# Importing The required libraries and previewing data

In [ ]:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as colors
pio.templates.default= "plotly_white"
import matplotlib.pyplot as plt       #For Outlier Visualization
import seaborn as sns                 #For Outlier Visualization
df = pd.read_csv("data.csv", encoding="latin1")
print(df.head())
print(df.nunique())
```

```
   Row ID        Order ID  Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156   11/8/2016  11/11/2016    Second Class    CG-12520
1       2  CA-2016-152156   11/8/2016  11/11/2016    Second Class    CG-12520
2       3  CA-2016-138688   6/12/2016   6/16/2016    Second Class    DV-13045
3       4  US-2015-108966  10/11/2015  10/18/2015  Standard Class    SO-20335
4       5  US-2015-108966  10/11/2015  10/18/2015  Standard Class    SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID         Category Sub-Category  \
0       42420   South  FUR-BO-10001798        Furniture    Bookcases
1       42420   South  FUR-CH-10000454        Furniture       Chairs
2       90036    West  OFF-LA-10000240  Office Supplies       Labels
3       33311   South  FUR-TA-10000577        Furniture       Tables
4       33311   South  OFF-ST-10000760  Office Supplies      Storage

                                        Product Name     Sales  Quantity  \
0                  Bush Somerset Collection Bookcase  261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400         3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200         2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775         5
4                     Eldon Fold 'N Roll Cart System   22.3680         2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
Row ID           9994
Order ID         5009
Order Date       1237
Ship Date        1334
Ship Mode           4
Customer ID       793
Customer Name     793
Segment             3
Country             1
City              531
State              49
Postal Code       631
Region              4
Product ID       1862
```

```
Category           3
Sub-Category      17
Product Name    1850
Sales           5825
Quantity          14
Discount          12
Profit          7287
dtype: int64
```

Descriptive stats

```
In [ ]: df.describe()
```

Out[ ]:

| | Row ID | Postal Code | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|
| **count** | 9994.000000 | 9994.000000 | 9994.000000 | 9994.000000 | 9994.000000 | 9994.000000 |
| **mean** | 4997.500000 | 55190.379428 | 229.858001 | 3.789574 | 0.156203 | 28.656896 |
| **std** | 2885.163629 | 32063.693350 | 623.245101 | 2.225110 | 0.206452 | 234.260108 |
| **min** | 1.000000 | 1040.000000 | 0.444000 | 1.000000 | 0.000000 | -6599.978000 |
| **25%** | 2499.250000 | 23223.000000 | 17.280000 | 2.000000 | 0.000000 | 1.728750 |
| **50%** | 4997.500000 | 56430.500000 | 54.490000 | 3.000000 | 0.200000 | 8.666500 |
| **75%** | 7495.750000 | 90008.000000 | 209.940000 | 5.000000 | 0.200000 | 29.364000 |
| **max** | 9994.000000 | 99301.000000 | 22638.480000 | 14.000000 | 0.800000 | 8399.976000 |

```
In [ ]: 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
```

# Converting date time from obeject to date time data type

In [ ]:

```
df['Order Date']=pd.to_datetime(df['Order Date'])
df['Ship Date']=pd.to_datetime(df['Ship Date'])
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   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 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: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

# Adding new Attributes like month, year, time taken, Profit Margin

In [ ]:

```
df['Month']=df['Order Date'].dt.month
df['Year']=df['Order Date'].dt.year
df['Day']=df['Order Date'].dt.dayofweek
df['Time Taken']=df['Ship Date']-df['Order Date']
df['Profit Margin %'] = (df['Profit'] / df['Sales']) * 100
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 26 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   datetime64[ns] 
 3   Ship Date        9994 non-null   datetime64[ns] 
 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        
 21  Month            9994 non-null   int32          
 22  Year             9994 non-null   int32          
 23  Day              9994 non-null   int32          
 24  Time Taken       9994 non-null   timedelta64[ns]
 25  Profit Margin %  9994 non-null   float64        
dtypes: datetime64[ns](2), float64(4), int32(3), int64(3), object(13), timedelta64[ns](1)
memory usage: 1.9+ MB
```

# Data Cleaning

Checking Null values

In [ ]:

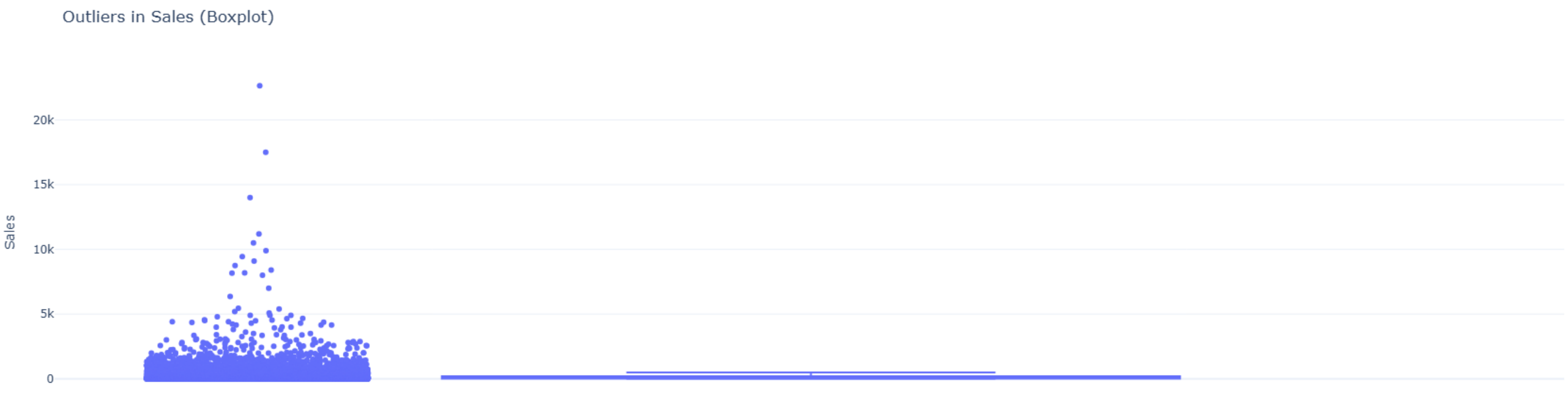
```python
print(df.isnull().sum())
```

```
Row ID              0
Order ID            0
Order Date          0
Ship Date           0
Ship Mode           0
Customer ID         0
Customer Name       0
Segment             0
Country             0
City                0
State               0
Postal Code         0
Region              0
Product ID          0
Category            0
Sub-Category        0
Product Name        0
Sales               0
Quantity            0
Discount            0
Profit              0
Month               0
Year                0
Day                 0
Time Taken          0
Profit Margin %     0
dtype: int64
```

Visualizing Outliers

In [ ]:

```python
fig = px.box(df, y="Sales", points="all", title="Outliers in Sales (Boxplot)")
fig.show()
```



Excluding Outliers

```
In [ ]: Q1 = df['Sales'].quantile(0.25)
        Q3 = df['Sales'].quantile(0.75)
        IQR = Q3 - Q1
        df = df[(df['Sales'] >= Q1 - 1.5*IQR) & (df['Sales'] <= Q3 + 1.5*IQR)]
        df.describe()
```

Out[ ]:

| | Row ID | Order Date | Ship Date | Postal Code | Sales | Quantity | Discount | Profit | Month | Year | Day | Tin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 8827.000000 | 8827 | 8827 | 8827.000000 | 8827.000000 | 8827.000000 | 8827.000000 | 8827.000000 | 8827.000000 | 8827.000000 | 8827.000000 | |
| **mean** | 5016.375892 | 2016-05-02 15:45:12.541067264 | 2016-05-06 14:57:05.036818688 | 55373.635663 | 92.864853 | 3.608587 | 0.157606 | 11.198644 | 7.784072 | 2015.731732 | 3.218647 | 23:11:52.49 |
| **min** | 1.000000 | 2014-01-03 00:00:00 | 2014-01-07 00:00:00 | 1040.000000 | 0.444000 | 1.000000 | 0.000000 | -1181.282400 | 1.000000 | 2014.000000 | 0.000000 | 0 days |
| **25%** | 2508.500000 | 2015-05-28 00:00:00 | 2015-05-31 00:00:00 | 23320.000000 | 15.008000 | 2.000000 | 0.000000 | 1.702400 | 5.000000 | 2015.000000 | 1.000000 | 3 days |
| **50%** | 5028.000000 | 2016-07-01 00:00:00 | 2016-07-03 00:00:00 | 59801.000000 | 40.880000 | 3.000000 | 0.200000 | 7.437600 | 9.000000 | 2016.000000 | 4.000000 | 4 days |
| **75%** | 7517.500000 | 2017-05-15 00:00:00 | 2017-05-19 00:00:00 | 90008.000000 | 124.225000 | 5.000000 | 0.200000 | 21.335400 | 11.000000 | 2017.000000 | 5.000000 | 5 days |
| **max** | 9994.000000 | 2017-12-30 00:00:00 | 2018-01-05 00:00:00 | 99301.000000 | 498.260000 | 14.000000 | 0.800000 | 240.859500 | 12.000000 | 2017.000000 | 6.000000 | 7 days |
| **std** | 2888.406540 | NaN | NaN | 31974.536840 | 114.045078 | 2.129308 | 0.211531 | 49.066101 | 3.289189 | 1.122496 | 2.120012 | 17:54:05.81 |

this shows that there are no null values in our dataset

Dropping duplicate entries

In [ ]:

```python
df = df.drop_duplicates(subset=["Order ID", "Product ID"])
df = df.reset_index(drop=True)
print(df[['Sales', 'Quantity', 'Discount', 'Profit']].describe())
df.head()
```

```
             Sales     Quantity     Discount       Profit
count  8821.000000  8821.000000  8821.000000  8821.000000
mean     92.830465     3.607981     0.157634    11.190131
std     114.044390     2.129173     0.211577    49.078632
min       0.444000     1.000000     0.000000 -1181.282400
25%      15.008000     2.000000     0.000000     1.702400
50%      40.776000     3.000000     0.200000     7.434000
75%     124.200000     5.000000     0.200000    21.295400
max     498.260000    14.000000     0.800000   240.859500
```

Out[ ]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | Product Name | Sales | Quantity | Discount | Profit | Month | Year | Day | Time Taken |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 2016-11-08 | 2016-11-11 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | Bush Somerset Collection Bookcase | 261.960 | 2 | 0.0 | 41.9136 | 11 | 2016 | 1 | 3 days |
| **1** | 3 | CA-2016-138688 | 2016-06-12 | 2016-06-16 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | Self-Adhesive Address Labels for Typewriters b... | 14.620 | 2 | 0.0 | 6.8714 | 6 | 2016 | 6 | 4 days |
| **2** | 5 | US-2015-108966 | 2015-10-11 | 2015-10-18 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | Eldon Fold 'N Roll Cart System | 22.368 | 2 | 0.2 | 2.5164 | 10 | 2015 | 6 | 7 days |
| **3** | 6 | CA-2014-115812 | 2014-06-09 | 2014-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... | Eldon Expressions Wood and Plastic Desk Access... | 48.860 | 7 | 0.0 | 14.1694 | 6 | 2014 | 0 | 5 days |
| **4** | 7 | CA-2014-115812 | 2014-06-09 | 2014-06-14 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... | Newell 322 | 7.280 | 4 | 0.0 | 1.9656 | 6 | 2014 | 0 | 5 days |

5 rows × 26 columns

Standardizing Text columns

In [ ]:

```
text_cols = df.select_dtypes(include='object').columns
df[text_cols] = df[text_cols].apply(lambda x: x.str.strip().str.title())
df.head()
```

Out[ ]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | Product Name | Sales | Quantity | Discount | Profit | Month | Year | Day | Time Taken |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Ca-2016-152156 | 2016-11-08 | 2016-11-11 | Second Class | Cg-12520 | Claire Gute | Consumer | United States | Henderson | ... | Bush Somerset Collection Bookcase | 261.960 | 2 | 0.0 | 41.9136 | 11 | 2016 | 1 | 3 days |
| **1** | 3 | Ca-2016-138688 | 2016-06-12 | 2016-06-16 | Second Class | Dv-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | Self-Adhesive Address Labels For Typewriters B... | 14.620 | 2 | 0.0 | 6.8714 | 6 | 2016 | 6 | 4 days |
| **2** | 5 | Us-2015-108966 | 2015-10-11 | 2015-10-18 | Standard Class | So-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | Eldon Fold 'N Roll Cart System | 22.368 | 2 | 0.2 | 2.5164 | 10 | 2015 | 6 | 7 days |
| **3** | 6 | Ca-2014-115812 | 2014-06-09 | 2014-06-14 | Standard Class | Bh-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... | Eldon Expressions Wood And Plastic Desk Access... | 48.860 | 7 | 0.0 | 14.1694 | 6 | 2014 | 0 | 5 days |
| **4** | 7 | Ca-2014-115812 | 2014-06-09 | 2014-06-14 | Standard Class | Bh-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | ... | Newell 322 | 7.280 | 4 | 0.0 | 1.9656 | 6 | 2014 | 0 | 5 days |

5 rows × 26 columns

In [ ]:

```
print("\n",df.nunique())
```

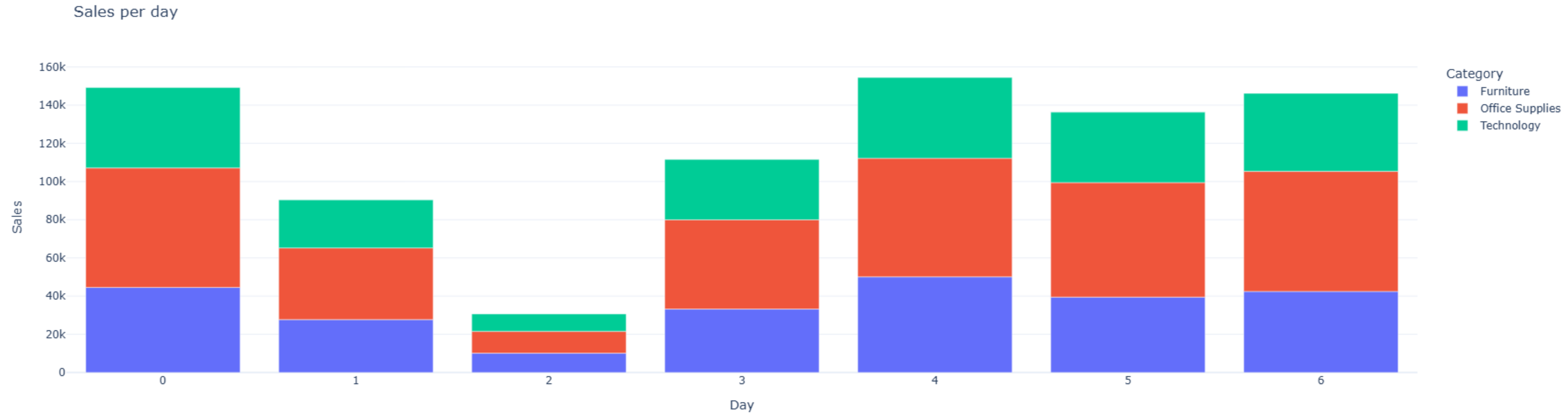
```
 Row ID                 8821
Order ID                4725
Order Date              1224
Ship Date               1316
Ship Mode                  4
Customer ID              790
Customer Name            790
Segment                    3
Country                    1
City                     527
State                     48
Postal Code              626
Region                     4
Product ID              1769
Category                   3
Sub-Category              17
Product Name            1754
Sales                   4881
Quantity                  14
Discount                  12
Profit                  6306
Month                     12
Year                       4
Day                        7
Time Taken                 8
Profit Margin %          470
dtype: int64
```

# Sales Analysis

Categorical Sales per day

In [ ]:

```
day_sales=df.groupby(['Day','Category'])['Sales'].sum().reset_index()
day_sales
fig=px.bar(day_sales, x='Day', y='Sales', color='Category', title='Sales per day')
fig.show()
```



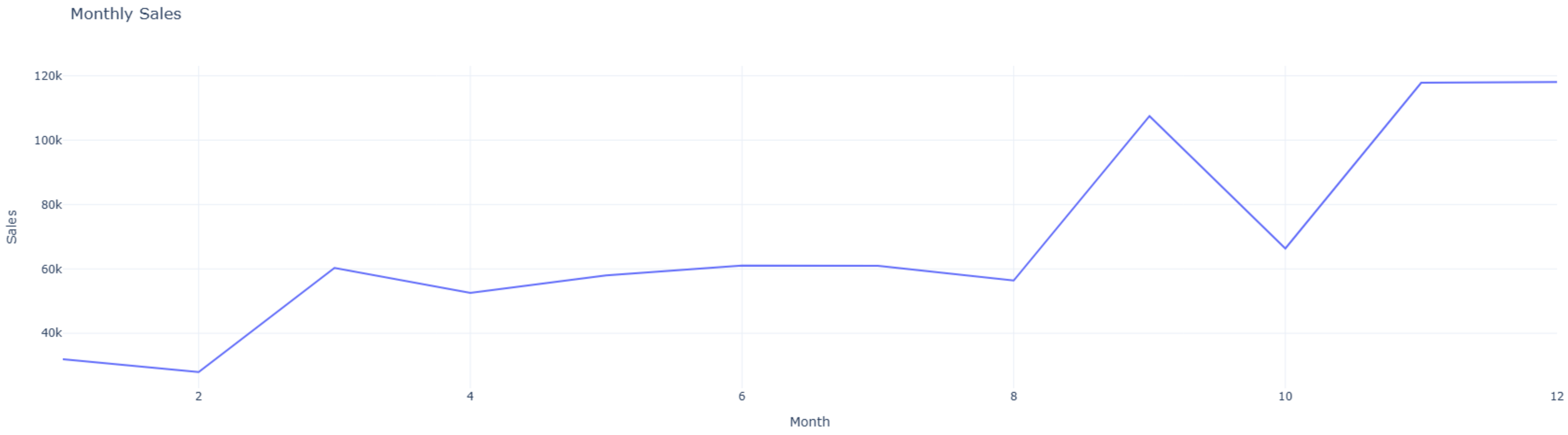
Monthly sales Analysis

```
In [ ]: monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
        print(monthly_sales)
```

```
    Month        Sales
0       1   31911.0090
1       2   27949.8704
2       3   60305.6104
3       4   52582.9201
4       5   58021.9144
5       6   61065.0348
6       7   60937.8290
7       8   56371.0245
8       9  107477.7543
9      10   66334.0502
10     11  117834.1100
11     12  118066.4003
```

Plotting Montly sales graph

```
In [ ]: fig=px.line(monthly_sales, x='Month', y='Sales', title='Monthly Sales')
        fig.show()
```



Categorical and Sub-Categorical Sales Analysis

```
In [ ]: Cate_sales = df.groupby('Category')['Sales'].sum().reset_index()
        subcate_sales = df.groupby('Sub-Category')['Sales'].sum().reset_index()
        print(Cate_sales)
        print("\n",subcate_sales)
```

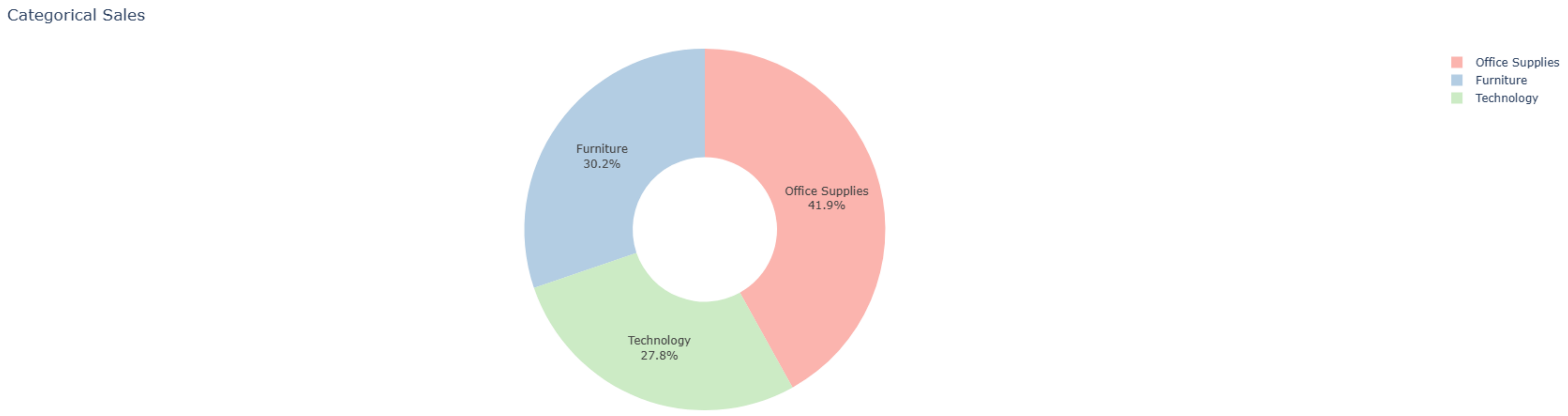
```
          Category        Sales
0        Furniture  247596.3264
1  Office Supplies  343240.1440
2       Technology  228021.0570

    Sub-Category        Sales
0    Accessories   92630.3880
1     Appliances   46729.7030
2            Art   26005.7680
3        Binders   60922.1550
4      Bookcases   40553.0319
5         Chairs   92644.0950
6        Copiers    5339.8140
7      Envelopes   15871.7460
8      Fasteners    3024.2800
9    Furnishings   69562.1520
10        Labels   11070.6480
11      Machines   12465.3170
12         Paper   75763.1560
13        Phones  117585.5380
14       Storage   94705.9560
15      Supplies    9146.7320
16        Tables   44837.0475
```

Categorical Sales Plot

In [ ]:

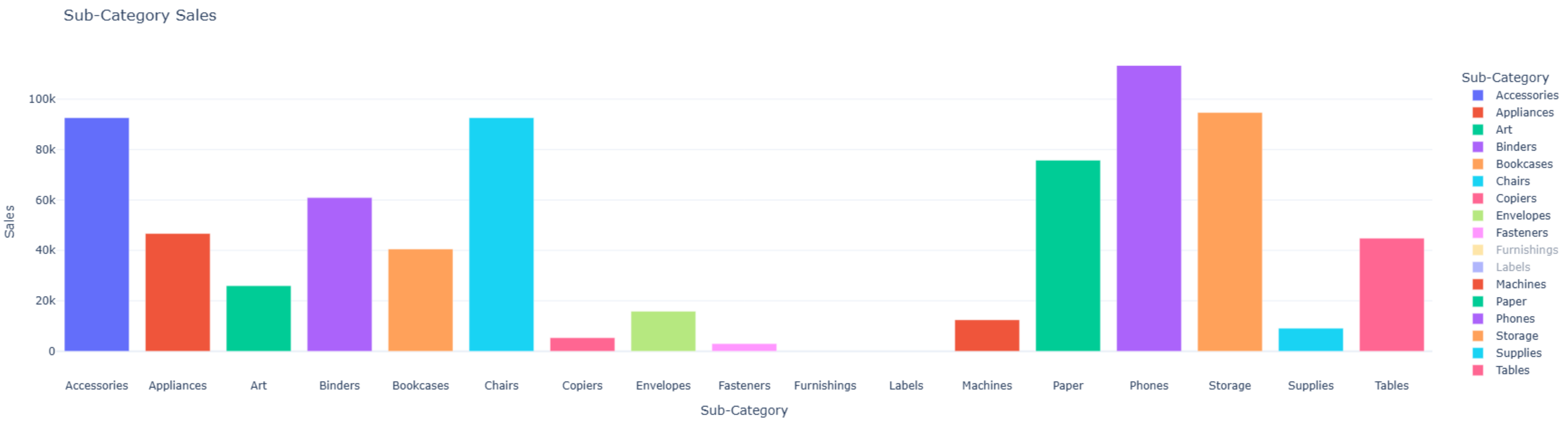
```
fig=px.pie(Cate_sales, values='Sales', names='Category', hole=0.4, color_discrete_sequence=px.colors.qualitative.Pastel1)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title='Categorical Sales')
fig.show()
```



Sub Categorical Sales Plot

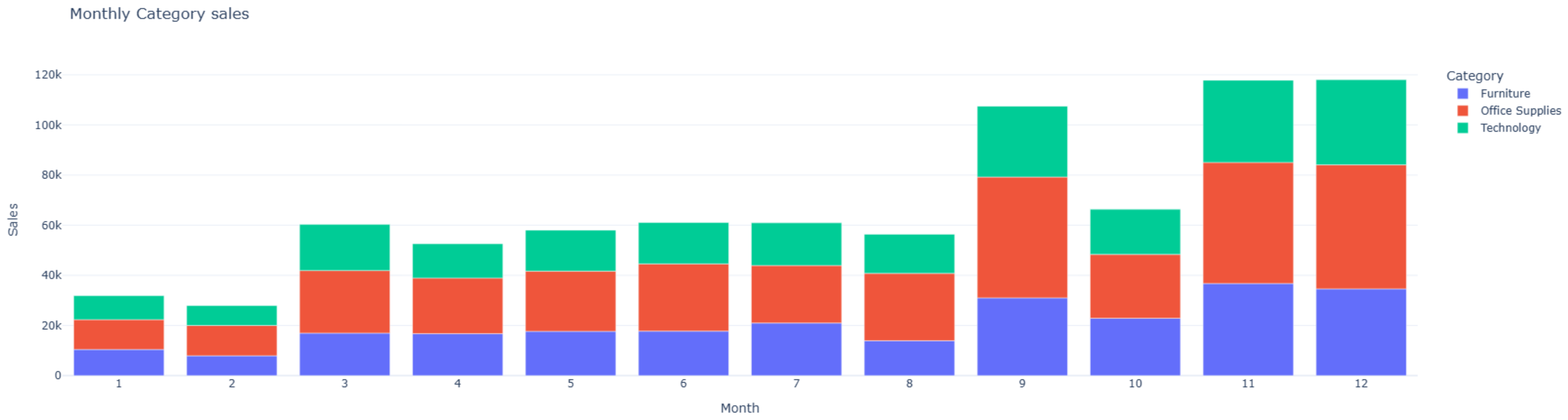
In [ ]:

```
fig=px.bar(subcate_sales, x='Sub-Category', y='Sales', color='Sub-Category', title='Sub-Category Sales')
fig.show()
```



Monthly sales by category

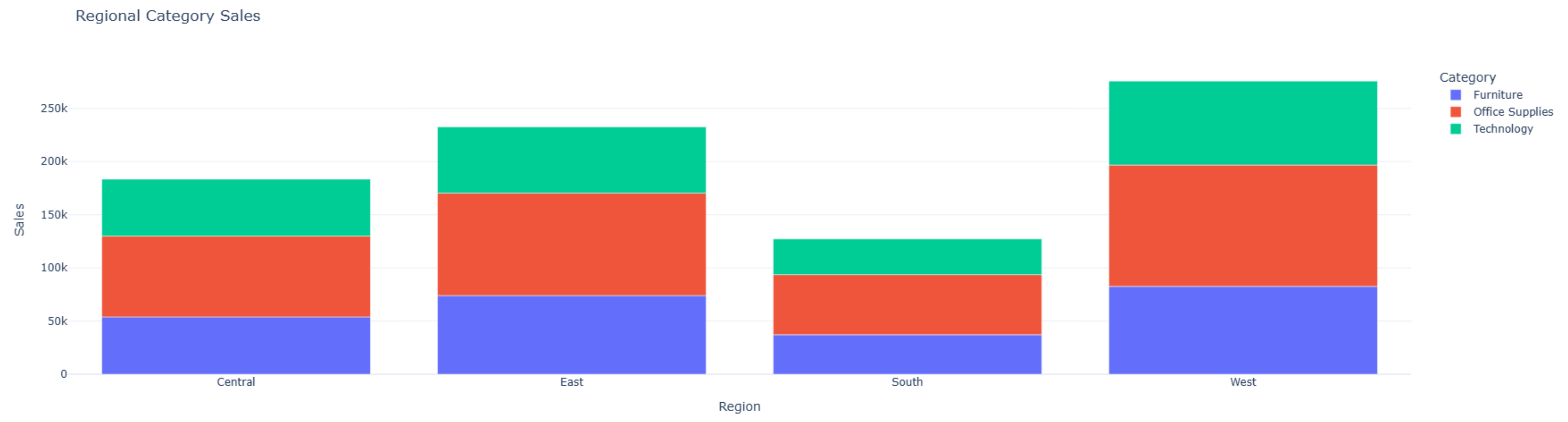
```
In [ ]: monthly_category_sales = df.groupby(['Month', 'Category'])['Sales'].sum().reset_index()
        fig = px.bar(monthly_category_sales, x='Month', y='Sales', color='Category',title='Monthly Category sales')
        fig.update_layout(xaxis = dict(tickmode = 'linear', tick0 = 1, dtick = 1))
        fig.show()
```



Regional Categorical Sales

In [ ]:

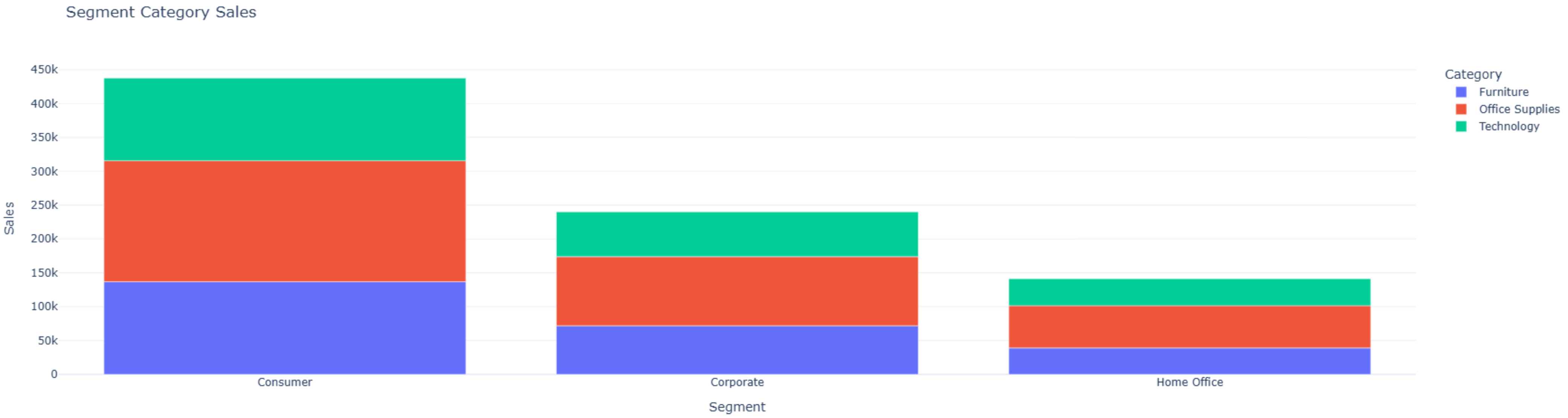
```
Regional_category_sales = df.groupby(['Region', 'Category'])['Sales'].sum().reset_index()
Regional_category_sales
fig=px.bar(Regional_category_sales, x='Region', y='Sales', color='Category', title='Regional Category Sales')
fig.show()
```



Segmential Categorical Sales

In [ ]:

```
Segment_category_sales = df.groupby(['Segment', 'Category'])['Sales'].sum().reset_index()
Segment_category_sales
fig=px.bar(Segment_category_sales, x='Segment', y='Sales', color='Category', title='Segment Category Sales')
fig.show()
```



# Profit Analysis

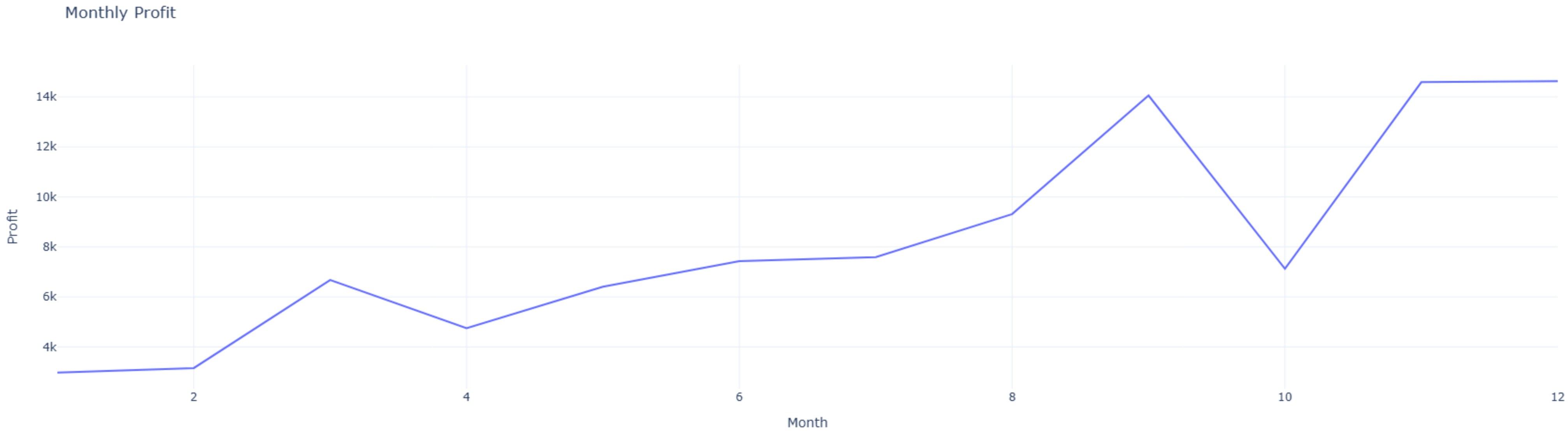
Monthly Profit Analysis

```
In [ ]: Month_profit=df.groupby('Month')['Profit'].sum().reset_index()
        print(Month_profit)
```

```
    Month       Profit
0       1    2974.4755
1       2    3156.3250
2       3    6679.3697
3       4    4748.7379
4       5    6408.6784
5       6    7432.7252
6       7    7592.7777
7       8    9314.4583
8       9   14055.0039
9      10    7128.7606
10     11   14588.1267
11     12   14628.7060
```

Monthly Profit Plot

```
In [ ]: fig=px.line(Month_profit, x='Month', y='Profit', title='Monthly Profit')
        fig.show()
```



Categorical and Sub-Categorical Profit Analysis

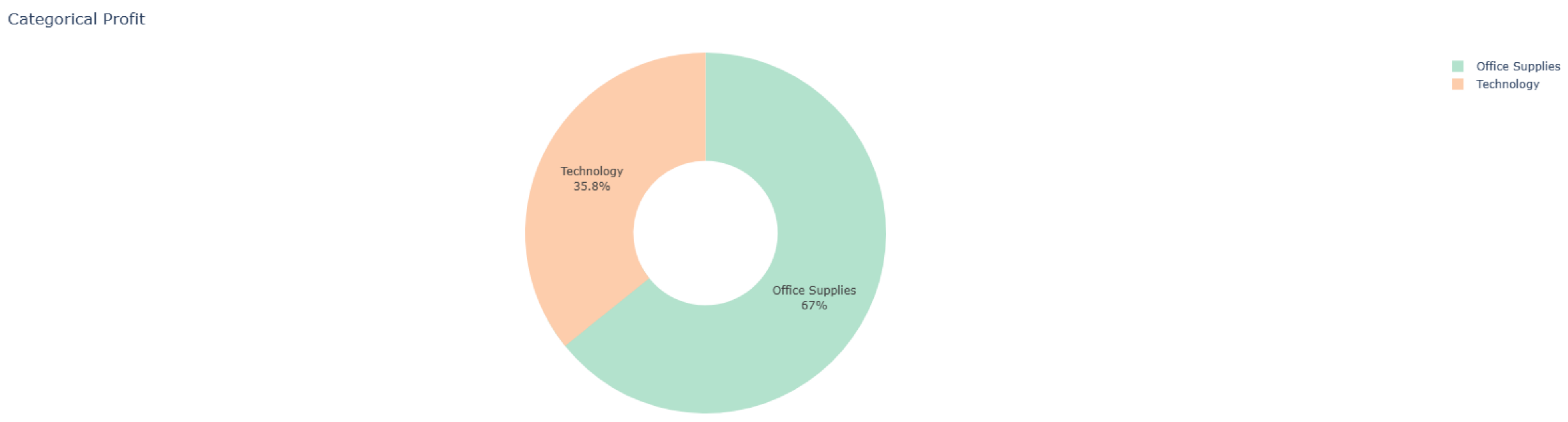
In [ ]:
```
Cate_profit=df.groupby('Category')['Profit'].sum().reset_index()
subcate_profit=df.groupby('Sub-Category')['Profit'].sum().reset_index()
print(Cate_profit)
print("\n",subcate_profit)
```

```
          Category      Profit
0        Furniture  -2702.4765
1  Office Supplies  66094.0111
2       Technology  35316.6103

    Sub-Category      Profit
0   Accessories  21247.9872
1    Appliances   2028.9706
2           Art   6416.4846
3       Binders   4859.8935
4     Bookcases  -4693.1152
5        Chairs   1638.1169
6       Copiers   1301.9544
7     Envelopes   6760.1053
8     Fasteners    949.5182
9   Furnishings   8988.7917
10       Labels   4932.7996
11     Machines  -1649.9229
12        Paper  32751.2701
13       Phones  14416.5916
14      Storage   6735.5406
15     Supplies    659.4286
16       Tables  -8636.2699
```

Category Profit Plot

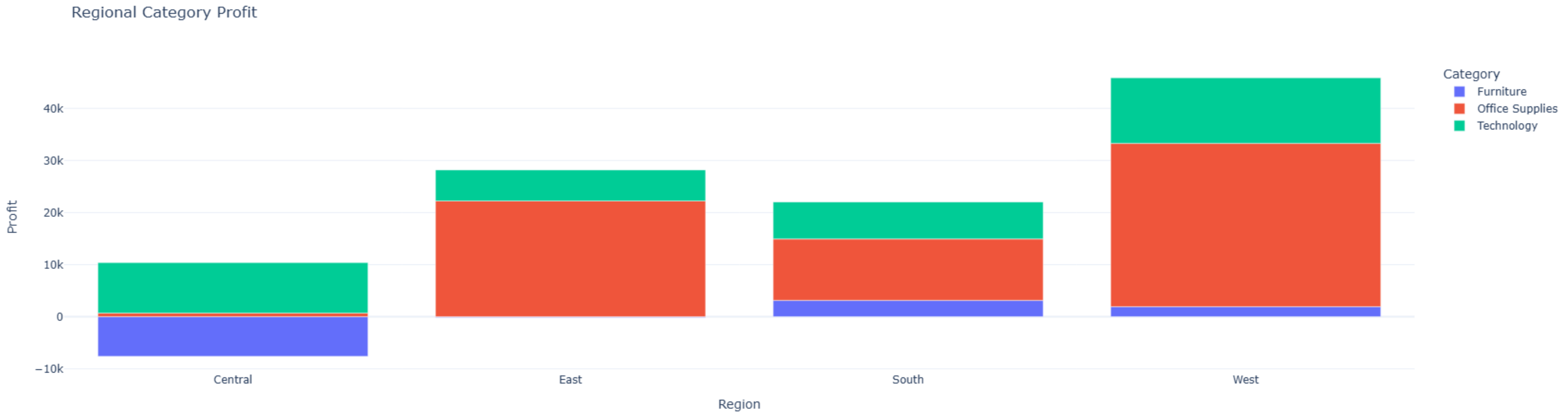
In [ ]:
```
fig=px.pie(Cate_profit, values='Profit', names='Category', hole=0.4, color_discrete_sequence=px.colors.qualitative.Pastel2)
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title='Categorical Profit')
fig.show()
```



Regional Categorical Profit

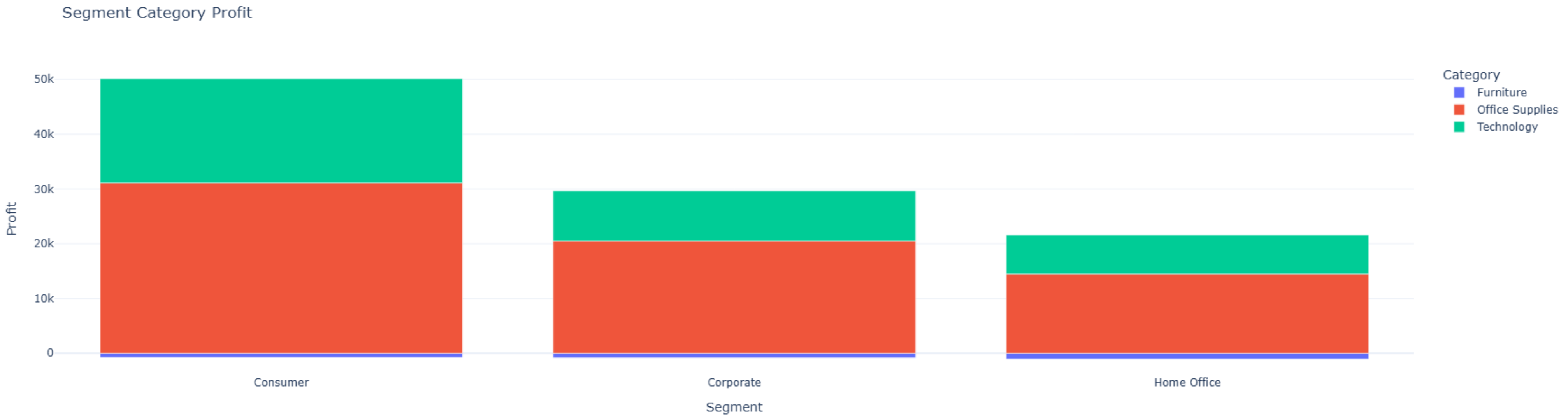
In [ ]:

```
Regional_category_Profit = df.groupby(['Region', 'Category'])['Profit'].sum().reset_index()
Regional_category_Profit
fig=px.bar(Regional_category_Profit, x='Region', y='Profit', color='Category', title='Regional Category Profit')
fig.show()
```



Segmential Categorical Profit

```
In [ ]: Segment_category_Profit = df.groupby(['Segment', 'Category'])['Profit'].sum().reset_index()
        Segment_category_Profit
        fig=px.bar(Segment_category_Profit, x='Segment', y='Profit', color='Category', title='Segment Category Profit')
        fig.show()
```

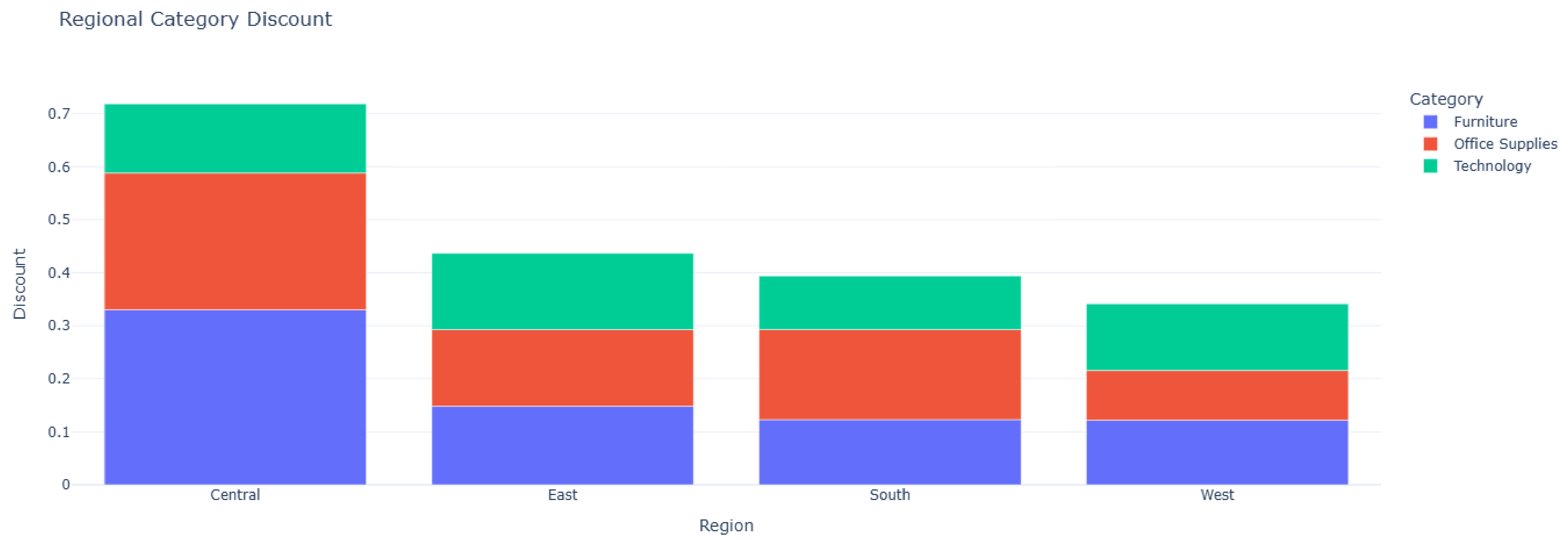


# Discount Analysis

Regional Discount per category

In [213…

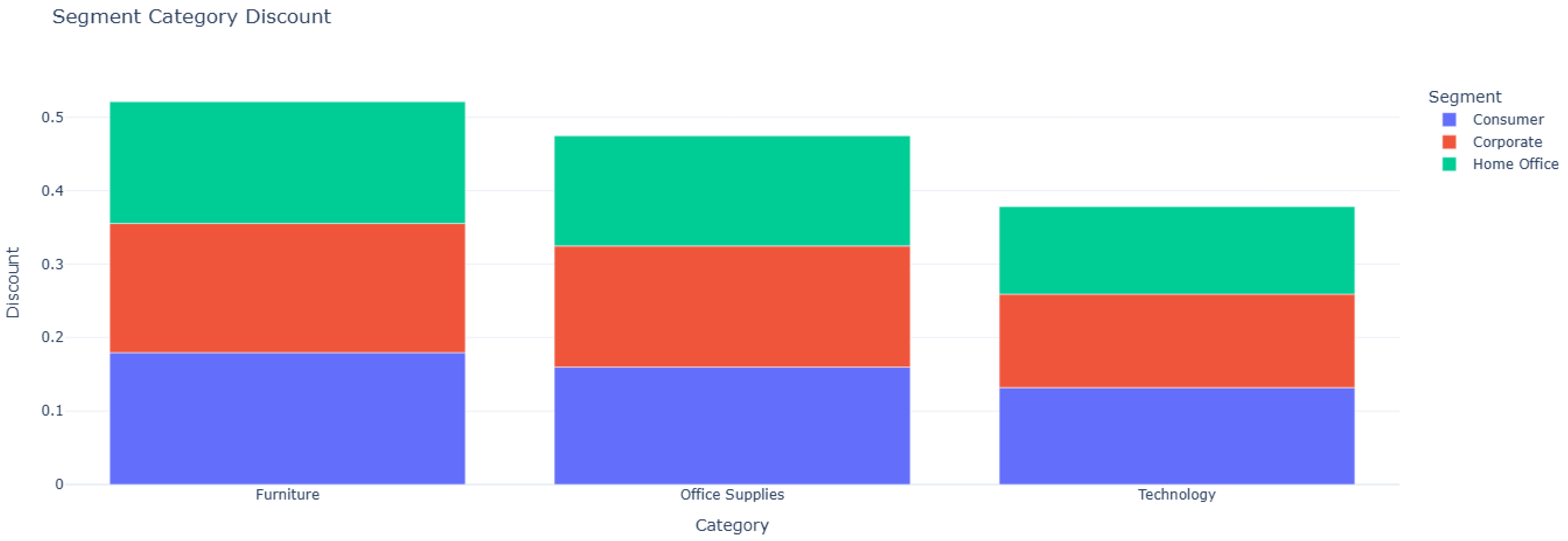
```
category_discount = df.groupby(['Region', 'Category'])['Discount'].mean().reset_index()
category_discount
fig=px.bar(category_discount, x='Region', y='Discount', color='Category', title='Regional Category Discount')
fig.show()
```



Segment Discount per Category

In [214…

```
category_discount = df.groupby(['Segment', 'Category'])['Discount'].mean().reset_index()
category_discount
fig=px.bar(category_discount, x='Category', y='Discount', color='Segment', title='Segment Category Discount')
fig.show()
```


Segment Category Discount
Segment
Consumer
Corporate
Home Office
Discount
0
0.1
0.2
0.3
0.4
0.5
Furniture
Office Supplies
Technology
Category