SALES ANALYSIS

The aim of is to Analyse the sales and its various trends using the data from the given dataset.The dataset has been taken from Kaggle.

OBJECTIVE

Upon initial inspection of the data, we can start thinking of some questions about it that we would want to answer :

> **What is the overall sales trend?**

> **Which are the Top 10 products by sales?**

> **Which is the most preferred Ship Mode?**

> **Which are the Most Profitable Category and Sub-Category?**

IMPORTING THE REQUIRED LIBRARIES

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns
```

IMPORTING THE DATASET

```
df = pd.read_excel('/content/drive/MyDrive/datasets/superstore_sales.xlsx')
```

DATA PROCESSING

```
# First five rows of the dataset
df.head()
```

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state
51285	CA-2014-115427	2014-12-31	2015-01-04	Standard Class	Erica Bern	Corporate	California
51286	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	Liz Preis	Consumer	Souss-Massa-Draâ
51287	MX-2014-110527	2014-12-31	2015-01-02	Second Class	Charlotte Melton	Consumer	Managua
51288	MX-2014-114783	2014-12-31	2015-01-06	Standard Class	Tamara Dahlen	Consumer	Chihuahua
51289	CA-2014-156720	2014-12-31	2015-01-04	Standard Class	Jill Matthias	Consumer	Colorado

5 rows × 21 columns

```
# Last five rows of the dataset
df.tail()
```

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state
51285	CA-2014-115427	2014-12-31	2015-01-04	Standard Class	Erica Bern	Corporate	California
51286	MO-2014-2560	2014-12-31	2015-01-05	Standard Class	Liz Preis	Consumer	Souss-Massa-Draâ
51287	MX-2014-110527	2014-12-31	2015-01-02	Second Class	Charlotte Melton	Consumer	Managua
51288	MX-2014-114783	2014-12-31	2015-01-06	Standard Class	Tamara Dahlen	Consumer	Chihuahua
51289	CA-2014-156720	2014-12-31	2015-01-04	Standard Class	Jill Matthias	Consumer	Colorado

5 rows × 21 columns

```
# Shape of the dataset
df.shape
```

```
(51290, 21)
```

```
# Columns present in the dataset
df.columns
```

```
Index(['order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_name',
       'segment', 'state', 'country', 'market', 'region', 'product_id',
       'category', 'sub_category', 'product_name', 'sales', 'quantity',
       'discount', 'profit', 'shipping_cost', 'order_priority', 'year'],
      dtype='object')
```

```
#Checking the data types of the columns
df.dtypes
```

	0
order_id	object
order_date	datetime64[ns]
ship_date	datetime64[ns]
ship_mode	object
customer_name	object
segment	object
state	object
country	object
market	object
region	object
product_id	object
category	object
sub_category	object
product_name	object
sales	float64
quantity	int64
discount	float64
profit	float64
shipping_cost	float64
order_priority	object
year	int64

dtype: object

```
# A concise summary of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        51290 non-null  object
 1   order_date      51290 non-null  datetime64[ns]
 2   ship_date       51290 non-null  datetime64[ns]
 3   ship_mode       51290 non-null  object
 4   customer_name   51290 non-null  object
 5   segment         51290 non-null  object
 6   state           51290 non-null  object
 7   country         51290 non-null  object
 8   market          51290 non-null  object
 9   region          51290 non-null  object
 10  product_id      51290 non-null  object
 11  category        51290 non-null  object
 12  sub_category    51290 non-null  object
 13  product_name    51290 non-null  object
 14  sales           51290 non-null  float64
 15  quantity        51290 non-null  int64
 16  discount        51290 non-null  float64
 17  profit          51290 non-null  float64
 18  shipping_cost   51290 non-null  float64
 19  order_priority  51290 non-null  object
 20  year            51290 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(2), object(13)
memory usage: 8.2+ MB
```

```
# Checking missing values
df.isna().sum()
```

	0
order_id	0
order_date	0
ship_date	0
ship_mode	0
customer_name	0
segment	0
state	0
country	0
market	0
region	0
product_id	0
category	0
sub_category	0
product_name	0
sales	0
quantity	0
discount	0
profit	0
shipping_cost	0
order_priority	0
year	0

dtype: int64

We’re lucky we have such a nice data set and with no missing values.

Next, we can look at some descriptive statistics of the data frame with the describe method.

This shows some descriptive statistics on the data set. Notice, it only shows the statistics on the numerical columns.

```
# Generating descriptive statistics summary
df.describe().round()
```

	order_date	ship_date	sales	quantity	discount	profit	ship
count	51290	51290	51290.0	51290.0	51290.0	51290.0	
mean	2013-05-11 21:26:49.155780864	2013-05-15 20:42:42.745174528	246.0	3.0	0.0	29.0	
min	2011-01-01 00:00:00	2011-01-03 00:00:00	0.0	1.0	0.0	-6600.0	
25%	2012-06-19 00:00:00	2012-06-23 00:00:00	31.0	2.0	0.0	0.0	
50%	2013-07-08 00:00:00	2013-07-12 00:00:00	85.0	3.0	0.0	9.0	

EXPLORATORY DATA ANALYSIS

- WHAT IS THE OVERALL SALES TREND?

```
# Getting month year from order_date
df['month_year'] = df['order_date'].apply(lambda x: x.strftime('%Y-%m'))
print(type("month_year"))
```

```
<class 'str'>
```

```
# Grouping month_year by sales
df_temp = df.groupby('month_year', as_index=False)['sales'].sum()

# Check the resulting DataFrame
print(df_temp)
```

```
   month_year         sales
0     2011-01   98898.48886
1     2011-02   91152.15698
2     2011-03  145729.36736
3     2011-04  116915.76418
4     2011-05  146747.83610
5     2011-06  215207.38022
6     2011-07  115510.41912
7     2011-08  207581.49122
8     2011-09  290214.45534
9     2011-10  199071.26404
10    2011-11  298496.53752
11    2011-12  333925.73460
12    2012-01  135780.72024
13    2012-02  100510.21698
14    2012-03  163076.77116
15    2012-04  161052.26952
16    2012-05  208364.89124
17    2012-06  256175.69842
```

```
18    2012-07  145236.78512
19    2012-08  303142.94238
20    2012-09  289389.16564
21    2012-10  252939.85020
22    2012-11  323512.41690
23    2012-12  338256.96660
24    2013-01  199185.90738
25    2013-02  167239.65040
26    2013-03  198594.03012
27    2013-04  177821.31684
28    2013-05  260498.56470
29    2013-06  396519.61190
30    2013-07  229928.95200
31    2013-08  326488.78936
32    2013-09  376619.24568
33    2013-10  293406.64288
34    2013-11  373989.36010
35    2013-12  405454.37802
36    2014-01  241268.55566
37    2014-02  184837.35556
38    2014-03  263100.77262
39    2014-04  242771.86130
40    2014-05  288401.04614
41    2014-06  401814.06310
42    2014-07  258705.68048
43    2014-08  456619.94236
44    2014-09  481157.24370
45    2014-10  422766.62916
46    2014-11  555279.02700
47    2014-12  503143.69348
```

```
# Setting the figure size
plt.figure(figsize=(16, 5))
plt.plot(df_temp['month_year'], df_temp['sales'], color='#b80045')
plt.xticks(rotation='vertical', size=8)
plt.show()
```

- WHICH ARE THE TOP 10 PRODUCTS BY SALES?

```
# Grouping products by sales, summing only the 'sales' column
prod_sales = df.groupby('product_name', as_index=False).agg({'sales': 'sum'})

# Sorting the dataframe in descending order
prod_sales.sort_values(by='sales', ascending=False, inplace=True)

# Top 10 products by sales
top_10_products = prod_sales.head(10)
print(top_10_products)
```

```
                                         product_name       sales
310                       Apple Smart Phone, Full Size  86935.7786
970                       Cisco Smart Phone, Full Size  76441.5306
2415                    Motorola Smart Phone, Full Size  73156.3030
2501                       Nokia Smart Phone, Full Size  71904.5555
866                Canon imageCLASS 2200 Advanced Copier  61599.8240
1837         Hon Executive Leather Armchair, Adjustable  58193.4841
2631  Office Star Executive Leather Armchair, Adjust...  50661.6840
1714  Harbour Creations Executive Leather Armchair, ...  50121.5160
2988                       Samsung Smart Phone, Cordless  48653.4600
2502                  Nokia Smart Phone, with Caller ID  47877.7857
```

```
# Creating a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='sales', y='product_name', data=top_10_products, palette='viridis')

# Adding titles and labels
plt.title('Top 10 Products by Sales', fontsize=16)
```

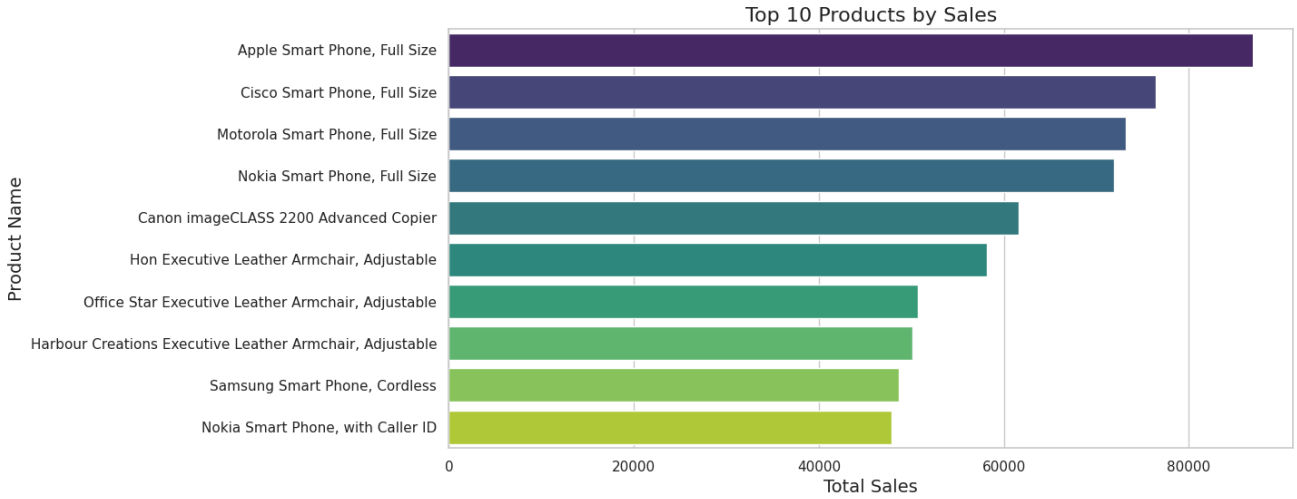
```
plt.xlabel('Total Sales', fontsize=14)
plt.ylabel('Product Name', fontsize=14)

plt.show()
```

```
<ipython-input-43-b999182d866b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x='sales', y='product_name', data=top_10_products, palette='viridis')
```

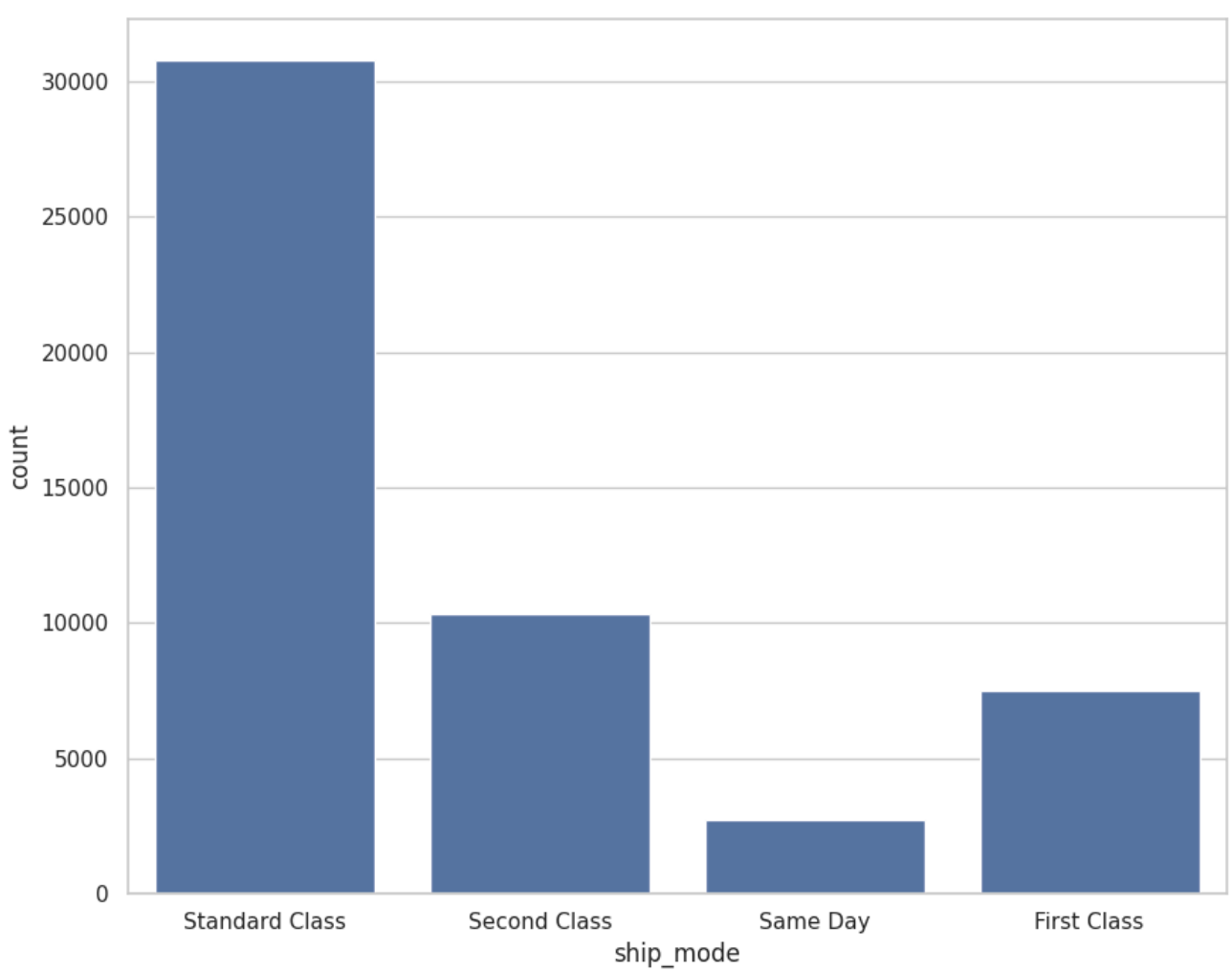


- WHAT IS THE MOST PREFERRED SHIP MODE?

```
# Setting the figure size
plt.figure(figsize=(10, 8))

# countplot: Shows the counts of observations in each categorical bin using bars
sns.countplot(x='ship_mode', data=df)

plt.show()
```



- WHICH ARE THE MOST PROFITABLE CATEGORY AND SUB-CATEGORY?

```
# Grouping products by Category and Sub-Category
cat_subcat = df.groupby(['category', 'sub_category'])['profit'].sum().reset_index()

# Sorting the values
cat_subcat.sort_values(by=['category', 'profit'], ascending=[True, False], inplace=True)

print(cat_subcat)
```

```
            category sub_category        profit
    0         Furniture    Bookcases  161924.41950
    1         Furniture       Chairs  141973.79750
    2         Furniture  Furnishings   46967.42550
    3         Furniture       Tables  -64083.38870
    4   Office Supplies   Appliances  141680.58940
    11  Office Supplies      Storage  108461.48980
    6   Office Supplies      Binders   72449.84600
    10  Office Supplies        Paper   59207.68270
    5   Office Supplies          Art   57953.91090
```