In [3]:

```
# Loading important libraries and packages
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import ttest_ind
import holidays
import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats
import warnings
import matplotlib.pyplot as plt
import plotly.offline as offline
offline.init_notebook_mode()

warnings.filterwarnings('ignore')
sns.set()
```

In [4]:

```
# Loading in the daTASET
df = pd.read_csv('superstore_train.csv')
df
```

Out[4]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Hen
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	A
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lauc
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Lauc
...	...	...	...	...	...	...	...	...	...	
7995	7996	US-2015-165743	2015-11-20	2015-11-23	Second Class	MM-18055	Michelle Moray	Consumer	United States	
7996	7997	US-2017-105998	2017-11-03	2017-11-05	First Class	CR-12580	Clay Rozendal	Home Office	United States	San
7997	7998	US-2017-105998	2017-11-03	2017-11-05	First Class	CR-12580	Clay Rozendal	Home Office	United States	San
7998	7999	US-2014-148194	2014-05-04	2014-05-07	First Class	BS-11365	Bill Shonely	Corporate	United States	
7999	8000	US-2014-148194	2014-05-04	2014-05-07	First Class	BS-11365	Bill Shonely	Corporate	United States	

8000 rows × 21 columns

In [5]:
```
# Getting information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         8000 non-null   int64  
 1   Order ID       8000 non-null   object 
 2   Order Date     8000 non-null   object 
 3   Ship Date      8000 non-null   object 
 4   Ship Mode      8000 non-null   object 
 5   Customer ID    8000 non-null   object 
 6   Customer Name  8000 non-null   object 
 7   Segment        8000 non-null   object 
 8   Country        8000 non-null   object 
 9   City           8000 non-null   object 
 10  State          8000 non-null   object 
 11  Postal Code    8000 non-null   int64  
 12  Region         8000 non-null   object 
 13  Product ID     8000 non-null   object 
 14  Category       8000 non-null   object 
 15  Sub-Category   8000 non-null   object 
 16  Product Name   8000 non-null   object 
 17  Sales          8000 non-null   float64
 18  Quantity       8000 non-null   int64  
 19  Discount       8000 non-null   float64
 20  Profit         8000 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.3+ MB
```

In [6]:
```
# Checking for missing values in the dataset.
df.isna().sum()
```

Out[6]:
```
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
dtype: int64
```

There are no missing or null values in the dataset.

In [7]:
```
# Checking for duplicated values
print('There are {} duplicated rows in the dataset'.format(df.duplicated().sum()))
```

```
There are 0 duplicated rows in the dataset
```

In [8]:
```
# Getting the unique count of values in each columns in the dataset.
uni_cols = df.nunique().reset_index()
uni_cols.columns = ['Columns', 'Count of Unique Values']
uni_cols
```

Out[8]:

	Columns	Count of Unique Values
0	Row ID	8000
1	Order ID	3962
2	Order Date	1181
3	Ship Date	1279
4	Ship Mode	4
5	Customer ID	784
6	Customer Name	784
7	Segment	3
8	Country	1
9	City	502
10	State	49
11	Postal Code	598
12	Region	4
13	Product ID	1831
14	Category	3
15	Sub-Category	17
16	Product Name	1819
17	Sales	5186
18	Quantity	14
19	Discount	12
20	Profit	6295

In [9]:
```
# Getting summary statistics of the numerical columns in the dataset
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Row ID	8000.0	4000.500000	2309.545410	1.000	2000.7500	4000.5000	6000.2500	8000.000
Postal Code	8000.0	55225.850000	32049.060112	1040.000	22980.0000	55901.0000	90008.0000	99301.000
Sales	8000.0	229.823459	621.031927	0.444	17.3400	54.3520	211.8700	22638.480
Quantity	8000.0	3.786875	2.211349	1.000	2.0000	3.0000	5.0000	14.000
Discount	8000.0	0.152864	0.203951	0.000	0.0000	0.1500	0.2000	0.800
Profit	8000.0	28.134233	227.328814	-6599.978	1.8147	8.7651	29.9505	8399.976

Summary:

- The average sales per transaction is about 230 dollars with a standard deviation of 54.35 dollars. This huge value of standard deviation shows there is a huge spread of values in the dataset. This is confirmed as it can be seen that the maximum sales per transaction recorded is 22,638 dollars and a minimum sales per transaction of about 44 cents.

- The average quantity of goods bought per transaction is approximately 3.79 units with a standard deviation of about 2.2 units. The minimum quantity of goods per transaction is just one unit and a maximum of about 14 units.

- The average profit per transaction is about 28 dollars with a standard deviation of 227.33 dollars. The minimum profit per transaction is a loss of 6599.978 dollars. This is due to discounts The maximum profit per transaction is 8399.976 dollars.

In [10]:
```
# Converting the 'Order Date column to a datetime format to aid further analysis'
df['Order Date'] = pd.to_datetime(df['Order Date'])
```

In [11]:
```
# Converting the 'Ship Date column to a datetime format to aid further analysis'
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

In [12]:
```
# Getting information about the dataset to see changes.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         8000 non-null   int64
 1   Order ID       8000 non-null   object
 2   Order Date     8000 non-null   datetime64[ns]
 3   Ship Date      8000 non-null   datetime64[ns]
 4   Ship Mode      8000 non-null   object
 5   Customer ID    8000 non-null   object
 6   Customer Name  8000 non-null   object
 7   Segment        8000 non-null   object
 8   Country        8000 non-null   object
 9   City           8000 non-null   object
 10  State          8000 non-null   object
 11  Postal Code    8000 non-null   int64
 12  Region         8000 non-null   object
 13  Product ID     8000 non-null   object
 14  Category       8000 non-null   object
 15  Sub-Category   8000 non-null   object
 16  Product Name   8000 non-null   object
 17  Sales          8000 non-null   float64
 18  Quantity       8000 non-null   int64
 19  Discount       8000 non-null   float64
 20  Profit         8000 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.3+ MB
```

The 'Order Date' and 'Ship Date' columns has been successfully converted to a datetime format.

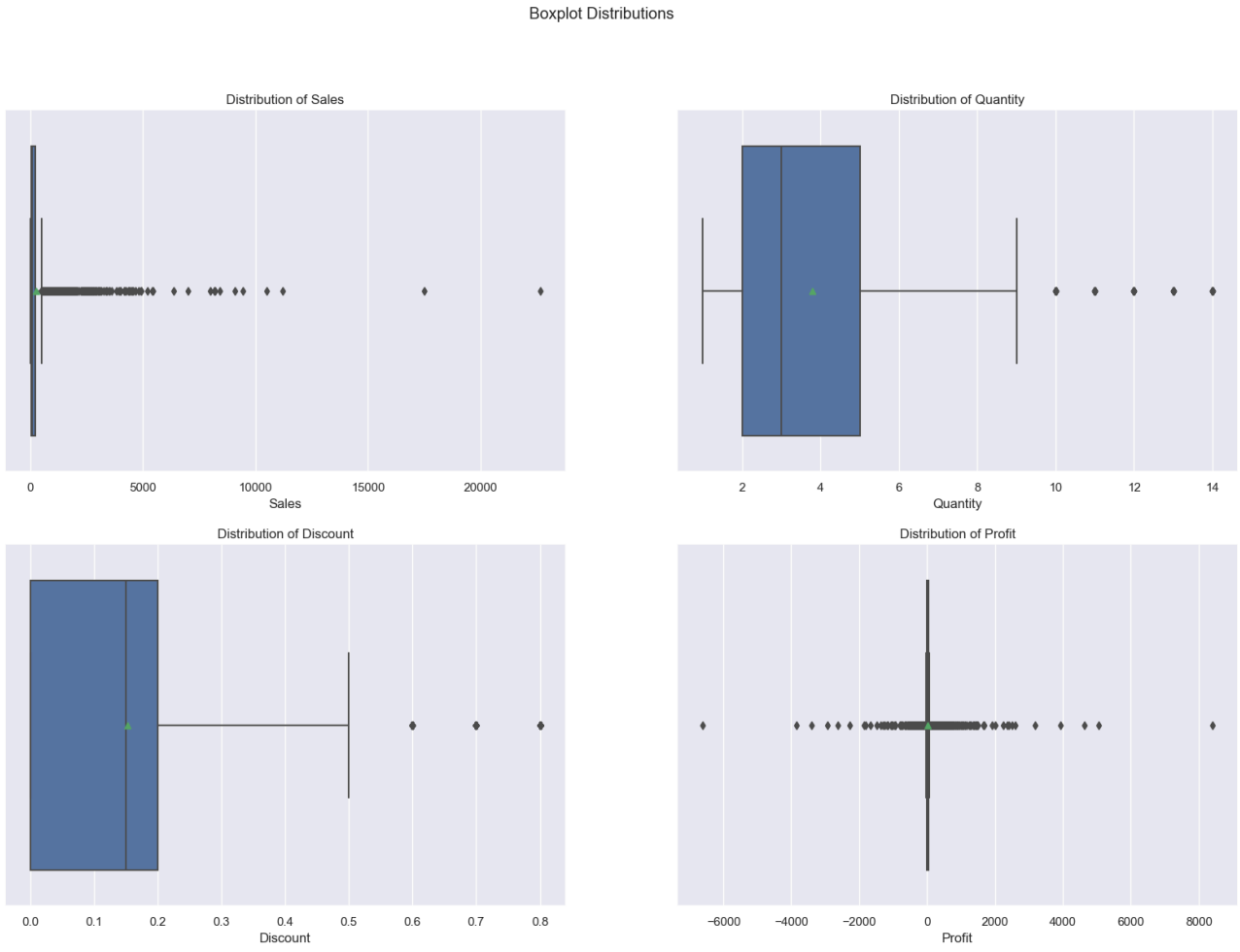
In [13]:
```
# Convverting the 'Segment', 'Region', 'Ship Mode','Category', 'Sub-Category' colum
df[['Segment', 'Region', 'Ship Mode', 'Category', 'Sub-Category']] = df[['Segment',
                                                                        'Region',
```

In [14]:
```
# Getting information about the columns in the dataset to see further changes.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Row ID         8000 non-null   int64         
 1   Order ID       8000 non-null   object        
 2   Order Date     8000 non-null   datetime64[ns]
 3   Ship Date      8000 non-null   datetime64[ns]
 4   Ship Mode      8000 non-null   category      
 5   Customer ID    8000 non-null   object        
 6   Customer Name  8000 non-null   object        
 7   Segment        8000 non-null   category      
 8   Country        8000 non-null   object        
 9   City           8000 non-null   object        
 10  State          8000 non-null   object        
 11  Postal Code    8000 non-null   int64         
 12  Region         8000 non-null   category      
 13  Product ID     8000 non-null   object        
 14  Category       8000 non-null   category      
 15  Sub-Category   8000 non-null   category      
 16  Product Name   8000 non-null   object        
 17  Sales          8000 non-null   float64       
 18  Quantity       8000 non-null   int64         
 19  Discount       8000 non-null   float64       
 20  Profit         8000 non-null   float64       
dtypes: category(5), datetime64[ns](2), float64(3), int64(3), object(8)
memory usage: 1.0+ MB
```

The 'Segment', 'Region', 'Ship Mode','Category', 'Sub-Category' columns has been successfully converted to a categorical format.

In [15]:
```
num_df = df.drop(['Row ID', 'Postal Code'], axis=1).select_dtypes(
    include=['int', 'float']).columns

#  Categorical Data
a = 2  # number of rows
b = 2  # number of columns
c = 1  # initialize plot counter

fig = plt.figure(figsize=(20, 13))
for i in num_df:
    plt.subplot(a, b, c)
    plt.title('Distribution of {}'.format(i))
    plt.xlabel(i)
    sns.boxplot(data=df, x=i, showmeans=True)
    c = c + 1


fig.suptitle('Boxplot Distributions')
# fig.tight_layout()
plt.show()
```



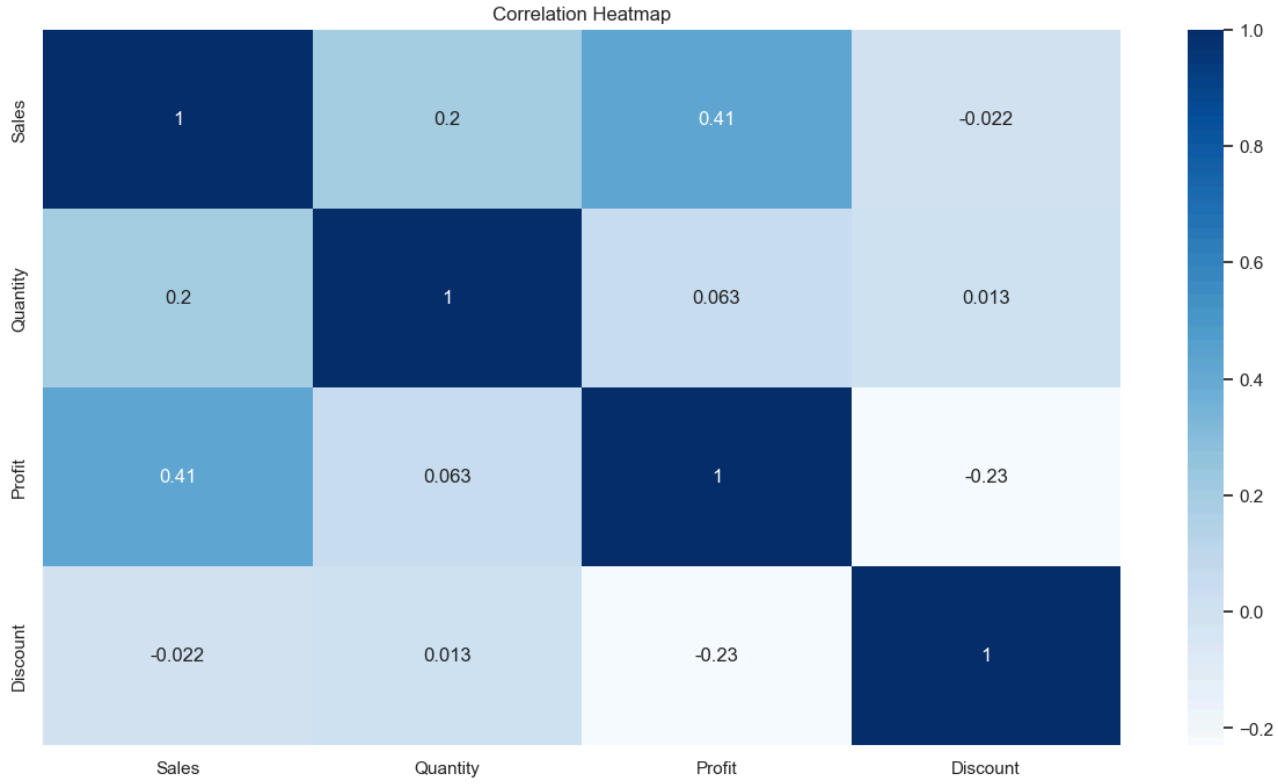
There are huge outlier values in the sales and profit column. This is due to the nature of the high variance between these two columns.

In [16]:
```
df[['Sales', 'Quantity', 'Profit', 'Discount']].corr()
```

Out[16]:

	Sales	Quantity	Profit	Discount
Sales	1.000000	0.196896	0.414750	-0.021971
Quantity	0.196896	1.000000	0.063334	0.013312
Profit	0.414750	0.063334	1.000000	-0.228720
Discount	-0.021971	0.013312	-0.228720	1.000000

In [17]:
```
plt.figure(figsize = (15, 8))
sns.heatmap(df[['Sales', 'Quantity', 'Profit', 'Discount']].corr(), annot = True, c
plt.title('Correlation Heatmap')
plt.show()
```



Summary:

- There is a positive correlation of approximately 0.197 between sales and quantity. This suggests that as the quantity of items sold increases, sales also tend to increase, but the correlation is not very strong.
- There is a positive correlation of approximately 0.415 between sales and profit. This indicates that higher sales are associated with higher profits, with a moderate positive relationship.
- There is a negative correlation of approximately -0.022 between sales and discount. This suggests that there is a weak negative relationship between the amount of discount applied and the sales amount, meaning higher discounts may slightly decrease sales, though the correlation is close to zero.
- There is a very weak positive correlation of approximately 0.063 between quantity and profit. This indicates that there's a slight tendency for higher quantities sold to be associated with higher profits, but the correlation is very weak.
- There is a very weak positive correlation of approximately 0.013 between quantity and discount. This suggests that there's almost no relationship between the quantity sold and the amount of discount applied.

- There is a negative correlation of approximately -0.229 between profit and discount. This indicates

In [18]:
```
total_sales = df['Sales'].sum()
print('The total sales is ${}'.format(total_sales))
```

```
The total sales is $1838587.6724
```

In [19]:
```
total_profit = df['Profit'].sum()
print('The total profit gained is ${}'.format(total_profit))
```

```
The total profit gained is $225073.86259999996
```

In [20]:
```
total_quantity = df['Quantity'].sum()
print('The total quantity of goods sold is {}'.format(total_quantity))
```

```
The total quantity of goods sold is 30295
```

Summary:

Over the course of four years;

- The total sales made is 1838587.67 dollars
- The total profit gained over the course of four years is 225073.86 dollars
- The total quantities of goods bought is 30, 295 units.

1 Product Analysis

In [21]:
```
# Getting the amount of times each
df['Product Name'].value_counts()
```

Out[21]:
```
Staples                                                                 43
Staple envelope                                                         40
Easy-staple paper                                                       33
Avery Non-Stick Binders                                                 20
Staples in misc. colors                                                 17
                                                                        ..
Avery 5                                                                  1
Sauder Facets Collection Locker/File Cabinet, Sky Alder Finish           1
Star Micronics TSP800 TSP847IIU Receipt Printer                          1
Nu-Dell Oak Frame                                                        1
Bush Westfield Collection Bookcases, Dark Cherry Finish, Fully Assembled  1
Name: Product Name, Length: 1819, dtype: int64
```

In [22]:
```
# Getting the top 10 products in terms of sales.
product_sales = df.groupby('Product Name').agg({'Sales': 'sum',
                                                'Quantity': 'sum',
                                                'Profit': 'sum'}).reset_index()
top_performing_products = product_sales.sort_values(
    'Sales', ascending=False).head(10)
print('Top 10 products in terms of Sales')
top_performing_products
```

```
Top 10 products in terms of Sales
```

Out[22]:

	Product Name	Sales	Quantity	Profit
399	Canon imageCLASS 2200 Advanced Copier	47599.864	16	18479.9472
438	Cisco TelePresence System EX90 Videoconferenci...	22638.480	6	-1811.0784
792	Hewlett Packard LaserJet 3310 Copier	17039.716	33	6743.8876
799	High Speed Automatic Electric Letter Opener	17030.312	11	-262.0048
968	Lexmark MX611dhe Monochrome Laser Printer	16829.901	18	-4589.9730
1026	Martin Yale Chadless Opener Electric Letter Op...	16656.200	22	-1299.1836
639	Fellowes PB500 Electric Punch Plastic Comb Bin...	16014.474	20	3304.5740
881	Ibico EPK-21 Electric Binding System	15875.916	13	3345.2823
672	GBC DocuBind P400 Electric Binding System	15787.484	19	1823.7266
18	3D Systems Cube Printer, 2nd Generation, Magenta	14299.890	11	3717.9714

In [23]:
```
# Getting the least 10 products in terms of Sales
under_performing_products = product_sales.sort_values(
    'Sales', ascending=True).head(10)
print('Top 10 Worst performing products in terms of Sales')
under_performing_products
```

```
Top 10 Worst performing products in terms of Sales
```

Out[23]:

	Product Name	Sales	Quantity	Profit
848	Hoover Commercial Lightweight Upright Vacuum	1.392	2	-3.7584
597	Eureka Disposable Bags for Sanitaire Vibra Gro...	1.624	2	-4.4660
809	Holmes Visible Mist Ultrasonic Humidifier with...	2.264	1	-5.2072
1627	Wirebound Voice Message Log Book	3.808	1	1.2376
853	Hoover Replacement Belt for Commercial Guardsm...	3.996	3	-0.6660
525	Dixon Ticonderoga Pencils	4.768	2	0.4172
1095	Newell 308	5.040	3	1.2600
204	Avery 5	5.760	2	2.8224
1744	Xerox 1983	5.980	1	2.9302
949	Kleencut Forged Office Shears by Acme United C...	6.240	3	1.8720

In [24]:
```
# # Getting the least 10 products in terms of Profits
top_profitable_products = product_sales.sort_values(
    'Profit', ascending=False).head(10)
print('Top 10 Most Profitable Products')
top_profitable_products
```

```
Top 10 Most Profitable Products
```

Out[24]:

	Product Name	Sales	Quantity	Profit
399	Canon imageCLASS 2200 Advanced Copier	47599.864	16	18479.9472
792	Hewlett Packard LaserJet 3310 Copier	17039.716	33	6743.8876
161	Ativa V4110MDD Micro-Cut Shredder	7699.890	11	3772.9461
18	3D Systems Cube Printer, 2nd Generation, Magenta	14299.890	11	3717.9714
1253	Plantronics Savi W720 Multi-Device Wireless He...	9367.290	24	3696.2820
881	Ibico EPK-21 Electric Binding System	15875.916	13	3345.2823
639	Fellowes PB500 Electric Punch Plastic Comb Bin...	16014.474	20	3304.5740
393	Canon Imageclass D680 Copier / Fax	8959.872	16	2799.9600
791	Hewlett Packard 610 Color Digital Copier / Pri...	6999.860	16	2599.9480
638	Fellowes PB300 Plastic Comb Binding Machine	8070.192	27	2518.0551

In [25]:
```
# Getting the worst performing products in terms of profits
least_profitable_products = product_sales.sort_values(
    'Profit', ascending=True).head(10)
print('Top 10 least profitable products')
least_profitable_products
```

```
Top 10 least profitable products
```

Out[25]:

	Product Name	Sales	Quantity	Profit
469	Cubify CubeX 3D Printer Double Head Print	11099.9630	9	-8879.9704
968	Lexmark MX611dhe Monochrome Laser Printer	16829.9010	18	-4589.9730
470	Cubify CubeX 3D Printer Triple Head Print	7999.9800	4	-3839.9904
371	Bush Advantage Collection Racetrack Conference...	5217.7830	21	-2087.1132
438	Cisco TelePresence System EX90 Videoconferenci...	22638.4800	6	-1811.0784
1327	Riverside Palais Royal Lawyers Bookcase, Royal...	11206.0656	19	-1682.6718
677	GBC Ibimaster 500 Manual ProClick Binding System	4413.6840	16	-1674.1560
1026	Martin Yale Chadless Opener Electric Letter Op...	16656.2000	22	-1299.1836
691	GBC ProClick 150 Presentation Binding System	884.7440	13	-1292.3582
16	3.6 Cubic Foot Counter Height Office Refrigerator	471.3920	8	-1225.6192

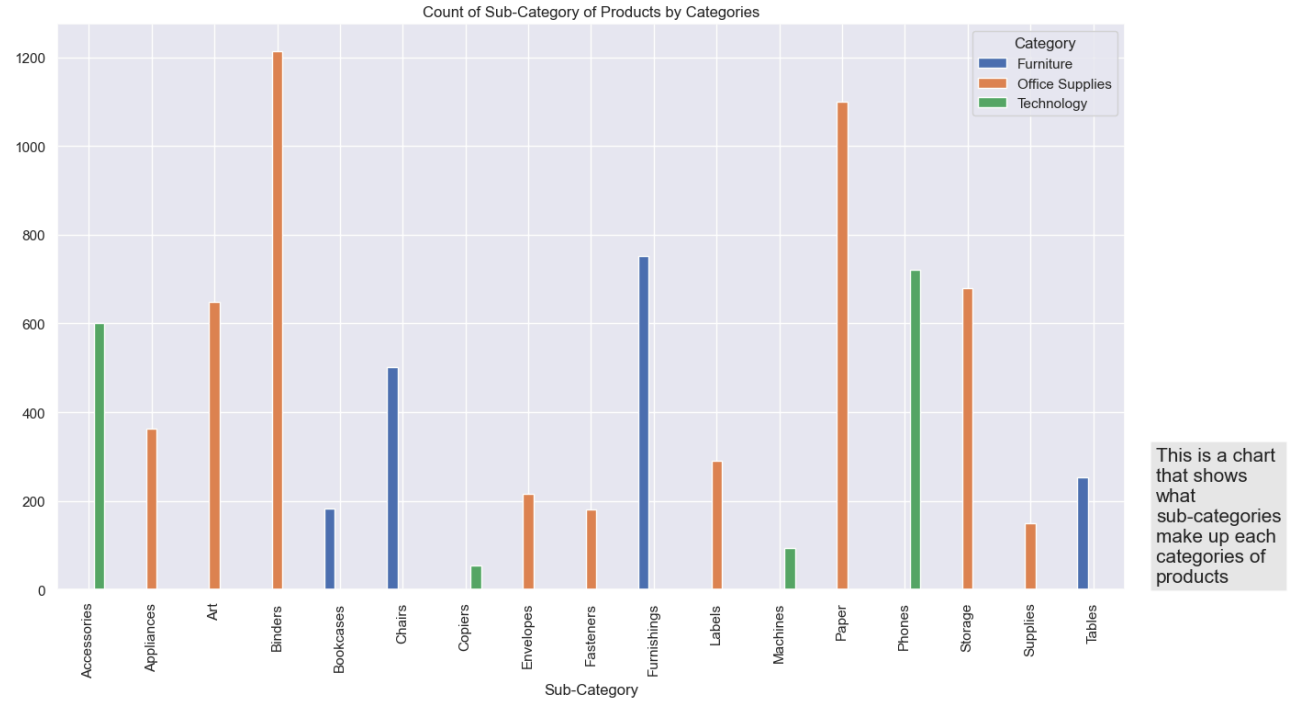
1.1 Summary:

- The product with the most sales is the 'Canon imageCLASS 2200 Advanced Copier' which made a sales of 47,599.86 dollars. This is about 2.6% of the total sales made over the years. This product is also the most profitable as it made a profit about 18480 dollars.

- The sales of the top ten best performing goods amount to 10.8% of the total sales.
- The product with the least sales is the 'Hoover Commercial Lightweight Upright Vacuum' which made a sales of 1.39 dollars, has only 2 units sold and made a loss of 3.86 dollars. _ The commonly ordered product is 'Staples' which has a count of 43 and 203 quantities ordered.

2 Category and Sub-Category Analysis.

In [26]:
```
# Checking out what sub-categories makes up each categories of products
pd.crosstab(index=df['Sub-Category'], columns=df['Category']
            ).plot(kind='bar', figsize=(15, 8))
plt.title('Count of Sub-Category of Products by Categories')
plt.text(17, 10, s='This is a chart that shows what sub-categories make up each cat
         bbox=dict(facecolor='lightgray', alpha=0.5), va='bottom', wrap=True)
plt.show()
```



In [27]:
```
category_sales = df.groupby('Category').agg({'Sales': 'sum',
                                             'Quantity': 'sum',
                                             'Profit': 'sum'}).reset_index()
category_sales = category_sales.sort_values('Sales', ascending=False)
category_sales
```

Out[27]:

	Category	Sales	Quantity	Profit
2	Technology	679802.1860	5534	111435.7920
0	Furniture	587825.1654	6426	16875.2353
1	Office Supplies	570960.3210	18335	96762.8353

The technology category has the most sale with a total sale of 679802.19 dollars which is about 37% of the total sales generated and the highest profit generated of about 111, 435.79 dollars which makes up more than 49% of total profits generated. The 'Office Supplies' category has the least sales with 570960.32 dollars generated. The 'Office Supplies' category made the second highest total profit with 96762.84 dollars generated which contributes to 43% of the total profit generated.

In [28]:
```
# Checking the high performing subcategories of products on the basis of the sales
sub_category_sales = df.groupby('Sub-Category').agg({'Sales': 'sum',
                                                     'Quantity': 'sum',
                                                     'Profit': 'sum'}).reset_index(
sub_category_sales = sub_category_sales.sort_values('Sales', ascending=False)
sub_category_sales
```

Out[28]:

	Sub-Category	Sales	Quantity	Profit
13	Phones	267838.3340	2668	36534.3185
5	Chairs	264689.1690	1908	22956.2429
14	Storage	179766.5260	2534	17622.5872
16	Tables	162418.1700	1001	-13392.9439
11	Machines	159003.2800	379	-3024.8580
3	Binders	146508.3560	4733	20775.3364
0	Accessories	133812.1700	2299	34178.3898
6	Copiers	119148.4020	188	43747.9417
4	Bookcases	87723.1144	702	-3539.4843
1	Appliances	87090.6030	1333	14819.7536
9	Furnishings	72994.7120	2815	10851.4206
12	Paper	64097.9980	4152	27916.9717
15	Supplies	44201.8140	508	-1239.6738
2	Art	21913.1080	2399	5243.7949
7	Envelopes	14332.5060	774	6119.9244
10	Labels	10540.8600	1146	4685.5425
8	Fasteners	2508.5500	756	818.5984

The 'phones' subcategory has the most sales and made the most profit of all subcategories. The least performing subcategory in terms of sales is the 'fasteners' subcategory, while in terms of profit, the least performing is the 'Tables' subcategory.

In [29]:
```
# Creating a column called 'profit margin' to hold the value of the profit margin f
df['profit margin'] = round((df['Profit'] / df['Sales']) * 100, 2)
```

In [30]:
```
df.head()
```

Out[30]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	H
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	H
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	

In [31]:
```
df['profit margin'].describe()
```

Out[31]:
```
count    8000.000000
mean       12.796767
std        45.621084
min      -275.000000
25%         7.500000
50%        27.000000
75%        36.250000
max        50.000000
Name: profit margin, dtype: float64
```

The mean profit margin per order is about 12.8%. The is a occurence of a profit margin of -275% which is the minimum profit margin ever recorded. This will be further investigated.

In [32]:
```
df[df['profit margin'] == df['profit margin'].min()]
```

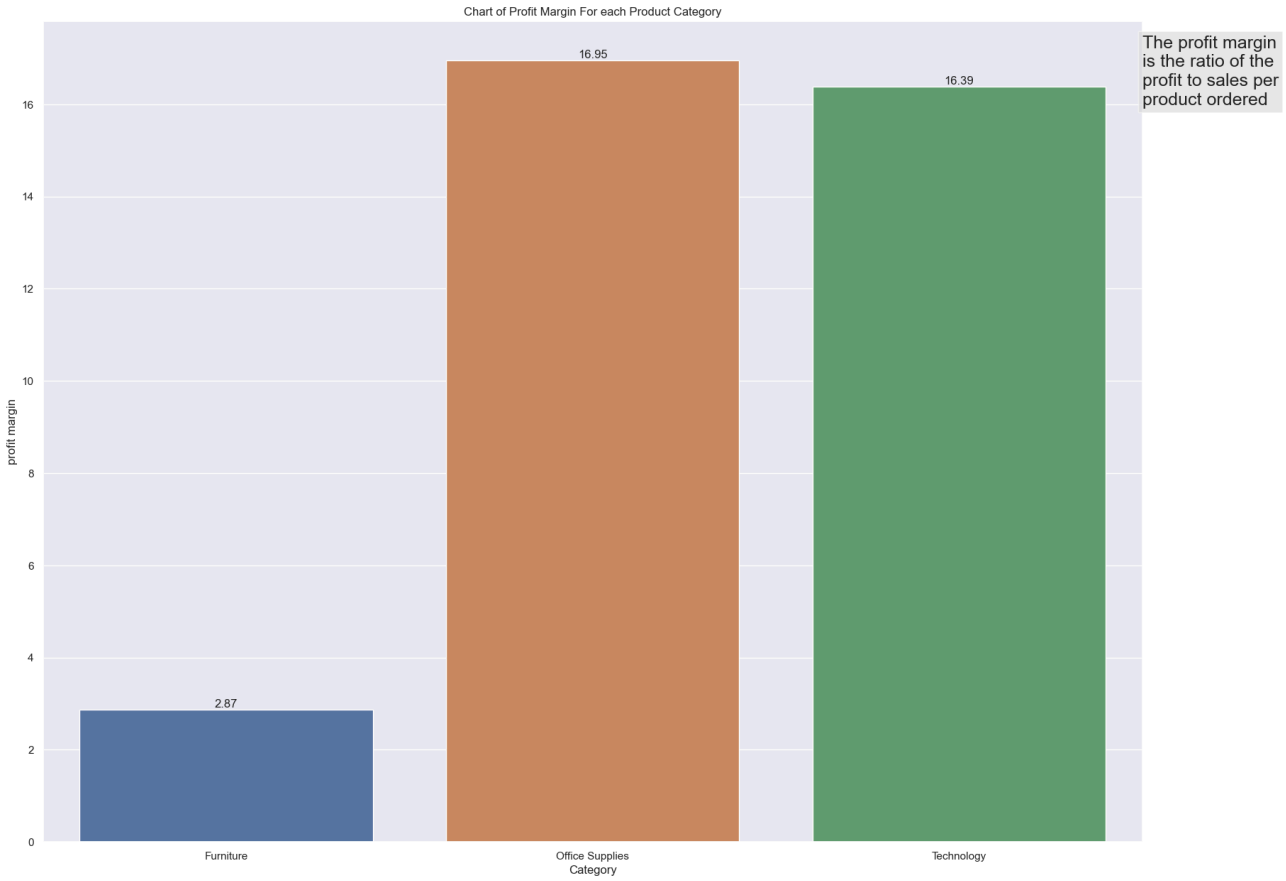
Out[32]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Ci
261	262	US-2017-155299	2017-06-08	2017-06-12	Standard Class	DI-13600	Dorris liebe	Corporate	United States	Pasa
676	677	US-2017-119438	2017-03-18	2017-03-23	Standard Class	CD-11980	Carol Darley	Consumer	United States	

2 rows × 22 columns

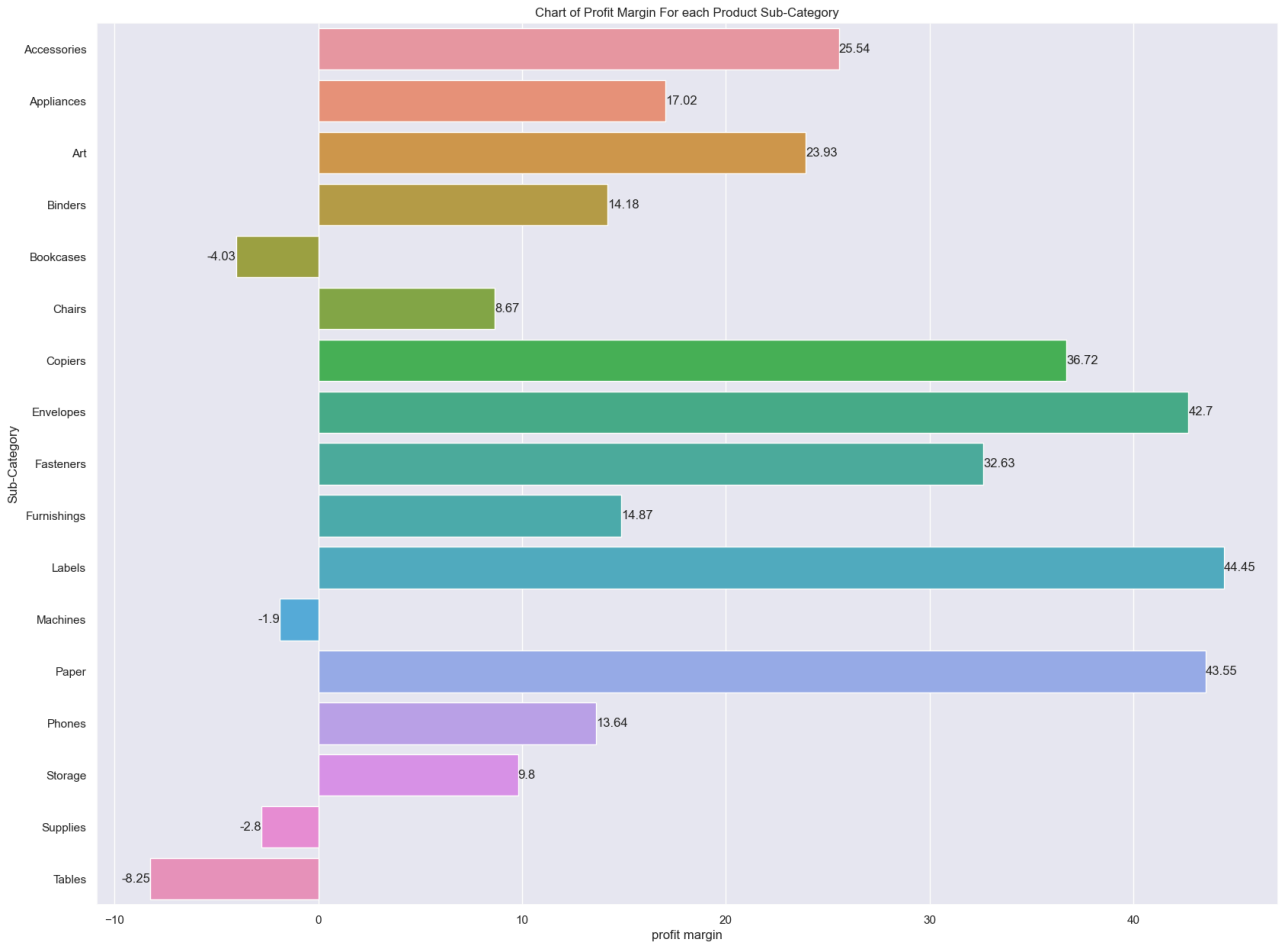
The minimum profit margin recorded was as a result of the 80% of the products bought. The two products bought having this minimum are both under the 'appliance' category.

In [33]:
```
category_sales['profit margin'] = round(
    (category_sales['Profit'] / category_sales['Sales']) * 100, 2)

fig = plt.figure(figsize=(20, 15))
ax = sns.barplot(x='Category', y='profit margin', data=category_sales)
plt.title('Chart of Profit Margin For each Product Category')
plt.text(2.5, 17.5, s='The profit margin is the ratio of the profit to sales per pr
         bbox=dict(facecolor='lightgray', alpha=0.5), va='top', wrap=True)
for container in ax.containers:
    ax.bar_label(container)
```



The 'office supplies' category has the highest profit margin of 16.95% closely followed by the 'Technology' category with a profit margin of 16.39%. The 'furniture' category has the lowest profit margin of 2.87%.

In [34]:
```
# Graph of profit margin for each product sub-categories.
sub_category_sales['profit margin'] = round(
    (sub_category_sales['Profit'] / sub_category_sales['Sales']) * 100, 2)

fig = plt.figure(figsize=(20, 15))
ax = sns.barplot(y='Sub-Category', x='profit margin',
                 data=sub_category_sales, errorbar=None)
plt.title('Chart of Profit Margin For each Product Sub-Category')
for container in ax.containers:
    ax.bar_label(container)
```



The 'Labels' subcategory has the highest profit margin of over 44% despite the 'Phones' subcategory having the highest sales and profit generated. This means that the 'Labels' subcategory is a highly profitable sub-category. This is closely followed by the 'Paper' subcategory. The 'Phones' subcategory with the highest sales and profit generated has a profit margin of about 13.64%. This shows that it is not as highly profitable as the 'paper', 'labels', 'Envelopes' and 'Fasteners' columns. The top 4 subcategories with the highest profit margin all fall under the 'Office Supplies' category.

Summary:

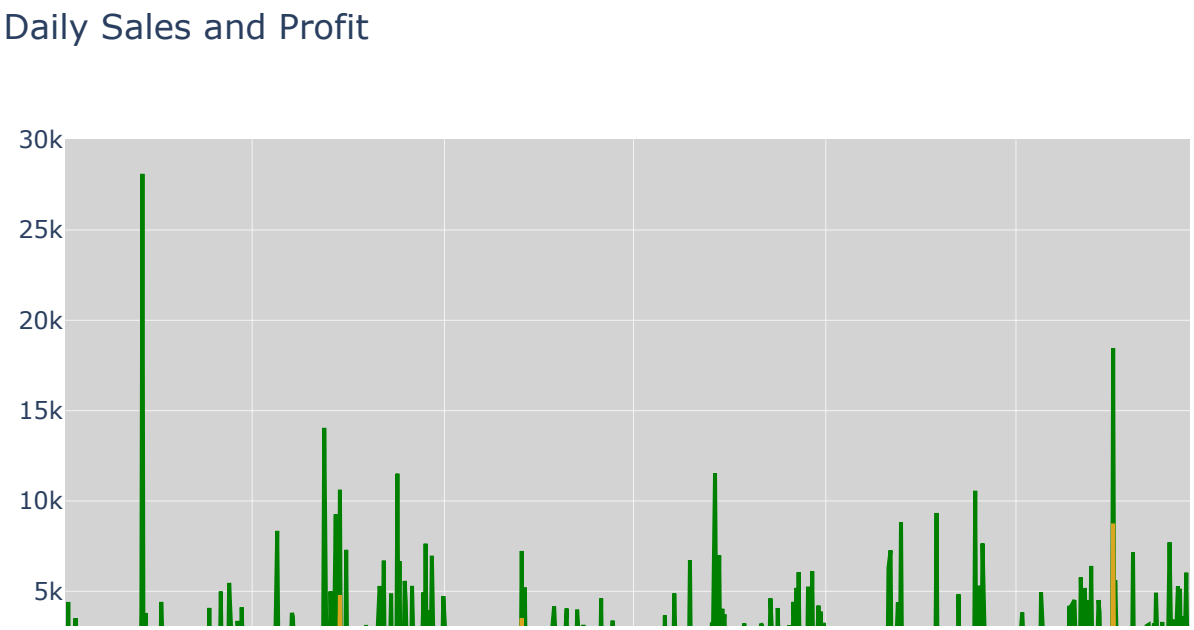
- The technology category has the most sale with a total sale of 679802.19 dollars and the highest profit generated of about 111,435.79 dollars.
- The 'Office Supplies' category has the least sales with 570,960.32 dollars generated. The 'Office Supplies' category made the second highest total profit with 96,762.84 dollars generated.
- The 'Phones' subcategory has the most sales and made the most profit of all subcategories but has the tenth highest profit margin.
- The least performing subcategory in terms of sales is the 'fasteners' subcategory, while in terms of profit, the least performing is the 'Tables' subcategory.
- The 4 most profitable subcategories i.e categories with the highest profit margin all fall under the 'Office Supplies' category.
- The 'Labels' subcategory has the highest profit margin of over 44%.

- The subcategory with the least profit margin is the 'Tables' subcategory with a profit margin of -8.25%

3 Time Series Analysis

In [35]:
```
# Grouping sales and profit by 'Order Date'
daily_sales = df.groupby('Order Date')['Sales'].sum().reset_index()
daily_profit = df.groupby('Order Date')['Profit'].sum().reset_index()
```

In [36]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=daily_sales['Order Date'], y=daily_sales['Sales'], mode=
                         name='Daily_Sales', line={'color': 'Green'}))
fig.add_trace(go.Scatter(x=daily_profit['Order Date'], y=daily_profit['Profit'], mo
                         name='Daily_Profit', line={'color': 'goldenrod'}))
fig.update_layout(plot_bgcolor='lightgray',
                  title="Daily Sales and Profit"
                  )
offline.iplot(fig)
```



The highest sales ever generated in one day across the years was on March 18 2014, this is followed by the sales generated on October 2 2016 which also has the maximum profit generated. The highest loss ever generated was on the November 25 2016 which was a 'Black Friday'.

In [37]:

```
# Investigating the transactions on November 25 2016
df[df['Order Date'] == '2016-11-25']
```

Out[37]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
3463	3464	CA-2016-165470	2016-11-25	2016-11-30	Standard Class	HJ-14875	Heather Jas	Home Office	United States	Jack
6573	6574	CA-2016-160535	2016-11-25	2016-11-30	Standard Class	BP-11050	Barry Pond	Corporate	United States	P
6574	6575	CA-2016-160535	2016-11-25	2016-11-30	Standard Class	BP-11050	Barry Pond	Corporate	United States	P
6977	6978	US-2016-119298	2016-11-25	2016-11-28	First Class	EP-13915	Emily Phan	Consumer	United States	Jo
7686	7687	CA-2016-169838	2016-11-25	2016-11-29	Standard Class	BB-11545	Brenda Bowman	Corporate	United States	
7687	7688	CA-2016-169838	2016-11-25	2016-11-29	Standard Class	BB-11545	Brenda Bowman	Corporate	United States	
7688	7689	CA-2016-169838	2016-11-25	2016-11-29	Standard Class	BB-11545	Brenda Bowman	Corporate	United States	
7771	7772	CA-2016-108196	2016-11-25	2016-12-02	Standard Class	CS-12505	Cindy Stewart	Consumer	United States	La
7772	7773	CA-2016-108196	2016-11-25	2016-12-02	Standard Class	CS-12505	Cindy Stewart	Consumer	United States	La
7773	7774	CA-2016-108196	2016-11-25	2016-12-02	Standard Class	CS-12505	Cindy Stewart	Consumer	United States	La

10 rows × 22 columns

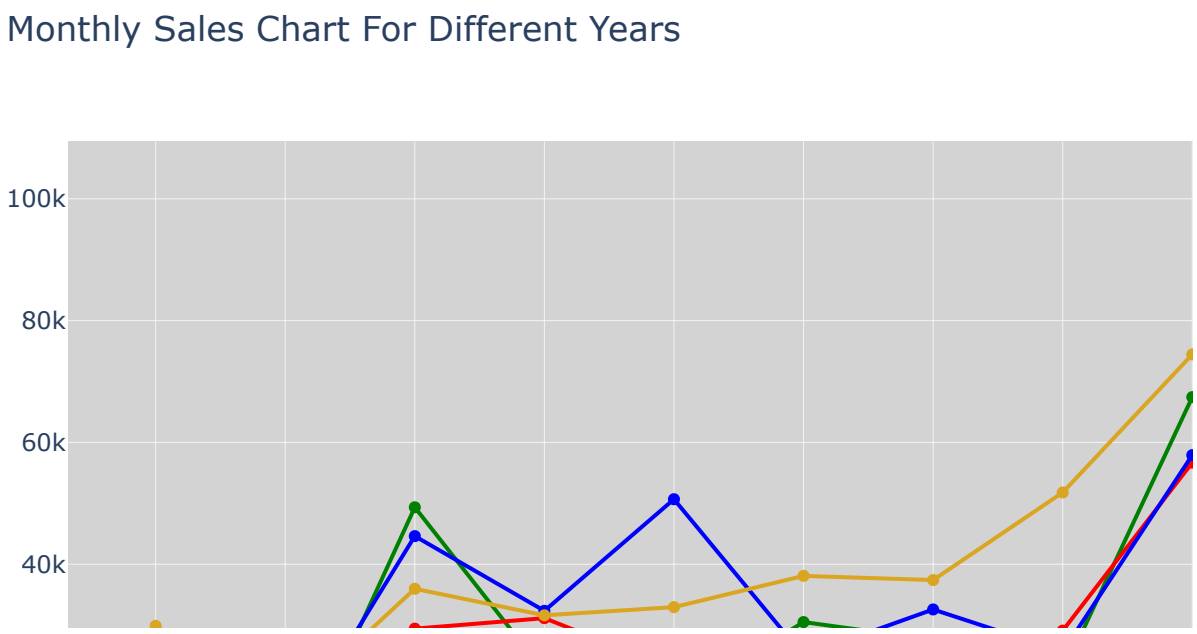
On this particular black friday, the maximum loss of 6599.98 dollars was recorded due to a 70% discount on prices. Two particular transactions with a discount of 70% occured on this particular black friday.

In [38]:
```
# Creating a column called 'month' to hold the month the product were ordered.
df['month'] = df['Order Date'].dt.month_name()
```

In [39]:
```
# splitting the dataframe into four based on year
df_2014 = df[df['Order Date'].dt.year == 2014]
df_2015 = df[df['Order Date'].dt.year == 2015]
df_2016 = df[df['Order Date'].dt.year == 2016]
df_2017 = df[df['Order Date'].dt.year == 2017]
```

In [40]:
```
new_order = ['January', 'February', 'March', 'April', 'May', 'June',
             'July', 'August', 'September', 'October', 'November', 'December']


# Grouping each split dataframe on month
sales_2014 = df_2014.groupby('month').agg({'Sales': 'sum', 'Quantity': 'sum',
                                           'Profit': 'sum'}).reindex(new_order, axi
sales_2015 = df_2015.groupby('month').agg({'Sales': 'sum', 'Quantity': 'sum',
                                           'Profit': 'sum'}).reindex(new_order, axi
sales_2016 = df_2016.groupby('month').agg({'Sales': 'sum', 'Quantity': 'sum',
                                           'Profit': 'sum'}).reindex(new_order, axi
sales_2017 = df_2017.groupby('month').agg({'Sales': 'sum', 'Quantity': 'sum',
                                           'Profit': 'sum'}).reindex(new_order, axi
```

In [41]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=sales_2014['month'], y=sales_2014['Sales'], mode='lines
fig.add_trace(go.Scatter(x=sales_2015['month'], y=sales_2015['Sales'], mode='lines
fig.add_trace(go.Scatter(x=sales_2016['month'], y=sales_2016['Sales'], mode='lines
fig.add_trace(go.Scatter(x=sales_2017['month'], y=sales_2017['Sales'], mode='lines
fig.update_layout(plot_bgcolor='lightgray',↔
offline.iplot(fig)
```



Analysis of the graph reveals an overall positive trend in sales performance over the four-year period. There is a noticeable upward trajectory, indicating consistent growth in sales over time.

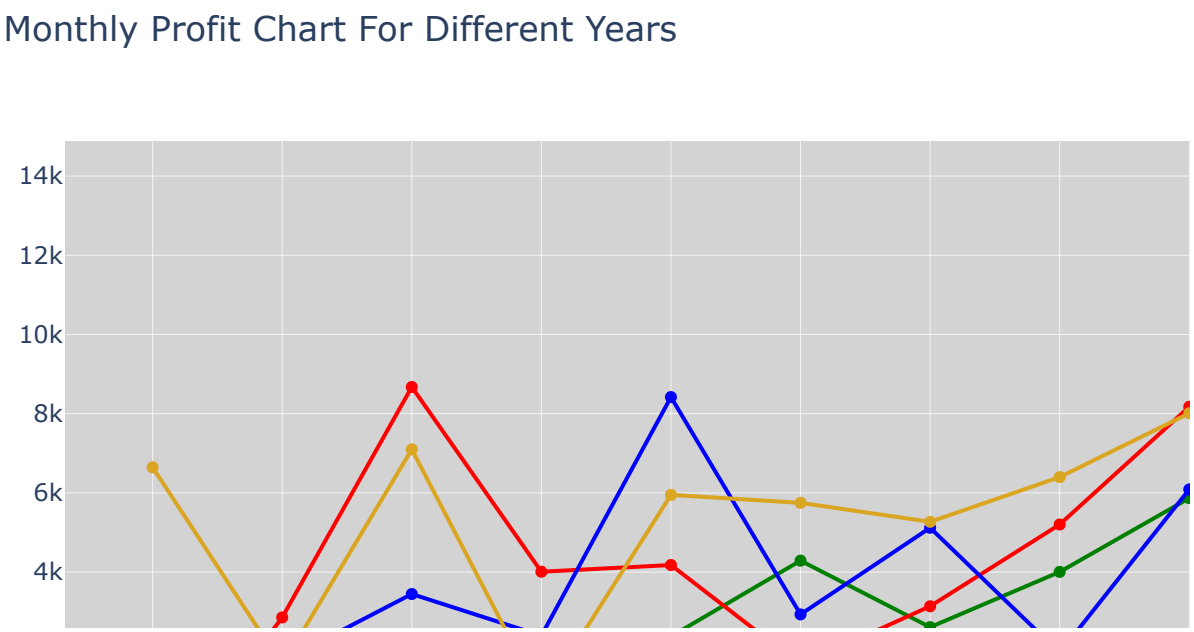
Year-on-year comparison highlights variations in sales performance. For instance, while 2014 exhibited the lowest growth, 2017 experienced a significant surge in sales, likely attributed to the high demand of new products and effective marketing campaigns.

- Subsequent years maintained the upward trend, albeit with varying degrees of growth.

Monthly sales patterns indicate fluctuations influenced by seasonal factors and promotional activities. For example, months like (March, September, November) for each year consistently record higher sales volumes, driven by increased consumer spending and promotional events.

- Conversely, sales figures typically dip during the months like (October, August, February) for each year, reflecting reduced consumer activity and demand.

In [42]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=sales_2014['month'], y=sales_2014['Profit'], mode='lines
                         name='2014', line={'color': 'Green'}))
fig.add_trace(go.Scatter(x=sales_2015['month'], y=sales_2015['Profit'], mode='lines
                         name='2015', line={'color': 'red'}))
fig.add_trace(go.Scatter(x=sales_2016['month'], y=sales_2016['Profit'], mode='lines
                         name='2016', line={'color': 'blue'}))
fig.add_trace(go.Scatter(x=sales_2017['month'], y=sales_2017['Profit'], mode='lines
                         name='2017', line={'color': 'goldenrod'}))
fig.update_layout(plot_bgcolor='lightgray',
                  title="Monthly Profit Chart For Different Years"
                  )
offline.iplot(fig)
```



Analysis of the graph reveals an overall positive trend in profit gaiined over the four-year period. There is a noticeable upward trajectory, indicating consistent growth in profit over time.

Year-on-year comparison highlights variations in profit performance.

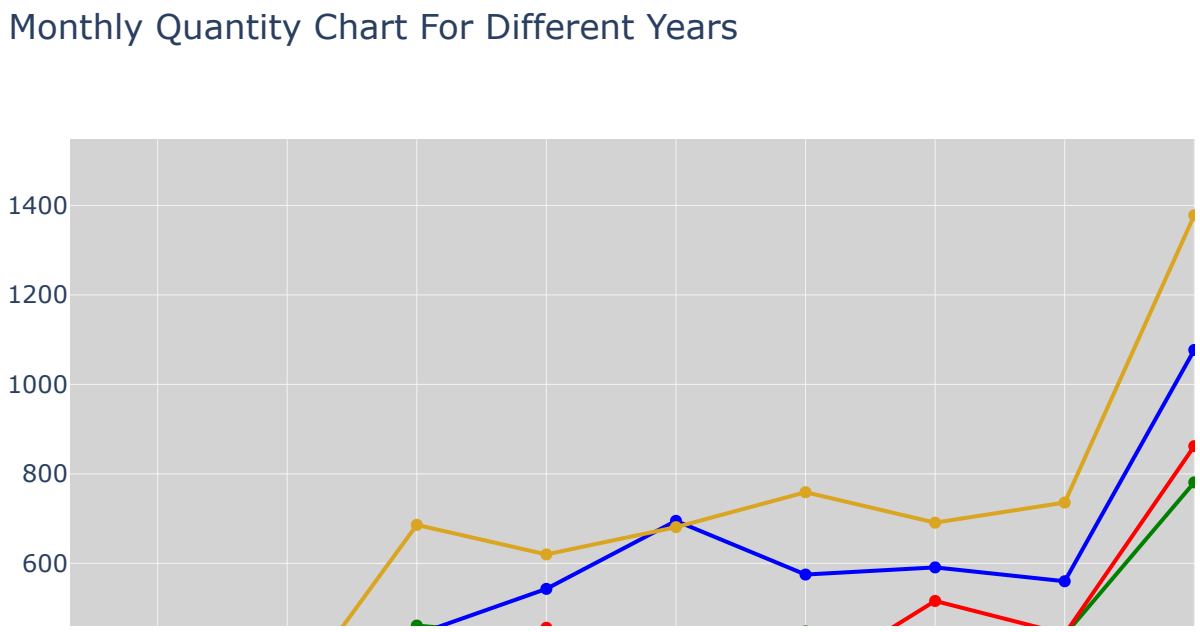
- For example, a huge loss can be seen at the early months of 2014, 2015 and also 2017.
- 2016 saw robust profitability, which can be attributed to strong market demand higher than those of previous years.

Monthly profit patterns exhibit fluctuations influenced by seasonal factors, market dynamics, and operational efficiencies. For instance, peak profit months coincide with high-demand periods, while lower profit months may result from low demands.

- The chart depicts a monthly profit trend over a four-year period. It reveals a seasonal pattern with higher profits in the months like (March, September, November) with variation across the years. For instance, 2014 didnot experience this general peak in sales for the month of March and also 2016 didnot experience this peak for the month of November. October 2016 produced the highest

profit ever till date. This pattern suggests potential seasonality in the company's sales or service offerings.

In [43]:
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=sales_2014['month'], y=sales_2014['Quantity'], mode='lir
fig.add_trace(go.Scatter(x=sales_2015['month'], y=sales_2015['Quantity'], mode='lir
fig.add_trace(go.Scatter(x=sales_2016['month'], y=sales_2016['Quantity'], mode='lir
fig.add_trace(go.Scatter(x=sales_2017['month'], y=sales_2017['Quantity'], mode='lir
fig.update_layout(plot_bgcolor='lightgray',
                  title="Monthly Quantity Chart For Different Years"
                  )
offline.iplot(fig)
```



Analysis of the graph reveals an overall positive trend in quantities demanded over the four-year period. There is a noticeable upward trajectory, indicating consistent growth in demand over time.

Year-on-year comparison highlights variations in sales demand. For instance, while 2014 exhibited the lowest growth, 2017 experienced a significant surge in sales, likely attributed to effective marketing campaigns.

Subsequent years maintained the upward trend, albeit with varying degrees of growth. Monthly sales patterns indicate fluctuations influenced by seasonal factors and promotional activities. For example, months like (March, September, November) for each year consistently record higher demand volumes.

Conversely, sales figures typically dip during the months like (October, August, February) for each year, reflecting reduced consumer activity and demand.

In [44]:
```
# Creating a column named Year to hold the year a transaction was carried out.
df['Year'] = df['Order Date'].dt.year
```

In [45]:
```
# Grouping the dataframe by year
df_year = df.groupby('Year').agg({'Sales': 'sum', 'Quantity': 'sum',
                                  'Profit': 'sum'}).reset_index()
df_year['profit_margin'] = round(
    (df_year['Profit'] / df_year['Sales']) * 100, 2)
df_year['Sales Growth'] = round(df_year['Sales'].pct_change() * 100, 2)
df_year['Profit Growth'] = round(df_year['Profit'].pct_change() * 100, 2)
df_year['Demand Growth'] = round(df_year['Quantity'].pct_change() * 100, 2)
df_year.fillna(0)
```

Out[45]:

	Year	Sales	Quantity	Profit	profit_margin	Sales Growth	Profit Growth	Demand Growth
0	2014	410701.4016	6099	43807.4519	10.67	0.00	0.00	0.00
1	2015	372506.5321	6342	51005.2479	13.69	-9.30	16.43	3.98
2	2016	468985.3225	7769	59842.3817	12.76	25.90	17.33	22.50
3	2017	586394.4162	10085	70418.7811	12.01	25.03	17.67	29.81

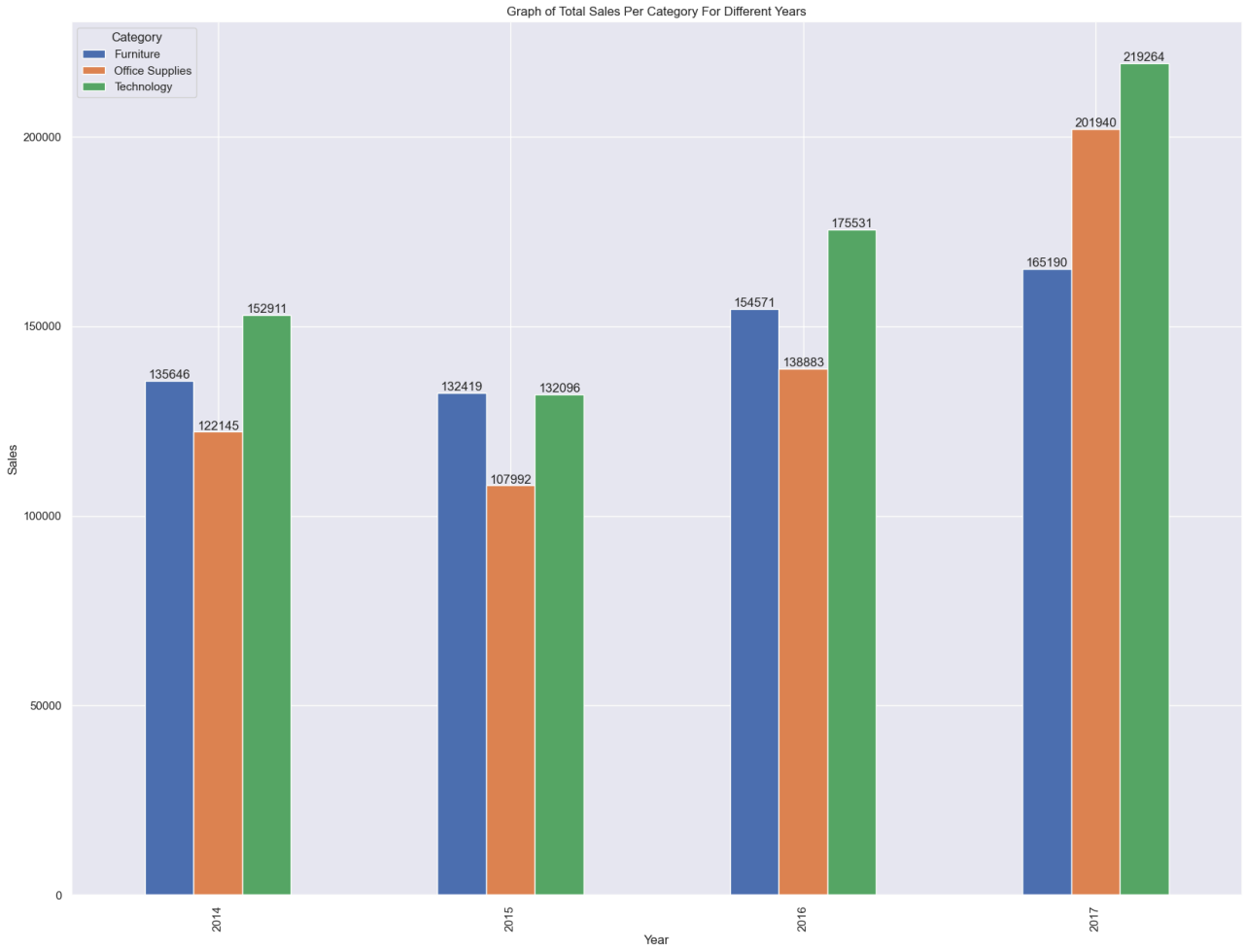
Summary:

- In 2014, the superstore achieved a moderate level of sales and profit. The profit margin, which indicates the efficiency of the superstore in generating profit from sales, was at 10.67%, reflecting a decent profitability level.
- In 2015, there was a decrease in sales by 9.3%, which might indicate a challenging market environment or internal factors affecting sales performance. However, despite the decrease in sales, there was a notable increase in profit by 16.43%, leading to an improved profit margin of 13.69%. The demand growth also showed a modest increase of 3.98%.
- In 2016, the superstore experienced significant growth in sales by 25.9%, indicating a strong performance and possibly successful market strategies or increased demand. Profit also grew by 17.33%, maintaining a relatively stable profit margin of 12.76%. The substantial increase in demand growth by 22.5% suggests a favorable market response to the superstore's products.
- In 2017, both sales and profit continued to grow, with sales increasing by 12.01% and profit by 25.03%. Despite the growth, the profit margin slightly decreased to 12.01%, possibly due to increased expenses or changes in pricing strategies. The demand growth remained strong at 17.67%, indicating sustained market interest in the superstore's products.

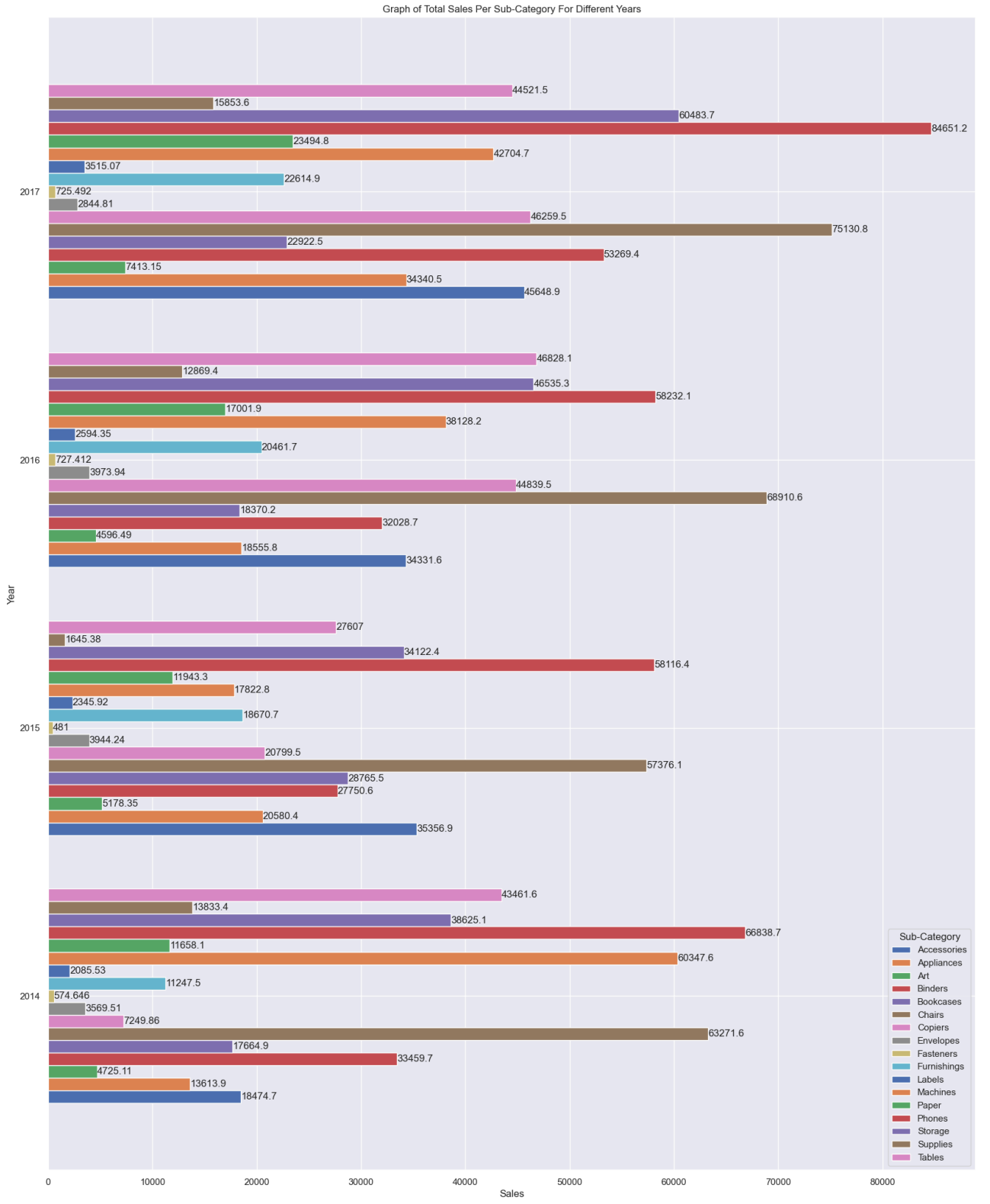
Overall, the company showed a positive trend of growth in sales, profit, and demand over the years, with some fluctuations in performance metrics such as profit margin. However, the general trajectory suggests an improving and healthy business performance during the period.

In [46]:

```
ax = df.pivot_table(index='Year', columns='Category',↔
for container in ax.containers:↔
plt.title('Graph of Total Sales Per Category For Different Years')
plt.ylabel('Sales')
plt.show()
```



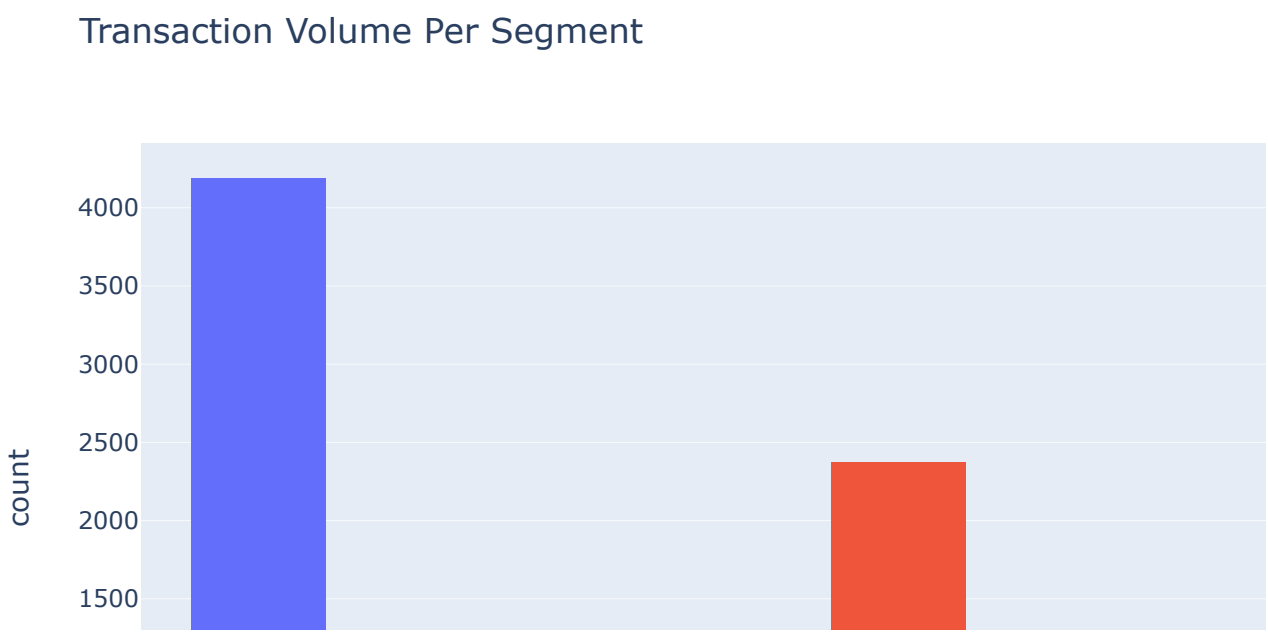
In [47]:
```
ax = df.pivot_table(index='Year', columns='Sub-Category',
                    values='Sales', aggfunc='sum').plot(kind='barh', width=0.8, fig
for container in ax.containers:
    ax.bar_label(container)
plt.title('Graph of Total Sales Per Sub-Category For Different Years')
plt.xlabel('Sales')
plt.show()
```



Generaly, there is a positive trend for total sales per category across the years. In 2015, the total sales in each category was at a minimum. This is also seen in the chart containing the Total Sales Per Sub-Category For Different Years while all sub categories had the lowest sales in 2015.

4 Segment Analysis

In [48]:
```
# Plotting a countplot for the 'Segment' categories.
fig = px.histogram(df, x='Segment', color='Segment',
                   barmode='group', title = 'Transaction Volume Per Segment')
offline.iplot(fig)
```



In [49]:
```
df_seg = df.groupby('Segment').agg({'Sales': 'sum', 'Quantity': 'sum',
                                    'Profit': 'sum'}).reset_index()
df_seg['profit_margin'] = round((df_seg['Profit'] / df_seg['Sales']) * 100, 2)

df_seg
```

Out[49]:

	Segment	Sales	Quantity	Profit	profit_margin
0	Consumer	906473.0977	15802	103683.1362	11.44
1	Corporate	564515.3362	9129	69434.1787	12.30
2	Home Office	367599.2385	5364	51956.5477	14.13

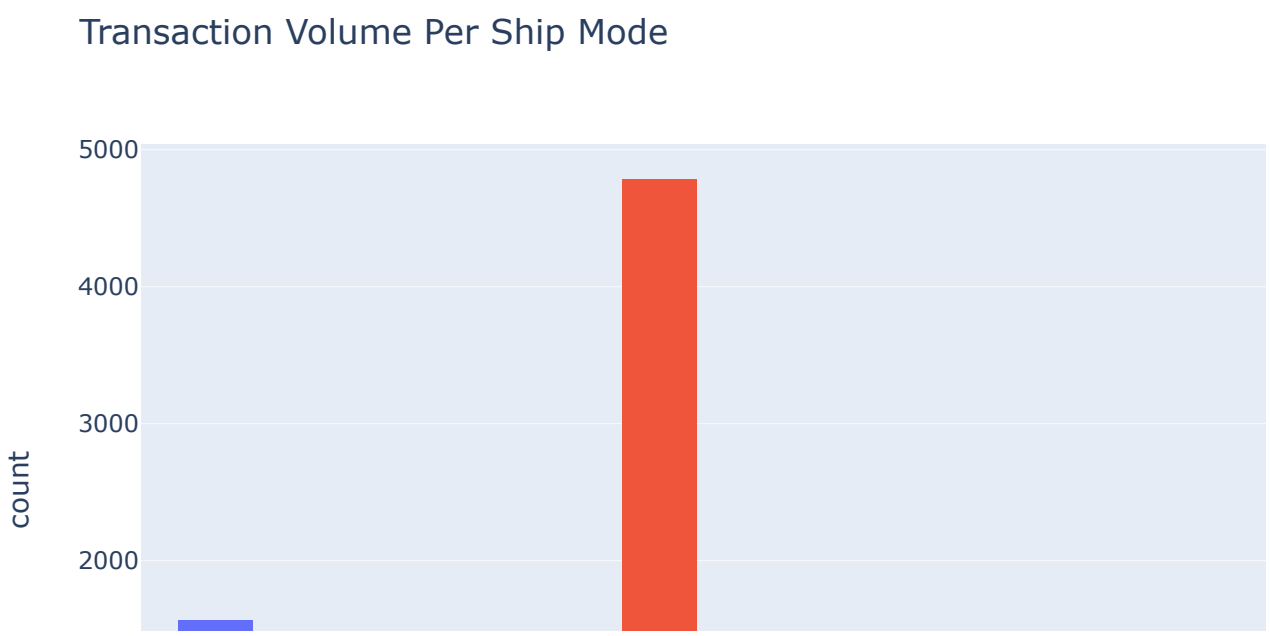
Summary:

- Consumer Segment: This segment has the highest number of transactions of 4192 transactions. Generated the highest sales revenue of approximately 906,473.10, with 15,802 products sold. The profit earned was 103,683.14, resulting in a profit margin of 11.44%.

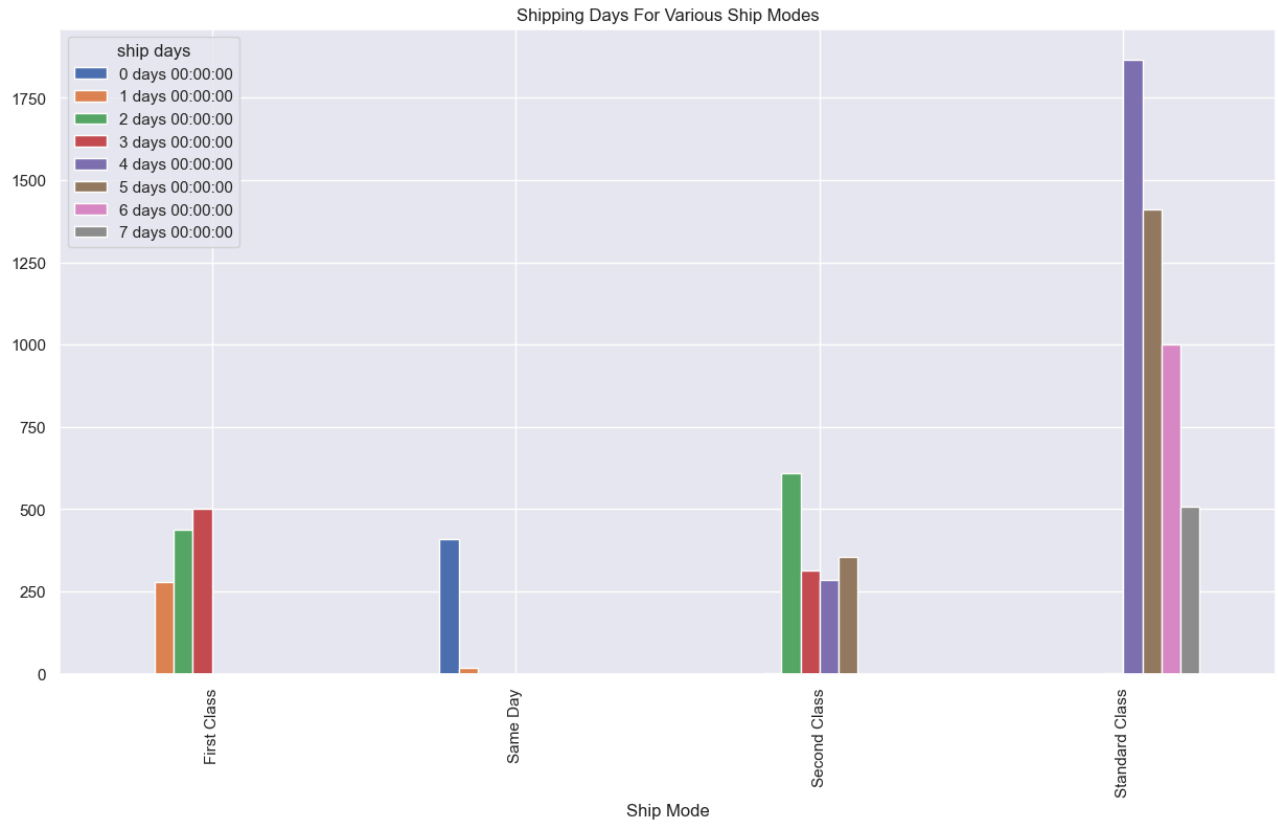
- Corporate Segment: This segment has the second highest number of transactions with a count of 2376 transactions. Achieved sales revenue of around 564,515.34, with 9,129 products sold. The profit earned was 69,434.18, leading to a profit margin of 12.3%.
- Home Office Segment: This segment has the least number of transactions with a count of 1432 transactions.Contributed sales revenue of roughly 367,599.24, with 5,364 products sold. The profit earned was 51,956.55, resulting in a profit margin of 14.13%.

5 Ship Mode Analysis

In [50]:
```
# Creating a countplot for the 'Ship Mode' Categories
fig = px.histogram(df, x='Ship Mode', color='Ship Mode',
                   barmode='group', title = 'Transaction Volume Per Ship Mode')
offline.iplot(fig)
```



In [51]:
```
df['ship days'] = df['Ship Date'] - df['Order Date']
pd.crosstab(index = df['Ship Mode'], columns = df['ship days']).plot(kind = 'bar',
plt.title('Shipping Days For Various Ship Modes')
plt.show()
```



The chart above represents the amount of days it takes to ship a product based on the 'Ship Mode'.

- First Class: Shipping days ranges between 1-3 days.
- Same Day: Shipping days ranges between 0-1 day. This is the fastest ship mode.
- Second Class: Shipping days ranges between 2-5 days.
- Standard Class: Shipping days ranges between 4-7 days.

In [52]:
```
df_sm = df.groupby('Ship Mode').agg({'Sales': 'sum', 'Quantity': 'sum',
                                     'Profit': 'sum'}).reset_index()
df_sm['profit_margin'] = round((df_sm['Profit'] / df_sm['Sales']) * 100, 2)
df_sm
```

Out[52]:

	Ship Mode	Sales	Quantity	Profit	profit_margin
0	First Class	2.671614e+05	4515	33120.0668	12.40
1	Same Day	1.040688e+05	1518	10249.1686	9.85
2	Second Class	3.790634e+05	6081	47263.1369	12.47
3	Standard Class	1.088294e+06	18181	134441.4903	12.35

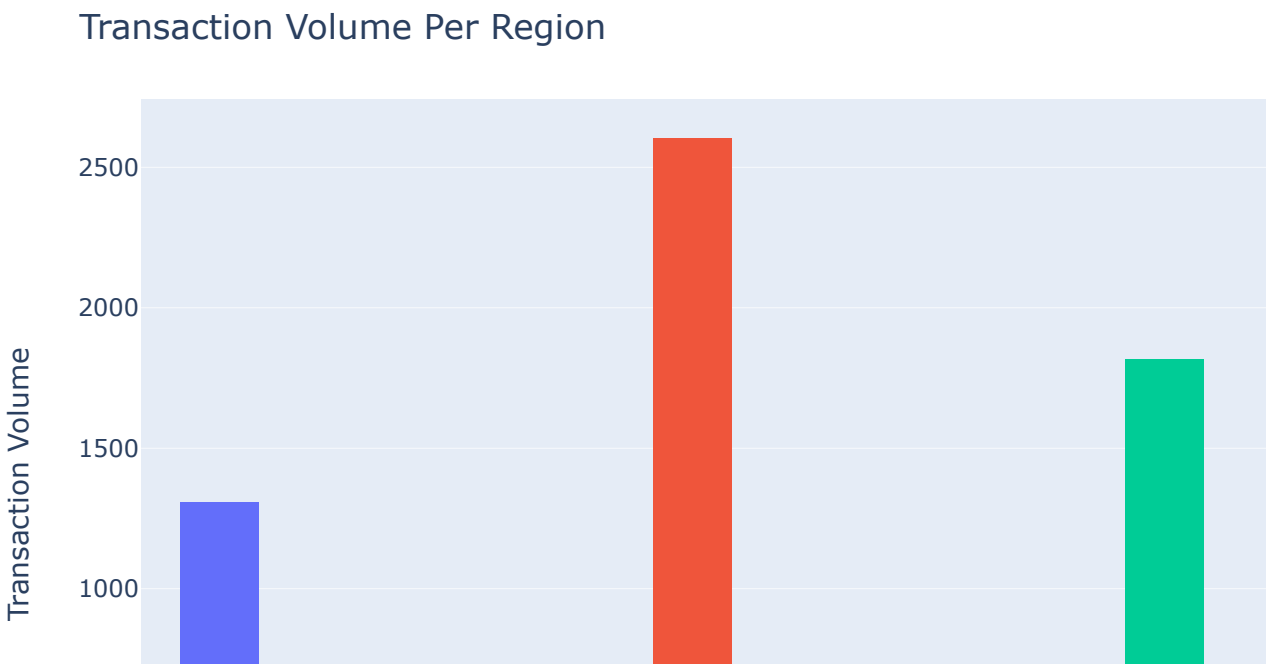
Summary

- First Class: Generated sales revenue of approximately 267,161.45, with 4,515 products shipped. The profit earned was 33,120.07, resulting in a profit margin of 12.4%.
- Same Day: Least popular shipping mode category. Achieved sales revenue of around 104,068.77, with 1,518 products shipped. The profit earned was $10,249.17, leading to a profit margin of 9.85%.

- Second Class: Contributed sales revenue of roughly 379,063.37, with 6,081 products shipped. The profit earned was 47,263.14, resulting in a profit margin of 12.47%.
- Standard Class: Most popular shipping mode. Accounted for the highest sales revenue of approximately 1,088,294.09, with 18,181 products shipped. The profit earned was 134,441.49, leading to a profit margin of 12.35%.

6 Region Analysis

In [53]:
```
# Getting a countplot for the 'REgion' category
fig = px.histogram(df, x='Region', color='Region',
                   barmode='group')
fig.update_layout(
    title= 'Transaction Volume Per Region',
    yaxis_title="Transaction Volume"
)
offline.iplot(fig)
```



- With a transaction volume of 2605, the West region has the highest recorded transactions. This high transaction volume suggests a significant level of commercial activity or customer engagement in the region.
- Following closely behind, the East region has a transaction volume of 2270. This indicates robust market activity and suggests that the East region is also a key contributor to overall sales and business operations.
- The Central region records a transaction volume of 1816, indicating a substantial but slightly lower level of commercial activity compared to the West and East regions. However, it still represents a significant portion of the overall transaction volume.

- With a transaction volume of 1309, the South region has the lowest recorded transactions among

In [54]:
```
df_reg = df.groupby('Region').agg({'Sales': 'sum', 'Quantity': 'sum',
                                   'Profit': 'sum'}).reset_index()
df_reg['profit_margin'] = round((df_reg['Profit'] / df_reg['Sales']) * 100, 2)

df_reg
```

Out[54]:

	Region	Sales	Quantity	Profit	profit_margin
0	Central	394604.9834	6895	34094.4879	8.64
1	East	551176.3040	8437	72473.6505	13.15
2	South	317934.8110	5017	38766.3684	12.19
3	West	574871.5740	9946	79739.3558	13.87

Summary:

- The West region leads in total sales, followed by the East, Central, and South regions, indicating significant economic activity in the Western region.
- Despite not leading in sales, the West region has the highest profit and profit margin, indicating efficient cost management or higher-margin sales. The Central region, despite having the lowest sales, shows a competitive profit margin.
- The East and South regions show relatively similar performance in terms of sales, profit, and profit margin, suggesting consistency in business operations across these regions.

7 State Analysis

In [55]:
```
df_state_count = df.State.value_counts().reset_index()
df_state_count.columns = ['State', 'Transaction Volume']
df_state_count
```

Out[55]:

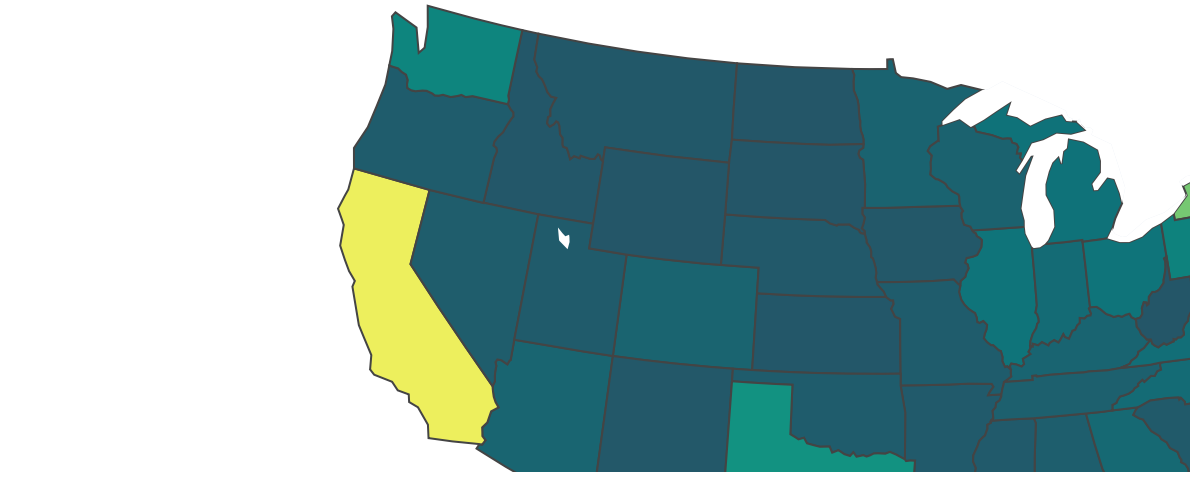
	State	Transaction Volume
0	California	1608
1	New York	912
2	Texas	734
3	Pennsylvania	480
4	Washington	397
5	Illinois	384
6	Ohio	361
7	Florida	296
8	Michigan	213
9	North Carolina	200
10	Arizona	191
11	Virginia	172
12	Colorado	165
13	Georgia	151
14	Tennessee	132
15	Kentucky	122
16	Indiana	122
17	Oregon	107
18	Massachusetts	100
19	New Jersey	95
20	Maryland	91
21	Wisconsin	88
22	Minnesota	80
23	Delaware	74
24	Connecticut	66
25	Alabama	54
26	Arkansas	54
27	Oklahoma	52
28	Mississippi	49
29	Utah	49
30	Missouri	49
31	Rhode Island	47
32	South Carolina	42
33	Louisiana	37
34	Nebraska	34
35	Nevada	31
36	New Mexico	29
37	New Hampshire	24

	State	Transaction Volume
38	Kansas	23
39	Iowa	21
40	Idaho	15
41	Montana	12
42	District of Columbia	10
43	South Dakota	9
44	North Dakota	7
45	Maine	5
46	West Virginia	3
47	Vermont	2
48	Wyoming	1

- With the highest transaction count of 1608, California stands out as the state with the most transactions.
- Following closely behind, New York has 912 transactions.
- Texas ranks third with 734 transactions.
- Vermont and Wyoming are the least two states in terms of transaction volume with a volume of 2 and 1 respectively.

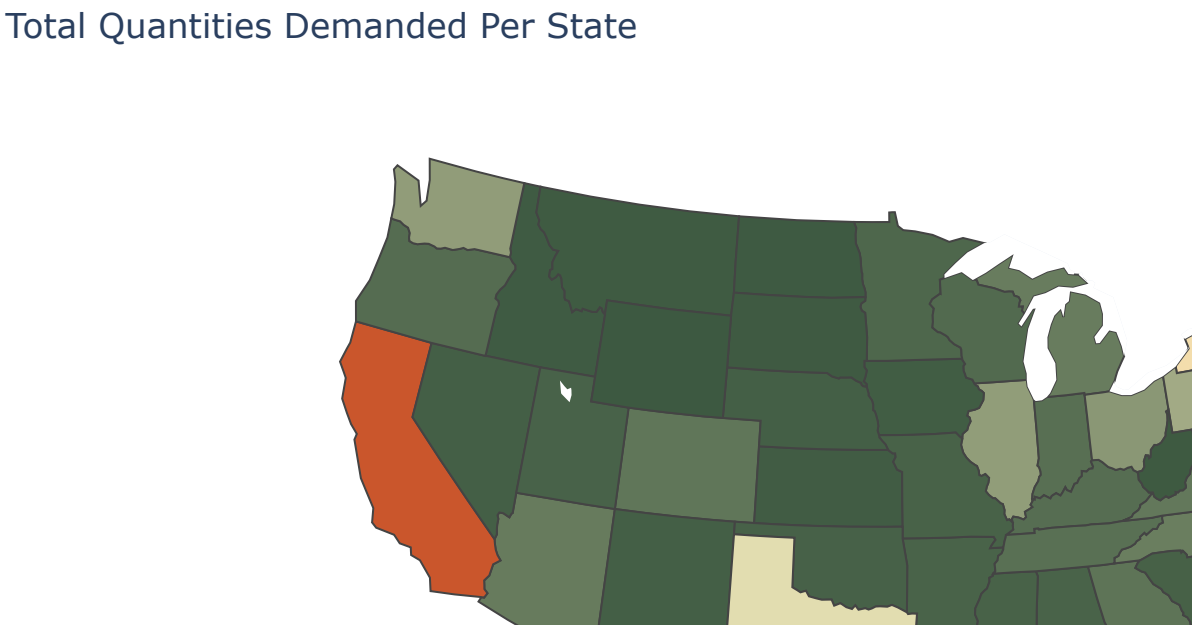
In [56]:
```
abbreviations = {
    "Alabama": "AL", "Alaska": "AK", "Arizona": "AZ", "Arkansas": "AR",
    "California": "CA", "Colorado": "CO", "Connecticut": "CT", "Delaware": "DE",
    "Florida": "FL", "Georgia": "GA", "Hawaii": "HI", "Idaho": "ID", "Illinois": "I
    "Indiana": "IN", "Iowa": "IA", "Kansas": "KS", "Kentucky": "KY", "Louisiana": '
    "Maine": "ME", "Maryland": "MD", "Massachusetts": "MA", "Michigan": "MI", "Minn
    "Mississippi": "MS", "Missouri": "MO", "Montana": "MT", "Nebraska": "NE", "Neva
    "New Hampshire": "NH", "New Jersey": "NJ", "New Mexico": "NM", "New York": "NY'
    "North Carolina": "NC", "North Dakota": "ND", "Ohio": "OH", "Oklahoma": "OK",
    "Oregon": "OR", "Pennsylvania": "PA", "Rhode Island": "RI", "South Carolina": '
    "South Dakota": "SD", "Tennessee": "TN", "Texas": "TX", "Utah": "UT", "Vermont'
    "Virginia": "VA", "Washington": "WA", "West Virginia": "WV", "Wisconsin": "WI",
}
df['Abbreviation'] = df['State'].map(abbreviations)

# Group by state and calculate the sum of sales
sum_of_sales = df.groupby('State').agg({'Sales': 'sum', 'Quantity': 'sum',
                                        'Profit': 'sum'}).reset_index()

# Add Abbreviation to sum_of_sales
sum_of_sales['Abbreviation'] = sum_of_sales['State'].map(abbreviations)
```

In [57]:
```
fig = px.choropleth(sum_of_sales, locations='Abbreviation', locationmode="USA-state
                    color='Sales', color_continuous_scale='Aggrnyl', scope="usa",
                    title = 'Total Sales Per State')
offline.iplot(fig)
```

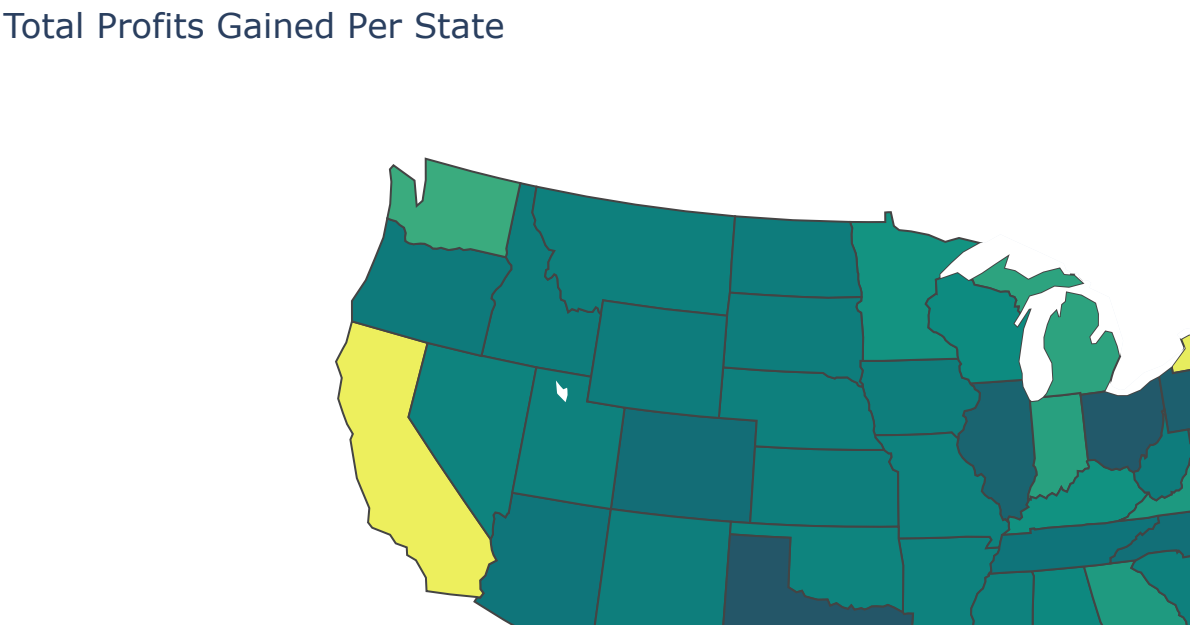


In [58]:

```
fig = px.choropleth(sum_of_sales, locations='Abbreviation', locationmode="USA-state
                    color='Quantity', color_continuous_scale='Fall', scope="usa",
                   title = 'Total Quantities Demanded Per State')
offline.iplot(fig)
```

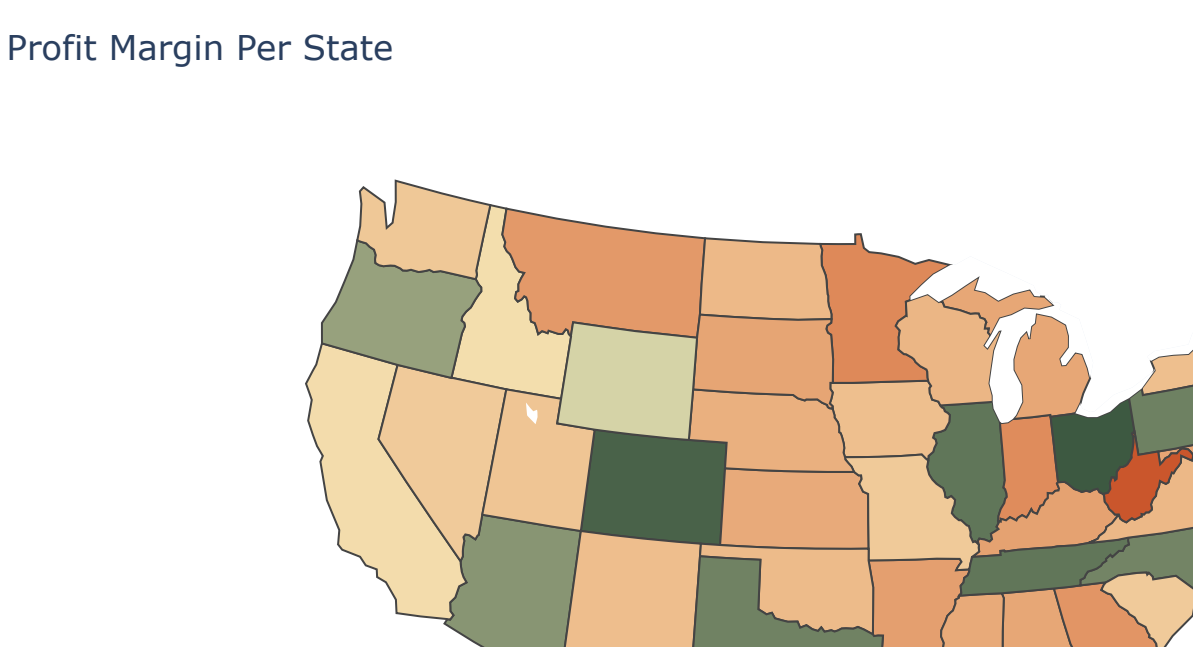


In [59]:
```
fig = px.choropleth(sum_of_sales, locations='Abbreviation', locationmode="USA-state
                    color='Profit', color_continuous_scale='Aggrnyl', scope="usa",
                   title = 'Total Profits Gained Per State')
offline.iplot(fig)
```



In [60]:
```
sum_of_sales['profit_margin'] = round(
    (sum_of_sales['Profit'] / sum_of_sales['Sales']) * 100, 2)
```

In [61]:
```
fig = px.choropleth(sum_of_sales, locations='Abbreviation', locationmode="USA-state
                    color='profit_margin', color_continuous_scale='Fall', scope="us
                   title = 'Profit Margin Per State')
offline.iplot(fig)
```



Summary:

- Total sales per state range from 536.48 in West Virginia to 359,678.19 in California. The mean sales amount per state is approximately 41,838.71.
- Total Quantity sold per state ranges from 4 items in Wyoming to 6,157 items in California. The mean quantity sold per state is approximately 536.38 items.
- Total Profit ranges from -17,096.23 in Texas to 59,671.63 in California. The mean profit is approximately 3,207.49.
- Profit margin ranges from -24.10% in Ohio to 49.00% in West Virginia. The mean profit margin is approximately 19.77%.
- West Virginia has the highest profit margin at 49.00% and a transaction volume of just 3 units, indicating high profitability relative to sales. Ohio, on the other hand, has the lowest profit margin at -24.10%, indicating losses exceeding the sales amount.
- While some states like California, Indiana, and Minnesota show both high sales and high profitability, others like Texas and Ohio have high sales but negative profitability, suggesting potential issues with cost management or pricing strategies.
- States like California, New York, and Texas consistently appear among the top performers in terms of sales, quantity, and profit, indicating their economic significance to the superstore.

8 Discount Analysis

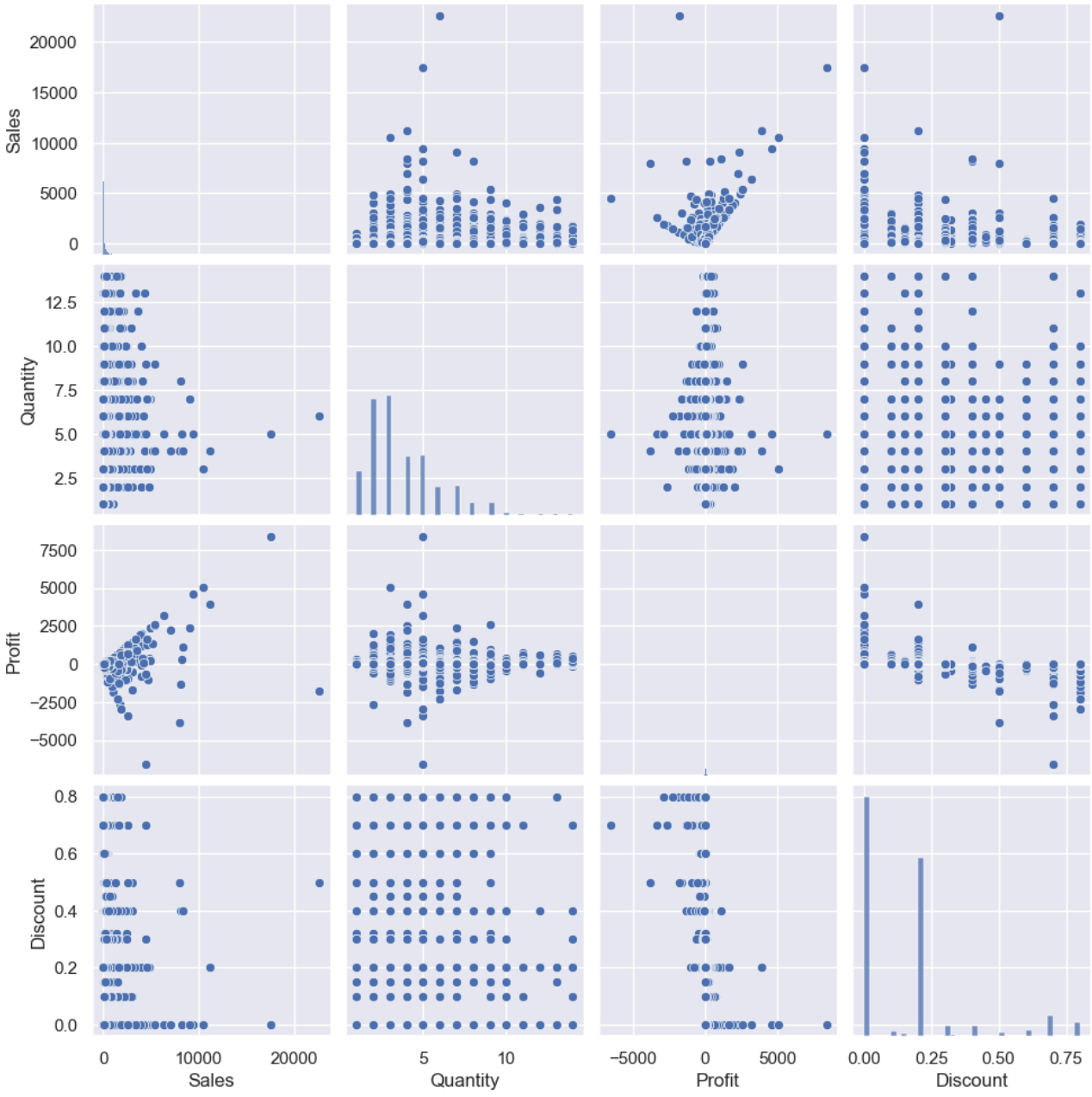
In [62]:
```
# Getting the count of transactions for each discount rate↔
```

Out[62]:
```
0.00    3897
0.20    2916
0.70     341
0.80     219
0.30     169
0.40     166
0.60      99
0.10      73
0.50      53
0.15      41
0.32      18
0.45       8
Name: Discount, dtype: int64
```

- The most frequent discount rate is 0.00, appearing 3897 times in the dataset. This indicates that a large portion of transactions does not involve any discount.
- The second most common discount rate is 0.20, with 2916 occurrences. This suggests that a significant number of transactions involve a 20% discount.
- Discount rates of 0.70, 0.80, 0.30, 0.40, 0.60, 0.10, 0.50, 0.15, and others have lower frequencies, indicating less common usage compared to the previously mentioned discount rates.

In [63]:
```
# Plotting a multiple scatter matrix for the 'Sales', 'Quantity', 'Profit', 'Discou
```

```
<Figure size 1500x800 with 0 Axes>
```



In [64]:
```
# Creating a new column called 'is_discounted' to specify if a transaction is a dis
```

In [65]:
```
df_dis = df.groupby('is_discounted').agg({'Sales': 'sum', 'Quantity': 'sum',
                                          'Profit': 'sum'}).reset_index()
df_dis
```

Out[65]:

	is_discounted	Sales	Quantity	Profit
0	Discounted	958181.0824	15474	-31930.6751
1	Non-Discounted	880406.5900	14821	257004.5377

- The total sales amount is higher for discounted transactions (958181.08) compared to non-discounted ones (880406.59). This suggests that discounted items may attract more customers or encourage larger purchases.

- Although the difference in total quantity sold between discounted (15474 units) and non-discounted (14821 units) transactions is relatively small, discounted transactions still have a slightly higher quantity sold. This aligns with the higher total sales amount observed in discounted transactions.
- Interestingly, while discounted transactions have higher total sales, they seem to result in a negative total profit, indicating a potential loss. On the other hand, non-discounted transactions

In [66]:
```
fig = px.histogram(df, x='Segment', color='is_discounted',
                   barmode='group',
                   title = 'Transaction Volume Per Segment For Discounted and Non D
offline.iplot(fig)
```

In [67]:
```
fig = px.histogram(df, x='Category', color='is_discounted',
                   barmode='group',
                   title = 'Transaction Volume Per Product Category For Discounted
                  )
offline.iplot(fig)
```

In [68]:
```
fig = px.histogram(df, x='Sub-Category', color='is_discounted',
                   barmode='group',
                   title = 'Transaction Volume Per Sub-Category For Discounted and
offline.iplot(fig)
```

In [69]:
```
# Getting the mean sales, quantities for discounted and non discounted transactions
```

Out[69]:

	is_discounted	Sales	Quantity
0	Discounted	233.531826	3.771387
1	Non-Discounted	225.919063	3.803182

- On average, the sales amount per transaction is slightly higher for discounted transactions compared to non-discounted transactions. However, the difference is relatively small.
- The average quantity of items sold per transaction is slightly lower for discounted transactions compared to non-discounted transactions. Again, the difference is minimal.

A hypothesis test will be carried out to determine if these differences are significant.

8.1 Hypothesis Test (1)

In this section, a t-test will be conducted to compare sales between discounted transactions and non discounted transactions.

- **Null Hypothesis (H0)**: There is no significant difference in sales and quantities demanded between discounted and non-discounted transactions.

- **Alternative Hypothesis (H1)**: There is a significant difference in sales and quantities demanded between discounted and non-discounted.

Alpha: 0.05

Results:

- If the p-value is less than the chosen significance level (alpha), reject the null hypothesis.
- If the p-value is greater than the significance level, fail to reject the null hypothesis.

In [70]:
```
categories = df['is_discounted'].unique()

# Perform pairwise t-tests for each numerical column across categories
for col in df[['Sales', 'Quantity']]:
    print(f"Pairwise t-tests for {col}:")
    for i in range(len(categories)):
        for j in range(i + 1, len(categories)):
            cat1 = categories[i]
            cat2 = categories[j]
            group1 = df[df['is_discounted'] == cat1][col]
            group2 = df[df['is_discounted'] == cat2][col]
            t_statistic, p_value = ttest_ind(group1, group2)
            print(f"T-test between {col} for {cat1} and {cat2}:")
            print("T-statistic:", t_statistic)
            print("P-value:", p_value)
            if p_value < 0.05:
                print("Reject the null hypothesis. There is a significant differenc
            else:
                print(
                    "Fail to reject the null hypothesis. There is no significant di
            print()
```

```
Pairwise t-tests for Sales:
T-test between Sales for Non-Discounted and Discounted:
T-statistic: -0.5479997204550883
P-value: 0.583707388211592
Fail to reject the null hypothesis. There is no significant difference.

Pairwise t-tests for Quantity:
T-test between Quantity for Non-Discounted and Discounted:
T-statistic: 0.6427743984583121
P-value: 0.520388923228463
Fail to reject the null hypothesis. There is no significant difference.
```

Summary:

1. **Sales T-Test**:
    - **T-Statistic**: -0.548
    - **P-Value**: 0.584
    - **Interpretation**: With a p-value of 0.584, which is greater than the common significance level of 0.05, we fail to reject the null hypothesis. This suggests that there is no significant difference in sales between non-discounted and discounted transactions.
2. **Quantity T-Test**:
    - **T-Statistic**: 0.643
    - **P-Value**: 0.520
    - **Interpretation**: Similarly, with a p-value of 0.520, which is greater than 0.05, we fail to reject the null hypothesis. This indicates that there is no significant difference in the quantity of items sold between non-discounted and discounted transactions.

In both cases, since the p-values are greater than the significance level, we do not have enough evidence to conclude that there is a significant difference between non-discounted and discounted

9 Holiday Transactions Analysis

In [71]:
```
# Creating a column to hold whether a transaction was carried out on a holiday.
us_holidays = holidays.US()
holiday = df[df["Order Date"].apply(lambda d: d in us_holidays)]
df['is_holiday'] = np.where(df.index.isin(holiday.index), 'true', 'false')
```

In [72]:
```
# Checking out the transactions that were carried out on holidays
df[df['is_holiday'] == 'true']
```

Out[72]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	Ci
16	17	CA-2014-105893	2014-11-11	2014-11-18	Standard Class	PK-19075	Pete Kriz	Consumer	United States	Mac
106	107	CA-2017-119004	2017-11-23	2017-11-28	Standard Class	JM-15250	Janet Martin	Consumer	United States	Cha
107	108	CA-2017-119004	2017-11-23	2017-11-28	Standard Class	JM-15250	Janet Martin	Consumer	United States	Cha
108	109	CA-2017-119004	2017-11-23	2017-11-28	Standard Class	JM-15250	Janet Martin	Consumer	United States	Cha
110	111	CA-2017-146780	2017-12-25	2017-12-30	Standard Class	CV-12805	Cynthia Voltz	Corporate	United States	York
...	...	...	...	...	...	...	...	...	...	
7945	7946	CA-2017-134194	2017-12-25	2018-01-01	Standard Class	GA-14725	Guy Armstrong	Consumer	United States	D
7946	7947	CA-2017-134194	2017-12-25	2018-01-01	Standard Class	GA-14725	Guy Armstrong	Consumer	United States	D
7967	7968	CA-2016-157707	2016-10-10	2016-10-12	First Class	CC-12610	Corey Catlett	Corporate	United States	De
7968	7969	CA-2016-157707	2016-10-10	2016-10-12	First Class	CC-12610	Corey Catlett	Corporate	United States	De
7969	7970	CA-2016-157707	2016-10-10	2016-10-12	First Class	CC-12610	Corey Catlett	Corporate	United States	De

381 rows × 28 columns

In [73]:
```
# Getting the mean sales , quantity, profit and profit margin for transactions cond
```

Out[73]:

	is_holiday	Sales	Quantity	Profit	profit margin
0	false	230.457793	3.794199	28.135596	12.919299
1	true	217.138436	3.640420	28.106977	10.346457

- The average sales amount per transaction is slightly higher on non-holidays compared to holidays. However, the difference is relatively small.
- Similarly, the average quantity of items sold per transaction is slightly higher on non-holidays than on holidays.
- The average profit per transaction and profit margin are comparable between holiday and non-holiday periods. While there is a slight difference, it may not be statistically significant.

Next, a hypothesis test will be carried out to determine if these differences in sales, quantity etc are significant.

9.1 Hypothesis Test (2)

In this section, a t-test will be conducted to compare sales, quantities, profit and profit margin between holiday transactions and non holiday transactions.

- **Null Hypothesis (H0):** There is no significant difference in sales, quantities, profit and profit margin between holidays and non-holidays.
- **Alternative Hypothesis (H1):** There is a significant difference in sales, quantities, profit and profit margin between holidays and non-holidays. Alpha: 0.05
- **Results**:
  - If the p-value is less than the chosen significance level (alpha), reject the null hypothesis.
  - If the p-value is greater than the significance level, fail to reject the null hypothesis.

In [74]:
```
# Get unique categories
categories = df['is_holiday'].unique()

# Perform pairwise t-tests for each numerical column across categories
for col in df.select_dtypes(include='float'):
    print(f"Pairwise t-tests for {col}:")
    for i in range(len(categories)):
        for j in range(i + 1, len(categories)):
            cat1 = categories[i]
            cat2 = categories[j]
            group1 = df[df['is_holiday'] == cat1][col]
            group2 = df[df['is_holiday'] == cat2][col]
            t_statistic, p_value = ttest_ind(group1, group2)
            print(f"T-test between {col} for {cat1} and {cat2}:")
            print("T-statistic:", t_statistic)
            print("P-value:", p_value)
            if p_value < 0.05:
                print("Reject the null hypothesis. There is a significant differenc
            else:
                print(
                    "Fail to reject the null hypothesis. There is no significant di
            print()
```

```
Pairwise t-tests for Sales:
T-test between Sales for false and true:
T-statistic: 0.408519903216694
P-value: 0.6829029582648267
Fail to reject the null hypothesis. There is no significant difference.

Pairwise t-tests for Discount:
T-test between Discount for false and true:
T-statistic: -1.5596988974894843
P-value: 0.11887061825208911
Fail to reject the null hypothesis. There is no significant difference.

Pairwise t-tests for Profit:
T-test between Profit for false and true:
T-statistic: 0.0023979368385411313
P-value: 0.9980867848564229
Fail to reject the null hypothesis. There is no significant difference.

Pairwise t-tests for profit margin:
T-test between profit margin for false and true:
T-statistic: 1.074281886829218
P-value: 0.28272876924645063
Fail to reject the null hypothesis. There is no significant difference.
```

1. **Sales T-Test**:
   - **T-Statistic**: 0.409
   - **P-Value**: 0.683
   - **Interpretation**: With a p-value of 0.683, we fail to reject the null hypothesis. This suggests that there is no significant difference in sales between non-holiday and holiday periods.
2. **Discount T-Test**:
   - **T-Statistic**: -1.560
   - **P-Value**: 0.119
   - **Interpretation**: Similarly, with a p-value of 0.119, we fail to reject the null hypothesis. This indicates that there is no significant difference in the presence of discounts between non-holiday and holiday periods.
3. **Profit T-Test**:

- **T-Statistic**: 0.00240
- **P-Value**: 0.998
- **Interpretation**: With a p-value of 0.998, we fail to reject the null hypothesis. This suggests that there is no significant difference in profit between non-holiday and holiday periods.

4. **Profit Margin T-Test**:

- **T-Statistic**: 1.074
- **P-Value**: 0.283
- **Interpretation**: Similarly, with a p-value of 0.283, we fail to reject the null hypothesis. This indicates that there is no significant difference in profit margin between non-holiday and holiday periods.

Overall, these results suggest that there are no significant differences between non-holiday and holiday periods in terms of sales, presence of discounts, profit, or profit margin.

In [75]:
```
# Grouping the dataset by the transaction volume carried out on each day and sortir
temp_df = df.groupby('Order Date')['Quantity'].count().reset_index().sort_values('Q
temp_df.columns = ['Order Date', 'Transaction Volume']
temp_df
```

Out[75]:

	Order Date	Transaction Volume
763	2016-09-05	37
1160	2017-12-09	31
1070	2017-09-02	30
820	2016-11-10	29
1153	2017-12-02	28
1159	2017-12-08	27
850	2016-12-11	27
1152	2017-12-01	27
1133	2017-11-12	27
1076	2017-09-09	26

In [76]:
```
# Grouping the dataset by the transaction volume carried out on each day and sortir
temp_df = df.groupby('Order Date')['Quantity'].sum().reset_index().sort_values('Qua
temp_df.columns = ['Order Date', 'Quantities Demanded']
temp_df
```

Out[76]:

	Order Date	Quantities Demanded
763	2016-09-05	141
820	2016-11-10	125
1153	2017-12-02	121
1140	2017-11-19	114
850	2016-12-11	108
1098	2017-10-02	108
539	2015-12-06	108
1160	2017-12-09	108
185	2014-09-08	106
1152	2017-12-01	106

The day with the highest number of transaction volume of 37 is 2016-09-05. September 5th in the united States is Labor day which is celebrated in honour of workers. This day is also the day with the highest number of quantities demanded.

Although there is no significant difference in sales metrics between transactions carried out on US federal holidays and those not carried out federal holidays, It doesnot mean that sales and quantities are do not increase on some certain days or certain holidays as can be seen previously where a huge negative profit or loss was discovered on the 'Black Friday' in 2016 i.e November 25th 2016.

10 Recommendations

1. **Customer Engagement**: Continue to prioritize customer satisfaction and engagement to sustain demand growth. This could involve gathering feedback, offering personalized experiences, or implementing loyalty programs.

2. **Market Activity Distribution**: The distribution of transaction volumes across regions provides insights into the geographical spread of business activities. Higher transaction volumes in the West and East regions suggest stronger market presence and potentially more significant revenue generation in those areas.

3. **Opportunities for Growth**: Regions with lower transaction volumes, such as the South, may present opportunities for targeted efforts to increase market share or expand business operations.

4. **Performance Monitoring**: Continuously monitor performance by region and track key metrics such as sales, profit, and profit margin. Identifying trends will aid strategic decision-making and resource allocation.

5. **Profit Margin Analysis**: Investigate the factors driving variations in profit margins across states. Identifying states with low or negative profit margins will aid in cost optimization and pricing adjustments.

6. **Customer Segmentation Analysis**: Conducting deeper analysis to understand the unique needs, preferences, and behaviors of each customer segment. This insights will then be used to develop targeted marketing campaigns and product strategies to meet or satisfy specific customer segments.

7. Explore customer behavior patterns associated with discounted and non-discounted transactions. Too much discounts may customers behaviour where customers wait for a discount before purchasing a product and also make potential customers question the quality of your products. Understanding customer preferences and responses to discounts can help refine marketing and pricing strategies.

8. Further analysis should be carried out to evaluate the performance of different discount rates in terms of their impact on sales volume, revenue, and profitability. This analysis can help identify which discount rates are most effective in achieving desired outcomes.

9. **Segmented Discounting**: Consider implementing segmented discounting strategies based on customer segments, product categories, or purchasing behavior. Tailoring discounts to specific customer segments or product lines can enhance their effectiveness and optimize resource allocation.

10. **Inventory Management**: Adjust inventory levels to anticipate increased demand during peak months (March, September, November) and reduce stock during slower periods (October, February, August).

    - Ensure sufficient stock availability for popular products or those experiencing higher demand during peak months to prevent stockouts and capitalize on sales opportunities.
11. **Supply Chain Optimization**: Optimize supply chain processes to ensure timely delivery of goods and materials, especially during peak months, to meet increased demand without disruptions.

    - Establish strong relationships with suppliers to negotiate favorable terms and secure adequate inventory levels ahead of peak seasons.
12. **Forecasting and Planning**: Utilize historical sales data and demand forecasts to develop accurate sales projections and resource allocation plans for each month.

    - Implement agile planning processes to adapt quickly to changing market conditions and