PANDAS SALES ANALYSIS

OBJECTIVE

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

1. What is the overall sales trend?
2. What are the top 10 products by sales?
3. What are the most selling products?
4. Which is the most preferred ship Mode?
5. Which are the most profitable category and sub-category?

IMPORTING REQUIRED LIBRARIES

In [82]:
```
# pip install openpyxl
```

In [83]:
```
# Data Manupulation
import pandas as pd

# Data Visualisation
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

- ### IMPORTING THE DATASET

In [84]:
```
df = pd.read_excel('superstore_sales.xlsx')
```

DATA AUDIT

In [85]:
```
# First five rows of the dataset
df.head()
```

Out[85]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria
1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australia
2	HU-2011-1220	2011-01-01	2011-01-05	Second Class	Annie Thurman	Consumer	Budapest	Hungary
3	IT-2011-3647632	2011-01-01	2011-01-05	Second Class	Eugene Moren	Home Office	Stockholm	Sweden
4	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australia

5 rows × 21 columns

In [86]:
```
# Last five rows of the dataset
df.tail()
```

Out[86]:

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

5 rows × 21 columns

In [87]:
```
# Shape of the dataset
df.shape
```

Out[87]:
```
(51290, 21)
```

In [88]:
```
# Columns present in the dataset
df.columns
```

Out[88]:
```
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')
```

In [89]:
```
# 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.isnull().sum()

In [90]:
```
# Getting descriptive statistics summary
df.describe()
```

Out[90]:

	sales	quantity	discount	profit	shipping_cost	year
count	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000	51290.000000
mean	246.490581	3.476545	0.142908	28.641740	26.375818	2012.777208
std	487.565361	2.278766	0.212280	174.424113	57.296810	1.098931
min	0.444000	1.000000	0.000000	-6599.978000	0.002000	2011.000000
25%	30.758625	2.000000	0.000000	0.000000	2.610000	2012.000000
50%	85.053000	3.000000	0.000000	9.240000	7.790000	2013.000000
75%	251.053200	5.000000	0.200000	36.810000	24.450000	2014.000000
max	22638.480000	14.000000	0.850000	8399.976000	933.570000	2014.000000

EXPLORATORY DATA ANALYSIS

- ### 1. WHAT IS THE OVERALL SALES TREND?

In [91]:
```
df['order_date'].min()
```
Out[91]:
```
Timestamp('2011-01-01 00:00:00')
```

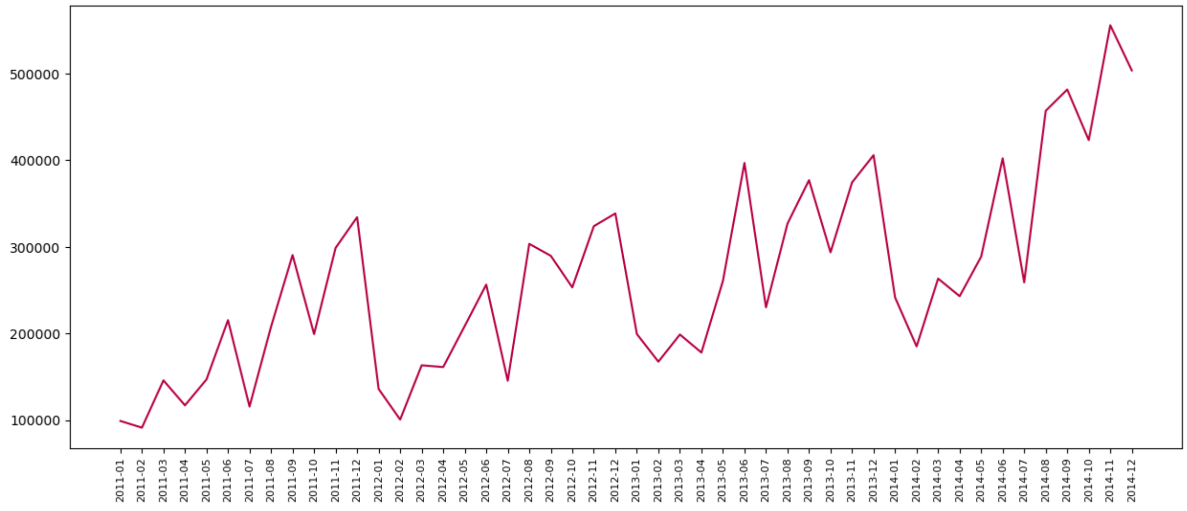
In [92]:
```
df['order_date'].max()
```
Out[92]:
```
Timestamp('2014-12-31 00:00:00')
```

In [93]:
```
# Getting month year from the dataset
df['month_year'] = df['order_date'].apply(lambda x: x.strftime('%Y-%m'))
```

In [94]:
```
df['month_year']
```
Out[94]:
```
0        2011-01
1        2011-01
2        2011-01
3        2011-01
4        2011-01
          ...
51285    2014-12
51286    2014-12
51287    2014-12
51288    2014-12
51289    2014-12
Name: month_year, Length: 51290, dtype: object
```

In [95]:
```
# Grouping month year
df_trend = df.groupby('month_year').sum()['sales'].reset_index()
```
```
C:\Users\SHANTANU GARAIN\AppData\Local\Temp\ipykernel_10776\43846726.py:2: FutureW
arning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. I
n a future version, numeric_only will default to False. Either specify numeric_onl
y or select only columns which should be valid for the function.
  df_trend = df.groupby('month_year').sum()['sales'].reset_index()
```

In [96]:
```
# Setting the figure size
plt.figure(figsize=(15,6))
plt.plot(df_trend['month_year'], df_trend['sales'], color='#b80045')
plt.xticks(rotation='vertical', size=8)
plt.show()
```



- 2. WHICH ARE THE TOP 10 PRODUCTS BY SALES?

In [97]:
```
# Grouping product name column
prod_sales = pd.DataFrame(df.groupby('product_name').sum()['sales'])
```

```
C:\Users\SHANTANU GARAIN\AppData\Local\Temp\ipykernel_10776\2716989281.py:2: Futur
eWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numeric_on
ly or select only columns which should be valid for the function.
  prod_sales = pd.DataFrame(df.groupby('product_name').sum()['sales'])
```

In [98]:
```
prod_sales = prod_sales.sort_values('sales', ascending=False)
```

In [99]:
```
# Top 10 product by sales
prod_sales[:10]
```

Out[99]:

	sales
product_name	
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306
Motorola Smart Phone, Full Size	73156.3030
Nokia Smart Phone, Full Size	71904.5555
Canon imageCLASS 2200 Advanced Copier	61599.8240
Hon Executive Leather Armchair, Adjustable	58193.4841
Office Star Executive Leather Armchair, Adjustable	50661.6840
Harbour Creations Executive Leather Armchair, Adjustable	50121.5160
Samsung Smart Phone, Cordless	48653.4600
Nokia Smart Phone, with Caller ID	47877.7857

- 3. WHICH ARE THE MOST SELLING PRODUCTS

In [100…
```
#Grouping product name
most_sell_prod = pd.DataFrame(df.groupby('product_name').sum()['quantity'])
```

```
C:\Users\SHANTANU GARAIN\AppData\Local\Temp\ipykernel_10776\2293239909.py:2: Futur
eWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numeric_on
ly or select only columns which should be valid for the function.
  most_sell_prod = pd.DataFrame(df.groupby('product_name').sum()['quantity'])
```

In [101…

```
# Sorting most_sell_product
most_sell_prod = most_sell_prod.sort_values('quantity', ascending=False)
```

In [102…

```
most_sell_prod[:10]
```

Out[102]:

	quantity
product_name	
Staples	876
Cardinal Index Tab, Clear	337
Eldon File Cart, Single Width	321
Rogers File Cart, Single Width	262
Sanford Pencil Sharpener, Water Color	259
Stockwell Paper Clips, Assorted Sizes	253
Avery Index Tab, Clear	252
Ibico Index Tab, Clear	251
Smead File Cart, Single Width	250
Stanley Pencil Sharpener, Water Color	242

- ## 4. WHAT IS THE MOST PREFERED SHIP MODE?

In [103…

```
# Setting figure size
plt.figure(figsize=(10,8.5))

# Plotting shipmode
sns.countplot(df['ship_mode'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[103], line 5
      2 plt.figure(figsize=(10,8.5))
      4 # Plotting shipmode
----> 5 sns.countplot(df['ship_mode'])

File ~\.conda\envs\CampusX\lib\site-packages\seaborn\categorical.py:2943, in count
plot(data, x, y, hue, order, hue_order, orient, color, palette, saturation, width,
dodge, ax, **kwargs)
   2940 elif x is not None and y is not None:
   2941     raise ValueError("Cannot pass values for both `x` and `y`")
-> 2943 plotter = _CountPlotter(
   2944     x, y, hue, data, order, hue_order,
   2945     estimator, errorbar, n_boot, units, seed,
   2946     orient, color, palette, saturation,
   2947     width, errcolor, errwidth, capsize, dodge
   2948 )
   2950 plotter.value_label = "count"
   2952 if ax is None:

File ~\.conda\envs\CampusX\lib\site-packages\seaborn\categorical.py:1530, in _BarP
lotter.__init__(self, x, y, hue, data, order, hue_order, estimator, errorbar, n_bo
ot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, ca
psize, dodge)
   1525 def __init__(self, x, y, hue, data, order, hue_order,
   1526              estimator, errorbar, n_boot, units, seed,
   1527              orient, color, palette, saturation, width,
   1528              errcolor, errwidth, capsize, dodge):
   1529     """Initialize the plotter."""
-> 1530     self.establish_variables(x, y, hue, data, orient,
   1531                              order, hue_order, units)
   1532     self.establish_colors(color, palette, saturation)
   1533     self.estimate_statistic(estimator, errorbar, n_boot, seed)

File ~\.conda\envs\CampusX\lib\site-packages\seaborn\categorical.py:516, in _Categ
oricalPlotter.establish_variables(self, x, y, hue, data, orient, order, hue_order,
units)
    513     plot_data = data
    515 # Convert to a list of arrays, the common representation
--> 516 plot_data = [np.asarray(d, float) for d in plot_data]
    518 # The group names will just be numeric indices
    519 group_names = list(range(len(plot_data)))

File ~\.conda\envs\CampusX\lib\site-packages\seaborn\categorical.py:516, in <listc
omp>(.0)
    513     plot_data = data
    515 # Convert to a list of arrays, the common representation
--> 516 plot_data = [np.asarray(d, float) for d in plot_data]
    518 # The group names will just be numeric indices
    519 group_names = list(range(len(plot_data)))

File ~\.conda\envs\CampusX\lib\site-packages\pandas\core\series.py:893, in Series.
__array__(self, dtype)
    846 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
    847     """
    848     Return the values as a NumPy array.
    849
   (...)
    891           dtype='datetime64[ns]')
    892     """
--> 893     return np.asarray(self._values, dtype)

ValueError: could not convert string to float: 'Standard Class'
```

```
<Figure size 1000x850 with 0 Axes>
```

- **5. WHICH ARE THE MOST PROFITABLE CATEGORY AND SUB-CATEGORY?**

In [108…

```
# Grouping category and subcategory
cat_subcat_profit = pd.DataFrame(df.groupby(['category', 'sub_category']).sum()['pr
```

```
C:\Users\SHANTANU GARAIN\AppData\Local\Temp\ipykernel_10776\861961321.py:2: Future
Warning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated.
In a future version, numeric_only will default to False. Either specify numeric_on
ly or select only columns which should be valid for the function.
  cat_subcat_profit = pd.DataFrame(df.groupby(['category', 'sub_category']).sum()
['profit'])
```

In [109…

```
# Sorting the result
cat_subcat_profit.sort_values(['category', 'profit'], ascending=False)
```

Out[109]:

		profit
category	**sub_category**	
Technology	**Copiers**	258567.54818
	Phones	216717.00580
	Accessories	129626.30620
	Machines	58867.87300
Office Supplies	**Appliances**	141680.58940
	Storage	108461.48980
	Binders	72449.84600
	Paper	59207.68270
	Art	57953.91090
	Envelopes	29601.11630
	Supplies	22583.26310
	Labels	15010.51200
	Fasteners	11525.42410
Furniture	**Bookcases**	161924.41950
	Chairs	141973.79750
	Furnishings	46967.42550
	Tables	-64083.38870

In [ ]: