In [255…
```
from IPython import display
display.Image("/content/drive/MyDrive/Data_Engineering/Best-Red-Bull-Flavors-Featured-Image-1024x683.jpeg", width = 170
```

Out[255]:

Account Data Analysis

Analyze data about accounts to identify key trends and opportunities for sales growth and communicate your insights.

Overcoming Sales Objections

Respond to objections raised during the sales process to help close the sale.

In [256…
```
#Total sales by Account Type and Year.
#Sales growth/trends by Account Type.
#Sales growth/trends by Year.
#Best and worst performing accounts (overall, and by account type).
#Effect of assortment (product lines) presence on sales.
#Effectiveness of the different marketing/promotion programs.
```

1. Data processing

In [257…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import matplotlib.colors as mcolors
from IPython import display
!pip install pypyodbc
!pip install pyodbc
```

```
Requirement already satisfied: pypyodbc in /usr/local/lib/python3.10/dist-packages (1.3.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from pypyodbc) (67.7.2)
Requirement already satisfied: pyodbc in /usr/local/lib/python3.10/dist-packages (5.1.0)
```

In [258…
```
!apt-get install -y unixodbc-dev
!apt-get install -y libssl1.0.0 libssl-dev
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
unixodbc-dev is already the newest version (2.3.9-5ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package libssl1.0.0
E: Couldn't find any package by glob 'libssl1.0.0'
E: Couldn't find any package by regex 'libssl1.0.0'
```

In [259…

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
nt=True).
```

In [260…

```
# Script for Loading and Analyzing Account Sales Data from Excel File
df = pd.read_excel('/content/drive/MyDrive/Data_Engineering/Account Sales Data for Analysis v2.xlsx', header=3)
```

In [261…

```
df.head()
```

Out[261]:

	Account Name	Account Address	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	20
0	Bar 1	2131 Patterson Road, Brooklyn NY 11201	Dorothy Rizzo	(880) 283-6803	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1982	5388	7063	72
1	Bar 2	3685 Morningview Lane, New York NY 10013	Lawson Moore	(711) 426-7350	Bar	Yes	Yes	Yes	No	Yes	Yes	Yes	2786	3804	4121	62
2	Bar 3	2285 Ladybug Drive, New York NY 10013	Vin Hudson	(952) 952-5573	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1209	1534	1634	43
3	Bar 4	2930 Southern Street, New York NY 10005	Susana Huels	(491) 505-6064	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	906	1251	2897	44
4	Bar 5	2807 Geraldine Lane, New York NY 10004	Shanna Hettinger	(412) 570-0596	Bar	Yes	Yes	No	Yes	Yes	Yes	Yes	1421	1893	2722	44

In [262…
```
df_To_Calculated_CAGR = df.copy()
```

In [263…
```
# Filtering Account Sales Data for Accounts Containing 'Bar
dfBars = df[df['Account Name'].str.contains("Bar")]
```

In [264…
```
dfBars.head()
```

Out[264]:

	Account Name	Account Address	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	20
0	Bar 1	2131 Patterson Road, Brooklyn NY 11201	Dorothy Rizzo	(880) 283-6803	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1982	5388	7063	72
1	Bar 2	3685 Morningview Lane, New York NY 10013	Lawson Moore	(711) 426-7350	Bar	Yes	Yes	Yes	No	Yes	Yes	Yes	2786	3804	4121	62
2	Bar 3	2285 Ladybug Drive, New York NY 10013	Vin Hudson	(952) 952-5573	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1209	1534	1634	43
3	Bar 4	2930 Southern Street, New York NY 10005	Susana Huels	(491) 505-6064	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	906	1251	2897	44
4	Bar 5	2807 Geraldine Lane, New York NY 10004	Shanna Hettinger	(412) 570-0596	Bar	Yes	Yes	No	Yes	Yes	Yes	Yes	1421	1893	2722	44

In [264…

In [265…
```
# Selecting Relevant Columns for 'Bar' Accounts Sales Data from 2017 to 2021
dfBarsTable = dfBars[['Account Name', 2017,2018,2019,2020,2021]]
```

In [266…
```
#Setting 'Account Name' as Index for Filtered 'Bar' Accounts Sales Data
dfBarsTable_setindex = dfBarsTable.set_index('Account Name')
```

In [267…
```
A = dfBarsTable_setindex
```

In [268…
```
#Calculating Total Sales Quantity per Bar Account and Adding to DataFrame
A['Total_Sales Quantity/Bar/Account Name'] = dfBarsTable_setindex.sum(axis=1)
```

In [269…
```
A.head()
```

Out[269]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Bar/Account Name
Bar 1	1982	5388	7063	7208	9093	30734
Bar 2	2786	3804	4121	6210	6909	23830
Bar 3	1209	1534	1634	4302	9768	18447
Bar 4	906	1251	2897	4499	9428	18981
Bar 5	1421	1893	2722	4410	5873	16319

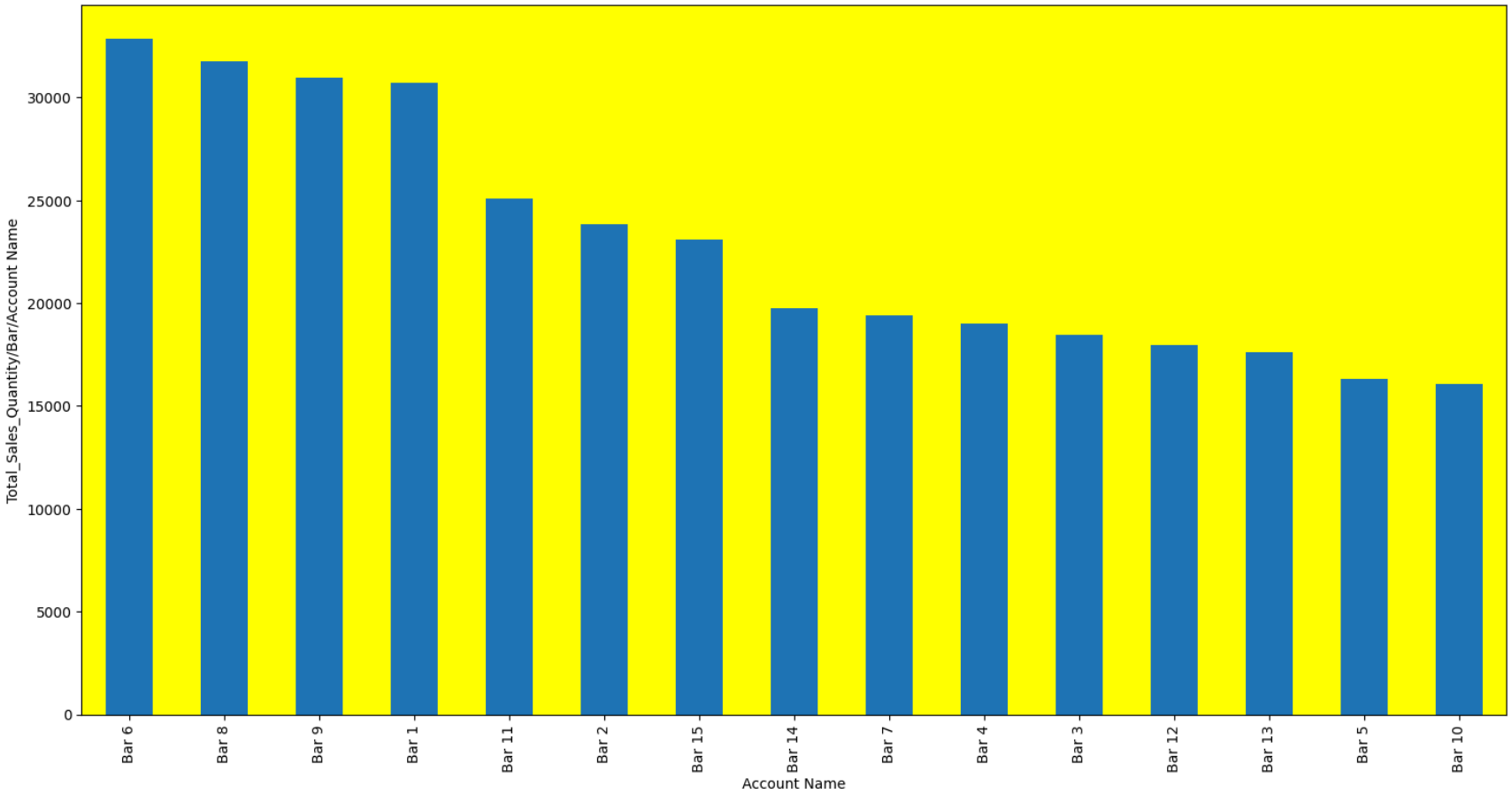
In [270…
```
# Sorting Bar Accounts by Total Sales Quantity in Descending Order
A.sort_values(by='Total_Sales Quantity/Bar/Account Name', ascending=False)
```

Out[270]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Bar/Account Name
Bar 6	2341	6105	7777	7891	8758	32872
Bar 8	1581	4799	6582	9024	9759	31745
Bar 9	9766	8049	5556	5202	2373	30946
Bar 1	1982	5388	7063	7208	9093	30734
Bar 11	7555	6551	5188	3436	2359	25089
Bar 2	2786	3804	4121	6210	6909	23830
Bar 15	9058	4839	4776	4024	369	23066
Bar 14	861	1314	1810	6510	9271	19766
Bar 7	9252	8499	991	448	211	19401
Bar 4	906	1251	2897	4499	9428	18981
Bar 3	1209	1534	1634	4302	9768	18447
Bar 12	1532	2678	4068	4278	5382	17938
Bar 13	24	1797	3548	3668	8592	17629
Bar 5	1421	1893	2722	4410	5873	16319
Bar 10	1530	1620	2027	4881	6002	16060

In [271…

```
# Plotting Total Sales Quantity per Bar Account in Descending Order with Customized Bar Chart
ax = A.sort_values(by='Total_Sales Quantity/Bar/Account Name', ascending=False)['Total_Sales Quantity/Bar/Account Name'
ax.set_facecolor("yellow")
plt.ylabel('Total_Sales_Quantity/Bar/Account Name')
plt.show()
```



In [272…

```
# Sorting sales data for the year 2021 in descending order
A[2021].sort_values(ascending=False)
```

Out[272]:
```
Account Name
Bar 3     9768
Bar 8     9759
Bar 4     9428
Bar 14    9271
Bar 1     9093
Bar 6     8758
Bar 13    8592
Bar 2     6909
Bar 10    6002
Bar 5     5873
Bar 12    5382
Bar 9     2373
Bar 11    2359
Bar 15     369
Bar 7      211
Name: 2021, dtype: int64
```

In [273…
```
# Sorting sales data for the year 2021 in descending order and resetting index
DataFrameBar = pd.DataFrame(A[2021].sort_values(ascending=False)).reset_index()
DataFrameBar.head()
```

Out[273]:

	Account Name	2021
0	Bar 3	9768
1	Bar 8	9759
2	Bar 4	9428
3	Bar 14	9271
4	Bar 1	9093

In [274…
```
# Creating a list of numbers from 0 to the input value
input = 15
output = list(range(input + 1))
print(output)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

In [275…
```
# Removing the first element (0) from the list 'output'
pop_0 = output.pop(0)
```

In [276…
```
# Adding ranks from the list 'output' to the DataFrame 'DataFrameBar'
DataFrameBar['Rank'] = output
```

In [277…
```
DataFrameBar.head()
```

Out[277]:

	Account Name	2021	Rank
0	Bar 3	9768	1
1	Bar 8	9759	2
2	Bar 4	9428	3
3	Bar 14	9271	4
4	Bar 1	9093	5

In [278…
```
# Creating a DataFrame with columns 'Rank', 'Account Name', and sales data for 2021
DataFrameBarRank = DataFrameBar[['Rank','Account Name',2021]]
```

In [279…
```
# Renaming columns in DataFrameBarRank
DataFrameBarRank.rename(columns={'Account Name': 'Bar_Ac_Name',
                                 2021: 'Bar_Sale_2021'},
          inplace=True, errors='raise')
```

In [280…
```
DataFrameBarRank.head()
```

Out[280]:

	Rank	Bar_Ac_Name	Bar_Sale_2021
0	1	Bar 3	9768
1	2	Bar 8	9759
2	3	Bar 4	9428
3	4	Bar 14	9271
4	5	Bar 1	9093

In [281…
```
# Transposing the DataFrame dfBarsTable_setindex
dfBarsTable_setindex_Transpose = dfBarsTable_setindex.T
```

In [282…
```
dfBarsTable_setindex_Transpose
```

Out[282]:

Account Name	Bar 1	Bar 2	Bar 3	Bar 4	Bar 5	Bar 6	Bar 7	Bar 8	Bar 9	Bar 10	Bar 11	Bar 12	Bar 13	Bar 14	Bar 15
2017	1982	2786	1209	906	1421	2341	9252	1581	9766	1530	7555	1532	24	861	9058
2018	5388	3804	1534	1251	1893	6105	8499	4799	8049	1620	6551	2678	1797	1314	4839
2019	7063	4121	1634	2897	2722	7777	991	6582	5556	2027	5188	4068	3548	1810	4776
2020	7208	6210	4302	4499	4410	7891	448	9024	5202	4881	3436	4278	3668	6510	4024
2021	9093	6909	9768	9428	5873	8758	211	9759	2373	6002	2359	5382	8592	9271	369
Total_Sales Quantity/Bar/Account Name	30734	23830	18447	18981	16319	32872	19401	31745	30946	16060	25089	17938	17629	19766	23066

In [283…
```
# Selecting all rows except the last one from the transposed DataFrame dfBarsTable_setindex
dfBarsTable_setindex_Transpose.iloc[:-1 , :]
```

Out[283]:

Account Name	Bar 1	Bar 2	Bar 3	Bar 4	Bar 5	Bar 6	Bar 7	Bar 8	Bar 9	Bar 10	Bar 11	Bar 12	Bar 13	Bar 14	Bar 15
2017	1982	2786	1209	906	1421	2341	9252	1581	9766	1530	7555	1532	24	861	9058
2018	5388	3804	1534	1251	1893	6105	8499	4799	8049	1620	6551	2678	1797	1314	4839
2019	7063	4121	1634	2897	2722	7777	991	6582	5556	2027	5188	4068	3548	1810	4776
2020	7208	6210	4302	4499	4410	7891	448	9024	5202	4881	3436	4278	3668	6510	4024
2021	9093	6909	9768	9428	5873	8758	211	9759	2373	6002	2359	5382	8592	9271	369

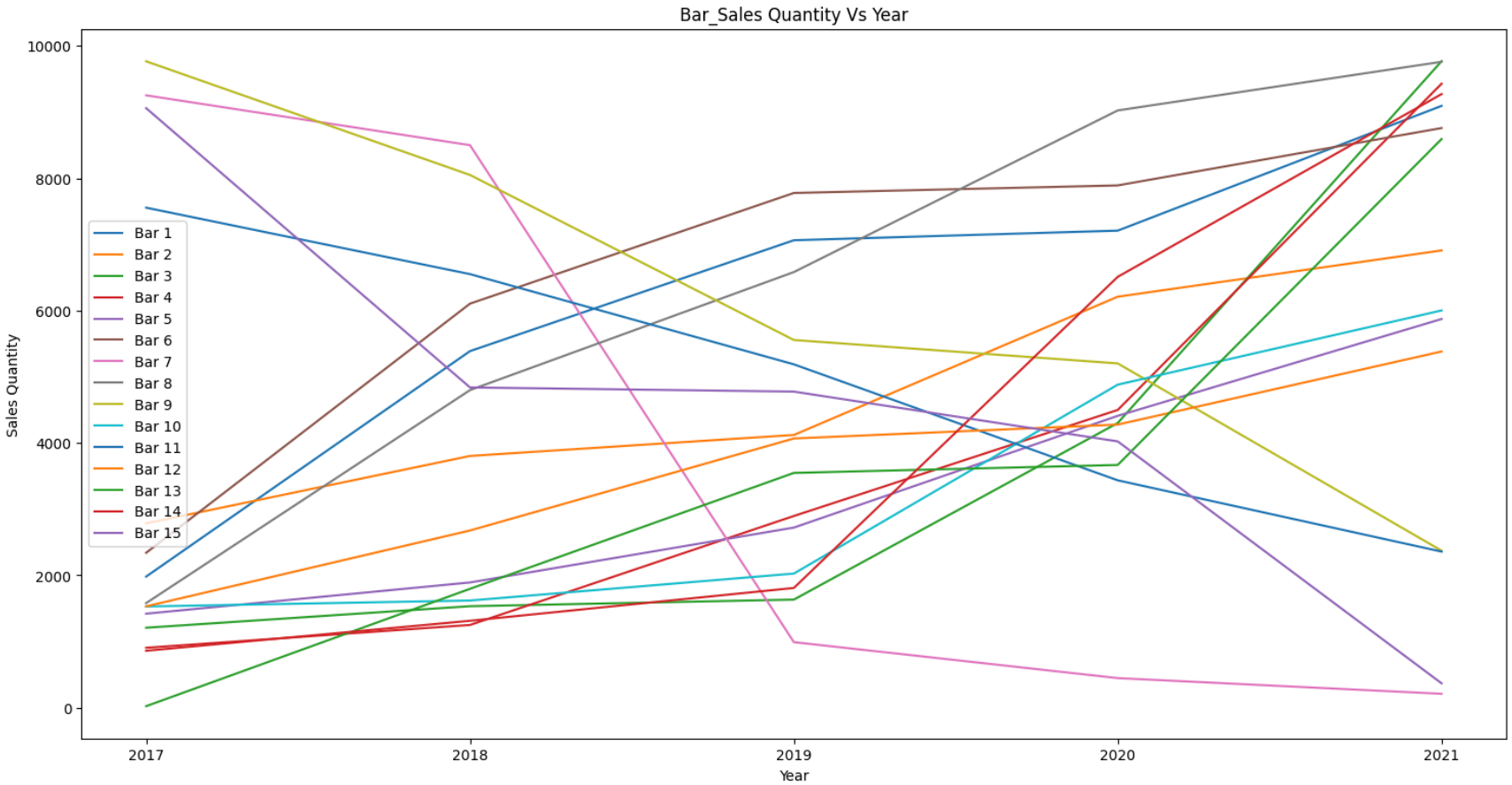
In [284…
```
# Assigning selected data to variable X
X = dfBarsTable_setindex_Transpose.iloc[:-1 , :]
```

In [285…
```
# Adding a 'Years' column to DataFrame X
X['Years'] = ['2017','2018','2019','2020','2021']
```

```
<ipython-input-285-eb12921174f3>: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
  X['Years'] = ['2017','2018','2019','2020','2021']
```

In [286…

```
# Creating a figure and plotting the data
plt.figure(figsize=(18,9))
plt.plot(X.set_index('Years')[0:15])  # Plotting data for first 15 bars
plt.title('Bar_Sales Quantity Vs Year')
plt.xlabel('Year')
plt.ylabel('Sales Quantity')
#plt.legend(['Bar 1', 'Bar 2', 'Bar 3', 'Bar 4', 'Bar 5', 'Bar 6',
#'Bar 7', 'Bar 8', 'Bar 9', 'Bar 10','Bar 11', 'Bar 12', 'Bar 13', 'Bar 14', 'Bar 15']) # Adding legend for the first 15
plt.legend(X.columns[:15]) # Adding legend for the first 15 bars
plt.show()
```



In [287…
```
# Filtering DataFrame for accounts containing "Restaurant"
dfRestaurants = df[df['Account Name'].str.contains("Restaurant")]
```

In [288…
```
dfRestaurants.head()
```

Out[288]:

	Account Name	Account Address	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019
15	Restaurant 1	9848 Linden St, New York NY 10011	Dan Hill	(248) 450-0797	Restaurant	Yes	Yes	No	No	No	No	No	3501	7079	7438
16	Restaurant 2	805 South Pilgrim Court, Brooklyn NY 11225	Javier George	(964) 214-3742	Restaurant	Yes	Yes	No	No	No	No	No	3916	4218	5072
17	Restaurant 3	9132 Redwood Rd, Bronx NY 10466	Christopher Evans	(831) 406-6300	Restaurant	Yes	Yes	No	Yes	No	Yes	No	700	5721	6247
18	Restaurant 4	3 Warren Drive, New York NY 10040	Julie Ross	(778) 387-0744	Restaurant	Yes	Yes	No	No	No	No	No	9773	9179	8390
19	Restaurant 5	402 Bridgeton Lane, Bronx NY 10468	Bill Callahan	(617) 419-7996	Restaurant	Yes	Yes	No	Yes	No	Yes	No	73	3485	4592

In [289…
```
# Selecting relevant columns for 'Restaurant' accounts sales data from 2017 to 2021
dfRestaurantsTable = dfRestaurants[['Account Name', 2017,2018,2019,2020,2021]]
```

In [290…
```
dfRestaurantsTable.head()
```

Out[290]:

	Account Name	2017	2018	2019	2020	2021
15	Restaurant 1	3501	7079	7438	7443	9225
16	Restaurant 2	3916	4218	5072	5201	7588
17	Restaurant 3	700	5721	6247	8495	9236
18	Restaurant 4	9773	9179	8390	8256	3815
19	Restaurant 5	73	3485	4592	5143	8100

In [291…
```
# Setting 'Account Name' as index for filtered 'Restaurant' accounts sales data
dfRestaurantsTable_setindex = dfRestaurantsTable.set_index('Account Name')
```

In [292…
```
dfRestaurantsTable_setindex.head()
```

Out[292]:

Account Name	2017	2018	2019	2020	2021
Restaurant 1	3501	7079	7438	7443	9225
Restaurant 2	3916	4218	5072	5201	7588
Restaurant 3	700	5721	6247	8495	9236
Restaurant 4	9773	9179	8390	8256	3815
Restaurant 5	73	3485	4592	5143	8100

In [293…
```
# Assigning filtered and indexed data to variable B
B = dfRestaurantsTable_setindex
```

In [294…
```
# Calculating total sales quantity per restaurant account and adding to DataFrame B
B['Total_Sales Quantity/Restaurant/Account Name'] = B.sum(axis=1)
```

In [295…
```
B.head()
```

Out[295]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Restaurant/Account Name
Restaurant 1	3501	7079	7438	7443	9225	34686
Restaurant 2	3916	4218	5072	5201	7588	25995
Restaurant 3	700	5721	6247	8495	9236	30399
Restaurant 4	9773	9179	8390	8256	3815	39413
Restaurant 5	73	3485	4592	5143	8100	21393

In [296…

```
# Sorting restaurant accounts by total sales quantity in descending order
B.sort_values(by='Total_Sales Quantity/Restaurant/Account Name', ascending=False)
```

Out[296]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Restaurant/Account Name
Restaurant 4	9773	9179	8390	8256	3815	39413
Restaurant 1	3501	7079	7438	7443	9225	34686
Restaurant 3	700	5721	6247	8495	9236	30399
Restaurant 10	570	1322	7279	8443	9571	27185
Restaurant 14	712	4182	6087	7494	8599	27074
Restaurant 13	6309	6227	5123	4968	3857	26484
Restaurant 2	3916	4218	5072	5201	7588	25995
Restaurant 7	1368	3447	4535	5476	9983	24809
Restaurant 8	8331	7667	5952	1998	375	24323
Restaurant 9	1779	2124	2844	6877	9570	23194
Restaurant 5	73	3485	4592	5143	8100	21393
Restaurant 11	6156	6110	5791	1759	969	20785
Restaurant 12	209	621	3098	7118	8433	19479
Restaurant 6	238	1235	1822	7074	8207	18576
Restaurant 15	2390	2415	3461	3850	4657	16773

In [297…
```
# Transposing the DataFrame dfRestaurantsTable_setindex
dfRestaurantsTable_setindex_Transpose = dfRestaurantsTable_setindex.T
```

In [298…
```
# Selecting all rows except the last one from the transposed DataFrame dfRestaurantsTable_setindex
q = dfRestaurantsTable_setindex_Transpose.iloc[:-1 , :]
```

In [299…
```
# Adding a 'Years' column to DataFrame q
q['Years'] = ['2017','2018','2019','2020','2021']
```

```
<ipython-input-299-21eb0571b3e7>: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
  q['Years'] = ['2017','2018','2019','2020','2021']
```

In [300…

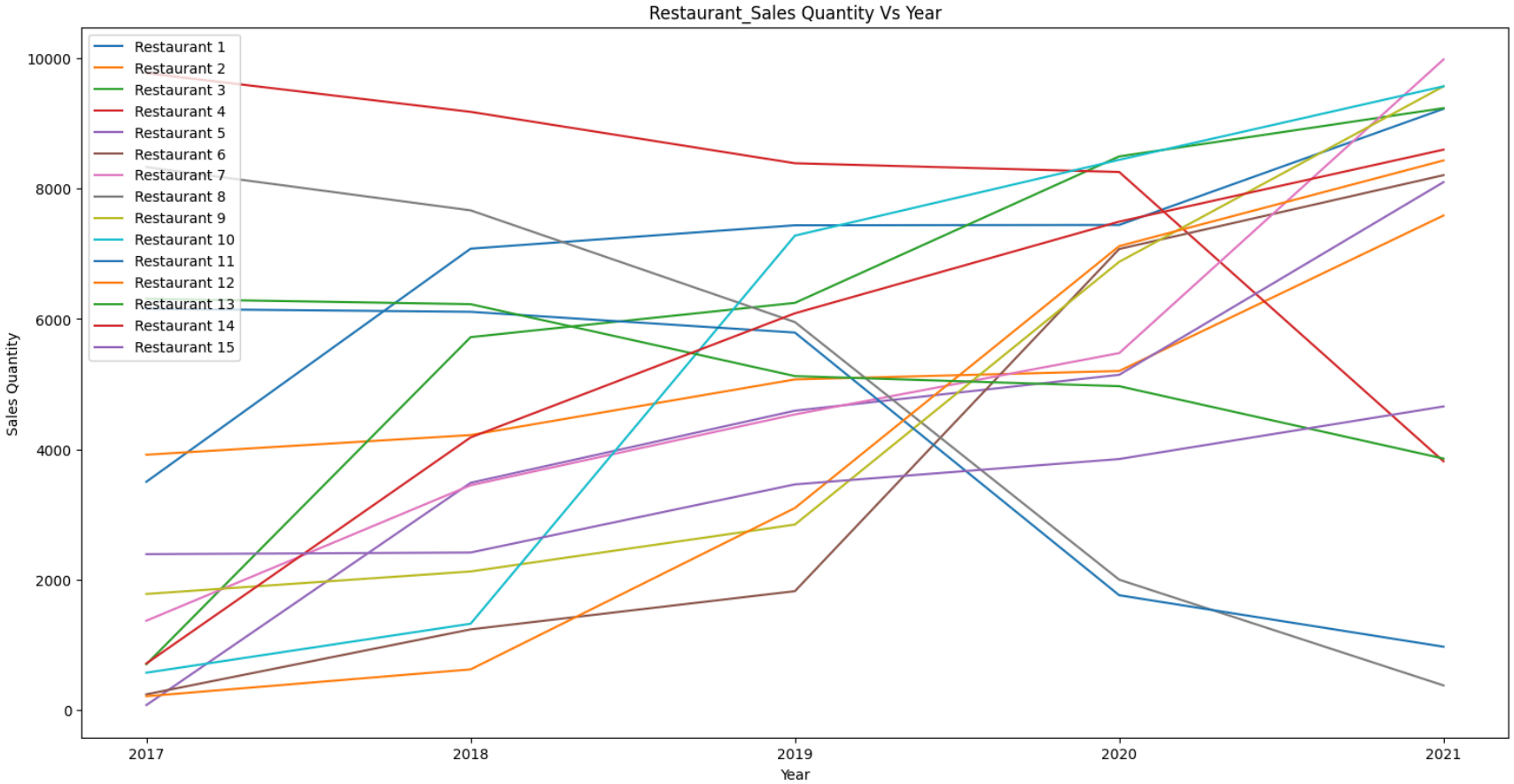
```
# Setting 'Years' as index and converting all values to strings
q.set_index('Years').astype(str)
```

Out[300]:

Account Name	Restaurant 1	Restaurant 2	Restaurant 3	Restaurant 4	Restaurant 5	Restaurant 6	Restaurant 7	Restaurant 8	Restaurant 9	Restaurant 10	Restaurant 11
Years											
2017	3501	3916	700	9773	73	238	1368	8331	1779	570	6156
2018	7079	4218	5721	9179	3485	1235	3447	7667	2124	1322	6110
2019	7438	5072	6247	8390	4592	1822	4535	5952	2844	7279	5791
2020	7443	5201	8495	8256	5143	7074	5476	1998	6877	8443	1759
2021	9225	7588	9236	3815	8100	8207	9983	375	9570	9571	969

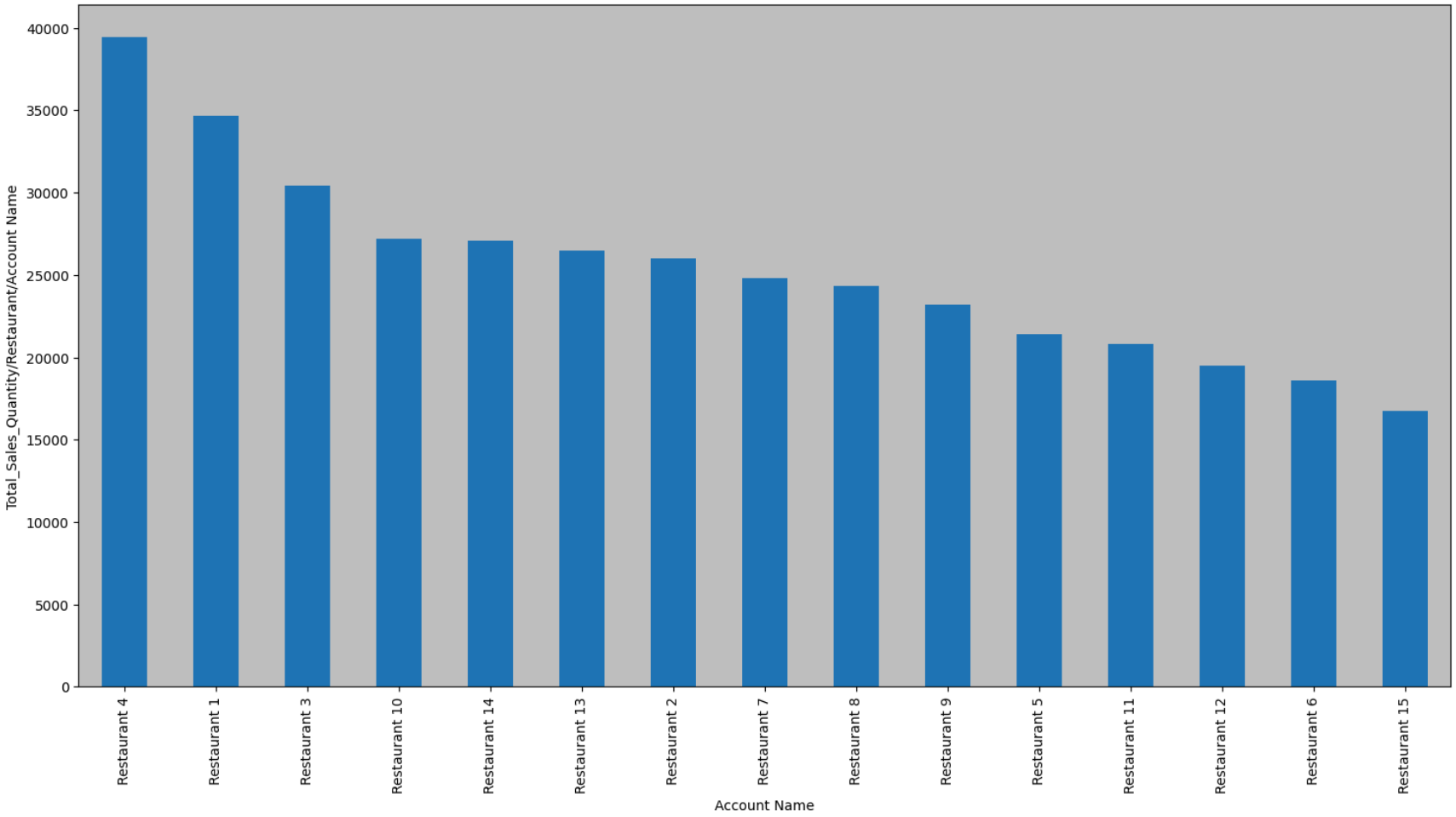
In [301…

```
# Creating a figure and plotting the data
plt.figure(figsize=(18,9))
plt.plot(q.set_index('Years')[0:15])
plt.title('Restaurant_Sales Quantity Vs Year')
plt.xlabel('Year')
plt.ylabel('Sales Quantity')
#plt.legend(['Restaurant 1', 'Restaurant 2', 'Restaurant 3', 'Restaurant 4', 'Restaurant 5', 'Restaurant 6',
#'Restaurant 7', 'Restaurant 8', 'Restaurant 9', 'Restaurant 10',
#'Restaurant 11', 'Restaurant 12', 'Restaurant 13', 'Restaurant 14', 'Restaurant 15'])# Adding legend for the first 15 r
plt.legend(q.columns[:15])  # Adding legend for the first 15 restaurants
plt.show()
```



In [302…

```
# Sorting restaurant accounts by total sales quantity in descending order
ax = B.sort_values(by='Total_Sales Quantity/Restaurant/Account Name', ascending=False)['Total_Sales Quantity/Restaurant/
# Creating a bar plot
ax.set_facecolor("silver")
plt.ylabel('Total_Sales_Quantity/Restaurant/Account Name')
plt.show()
```



In [303…

```
# Sorting sales data for the year 2021 in descending order
B[2021].sort_values(ascending=False)
```

Out[303]:
```
Account Name
Restaurant 7     9983
Restaurant 10    9571
Restaurant 9     9570
Restaurant 3     9236
Restaurant 1     9225
Restaurant 14    8599
Restaurant 12    8433
Restaurant 6     8207
Restaurant 5     8100
Restaurant 2     7588
Restaurant 15    4657
Restaurant 13    3857
Restaurant 4     3815
Restaurant 11     969
Restaurant 8      375
Name: 2021, dtype: int64
```

In [304…
```
# Creating a DataFrame with sorted sales data for the year 2021 and resetting index
DataFrameRestaurant = pd.DataFrame(B[2021].sort_values(ascending=False)).reset_index()
DataFrameRestaurant.head()
```

Out[304]:

	Account Name	2021
0	Restaurant 7	9983
1	Restaurant 10	9571
2	Restaurant 9	9570
3	Restaurant 3	9236
4	Restaurant 1	9225

In [305…
```
# Renaming columns in DataFrameRestaurant
DataFrameRestaurant.rename(columns={'Account Name': 'Restaurant_Ac_Name',
                                    2021: 'Restaurant_Sale_2021'},
          inplace=True, errors='raise')
```

In [306…
```
DataFrameRestaurant.head()
```

Out[306]:

	Restaurant_Ac_Name	Restaurant_Sale_2021
0	Restaurant 7	9983
1	Restaurant 10	9571
2	Restaurant 9	9570
3	Restaurant 3	9236
4	Restaurant 1	9225

In [307…

```
# Filtering DataFrame for accounts containing "Nightclub"
dfNightclubs = df[df['Account Name'].str.contains("Nightclub")]
```

In [308…

```
dfNightclubs.head()
```

Out[308]:

	Account Name	Account Address	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2
30	Nightclub 1	77 Stillwater St, Brooklyn NY 11213	John Mackey	(831) 581-1892	Club	Yes	Yes	Yes	No	No	Yes	No	2519	3938	5190	8
31	Nightclub 2	7061 Bishop St, Yonkers NY 10701	Raymond Heywin	(571) 843-1746	Club	Yes	Yes	Yes	Yes	Yes	Yes	No	138	286	6750	8
32	Nightclub 3	7223 Cedarwood Ave, Brooklyn NY 11221	Janie Roberson	(924) 516-6566	Club	Yes	Yes	Yes	No	No	Yes	Yes	8873	8484	7883	7
33	Nightclub 4	62 Lafayette Ave, Bronx NY 10462	Brooke Hayes	(247) 999-3394	Club	Yes	Yes	Yes	No	No	Yes	Yes	3297	4866	4928	8
34	Nightclub 5	7839 Elm St, Staten Island NY 10306	Lee Niemeyer	(920) 451-3973	Club	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1092	3140	4123	4

In [309…

```
# Selecting relevant columns for 'Nightclub' accounts sales data from 2017 to 2021
dfNightclubsTable = dfNightclubs[['Account Name', 2017,2018,2019,2020,2021]]
```

In [310…

```
dfNightclubsTable.head()
```

Out[310]:

	Account Name	2017	2018	2019	2020	2021
30	Nightclub 1	2519	3938	5190	8203	8780
31	Nightclub 2	138	286	6750	8254	8656
32	Nightclub 3	8873	8484	7883	7499	6592
33	Nightclub 4	3297	4866	4928	8451	9585
34	Nightclub 5	1092	3140	4123	4366	9482

In [311…
```
# Setting 'Account Name' as index for filtered 'Nightclub' accounts sales data
dfNightclubsTable_setindex = dfNightclubsTable.set_index('Account Name')
```

In [312…
```
dfNightclubsTable_setindex.head()
```

Out[312]:

Account Name	2017	2018	2019	2020	2021
Nightclub 1	2519	3938	5190	8203	8780
Nightclub 2	138	286	6750	8254	8656
Nightclub 3	8873	8484	7883	7499	6592
Nightclub 4	3297	4866	4928	8451	9585
Nightclub 5	1092	3140	4123	4366	9482

In [313…
```
# Assigning filtered and indexed data to variable C
C = dfNightclubsTable_setindex
```

In [314…
```
# Calculating total sales quantity per nightclub account and adding to DataFrame C
C['Total_Sales Quantity/Nightclub/Account Name'] = C.sum(axis=1)
```

In [315…
```
C.head()
```

Out[315]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Nightclub/Account Name
Nightclub 1	2519	3938	5190	8203	8780	28630
Nightclub 2	138	286	6750	8254	8656	24084
Nightclub 3	8873	8484	7883	7499	6592	39331
Nightclub 4	3297	4866	4928	8451	9585	31127
Nightclub 5	1092	3140	4123	4366	9482	22203

In [316…

```
# Sorting nightclub accounts by total sales quantity in descending order
C.sort_values(by='Total_Sales Quantity/Nightclub/Account Name', ascending=False)
```

Out[316]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Nightclub/Account Name
Nightclub 3	8873	8484	7883	7499	6592	39331
Nightclub 4	3297	4866	4928	8451	9585	31127
Nightclub 15	431	6231	7478	8039	8271	30450
Nightclub 13	8891	5952	5914	5405	4031	30193
Nightclub 14	1290	4033	6956	7929	8834	29042
Nightclub 9	488	5535	5775	7661	9206	28665
Nightclub 1	2519	3938	5190	8203	8780	28630
Nightclub 6	2541	3794	3984	8803	9338	28460
Nightclub 7	742	3751	4423	8733	9909	27558
Nightclub 2	138	286	6750	8254	8656	24084
Nightclub 12	1038	3615	3712	5819	9589	23773
Nightclub 11	7840	5804	4259	4243	907	23053
Nightclub 5	1092	3140	4123	4366	9482	22203
Nightclub 8	7703	6957	3898	1857	1512	21927
Nightclub 10	376	889	4373	6803	7578	20019

In [317…
```
# Transposing the DataFrame dfNightclubsTable_setindex
dfNightclubsTable_setindex_Transpose = dfNightclubsTable_setindex.T
```

In [318…
```
# Selecting all rows except the last one from the transposed DataFrame dfNightclubsTable_setindex
L = dfNightclubsTable_setindex_Transpose.iloc[:-1 , :]
```

In [319…
```
L.head()
```

Out[319]:

Account Name	Nightclub 1	Nightclub 2	Nightclub 3	Nightclub 4	Nightclub 5	Nightclub 6	Nightclub 7	Nightclub 8	Nightclub 9	Nightclub 10	Nightclub 11	Nightclu 1
2017	2519	138	8873	3297	1092	2541	742	7703	488	376	7840	103
2018	3938	286	8484	4866	3140	3794	3751	6957	5535	889	5804	361
2019	5190	6750	7883	4928	4123	3984	4423	3898	5775	4373	4259	371
2020	8203	8254	7499	8451	4366	8803	8733	1857	7661	6803	4243	581
2021	8780	8656	6592	9585	9482	9338	9909	1512	9206	7578	907	958

In [320…

```
# Adding a 'Years' column to DataFrame L
L['Years'] = ['2017','2018','2019','2020','2021']
L.head()
```

```
<ipython-input-320-95f8ba15d477>: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
  L['Years'] = ['2017','2018','2019','2020','2021']
```

Out[320]:

Account Name	Nightclub 1	Nightclub 2	Nightclub 3	Nightclub 4	Nightclub 5	Nightclub 6	Nightclub 7	Nightclub 8	Nightclub 9	Nightclub 10	Nightclub 11	Nightclu 1
2017	2519	138	8873	3297	1092	2541	742	7703	488	376	7840	103
2018	3938	286	8484	4866	3140	3794	3751	6957	5535	889	5804	361
2019	5190	6750	7883	4928	4123	3984	4423	3898	5775	4373	4259	371
2020	8203	8254	7499	8451	4366	8803	8733	1857	7661	6803	4243	581
2021	8780	8656	6592	9585	9482	9338	9909	1512	9206	7578	907	958

In [321…

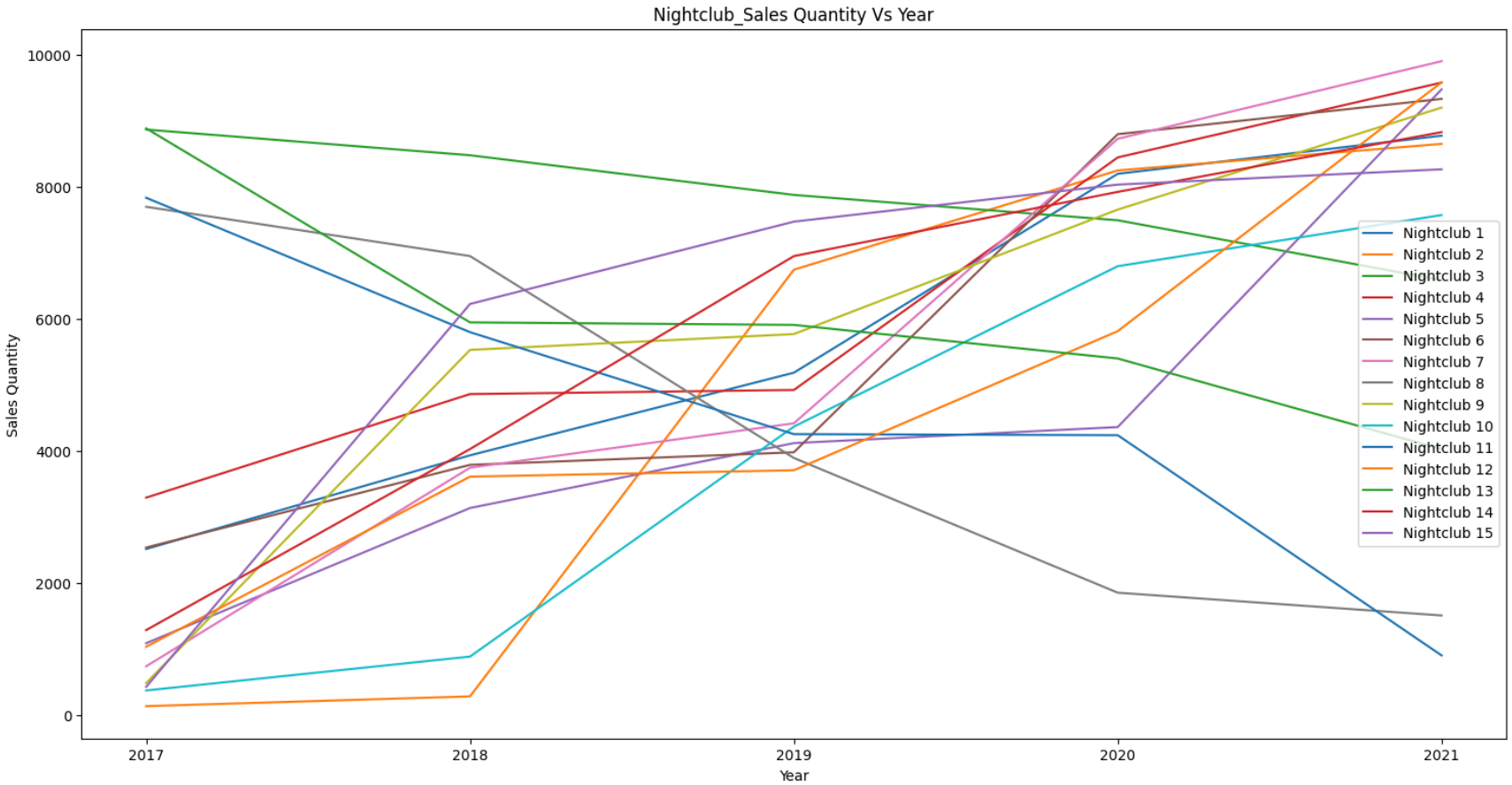
```
# Setting 'Years' as index and converting all values to strings in DataFrame L
L.set_index('Years').astype(str)
```

Out[321]:

Account Name	Nightclub 1	Nightclub 2	Nightclub 3	Nightclub 4	Nightclub 5	Nightclub 6	Nightclub 7	Nightclub 8	Nightclub 9	Nightclub 10	Nightclub 11	Nightclu 1
Years												
2017	2519	138	8873	3297	1092	2541	742	7703	488	376	7840	103
2018	3938	286	8484	4866	3140	3794	3751	6957	5535	889	5804	361
2019	5190	6750	7883	4928	4123	3984	4423	3898	5775	4373	4259	371
2020	8203	8254	7499	8451	4366	8803	8733	1857	7661	6803	4243	581
2021	8780	8656	6592	9585	9482	9338	9909	1512	9206	7578	907	958

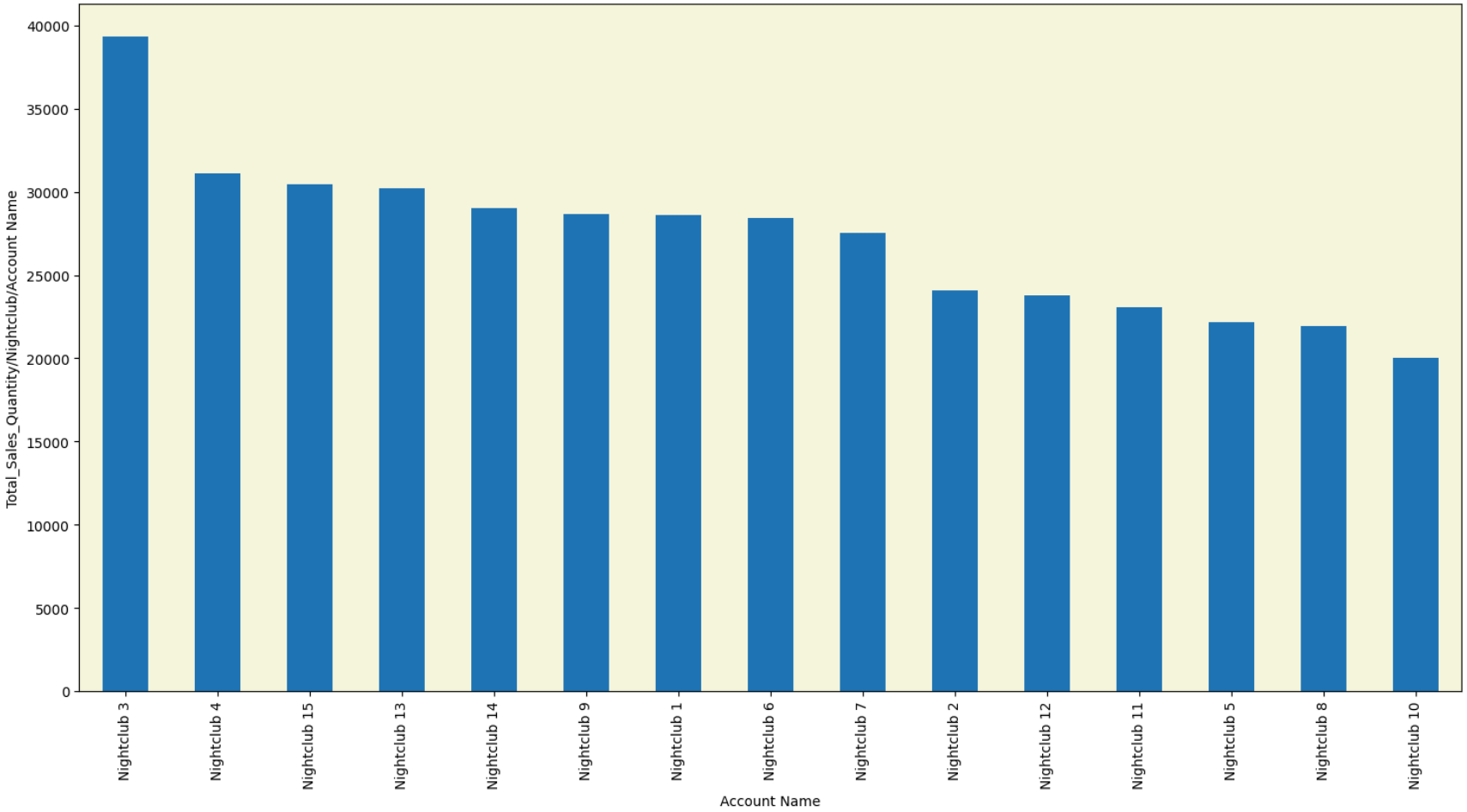
In [322…

```
# Creating a figure and plotting the data
plt.figure(figsize=(18,9))
plt.plot(L.set_index('Years')[0:15])# Plotting data for first 15 nightclubs
plt.title('Nightclub_Sales Quantity Vs Year')
plt.xlabel('Year')
plt.ylabel('Sales Quantity')
#plt.legend(['Nightclub 1', 'Nightclub 2', 'Nightclub 3', 'Nightclub 4', 'Nightclub 5', 'Nightclub 6',
#'Nightclub 7', 'Nightclub 8', 'Nightclub 9', 'Nightclub 10',
#'Nightclub 11', 'Nightclub 12', 'Nightclub 13', 'Nightclub 14', 'Nightclub 15'])# Adding legend for the first 15 nightc
plt.legend(L.columns[:15])  # Adding legend for the first 15 nightclubs
plt.show()
```



In [323…

```
# Sorting nightclub accounts by total sales quantity in descending order
ax = C.sort_values(by='Total_Sales Quantity/Nightclub/Account Name', ascending=False)['Total_Sales Quantity/Nightclub/A
# Creating a bar plot
ax.set_facecolor("beige")
plt.ylabel('Total_Sales_Quantity/Nightclub/Account Name')
plt.show()
```



In [324…

```
# Sorting sales data for the year 2021 in descending order
C[2021].sort_values(ascending=False)
```

Out[324]:
```
Account Name
Nightclub 7     9909
Nightclub 12    9589
Nightclub 4     9585
Nightclub 5     9482
Nightclub 6     9338
Nightclub 9     9206
Nightclub 14    8834
Nightclub 1     8780
Nightclub 2     8656
Nightclub 15    8271
Nightclub 10    7578
Nightclub 3     6592
Nightclub 13    4031
Nightclub 8     1512
Nightclub 11     907
Name: 2021, dtype: int64
```

In [325…
```
# Creating a DataFrame with sorted sales data for nightclubs in the year 2021 and resetting index
DataFrameNightclub = pd.DataFrame(C[2021].sort_values(ascending=False)).reset_index()
DataFrameNightclub.head()
```

Out[325]:

	Account Name	2021
0	Nightclub 7	9909
1	Nightclub 12	9589
2	Nightclub 4	9585
3	Nightclub 5	9482
4	Nightclub 6	9338

In [326…
```
# Renaming columns in DataFrameNightclub
DataFrameNightclub.rename(columns={'Account Name': 'Nightclub_Ac_Name',
                                   2021: 'Nightclub_Sale_2021'},
          inplace=True, errors='raise')
DataFrameNightclub.head()
```

Out[326]:

	Nightclub_Ac_Name	Nightclub_Sale_2021
0	Nightclub 7	9909
1	Nightclub 12	9589
2	Nightclub 4	9585
3	Nightclub 5	9482
4	Nightclub 6	9338

In [327…

```
# Filtering DataFrame for accounts containing "Event Venue"
dfEventVenues = df[df['Account Name'].str.contains("Event Venue")]
```

In [328…

```
dfEventVenues.head()
```

Out[328]:

	Account Name	Account Address	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020
45	Event Venue 1	7184 Center Court, Brooklyn NY 11208	Richard Breaux	(685) 981-8556	Hotel	Yes	No	No	No	No	Yes	No	8156	1245	791	338
46	Event Venue 2	815 2nd St, New York NY 10028	Craig Collins	(828) 840-2736	Hotel	Yes	Yes	Yes	Yes	No	Yes	No	299	657	6238	8922
47	Event Venue 3	9875 Franklin Rd, Brooklyn NY 11223	Donna Lam	(931) 618-9558	Hotel	Yes	Yes	Yes	No	No	Yes	No	1323	4963	6292	6728
48	Event Venue 4	601 Bank Ave, Brooklyn NY 11218	Teresa Vasbinder	(261) 690-0303	Hotel	Yes	No	No	No	No	Yes	No	8466	4079	2797	2245
49	Event Venue 5	21 Yukon St, Bronx NY 10451	Andre Mobley	(597) 701-9429	Hotel	Yes	Yes	Yes	No	No	Yes	No	870	2428	7386	8835

In [329…
```
# Selecting relevant columns for 'Event Venue' accounts sales data from 2017 to 2021
dfEventVenuesTable = dfEventVenues[['Account Name', 2017,2018,2019,2020,2021]]
```

In [330…
```
dfEventVenuesTable.head()
```

Out[330]:

	Account Name	2017	2018	2019	2020	2021
45	Event Venue 1	8156	1245	791	338	44
46	Event Venue 2	299	657	6238	8922	9081
47	Event Venue 3	1323	4963	6292	6728	8202
48	Event Venue 4	8466	4079	2797	2245	1696
49	Event Venue 5	870	2428	7386	8835	9766

In [331…

```
# Setting 'Account Name' as index for filtered 'Event Venue' accounts sales data
dfEventVenuesTable_setindex = dfEventVenuesTable.set_index('Account Name')
dfEventVenuesTable_setindex.head()
```

Out[331]:

Account Name	2017	2018	2019	2020	2021
Event Venue 1	8156	1245	791	338	44
Event Venue 2	299	657	6238	8922	9081
Event Venue 3	1323	4963	6292	6728	8202
Event Venue 4	8466	4079	2797	2245	1696
Event Venue 5	870	2428	7386	8835	9766

In [332…

```
# Assigning filtered and indexed data to variable D
D = dfEventVenuesTable_setindex
```

In [333…

```
# Calculating total sales quantity per event venue account and adding to DataFrame D
D['Total_Sales Quantity/Event Venue/Account Name'] = D.sum(axis=1)
```

In [334…

```
D.head()
```

Out[334]:

Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Event Venue/Account Name
Event Venue 1	8156	1245	791	338	44	10574
Event Venue 2	299	657	6238	8922	9081	25197
Event Venue 3	1323	4963	6292	6728	8202	27508
Event Venue 4	8466	4079	2797	2245	1696	19283
Event Venue 5	870	2428	7386	8835	9766	29285

In [335…

```
# Sorting event venue accounts by total sales quantity in descending order
D.sort_values(by='Total_Sales Quantity/Event Venue/Account Name', ascending=False)
```

Out[335]:

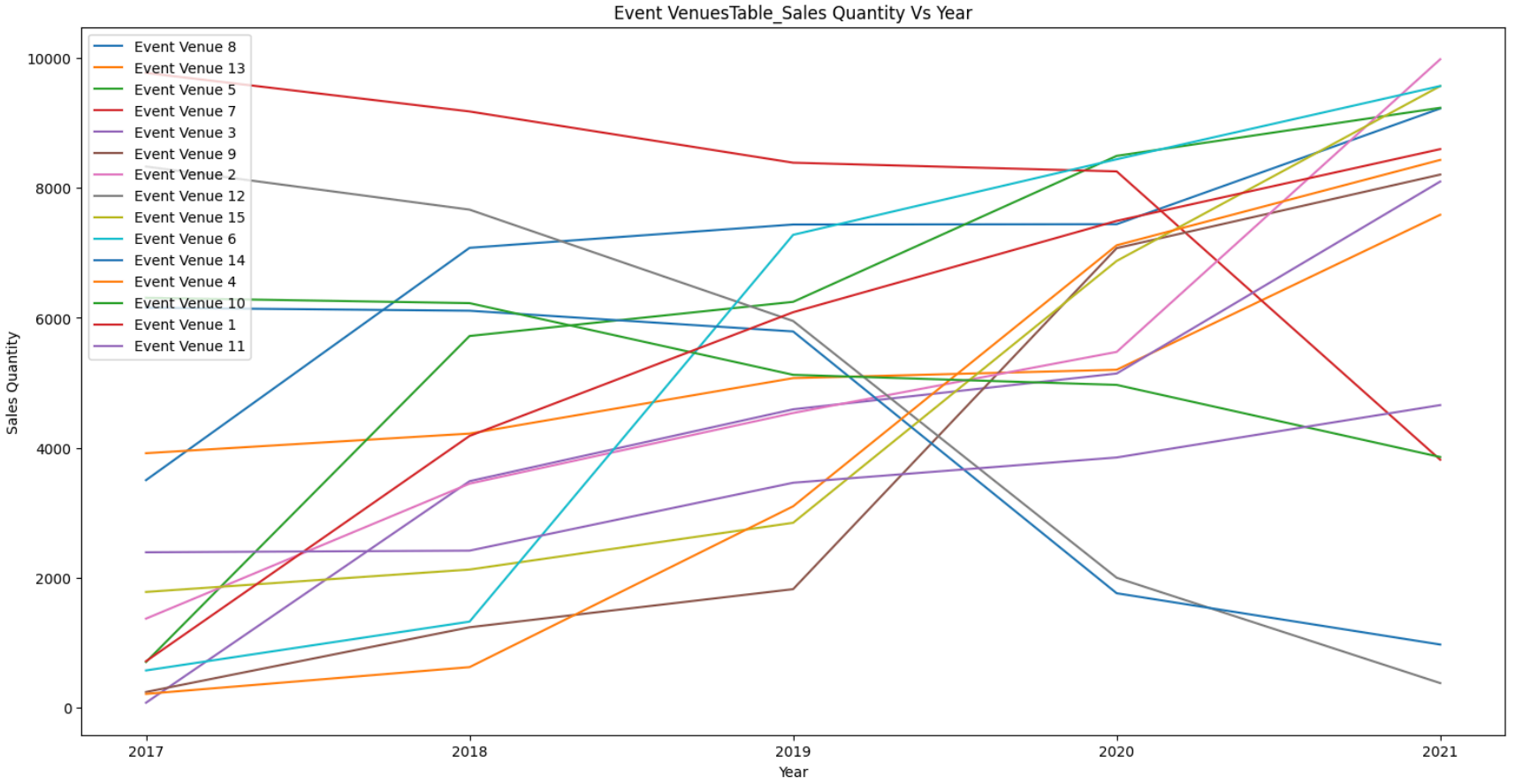
Account Name	2017	2018	2019	2020	2021	Total_Sales Quantity/Event Venue/Account Name
Event Venue 8	9791	9610	7534	5080	4936	36951
Event Venue 13	1263	2517	8042	8222	9686	29730
Event Venue 5	870	2428	7386	8835	9766	29285
Event Venue 7	1082	3353	6351	8550	9272	28608
Event Venue 3	1323	4963	6292	6728	8202	27508
Event Venue 9	1357	4189	5407	6233	9681	26867
Event Venue 2	299	657	6238	8922	9081	25197
Event Venue 12	8034	6541	3311	3254	2687	23827
Event Venue 15	1014	2254	4534	6796	7730	22328
Event Venue 6	1497	1768	2804	5718	9822	21609
Event Venue 14	1032	3919	4466	5568	6476	21461
Event Venue 4	8466	4079	2797	2245	1696	19283
Event Venue 10	576	2628	3612	5066	5156	17038
Event Venue 1	8156	1245	791	338	44	10574
Event Venue 11	128	416	747	1028	6357	8676

In [336…

```
# Transposing the DataFrame dfEventVenuesTable_setindex
dfEventVenuesTable_setindex_Transpose = dfEventVenuesTable_setindex.T
# Selecting all rows except the last one from the transposed DataFrame dfEventVenuesTable_setindex
O = dfEventVenuesTable_setindex_Transpose.iloc[:-1 , :]
# Adding a 'Years' column to DataFrame O
O['Years'] = ['2017','2018','2019','2020','2021']
# Converting all values to strings in DataFrame O
O.set_index('Years').astype(str)
# Creating a figure and plotting the data
plt.figure(figsize=(18,9))
plt.plot(q.set_index('Years')[0:15])
plt.title('Event VenuesTable_Sales Quantity Vs Year')
plt.xlabel('Year')
plt.ylabel('Sales Quantity')
```

```
plt.legend(['Event Venue 8', 'Event Venue 13', 'Event Venue 5', 'Event Venue 7', 'Event Venue 3', 'Event Venue 9',
'Event Venue 2', 'Event Venue 12', 'Event Venue 15', 'Event Venue 6',
'Event Venue 14', 'Event Venue 4', 'Event Venue 10', 'Event Venue 1', 'Event Venue 11'])
plt.show()
```

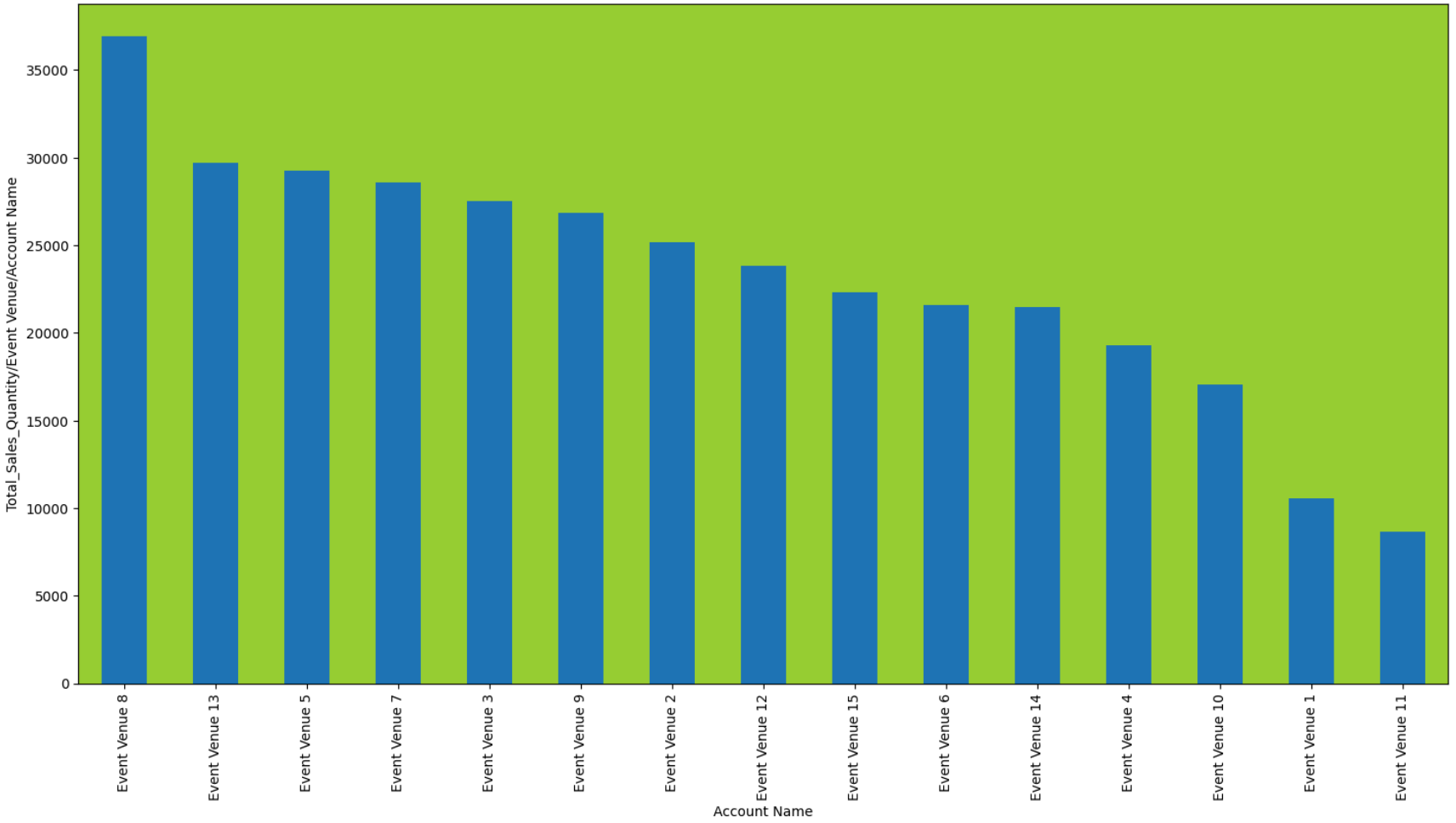
```
<ipython-input-336-da07c6436edf>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  O['Years'] = ['2017','2018','2019','2020','2021']
```



In [337…

```
# Sorting event venue accounts by total sales quantity in descending order
ax = D.sort_values(by='Total_Sales Quantity/Event Venue/Account Name', ascending=False)['Total_Sales Quantity/Event Ven
ax.set_facecolor("yellowgreen")
```

```
plt.ylabel('Total_Sales_Quantity/Event Venue/Account Name')
plt.show()
```



In [337…

In [338…

```
# Sorting sales data for the year 2021 in descending order
D[2021].sort_values(ascending=False)
```

Out[338]:
```
Account Name
Event Venue 6     9822
Event Venue 5     9766
Event Venue 13    9686
Event Venue 9     9681
Event Venue 7     9272
Event Venue 2     9081
Event Venue 3     8202
Event Venue 15    7730
Event Venue 14    6476
Event Venue 11    6357
Event Venue 10    5156
Event Venue 8     4936
Event Venue 12    2687
Event Venue 4     1696
Event Venue 1       44
Name: 2021, dtype: int64
```

In [339…
```
# Creating a DataFrame with sorted sales data for Event Venues in the year 2021 and resetting index
DataFrameEventVenue = pd.DataFrame(D[2021].sort_values(ascending=False)).reset_index()
DataFrameEventVenue.head()
```

Out[339]:

	Account Name	2021
0	Event Venue 6	9822
1	Event Venue 5	9766
2	Event Venue 13	9686
3	Event Venue 9	9681
4	Event Venue 7	9272

In [340…
```
# Renaming columns in DataFrameEventVenue
DataFrameEventVenue.rename(columns={'Account Name': 'EventVenue_Ac_Name',
                                    2021: 'EventVenue_Sale_2021'},
          inplace=True, errors='raise')
DataFrameEventVenue.head()
```

Out[340]:

	EventVenue_Ac_Name	EventVenue_Sale_2021
0	Event Venue 6	9822
1	Event Venue 5	9766
2	Event Venue 13	9686
3	Event Venue 9	9681
4	Event Venue 7	9272

Rankings and Sales Data for Accounts in 2021

In [341…
```
# Concatenating DataFrames DataFrameBarRank, DataFrameRestaurant, DataFrameNightclub, and DataFrameEventVenue horizonta
data = pd.concat([DataFrameBarRank,DataFrameRestaurant,DataFrameNightclub,DataFrameEventVenue], axis=1, join='inner')
```

In [342…
```
# Renaming columns
data.rename(columns={'Account Name': 'EventVenue_Ac_Name', 2021: 'EventVenue_Sale_2021'}, inplace=True)
```

In [343…
```
data
```

Out[343]:

	Rank	Bar_Ac_Name	Bar_Sale_2021	Restaurant_Ac_Name	Restaurant_Sale_2021	Nightclub_Ac_Name	Nightclub_Sale_2021	EventVenue_Ac_
0	1	Bar 3	9768	Restaurant 7	9983	Nightclub 7	9909	Event V
1	2	Bar 8	9759	Restaurant 10	9571	Nightclub 12	9589	Event V
2	3	Bar 4	9428	Restaurant 9	9570	Nightclub 4	9585	Event Ve
3	4	Bar 14	9271	Restaurant 3	9236	Nightclub 5	9482	Event V
4	5	Bar 1	9093	Restaurant 1	9225	Nightclub 6	9338	Event V
5	6	Bar 6	8758	Restaurant 14	8599	Nightclub 9	9206	Event V
6	7	Bar 13	8592	Restaurant 12	8433	Nightclub 14	8834	Event V
7	8	Bar 2	6909	Restaurant 6	8207	Nightclub 1	8780	Event Ve
8	9	Bar 10	6002	Restaurant 5	8100	Nightclub 2	8656	Event Ve
9	10	Bar 5	5873	Restaurant 2	7588	Nightclub 15	8271	Event Ve
10	11	Bar 12	5382	Restaurant 15	4657	Nightclub 10	7578	Event Ve
11	12	Bar 9	2373	Restaurant 13	3857	Nightclub 3	6592	Event V
12	13	Bar 11	2359	Restaurant 4	3815	Nightclub 13	4031	Event Ve
13	14	Bar 15	369	Restaurant 11	969	Nightclub 8	1512	Event V
14	15	Bar 7	211	Restaurant 8	375	Nightclub 11	907	Event V

Best and Worst Performing Accounts by Account Type (5 Year CAGR)

In [343…

In [344…

```
# Calculate CAGR function
def calculate_cagr(beginning_value, ending_value, num_years):
    return (ending_value / beginning_value) ** (1 / num_years) - 1

# Calculate CAGR for each account
df_To_Calculated_CAGR['CAGR'] = df_To_Calculated_CAGR.apply(
    lambda row: calculate_cagr(row[2017], row[2021], 4), axis=1)
```

```
# Order by CAGR in descending order and select relevant columns
df_To_Calculated_CAGR_sorted = df_To_Calculated_CAGR[['CAGR', 'Account Type', 'Account Name']].sort_values(by='CAGR', a

# Set 'Account Name' as the index
df_To_Calculated_CAGR_sorted.set_index('Account Name', inplace=True)

# Display the DataFrame with CAGR
df_To_Calculated_CAGR_sorted[['CAGR', 'Account Type']]
```

Out[344]:

Account Name	CAGR	Account Type
Bar 13	3.349815	Bar
Restaurant 5	2.245567	Restaurant
Nightclub 2	1.814230	Club
Event Venue 11	1.654670	Hotel
Restaurant 12	1.520339	Restaurant
Restaurant 6	1.423270	Restaurant
Event Venue 2	1.347554	Hotel
Nightclub 10	1.118808	Club
Nightclub 15	1.093005	Club
Nightclub 9	1.084072	Club
Restaurant 10	1.024280	Restaurant
Nightclub 7	0.911642	Club
Restaurant 3	0.905884	Restaurant
Restaurant 14	0.864198	Restaurant
Event Venue 5	0.830414	Hotel
Bar 14	0.811469	Bar
Bar 4	0.796068	Bar
Nightclub 12	0.743388	Club
Event Venue 10	0.729707	Hotel
Nightclub 5	0.716601	Club
Event Venue 7	0.710947	Hotel
Bar 3	0.685951	Bar
Event Venue 13	0.664122	Hotel
Event Venue 15	0.661634	Hotel

Account Name	CAGR	Account Type
Restaurant 7	0.643591	Restaurant
Event Venue 9	0.634312	Hotel
Nightclub 14	0.617677	Club
Event Venue 6	0.600459	Hotel
Event Venue 14	0.582730	Hotel
Event Venue 3	0.577938	Hotel
Bar 8	0.576226	Bar
Restaurant 9	0.522944	Restaurant
Bar 1	0.463527	Bar
Bar 5	0.425826	Bar
Bar 10	0.407347	Bar
Bar 6	0.390756	Bar
Nightclub 6	0.384562	Club
Bar 12	0.369056	Bar
Nightclub 1	0.366365	Club
Nightclub 4	0.305775	Club
Restaurant 1	0.274071	Restaurant
Bar 2	0.254898	Bar
Restaurant 15	0.181482	Restaurant
Restaurant 2	0.179835	Restaurant
Nightclub 3	-0.071597	Club
Restaurant 13	-0.115756	Restaurant
Event Venue 8	-0.157370	Hotel
Nightclub 13	-0.179430	Club

Account Name	CAGR	Account Type
Restaurant 4	-0.209564	Restaurant
Event Venue 12	-0.239527	Hotel
Bar 11	-0.252479	Bar
Bar 9	-0.297906	Bar
Event Venue 4	-0.330983	Hotel
Nightclub 8	-0.334385	Club
Restaurant 11	-0.370122	Restaurant
Nightclub 11	-0.416793	Club
Restaurant 8	-0.539390	Restaurant
Bar 15	-0.550739	Bar
Bar 7	-0.611392	Bar
Event Venue 1	-0.728985	Hotel

In [345…
```
CopyOfdf_To_Calculated_CAGR_sorted =df_To_Calculated_CAGR_sorted[['CAGR', 'Account Type']].copy()
```

Top 2 Accounts in green color and Bottom 2 Accounts in red color for Bar Account

---

In [346…
```
# Order by CAGR in descending order and select relevant columns
df_To_Calculated_CAGR_sorted = df_To_Calculated_CAGR[['CAGR', 'Account Type', 'Account Name']].sort_values(by='CAGR', a

# Set 'Account Name' as the index
df_To_Calculated_CAGR_sorted.set_index('Account Name', inplace=True)

# Filtering the DataFrame
filtered_df = df_To_Calculated_CAGR_sorted[df_To_Calculated_CAGR_sorted['Account Type'] == 'Bar']

# Selecting the columns 'CAGR' and 'Account Type'
result = filtered_df[['CAGR', 'Account Type']]
```

```
# Function to apply the color
def highlight_rows(row):
    if row.name in result.index[:2]:  # First two rows
        return ['background-color: green'] * len(row)
    elif row.name in result.index[-2:]:  # Last two rows
        return ['background-color: red'] * len(row)
    else:
        return [''] * len(row)

# Applying the function to the DataFrame
styled_result = result.style.apply(highlight_rows, axis=1)

# Display the styled DataFrame
styled_result
```

Out[346]:

Account Name	CAGR	Account Type
Bar 13	3.349815	Bar
Bar 14	0.811469	Bar
Bar 4	0.796068	Bar
Bar 3	0.685951	Bar
Bar 8	0.576226	Bar
Bar 1	0.463527	Bar
Bar 5	0.425826	Bar
Bar 10	0.407347	Bar
Bar 6	0.390756	Bar
Bar 12	0.369056	Bar
Bar 2	0.254898	Bar
Bar 11	-0.252479	Bar
Bar 9	-0.297906	Bar
Bar 15	-0.550739	Bar
Bar 7	-0.611392	Bar

Top 2 Accounts in green color and Bottom 2 Accounts in red color for Restaurant Account

---

In [347…

```
# Order by CAGR in descending order and select relevant columns
df_To_Calculated_CAGR_sorted = df_To_Calculated_CAGR[['CAGR', 'Account Type', 'Account Name']].sort_values(by='CAGR', a

# Set 'Account Name' as the index
df_To_Calculated_CAGR_sorted.set_index('Account Name', inplace=True)

# Filtering the DataFrame
filtered_df = df_To_Calculated_CAGR_sorted[df_To_Calculated_CAGR_sorted['Account Type'] == 'Restaurant']

# Selecting the columns 'CAGR' and 'Account Type'
result = filtered_df[['CAGR', 'Account Type']]

# Function to apply the color
def highlight_rows(row):
    if row.name in result.index[:2]:  # First two rows
        return ['background-color: green'] * len(row)
    elif row.name in result.index[-2:]:  # Last two rows
        return ['background-color: red'] * len(row)
    else:
        return [''] * len(row)

# Applying the function to the DataFrame
styled_result = result.style.apply(highlight_rows, axis=1)

# Display the styled DataFrame
```

Out[347]:

Account Name	CAGR	Account Type
Restaurant 5	2.245567	Restaurant
Restaurant 12	1.520339	Restaurant
Restaurant 6	1.423270	Restaurant
Restaurant 10	1.024280	Restaurant
Restaurant 3	0.905884	Restaurant
Restaurant 14	0.864198	Restaurant
Restaurant 7	0.643591	Restaurant
Restaurant 9	0.522944	Restaurant
Restaurant 1	0.274071	Restaurant
Restaurant 15	0.181482	Restaurant
Restaurant 2	0.179835	Restaurant
Restaurant 13	-0.115756	Restaurant
Restaurant 4	-0.209564	Restaurant
Restaurant 11	-0.370122	Restaurant
Restaurant 8	-0.539390	Restaurant

Top 2 Accounts in green color and Bottom 2 Accounts in red color for Hotel Account

In [348…

```
# Order by CAGR in descending order and select relevant columns
df_To_Calculated_CAGR_sorted = df_To_Calculated_CAGR[['CAGR', 'Account Type', 'Account Name']].sort_values(by='CAGR', a

# Set 'Account Name' as the index
df_To_Calculated_CAGR_sorted.set_index('Account Name', inplace=True)

# Filtering the DataFrame
filtered_df = df_To_Calculated_CAGR_sorted[df_To_Calculated_CAGR_sorted['Account Type'] == 'Hotel']

# Selecting the columns 'CAGR' and 'Account Type'
result = filtered_df[['CAGR', 'Account Type']]
```

```
# Function to apply the color
def highlight_rows(row):
    if row.name in result.index[:2]:  # First two rows
        return ['background-color: green'] * len(row)
    elif row.name in result.index[-2:]:  # Last two rows
        return ['background-color: red'] * len(row)
    else:
        return [''] * len(row)

# Applying the function to the DataFrame
styled_result = result.style.apply(highlight_rows, axis=1)

# Display the styled DataFrame
styled_result
```

Out[348]:

Account Name	CAGR	Account Type
Event Venue 11	1.654670	Hotel
Event Venue 2	1.347554	Hotel
Event Venue 5	0.830414	Hotel
Event Venue 10	0.729707	Hotel
Event Venue 7	0.710947	Hotel
Event Venue 13	0.664122	Hotel
Event Venue 15	0.661634	Hotel
Event Venue 9	0.634312	Hotel
Event Venue 6	0.600459	Hotel
Event Venue 14	0.582730	Hotel
Event Venue 3	0.577938	Hotel
Event Venue 8	-0.157370	Hotel
Event Venue 12	-0.239527	Hotel
Event Venue 4	-0.330983	Hotel
Event Venue 1	-0.728985	Hotel

Top 2 Accounts in green color and Bottom 2 Accounts in red color for Club Account

In [349…

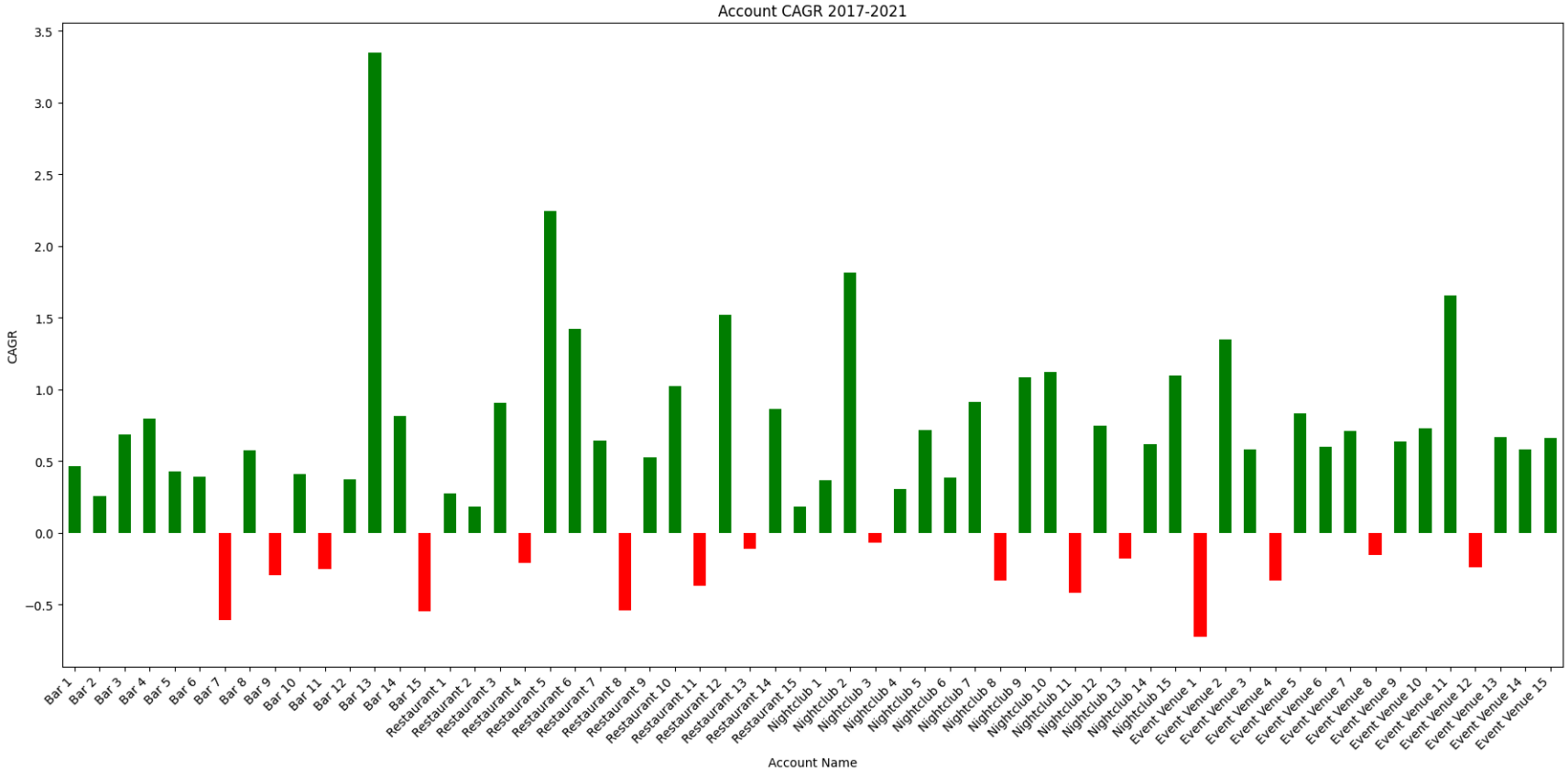
```
# Order by CAGR in descending order and select relevant columns
df_To_Calculated_CAGR_sorted = df_To_Calculated_CAGR[['CAGR', 'Account Type', 'Account Name']].sort_values(by='CAGR', a

# Set 'Account Name' as the index
df_To_Calculated_CAGR_sorted.set_index('Account Name', inplace=True)

# Filtering the DataFrame
filtered_df = df_To_Calculated_CAGR_sorted[df_To_Calculated_CAGR_sorted['Account Type'] == 'Club']

# Selecting the columns 'CAGR' and 'Account Type'
result = filtered_df[['CAGR', 'Account Type']]

# Function to apply the color
def highlight_rows(row):
    if row.name in result.index[:2]:  # First two rows
        return ['background-color: green'] * len(row)
    elif row.name in result.index[-2:]:  # Last two rows
        return ['background-color: red'] * len(row)
    else:
        return [''] * len(row)

# Applying the function to the DataFrame
styled_result = result.style.apply(highlight_rows, axis=1)

# Display the styled DataFrame
styled_result
```

Out[349]:

Account Name	CAGR	Account Type
Nightclub 2	1.814230	Club
Nightclub 10	1.118808	Club
Nightclub 15	1.093005	Club
Nightclub 9	1.084072	Club
Nightclub 7	0.911642	Club
Nightclub 12	0.743388	Club
Nightclub 5	0.716601	Club
Nightclub 14	0.617677	Club
Nightclub 6	0.384562	Club
Nightclub 1	0.366365	Club
Nightclub 4	0.305775	Club
Nightclub 3	-0.071597	Club
Nightclub 13	-0.179430	Club
Nightclub 8	-0.334385	Club
Nightclub 11	-0.416793	Club

In [350…

```
import matplotlib.pyplot as plt
# Calculate colors based on positive or negative CAGR
colors = ['green' if cagr > 0 else 'red' for cagr in df_To_Calculated_CAGR['CAGR']]
# Set 'Account Name' as the index
df_To_Calculated_CAGR.set_index('Account Name', inplace=True)
# Plotting the bar chart
df_To_Calculated_CAGR['CAGR'].plot(kind='bar', figsize=(18, 9), color=colors)
plt.title('Account CAGR 2017-2021')
plt.xlabel('Account Name')
plt.ylabel('CAGR')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



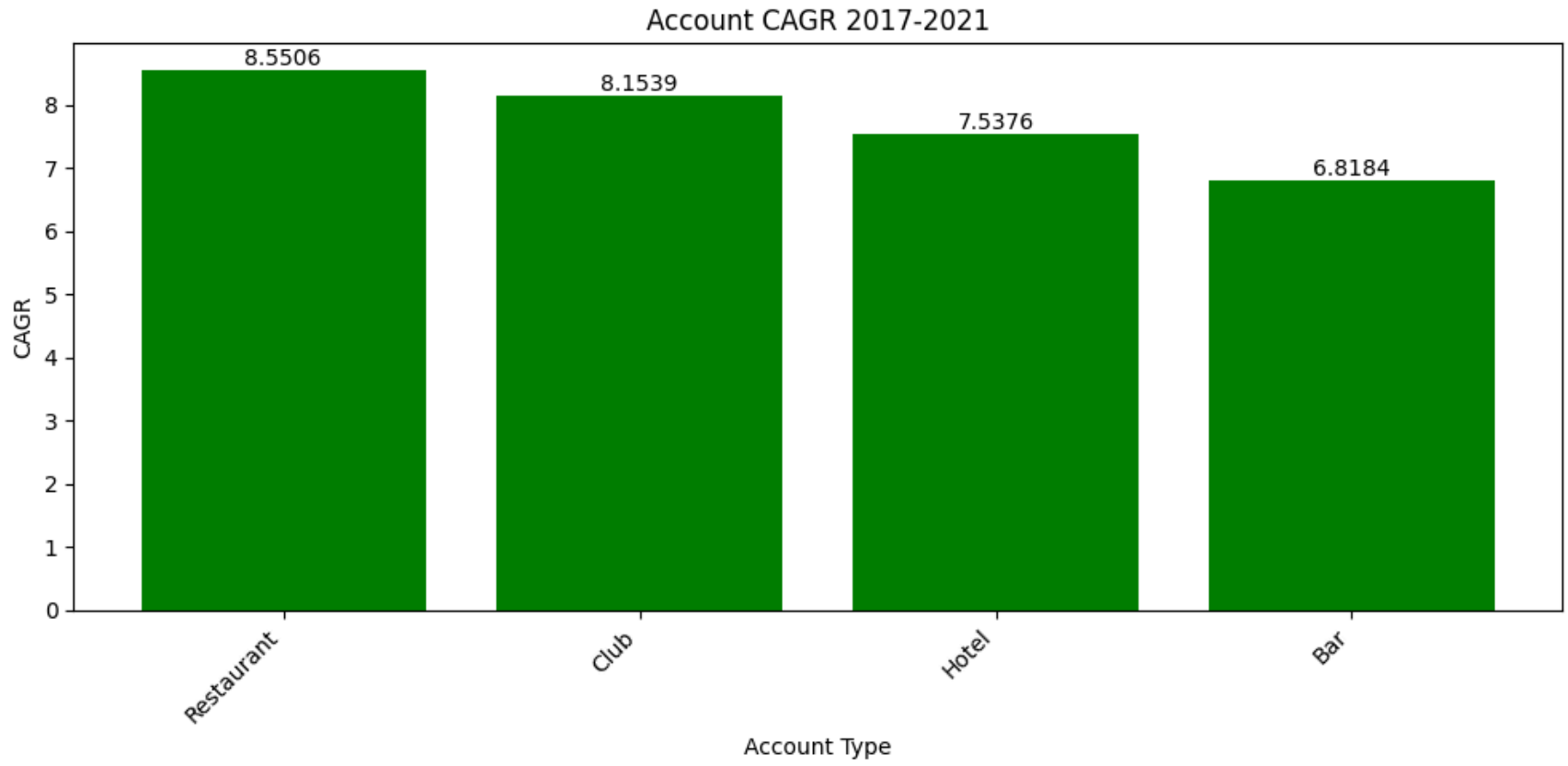
In [350…

In [351…
```
dfAccountTypeCAGR=df_To_Calculated_CAGR[['CAGR','Account Type']].copy()
```

In [352…
```
dfAccountTypeCAGR.set_index('Account Type', inplace=True)
```

In [353…
```
# Group by Account Type and sum the CAGR
dfAccountTypeCAGR = dfAccountTypeCAGR.groupby('Account Type')['CAGR'].sum().reset_index()
# Sort by descending CAGR
dfAccountTypeCAGR = dfAccountTypeCAGR.sort_values(by='CAGR', ascending=False)
# Calculate colors based on positive or negative CAGR
colors = ['green' if cagr > 0 else 'red' for cagr in dfAccountTypeCAGR['CAGR']]

# Plotting the bar chart
plt.figure(figsize=(10, 5))
plt.bar(dfAccountTypeCAGR['Account Type'], dfAccountTypeCAGR['CAGR'], color=colors)
```

```
plt.title('Account CAGR 2017-2021')
plt.xlabel('Account Type')
plt.ylabel('CAGR')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.tight_layout()
bars = plt.bar(dfAccountTypeCAGR['Account Type'], dfAccountTypeCAGR['CAGR'], color=colors)
# Add value labels on top of the bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 4), ha='center', va='bottom')

plt.show()
```



In [353…

In [354…

```
# Group by 'Account Type' and sum 'CAGR'
grouped = df_To_Calculated_CAGR.groupby('Account Type')['CAGR'].sum().reset_index()
grouped.rename(columns={'CAGR': 'CAGRAcType'}, inplace=True)
```

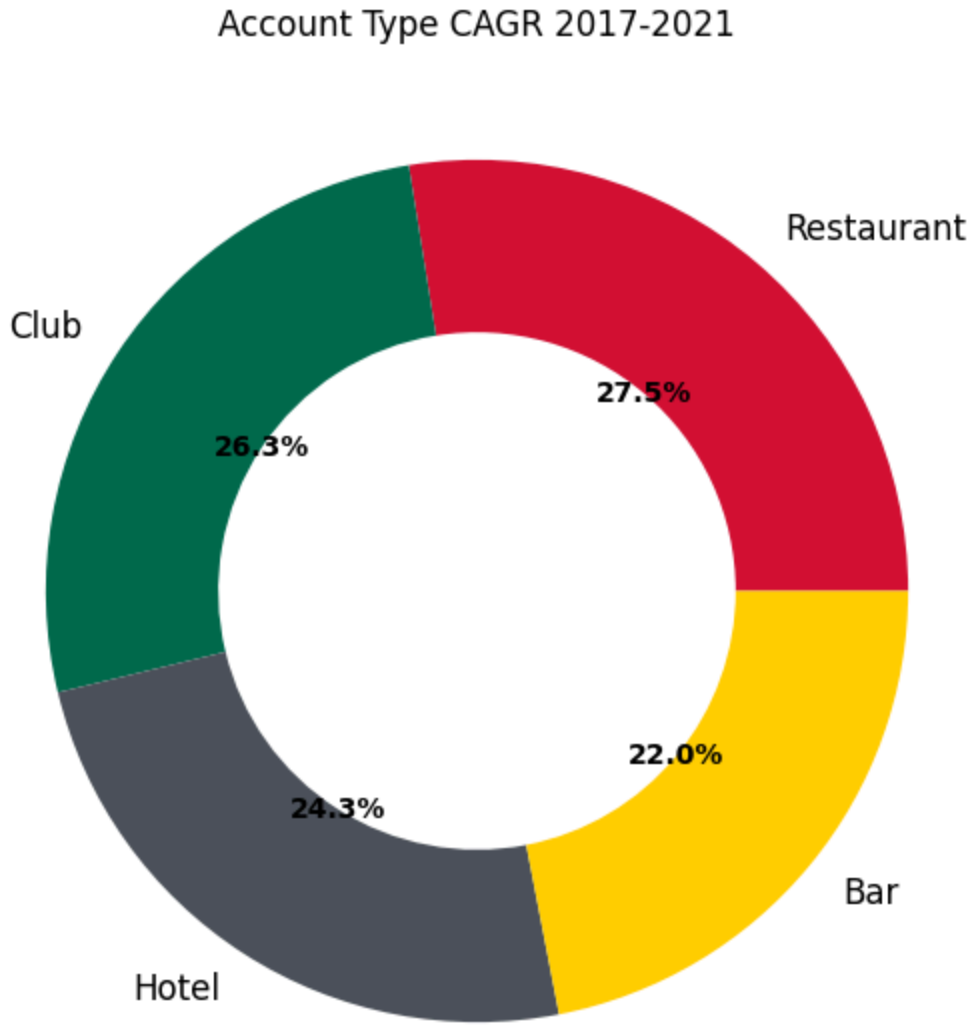
```
# Calculate total sum of 'CAGR'
total_sum = grouped['CAGRAcType'].sum()

# Calculate percentage
grouped['% CAGRAcType'] = (grouped['CAGRAcType'] / total_sum) * 100
# Sort by 'CAGRAcType' descending
grouped = grouped.sort_values(by='CAGRAcType', ascending=False)
grouped
```

Out[354]:

	Account Type	CAGRAcType	% CAGRAcType
3	Restaurant	8.550629	27.528865
1	Club	8.153919	26.251652
2	Hotel	7.537624	24.267482
0	Bar	6.818422	21.952001

In [355…

```
# Define colors
colors = ['#D21034', '#006A4E', '#4e545c', '#FFCE00', '#FFFFFF']

# Plotting the donut chart
plt.figure(figsize=(12, 7))
wedges, texts, autotexts = plt.pie(grouped['% CAGRAcType'],
                                   labels=grouped['Account Type'],
                                   autopct='%.1f%%',
                                   colors=colors,
                                   wedgeprops=dict(width=0.4))

# Customize the appearance
plt.setp(autotexts, size=10, weight="bold", color="black")
plt.setp(texts, size=12)

plt.title('Account Type CAGR 2017-2021')
plt.show()
```



In [355…

In [356…
```
# df1['Stateright'] = df1['State'].str[-2:]
df['Zipcode'] = df['Account Address'].str[-5:]
```

In [357…
```
# Remove both the first few & last few characters from a column of texts
```

In [358…
```
df['State'] = df['Account Address'].str[:-5].str[-3:]
```

In [359…
```
df['Account Address'].str[:-8]
```

Out[359]:

0	2131 Patterson Road, Brooklyn
1	3685 Morningview Lane, New York
2	2285 Ladybug Drive, New York
3	2930 Southern Street, New York
4	2807 Geraldine Lane, New York
5	7778 Cherry Road, Bronx
6	48 Winchester Avenue, New York
7	8735 Squaw Creek Drive, Brooklyn
8	267 Third Road, New York
9	102 Coffee Court, Bronx
10	44 W. Pheasant Street, Brooklyn
11	7488 N. Marconi Ave, Brooklyn
12	9575 Shipley Court, Brooklyn
13	8156 Lake View Street, New York,
14	44 Madison Dr, New York
15	9848 Linden St, New York
16	805 South Pilgrim Court, Brooklyn
17	9132 Redwood Rd, Bronx
18	3 Warren Drive, New York
19	402 Bridgeton Lane, Bronx
20	6 E. Nichols Ave, New York
21	323 North Edgewood St, Bronx
22	484 Thorne St, New York
23	861 Gonzales Lane, Bronx
24	267 Randall Mill Dr, New York
25	12 Lees Creek St, Brooklyn
26	240 W. Manhattan St, Bronx
27	62 Lower River Road, Staten Island,
28	48 S. Brandywine St, New York
29	5 Tallwood St, Brooklyn
30	77 Stillwater St, Brooklyn
31	7061 Bishop St, Yonkers
32	7223 Cedarwood Ave, Brooklyn
33	62 Lafayette Ave, Bronx
34	7839 Elm St, Staten Island
35	429 Stonybrook Dr, Brooklyn
36	640 Beechwood Dr, Bronx
37	9453 N. Wagon Lane, Brooklyn
38	81 San Carlos Road, Bronx
39	596 Coffee St, Bronx
40	92 Princess St, New York
41	9151 River St, Brooklyn
42	424 Hall Ave, New York
43	81 Crescent St, Brooklyn
44	7217 Birch Hill Dr, New York

```
45            7184 Center Court, Brooklyn
46                  815 2nd St, New York
47             9875 Franklin Rd, Brooklyn
48                 601 Bank Ave, Brooklyn
49                    21 Yukon St, Bronx
50           18 N. Woodland Ave, New York
51              65 Lower River Ave, Bronx
52             8680 Alderwood St, New York
53             8388 Gonzales St, Brooklyn
54               9760 Taylor Dr, Brooklyn
55             419 E. Henry Ave, New York
56                  8083 8th St, Brooklyn
57             2 Rock Maple Ave, New York
58        9577 Nicolls Ave, Staten Island
59             174 Del Monte St, Brooklyn
Name: Account Address, dtype: object
```

In [360…
```
df[['Street','string,','City']] = df['Account Address'].str[:-8].str.partition(',')
```

In [361…
```
df1 = df
```

In [361…
```

```

In [362…
```
df1.drop(['Account Name','Account Address', 'string,'], axis = 1, inplace = True)
```

In [363…
```
df1.head()
```

Out[363]:

	Decision Maker	Phone Number	Account Type	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	Zipcode	Sta
0	Dorothy Rizzo	(880) 283-6803	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1982	5388	7063	7208	9093	11201	
1	Lawson Moore	(711) 426-7350	Bar	Yes	Yes	Yes	No	Yes	Yes	Yes	2786	3804	4121	6210	6909	10013	
2	Vin Hudson	(952) 952-5573	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1209	1534	1634	4302	9768	10013	
3	Susana Huels	(491) 505-6064	Bar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	906	1251	2897	4499	9428	10005	
4	Shanna Hettinger	(412) 570-0596	Bar	Yes	Yes	No	Yes	Yes	Yes	Yes	1421	1893	2722	4410	5873	10004	

Total sales by Account Type and Year

In [364…
```
df1groupby = df1.groupby('Account Type')[[2017, 2018, 2019, 2020, 2021]].sum()
```

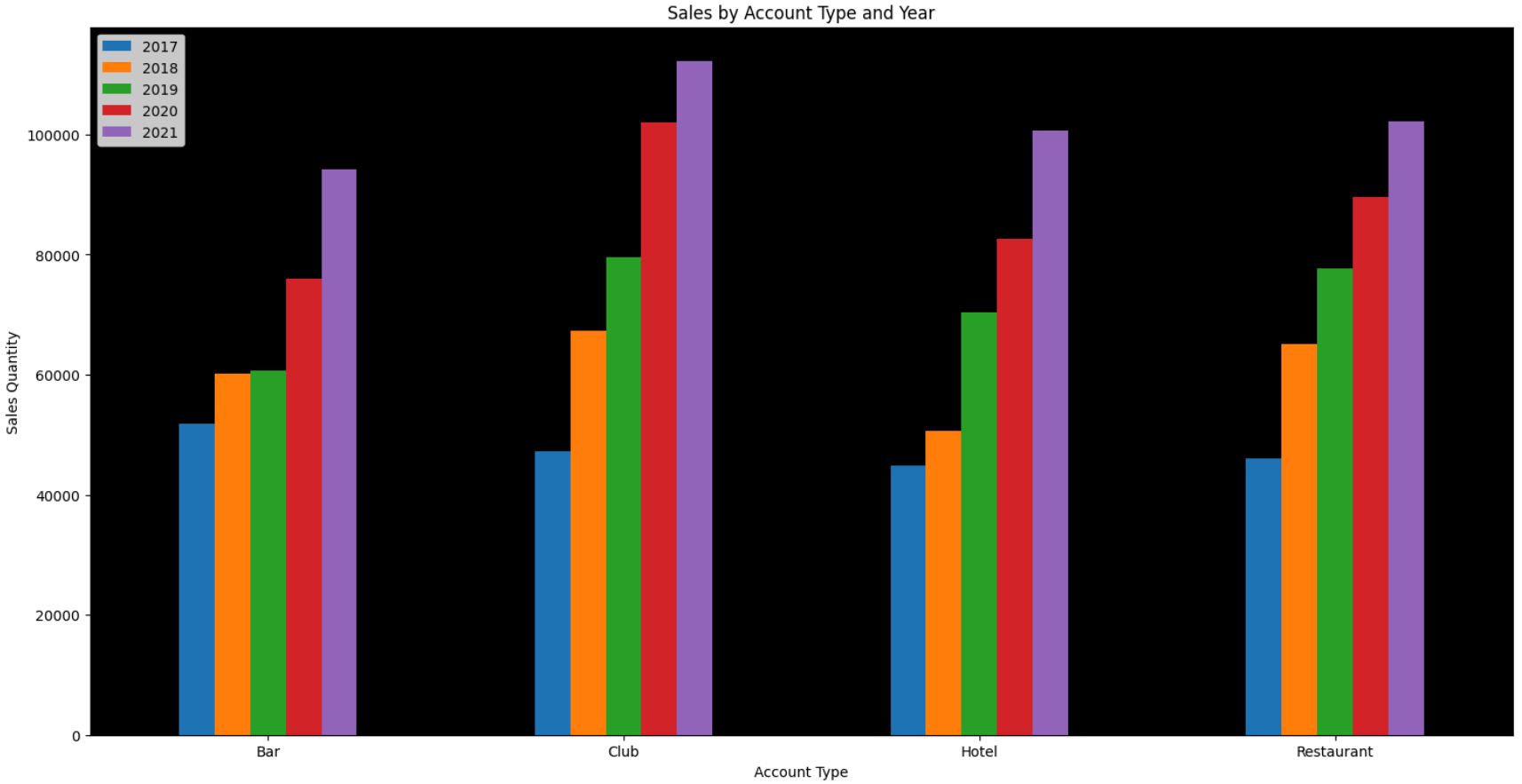
In [365…
```
df1groupby
```

Out[365]:

Account Type	2017	2018	2019	2020	2021
Bar	51804	60121	60760	75991	94147
Club	47259	67275	79646	102065	112270
Hotel	44888	50567	70312	82583	100592
Restaurant	46025	65032	77731	89595	102185

In [366…

```
# Group by 'Account Type' and sum across years
df1groupby = df1.groupby('Account Type')[[2017, 2018, 2019, 2020, 2021]].sum()

# Plotting the bar chart
ax = df1groupby.plot.bar(figsize=(18, 9))
ax.set_facecolor("black")
plt.title('Sales by Account Type and Year')
plt.xlabel('Account Type')
plt.ylabel('Sales Quantity')
plt.xticks(rotation=0)
plt.show()
```



In [367…

```
df1groupbyT = df1groupby.T
df1groupbyT
```

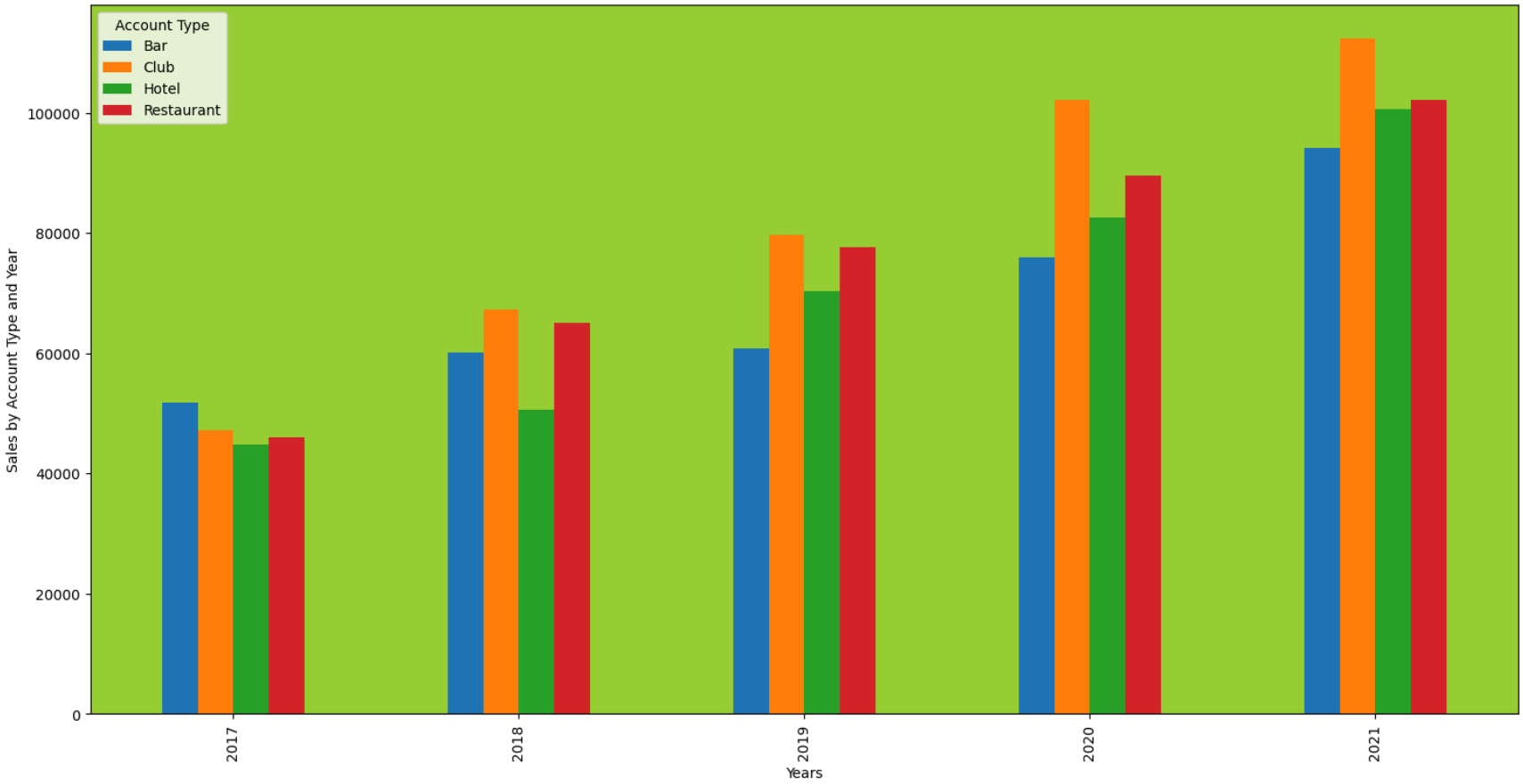
Out[367]:

Account Type	Bar	Club	Hotel	Restaurant
2017	51804	47259	44888	46025
2018	60121	67275	50567	65032
2019	60760	79646	70312	77731
2020	75991	102065	82583	89595
2021	94147	112270	100592	102185

Total Sales by Account Type and Year

In [368…

```
ax = df1groupbyT.plot.bar(figsize=(18,9))
ax.set_facecolor("yellowgreen")
plt.xlabel('Years')
plt.ylabel('Sales by Account Type and Year')
plt.show()
```



In [369…
```
ASSMM = df1.copy()
```

In [370…
```
# Selection of  the specified columns
selected_columns = ['Regular', 'Sugar Free', 'Yellow Edition', 'Cooler?', 'Digital screen?', 'Menu inclusion?', 'Poster
df_ASSMM_selected = ASSMM[selected_columns]
# OutPut
df_ASSMM_selected
```

Out[370]:

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021
0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1982	5388	7063	7208	9093
1	Yes	Yes	Yes	No	Yes	Yes	Yes	2786	3804	4121	6210	6909
2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1209	1534	1634	4302	9768
3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	906	1251	2897	4499	9428
4	Yes	Yes	No	Yes	Yes	Yes	Yes	1421	1893	2722	4410	5873
5	Yes	Yes	Yes	No	Yes	Yes	No	2341	6105	7777	7891	8758
6	Yes	No	No	No	No	Yes	No	9252	8499	991	448	211
7	Yes	Yes	Yes	Yes	No	Yes	No	1581	4799	6582	9024	9759
8	Yes	No	No	No	No	Yes	No	9766	8049	5556	5202	2373
9	Yes	Yes	No	Yes	No	Yes	No	1530	1620	2027	4881	6002
10	Yes	No	No	No	No	No	No	7555	6551	5188	3436	2359
11	Yes	No	No	No	No	No	No	1532	2678	4068	4278	5382
12	Yes	Yes	Yes	Yes	Yes	Yes	Yes	24	1797	3548	3668	8592
13	Yes	Yes	Yes	Yes	Yes	Yes	Yes	861	1314	1810	6510	9271
14	Yes	Yes	No	No	No	No	No	9058	4839	4776	4024	369
15	Yes	Yes	No	No	No	No	No	3501	7079	7438	7443	9225
16	Yes	Yes	No	No	No	No	No	3916	4218	5072	5201	7588
17	Yes	Yes	No	Yes	No	Yes	No	700	5721	6247	8495	9236
18	Yes	Yes	No	No	No	No	No	9773	9179	8390	8256	3815
19	Yes	Yes	No	Yes	No	Yes	No	73	3485	4592	5143	8100
20	Yes	Yes	No	Yes	No	Yes	No	238	1235	1822	7074	8207
21	Yes	Yes	No	Yes	No	Yes	No	1368	3447	4535	5476	9983
22	Yes	No	No	No	Yes	No	No	8331	7667	5952	1998	375
23	Yes	Yes	No	Yes	Yes	Yes	No	1779	2124	2844	6877	9570
24	Yes	Yes	No	Yes	Yes	Yes	No	570	1322	7279	8443	9571

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021
25	Yes	No	No	No	Yes	No	No	6156	6110	5791	1759	969
26	Yes	Yes	No	Yes	Yes	Yes	No	209	621	3098	7118	8433
27	Yes	Yes	No	No	No	No	No	6309	6227	5123	4968	3857
28	Yes	Yes	No	Yes	No	Yes	No	712	4182	6087	7494	8599
29	Yes	Yes	No	No	No	No	No	2390	2415	3461	3850	4657
30	Yes	Yes	Yes	No	No	Yes	No	2519	3938	5190	8203	8780
31	Yes	Yes	Yes	Yes	Yes	Yes	No	138	286	6750	8254	8656
32	Yes	Yes	Yes	No	No	Yes	Yes	8873	8484	7883	7499	6592
33	Yes	Yes	Yes	No	No	Yes	Yes	3297	4866	4928	8451	9585
34	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1092	3140	4123	4366	9482
35	Yes	Yes	Yes	No	No	Yes	Yes	2541	3794	3984	8803	9338
36	Yes	Yes	Yes	Yes	Yes	Yes	Yes	742	3751	4423	8733	9909
37	Yes	No	No	No	No	Yes	Yes	7703	6957	3898	1857	1512
38	Yes	Yes	Yes	Yes	Yes	Yes	Yes	488	5535	5775	7661	9206
39	Yes	Yes	Yes	Yes	Yes	Yes	Yes	376	889	4373	6803	7578
40	Yes	No	No	No	No	Yes	Yes	7840	5804	4259	4243	907
41	Yes	Yes	Yes	Yes	Yes	Yes	Yes	1038	3615	3712	5819	9589
42	Yes	Yes	No	No	No	No	No	8891	5952	5914	5405	4031
43	Yes	Yes	Yes	Yes	No	No	No	1290	4033	6956	7929	8834
44	Yes	Yes	Yes	Yes	Yes	No	No	431	6231	7478	8039	8271
45	Yes	No	No	No	No	Yes	No	8156	1245	791	338	44
46	Yes	Yes	Yes	Yes	No	Yes	No	299	657	6238	8922	9081
47	Yes	Yes	Yes	No	No	Yes	No	1323	4963	6292	6728	8202
48	Yes	No	No	No	No	Yes	No	8466	4079	2797	2245	1696
49	Yes	Yes	Yes	No	No	Yes	No	870	2428	7386	8835	9766

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021
50	Yes	Yes	Yes	No	No	Yes	No	1497	1768	2804	5718	9822
51	Yes	Yes	Yes	No	No	Yes	No	1082	3353	6351	8550	9272
52	Yes	Yes	No	No	No	Yes	No	9791	9610	7534	5080	4936
53	Yes	Yes	Yes	No	No	Yes	No	1357	4189	5407	6233	9681
54	Yes	No	No	No	No	Yes	No	576	2628	3612	5066	5156
55	Yes	Yes	Yes	Yes	No	No	No	128	416	747	1028	6357
56	Yes	No	No	No	No	No	No	8034	6541	3311	3254	2687
57	Yes	Yes	Yes	No	No	No	No	1263	2517	8042	8222	9686
58	Yes	Yes	Yes	No	No	No	No	1032	3919	4466	5568	6476
59	Yes	Yes	Yes	No	No	No	No	1014	2254	4534	6796	7730

In [371…

```
# Replace values in the specified columns
df_ASSMM_selected['Regular'] = df_ASSMM_selected['Regular'].apply(lambda x: 'Regular' if x == 'Yes' else '')
df_ASSMM_selected['Sugar Free'] = df_ASSMM_selected['Sugar Free'].apply(lambda x: 'SugarFree' if x == 'Yes' else '')
df_ASSMM_selected['Yellow Edition'] = df_ASSMM_selected['Yellow Edition'].apply(lambda x: 'YellowEdition' if x == 'Yes'
df_ASSMM_selected['Cooler?'] = df_ASSMM_selected['Cooler?'].apply(lambda x: 'Cooler' if x == 'Yes' else '')
df_ASSMM_selected['Digital screen?'] = df_ASSMM_selected['Digital screen?'].apply(lambda x: 'Digitalscreen' if x == 'Ye
df_ASSMM_selected['Menu inclusion?'] = df_ASSMM_selected['Menu inclusion?'].apply(lambda x: 'MenuInclusion' if x == 'Ye
df_ASSMM_selected['Posters?'] = df_ASSMM_selected['Posters?'].apply(lambda x: 'Posters' if x == 'Yes' else '')

# Sum the values of the specified columns to create a new column 'RevenuFor5Years'
df_ASSMM_selected['RevenuFor5Years'] =df_ASSMM_selected[[2017, 2018, 2019, 2020, 2021]].sum(axis=1)

# Display the modified dataframe
df_ASSMM_selected
```

```
<ipython-input-371-18e9531cf29b>: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
  df_ASSMM_selected['Regular'] = df_ASSMM_selected['Regular'].apply(lambda x: 'Regular' if x == 'Yes' else '')
<ipython-input-371-18e9531cf29b>:3: 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
  df_ASSMM_selected['Sugar Free'] = df_ASSMM_selected['Sugar Free'].apply(lambda x: 'SugarFree' if x == 'Yes' else '')
<ipython-input-371-18e9531cf29b>:4: 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
  df_ASSMM_selected['Yellow Edition'] = df_ASSMM_selected['Yellow Edition'].apply(lambda x: 'YellowEdition' if x == 'Ye
s' else '')
<ipython-input-371-18e9531cf29b>:5: 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
  df_ASSMM_selected['Cooler?'] = df_ASSMM_selected['Cooler?'].apply(lambda x: 'Cooler' if x == 'Yes' else '')
<ipython-input-371-18e9531cf29b>:6: 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
  df_ASSMM_selected['Digital screen?'] = df_ASSMM_selected['Digital screen?'].apply(lambda x: 'Digitalscreen' if x ==
'Yes' else '')
<ipython-input-371-18e9531cf29b>: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
  df_ASSMM_selected['Menu inclusion?'] = df_ASSMM_selected['Menu inclusion?'].apply(lambda x: 'MenuInclusion' if x ==
'Yes' else '')
```

```
<ipython-input-371-18e9531cf29b>:8: 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
  df_ASSMM_selected['Posters?'] = df_ASSMM_selected['Posters?'].apply(lambda x: 'Posters' if x == 'Yes' else '')
<ipython-input-371-18e9531cf29b>:11: 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
  df_ASSMM_selected['RevenuFor5Years'] =df_ASSMM_selected[[2017, 2018, 2019, 2020, 2021]].sum(axis=1)
```

Out[371]:

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years
0	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1982	5388	7063	7208	9093	30734
1	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion	Posters	2786	3804	4121	6210	6909	23830
2	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1209	1534	1634	4302	9768	18447
3	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	906	1251	2897	4499	9428	18981
4	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion	Posters	1421	1893	2722	4410	5873	16319
5	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion		2341	6105	7777	7891	8758	32872
6	Regular					MenuInclusion		9252	8499	991	448	211	19401
7	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		1581	4799	6582	9024	9759	31745
8	Regular					MenuInclusion		9766	8049	5556	5202	2373	30946
9	Regular	SugarFree		Cooler		MenuInclusion		1530	1620	2027	4881	6002	16060
10	Regular							7555	6551	5188	3436	2359	25089
11	Regular							1532	2678	4068	4278	5382	17938
12	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	24	1797	3548	3668	8592	17629
13	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	861	1314	1810	6510	9271	19766
14	Regular	SugarFree						9058	4839	4776	4024	369	23066
15	Regular	SugarFree						3501	7079	7438	7443	9225	34686
16	Regular	SugarFree						3916	4218	5072	5201	7588	25995
17	Regular	SugarFree		Cooler		MenuInclusion		700	5721	6247	8495	9236	30399
18	Regular	SugarFree						9773	9179	8390	8256	3815	39413
19	Regular	SugarFree		Cooler		MenuInclusion		73	3485	4592	5143	8100	21393
20	Regular	SugarFree		Cooler		MenuInclusion		238	1235	1822	7074	8207	18576
21	Regular	SugarFree		Cooler		MenuInclusion		1368	3447	4535	5476	9983	24809
22	Regular				Digitalscreen			8331	7667	5952	1998	375	24323
23	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		1779	2124	2844	6877	9570	23194

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years
24	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		570	1322	7279	8443	9571	27185
25	Regular				Digitalscreen			6156	6110	5791	1759	969	20785
26	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		209	621	3098	7118	8433	19479
27	Regular	SugarFree						6309	6227	5123	4968	3857	26484
28	Regular	SugarFree		Cooler		MenuInclusion		712	4182	6087	7494	8599	27074
29	Regular	SugarFree						2390	2415	3461	3850	4657	16773
30	Regular	SugarFree	YellowEdition			MenuInclusion		2519	3938	5190	8203	8780	28630
31	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion		138	286	6750	8254	8656	24084
32	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	8873	8484	7883	7499	6592	39331
33	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	3297	4866	4928	8451	9585	31127
34	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1092	3140	4123	4366	9482	22203
35	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	2541	3794	3984	8803	9338	28460
36	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	742	3751	4423	8733	9909	27558
37	Regular					MenuInclusion	Posters	7703	6957	3898	1857	1512	21927
38	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	488	5535	5775	7661	9206	28665
39	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	376	889	4373	6803	7578	20019
40	Regular					MenuInclusion	Posters	7840	5804	4259	4243	907	23053
41	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1038	3615	3712	5819	9589	23773
42	Regular	SugarFree						8891	5952	5914	5405	4031	30193
43	Regular	SugarFree	YellowEdition	Cooler				1290	4033	6956	7929	8834	29042
44	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen			431	6231	7478	8039	8271	30450
45	Regular					MenuInclusion		8156	1245	791	338	44	10574
46	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		299	657	6238	8922	9081	25197
47	Regular	SugarFree	YellowEdition			MenuInclusion		1323	4963	6292	6728	8202	27508

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years
48	Regular					MenuInclusion		8466	4079	2797	2245	1696	19283
49	Regular	SugarFree	YellowEdition			MenuInclusion		870	2428	7386	8835	9766	29285
50	Regular	SugarFree	YellowEdition			MenuInclusion		1497	1768	2804	5718	9822	21609
51	Regular	SugarFree	YellowEdition			MenuInclusion		1082	3353	6351	8550	9272	28608
52	Regular	SugarFree				MenuInclusion		9791	9610	7534	5080	4936	36951
53	Regular	SugarFree	YellowEdition			MenuInclusion		1357	4189	5407	6233	9681	26867
54	Regular					MenuInclusion		576	2628	3612	5066	5156	17038
55	Regular	SugarFree	YellowEdition	Cooler				128	416	747	1028	6357	8676
56	Regular							8034	6541	3311	3254	2687	23827
57	Regular	SugarFree	YellowEdition					1263	2517	8042	8222	9686	29730
58	Regular	SugarFree	YellowEdition					1032	3919	4466	5568	6476	21461
59	Regular	SugarFree	YellowEdition					1014	2254	4534	6796	7730	22328

In [372…

```
# Concatenate columns to create 'Assortment' and 'Multimedia' columns
df_ASSMM_selected['Assortment'] = df_ASSMM_selected[['Regular', 'Sugar Free', 'Yellow Edition']].apply(lambda x: '_'.jo
df_ASSMM_selected['Multimedia'] = df_ASSMM_selected[['Cooler?', 'Digital screen?', 'Menu inclusion?']].apply(lambda x:
# OutPut
df_ASSMM_selected
```

```
<ipython-input-372-ae73f934efea>: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
  df_ASSMM_selected['Assortment'] = df_ASSMM_selected[['Regular', 'Sugar Free', 'Yellow Edition']].apply(lambda x: '_'.
join(filter(None, x)), axis=1)
<ipython-input-372-ae73f934efea>:3: 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
  df_ASSMM_selected['Multimedia'] = df_ASSMM_selected[['Cooler?', 'Digital screen?', 'Menu inclusion?']].apply(lambda
x: '_'.join(filter(None, x)), axis=1)
```

Out[372]:

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
0	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1982	5388	7063	7208	9093	30734	Regula
1	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion	Posters	2786	3804	4121	6210	6909	23830	Regula
2	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1209	1534	1634	4302	9768	18447	Regula
3	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	906	1251	2897	4499	9428	18981	Regula
4	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion	Posters	1421	1893	2722	4410	5873	16319	
5	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion		2341	6105	7777	7891	8758	32872	Regula
6	Regular					MenuInclusion		9252	8499	991	448	211	19401	
7	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		1581	4799	6582	9024	9759	31745	Regula
8	Regular					MenuInclusion		9766	8049	5556	5202	2373	30946	
9	Regular	SugarFree		Cooler		MenuInclusion		1530	1620	2027	4881	6002	16060	
10	Regular							7555	6551	5188	3436	2359	25089	
11	Regular							1532	2678	4068	4278	5382	17938	
12	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	24	1797	3548	3668	8592	17629	Regula
13	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	861	1314	1810	6510	9271	19766	Regula
14	Regular	SugarFree						9058	4839	4776	4024	369	23066	
15	Regular	SugarFree						3501	7079	7438	7443	9225	34686	
16	Regular	SugarFree						3916	4218	5072	5201	7588	25995	
17	Regular	SugarFree		Cooler		MenuInclusion		700	5721	6247	8495	9236	30399	
18	Regular	SugarFree						9773	9179	8390	8256	3815	39413	
19	Regular	SugarFree		Cooler		MenuInclusion		73	3485	4592	5143	8100	21393	
20	Regular	SugarFree		Cooler		MenuInclusion		238	1235	1822	7074	8207	18576	
21	Regular	SugarFree		Cooler		MenuInclusion		1368	3447	4535	5476	9983	24809	
22	Regular				Digitalscreen			8331	7667	5952	1998	375	24323	
23	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		1779	2124	2844	6877	9570	23194	

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
24	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		570	1322	7279	8443	9571	27185	
25	Regular				Digitalscreen			6156	6110	5791	1759	969	20785	
26	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		209	621	3098	7118	8433	19479	
27	Regular	SugarFree						6309	6227	5123	4968	3857	26484	
28	Regular	SugarFree		Cooler		MenuInclusion		712	4182	6087	7494	8599	27074	
29	Regular	SugarFree						2390	2415	3461	3850	4657	16773	
30	Regular	SugarFree	YellowEdition			MenuInclusion		2519	3938	5190	8203	8780	28630	Regula
31	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion		138	286	6750	8254	8656	24084	Regula
32	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	8873	8484	7883	7499	6592	39331	Regula
33	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	3297	4866	4928	8451	9585	31127	Regula
34	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1092	3140	4123	4366	9482	22203	Regula
35	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	2541	3794	3984	8803	9338	28460	Regula
36	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	742	3751	4423	8733	9909	27558	Regula
37	Regular					MenuInclusion	Posters	7703	6957	3898	1857	1512	21927	
38	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	488	5535	5775	7661	9206	28665	Regula
39	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	376	889	4373	6803	7578	20019	Regula
40	Regular					MenuInclusion	Posters	7840	5804	4259	4243	907	23053	
41	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1038	3615	3712	5819	9589	23773	Regula
42	Regular	SugarFree						8891	5952	5914	5405	4031	30193	
43	Regular	SugarFree	YellowEdition	Cooler				1290	4033	6956	7929	8834	29042	Regula
44	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen			431	6231	7478	8039	8271	30450	Regula
45	Regular					MenuInclusion		8156	1245	791	338	44	10574	
46	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		299	657	6238	8922	9081	25197	Regula
47	Regular	SugarFree	YellowEdition			MenuInclusion		1323	4963	6292	6728	8202	27508	Regula

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
48	Regular					MenuInclusion		8466	4079	2797	2245	1696	19283	
49	Regular	SugarFree	YellowEdition			MenuInclusion		870	2428	7386	8835	9766	29285	Regula
50	Regular	SugarFree	YellowEdition			MenuInclusion		1497	1768	2804	5718	9822	21609	Regula
51	Regular	SugarFree	YellowEdition			MenuInclusion		1082	3353	6351	8550	9272	28608	Regula
52	Regular	SugarFree				MenuInclusion		9791	9610	7534	5080	4936	36951	
53	Regular	SugarFree	YellowEdition			MenuInclusion		1357	4189	5407	6233	9681	26867	Regula
54	Regular					MenuInclusion		576	2628	3612	5066	5156	17038	
55	Regular	SugarFree	YellowEdition	Cooler				128	416	747	1028	6357	8676	Regula
56	Regular							8034	6541	3311	3254	2687	23827	
57	Regular	SugarFree	YellowEdition					1263	2517	8042	8222	9686	29730	Regula
58	Regular	SugarFree	YellowEdition					1032	3919	4466	5568	6476	21461	Regula
59	Regular	SugarFree	YellowEdition					1014	2254	4534	6796	7730	22328	Regula

In [373…
```
# Replace blank values in 'Multimedia' column with 'NoMultimedia'
df_ASSMM_selected['Multimedia'] = df_ASSMM_selected['Multimedia'].replace('', 'NoMultimedia')
```

```
<ipython-input-373-25dfd61c9273>: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
  df_ASSMM_selected['Multimedia'] = df_ASSMM_selected['Multimedia'].replace('', 'NoMultimedia')
```

In [374…
```
df_ASSMM_selected
```

Out[374]:

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
0	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1982	5388	7063	7208	9093	30734	Regula
1	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion	Posters	2786	3804	4121	6210	6909	23830	Regula
2	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1209	1534	1634	4302	9768	18447	Regula
3	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	906	1251	2897	4499	9428	18981	Regula
4	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion	Posters	1421	1893	2722	4410	5873	16319	
5	Regular	SugarFree	YellowEdition		Digitalscreen	MenuInclusion		2341	6105	7777	7891	8758	32872	Regula
6	Regular					MenuInclusion		9252	8499	991	448	211	19401	
7	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		1581	4799	6582	9024	9759	31745	Regula
8	Regular					MenuInclusion		9766	8049	5556	5202	2373	30946	
9	Regular	SugarFree		Cooler		MenuInclusion		1530	1620	2027	4881	6002	16060	
10	Regular							7555	6551	5188	3436	2359	25089	
11	Regular							1532	2678	4068	4278	5382	17938	
12	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	24	1797	3548	3668	8592	17629	Regula
13	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	861	1314	1810	6510	9271	19766	Regula
14	Regular	SugarFree						9058	4839	4776	4024	369	23066	
15	Regular	SugarFree						3501	7079	7438	7443	9225	34686	
16	Regular	SugarFree						3916	4218	5072	5201	7588	25995	
17	Regular	SugarFree		Cooler		MenuInclusion		700	5721	6247	8495	9236	30399	
18	Regular	SugarFree						9773	9179	8390	8256	3815	39413	
19	Regular	SugarFree		Cooler		MenuInclusion		73	3485	4592	5143	8100	21393	
20	Regular	SugarFree		Cooler		MenuInclusion		238	1235	1822	7074	8207	18576	
21	Regular	SugarFree		Cooler		MenuInclusion		1368	3447	4535	5476	9983	24809	
22	Regular				Digitalscreen			8331	7667	5952	1998	375	24323	
23	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		1779	2124	2844	6877	9570	23194	

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
24	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		570	1322	7279	8443	9571	27185	
25	Regular				Digitalscreen			6156	6110	5791	1759	969	20785	
26	Regular	SugarFree		Cooler	Digitalscreen	MenuInclusion		209	621	3098	7118	8433	19479	
27	Regular	SugarFree						6309	6227	5123	4968	3857	26484	
28	Regular	SugarFree		Cooler		MenuInclusion		712	4182	6087	7494	8599	27074	
29	Regular	SugarFree						2390	2415	3461	3850	4657	16773	
30	Regular	SugarFree	YellowEdition			MenuInclusion		2519	3938	5190	8203	8780	28630	Regula
31	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion		138	286	6750	8254	8656	24084	Regula
32	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	8873	8484	7883	7499	6592	39331	Regula
33	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	3297	4866	4928	8451	9585	31127	Regula
34	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1092	3140	4123	4366	9482	22203	Regula
35	Regular	SugarFree	YellowEdition			MenuInclusion	Posters	2541	3794	3984	8803	9338	28460	Regula
36	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	742	3751	4423	8733	9909	27558	Regula
37	Regular					MenuInclusion	Posters	7703	6957	3898	1857	1512	21927	
38	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	488	5535	5775	7661	9206	28665	Regula
39	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	376	889	4373	6803	7578	20019	Regula
40	Regular					MenuInclusion	Posters	7840	5804	4259	4243	907	23053	
41	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen	MenuInclusion	Posters	1038	3615	3712	5819	9589	23773	Regula
42	Regular	SugarFree						8891	5952	5914	5405	4031	30193	
43	Regular	SugarFree	YellowEdition	Cooler				1290	4033	6956	7929	8834	29042	Regula
44	Regular	SugarFree	YellowEdition	Cooler	Digitalscreen			431	6231	7478	8039	8271	30450	Regula
45	Regular					MenuInclusion		8156	1245	791	338	44	10574	
46	Regular	SugarFree	YellowEdition	Cooler		MenuInclusion		299	657	6238	8922	9081	25197	Regula
47	Regular	SugarFree	YellowEdition			MenuInclusion		1323	4963	6292	6728	8202	27508	Regula

	Regular	Sugar Free	Yellow Edition	Cooler?	Digital screen?	Menu inclusion?	Posters?	2017	2018	2019	2020	2021	RevenuFor5Years	
48	Regular					MenuInclusion		8466	4079	2797	2245	1696	19283	
49	Regular	SugarFree	YellowEdition			MenuInclusion		870	2428	7386	8835	9766	29285	Regula
50	Regular	SugarFree	YellowEdition			MenuInclusion		1497	1768	2804	5718	9822	21609	Regula
51	Regular	SugarFree	YellowEdition			MenuInclusion		1082	3353	6351	8550	9272	28608	Regula
52	Regular	SugarFree				MenuInclusion		9791	9610	7534	5080	4936	36951	
53	Regular	SugarFree	YellowEdition			MenuInclusion		1357	4189	5407	6233	9681	26867	Regula
54	Regular					MenuInclusion		576	2628	3612	5066	5156	17038	
55	Regular	SugarFree	YellowEdition	Cooler				128	416	747	1028	6357	8676	Regula
56	Regular							8034	6541	3311	3254	2687	23827	
57	Regular	SugarFree	YellowEdition					1263	2517	8042	8222	9686	29730	Regula
58	Regular	SugarFree	YellowEdition					1032	3919	4466	5568	6476	21461	Regula
59	Regular	SugarFree	YellowEdition					1014	2254	4534	6796	7730	22328	Regula

In [375…
```
# Create a pivot table
pivot_table1 = df_ASSMM_selected.pivot_table(values='RevenuFor5Years', index='Assortment', aggfunc='sum')
# Order by descending
pivot_table1 = pivot_table1.sort_values(by='RevenuFor5Years', ascending=False)
```

In [376…
```
pivot_table1
```

Out[376]:

Assortment	RevenuFor5Years
Regular_SugarFree_YellowEdition	768615
Regular_SugarFree	458049
Regular	254184

In [377…
```
pivot_table2 = df_ASSMM_selected.pivot_table(values='RevenuFor5Years', index='Multimedia', aggfunc='sum')
# Order by descending
```

```
pivot_table2 = pivot_table2.sort_values(by='RevenuFor5Years', ascending=False)
```

In [378…

```
pivot_table2
```

Out[378]:

Multimedia	RevenuFor5Years
MenuInclusion	440598
Cooler_Digitalscreen_MenuInclusion	338036
NoMultimedia	336983
Cooler_MenuInclusion	195253
Digitalscreen_MenuInclusion	56702
Digitalscreen	45108
Cooler	37718
Cooler_Digitalscreen	30450

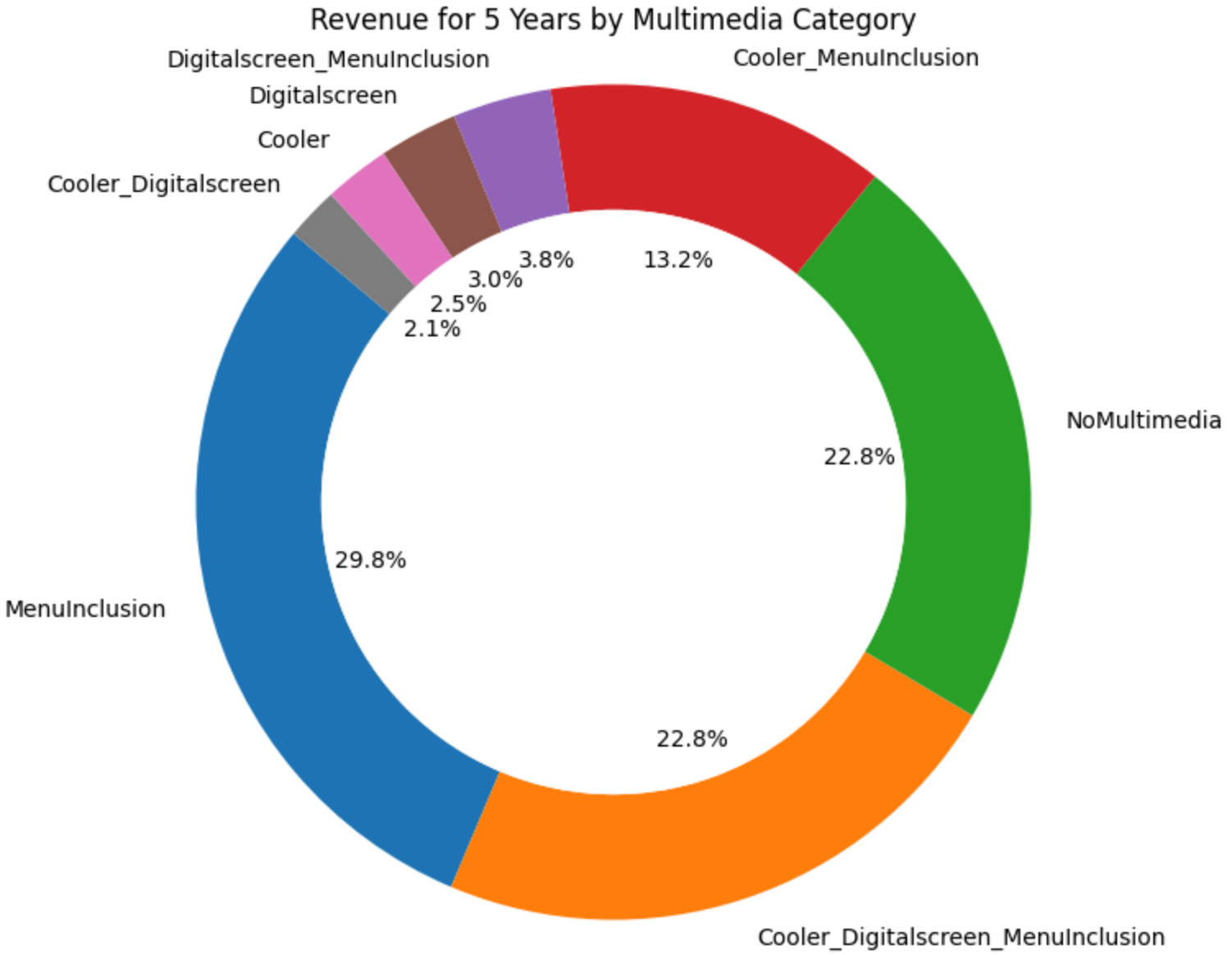
In [378…

In [379…

```
import pandas as pd
import matplotlib.pyplot as plt

# Data for the pie chart
data = {
    'Multimedia': [
        'MenuInclusion',
        'Cooler_Digitalscreen_MenuInclusion',
        'NoMultimedia',
        'Cooler_MenuInclusion',
        'Digitalscreen_MenuInclusion',
        'Digitalscreen',
        'Cooler',
        'Cooler_Digitalscreen'
    ],
    'RevenuFor5Years': [
        440598,
        338036,
        336983,
```

```
        195253,
        56702,
        45108,
        37718,
        30450
    ]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Check if the DataFrame is created correctly
print(df)

# Plotting the donut chart
fig, ax = plt.subplots(figsize=(10, 7))
wedges, texts, autotexts = ax.pie(df['RevenuFor5Years'], labels=df['Multimedia'], autopct='%1.1f%%', startangle=140, we

# Add a white circle at the center to create the donut effect
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that pie is drawn as a circle.
ax.axis('equal')

# Title
plt.title('Revenue for 5 Years by Multimedia Category')

# Center the figure
fig.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)

# Display the donut chart
plt.show()
```

```
                        Multimedia  RevenuFor5Years
0                     MenuInclusion           440598
1  Cooler_Digitalscreen_MenuInclusion           338036
2                      NoMultimedia           336983
3              Cooler_MenuInclusion           195253
4      Digitalscreen_MenuInclusion            56702
5                     Digitalscreen            45108
6                            Cooler            37718
7              Cooler_Digitalscreen            30450
```



In [380…

```
# 1. Count the occurrences of each category in the Assortment column
assortment_counts = df_ASSMM_selected['Assortment'].value_counts().reset_index()
assortment_counts.columns = ['Assortment', 'CountAssortment']

# 2. Group by the Assortment column and sum the RevenuFor5Years column
grouped_df = df_ASSMM_selected.groupby('Assortment')['RevenuFor5Years'].sum().reset_index()
```

```
# 3. Merge the count of occurrences with the grouped dataframe
merged_df = pd.merge(grouped_df, assortment_counts, on='Assortment')

# 4. Sort the results by RevenuFor5Years in descending order
sorted_merged_df = merged_df.sort_values(by='RevenuFor5Years', ascending=False).reset_index(drop=True)

sorted_merged_df
```

Out[380]:

	Assortment	RevenuFor5Years	CountAssortment
0	Regular_SugarFree_YellowEdition	768615	30
1	Regular_SugarFree	458049	18
2	Regular	254184	12

In [380…

Effect of assortment (product lines) presence on sales.

In [381…

```
# Calculate the mean (average) revenue per instance
sorted_merged_df['AverageRevenuePerInstance'] = sorted_merged_df['RevenuFor5Years'] / sorted_merged_df['CountAssortment
sorted_merged_df
```

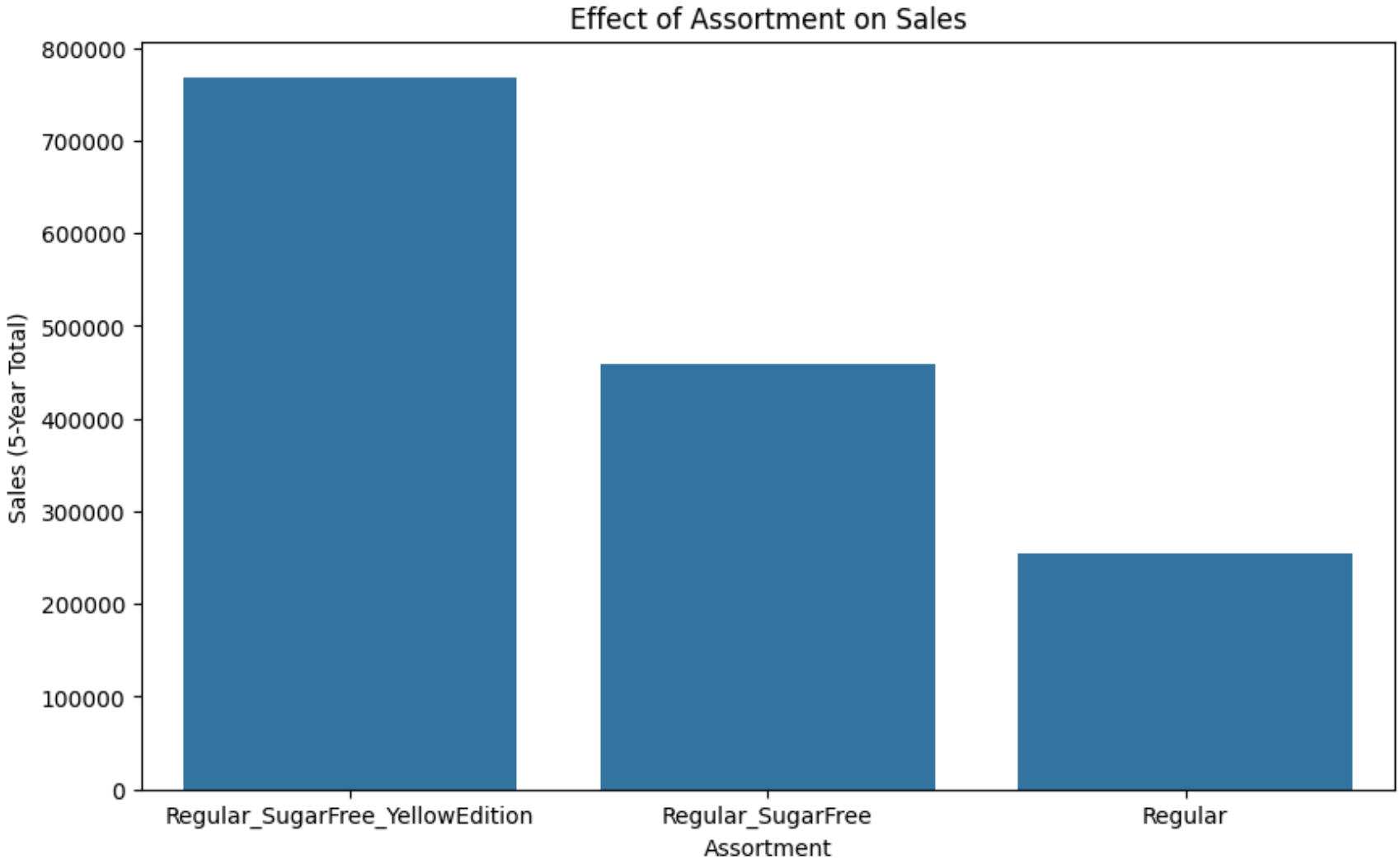
Out[381]:

	Assortment	RevenuFor5Years	CountAssortment	AverageRevenuePerInstance
0	Regular_SugarFree_YellowEdition	768615	30	25620.500000
1	Regular_SugarFree	458049	18	25447.166667
2	Regular	254184	12	21182.000000

In [382…

```
import matplotlib.pyplot as plt
import seaborn as sns
# Calculate the total sales for each assortment
assortment_sales = df_ASSMM_selected.groupby('Assortment')['RevenuFor5Years'].sum().reset_index()

# Sort the assortments by total sales in descending order
assortment_sales = assortment_sales.sort_values(by='RevenuFor5Years', ascending=False)
```

```
# Create a bar chart to visualize the results
plt.figure(figsize=(10, 6))
sns.barplot(x='Assortment', y='RevenuFor5Years', data=assortment_sales)
plt.title('Effect of Assortment on Sales')
plt.xlabel('Assortment')
plt.ylabel('Sales (5-Year Total)')
plt.show()
```



Conclusion:

The bar chart shows that the presence of certain assortments (product lines) has a significant impact on sales. For example, the assortment "Regular_SugarFree_YellowEdition" has the highest total sales, while the assortment "SugarFree" has the lowest total sales. This suggests that customers are more likely to purchase products from a wider range of product lines. Additionally, the presence of multiple product lines may allow for more targeted marketing and sales strategies, leading to increased sales.

In [382…

In [383…

```
# 1. Count the occurrences of each category in the Multimedia column
multimedia_counts = df_ASSMM_selected['Multimedia'].value_counts().reset_index()
multimedia_counts.columns = ['Multimedia','Count']

# 2. Group by the Multimedia column and sum the RevenuFor5Years column
grouped_df = df_ASSMM_selected.groupby('Multimedia')['RevenuFor5Years'].sum().reset_index()

# 3. Merge the count of occurrences with the grouped dataframe
merged_df = pd.merge(grouped_df, multimedia_counts, on='Multimedia')

# 4. Sort the results by RevenuFor5Years in descending order
sorted_merged_df2 = merged_df.sort_values(by='RevenuFor5Years', ascending=False).reset_index(drop=True)
sorted_merged_df2
```

Out[383]:

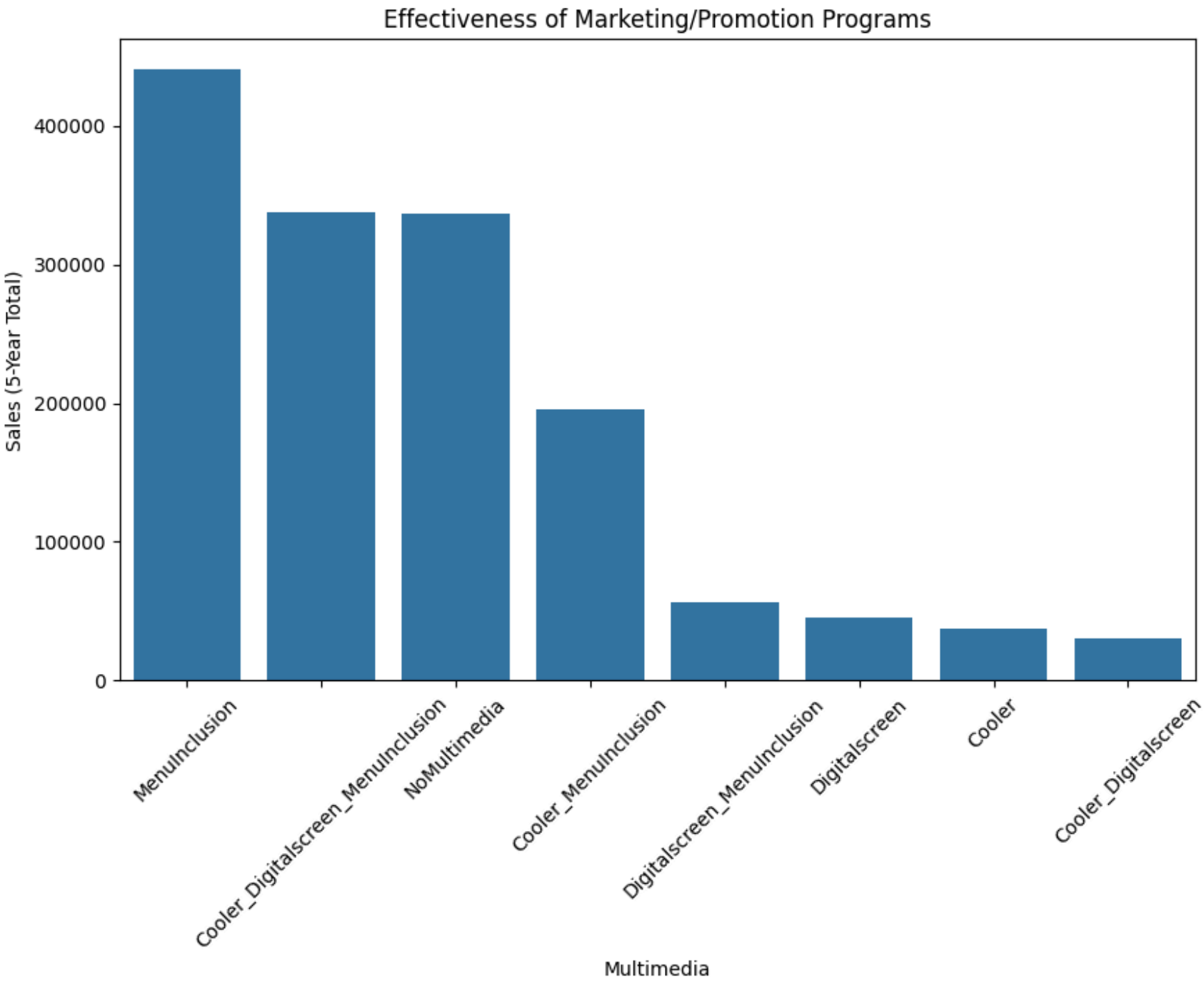
	Multimedia	RevenuFor5Years	Count
0	MenuInclusion	440598	17
1	Cooler_Digitalscreen_MenuInclusion	338036	15
2	NoMultimedia	336983	13
3	Cooler_MenuInclusion	195253	8
4	Digitalscreen_MenuInclusion	56702	2
5	Digitalscreen	45108	2
6	Cooler	37718	2
7	Cooler_Digitalscreen	30450	1

In [383…

Effectiveness of the different marketing/promotion programs.

In [384…

```
# Calculate the total sales for each multimedia category
multimedia_sales = df_ASSMM_selected.groupby('Multimedia')['RevenuFor5Years'].sum().reset_index()

# Sort the multimedia categories by total sales in descending order
multimedia_sales = multimedia_sales.sort_values(by='RevenuFor5Years', ascending=False)
# Create a bar chart to visualize the results
plt.figure(figsize=(10, 6))
sns.barplot(x='Multimedia', y='RevenuFor5Years', data=multimedia_sales)
plt.title('Effectiveness of Marketing/Promotion Programs')
plt.xlabel('Multimedia')
plt.ylabel('Sales (5-Year Total)')
plt.xticks(rotation=45)
plt.show()
```


Effectiveness of Marketing/Promotion Programs
Sales (5-Year Total)
400000
300000
200000
100000
0
MenuInclusion
Cooler_Digitalscreen_MenuInclusion
NoMultimedia
Cooler_MenuInclusion
Digitalscreen_MenuInclusion
Digitalscreen
Cooler
Cooler_Digitalscreen
Multimedia

In [385…
```
# Step 1: Calculate the average revenue per instance
sorted_merged_df2['AverageRevenuePerInstance'] = sorted_merged_df2['RevenuFor5Years'] / sorted_merged_df2['Count']
# Step 2: Sort the DataFrame by the Average Revenue Per Instance in descending order
sorted_merged_df2 = sorted_merged_df2.sort_values(by='AverageRevenuePerInstance', ascending=False).reset_index(drop=Tru
sorted_merged_df2
```

Out[385]:

	Multimedia	RevenuFor5Years	Count	AverageRevenuePerInstance
0	Cooler_Digitalscreen	30450	1	30450.000000
1	Digitalscreen_MenuInclusion	56702	2	28351.000000
2	NoMultimedia	336983	13	25921.769231
3	MenuInclusion	440598	17	25917.529412
4	Cooler_MenuInclusion	195253	8	24406.625000
5	Digitalscreen	45108	2	22554.000000
6	Cooler_Digitalscreen_MenuInclusion	338036	15	22535.733333
7	Cooler	37718	2	18859.000000

Conclusion:

The most effective program, based on average revenue per instance, is Digitalscreen_MenuInclusion. Surprisingly, NoMultimedia performs better than most multimedia programs, ranking second. MenuInclusion on its own is quite effective, ranking third. Cooler by itself is the least effective, suggesting that simply having a cooler without other multimedia components doesn't generate much revenue. By focusing on combinations that include Digitalscreen and MenuInclusion, businesses can potentially maximize their revenue from these marketing/promotion programs.

In [387…
```
from IPython import display
display.Image("/content/drive/MyDrive/Data_Engineering/bradley-wright-phillips_3470989.jpeg", width = 1700, height = 80
```

Out[387]: