

1 : Sales Analysis

NB!! This pdf file consist of 5 project

By Nelio Lino Nhacolo

import libraries:

```
In [4]: import pandas as pd
import os
```

Merging 12 months of sales data into a single csv file:

Task 1 : Merge the 12 months of sales data into a single csv file

```
In [5]: df = pd.read_csv("C:/Users/nelio/Documents/python/Jupyter/Pandas-Data-Science-Tasks-master/SalesAnalysis/Sales_D

files = [file for file in os.listdir("C:/Users/nelio/Documents/python/Jupyter/Pandas-Data-Science-Tasks-master/S

all_month_data = pd.DataFrame()

for file in files:
    df = pd.read_csv("C:/Users/nelio/Documents/python/Jupyter/Pandas-Data-Science-Tasks-master/SalesAnalysis/Sal
    all_month_data = pd.concat([all_month_data, df])

all_month_data.to_csv("all_data.csv", index= False)
```

Read in updated dataframe

In [6]: *#We read all the merged files that we created*

```
all_data = pd.read_csv("C:/Users/nelio/Documents/python/Jupyter/Pandas-Data-Science-Tasks-master/SalesAnalysis/C
all_data.head()
```

Out[6]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

Clean up the data:

Drop rows of NAN

In [7]: *#See all columns with non values and drop them*

```
nan_df = all_data[all_data.isna().any(axis=1)]
nan_df.head()

all_data = all_data.dropna(how='all')
all_data.head()
```

Out[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Now we find 'Or' and delete it

Now we find 'Or' and delete it

```
In [8]: #Now we pass a conditons to find out whats causing the error
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
```

Convert columns to the correct data type

```
In [9]: all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
```

Augment data with additional columns:

Task 2: Add Month Column

```
In [10]: #We add a table and convert it into int
#NB!! best clean the data first
all_data['month'] = all_data["Order Date"].str[0:2]
all_data['month'] = all_data['month'].astype('int32')
all_data.head()
```

Out[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

Task 3: Add a sales column

```
In [11]: all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']  
all_data.head()
```

Out[11]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

Task 4: Add a city column

```
In [12]: # Let's use .apply()
#We can't get the city and leave out post code
#We can remove the zip code
def get_city(address):
    return address.split(',')[1]

def get_state(address):
    return address.split(',')[2].split(" ")[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)} ({get_state(x)})")
all_data.head()
```

Out[12]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)

Question 1:What was the best month for sales? How much was earned that month?

```
In [13]: results = all_data.groupby('month').sum()  
results
```

Out[13]:

	Quantity Ordered	Price Each	Sales
month			
1	10903	1811768.38	1822256.73
2	13449	2188884.72	2202022.42
3	17005	2791207.83	2807100.38
4	20558	3367671.02	3390670.24
5	18667	3135125.13	3152606.75
6	15253	2562025.61	2577802.26
7	16072	2632539.56	2647775.76
8	13448	2230345.42	2244467.88
9	13109	2084992.09	2097560.13
10	22703	3715554.83	3736726.88
11	19798	3180600.68	3199603.20
12	28114	4588415.41	4613443.34

In [18]: *#The best month was december, we can also plot the results*

```
import matplotlib.pyplot as plt
```

```
months = range(1,13)
```

```
print(months)
```

```
plt.bar(months,all_data.groupby(['month']).sum()['Sales'])
```

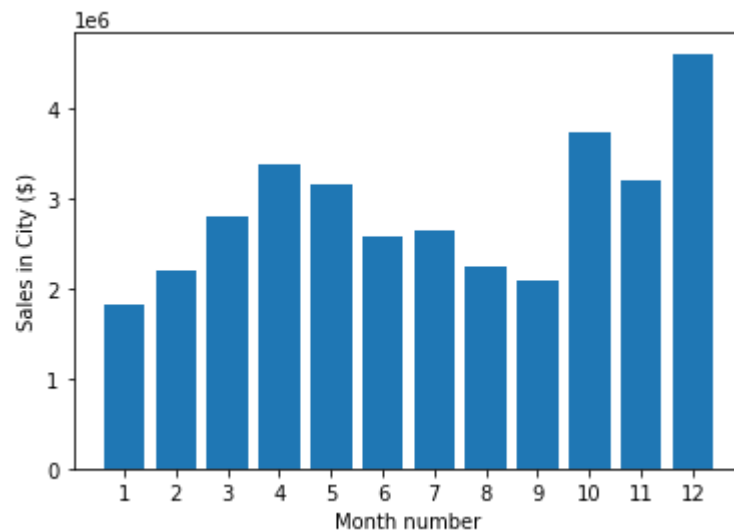
```
plt.xticks(months)
```

```
plt.ylabel('Sales in City ($)')
```

```
plt.xlabel('Month number')
```

```
plt.show()
```

```
range(1, 13)
```



Question 2: What city had the highest of sales?

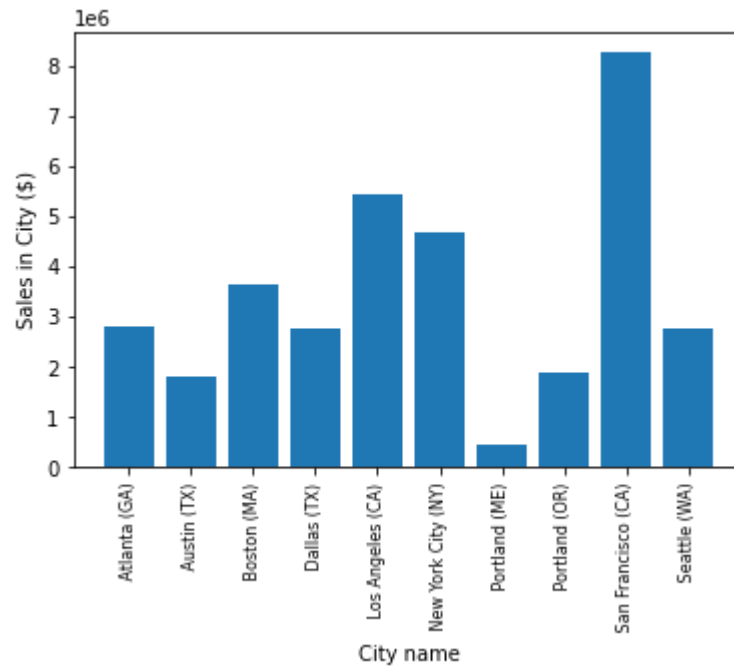
```
In [17]: results = all_data.groupby('City').sum()  
results
```

Out[17]:

	Quantity Ordered	Price Each	month	Sales
City				
Atlanta (GA)	16602	2779908.20	104794	2795498.58
Austin (TX)	11153	1809873.61	69829	1819581.75
Boston (MA)	22528	3637409.77	141112	3661642.01
Dallas (TX)	16730	2752627.82	104620	2767975.40
Los Angeles (CA)	33289	5421435.23	208325	5452570.80
New York City (NY)	27932	4635370.83	175741	4664317.43
Portland (ME)	2750	447189.25	17144	449758.27
Portland (OR)	11303	1860558.22	70621	1870732.34
San Francisco (CA)	50239	8211461.74	315520	8262203.91
Seattle (WA)	16553	2733296.01	104941	2747755.48


```
In [19]: cities = [city for city, df in all_data.groupby("City")]

plt.bar(cities, all_data.groupby(['City']).sum()['Sales'])
plt.xticks(cities, rotation="vertical", size=8)
plt.ylabel('Sales in City ($)')
plt.xlabel('City name')
plt.show()
```



Question 3: What time should we display advertisements to maximize likelihood of customer's buying product?

```
In [20]: #We will use the date time library
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
```

```
In [21]: all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data['Count'] = 1
all_data.head()
```

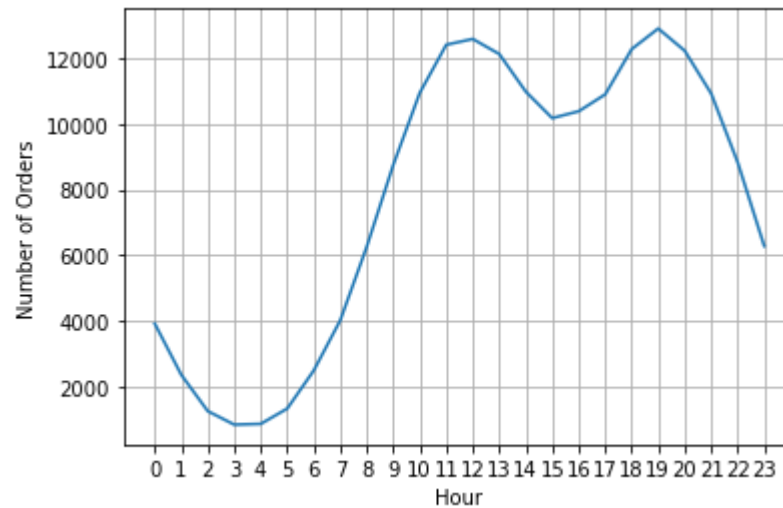
Out[21]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales	City	Hour	Minute	Count
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)	8	46	1
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)	22	30	1
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)	14	38	1
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	14	38	1
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)	9	27	1

```
In [22]: keys = [pair for pair, df in all_data.groupby(['Hour'])]

plt.plot(keys, all_data.groupby(['Hour']).count()['Count'])
plt.xticks(keys)
plt.xlabel('Hour')
plt.ylabel("Number of Orders")
plt.grid()
plt.show()

# My recommendation is around 11am(11) or 7pm(19)
```



Question 4: What products are most often sold together?

```
In [23]: #We check any duplicating ID's
#Then we create a table and group them together
#Then we drop the duplicates

df = all_data[all_data['Order ID'].duplicated(keep=False)]

df ['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ",".join(x))

df = df[["Order ID", 'Grouped']].drop_duplicates()
df.head()
```

C:\Users\nelio\AppData\Local\Temp\ipykernel_54424\4160259238.py:7: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df ['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ",".join(x))
```

Out[23]:

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable
30	176585	Bose SoundSport Headphones,Bose SoundSport Hea...
32	176586	AAA Batteries (4-pack),Google Phone
119	176672	Lightning Charging Cable,USB-C Charging Cable

```
In [24]: #Count paires that occur the most
from itertools import combinations
from collections import Counter

count = Counter()
for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

for keys, value in count.most_common(10):
    print(keys, value)
```

```
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

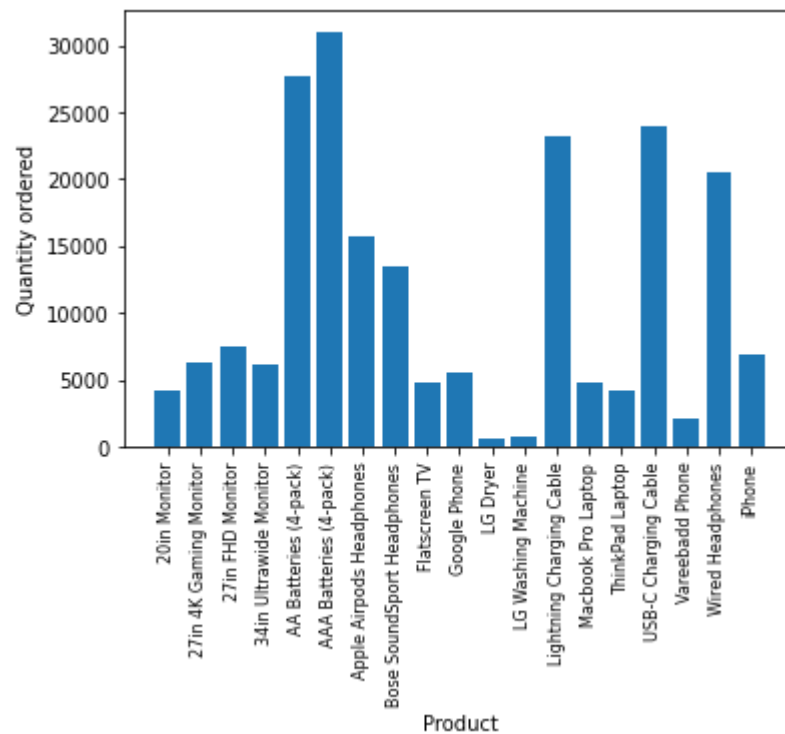
Question 5: What product sold the most? Why do you think it sold the most?

```
In [25]: product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

products = [product for product, df in product_group]

plt.bar(products, quantity_ordered)
plt.ylabel("Quantity ordered")
plt.xlabel("Product")
plt.xticks(products, rotation="vertical", size=8)

plt.show()
```



```
In [ ]: #Triple A batteries, Double A batteries, USB-C Charging Cable, Wired Headphones, Lightning Charging Cable were th  
#That might be because they are cheaper and in demand(more people use them daily)
```

```
In [28]: #Proving my hypothesis
prices = all_data.groupby('Product').mean()['Price Each']

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(products, qauntity_ordered, color='blue')
ax2.plot(products, prices, color='r')

ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='blue')
ax2.set_ylabel('Price ($)', color='r')
ax1.set_xticklabels(products, rotation='vertical', size=8)

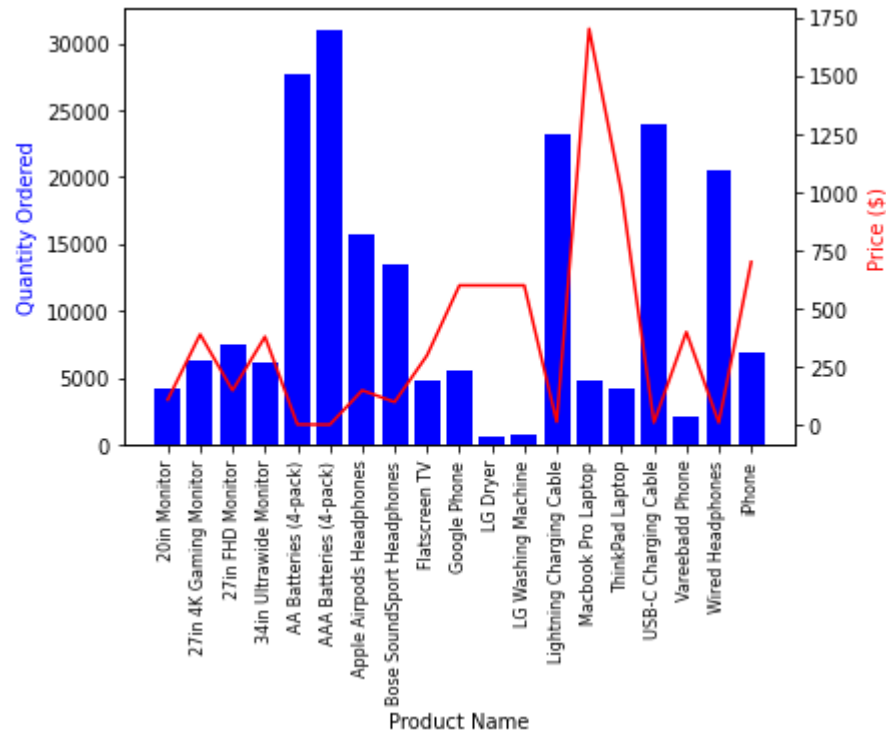
fig.show()
```

C:\Users\nelio\AppData\Local\Temp\ipykernel_54424\1855865478.py:13: UserWarning: FixedFormatter should only be used together with FixedLocator

```
ax1.set_xticklabels(products, rotation='vertical', size=8)
```

C:\Users\nelio\AppData\Local\Temp\ipykernel_54424\1855865478.py:15: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.

```
fig.show()
```

In [105]: *#So whenever the Quatinty ordered is high, the price is low.
 #And when the Qauntity ordered is low, that means the price is high.*

*#We can see that the pice for Macbook pro and ThinkPad Laptops is higher than the LG Dryer and LG washing machin
 #that might be because there are more students, organizations and small business that use Laptops, so they are i*

Thank you!!

In []:

In []:

2: Video game sales

Task 1: import data

```
In [2]: #shape: tells us how many records we have and how many columns we have
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.impute import SimpleImputer

game_data = pd.read_csv('VideoGamesSales.csv')
game_data.shape
```

Out[2]: (16598, 11)

```
In [3]: #describe: returns basic information of each columns count, mean, std, min etc..
game_data.describe()
```

Out[3]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	16598.000000
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	0.537441
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	1.555028
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.470000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

```
In [4]: #Will return the values
game_data.values
```

```
Out[4]: array([[1, 'Wii Sports', 'Wii', ..., 3.77, 8.46, 82.74],
 [2, 'Super Mario Bros.', 'NES', ..., 6.81, 0.77, 40.24],
 [3, 'Mario Kart Wii', 'Wii', ..., 3.79, 3.31, 35.82],
 ...,
 [16598, 'SCORE International Baja 1000: The Official Game', 'PS2',
 ..., 0.0, 0.0, 0.01],
 [16599, 'Know How 2', 'DS', ..., 0.0, 0.0, 0.01],
 [16600, 'Spirits & Spells', 'GBA', ..., 0.0, 0.0, 0.01]],
 dtype=object)
```

Task 2: Lets clean the data

```
In [5]: #We check the null values
null_values = game_data[game_data.isna().any(axis=1)]
null_values.head()
```

Out[5]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
179	180	Madden NFL 2004	PS2	NaN	Sports	Electronic Arts	4.26	0.26	0.01	0.71	5.23
377	378	FIFA Soccer 2004	PS2	NaN	Sports	Electronic Arts	0.59	2.36	0.04	0.51	3.49
431	432	LEGO Batman: The Videogame	Wii	NaN	Action	Warner Bros. Interactive Entertainment	1.86	1.02	0.00	0.29	3.17
470	471	wwe Smackdown vs. Raw 2006	PS2	NaN	Fighting	NaN	1.57	1.02	0.00	0.41	3.00
607	608	Space Invaders	2600	NaN	Shooter	Atari	2.36	0.14	0.00	0.03	2.53

```
In [6]: #Now lets clean the data
game_data = game_data.dropna(how='all')
game_data.head()
```

Out[6]:

	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

Task 2: remove unwanted and modify columns

```
In [7]: #Let's remove columns that we might not need or use
#Let's remove the Platform and the Publisher column

game_data.drop(['Platform', 'Publisher'], axis=1, inplace=True)
game_data.head()
```

Out[7]:

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

In [8]: `game_data.head()`

Out[8]:

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

In [9]: *#Let's modify*
#It's to name your columns in a way that anyone can understand them
`game_data.rename(columns={'NA_Sales': 'National_Sales', 'EU_Sales': 'European_Sales', 'JP_Sales': 'Japan'}, inplace=True)`
`game_data.head()`

Out[9]:

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

Task 3: Lets say we want to see the total sales in each Name followed by sales.

```
In [28]: total_sales = game_data.groupby('Name')[['Name', 'National_Sales', 'European_Sales', 'Japan', 'Other_Sales', 'Global_Sales']]
total_sales.head()
```

Out[28]:

	Name	National_Sales	European_Sales	Japan	Other_Sales	Global_Sales
0	'98 Koshien	0.15	0.10	0.12	0.03	0.41
1	.hack//G.U. Vol.1//Rebirth	0.00	0.00	0.17	0.00	0.17
2	.hack//G.U. Vol.2//Reminisce	0.11	0.09	0.00	0.03	0.23
3	.hack//G.U. Vol.2//Reminisce (jp sales)	0.00	0.00	0.16	0.00	0.16
4	.hack//G.U. Vol.3//Redemption	0.00	0.00	0.17	0.00	0.17

Task 4: Lets say we want to find out Global_sales which are more than a 100

```
In [29]: Larger = total_sales[total_sales['Global_Sales']>100]
Larger.head()
```

Out[29]:

	Name	National_Sales	European_Sales	Japan	Other_Sales	Global_Sales
350	Animal Crossing: Wild World	2.55	3.52	5.33	0.88	12.27
487	Assassin's Creed	5.20	4.48	0.16	1.46	11.30
490	Assassin's Creed II	5.65	3.96	0.29	1.49	11.41
492	Assassin's Creed III	6.24	4.93	0.19	1.74	13.10
494	Assassin's Creed IV: Black Flag	6.17	5.18	0.20	1.65	13.16

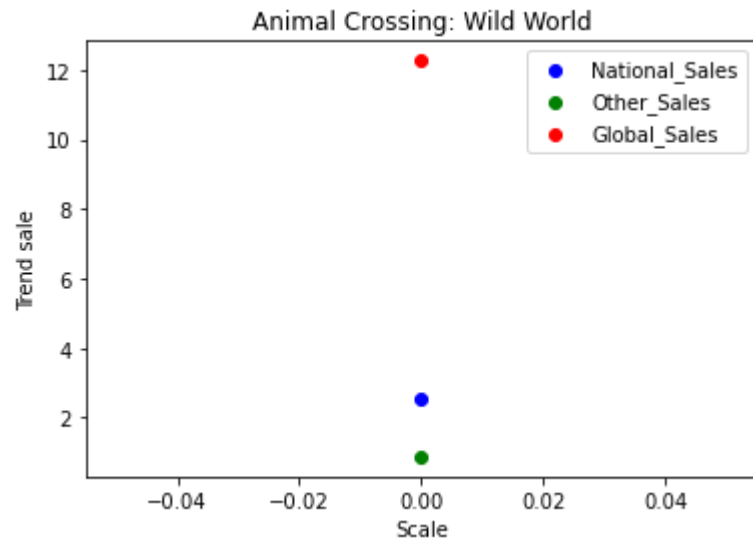
Task 5: Analyse the data

```
In [30]: #Let's first check, how many unique names are there
names = Larger['Name'].unique()
len(names)
```

Out[30]: 96

```
In [53]: #Now let's see what is the trend of Names in respective National_Sales, Other_Sales and Global_sales
#We gonna loop over each name.
```

```
for n in range(0, len(names)):
    C = Larger[Larger['Name'] == names[n]].reset_index()
    plt.scatter(np.arange(0, len(C)), C['National_Sales'], color='blue', label='National_Sales')
    plt.scatter(np.arange(0, len(C)), C['Other_Sales'], color='green', label='Other_Sales')
    plt.scatter(np.arange(0, len(C)), C['Global_Sales'], color='red', label='Global_Sales')
    plt.title(names[n])
    plt.xlabel("Scale")
    plt.ylabel('Trend sale')
    plt.legend()
    plt.show()
```



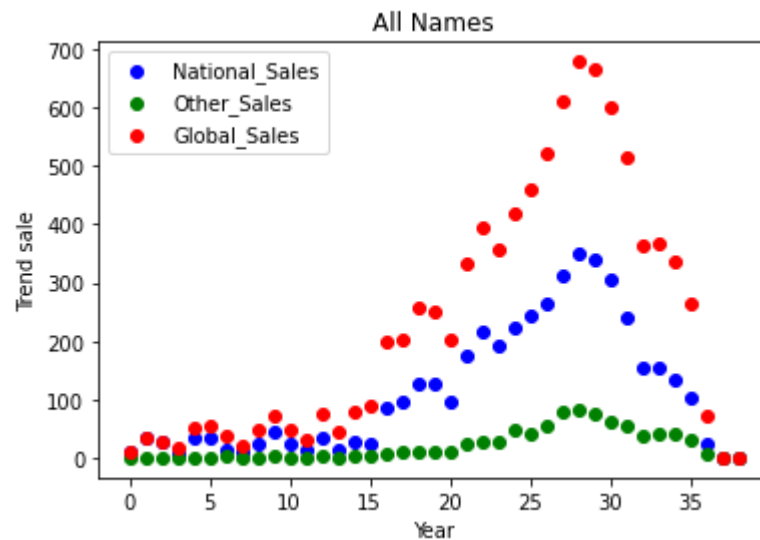
Task6: Lets see the overall sales of each column grouped by year

```
In [20]: over_all = game_data.groupby(['Year'])[['National_Sales', 'Other_Sales', 'Global_Sales']].sum().reset_index()
over_all.head()
```

Out[20]:

	Year	National_Sales	Other_Sales	Global_Sales
0	1980.0	10.59	0.12	11.38
1	1981.0	33.40	0.32	35.77
2	1982.0	26.92	0.31	28.86
3	1983.0	7.76	0.14	16.79
4	1984.0	33.28	0.70	50.36

```
In [27]: C = over_all
plt.scatter(np.arange(0,len(C)),C['National_Sales'],color='blue',label='National_Sales')
plt.scatter(np.arange(0,len(C)),C['Other_Sales'],color='green',label='Other_Sales')
plt.scatter(np.arange(0,len(C)),C['Global_Sales'],color='red',label='Global_Sales')
plt.title('All Names')
plt.xlabel("Year")
plt.ylabel('Trend sale')
plt.legend()
plt.show()
```



Thank you!!

In []:

In []:

3: Online music store

Task 1: Import data

```
In [36]: import pandas as pd
music_info = pd.read_csv('music.csv')
music_info
```

Out[36]:

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

Task 2: Clean the data

```
In [ ]: #In this data set we don't have any null values, so we don't need to do any cleaning.  
#I included this task just to show a step by step guide to follow.
```

Task 3: Split the data into Training/ test Sets

```
In [40]: #When we train a model we give them two sets, the input set(age,gender) and the output set(genre).  
inputSet = music_info.drop(columns=['genre'])  
outputSet = music_info['genre']
```

```
Out[40]: 0      HipHop  
1      HipHop  
2      HipHop  
3       Jazz  
4       Jazz  
5       Jazz  
6    Classical  
7    Classical  
8    Classical  
9       Dance  
10      Dance  
11      Dance  
12    Acoustic  
13    Acoustic  
14    Acoustic  
15    Classical  
16    Classical  
17    Classical  
Name: genre, dtype: object
```

Task 4: Create a Model

```
In [42]: #We are now gonna create a model using an algorithmn  
#Each algorithmn has it's own pros and cons  
#We gonna ask our model to predict the type of music our uses like based on gender and age.  
from sklearn.tree import DecisionTreeClassifier  
  
model = DecisionTreeClassifier()  
model.fit(inputSet, outputSet)  
predictions = model.predict([ [21, 1], [22, 0] ])  
predictions
```

C:\Users\nelio\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
warnings.warn(

```
Out[42]: array(['HipHop', 'Dance'], dtype=object)
```

Task 5: Train the model

```
In [54]: #Now we gonna make our model calculate accurately  
#We gonna allocate 70-80% of our data to training and 20% to testing  
  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score  
inputSet_train, inputSet_test, outputSet_train, outputSet_test = train_test_split(inputSet, outputSet, test_size=0.2)  
model.fit(inputSet_train, outputSet_train)  
predictions = model.predict(inputSet_test)  
  
#Will return an accuracy score from 0 - 1 and they will change each time  
#Press control enter if u don't want to create a cell below each time u run this cell.  
score = accuracy_score(outputSet_test, predictions)  
score
```

```
Out[54]: 1.0
```

Task 6: Make Predictions

```
In [60]: #Please keep in mind that we dealing with a small data set  
#We gonna create model persistence: we gonna build and train our model,  
#save it a file and use it whenever we want(It's like an intellegent person).  
#You can also use (from sklearn.externals import joblib).  
  
import joblib  
#we just created a musci-recommendation file(will be attached along side this pdf file.)  
joblib.dump(model, 'music-recommendation.joblib')  
  
#We load the file  
model = joblib.load('music-recommendation.joblib')  
  
predictions = model.predict([ [21, 1], [22, 0] ])  
predictions
```

```
C:\Users\nelio\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names  
warnings.warn(
```

```
Out[60]: array(['HipHop', 'Dance'], dtype=object)
```

Task 7: Evaluate and Improve

```
In [64]: #We gonna export this model in a virtual format, to see how it makes predictions.
#It's gonna be cool.
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

music_info = pd.read_csv('music.csv')
inputSet = music_info.drop(columns=['genre'])
outputSet = music_info['genre']

model = DecisionTreeClassifier()
model.fit(inputSet, outputSet)

#we gonna use key words arguments and parameters
#filled: each box or not is field with colors
#rounded: each box has a rounded shape
#label: they all have labels
#class-names: display the class for it node
#feature_names:we can see the rules in our nodes.

tree.export_graphviz(model, out_file="music-recommendation.dot",
                      feature_names=['age', 'gender'],
                      class_names=sorted(outputSet.unique()),
                      label='all',
                      rounded=True,
                      filled=True)
```

Open Visual studio code

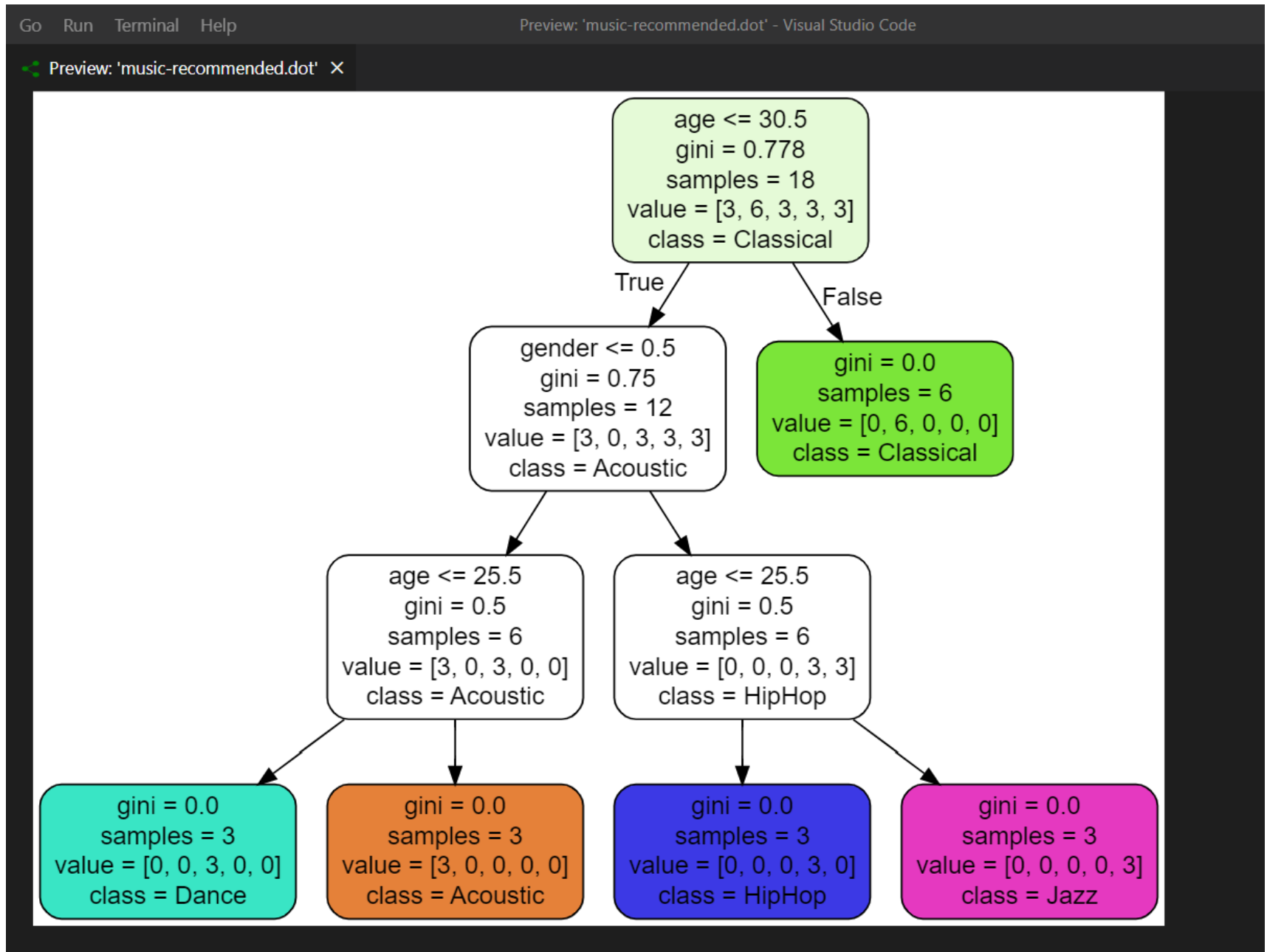
drag and drop the .dot file

click the extension and download graphviz(dot)

restart vsc and top left corner click the three toggles

You will see "open preview", click on it.

Run to see the image



Thank you!!

In []:

In []:

4: Commercial data

In [1]: *#Our job is to have a look at the commercial data for the year and present it.*

Task 1: Import Libraries & Load Dataset

```
In [2]: # Imports:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
from pandas_profiling import ProfileReport
from autoviz.AutoViz_Class import AutoViz_Class
AV = AutoViz_Class()
```

Imported v0.1.55. After importing, execute '%matplotlib inline' to display charts in Jupyter.

```
AV = AutoViz_Class()
```

```
dfte = AV.AutoViz(filename, sep=',', depVar='', dfte=None, header=0, verbose=1, lowess=False,
                  chart_format='svg',max_rows_analyzed=150000,max_cols_analyzed=30, save_plot_dir=None)
```

Update: verbose=0 displays charts in your local Jupyter notebook.

verbose=1 additionally provides EDA data cleaning suggestions. It also displays charts.

verbose=2 does not display charts but saves them in AutoViz_Plots folder in local machine.

chart_format='bokeh' displays charts in your local Jupyter notebook.

chart_format='server' displays charts in your browser: one tab for each chart type

chart_format='html' silently saves interactive HTML files in your local machine




```
In [3]: #Template we will use:
#Load the file:
template_style = "plotly_white"
data = pd.read_excel('data.xlsx', engine="openpyxl")
data.head()
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category
0	13	CA-2017-114412	2020-04-15	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	28027	South	OFF-PA-10002365	Office Supplies
1	24	US-2017-156909	2020-07-16	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUR-CH-10002774	Furniture
2	35	CA-2017-107727	2020-10-19	Second Class	MA-17560	Matt Abelman	Home Office	United States	Houston	Texas	77095	Central	OFF-PA-10000249	Office Supplies
3	42	CA-2017-120999	2020-09-10	Standard Class	LC-16930	Linda Cazamias	Corporate	United States	Naperville	Illinois	60540	Central	TEC-PH-10004093	Technology
4	44	CA-2017-139619	2020-09-19	Standard Class	ES-14080	Erin Smith	Corporate	United States	Melbourne	Florida	32935	South	OFF-ST-10003282	Office Supplies



Task 2: Explore Dataset

In [4]: *# Basic info about the data*
 data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3312 entries, 0 to 3311
Data columns (total 20 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Row ID                 3312 non-null  int64
1   Order ID               3312 non-null  object
2   Order Date             3312 non-null  datetime64[ns]
3   Ship Mode               3312 non-null  object
4   Customer ID            3312 non-null  object
5   Customer Name          3312 non-null  object
6   Segment                3312 non-null  object
7   Country                 3312 non-null  object
8   City                   3312 non-null  object
9   State                  3312 non-null  object
10  Postal Code             3312 non-null  int64
11  Region                  3312 non-null  object
12  Product ID              3312 non-null  object
13  Category                3312 non-null  object
14  Sub-Category            3312 non-null  object
15  Product Name            3312 non-null  object
16  Sales                   3312 non-null  float64
17  Quantity                3312 non-null  int64
18  Discount                3312 non-null  float64
19  Profit                  3312 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(3), object(13)
memory usage: 517.6+ KB
```

```
In [5]: #Describe Method  
data.describe()
```

Out[5]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	3312.000000	3312.000000	3312.000000	3312.000000	3312.000000	3312.000000
mean	5087.107488	56186.515097	221.381418	3.766908	0.156467	28.212340
std	2817.482266	31980.375516	585.257531	2.221776	0.207429	241.864342
min	13.000000	1841.000000	0.444000	1.000000	0.000000	-3839.990400
25%	2655.750000	27978.750000	17.018000	2.000000	0.000000	1.763200
50%	5183.500000	60472.500000	53.810000	3.000000	0.200000	8.296800
75%	7498.250000	90032.000000	205.105700	5.000000	0.200000	28.315125
max	9994.000000	99301.000000	13999.960000	14.000000	0.800000	6719.980800

```
In [6]: # Check unique values for a specific column
data['Customer ID'].unique()
```

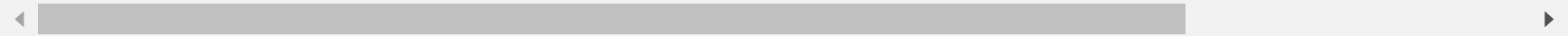
```
Out[6]: array(['AA-10480', 'SF-20065', 'MA-17560', 'LC-16930', 'ES-14080',
               'TB-21520', 'KB-16600', 'CS-12400', 'PO-18865', 'PG-18895',
               'RB-19705', 'PN-18775', 'KD-16345', 'JM-15250', 'CV-12805',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-16690', 'BB-10990', 'AG-10495',
               'JH-15910', 'SH-20395', 'DJ-13510', 'CC-12550', 'TD-20995',
               'VB-21745', 'KW-16435', 'GT-14755', 'AR-10405', 'MV-18190',
               'VP-21730', 'SS-20140', 'RF-19840', 'KH-16510', 'KC-16675',
               'CJ-12010', 'SP-20860', 'MP-17965', 'NF-18385', 'JM-15265',
               'KH-16630', 'BB-11545', 'TB-21595', 'EB-13705', 'DS-13180',
               'MT-18070', 'LS-17245', 'BN-11515', 'AD-10180', 'MC-17605',
               'PH-18790', 'JC-15340', 'EP-13915', 'AS-10135', 'RC-19960',
               'GT-14710', 'DK-13225', 'LP-17080', 'FP-14320', 'EB-13840',
               'JF-15415', 'CS-11950', 'DV-13465', 'CK-12595', 'NG-18355',
               'MO-17800', 'AT-10735', 'BD-11605', 'JL-15835', 'SC-20305',
               'TW-21025', 'CC-12430', 'TH-21235', 'RB-19570', 'CD-11980',
               'GT-14635', 'RA-19285', 'AP-10915', 'RS-19765', 'KC-16540',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-16690', 'BB-10990', 'AG-10495',
               'JH-15910', 'SH-20395', 'DJ-13510', 'CC-12550', 'TD-20995',
               'VB-21745', 'KW-16435', 'GT-14755', 'AR-10405', 'MV-18190',
               'VP-21730', 'SS-20140', 'RF-19840', 'KH-16510', 'KC-16675',
               'CJ-12010', 'SP-20860', 'MP-17965', 'NF-18385', 'JM-15265',
               'KH-16630', 'BB-11545', 'TB-21595', 'EB-13705', 'DS-13180',
               'MT-18070', 'LS-17245', 'BN-11515', 'AD-10180', 'MC-17605',
               'PH-18790', 'JC-15340', 'EP-13915', 'AS-10135', 'RC-19960',
               'GT-14710', 'DK-13225', 'LP-17080', 'FP-14320', 'EB-13840',
               'JF-15415', 'CS-11950', 'DV-13465', 'CK-12595', 'NG-18355',
               'MO-17800', 'AT-10735', 'BD-11605', 'JL-15835', 'SC-20305',
               'TW-21025', 'CC-12430', 'TH-21235', 'RB-19570', 'CD-11980',
               'GT-14635', 'RA-19285', 'AP-10915', 'RS-19765', 'KC-16540',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-16690', 'BB-10990', 'AG-10495',
               'JH-15910', 'SH-20395', 'DJ-13510', 'CC-12550', 'TD-20995',
               'VB-21745', 'KW-16435', 'GT-14755', 'AR-10405', 'MV-18190',
               'VP-21730', 'SS-20140', 'RF-19840', 'KH-16510', 'KC-16675',
               'CJ-12010', 'SP-20860', 'MP-17965', 'NF-18385', 'JM-15265',
               'KH-16630', 'BB-11545', 'TB-21595', 'EB-13705', 'DS-13180',
               'MT-18070', 'LS-17245', 'BN-11515', 'AD-10180', 'MC-17605',
               'PH-18790', 'JC-15340', 'EP-13915', 'AS-10135', 'RC-19960',
               'GT-14710', 'DK-13225', 'LP-17080', 'FP-14320', 'EB-13840',
               'JF-15415', 'CS-11950', 'DV-13465', 'CK-12595', 'NG-18355',
               'MO-17800', 'AT-10735', 'BD-11605', 'JL-15835', 'SC-20305',
               'TW-21025', 'CC-12430', 'TH-21235', 'RB-19570', 'CD-11980',
               'GT-14635', 'RA-19285', 'AP-10915', 'RS-19765', 'KC-16540',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-16690', 'BB-10990', 'AG-10495',
               'JH-15910', 'SH-20395', 'DJ-13510', 'CC-12550', 'TD-20995',
               'VB-21745', 'KW-16435', 'GT-14755', 'AR-10405', 'MV-18190',
               'VP-21730', 'SS-20140', 'RF-19840', 'KH-16510', 'KC-16675',
               'CJ-12010', 'SP-20860', 'MP-17965', 'NF-18385', 'JM-15265',
               'KH-16630', 'BB-11545', 'TB-21595', 'EB-13705', 'DS-13180',
               'MT-18070', 'LS-17245', 'BN-11515', 'AD-10180', 'MC-17605',
               'PH-18790', 'JC-15340', 'EP-13915', 'AS-10135', 'RC-19960',
               'GT-14710', 'DK-13225', 'LP-17080', 'FP-14320', 'EB-13840',
               'JF-15415', 'CS-11950', 'DV-13465', 'CK-12595', 'NG-18355',
               'MO-17800', 'AT-10735', 'BD-11605', 'JL-15835', 'SC-20305',
               'TW-21025', 'CC-12430', 'TH-21235', 'RB-19570', 'CD-11980',
               'GT-14635', 'RA-19285', 'AP-10915', 'RS-19765', 'KC-16540',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-16690', 'BB-10990', 'AG-10495',
               'JH-15910', 'SH-20395', 'DJ-13510', 'CC-12550', 'TD-20995',
               'VB-21745', 'KW-16435', 'GT-14755', 'AR-10405', 'MV-18190',
               'VP-21730', 'SS-20140', 'RF-19840', 'KH-16510', 'KC-16675',
               'CJ-12010', 'SP-20860', 'MP-17965', 'NF-18385', 'JM-15265',
               'KH-16630', 'BB-11545', 'TB-21595', 'EB-13705', 'DS-13180',
               'MT-18070', 'LS-17245', 'BN-11515', 'AD-10180', 'MC-17605',
               'PH-18790', 'JC-15340', 'EP-13915', 'AS-10135', 'RC-19960',
               'GT-14710', 'DK-13225', 'LP-17080', 'FP-14320', 'EB-13840',
               'JF-15415', 'CS-11950', 'DV-13465', 'CK-12595', 'NG-18355',
               'MO-17800', 'AT-10735', 'BD-11605', 'JL-15835', 'SC-20305',
               'TW-21025', 'CC-12430', 'TH-21235', 'RB-19570', 'CD-11980',
               'GT-14635', 'RA-19285', 'AP-10915', 'RS-19765', 'KC-16540',
               'TS-21610', 'DW-13585', 'SH-19975', 'SG-20080', 'SJ-20500',
               'VM-21685', 'FH-14365', 'MB-17305', 'LC-17140', 'HK-14890',
               'LE-16810', 'JH-15985', 'DB-13120', 'CC-12670', 'DL-13600',
               'CB-12535', 'CA-12310', 'KH-
```

```
In [7]: #NaN Values
data.isna()
```

Out[7]:

	Row ID	Order ID	Order Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	F
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
...	
3307	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3308	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3309	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3310	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3311	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	

3312 rows × 20 columns



```
In [9]: # WE WILL USE PANDAS PROFILING TO GET A BETTER UNDERSTANDING.
profile = ProfileReport(data,title = "Sales profiling")

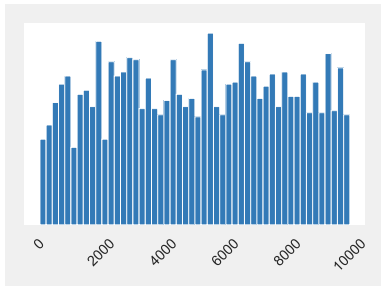
# View In Notebook
profile.to_widgets()
```

Summarize dataset: 100%

69/69 [00:09<00:00, 8.10it/s, Completed]

Generate report structure: 100%

1/1 [00:03<00:00, 3.42s/it]

Overview	Variables	Interactions	Correlations	Missing values	Sample
<div>▼ Row ID</div> <div><div><div>Row ID</div><div>Real number ($\mathbb{R}_{\geq 0}$)</div><div>UNIQUE</div></div><div><div>Distinct</div><div>3312</div><div>Minimum</div><div>13</div></div><div><div>Distinct (%)</div><div>100.0%</div><div>Maximum</div><div>9994</div></div><div><div>Missing</div><div>0</div><div>Zeros</div><div>0</div></div><div><div>Missing (%)</div><div>0.0%</div><div>Zeros (%)</div><div>0.0%</div></div><div><div>Infinite</div><div>0</div><div>Negative</div><div>0</div></div><div><div>Infinite (%)</div><div>0.0%</div><div>Negative (%)</div><div>0.0%</div></div><div><div>Mean</div><div>5087.107488</div><div>Memory size</div><div>26.0 KiB</div></div></div> <div></div> <div>Toggle details</div>					
<div>► Order ID</div>					
<div>► Order Date</div>					
<div>► Ship Mode</div>					

▶ **Customer ID**

▶ **Customer Name**

▶ **Segment**

▶ **Country**

▶ **City**

▶ **State**

▶ **Postal Code**

▶ **Region**

▶ **Product ID**

▶ **Category**

▶ **Sub-Category**

▶ **Product Name**

▶ **Sales**

▶ **Quantity**

▶ **Discount**

▶ **Profit**

Report generated by YData (https://ydata.ai/?utm_source=opensource&utm_medium=pandasprofiling&utm_campaign=report).

```
In [10]: # Export Pandas Profiling Report to HTML  
profile.to_file('output.html')
```

Render HTML: 100%

1/1 [00:01<00:00, 1.81s/it]

Export report to file: 100%

1/1 [00:00<00:00, 31.21it/s]


```
In [11]: #Now we gonna use Autovis, autoviz performs autometric visualization
AV = AutoViz_Class()
df_autoviz = AV.AutoViz('data.xlsx')
```

Shape of your Data Set loaded: (3312, 20)

```
#####
##### C L A S S I F Y I N G   V A R I A B L E S #####
#####
Classifying variables in data set...
```

	Nuniques	dtype	Nulls	Nullpercent	NuniquePercent	Value counts Min	Data cleaning improvement suggestions
Row ID	3312	int64	0	0.000000	100.000000	0	possible ID column: drop
Profit	2913	float64	0	0.000000	87.952899	0	highly skewed column: remove outliers or do box-cox transform
Sales	2623	float64	0	0.000000	79.196860	0	highly skewed column: remove outliers or do box-cox transform
Order ID	1687	object	0	0.000000	50.935990	1	combine rare categories
Product ID	1525	object	0	0.000000	46.044686	1	combine rare categories
Product Name	1511	object	0	0.000000	45.621981	1	combine rare categories
Customer ID	693	object	0	0.000000	20.923913	1	combine rare categories
Customer Name	693	object	0	0.000000	20.923913	1	combine rare categories
Postal Code	437	int64	0	0.000000	13.194444	0	
City	350	object	0	0.000000	10.567633	1	combine rare categories
Order Date	322	datetime64[ns]	0	0.000000	9.722222	0	
State	47	object	0	0.000000	1.419082	2	combine rare categories
Sub-Category	17	object	0	0.000000	0.513285	22	
Quantity	14	int64	0	0.000000	0.422705	0	
Discount	12	float64	0	0.000000	0.362319	0	skewed column: cap or drop possible outliers
Ship Mode	4	object	0	0.000000	0.120773	186	

	Nuniques	dtype	Nulls	Nullpercent	NuniquePercent	Value counts Min	Data cleaning improvement suggestions
Region	4	object	0	0.000000	0.120773	518	
Category	3	object	0	0.000000	0.090580	624	
Segment	3	object	0	0.000000	0.090580	664	
Country	1	object	0	0.000000	0.030193	3312	invariant values: drop column

20 Predictors classified...

2 variables removed since they were ID or low-information variables

List of variables removed: ['Row ID', 'Country']

Number of All Scatter Plots = 6

[nltk_data] Error loading popular: <urlopen error [Errno 11001]

[nltk_data] getaddrinfo failed>

All Plots done

Time to run AutoViz = 37 seconds

AUTO VISUALIZATION Completed

Task 4: Data Preperation & Analysis

TASKS:

- What was the highest Sale in 2020?
- What is average discount rate of charis?
- Add extra columns to seperate Year & Month from the Order Date
- Add a new column to calculate the Profit Margin for each sales record
- Export manipulated dataframe to Excel
- Create a new dataframe to reflect total Profit & Sales by Sub-Category
- Develop a function, to return a dataframe which is grouped by a particular column (as an input)

What was the highest Sale in 2020?

In [44]: `data.nlargest(1, 'Sales')`

Out[44]:

	Row ID	Order ID	Order Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category
2710	8154	CA-2017-140151	2020-03-23	First Class	RB-19360	Raymond Buch	Consumer	United States	Seattle	Washington	98115	West	TEC-CO-10004722	Technology



What is average discount rate of charis?

In [48]: `# Create a Boolean mask
mask = data['Sub-Category'] == 'Chairs'

#Use a boolean mask to filter dataframe
data[mask]['Discount'].mean()`

Out[48]: 0.1673684210526316

Add an extra column for "Order Month" & "Order Year"

```
In [49]: data['Order Date Year'] = data['Order Date'].dt.year
data['Order Date Month'] = data['Order Date'].dt.month
data.head(5)
```

Out[49]:

Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit	Order Date Year
United States	Concord	North Carolina	28027	South	OFF-PA-10002365	Office Supplies	Paper	Xerox 1967	15.552	3	0.2	5.4432	2020
United States	Philadelphia	Pennsylvania	19140	East	FUR-CH-10002774	Furniture	Chairs	Global Deluxe Stacking Chair, Gray	71.372	2	0.3	-1.0196	2020
United States	Houston	Texas	77095	Central	OFF-PA-10000249	Office Supplies	Paper	Easy-staple paper	29.472	3	0.2	9.9468	2020
United States	Naperville	Illinois	60540	Central	TEC-PH-10004093	Technology	Phones	Panasonic Kx-TS550	147.168	4	0.2	16.5564	2020
United States	Melbourne	Florida	32935	South	OFF-ST-10003282	Office Supplies	Storage	Advantus 10-Drawer Portable Organizer, Chrome Metal Frame, Smoke Drawers	95.616	2	0.2	9.5616	2020



Add a new column to calculate the Profit Margin for each sales record

```
In [50]: data['Profit Margin'] = data['Profit'] / data['Sales']
data.head(4)
```

Out[50]:

	Row ID	Order ID	Order Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category
0	13	CA-2017-114412	2020-04-15	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	28027	South	OFF-PA-10002365	Office Supplies
1	24	US-2017-156909	2020-07-16	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUR-CH-10002774	Furniture
2	35	CA-2017-107727	2020-10-19	Second Class	MA-17560	Matt Abelman	Home Office	United States	Houston	Texas	77095	Central	OFF-PA-10000249	Office Supplies
3	42	CA-2017-120999	2020-09-10	Standard Class	LC-16930	Linda Cazamias	Corporate	United States	Naperville	Illinois	60540	Central	TEC-PH-10004093	Technology

Export manipulated dataframe to Excel

```
In [52]: data.to_excel("C:/Users/nelio/Documents/python/Jupyter/data-analysis-python-master/output/data_output.xlsx", index=False)
```

Create a new dataframe to reflect total Profit & Sales by Sub-Category

```
In [53]: # Group by Sub-Category [SUM]
data_group = data.groupby('Sub-Category').sum()
data_group.head()

# Reset Index(If you wany)
data_group = data.groupby.reset_index(inplace=True)
data_group.head()
```

Out[53]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit	Order Date Year	Order Date Month	Profit Margin
Sub-Category									
Accessories	1449605	15762114	59946.2320	1079	19.80	15672.3570	555500	2153	62.655000
Appliances	874517	9377881	42926.9320	654	29.10	7865.2683	333300	1255	-36.133889
Art	1377681	16355998	8863.0680	1101	22.20	2221.9631	569640	2146	70.995000
Binders	2500416	29015384	72788.0450	2067	189.10	7669.7418	1010000	3841	-107.832500
Bookcases	384522	4483249	30024.2797	276	16.32	-583.6261	153520	552	-11.233641

Develop a function, to return a dataframe which is grouped by a particular column (as an input)

```
In [14]: # Groupby as a function
def grouped_data(column_name):
    data_tmp = data.groupby(column_name).sum()
    data_tmp.reset_index(inplace=True)
    return data_tmp

#Group DataFrame by Segment
grouped_data('Segment')
```

Out[14]:

	Segment	Row ID	Postal Code	Sales	Quantity	Discount	Profit
0	Consumer	8561367	92871614	331904.6999	6282	264.12	45568.2391
1	Corporate	5023960	54879479	241847.8244	3758	154.68	26782.3633
2	Home Office	3263173	38338645	159462.7309	2436	99.42	21088.6672

Task 5: Further Deep Dive & Visualalization

We will focus on presentable charts using plotly

Distribution Sales [Histogram]

```
In [15]: #Stats Overview for Sales
data['Sales'].describe()
```

```
Out[15]: count      3312.000000
mean         221.381418
std          585.257531
min           0.444000
25%          17.018000
50%          53.810000
75%         205.105700
max        13999.960000
Name: Sales, dtype: float64
```

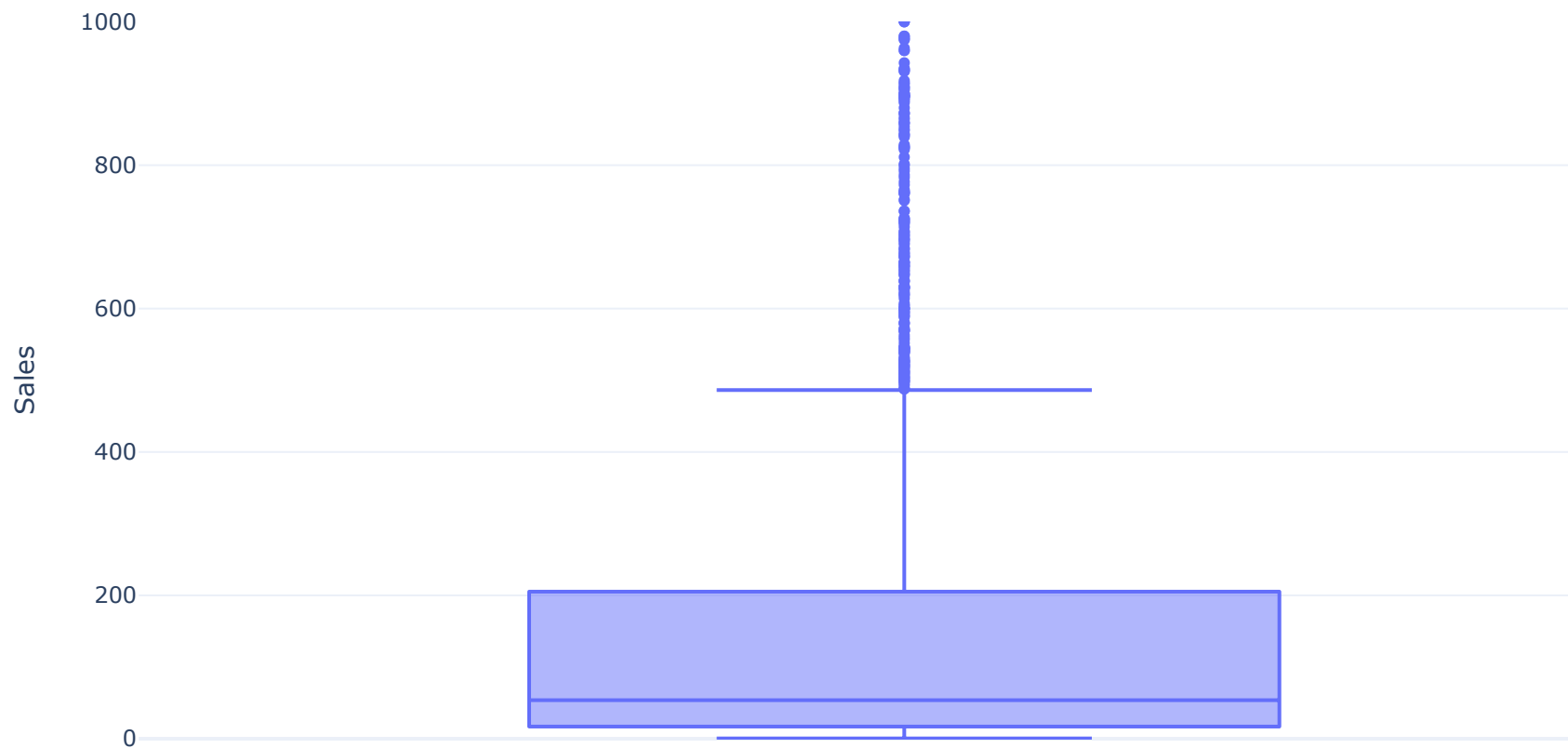
```
In [16]: # Create Chart
fig = px.histogram(data,
                    x="Sales",
                    template=template_style)

# Plot Chart
fig.show()
```



Show the distribution and skewness of Sales [Boxplot]


```
In [17]: # Create Chart
fig = px.box(data,
             y="Sales",
             range_y=[0,1000],
             template=template_style)
# Plot Chart
fig.show()
```



Plot Sales by Sub-Category

```
In [18]: # Create Dataframe  
data = grouped_data('Sub-Category')  
data.head()
```

Out[18]:

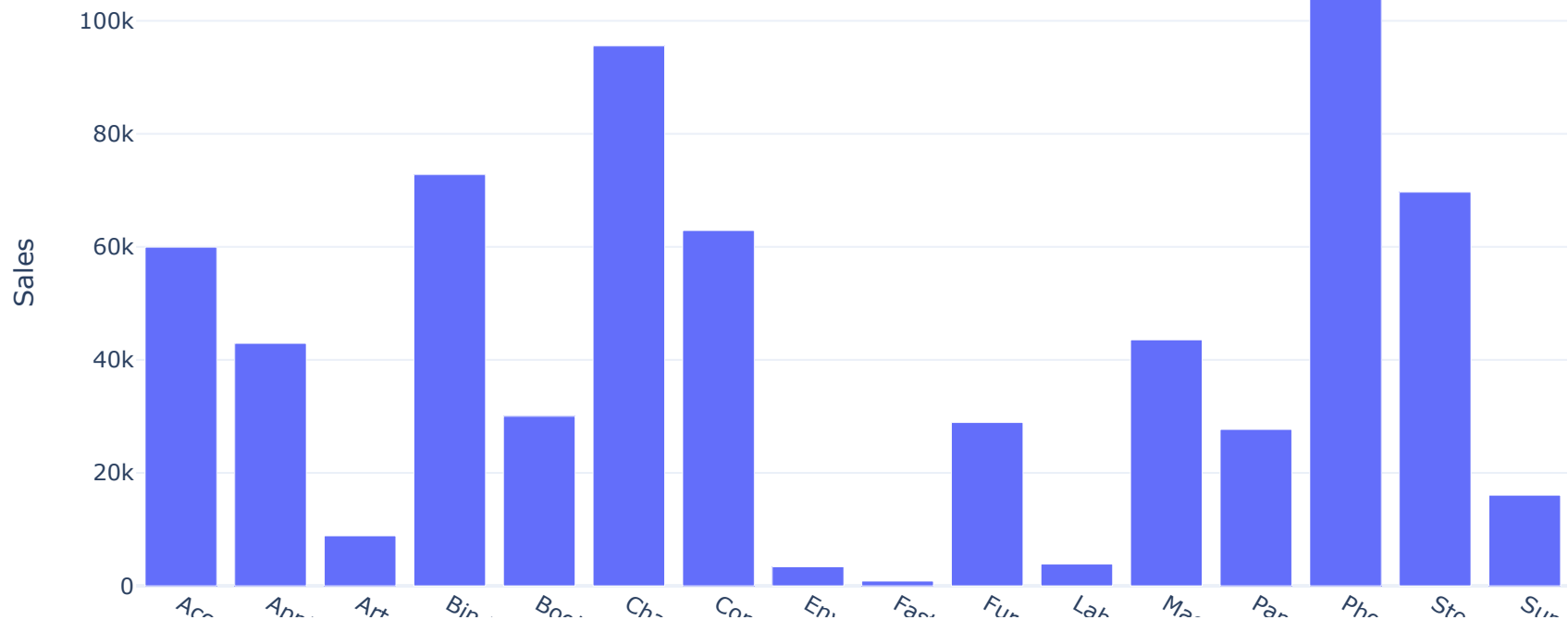
	Sub-Category	Row ID	Postal Code	Sales	Quantity	Discount	Profit
0	Accessories	1449605	15762114	59946.2320	1079	19.80	15672.3570
1	Appliances	874517	9377881	42926.9320	654	29.10	7865.2683
2	Art	1377681	16355998	8863.0680	1101	22.20	2221.9631
3	Binders	2500416	29015384	72788.0450	2067	189.10	7669.7418
4	Bookcases	384522	4483249	30024.2797	276	16.32	-583.6261

```
In [20]: # Create Chart
fig = px.bar(data,
             x='Sub-Category',
             y='Sales',
             title='<b>Sales by Sub Category</b>',
             template = template_style)

# Display Plot
fig.show()

# Export Chart to HTML
#plotly.offline.plot(fig, filename='output/Sales_Sub_Cat.html', auto_open=False)
```

Sales by Sub Category



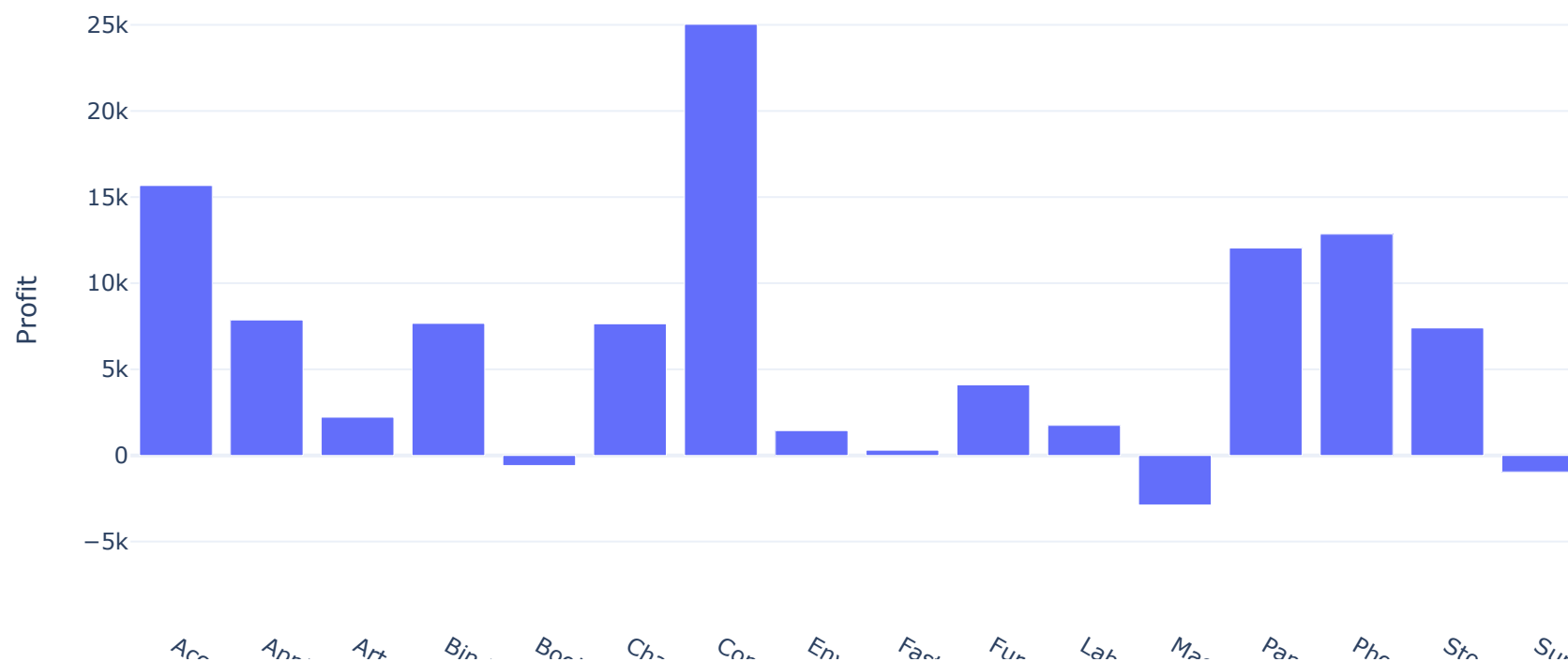
Plot Profit by Sub-Category

```
In [22]: # Create Chart
fig = px.bar(data,
             x='Sub-Category',
             y='Profit',
             title='<b>Profit by Sub Category</b>',
             template = template_style)

# Display Plot
fig.show()

# Export Chart to HTML
#plotly.offline.plot(fig, filename='output/Profit_Sub_Cat.html', auto_open=False)
```

Profit by Sub Category



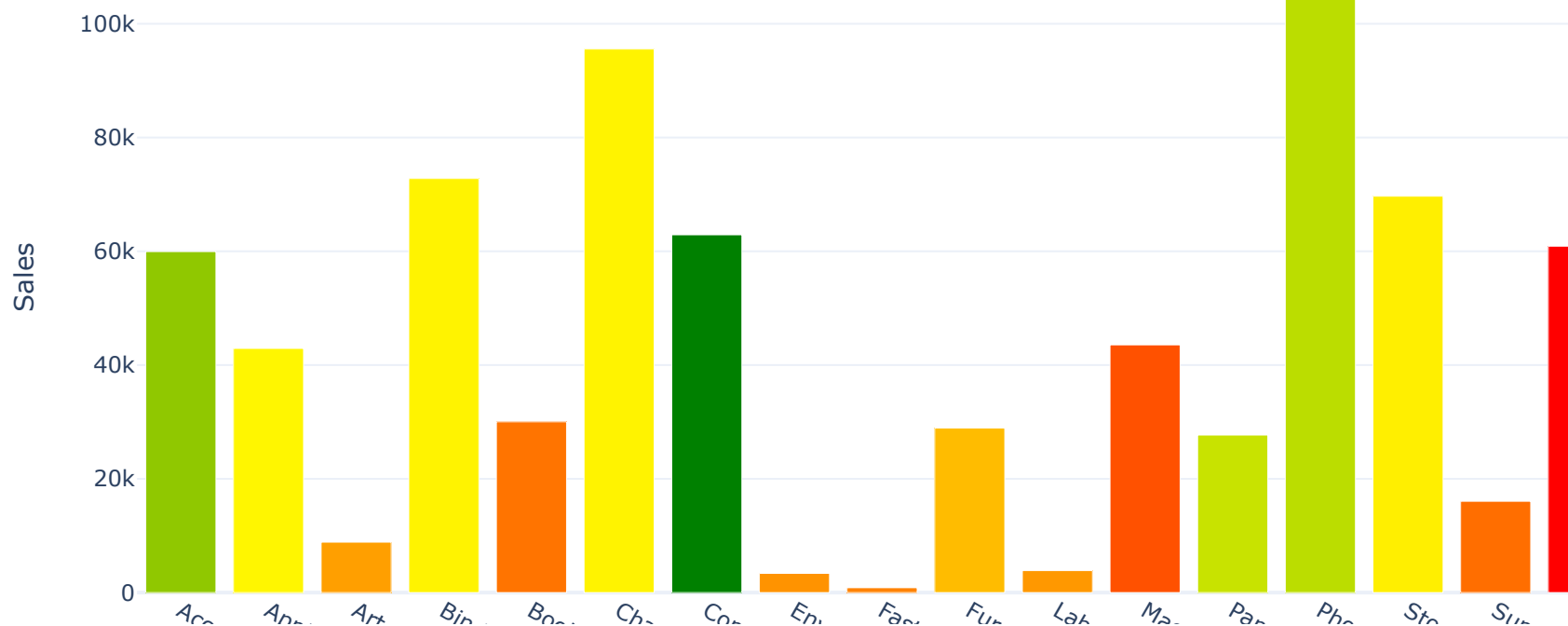
Plot Sales & Profit by Sub-Category

```
In [23]: # Create Chart
fig = px.bar(data,
             x='Sub-Category',
             y='Sales', color='Profit',
             color_continuous_scale=["red", "yellow", "green"],
             template = template_style,
             title = '<b>Sales & Profit by Sub Category</b>')

# Display Plot
fig.show()

# Export Chart to HTML
#plotly.offline.plot(fig, filename = 'output/Profit_Sales_Sub_Cat.html', auto_open=False)
```

Sales & Profit by Sub Category



Inspect Negative Profit of Tables

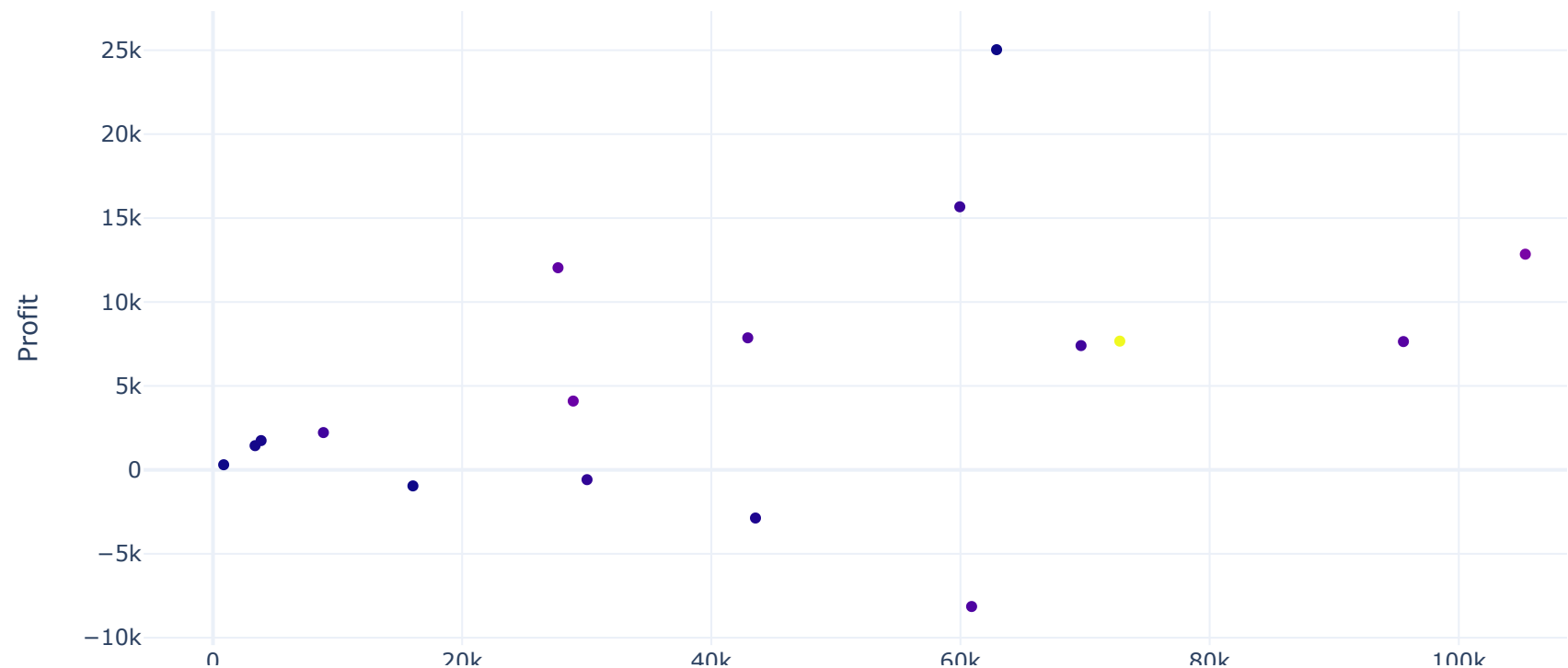
Is there any linear correlation between Sales/Profit & Discount? [Scatterplot]


```
In [25]: # Create Chart
fig = px.scatter(data,
                 x='Sales',
                 y='Profit',
                 color='Discount',
                 template = template_style,
                 title = '<b>Scatterplot Sales/Profit</b>')

# Display Plot
fig.show()

# Export Chart to HTML
#plotly.offline.plot(fig, filename='output/Sales_Profit_Scatterplot.html', auto_open=False)
```

Scatterplot Sales/Profit



Check Discount mean by Sub Category

```
In [27]: # Create new dataframe: Group by 'Sub-Category' and aggregate the mean of 'Discount'
df_discount = data.groupby('Sub-Category').agg({'Discount':'mean',
                                                'Profit':'sum'})

# Display first 5 rows of new dataframe
df_discount.head()
```

Out[27]:

	Discount	Profit
Sub-Category		
Accessories	19.80	15672.3570
Appliances	29.10	7865.2683
Art	22.20	2221.9631
Binders	189.10	7669.7418
Bookcases	16.32	-583.6261

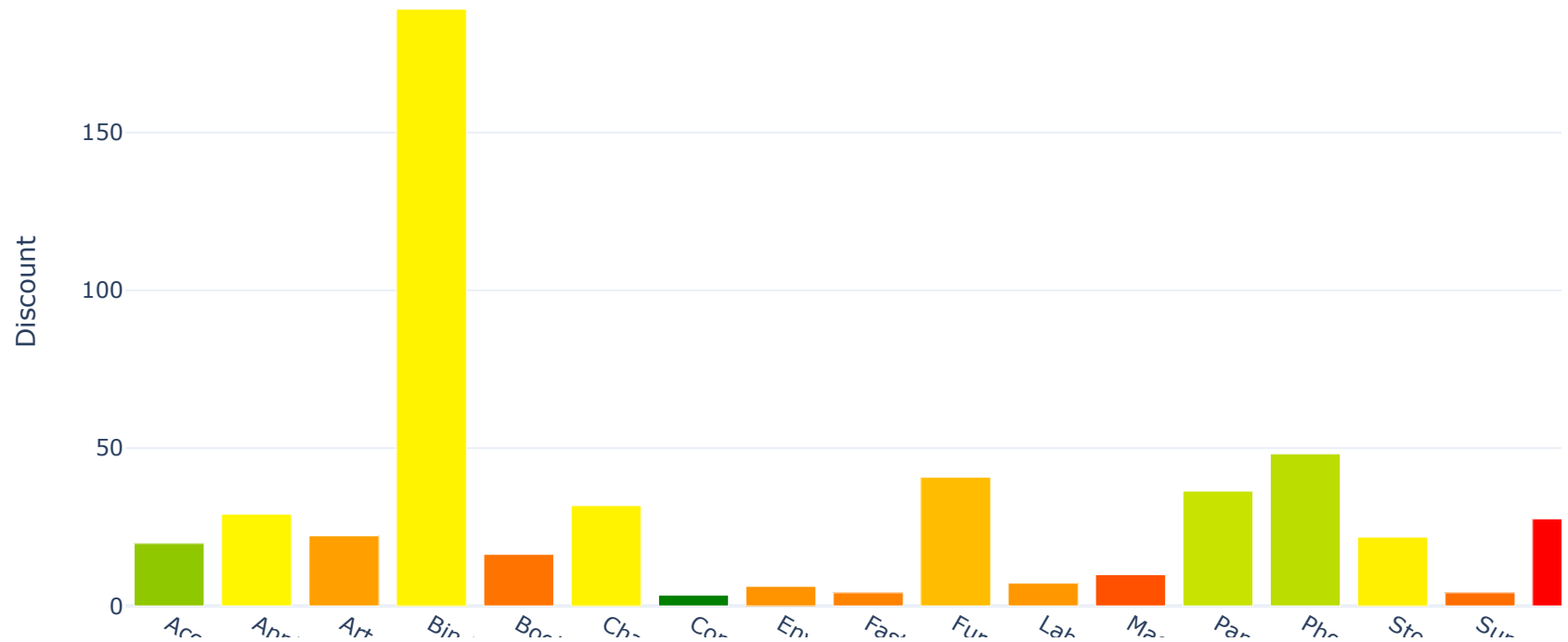
Plot Mean Discount by Sub Category

```
In [28]: # Create Chart
fig = px.bar(df_discount,
             x=df_discount.index,
             y='Discount',
             color='Profit',
             color_continuous_scale=['red', 'yellow', 'green'],
             template = template_style,
             title = '<b>Mean Discount by Sub Category</b>')

# Display Plot
fig.show()

# Export Chart to HTML
```

Mean Discount by Sub Category



Thank you, I end my projects here. 🏠

Rerefence(bestfriend):

StackOverflow

Google

In [38]: `#pip install -U notebook-as-pdf`

In []: