```
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
%matplotlib inline
```

```
superstore_df = pd.read_csv('supermarket_sales - Sheet1.csv')
superstore_df
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085
...	...	...	...	...	...	...	...	...	..
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190

```
# Check the number of rows and columns
superstore_df.shape
```

```
(1000, 17)
```

```
# check the size
superstore_df.size
```

```
17000
```

```
# To check the number of missing values in each column
superstore_df.isnull().sum()
```

```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

```
superstore_df.columns
```

```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
```

```
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
       'Rating'],
      dtype='object')
```

```
superstore_df.describe()
```

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905	49.650000	10.00000

```
# check the info of the dataset
superstore_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

```
# copy the data
updated_superstore_df = superstore_df.copy()
```

```
updated_superstore_df
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Ti
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:

```
# check if there is any duplicates
updated_superstore_df[updated_superstore_df.duplicated()]
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross

```
# covert the datatypes of both columns to datatime
updated_superstore_df['Date'] = pd.to_datetime(updated_superstore_df.Date)
updated_superstore_df['Time']=pd.to_datetime(updated_superstore_df['Time'], format='%H:%M')

### Parse  columns
updated_superstore_df["Month"] = pd.DatetimeIndex(updated_superstore_df.Date).month_name()
updated_superstore_df["Year"] = pd.DatetimeIndex(updated_superstore_df.Date).year
updated_superstore_df["weekday"] = pd.DatetimeIndex(updated_superstore_df.Date).day_name()
updated_superstore_df["Day"] = pd.DatetimeIndex(updated_superstore_df.Date).day
updated_superstore_df["week"] = updated_superstore_df['Date'].dt.isocalendar().week
updated_superstore_df['Hour']= updated_superstore_df['Time'].dt.hour
```

```
updated_superstore_df
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	cogs	g
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	522.83	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	...	76.40	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	...	324.31	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	465.76	
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	...	604.17	
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	...	40.35	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	...	973.80	
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	...	31.84	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	...	65.82	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	...	618.38	

1000 rows × 23 columns

```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size']= 14
matplotlib.rcParams['figure.figsize']=(10, 5)
matplotlib.rcParams['figure.facecolor']= '#00000000'
```

```
updated_superstore_df.head(5)
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	cogs	gros
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	522.83	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	...	76.40	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	...	324.31	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	465.76	

```
updated_superstore_df.tail(5)
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	cogs	gr
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	...	40.35	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	...	973.80	
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	...	31.84	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	...	65.82	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	...	618.38	

5 rows × 23 columns

```
updated_superstore_df.cogs.mean()
```

```
307.58738
```

```
expensive_price =updated_superstore_df['Unit price'] == updated_superstore_df['Unit price'].max()
expensive_price
```

```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Name: Unit price, Length: 1000, dtype: bool
```

```
expensive_product = updated_superstore_df[["Product line","Unit price"]][expensive_price]
expensive_product
```

index	Product line
122	Sports and travel
983	Health and beauty

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Categorical distributions

```
health_and_beauty_details = updated_superstore_df[updated_superstore_df["Product line"] == "Health and beauty"]
health_and_beauty_details
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	cogs	gross
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	522.83	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	465.76	
8	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.6260	76.1460	...	72.52	
14	829-34-3910	A	Yangon	Normal	Female	Health and beauty	71.38	10	35.6900	749.4900	...	713.80	
16	656-95-9349	A	Yangon	Member	Female	Health and beauty	68.93	7	24.1255	506.6355	...	482.51	
...	...	...	...	...	...	...	...	...	...	...	...	...	
983	148-41-7930	C	Naypyitaw	Normal	Male	Health and beauty	99.96	7	34.9860	734.7060	...	699.72	
986	764-44-8999	B	Mandalay	Normal	Female	Health and beauty	14.76	2	1.4760	30.9960	...	29.52	
987	552-44-5977	B	Mandalay	Member	Male	Health and beauty	62.00	8	24.8000	520.8000	...	496.00	
989	430-53-4718	B	Mandalay	Member	Male	Health and beauty	75.37	8	30.1480	633.1080	...	602.96	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	...	40.35	

152 rows × 23 columns

```
expensive_purchases = updated_superstore_df.groupby("Product line")[["Total"]].max().sort_values("Total", ascending=False)
expensive_purchases
```

1 to 6 of 6 entries Filter

Product line	Total
Fashion accessories	1042.65
Food and beverages	1034.46
Home and lifestyle	1023.75
Sports and travel	1002.12
Health and beauty	950.25
Electronic accessories	942.4485

Show 25 per page

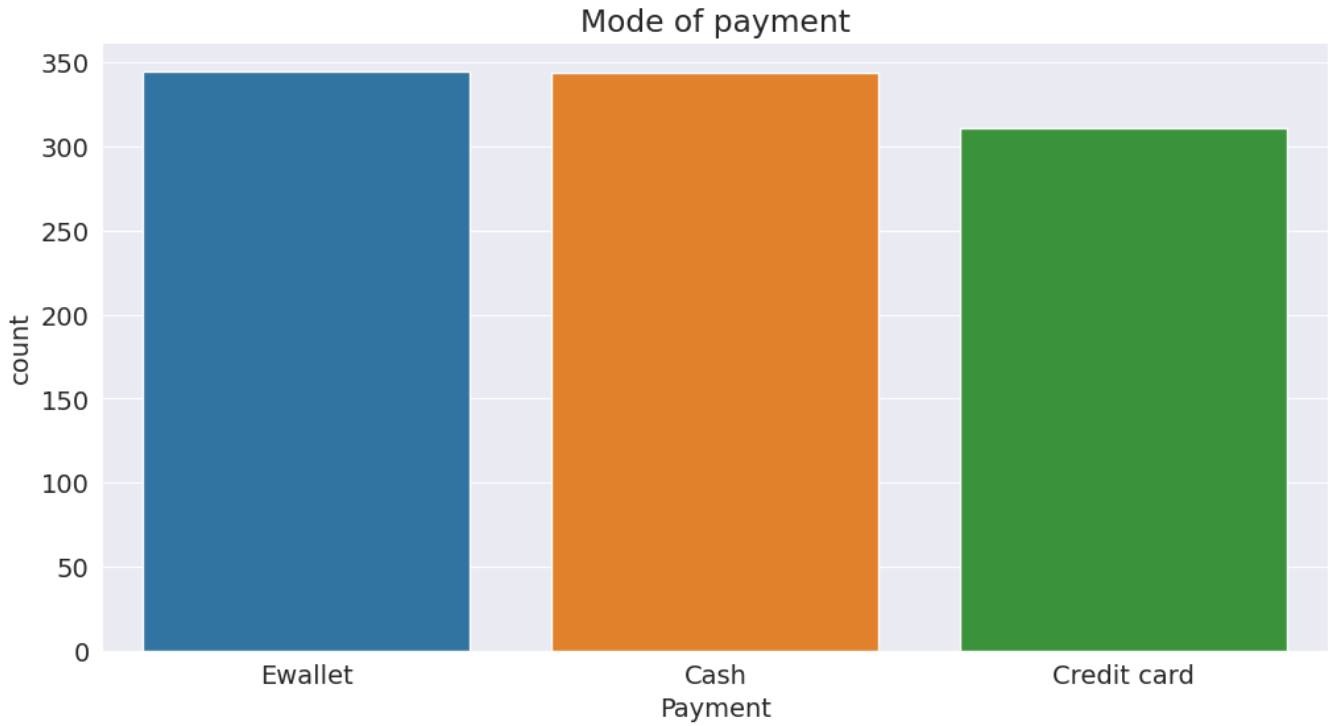
Like what you see? Visit the data table notebook to learn more about interactive tables.

Values


```
mode_of_payment = updated_superstore_df.Payment.value_counts()
mode_of_payment
```

```
Ewallet        345
Cash           344
Credit card    311
Name: Payment, dtype: int64
```

```
plt.figure(figsize=(12,6))
plt.title("Mode of payment")
sns.countplot(data=updated_superstore_df, x='Payment');
```




```
#Branch
branch_profitability = updated_superstore_df.groupby("Branch")[["gross income"]].sum()
branch_profitability
```

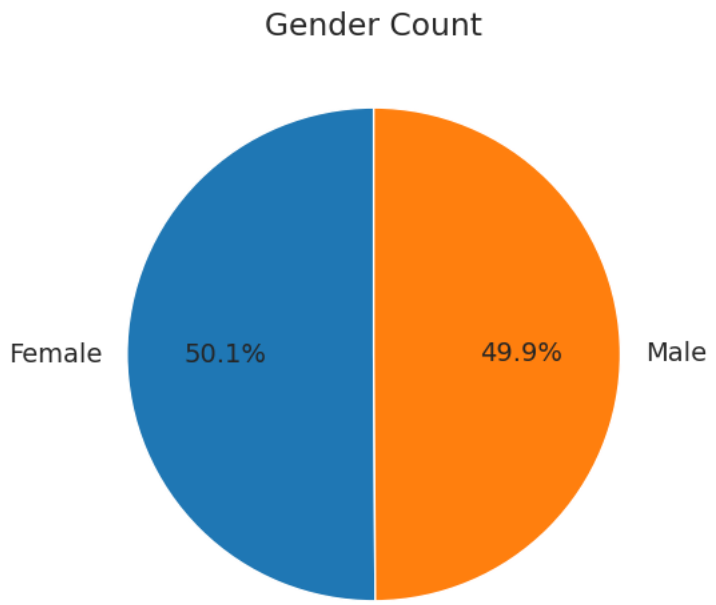
Branch	gross income
A	5057.1605
B	5057.0320
C	5265.1765

```
plt.figure(figsize=(10, 5))
plt.title("Branch profitability")
sns.barplot(x=branch_profitability.index,y= branch_profitability['gross income'], palette='CMRmap');
```


```
#Gender
gender_count = updated_superstore_df.Gender.value_counts()
gender_count
```

```
Female    501
Male      499
Name: Gender, dtype: int64
```

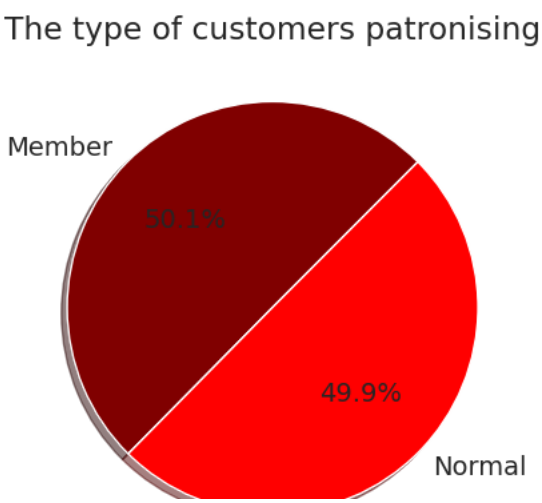
```
plt.figure(figsize=(12,6))
plt.title("Gender Count")
plt.pie(gender_count, labels=gender_count.index,autopct='%1.1f%%', startangle=90);
```




```
updated_superstore_df.groupby('Gender')[["Total"]].sum().reset_index()
```

	Gender	Total
0	Female	167882.925
1	Male	155083.824

```
#customer type
plt.figure(figsize=(8,5))
plt.title("The type of customers patronising")
plt.pie(updated_superstore_df['Customer type'].value_counts(),labels=updated_superstore_df['Customer type'].value_counts().index,autopct=
```



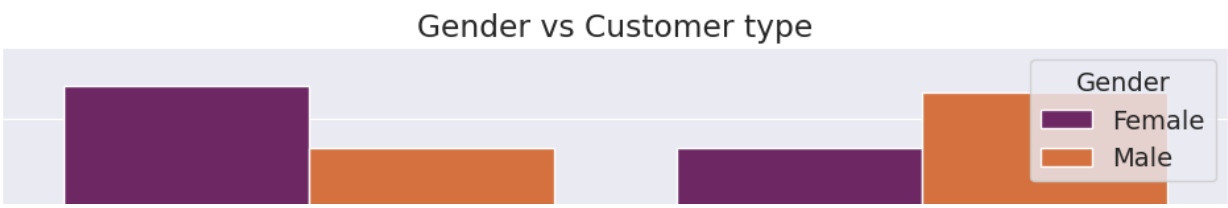

```
#Gender Vs Customer_type
gender_customer_count=updated_superstore_df.groupby(['Gender','Customer type']).size().reset_index()
gender_customer_count
```

	Gender	Customer type	0
0	Female	Member	261
1	Female	Normal	240
2	Male	Member	240
3	Male	Normal	259

```
# rename the columns
gender_customer_count.columns=['Gender','Customer type',"Counts"]
gender_customer_count
```

	Gender	Customer type	Counts
0	Female	Member	261
1	Female	Normal	240
2	Male	Member	240
3	Male	Normal	259

```
plt.figure(figsize=(12, 8))
plt.title("Gender vs Customer type")
sns.barplot(data=gender_customer_count,x='Customer type', y='Counts',hue="Gender",palette='inferno');
```



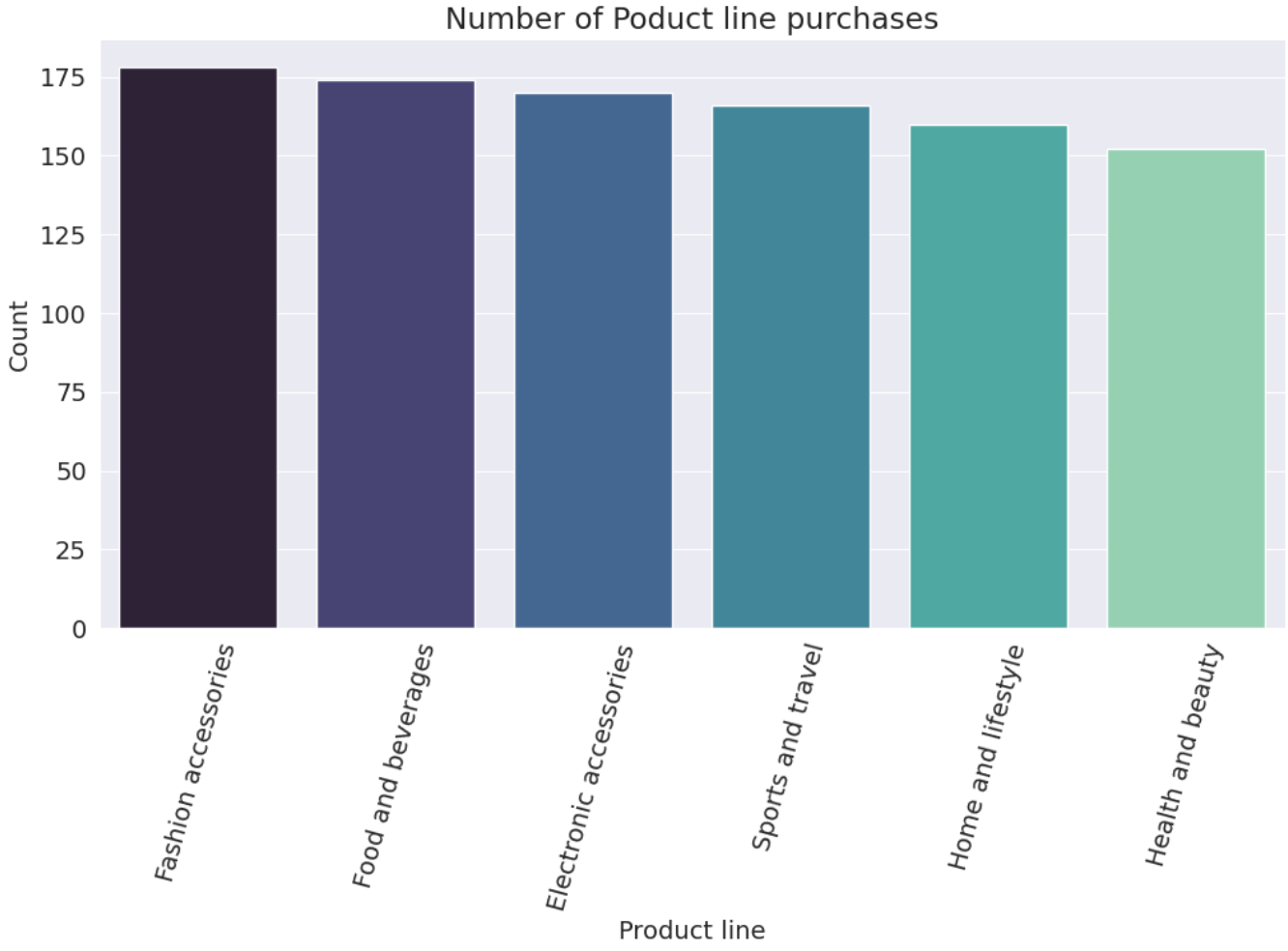

```
#Product_Line
updated_superstore_df['Product line'].unique()
```

```
array(['Health and beauty', 'Electronic accessories',
       'Home and lifestyle', 'Sports and travel', 'Food and beverages',
       'Fashion accessories'], dtype=object)
```

```
# number of purchases
product_line_count = updated_superstore_df['Product line'].value_counts().reset_index()
product_line_count.columns=['Product line','Count']
product_line_count
```

	Product line	Count
0	Fashion accessories	178
1	Food and beverages	174
2	Electronic accessories	170
3	Sports and travel	166
4	Home and lifestyle	160
5	Health and beauty	152

```
plt.figure(figsize=(12,6));
plt.title("Number of Poduct line purchases")
plt.xticks(rotation = 75);
plt.ylabel("Count")
sns.barplot(data=product_line_count,x= "Product line",y='Count', palette="mako");
```



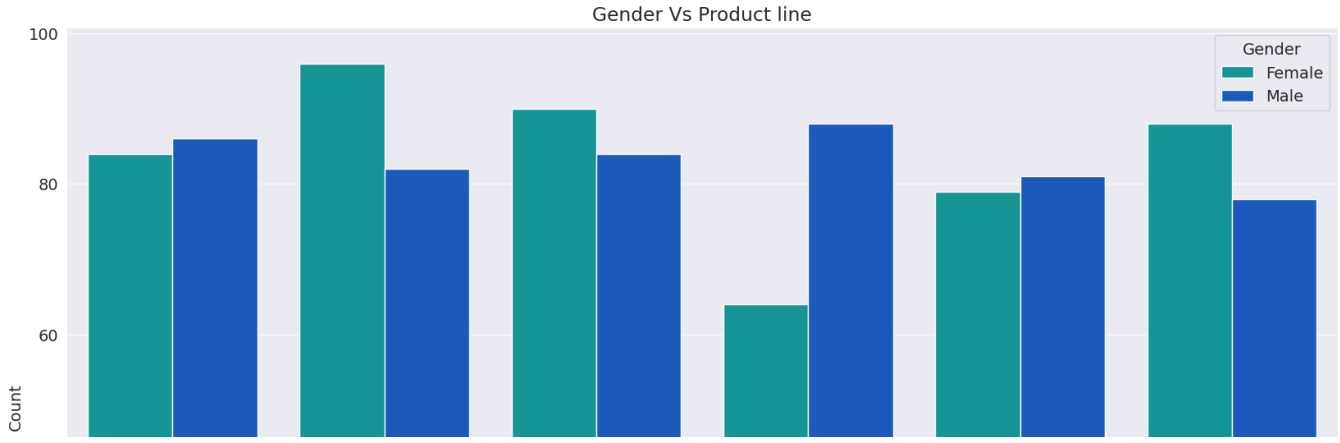
```
gender_product_count=updated_superstore_df.groupby(['Gender','Product line']).size().reset_index()
gender_product_count
```

	Gender	Product line	0
0	Female	Electronic accessories	84
1	Female	Fashion accessories	96
2	Female	Food and beverages	90
3	Female	Health and beauty	64
4	Female	Home and lifestyle	79
5	Female	Sports and travel	88
6	Male	Electronic accessories	86
7	Male	Fashion accessories	82
8	Male	Food and beverages	84
9	Male	Health and beauty	88
10	Male	Home and lifestyle	81
11	Male	Sports and travel	78

```
gender_product_count.columns=['Gender','Product line','Count']
gender_product_count
```

	Gender	Product line	Count
0	Female	Electronic accessories	84
1	Female	Fashion accessories	96
2	Female	Food and beverages	90
3	Female	Health and beauty	64
4	Female	Home and lifestyle	79
5	Female	Sports and travel	88
6	Male	Electronic accessories	86
7	Male	Fashion accessories	82
8	Male	Food and beverages	84
9	Male	Health and beauty	88
10	Male	Home and lifestyle	81
11	Male	Sports and travel	78

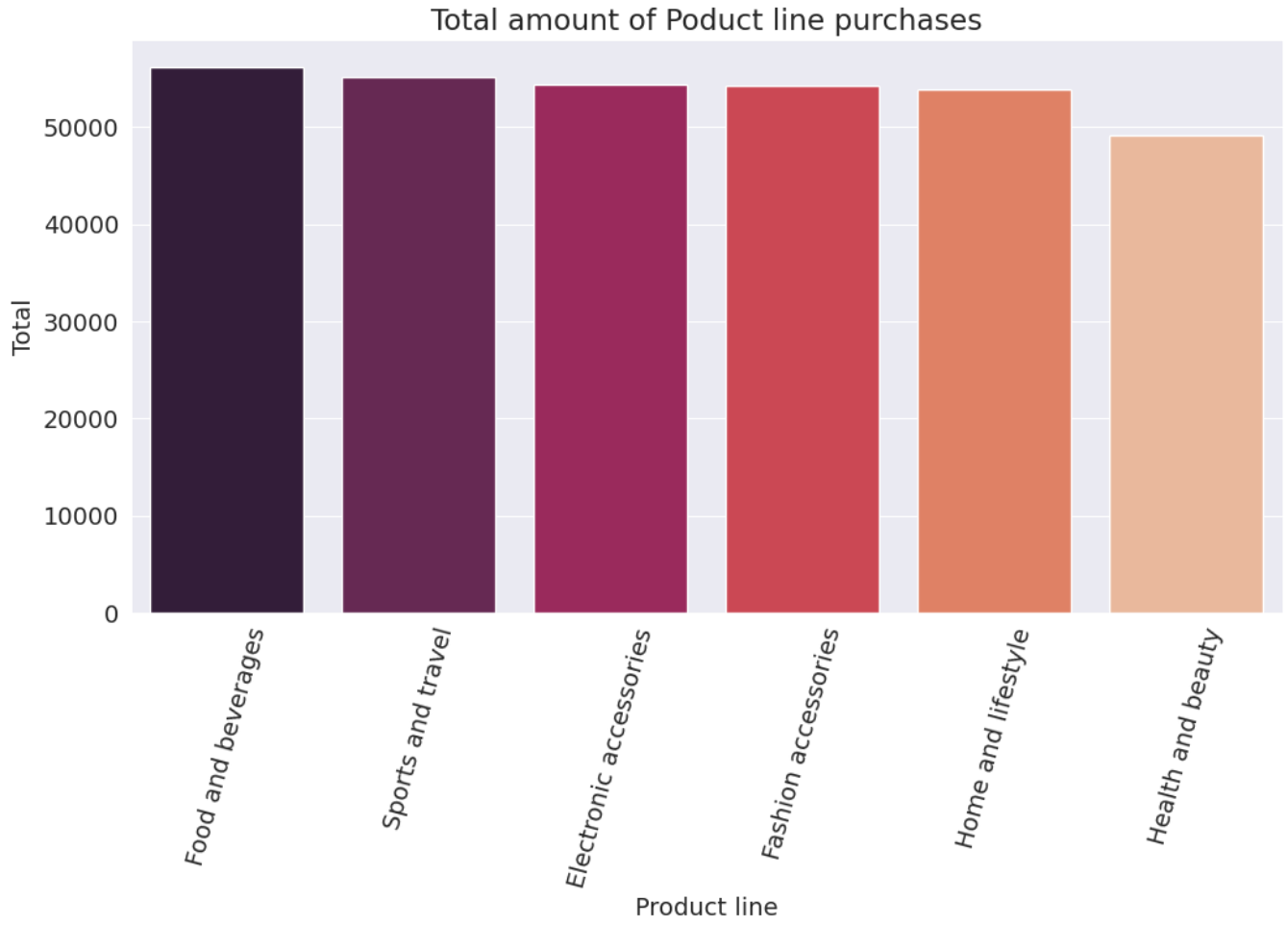
```
plt.figure(figsize=(20,12))
plt.title('Gender Vs Product line')
sns.barplot(data=gender_product_count,x='Product line', y='Count',hue="Gender", palette='winter_r');
```



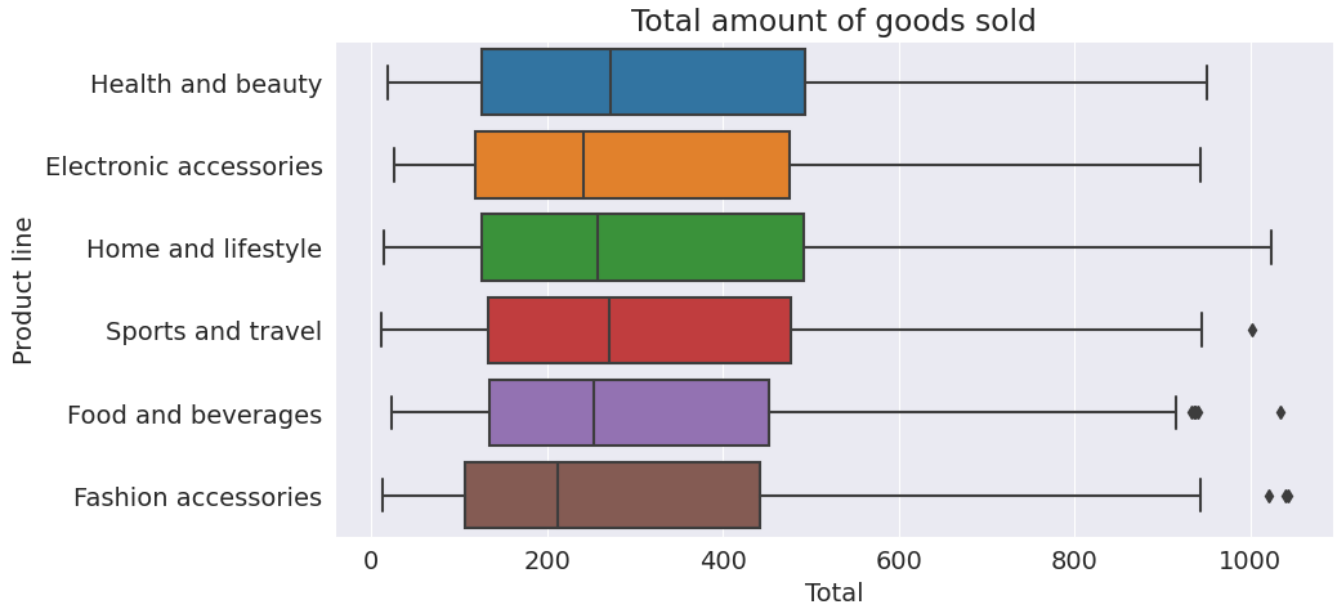

```
#Total amount of productline purchases
total_purchase = updated_superstore_df.groupby("Product line")[["Total"]].sum().sort_values("Total",ascending=False)
total_purchase
```

Product line	Total
Food and beverages	56144.8440
Sports and travel	55122.8265
Electronic accessories	54337.5315
Fashion accessories	54305.8950
Home and lifestyle	53861.9130
Health and beauty	49193.7390

```
plt.figure(figsize=(12,6));
plt.title("Total amount of Poduct line purchases")
plt.xticks(rotation = 75);
sns.barplot(x=total_purchase.index, y= total_purchase.Total, palette='rocket');
```



```
plt.title("Total amount of goods sold")
sns.boxplot(data=updated_superstore_df,y='Product line',x='Total');
```




```
#Finding how the customers rate their shopping experience
plt.figure(figsize=(10,6))
plt.title("Shopping Experience Rating")
sns.histplot(updated_superstore_df.Rating,bins=7,color='green');
```


```
def rating(x):
    if x['Rating'] <5:
        return 'Poor'
    elif x['Rating'] >=5 and  x['Rating'] <=7:
        return 'Fair'
    elif x['Rating'] >=7 and x['Rating']  <=8:
        return 'Good'
    else:
        return "Excellent"
updated_superstore_df['Sales_Rating']= updated_superstore_df.apply(rating,axis=1)
```

```
plt.title("Supermarket Rating")
sns.countplot(data=updated_superstore_df, x='Sales_Rating', palette='cubehelix');
```

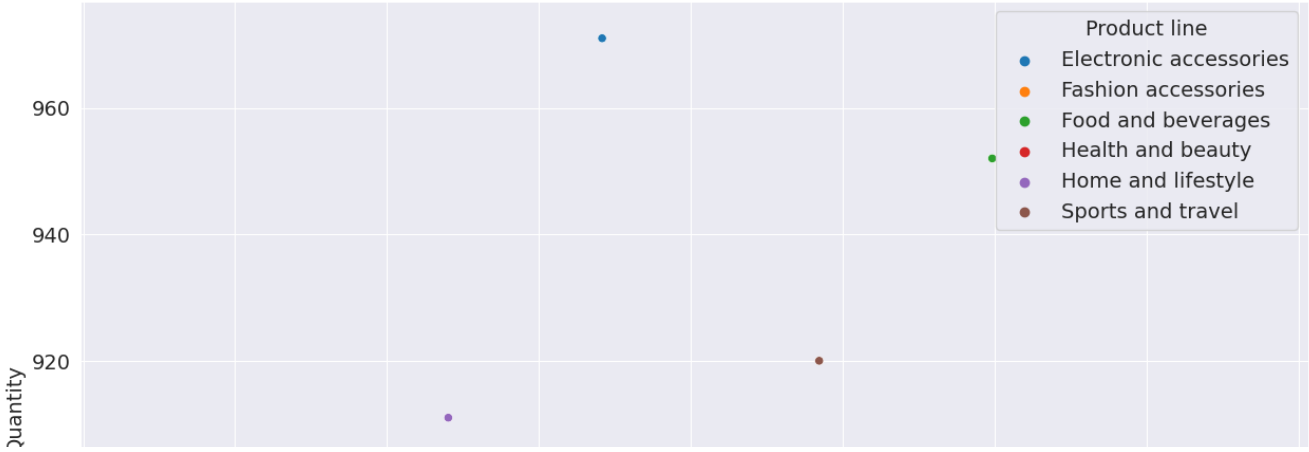

```
updated_superstore_df["Rating"].mean()
```

```
6.9727
```

```
Quantity_bought = updated_superstore_df.groupby("Product line")[["Unit price", 'Quantity']].sum()
Quantity_bought
```

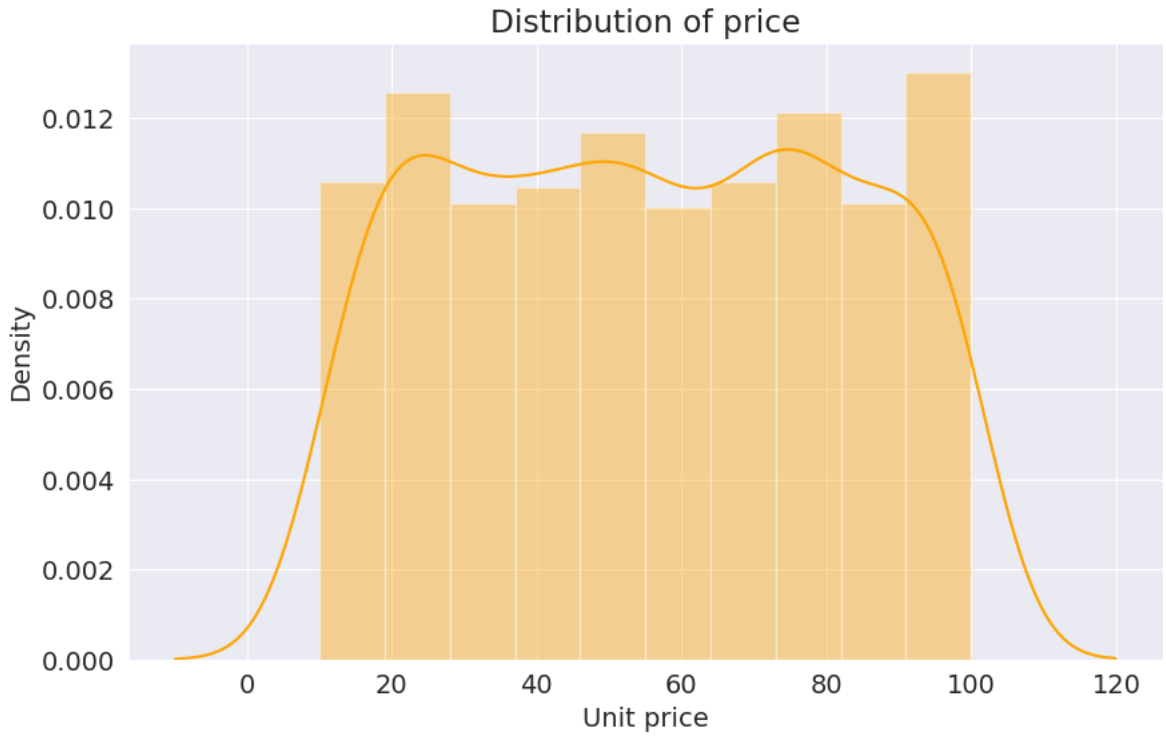
Product line	Unit price	Quantity
Electronic accessories	9103.77	971
Fashion accessories	10173.35	902
Food and beverages	9745.54	952
Health and beauty	8337.88	854
Home and lifestyle	8850.71	911
Sports and travel	9460.88	920

```
plt.figure(figsize=(15,10))
sns.scatterplot(data=Quantity_bought,x='Unit price', y="Quantity", hue='Product line');
```

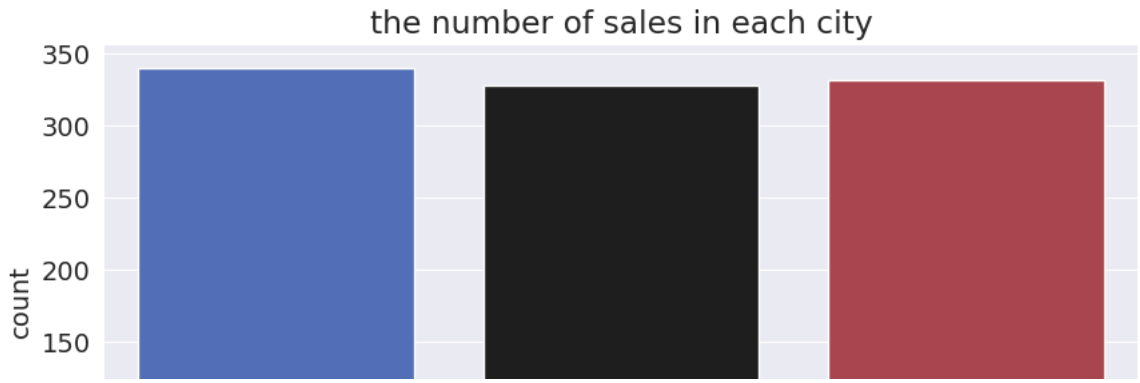



```
#UNIT PRICE
plt.figure(figsize=(10,6))
plt.title('Distribution of price')
sns.distplot(updated_superstore_df['Unit price'],color='orange',kde=True)
plt.show()
```

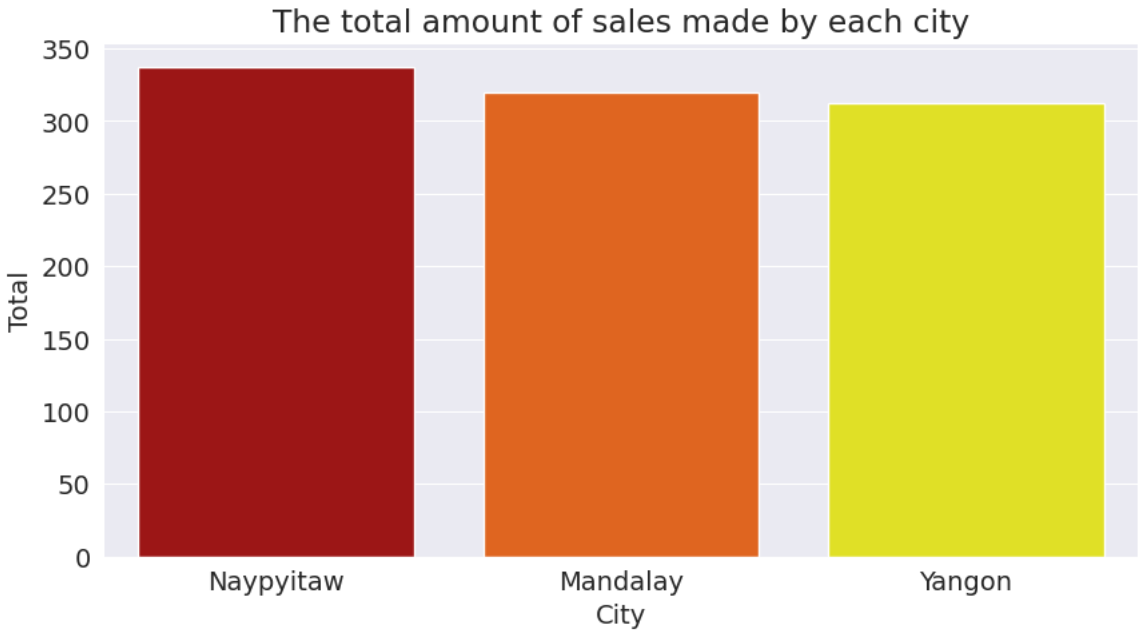
```
<ipython-input-51-350ae87e66c8>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(updated_superstore_df['Unit price'],color='orange',kde=True)
```




```
#CITY
# The number of sales in each city
plt.title('the number of sales in each city')
sns.countplot(data=updated_superstore_df, x="City", palette='icefire');
```




```
# The average sales made in each city
city_amount=updated_superstore_df.groupby('City')[["Total"]].mean().reset_index().sort_values('Total',ascending=False)
city_amount
plt.title("The total amount of sales made by each city")
sns.barplot(data=city_amount, x="City", y="Total", palette='hot');
```




```
#Quantity
plt.figure(figsize=(10,6))
plt.title("quantities sold")
plt.hist(updated_superstore_df['Quantity'])
plt.show()
```

quantities sold

```
#TIME
# create a function to parse the time column showing day and night
# if the time exceeds 16:00:00 then it is night
time = pd.to_datetime(['16:00:00']).time
def day_night(x):
    if x['Time'].time() > time:
        return 'Night'
    else:
        return "Day"
updated_superstore_df['Day/Night']= updated_superstore_df.apply(day_night, axis=1)
```

```
updated_superstore_df
```

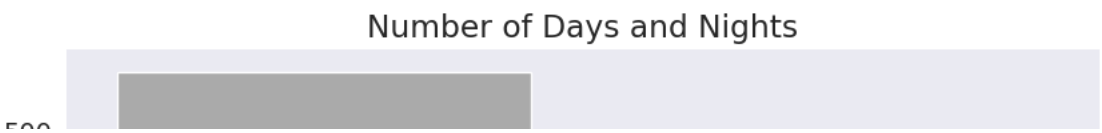
	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	gross income	Rating	Month	Year
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	26.1415	9.1	January	2019
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	...	3.8200	9.6	March	2019
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	...	16.2155	7.4	March	2019
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	23.2880	8.4	January	2019
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	...	30.2085	5.3	February	2019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	...	2.0175	6.2	January	2019
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	...	48.6900	4.4	March	2019
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	...	1.5920	7.7	February	2019
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	...	3.2910	4.1	February	2019
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	...	30.9190	6.6	February	2019

1000 rows × 25 columns

```
# number of days and night sales were made
day_night_count=updated_superstore_df['Day/Night'].value_counts()
day_night_count
```

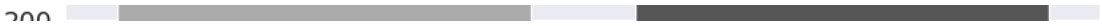
```
Day      568
Night    432
Name: Day/Night, dtype: int64
```

```
plt.title("Number of Days and Nights")
sns.countplot(data=updated_superstore_df,x='Day/Night', palette='gist_yarg');
```

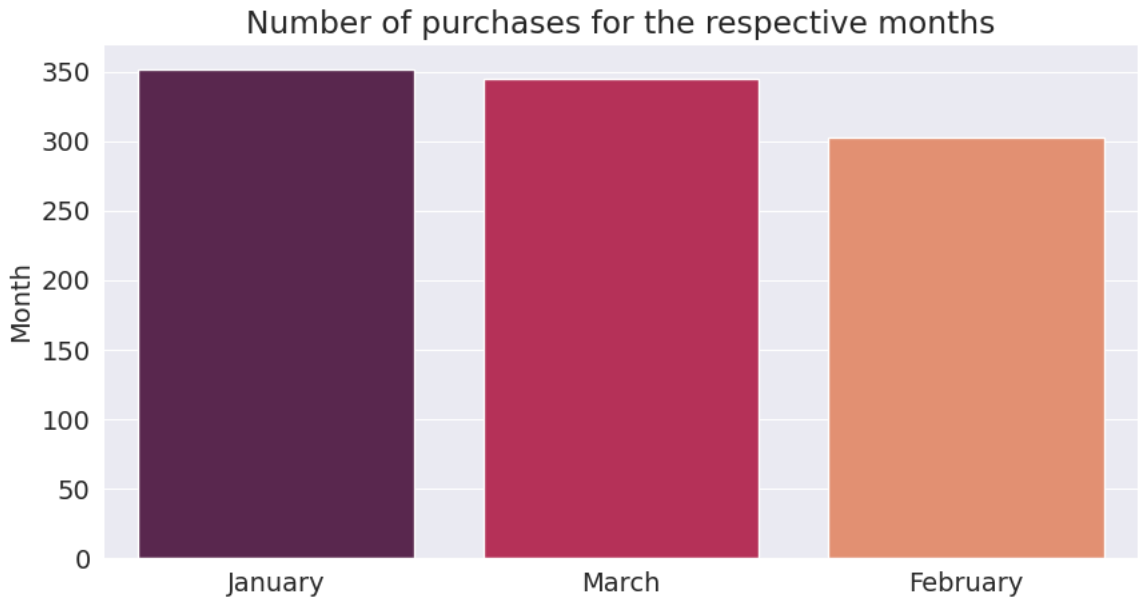



```
#DATE
# Number of sales made in each month
month_counts=updated_superstore_df['Month'].value_counts()
month_counts
```

```
January     352
March       345
February    303
Name: Month, dtype: int64
```



```
plt.title("Number of purchases for the respective months")
sns.barplot(x=month_counts.index,y=month_counts, palette='rocket');
```




```
# Total amount of goods sold in each month
sum_of_goods_sold=updated_superstore_df.groupby(['Month','Day/Night'])[['Total']].sum().reset_index()
sum_of_goods_sold
```

	Month	Day/Night	Total
0	February	Day	55866.4890
1	February	Night	41352.8850
2	January	Day	71949.1815
3	January	Night	44342.6865
4	March	Day	56780.1570
5	March	Night	52675.3500

```
!pip install sort-dataframeby-monthorweek
!pip install sorted-months-weekdays
```

```
Collecting sort-dataframeby-monthorweek
  Downloading sort_dataframeby_monthorweek-0.4.tar.gz (2.8 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: sort-dataframeby-monthorweek
  Building wheel for sort-dataframeby-monthorweek (setup.py) ... done
  Created wheel for sort-dataframeby-monthorweek: filename=sort_dataframeby_monthorweek-0.4-py3-none-any.whl size=3419 sha256=9fae7
  Stored in directory: /root/.cache/pip/wheels/6b/fd/39/06eaac8d65d641f9f50856a48b57a5ec8351be2874beff01ec
Successfully built sort-dataframeby-monthorweek
Installing collected packages: sort-dataframeby-monthorweek
Successfully installed sort-dataframeby-monthorweek-0.4
Collecting sorted-months-weekdays
  Downloading sorted_months_weekdays-0.2.tar.gz (2.7 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: sorted-months-weekdays
  Building wheel for sorted-months-weekdays (setup.py) ... done
  Created wheel for sorted-months-weekdays: filename=sorted_months_weekdays-0.2-py3-none-any.whl size=3223 sha256=9fbd07d6a9531a746
  Stored in directory: /root/.cache/pip/wheels/bd/b4/f6/2c29a96668a9a13a568134857fd7b7a5186261f00f1d35661c
Successfully built sorted-months-weekdays
```

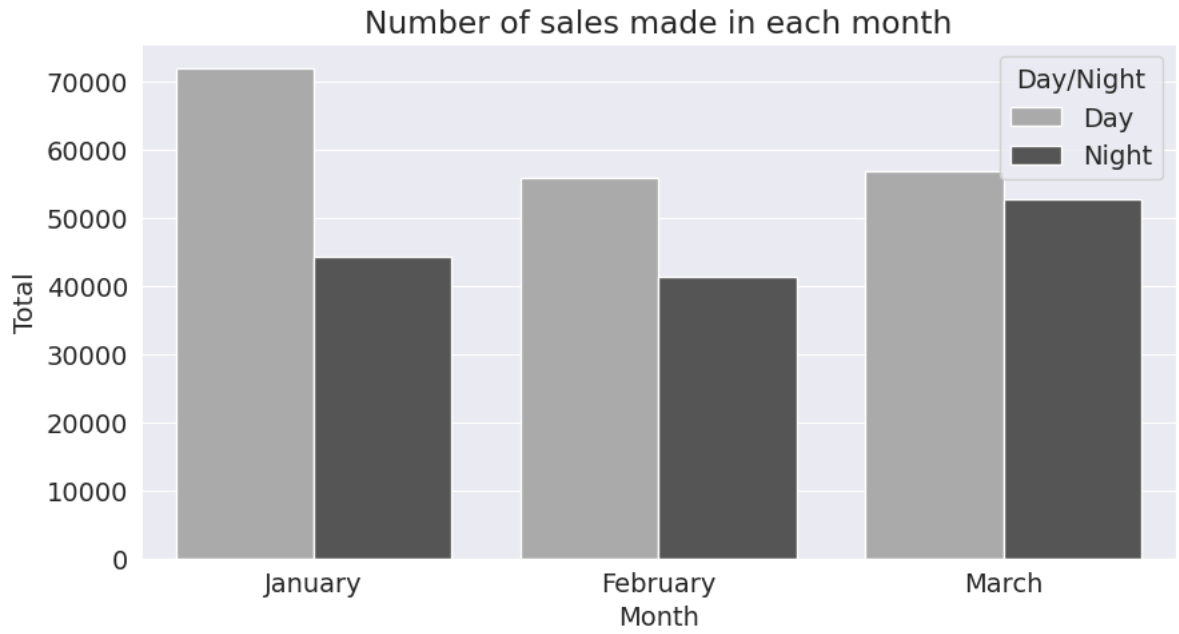
```
Installing collected packages: sorted-months-weekdays
Successfully installed sorted-months-weekdays-0.2
```

```
# import the sort module to help sort my month
import sort_dataframeby_monthorweek as sd
```

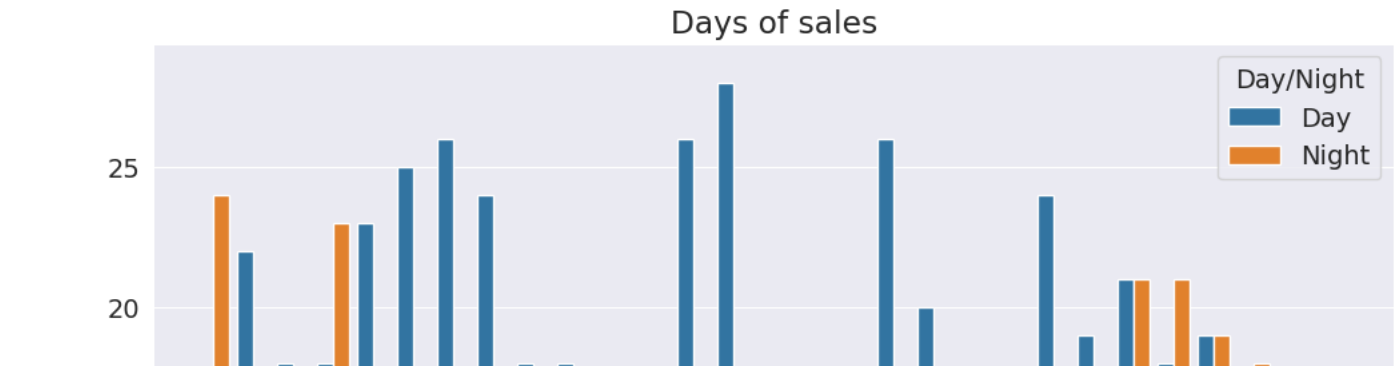
```
sort_month= sd.Sort_Dataframeby_Month(sum_of_goods_sold,'Month')
sort_month
```

	Month	Day/Night	Total
0	January	Day	71949.1815
1	January	Night	44342.6865
2	February	Day	55866.4890
3	February	Night	41352.8850
4	March	Day	56780.1570
5	March	Night	52675.3500

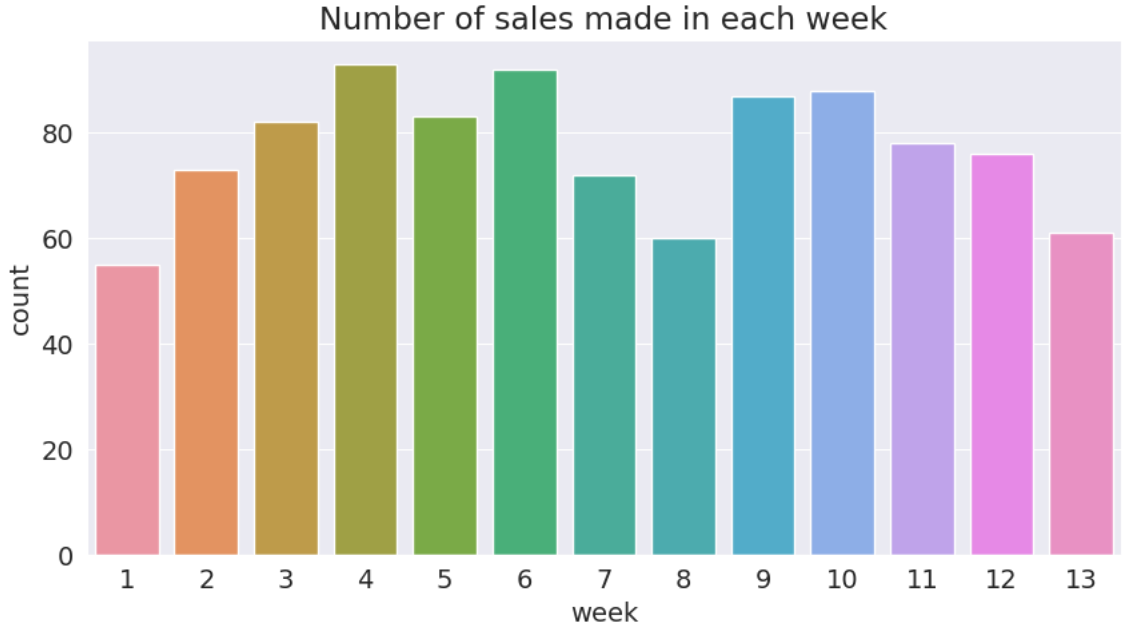
```
plt.title("Number of sales made in each month")
sns.barplot(data=sort_month,x='Month',y="Total", hue='Day/Night', palette='gist_yarg');
```




```
plt.figure(figsize=(12,8))
plt.title("Days of sales")
sns.countplot(data=updated_superstore_df, x='Day',hue="Day/Night");
```




```
plt.title("Number of sales made in each week")
sns.countplot(data=updated_superstore_df, x='week');
```



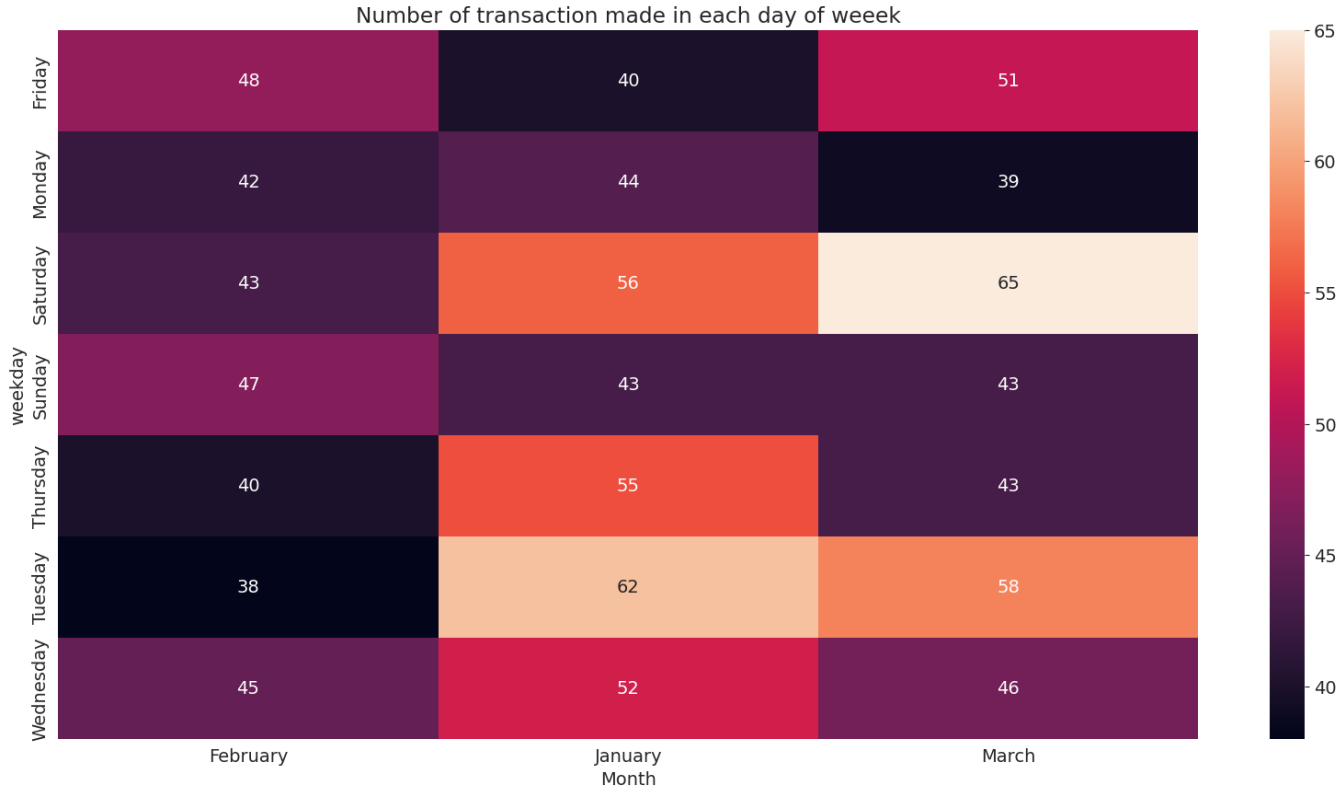

```
# the day of the week with the highest number of sales
def count_rows(rows):
    return len(rows)


by_cross = updated_superstore_df.groupby(["weekday",'Month']).apply(count_rows)
by_cross
```

```
weekday    Month
Friday     February    48
           January     40
           March       51
Monday     February    42
           January     44
           March       39
Saturday   February    43
           January     56
           March       65
Sunday     February    47
           January     43
           March       43
Thursday   February    40
           January     55
           March       43
Tuesday    February    38
           January     62
           March       58
Wednesday  February    45
           January     52
           March       46
dtype: int64
```

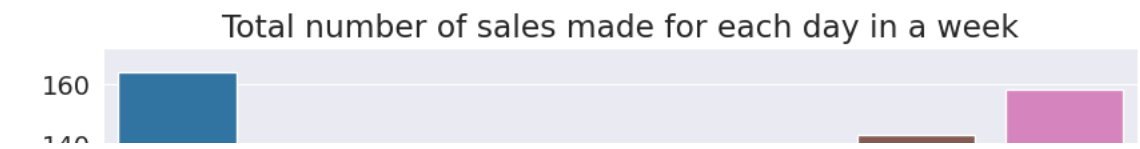
```
pivot=by_cross.unstack()
pivot
```

Month	February	January	March
weekday			
Friday	48	40	51
Monday	42	44	39
Saturday	43	56	65
Sunday	47	43	43
Thursday	40	55	43

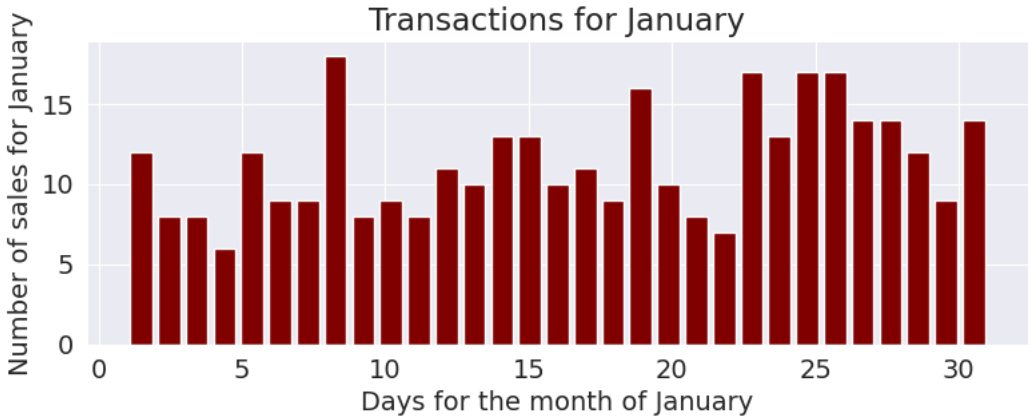
```
plt.figure(figsize=(20,10))
plt.title("Number of transaction made in each day of weeek")
sns.heatmap(pivot,annot=True);
```

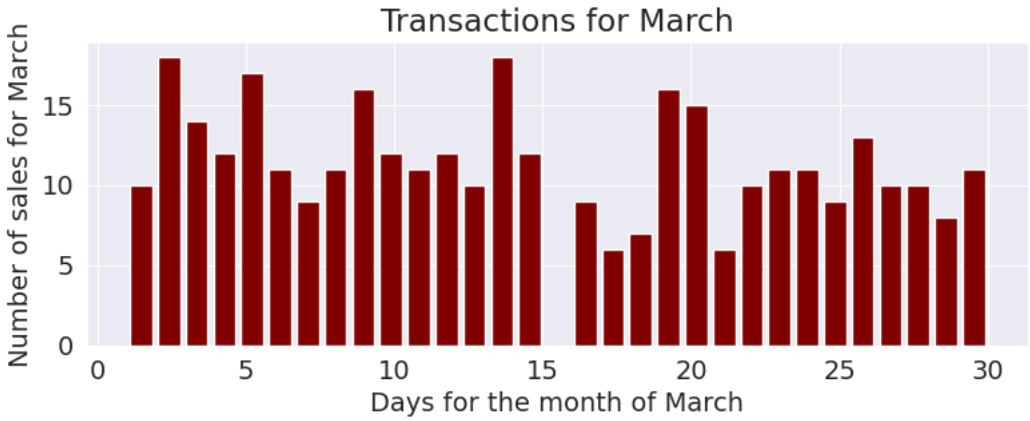


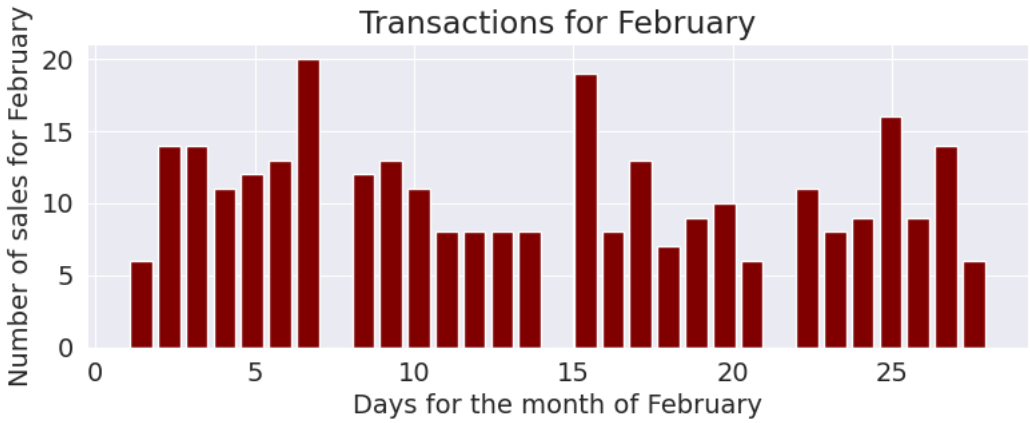

```
plt.title("Total number of sales made for each day in a week")
sns.countplot(data=updated_superstore_df, x='weekday');
```




```
#How did the supermarket fare for each month
for i , months in enumerate(updated_superstore_df['Month'].unique(),1):
    plt.figure(figsize=(20,10))
    plt.subplot(3,2,i)
    plt.xlabel('Days for the month of {}'.format(months))
    plt.title("Transactions for {}".format(months))
    plt.ylabel('Number of sales for {}'.format(months))
    df_out= updated_superstore_df[updated_superstore_df['Month']==months]
    plt.hist(df_out['Day'], rwidth=0.8, bins=31, color='maroon')
```








```
updated_superstore_df.head()
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	...	gross incom
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	...	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	...	3.820
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	...	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	...	23.288

3

"'Inference and Conclusion The most expensive item in the supermarket is worth 99.96. The items are located in both the Health and beauty and Sports and travel section. About 50.1 percent of the customers are female while the remaining 49.9 percent are male. In thesame vein, 50.1 percent of the customers are members while the remaining 49.9% are normal customers. The fashion and accessories section had the highest number of sales. The highest amount of sales was made in the Food and beverage section. Most of the customers gave the supermarket fair rating. The price of each goods is evenly distributed from 10 t0 100. The highest number of sales was made in the city of Yangon. The highest amount of sales was made in the city of Napytaw. E-Wallet was the most used mode of payment closely followed by cash payment. The quantities of goods sold was evenly distributed between 1 unit to 10units. About 568 sales were made during the day while the remaining 432 were made in the night. The highest number of transactions was made in month of January. 15th has the highest number of transaction. The supermarket made the most sales sales in the 4th week. The highest number of transaction was made on the 8th in January, 7th in febuary and 1st and 14th in march. The supermarket made the highest profit in branch C. Price has does not have effect on the quantities demanded. Inference from analysis, the relationship between Unit price and quantity using a scatterplot we can see that price does not neccessarily influence the demand for goods. So to make more profit it is only advisable for the owner to purchase more of the goods that are perceived to have higher prices. The average rating of the supermarket is 6.9 which is fair.This is not good enough for a supermarket looking to get more patronage and more profit margin. The owner should try to improve the shopping experience of its customer. As the saying goes "The customer is the king"."'