Superstore Sales

Selecting real world data set

```
!pip install jovian opendatasets --upgrade --quiet
```

```
#importing libaries
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

```
# importing data set

Store_df = pd.read_excel('superstore_sales.xlsx')
```

Data preparation & cleaning

Now that we have the data imported, will prepare the data for further analysis by pre-anaylsing cleaning the data set.

```
#Sample of dataset
Store_df.head()
```

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

5 rows × 21 columns

```
##checking data type
Store_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
```

There is no error in data type following data type

```
## Checking the null value
Store_df.isnull().sum()
```

```
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
```

The data set is clean there is no empty value is found

```
Store_df.describe().round()
```

	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

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "shashi-tron/untitled1" on https://jovian.com/
[jovian] Committed successfully! https://jovian.com/shashi-tron/untitled1
```

```
'https://jovian.com/shashi-tron/untitled1'
```

EXPLORATORY DATA ANALYSIS

- ## WHAT IS THE OVERALL SALES TREND?

Here in exploratory data analysis , lets us understand overall trend of sales

```
Store_df.head(2)
```

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	Africa	...
1	IN-2011-47883	2011-01-01	2011-01-08	Standard Class	Joseph Holt	Consumer	New South Wales	Australia	APAC	Oceania	...

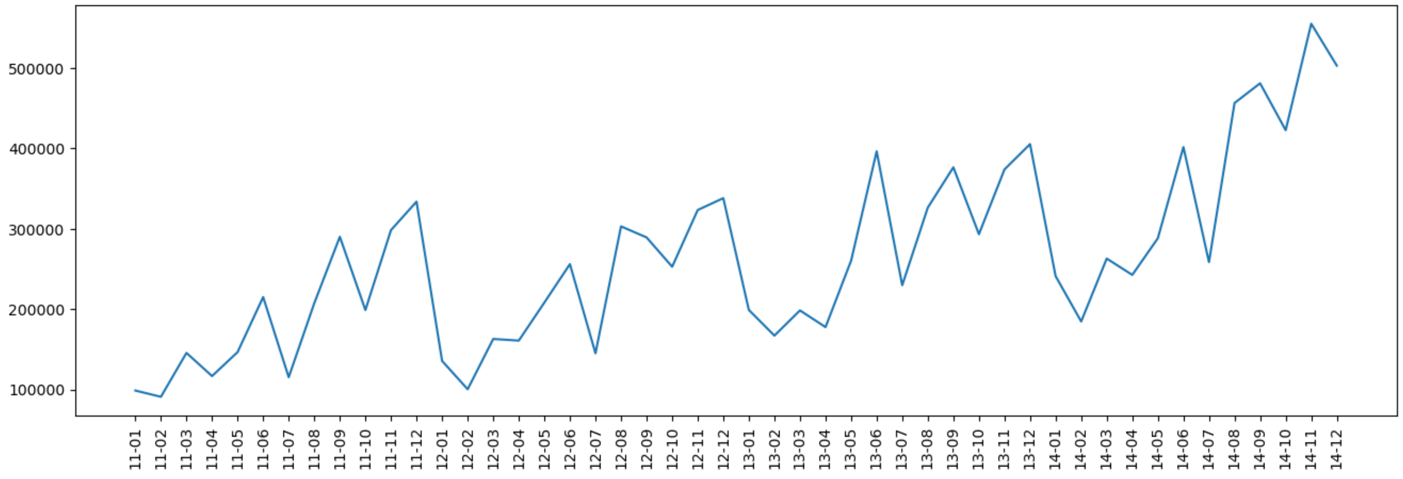
2 rows × 21 columns

```
## first we get month and year from order_date column by creating new column called mor
Store_df['month_year'] = Store_df['order_date'].apply(lambda x:x.strftime('%y-%m'))
```

```
## grouping sales by month
Sales_df = Store_df.groupby('month_year').sum()['sales'].reset_index()
```

```
C:\Users\angdi\AppData\Local\Temp\ipykernel_3020\3137860707.py:2: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  Sales_df = Store_df.groupby('month_year').sum()['sales'].reset_index()
```

```
## visulizing sales trend
plt.figure(figsize=(16,5))
plt.plot(Sales_df['month_year'],Sales_df['sales'])
plt.xticks(rotation = 'vertical',size = 10)
plt.show()
```



The sales visiulaztion show trend of sales with expontinal growth , this shows us sales by store perfoming very good.
as store there is still area of growth of sales which need perfomed well.

insight

- since 2011 - 2014 , month of janury and february there is low sales volume which appears to be cyclic , Store need attractive promotion and Discounts which helps increases sale volume.
- every year in month of july there is drastic drop in sales , due uncertain reason which need be noted and perfome requried action.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "shashi-tron/untitled1" on https://jovian.com/
[jovian] Committed successfully! https://jovian.com/shashi-tron/untitled1
```

```
'https://jovian.com/shashi-tron/untitled1'
```

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

- 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?

1. What are the top 10 products by sales ?

```
Store_df.head(1)
```

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...	s
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	Africa	...	

1 rows × 22 columns

```
#grouping product by sales
product_sales =pd.DataFrame(Store_df.groupby('product_name').sum()['sales'])
```

```
C:\Users\angdi\AppData\Local\Temp\ipykernel_3020\3721149701.py:2: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  product_sales =pd.DataFrame(Store_df.groupby('product_name').sum()['sales'])
```

```
#Sorting data frame ascending order
product_sales.sort_values(by=['sales'],inplace=True,ascending=False)
```

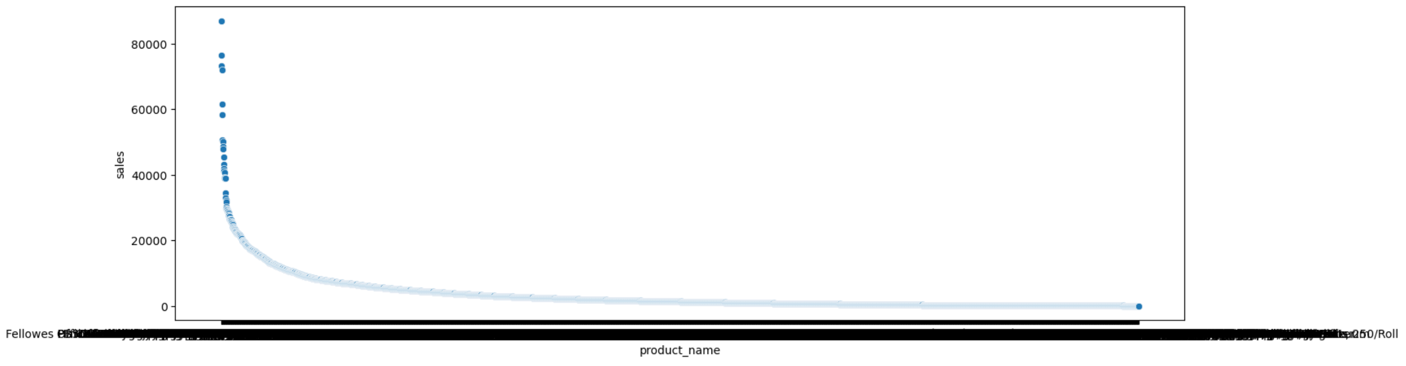
```
#top 10 product sales
product_sales.head(10)
```

	sales
product_name	
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306

	sales
product_name	
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

```
#visulization top 10 products
plt.figure(figsize=(16,5))
sns.scatterplot(x='product_name', y='sales', data=product_sales)
```

```
<Axes: xlabel='product_name', ylabel='sales'>
```



```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "shashi-tron/untitled1" on https://jovian.com/
[jovian] Committed successfully! https://jovian.com/shashi-tron/untitled1
```

```
'https://jovian.com/shashi-tron/untitled1'
```

2.Which are the Most Selling Products?

```
Store_df.head(1)
```

	order_id	order_date	ship_date	ship_mode	customer_name	segment	state	country	market	region	...	s
0	AG-2011-2040	2011-01-01	2011-01-06	Standard Class	Toby Braunhardt	Consumer	Constantine	Algeria	Africa	Africa	...	

1 rows × 22 columns

```
# Grouping products by Quantity
best_selling_prods = pd.DataFrame(Store_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.head(10)
```

```
C:\Users\angdi\AppData\Local\Temp\ipykernel_3020\3778220150.py:2: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  best_selling_prods = pd.DataFrame(Store_df.groupby('product_name').sum()['quantity'])
```

	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

```
Store_df.category.unique()
```

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

```
  - here most selling products in terms of qunatity comes under category are 'office su
```

3.Which is the most preferred Ship Mode?

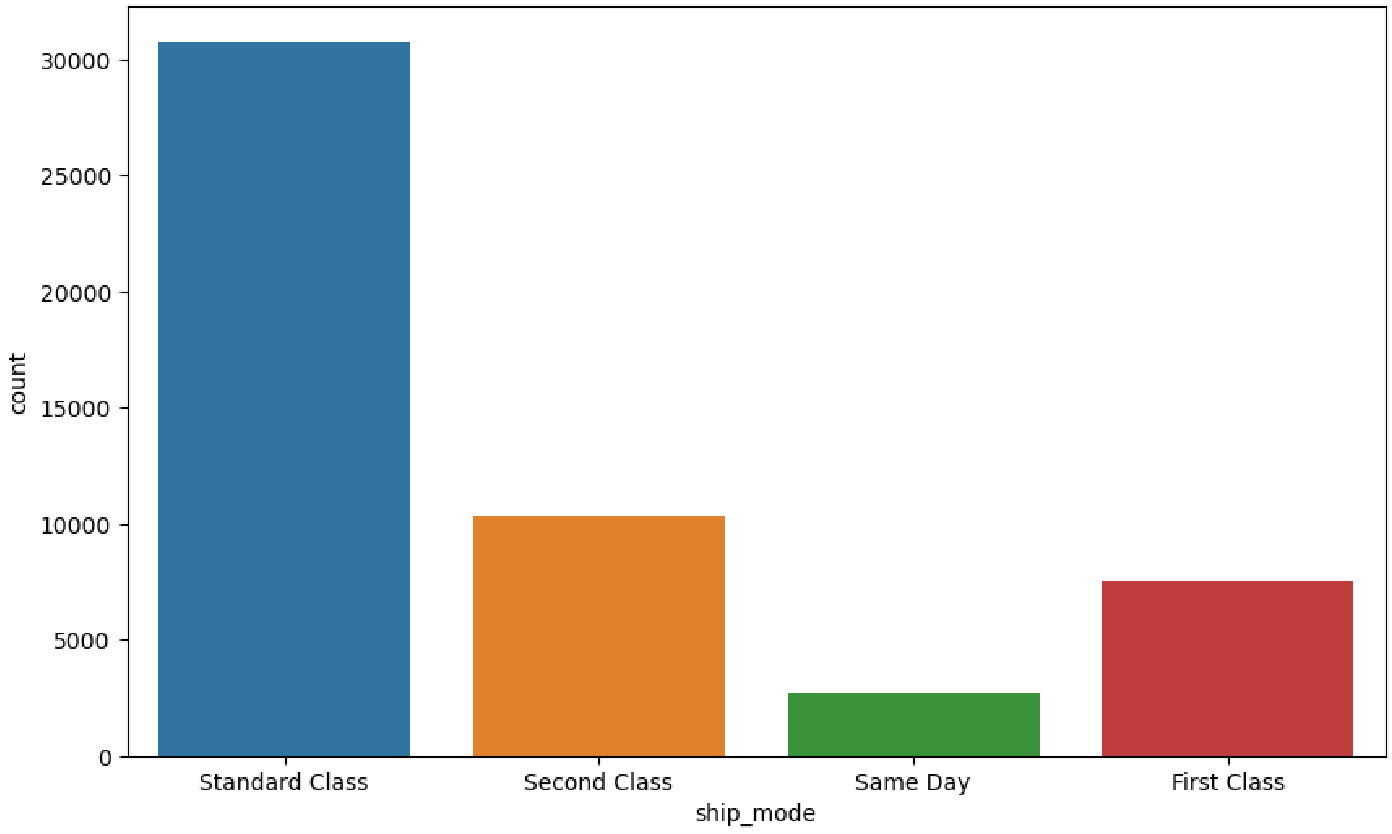
```
#Finding type of shiping mode
Store_df.ship_mode.unique()
```

```
array(['Standard Class', 'Second Class', 'Same Day', 'First Class'],
      dtype=object)
```

```
#lets visulize the most preferred ship mode

plt.figure(figsize=(10,6))
```

```
sns.countplot(x='ship_mode',data=Store_df)
plt.show()
```



-- There is clear majority shows 'Standard class' is prefferd

4.Which are the Most Profitable Category and Sub-Category?

```
#lets group category and sub category first
cat_subcat = pd.DataFrame(Store_df.groupby(['category', 'sub_category']).sum()['profit'
```

```
C:\Users\angdi\AppData\Local\Temp\ipykernel_3020\2702117022.py:2: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  cat_subcat = pd.DataFrame(Store_df.groupby(['category', 'sub_category']).sum()
['profit'])
```

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

		profit
category	**sub_category**	
Technology	**Copiers**	258567.54818
	Phones	216717.00580
	Accessories	129626.30620
	Machines	58867.87300

		profit
category	**sub_category**	
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

```python
import jovian
```

```python
jovian.commit()
```