Store Sales and Profit Analysis using Python

Problem Statement Store sales and profit analysis help businesses identify areas for improvement and make data-driven decisions to optimize their operations, pricing, marketing, and inventory management strategies to drive revenue and growth. I hope you liked this Project on the task of analyzing the sales and profit of a store using Python. Feel free to ask valuable questions in the comments section below.

Import Library

In [1]:
```
import pandas as pd
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\Syed Arif\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWar
ning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of Sc
iPy (detected version 1.25.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Uploading Csv fle

In [3]:
```
df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\SuperStore_Sales_Dataset.csv")
```

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

In [4]:
```
df.head()
```

Out[4]:

	Row ID+O6G3A1:R6	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
0	4918	CA-2019-160304	01-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaitl
1	4919	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaitl
2	4920	CA-2019-160304	02-01-2019	07-01-2019	Standard Class	BM-11575	Brendan Murry	Corporate	United States	Gaitl
3	3074	CA-2019-125206	03-01-2019	05-01-2019	First Class	LR-16915	Lena Radford	Consumer	United States	Los
4	8604	US-2019-116365	03-01-2019	08-01-2019	Standard Class	CA-12310	Christine Abelman	Corporate	United States	Sar

5 rows × 23 columns

.tail()

tail is used to show rows by Descending order

In [5]:
```
df.tail()
```

Out[5]:

	Row ID+O6G3A1:R6	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
5896	907	CA-2020-143259	30-12-2020	03-01-2021	Standard Class	PO-18865	Patrick O'Donnell	Consumer	United States	`
5897	1297	CA-2020-115427	30-12-2020	03-01-2021	Standard Class	EB-13975	Erica Bern	Corporate	United States	
5898	5092	CA-2020-156720	30-12-2020	03-01-2021	Standard Class	JM-15580	Jill Matthias	Consumer	United States	L
5899	909	CA-2020-143259	30-12-2020	03-01-2021	Standard Class	PO-18865	Patrick O'Donnell	Consumer	United States	`
5900	5093	CA-2020-151450	31-12-2020	04-01-2021	Standard Class	JM-15580	Jill Matthias	Consumer	United States	L

5 rows × 23 columns

.shape

It show the total no of rows & Column in the dataset

In [6]:
```
df.shape
```

Out[6]:
```
(5901, 23)
```

.Columns

It show the no of each Column

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Row ID+O6G3A1:R6', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mod
e',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Region', 'Product ID', 'Category', 'Sub-Category', 'Product Name',
       'Sales', 'Quantity', 'Profit', 'Returns', 'Payment Mode', 'ind1',
       'ind2'],
      dtype='object')
```

.dtypes

This Attribute show the data type of each column

In [8]:
```
df.dtypes
```

Out[8]:
```
Row ID+O6G3A1:R6      int64
Order ID             object
Order Date           object
Ship Date            object
Ship Mode            object
Customer ID          object
Customer Name        object
Segment              object
Country              object
City                 object
State                object
Region               object
Product ID           object
Category             object
Sub-Category         object
Product Name         object
Sales               float64
Quantity              int64
Profit              float64
Returns             float64
Payment Mode         object
ind1                float64
ind2                float64
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [9]:
```
df["Category"].unique()
```

Out[9]:
```
array(['Furniture', 'Technology', 'Office Supplies'], dtype=object)
```

.nuique()

It will show the total no of unque value from whole data frame

In [10]:
```
df.nunique()
```

Out[10]:
```
Row ID+O6G3A1:R6    5900
Order ID            3003
Order Date           643
Ship Date            690
Ship Mode              4
Customer ID          773
Customer Name        773
Segment                3
Country                1
City                 452
State                 49
Region                 4
Product ID          1755
Category               3
Sub-Category          17
Product Name        1742
Sales               5109
Quantity              14
Profit              4739
Returns                1
Payment Mode           3
ind1                   0
ind2                   0
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [11]:
```
df.describe()
```

Out[11]:

	Row ID+O6G3A1:R6	Sales	Quantity	Profit	Returns	ind1	ind2
count	5901.000000	5901.000000	5901.000000	5901.000000	287.0	0.0	0.0
mean	5022.422471	265.345589	3.781901	29.700408	1.0	NaN	NaN
std	2877.977184	474.260645	2.212917	259.589138	0.0	NaN	NaN
min	1.000000	0.836000	1.000000	-6599.978000	1.0	NaN	NaN
25%	2486.000000	71.976000	2.000000	1.795500	1.0	NaN	NaN
50%	5091.000000	128.648000	3.000000	8.502500	1.0	NaN	NaN
75%	7456.000000	265.170000	5.000000	28.615000	1.0	NaN	NaN
max	9994.000000	9099.930000	14.000000	8399.976000	1.0	NaN	NaN

.value_counts

It Shows all the unique values with their count

In [12]:
```
df["Category"].value_counts()
```

Out[12]:
```
Office Supplies    3569
Furniture          1249
Technology         1083
Name: Category, dtype: int64
```

.isnull()

It shows the how many null values

In [13]:
```
df.isnull()
```

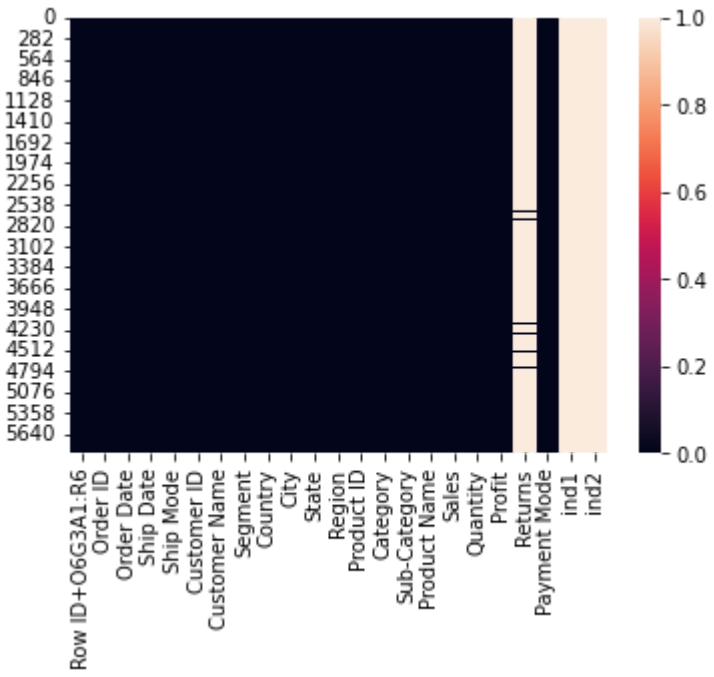
Out[13]:

	Row ID+O6G3A1:R6	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
5896	False	False	False	False	False	False	False	False	False	False
5897	False	False	False	False	False	False	False	False	False	False
5898	False	False	False	False	False	False	False	False	False	False
5899	False	False	False	False	False	False	False	False	False	False
5900	False	False	False	False	False	False	False	False	False	False

5901 rows × 23 columns

In [14]:
```
sns.heatmap(df.isnull())
```

Out[14]: <AxesSubplot:>



In [15]:
```
df.duplicated().sum()
```

Out[15]: 0

In [16]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])

df['Order Month'] = df['Order Date'].dt.month
df['Order Year'] = df['Order Date'].dt.year
df['Order Day of Week'] = df['Order Date'].dt.dayofweek
```

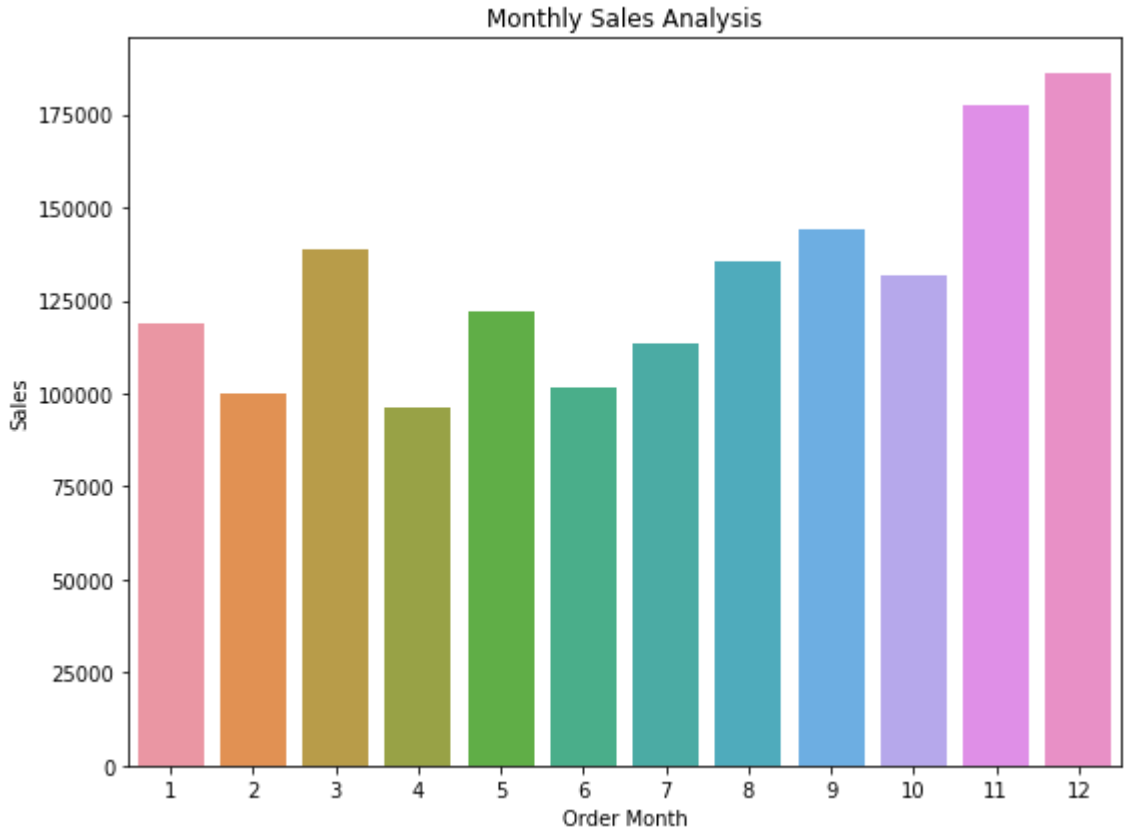
```
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\Syed Arif\anaconda3\lib\site-packages\pandas\core\tools\datetimes.
py:1047: UserWarning: Parsing '21-04-2019' in DD/MM/YYYY format. Provide fo
rmat or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\Syed Arif\anaconda3\lib\site-packages\pandas\core\tools\datetimes.
py:1047: UserWarning: Parsing '23-04-2019' in DD/MM/YYYY format. Provide fo
rmat or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\Syed Arif\anaconda3\lib\site-packages\pandas\core\tools\datetimes.
py:1047: UserWarning: Parsing '20-04-2019' in DD/MM/YYYY format. Provide fo
rmat or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\Syed Arif\anaconda3\lib\site-packages\pandas\core\tools\datetimes.
py:1047: UserWarning: Parsing '22-04-2019' in DD/MM/YYYY format. Provide fo
rmat or specify infer_datetime_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\Syed Arif\anaconda3\lib\site-packages\pandas\core\tools\datetimes.
py:1047: UserWarning: Parsing '25-04-2019' in DD/MM/YYYY format. Provide fo
```

In [17]:
```
sales_by_month = df.groupby('Order Month')['Sales'].sum().reset_index()
sales_by_month
```

Out[17]:

	Order Month	Sales
0	1	118934.4350
1	2	100157.8994
2	3	138636.1288
3	4	96020.2796
4	5	122174.7682
5	6	101640.2685
6	7	113457.4825
7	8	135676.2924
8	9	143865.7503
9	10	131574.3122
10	11	177411.8250
11	12	186254.8813

In [18]:
```
# Create a scatter plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=sales_by_month, x='Order Month', y='Sales')
plt.title('Monthly Sales Analysis')
plt.xlabel('Order Month')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```

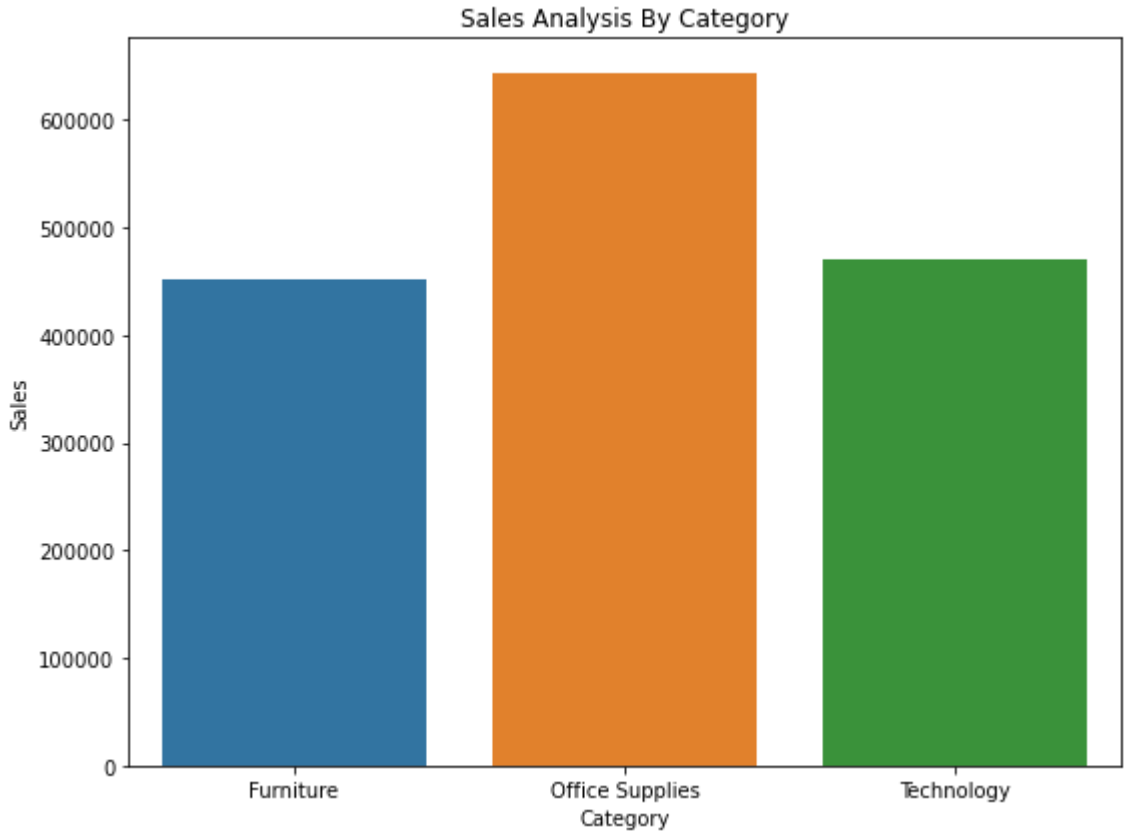


In [19]:
```
sales_by_category = df.groupby('Category')['Sales'].sum().reset_index()
sales_by_category
```

Out[19]:

	Category	Sales
0	Furniture	451508.6452
1	Office Supplies	643707.6870
2	Technology	470587.9910

In [20]:
```
# Create a scatter plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=sales_by_category, x='Category', y='Sales')
plt.title('Sales Analysis By Category')
plt.xlabel('Category')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```

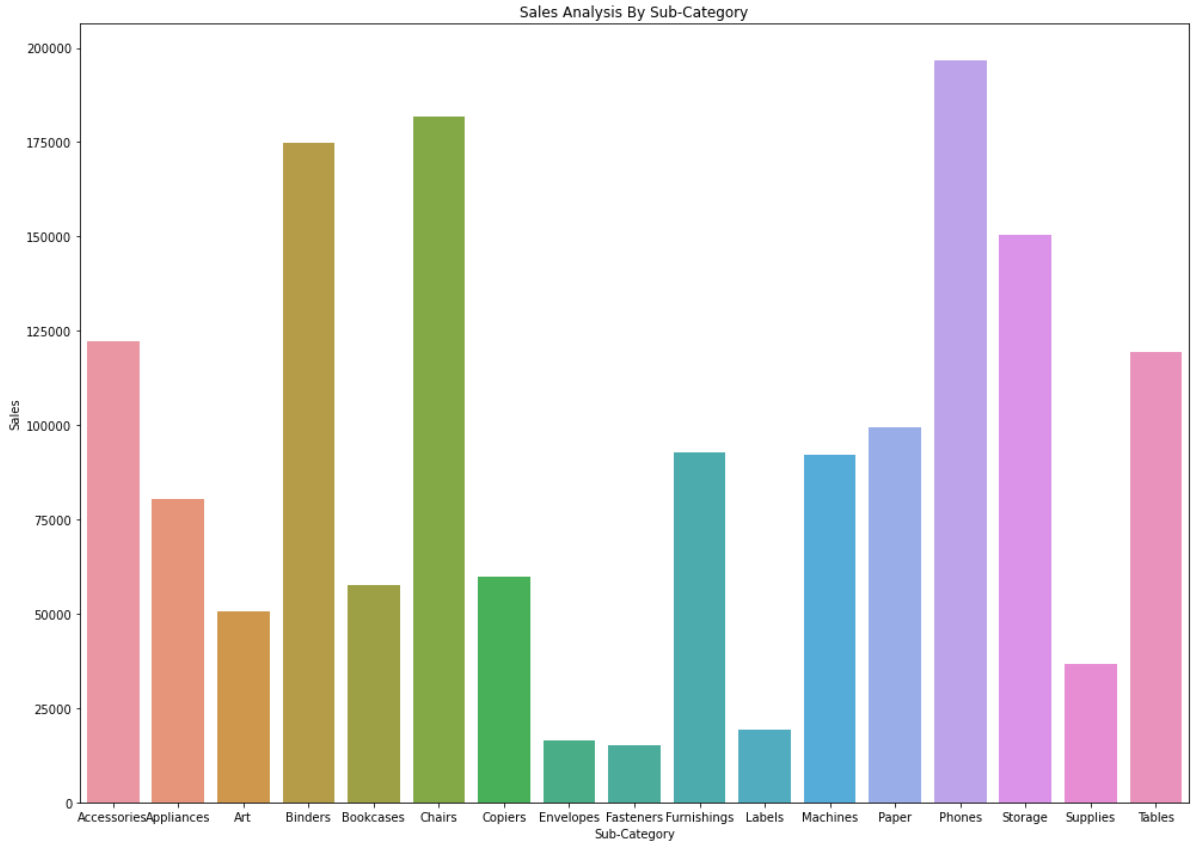


In [21]:
```
sales_by_sub_category = df.groupby('Sub-Category')['Sales'].sum().reset_index()
sales_by_sub_category
```

Out[21]:

	Sub-Category	Sales
0	Accessories	122301.0860
1	Appliances	80305.2470
2	Art	50762.9760
3	Binders	174978.3900
4	Bookcases	57577.6862
5	Chairs	181945.9980
6	Copiers	59735.7980
7	Envelopes	16542.4640
8	Fasteners	15205.2380
9	Furnishings	92691.2180
10	Labels	19397.4560
11	Machines	91987.5610
12	Paper	99453.6120
13	Phones	196563.5460
14	Storage	150341.3180
15	Supplies	36720.9860
16	Tables	119293.7430

In [22]:
```
# Create a scatter plot to visualize the correlation
plt.figure(figsize=(14, 10))
sns.barplot(data=sales_by_sub_category , x='Sub-Category', y='Sales')
plt.title('Sales Analysis By Sub-Category')
plt.xlabel('Sub-Category')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```

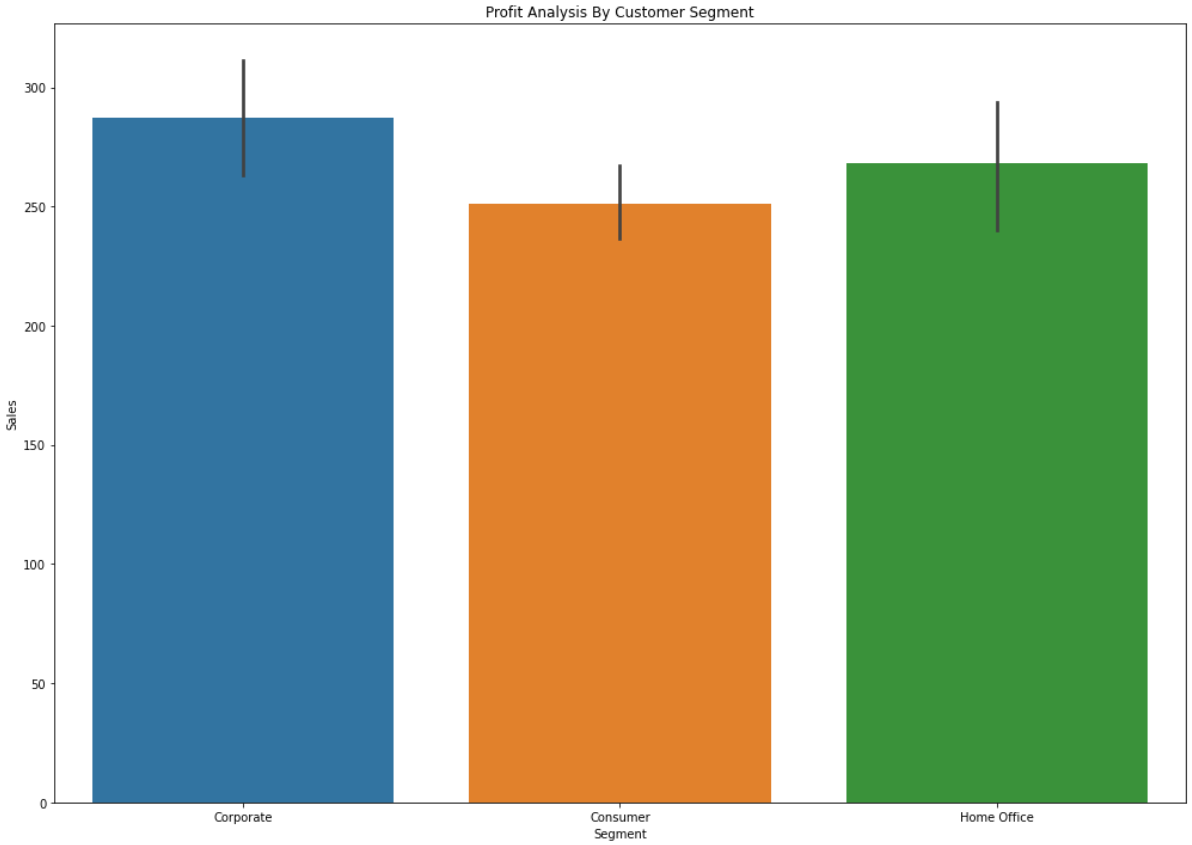


In [23]:
```
Profit_by_sub_category = df.groupby('Sub-Category')['Profit'].sum().reset_index
Profit_by_sub_category
```

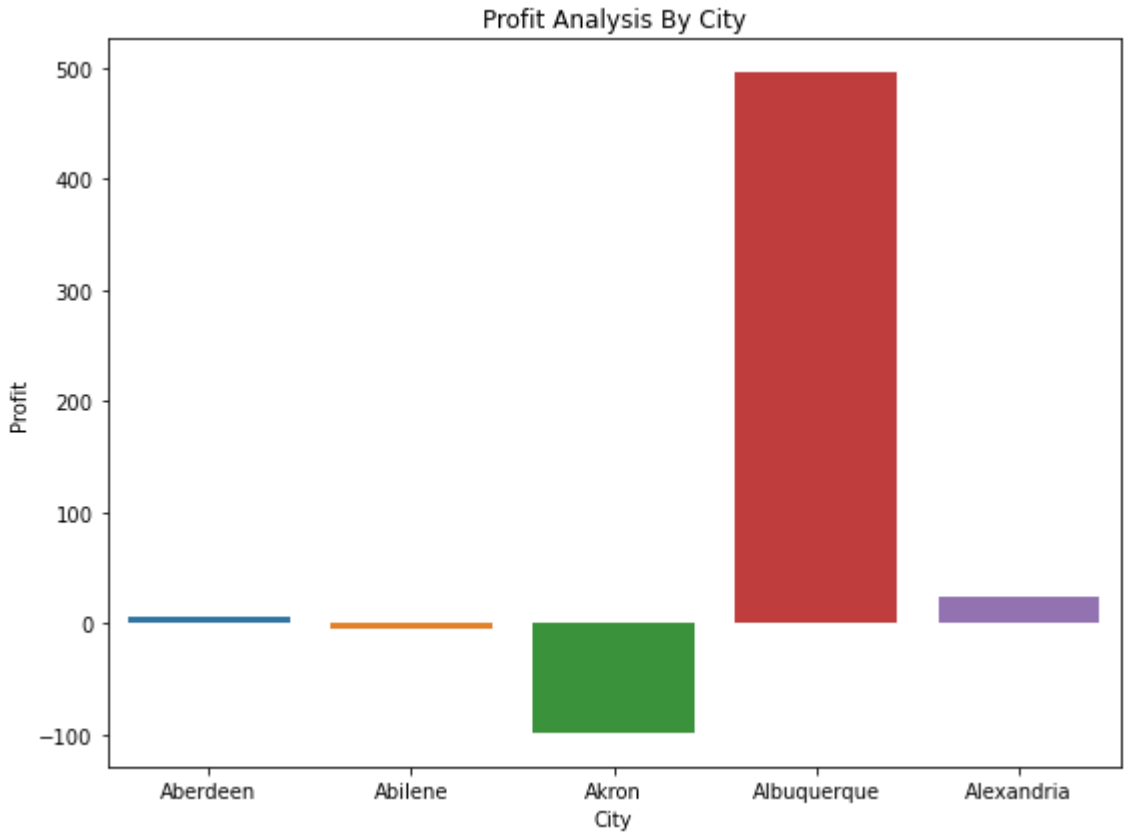
Out[23]:

	Sub-Category	Profit
0	Accessories	25336.6455
1	Appliances	13166.6098
2	Art	3635.9257
3	Binders	17885.3759
4	Bookcases	-342.8883
5	Chairs	13406.7032
6	Copiers	42774.5828
7	Envelopes	3508.5073
8	Fasteners	598.4175
9	Furnishings	8034.4328
10	Labels	2937.2212
11	Machines	38.1024
12	Paper	21112.3779
13	Phones	22308.9179
14	Storage	13607.0875
15	Supplies	-1654.2767
16	Tables	-11091.6365

In [24]:
```
# Create a scatter plot to visualize the correlation
plt.figure(figsize=(14, 10))
sns.barplot(data=df , x='Segment', y ='Sales')
plt.title('Profit Analysis By Customer Segment')
plt.xlabel('Segment')
plt.ylabel('Sales')
plt.tight_layout()
plt.show()
```



In [27]:
```
Profit_by_city = df.groupby('City')['Profit'].sum().reset_index().head()
Profit_by_city
# Create a scatter plot to visualize the correlation
plt.figure(figsize=(8, 6))
sns.barplot(data=Profit_by_city , x='City', y ='Profit')
plt.title('Profit Analysis By City')
plt.xlabel('City')
plt.ylabel('Profit')
plt.tight_layout()
plt.show()
```



In [ ]: