values as is. We decide to keep those missing values, as filling them in with mean or median values would alter the data, as roughly half of those data points are currently missing. We notice that those rows missing scores are also missing ratings. Therefore, the negative effect of these games on the data is limited. Also, we refrain from dropping those rows, as they still have crucial sales, genre, name, and platform data.

Feature Engineering

In [ ]:
```
# Changing game names to lowercase, platforms to uppercase
df['name'] = df['name'].str.lower()
df['platform'] = df['platform'].str.upper()
```

In [ ]:
```
# scaling down critic score
df['critic_score'] = df['critic_score'] / 10
```

In [ ]:
```
# Creating average score column
df['average_score'] = df[['critic_score', 'user_score']].mean(axis=1)
```

In [ ]:
```
# Adding total sales column
df['total_sales'] = df[['na_sales','eu_sales', 'jp_sales', 'other_sales']].sum(axis=1)
```

In [ ]:
```
# Adding average sales column
df['average_sales'] = df[['na_sales','eu_sales', 'jp_sales', 'other_sales']].mean(axis=1)
```

We scale down critic score to the same scale as user score, so we can get an average score column. Average score is the mean of user and critic scores. We add a total sales and an average sales column to the data.

Analyzing Data

Game Releases

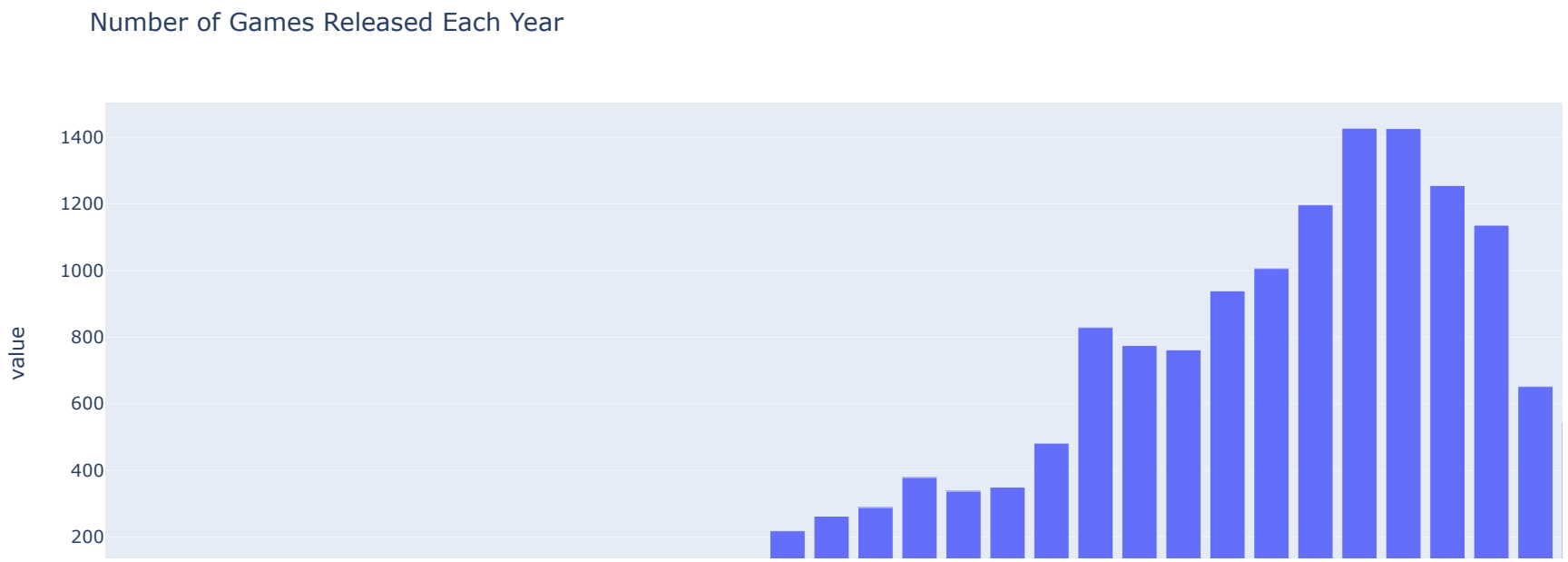
In [ ]:
```
# Number of games released in each year
game_years = df['year_of_release'].value_counts()
game_years = pd.DataFrame(game_years)
```

In [ ]:
```
# The years with the largest number of games released
game_years.head()
```

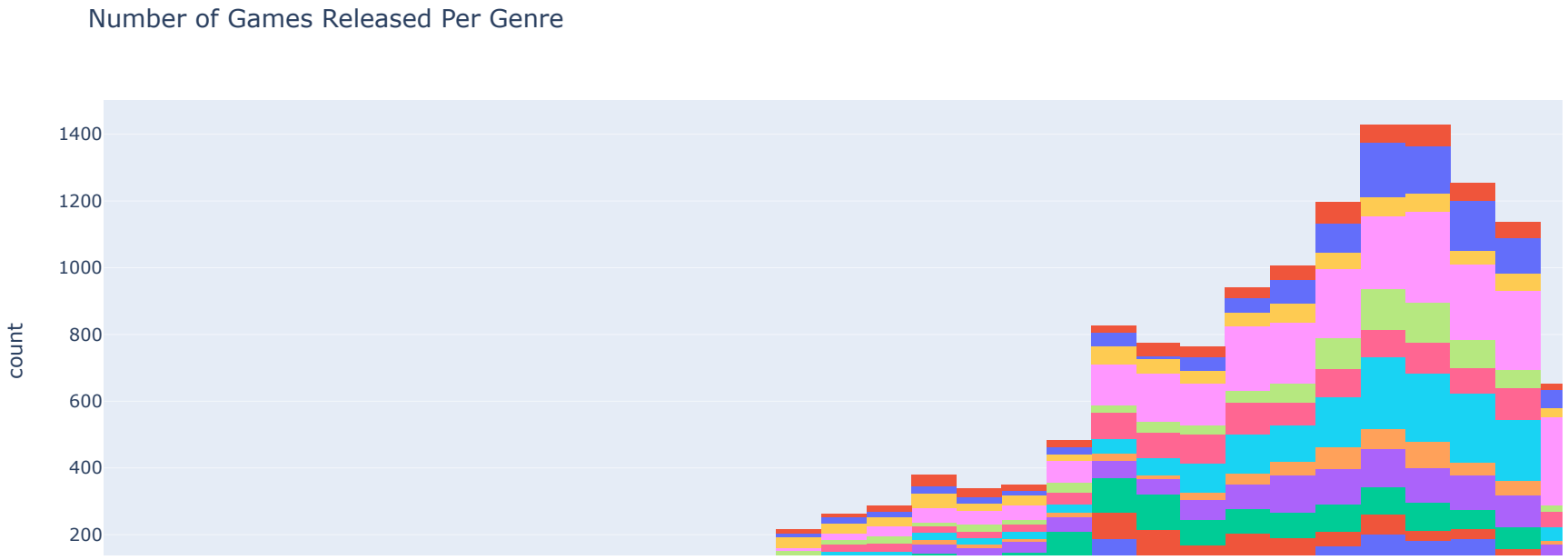
Out[ ]:

	year_of_release
2008	1427
2009	1426
2010	1255
2007	1197
2011	1136

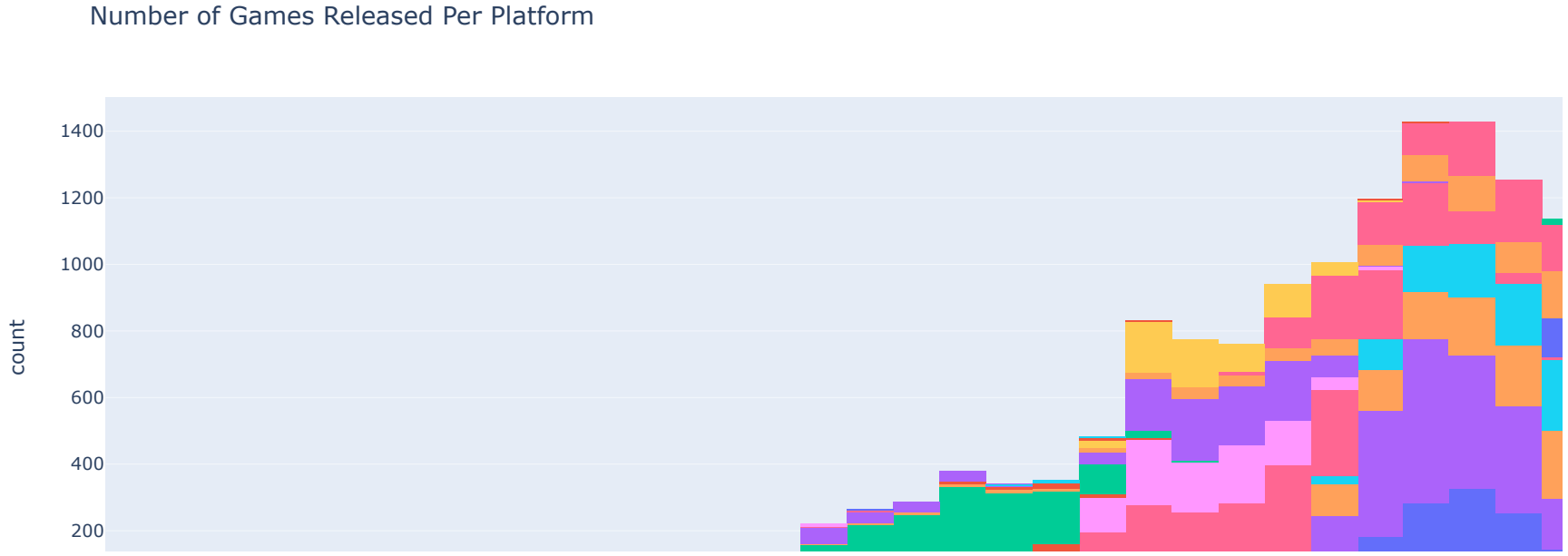
In [ ]:
```
# plot of games released each year
px.bar(game_years, title='Number of Games Released Each Year').show()
```



In [ ]:
```
# games released per genre
px.histogram(df, x='year_of_release', color='genre', title='Number of Games Released Per Genre').show()
```



In [ ]:
```
# games released per platform
px.histogram(df, x='year_of_release', color='platform', title='Number of Games Released Per Platform').show()
```



Conclusions

We see a steady increase in the number of games released from the early 90's. Game releases peak in 2008, then sharply decrease going into 2016. Sports and action games are the genres that have the most releases throughout the time period. The data also shows the changing of the various gaming platforms over time, as new platforms replace old generations.

In [ ]:
```
# Looking at all the different platforms
df['platform'].unique()
```

Out[ ]:
```
array(['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'], dtype=object)
```

In [ ]:
```
# Looking at all the all the genres
df['genre'].unique()
```

Out[ ]:
```
array(['Sports', 'Platform', 'Racing', 'Role-Playing', 'Puzzle', 'Misc',
       'Shooter', 'Simulation', 'Action', 'Fighting', 'Adventure',
       'Strategy'], dtype=object)
```

In [ ]:
```
# Median critic score of each platform
df.groupby('platform')['critic_score'].median()
```

Out[ ]:
```
platform
2600     NaN
3DO      NaN
3DS     6.80
DC      8.80
DS      6.60
GB       NaN
GBA     6.90
GC      7.00
GEN      NaN
GG       NaN
N64      NaN
NES      NaN
NG       NaN
PC      7.80
PCFX     NaN
PS      7.35
PS2     7.00
PS3     7.30
PS4     7.30
PSP     6.80
PSV     7.10
SAT      NaN
SCD      NaN
SNES     NaN
TG16     NaN
WII     6.50
WIIU    7.35
WS       NaN
X360    7.10
XB      7.20
XONE    7.60
Name: critic_score, dtype: float64
```

In [ ]:
```
# Counts of different user scores
df['user_score'].value_counts()
```

Out[ ]:
```
0.0    2377
7.8     322
8.0     285
8.2     276
8.3     252
       ...
1.5       2
0.3       2
1.1       2
1.9       2
9.7       1
Name: user_score, Length: 95, dtype: int64
```

In [ ]:
```
# Median of user score per platform
df.groupby('platform')['user_score'].median()
```

Out[ ]:
```
platform
2600     NaN
3DO      NaN
3DS     6.30
DC      8.80
DS      0.00
GB       NaN
GBA     0.00
GC      7.60
GEN      NaN
GG       NaN
N64      NaN
NES      NaN
NG       NaN
PC      7.40
PCFX     NaN
PS      7.80
PS2     7.75
PS3     6.90
PS4     7.00
PSP     7.10
PSV     7.50
SAT      NaN
SCD      NaN
SNES     NaN
TG16     NaN
WII     4.50
WIIU    7.00
WS       NaN
X360    6.80
XB      7.50
XONE    6.60
Name: user_score, dtype: float64
```

In [ ]:
```
# Total sales based on median user score
df.groupby('total_sales')['user_score'].median()
```

Out[ ]:
```
total_sales
0.00     NaN
0.01     5.2
0.02     5.4
0.03     4.8
0.04     5.1
        ...
31.38    NaN
32.77    8.0
35.52    8.3
40.24    NaN
82.54    8.0
Name: user_score, Length: 1004, dtype: float64
```

In [ ]:
```
# Count of the different ratings
df['rating'].value_counts()
```

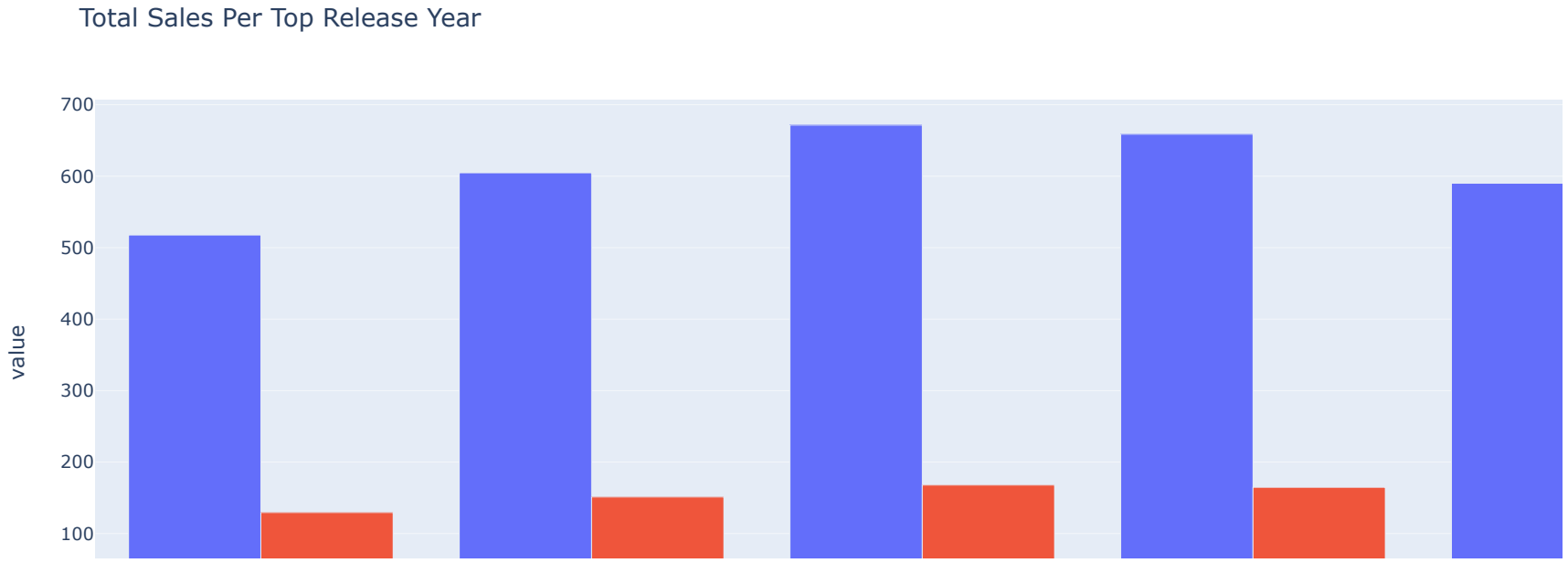
Out[ ]:
```
E       3920
T       2905
M       1536
E10+    1393
EC         8
K-A        3
AO         1
RP         1
Name: rating, dtype: int64
```

In [ ]:
```
# Platform average and total sales
top_platforms = df.pivot_table(index='platform', values=['total_sales', 'average_sales'], aggfunc='sum').reset_index()
```

Top Grossing Years

In [ ]:
```
# top grossing year
top_five_years = df.pivot_table(index='year_of_release', values=['total_sales', 'average_sales'], aggfunc='sum').reset_index()
top_five_years = top_five_years.nlargest(5, columns='total_sales')
```

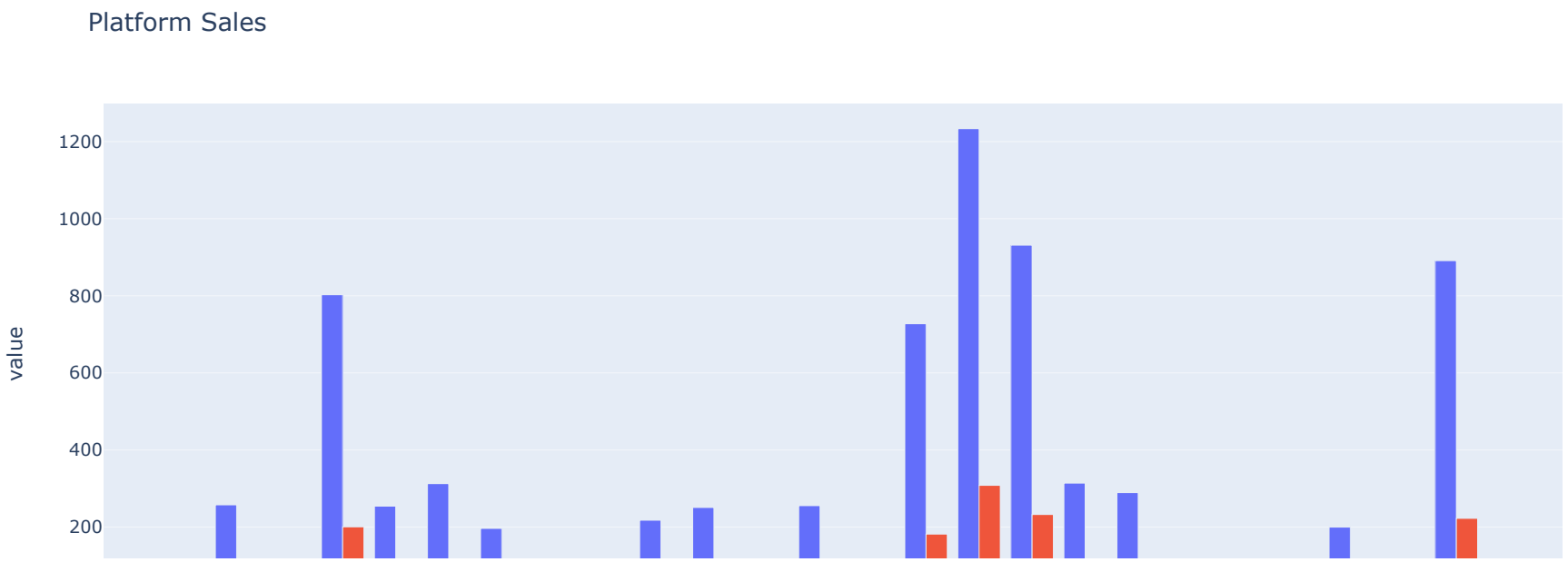
In [ ]:
```
# top 5 grossing years
px.bar(top_five_years, x='year_of_release', y=['total_sales', 'average_sales'], barmode='group', title='Total Sales Per Top Release Year').show()
```



When it comes to sales, the top grossing years are from 2006 to 2010. The year 2009 had the most game sales in the entire 37 year period from 1985 to 2016. This represents the era of the PS3, X360, and the WII, which explains why these platforms were so popular.

Total and Average Sales Per Platform

In [ ]:
```
# platform total and average sales
px.bar(top_platforms, x='platform', y=['total_sales', 'average_sales'], barmode='group', title='Platform Sales').show()
```



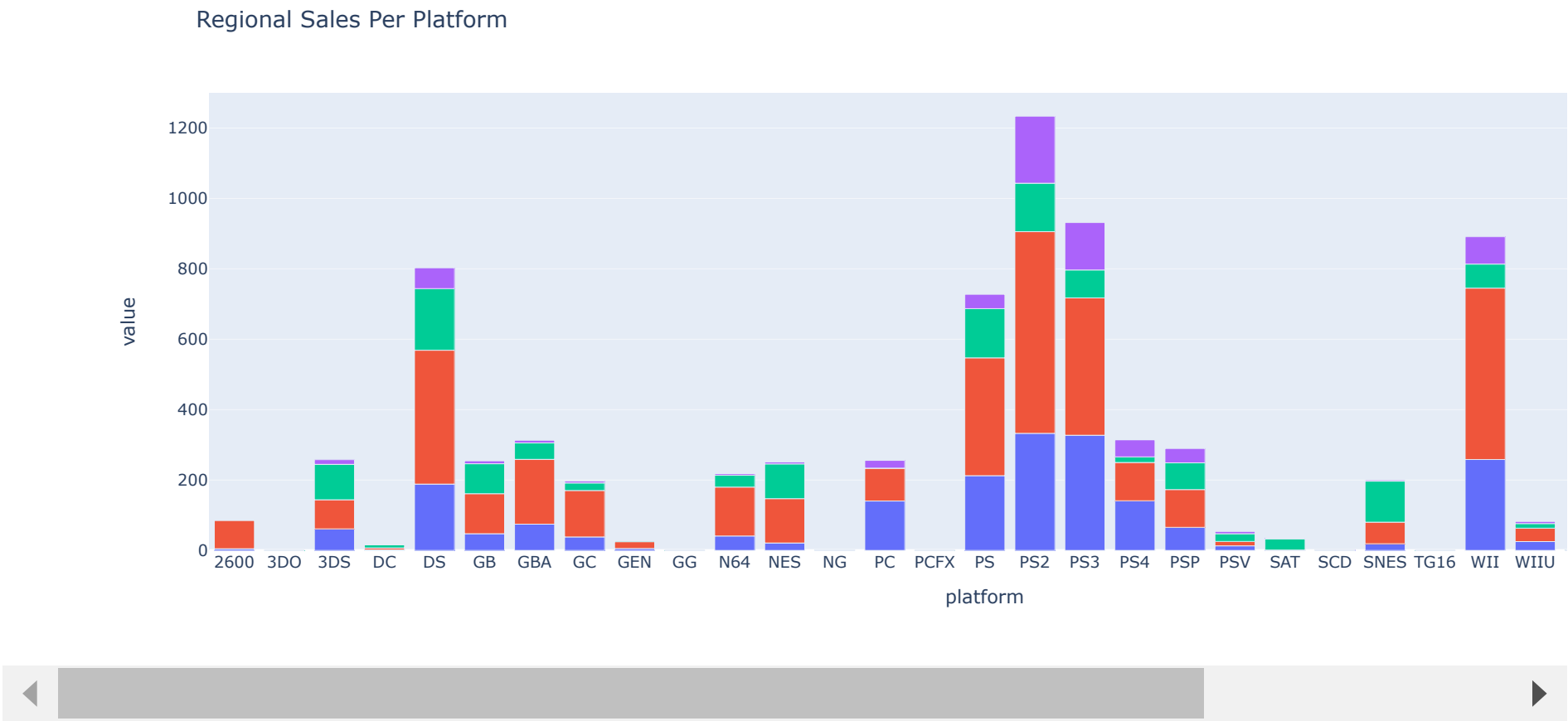
We see that the PS2 was the most successful platform from 1985 to 2016. Other popular platforms are the X360, PS3, WII and DS. We also see that the upgraded version of these platforms are not yet as popular. This may be due to their competition with the predecessors, as the new versions are released about half a decade int the predecessors run. Looking to the future, one would expect the PS4 to dominate the other platforms, while the XONE, WIIU and 3DS would follow as the new generation platforms.

We will analyze the historic data for context, then analyze the most recent time periods to make predictions for 2017.

Platform Sales by Region

In [ ]:
```
# Platform sales by region, average and total
platform_sales = df.pivot_table(index='platform', values=['na_sales', 'eu_sales', 'jp_sales', 'other_sales', 'total_sales', 'average_sales'], aggfunc='sur
```

In [ ]:
```
# platform regional sales
px.bar(platform_sales, x='platform', y=['eu_sales','na_sales',
'jp_sales', 'other_sales'], width=1400, title='Regional Sales Per Platform').show()
```



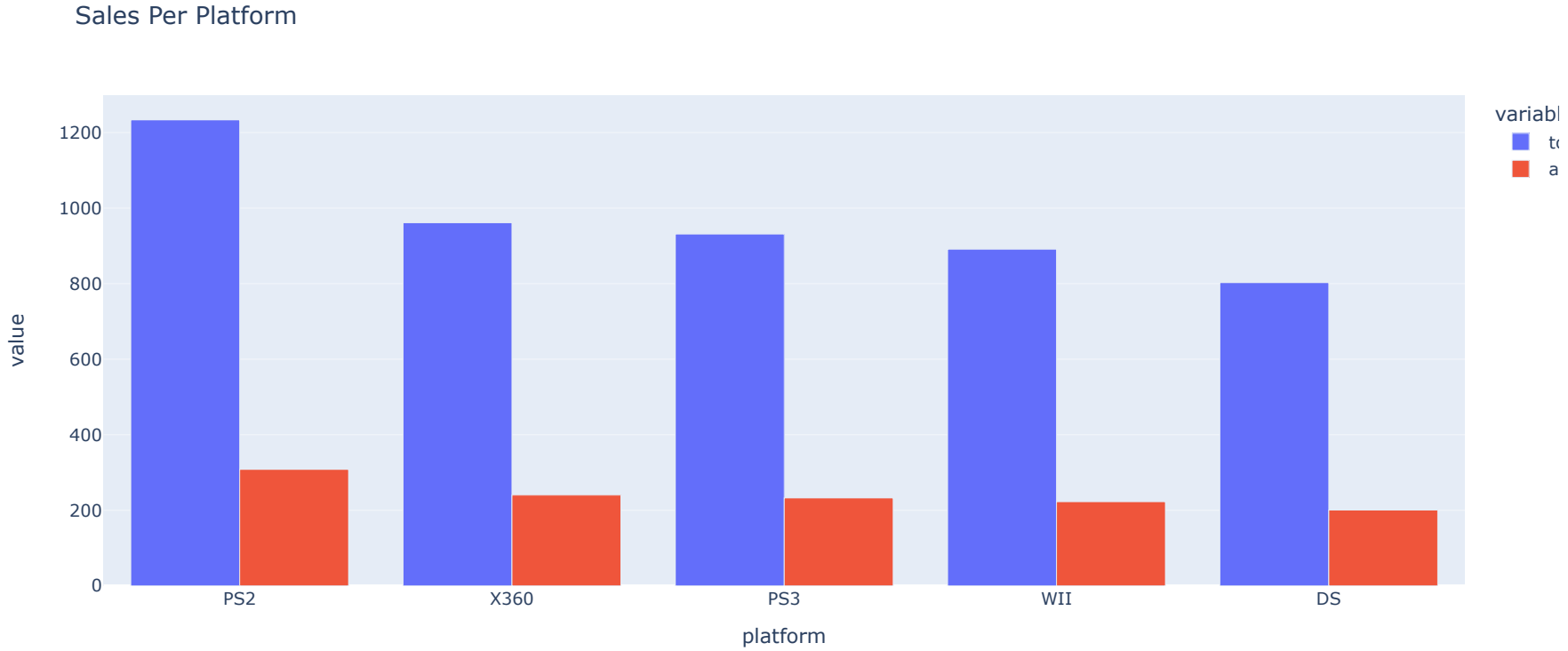
Platform sales by region breaks down the sales based on the variable regions in our dataset. We see that North America leads in the sales of most of the platforms, followed by Europe. Japan regional sales shows which platforms users prefer there.

Top Grossing Platforms

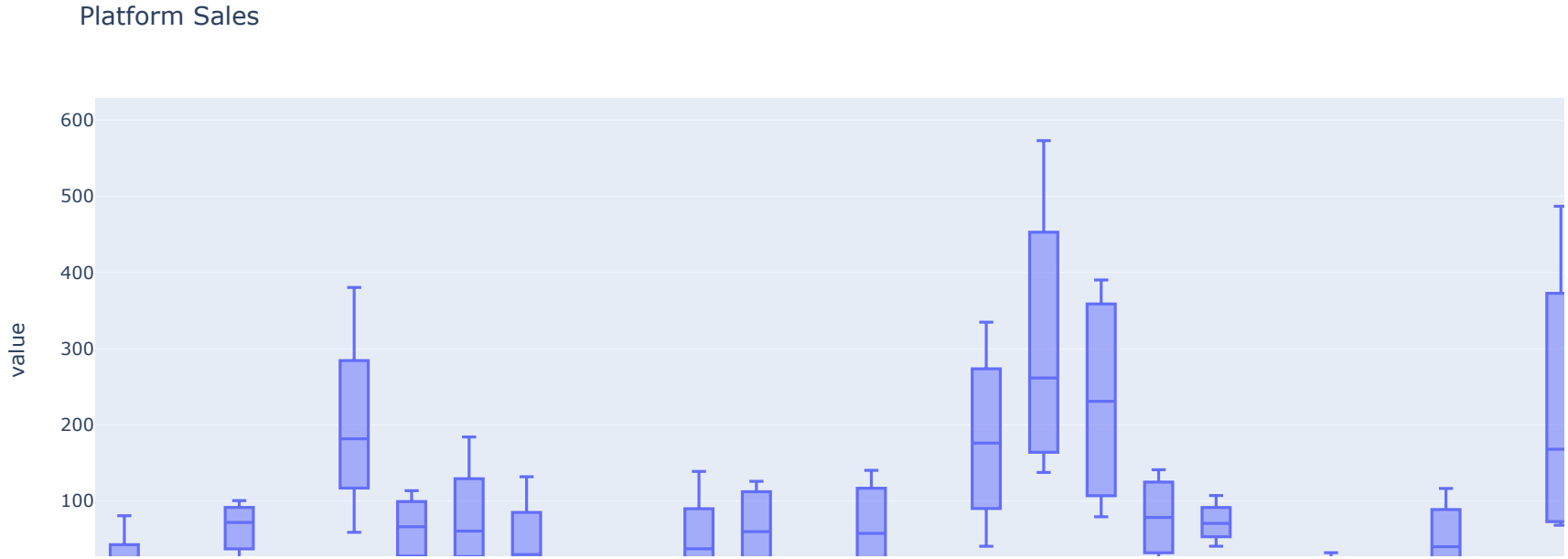
In [ ]:
```
# top grossing platforms
top_five_platforms = df.pivot_table(index='platform', values=['eu_sales','na_sales',
'jp_sales', 'other_sales', 'total_sales', 'average_sales'], aggfunc='sum').reset_index()
top_five_platforms = top_five_platforms.nlargest(5, columns='total_sales')
```

In [ ]:
```
# platform sales
px.bar(top_five_platforms, x='platform', y='total_sales',
 color='platform', title='Total Sales Per Platform').show()
```

In [ ]:
```
# platform total and average sales
px.bar(top_five_platforms, x='platform', y=['total_sales', 'average_sales'],
barmode='group', title='Sales Per Platform', width=1200).show()
```



In [ ]:
```
# platform sales boxplot
px.box(platform_sales, x='platform', y=['na_sales', 'eu_sales', 'jp_sales', 'other_sales'], title='Platform Sales').show()
```

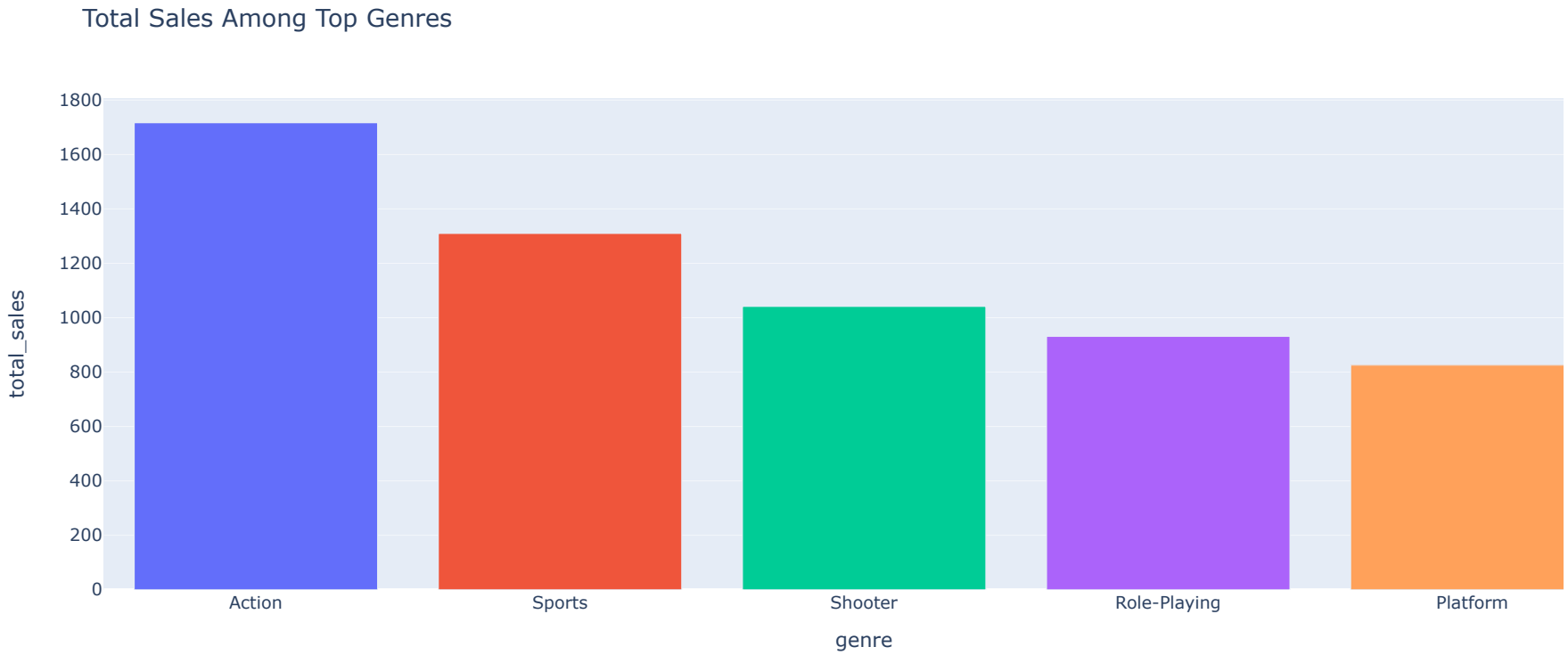


As we saw from previous results, the top grossing platforms were the PS2, X360, PS3, WII and the DS. Note that these platforms are not the latest platforms to come to market, so they have had a longer lifespan to build sales. Surprisingly, the PS2 is still the leader, despite the PS3 holding the number 3 position, and the PS4 being released. The box plots illustrate the differences in the statistics of the platform sales. We see some platforms have a wide range, while others have an extremely tight range.

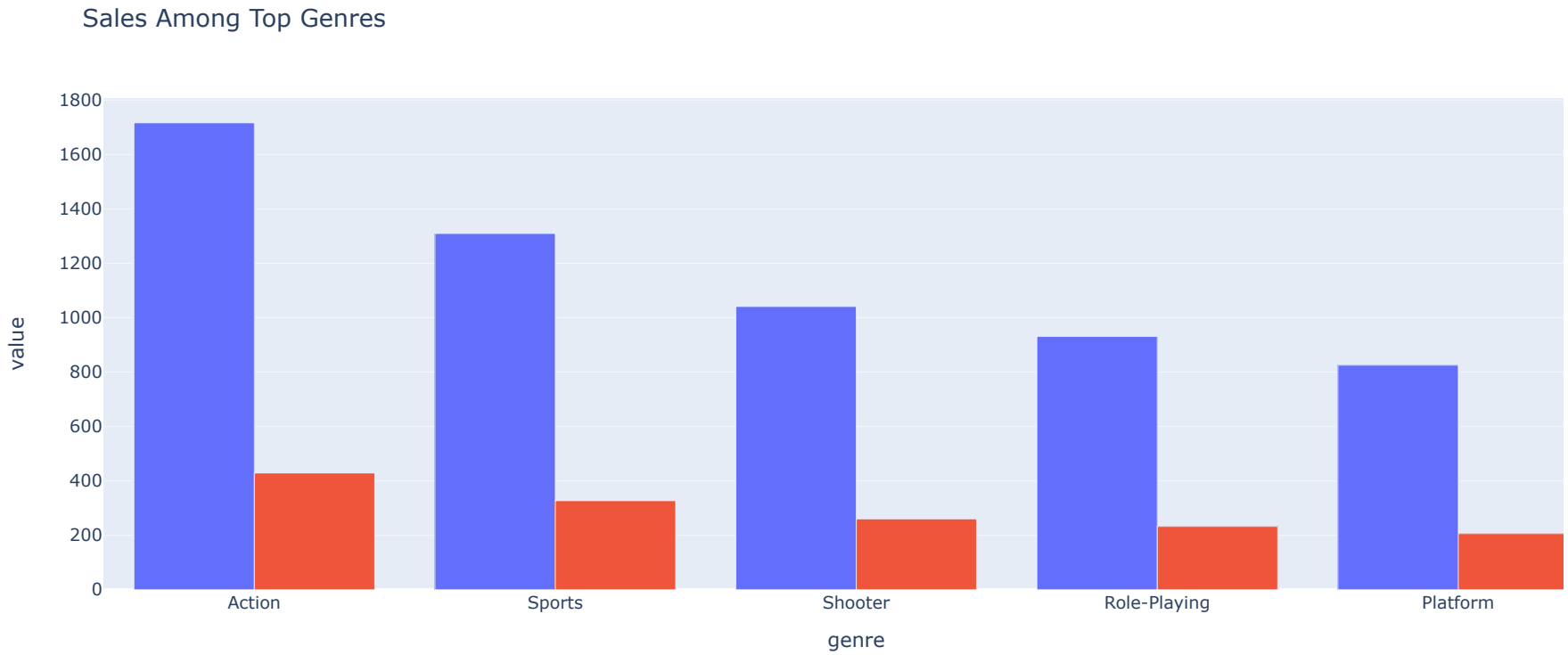
Top Grossing Genres

In [ ]:
```
# top grossing genre
top_five_genres = df.pivot_table(index='genre', values=['total_sales', 'average_sales'], aggfunc='sum').reset_index()
top_five_genres = top_five_genres.nlargest(5, columns='total_sales')
```

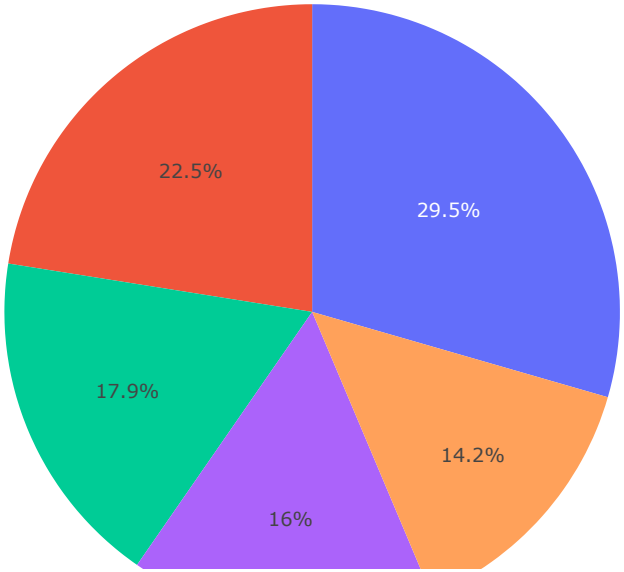
In [ ]:
```
# sales of top 5 genres
px.bar(top_five_genres, x='genre', y='total_sales',
 color='genre', width=1300, title='Total Sales Among Top Genres').show()
```



In [ ]:
```
# total and average sales, top 5 genres
px.bar(top_five_genres, x='genre', y=['total_sales', 'average_sales'],
width=1300, barmode='group', title='Sales Among Top Genres').show()
```



In [ ]:
```
# share of top 5 genres
px.pie(top_five_genres, values='total_sales', names='genre').show()
```



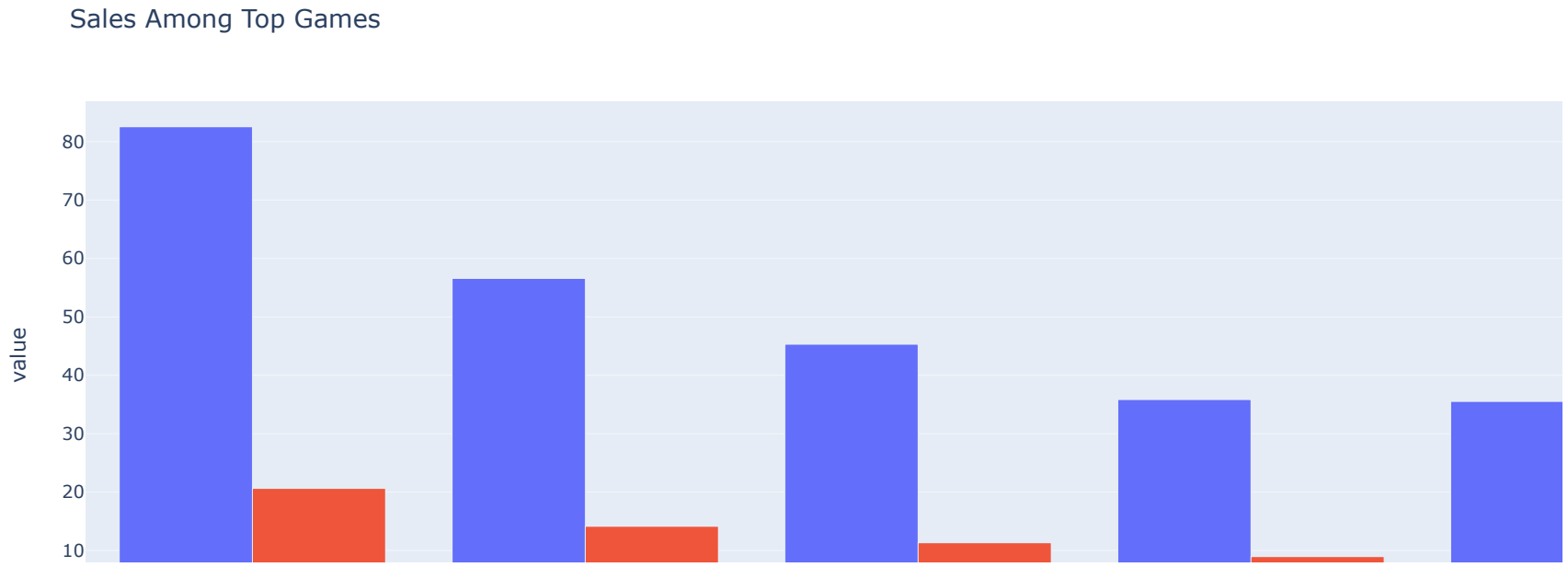
The top grossing genres were action, sports, shooter, role playing, and platform. Now of the top 5 platforms, action games take a 29.6% market share, while sports takes up 22.6%.

Top Grossing Games

In [ ]:
```
# top grossing games
top_five_games = df.pivot_table(index='name', values=['total_sales', 'average_sales'], aggfunc='sum').reset_index()
top_five_games = top_five_games.nlargest(5, columns='total_sales')
```

In [ ]:
```
# sales of top 5 games
px.bar(top_five_games, x='name', y='total_sales',
color='name', title='Sales Among Top Games').show()
```

In [ ]:
```
# total and average sales, top 5 games
px.bar(top_five_games, x='name', y=['total_sales', 'average_sales'], barmode='group', title='Sales Among Top Games').show()
```



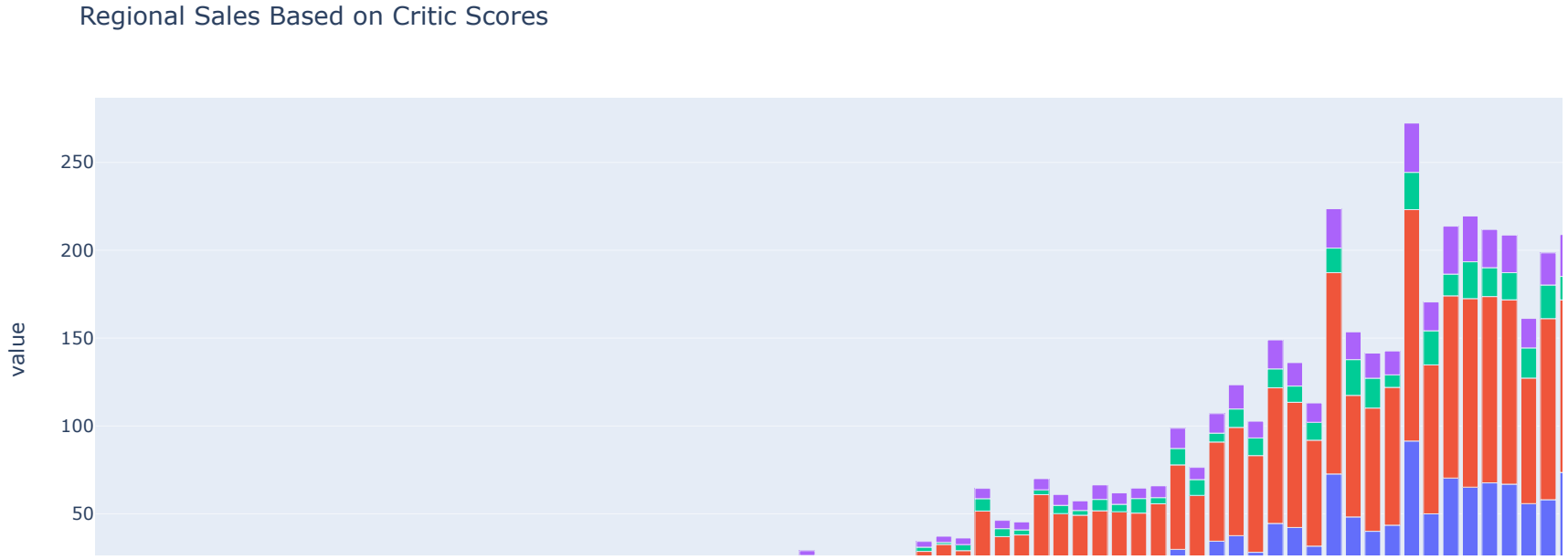
The top grossing games of this time frame are WII sports, GTA V, Super Mario Bros, Tetris, and Mario Kart WII. These entries explain why the WII is one of the top 5 platforms by sales. This is also significant, as WII games are not played across platforms. WII sports being a sports genre game supports the conclusion of sports being one of the top selling genres. Since GTA V is an action game, this supports that genre being in the top five as well.

Sales Based on Critic Scores

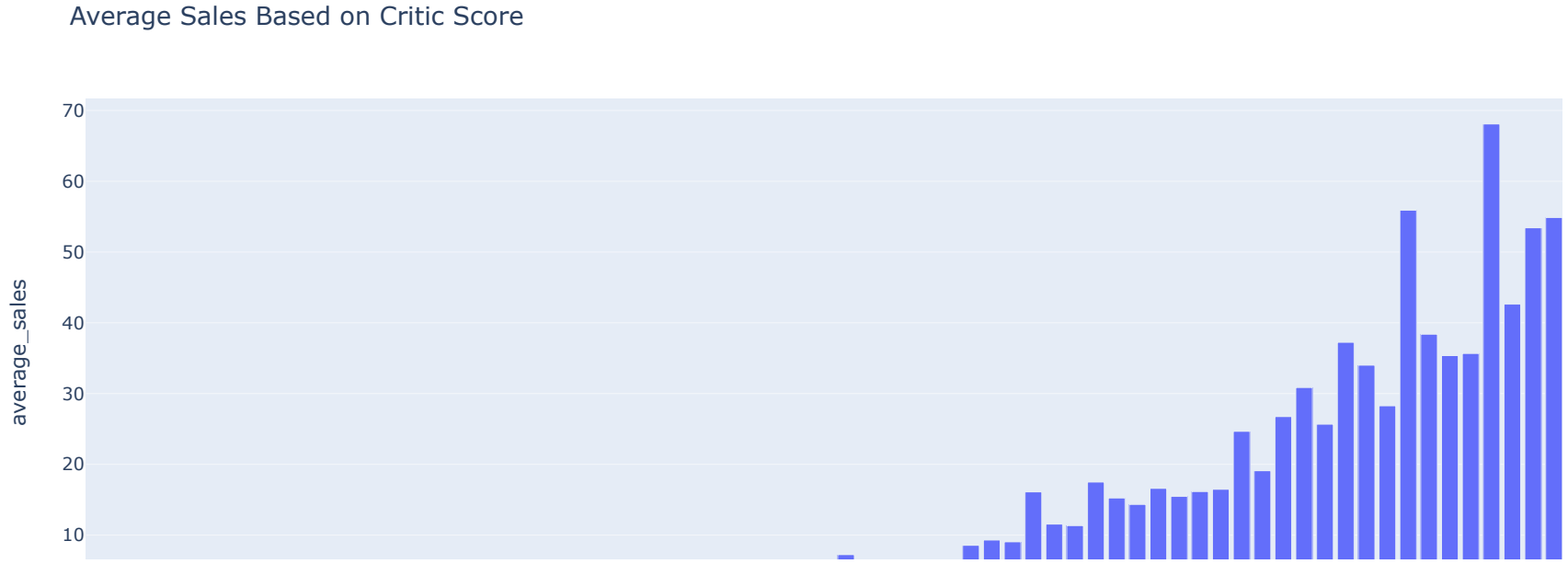
In [ ]:
```
# Sales based on critic scores
critic_sales = df.pivot_table(index='critic_score', values=['eu_sales','na_sales',
'jp_sales', 'other_sales', 'total_sales', 'average_sales'], aggfunc='sum').reset_index()
```

In [ ]:
```
# top grossing critic score
top_five_critic = critic_sales.nlargest(5, columns=['total_sales', 'average_sales'])
```

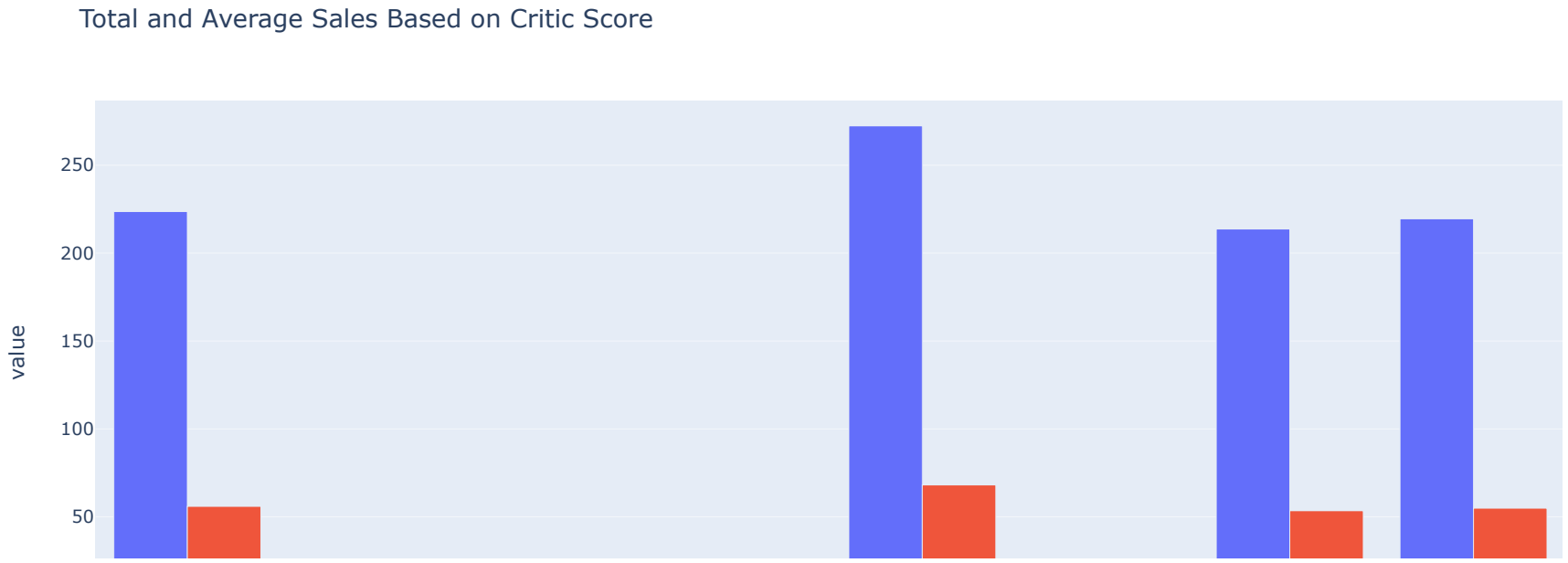
In [ ]:
```
# regional sales, critic scores
px.bar(critic_sales, x='critic_score', y=['eu_sales','na_sales',
'jp_sales', 'other_sales'], title='Regional Sales Based on Critic Scores').show()
```



In [ ]:
```
# average sales, critic scores
px.bar(critic_sales, x='critic_score', y='average_sales', title='Average Sales Based on Critic Score').show()
```



In [ ]:
```
# total and average sales, critic scores
px.bar(top_five_critic, x='critic_score', y=['total_sales', 'average_sales'], barmode='group',
    title='Total and Average Sales Based on Critic Score').show()
```



In [ ]:
```
# correlation of sales and critic scores
df[['critic_score', 'eu_sales', 'jp_sales', 'na_sales', 'total_sales']].corr()
```

Out[ ]:

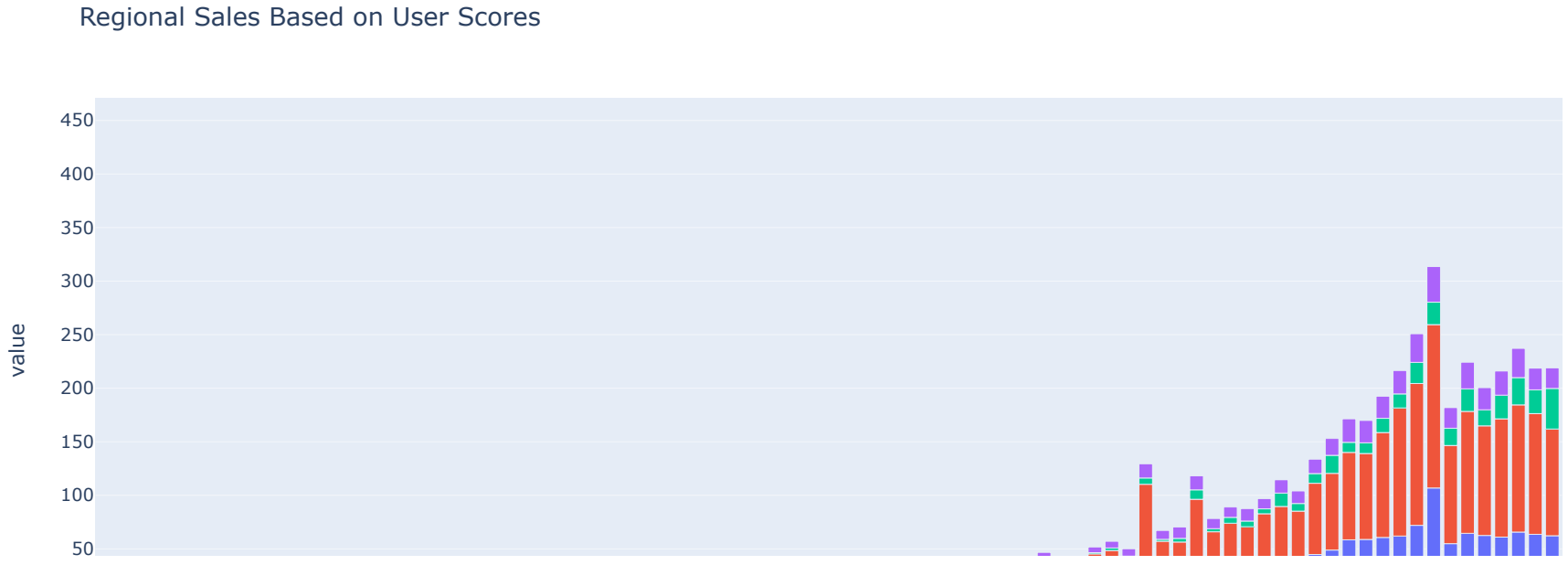
	critic_score	eu_sales	jp_sales	na_sales	total_sales
critic_score	1.000000	0.221330	0.153510	0.240150	0.245414
eu_sales	0.221330	1.000000	0.435892	0.766545	0.901673
jp_sales	0.153510	0.435892	1.000000	0.451159	0.613303
na_sales	0.240150	0.766545	0.451159	1.000000	0.941241
total_sales	0.245414	0.901673	0.613303	0.941241	1.000000

We see the bar graph of total sales skewed to the left, and distributed around a critic score of 8. The data suggest critic scores are a poor determinate of total sales. We see that the games with the highest scores, do not have the highest sales. The data shows us that games with a critic score of 8, have the highest total sales. Yet, games with lower critic scores do have lower sales. Correlation data confirms a weak correlation between critic score and sales.

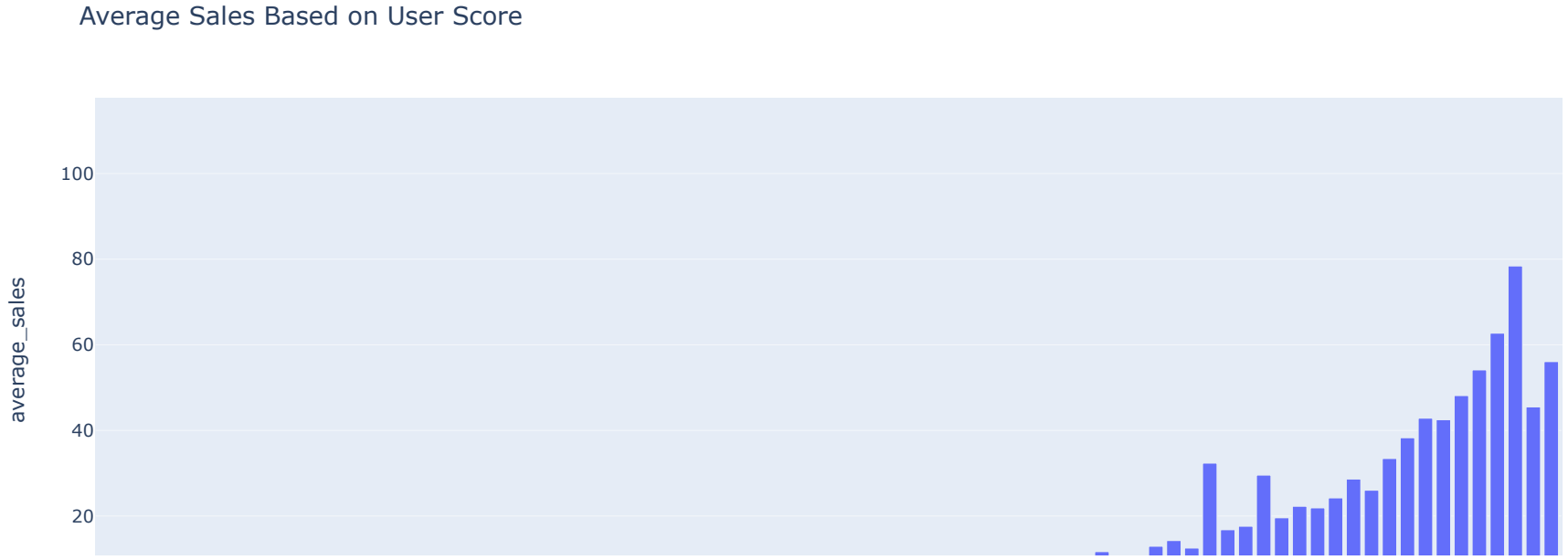
Sales Based on User Scores

In [ ]:
```
# Sorting by user score, then getting data on sales
user_sales = df.pivot_table(index='user_score', values=['eu_sales','na_sales',
'jp_sales', 'other_sales', 'total_sales', 'average_sales'], aggfunc='sum').reset_index()
```

In [ ]:
```
#regional sales
fig = px.bar(user_sales, x='user_score', y=['eu_sales','na_sales',
'jp_sales', 'other_sales'], title='Regional Sales Based on User Scores', )

fig.update_xaxes(range=[0.1, 10])
fig.show()
```

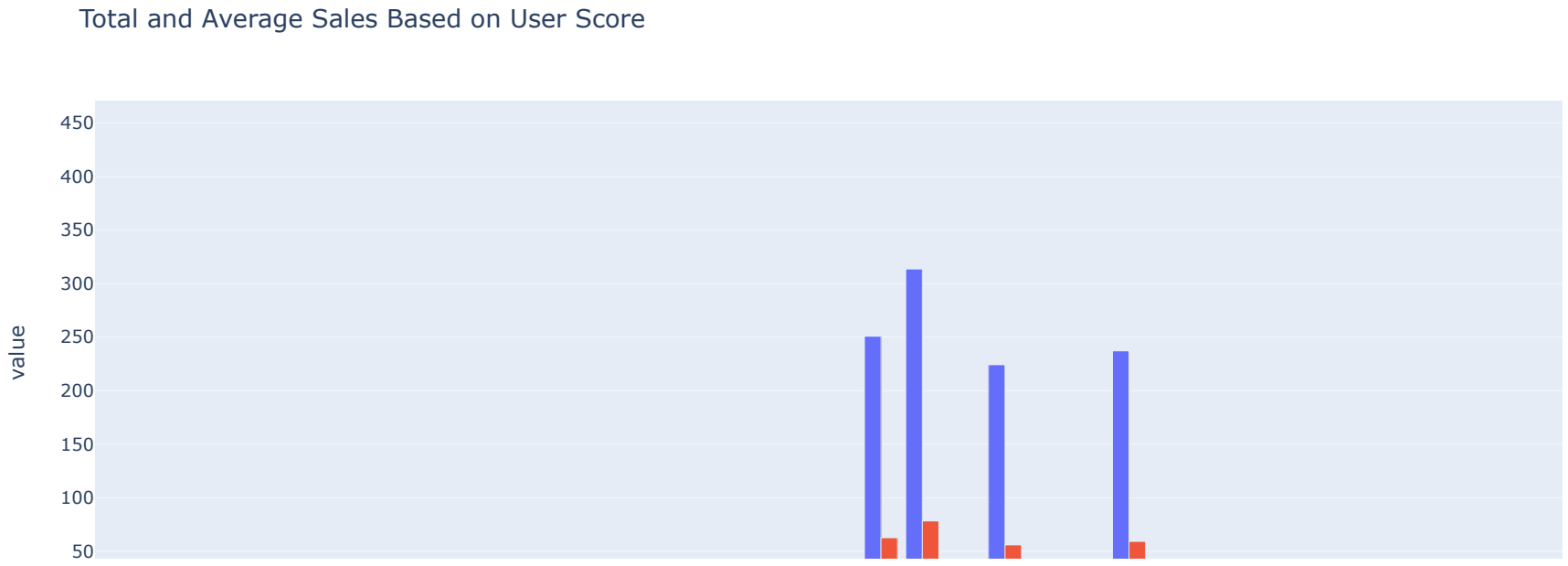


In [ ]:
```
# average sales
fig = px.bar(user_sales, x='user_score', y='average_sales', title='Average Sales Based on User Score')

fig.update_xaxes(range=[0.1, 10])
fig.show()
```



In [ ]:
```
# top grossing user score
top_five_user = user_sales.nlargest(5, columns=['total_sales', 'average_sales'])
```

In [ ]:
```
# sales of top scores
fig = px.bar(top_five_user, x='user_score', y=['total_sales', 'average_sales'], barmode='group',
title='Total and Average Sales Based on User Score')

fig.update_xaxes(range=[6, 10])
fig.show()
```



In [ ]:
```
# correlation of user score and sales
df[['user_score', 'eu_sales', 'jp_sales', 'na_sales', 'total_sales']].corr()
```

Out[ ]:

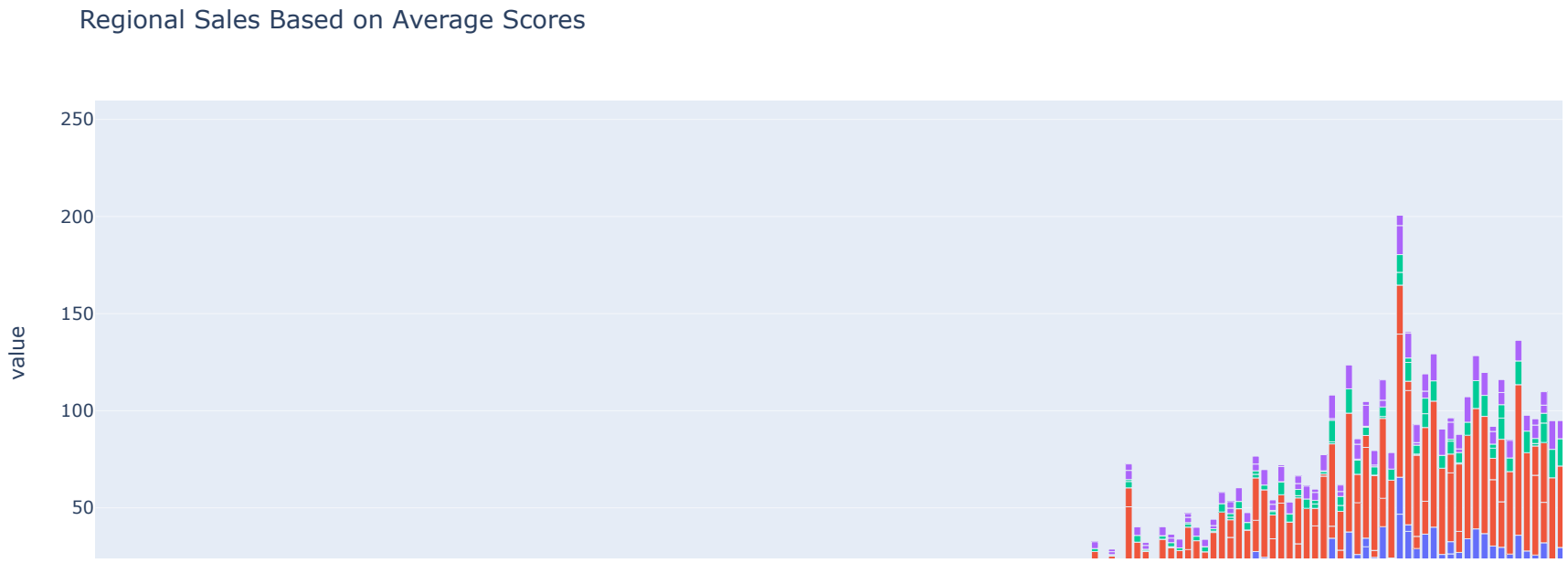
	user_score	eu_sales	jp_sales	na_sales	total_sales
user_score	1.000000	0.144300	0.142276	0.153533	0.165091
eu_sales	0.144300	1.000000	0.435892	0.766545	0.901673
jp_sales	0.142276	0.435892	1.000000	0.451159	0.613303
na_sales	0.153533	0.766545	0.451159	1.000000	0.941241
total_sales	0.165091	0.901673	0.613303	0.941241	1.000000

We see the same trend in user scores and critic scores, when looking at sales data. We see a left skewed distribution around a score of 8. Then, higher rated games do not show higher total sales. Games with the user score of 8 show the highest total sales. Correlation confirms a weak correlation between user score and sales.

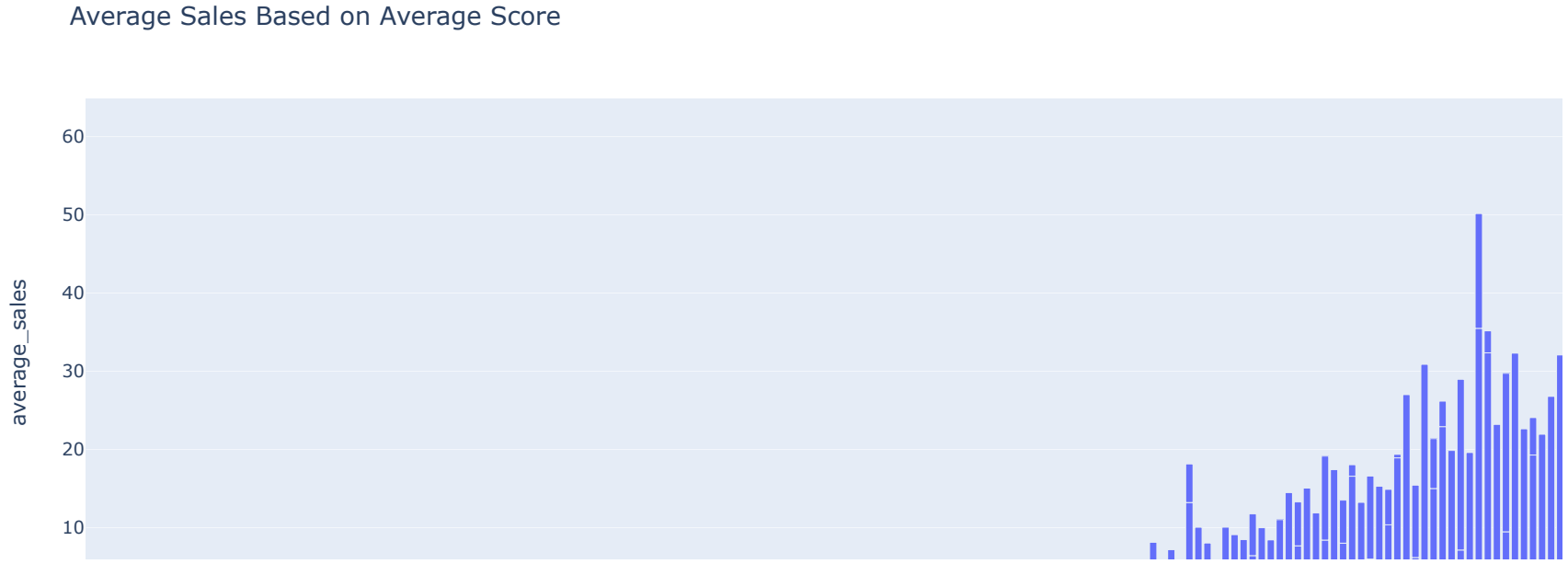
Sales Based on Average Scores

In [ ]:
```
# Sorting data by average score, then looking at sales
average_score_sales = df.pivot_table(index='average_score', values=['eu_sales','na_sales',
'jp_sales', 'other_sales', 'total_sales', 'average_sales'], aggfunc='sum').reset_index()
```

In [ ]:
```
# regional sales and average scores
fig = px.bar(average_score_sales, x='average_score', y=['eu_sales','na_sales',
'jp_sales', 'other_sales'], title='Regional Sales Based on Average Scores')

fig.update_xaxes(range=[0.1, 10])
fig.show()
```



In [ ]:
```
# average sales, average score
fig = px.bar(average_score_sales, x='average_score', y='average_sales', title='Average Sales Based on Average Score')

fig.update_xaxes(range=[0.1, 10])
fig.show()
```



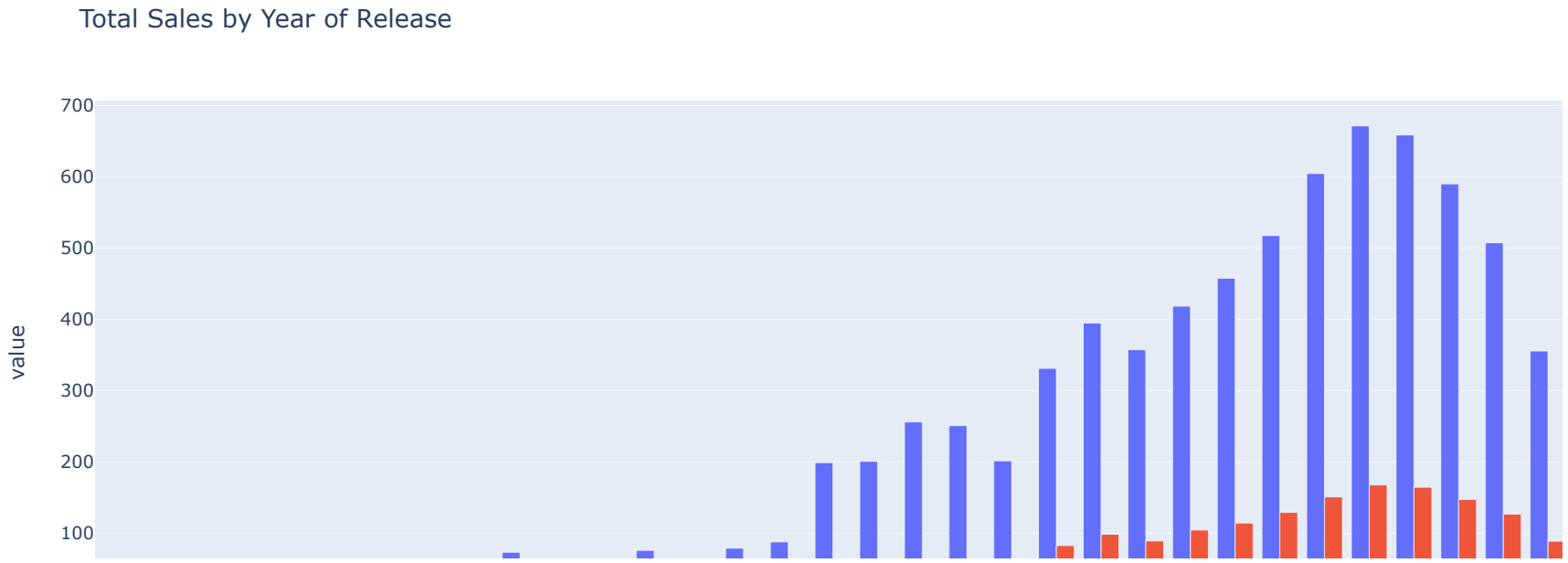
Taking the average of user and critic scores slightly changes the top selling score to a 7.8. The shape and distribution of the sales stays mostly the same.

Yearly Sales

In [ ]:
```
# Dataframe of sales based on release year
yearly_sales = df.groupby('year_of_release')['total_sales', 'average_sales'].sum().sort_values(by='total_sales', ascending=False).reset_index()
```

```
C:\Users\XIX\AppData\Local\Temp\ipykernel_23028\1022505698.py:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [ ]:
```
# total sales, year of release
px.bar(yearly_sales, x='year_of_release', y=['total_sales', 'average_sales'], title='Total Sales by Year of Release', barmode='group').show()
```

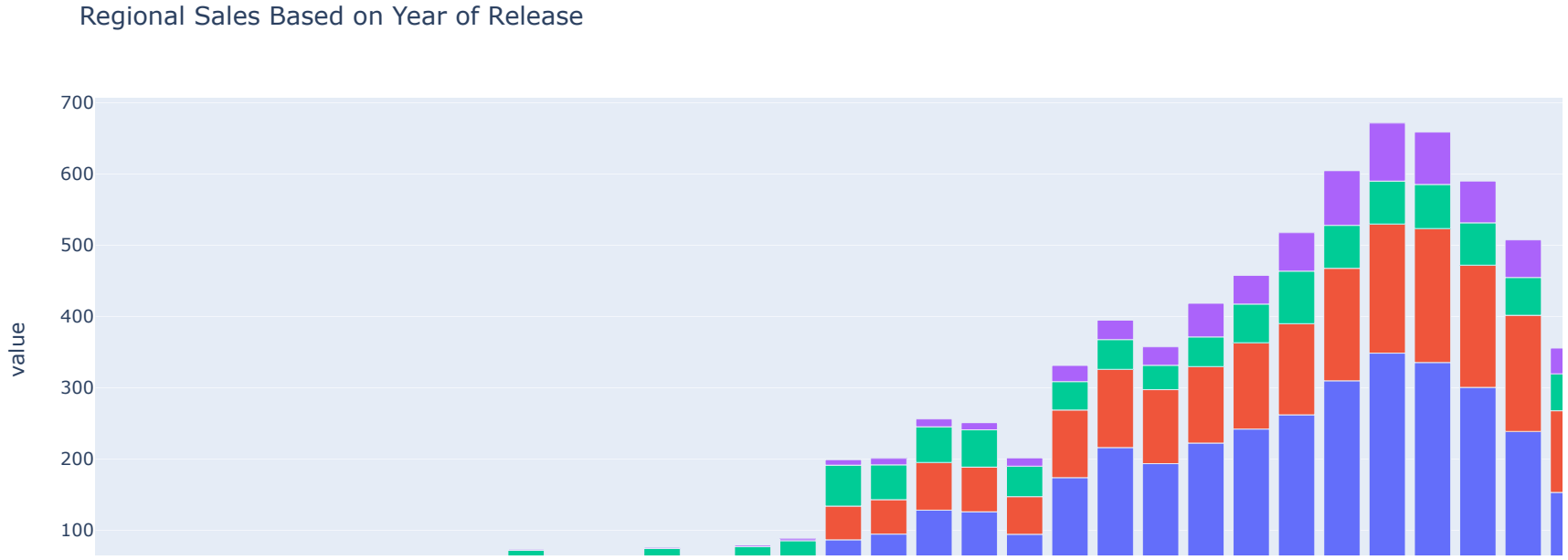


We see how sales are generally distributed from 2001 to 2013. The total sales rose from 1996 to its peak in 2008. Since 2008, we have seen a steady decline in sales, back to the levels of the late 80's.

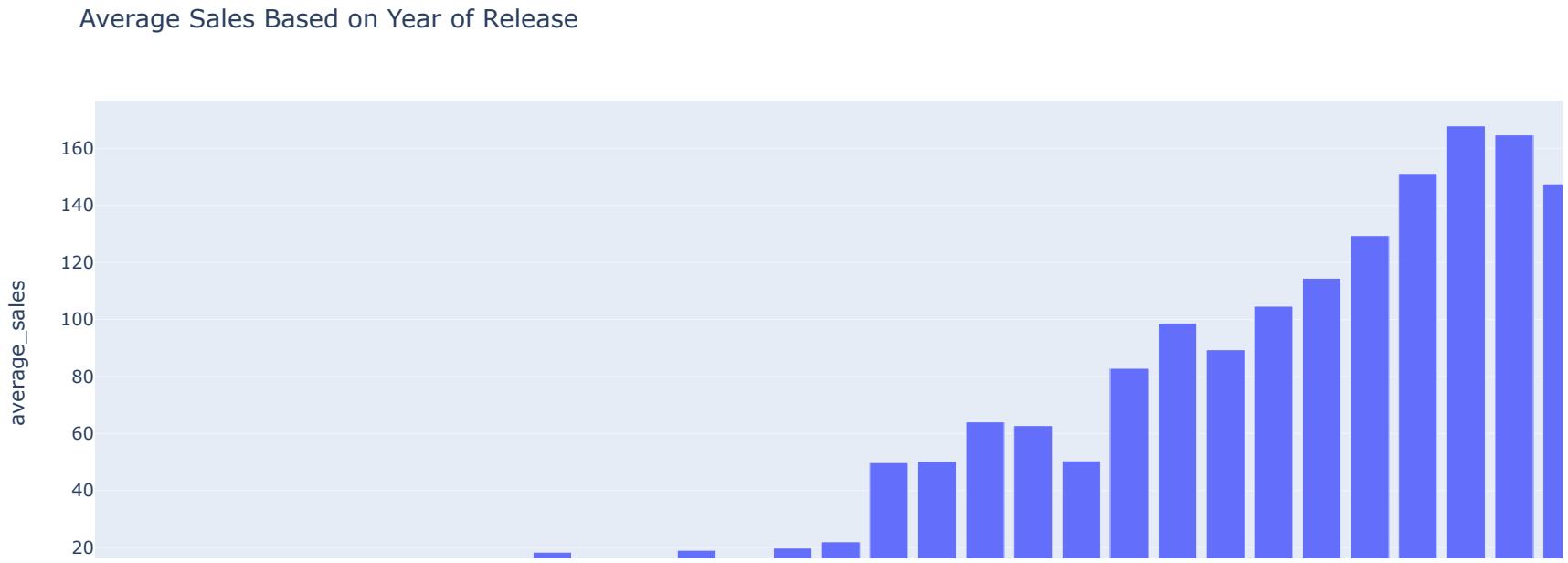
Yearly Sales Based on Region

In [ ]:
```
# yearly regional sales pivot table
yearly_region_sales = df.pivot_table(index='year_of_release', values=['na_sales', 'eu_sales', 'jp_sales', 'other_sales', 'total_sales', 'average_sales'],
```

In [ ]:
```
# plot pivot table
px.bar(yearly_region_sales, x='year_of_release',
y=['na_sales', 'eu_sales', 'jp_sales', 'other_sales'], title='Regional Sales Based on Year of Release').show()
```



In [ ]:
```
# plot pivot table, average sales
px.bar(yearly_region_sales, x='year_of_release',
y='average_sales', title='Average Sales Based on Year of Release').show()
```



Customers in North America tend to buy more games than customers of the other regions, generally throughout time. Sales in the Japanese region was most significant in the early 90's. The European region has seen a boost in sales starting from the mid 90's.

Lifespan of the Platforms

In [ ]:
```
# lifespan of the different platforms
platform_lifespan = df.groupby('platform')['year_of_release'].value_counts()
platform_lifespan = pd.DataFrame(platform_lifespan)
```

In [ ]:
```
# reset the index for cleaner dataframe
platform_years = df.groupby(['platform', 'year_of_release'])['total_sales', 'average_sales'].count().reset_index(level=0)
platform_years = platform_years.reset_index()
```

```
C:\Users\XIX\AppData\Local\Temp\ipykernel_23028\3156860180.py:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [ ]:
```
# Make a pivot table of platform and years of release, with total sales values
platform_pivot = platform_years.pivot_table(index='platform', columns='year_of_release', values='total_sales',)
```

In [ ]:
```
# filter pivot for recent years
relevant_pp = platform_pivot.loc[:, 2013:2016].dropna()
```

In [ ]:
```
# Adding a lifespan count column
relevant_pp['count'] = platform_pivot.loc[:, 2013:2016].dropna().sum(axis=1)
```

In [ ]:
```
# Extracting the count colum to show lifespan of the platforms, descending order
platform_count_years = relevant_pp.count(axis=1)
platform_count_years = platform_count_years.sort_values(ascending=False)
platform_count_years.reset_index()
```

Out[ ]:

	platform	0
0	3DS	5
1	PC	5
2	PS3	5
3	PS4	5
4	PSV	5
5	WII	5
6	WIIU	5
7	X360	5
8	XONE	5

In [ ]:
```
# statistical overview
platform_count_years.describe()
```

Out[ ]:
```
count    9.0
mean     5.0
std      0.0
min      5.0
25%      5.0
50%      5.0
75%      5.0
max      5.0
dtype: float64
```

The platform with the longest lifespan, as of 2016, is the PC. This is intuitive, as the PC has been around for a long time, and so have PC games. This is an outlier, as most platforms now have been around for 5 years. When looking at data from 2013-2016, the median and mean lifespan is 5 years. Therefore, the measure of a successful platform is one that has sales for at least 5 years. Furthermore, we can generally expect a successful platform to survive more than 5 years.

Data for every year is not significant for what we are trying to achieve.

Top Platforms from 2000-2016

In [ ]:
```
# Filtering platforms by choices
platform_pivot_filtered = platform_pivot.loc[['PS2', 'PS3', 'WII', 'X360', 'DS']].dropna(axis=1, how='all')
```

```
platform_pivot_filtered
```

Out[ ]:

year_of_release	1985	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
platform																		
PS2	NaN	82.0	185.0	280.0	256.0	259.0	260.0	259.0	214.0	191.0	96.0	38.0	7.0	NaN	NaN	NaN	NaN	NaN
PS3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	27.0	90.0	138.0	162.0	181.0	215.0	147.0	126.0	108.0	73.0	38.0
WII	NaN	NaN	NaN	NaN	NaN	NaN	NaN	44.0	185.0	282.0	325.0	253.0	143.0	31.0	12.0	6.0	4.0	1.0
X360	NaN	NaN	NaN	NaN	NaN	NaN	18.0	93.0	123.0	146.0	172.0	182.0	206.0	106.0	75.0	63.0	35.0	13.0
DS	1.0	NaN	NaN	NaN	NaN	23.0	118.0	201.0	376.0	492.0	403.0	323.0	153.0	23.0	8.0	NaN	NaN	NaN

In [ ]:
```
# filter out DS in 1985
final_platform_pivot =platform_pivot_filtered.loc[:,2000:2016]
final_platform_pivot = final_platform_pivot.reset_index()
final_platform_pivot
```

Out[ ]:

year_of_release	platform	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
0	PS2	82.0	185.0	280.0	256.0	259.0	260.0	259.0	214.0	191.0	96.0	38.0	7.0	NaN	NaN	NaN	NaN	NaN
1	PS3	NaN	NaN	NaN	NaN	NaN	NaN	27.0	90.0	138.0	162.0	181.0	215.0	147.0	126.0	108.0	73.0	38.0
2	WII	NaN	NaN	NaN	NaN	NaN	NaN	44.0	185.0	282.0	325.0	253.0	143.0	31.0	12.0	6.0	4.0	1.0
3	X360	NaN	NaN	NaN	NaN	NaN	18.0	93.0	123.0	146.0	172.0	182.0	206.0	106.0	75.0	63.0	35.0	13.0
4	DS	NaN	NaN	NaN	NaN	23.0	118.0	201.0	376.0	492.0	403.0	323.0	153.0	23.0	8.0	NaN	NaN	NaN

In [ ]:
```
# Total Sales column
final_platform_pivot['total'] = final_platform_pivot.sum(axis=1)
final_platform_pivot
```

```
C:\Users\XIX\AppData\Local\Temp\ipykernel_23028\324230748.py:2: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select onl
y valid columns before calling the reduction.
```

Out[ ]:

year_of_release	platform	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	total
0	PS2	82.0	185.0	280.0	256.0	259.0	260.0	259.0	214.0	191.0	96.0	38.0	7.0	NaN	NaN	NaN	NaN	NaN	2127.0
1	PS3	NaN	NaN	NaN	NaN	NaN	NaN	27.0	90.0	138.0	162.0	181.0	215.0	147.0	126.0	108.0	73.0	38.0	1305.0
2	WII	NaN	NaN	NaN	NaN	NaN	NaN	44.0	185.0	282.0	325.0	253.0	143.0	31.0	12.0	6.0	4.0	1.0	1286.0
3	X360	NaN	NaN	NaN	NaN	NaN	18.0	93.0	123.0	146.0	172.0	182.0	206.0	106.0	75.0	63.0	35.0	13.0	1232.0
4	DS	NaN	NaN	NaN	NaN	23.0	118.0	201.0	376.0	492.0	403.0	323.0	153.0	23.0	8.0	NaN	NaN	NaN	2120.0

In [ ]:
```
# Look at the names of the columns
final_platform_pivot.columns
```

Out[ ]:
```
Index(['platform',       2000,       2001,       2002,       2003,       2004,
             2005,       2006,       2007,       2008,       2009,       2010,
             2011,       2012,       2013,       2014,       2015,       2016,
            'total'],
      dtype='object', name='year_of_release')
```

We filter out the DS value in 1985, as that was likely the first generation of the platform. The relevant version is the current one.

In [ ]:
```
# list of years
final_platform_pivot.columns.to_list()
```

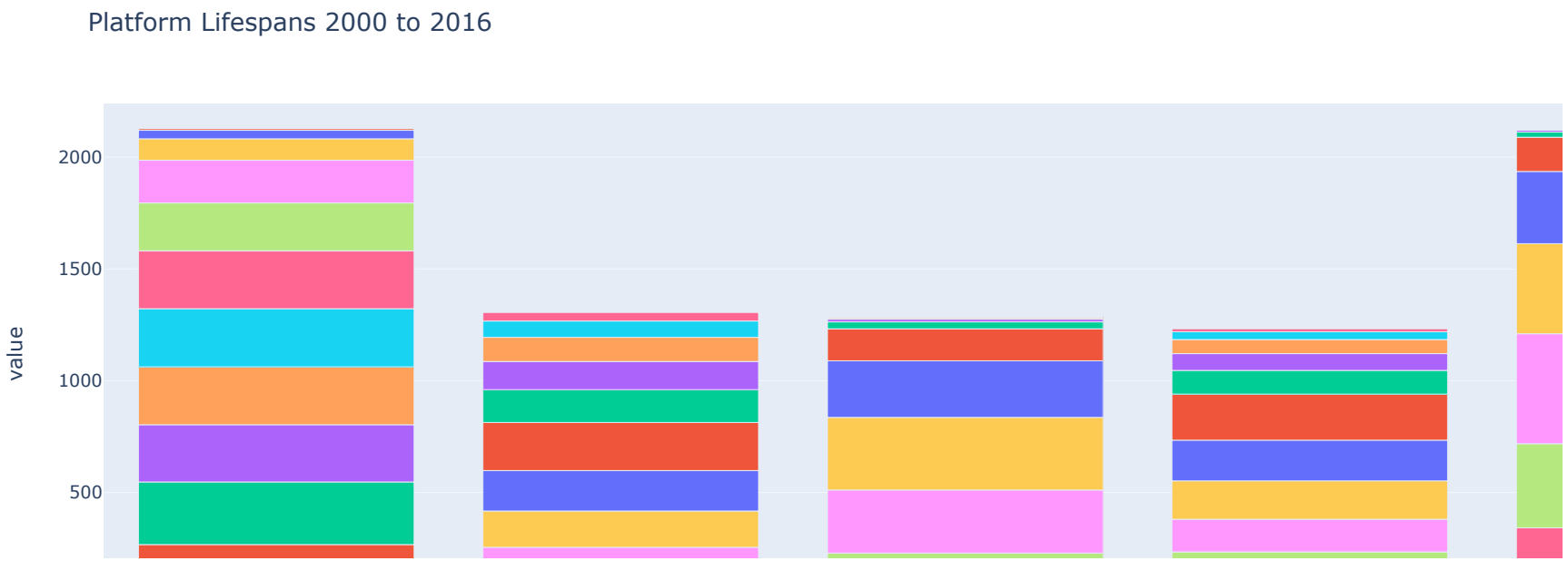
Out[ ]:
```
['platform',
 2000,
 2001,
 2002,
 2003,
 2004,
 2005,
 2006,
 2007,
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 'total']
```

Official Release Dates

```
PS2 march, 2000

PS3 november, 2006

WII november, 2006

X360 november, 2005

DS november, 2004
```

We see that the pivot table of release years correctly illustrates the lifespan of the platforms. The PS2 was released in 2000, while the PS3 and WII were released in 2006. The XBOX 360 was released in 2005, while the new generation DS was released in 2004. The total lifespan of the PS2 was 12 years, but the platform was replaced by the PS3 in 2006, which led to steady declines in PS2 game sales. This could be attributed to people upgrading their platform to the PS3. The sales of the PS3 slowly increased as the PS2 sales decreased. It took 3 years for PS3 game sales to overtake PS2 game sales. We also see that the Xbox 360 came out in 2006, as the games sales of it and the PS3 would compete with each other. The lifespan of the PS3 seems to continue beyond 2016, while the Xbox 360 started leveling off in 2012. This is interesting, as Xbox game sales peaked first, and was not overtaken by PS3 game sales until 2011. The WII platform was released in 2005, and still sees a small number of sales, only beginning to level off in 2016. The Nintendo DS was released in 2004, and saw a lifespan of 10 years, leveling off in 2012 to no game sales in 2014.

In [ ]:
```
# platform lifespans
px.bar(final_platform_pivot, x='platform',
y=[ 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016,],
title='Platform Lifespans 2000 to 2016').show()
```



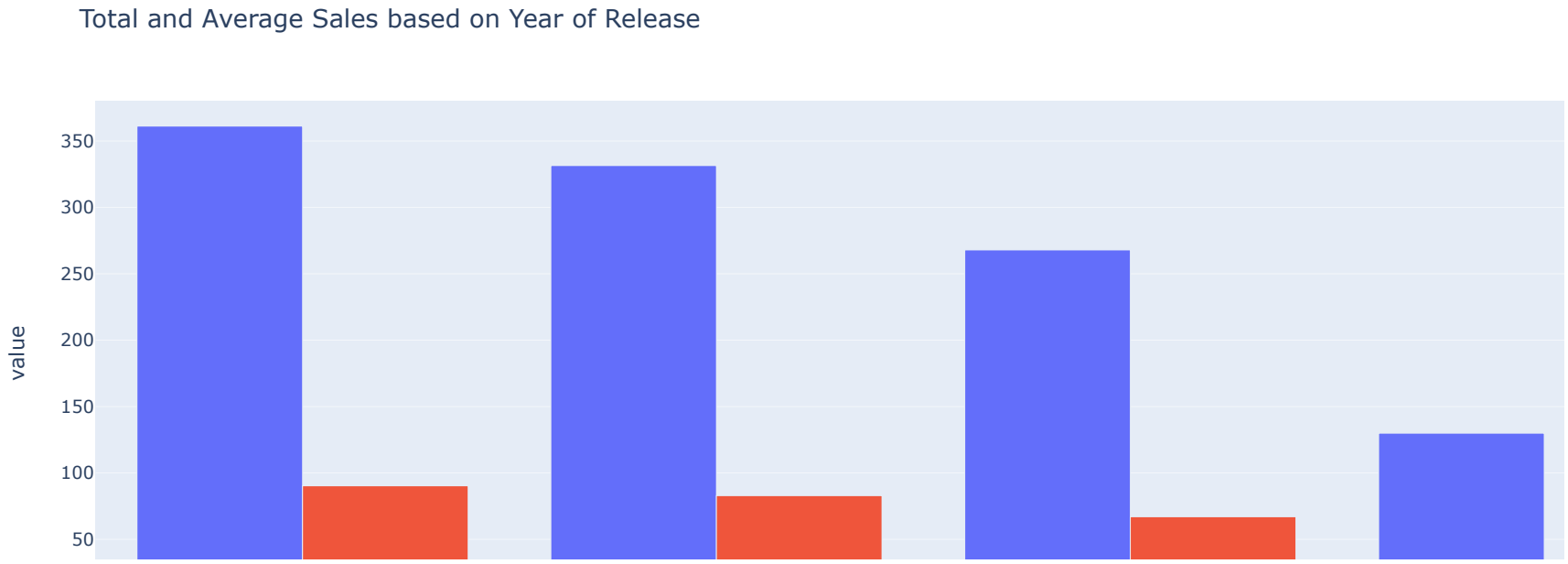
Top Platforms of Period 2013-2016

In [ ]:
```
# Filter data by last 4 years
last_period_sales = yearly_sales.sort_values('year_of_release', ascending=False).head(4)
```

In [ ]:
```
# Filter parameters
key_platforms = ('PS2', 'X360', 'PS3', 'WII', 'DS')
key_years = [2013, 2014, 2015, 2016]
```

In [ ]:
```
# Filter dataframe based on platforms and years
df_filtered = df[df['platform'].isin(key_platforms) & df['year_of_release'].isin(key_years)]
```

In [ ]:
```
# total and average sales, recent years
px.bar(last_period_sales, x='year_of_release', y=['total_sales', 'average_sales'], barmode='group',
    title='Total and Average Sales based on Year of Release').show()
```



Sales of Last Four Years

Only data from the last period is relevant, as we are attempting to make a prediction for 2017. As such, older platforms and older years are not needed in the data for overall analysis. Here, we will use the last 4 years.

Most Relevant Platforms 2013-2016

In [ ]:
```
# Most relevant platforms from recent time period
platform_pivot.loc[['3DS', 'PC', 'PS4', 'PSV', 'XONE'], 2000:2016]
```

Out[ ]:

year_of_release	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
platform																	
3DS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	116.0	93.0	91.0	80.0	86.0	46.0
PC	7.0	15.0	19.0	33.0	30.0	37.0	52.0	62.0	76.0	107.0	90.0	139.0	61.0	38.0	47.0	50.0	54.0
PS4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	16.0	75.0	137.0	164.0
PSV	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	18.0	53.0	63.0	100.0	110.0	85.0
XONE	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	19.0	61.0	80.0	87.0

In [ ]:
```
# Filtering platforms from 2013 to 2016, by total sales in period
relevant_platforms = platform_pivot.loc[:,2013:2016].dropna(how='all')
```

These are the most relevant platforms, in the most relevant periods. These platforms represent a sample of the various platforms still being played. Furthermore, these are newer generation platforms that came out in the last couple of years, and have a few more years left in their life cycles. The oldest Platform is the PC. Platforms 3DS and PSV represent handheld options, while PS4 and XONE represent the period consoles for Sony and Microsoft, respectively.

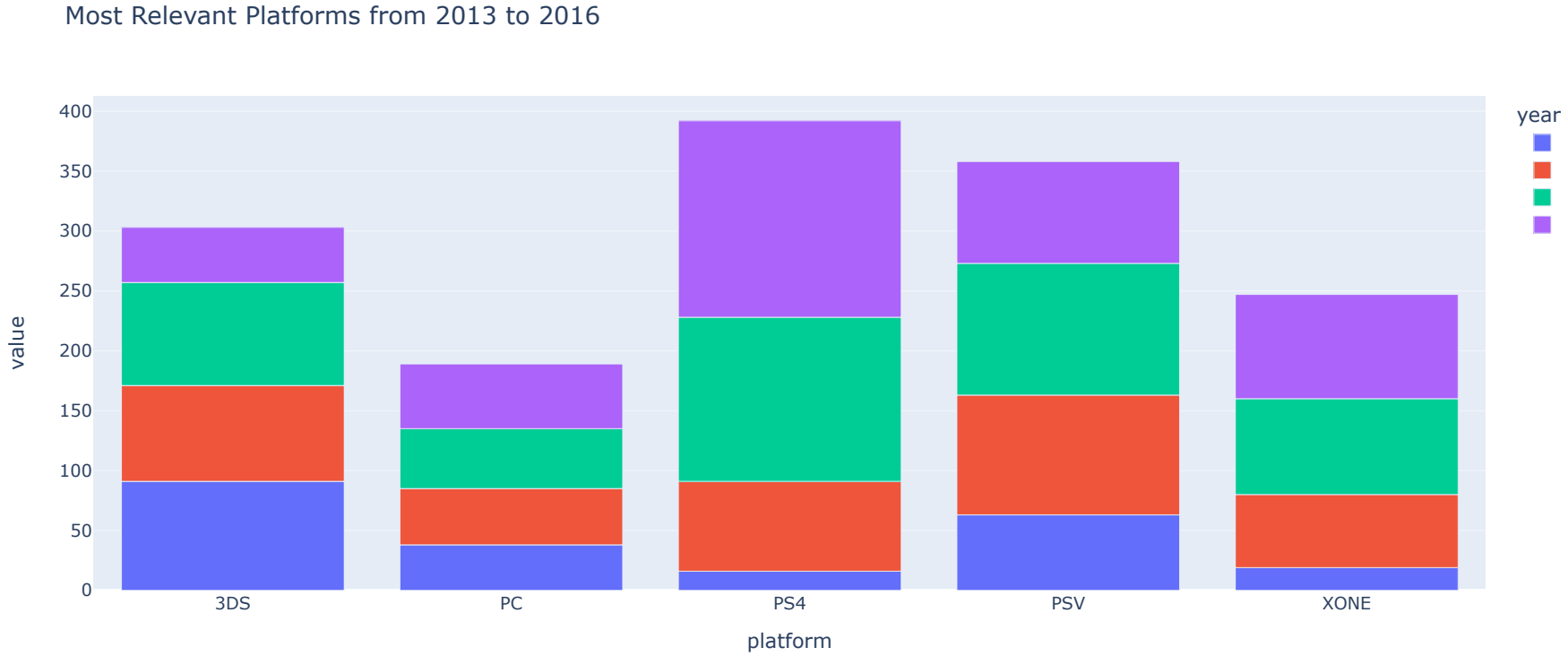
Top 5 Most Relevant Platforms from 2013 to 2016

In [ ]:
```
# Filter for 5 platforms in the time period
most_relevant_platforms = relevant_platforms.loc[['3DS', 'PC', 'PS4', 'PSV', 'XONE']]
most_relevant_platforms
```

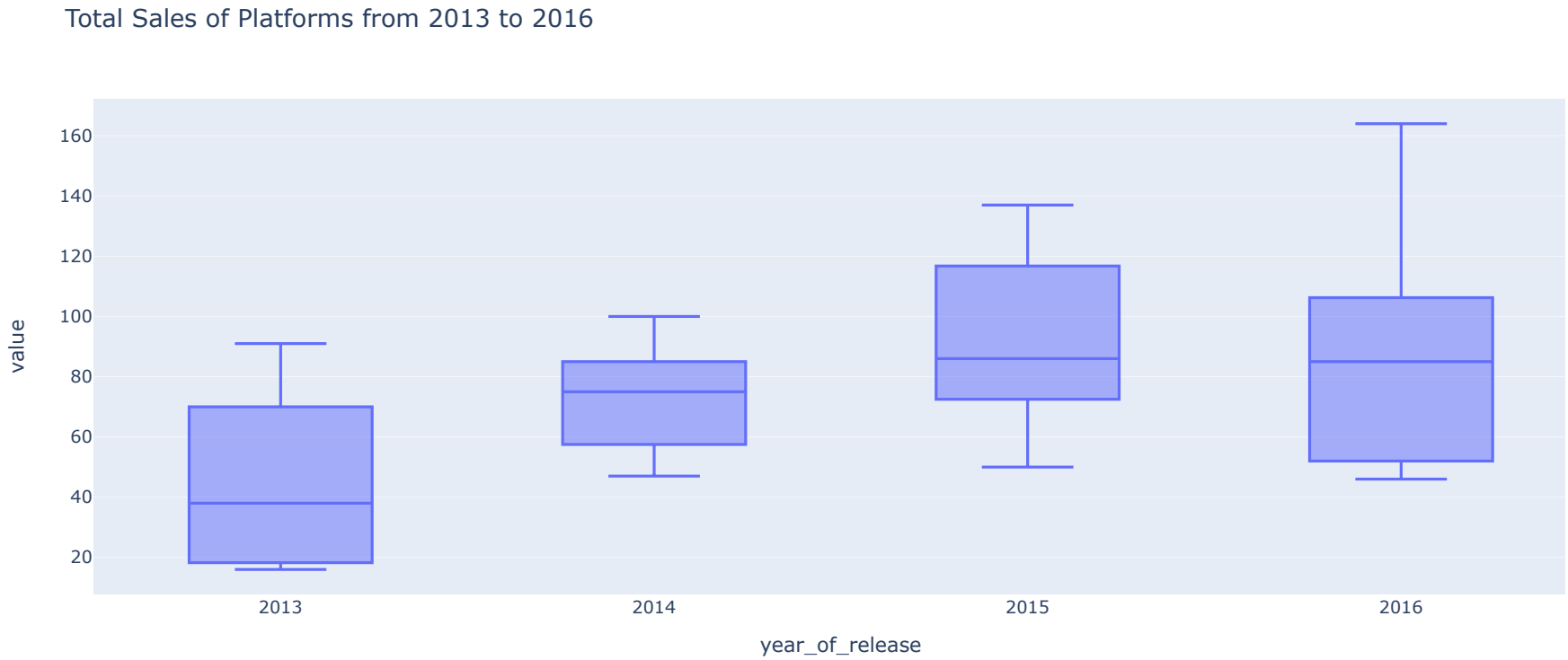
Out[ ]:

year_of_release	2013	2014	2015	2016
platform				
3DS	91.0	80.0	86.0	46.0
PC	38.0	47.0	50.0	54.0
PS4	16.0	75.0	137.0	164.0
PSV	63.0	100.0	110.0	85.0
XONE	19.0	61.0	80.0	87.0

In [ ]:
```
# platform game releases, year
px.bar(most_relevant_platforms, title='Most Relevant Platforms from 2013 to 2016', width=1200).show()
```



In [ ]:
```
# total sales boxplot
px.box(most_relevant_platforms, title='Total Sales of Platforms from 2013 to 2016', width=1200).show()
```



The most profitable platform is the PS4, and the game sales are still increasing. PC sales are the lowest, yet PC as a platform has stood the test of time and still persists. The other platforms we need to pay attention to are PSV, XONE and 3DS.

Differences in Sales by Platform 2013 to 2016

In [ ]:
```
# Take the most relevant platform dataset
df2 = most_relevant_platforms[:]
```

In [ ]:
```
# Adding a column that totals platform sales
df2['period_total'] = df2.sum(axis=1)
```

```
C:\Users\XIX\AppData\Local\Temp\ipykernel_23028\1332112292.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [ ]:
```
# List of most relevant platforms
most_relevant_platforms_list = ('3DS', 'PS4', 'PC', 'XONE', 'PSV')
```

In [ ]:
```
# Filter dataset by relevant platforms list, to get regional sales
df4 = platform_sales[platform_sales['platform'].isin(most_relevant_platforms_list)]
```

In [ ]:
```
# Sorted by EU sales
df4.sort_values(by='eu_sales')
```

Out[ ]:

	platform	average_sales	eu_sales	jp_sales	na_sales	other_sales	total_sales
20	PSV	13.4525	13.07	21.84	12.47	6.43	53.81
30	XONE	39.8300	51.59	0.34	93.12	14.27	159.32
2	3DS	64.4525	61.27	100.62	82.65	13.27	257.81
13	PC	63.9400	140.37	0.17	93.34	21.88	255.76
18	PS4	78.5350	141.09	15.96	108.74	48.35	314.14

In [ ]:
```
# Sorted by JP sales
df4.sort_values(by='jp_sales')
```

Out[ ]:

	platform	average_sales	eu_sales	jp_sales	na_sales	other_sales	total_sales
13	PC	63.9400	140.37	0.17	93.34	21.88	255.76
30	XONE	39.8300	51.59	0.34	93.12	14.27	159.32
18	PS4	78.5350	141.09	15.96	108.74	48.35	314.14
20	PSV	13.4525	13.07	21.84	12.47	6.43	53.81
2	3DS	64.4525	61.27	100.62	82.65	13.27	257.81

In [ ]:
```
# Sorted by NA sales
df4.sort_values(by='na_sales')
```

Out[ ]:

	platform	average_sales	eu_sales	jp_sales	na_sales	other_sales	total_sales
20	PSV	13.4525	13.07	21.84	12.47	6.43	53.81
2	3DS	64.4525	61.27	100.62	82.65	13.27	257.81
30	XONE	39.8300	51.59	0.34	93.12	14.27	159.32
13	PC	63.9400	140.37	0.17	93.34	21.88	255.76
18	PS4	78.5350	141.09	15.96	108.74	48.35	314.14

In [ ]:
```
# Cum sum of sales in regions
df4_sum =df4.cumsum()
df4_sum = df4_sum[4:]
df4_sum
```

Out[ ]:

	platform	average_sales	eu_sales	jp_sales	na_sales	other_sales	total_sales
30	3DSPCPS4PSVXONE	260.21	407.39	138.93	390.32	104.2	1040.84

3DS Platform

In [ ]:
```
# DS ratios of total sales
ds = pd.DataFrame((df4.iloc[0,1:] / df4_sum.iloc[0, 1:]) * 100)
ds.columns= ['DS']
ds.reset_index()
```

Out[ ]:

	index	DS
0	average_sales	24.769417
1	eu_sales	15.039643
2	jp_sales	72.424962
3	na_sales	21.174933
4	other_sales	12.735125
5	total_sales	24.769417

In [ ]:
```
# DS ratio of total sales list
ds_sales = ds['DS'].iloc[1:5].to_list()
ds_sales
```

Out[ ]:
```
[15.039642602911215, 72.42496221118549, 21.17493338798934, 12.735124760076774]
```

In [ ]:
```
# DS regional sales numbers
ds_box = pd.DataFrame(df4.iloc[0,1:])
ds_box.columns= ['3DS']
ds_box.reset_index()
```

Out[ ]:

	index	3DS
0	average_sales	64.4525
1	eu_sales	61.27
2	jp_sales	100.62
3	na_sales	82.65
4	other_sales	13.27
5	total_sales	257.81

In [ ]:
```
# DS regional sales numbers list
ds_box_sales = ds_box['3DS'].iloc[1:5].to_list()
```

PC Platform

In [ ]:
```
# PC ratios of total sales
pc = pd.DataFrame((df4.iloc[1,1:] / df4_sum.iloc[0, 1:]) * 100)
pc.columns = ['PC']
pc.reset_index()
```

Out[ ]:

	index	PC
0	average_sales	24.572461
1	eu_sales	34.455927
2	jp_sales	0.122364
3	na_sales	23.913712
4	other_sales	20.998081
5	total_sales	24.572461

In [ ]:
```
# PC ratio of total sales list
pc_sales = pc['PC'].iloc[1:5].to_list()
```

In [ ]:
```
# PC regional sales numbers
pc_box = pd.DataFrame(df4.iloc[1,1:])
pc_box.columns= ['PC']
pc_box.reset_index()
```

Out[ ]:

	index	PC
0	average_sales	63.94
1	eu_sales	140.37
2	jp_sales	0.17
3	na_sales	93.34
4	other_sales	21.88
5	total_sales	255.76

In [ ]:
```
# PC regional sales numbers  list
pc_box_sales = pc_box['PC'].iloc[1:5].to_list()
```

PS4 Platform

In [ ]:
```
# PS4 ratios of total sales
ps4 = pd.DataFrame((df4.iloc[2,1:] / df4_sum.iloc[0, 1:]) * 100)
ps4.columns = ['PS4']
ps4.reset_index()
```

Out[ ]:

	index	PS4
0	average_sales	30.181392
1	eu_sales	34.632662
2	jp_sales	11.4878
3	na_sales	27.859192
4	other_sales	46.401152
5	total_sales	30.181392

In [ ]:
```
# PS4 ratios of total sales list
ps4_sales = ps4['PS4'].iloc[1:5].to_list()
```

In [ ]:
```
# PS4 regional sales numbers
ps4_box = pd.DataFrame(df4.iloc[2,1:])
ps4_box.columns = ['PS4']
ps4_box.reset_index()
```

Out[ ]:

	index	PS4
0	average_sales	78.535
1	eu_sales	141.09
2	jp_sales	15.96
3	na_sales	108.74
4	other_sales	48.35
5	total_sales	314.14

In [ ]:
```
# PS4 regional sales numbers list
ps4_box_sales = ps4_box['PS4'].iloc[1:5].to_list()
```

PSV Platform

In [ ]:
```
# PSV ratios of total sales
psv = pd.DataFrame((df4.iloc[3,1:] / df4_sum.iloc[0, 1:]) * 100)
psv.columns = ['PSV']
psv.reset_index()
```

Out[ ]:

	index	PSV
0	average_sales	5.169863
1	eu_sales	3.208228
2	jp_sales	15.720147
3	na_sales	3.194815
4	other_sales	6.170825
5	total_sales	5.169863

In [ ]:
```
# PSV ratios of sales list
psv_sales = psv['PSV'].iloc[1:5].to_list()
```

In [ ]:
```
# PSV sales numbers
psv_box = pd.DataFrame(df4.iloc[3,1:])
psv_box.columns = ['PSV']
psv_box.reset_index()
```

Out[ ]:

	index	PSV
0	average_sales	13.4525
1	eu_sales	13.07
2	jp_sales	21.84
3	na_sales	12.47
4	other_sales	6.43
5	total_sales	53.81

In [ ]:
```
# PSV sales numbers list
psv_box_sales = psv_box['PSV'].iloc[1:5].to_list()
```

XONE Platform

In [ ]:
```
# XONE ratios of total sales
xone = pd.DataFrame((df4.iloc[4,1:] / df4_sum.iloc[0, 1:]) * 100)
xone.columns = ['XONE']
xone.reset_index()
```

Out[ ]:

	index	XONE
0	average_sales	15.306868
1	eu_sales	12.663541
2	jp_sales	0.244728
3	na_sales	23.857348
4	other_sales	13.694818
5	total_sales	15.306868

In [ ]:
```
# XONE ratios of sales list
xone_sales = xone['XONE'].iloc[1:5].to_list()
```

In [ ]:
```
# XONE regional sales
xone_box = pd.DataFrame(df4.iloc[4,1:])
xone_box.columns = ['XONE']
xone_box.reset_index()
```

Out[ ]:

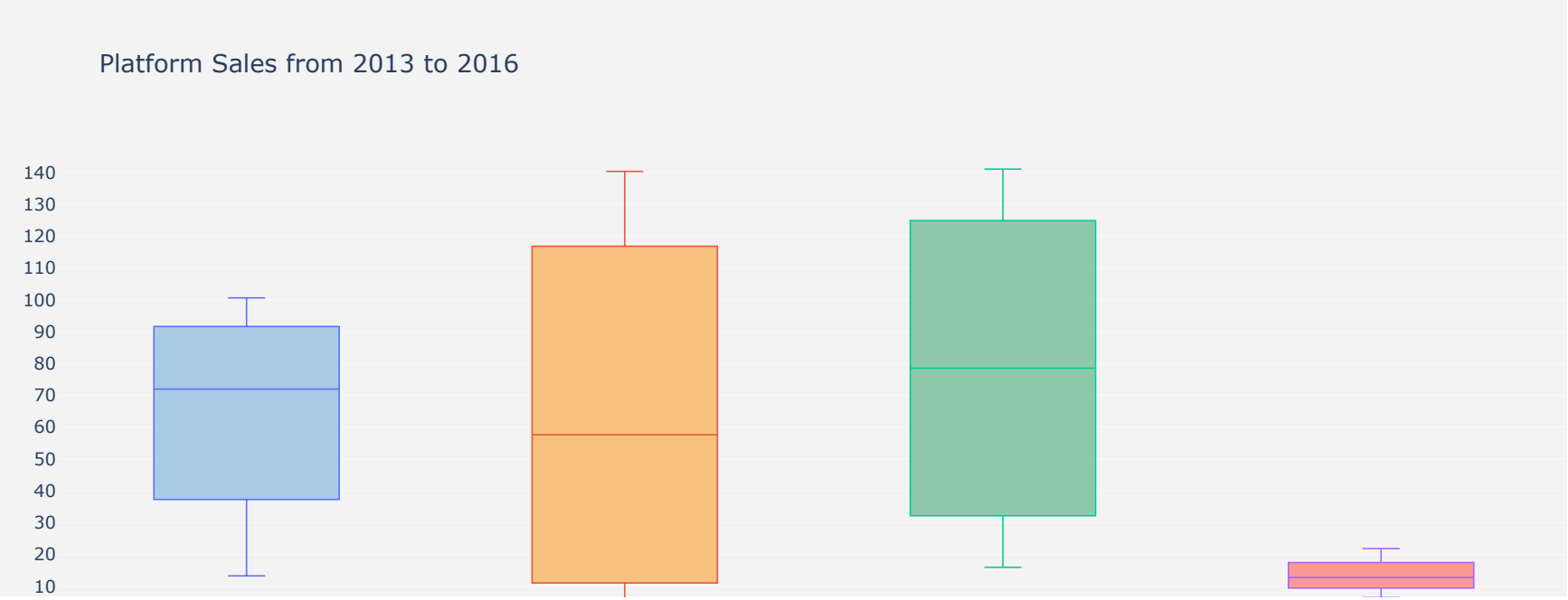
	index	XONE
0	average_sales	39.83
1	eu_sales	51.59
2	jp_sales	0.34
3	na_sales	93.12
4	other_sales	14.27
5	total_sales	159.32

In [ ]:
```
# XONE regional sales list
xone_box_sales = xone_box['XONE'].iloc[1:5].to_list()
```

Overall Platform Results

In [ ]:
```
x_data = ['3DS', 'PC', 'PS4', 'PSV', 'XONE']

y_data = [ds_box_sales , pc_box_sales, ps4_box_sales, psv_box_sales, xone_box_sales]

colors = ['rgba(93, 164, 214, 0.5)', 'rgba(255, 144, 14, 0.5)', 'rgba(44, 160, 101, 0.5)',
          'rgba(255, 65, 54, 0.5)', 'rgba(207, 114, 255, 0.5)']

fig = go.Figure()

for xd, yd, cls in zip(x_data, y_data, colors):
        fig.add_trace(go.Box(
            y=yd,
            name=xd,
            jitter=0.5,
            whiskerwidth=0.2,
            fillcolor=cls,
            marker_size=2,
            line_width=1)
        )

fig.update_layout(
    title='Platform Sales from 2013 to 2016',
    yaxis=dict(
        autorange=True,
        showgrid=True,
        zeroline=True,
        dtick=10,
        gridcolor='rgb(255, 255, 255)',
        gridwidth=1,
        zerolinecolor='rgb(255, 255, 255)',
        zerolinewidth=2,
    ),
    margin=dict(
        l=40,
        r=30,
        b=80,
```

```
        t=100,
    ),
    paper_bgcolor='rgb(243, 243, 243)',
    plot_bgcolor='rgb(243, 243, 243)',
    showlegend=False
)

fig.show()
```



Conclusions

The box plots show the differences, and similarities in the total sales of each platform from 2013 to 2016. All platforms have a fairly broad spectrum, except for the PSV, which has a tight range. Statistical testing will determine if the mean sales of the regions are different from one another.

Market Share per Platform, by Region

In [ ]:
```
# DS data merged with PC
pt_1 = ds.T.reset_index().merge(pc.T.reset_index(), how='outer')
```

```
C:\Users\XIX\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: FutureWarning:

In a future version, the Index constructor will not infer numeric dtypes when passed object-dtype sequences (matching Series behavior)
```

In [ ]:
```
# Merge again with PS4
pt_2 = pt_1.merge(ps4.T.reset_index(), how='outer')
```

```
C:\Users\XIX\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: FutureWarning:

In a future version, the Index constructor will not infer numeric dtypes when passed object-dtype sequences (matching Series behavior)
```

In [ ]:
```
# Merge again with PSV
pt_3 = pt_2.merge(psv.T.reset_index(), how='outer')
```

```
C:\Users\XIX\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: FutureWarning:

In a future version, the Index constructor will not infer numeric dtypes when passed object-dtype sequences (matching Series behavior)
```

In [ ]:
```
# Platform Market share percentages
pt_final = pt_3.merge(xone.T.reset_index(), how='outer')
pt_final
```

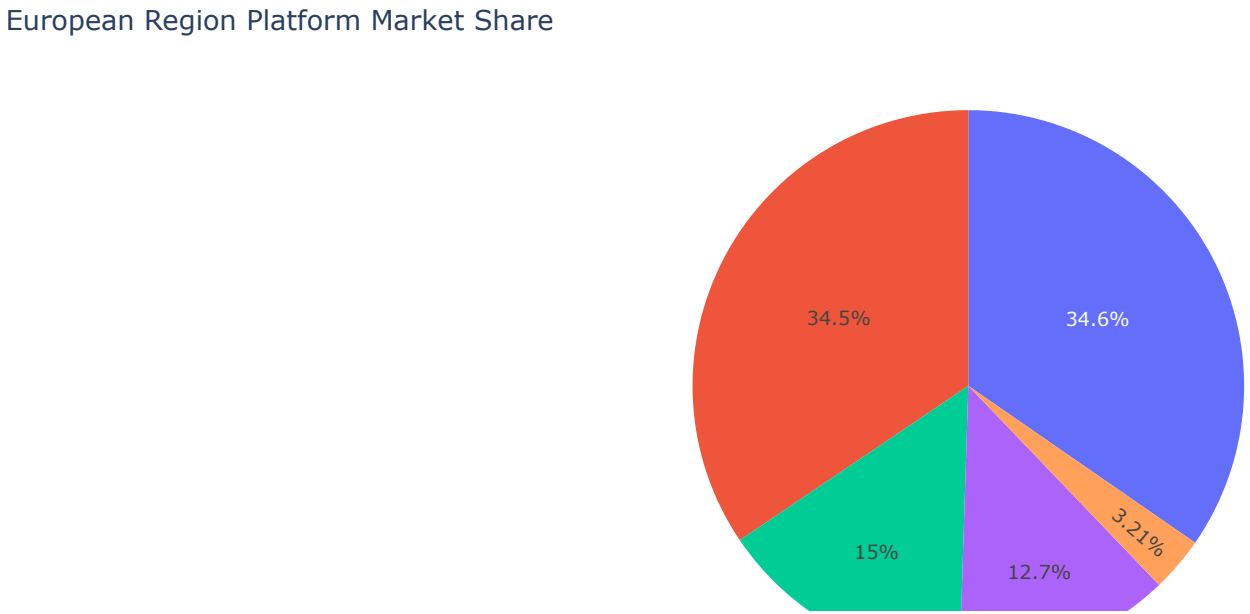
```
C:\Users\XIX\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: FutureWarning:

In a future version, the Index constructor will not infer numeric dtypes when passed object-dtype sequences (matching Series behavior)
```

Out[ ]:

	index	average_sales	eu_sales	jp_sales	na_sales	other_sales	total_sales
0	DS	24.769417	15.039643	72.424962	21.174933	12.735125	24.769417
1	PC	24.572461	34.455927	0.122364	23.913712	20.998081	24.572461
2	PS4	30.181392	34.632662	11.4878	27.859192	46.401152	30.181392
3	PSV	5.169863	3.208228	15.720147	3.194815	6.170825	5.169863
4	XONE	15.306868	12.663541	0.244728	23.857348	13.694818	15.306868

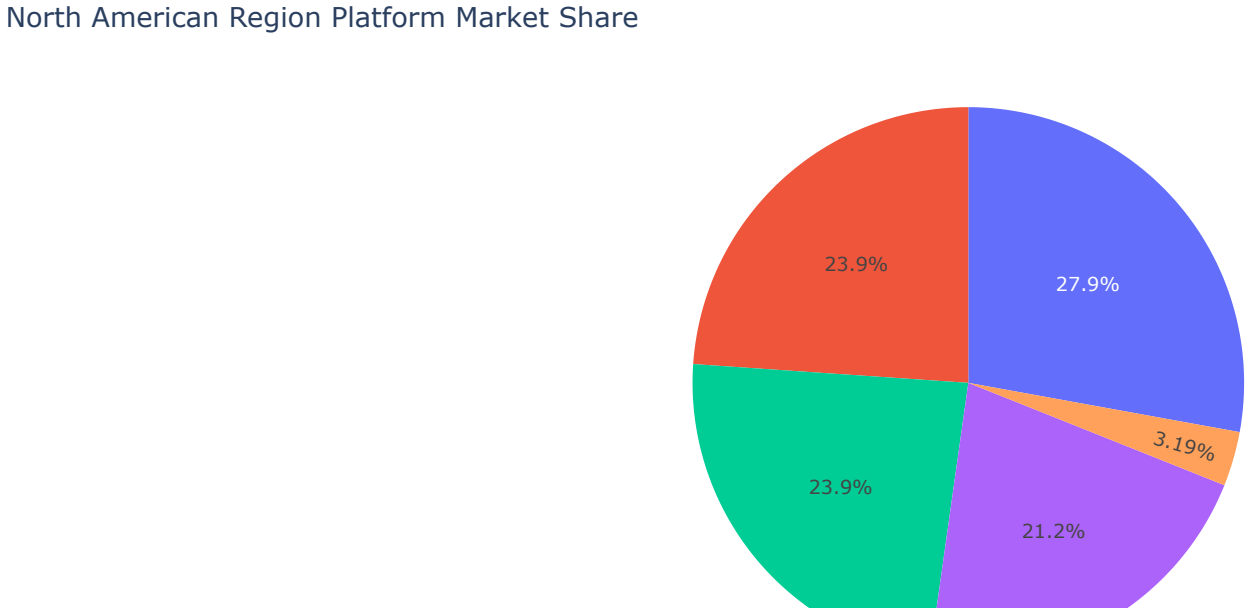
Euorope

In [ ]:
```
# platform market share, Europe
px.pie(pt_final, values=pt_final['eu_sales'], names=['3DS', 'PC', 'PS4', 'PSV', 'XONE'], title='European Region Platform Market Share').show()
```



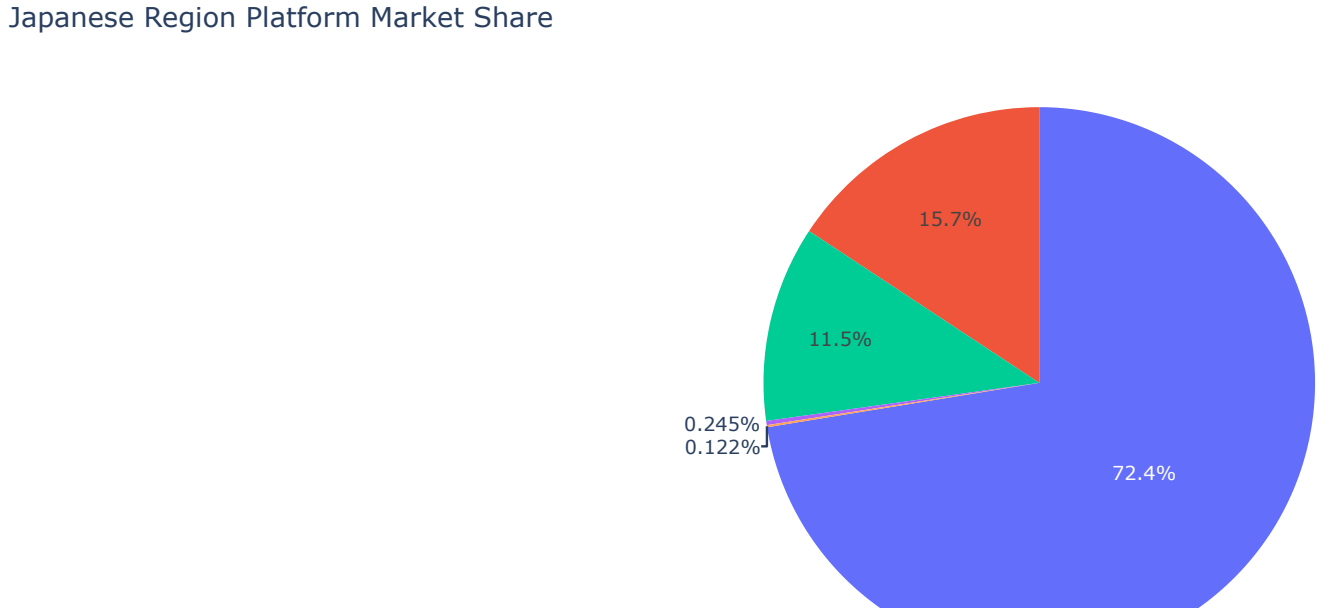
North America

In [ ]:
```
# platform market share, North America
px.pie(pt_final, values=pt_final['na_sales'], names=['3DS', 'PC', 'PS4', 'PSV', 'XONE'], title='North American Region Platform Market Share').show()
```



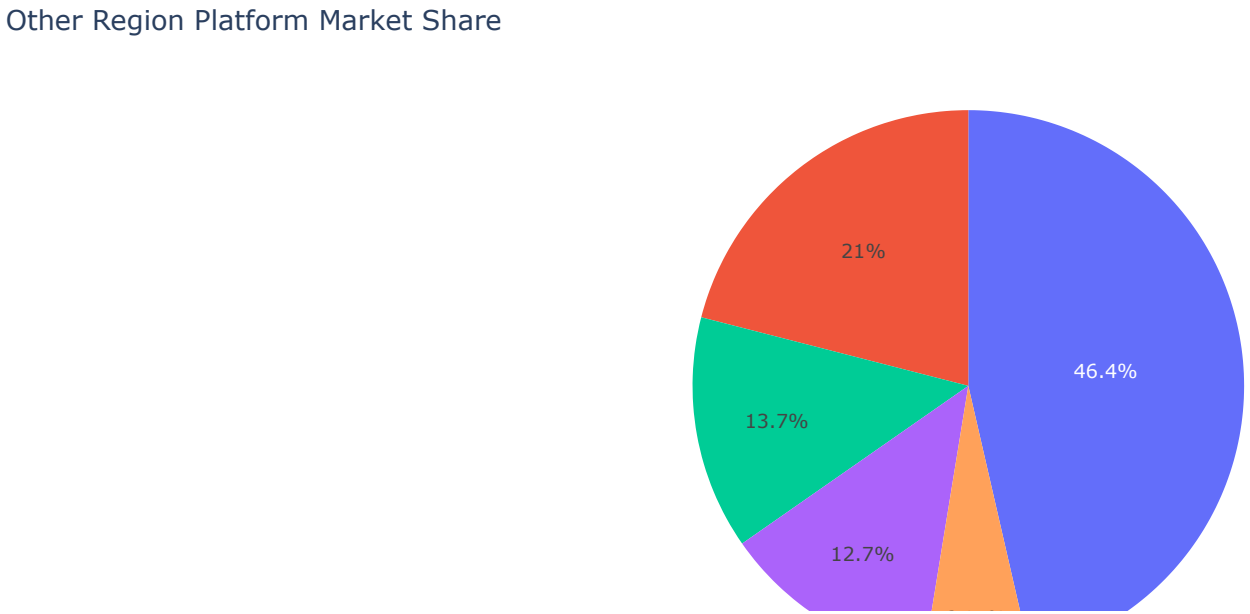
Japan

In [ ]:
```
# platform market share, Japan
px.pie(pt_final, values=pt_final['jp_sales'], names=['3DS', 'PC', 'PS4', 'PSV', 'XONE'], title='Japanese Region Platform Market Share').show()
```



Other

In [ ]:
```
# platform market share, Other
px.pie(pt_final, values=pt_final['other_sales'], names=['3DS', 'PC', 'PS4', 'PSV', 'XONE'], title='Other Region Platform Market Share').show()
```



Conclusions

The market share of each relevant platform, depends on the sales region. In the European region, a vast majority of games are sold for PC and PS4 platforms. In the North American region, market share is more or less evenly split among PC, PS4, XONE and 3DS platforms. In the Japanese region, the 3DS takes up 72.4% market share among the top 5 platforms. The other region sees about a 46% market share for the PS4, and then a 21% market share for PC, among the top platforms.

Statistical Testing of Mean Sales

Comparison of 3DS and PC

The mean sales of 3DS and PC are the same ?

```
Null Hypothesis : The mean sales of 3DS and PC are the same
```

In [ ]:
```
# Comparison of 3DS and PC

#Null Hypothesis

## The mean sales of 3DS and PC are the same


alpha = 0.05

results = st.ttest_ind(ds_box_sales, pc_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.9895189927444241
We can't reject the null hypothesis
```

Comparison of 3DS and PS4

The mean sales of 3DS and PS4 are the same ?

Null Hypothesis : The mean sales of 3DS and PS4 are the same

In [ ]:
```
# Comparison of 3DS and PS4

#Null Hypothesis

## The mean sales of 3DS and PS4 are the same


alpha = 0.05

results = st.ttest_ind(ds_box_sales, ps4_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.6936421504882218
We can't reject the null hypothesis
```

Comparison of 3DS and PSV

The mean sales of 3DS and PSV are the same ?

Null Hypothesis : The mean sales of 3DS and PSV are the same

In [ ]:
```
# Comparison of 3DS and PSV

#Null Hypothesis

## The mean sales of 3DS and PSV are the same


alpha = 0.05

results = st.ttest_ind(ds_box_sales, psv_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.03720070588836854
We reject the null hypothesis, the means are different
```

Comparison of 3DS and XONE

The mean sales of 3DS and XONE are the same ?

Null Hypothesis : The mean sales of 3DS and XONE are the same

In [ ]:
```
# Comparison of 3DS and XONE

#Null Hypothesis

## The mean sales of 3DS and XONE are the same


alpha = 0.05

results = st.ttest_ind(ds_box_sales, xone_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.41423844518684094
We can't reject the null hypothesis
```

Comparison of PC and PS4

The mean sales of PC and PS4 are the same ?

Null Hypothesis : The mean sales of PC and PS4 are the same

In [ ]:
```
# Comparison of PC and PS4

#Null Hypothesis

## The mean sales of PC and PS4 are the same


alpha = 0.05

results = st.ttest_ind(pc_box_sales, ps4_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.7458914072175874
We can't reject the null hypothesis
```

Comparison of PC and PSV

The mean sales of PC and PSV are the same ?

Null Hypothesis : The mean sales of PC and PSV are the same

In [ ]:
```
# Comparison of PC and PSV

#Null Hypothesis

## The mean sales of PC and PSV are the same


alpha = 0.05

results = st.ttest_ind(pc_box_sales, psv_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.17112281031665025
We can't reject the null hypothesis
```

Comparison of PC and XONE

The mean sales of PC and XONE are the same ?

Null Hypothesis : The mean sales of PC and XONE are the same

In [ ]:
```
# Comparison of PC and XONE

#Null Hypothesis

## The mean sales of PC and XONE are the same


alpha = 0.05

results = st.ttest_ind(pc_box_sales, xone_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.5536327634712646
We can't reject the null hypothesis
```

Comparison of PS4 and PSV

The mean sales of PS4 and PSV are the same ?

Null Hypothesis : The mean sales of PS4 and PSV are the same

In [ ]:
```
# Comparison of PS4 and PSV

#Null Hypothesis

## The mean sales of PS4 and PSV are the same


alpha = 0.05

results = st.ttest_ind(ps4_box_sales, psv_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
```

```
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.06275189115886726
We can't reject the null hypothesis
```

Comparison of PS4 and XONE

The mean sales of PS4 and XONE are the same ?

Null Hypothesis : The mean sales of PS4 and XONE are the same

In [ ]:
```
# Comparison of PS4 and XONE

#Null Hypothesis

## The mean sales of PS4 and XONE are the same


alpha = 0.05

results = st.ttest_ind(ps4_box_sales, xone_box_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.3132900002296261
We can't reject the null hypothesis
```

Comparison of PSV and XONE

The mean sales of PSV and XONE are the same ?

Null Hypothesis : The mean sales of PSV and XONE are the same

In [ ]:
```
# Comparison of PSV and XONE

#Null Hypothesis

## The mean sales of PSV and XONE are the same


alpha = 0.05

results = st.ttest_ind(psv_box_sales, xone_box_sales)
```

```
print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.2565719484054285
We can't reject the null hypothesis
```

Conclusions

Using statistical t test of the mean sales of the different platforms, we do not see many differences. The only difference we can attest to with more than a 95% degree of certainty is between the 3DS and the PSV. We fail to reject the null hypotheses that the mean sales for the other platforms are different from each other. We used the standard alpha of 0.05, so that we were at least 95% sure of our results. We chose the null hypotheses to be the mean sales of the corresponding platforms were the same. The lack of difference seen could be attributed to the variance, and hence, the range of the sales.

Differences in Sales by Region

In [ ]:
```
# Filtering data frame for most relevant platforms and key years 2013-2016
df3 = df[df['platform'].isin(most_relevant_platforms_list) & df['year_of_release'].isin(key_years)]
df3.head()
```

Out[ ]:

	name	platform	year_of_release	genre	na_sales	eu_sales	jp_sales	other_sales	critic_score	user_score	rating	average_score	total_sales	average_sales
31	call of duty: black ops 3	PS4	2015	Shooter	6.03	5.86	0.36	2.38	NaN	NaN	NaN	NaN	14.63	3.6575
33	pokemon x/pokemon y	3DS	2013	Role-Playing	5.28	4.19	4.35	0.78	NaN	NaN	NaN	NaN	14.60	3.6500
42	grand theft auto v	PS4	2014	Action	3.96	6.31	0.38	1.97	9.7	8.3	M	9.00	12.62	3.1550
47	pokemon omega ruby/pokemon alpha sapphire	3DS	2014	Role-Playing	4.35	3.49	3.10	0.74	NaN	NaN	NaN	NaN	11.68	2.9200
77	fifa 16	PS4	2015	Sports	1.12	6.12	0.06	1.28	8.2	4.3	E	6.25	8.58	2.1450

In [ ]:
```
# Filter by EU region sales
df_eu = df4[['platform', 'eu_sales']]
```

In [ ]:
```
# Convert EU data to list
df_eu_sales = df_eu['eu_sales'].to_list()
```

In [ ]:
```
# Filter by JP region sales
df_jp = df4[['platform', 'jp_sales']]
```

In [ ]:
```
# Convert JP data to list
df_jp_sales = df_jp['jp_sales'].to_list()
```

In [ ]:
```
# Filter for NA region sales
df_na = df4[['platform', 'na_sales']]
```

In [ ]:
```
# Convert NA sales to list
df_na_sales = df_na['na_sales'].to_list()
```

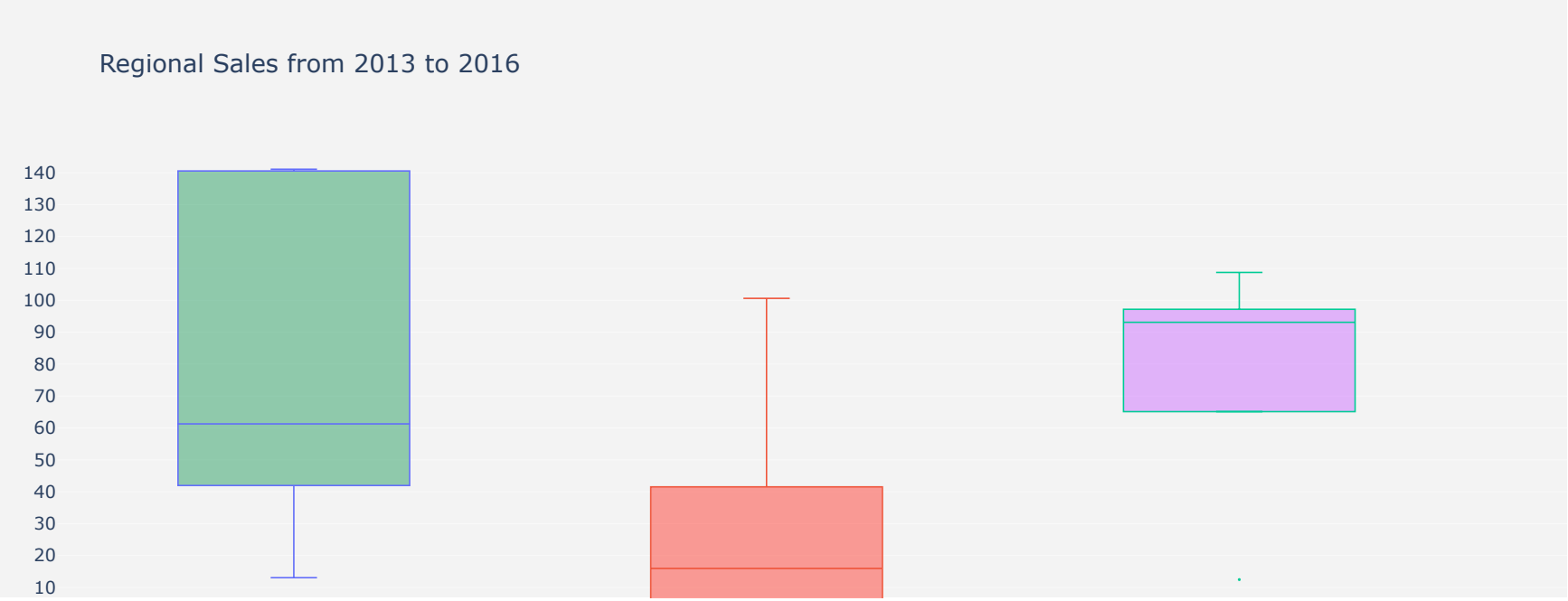
In [ ]:
```
# Filter for Other region sales
df_other = df4[['platform', 'other_sales']]
```

In [ ]:
```
# Convert other region sales to list
df_other_sales = df_other['other_sales'].to_list()
```

Overal Results

In [ ]:
```
x_data = ['Europe', 'Japan', 'North America', 'Other']

y_data = [df_eu_sales , df_jp_sales, df_na_sales, df_other_sales]

colors = ['rgba(44, 160, 101, 0.5)',
          'rgba(255, 65, 54, 0.5)', 'rgba(207, 114, 255, 0.5)', 'rgba(127, 96, 0, 0.5)']

fig = go.Figure()

for xd, yd, cls in zip(x_data, y_data, colors):
        fig.add_trace(go.Box(
            y=yd,
            name=xd,
            jitter=0.5,
            whiskerwidth=0.2,
            fillcolor=cls,
            marker_size=2,
            line_width=1)
        )

fig.update_layout(
    title='Regional Sales from 2013 to 2016',
    yaxis=dict(
        autorange=True,
        showgrid=True,
        zeroline=True,
        dtick=10,
        gridcolor='rgb(255, 255, 255)',
        gridwidth=1,
        zerolinecolor='rgb(255, 255, 255)',
        zerolinewidth=2,
    ),
    margin=dict(
        l=40,
        r=30,
        b=80,
        t=100,
    ),
    paper_bgcolor='rgb(243, 243, 243)',
```

```
    plot_bgcolor='rgb(243, 243, 243)',
    showlegend=False
)

fig.show()
```



Conclusions

We see the data for the regional sales from 2013 to 2016. It is difficult to predict differences, so we would need to conduct statistical testing of means.

Statistical Testing of Sales Means by Region

Europe and Japan

```
Null Hypothesis : The mean sales of EU and JP are the same
```

In [ ]:
```
# Comparison of EU and JP

#Null Hypothesis

## The mean sales of EU and JP are the same


alpha = 0.05

results = st.ttest_ind(df_eu_sales, df_jp_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.12795336461497347
We can't reject the null hypothesis
```

Europe and North America

Null Hypothesis : The mean sales of EU and NA are the same

In [ ]:
```
# Comparison of EU and NA

#Null Hypothesis

## The mean sales of EU and NA are the same


alpha = 0.05

results = st.ttest_ind(df_eu_sales, df_na_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.9139153517721482
We can't reject the null hypothesis
```

Europe and Other

Null Hypothesis : The mean sales of EU and Other are the same

In [ ]:
```
# Comparison of EU and Other

#Null Hypothesis

## The mean sales of EU and Other are the same


alpha = 0.05

results = st.ttest_ind(df_eu_sales, df_other_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.05156498857359916
We can't reject the null hypothesis
```

Japan and North America

Null Hypothesis : The mean sales of JP and NA are the same

In [ ]:
```
# Comparison of JP and NA

#Null Hypothesis

## The mean sales of JP and NA are the same


alpha = 0.05

results = st.ttest_ind(df_jp_sales, df_na_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.08131759488099968
We can't reject the null hypothesis
```

Japan and Other

Null Hypothesis : The mean sales of JP and NA are the same

In [ ]:
```
# Comparison of JP and NA

#Null Hypothesis

## The mean sales of JP and NA are the same


alpha = 0.05

results = st.ttest_ind(df_jp_sales, df_na_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.08131759488099968
We can't reject the null hypothesis
```

North America and Other

Null Hypothesis : The mean sales of NA and Other are the same

In [ ]:
```
# Comparison of NA and Other

#Null Hypothesis

## The mean sales of NA and Other are the same


alpha = 0.05

results = st.ttest_ind(df_na_sales, df_other_sales)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  0.014543290639873504
We reject the null hypothesis, the means are different
```

Conclusions

We see that most of the mean regional sales are not different from each other. The one exception lies with the mean sales, of North American and Other. We chose our null hypotheses to be similarities in mean sales, and our alpha value to be 0.05. This alpha allows us to be at least 95% certain of our results.

User and Professional Reviews Effect on PS4 Sales

In [ ]:
```
# Creating a filter
ps4_list = ['PS4']
```

In [ ]:
```
# filtering data frame for PS4
df_ps4 = df3[df3['platform'].isin(ps4_list)]
```

In [ ]:
```
# Sorting the resulting data set by scores and total sales
ps4_score_sales = df_ps4[['user_score', 'critic_score','average_score', 'total_sales']].sort_values(by='total_sales', ascending=False)
```

In [ ]:
```
# correlation between scores and total sales
ps4_score_sales.corr()
```

Out[ ]:

	user_score	critic_score	average_score	total_sales
user_score	1.000000	0.520752	0.922750	0.023279
critic_score	0.520752	1.000000	0.839231	0.406568
average_score	0.922750	0.839231	1.000000	0.201518
total_sales	0.023279	0.406568	0.201518	1.000000

In [ ]:
```
# sales based on review scores
px.scatter(ps4_score_sales, x=['user_score', 'critic_score', 'average_score'],
y='total_sales', trendline='ols', trendline_color_override='black',
title='PS4 Total Sales Based on Review Scores').show()
```

Conclusions

We see a very weak correlation between reviews and sales, when considering user and critic reviews. Critic scores show a stronger relationship with sales, when compared to user scores, yet that relationship is fairly weak. Therefore, scores are not a strong predictor of total sales.

Which platforms are leading in sales from 2013 to 2016?

Top Platform Sales

In [ ]:
```
# Filter by 2013 to 2016
df5 = df[df['year_of_release'].isin(key_years)]
```

In [ ]:
```
# group the platforms by sales
df6 = df5.groupby('platform')['eu_sales', 'jp_sales', 'na_sales', 'other_sales', 'average_sales', 'total_sales'].sum()
```

```
C:\Users\XIX\AppData\Local\Temp\ipykernel_23028\3857988385.py:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [ ]:
```
# Sort platforms by highest total sales
df6.sort_values(by='total_sales',ascending=False).head(10)
```

Out[ ]:

platform	eu_sales	jp_sales	na_sales	other_sales	average_sales	total_sales
PS4	141.09	15.96	108.74	48.35	78.5350	314.14
PS3	67.81	23.35	63.50	26.77	45.3575	181.43
XONE	51.59	0.34	93.12	14.27	39.8300	159.32
3DS	30.96	67.81	38.20	6.28	35.8125	143.25
X360	42.52	0.51	81.66	12.11	34.2000	136.80
WIIU	19.85	10.88	29.21	4.69	16.1575	64.63
PC	25.36	0.00	11.11	2.96	9.8575	39.43
PSV	6.10	18.59	5.04	3.26	8.2475	32.99
WII	5.93	0.05	6.56	1.12	3.4150	13.66
PSP	0.17	3.29	0.00	0.04	0.8750	3.50

The PS4 leads in sales for the relevant time period. Third, is the PS3, which appears to be shrinking as PS4 sales increase. This is because the PS4 is the updated version of the PS3, so they compete in sales. The sales trends suggest customers are deciding to upgrade their platform. The next platform on the list is the XONE, which is the updated version of the X360. We see similar trends with playstation. Fourth on the total sales ranking ins the 3DS, and 6th is the WIIU.

Comparing Game Sales on Various Platforms

In [ ]:
```
# Top games dataset
top_five_games
```

Out[ ]:

	name	average_sales	total_sales
10927	wii sports	20.6350	82.54
3691	grand theft auto v	14.1450	56.58
9300	super mario bros.	11.3275	45.31
9660	tetris	8.9600	35.84
5501	mario kart wii	8.8800	35.52

Wii sports and mario kart are only on the Wii platform. Super smash bros and Tetris may not be on cross platforms either.

In [ ]:
```
```

In [ ]:
```
# Top 20 Games by total sales from 2013-2016
df5.pivot_table(index='name', values=['total_sales', 'average_sales'],
aggfunc='sum').reset_index().sort_values(by='total_sales', ascending=False).head(20)
```

Out[ ]:

	name	average_sales	total_sales
428	grand theft auto v	14.1450	56.58
139	call of duty: ghosts	6.8475	27.39
138	call of duty: black ops 3	6.4175	25.67
660	minecraft	6.0400	24.16
137	call of duty: advanced warfare	5.4925	21.97
358	fifa 15	4.3425	17.37
357	fifa 14	4.1150	16.46
359	fifa 16	4.0750	16.30
801	pokemon x/pokemon y	3.6500	14.60
103	battlefield 4	3.4850	13.94
66	assassin's creed iv: black flag	3.2650	13.06
336	fallout 4	3.1675	12.67
218	destiny	3.1350	12.54
1021	super smash bros. for wii u and 3ds	3.1050	12.42
984	star wars battlefront (2015)	3.0475	12.19
797	pokemon omega ruby/pokemon alpha sapphire	2.9200	11.68
360	fifa 17	2.8700	11.48
1073	the last of us	2.6450	10.58
584	lego marvel super heroes	2.3525	9.41
1200	watch dogs	2.2950	9.18

Looking at Grand Theft Auto V, Call of Duty: Ghost, Minecraft, and FIFA 15, as they appear on a number of platforms. Nevertheless, many games found on Nintendo platforms are not present on the other console platforms.

In [ ]:
```
# Pivot table of games and their platform, aggregated by total sales
df7 = df5.pivot_table(index=['name', 'platform'], values='total_sales', aggfunc='sum').reset_index()
```

In [ ]:
```
# Creating filters
cross_games = ('grand theft auto v', 'call of duty: ghosts', 'minecraft', 'fifa 15')
cross_platforms = ('PS4', 'XONE', 'WIIU', 'PC', '3DS')
```

In [ ]:
```
# Applying filter to our data set
df_cross = df7[df7['name'].isin(cross_games) & df7['platform'].isin(cross_platforms)]
```

In [ ]:
```
# Seeing which platforms have the most games in our selection
df7[df7['platform'].isin(cross_platforms)].sort_values(by=['name', 'platform']).value_counts(subset='platform')
```

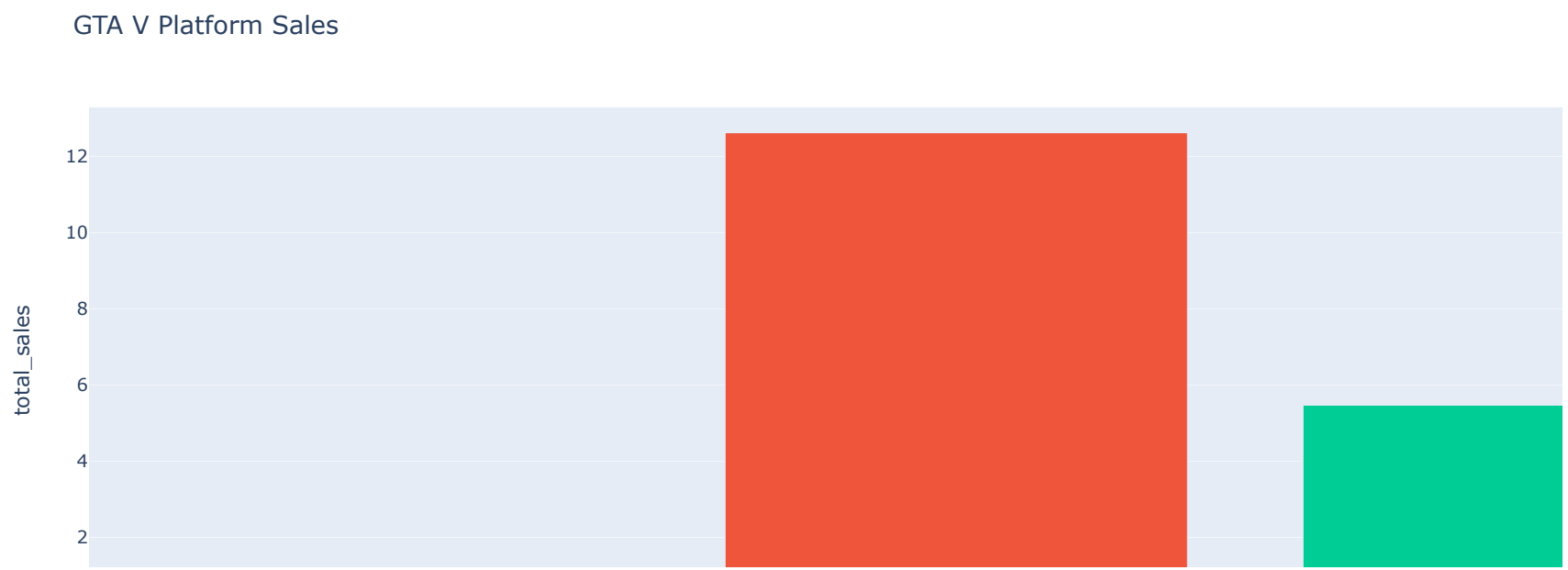
Out[ ]:
```
platform
PS4     392
3DS     303
XONE    247
PC      189
WIIU    115
dtype: int64
```

In [ ]:
```
# Making a COD data set
cod = df_cross[df_cross['name']=='call of duty: ghosts']
```

In [ ]:
```
#making a GTA data set
gta = df_cross[df_cross['name']=='grand theft auto v']
```

In [ ]:
```
px.bar(cod, x='platform', y='total_sales', color='platform', title='COD: Ghost Platform Sales').show()
```

In [ ]:
```
# GTA V sales
px.bar(gta, x='platform', y='total_sales', color='platform', title='GTA V Platform Sales').show()
```



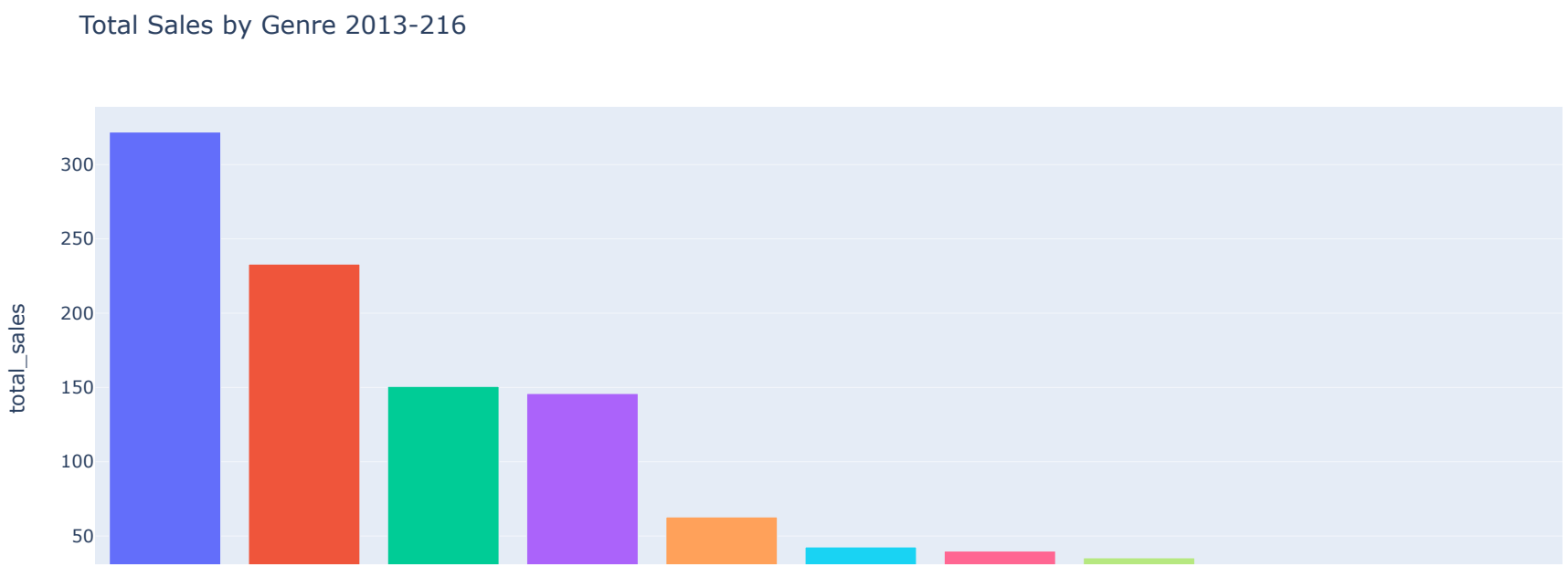
The data shows that very few games are sold on multiple platforms, and even less sold on WIIU and 3DS, with other platforms. This suggests the WIIU and 3DS ecosystems are mostly closed off to the other platforms, as 3DS has the second highest amount of games, yet we do not see those games on other platforms. We do see cross platform games among PC, PS4, and XONE consoles. Where games appear cross platform on PS4, PC, and XONE, we see PS4 games sales dominate. This is further demonstrated by the PS4 leading in the number of different games sold. Sales are then second most popular on the XONE platform. We see this trend with Call of Duty: Ghosts, and Grand Theft Auto V, two of the most popular platform games.

General Distribution of Games by Genre and Rating

Genre

In [ ]:
```
# Sorting data by genre
genres = df5.groupby('genre')['total_sales'].sum().sort_values(ascending=False).reset_index()
```

In [ ]:
```
# sales by genre
px.bar(genres, x='genre', y='total_sales', color='genre', title='Total Sales by Genre 2013-216').show()
```

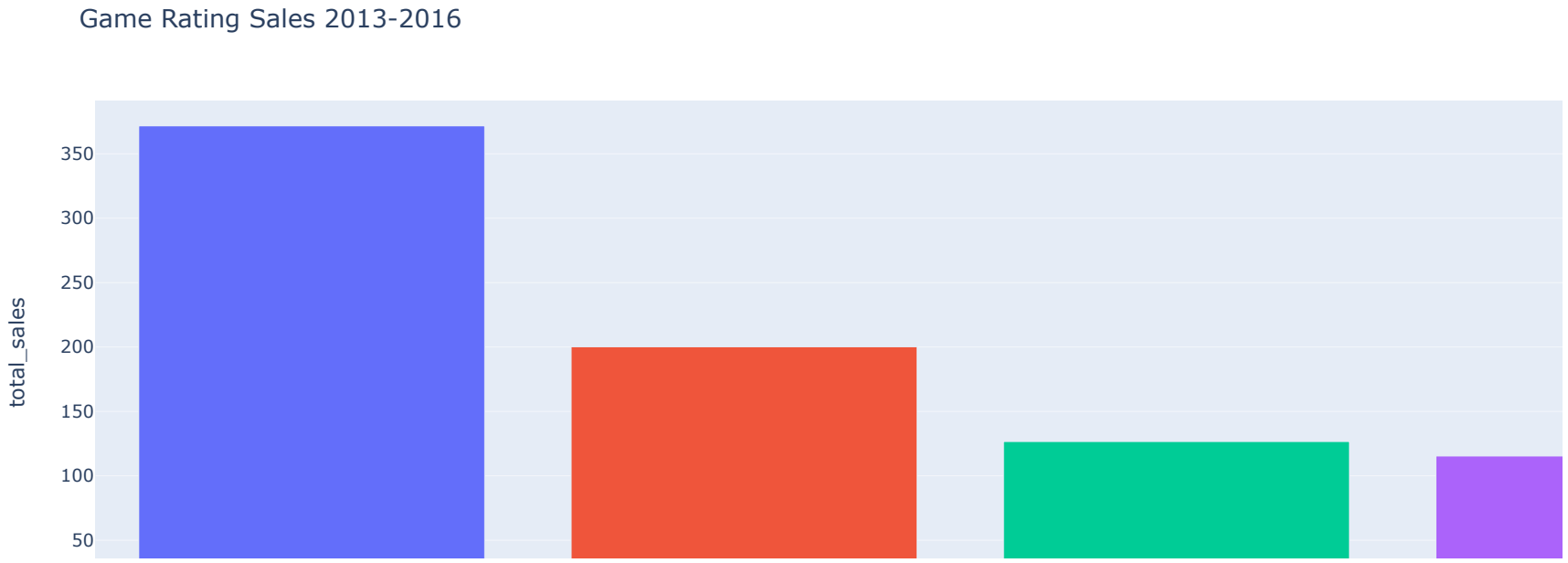


Rating

In [ ]:
```
# creating a rating pivot table
df_rating = df5.pivot_table(index=['name', 'platform', 'rating'], values='total_sales', aggfunc='sum').reset_index()
```

In [ ]:
```
# Indexing for the total sales based on rating
df_rating_sales = df_rating.groupby('rating')['total_sales'].sum().sort_values(ascending=False).reset_index()
```

In [ ]:
```
# game rating sales
px.bar(df_rating_sales, x='rating', y='total_sales', color='rating', title='Game Rating Sales 2013-2016').show()
```



Conclusions

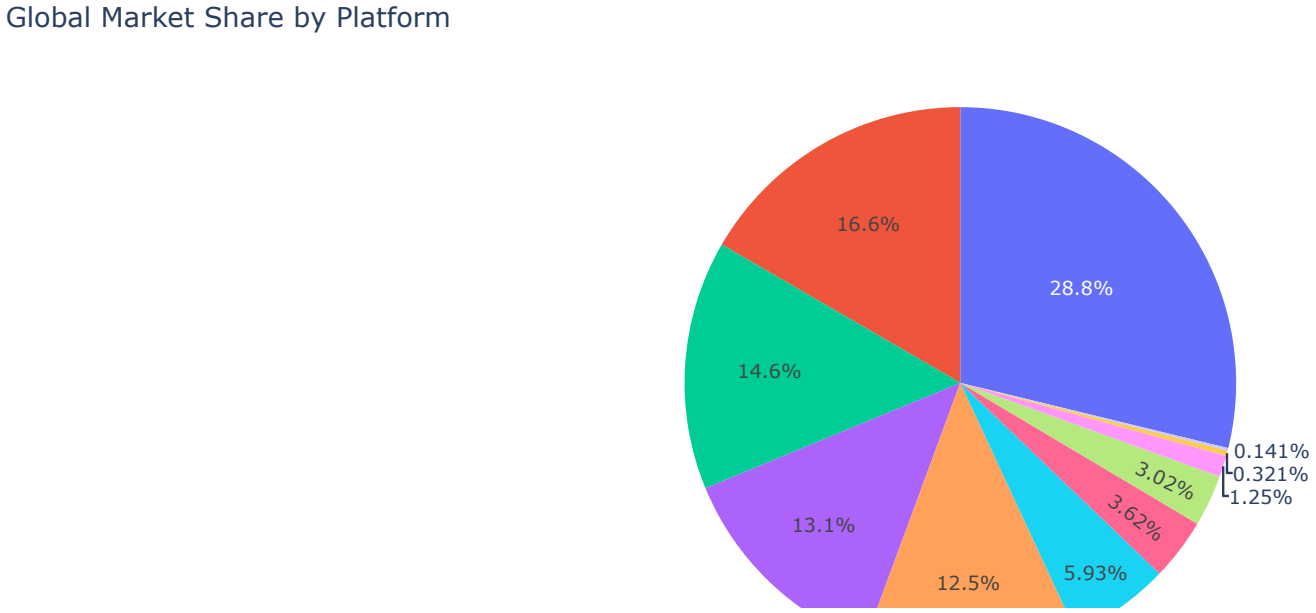
Action games are the most profitable genre, followed by shooter and sports games. The genres with lower sales may have a smaller audience, or are geared for a subset of customers, such as children. We can see when evaluating sales based on rating, mature games sell more. A logical assumption would be that action and shooter games are for mature audiences.

Top Platforms Per Region

In [ ]:
```
# Filtering dataframe for key years 2013 to 2016
df8 = df[df['year_of_release'].isin(key_years)]
```

In [ ]:
```
# grouping result by platform, then looking at sales data
df9 = df8.groupby('platform')[['total_sales','eu_sales', 'jp_sales', 'na_sales', 'other_sales']].sum().reset_index()
```

In [ ]:
```
# platfrom global market share
px.pie(df9, values='total_sales', names='platform', title='Global Market Share by Platform').show()
```



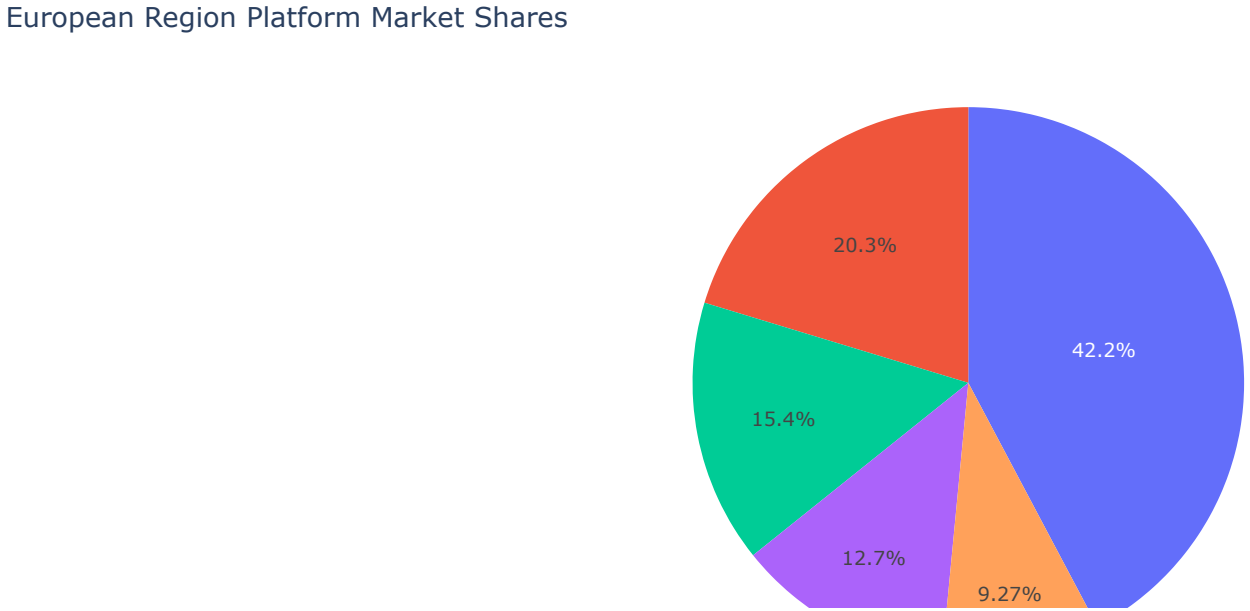
European Region Platform Market Shares

In [ ]:
```
# European top platforms
df_top_eu = df9.sort_values(by='eu_sales', ascending=False).head()
df_top_eu = df_top_eu[['platform','eu_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in europe
df_top_eu_sum =df_top_eu.cumsum()
df_top_eu_sum = df_top_eu_sum[4:]
df_top_eu_sum = df_top_eu_sum['eu_sales'].to_list()
```

In [ ]:
```
# multiply market share by 100
df_top_eu['market_share']= (df_top_eu['eu_sales'] / df_top_eu_sum) *100
```

In [ ]:
```
# european market share
px.pie(df_top_eu, values='market_share', names='platform', title='European Region Platform Market Shares').show()
```



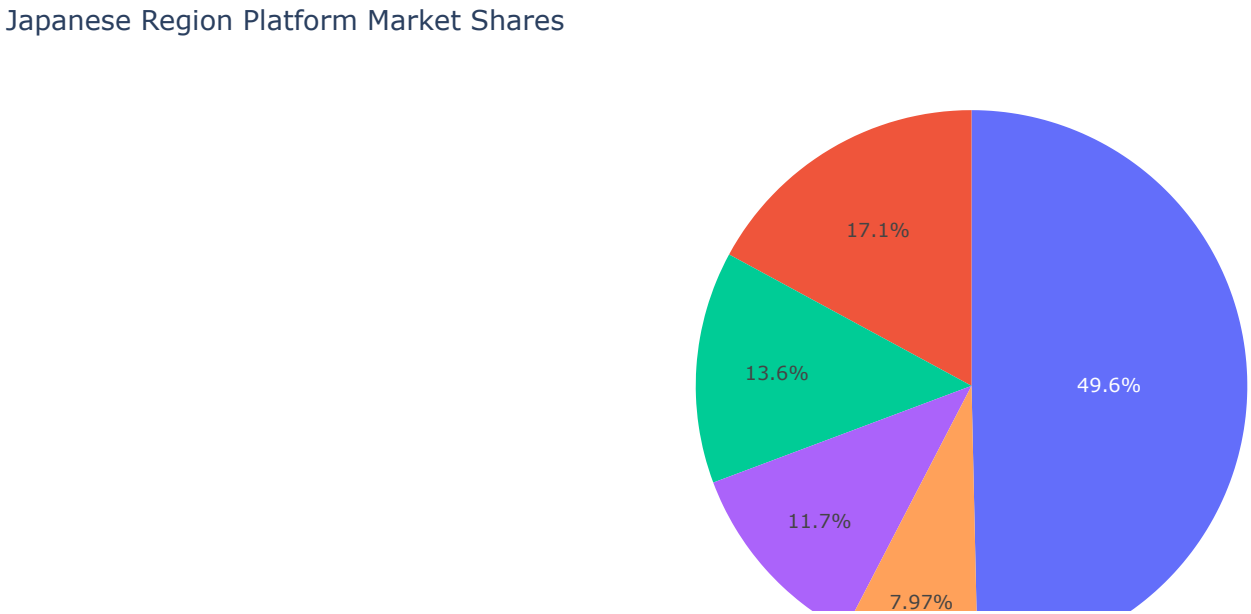
Japanese Region Platform Market Shares

In [ ]:
```
# top japanese platforms
df_top_jp = df9.sort_values(by='jp_sales', ascending=False).head()
df_top_jp = df_top_jp[['platform','jp_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in japan
df_top_jp_sum =df_top_jp.cumsum()
df_top_jp_sum = df_top_jp_sum[4:]
df_top_jp_sum = df_top_jp_sum['jp_sales'].to_list()
```

In [ ]:
```
# market share percentage
df_top_jp['market_share']= (df_top_jp['jp_sales'] / df_top_jp_sum) *100
```

In [ ]:
```
# Japan platform market share
px.pie(df_top_jp, values='market_share', names='platform', title='Japanese Region Platform Market Shares').show()
```



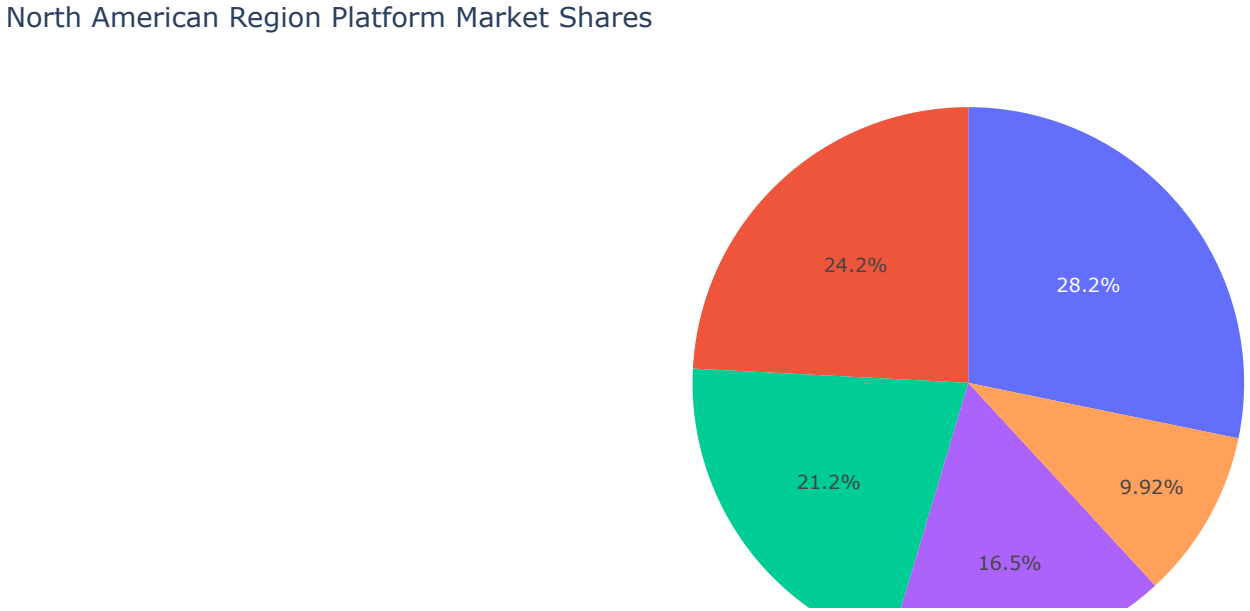
North American Region Platform Market Shares

In [ ]:
```
# Top north american platforms
df_top_na = df9.sort_values(by='na_sales', ascending=False).head()
df_top_na = df_top_na[['platform','na_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in north america
df_top_na_sum =df_top_na.cumsum()
df_top_na_sum = df_top_na_sum[4:]
df_top_na_sum = df_top_na_sum['na_sales'].to_list()
```

In [ ]:
```
# market share percentage
df_top_na['market_share']= (df_top_na['na_sales'] / df_top_na_sum) *100
```

In [ ]:
```
# north america platform market share
px.pie(df_top_na, values='market_share', names='platform', title='North American Region Platform Market Shares').show()
```



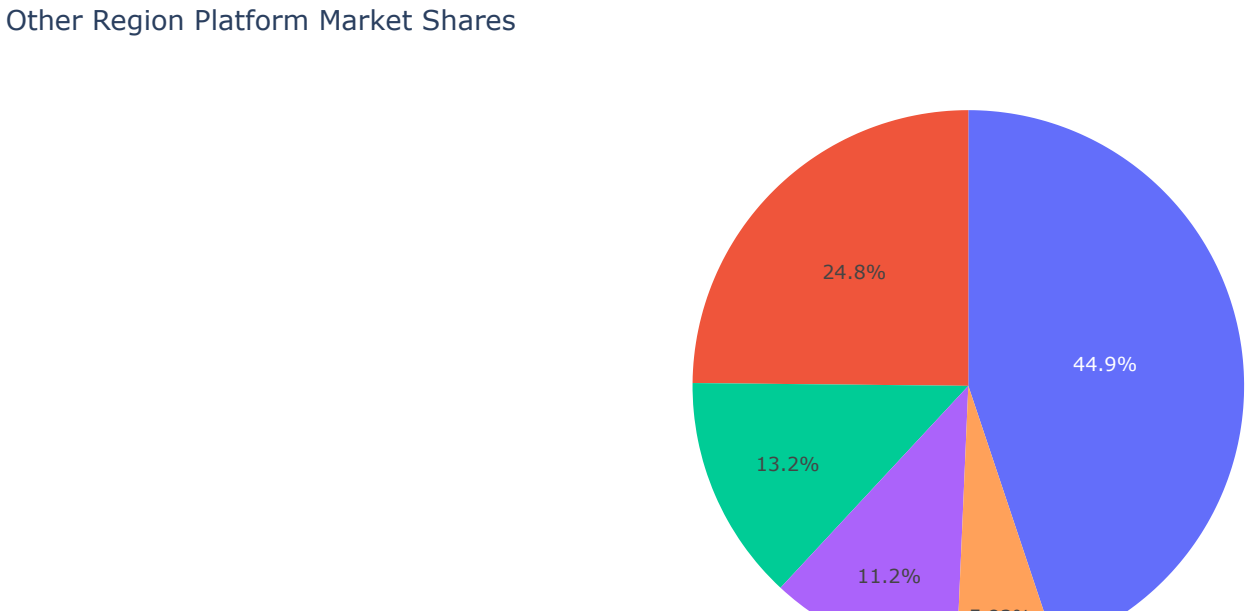
Other Region Platform Market Shares

In [ ]:
```
# other region top platform sales
df_top_other = df9.sort_values(by='other_sales', ascending=False).head()
df_top_other = df_top_other[['platform','other_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in north america
df_top_other_sum =df_top_other.cumsum()
df_top_other_sum = df_top_other_sum[4:]
df_top_other_sum = df_top_other_sum['other_sales'].to_list()
```

In [ ]:
```
# market share percentage
df_top_other['market_share']= (df_top_other['other_sales'] / df_top_other_sum) *100
```

In [ ]:
```
# other region market share
px.pie(df_top_other, values='market_share', names='platform', title='Other Region Platform Market Shares').show()
```



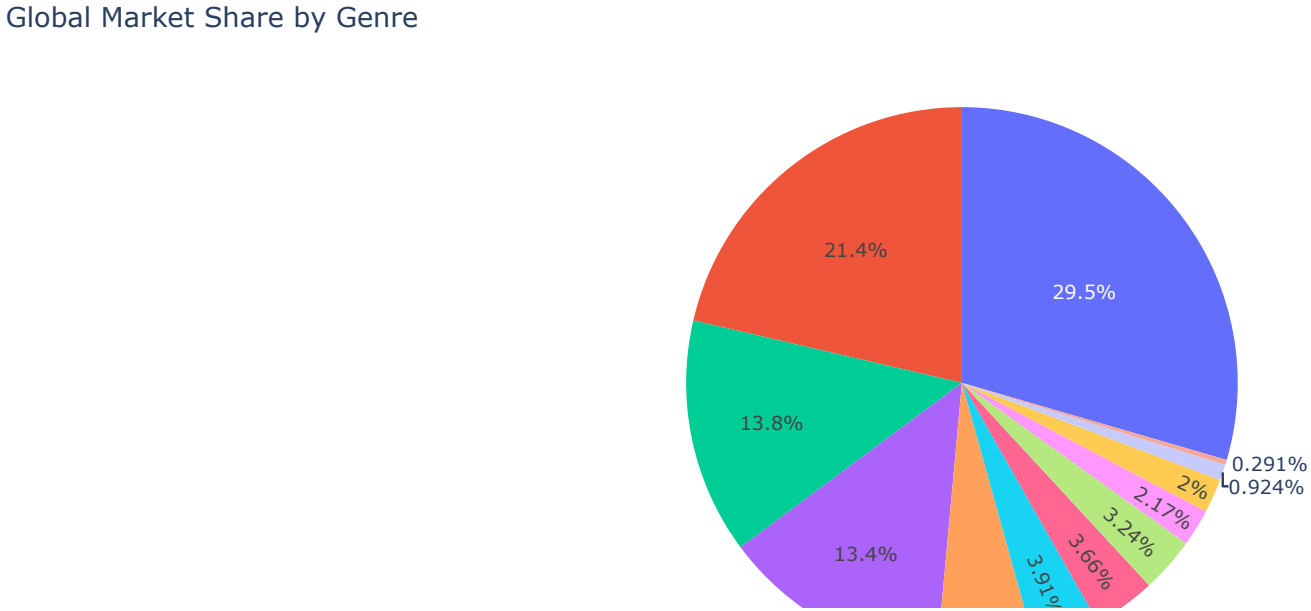
Conclusions

The platforms with the top market shares generally differ region to region. The PS4 has the largest market share in Europe, followed by the PS3, XONE, X360, and then the 3DS. Japan market share is dominated by the 3DS. The 3DS is followed by the PS3, PSV, PS4, and then the WIIU. The North American market leads with the PS4, and then close after is the XONE. after those two, market share is split among the X360, PS3, and 3DS in that order. Looking at the other region, PS4 game sales take a lion share of the market at 44.8%. Following that platform is the PS3, XONE, X360, and then the 3DS. Overall, the PS4 seems to predominate most markets out of the Japanese region, where it takes only 11.7% of the market. The PS3 and XONE are also fairly strong in their market shares, but only out of the Japanese region. Japan has different tastes when it comes to platforms. This region prefers the 3DS, and is also somewhat fond of the WIIU, which is a platform not seen in the top five of any other regions.

Top Genres Per Region

In [ ]:
```
# Categorizing datafarame by genre and sum of sales
df10 = df8.groupby('genre')[['total_sales','eu_sales', 'jp_sales', 'na_sales', 'other_sales']].sum().reset_index()
```

In [ ]:
```
# market share, genre
px.pie(df10, values='total_sales', names='genre', title='Global Market Share by Genre').show()
```



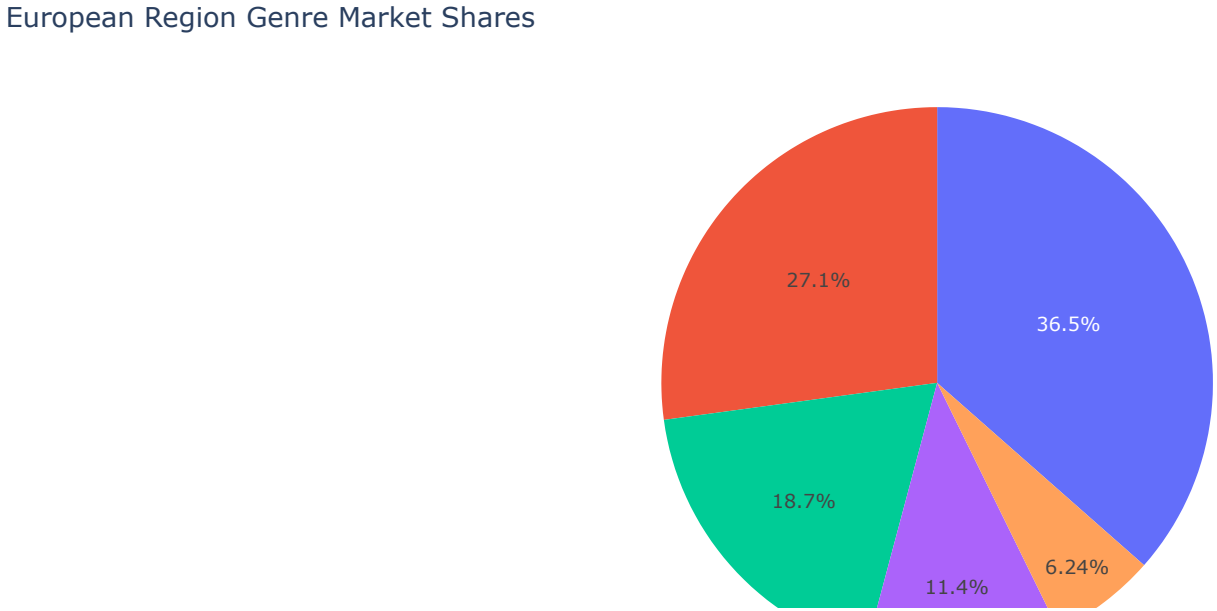
European Region Top 5 Genres Market Share

In [ ]:
```
# Top 5 european region genres
df_genre_eu = df10.sort_values(by='eu_sales', ascending=False).head()
df_genre_eu = df_genre_eu[['genre','eu_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in europe
df_genre_eu_sum =df_genre_eu.cumsum()
df_genre_eu_sum = df_genre_eu_sum[4:]
df_genre_eu_sum = df_genre_eu_sum['eu_sales'].to_list()
```

In [ ]:
```
# Creating market share column
df_genre_eu['market_share']= (df_genre_eu['eu_sales'] / df_genre_eu_sum) *100
```

In [ ]:
```
# European genre market share
px.pie(df_genre_eu, values='market_share', names='genre', title='European Region Genre Market Shares').show()
```



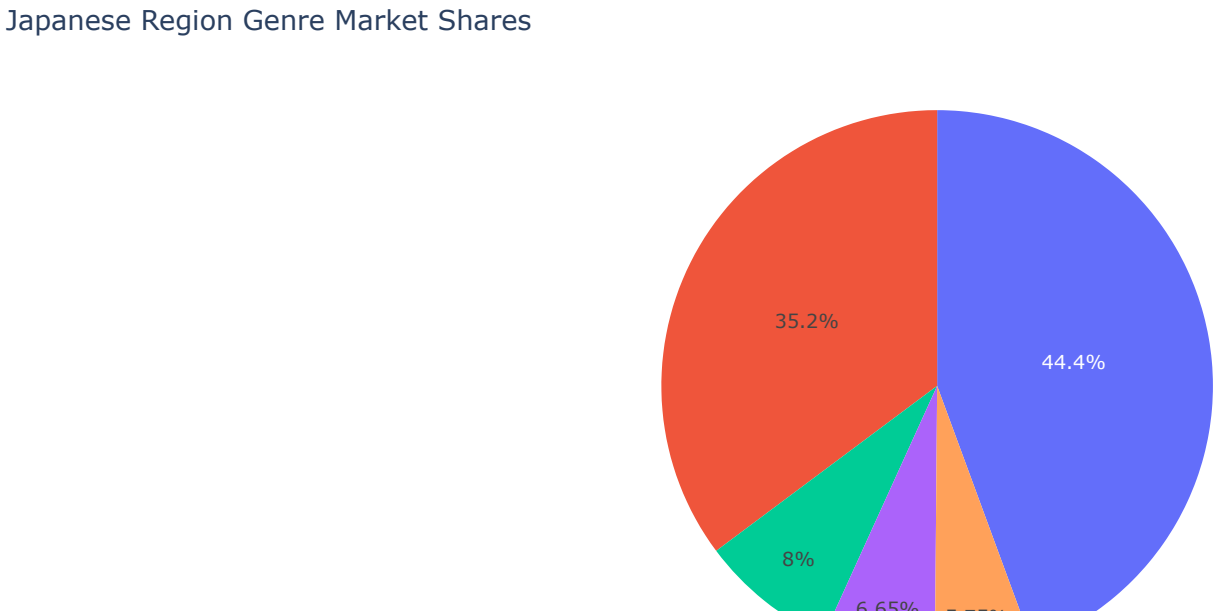
Japanese Region Top 5 Genres Market Share

In [ ]:
```
# Top 5 japanese region genres
df_genre_jp = df10.sort_values(by='jp_sales', ascending=False).head()
df_genre_jp = df_genre_jp[['genre','jp_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in japan
df_genre_jp_sum =df_genre_jp.cumsum()
df_genre_jp_sum = df_genre_jp_sum[4:]
df_genre_jp_sum = df_genre_jp_sum['jp_sales'].to_list()
```

In [ ]:
```
# Creating market share column
df_genre_jp['market_share']= (df_genre_jp['jp_sales'] / df_genre_jp_sum) *100
```

In [ ]:
```
# Japan genre market share
px.pie(df_genre_jp, values='market_share', names='genre', title='Japanese Region Genre Market Shares')
```



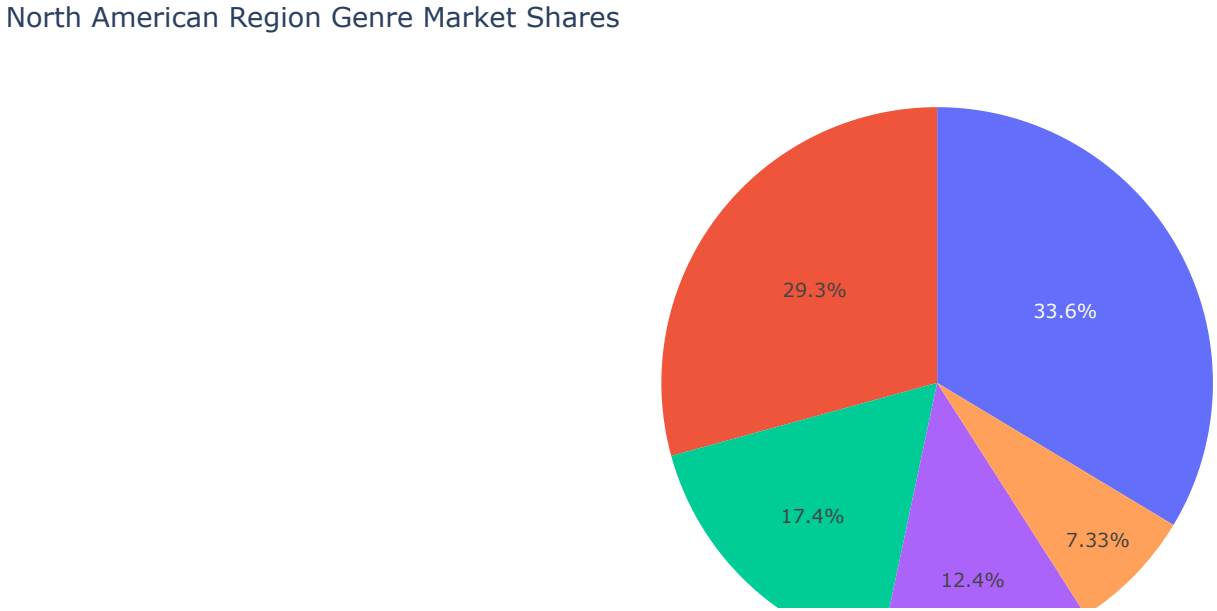
North American Region Top 5 Genres Market Share

In [ ]:
```
# Top 5 north american region genres
df_genre_na = df10.sort_values(by='na_sales', ascending=False).head()
df_genre_na = df_genre_na[['genre','na_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in japan
df_genre_na_sum =df_genre_na.cumsum()
df_genre_na_sum = df_genre_na_sum[4:]
df_genre_na_sum = df_genre_na_sum['na_sales'].to_list()
```

In [ ]:
```
# Creating market share column
df_genre_na['market_share']= (df_genre_na['na_sales'] / df_genre_na_sum) *100
```

In [ ]:
```
# North America genre market share
px.pie(df_genre_na, values='market_share', names='genre', title='North American Region Genre Market Shares').show()
```



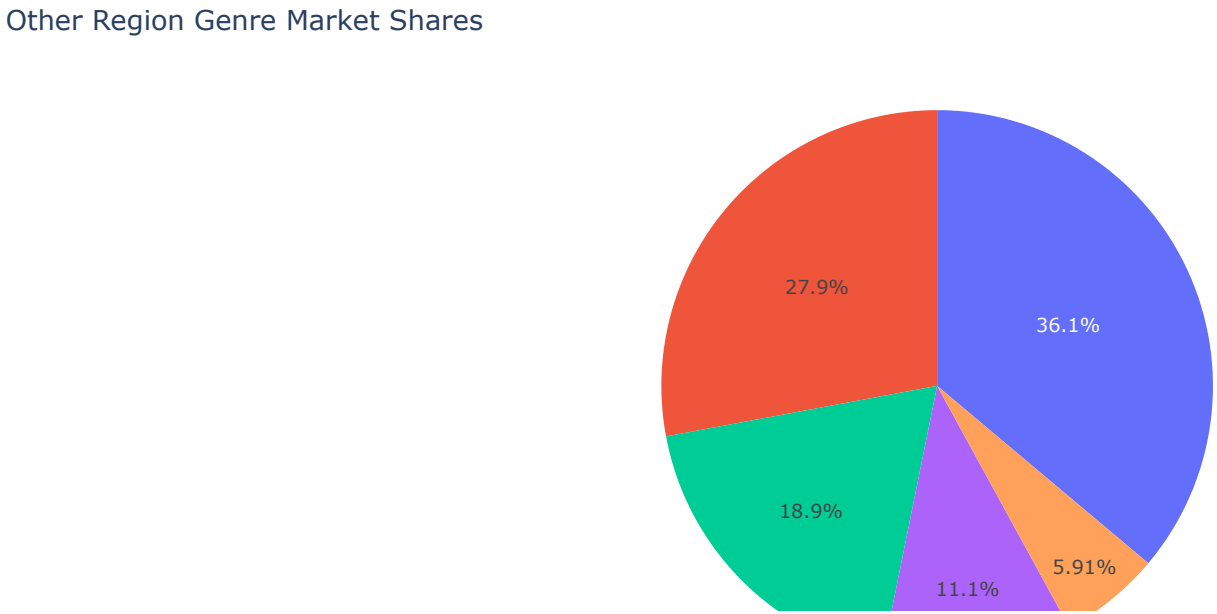
Other Region Top 5 Genres Market Share

In [ ]:
```
# Top 5 other region genres
df_genre_other = df10.sort_values(by='other_sales', ascending=False).head()
df_genre_other = df_genre_other[['genre','other_sales']].reset_index()
```

In [ ]:
```
# Cum sum of sales in japan
df_genre_other_sum =df_genre_other.cumsum()
df_genre_other_sum = df_genre_other_sum[4:]
df_genre_other_sum = df_genre_other_sum['other_sales'].to_list()
```

In [ ]:
```
# Creating market share column
df_genre_other['market_share']= (df_genre_other['other_sales'] / df_genre_other_sum) *100
```

In [ ]:
```
# Other genre market share
px.pie(df_genre_other, values='market_share', names='genre', title='Other Region Genre Market Shares').show()
```



Conclusions

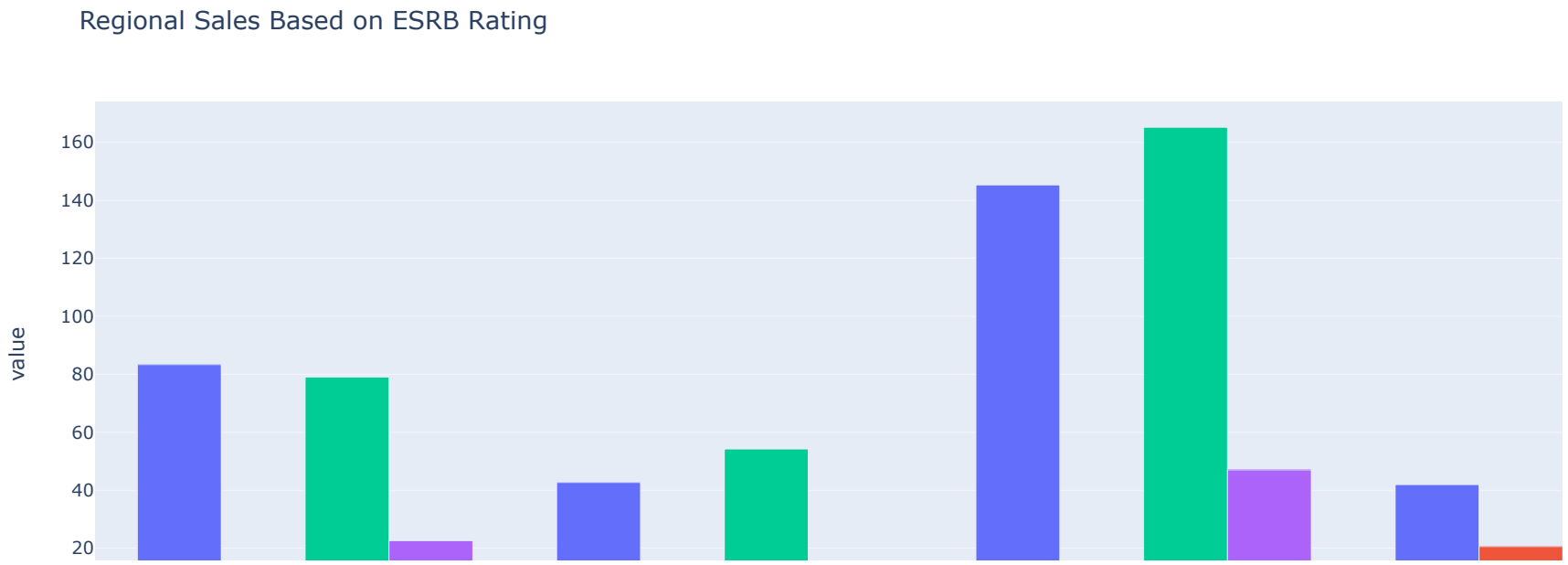
The European region prefers action games, followed by shooters and sports. Role playing and racing take a small portion of the market share. In the Japanese region, role playing and then action games take a majority of the market share. Miscellaneous, fighting, and shooter genres take a smaller portion of the Japanese market. In North America, action games predominate, followed by shooter. Then we have sports, role playing, and finally miscellaneous. In other region, we see action is first, shooter is second, and sports is third. The last two are role playing and miscellaneous. Overall, the main trend is action, shooter, sports, role playing, in all regions beside Japan. In Japan, they prefer role playing and action games.

ESRB Ratings Affect Regional Sales

In [ ]:
```
# creating pivot based on ratings, then collect sales
df_esrb = df5.pivot_table(index=['name', 'rating'],
values=['eu_sales','jp_sales', 'na_sales', 'other_sales','total_sales'],
aggfunc='sum').reset_index()
```

In [ ]:
```
# grouping by rating, taking sum of sales data
df_esrb_sales = df_esrb.groupby('rating')[['eu_sales','jp_sales', 'na_sales', 'other_sales']].sum().reset_index()
```

In [ ]:
```
# sales based on rating
px.bar(df_esrb_sales, x='rating', y=['eu_sales','jp_sales', 'na_sales', 'other_sales'],
    barmode='group', title='Regional Sales Based on ESRB Rating').show()
```



Conclusions

In the North American region, customers prefer games rated M, and then rated E. The same holds true for the European and other region. In the Japanese region, customers prefer T, then E, followed by M. It appears that these trends share a relationship with the genre preferences of the respective regions. It would stand to reason that shooter and action games are rated M, while sports games are rated E. Role playing games must therefore be rated as T or E.

Hypothesis Testing

Average user rating of XONE and PC are the same

```
Null Hypothesis : The mean user rating of XONE and PC are the same
```

In [ ]:
```
# list of platfroms to filter
xone_list = ['XONE']
pc_list = ['PC']
```

In [ ]:
```
# Filtered dataframe
df_xone_pc = df[df['platform'].isin(xone_list) | df['platform'].isin(pc_list)].dropna()
```

Have to drop missing values for hypothesis testing to work

In [ ]:
```
# Filter again by XONE
xone_user_list = df_xone_pc[df_xone_pc['platform'].isin(xone_list)]
```

In [ ]:
```
# List of user scores
xone_user_list = xone_user_list['user_score'].to_list()
```

In [ ]:
```
# Filter again by PC
pc_user_list = df_xone_pc[df_xone_pc['platform'].isin(pc_list)]
```

In [ ]:
```
# Convert values to list
pc_user_list = pc_user_list['user_score'].to_list()
```

In [ ]:
```
# Comparison of XONE and PC user ratings

#Null Hypothesis

## The mean user ratings of XONE and PC are the same


alpha = 0.05

results = st.ttest_ind(xone_user_list, pc_user_list)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  1.3810936500327673e-05
We reject the null hypothesis, the means are different
```

The average user ratings of XONE and PC platforms are different.

Average user rating of action and sports are the same

Null Hypothesis : The mean user rating of action and sports are the same

In [ ]:
```
# Creating action and sports filter
action = ['Action']
```

```
sports = ['Sports']
```

In [ ]:
```
# Filtered dataframe
df_action_sports = df[df['genre'].isin(action) | df['genre'].isin(sports)].dropna()
```

In [ ]:
```
# Filter again by action
action_user_list = df_action_sports[df_action_sports['genre'].isin(action)]
```

In [ ]:
```
# List of user scores
action_user_list = action_user_list['user_score'].to_list()
```

In [ ]:
```
# Filter again by action
sports_user_list = df_action_sports[df_action_sports['genre'].isin(sports)]
```

In [ ]:
```
# List of user scores
sports_user_list = sports_user_list['user_score'].to_list()
```

In [ ]:
```
# Comparison of Action and Sports user ratings

#Null Hypothesis

## The mean user ratings of Action and Sports are the same


alpha = 0.05

results = st.ttest_ind(action_user_list, sports_user_list)

print('p-value: ', results.pvalue)

if results.pvalue < alpha:
    print("We reject the null hypothesis, the means are different")
else:
    print("We can't reject the null hypothesis")
```

```
p-value:  5.896027231289071e-06
We reject the null hypothesis, the means are different
```

The mean action and sports user ratings are not statistically different.

Overall Conclusions

Overall, we can look at our data to make predictions for 2017. First, inferences would need to be made as to the region we are most interested in marketing to. Since the North American region generally leads in sales, and since the European region follows a similar trend, we can maximize our results by focusing on North America. Since the most popular platform is the PS4, we will be looking for a game on that platform. Since the most popular genre in North America is action, we will be looking for the next big action game. Furthermore, we would want a game that could be played on XONE or PC as well, to maximize sales. Since ratings and user scores do not influence sales, we will not care too much about those factors. Yet, a game in the action genre will likely be M for mature. It is highly likely that the next iteration of Call of Duty or GTA will be widely successful. Alternatively, if we want to market to the Japanese region, we will look for a role playing game on the 3DS. However, profits are likely to be higher working with the previous strategy.