In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

%matplotlib inline
```

Data cleaning

In [2]:
```
# reading data from excel sheet (uses xlrd python module)
superstore_data = pd.read_excel('./data/US Superstore data.xls', 'Orders', index_col=None)
superstore_data.head()
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	F
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	

5 rows × 21 columns

In [3]:
```
# converting to csv
superstore_data.to_csv('./data/superstore_data.csv', encoding='utf-8', index=False)
```

In [4]:
```
# reading content from csv file
superstore_data_df = pd.read_csv('./data/superstore_data.csv')
superstore_data_df.head()
```

Out[4]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies

5 rows × 21 columns

In [5]:
```
superstore_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

Mode making maximum profit

In [6]:
```
superstore_data_df.groupby(by=['Ship Mode'], sort=True).mean()
```

```
C:\Users\heman\AppData\Local\Temp\ipykernel_12768\2879827036.py:1: FutureWarning: The default value of numeri
c_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Eithe
r specify numeric_only or select only columns which should be valid for the function.
  superstore_data_df.groupby(by=['Ship Mode'], sort=True).mean()
```

Out[6]:

Ship Mode	Row ID	Postal Code	Sales	Quantity	Discount	Profit
First Class	4875.510403	54765.611834	228.497024	3.701560	0.164610	31.839948
Same Day	5128.909761	57536.082873	236.396179	3.609576	0.152394	29.266591
Second Class	4936.759383	55626.009254	236.089239	3.816452	0.138895	29.535545
Standard Class	5036.776977	54944.447051	227.583067	3.819873	0.160023	27.494770

In [7]:
```
mode_profits = superstore_data_df.groupby(by=['Ship Mode']).sum(numeric_only=True)
```

In [8]:
```
mode_profits
```

Out[8]:

Ship Mode	Row ID	Postal Code	Sales	Quantity	Discount	Profit
First Class	7498535	84229511	3.514284e+05	5693	253.17	48969.8399
Same Day	2784998	31242093	1.283631e+05	1960	82.75	15891.7589
Second Class	9601997	108192588	4.591936e+05	7423	270.15	57446.6354
Standard Class	30059485	327908460	1.358216e+06	22797	955.02	164088.7875

In [9]:
```
mode_profits.index
```

Out[9]:
```
Index(['First Class', 'Same Day', 'Second Class', 'Standard Class'], dtype='object', name='Ship Mode')
```

In [10]:
```
mode_profits.columns
```

Out[10]:
```
Index(['Row ID', 'Postal Code', 'Sales', 'Quantity', 'Discount', 'Profit'], dtype='object')
```

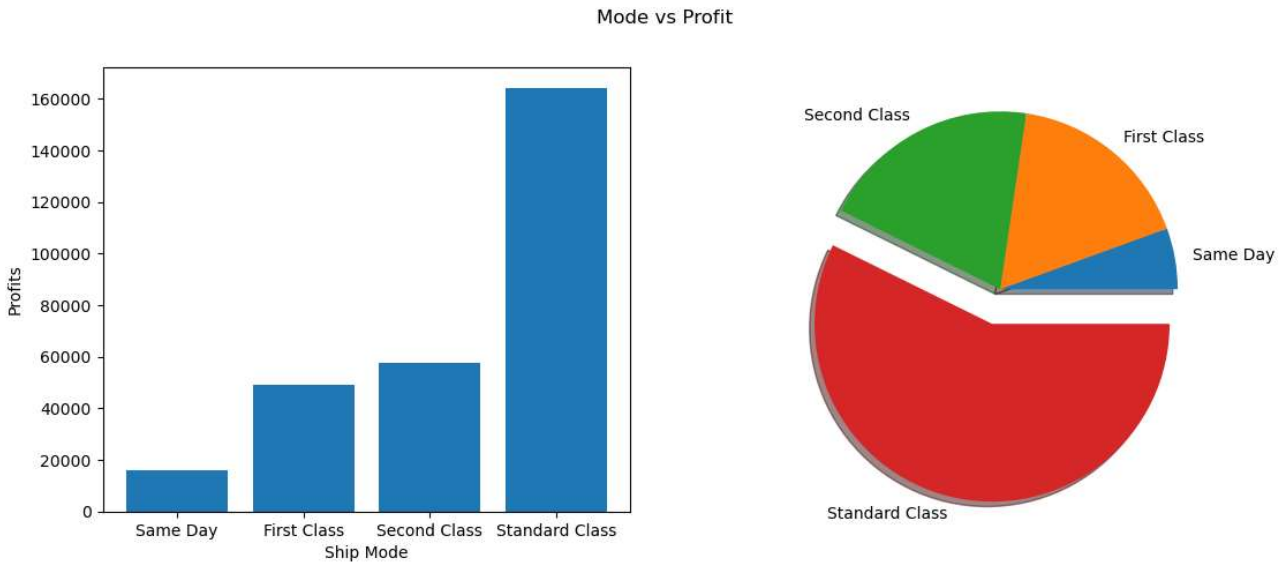
In [11]:
```
mode_profits = mode_profits["Profit"]
```

In [12]:
```
mode_profits = mode_profits.sort_values(ascending=True)
mode_profits
```

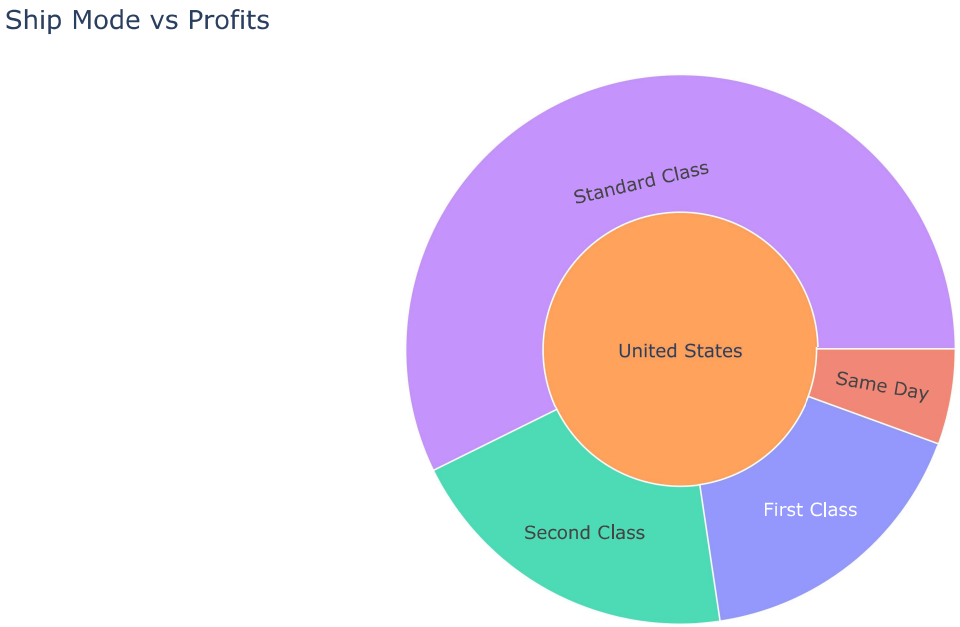
Out[12]:
```
Ship Mode
Same Day           15891.7589
First Class        48969.8399
Second Class       57446.6354
Standard Class    164088.7875
Name: Profit, dtype: float64
```

Graphical representation of the mode and profits

In [13]:
```
# setup plot
fig, (ax0, ax1) = plt.subplots(nrows = 1, ncols = 2, figsize=(13, 5))

# customizing figure
fig.suptitle("Mode vs Profit")

# adding data to ax0
ax0.bar(mode_profits.index, mode_profits.values)

# customizing ax0
ax0.set(xlabel = "Ship Mode", ylabel="Profits")

# adding data to ax1
ax1.pie(x = mode_profits.values, labels = mode_profits.index, explode=[0, 0, 0, 0.2], shadow=True, radius=1)

plt.show()
```



In [14]:
```
fig = px.sunburst(superstore_data_df, path=['Country', 'Ship Mode'],
                  values='Profit',color='Ship Mode',
                  hover_data =['Sales','Quantity','Profit'])
fig.update_layout(height=500,title_text='Ship Mode vs Profits')
fig.show()
```



Which product is performing good ?

In [15]:
```
superstore_data_df.head()
```

Out[15]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies

5 rows × 21 columns

Which segment buys the most products?

In [16]:
```
segment_analysis = superstore_data_df.groupby(by=['Segment']).sum(numeric_only=True)
```

In [17]:
```
segment_analysis
```

Out[17]:

Segment	Row ID	Postal Code	Sales	Quantity	Discount	Profit
Consumer	25581329	288878609	1.161401e+06	19521	820.91	134119.2092
Corporate	15504734	164536330	7.061464e+05	11608	477.85	91979.1340
Home Office	8858952	98157713	4.296531e+05	6744	262.33	60298.6785

In [18]:
```
segment_analysis.drop(["Row ID", "Postal Code", "Discount"], axis = 1, inplace=True)
```

In [19]:
```
segment_analysis = segment_analysis.sort_values(by="Profit", ascending=True)
segment_analysis
```

Out[19]:

Segment	Sales	Quantity	Profit
Home Office	4.296531e+05	6744	60298.6785
Corporate	7.061464e+05	11608	91979.1340
Consumer	1.161401e+06	19521	134119.2092

Graphical representation of Segment and sales

In [20]:
```
# categories of the segment
categories = segment_analysis.index
categories
```

Out[20]:
```
Index(['Home Office', 'Corporate', 'Consumer'], dtype='object', name='Segment')
```

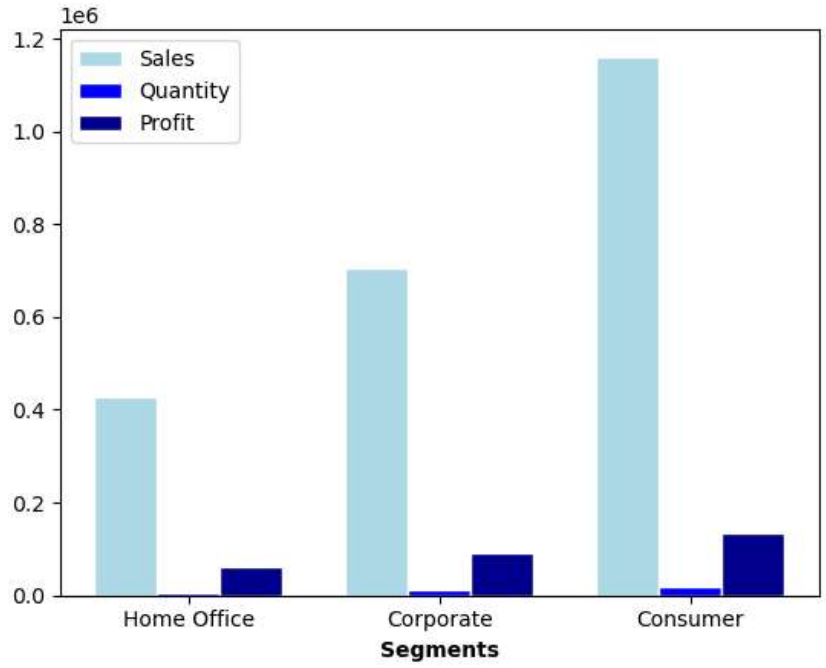
In [21]:
```
# getting individual data of the categories
category_sales = [ float(x) for x in segment_analysis["Sales"]]
category_quantity = segment_analysis["Quantity"]
category_profit = segment_analysis["Profit"]

category_sales
```

Out[21]:
```
[429653.1485, 706146.3668, 1161401.345]
```

In [22]:
```
## customizing bars

# number of categories
num_categories = len(categories)

# bar width
bar_width = 0.25

# setting position of bars on x-axis
bar1 = np.arange(num_categories)
bar2 = [x + bar_width for x in bar1]
bar3 = [x + bar_width for x in bar2]

print(bar1, bar2, bar3)
```

```
[0 1 2] [0.25, 1.25, 2.25] [0.5, 1.5, 2.5]
```

In [23]:
```
plt.bar(bar1, category_sales, color="#ADD8E6", width = bar_width, edgecolor = 'white', label = 'Sales' )
plt.bar(bar2, category_quantity, color="#0000FF", width = bar_width, edgecolor = 'white', label = 'Quantity')
plt.bar(bar3, category_profit, color="#00008B", width = bar_width, edgecolor = 'white', label = 'Profit')

plt.xlabel('Segments', fontweight='bold')
plt.xticks([r + bar_width for r in range(num_categories)], categories)

plt.legend()
plt.show()
```



Plotting using plotly

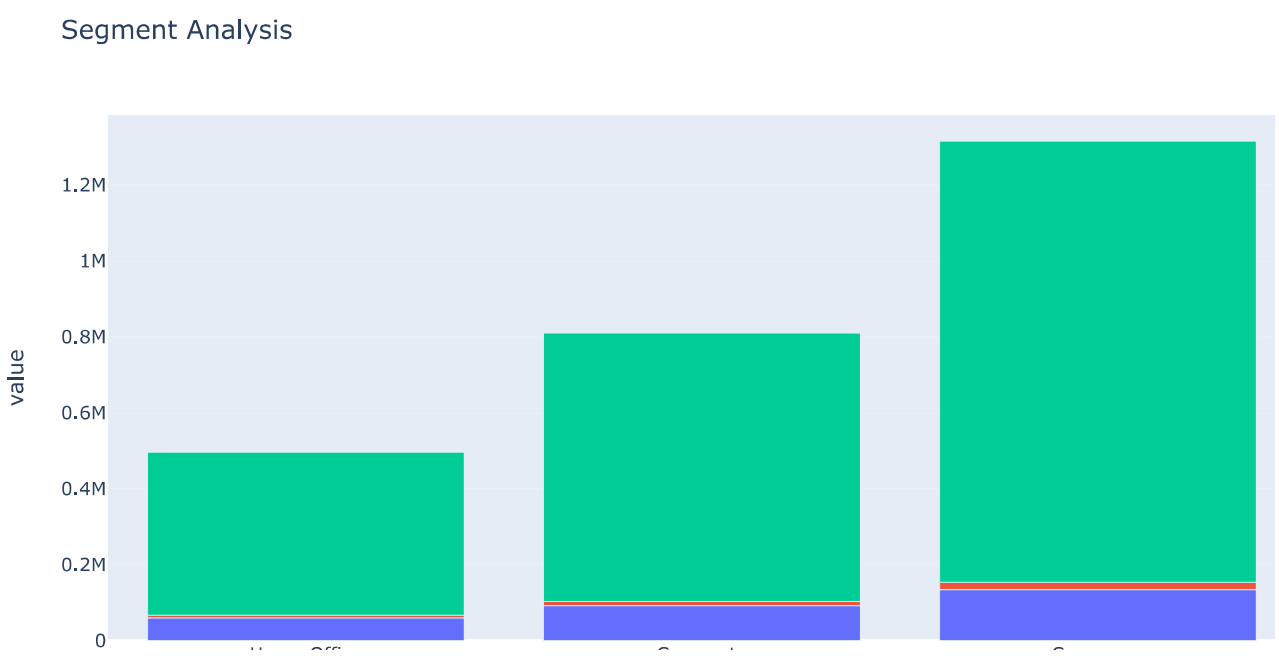
In [24]:
```
categories
```

Out[24]:
```
Index(['Home Office', 'Corporate', 'Consumer'], dtype='object', name='Segment')
```

In [25]:
```
category_profit, category_quantity, category_sales
```

Out[25]:
```
(Segment
 Home Office     60298.6785
 Corporate       91979.1340
 Consumer       134119.2092
 Name: Profit, dtype: float64,
 Segment
 Home Office     6744
 Corporate      11608
 Consumer       19521
 Name: Quantity, dtype: int64,
 [429653.1485, 706146.3668, 1161401.345])
```

In [26]:
```
px.bar(segment_analysis, x = categories, y = ["Profit", "Quantity", "Sales"], title="Segment Analysis")
```



10 Best customer of all time

In [27]:
```
superstore_data_df.head()
```

Out[27]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	F
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	

5 rows × 21 columns

In [28]:
```
superstore_data_df.count()
```

Out[28]:
```
Row ID           9994
Order ID         9994
Order Date       9994
Ship Date        9994
Ship Mode        9994
Customer ID      9994
Customer Name    9994
Segment          9994
Country          9994
City             9994
State            9994
Postal Code      9994
Region           9994
Product ID       9994
Category         9994
Sub-Category     9994
Product Name     9994
Sales            9994
Quantity         9994
Discount         9994
Profit           9994
dtype: int64
```

In [29]:
```
len(superstore_data_df.index), superstore_data_df.shape[0]
```

Out[29]:
```
(9994, 9994)
```

In [30]:
```
len(superstore_data_df["Customer Name"].unique())
```

Out[30]:
```
793
```

In [31]:
```
# finding the sales and profit ammounts done through a customer

best_customers = superstore_data_df.loc[:, ['Customer Name', 'Sales', 'Profit']]
best_customers
```

Out[31]:

	Customer Name	Sales	Profit
0	Claire Gute	261.9600	41.9136
1	Claire Gute	731.9400	219.5820
2	Darrin Van Huff	14.6200	6.8714
3	Sean O'Donnell	957.5775	-383.0310
4	Sean O'Donnell	22.3680	2.5164
...	...	...	...
9989	Tom Boeckenhauer	25.2480	4.1028
9990	Dave Brooks	91.9600	15.6332
9991	Dave Brooks	258.5760	19.3932
9992	Dave Brooks	29.6000	13.3200
9993	Chris Cortes	243.1600	72.9480

9994 rows × 3 columns

In [32]:
```
best_customers.groupby(["Customer Name"]).sum().head()
```

Out[32]:

Customer Name	Sales	Profit
Aaron Bergman	886.156	129.3465
Aaron Hawkins	1744.700	365.2152
Aaron Smayling	3050.692	-253.5746
Adam Bellavance	7755.620	2054.5885
Adam Hart	3250.337	281.1890

In [33]:
```
best_customers = best_customers.groupby("Customer Name", as_index=False).sum()
```

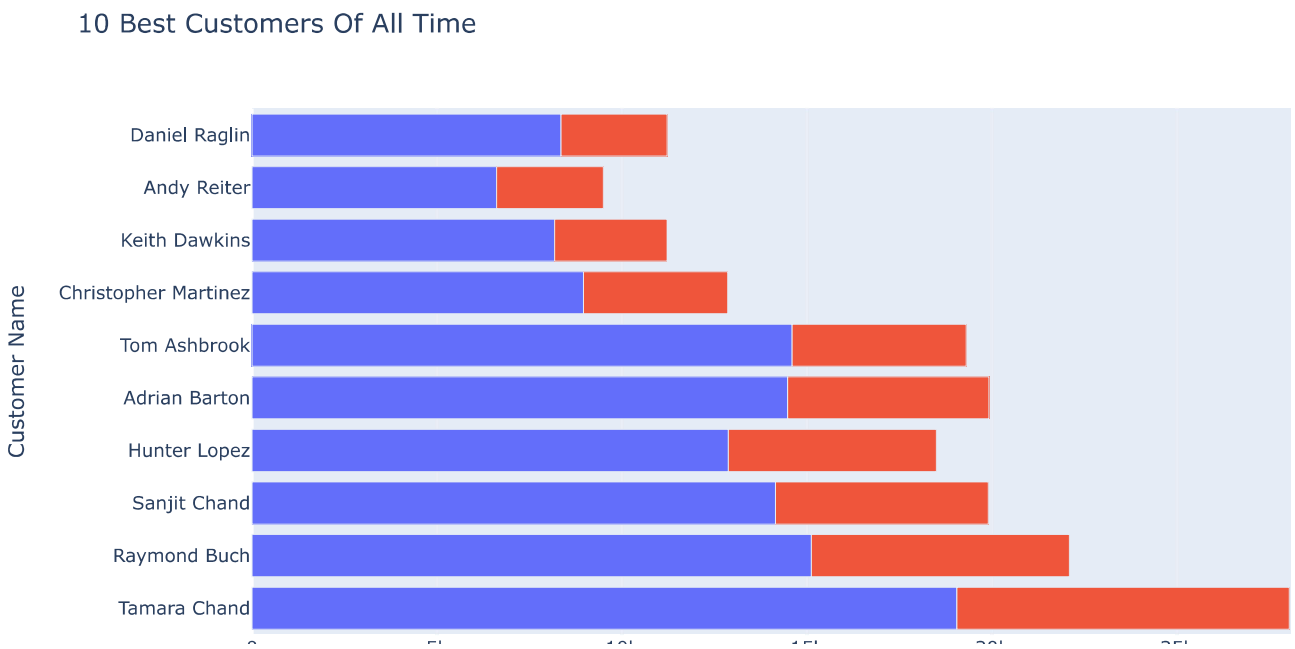
In [34]:
```
best_customers.sort_values('Profit', ascending=False, inplace=True)
```

In [35]:
```
ten_best_customers = best_customers[:10]
ten_best_customers
```

Out[35]:

	Customer Name	Sales	Profit
730	Tamara Chand	19052.218	8981.3239
622	Raymond Buch	15117.339	6976.0959
671	Sanjit Chand	14142.334	5757.4119
334	Hunter Lopez	12873.298	5622.4292
6	Adrian Barton	14473.571	5444.8055
757	Tom Ashbrook	14595.620	4703.7883
157	Christopher Martinez	8954.020	3899.8904
431	Keith Dawkins	8181.256	3038.6254
35	Andy Reiter	6608.448	2884.6208
194	Daniel Raglin	8350.868	2869.0760

In [36]:
```
px.bar(ten_best_customers,
       x = ["Sales", "Profit"],
       y = "Customer Name",
       orientation='h',
       title="10 Best Customers Of All Time", )
```



Total profit of the bussiness

In [37]:
```
superstore_data_df.head()
```

Out[37]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies

5 rows × 21 columns

In [38]:
```
superstore_data_df["Order Date"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 9994 entries, 0 to 9993
Series name: Order Date
Non-Null Count  Dtype
--------------  -----
9994 non-null   object
dtypes: object(1)
memory usage: 78.2+ KB
```

In [39]:
```
store_profits = superstore_data_df.loc[:, ["Order Date", "Profit"]]
store_profits
```

Out[39]:

	Order Date	Profit
0	2016-11-08	41.9136
1	2016-11-08	219.5820
2	2016-06-12	6.8714
3	2015-10-11	-383.0310
4	2015-10-11	2.5164
...	...	...
9989	2014-01-21	4.1028
9990	2017-02-26	15.6332
9991	2017-02-26	19.3932
9992	2017-02-26	13.3200
9993	2017-05-04	72.9480

9994 rows × 2 columns

In [40]:
```
store_profits["Order Date"].str[:4]
```

Out[40]:
```
0       2016
1       2016
2       2016
3       2015
4       2015
        ...
9989    2014
9990    2017
9991    2017
9992    2017
9993    2017
Name: Order Date, Length: 9994, dtype: object
```

In [41]:
```
store_profits["Order year"] = store_profits["Order Date"].str[:4]
```

In [42]:
```
store_profits.head()
```

Out[42]:

	Order Date	Profit	Order year
0	2016-11-08	41.9136	2016
1	2016-11-08	219.5820	2016
2	2016-06-12	6.8714	2016
3	2015-10-11	-383.0310	2015
4	2015-10-11	2.5164	2015

In [43]:
```
# Droping Order Date column
store_profits.drop(["Order Date"], axis=1, inplace=True)
store_profits.head()
```

Out[43]:

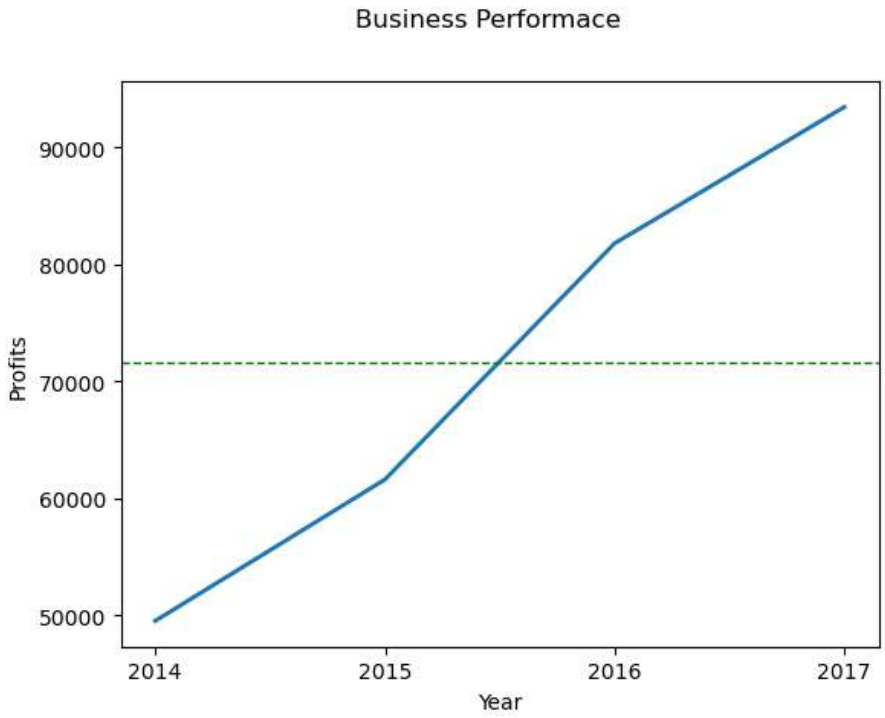
	Profit	Order year
0	41.9136	2016
1	219.5820	2016
2	6.8714	2016
3	-383.0310	2015
4	2.5164	2015

In [44]:
```
store_profits = store_profits.groupby(["Order year"], as_index=False).sum(numeric_only=True)
store_profits
```

Out[44]:

	Order year	Profit
0	2014	49543.9741
1	2015	61618.6037
2	2016	81795.1743
3	2017	93439.2696

In [45]:
```
# Plotting Graph

# preparing plot
fig, ax = plt.subplots()

# preparing data and plotting data
ax.plot(store_profits["Order year"], store_profits["Profit"], linewidth=2.0)

# customizing ax0
ax.set(xlabel = "Year", ylabel="Profits")
ax.axhline(store_profits["Profit"].mean(), color='green', linewidth=1.0, linestyle="--")

# customizing fig
fig.suptitle("Business Performace")

plt.show()
```



Sales analysis of each year

In [46]:
```
store_profits_year = superstore_data_df.loc[:, ["Order Date", "Profit"]]
store_profits_year.head()
```

Out[46]:

	Order Date	Profit
0	2016-11-08	41.9136
1	2016-11-08	219.5820
2	2016-06-12	6.8714
3	2015-10-11	-383.0310
4	2015-10-11	2.5164

In [47]:
```
store_profits_year["Order Date"].str[5:7]
```

Out[47]:
```
0       11
1       11
2       06
3       10
4       10
        ..
9989    01
9990    02
9991    02
9992    02
9993    05
Name: Order Date, Length: 9994, dtype: object
```

In [48]:
```
store_profits_year["Month"] = store_profits_year["Order Date"].str[5:7]
store_profits_year["Year"] = store_profits_year["Order Date"].str[:4]
store_profits_year.head()
```

Out[48]:

	Order Date	Profit	Month	Year
0	2016-11-08	41.9136	11	2016
1	2016-11-08	219.5820	11	2016
2	2016-06-12	6.8714	06	2016
3	2015-10-11	-383.0310	10	2015
4	2015-10-11	2.5164	10	2015

In [49]:
```
store_profits_year.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Order Date  9994 non-null   object 
 1   Profit      9994 non-null   float64
 2   Month       9994 non-null   object 
 3   Year        9994 non-null   object 
dtypes: float64(1), object(3)
memory usage: 312.4+ KB
```

In [50]:
```
store_profits_year_mod = store_profits_year.groupby(["Month", "Year"], as_index=False).sum(numeric_only=True)
store_profits_year_mod.head()
```

Out[50]:

	Month	Year	Profit
0	01	2014	2450.1907
1	01	2015	-3281.0070
2	01	2016	2824.8233
3	01	2017	7140.4391
4	02	2014	862.3084

In [51]:
```
store_profits_year_mod["Month"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 48 entries, 0 to 47
Series name: Month
Non-Null Count  Dtype  
--------------  -----  
48 non-null     object 
dtypes: object(1)
memory usage: 516.0+ bytes
```

In [52]:
```
store_profits_year_mod["Month"] = store_profits_year_mod["Month"].astype('int')
store_profits_year_mod["Year"] = store_profits_year_mod["Year"].astype('int')
store_profits_year_mod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Month   48 non-null     int32  
 1   Year    48 non-null     int32  
 2   Profit  48 non-null     float64
dtypes: float64(1), int32(2)
memory usage: 900.0 bytes
```

In [53]:
```
store_profits_year_mod.sort_values(by=['Year', 'Month'], ascending=True).head()
```

Out[53]:

	Month	Year	Profit
0	1	2014	2450.1907
4	2	2014	862.3084
8	3	2014	498.7299
12	4	2014	3488.8352
16	5	2014	2738.7096

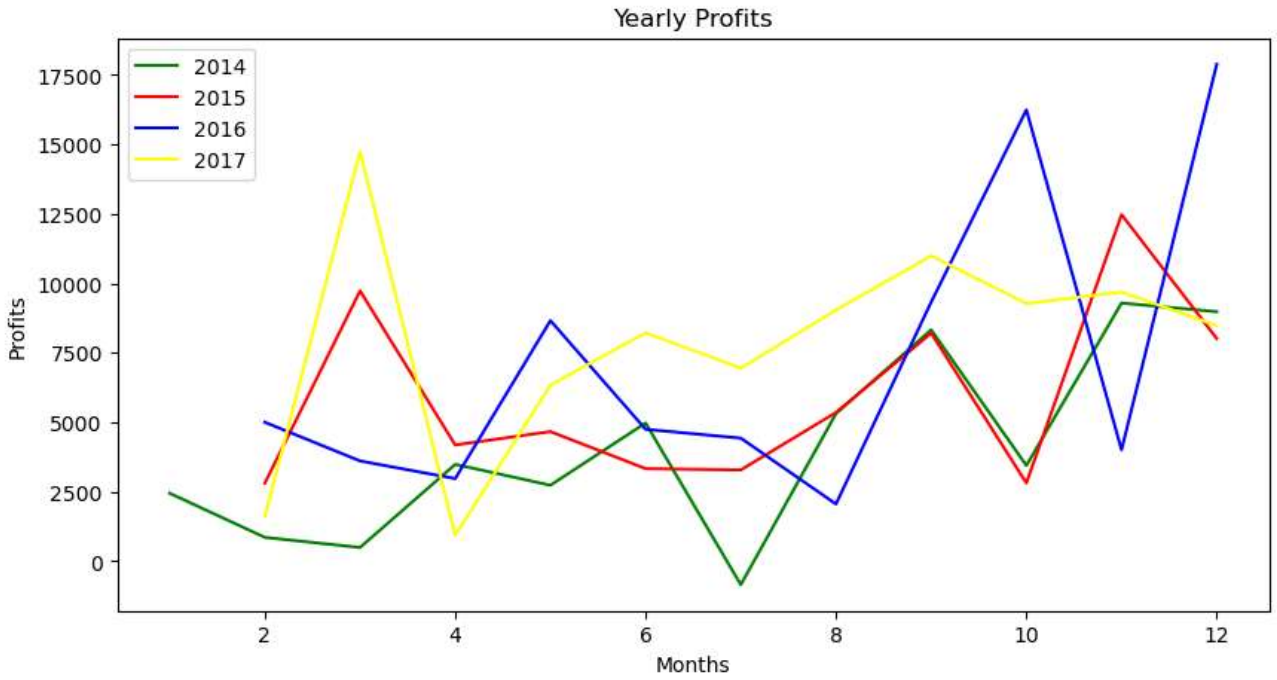
In [54]:
```
store_profits_year_mod.sort_values(by=['Year', 'Month'], ascending=True, inplace=True)
```

In [55]:
```
# preparing data for analysis

store_profits_2014 = store_profits_year_mod.iloc[:12]
store_profits_2015 = store_profits_year_mod.iloc[13:24]
store_profits_2016 = store_profits_year_mod.iloc[25:36]
store_profits_2017 = store_profits_year_mod.iloc[37:48]
```

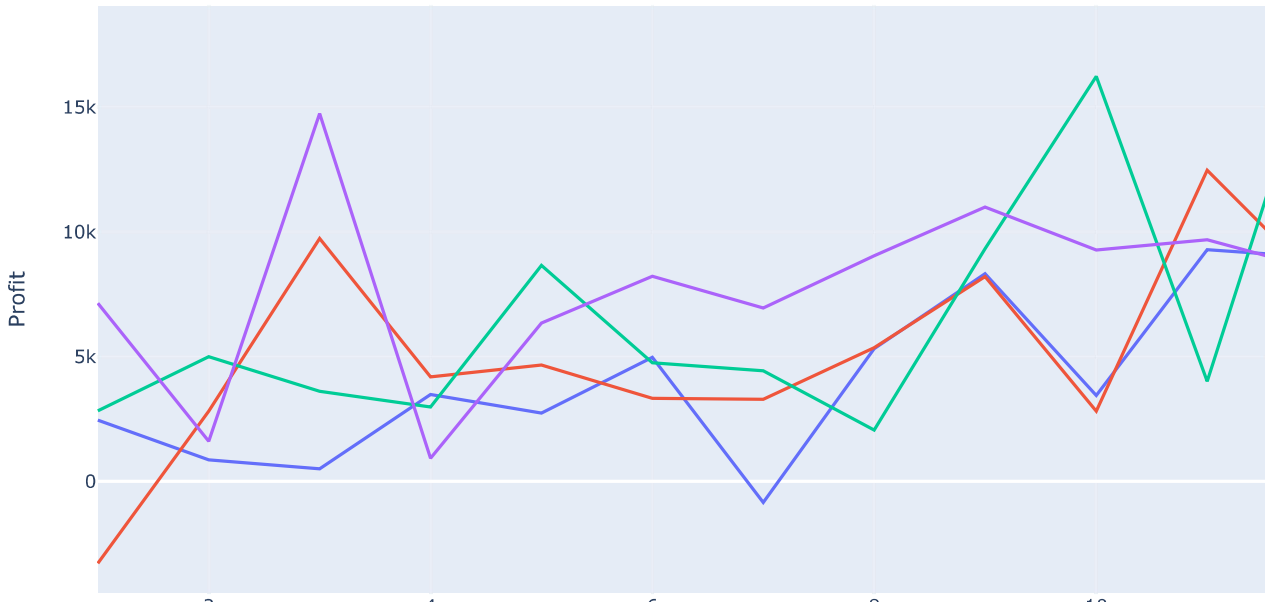
In [56]:
```
store_profits_2014, store_profits_2015, store_profits_2016, store_profits_2017
```

Out[56]:
```
(    Month  Year     Profit
 0       1  2014  2450.1907
 4       2  2014   862.3084
 8       3  2014   498.7299
 12      4  2014  3488.8352
 16      5  2014  2738.7096
 20      6  2014  4976.5244
 24      7  2014  -841.4826
 28      8  2014  5318.1050
 32      9  2014  8328.0994
 36     10  2014  3448.2573
 40     11  2014  9292.1269
 44     12  2014  8983.5699,
     Month  Year     Profit
 5       2  2015   2813.8508
 9       3  2015   9732.0978
 13      4  2015   4187.4962
 17      5  2015   4667.8690
 21      6  2015   3335.5572
 25      7  2015   3288.6483
 29      8  2015   5355.8084
 33      9  2015   8209.1627
 37     10  2015   2817.3660
 41     11  2015  12474.7884
 45     12  2015   8016.9659,
     Month  Year     Profit
 6       2  2016   5004.5795
 10      3  2016   3611.9680
 14      4  2016   2977.8149
 18      5  2016   8662.1464
 22      6  2016   4750.3781
 26      7  2016   4432.8779
 30      8  2016   2062.0693
 34      9  2016   9328.6576
 38     10  2016  16243.1425
 42     11  2016   4011.4075
 46     12  2016  17885.3093,
     Month  Year     Profit
 7       2  2017   1613.8720
 11      3  2017  14751.8915
 15      4  2017    933.2900
 19      5  2017   6342.5828
 23      6  2017   8223.3357
 27      7  2017   6952.6212
 31      8  2017   9040.9557
 35      9  2017  10991.5556
 39     10  2017   9275.2755
 43     11  2017   9690.1037
 47     12  2017   8483.3468)
```

In [57]:
```
# Plotting graphs

fig, ax = plt.subplots(figsize=(10, 5))

# setting up figure

ax.plot(store_profits_2014["Month"], store_profits_2014["Profit"], color="green", label="2014")
ax.plot(store_profits_2015["Month"], store_profits_2015["Profit"], color="red", label="2015")
ax.plot(store_profits_2016["Month"], store_profits_2016["Profit"], color="blue", label="2016")
ax.plot(store_profits_2017["Month"], store_profits_2017["Profit"], color="yellow", label="2017")

# customizing plot
ax.legend()
ax.set(xlabel="Months", ylabel="Profits", title="Yearly Profits")

# display figure
plt.show();
```



In [58]:
```
# Interactive graph
fig = px.line(store_profits_year_mod, x="Month", y="Profit", color="Year")
fig.show()
```



Regional sales

In [59]:
```
# Preparing data for Regional sales analysis
superstore_data_df.head(2)
```

Out[59]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	C
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bc
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	

2 rows × 21 columns

In [60]:
```
regional_sales_df = superstore_data_df.loc[:, ["Order Date", "Country", "City", "Region", "Sales", "Profit"]]
regional_sales_df.head()
```

Out[60]:

	Order Date	Country	City	Region	Sales	Profit
0	2016-11-08	United States	Henderson	South	261.9600	41.9136
1	2016-11-08	United States	Henderson	South	731.9400	219.5820
2	2016-06-12	United States	Los Angeles	West	14.6200	6.8714
3	2015-10-11	United States	Fort Lauderdale	South	957.5775	-383.0310
4	2015-10-11	United States	Fort Lauderdale	South	22.3680	2.5164

In [61]:
```
regional_sales_df["City"].count(), len(regional_sales_df["City"].unique()), regional_sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Order Date  9994 non-null   object 
 1   Country     9994 non-null   object 
 2   City        9994 non-null   object 
 3   Region      9994 non-null   object 
 4   Sales       9994 non-null   float64
 5   Profit      9994 non-null   float64
dtypes: float64(2), object(4)
memory usage: 468.6+ KB
```

Out[61]:
```
(9994, 531, None)
```

Fetching latitudes and logitudes of cities

In [62]:
```
#pip install geopy
```

In [63]:
```
from geopy.geocoders import Nominatim
```

In [64]:
```
geolocator = Nominatim(user_agent="Geolocation")
```

In [65]:
```
# fetching locations for regional_sales_df
locations = regional_sales_df["City"].unique()
locations_series = pd.Series(data = locations)
locations_series
```

Out[65]:
```
0            Henderson
1          Los Angeles
2      Fort Lauderdale
3              Concord
4              Seattle
             ...      
526       San Clemente
527    San Luis Obispo
528         Springdale
529               Lodi
530              Mason
Length: 531, dtype: object
```

In [66]:
```
# fetching adjacent latitude and logitude
longitudes = []
latitudes = []

# longitudes = locations.apply(lambda x: geolocator.geocode(x).longitude)
# latitudes = locaitons.apply(lambda x: geolocator.geocode(x).latitude)

for i in range(0, len(locations)):
    coordinates = geolocator.geocode(locations[i])
    longitudes.append(coordinates.longitude)
    latitudes.append(coordinates.latitude)

longitudes, latitudes
```

Out[66]:
```
([-95.7893178,
  -118.242766,
  -80.1433786,
  -71.537476,
  -122.330062,
  -97.3327459,
  -89.3837613,
  -111.9395211,
  -122.419906,
  -121.988571,
  -75.1635262,
  -111.6944313,
  -95.3676974,
  -96.7297206,
  -88.1479278,
  144.9631732,
  -93.1659179,
  -83.4005321,
  1.3134228,
```

In [67]:
```
location_info_df = pd.DataFrame({'City': locations_series,
                                 'Latitude': pd.Series(latitudes),
                                 'Longitude': pd.Series(longitudes)})
location_info_df.sort_values(by="City", ascending=True, inplace=True)
location_info_df.head()
```

Out[67]:

	City	Latitude	Longitude
520	Aberdeen	57.148243	-2.092809
488	Abilene	32.446450	-99.747591
59	Akron	41.083064	-81.518485
305	Albuquerque	35.084103	-106.650985
166	Alexandria	38.805110	-77.047023

In [69]:
```
# final dataframe for Regional analysis
regional_analysis_df = regional_sales_df.groupby(by=["City"], as_index=False).sum(numeric_only=True)
regional_analysis_df.head()
```

Out[69]:

	City	Sales	Profit
0	Aberdeen	25.500	6.6300
1	Abilene	1.392	-3.7584
2	Akron	2729.986	-186.6356
3	Albuquerque	2220.160	634.0881
4	Alexandria	5519.570	318.6183

In [70]:
```
regional_analysis_df["Latitude"] = location_info_df["Latitude"]
regional_analysis_df["Longitude"] = location_info_df["Longitude"]
regional_analysis_df.head()
```

Out[70]:

	City	Sales	Profit	Latitude	Longitude
0	Aberdeen	25.500	6.6300	32.182598	-95.789318
1	Abilene	1.392	-3.7584	34.053691	-118.242766
2	Akron	2729.986	-186.6356	26.122308	-80.143379
3	Albuquerque	2220.160	634.0881	43.207178	-71.537476
4	Alexandria	5519.570	318.6183	47.603832	-122.330062

In [71]:
```
central_latitude = regional_analysis_df['Latitude'].mean()
central_longitude = regional_analysis_df['Longitude'].mean()
central_latitude, central_longitude
```

Out[71]: (37.40520806120527, -87.2209418998167)

1. Sales Concentration

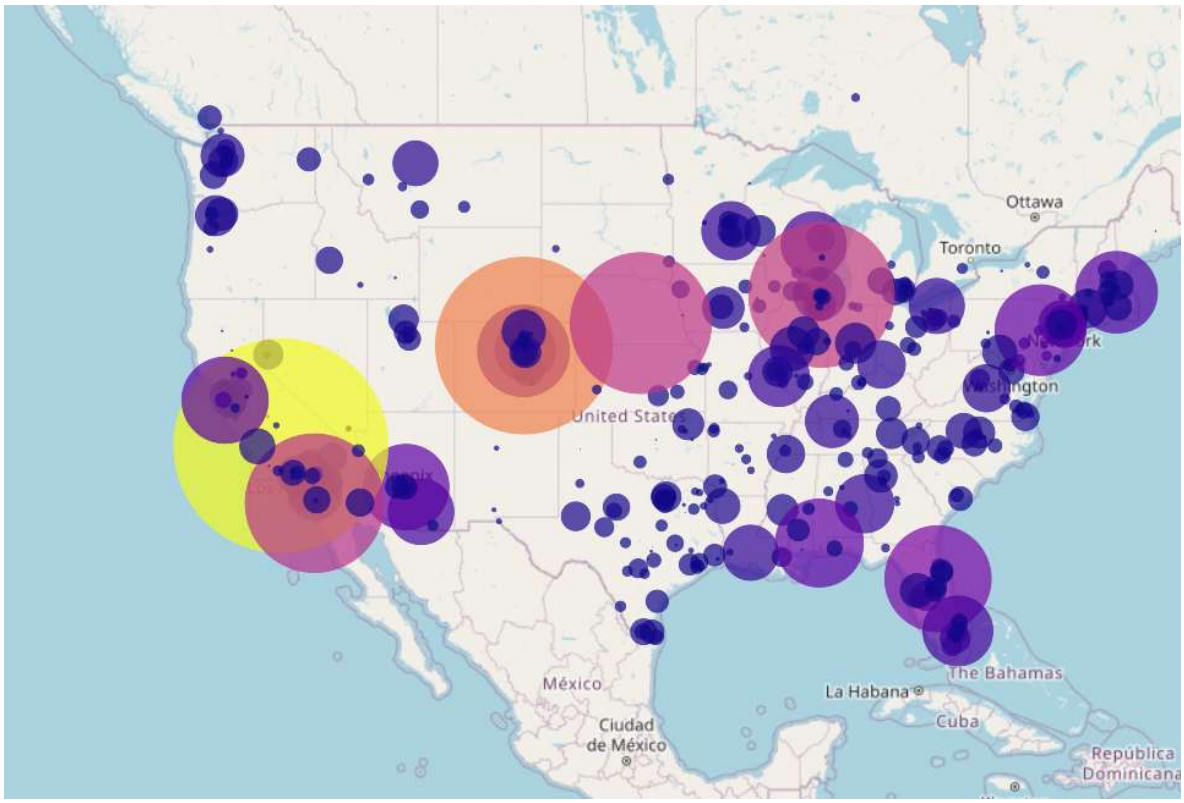
In [72]:
```
fig = px.scatter_mapbox(
    regional_analysis_df,
    title="map",
    lat='Latitude',
    lon='Longitude',
    color='Sales',
    size='Sales',
    size_max=100,
    hover_name='Sales',
    hover_data=['City', 'Sales', 'Profit'],
)



fig.layout.update(
    height=700,
    coloraxis_showscale=True,
    mapbox_style="open-street-map",
    mapbox=dict(center=dict(lat=central_latitude, lon=-100), zoom=3)
);
```

In [73]:
```
fig.data[0]
```

Out[73]:
```
Scattermapbox({
    'customdata': array([['Aberdeen', 25.5, 6.63],
                         ['Abilene', 1.3919999999999997, -3.758400000000001],
                         ['Akron', 2729.986, -186.63560000000012],
                         ...,
                         ['York', 817.9780000000001, -102.16920000000002],
                         ['Yucaipa', 50.8, 13.208000000000002],
                         ['Yuma', 840.8650000000001, -465.9909]], dtype=object),
    'hovertemplate': ('<b>%{hovertext}</b><br><br>Sal' ... '{customdata[2]}<extra></extra>'),
    'hovertext': array([2.550000e+01, 1.392000e+00, 2.729986e+03, ..., 8.179780e+02,
                        5.080000e+01, 8.408650e+02]),
    'lat': array([32.182598 , 34.0536909, 26.1223084, ..., 36.1867442, 45.2613104,
                  30.705029 ]),
    'legendgroup': '',
    'lon': array([ -95.7893178 , -118.242766  ,  -80.1433786 , ...,  -94.1288142 ,
                    9.49167806,  -99.2233284 ]),
    'marker': {'color': array([2.550000e+01, 1.392000e+00, 2.729986e+03, ..., 8.179780e+02,
                               5.080000e+01, 8.408650e+02]),
               'coloraxis': 'coloraxis',
               'size': array([2.550000e+01, 1.392000e+00, 2.729986e+03, ..., 8.179780e+02,
                              5.080000e+01, 8.408650e+02]),
               'sizemode': 'area',
               'sizeref': 25.6368161},
    'mode': 'markers',
    'name': '',
    'showlegend': False,
    'subplot': 'mapbox'
})
```

In [74]:
```
fig.show()
```

map



2. Tabel representation

In [75]:
```
regional_analysis_df[['City', 'Sales', 'Profit']].sort_values(by='Profit', ascending=False).head(10).style\
.background_gradient(cmap='Greens', subset=['Sales'])\
.background_gradient(cmap='Reds', subset=['Profit'])
```

Out[75]:

	City	Sales	Profit
329	New York City	256368.161000	62036.983700
266	Los Angeles	175851.341000	30440.757900
452	Seattle	119540.742000	29156.096700
438	San Francisco	112669.092000	17507.385400
123	Detroit	42446.944000	13181.790800
233	Lafayette	25036.200000	10018.387600
215	Jackson	24963.858000	7581.682800
21	Atlanta	17197.840000	6993.662900
300	Minneapolis	16870.540000	6824.584600
437	San Diego	47521.029000	6377.196000

In [ ]: