" SALES ANALYSIS PROJECT "

Author : Shalu Anand

Language : Python

Using : Super Store Dataset

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 are the Most Selling Products?**
- **Which is the most preferred Ship Mode?**
- **Which are the Most Profitable Category and Sub-Category?**

IMPORTING REQUIRED LIBRARIES

In [1]:

```
# Data Manipulation
import pandas as pd

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

import seaborn as sns
```

IMPORTING THE DATASET

In [2]:

```
# Importing dataset
df = pd.read_excel('superstore_sales.xlsx')
```

DATA AUDIT   (FRIENDSHIP WITH DATA WHICH WE HAVE)

**You can't make your data work for you until you know what data you're talking about.
To get a quick idea of what the data looks like, we can call the head function on the data frame. By default, this returns the top five rows, but it can take in a parameter of how many rows to return.
Data auditing is the process of conducting a data audit to assess how company's data is fit for given purpose.
This involves profiling the data and assessing the impact of poor quality data on the organization's performance and profits.**

In [3]:

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

Out[3]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...	ca
0	CA-2012-124891	2012-07-31	2012-07-31	Same Day	Rick Hansen	Consumer	New York	United States	US	East	...	Tech
1	IN-2013-77878	2013-02-05	2013-02-07	Second Class	Justin Ritter	Corporate	New South Wales	Australia	APAC	Oceania	...	Fu
2	IN-2013-71249	2013-10-17	2013-10-18	First Class	Craig Reiter	Consumer	Queensland	Australia	APAC	Oceania	...	Tech
3	ES-2013-1579342	2013-01-28	2013-01-30	First Class	Katherine Murray	Home Office	Berlin	Germany	EU	Central	...	Tech
4	SG-2013-4320	2013-11-05	2013-11-06	Same Day	Rick Hansen	Consumer	Dakar	Senegal	Africa	Africa	...	Tech

5 rows × 21 columns

In [4]:

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

Out[4]:

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...	ca
51285	IN-2014-62366	2014-06-19	2014-06-19	Same Day	Katrina Edelman	Corporate	Hiroshima	Japan	APAC	North Asia	...	S
51286	US-2014-102288	2014-06-20	2014-06-24	Standard Class	Zuschuss Carroll	Consumer	Texas	United States	US	Central	...	S
51287	US-2013-155768	2013-12-02	2013-12-02	Same Day	Laurel Beltran	Home Office	California	United States	US	West	...	S
51288	MX-2012-140767	2012-02-18	2012-02-22	Standard Class	Ross Baird	Home Office	São Paulo	Brazil	LATAM	South	...	S
51289	MX-2012-134460	2012-05-22	2012-05-26	Second Class	Mick Crebagga	Consumer	Managua	Nicaragua	LATAM	Central	...	S

5 rows × 21 columns

In [5]: FROM THESE COMMANDS WE WONT BE ABLE TO FIND EXACTLY HOW MANY ROWS AND COLUMNS ARE THERE, SO FOR THAT WE WILL BE USING SHAPE COMMAND.

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

Out[5]:

```
#(number of rows, number of columns)
(51290, 21)
```

In [6]:

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

Out[6]:

```
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')
```

This looks a lot like an Excel spreadsheet, doesn't it? Under the hood, the data frame is a two-dimensional data structure and each column can have different types. To show that, we can call dtypes attribute on the data frame to see what each column types are.

In [7]:

```
# 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
```

Now we can do further analysis on our data to answer our questions. Before that, we should see if there are any missing values in our data set.To check if there are any missing values in the entire data set we use the isnull function, then see if there are any values.

We're lucky we have such a nice data set and with no missing values. While we won't focus on it in this post, a data scientist will spend their time cleaning (or wrangling) the data. Since we don't have any missing data, we can start doing further analysis on our data.

In [8]:

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

Out[8]:

```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             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
```

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. From here you can see the following statistics:

- **Row count, which aligns to what the shape attribute showed us.**
- **The mean, or average.**
- **The standard deviation, or how spread out the data is.**
- **The minimum and maximum value of each column**
- **The number of items that fall within the first, second, and third percentiles.**

In [9]:

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

Out[9]:

	sales	quantity	discount	profit	shipping_cost	year
count	51290.0	51290.0	51290.0	51290.0	51290.0	51290.0
mean	246.0	3.0	0.0	29.0	26.0	2013.0
std	488.0	2.0	0.0	174.0	57.0	1.0
min	0.0	1.0	0.0	-6600.0	0.0	2011.0
25%	31.0	2.0	0.0	0.0	3.0	2012.0
50%	85.0	3.0	0.0	9.0	8.0	2013.0
75%	251.0	5.0	0.0	37.0	24.0	2014.0
max	22638.0	14.0	1.0	8400.0	934.0	2014.0

EXPLORATORY DATA ANALYSIS

- **WHAT IS THE OVERALL SALES TREND?**

In [10]:

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

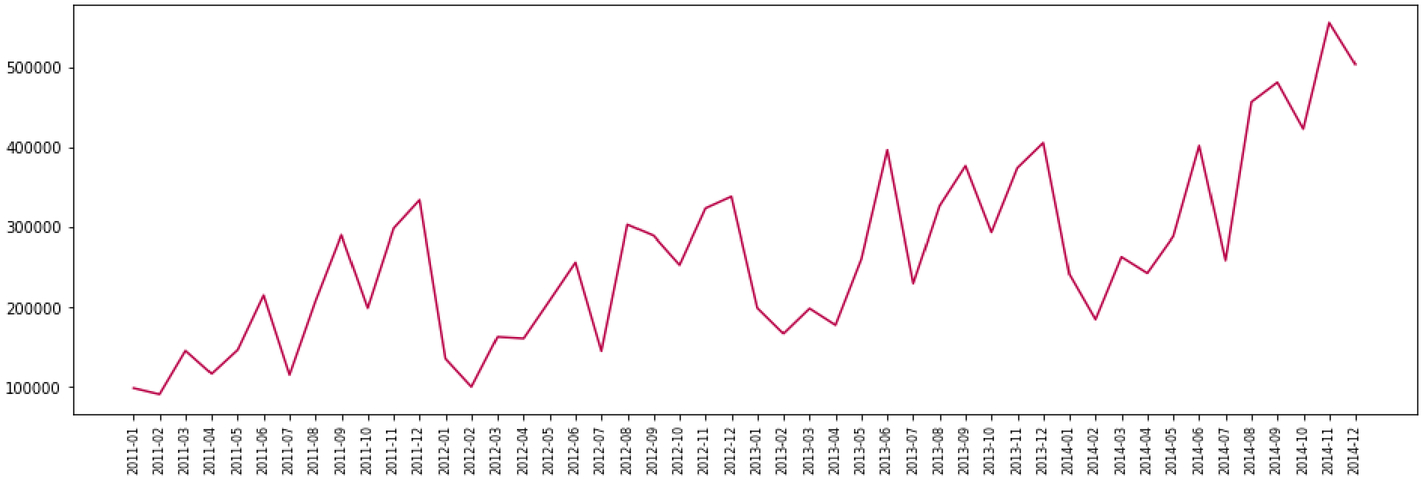
In [11]:

```
# grouping month_year by sales
df_temp = df.groupby('month_year').sum()['sales'].reset_index()
```

In [12]:

In [12]:

```
# Setting the figure size
plt.figure(figsize=(16, 5))
plt.plot(df_temp['month_year'], df_temp['sales'], color='#b80045') #we define x and then y axis
plt.xticks(rotation='vertical', size=8)              #90 degree tilt
plt.show()            #will help you in hiding unnecessary lists
```



- **WHICH ARE THE TOP 10 PRODUCTS BY SALES?**

In [13]:

```
# Grouping products by sales
prod_sales = pd.DataFrame(df.groupby('product_name').sum()['sales'])

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

# Top 10 products by sales
prod_sales[:10]
```

Out[13]:

	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

- **WHICH ARE THE MOST SELLING PRODUCTS?**

In [14]:

```
# Grouping products by Quantity
best_selling_prods = pd.DataFrame(df.groupby('product_name').sum()['quantity'])

# Sorting the dataframe in descending order
```

```
best_selling_prods.sort_values(by=['quantity'], inplace=True, ascending=False)

# Most selling products
best_selling_prods[:10]
```

Out[14]:

	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

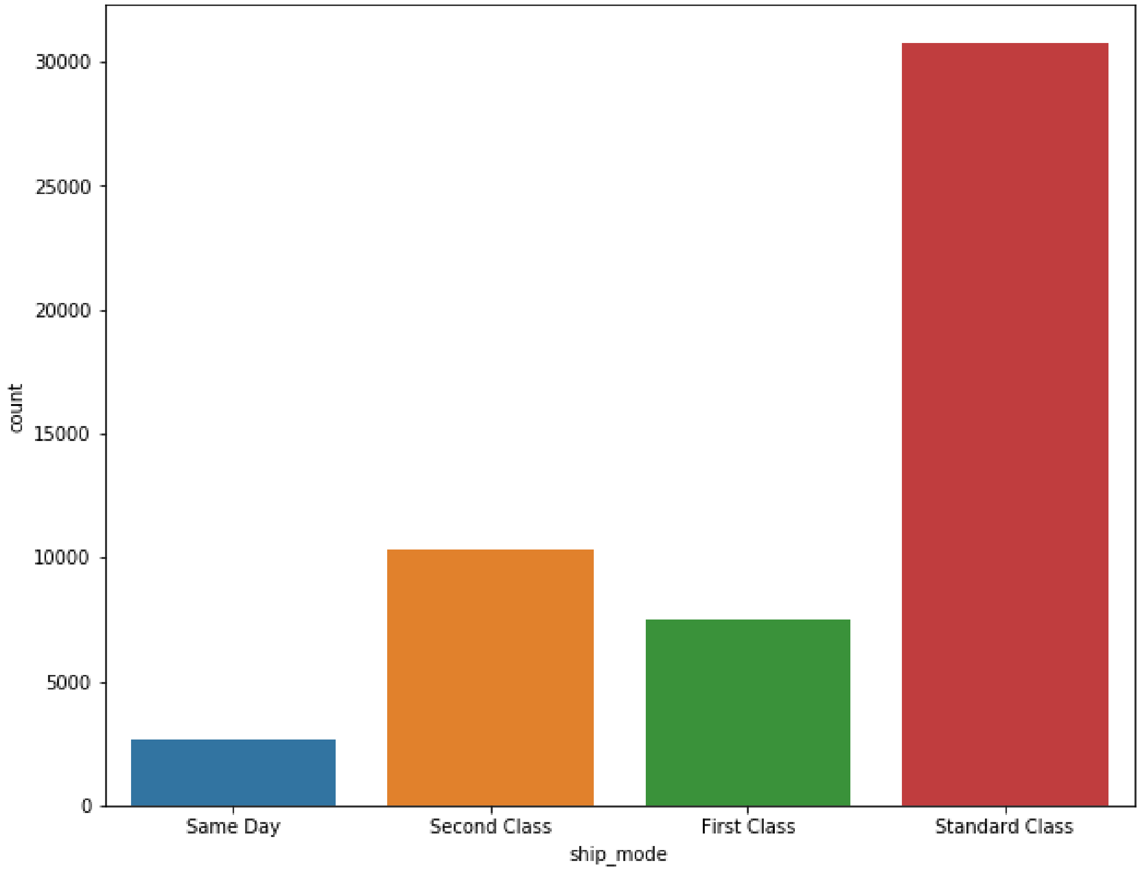
- **WHAT IS THE MOST PREFERRED SHIP MODE?**

In [15]:

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

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

# Display the figure
plt.show()
```



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

In [16]:

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

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

Out[16]:

		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

THANK YOU
END

Author : Shalu Anand
Excel file resource : GOOGLE

LIBRARIES USED : PANDAS , MATPLOTLIB , SEABORN