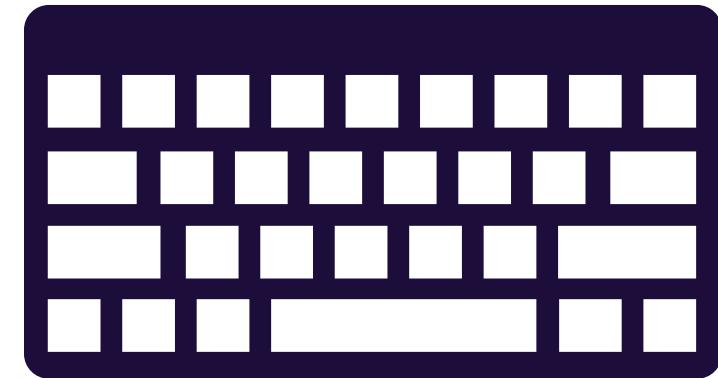


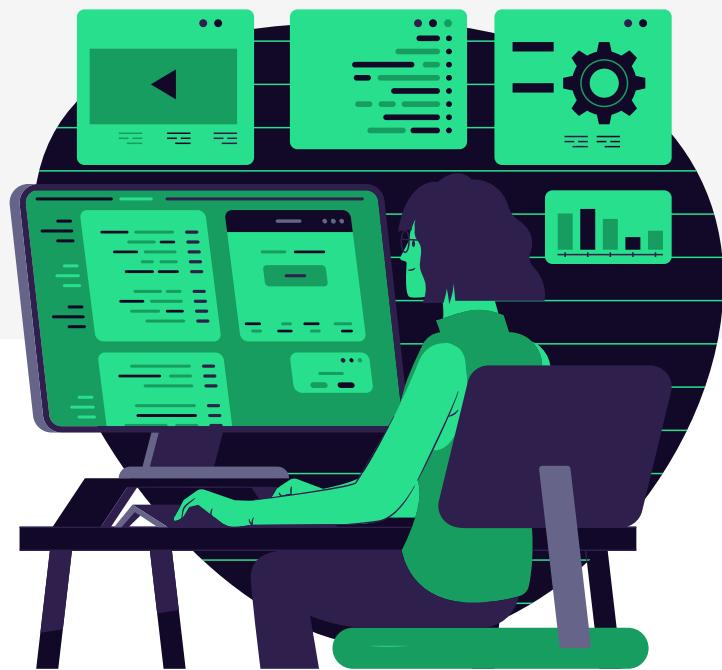
```
(base) C:\Users\ITMGT-A>cd superstore_dataset
```

```
Group6 =  
["Buenaventura", "Sta. Maria", "Verches"]
```



Importing Necessary Libraries

```
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
  
import geopandas as gpd
```



Sourced Superstore Dataset

\SuperstoreDataset.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount	Profit
2	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Colle	261.96	2	0	41.9136
3	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric U	731.94	3	0	219.582
4	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address	14.62	2	0	6.8714
5	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Ser	957.5775	5	0.45	-383.031
6	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll C	22.368	2	0.2	2.5164
7	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	FUR-FU-10001487	Furniture	Furnishings	Eldon Expressions W	48.86	7	0	14.1694
8	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-AR-10002833	Office Supplies	Art	Newell 322	7.28	4	0	1.9656
9	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	TEC-PH-10002275	Technology	Phones	Mitel 5320 IP Phone	907.152	6	0.2	90.7152
10	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-BI-10003910	Office Supplies	Binders	DXL Angle-View Bin	18.504	3	0.2	5.7825
11	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-AP-10002892	Office Supplies	Appliances	Belkin F5C206VTEL	114.9	5	0	34.47
12	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	FUR-TA-10001539	Furniture	Tables	Chromcraft Rectangu	1706.184	9	0.2	85.3092
13	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	TEC-PH-10002033	Technology	Phones	Konftel 250 Conferen	911.424	4	0.2	68.3568
14	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	28027	South	OFF-PA-10002365	Office Supplies	Paper	Xerox 1967	15.552	3	0.2	5.4432
15	Standard Class	IM-15070	Irene Maddox	Consumer	United States	Seattle	Washington	98103	West	OFF-BI-10003656	Office Supplies	Binders	Fellowes PB200 Plat	407.976	3	0.2	132.5922
16	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OFF-AP-10002311	Office Supplies	Appliances	Holmes Replacemen	68.81	5	0.8	-123.858
17	Standard Class	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OFF-BI-10000756	Office Supplies	Binders	Storex DuraTech Re	2.544	3	0.8	-3.816
18	Standard Class	PK-19075	Pete Kriz	Consumer	United States	Madison	Wisconsin	53711	Central	OFF-ST-10004186	Office Supplies	Storage	Stur-D-Stor Shelving	665.88	6	0	13.3176
19	Second Class	AG-10270	Alejandro Grove	Consumer	United States	West Jordan	Utah	84084	West	OFF-ST-10000107	Office Supplies	Storage	Fellowes Super Stor	55.5	2	0	9.99
20	Second Class	ZD-21925	Zuschuss Donatelli	Consumer	United States	San Francisco	California	94109	West	OFF-AR-10003056	Office Supplies	Art	Newell 341	8.56	2	0	2.4824
21	Second Class	ZD-21925	Zuschuss Donatelli	Consumer	United States	San Francisco	California	94109	West	TEC-PH-10001949	Technology	Phones	Cisco SPA 501G IP I	213.48	3	0.2	16.011
22	Second Class	ZD-21925	Zuschuss Donatelli	Consumer	United States	San Francisco	California	94109	West	OFF-BI-10002215	Office Supplies	Binders	Wilson Jones Hangin	22.72	4	0.2	7.384
23	Standard Class	KB-16585	Ken Black	Corporate	United States	Fremont	Nebraska	68025	Central	OFF-AR-10000246	Office Supplies	Art	Newell 318	19.46	7	0	5.0596
24	Standard Class	KB-16585	Ken Black	Corporate	United States	Fremont	Nebraska	68025	Central	OFF-AP-10001492	Office Supplies	Appliances	Acco Six-Outlet Pow	60.34	7	0	15.6884
25	Second Class	SF-20065	Sandra Flanagan	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUR-CH-10002774	Furniture	Chairs	Global Deluxe Stack	71.372	2	0.3	-1.0196
26	Standard Class	EB-13870	Emily Burns	Consumer	United States	Orem	Utah	84057	West	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Ser	1044.63	3	0	240.2649
27	Second Class	EH-13945	Eric Hoffmann	Consumer	United States	Los Angeles	California	90049	West	OFF-BI-10001634	Office Supplies	Binders	Wilson Jones Active	11.648	2	0.2	4.2224
28	Second Class	EH-13945	Eric Hoffmann	Consumer	United States	Los Angeles	California	90049	West	TEC-AC-10003027	Technology	Accessories	Imation 8GB Mini Tr	90.57	3	0	11.7741
29	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUR-BO-10004834	Furniture	Bookcases	Riverside Palais Roy	3083.43	7	0.5	-1665.0522
30	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OFF-BI-10000474	Office Supplies	Binders	Avery Recycled Flex	9.618	2	0.7	-7.0532
31	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	FUR-FU-10004848	Furniture	Furnishings	Howard Miller 13-3/4	124.2	3	0.2	15.525
32	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OFF-EN-10001509	Office Supplies	Envelopes	Poly String Tie Enve	3.264	2	0.2	1.1016
33	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OFF-AR-10004042	Office Supplies	Art	BOSTON Model 180	86.304	6	0.2	9.7092
34	Standard Class	TB-21520	Tracy Blumstein	Consumer	United States	Philadelphia	Pennsylvania	19140	East	OFF-BI-10001525	Office Supplies	Binders	Acco Pressboard Co	6.858	6	0.7	-5.715

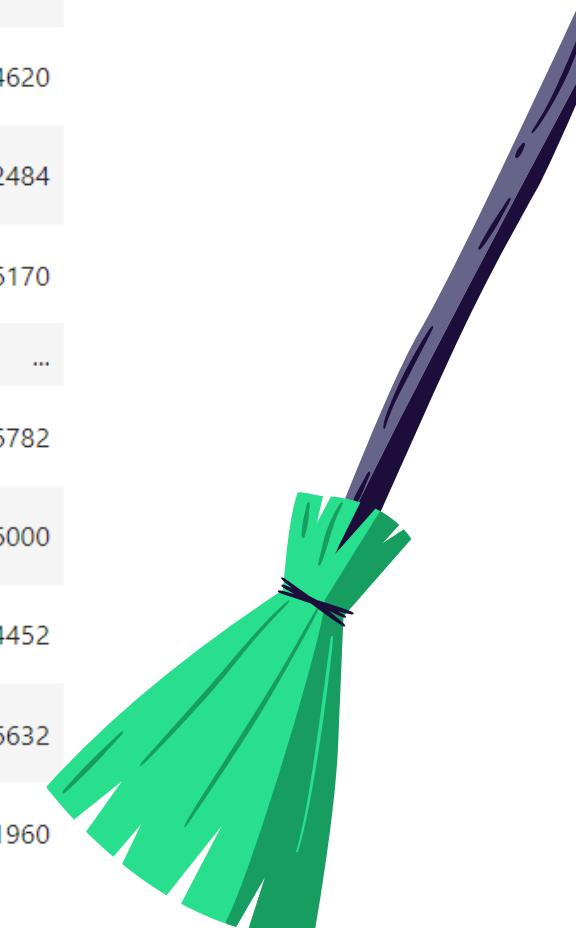
Reading and Cleaning Data



```
dataset = pd.read_csv("SuperstoreDataset.csv")
dataset = dataset.drop(columns=["Customer Name", "Country", "Postal Code", "Unnamed: 17"]).sort_values(["Region", "State", "City"], ascending=[True, True, True])[[["Region", "State", "City"]]
```

	Region	State	City	Customer ID	Segment	Product ID	Product Name	Ship Mode	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Central	Illinois	Arlington Heights	SC-20845	Consumer	OFF-AR-10003394	Newell 332	Standard Class	Office Supplies	Art	14.112	6	0.2	1.2348
1	Central	Illinois	Aurora	EH-14125	Home Office	FUR-CH-10001215	Global Troy Executive Leather Low-Back Tilter	Same Day	Furniture	Chairs	701.372	2	0.3	-50.0980
2	Central	Illinois	Aurora	EH-14125	Home Office	OFF-BI-10004654	Avery Binding System Hidden Tab Executive Styl...	Same Day	Office Supplies	Binders	2.308	2	0.8	-3.4620
3	Central	Illinois	Aurora	JK-16120	Home Office	FUR-FU-10003394	Tenex "The Solids" Textured Chair Mats	Standard Class	Furniture	Furnishings	83.952	3	0.6	-90.2484
4	Central	Illinois	Aurora	CK-12760	Corporate	FUR-TA-10002958	Bevis Oval Conference Table, Walnut	Standard Class	Furniture	Tables	652.450	5	0.5	-430.6170
...
9989	West	Washington	Vancouver	AR-10825	Corporate	TEC-PH-10003273	AT&T TR1909W	Second Class	Technology	Phones	302.376	3	0.2	22.6782
9990	West	Washington	Vancouver	AR-10825	Corporate	TEC-AC-10001142	First Data FD10 PIN Pad	Second Class	Technology	Accessories	316.000	4	0.0	31.6000
9991	West	Washington	Vancouver	FM-14290	Home Office	OFF-AR-10002956	Boston 16801 Nautilus Battery Pencil Sharpener	Standard Class	Office Supplies	Art	44.020	2	0.0	11.4452
9992	West	Washington	Vancouver	JW-15220	Corporate	FUR-FU-10002885	Magna Visual Magnetic Picture Hangers	Standard Class	Furniture	Furnishings	9.640	2	0.0	3.6632
9993	West	Wyoming	Cheyenne	MA-17995	Home Office	FUR-CH-10001215	Global Troy Executive Leather Low-Back Tilter	Standard Class	Furniture	Chairs	1603.136	4	0.2	100.1960

9994 rows × 14 columns



\data_by_profit.ipnyb



Separating States by Regions and Creating a Spatial File for Map Visualization

```
regions = ["Central", "West", "East", "South"]
region_data = {}

gdf = gpd.read_file("us_states")
gdf = gdf.rename(columns={"NAME": "state"})
gdf_dict = {}

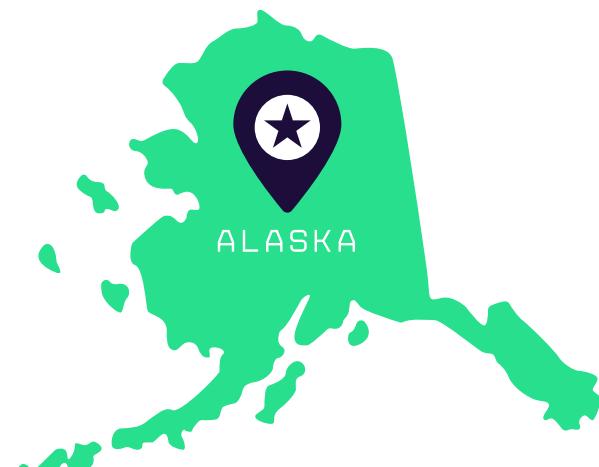
for region in regions:
    region_data[region] = dataset[dataset["Region"] == region]
    region_data[region].sort_values(by="State")
    set_region = set(region_data[region]["State"].unique())
    gdf_dict[region] = gdf.loc[gdf["state"].isin(set_region)].reset_index().drop(columns=["GEO_ID", "STATE", "index", "CENSUSAREA", "LSAD"])
    region_data[region] = pd.DataFrame(region_data[region].groupby("State").sum()).reset_index()
    region_data[region] = pd.concat([region_data[region], gdf_dict[region]], axis=1)
    region_data[region] = region_data[region].sort_values(by="Profit", ascending=False).reset_index().drop(columns = ["Quantity", "Discount", "index", "state"])

region_gdf = pd.concat([region_data["Central"], region_data["East"], region_data["West"], region_data["South"]])
region_gdf = gpd.GeoDataFrame(region_gdf, geometry='geometry')
region_gdf.to_file("states.geojson", driver="GeoJSON")

for region in regions:
    region_data[region].drop(columns=["geometry"], inplace = True)
    region_data[region].to_csv("region_data/{}_region_data.csv".format(region.lower()), index=False)
```



Sourced data for State geometry from
<https://eric.clst.org/tech/usgeojson/>



	GEO_ID	STATE	state	LSAD	CENSUSAREA	geometry
0	0400000US01	01	Alabama	None	50645.326	MULTIPOLYGON (((-88.12466 30.28364, -88.08681 ...
1	0400000US02	02	Alaska	None	570640.950	MULTIPOLYGON (((-166.10574 53.98861, -166.0752...)
2	0400000US04	04	Arizona	None	113594.084	POLYGON ((-112.53859 37.00067, -112.53454 37.0...
3	0400000US05	05	Arkansas	None	52035.477	POLYGON ((-94.04296 33.01922, -94.04304 33.079...
4	0400000US06	06	California	None	155779.220	MULTIPOLYGON (((-122.42144 37.86997, -122.4213...
5	0400000US08	08	Colorado	None	103641.888	POLYGON ((-106.19055 40.99761, -106.06118 40.9...
6	0400000US09	09	Connecticut	None	4842.355	POLYGON ((-71.79924 42.00806, -71.79792 41.935...
7	0400000US10	10	Delaware	None	1948.543	MULTIPOLYGON (((-75.56493 39.58325, -75.57627 ...
8	0400000US11	11	District of Columbia	None	61.048	POLYGON ((-77.03860 38.79151, -77.03890 38.800...
9	0400000US12	12	Florida	None	53624.759	MULTIPOLYGON (((-82.82158 27.96444, -82.82980 ...
10	0400000US13	13	Georgia	None	57513.485	POLYGON ((-84.81048 34.98761, -84.80918 34.987...
11	0400000US15	15	Hawaii	None	6422.628	MULTIPOLYGON (((-155.77823 20.24574, -155.7727...
12	0400000US16	16	Idaho	None	82643.117	POLYGON ((-111.04416 43.02005, -111.04413 43.0...
13	0400000US17	17	Illinois	None	55518.930	POLYGON ((-89.36603 42.50027, -89.36156 42.500...
14	0400000US18	18	Indiana	None	35826.109	POLYGON ((-84.80248 40.52805, -84.80255 40.501...
15	0400000US19	19	Iowa	None	55857.130	POLYGON ((-91.21771 43.50055, -91.21827 43.497...
16	0400000US20	20	Kansas	None	81758.717	POLYGON ((-99.54112 36.99957, -99.55807 36.999...
17	0400000US21	21	Kentucky	None	39486.338	MULTIPOLYGON (((-89.48511 36.49769, -89.49254 ...
18	0400000US22	22	Louisiana	None	43203.905	MULTIPOLYGON (((-88.86507 29.75271, -88.88976 ...

Alaska, Hawaii, and Puerto Rico were filtered from this

New Dataframes for each Region



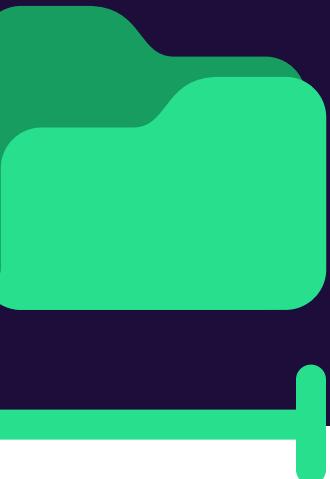
region_data["Central"]			
	State	Sales	Profit
0	Michigan	76269.6140	24463.1876
1	Indiana	53555.3600	18382.9363
2	Minnesota	29863.1500	10823.1874
3	Wisconsin	32114.6100	8401.8004
4	Missouri	22205.1500	6436.2105
5	Oklahoma	19683.3900	4853.9560
6	Nebraska	7464.9300	2037.0942
7	Iowa	4579.7600	1183.8119
8	Kansas	2914.3100	836.4435
9	South Dakota	1315.5600	394.8283
10	North Dakota	919.9100	230.1497
11	Illinois	80166.1010	-12607.8870
12	Texas	170188.0458	-25729.3563

region_data["East"]			
	State	Sales	Profit
0	New York	310876.271	74038.5486
1	Delaware	27451.069	9977.3748
2	New Jersey	35764.312	9772.9138
3	Rhode Island	22627.956	7285.6293
4	Maryland	23705.523	7031.1788
5	Massachusetts	28634.434	6785.5016
6	Connecticut	13384.357	3511.4918
7	Vermont	8929.370	2244.9783
8	New Hampshire	7292.524	1706.5028
9	District of Columbia	2865.020	1059.5893
10	Maine	1270.530	454.4862
11	West Virginia	1209.824	185.9216
12	Pennsylvania	116511.914	-15559.9603
13	Ohio	78258.136	-16971.3766

region_data["West"]			
	State	Sales	Profit
0	California	457687.6315	76381.3871
1	Washington	138641.2700	33402.6517
2	Nevada	16729.1020	3316.7659
3	Utah	11220.0560	2546.5335
4	Montana	5589.3520	1833.3285
5	New Mexico	4783.5220	1157.1161
6	Idaho	4382.4860	826.7231
7	Wyoming	1603.1360	100.1960
8	Oregon	17431.1500	-1190.4705
9	Arizona	35282.0010	-3427.9246
10	Colorado	32108.1180	-6527.8579

region_data["South"]			
	State	Sales	Profit
0	Virginia	70636.720	18597.9504
1	Georgia	49095.840	16250.0433
2	Kentucky	36591.750	11199.6966
3	Alabama	19510.640	5786.8253
4	Arkansas	11678.130	4008.6871
5	Mississippi	10771.340	3172.9762
6	Louisiana	9217.030	2196.1023
7	South Carolina	8481.710	1769.0566
8	Florida	89473.708	-3399.3017
9	Tennessee	30661.873	-5341.6936
10	North Carolina	55603.164	-7490.9122

\region_data\



	A	B	C		A	B	C		A	B	C		A	B	C
1	State	Sales	Profit	1	State	Sales	Profit	1	State	Sales	Profit	1	State	Sales	Profit
2	Michigan	76269.614	24463.1876	2	New York	310876.271	74038.5486	2	California	457687.6315	76381.3871	2	Virginia	70636.72	18597.9504
3	Indiana	53555.36	18382.9363	3	Delaware	27451.069	9977.3748	3	Washington	138641.27	33402.6517	3	Georgia	49095.84	16250.0433
4	Minnesota	29863.15	10823.1874	4	New Jersey	35764.312	9772.9138	4	Nevada	16729.102	3316.7659	4	Kentucky	36591.75	11199.6966
5	Wisconsin	32114.61	8401.8004	5	Rhode Island	22627.956	7285.6293	5	Utah	11220.056	2546.5335	5	Alabama	19510.64	5786.8253
6	Missouri	22205.15	6436.2105	6	Maryland	23705.523	7031.1788	6	Montana	5589.352	1833.3285	6	Arkansas	11678.13	4008.6871
7	Oklahoma	19683.39	4853.956	7	Massachusetts	28634.434	6785.5016	7	New Mexico	4783.522	1157.1161	7	Mississippi	10771.34	3172.9762
8	Nebraska	7464.93	2037.0942	8	Connecticut	13384.357	3511.4918	8	Idaho	4382.486	826.7231	8	Louisiana	9217.03	2196.1023
9	Iowa	4579.76	1183.8119	9	Vermont	8929.37	2244.9783	9	Wyoming	1603.136	100.196	9	South Carolina	8481.71	1769.0566
10	Kansas	2914.31	836.4435	10	New Hampshire	7292.524	1706.5028	10	Oregon	17431.15	-1190.4705	10	Florida	89473.708	-3399.3017
11	South Dakota	1315.56	394.8283	11	District of Columbia	2865.02	1059.5893	11	Arizona	35282.001	-3427.9246	11	Tennessee	30661.873	-5341.6936
12	North Dakota	919.91	230.1497	12	Maine	1270.53	454.4862	12	Colorado	32108.118	-6527.8579	12	North Carolina	55603.164	-7490.9122
13	Illinois	80166.101	-12607.887	13	West Virginia	1209.824	185.9216	13				13			
14	Texas	170188.0458	-25729.3563	14	Pennsylvania	116511.914	-15559.9603	14				14			
15				15	Ohio	78258.136	-16971.3766	15				15			
16				16				16				16			
17				17				17				17			

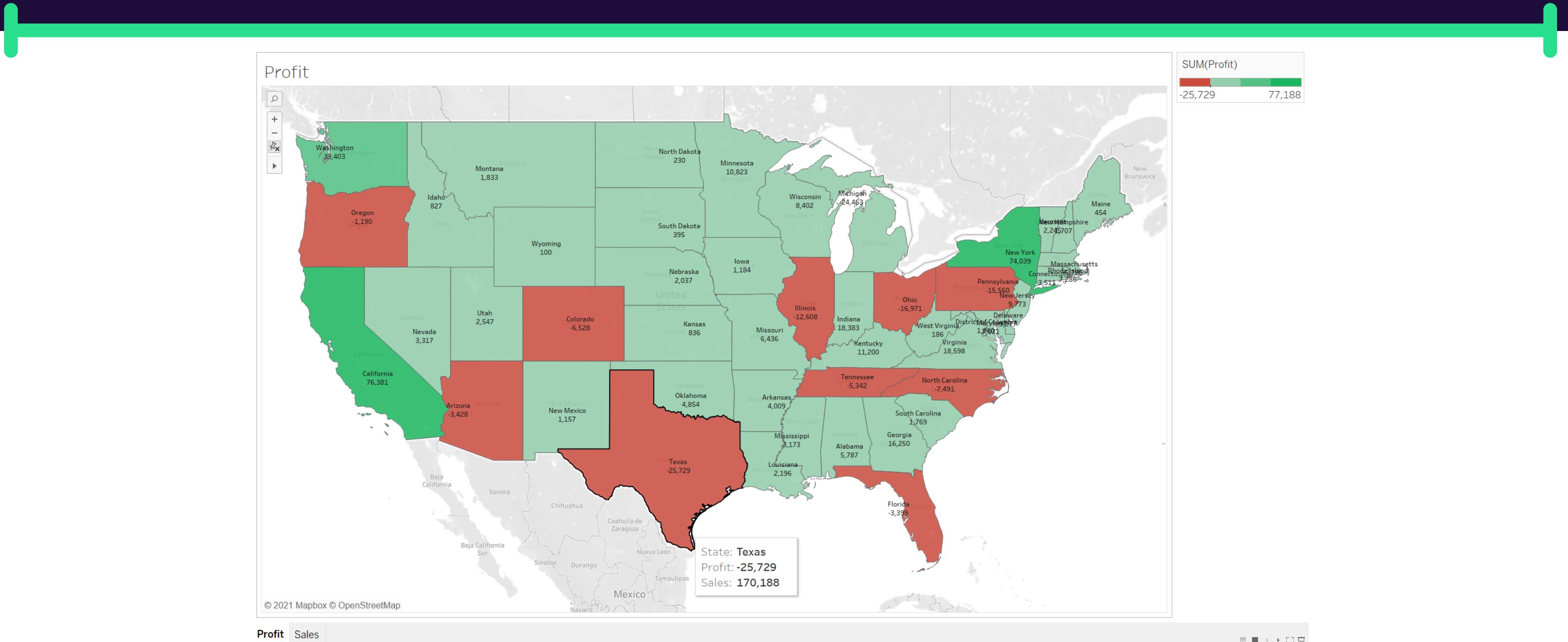
central_region_data east_region_data west_region_data south_region_data

```
# Generated Spatial file  
\states.geojson
```

[{"type": "FeatureCollection", "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::4269" } }, "features": [{"type": "Feature", "properties": { "State": "Michigan", "Sales": 76269.614, "Profit": 24463.1876 }, "geometry": { "type": "MultiPolygon", "coordinates": [[[[[-85.825954778925095, 45.404296308954493], [-85.833516344733809, 45.378174536161005], [-85.8733864189975], [-85.989412, 45.151069], [-85.976803, 45.138363], [-85.976434, 45.120706], [-85.980433, 45.113046], [-85.984095, 45.087073], [-85.982799, 45.072993], [-85.976883, 45.06266], [-85.99736, 45.055929], [-86.013073, 45.063774], [-88.449502, 48.163312], [-88.459697, 48.158551], [-88.469573, 48.152879], [-88.4857, 48.137683], [-88.511902, 48.121699], [-88.566938, 48.093719], [-88.578053, 48.084373], [-88.578395, 48.078003], [-88.575869, 48.075166], [-88.573924, 48.068 .828616], [-89.157738, 47.824015], [-89.19017, 47.831603], [-89.192681, 47.83343], [-89.201812, 47.850243], [-89.234533, 47.851718], [-89.235552, 47.853774], [-89.234535, 47.855373], [-89.228507, 47.858039], [-89.246774, 47.87], [-88.695353, 48.110549], [-88.684434, 48.115785]]], [[[-84.6128450885479, 45.834528302537201], [-84.623863, 45.846579], [-84.629239, 45.850014], [-84.6397, 45.852624], [-84.650783, 45.85921], [-84.651336, 45.862844], [-84.646876, 45.884642], [-84.5727952], [-84.469183, 45.732246], [-84.484128, 45.73071], [-84.500892, 45.737259], [-84.507476, 45.744644], [-84.509301, 45.754336], [-84.52506, 45.764168], [-84.549902, 45.789859], [-84.561562, 45.796213], [-84.58195, 45.802633], [-84.587572, 45.723774498198], [-85.651865706952208, 45.743139356864198], [-85.694352838174295, 45.722552427601599], [-85.696872074143698, 45.697249977280102], [[[-85.360951901267995, 45.8175540128859], [-85.408611, 45.760251], [-85.498777, 45.726291], [-85.494154, 45.705378], [-85.494016, 45.698476], [-85.5028, 45.690998], [-85.506104, 45.681148], [-85.503767, 45.670472], [-85.500451, 45.664298], [-85.490252, 45.652122], [-85.487026, 45.621211], [-85.491347, 706, 45.562278817688899], [-86.622023187090093, 45.556330165174295], [-86.636894818376504, 45.542053399139299], [-86.648792123405698, 45.5432431296422], [-86.661284293686293, 45.574176122718001], [-86.6917052472020991, 45.595815886266195], [-86.71232769522646 , 41.737603], [-84.806134, 41.743115], [-84.805883, 41.760216], [-84.818873, 41.760059], [-84.825129909657392, 41.759990721213597], [-84.825196, 41.75999], [-84.932484, 41.759691], [-84.96086, 41.759438], [-84.961562, 41.759552], [-84.971551, 41.759527 759049], [-85.624987, 41.759093], [-85.632714, 41.759164], [-85.647683, 41.759125], [-85.650738, 41.759103], [-85.659750008000898, 41.759100801830094], [-85.724534, 41.759085], [-85.749992, 41.759091], [-85.750469, 41.75909], [-85.775039, 41.759147], [-86.800611, 41.760251], [-86.800707, 41.76024], [-86.801578, 41.76024], [-86.823628, 41.76024], [-86.824828, 41.76024], [-86.82355014709789, 41.760897704588203], [-86.82064932675307, 41.771956038517096], [-86.801772322803501 7, 41.873688936117404], [-86.643964572637103, 41.873757235062797], [-86.629866684294001, 41.885259229438199], [-86.629344189346483, 41.885722849775298], [-86.6238650305001, 41.890230450627499], [-86.623773948884406, 41.890281646210404], [-86.621259066230895 3511447296, 41.941178622649024], [-86.582197, 41.942241], [-86.58151758234494, 41.94376454954997], [-86.564272035243889, 41.981427469501398], [-86.564077051360499, 41.98287380321697], [-86.563904059667294, 41.98326 105139], [-86.485223, 42.118239], [-86.479030704282394, 42.12351829379599], [-86.466576009626493, 42.134138261292598], [-86.466262, 42.134406], [-86.464912763493601, 42.13575213036999], [-86.464897466870497, 42.135767391772], [-86.464355548752096, 42.13 664925692301, 42.219207443734199], [-86.383584656579302, 42.219309680769292], [-86.381434377554498, 42.222048444706502], [-86.380406604728691, 42.223357496749799], [-86.380035478688299, 42.223830191969389], [-86.364880140390099, 42.2423133218929898], [-86.356 208654, 42.69209], [-86.206834, 42.719424], [-86.208309, 42.762789], [-86.20885587580397, 42.767539639175901], [-86.210863, 42.783832], [-86.211815, 42.833236], [-86.210737, 42.859128], [-86.214138, 42.883555], [-86.216209, 42.919007], [-86.226305, 42 95, 43.253384544428002], [-86.357995846550196, 43.2587034783378], [-86.382882148394501, 43.296619782103697], [-86.383821353127203, 43.298050736858094], [-86.386095785812699, 43.3015160199338], [-86.386374420603389, 43.301940542686594], [-86.388492748905293, 4276401], [-86.438874990629998, 43.406427473980202], [-86.44847916024419, 43.413328922929801], [-86.448743, 43.432013], [-86.4545696785860588, 43.449474976590096], [-86.45608291238959, 43.4540098798790297], [-86.456844967858999, 43.456293792499004], [-86.45731902], [-86.50938653662098, 43.5604212681653], [-86.51468269505902, 43.567664452932604], [-86.515108012683399, 43.569073826615394], [-86.515838220875096, 43.5701689517754097], [-86.51629517726594, 43.5708542686962 44593], [-86.540639913228802, 43.6454995287093], [-86.538763379835203, 43.657061309579198], [-86.538482, 43.658795], [-86.538185937686805, 43.659402654929401], [-86.529685607962094, 43.676849209348894], [-86.529179, 43.677889], [-86.527666579021201, 43.67897 1797, 43.726436100536901], [-86.480267455452704, 43.7268192381771], [-86.464363610635302, 43.744686783537404], [-86.461554, 43.746685], [-86.460444124697801, 43.74836551778002], [-86.445531686740099, 43.770945196123897], [-86.445123, 43.771564], [-86.44495 70518606498], [-86.445545, 43.889726], [-86.4457300326862, 43.892897037430501], [-86.445937542987807, 43.895289563725299], [-86.446463099216899, 43.901349056134094], [-86.447915, 43.918089], [-86.4480091994118, 43.918416305978099], [-86.448147435503017, 43.991], [-86.5090515086967, 44.067408339188802], [-86.508764, 44.067881], [-86.508414691912904, 44.068205720765398], [-86.501242944680996, 44.074872658452001], [-86.500453, 44.075607], [-86.497936673468089, 44.07033231519302], [-86.469294741285395, 44.0932], [-86.3545921197951, 44.223811191314795], [-86.354312789646002, 44.224342389589303], [-86.351638, 44.229429], [-86.3492938280027, 44.2354587064033], [-86.343793, 44.249608], [-86.327287, 44.263057], [-86.326901730780193, 44.263780636992294], [-86.319063 3, 44.501682], [-86.233502662798301, 44.518277627563201], [-86.223788, 44.549043], [-86.220697, 44.566742], [-86.22545, 44.59459], [-86.231828, 44.609107], [-86.25395, 44.64808], [-86.259029, 44.663654], [-86.256796, 44.686769], [-86.254996, 44.691935], [-86.073506, 44.769803], [-86.073073003691007, 44.778392564279798], [-86.07112, 44.804717], [-86.065966, 44.821522], [-86.066031, 44.834852], [-86.071112, 44.854542], [-86.072468, 44.884788], [-86.07099, 44.895876], [-86.066745, 44.905685], [-86.051229], [-85.740836, 45.055575], [-85.712262, 45.065622], [-85.695715, 45.076461], [-85.681096, 45.092693], [-85.675671, 45.10554], [-85.674861, 45.116216], [-85.656024, 45.145788], [-85.618639, 45.186771], [-85.613174, 45.184624], [-85.611684, 45.11351], [-85.583198, 45.071304], [-85.573353, 45.068382], [-85.566066, 45.059201], [-85.56613, 45.043633], [-85.57016, 45.041278], [-85.573976, 45.043361], [-85.599652, 45.021749], [-85.609123, 45.013103], [-85.621878, 45.47550197269899], [-85.599873626083706, 44.765878307448098], [-85.599256, 44.765919], [-85.599103285054397, 44.765995935022694], [-85.593571, 44.768783], [-85.593474467435101, 44.769347823758196], [-85.59344868 4.90207295614095], [-85.556471038250109, 44.902223401852801], [-85.551566591436696, 44.906480736902097], [-85.539703, 44.916779], [-85.538945488578705, 44.9178854593655], [-85.538288497316699, 44.918845093919295], [-85.533553, 44.925762], [-85.5304769337954974021], [-85.464944, 44.961062], [-85.46665, 44.958844], [-85.470688275678199, 44.959237890298894], [-85.471260881896796, 44.959293741868294], [-85.472258, 44.959391], [-85.474274148444593, 44.958528880449201 , 44.896642716579], [-85.498007, 44.865451], [-85.500871692036, 44.858830302166396], [-85.502182, 44.855802], [-85.502385643553585, 44.8555510453156], [-85.50822729901891, 44.848352322747196], [-85.508167, 44.847872], [-85.51175148872011, 44.84711432654 82093, 44.808358918462986], [-85.560348196076887, 44.800772468497392], [-85.560840083712098, 44.804083989476597], [-85.560400870580091, 44.79594978422598], [-85.560423811086196, 44.794772402797797], [-85.5604887856442495, 44.778577036288297], [-85.502544, 44.781594], [-85.507112984217486, 44.784271307064799], [-85.509251, 44.787334], [-85.508461838085097, 44.788682274145401], [-85.5074725465607188, 44.790372467541005], [-85.1918], [-85.452683485397003, 44.8459688584894], [-85.442250259211306, 44.859816879816591], [-85.425804, 44.881646], [-85.423003, 44.895019], [-85.406173, 44.911773], [-85.3958, 44.931018], [-85.378286, 44.998587], [-85.381654, 45.018407], [-85.380659, 0.73617474147994, 45.364520319414497], [-85.066968996706407, 45.364950527399401], [-85.064250016316805, 45.36475839910393], [-85.060905191317687, 45.364522049243604], [-85.054805, 45.364091], [-85.046635252528489, 45 [-84.9900785, 45.425264], [-84.989223542308395, 45.4242252838737594], [-84.987143229049607, 45.422905679482398], [-84.984928327938391, 45.421471364473007], [-84.983836, 45.420764], [-84.97837303095087, 45.4201713654 10883783364699, 45.5262852433315], [-85.110954963638292, 45.52648844902799], [-85.115479, 45.539406], [-85.11578476852787, 45.5415330238366], [-85.115863985865954, 45.542125195786298], [-85.117406057446969, 45.552811223499099], [-85.117656012161291, 45.554 5012911795], [-85.10307665298908, 45.59265313008498], [-85.102916871351397, 45.593306403149696], [-85.101977291016198, 45.593550060965903], [-85.096798494926887, 45.5975761354713], [-85.095530705823094, 45.598561 5102275096], [-85.0153341, 45.651564], [-85.014352493990799, 45.652134159328895], [-85.013217751815091, 45.652788666060701], [-85.007026, 45.65636], [-85.0056454916995904, 45.6575038243043], [-85.001154, 45.661225], [-85.5337605502, 45.737326203814199], [-84.954274947478694, 45.738536621495896], [-84.955393748019006, 45.739071967270199], [-84.958129645013898, 45.740381093287098], [-84.968308937796095, 45.745251882297197], [-84.982328, 45.75196], [-84.982973299896798, 45.75226], [-84.92530082037501, 45.757021224480006], [-84.924664, 45.756897], [-84.922478451603396, 45.7558419129534], [-84.920691321845695, 45.754979164976298], [-84.919959528158698, 45.754643459359897], [-84.910398, 45.75001], [-84.866976, 45.752066], [-84.787224, 45.78818363595393], [-84.774014066796497, 45.788957291502896], [-84.77265, 45.782733], [-84.751571, 45.782733], [-84.742, 45.784134], [-84.734065, 45.788205], [-84.726192, 45.788205], [-84.157121, 45.585305], [-84.139462, 45.573714], [-84.128867, 45.562284], [-84.126532, 45.556616], [-84.126971, 45.542428], [-84.122309, 45.523788], [-84.116687, 45.51305], [-84.109238, 45.505171], [-84.095905, 45.497298], [-84.075792, 45.490537], [-83.541717, 45.34646], [-83.496704, 45.357536], [-83.488266, 45.355872], [-83.477794, 45.341891], [-83.468099068997693, 45.332450499817902], [-83.445672, 45.310612], [-83.43304, 45.303688], [-83.42514, 45.292875], [-83.363678, 45.166469], [-83.359895, 45.16302], [-

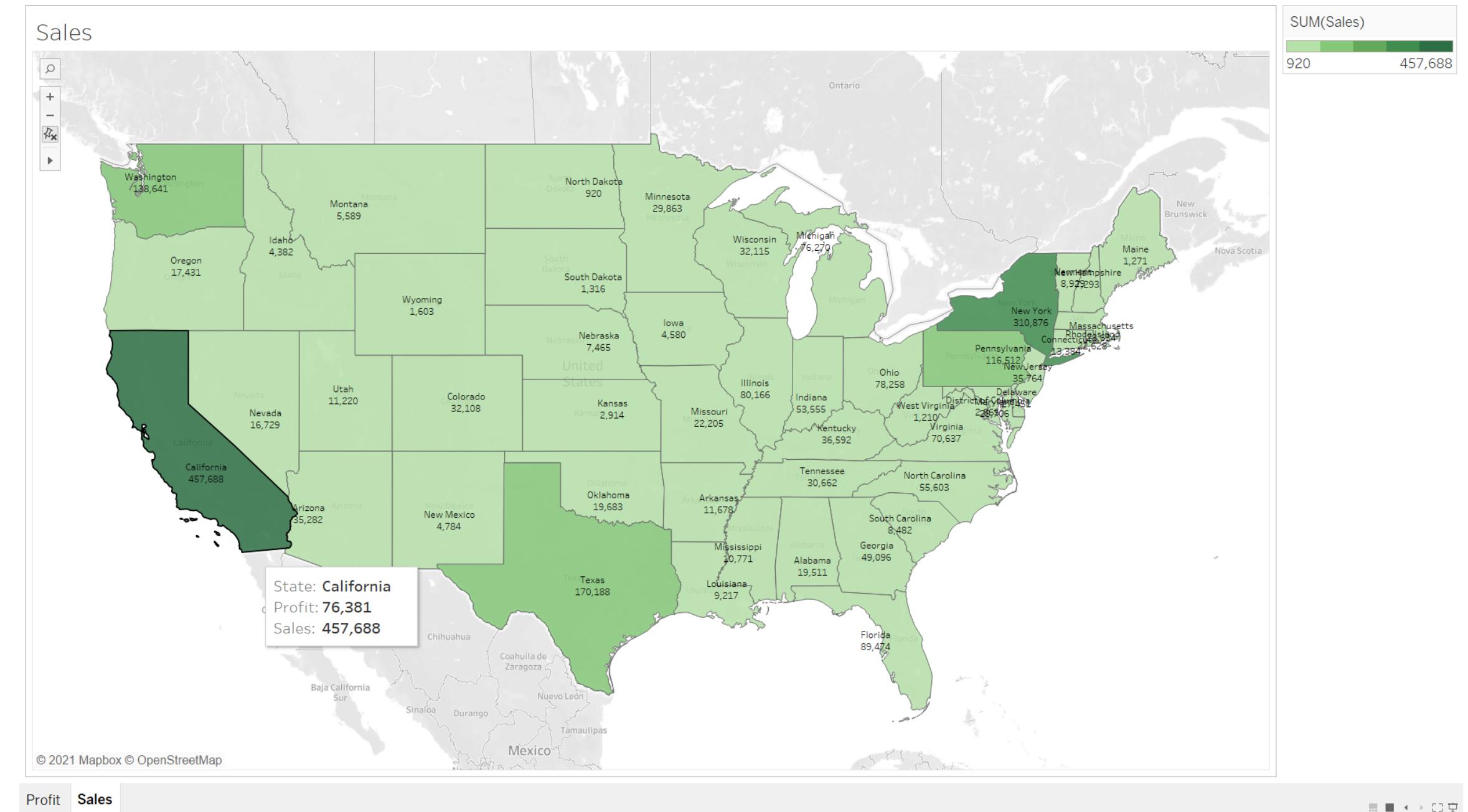
Map Visualization Using GeoPandas + Tableau

States based on Profit



Map Visualization Using GeoPandas + Tableau

States based on Sales



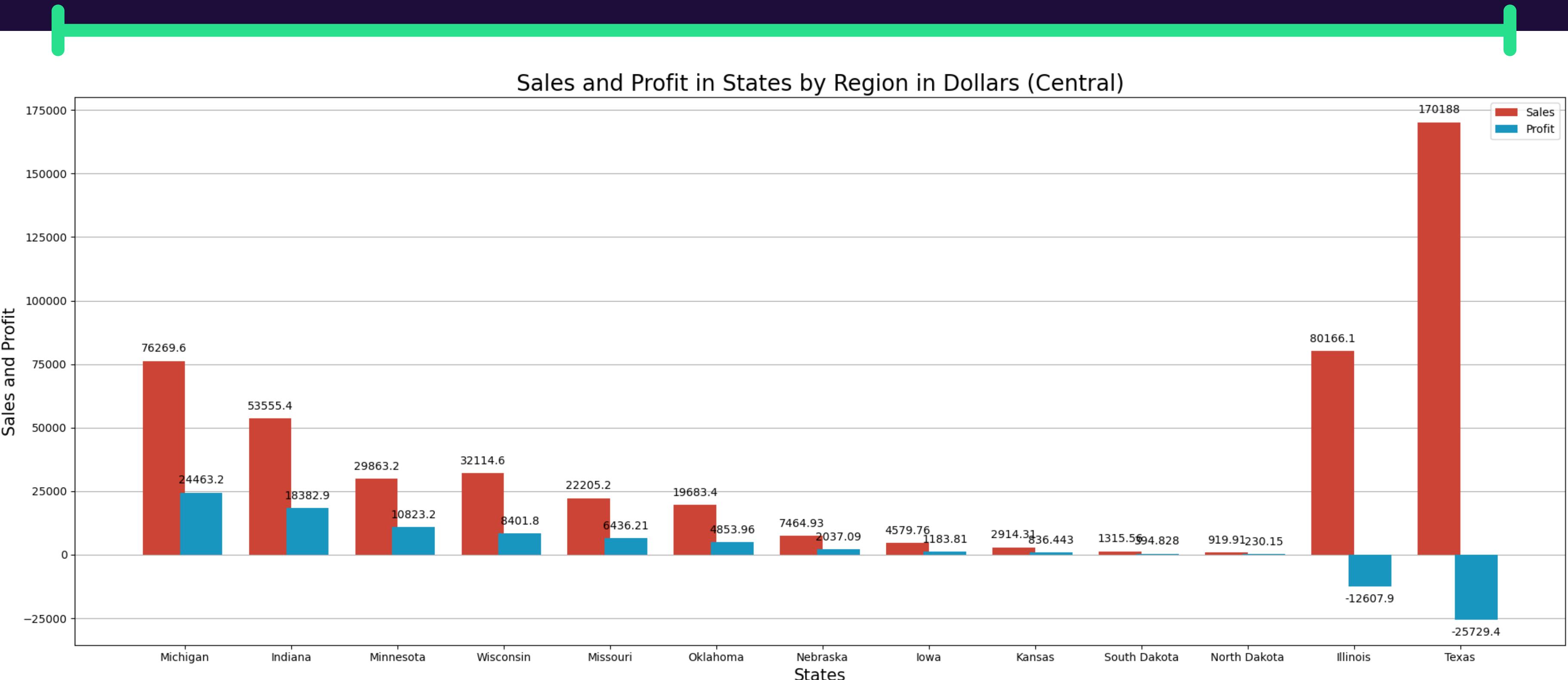
Creating Charts for Sales and Profit in each State by Region using Matplotlib

```
for region_df in regions:
    x = list(region_data[region_df]["State"])
    x_labels = np.arange(len(x))
    y1 = np.array(region_data[region_df]["Sales"])
    y2 = np.array(region_data[region_df]["Profit"])

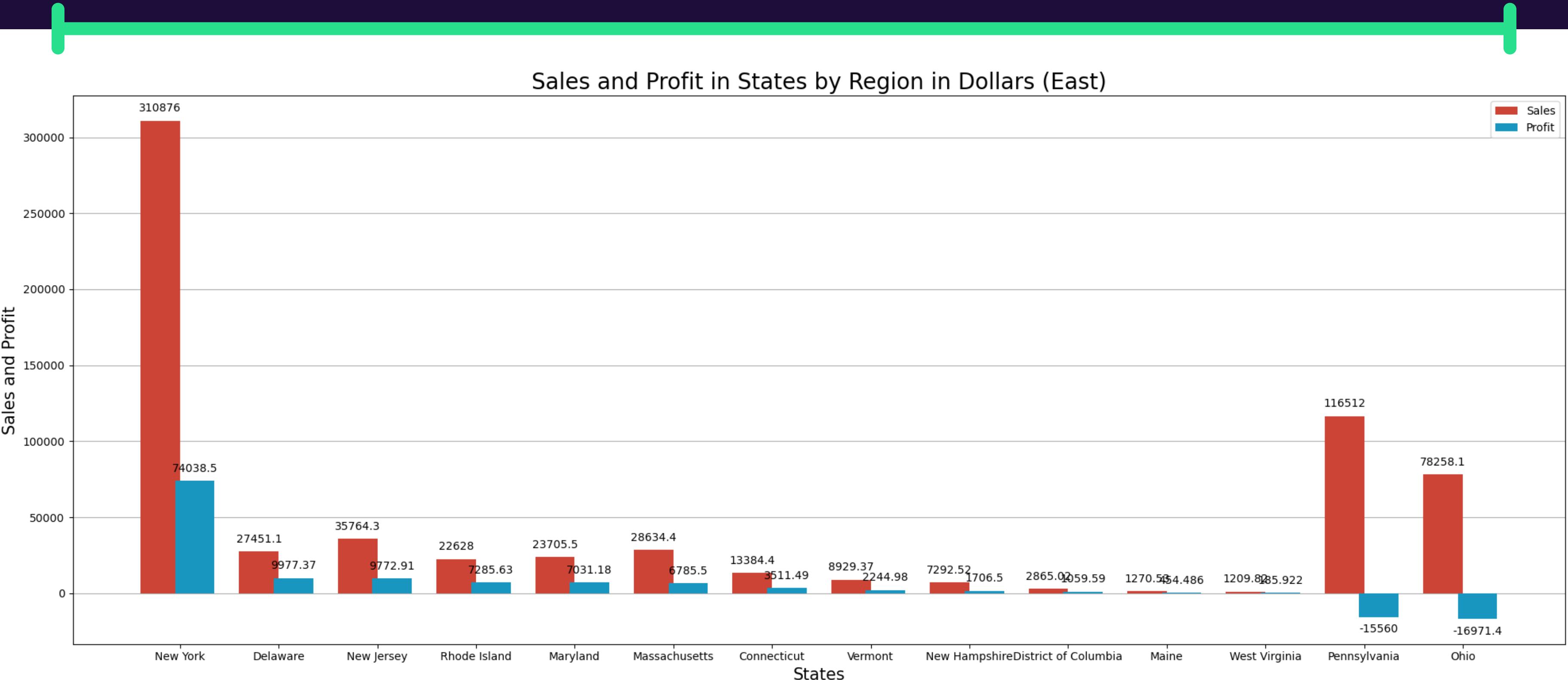
    fig, ax = plt.subplots(figsize=(20,8))
    sales_bar = ax.bar(x_labels - 0.2, y1, 0.4, label = 'Sales',color = "#cc4435",zorder = 3)
    profit_bar = ax.bar(x_labels + 0.15, y2, 0.4, label = 'Profit', color = "#1996bf", zorder = 3)

    ax.set_ylabel("Sales and Profit",fontsize = 15)
    ax.set_xlabel("States",fontsize = 15)
    ax.set_title("Sales and Profit in States by Region in Dollars {}".format(region_df),fontsize = 20)
    ax.set_xticks(x_labels)
    ax.set_xticklabels(x)
    ax.legend()
    ax.bar_label(sales_bar, padding=6)
    ax.bar_label(profit_bar, padding=6)
    ax.grid(axis = 'y',zorder = 0)
    fig.tight_layout()
    fig.savefig("sales_profit_charts/{}_sales_profit_chart.png".format(region_df.lower()))
```

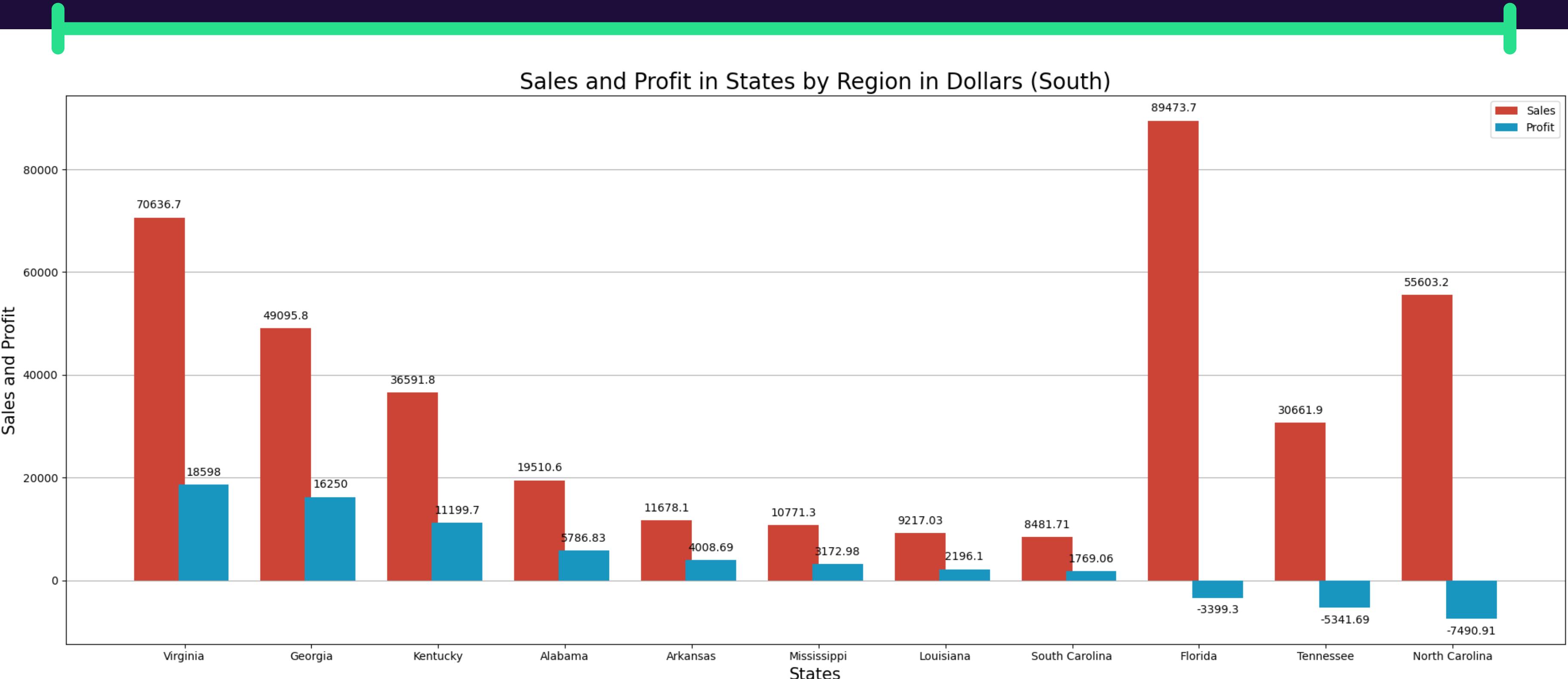
\sales_profit_charts\central_sales_profit_chart.png



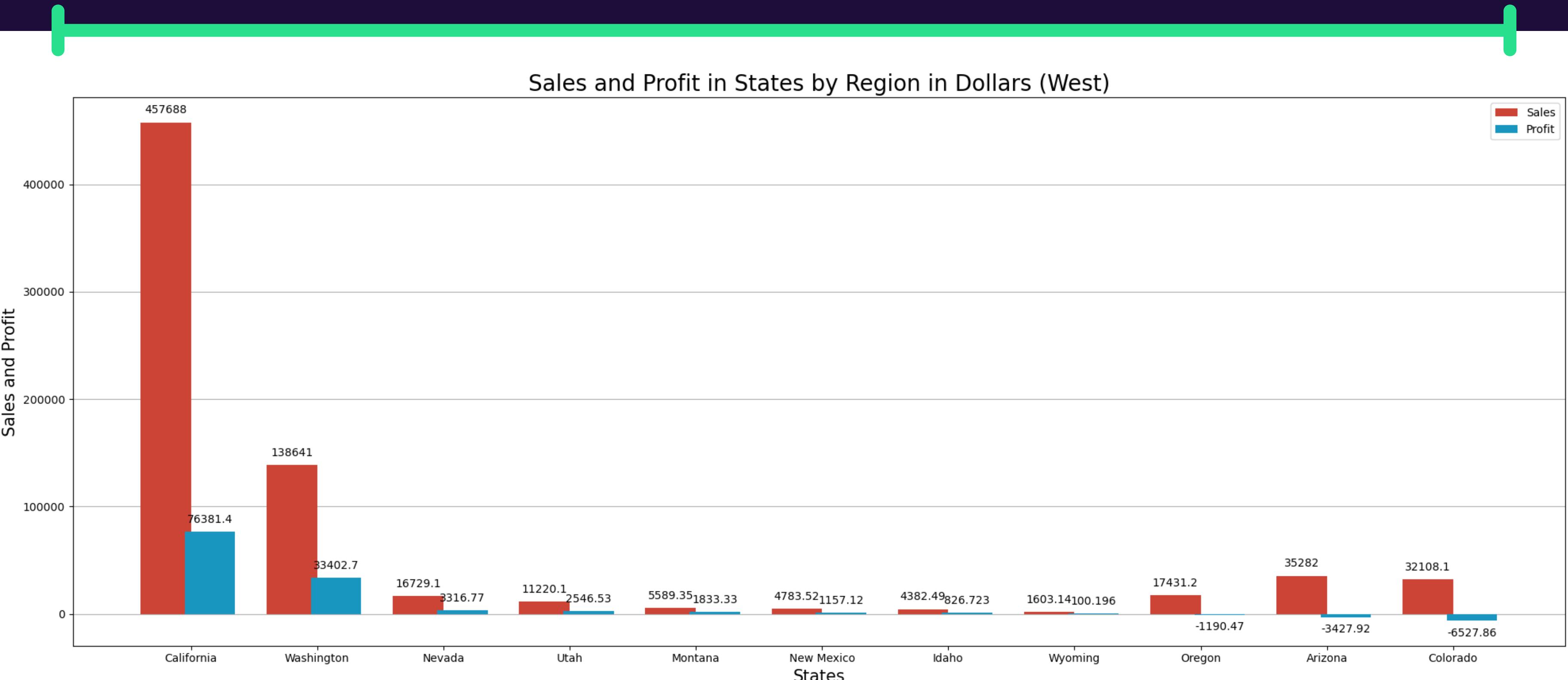
\sales_profit_charts\east_sales_profit_chart.png



\sales_profit_charts\south_sales_profit_chart.png



\sales_profit_charts\west_sales_profit_chart.png



Creating Charts for Profit Margin Ratio in each State by Region

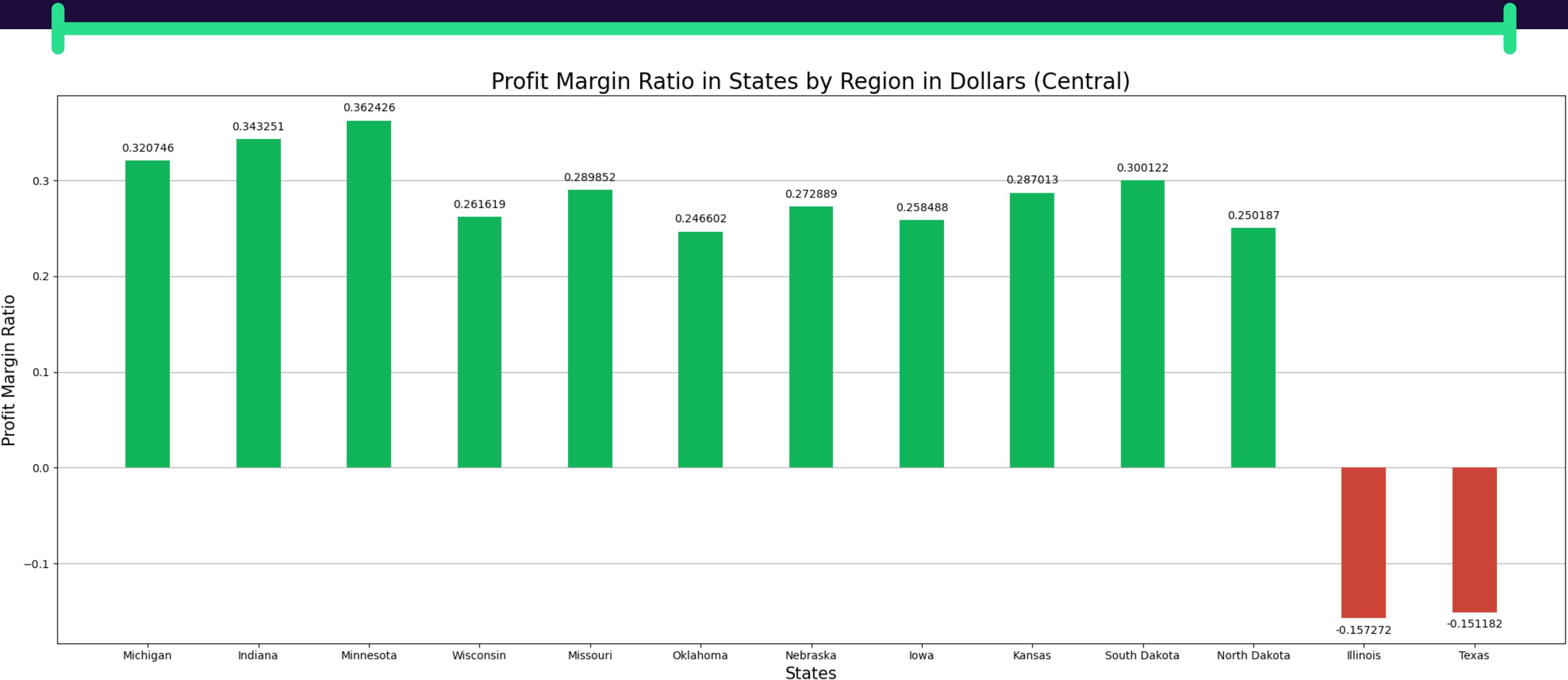
```
def get_ratio(r):
    return r["Profit"]/r["Sales"]

for region_df in regions:
    region_data[region_df]["Profit Margin Ratio"] = region_data[region_df].apply(get_ratio, axis=1)
    clrs = ["#cc4435" if (v < 0) else "#10b55a" for v in region_data[region_df]["Profit Margin Ratio"]]
    x = list(region_data[region_df]["State"])
    x_labels = np.arange(len(x))
    y = np.array(region_data[region_df]["Profit Margin Ratio"])

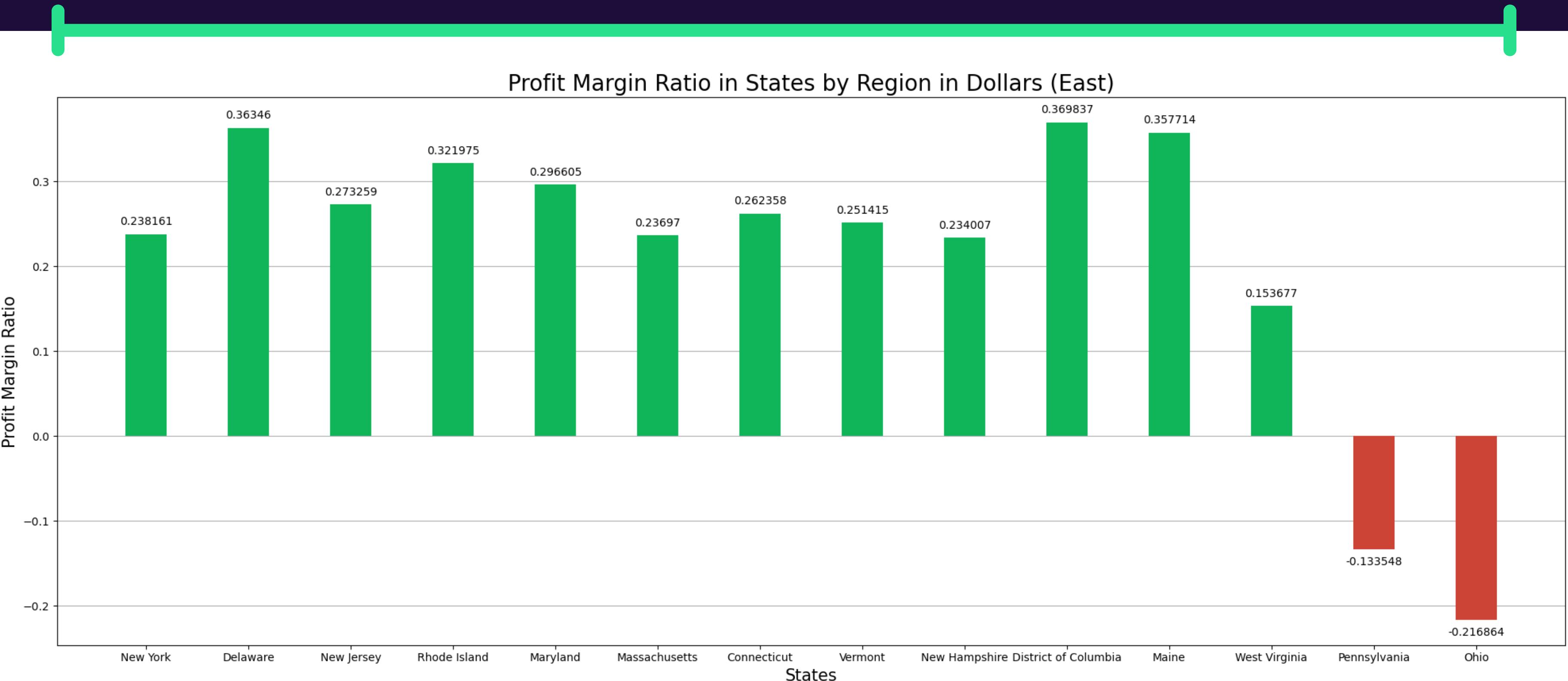
    fig, ax = plt.subplots(figsize=(20,8), dpi=100)
    sales_bar = ax.bar(x_labels, y, 0.4, label = 'Profit Margin Ratio', color = clrs, zorder = 3)

    ax.set_ylabel("Profit Margin Ratio", fontsize = 15)
    ax.set_xlabel("States", fontsize = 15)
    ax.set_title("Profit Margin Ratio in States by Region in Dollars {}".format(region_df), fontsize = 20)
    ax.set_xticks(x_labels)
    ax.set_xticklabels(x)
    ax.bar_label(sales_bar, padding=6)
    ax.grid(axis = 'y', zorder = 0)
    fig.tight_layout()
    plt.show()
    fig.savefig("profit_margin_charts/{}_profit_margin_chart.png".format(region_df.lower()))

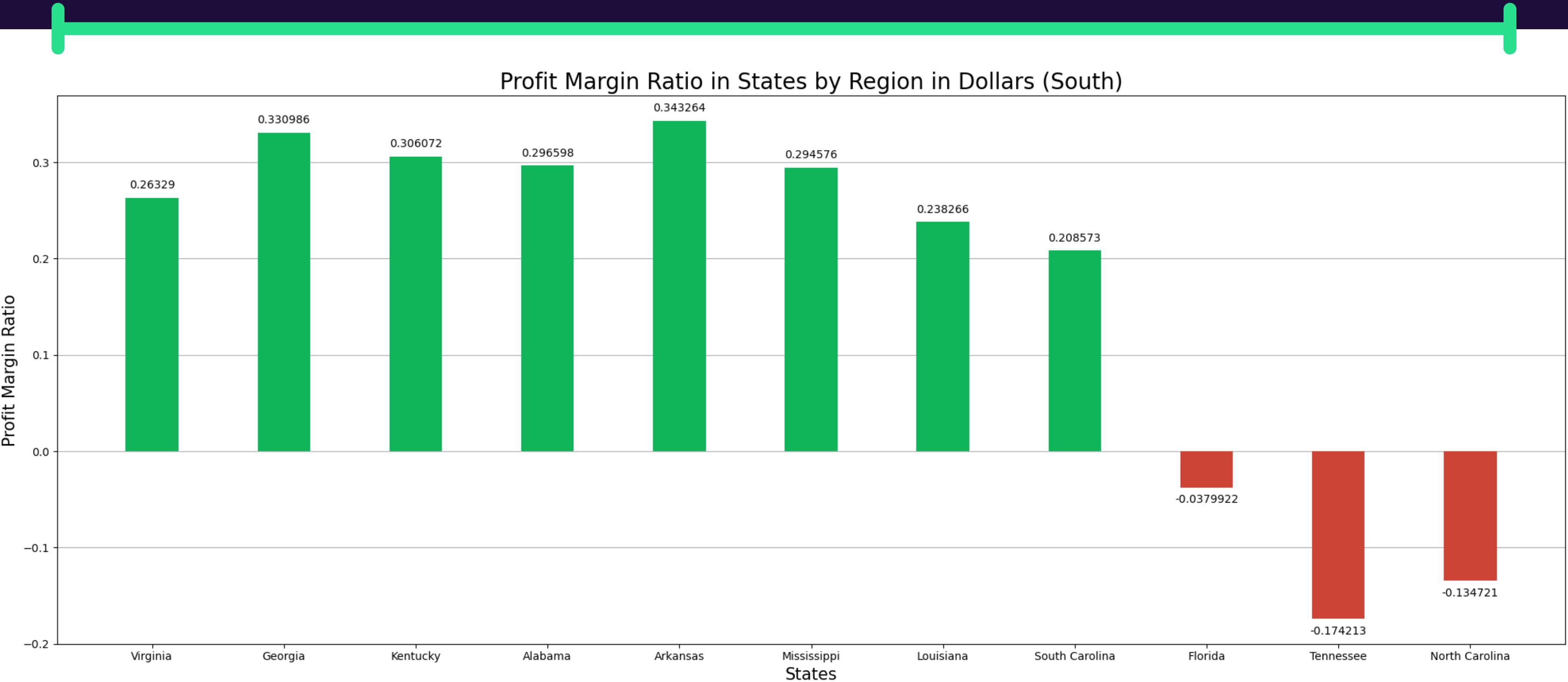
    region_data[region_df].drop(columns=["Profit Margin Ratio"], inplace = True)
```



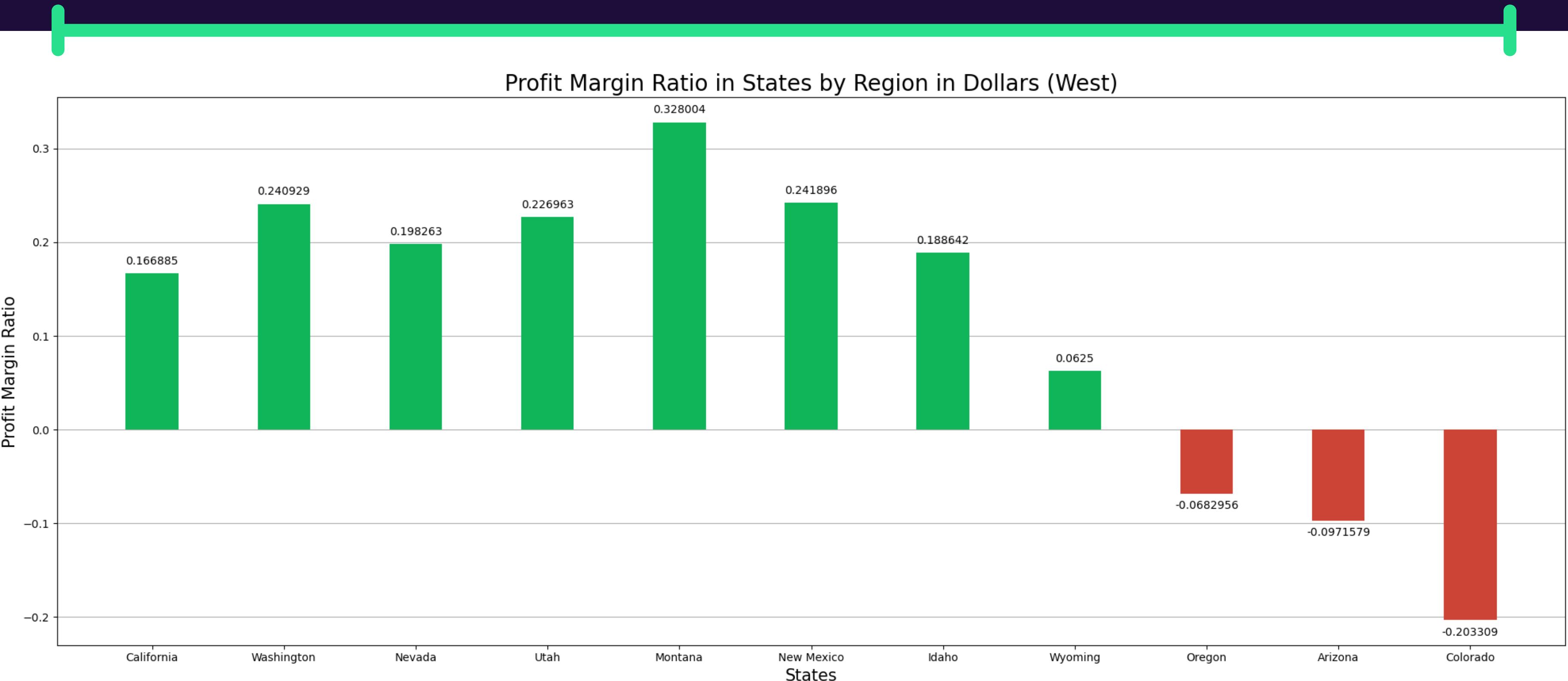
\profit_margin_charts\east_profit_margin_chart.png



\profit_margin_charts\south_profit_margin_chart.png



\profit_margin_charts\west_profit_margin_chart.png



Creating Charts for Profit Based on Products

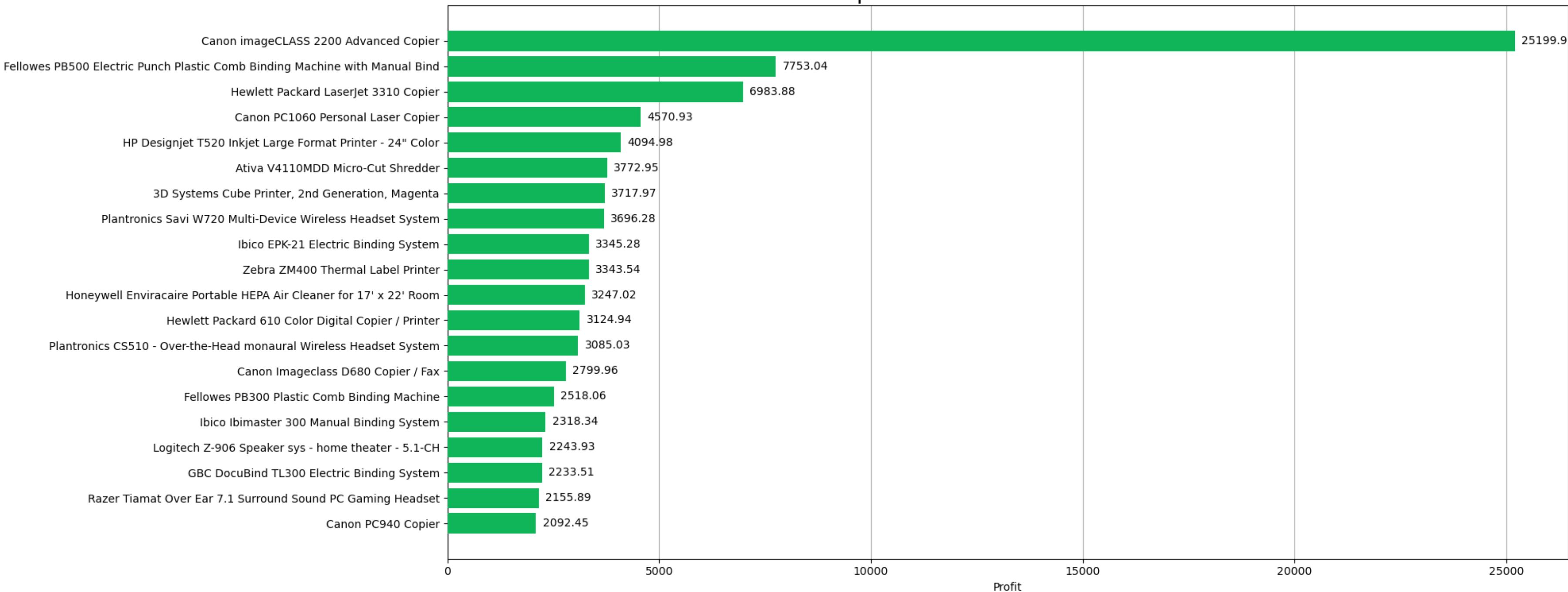
```
producttop = pd.DataFrame(dataset.groupby("Product Name").sum()).sort_values(by="Profit", ascending=False).reset_index().drop(columns = ["Quantity", "Discount"])
productbottom = pd.DataFrame(dataset.groupby("Product Name").sum()).sort_values(by="Profit", ascending=False).reset_index().drop(columns = ["Quantity", "Discount"])

top_y_labels = list(producttop["Product Name"])
top_y = np.arange(len(top_y_labels))
top_x = np.array(producttop["Profit"])
fig, ax = plt.subplots(figsize=(20,8), dpi=100)
top_profit = ax.barh(top_y, top_x, align='center', color = "#10b55a", zorder = 3)
ax.set_yticks(top_y)
ax.set_yticklabels(top_y_labels)
ax.invert_yaxis() # Labels read top-to-bottom
ax.set_xlabel('Profit')
ax.bar_label(top_profit, padding=6)
ax.set_title('Top 20 Products Based on Profit', fontsize = 20)
ax.grid(axis = 'x', zorder = 0)
plt.tight_layout()
fig.savefig("product_profit_charts/top_products_chart.png")
plt.show()

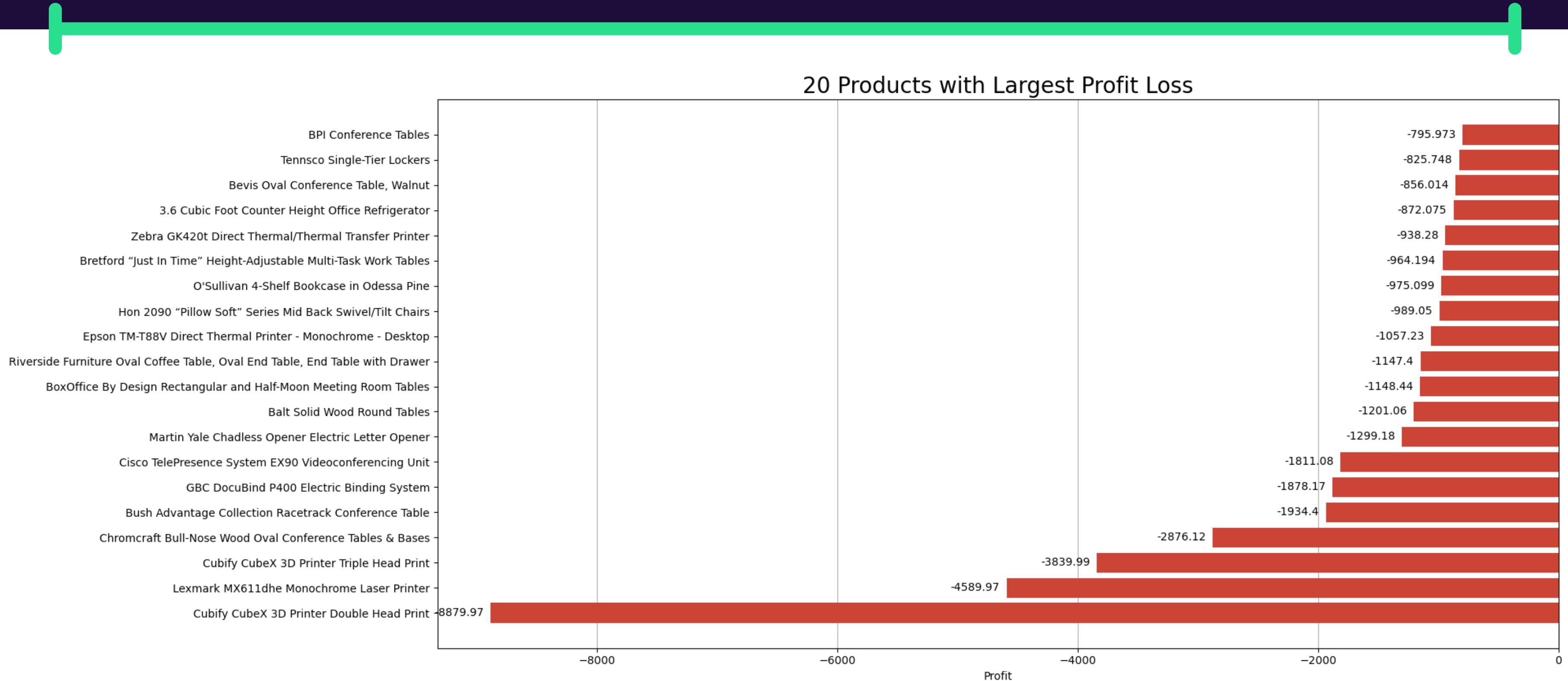
bottom_y_labels = list(productbottom["Product Name"])
bottom_y = np.arange(len(bottom_y_labels))
bottom_x = np.array(productbottom["Profit"])
fig, ax = plt.subplots(figsize=(20,8), dpi=100)
bottom_profit = ax.barh(bottom_y, bottom_x, align='center', color = "#cc4435", zorder = 3)
ax.set_yticks(bottom_y)
ax.set_yticklabels(bottom_y_labels)
ax.invert_yaxis() # Labels read top-to-bottom
ax.set_xlabel('Profit')
ax.bar_label(bottom_profit, padding=6)
ax.set_title('20 Products with Largest Profit Loss', fontsize = 20)
ax.grid(axis = 'x', zorder = 0)
plt.tight_layout()
fig.savefig("product_profit_charts/bottom_products_chart.png")
plt.show()
```

\product_profit_charts\top_products_chart.png

Top 20 Products Based on Profit



\product_profit_charts\bottom_products_chart.png



Creating Charts for Profit and Sales Based on Segments

```
segment_profits = pd.DataFrame(dataset.groupby("Segment").sum()).sort_values(by="Profit", ascending=False).reset_index().drop(columns = ["Quantity","Discount"])

labels = list(segment_profits["Segment"])
profits = list(segment_profits["Profit"])
sales = list(segment_profits["Sales"])
colors = ["#cc4435","#1996bf","#10b55a"]

fig, ax = plt.subplots(figsize=(10,10),dpi=100)
ax.pie(profits, labels=labels, autopct='%1.1f%%', startangle=90, colors=colors)
ax.axis('equal')
ax.set_title("Profits by Segment",fontsize = 20)
fig.savefig("segment_charts/profit_chart.png")
plt.show()

fig, ax = plt.subplots(figsize=(10,10),dpi=100)
ax.pie(sales, labels=labels, autopct='%1.1f%%', startangle=90, colors=colors)
ax.axis('equal')
ax.set_title("Sales by Segment",fontsize = 20)
fig.savefig("segment_charts/sales_chart.png")
plt.show()
```

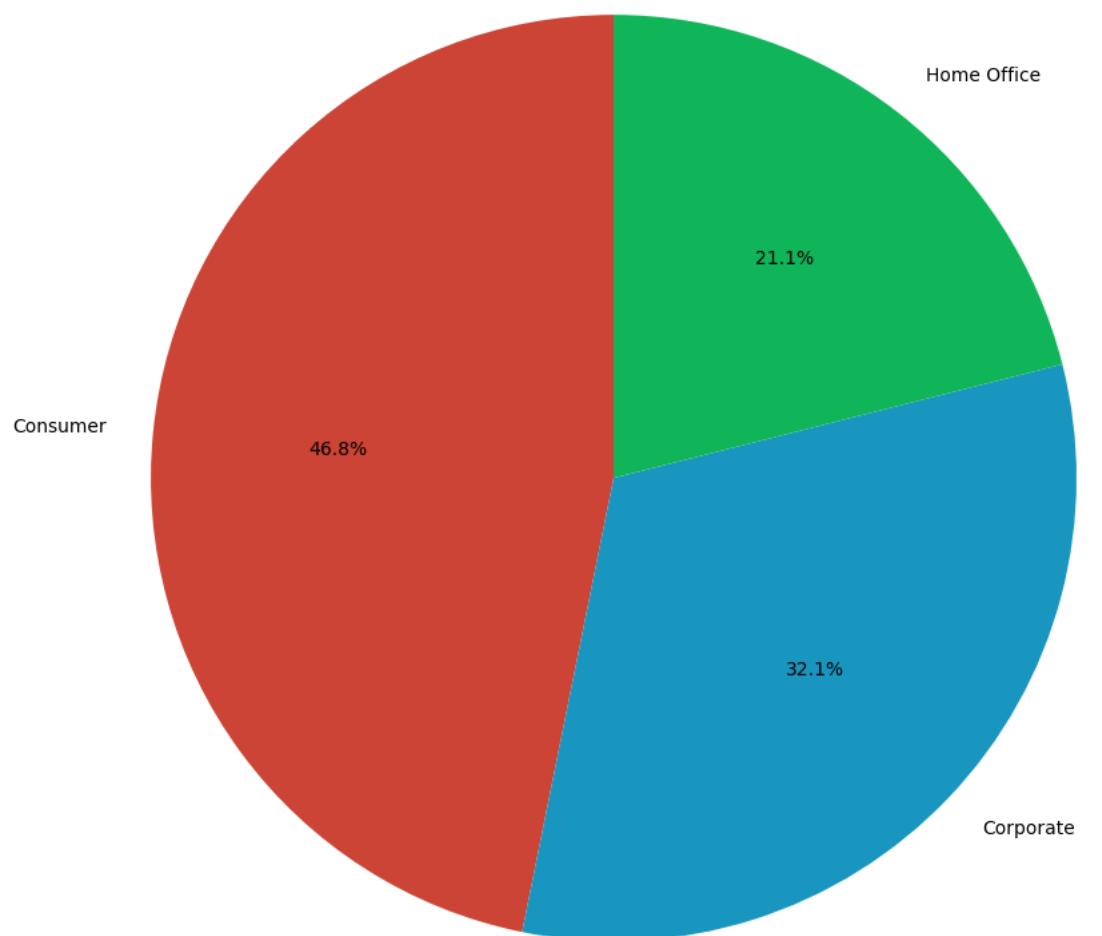
\segment_charts\profit_chart.png



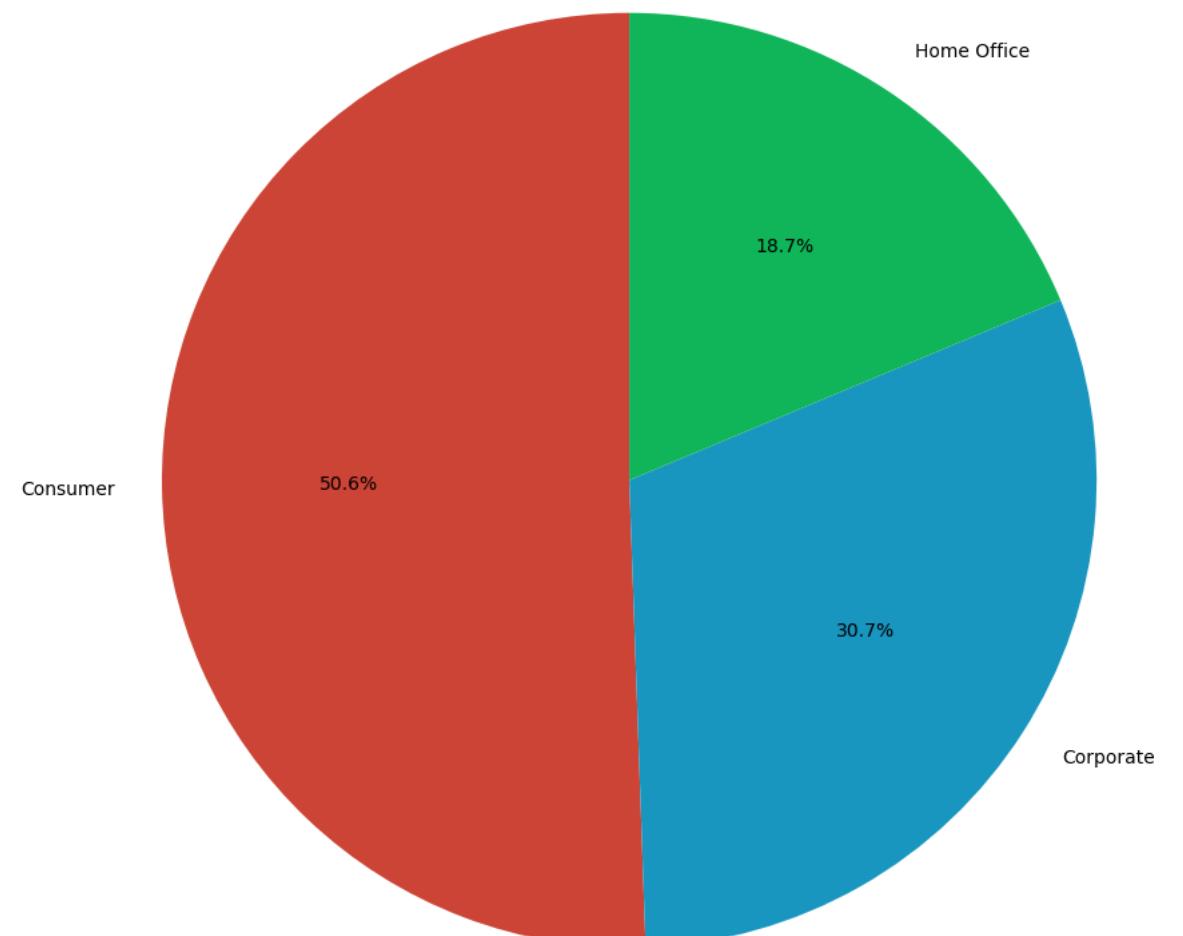
\segment_charts\sales_chart.png



Profits by Segment



Sales by Segment



*\FINAL_PROJECT (PRODUCTS CATEGORIES AND
SUBCATEGORIES).ipnyb*



Creating Charts for Sales Based on Segments by Region



```
#Isolating necessary columns for getting sales by segment per region
df1 = dataset[['Region','Segment','Sales']]
df1 = df1.sort_values(["Region","Segment"])

#Get Unique Regions and segments
df1["Region"].unique()
df1["Segment"].unique()

#Separate by Region
central_cond = df1["Region"] == "Central"
east_cond = df1["Region"] == "East"
south_cond = df1["Region"] == "South"
west_cond = df1["Region"] == "West"

central_df = df1.loc[central_cond,:]
east_df = df1.loc[east_cond,:]
south_df = df1.loc[south_cond,:]
west_df = df1.loc[west_cond,:]

#Getting sales and segment per region
agg_func = {'Region': 'first', 'Sales': 'sum'}
central_df = central_df.groupby(central_df['Segment']).aggregate(agg_func)
east_df = east_df.groupby(east_df['Segment']).aggregate(agg_func)
south_df = south_df.groupby(south_df['Segment']).aggregate(agg_func)
west_df = west_df.groupby(west_df['Segment']).aggregate(agg_func)

#Remove Index

central_df.reset_index(inplace=True,drop=False)
east_df.reset_index(inplace=True,drop=False)
south_df.reset_index(inplace=True,drop=False)
west_df.reset_index(inplace=True,drop=False)
```

Plotting Charts

```
#Plotting Pie Chart (Sales by segments in region)
import matplotlib.pyplot as plt
import numpy as np

#https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/
#http://www.learningaboutelectronics.com/Articles/How-to-create-a-pie-chart-in-matplotlib-with-Python.php

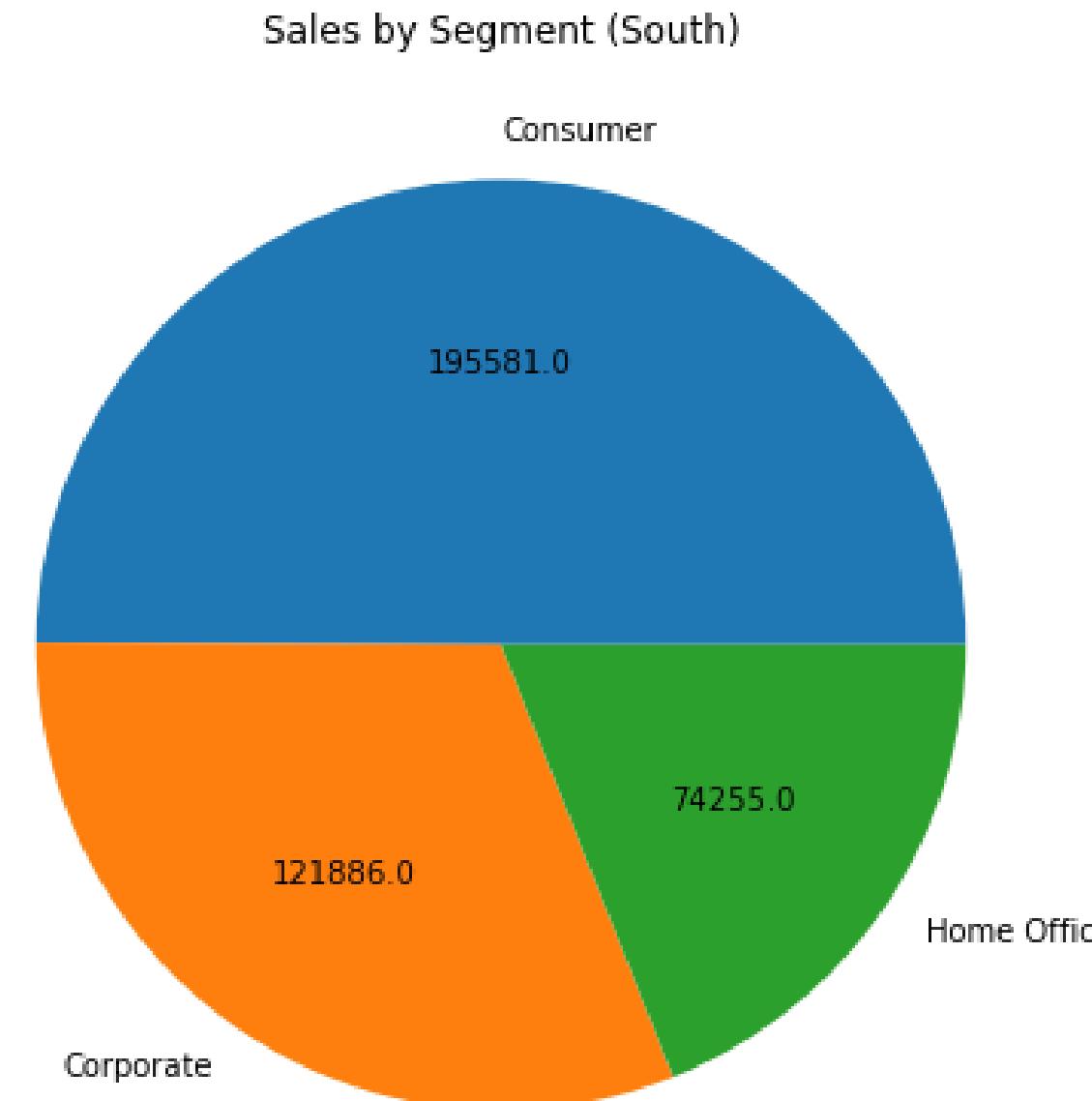
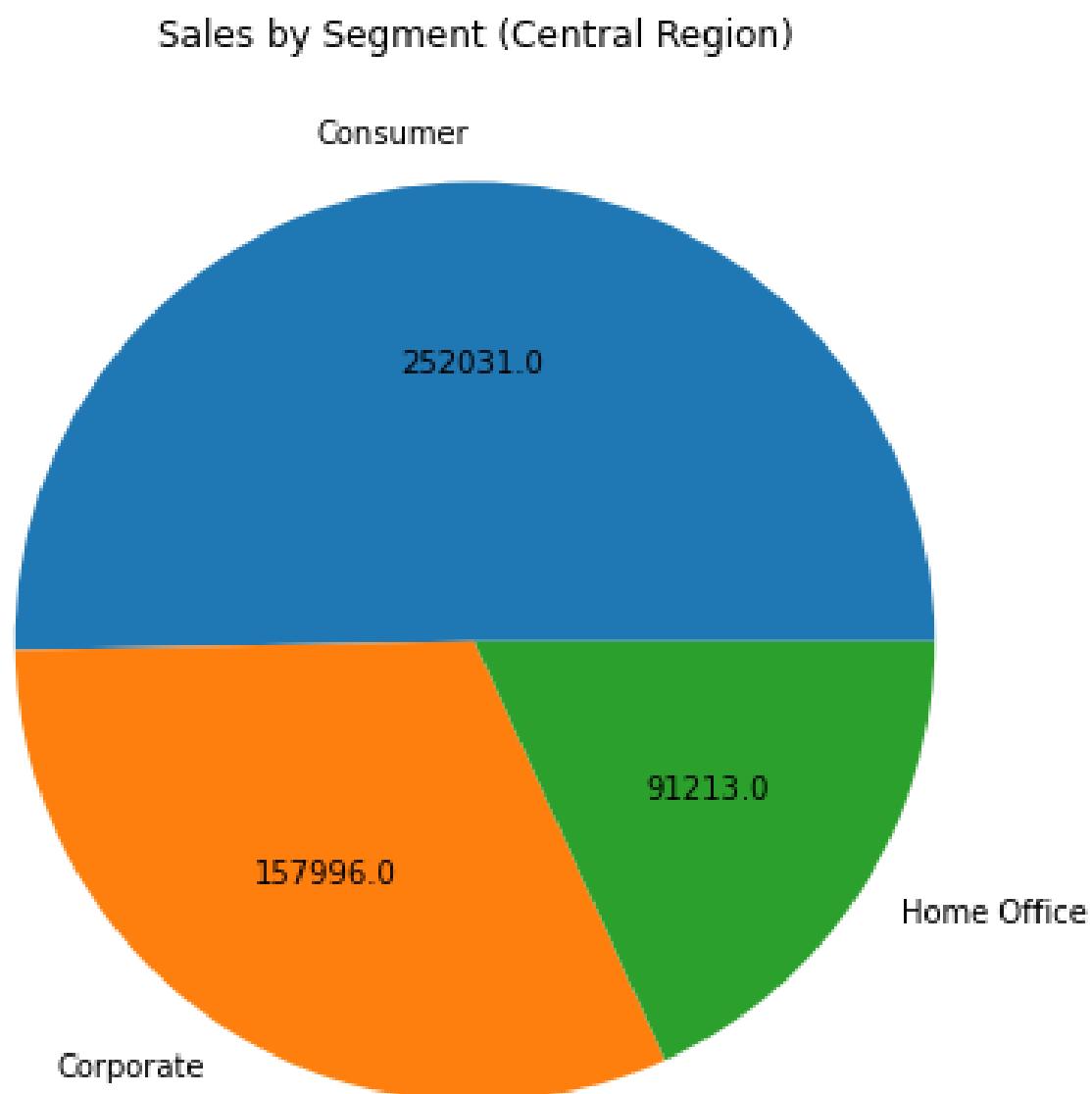
# Creating dataset
sub_cat = central_df["Segment"]
data = central_df["Sales"]

#https://stackoverflow.com/questions/41088236/how-to-have-actual-values-in-matplotlib-pie-chart-displayed-python
def absolute_value(val):
    a = np.round(val/100.*data.sum(), 0)
    return a

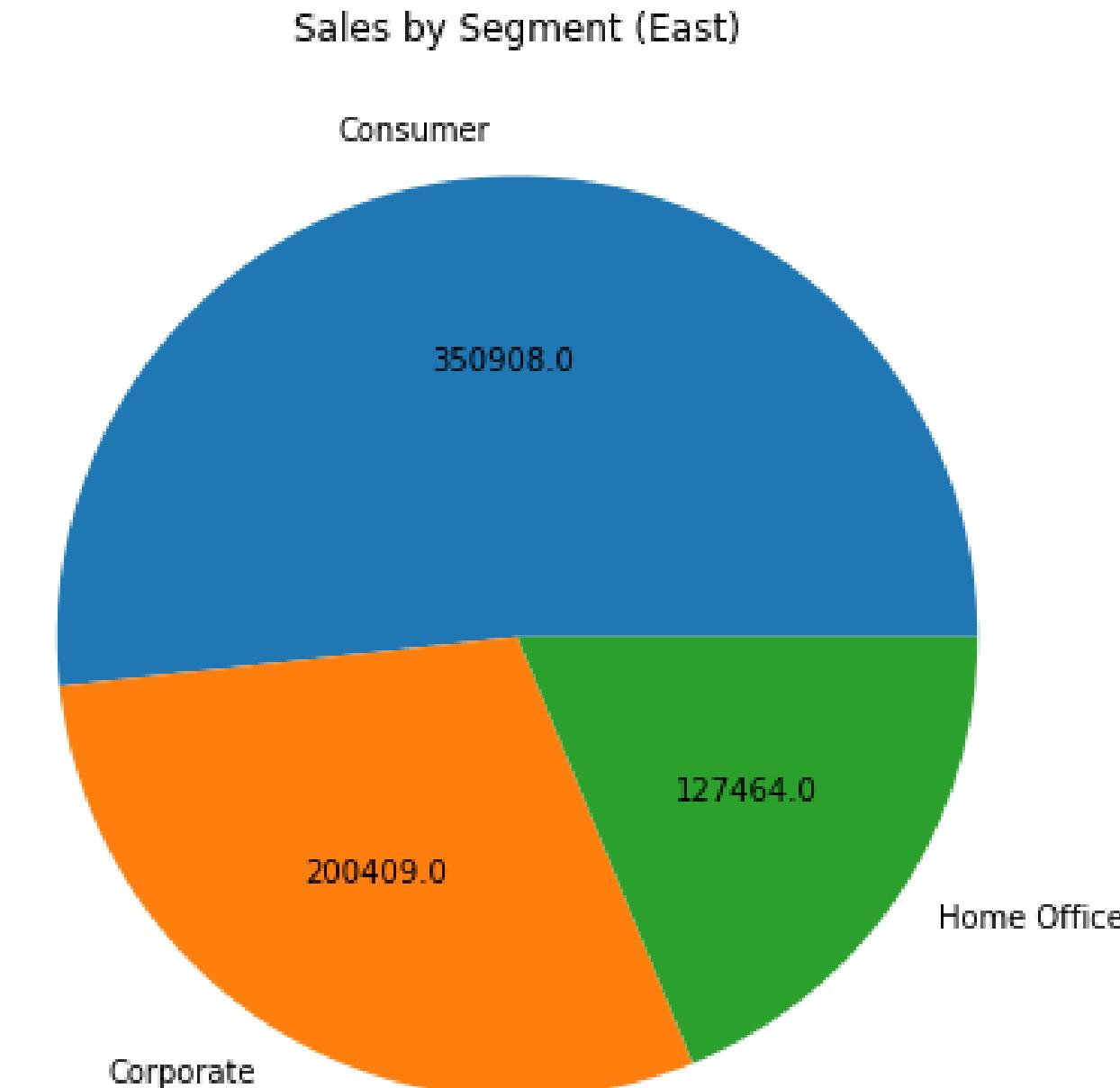
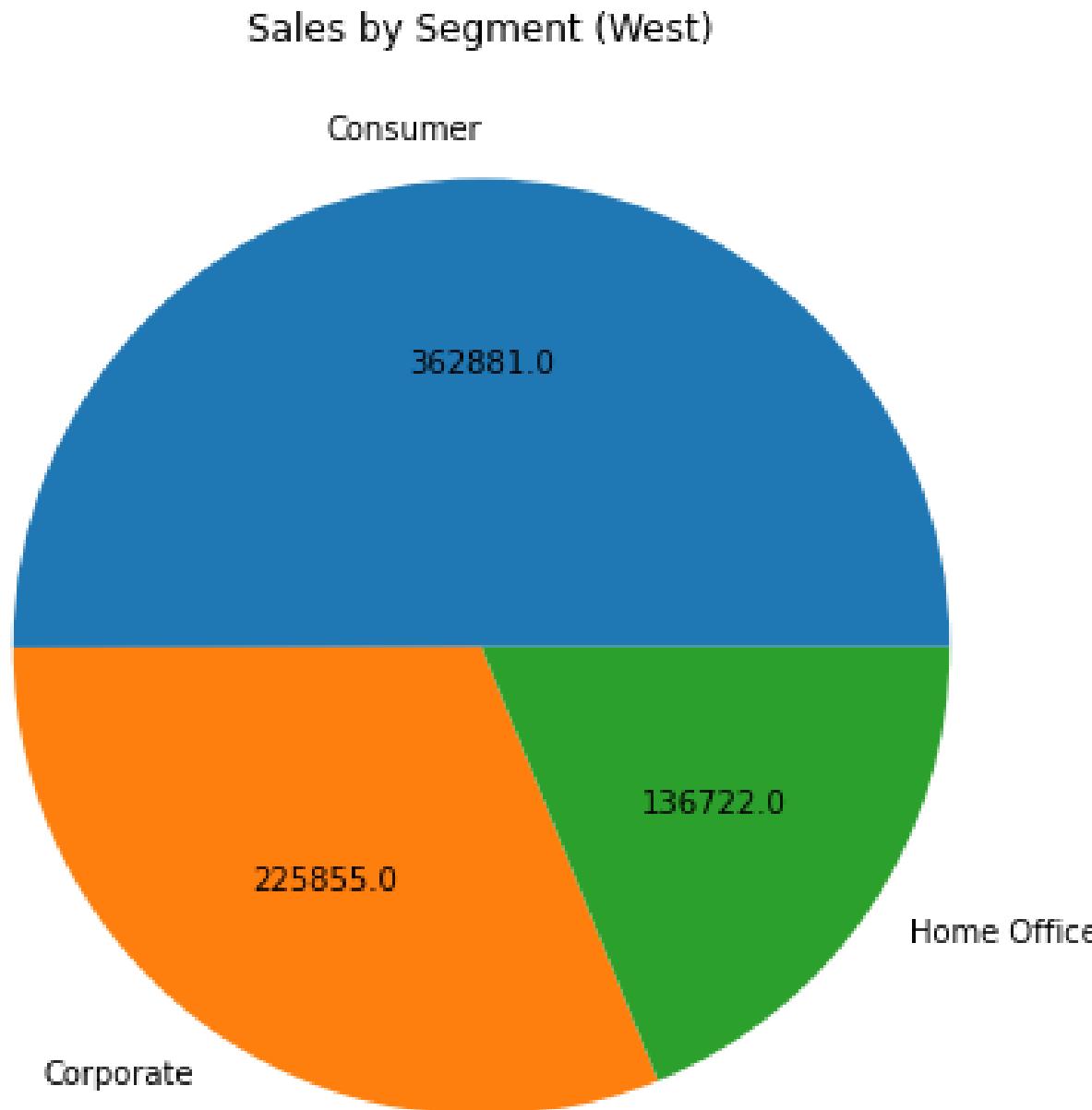
# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = sub_cat, autopct=absolute_value)
plt.title("Sales by Segment (Central Region)")

# show plot
fig.savefig("Sales by Segment (Central Region)")
plt.show()
```

Sales based on Segment by Region



Sales based on Segment by Region



Creating Graphs for Profits and Sales based on Product Sub-Category



```
#Isolating necessary columns for product wise sales
df1 = dataset[['Category','Sub-Category','Sales','Profit']]
df1 = df1.sort_values(["Category","Sub-Category"])

#Check for unique categories
df1["Category"].unique()

#Separate by unique category
furniture_cond = df1["Category"] == "Furniture"
office_cond = df1["Category"] == "Office Supplies"
tech_cond = df1["Category"] == "Technology"

furniture_df = df1.loc[furniture_cond,:]
office_df = df1.loc[office_cond,:]
tech_df = df1.loc[tech_cond,:]

#Getting sales and profit per subcategory
agg_func = {'Category': 'first', 'Sales': 'sum', 'Profit': 'sum'}
furniture_df = furniture_df.groupby(furniture_df['Sub-Category']).aggregate(agg_func)
office_df = office_df.groupby(office_df['Sub-Category']).aggregate(agg_func)
tech_df = tech_df.groupby(tech_df['Sub-Category']).aggregate(agg_func)
```

Creating Graphs for Profits and Sales based on Product Sub-Category



```
#Sort Sales and Profits in descending order

furniture_sales_df = furniture_df.sort_values(["Sales"], ascending=False)
furniture_profit_df = furniture_df.sort_values(["Profit"], ascending=False)
office_sales_df = office_df.sort_values(["Sales"], ascending=False)
office_profit_df = office_df.sort_values(["Profit"], ascending=False)
tech_sales_df = tech_df.sort_values(["Sales"], ascending=False)
tech_profit_df = tech_df.sort_values(["Profit"], ascending=False)

#Remove Index

furniture_sales_df.reset_index(inplace=True, drop=False)
furniture_profit_df.reset_index(inplace=True, drop=False)
office_sales_df.reset_index(inplace=True, drop=False)
office_profit_df.reset_index(inplace=True, drop=False)
tech_sales_df.reset_index(inplace=True, drop=False)
tech_profit_df.reset_index(inplace=True, drop=False)
```

Resulting DataFrames



	Sub-Category	Category	Sales	Profit
0	Paper	Office Supplies	78479.206	34053.5693
1	Binders	Office Supplies	203412.733	30221.7633
2	Storage	Office Supplies	223843.608	21278.8264
3	Appliances	Office Supplies	107532.161	18138.0054
4	Envelopes	Office Supplies	16476.402	6964.1767
5	Art	Office Supplies	27118.792	6527.7870
6	Labels	Office Supplies	12486.312	5546.2540
7	Fasteners	Office Supplies	3024.280	949.5182
8	Supplies	Office Supplies	46673.538	-1189.0995

	Sub-Category	Category	Sales	Profit
0	Storage	Office Supplies	223843.608	21278.8264
1	Binders	Office Supplies	203412.733	30221.7633
2	Appliances	Office Supplies	107532.161	18138.0054
3	Paper	Office Supplies	78479.206	34053.5693
4	Supplies	Office Supplies	46673.538	-1189.0995
5	Art	Office Supplies	27118.792	6527.7870
6	Envelopes	Office Supplies	16476.402	6964.1767
7	Labels	Office Supplies	12486.312	5546.2540
8	Fasteners	Office Supplies	3024.280	949.5182

Resulting DataFrames



	Sub-Category	Category	Sales	Profit
0	Copiers	Technology	149528.030	55617.8249
1	Phones	Technology	330007.054	44515.7306
2	Accessories	Technology	167380.318	41936.6357
3	Machines	Technology	189238.631	3384.7569

	Sub-Category	Category	Sales	Profit
0	Phones	Technology	330007.054	44515.7306
1	Machines	Technology	189238.631	3384.7569
2	Accessories	Technology	167380.318	41936.6357
3	Copiers	Technology	149528.030	55617.8249

Resulting DataFrames



	Sub-Category	Category	Sales	Profit
0	Chairs	Furniture	328449.1030	26590.1663
1	Furnishings	Furniture	91705.1640	13059.1436
2	Bookcases	Furniture	114879.9963	-3472.5560
3	Tables	Furniture	206965.5320	-17725.4811

	Sub-Category	Category	Sales	Profit
0	Chairs	Furniture	328449.1030	26590.1663
1	Tables	Furniture	206965.5320	-17725.4811
2	Bookcases	Furniture	114879.9963	-3472.5560
3	Furnishings	Furniture	91705.1640	13059.1436

Plotting Bar Graphs

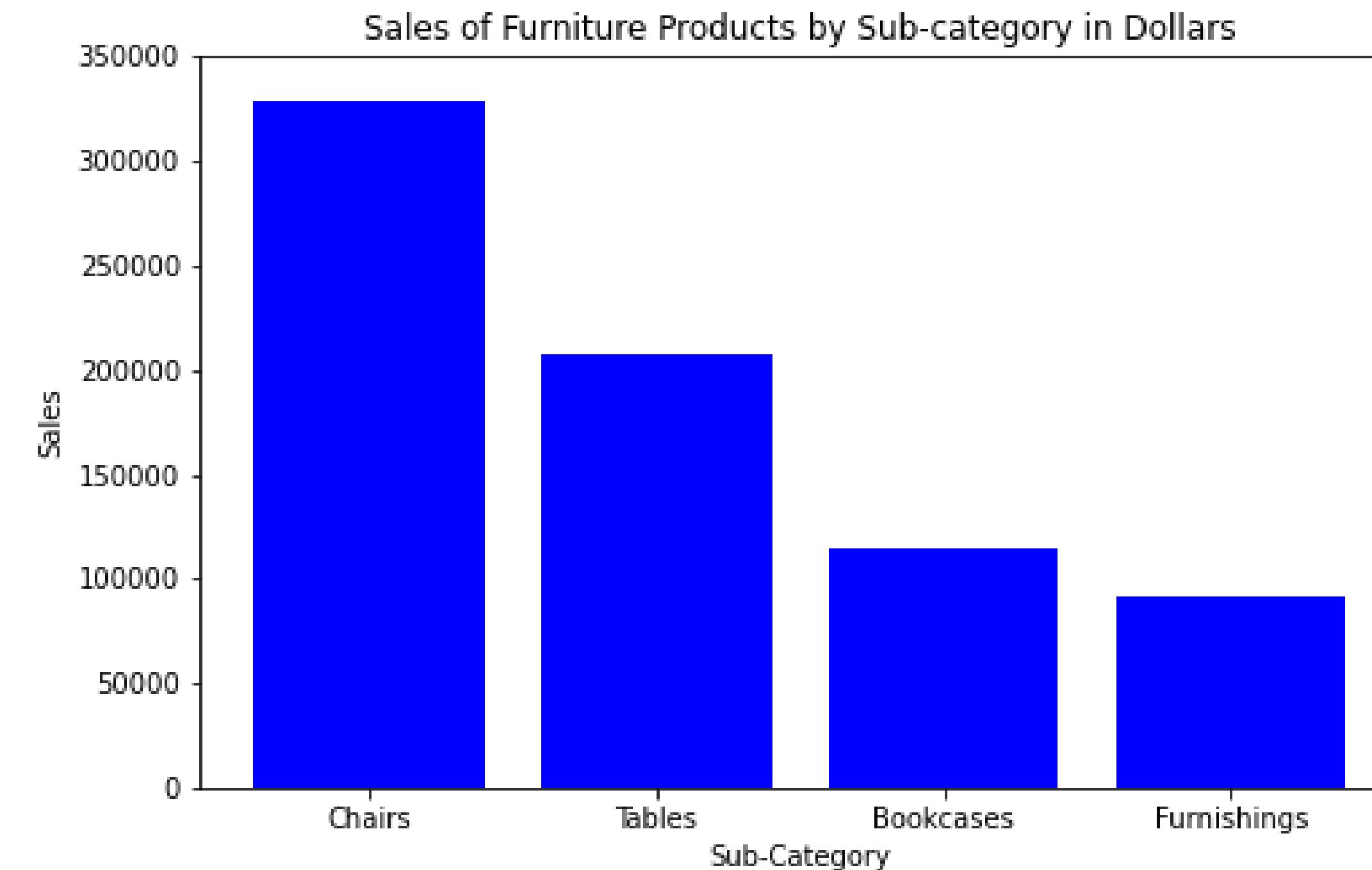
```
#Plotting Bar Graph (Furniture Sales)
import matplotlib.pyplot as plt
import numpy as np

#Getting x and y axis
X = list(furniture_sales_df["Sub-Category"])
Y = list(furniture_sales_df["Sales"])

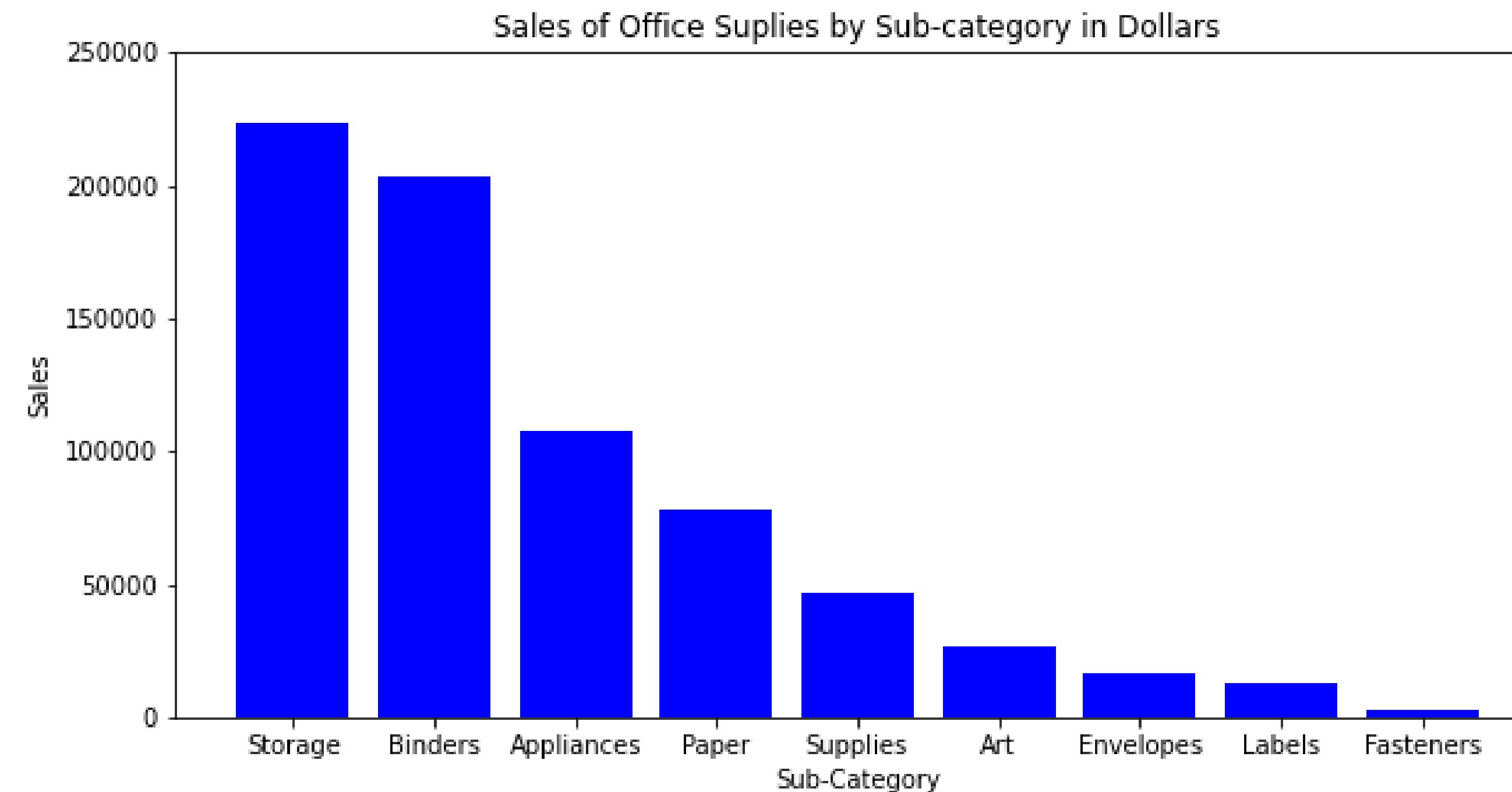
# Plotting data
f = plt.figure(figsize=(8,5))
plt.bar(X, Y, color='b')
plt.title("Sales of Furniture Products by Sub-category in Dollars")
plt.xlabel("Sub-Category")
plt.ylabel("Sales")
y_ticks = np.arange(0,400000,50000)
plt.yticks(y_ticks)

f.savefig("Sales of Furniture Products by Sub-category in Dollars.png")
plt.show()
```

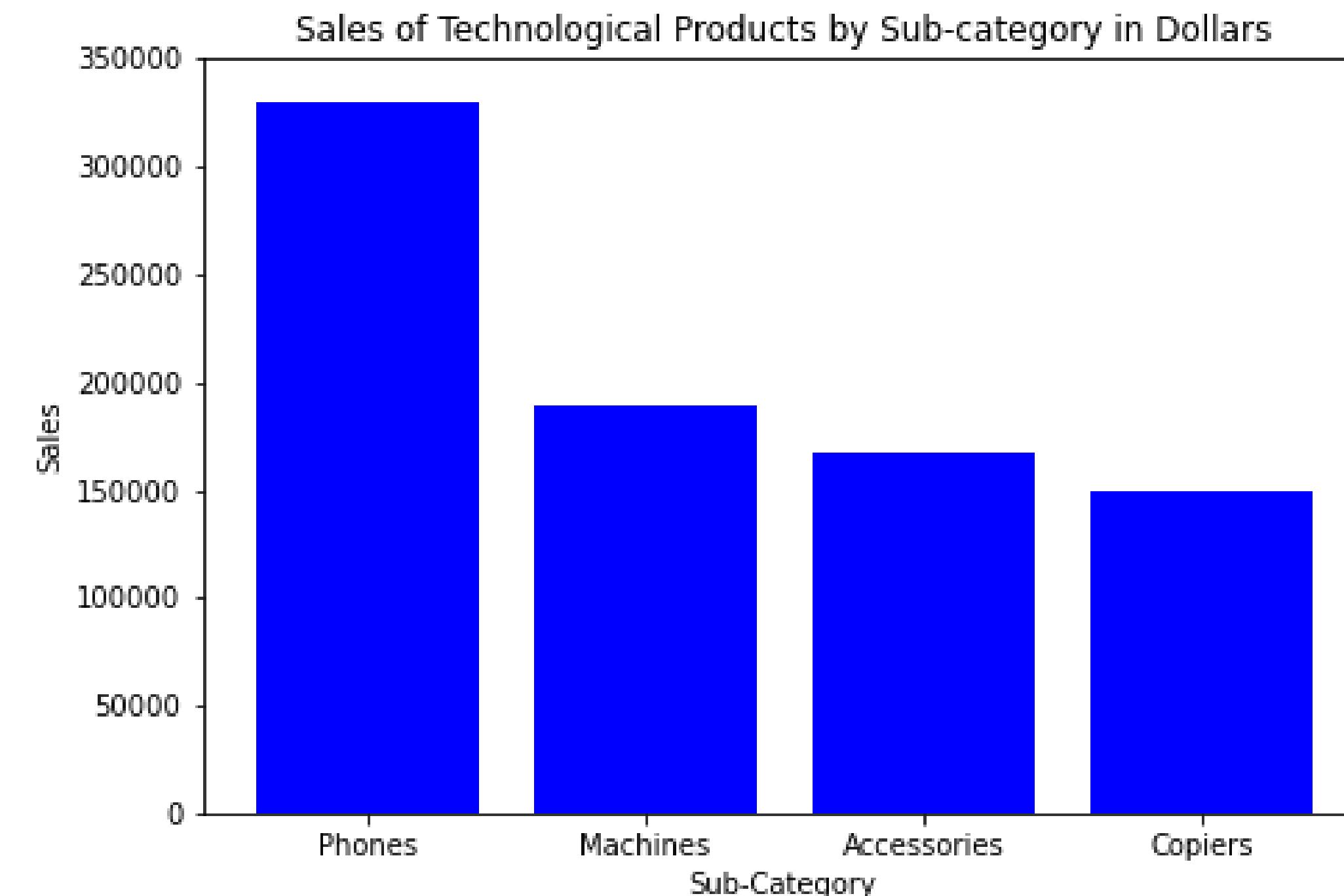
Sales based on Product Sub-Category



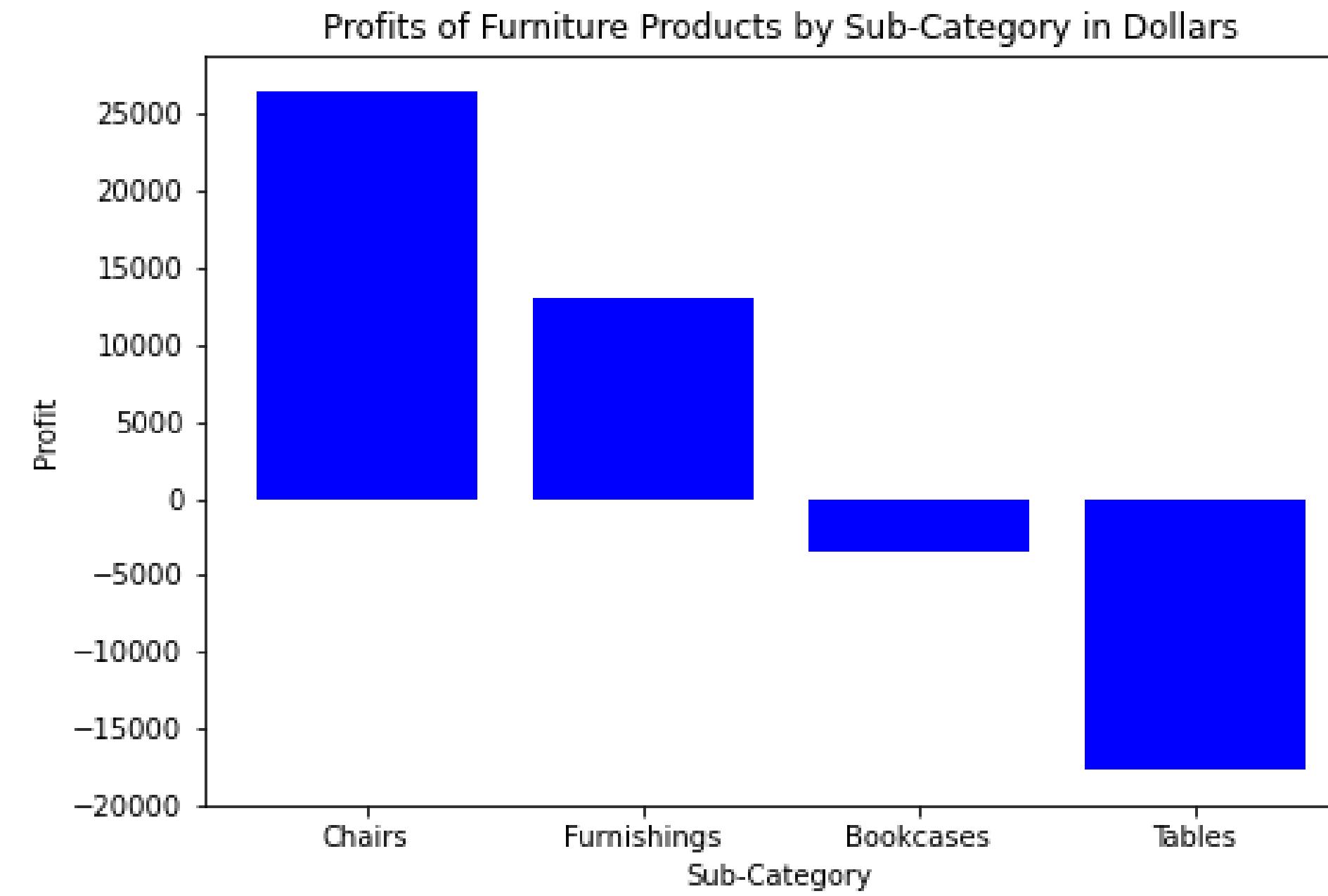
Sales based on Product Sub-Category



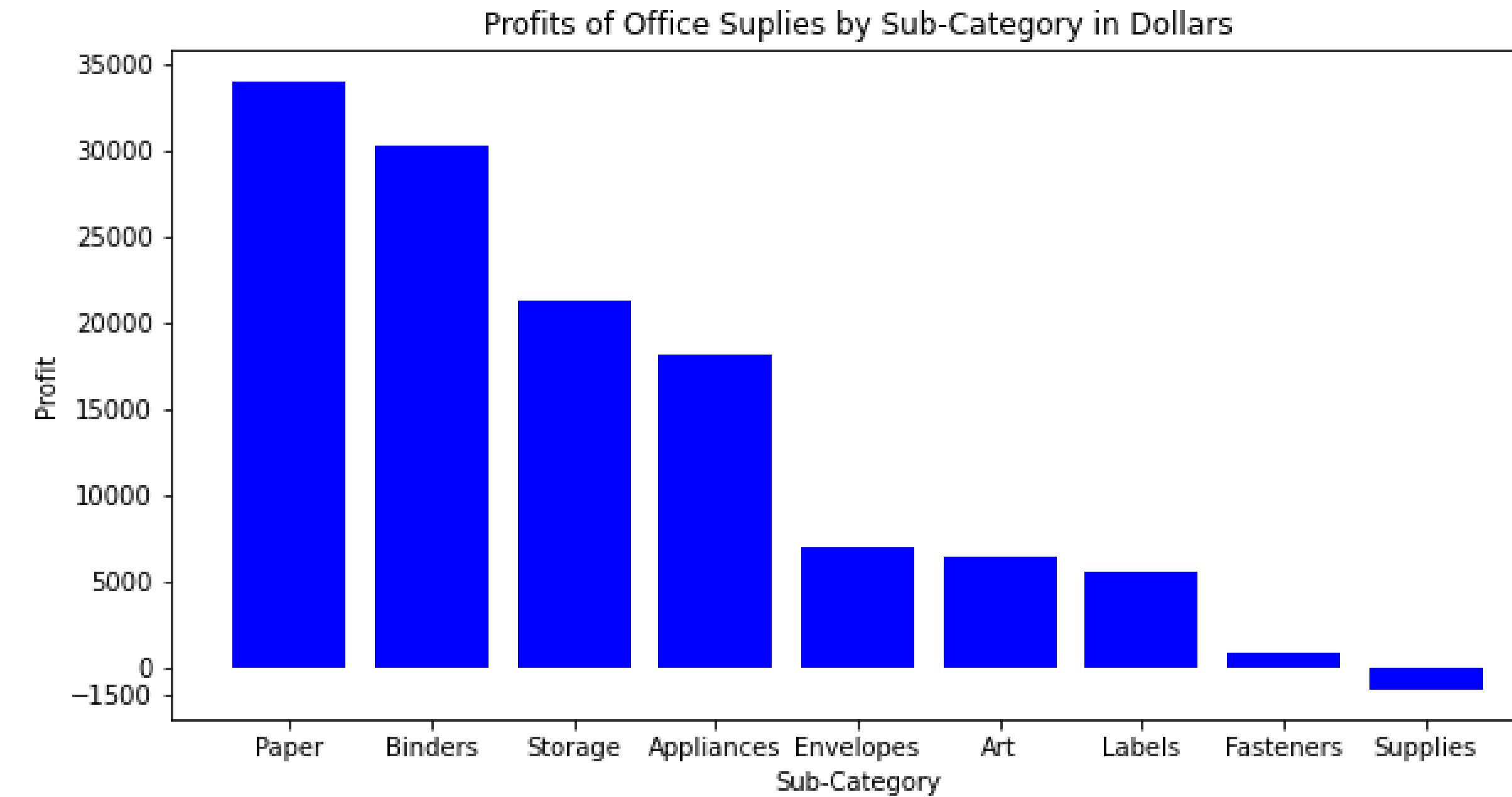
Sales based on Product Sub-Category



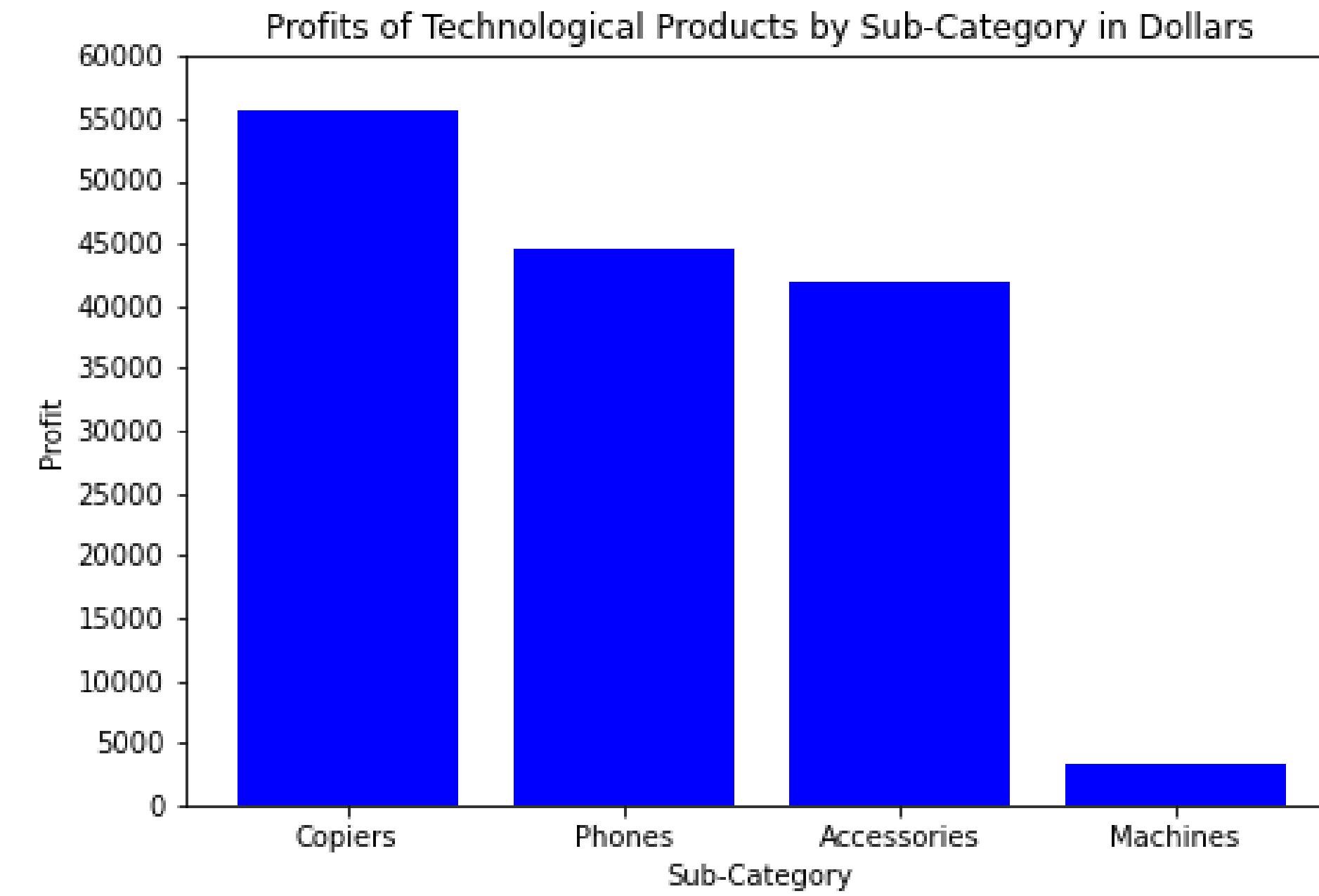
Profits based on Product Sub-Category



Profits based on Product Sub-Category



Profits based on Product Sub-Category



Creating Charts for showcasing most popular Product Sub-Categories by Quantity Sold



```
#Isolating necessary columns for getting most popular subcategory by quantity sold
df1 = dataset[['Category','Sub-Category','Quantity']]
df1 = df1.sort_values(["Category","Sub-Category"])

#Separate by unique Category
furniture_cond = df1["Category"] == "Furniture"
office_cond = df1["Category"] == "Office Supplies"
tech_cond = df1["Category"] == "Technology"

furniture_df1 = df1.loc[furniture_cond,:]
office_df1 = df1.loc[office_cond,:]
tech_df1 = df1.loc[tech_cond,:]

#Getting Total Quantity by Sub-Category
agg_func = {'Category': 'first', 'Quantity': 'sum'}
furniture_df1 = furniture_df1.groupby(furniture_df1['Sub-Category']).aggregate(agg_func)
office_df1 = office_df1.groupby(office_df1['Sub-Category']).aggregate(agg_func)
tech_df1 = tech_df1.groupby(tech_df1['Sub-Category']).aggregate(agg_func)

#Sort by Quantity
furniture_df1 = furniture_df1.sort_values(["Quantity"],ascending=False)
office_df1 = office_df1.sort_values(["Quantity"],ascending=False)
tech_df1 = tech_df1.sort_values(["Quantity"],ascending=False)

#Reset Index
furniture_df1.reset_index(inplace=True,drop=False)
office_df1.reset_index(inplace=True,drop=False)
tech_df1.reset_index(inplace=True,drop=False)
```

Resulting DataFrames



	Sub-Category	Category	Quantity
0	Phones	Technology	3289
1	Accessories	Technology	2976
2	Machines	Technology	440
3	Copiers	Technology	234

	Sub-Category	Category	Quantity
0	Furnishings	Furniture	3563
1	Chairs	Furniture	2356
2	Tables	Furniture	1241
3	Bookcases	Furniture	868



Resulting DataFrames



	Sub-Category	Category	Quantity
0	Binders	Office Supplies	5974
1	Paper	Office Supplies	5178
2	Storage	Office Supplies	3158
3	Art	Office Supplies	3000
4	Appliances	Office Supplies	1729
5	Labels	Office Supplies	1400
6	Fasteners	Office Supplies	914
7	Envelopes	Office Supplies	906
8	Supplies	Office Supplies	647

Plotting Charts

```
#Plotting Pie Chart (Furniture Sub-Categories by Quantity)
import matplotlib.pyplot as plt

#https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/
#http://www.Learningaboutelectronics.com/Articles/How-to-create-a-pie-chart-in-matplotlib-with-Python.php

# Creating dataset
sub_cat = furniture_df1["Sub-Category"]
data = furniture_df1["Quantity"]

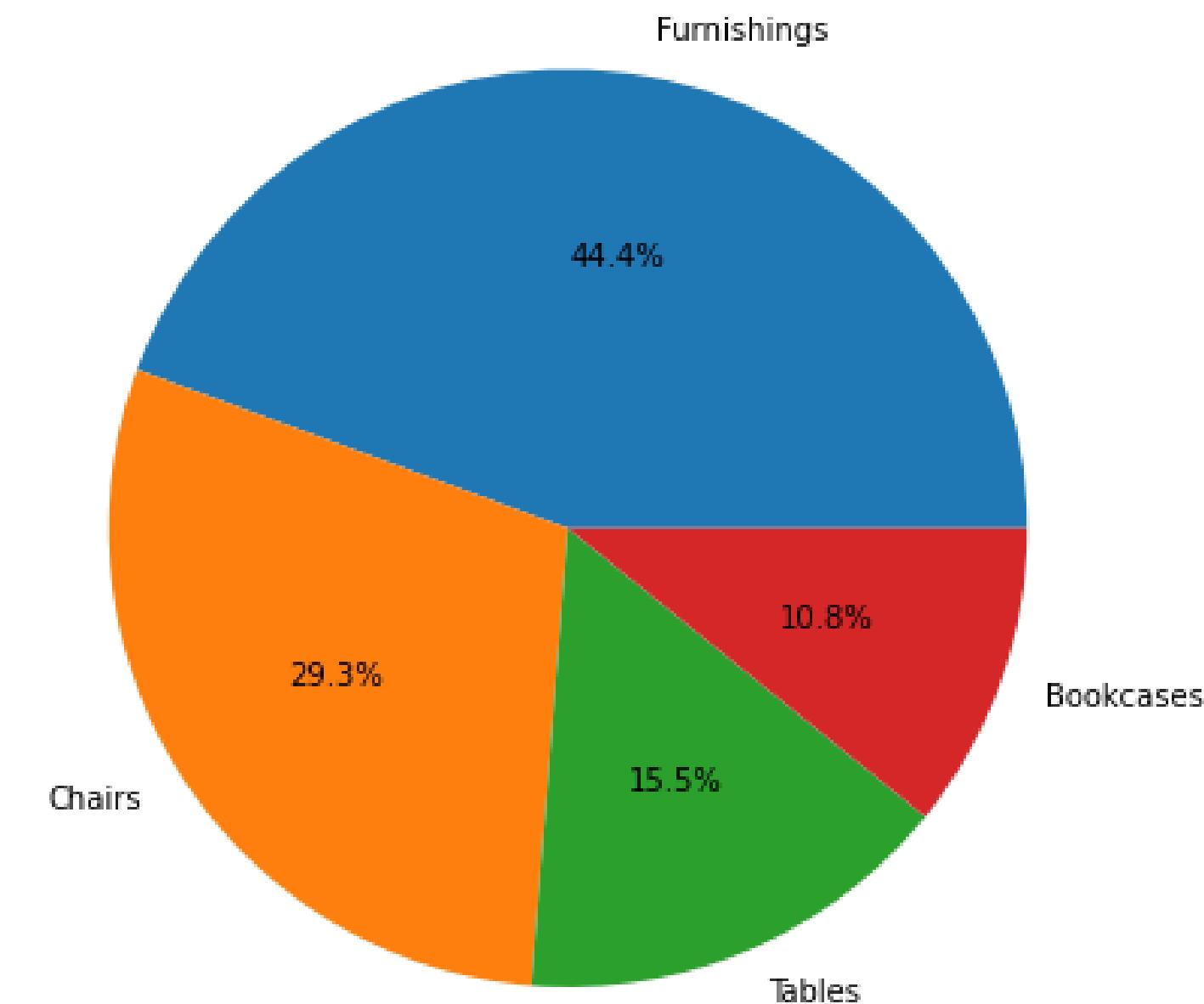
# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = sub_cat, autopct='%1.1f%%')
plt.title("Percentage of Furniture Product Sub-Categories by Quantity Sold")

# show plot
fig.savefig("Percentage of Furniture Product Sub-Categories by Quantity Sold")
plt.show()
```

Most Popular Sub-Categories by Quantity Sold



Percentage of Furniture Product Sub-Categories by Quantity Sold

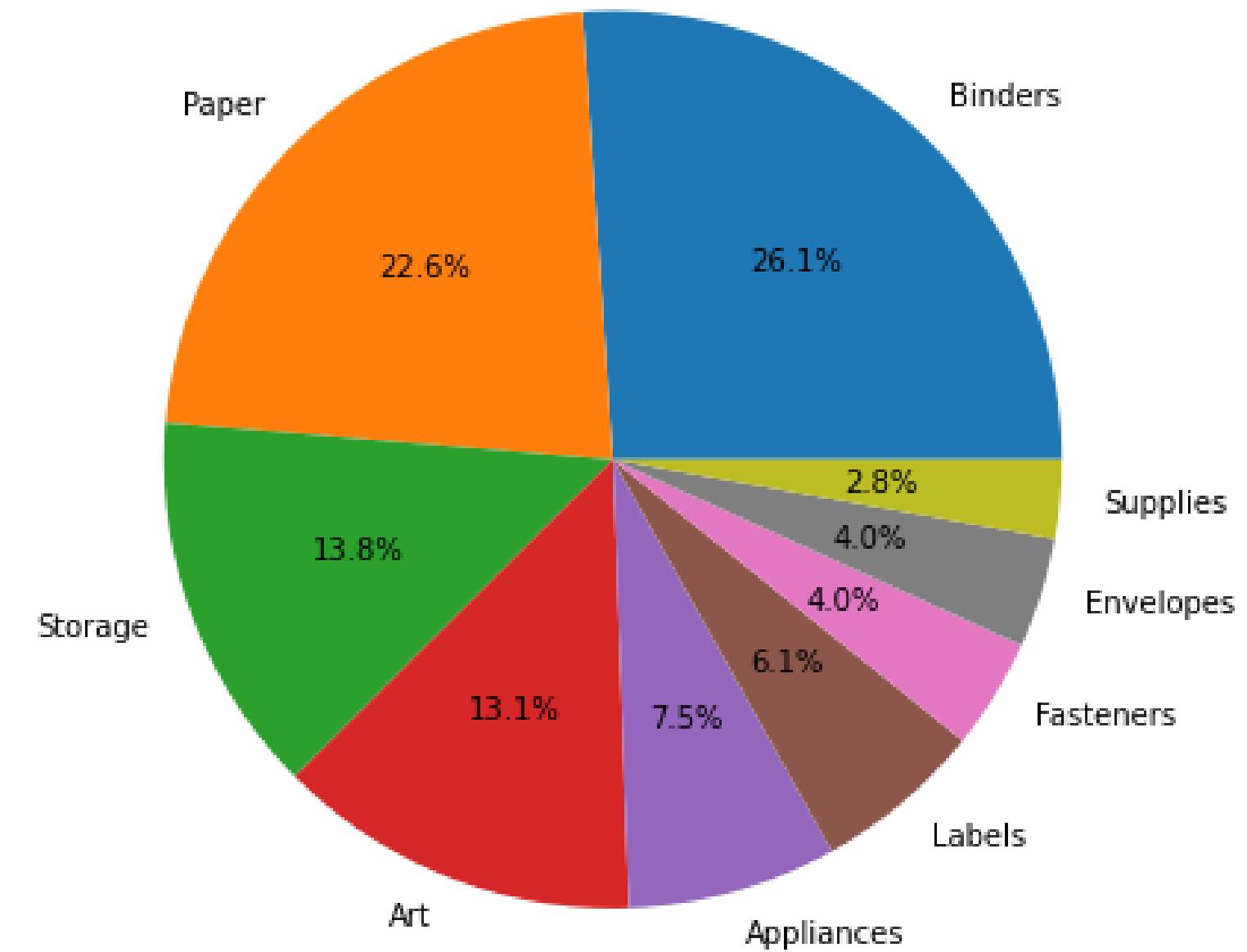


Most Popular Sub-Categories by Quantity Sold

Sold



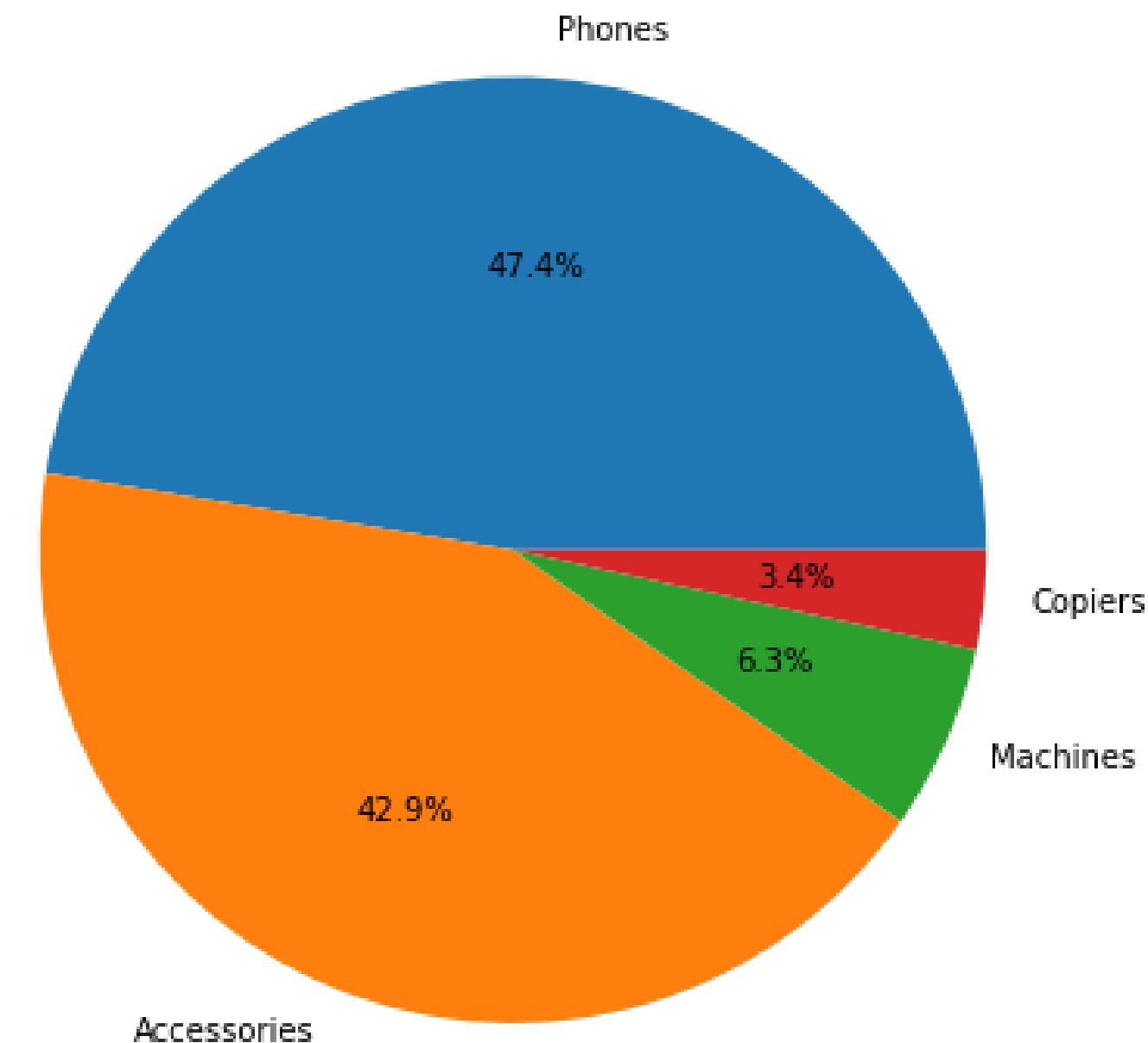
Percentage of Office Supplies Sub-Categories by Quantity Sold



Most Popular Sub-Categories by Quantity Sold



Percentage of Technological Product Sub-Categories by Quantity Sold



\data_by_profit_and_shipmode.ipnyb



Sorting data for Ship Mode Popularity

```
# counting data by ship mode  
shipmode = dataset.groupby('Ship Mode')  
shipmode_df = pd.DataFrame(shipmode['Ship Mode'].count())  
shipmode_df.index.name = None  
shipmode_df['Count']=shipmode_df['Ship Mode']  
shipmode_df = shipmode_df.drop(columns=['Ship Mode'])  
shipmode_df.index.name = 'Ship Mode'  
shipmode_df = shipmode_df.sort_values('Count').reset_index()
```

Resulting Data Frame

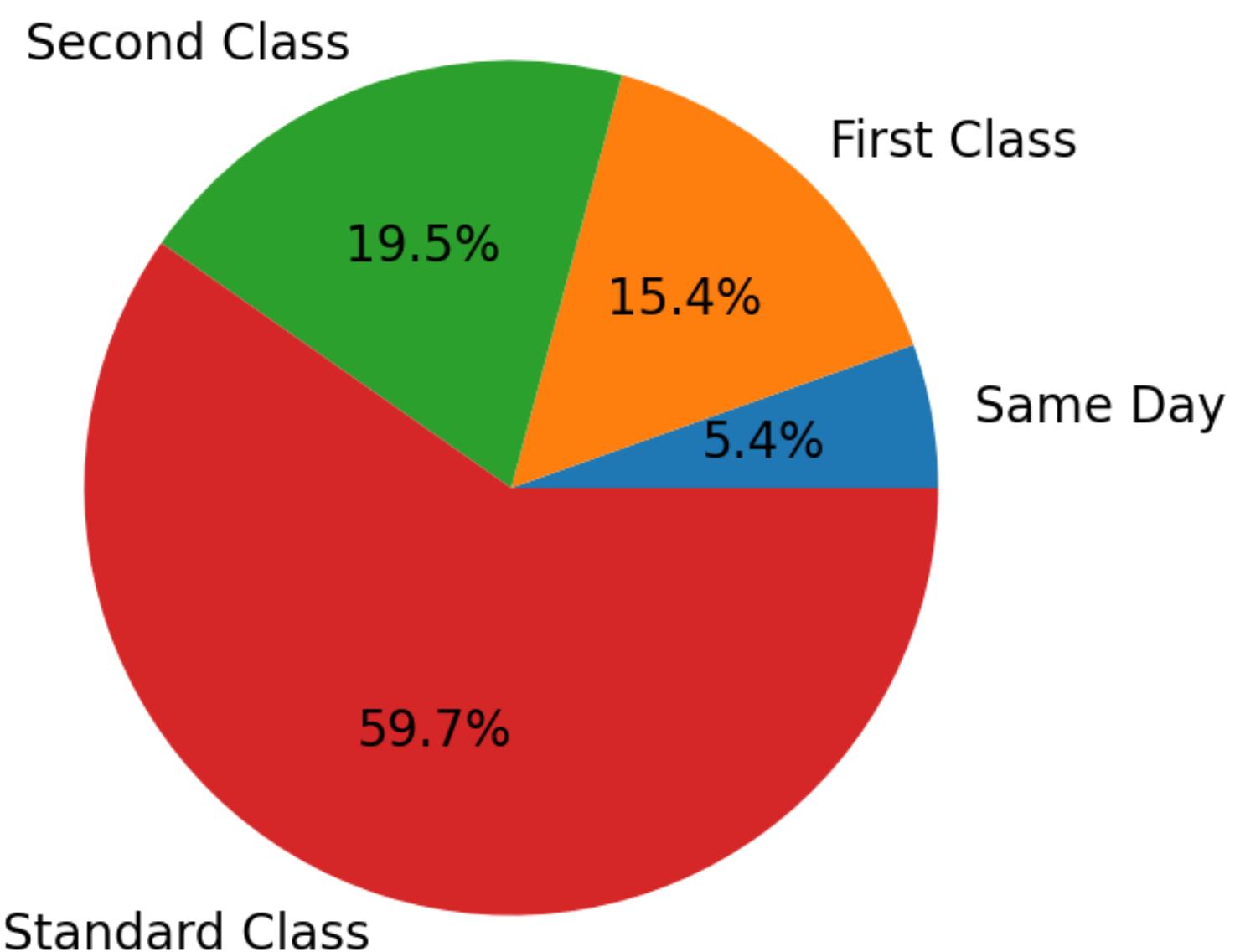
	Ship Mode	Count
0	Same Day	543
1	First Class	1538
2	Second Class	1945
3	Standard Class	5968

Charting data by Ship Mode Popularity

```
# chart by ship mode popularity
fig = plt.pie(shipmode_df['Count'],labels=list(shipmode_df['Ship Mode']),autopct='%1.1f%%') #https://datatofish.com/pie-chart-
matplotlib/
plt.title('Most Popular Ship Mode')
plt.savefig("./charts_by_ship_mode/Ship_Mode_Popularity.png", dpi=200)
```



Most Popular Ship Mode



Creating Dataframes for Profit Based on Ship Mode by Region

```
# dataframe per class
fstclass_df = dataset[dataset['Ship Mode']=='First Class'].drop(columns=["Region", "State", "City", "Customer ID", "Segment", "Product ID", "Product Name", "Category", "Sub-Category"])
stdclass_df = dataset[dataset['Ship Mode']=='Standard Class'].drop(columns=["Region", "State", "City", "Customer ID", "Segment", "Product ID", "Product Name", "Category", "Sub-Category"])
scdclass_df = dataset[dataset['Ship Mode']=='Second Class'].drop(columns=["Region", "State", "City", "Customer ID", "Segment", "Product ID", "Product Name", "Category", "Sub-Category"])
sameday_df = dataset[dataset['Ship Mode']=='Same Day'].drop(columns=["Region", "State", "City", "Customer ID", "Segment", "Product ID", "Product Name", "Category", "Sub-Category"])

# profit by region
fstclass_bylocation = dataset[dataset['Ship Mode']=='First Class'].drop(columns=["Customer ID", "Segment", "Product ID", "ProductName", "Category", "Sub-Category"])
fstclass_region = fstclass_bylocation.groupby('Region').sum().sort_values('Profit').reset_index()

stdclass_bylocation = dataset[dataset['Ship Mode']=='Standard Class'].drop(columns=["Customer ID", "Segment", "Product ID", "Product Name", "Category", "Sub-Category"])
stdclass_region = stdclass_bylocation.groupby('Region').sum().sort_values('Profit').reset_index()

scdclass_bylocation = dataset[dataset['Ship Mode']=='Second Class'].drop(columns=["Customer ID", "Segment", "Product ID", "ProductName", "Category", "Sub-Category"])
scdclass_region = scdclass_bylocation.groupby('Region').sum().sort_values('Profit').reset_index()

sameday_bylocation = dataset[dataset['Ship Mode']=='Same Day'].drop(columns=["Customer ID", "Segment", "Product ID", "ProductName", "Category", "Sub-Category"])
sameday_region = sameday_bylocation.groupby('Region').sum().sort_values('Profit').reset_index()
```

Dataframes for Profit Based on Ship Mode by Region

fstclass_region

	Region	Sales	Quantity	Discount	Profit
0	Central	58746.9154	1156	79.42	3707.2672
1	South	49332.5660	830	34.30	6892.3854
2	East	113587.0530	1805	76.50	15732.0141
3	West	129761.8885	1902	62.95	22638.1732

stdclass_region

	Region	Sales	Quantity	Discount	Profit
0	Central	318527.5600	5437	347.42	25352.3807
1	South	227613.5535	3761	156.80	26952.2330
2	West	406752.7985	7385	208.90	54760.9657
3	East	405321.8310	6214	241.90	57023.2081

scdclass_region

	Region	Sales	Quantity	Discount	Profit
0	Central	103550.0054	1795	104.60	9114.8349
1	East	116545.5240	2026	70.00	10787.2908
2	South	93758.6125	1294	36.15	14667.1469
3	West	145339.4275	2308	59.40	22877.3628

sameday_region

	Region	Sales	Quantity	Discount	Profit
0	South	21017.173	324	11.30	-1762.3350
1	Central	20415.410	392	26.90	1531.8797
2	East	43326.832	573	25.60	7980.2670
3	West	43603.710	671	18.95	8141.9472

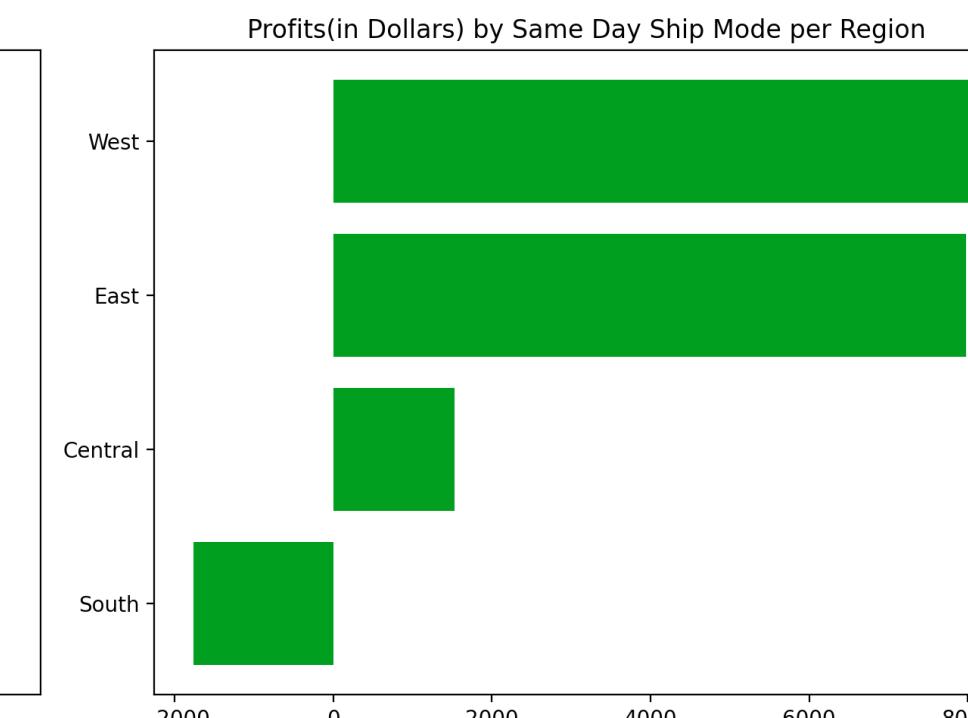
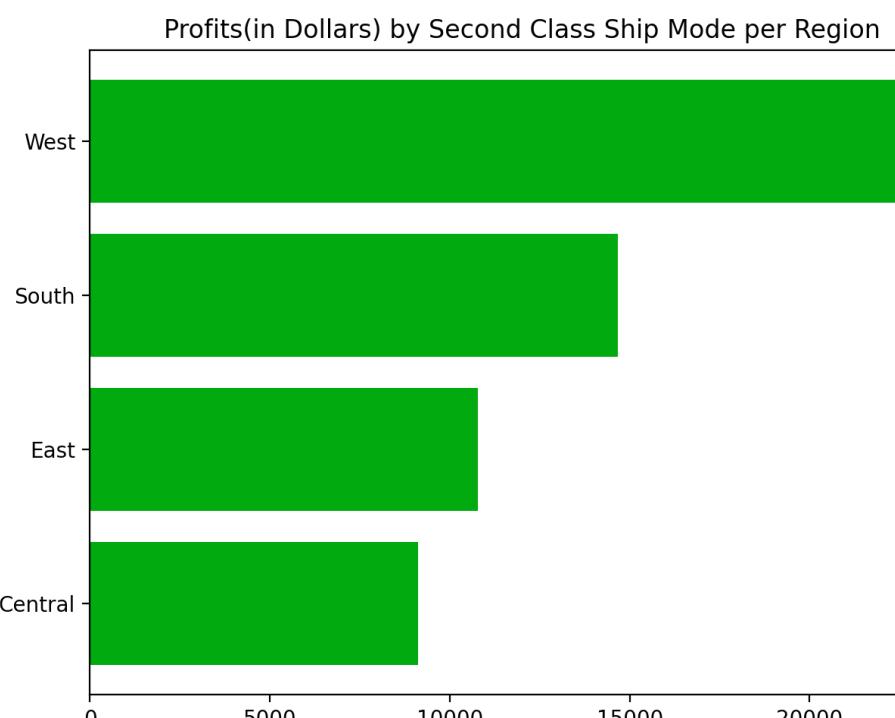
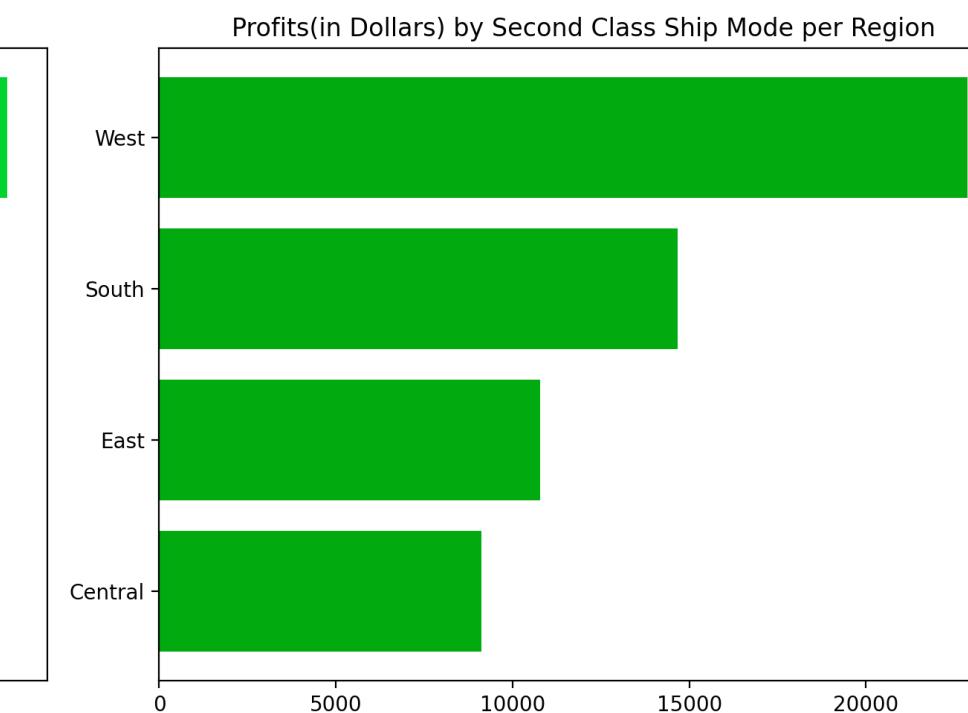
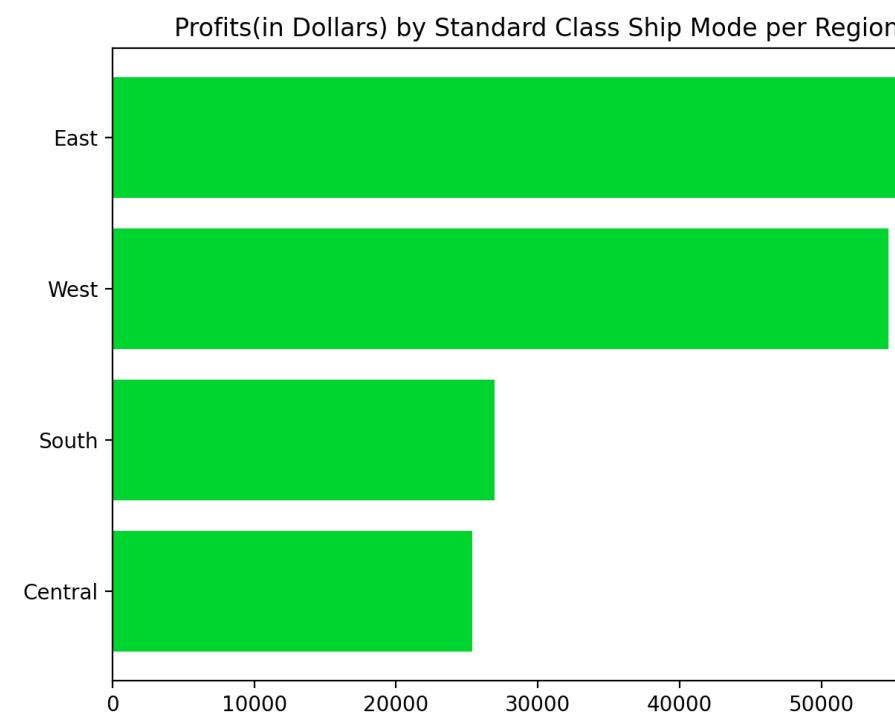
Creating Charts for Profit Based on Ship Mode by Region

```
# plotting profits per class by region
fig,ax = plt.subplots(1,4)
plt.figure(figsize=(15, 6))
fig.set_size_inches(26,5)

ax[0].barh(y = fstclass_region['Region'], width = fstclass_region['Profit'],color="#00e050' )
ax[0].set_title('Profits(in Dollars) by First Class Ship Mode per Region') # https://stackoverflow.com/questions/6963035/pyplot-axes-labels-for-subplots
ax[1].barh(y = stdclass_region['Region'],width = stdclass_region['Profit'],color='#00d42f' )
ax[1].set_title('Profits(in Dollars) by Standard Class Ship Mode per Region')
ax[2].barh(y = scdclass_region['Region'], width = scdclass_region['Profit'],color='#00aa0f' )
ax[2].set_title('Profits(in Dollars) by Second Class Ship Mode per Region')
ax[3].barh(y = sameday_region['Region'],width = sameday_region['Profit'],color='#009f20' )
ax[3].set_title('Profits(in Dollars) by Same Day Ship Mode per Region')

fig.tight_layout()
plt.show()
fig.savefig("./charts_by_ship_mode/Profits_by_ShipMode_per_Region.png", dpi=200)
```

\charts_by_ship_mode\Profits_by_ShipMode_per_Region.png



Creating Dataframes for Profit Based on Ship Mode by Region and State (First Class)



```
# profit by first class per region & state
# grouping by region
fstclass_bycentral = fstclass_bylocation[fstclass_bylocation['Region']=='Central'].drop(columns=['City','Ship Mode'])
fstclass_bycentral = fstclass_bycentral.groupby('State').sum().sort_values("Profit").reset_index()

fstclass_byeast = fstclass_bylocation[fstclass_bylocation['Region']=='East'].drop(columns=['City','Ship Mode'])
fstclass_byeast = fstclass_byeast.groupby('State').sum().sort_values("Profit").reset_index()

fstclass_bywest = fstclass_bylocation[fstclass_bylocation['Region']=='West'].drop(columns=['City','Ship Mode'])
fstclass_bywest = fstclass_bywest.groupby('State').sum().sort_values("Profit").reset_index()

fstclass_bysouth = fstclass_bylocation[fstclass_bylocation['Region']=='South'].drop(columns=['City','Ship Mode'])
fstclass_bysouth = fstclass_bysouth.groupby('State').sum().sort_values("Profit").reset_index()
```

Creating Charts for Profit Based on Ship Mode by Region and State (First Class)

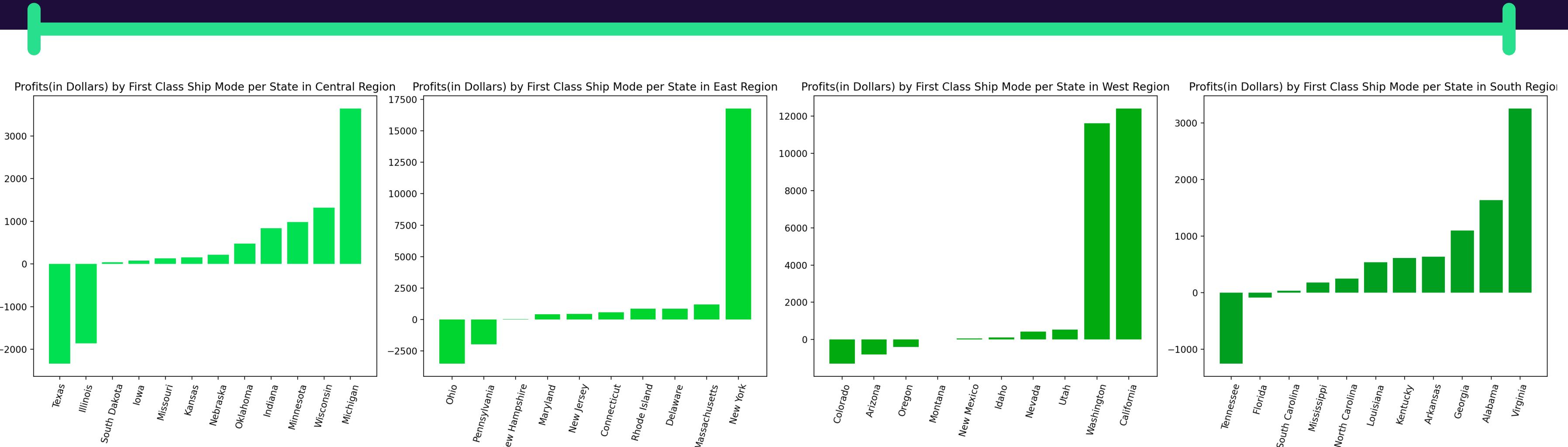


```
# plotting
fig,ax = plt.subplots(1,4)
fig.set_size_inches(24,6)

ax[0].bar(x = fstclass_bycentral['State'], height = fstclass_bycentral['Profit'],color="#00e050")
ax[0].tick_params(axis='x', rotation=75) # https://stackoverflow.com/questions/31186019/rotate-tick-labels-in-subplot-pyplot-matplotlib-gridspec/52461208
ax[0].set_title('Profits(in Dollars) by First Class Ship Mode per State in Central Region')
ax[1].bar(x = fstclass_byeast['State'], height = fstclass_byeast['Profit'],color="#00d42f")
ax[1].tick_params(axis='x', rotation=75)
ax[1].set_title('Profits(in Dollars) by First Class Ship Mode per State in East Region')
ax[2].bar(x = fstclass_bywest['State'], height = fstclass_bywest['Profit'],color="#00aa0f")
ax[2].tick_params(axis='x', rotation=75)
ax[2].set_title('Profits(in Dollars) by First Class Ship Mode per State in West Region')
ax[3].bar(x = fstclass_bysouth['State'], height = fstclass_bysouth['Profit'],color="#009f20")
ax[3].tick_params(axis='x', rotation=75)
ax[3].set_title('Profits(in Dollars) by First Class Ship Mode per State in South Region')

fig.tight_layout()
plt.show()
fig.savefig("./charts_by_ship_mode/Profits_by_FirstClass_ShipMode_per_State.png", dpi=200)
```

\charts_by_ship_mode\Profits_by_FirstClass_ShipMode_per_State.png



Creating Dataframes for Profit Based on Ship Mode by Region and State (Standard class)



```
# profit by standard class shipping per region & state
# grouping by region
stdclass_bycentral = stdclass_bylocation[stdclass_bylocation['Region']=='Central'].drop(columns=['City','Ship Mode'])
stdclass_bycentral = stdclass_bycentral.groupby('State').sum().sort_values("Profit").reset_index()

stdclass_byeast = stdclass_bylocation[stdclass_bylocation['Region']=='East'].drop(columns=['City','Ship Mode'])
stdclass_byeast = stdclass_byeast.groupby('State').sum().sort_values("Profit").reset_index()

stdclass_bywest = stdclass_bylocation[stdclass_bylocation['Region']=='West'].drop(columns=['City','Ship Mode'])
stdclass_bywest = stdclass_bywest.groupby('State').sum().sort_values("Profit").reset_index()

stdclass_bysouth = stdclass_bylocation[stdclass_bylocation['Region']=='South'].drop(columns=['City','Ship Mode'])
stdclass_bysouth = stdclass_bysouth.groupby('State').sum().sort_values("Profit").reset_index()
```

Creating Charts for Profit Based on Ship Mode by Region and State (Standard Class)

```
# plotting
fig,ax = plt.subplots(1,4)
fig.set_size_inches(24,6)

ax[0].bar(x = stdclass_bycentral['State'], height = stdclass_bycentral['Profit'],color="#00e050")
ax[0].tick_params(axis='x', rotation=80)
ax[0].set_title('Profits by Standard Class Ship Mode per State in Central Region')
ax[1].bar(x = stdclass_byeast['State'], height = stdclass_byeast['Profit'],color="#00d42f")
ax[1].tick_params(axis='x', rotation=80)
ax[1].set_title('Profits by Standard Class Ship Mode per State in East Region')
ax[2].bar(x = stdclass_bywest['State'], height = stdclass_bywest['Profit'],color="#00aa0f")
ax[2].tick_params(axis='x', rotation=80)
ax[2].set_title('Profits by Standard Class Ship Mode per State in West Region')
ax[3].bar(x = stdclass_bysouth['State'], height = stdclass_bysouth['Profit'],color="#009f20")
ax[3].tick_params(axis='x', rotation=80)
ax[3].set_title('Profits by Standard Class Ship Mode per State in South Region')

fig.tight_layout()
plt.show()
fig.savefig("./charts_by_ship_mode/Profits_by_StandardClass_ShipMode_per_State.png", dpi=200)
```

\charts_by_ship_mode\Profits_by_StandardClass_ShipMode_per_State.png



Creating Dataframes for Profit Based on Ship Mode by Region and State (Second class)

```
# profit by second class shipping per region & state
# grouping by region
scdclass_bycentral = scdclass_bylocation[scdclass_bylocation['Region']=='Central'].drop(columns=['City','Ship Mode'])
scdclass_bycentral = scdclass_bycentral.groupby('State').sum().sort_values("Profit").reset_index()

scdclass_byeast = scdclass_bylocation[scdclass_bylocation['Region']=='East'].drop(columns=['City','Ship Mode'])
scdclass_byeast = scdclass_byeast.groupby('State').sum().sort_values("Profit").reset_index()

scdclass_bywest = scdclass_bylocation[scdclass_bylocation['Region']=='West'].drop(columns=['City','Ship Mode'])
scdclass_bywest = scdclass_bywest.groupby('State').sum().sort_values("Profit").reset_index()

scdclass_bysouth = scdclass_bylocation[scdclass_bylocation['Region']=='South'].drop(columns=['City','Ship Mode'])
scdclass_bysouth = scdclass_bysouth.groupby('State').sum().sort_values("Profit").reset_index()
```

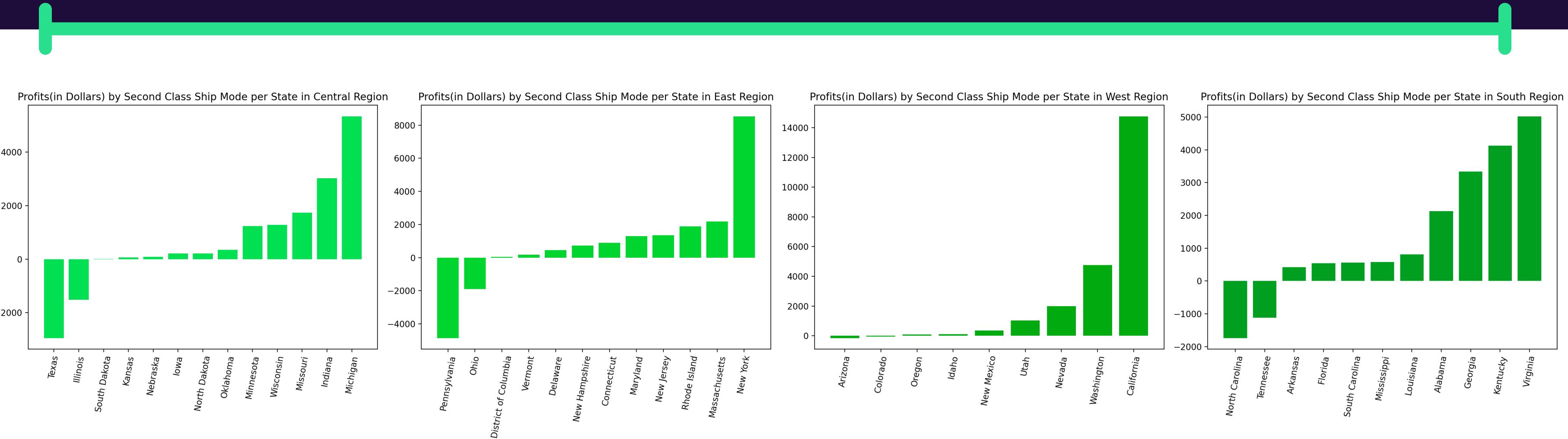
Creating Charts for Profit Based on Ship Mode by Region and State (Second Class)



```
# plotting
fig,ax = plt.subplots(1,4)
fig.set_size_inches(26,6)

ax[0].bar(x = scdclass_bycentral['State'], height = scdclass_bycentral['Profit'],color="#00e050")
ax[0].tick_params(axis='x', rotation=80)
ax[0].set_title('Profits(in Dollars) by Second Class Ship Mode per State in Central Region')
ax[1].bar(x = scdclass_byeast['State'], height = scdclass_byeast['Profit'],color="#00d42f")
ax[1].tick_params(axis='x', rotation=80)
ax[1].set_title('Profits(in Dollars) by Second Class Ship Mode per State in East Region')
ax[2].bar(x = scdclass_bywest['State'], height = scdclass_bywest['Profit'],color="#00aa0f")
ax[2].tick_params(axis='x', rotation=80)
ax[2].set_title('Profits(in Dollars) by Second Class Ship Mode per State in West Region')
ax[3].bar(x = scdclass_bysouth['State'], height = scdclass_bysouth['Profit'],color="#009f20")
ax[3].tick_params(axis='x', rotation=80)
ax[3].set_title('Profits(in Dollars) by Second Class Ship Mode per State in South Region')
fig.tight_layout()
plt.show()
fig.savefig("./charts_by_ship_mode/Profits_by_SecondClass_ShipMode_per_State.png", dpi=200)
```

\charts_by_ship_mode\Profits_by_SecondClass_ShipMode_per_State.png



Creating Dataframes for Profit Based on Ship Mode by Region and State (Same Day)



```
# profit by same day shipping per region & state
# grouping by region
sameday_bycentral = sameday_bylocation[sameday_bylocation['Region']=='Central'].drop(columns=['City','Ship Mode'])
sameday_bycentral = sameday_bycentral.groupby('State').sum().sort_values("Profit").reset_index()

sameday_byeast = sameday_bylocation[sameday_bylocation['Region']=='East'].drop(columns=['City','Ship Mode'])
sameday_byeast = sameday_byeast.groupby('State').sum().sort_values("Profit").reset_index()

sameday_bywest = sameday_bylocation[sameday_bylocation['Region']=='West'].drop(columns=['City','Ship Mode'])
sameday_bywest = sameday_bywest.groupby('State').sum().sort_values("Profit").reset_index()

sameday_bysouth = sameday_bylocation[sameday_bylocation['Region']=='South'].drop(columns=['City','Ship Mode'])
sameday_bysouth = sameday_bysouth.groupby('State').sum().sort_values("Profit").reset_index()
```

Creating Charts for Profit Based on Ship Mode by Region and State (Same Day)



```
# plotting
fig,ax = plt.subplots(1,4)
fig.set_size_inches(25,6)

ax[0].bar(x = sameday_bycentral['State'], height = sameday_bycentral['Profit'],color='#00e050')
ax[0].tick_params(axis='x', rotation=80)
ax[0].set_title('Profits(in Dollars) by Same Day Ship Mode per State in Central Region')

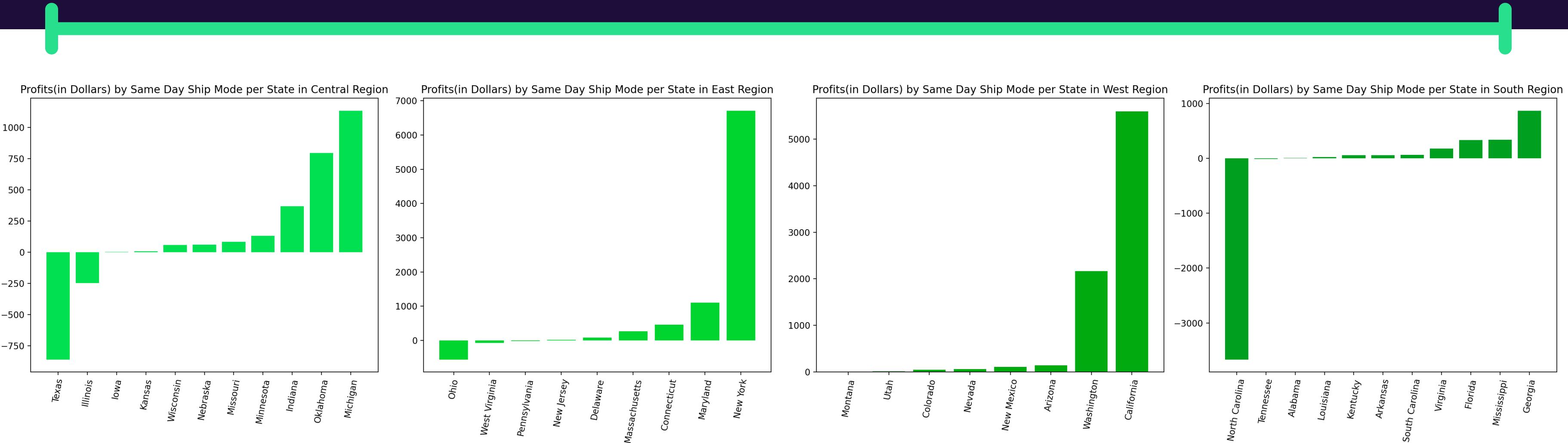
ax[1].bar(x = sameday_byeast['State'], height = sameday_byeast['Profit'],color='#00d42f')
ax[1].tick_params(axis='x', rotation=80)
ax[1].set_title('Profits(in Dollars) by Same Day Ship Mode per State in East Region')

ax[2].bar(x = sameday_bywest['State'], height = sameday_bywest['Profit'],color='#00aa0f')
ax[2].tick_params(axis='x', rotation=80)
ax[2].set_title('Profits(in Dollars) by Same Day Ship Mode per State in West Region')

ax[3].bar(x = sameday_bysouth['State'], height = sameday_bysouth['Profit'],color='#009f20')
ax[3].tick_params(axis='x', rotation=80)
ax[3].set_title('Profits(in Dollars) by Same Day Ship Mode per State in South Region')

fig.tight_layout()
plt.show()
fig.savefig("./charts_by_ship_mode/Profits_by_SameDay_ShipMode_per_State.png", dpi=200)
```

\charts_by_ship_mode\Profits_by_SameDay_ShipMode_per_State.png



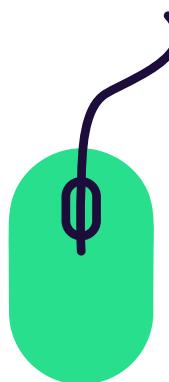
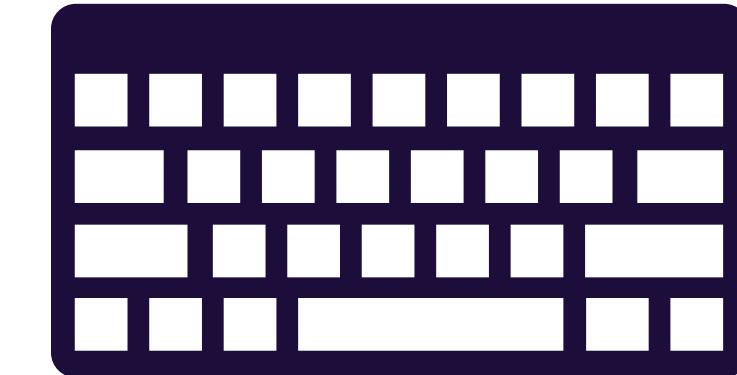


source for dataset:

https://www.kaggle.com/dhirajnirne/online-store-dataset



<



>