



인공지능 2021

목해은

VideoGames EDA 😊

2019108268 목해은





목차 😊

VideoGames EDA in Kaggle

- 1 모듈 import 및 설정
- 2 Loading data
- 3 Pandas Profiling
- 4 궁금증 해결하기
- 5 느낀점



1. 모듈 import 및 설정



01



모듈 import

목 해 은

```
# 데이터 조작 및 분석을 위한 Python 프로그래밍 언어 용으로 작성된 소프트웨어 라이브러리 입니다.  
import pandas as pd  
# NumPy는 행렬이나 일반적으로 대규모 다차원 배열을 쉽게 처리 할 수 있도록 지원하는 파이썬의 라이브러리이다.  
import numpy as np  
  
# 데이터 시각화 툴 plotly, matplotlib, seaborn  
import plotly.express as px  
import plotly.graph_objects as go  
import plotly.figure_factory as ff  
  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
# 브라우저에서 바로 그림으로 볼 수 있도록 함  
%matplotlib inline  
# pandas-profiling ; 방대한 양의 데이터를 가진 데이터프레임을 한번에 탐색하는 패키지  
from pandas_profiling import ProfileReport  
import pandas.util.testing as tm
```

```
# input 디렉토리 내 파일 가져옴  
import os  
for dirname, _, filenames in os.walk('/kaggle/input'):  
    for filename in filenames:  
        # os.path.join : 경로명 조작에 관한 모듈  
        print(os.path.join(dirname, filename))
```

- 데이터 조작, 분석을 위한 Pandas 라이브러리
- 행렬과 다차원 배열을 쉽게 처리할 수 있는 Numpy 라이브러리
- 시각화 툴 plotly, matplotlib, seaborn

- 데이터 프레임을 한번에 탐색할 수 있는 pandas-profiling

- os.path.join() : 경로명 조작에 관한 모듈

도구 불러오기

필요한 도구들을 불러오거나, 기본적인 경로 등을 설정한다.



설 정

옥 해 은

```
data_file_path = "/kaggle/input/videogamesales/vgsales.csv"
comanie_region_path = "/kaggle/input/videogamescompaniesregions/video-games-developers.csv"
total_sales_column = "Total_Sales"
```

- 파일 경로 설정

```
# Defining all our palette colours.
primary_blue = "#496595"
primary_blue2 = "#85a1c1"
primary_blue3 = "#3f4d63"
primary_grey = "#c6ccd8"
primary_black = "#202022"
primary_bgcolor = "#f4f0ea"

primary_green = px.colors.qualitative.Plotly[2]

plt.rcParams['axes.facecolor'] = primary_bgcolor

colors = [primary_blue, primary_blue2, primary_blue3, primary_grey, primary_black, primary_bgcolor, primary_green]
sns.palplot(sns.color_palette(colors))
```

- 차트 그림 색 설정



설정

기본적인 것을 설정한다.



2. Loading data

데이터 불러오기



02



데이터 불러오기

코어 데이터와 개발 회사 지역 데이터를 불러온다.

```
# Load core data
# 코어 데이터
data_df = pd.read_csv(data_file_path)
data_df.head()
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

```
# Load companie-region data
# 회사 지역 데이터
region_df = pd.read_csv(companie_region_path)
region_df.head()
```

	Developer	City	Administrative division	Country	Est.	Notable games, series or franchises	Notes
0	Overflow	Tokyo	NaN	Japan	1997	School DaysSummer DaysCross Days	Visual Novel brand (both developer and publisher)
1	11 bit studios	Warsaw	Masovian Voivodeship	Poland	2010	Frostpunk	Indie developer/publisher
2	1C Company	Moscow	NaN	Russia	1991	King's Bounty: Warriors of the North	Game localization. The game development subsid...
3	1-Up Studio	Tokyo	NaN	Japan	2000	Mother 3	Subsidiary of Nintendo. Formed by former emplo...
4	2K Czech	Brno	NaN	Czech Republic	1997	MafiaMafia II	Former subsidiary of 2K Games; previously know...



데이터 불러오기

코어 데이터와 개발 회사 지역 데이터를 merge

```
# how='left'는 왼쪽 데이터프레임 기준으로 merge
df = pd.merge(data_df, region_df[['Developer', 'Country']], left_on='Publisher', right_on='Developer', how='left')
df.head()
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Developer	Country
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74	Nintendo	Japan
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	Nintendo	Japan
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82	Nintendo	Japan
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00	Nintendo	Japan
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	Nintendo	Japan

Column :

- Rank
- Name
- Platform
- Year
- Genre
- Publisher
- NA_Sales
- EU_Sales
- JP_Sales
- Other_Sales
- Global_Sales
- Developer
- Country




3. Pandas Profiling




03



```
# EDA(탐색적 데이터 분석; 데이터의 성격 파악하는 과정)
# pandas-profiling ; 방대한 양의 데이터를 가진 데이터프레임을 한번에 탐색하는 패키지
vgames_profile = ProfileReport(df, title='Video Games Profile Report')
# vgames_profile 에 데이터를 프로파일링한 결과 리포트를 저장
```

variables: 100%  13/13 [00:37<00:00, 2.86s/it]

correlations [recoded]: 100%  6/6 [00:12<00:00, 2.10s/it]







interactions [continuous]: 100%  49/49 [00:16<00:00, 2.90it/s]

table: 100%  1/1 [00:06<00:00, 6.02s/it]

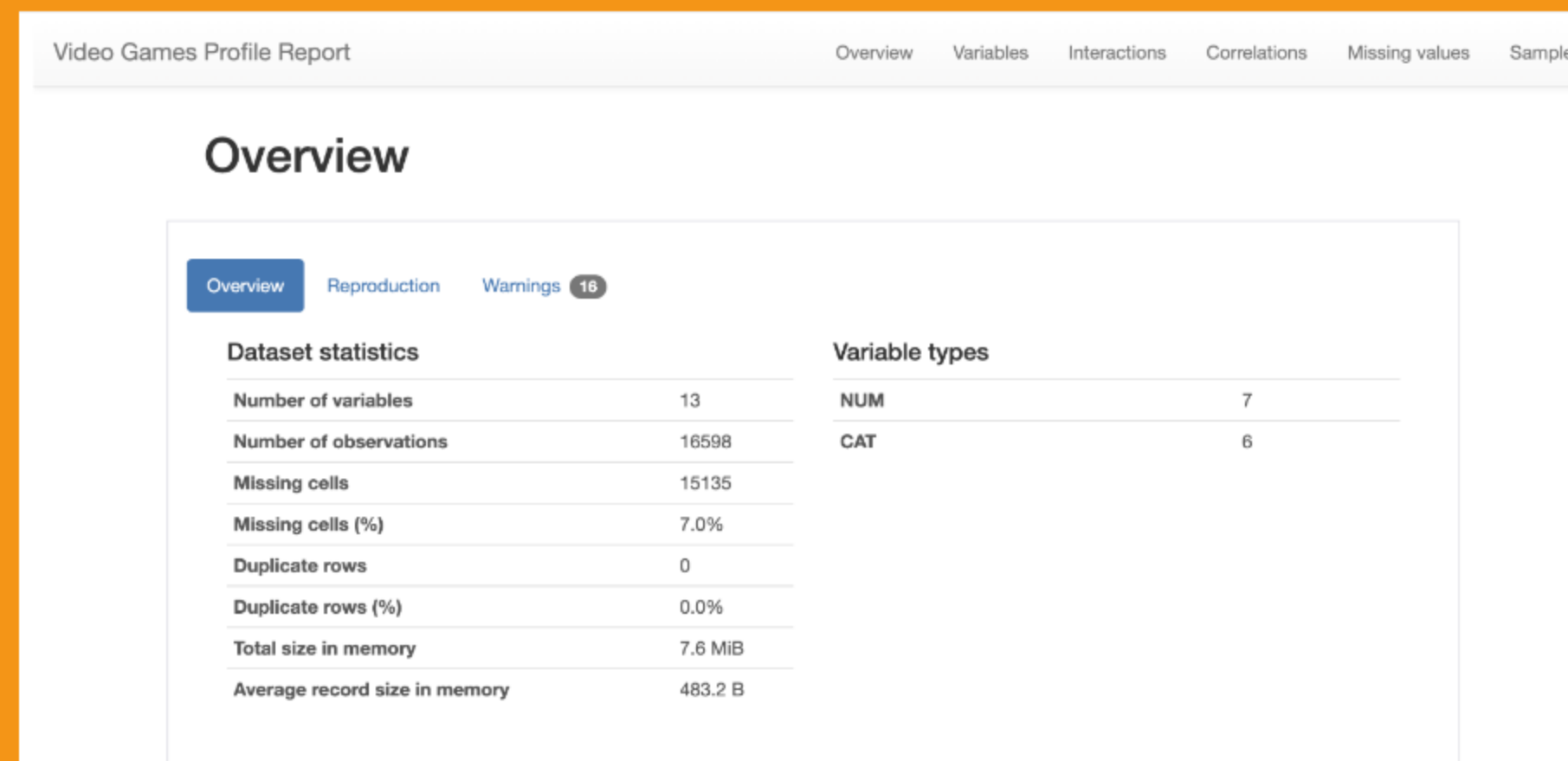
missing [dendrogram]: 100%  4/4 [00:05<00:00, 1.48s/it]

warnings [correlations]: 100%  3/3 [00:00<00:00, 40.07it/s]

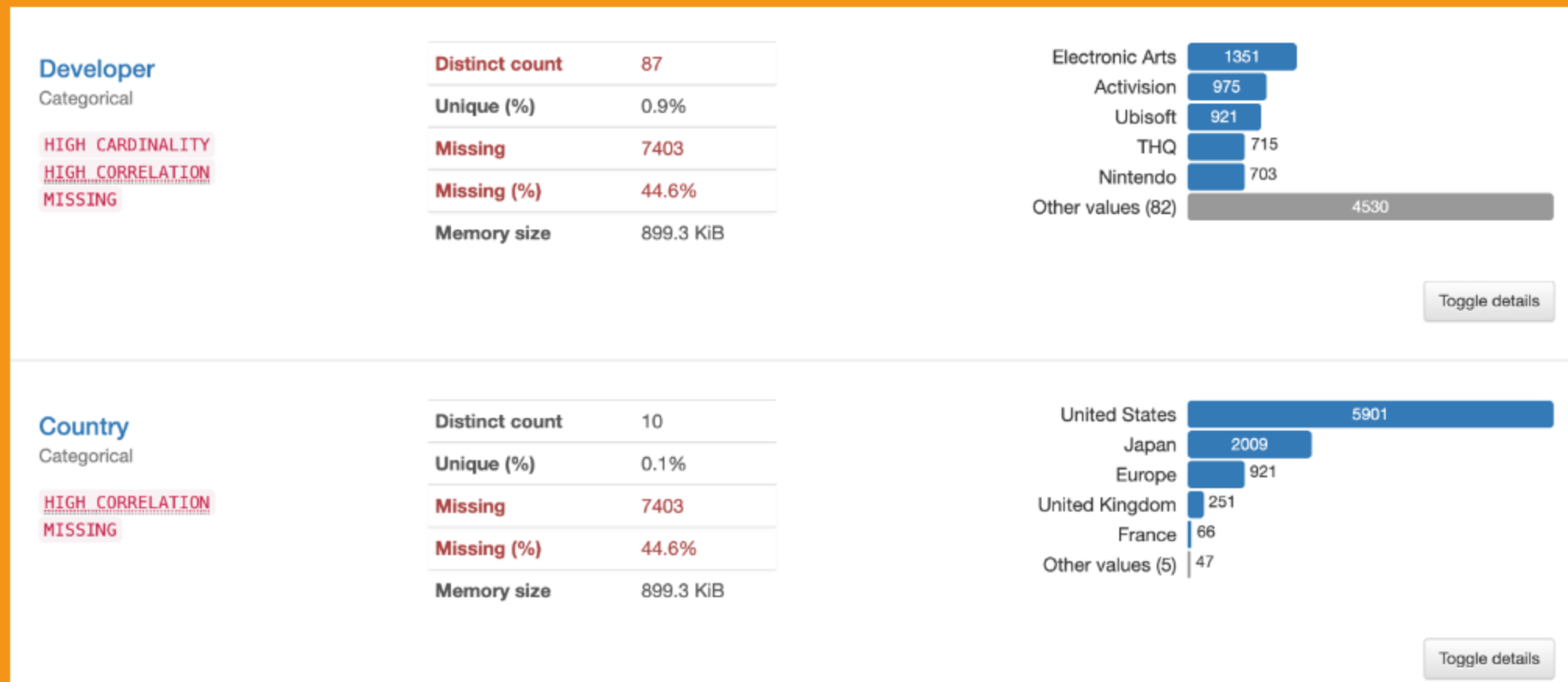
package: 100%  1/1 [00:00<00:00, 3.66it/s]

build report structure: 100%  1/1 [00:16<00:00, 16.82s/it]

Pandas Profiling을 통해 방대한 양의 데이터를 한번에 분석할 수 있었다.



overview, variables, interactions, correlations, missing value, sample
의 카테고리로 나누어 데이터를 분석한 것을 report로 볼 수 있다.



pandas profiling 결과,
변수(columns) 중 Developer 와 Country 항목에서
44.6%의 결측값이 있음을 알 수 있었다.

Developer와 Country 항목을 제외한 변수들 사이의 관계를 분석하는 것이 신뢰도가 높을 것으로 예상



4. 궁금증 해결

데이터 분석을 통해 궁금증을 해결하기



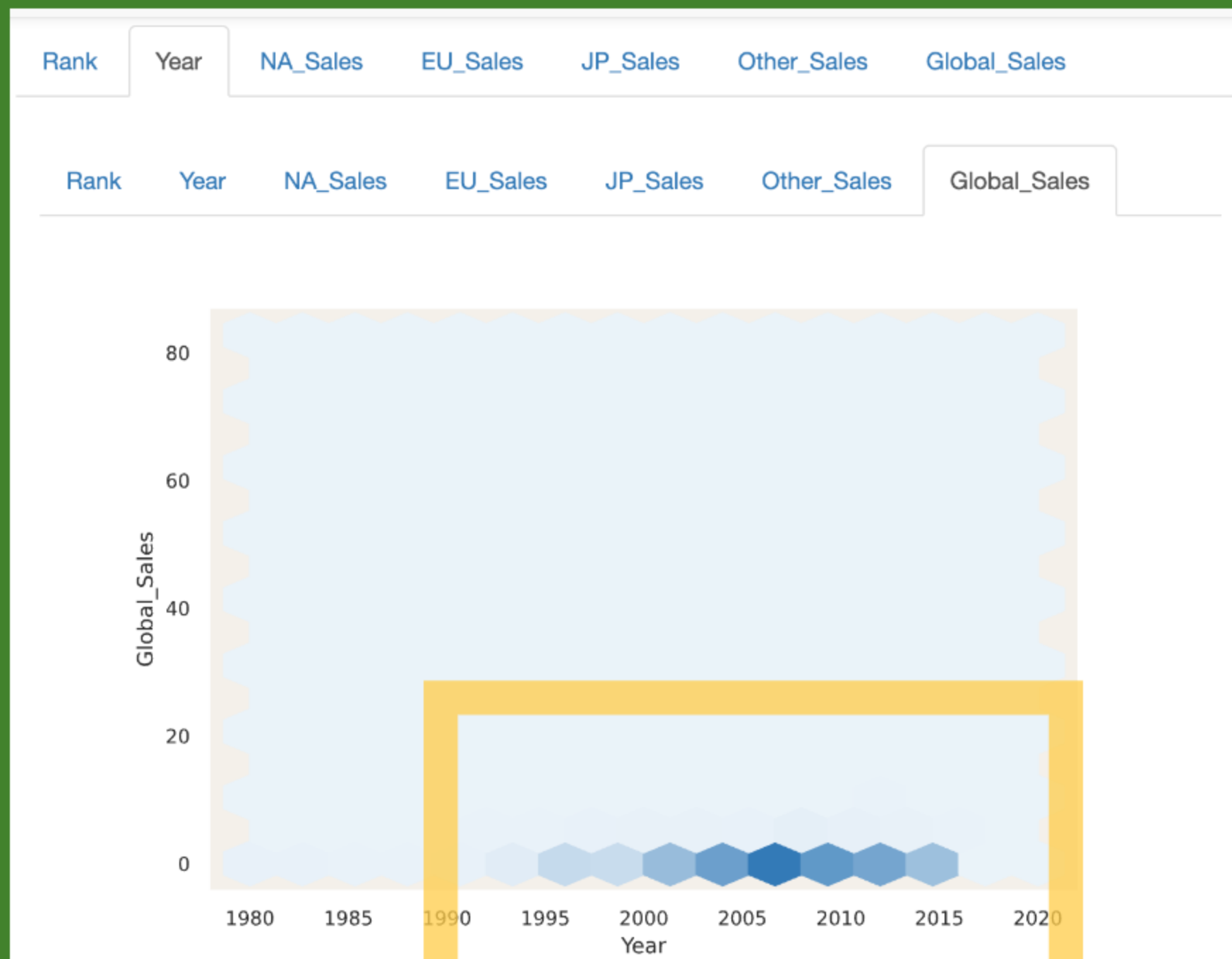
04





궁금증 1. 비디오 게임의 전성기는 언제일까?

Year - Global_Sales 의 관계



in pandas profiling report,
interactions 카테고리
Year - Global_Sales의 관계

2008~2010



궁금증 2.

각 나라별로 판매율이 높은 게임의 장르는 어떻게 다를까?

Nations_Sales - Genre 의 관계

각 나라별 판매 분포

```
if 'Total_Shipped' in df.columns:
    regions = ['NA', 'JP', 'PAL', 'Other']
else:
    regions = ['NA', 'JP', 'EU', 'Other']

region_sales_sufix = '_Sales'
regions_agg = {}

for region in regions:
    regions_agg[region + region_sales_sufix] = 'sum'

regions_agg[total_sales_column] = 'sum'
regions_agg
```

```
{'NA_Sales': 'sum',
 'JP_Sales': 'sum',
 'EU_Sales': 'sum',
 'Other_Sales': 'sum',
 'Total_Sales': 'sum'}
```

```
# Year 컬럼 기준으로 나눠 그룹 생성, agg함수로 집계
geo_tdf = tdf.groupby(['Year']).agg(regions_agg).reset_index()
geo_tdf = geo_tdf.sort_values('Year', ascending=True)
geo_tdf.head(10)
```

	Year	NA_Sales	JP_Sales	EU_Sales	Other_Sales	Total_Sales
0	1980.0	10.59	0.00	0.67	0.12	11.38
1	1981.0	33.40	0.00	1.96	0.32	35.77
2	1982.0	26.92	0.00	1.65	0.31	28.86
3	1983.0	7.76	8.10	0.80	0.14	16.79
4	1984.0	33.28	14.27	2.10	0.70	50.36
5	1985.0	33.73	14.56	4.74	0.92	53.94
6	1986.0	12.50	19.81	2.84	1.93	37.07
7	1987.0	8.46	11.63	1.41	0.20	21.74
8	1988.0	23.87	15.76	6.59	0.99	47.22
9	1989.0	45.15	18.36	8.44	1.50	73.45

regions 리스트에 컬럼에 존재하는 나라의 이름을 넣고, 접미사를 붙여 집계할 때 사용할 리스트 내 값을 "sum"으로 리셋한다. Year 컬럼을 기준으로 나눠 그룹을 생성하고, agg함수를 이용해 같은 그룹 내에 있는 값을 모두 더해 geo_tdf에 넣고 Year를 기준으로 오름차순으로 정렬하여 출력한다.



각 나라별 전체 판매 분포

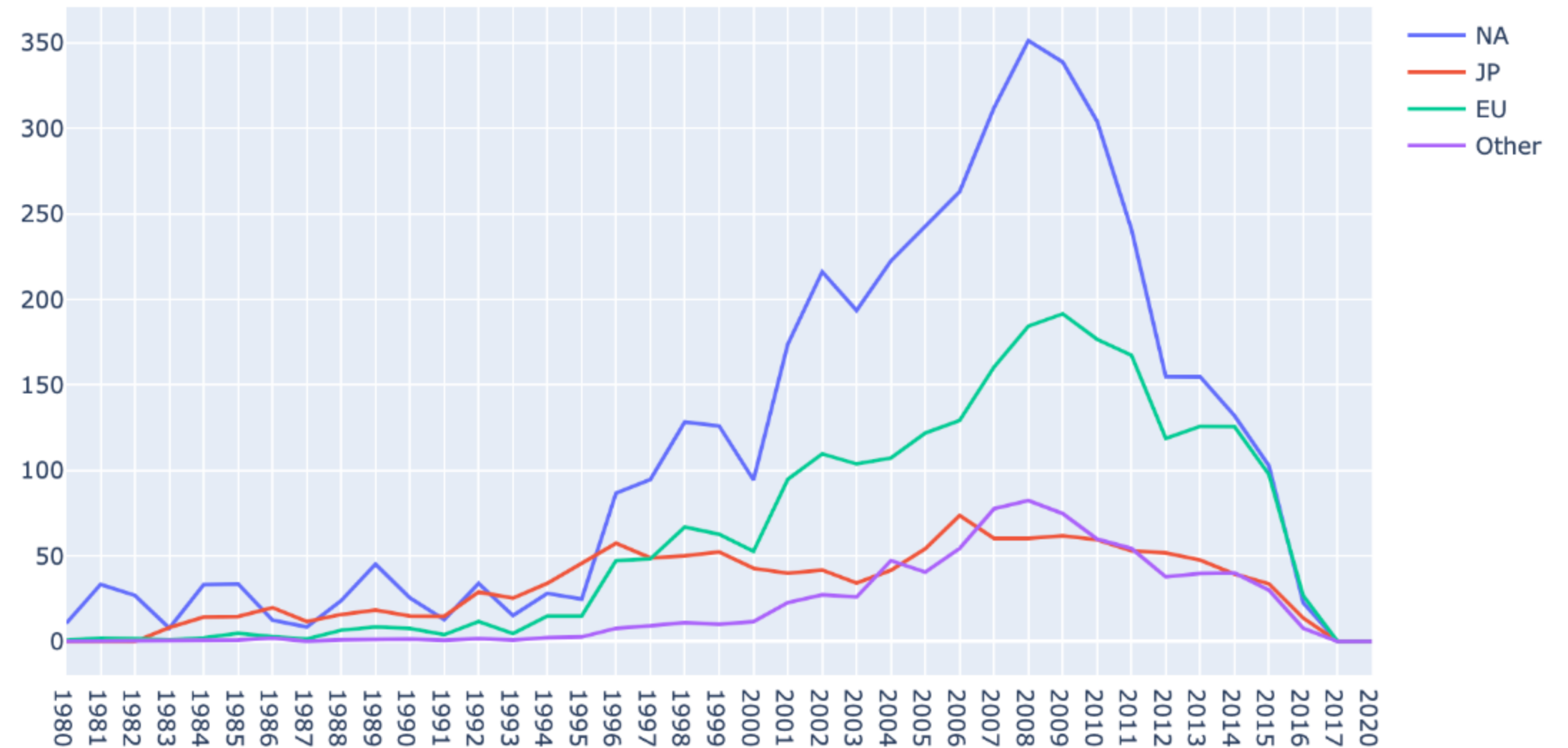
```
fig = go.Figure()

for region in regions:

    fig.add_trace(go.Scatter(
        x=geo_tdf['Year'],
        y=geo_tdf[region + region_sales_sufix],
        mode='lines',
        name=region,
    ))
fig.update_layout(title="Total sales per year by region (Millions)")
fig.update_xaxes(type='category')
fig.show()
```

add_trace를 이용해 x축, y축을 설정하고
선 그래프를 그린다.

Total sales per year by region (Millions)



NA >> EU >> JP >= Other

최근 4년 (2016~2019) 간, Genre - Region 의 관계

```
genre_last_tdf = tdf[tdf['Year'].isin([2016, 2017, 2018, 2019])]
genre_last_tdf = genre_last_tdf.groupby(['Genre']).agg(regions_agg)
genre_last_tdf = genre_last_tdf.sort_values(total_sales_column, ascending=False)
genre_last_tdf.head()
```

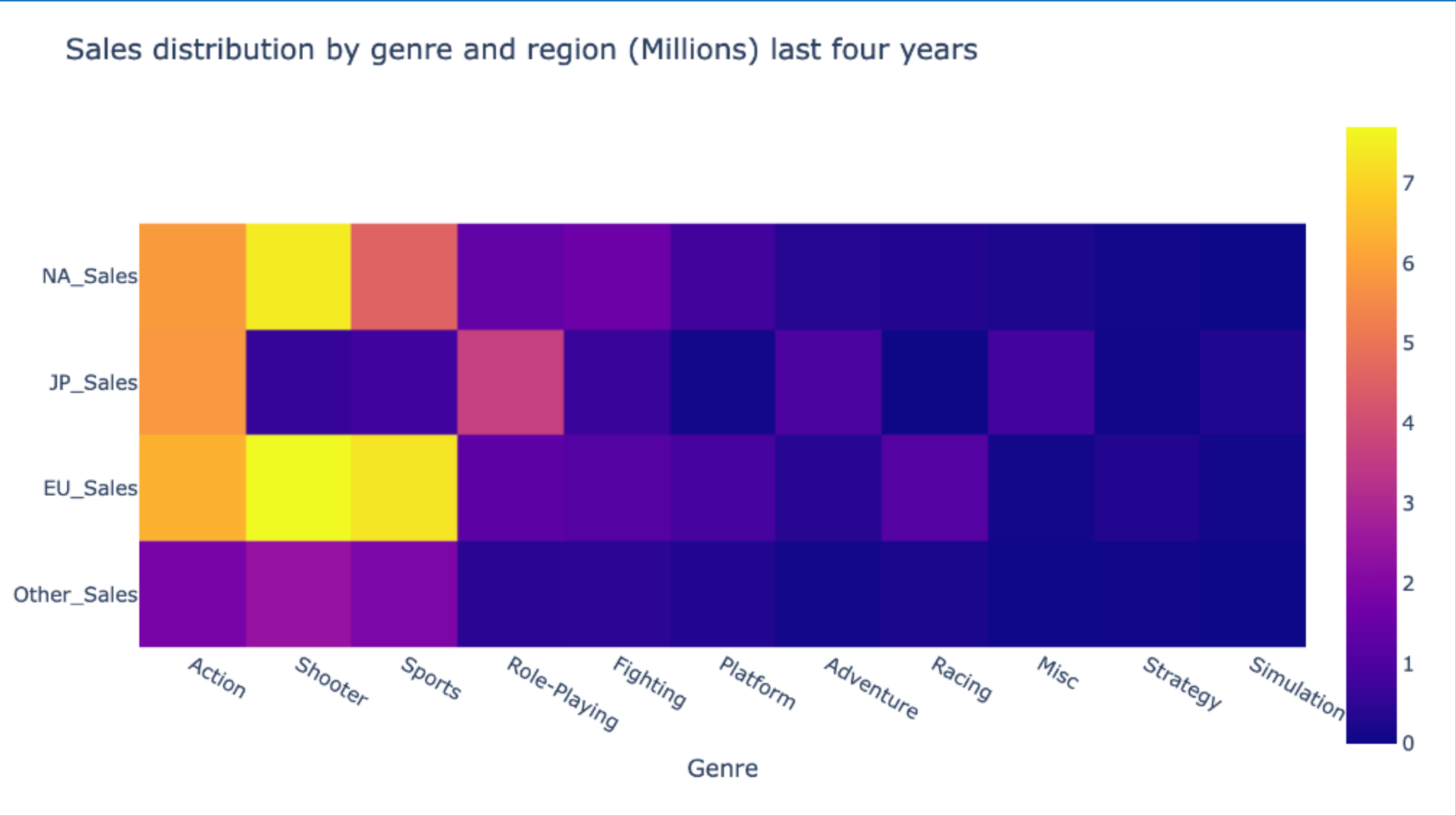
	NA_Sales	JP_Sales	EU_Sales	Other_Sales	Total_Sales
Genre					
Action	5.87	5.80	6.36	1.83	19.92
Shooter	7.44	0.61	7.70	2.42	18.22
Sports	4.57	0.78	7.36	1.92	14.60
Role-Playing	1.39	3.67	1.29	0.44	6.80
Fighting	1.60	0.64	1.15	0.46	3.86

먼저 Year를 기준으로 리스트를 만들고,

Genre 컬럼 기준으로 그룹 생성, agg 함수를 이용하여 각 그룹 내에서 집계.
total_sales 높은 순으로 정렬

```
fig = px.imshow(genre_last_tdf.drop(total_sales_column, 1).T)
fig.update_layout(title="Sales distribution by genre and region (Millions) last four years")
fig.show()
```

px (=plotly.express) 에서 제공하는 imshow 그래프로 시각화

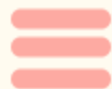


북미(NA) : 슈팅게임 > 액션 > 스포츠

일본(JP) : 액션 > **RPG**

유럽(EU) : 슈팅게임 > 스포츠 > 액션

Other : 슈팅게임 > 스포츠 > 액션



궁금증 3.

게임회사별로 잘 팔리는 장르가 어떻게 다를까?

Genre - Publisher 의 관계



```
publisher_tops = list(pub_tdf['Publisher'])  
len(publisher_tops)
```

```
# Re-create the df to only select top 10 Publishers  
pub_genre_df = tdf.groupby(['Publisher', 'Genre']).agg(regions_agg).reset_index()  
pub_genre_df = pub_genre_df[pub_genre_df['Publisher'].isin(publisher_tops[:10])]  
pub_genre_df = pub_genre_df[pub_genre_df['Genre'].isin(genre_tops)]  
pub_genre_df.head()  
  
pub_genre_pivot_df = pub_genre_df.pivot(index='Publisher', columns='Genre', values='total_sales_column')  
  
z = pub_genre_pivot_df.values  
x = pub_genre_pivot_df.columns.tolist()  
y = pub_genre_pivot_df.index.tolist()  
|  
z_text = np.around(z)  
  
# Create heatmap  
fig = ff.create_annotated_heatmap(z, x=x, y=y, annotation_text=z_text, colorscale='viridis')  
fig.update_xaxes(categoryorder='total_descending')  
fig.update_layout(title="Sales by publisher and genre (Millions)")  
fig.show()
```

Publisher 와 Genre를 묶어 그룹화한 후, 집계.

게임 회사 상위 10개를 추리고 pivot을 이용해 새롭게 데이터를 정리한다.

z축은 값(total_sales), x축은 장르, y축은 회사로 설정하여 히트맵을 만든다.

Sales by publisher and genre (Millions)

	Sports	Action	Shooter	Platform	Misc	Racing	Role-Playing	Simulation	Fighting
Ubisoft	23.0	143.0	68.0	21.0	98.0	16.0	17.0	44.0	7.0
Take-Two Interactive	77.0	211.0	54.0	3.0	11.0	21.0	6.0	1.0	0.0
THQ	13.0	89.0	15.0	41.0	24.0	40.0	1.0	8.0	73.0
Sony Computer Entertainment	59.0	95.0	58.0	104.0	81.0	111.0	44.0	9.0	28.0
Sega	72.0	30.0	17.0	61.0	20.0	22.0	15.0	3.0	15.0
Nintendo	218.0	128.0	70.0	426.0	181.0	151.0	285.0	85.0	53.0
Namco Bandai Games	19.0	37.0	10.0	3.0	30.0	7.0	54.0	10.0	61.0
Konami Digital Entertainment	98.0	70.0	9.0	15.0	18.0	1.0	14.0	32.0	2.0
Electronic Arts	469.0	115.0	158.0	7.0	20.0	146.0	35.0	90.0	31.0
Activision	75.0	142.0	295.0	33.0	77.0	17.0	47.0	8.0	3.0

Nintendo : Platform >> RPG > Sports

Electronic Arts(EA) : Sports

Activision : Shooter



궁금증 4.

가장 판매율이 높은 게임 플랫폼은 어떤 것일까?

Platform - Nations_Sales의 관계

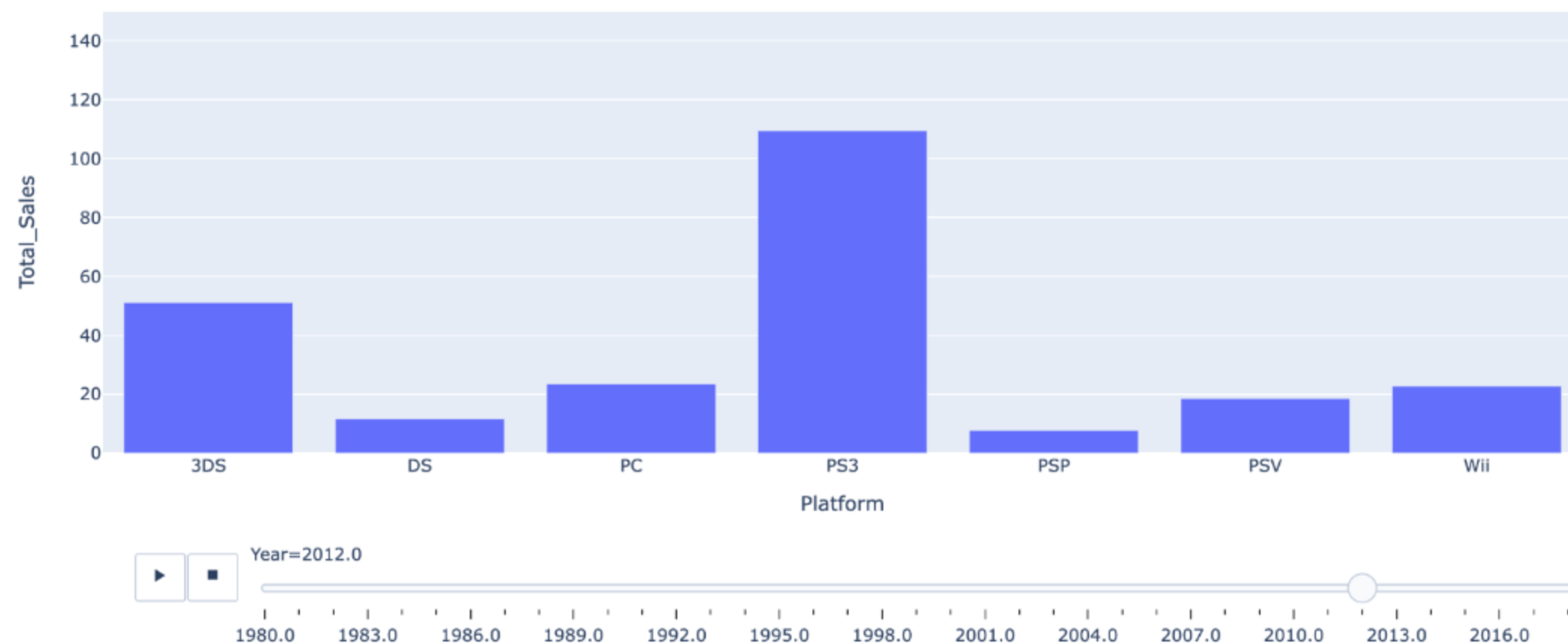
```
platform_tdf = tdf.groupby(['Platform', 'Year']).agg({total_sales_column: 'sum'}).reset_index()
platform_tdf = platform_tdf.sort_values('Year', ascending=True)
platform_tdf.head()
```

	Platform	Year	Total_Sales
0	2600	1980.0	11.38
1	2600	1981.0	35.77
2	2600	1982.0	28.86
3	2600	1983.0	5.83
79	NES	1983.0	10.96

Platform 과 Year 를 중심으로 그룹을 생성.
total_sales를 집계한다.

```
fig = px.bar(
    platform_tdf,
    x='Platform',
    y=total_sales_column,
    animation_frame='Year',
    range_y=[0, 150],
)
fig.update_xaxes(type='category')
fig.update_xaxes(categoryorder='category ascending')
fig.show()
```

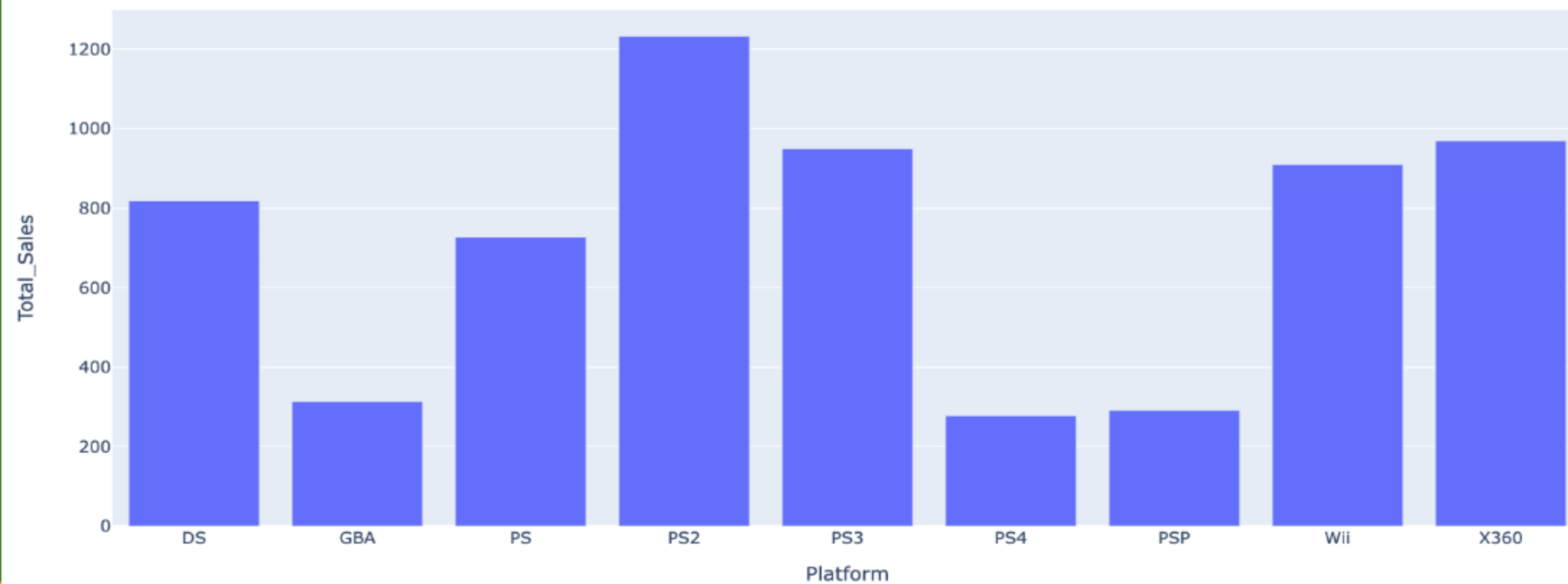
px.bar 를 이용해 x축은 Platform, y축은 total_sales_column
으로 만들고, Year에 따라 애니메이션이 가능한 그래프로 작성



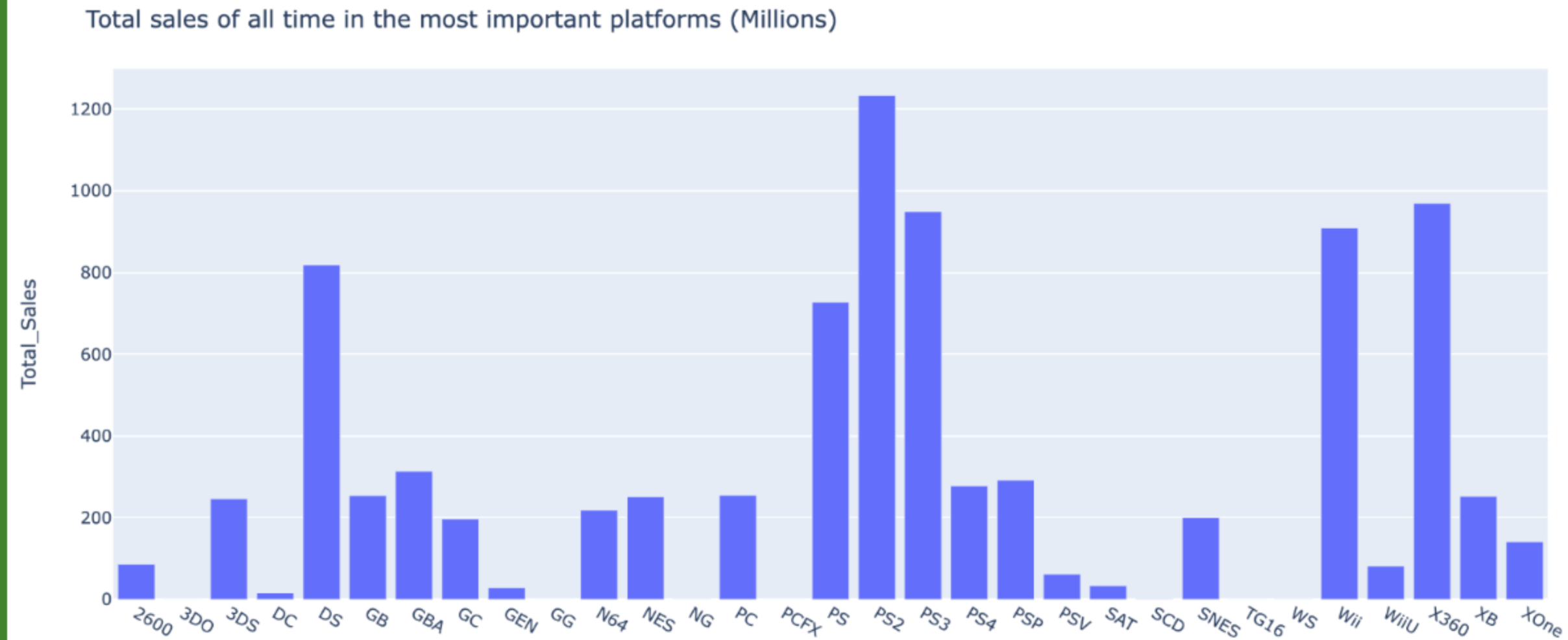

```
platform_sum_tdf = platform_tdf.groupby(['Platform']).agg({total_sales_column: 'sum'}).reset_index()
platform_sum_tdf = platform_sum_tdf[platform_sum_tdf[total_sales_column] > platform_sum_tdf[total_sales_column].sum() * 0.03]
```

```
fig = px.bar(
    platform_sum_tdf,
    x='Platform',
    y=total_sales_column,
)
fig.update_layout(title="Total sales of all time in the most important platforms (Millions)")
fig.update_xaxes(type='category')
fig.update_xaxes(categoryorder='category ascending')
fig.show()
```

Platform 별로 그룹으로 묶어
total_Sales 값을 aggregation (집계)하여
막대 그래프로 나타낸다(px.bar())



```
platform_sum_tdf = platform_tdf.groupby(['Platform']).agg({total_sales_column: 'sum'}).reset_index()  
#platform_sum_tdf = platform_sum_tdf[platform_sum_tdf[total_sales_column] > platform_sum_tdf[total_sales_column].sum() * 0.03]
```

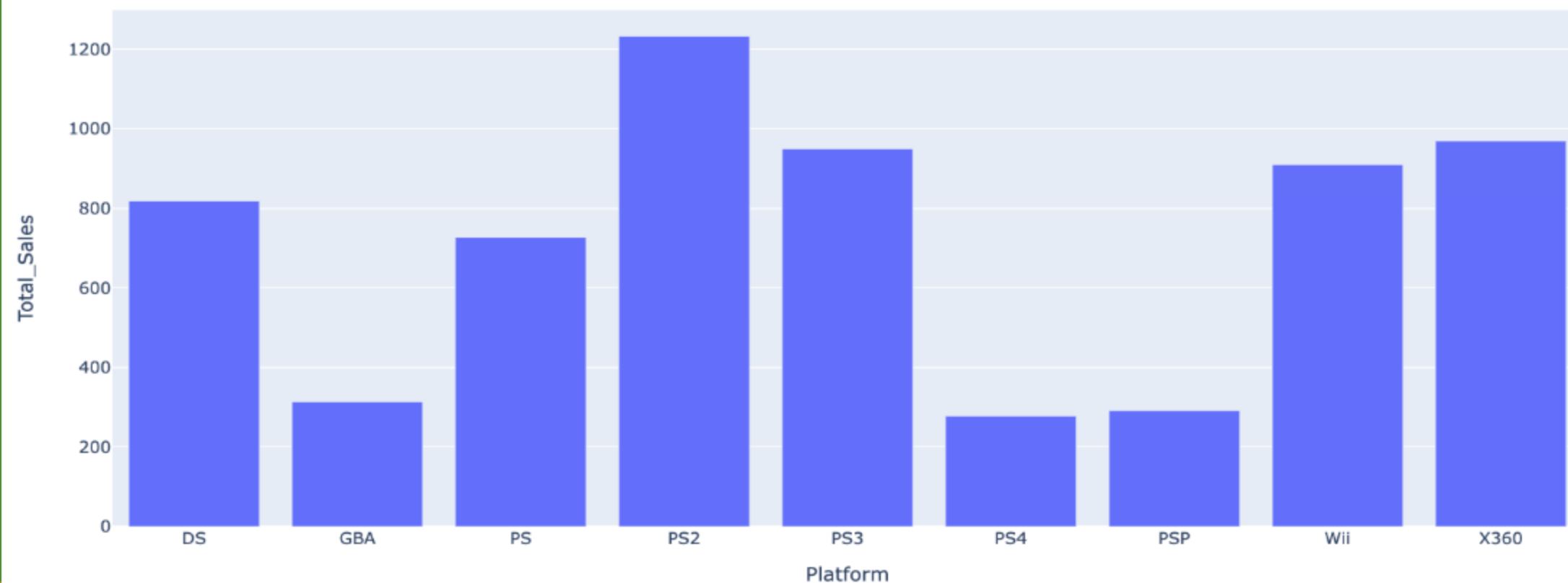


합계의 3% 이상의 값을 가진 Platform만 추출하여 나타내려 한 것..!

```
platform_sum_tdf = platform_tdf.groupby(['Platform']).agg({'total_sales_column': 'sum'}).reset_index()
platform_sum_tdf = platform_sum_tdf[platform_sum_tdf['total_sales_column'] > platform_sum_tdf['total_sales_column'].sum() * 0.03]
```

```
fig = px.bar(
    platform_sum_tdf,
    x='Platform',
    y='total_sales_column',
)
fig.update_layout(title="Total sales of all time in the most important platforms (Millions)")
fig.update_xaxes(type='category')
fig.update_xaxes(categoryorder='category ascending')
fig.show()
```

Platform 별로 그룹으로 묶어
total_Sales 값을 aggregation (집계)하여
막대 그래프로 나타낸다(px.bar())



PS2 > X360 > PS3 > DS > PS



5. 느낀점



05



내가 고른 데이터셋이 활용하기에 어렵지 않게 구성되어 있었고, 접근하기 쉬웠다. 데이터 분석이라고 하면 학문적이고 전문적인 데이터를 다루고 그런 결론을 도출해야 되지 않을까 라는 편견이 있었는데, 이번 과제를 준비하면서 데이터라는 것은 오히려 일상적인 것들에 대한 관심에서 시작되고 모아지는 것이라는 생각이 들었다. 처음에는 어디서부터 어떻게 이해해야하는지, 이해할 수는 있을지 복잡하고 두려웠지만, 코드를 하나하나 읽어보고 검색해보면서 어려워보이기만 했던 것들이 완전히는 아니더라도 조금씩 이해가 되는 것이 신기했다.

가장 마음에 들었던 기술은 Pandas profiling이었다. 간단한 코드 단 몇 줄만으로, 방대한 데이터를 정리해 한 눈에 보기 좋도록 만들었다. 세세한 분석에 앞서 전체적인 데이터의 형태를 보고, 이 데이터가 분석을 통해 결과적으로 의미를 가질 수 있을지에 대하여 가볍게 추측하기에 아주 좋은 툴이라고 생각한다.

데이터가 우리의 일상에 깊게 스며들어있고, 오히려 이젠 세상과 데이터가 동일한 것이라고 말할 수 있을만큼 중심이 된 이 시기에 인공지능이라는 것을 접할수는 있었으나 가깝게 다뤄볼 일은 없었다. 인공지능이 빠르게 발전하고 있다지만, 이번 과제를 통해 느낀 것은 인간의 대단함이었다. 이 많은 데이터들도 사람이 만든 것이고, 이 모든 툴도 사람이 만들었다는 것은 인공지능의 모든 것이 결국 인간의 손에서 나온다는 것을 의미하고 있었다.

훗날 인공지능의 분야가 아니더라도, 미래인들에게 인공지능보다 인간이 더 대단하다는 것을 보여줄 수 있는 어떤 것을 만드는 일에 나 또한 기여하고 싶다고 생각했다.



THANK 😊 YOU!

