[Workshop] Superstore (Exploratory Data Analysis - EDA)

1. Import Library and Setting

In [2]:
```python
# Importing library
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import warnings
import os

# Check library version
print("---Library version---", end = '\n')
print('pandas version: ', pd.__version__)
print('numpy version: ', np.__version__)
print('seaborn version: ', sns.__version__)
print('matplotlib version: ', mpl.__version__, end = '\n\n')

# Setting library
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 122)

mpl.font_manager.fontManager.addfont("fonts\Sarabun-Regular.ttf")
mpl.rc('font', family='Sarabun')
plt.rcParams ['font.family'] = ('Sarabun')

# ignore warnings
warnings.filterwarnings('ignore')


print("---Working Directory---", end = '\n')
print('Current Working Directory:', os.getcwd())
print('File of Directory:', os.listdir(os.getcwd()))
print('List of Directory (archive):', os.listdir(os.getcwd() + r'\archive'))
```

```
---Library version---
pandas version:  2.2.2
numpy version:  1.26.3
seaborn version:  0.13.2
matplotlib version:  3.9.2

---Working Directory---
Current Working Directory: C:\Users\KATANA\Desktop\Jupyter-Lab\workshop\Superstore_S
ales
File of Directory: ['.ipynb_checkpoints', 'archive', 'fonts', 'info.txt', 'superstor
e-analytic.ipynb']
List of Directory (archive): ['Superstore.csv', 'Superstore.xlsx']
```

2. Importing Data

In [13]:
```
data = pd.read_csv(r'archive/Superstore.csv', index_col='Row ID', encoding='ISO-885

with pd.option_context('display.max_rows', 100):
    display(data)
```

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
1	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Hender
2	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Hender
3	CA-2013-138688	13-06-2013	17-06-2013	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Ange
4	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderc
5	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	F Lauderc
...	...	...	...	...	...	...	...	...	
9990	CA-2011-110422	22-01-2011	24-01-2011	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Mi
9991	CA-2014-121258	27-02-2014	04-03-2014	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9992	CA-2014-121258	27-02-2014	04-03-2014	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9993	CA-2014-121258	27-02-2014	04-03-2014	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa M
9994	CA-2014-	05-05-	10-05-	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westmin

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	C
Row ID									
	119914	2014	2014						

9994 rows × 20 columns

In [14]:
```python
data.head()
```

Out[14]:

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
Row ID									
1	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
2	CA-2013-152156	09-11-2013	12-11-2013	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson
3	CA-2013-138688	13-06-2013	17-06-2013	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles
4	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale
5	US-2012-108966	11-10-2012	18-10-2012	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale

In [16]:
```python
print(f'Record: {data.shape[0]}, Columns: {data.shape[1]}')
```

```
Record: 9994, Columns: 20
```

3. Overview of Data

In [17]:
```
data.columns
```

Out[17]:
```
Index(['Order ID', 'Order Date', 'Ship Date', 'Ship Mode', 'Customer ID',
       'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name',
       'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [18]:
```
data.info()
```

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

4. Data preprocessing

Change data type in columns

In [25]:
```
data['Order Date'] = pd.to_datetime(data['Order Date'], format='%d-%m-%Y', dayfirst
data['Ship Date'] = pd.to_datetime(data['Ship Date'], format='%d-%m-%Y', dayfirst=T
data['Postal Code'] = data['Postal Code'].astype(str)
```

In [27]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9994 entries, 1 to 9994
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Order ID       9994 non-null   object        
 1   Order Date     9994 non-null   datetime64[ns]
 2   Ship Date      9994 non-null   datetime64[ns]
 3   Ship Mode      9994 non-null   object        
 4   Customer ID    9994 non-null   object        
 5   Customer Name  9994 non-null   object        
 6   Segment        9994 non-null   object        
 7   Country        9994 non-null   object        
 8   City           9994 non-null   object        
 9   State          9994 non-null   object        
 10  Postal Code    9994 non-null   object        
 11  Region         9994 non-null   object        
 12  Product ID     9994 non-null   object        
 13  Category       9994 non-null   object        
 14  Sub-Category   9994 non-null   object        
 15  Product Name   9994 non-null   object        
 16  Sales          9994 non-null   float64       
 17  Quantity       9994 non-null   int64         
 18  Discount       9994 non-null   float64       
 19  Profit         9994 non-null   float64       
dtypes: datetime64[ns](2), float64(3), int64(1), object(14)
memory usage: 1.6+ MB
```

data.describe()

In [26]:
```
data.describe()
```

Out[26]:

	Order Date	Ship Date	Sales	Quantity	Discount	
count	9994	9994	9994.000000	9994.000000	9994.000000	999
mean	2013-04-30 19:20:02.401441024	2013-05-04 18:20:49.229537792	229.858001	3.789574	0.156203	2
min	2011-01-04 00:00:00	2011-01-08 00:00:00	0.444000	1.000000	0.000000	-659
25%	2012-05-23 00:00:00	2012-05-27 00:00:00	17.280000	2.000000	0.000000	
50%	2013-06-27 00:00:00	2013-06-30 00:00:00	54.490000	3.000000	0.200000	
75%	2014-05-15 00:00:00	2014-05-19 00:00:00	209.940000	5.000000	0.200000	2
max	2014-12-31 00:00:00	2015-01-06 00:00:00	22638.480000	14.000000	0.800000	839
std	NaN	NaN	623.245101	2.225110	0.206452	23

NaN values in each columns

In [28]:

```
for i in data.columns:
    print(f'{i} : {data[i].isna().sum()}')
```

```
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
```

In [29]:

```
print('Object columns with contain NaN values', end='\n\n')
for column in data.columns:
    if data[column].dtype == 'O' and data[column].isna().any():
        print(f"NaN in : '{column}'")
```

```
Object columns with contain NaN values
```

Find Unique

In [47]:
```
selected_cols = ['Segment', 'Country', 'Category', 'Sub-Category', 'Product Name']

for col in selected_cols:
    unique_vals = data[col].unique()
    unique_count = data[col].nunique()
    print(f"Column: {col}")
    print(f"Unique count: {unique_count}" ,end='\n\n')
    print(f"Unique values: {unique_vals}")
    print("-" * 40, end='\n\n')
```

```
Column: Segment
Unique count: 3

Unique values: ['Consumer' 'Corporate' 'Home Office']
----------------------------------------

Column: Country
Unique count: 1

Unique values: ['United States']
----------------------------------------

Column: Category
Unique count: 3

Unique values: ['Furniture' 'Office Supplies' 'Technology']
----------------------------------------

Column: Sub-Category
Unique count: 17

Unique values: ['Bookcases' 'Chairs' 'Labels' 'Tables' 'Storage' 'Furnishings' 'Art'
 'Phones' 'Binders' 'Appliances' 'Paper' 'Accessories' 'Envelopes'
 'Fasteners' 'Supplies' 'Machines' 'Copiers']
----------------------------------------

Column: Product Name
Unique count: 1841

Unique values: ['Bush Somerset Collection Bookcase'
 'Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back'
 'Self-Adhesive Address Labels for Typewriters by Universal' ...
 'Eureka Hand Vacuum, Bagless' 'LG G2'
 'Eldon Jumbo ProFile Portable File Boxes Graphite/Black']
----------------------------------------
```

count duplicate value

In [54]:
```
data['Segment'].value_counts()[data['Segment'].value_counts() > 1]
```

Out[54]:
```
Segment
Consumer       5191
Corporate      3020
Home Office    1783
Name: count, dtype: int64
```

In [55]:
```
data['Order ID'].value_counts()[data['Order ID'].value_counts() > 1]
```

Out[55]:
```
Order ID
CA-2014-100111    14
CA-2014-157987    12
CA-2013-165330    11
US-2013-108504    11
CA-2012-131338    10
                  ..
CA-2013-115476     2
CA-2013-145625     2
CA-2013-111794     2
CA-2013-142370     2
CA-2012-120341     2
Name: count, Length: 2471, dtype: int64
```

Drop non-used columns

In [59]:
```
#data=data.drop('Row ID',axis=1)

data=data[[
        'Order ID',
        'Order Date',
        'Ship Date',
        'Ship Mode',
    #'Customer ID',
    #'Customer Name',
        'Segment',
    #'Country',
        'City',
        'State',
    #'Postal Code',
        'Region',
    #'Product ID',
        'Category',
        'Sub-Category',
        'Product Name',
        'Sales',
        'Quantity',
        'Discount',
        'Profit']]
data.head(3) # final dataframe, after columns were removed
```

Out[59]:

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Segment	City	State	Region	Category	C
1	CA-2013-152156	2013-11-09	2013-11-12	Second Class	Consumer	Henderson	Kentucky	South	Furniture	B
2	CA-2013-152156	2013-11-09	2013-11-12	Second Class	Consumer	Henderson	Kentucky	South	Furniture	
3	CA-2013-138688	2013-06-13	2013-06-17	Second Class	Corporate	Los Angeles	California	West	Office Supplies	

extracts specific date values from the `Order Date` column

In [60]:

```
data['month']=data['Order Date'].dt.month
data['year']=data['Order Date'].dt.year
data['year_month']=data['Order Date'].dt.to_period('M')
data['total_discount_in_dollars']=data['Sales'] * data['Discount'] # discount's equ
data['selling_price']=data['Sales'] / data['Quantity'] # calculates selling price f
data['(net)_profit_before_discount']=data['Sales'] * data['Discount'] + data['Profi
data['order_fulfillment_time']=data['Ship Date'] - data['Order Date'] # interval be
data['net_profit_per_unit_sold']=data['Profit'] / data['Quantity'] # net profit gen
data=data.rename(columns={'Profit':'net_profit'}) # renames Profit column with net_
data['profit_margin']=data['net_profit'] / data['Sales'] * 100 # for a 25% profit m
data['discounted_sales']=data['Sales'] - (data['Discount']*data['Sales']) # extract
```

In [61]:

```
data.head(3)
```

Out[61]:

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Segment	City	State	Region	Category
1	CA-2013-152156	2013-11-09	2013-11-12	Second Class	Consumer	Henderson	Kentucky	South	Furniture
2	CA-2013-152156	2013-11-09	2013-11-12	Second Class	Consumer	Henderson	Kentucky	South	Furniture
3	CA-2013-138688	2013-06-13	2013-06-17	Second Class	Corporate	Los Angeles	California	West	Office Supplies

In [63]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9994 entries, 1 to 9994
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Order ID                      9994 non-null   object
 1   Order Date                    9994 non-null   datetime64[ns]
 2   Ship Date                     9994 non-null   datetime64[ns]
 3   Ship Mode                     9994 non-null   object
 4   Segment                       9994 non-null   object
 5   City                          9994 non-null   object
 6   State                         9994 non-null   object
 7   Region                        9994 non-null   object
 8   Category                      9994 non-null   object
 9   Sub-Category                  9994 non-null   object
 10  Product Name                  9994 non-null   object
 11  Sales                         9994 non-null   float64
 12  Quantity                      9994 non-null   int64
 13  Discount                      9994 non-null   float64
 14  net_profit                    9994 non-null   float64
 15  month                         9994 non-null   int32
 16  year                          9994 non-null   int32
 17  year_month                    9994 non-null   period[M]
 18  total_discount_in_dollars     9994 non-null   float64
 19  selling_price                 9994 non-null   float64
 20  (net)_profit_before_discount  9994 non-null   float64
 21  order_fulfillment_time        9994 non-null   timedelta64[ns]
 22  net_profit_per_unit_sold      9994 non-null   float64
 23  profit_margin                 9994 non-null   float64
 24  discounted_sales              9994 non-null   float64
dtypes: datetime64[ns](2), float64(9), int32(2), int64(1), object(9), period[M](1),
timedelta64[ns](1)
memory usage: 1.9+ MB
```

5. Analysis and Understand insight of the Data

5.1 Correlation

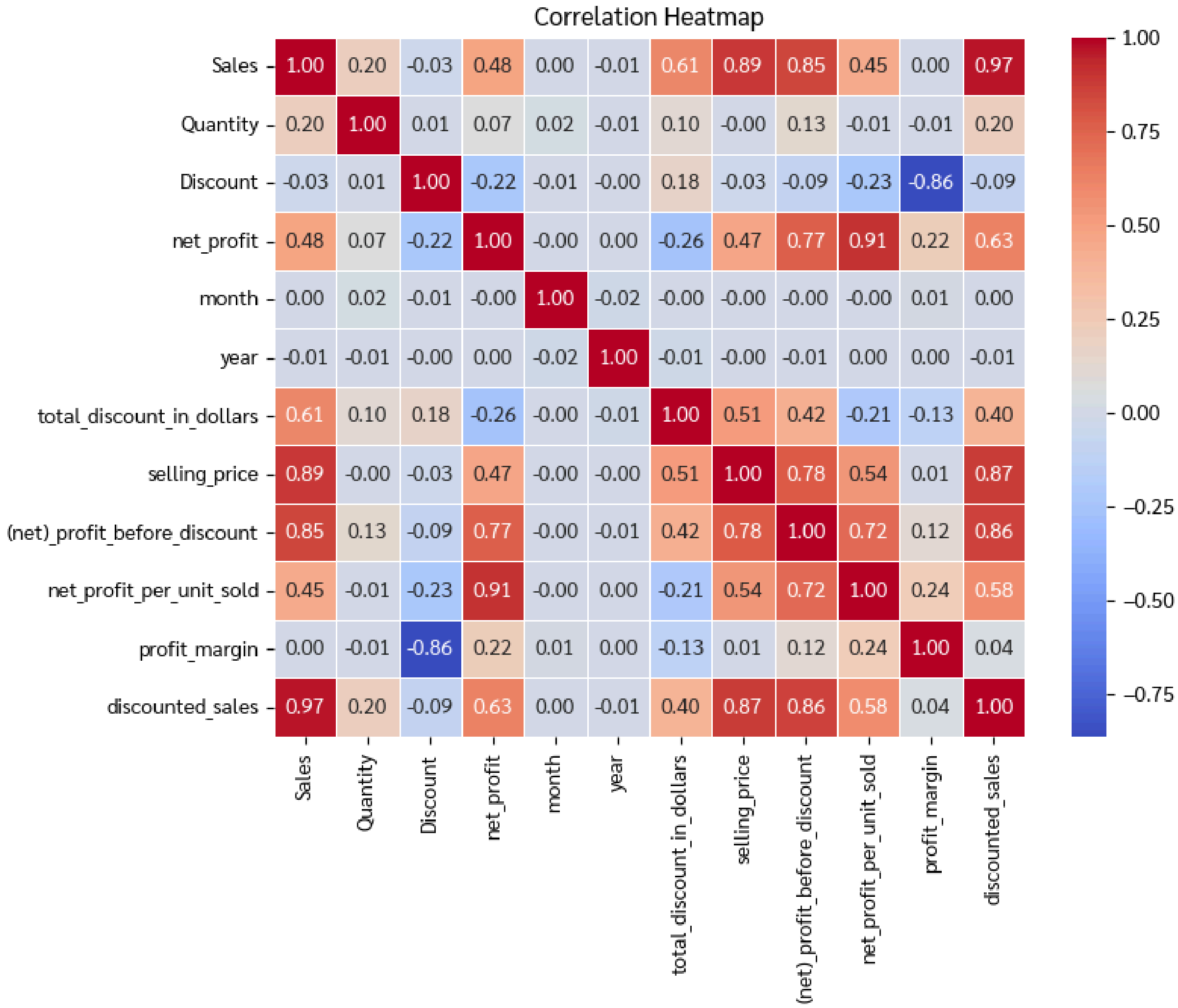
In [64]:
```python
data.corr(numeric_only=True)
```

Out[64]:

	Sales	Quantity	Discount	net_profit	month	yea
Sales	1.000000	0.200795	-0.028190	0.479064	0.000079	-0.00980
Quantity	0.200795	1.000000	0.008623	0.066253	0.023459	-0.00578
Discount	-0.028190	0.008623	1.000000	-0.219487	-0.005071	-0.00261
net_profit	0.479064	0.066253	-0.219487	1.000000	-0.000210	0.00461
month	0.000079	0.023459	-0.005071	-0.000210	1.000000	-0.01859
year	-0.009800	-0.005788	-0.002615	0.004618	-0.018596	1.00000
total_discount_in_dollars	0.610248	0.101840	0.176579	-0.259087	-0.003600	-0.01476
selling_price	0.889376	-0.003148	-0.032803	0.468312	-0.003506	-0.00307
(net)_profit_before_discount	0.853590	0.129552	-0.090270	0.770939	-0.002572	-0.00538
net_profit_per_unit_sold	0.447319	-0.007209	-0.232313	0.912199	-0.001229	0.00333
profit_margin	0.003444	-0.005280	-0.864452	0.223732	0.009846	0.00014
discounted_sales	0.970510	0.201171	-0.086325	0.632732	0.001187	-0.00683

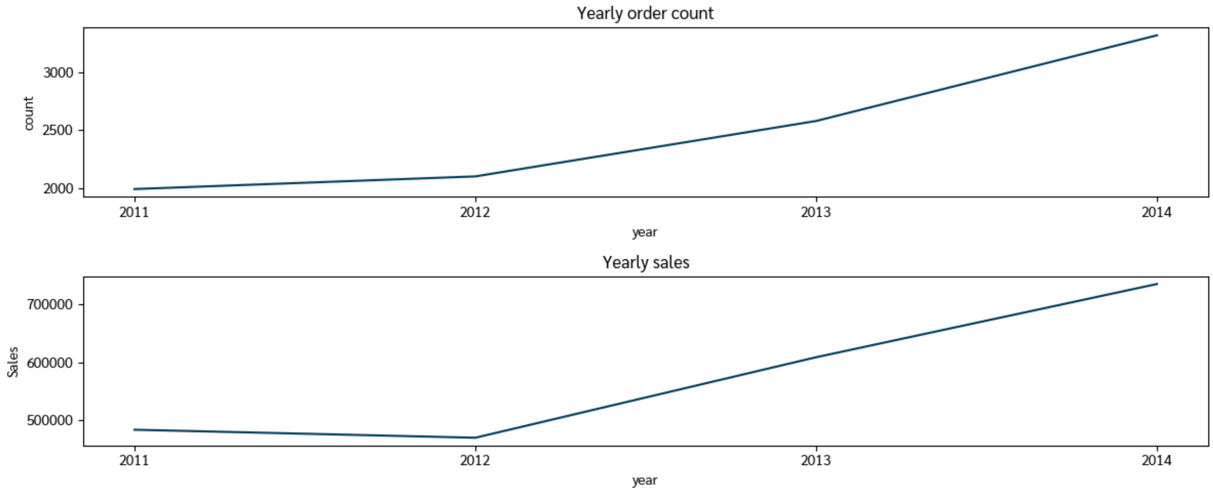
In [65]:

```
# Create a heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f", l
plt.title('Correlation Heatmap')
plt.show()
```



4.1. Sales Performance

In [70]:
```
plt.figure(figsize=(12,5))

plt.subplot(211)
data.groupby(['year'])['Order Date'].count().plot(c='#003f5c')
plt.ylabel('count')
plt.xticks(data.groupby(['year'])['Order Date'].count().index)
plt.title('Yearly order count')

plt.subplot(212)
data.groupby('year')['Sales'].sum().plot(c='#003f5c')
plt.ylabel('Sales')
plt.xticks(data.groupby('year')['Sales'].sum().index)
plt.title('Yearly sales')

plt.tight_layout()
plt.show()
```

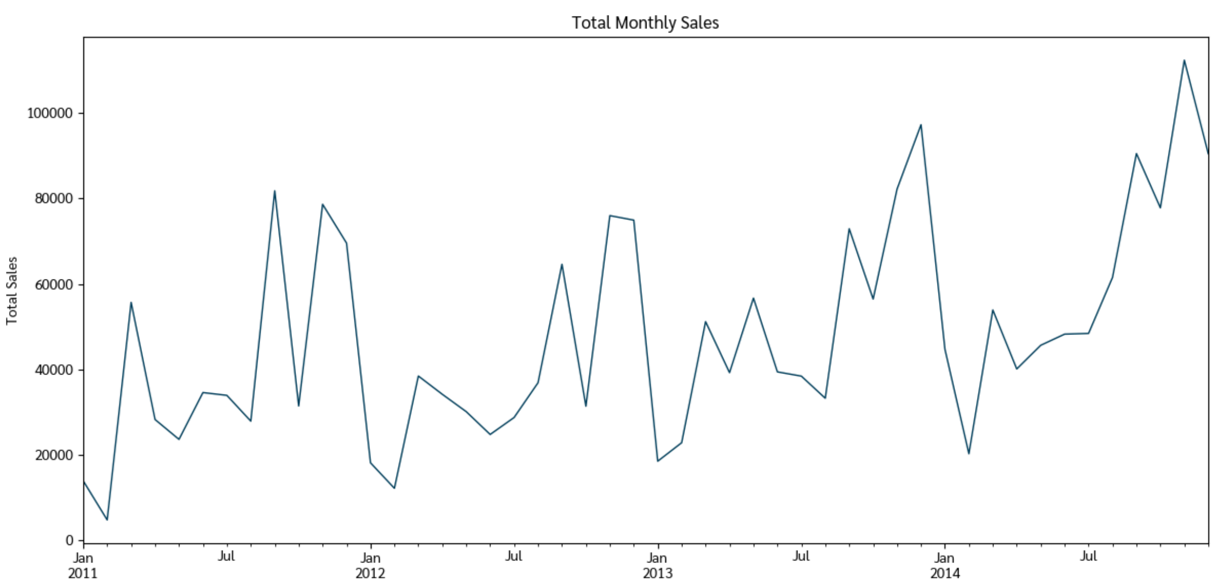


In [71]:
```python
print('Annual total sales: ')
data.groupby('year')['Sales'].sum()
```

```
Annual total sales:
```

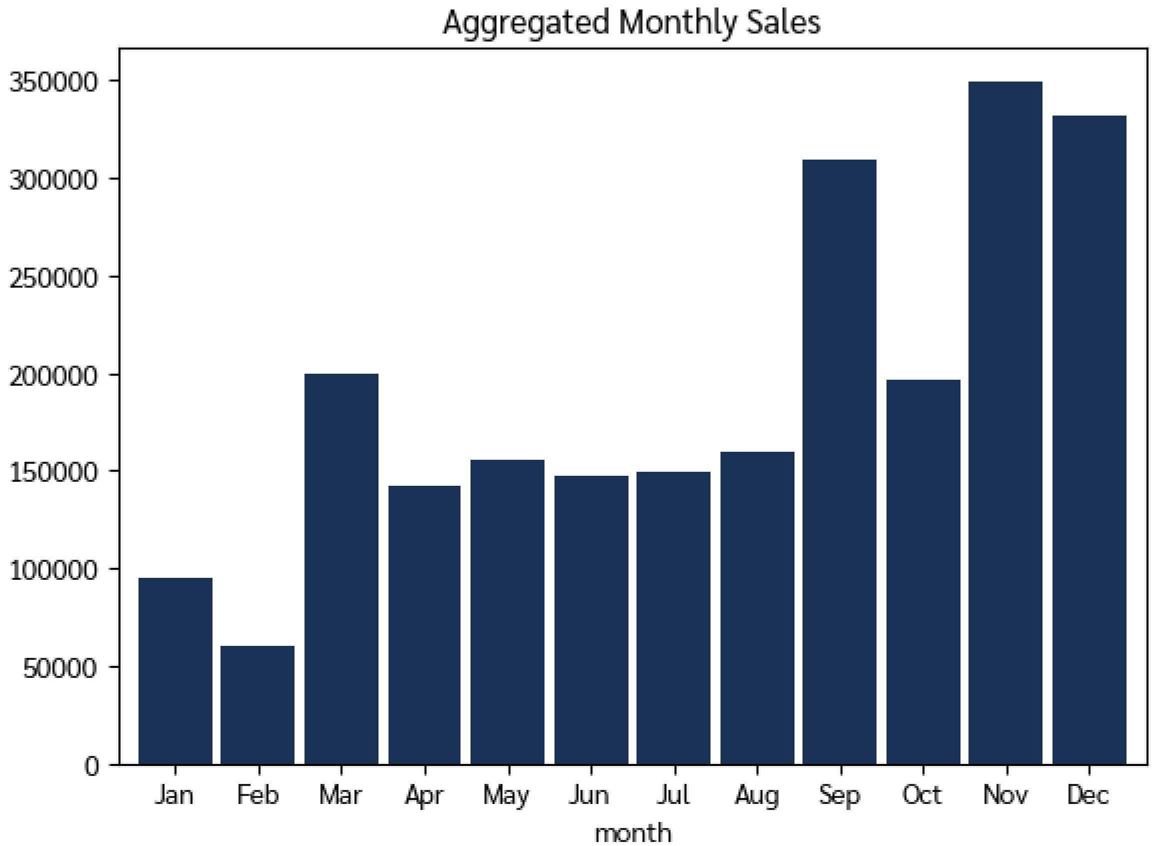
Out[71]:
```
year
2011    484247.4981
2012    470532.5090
2013    608473.8300
2014    733947.0232
Name: Sales, dtype: float64
```

In [72]:
```python
data.groupby('year_month')['Sales'].sum().plot(c='#003f5c',linewidth=1,figsize=(12,
plt.title('Total Monthly Sales')
plt.xlabel('Month')
plt.ylabel('Total Sales')

plt.tight_layout()
plt.show()
```



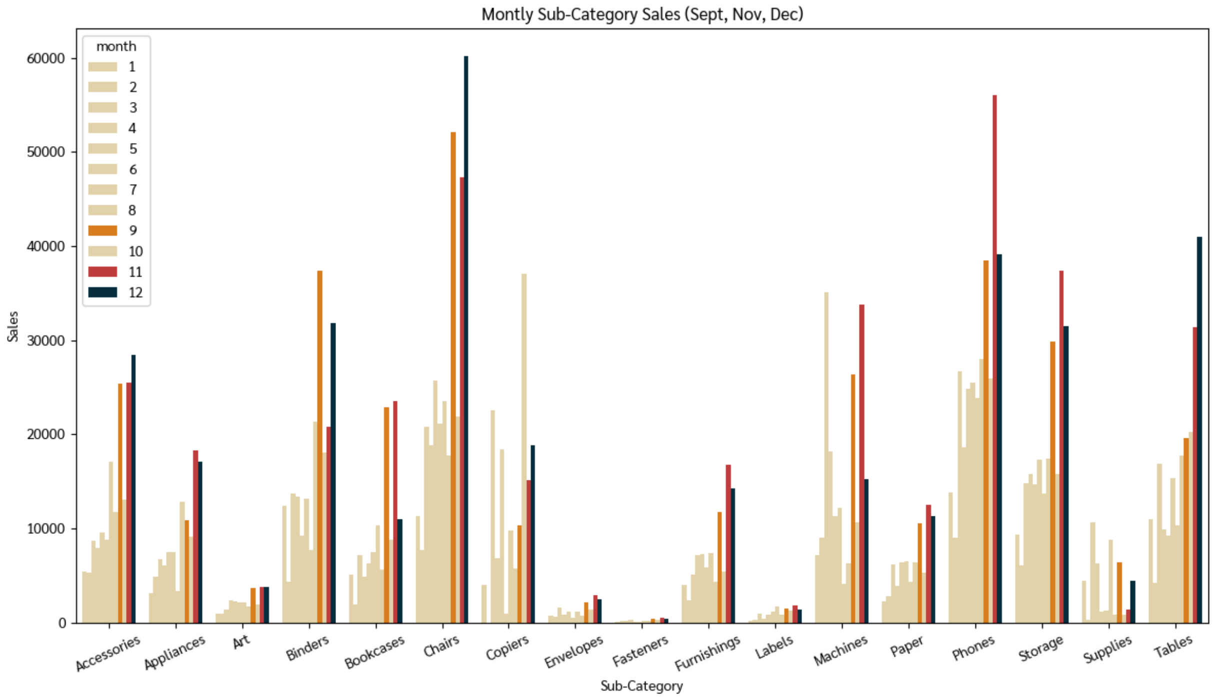
In [73]:
```python
data.groupby('month')['Sales'].sum().plot(kind='bar',color='#1d3557',figsize=(6,4.5
plt.title('Aggregated Monthly Sales')
```

```
plt.xticks(ticks=np.arange(0,12,1),labels=['Jan','Feb','Mar','Apr','May','Jun','Jul

plt.tight_layout()
plt.show()
```

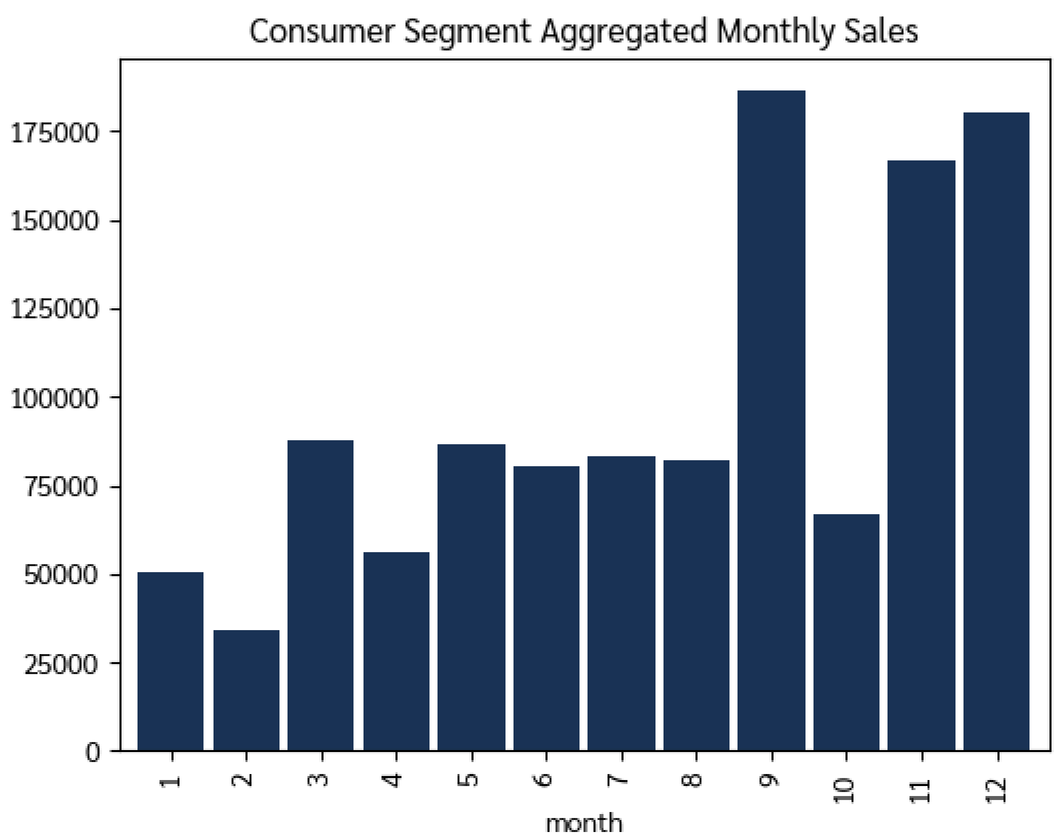


In [74]:

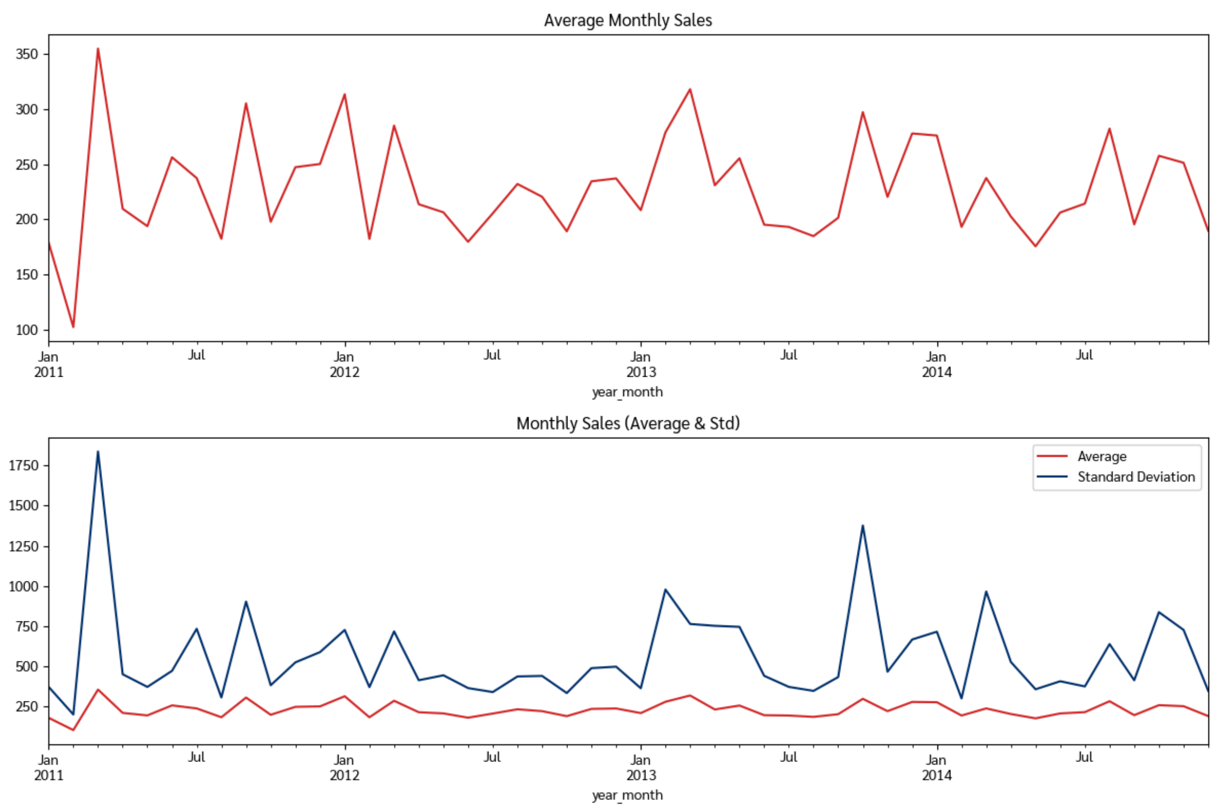
```
month_subcat=pd.DataFrame(data.groupby(['month','Sub-Category'])['Sales'].sum().res
month_subcat

plt.figure(figsize=(12,7))
sns.barplot(data=month_subcat,\
            x='Sub-Category',\
            y='Sales',\
            hue='month',\
            palette=['#e9d8a6','#e9d8a6','#e9d8a6',\
                     '#e9d8a6','#e9d8a6','#e9d8a6',\
                     '#e9d8a6','#e9d8a6','#f77f00',\
                     '#e9d8a6','#d62828','#003049'])
plt.title('Montly Sub-Category Sales (Sept, Nov, Dec)')
plt.xticks(rotation=25)

plt.tight_layout()
```



In [75]:
```
data.query('Segment == "Consumer"').groupby('month')['Sales'].sum().plot(kind='bar'
                                                                          figsize=(6
                                                                          width=.89,
                                                                          color='#1d
plt.title('Consumer Segment Aggregated Monthly Sales')
plt.show()
```



In [76]:
```
plt.figure(figsize=(12,8))

plt.subplot(211)
data.groupby('year_month')['Sales'].mean().plot(linewidth=1.5,color='#d62828')
plt.title('Average Monthly Sales')

plt.subplot(212)
data.groupby('year_month')['Sales'].mean().plot(linewidth=1.5,color='#d62828')
data.groupby('year_month')['Sales'].describe()['std'].plot(linewidth=1.5,color='#03
plt.title('Monthly Sales (Average & Std)')
plt.legend(['Average','Standard Deviation'])

plt.tight_layout()
plt.show()
```



Huge variation in sales within each month can be observed throughout the period. This is confirmed by the monthly sales' standard deviation above. Interestingly, this variation seems to have a pattern. Sales were more variable during March, and around September and October. Interestingly,from April 2012 until the end of the year, there seemed to have low variability in the sales. Along with this, the general sales trend in 2012 was slightly downward, as can be seen in the total yearly sales graph - when total yearly sales dipped a little from 2011 to 2012. On the other hand, sales were more variable in 2011, 2013, and 2014.

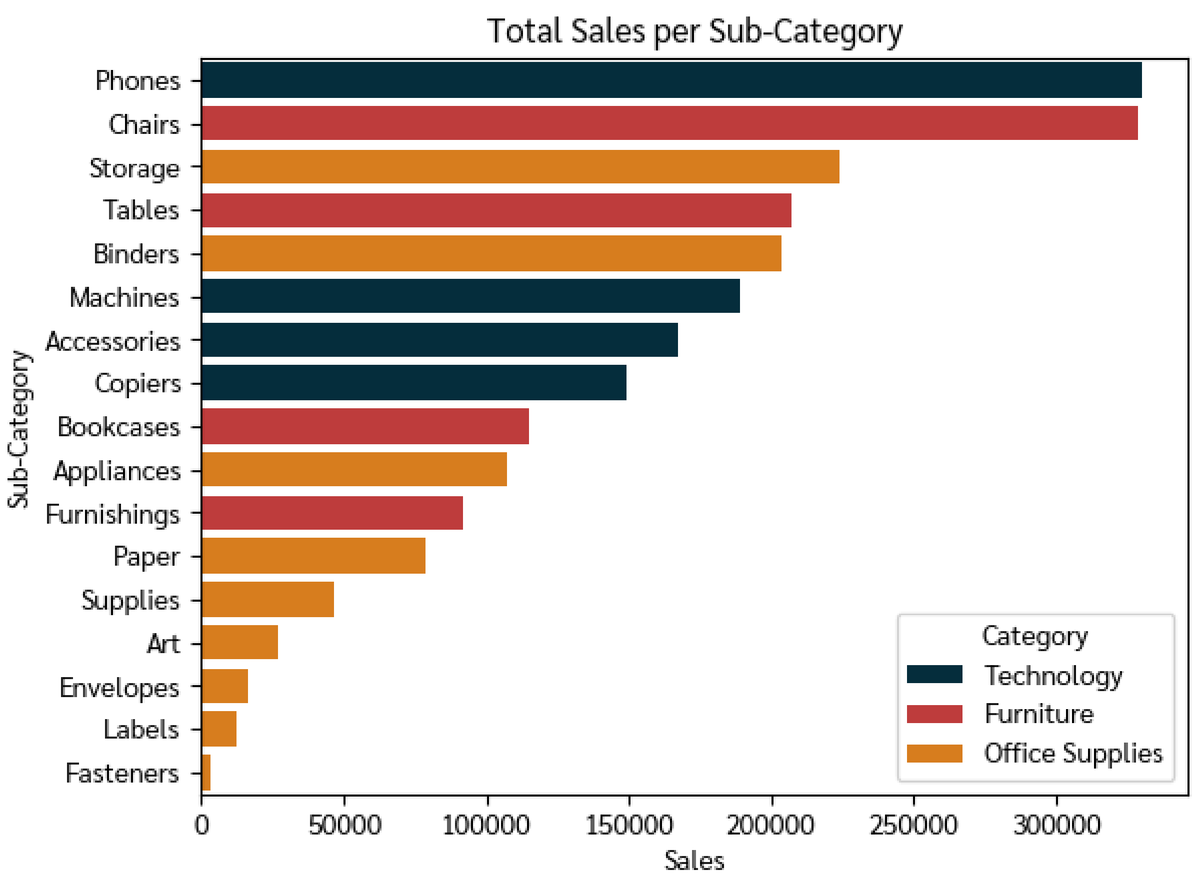
Store sales are typically subject to variablity in sales due to a number of observable factors such as seasonality, customer behaviors, and competitive landscape, among others.

Key findings:

1. Yearly sales had been growing during the 4 year period. Growth was slowest in 2012 and fastest in 2013.
2. Seasonal trends can be observed with sales. Sales generally increase towards the end of the year - November and December (holidays) and in September (possibly due to the opening of schools. Sales under consumer segment also incrased during these months. Increase in school and office supplies sales was also observed).
3. Sales had been very variable especially in March and around September and October. No significant variability was observed from April 2012, until the end of the year.

4.2. Product Categories

In [77]:
```
df_sales=pd.DataFrame(data.groupby(['Category','Sub-Category'])['Sales'].sum()).res

sns.barplot(x='Sales',y='Sub-Category',data=df_sales,hue='Category',palette=['#0030
plt.title('Total Sales per Sub-Category')

plt.show()
```

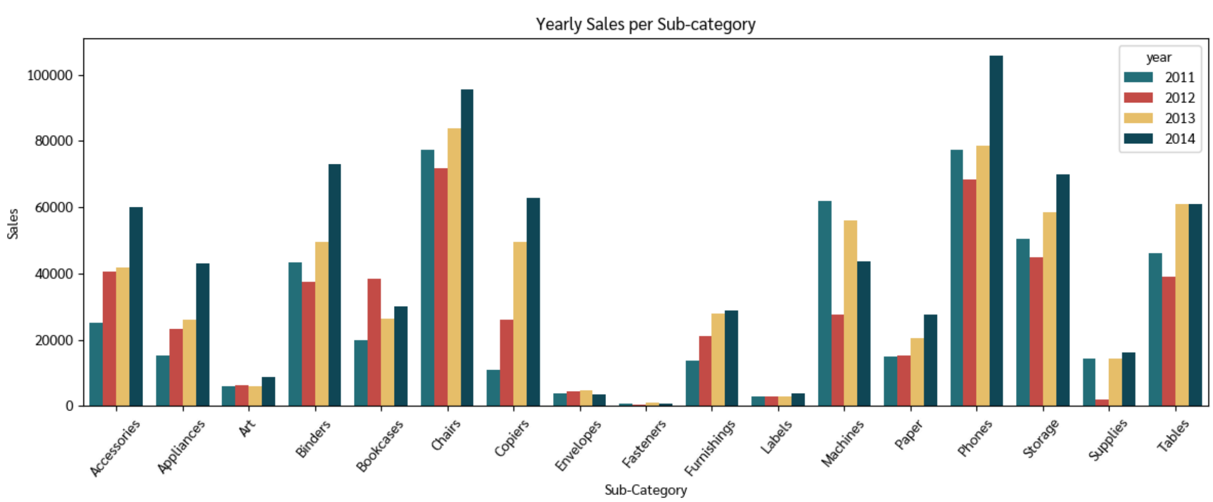


The visualization above shows a general overview of the magnitude of sales for each product sub-category. For technology category, phones are the top sales-generating products. Chairs products for furnitures category, and storage products for office supplies category. Throughout the 4-year period from 2011 - 2014, phones, chairs, and storage products are the three most sales-generating products. Along with them are tables, binders, and machine products.

Under the technology category, copier products are the least performing. For the furniture and office supplies category, furnishings and fasteners are the least performing.

It is worth noting that phones and chairs products sales, which are significantly higher than the rest of the sub-categories, belong to Technology and Furniture product categories, products that are generally expensive.

In [78]:
```
yearly_sales=pd.DataFrame(data.groupby(['Sub-Category','year'])['Sales'].sum()).res
yearly_sales

plt.figure(figsize=(12,5))
sns.barplot(data=yearly_sales,x='Sub-Category',y='Sales',hue='year',palette=['#177e
plt.xticks(rotation=50)
plt.title('Yearly Sales per Sub-category')

plt.tight_layout()
plt.show()
```



Shown is how sales on different products had changed over the 4-year period. For some product categories, sales had been fastest growing in 2014. This was not the case for bookcases, machines, supplies, and tables, which all saw a slow growth in sales in the same year. In 2012, products under binders, phones, storages, supplies, and tables experienced negative growth in sales, especially machine products.

In [79]:
```
yearly_sales['yearly_growth_rate'] = yearly_sales.groupby('Sub-Category')['Sales'].
print('Sales Annual Average Growth Rate:')
pd.DataFrame(yearly_sales.groupby('Sub-Category')['yearly_growth_rate'].mean().sort
```

```
Sales Annual Average Growth Rate:
```

Out[79]:

	yearly_growth_rate
Sub-Category	
Supplies	185.742343
Copiers	85.854740
Appliances	42.879917
Accessories	36.157557
Furnishings	29.479289
Bookcases	24.934417
Paper	24.118681
Binders	21.913015
Art	16.183280
Fasteners	15.954350
Tables	13.476293
Storage	12.922371
Phones	12.573212
Labels	12.083597
Machines	8.006001
Chairs	7.906943
Envelopes	-2.240016

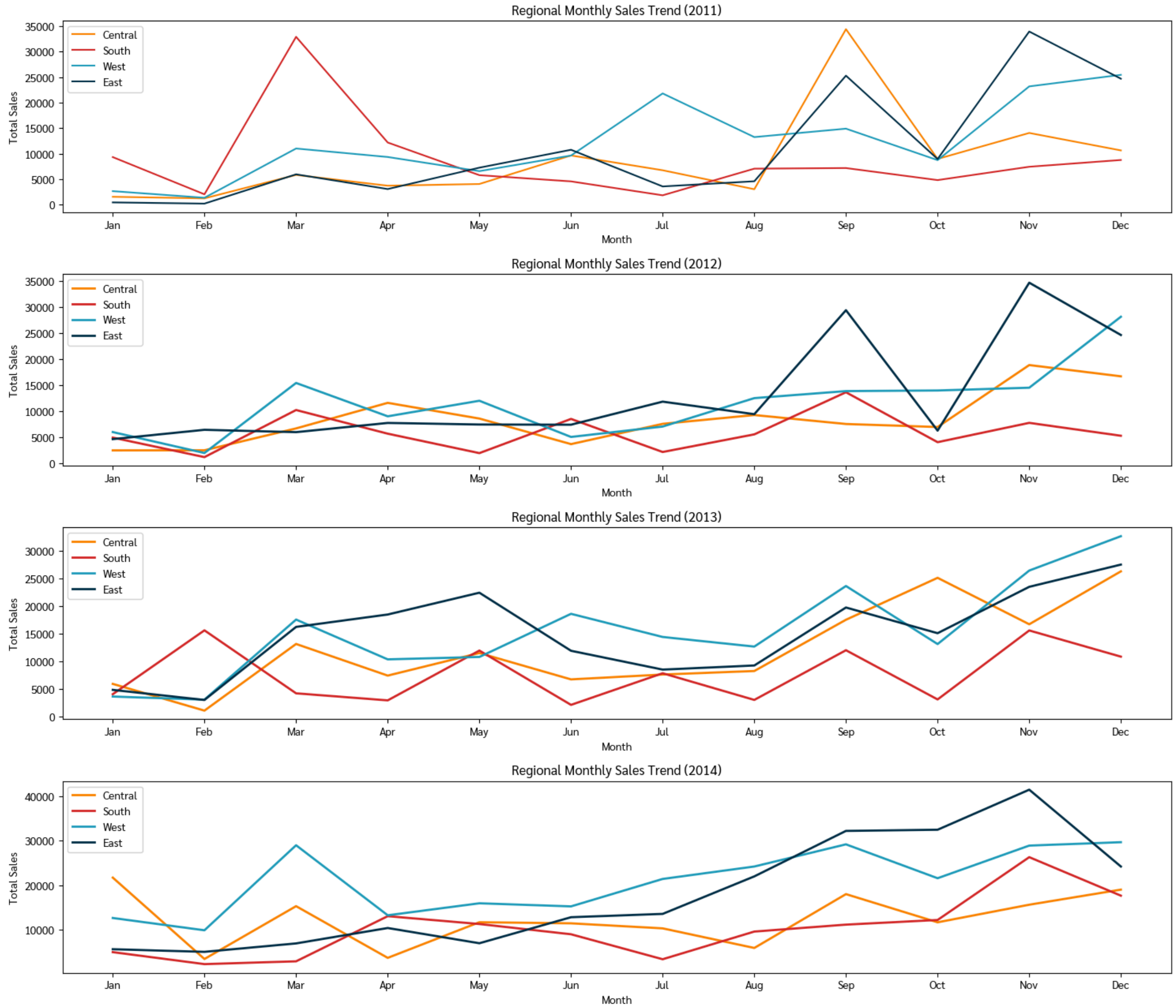
By average sales annual growth rate, envelope products had been the slowest while supplies products had been the fastest at 185% annual average growth rate (AAGR), followed by copier and appliances products at 86% and 43% AAGR, respectively.

4.3. Geographic Insights

In [80]:

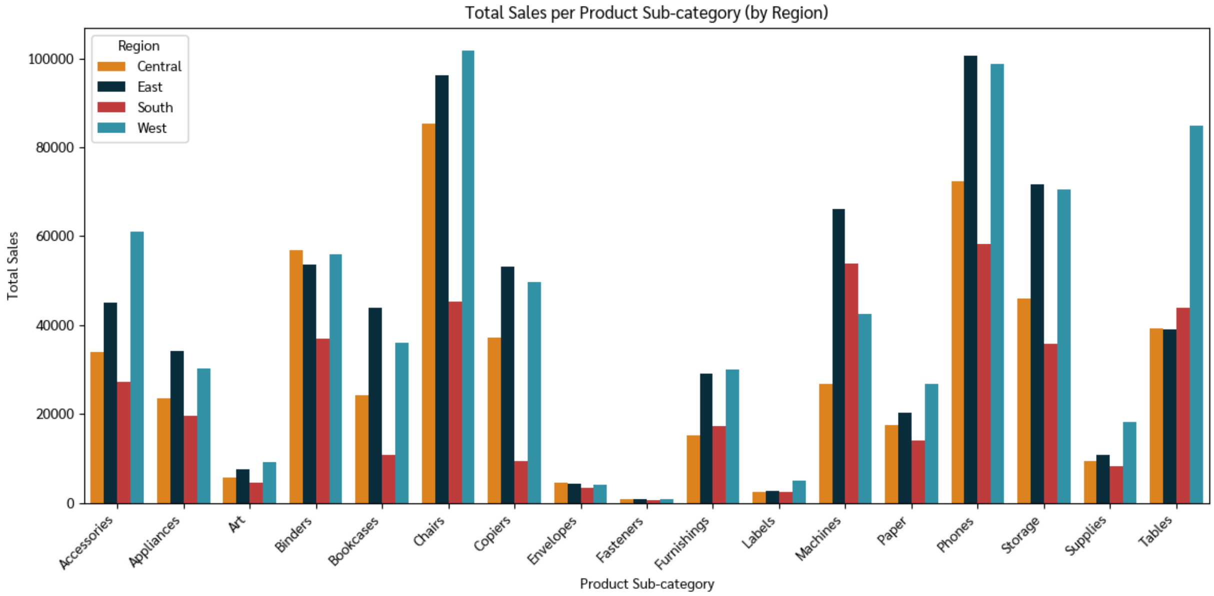
```
data11=data.query('year == 2011')
data12=data.query('year == 2012')
data13=data.query('year == 2013')
data14=data.query('year == 2014')

plt.figure(figsize=(15,13))

plt.subplot(411)
data11.query("Region == 'Central'").groupby('month')['Sales'].sum().plot(c='#fb8500
data11.query("Region == 'South'").groupby('month')['Sales'].sum().plot(c='#d62828',
data11.query("Region == 'West'").groupby('month')['Sales'].sum().plot(c='#219ebc',l
data11.query("Region == 'East'").groupby('month')['Sales'].sum().plot(c='#023047',l
```

```
plt.title('Regional Monthly Sales Trend (2011)')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(ticks=np.arange(1,13,1),labels=['Jan','Feb','Mar','Apr','May','Jun','Jul
plt.legend(['Central','South','West','East'])

plt.subplot(412)
data12.query("Region == 'Central'").groupby('month')['Sales'].sum().plot(c='#fb8500
data12.query("Region == 'South'").groupby('month')['Sales'].sum().plot(c='#d62828',
data12.query("Region == 'West'").groupby('month')['Sales'].sum().plot(c='#219ebc',l
data12.query("Region == 'East'").groupby('month')['Sales'].sum().plot(c='#023047',l

plt.title('Regional Monthly Sales Trend (2012)')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(ticks=np.arange(1,13,1),labels=['Jan','Feb','Mar','Apr','May','Jun','Jul
plt.legend(['Central','South','West','East'])

plt.subplot(413)
data13.query("Region == 'Central'").groupby('month')['Sales'].sum().plot(c='#fb8500
data13.query("Region == 'South'").groupby('month')['Sales'].sum().plot(c='#d62828',
data13.query("Region == 'West'").groupby('month')['Sales'].sum().plot(c='#219ebc',l
data13.query("Region == 'East'").groupby('month')['Sales'].sum().plot(c='#023047',l

plt.title('Regional Monthly Sales Trend (2013)')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(ticks=np.arange(1,13,1),labels=['Jan','Feb','Mar','Apr','May','Jun','Jul
plt.legend(['Central','South','West','East'])

plt.subplot(414)
data14.query("Region == 'Central'").groupby('month')['Sales'].sum().plot(c='#fb8500
data14.query("Region == 'South'").groupby('month')['Sales'].sum().plot(c='#d62828',
data14.query("Region == 'West'").groupby('month')['Sales'].sum().plot(c='#219ebc',l
data14.query("Region == 'East'").groupby('month')['Sales'].sum().plot(c='#023047',l

plt.title('Regional Monthly Sales Trend (2014)')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(ticks=np.arange(1,13,1),labels=['Jan','Feb','Mar','Apr','May','Jun','Jul
plt.legend(['Central','South','West','East'])

plt.tight_layout()
plt.show()
```



As shown before, seasonal trend occured, with sales increasing in holidays (November and December), opening of classes (September), and possibly Easter (March). This is also the case with regional sales data per year. Total sales has been generally higher most of the year in the West, followed by the East compared to the remaining two regions. Sales in the South has been lower each year compared to other regions with the exception in March 2011 when sales in the South was more than thrice the next best performer.

In [81]:
```
reg_sub=pd.DataFrame(data.groupby(['Region','Sub-Category'])['Sales'].sum()).reset_

plt.figure(figsize=(12, 6))
sns.barplot(data=reg_sub, x='Sub-Category', y='Sales', hue='Region',palette=['#fb85
plt.xlabel('Product Sub-category')
plt.ylabel('Total Sales')
plt.title('Total Sales per Product Sub-category (by Region)')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Region')
plt.tight_layout()
plt.show()
```



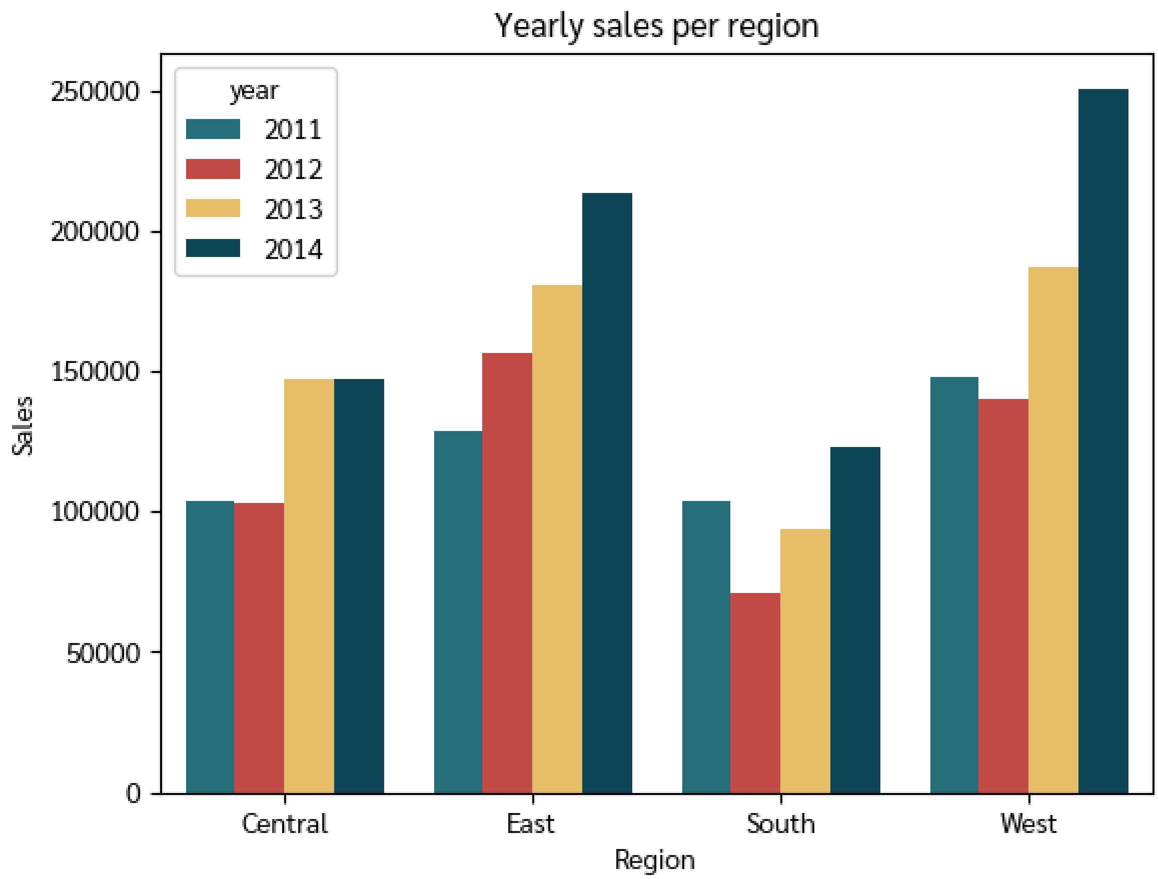
Sales for most sub-categories has been lower in the South and higher in the West. For certain products, sales has been significantly lower in the South. For instance, sales for chairs and copiers products in the South are significantly lower compared to all other regions, while machine and table products sales has been higher in the South than in other Regions. For most sub-categories, sales in the Central region was just slightly higher than that of the South. Another notable observation from this is that sales of certain products sub-category are significantly higher in the West than all other remaining regions. Those products are table, office supplies, and technology accessories.

The dataset is fictional and does not provide background information about the regions. However, based on the sales, it can be hypothesized that more offices and business districts are probably located in the West and in the East than in Central and South regions. Office supplies sales like storages, binders, and appliances are significantly lower in the South and Central than the remaining regions. Conversely, sales for these products and tech ones are higher in the West and in the East. However, it is worth noting that the second highest sales for machine products was in the South.

Take note that this sales refer to total sales from 2011 - 2014 in each region, which does not show how sales behaved throughout the years. This visualization rather intends to show a rough estimate of the magnitude of sales for different products in each region

In [82]:

```
year_s=pd.DataFrame(data.groupby(['Region','year'])['Sales'].sum()).reset_index()
year_p=pd.DataFrame(data.groupby(['Region','year'])['(net)_profit_before_discount']

sns.barplot(data=year_s,x='Region',y='Sales',hue='year',palette=['#177e89','#db3a34
plt.title('Yearly sales per region')
plt.show()
```



Shown is how sales for each region changed over time. In the Central region, a sharp positive growth was observed from 2013, while yearly positive growth was consistent in the East. For the South region, a negative growth was observed in 2012, but the region rebounded thereafter. Simililarly for the West, negative growth was observed in 2012, but rebounded and sustained positive growth thereafter.

It was observed that there was a slight dip in total sales in 2012. From the visualization above, it can be inferred that South had contributed the most to that dip, followed by the West and then the Central. Interestingly, the East region still grew positively in 2012.

In [83]:
```
year_s['yearly_growth_rate']=year_s.groupby('Region')['Sales'].pct_change() * 100

print('Sales Average Annual Growth Rate (AAGR, 2011-2014) :')
pd.DataFrame(year_s.groupby('Region')['yearly_growth_rate'].mean())
```

```
Sales Average Annual Growth Rate (AAGR, 2011-2014) :
```

Out[83]:

	yearly_growth_rate
Region	
Central	14.052441
East	18.361897
South	10.423004
West	20.759458

4.4. Profitability

In [84]:
```
yearly_summary = data.groupby('year')[['Sales','net_profit']].sum()

yearly_summary['profit_margin'] = (yearly_summary['net_profit'] / yearly_summary['S
yearly_summary
```

Out[84]:

	Sales	net_profit	profit_margin
year			
2011	484247.4981	49543.9741	10.231126
2012	470532.5090	61618.6037	13.095504
2013	608473.8300	81726.9308	13.431462
2014	733947.0232	93507.5131	12.740363

In [85]:
```
profit_margin_df=pd.DataFrame(data.groupby(['Category','Sub-Category'])['profit_mar
print("This table shows exact values on the average profit margin of each product s
profit_margin_df
```

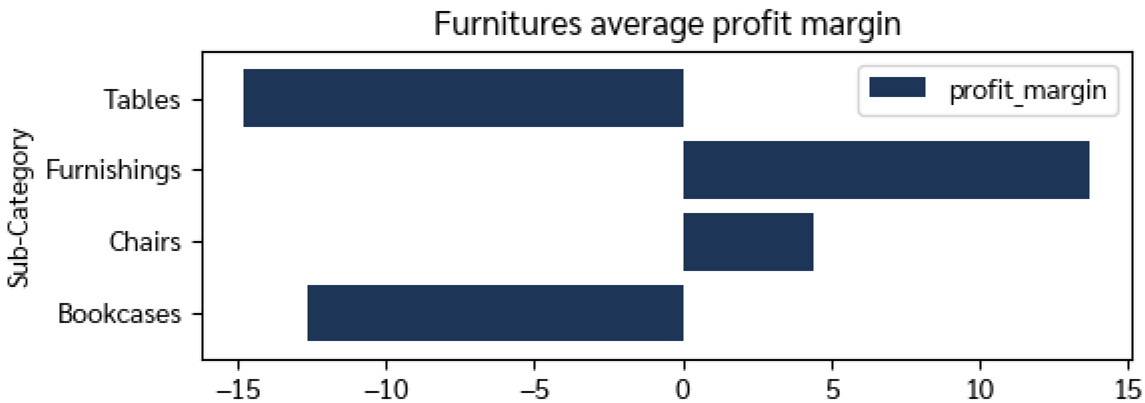
```
This table shows exact values on the average profit margin of each product sub-categ
ory:
```

Out[85]:

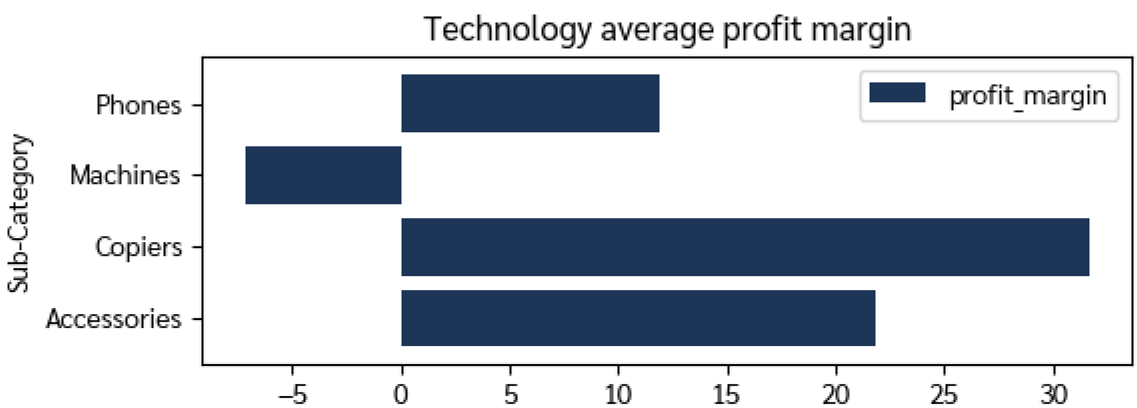
	Category	Sub-Category	profit_margin
0	Furniture	Bookcases	-12.664007
1	Furniture	Chairs	4.389963
2	Furniture	Furnishings	13.706635
3	Furniture	Tables	-14.772653
4	Office Supplies	Appliances	-15.686934
5	Office Supplies	Art	25.164573
6	Office Supplies	Binders	-19.959510
7	Office Supplies	Envelopes	42.313976
8	Office Supplies	Fasteners	29.917051
9	Office Supplies	Labels	42.966346
10	Office Supplies	Paper	42.560036
11	Office Supplies	Storage	8.911348
12	Office Supplies	Supplies	11.203947
13	Technology	Accessories	21.820968
14	Technology	Copiers	31.719363
15	Technology	Machines	-7.202622
16	Technology	Phones	11.922197

In [86]:

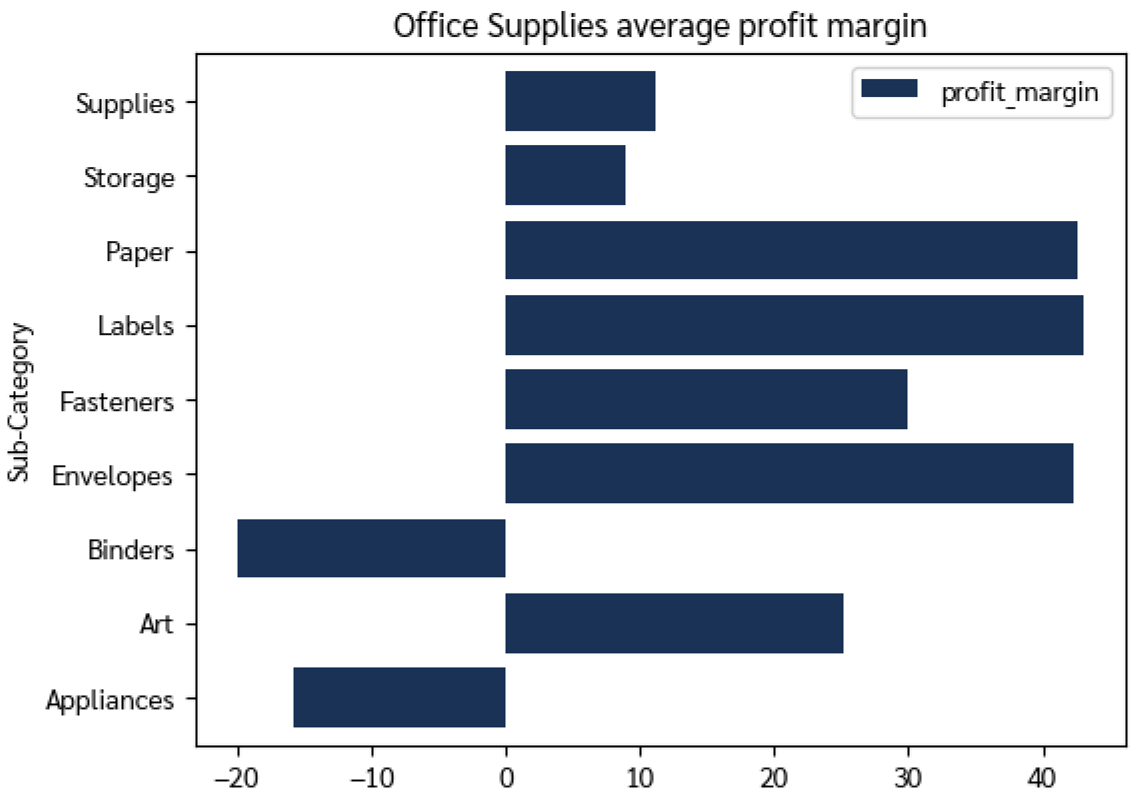
```
barr1=profit_margin_df[profit_margin_df['Category']==\
                      'Furniture'][['Sub-Category',\
                                    'profit_margin']].set_index('Sub-Category')

barr1.plot(kind='barh',\
           title='Furnitures average profit margin',\
           color='#1d3557',\
           figsize=(6,2),\
           width=.8)

plt.show()
```



In [87]:
```
barr3=profit_margin_df[profit_margin_df['Category']==\
                      'Technology'][['Sub-Category',\
                                    'profit_margin']].set_index('Sub-Category')

barr3.plot(kind='barh',title='Technology average profit margin',\
           color ='#1d3557',\
           figsize=(6,2),\
           width=.8)

plt.show()
```

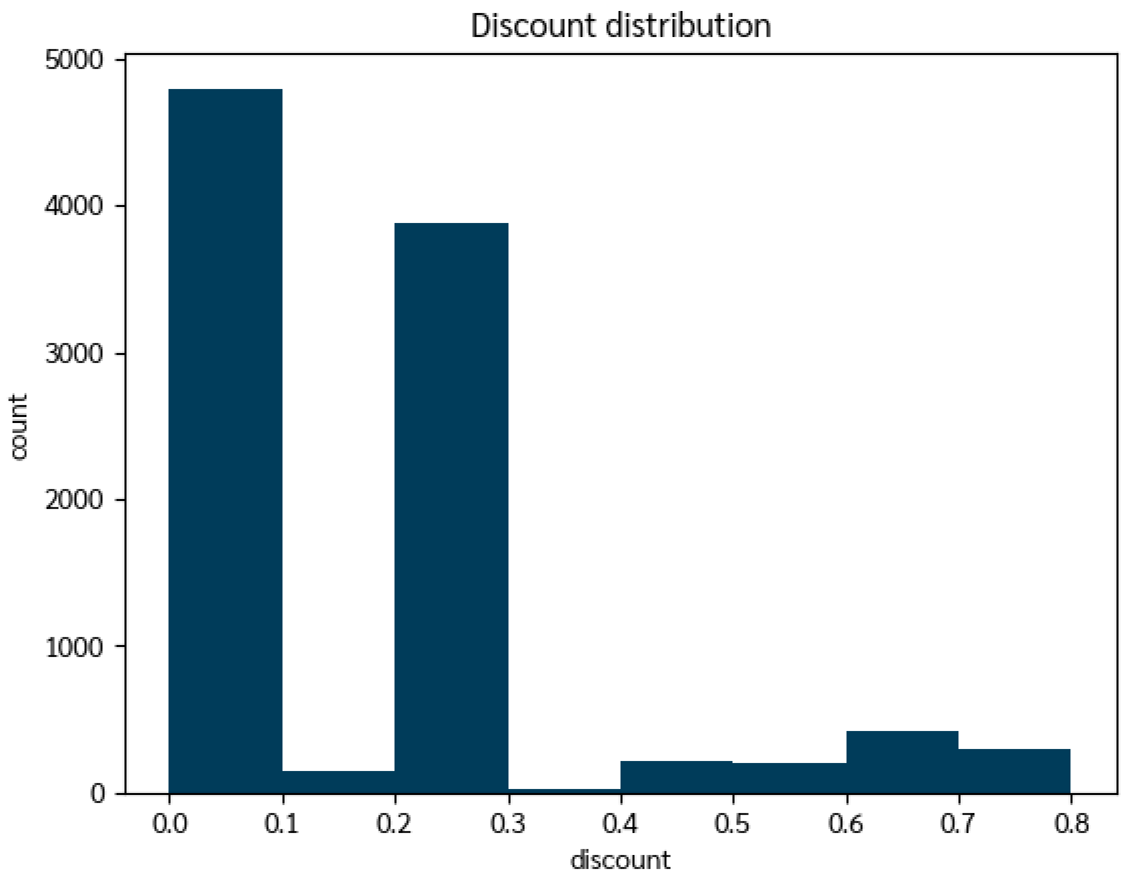


In [88]:
```
barr2=profit_margin_df[profit_margin_df['Category']==\
                      'Office Supplies'][['Sub-Category',\
                                    'profit_margin']].set_index('Sub-Category')

barr2.plot(kind='barh',\
           title='Office Supplies average profit margin',\
           color='#1d3557',\
           figsize=(6,4.5),\
           width=.8)

plt.show()
```

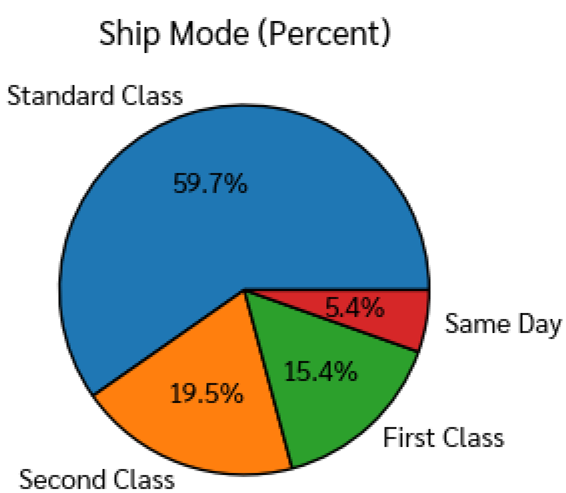


In [90]:

```
plt.hist(data=data,x='Discount',bins=8,color='#003f5c') # distribution of discount
plt.title('Discount distribution')
plt.xlabel('discount')
plt.ylabel('count')
plt.show()
```



In [91]:
```
i3=data['Ship Mode'].value_counts()/len(data)*100
i3

plt.figure(figsize=(5,3))
plt.pie(i3,labels=i3.index,autopct='%.1f%%',textprops={'fontsize':10},wedgeprops={'
plt.title('Ship Mode (Percent)')

plt.show()
```



In [92]:
```
print('Following are the average order fulfillment time for corresponding ship mode
```

```
print("Standard Class:",data[data['Ship Mode'] == 'Standard Class']['order_fulfillm
print("Second Class:",data[data['Ship Mode'] == 'Second Class']['order_fulfillment_
print("First Class:",data[data['Ship Mode'] == 'First Class']['order_fulfillment_ti
```

```
Following are the average order fulfillment time for corresponding ship modes:
Standard Class: 5 days 00:10:22.520107238
Second Class: 3 days 05:45:44.884318766
First Class: 2 days 04:22:09.518855656
```

5. Conclusion

The company experienced year-over-year sales growth, with 2013 showing the highest growth and 2012 the slowest. Seasonal sales trends were also evident, with spikes in November, December, and September. Sales were highly variable in March, and around September and October. Phones, chairs, and storage products led sales within their respective categories, while copiers, furnishings, and fasteners lagged.

By region, the West and East had higher sales across most categories, while the South had consistently lower sales. Central, on the other hand, showed positive growth in 2012, in contrast to all other regions in the same year. Furthermore, the West had the fastest average annual sales growth.

Despite fluctuating profitability, the company maintained a 10%+ profit margin from 2011 to 2014. Chairs, phones, and storage products were the least profitable, while furnishings, copiers, and labels were the most profitable sub-categories. Sales discounts significantly impacted profits, with tables and office supplies experiencing the largest drops.

In conclusion, while the company saw sales growth and maintained profitability, seasonal and regional variations played a significant role in its performance, and discounts affected profit margins.

5.1. Recommendation

Based on the the key findings, the following recommendations are presented:

- Given evident seasonal trends, consider adjusting inventory levels to cater increased demand during November, December, and September. With this, stockouts during peak periods and overstocking during off-peak can be prevented/minimized.
- Formulate strategies to boost sales in regions with lower performance, especially in the South. It is also very important to conduct further investigation/research on regional preferences, adjust product offerings or marketing to better cater to local/targeted markets.
- Further assess the impact of discounts on profit margins. A comprehensive review of discount strategy should be done to maintain profitability while attracting more customers or increasing sales.

- Expand or diversify product range. Consider introducing new products/categories or enhancing existing ones to tap into more/other customer segments.
- Examine the performance of product sub-categories in more detail. Identify which specific products within each category are the most and least profitable, and decide on whether to optimize or discontinue specific products.
- Optimize supply chain management to reduce variability in sales (among other factors for sales variability). Implement more efficient inventory management and demand forecasting techniques to mitigate stockouts and overstocking.
- Reevaluate regional growth strategies based on the relative performance of each region. Consider shifting resources and marketing efforts toward regions with higher growth potential.
- Continue efforts to control costs and maintain a good profit margin, even during periods of fluctuating profitability.