(1) (Exercices) Supermarket Sales Analysis

July 23, 2022

[270]:
```
print('Here, we are exploring supermarket sales.')
```

```
Here, we are exploring supermarket sales.
```

[271]:
```
print('Important note, the symbol "K" is the Kyat, currency of the Myanmar, and␣
↪does not mean a thousand.')
```

```
Important note, the symbol "K" is the Kyat, currency of the Myanmar, and does
not mean a thousand.
```

[272]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[273]:
```
market_sales_df = pd.read_csv('supermarket_sales - Sheet1.csv')
```

[274]:
```
market_sales_df.head()
```

[274]:

	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
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019						
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019						

```
3  20:33      Ewallet  465.76                4.761905       23.2880     8.4
4  10:37      Ewallet  604.17                4.761905       30.2085     5.3
```

[275]:
```
print('Count of existing data : ')
print('  ')
market_sales_df.notnull().count()
```

```
Count of existing data :


Invoice ID                1000
Branch                    1000
City                      1000
Customer type             1000
Gender                    1000
Product line              1000
Unit price                1000
Quantity                  1000
Tax 5%                    1000
Total                     1000
Date                      1000
Time                      1000
Payment                   1000
cogs                      1000
gross margin percentage   1000
gross income              1000
Rating                    1000
dtype: int64
```

[ ]:
```

```

[276]:
```
print('Missing values : ')
market_sales_df.isna().sum()
```

```
Missing values :

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
```

[277]:
```
print('Null values : ')
market_sales_df.isnull().sum()
```

```
Null values :
```

[277]:
```
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
```

[278]:
```
print('There are no missing or null values in the data set.')
```

```
There are no missing or null values in the data set.
```

[279]:
```
print('Start date of data sales : 2019-01-01')
print('  ')
date = market_sales_df.Date
date.replace('/','-')
sales_date = pd.to_datetime(market_sales_df.Date)
np.min(sales_date)
```

```
Start date of data sales : 2019-01-01
```

[279]:
```
Timestamp('2019-01-01 00:00:00')
```

[280]:
```
print('End date of data sales')
print('  ')
np.max(sales_date)
```

```
End date of data sales
```

[280]:
```
Timestamp('2019-03-30 00:00:00')
```

[281]:
```
print('The data has been collected during 88 days, 3 months.')
np.max(sales_date) - np.min(sales_date)
```

```
The data has been collected during 88 days, 3 months.
```

[281]:
```
Timedelta('88 days 00:00:00')
```

[282]:
```
print('Count of data by gender')
print('  ')
market_sales_df.Gender.value_counts()
```

```
Count of data by gender
```

[282]:
```
Female    501
Male      499
Name: Gender, dtype: int64
```

[283]:
```
print('  ')
print('The data is spread evenly accross both sex (2 more Males).')
print('  ')
```

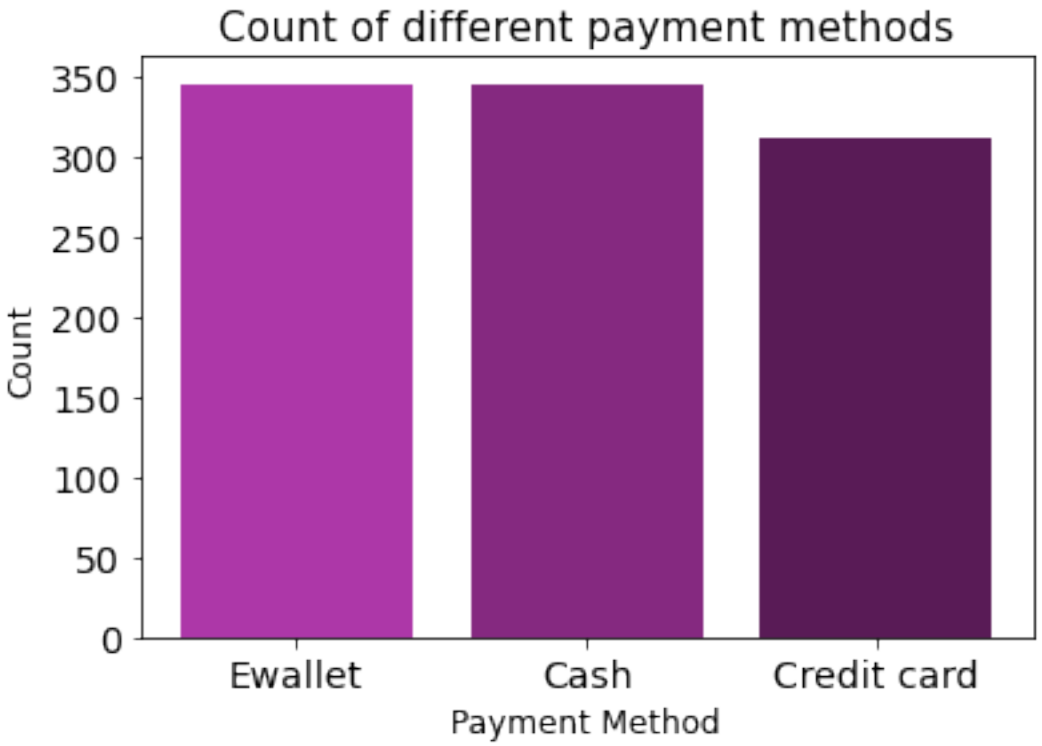
```
The data is spread evenly accross both sex (2 more Males).
```

[284]:
```
print('Count of different payment methods')
print('  ')
payment_count = market_sales_df.Payment.value_counts()
payment_count
```

```
Count of different payment methods
```

[284]:
```
Ewallet        345
Cash           344
Credit card    311
Name: Payment, dtype: int64
```

[285]:
```
payment_count_df = payment_count.reset_index().rename(columns={'index':'Type of␣
↪payment','Payment':'Count'})
x = payment_count_df['Type of payment']
y = payment_count_df['Count']
plt.bar(x,y,color=['#ad37a8','#852980','#591b56'])
plt.ylabel('Count',fontsize=12)
plt.xlabel('Payment Method',fontsize=12)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.title('Count of different payment methods',fontsize=15)
plt.show()
```



[286]:
```
print('The most used payment method is Ewallet, followed by Cash, ending with␣
↪credit card, even though each is very close to one another. There is no main␣
↪tendency. ')
```

```
The most used payment method is Ewallet, followed by Cash, ending with credit
card, even though each is very close to one another. There is no main tendency.
```

[287]:
```
print('Count of sales by member / non-member (normal) data')
market_sales_df['Customer type'].value_counts()
```

```
Count of sales by member / non-member (normal) data
```

[287]:
```
Member    501
Normal    499
Name: Customer type, dtype: int64
```

[288]:
```
print('The number of member customers buying is very close to the non-member␣
 ↪customers number.')
```

```
The number of member customers buying is very close to the non-member customers
number.
```

[289]:
```
print('Count of sales data by city, repsectively, Yangon, Mandalay and␣
 ↪Naypyitaw, all cities of the asian country Bruma (Myanmar).')
market_sales_df.City.value_counts()
```

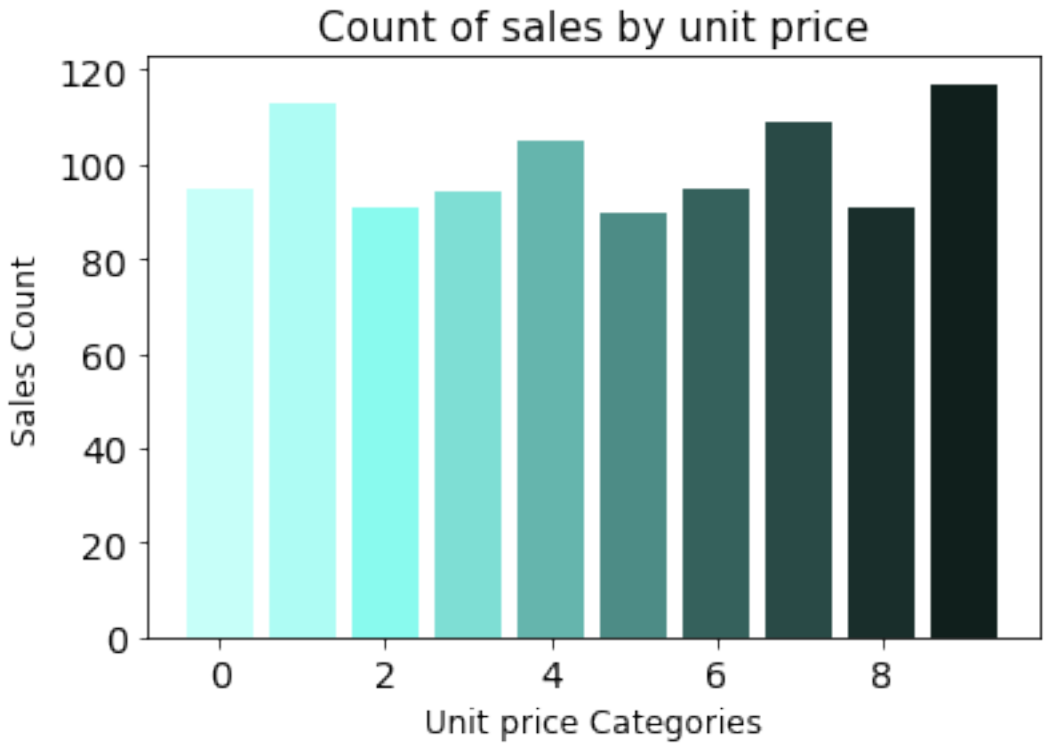
```
Count of sales data by city, repsectively, Yangon, Mandalay and Naypyitaw, all
cities of the asian country Bruma (Myanmar).
```

[289]:
```
Yangon       340
Mandalay     332
Naypyitaw    328
Name: City, dtype: int64
```

[290]:
```
print('Number of sales across city is even.')
```

```
Number of sales across city is even.
```

[291]:
```
sales_by_unit_price = ((pd.cut(market_sales_df['Unit price'], bins= 10).
 ↪value_counts()).sort_index())
sales_by_unit_price = sales_by_unit_price.reset_index().rename(columns={'index':
 ↪'Unit Price','Unit price':'Sales Count'})
x_1 = sales_by_unit_price['Unit Price'].reset_index()
y_1 = sales_by_unit_price['Sales Count']
plt.
 ↪bar(x_1['index'],y_1,color=['#c7fff9','#aefcf4','#89faee','#7eded4','#65b5ad','#4d8c86','#3
plt.ylabel('Sales Count',fontsize=12)
plt.xlabel('Unit price Categories',fontsize=12)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.title('Count of sales by unit price',fontsize=15)
plt.show()
sales_by_unit_price
```

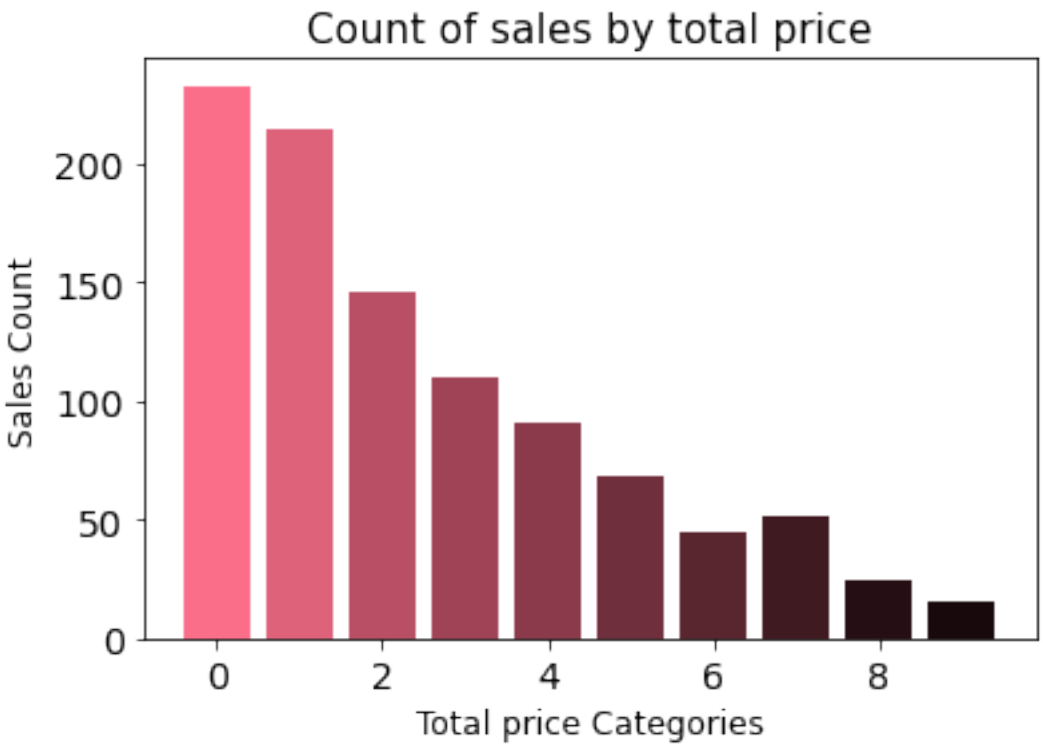


[291]:

	Unit Price	Sales Count
0	(9.99, 19.068]	95
1	(19.068, 28.056]	113
2	(28.056, 37.044]	91
3	(37.044, 46.032]	94
4	(46.032, 55.02]	105
5	(55.02, 64.008]	90
6	(64.008, 72.996]	95
7	(72.996, 81.984]	109
8	(81.984, 90.972]	91
9	(90.972, 99.96]	117

[292]:
```
sales_by_total_price = (pd.cut(market_sales_df['Total'], bins= 10).
 ↪value_counts()).sort_index()
sales_by_total_price = sales_by_total_price.reset_index().
 ↪rename(columns={'index':'Total Price','Total':'Sales Count'})
x_2 = sales_by_total_price['Total Price'].reset_index()
y_2 = sales_by_total_price['Sales Count']
plt.
 ↪bar(x_2['index'],y_2,color=['#fa6e8a','#de627a','#b84f64','#a14356','#8a3a4a','#702f3c','#5
plt.ylabel('Sales Count',fontsize=12)
plt.xlabel('Total price Categories',fontsize=12)
plt.xticks(fontsize=14)
```

```
plt.yticks(fontsize=14)
plt.title('Count of sales by total price',fontsize=15)
plt.show()
sales_by_total_price
```



[292]:

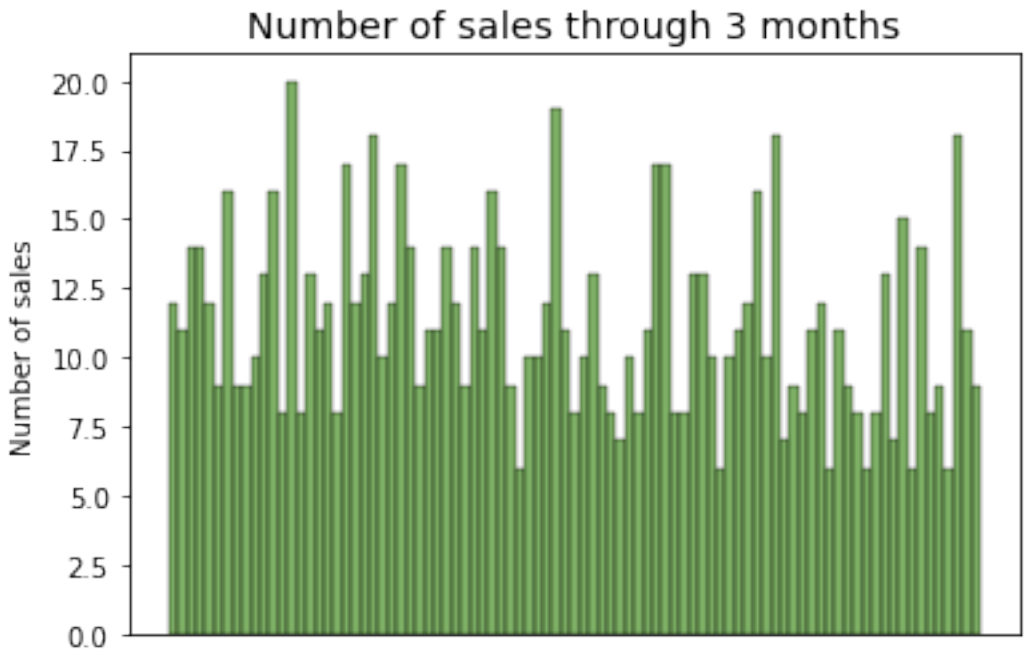
	Total Price	Sales Count
0	(9.647, 113.876]	233
1	(113.876, 217.073]	215
2	(217.073, 320.27]	146
3	(320.27, 423.467]	110
4	(423.467, 526.664]	91
5	(526.664, 629.861]	68
6	(629.861, 733.059]	45
7	(733.059, 836.256]	51
8	(836.256, 939.453]	25
9	(939.453, 1042.65]	16

[293]:
```
print('The most sales per unit price, have the highest unit price, between 91␣
↪and 100.')
print('Whereas the most sales per total price, have the lowest total price,␣
↪between 10 and 114.')
```

```
The most sales per unit price, have the highest unit price, between 91 and 100.
```

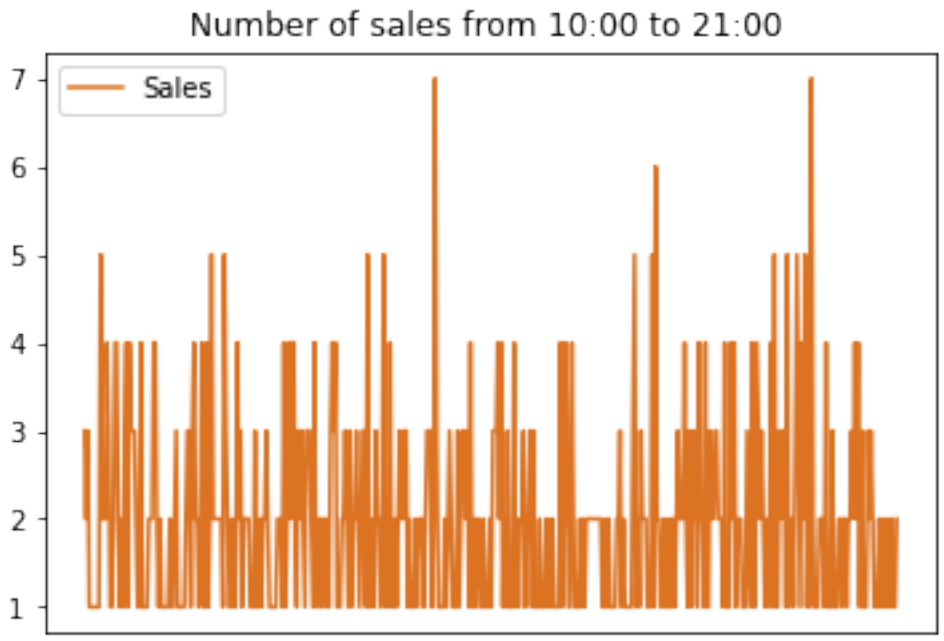
Whereas the most sales per total price, have the lowest total price, between 10 and 114.

[294]:
```
sns.histplot(market_sales_df['Date'].sort_index(),color='#519432')
plt.title('Number of sales through 3 months',fontsize=14)
plt.ylabel('Number of sales')
plt.xticks([])
plt.xlabel('')
plt.show()
```



[295]:
```
time_of_sale = market_sales_df.Time.sort_values()
times = time_of_sale.sort_values().reindex().value_counts().sort_index().
↪reset_index(level=0).rename(columns={'index':'TimeStamp','Time':'Sales'})
```

[296]:
```
times.plot(color='#db7323')
plt.xticks([])
plt.title('Number of sales from 10:00 to 21:00 ')
plt.show()
```

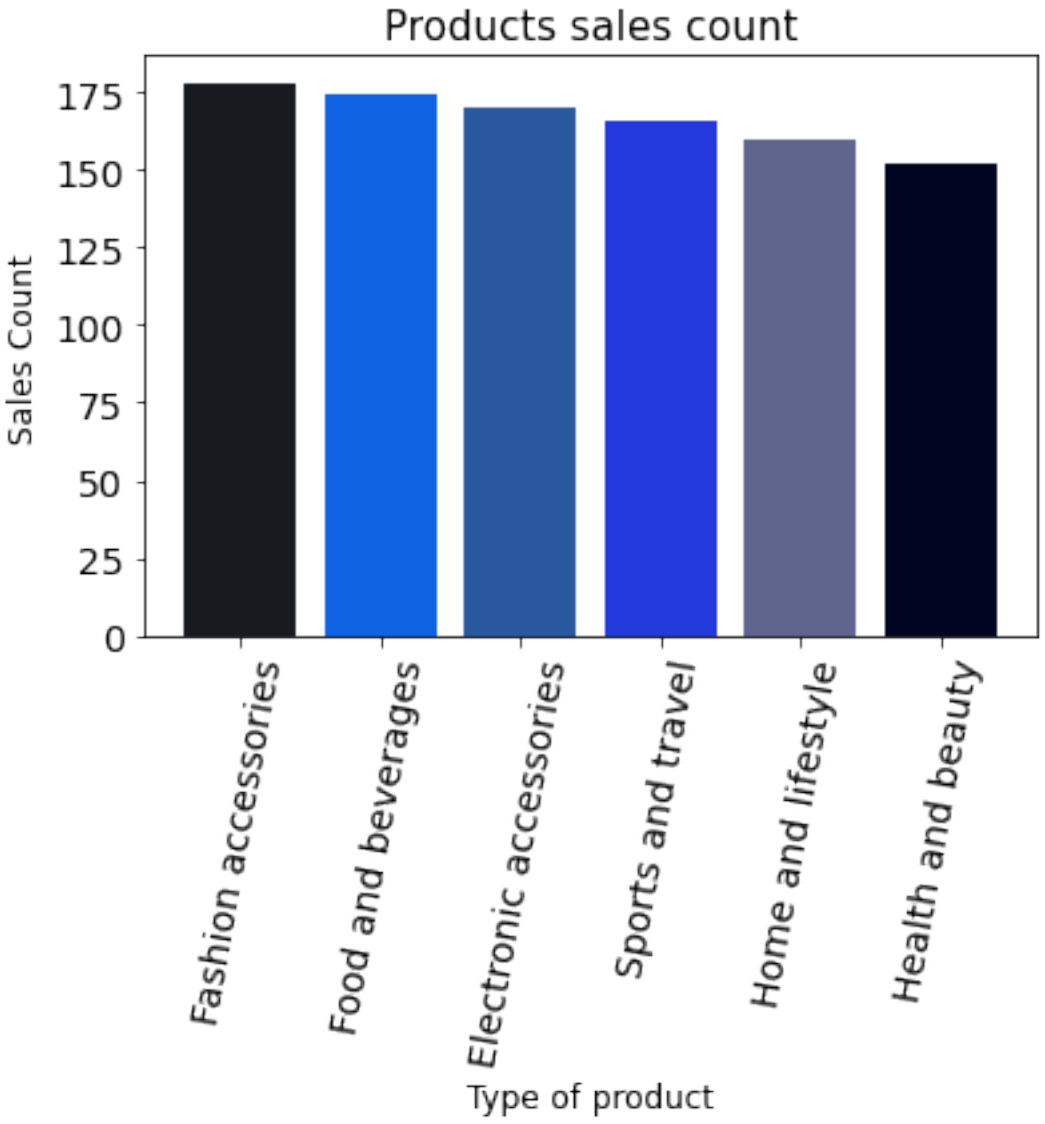


[297]:
```
print('  ')
print('Products sales count : ')
print('  ')
most_sold_products = market_sales_df['Product line'].value_counts()
print(most_sold_products)
print('Most sold products are fashion accessories, while health and beauty␣
 ↪products are the less sold.')
print('  ')
most_sold_products = most_sold_products.reset_index().rename(columns={'index':
 ↪'Type of product','Product line':'Sales Count'})
x_3 = most_sold_products['Type of product']
y_3 = most_sold_products['Sales Count']
plt.
 ↪bar(x_3,y_3,color=['#171a1f','#1064e3','#29589e','#2339db','#5f658c','#020521'])
plt.ylabel('Sales Count',fontsize=12)
plt.xlabel('Type of product',fontsize=12)
plt.xticks(fontsize=14,rotation=80)
plt.yticks(fontsize=14)
plt.title('Products sales count',fontsize=15)
plt.show()
most_sold_products
```

```
Products sales count :

Fashion accessories       178
```

```
Food and beverages        174
Electronic accessories    170
Sports and travel         166
Home and lifestyle        160
Health and beauty         152
Name: Product line, dtype: int64
Most sold products are fashion accessories, while health and beauty products are
the less sold.
```

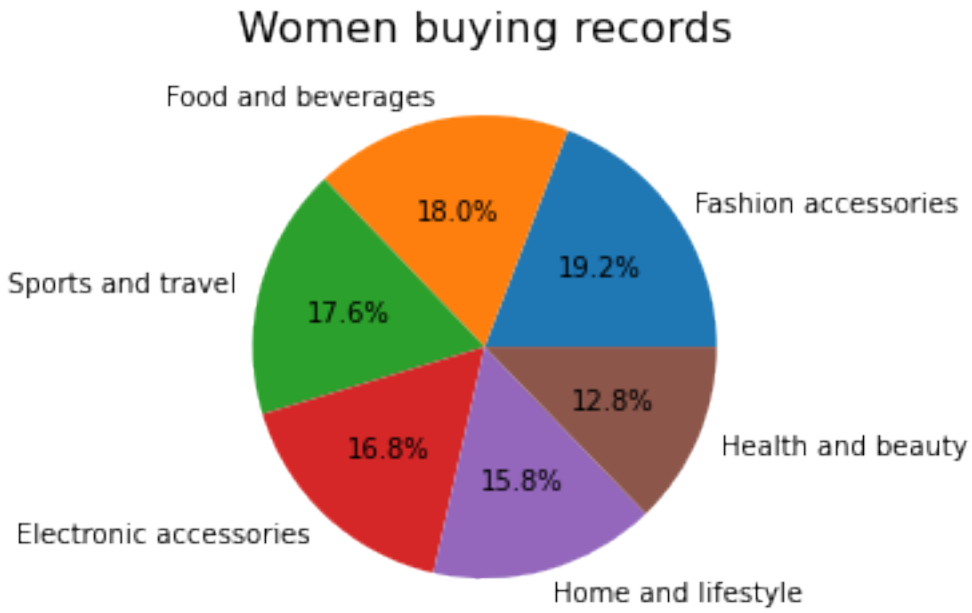


[297]:

	Type of product	Sales Count
0	Fashion accessories	178
1	Food and beverages	174

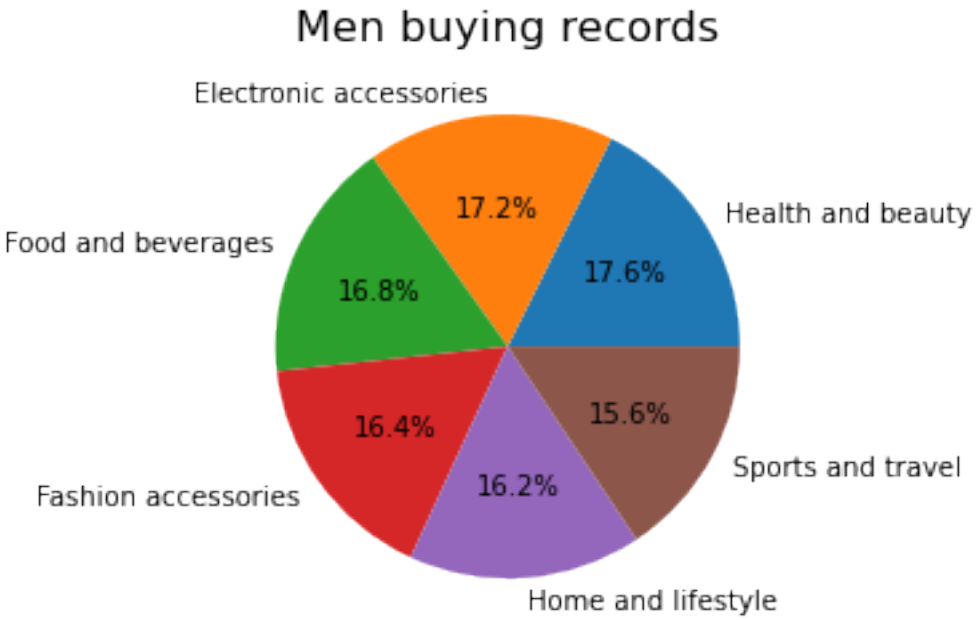
```
2  Electronic accessories          170
3       Sports and travel          166
4      Home and lifestyle          160
5       Health and beauty          152
```

[298]:
```
females = market_sales_df.loc[market_sales_df.Gender == 'Female']
women_buying_records = females['Product line'].value_counts()
women_buying_records_df = women_buying_records.reset_index().
 ↪rename(columns={'index':'Product lines','Product line':'Sales count'})
women_buying_records_df
plt.pie(women_buying_records,
        labels=(women_buying_records_df['Product lines']),
        autopct='%1.1f%%'
       )
plt.title('Women buying records',fontsize=16)
plt.show()
```



[299]:
```
males = market_sales_df.loc[market_sales_df.Gender == 'Male']
men_buying_records = males['Product line'].value_counts()
men_buying_records_df = men_buying_records.reset_index().
 ↪rename(columns={'index':'Product lines','Product line':'Sales count'})
men_buying_records_df
plt.pie(men_buying_records,
        labels=(men_buying_records_df['Product lines']),
        autopct='%1.1f%%'
```

```
        )
plt.title('Men buying records',fontsize=16)
plt.show()
```



[300]:
```
females = market_sales_df.loc[market_sales_df.Gender == 'Female']
print('Money spent by women :')
print('K'+ ' ' + str(females.Total.sum()))
males = market_sales_df.loc[market_sales_df.Gender == 'Male']
print('Money spent by men :')
print('K'+ ' ' + str(males.Total.sum()))
print('Women spend more money than men overall.')
```

```
Money spent by women :
K 167882.925
Money spent by men :
K 155083.824
Women spend more money than men overall.
```

[301]:
```
print('Average females spending')
print('K'+ ' ' + str(females.Total.mean()))
print('Average males spending')
print('K'+ ' ' + str(males.Total.mean()))
```

```
Average females spending
K 335.0956586826348
Average males spending
```

```
K 310.78922645290606
```

[302]:
```
members = market_sales_df.loc[market_sales_df['Customer type'] == 'Member']
print('Average members spending')
print('K'+ ' ' + str(members.Total.mean()))
non_members = market_sales_df.loc[market_sales_df['Customer type'] == 'Normal']
print('Average non members spending')
print('K'+ ' ' + str(non_members.Total.mean()))
print('Members spend on average more money than non members.')
```

```
Average members spending
K 327.7913053892216
Average non members spending
K 318.122855711423
Members spend on average more money than non members.
```

[303]:
```
print('Correlation Matrix : ')
corr_matrix = market_sales_df.corr()
corr_matrix
```

```
Correlation Matrix :
```

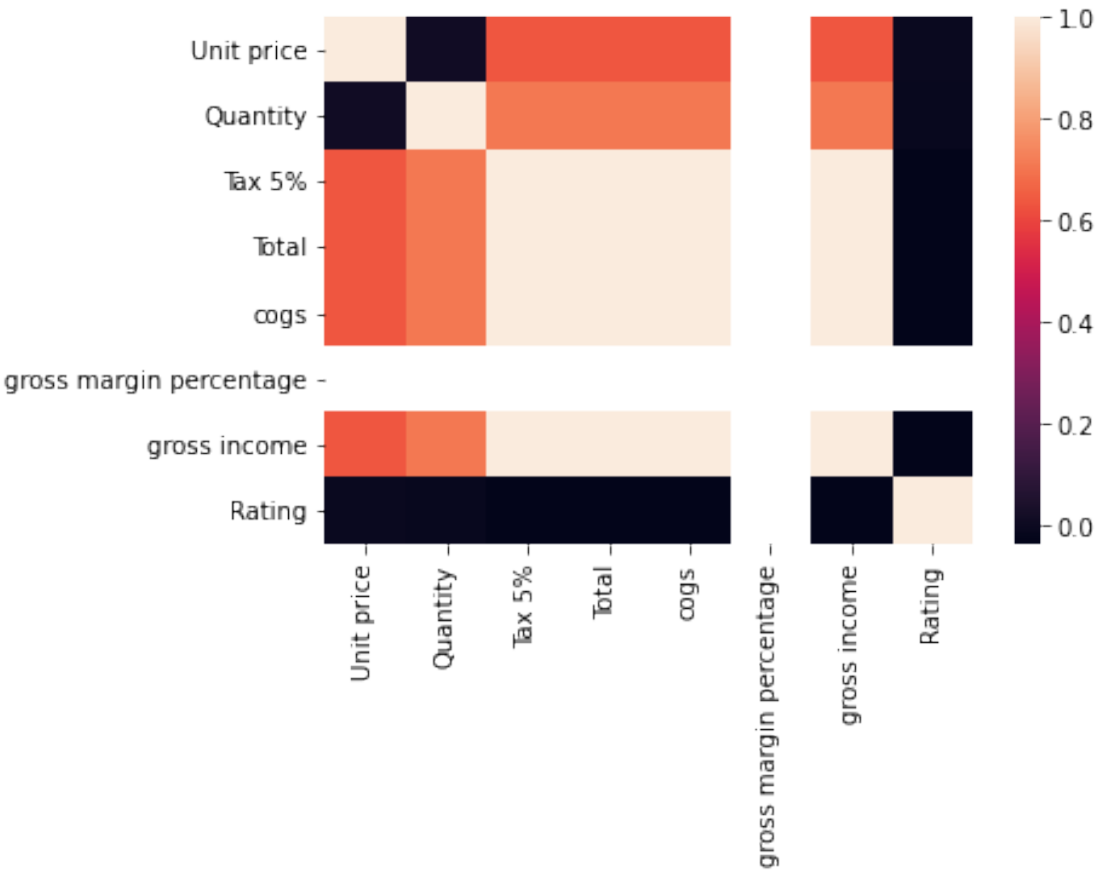
[303]:

	Unit price	Quantity	Tax 5%	Total	cogs	\
Unit price	1.000000	0.010778	0.633962	0.633962	0.633962	
Quantity	0.010778	1.000000	0.705510	0.705510	0.705510	
Tax 5%	0.633962	0.705510	1.000000	1.000000	1.000000	
Total	0.633962	0.705510	1.000000	1.000000	1.000000	
cogs	0.633962	0.705510	1.000000	1.000000	1.000000	
gross margin percentage	NaN	NaN	NaN	NaN	NaN	
gross income	0.633962	0.705510	1.000000	1.000000	1.000000	
Rating	-0.008778	-0.015815	-0.036442	-0.036442	-0.036442	

	gross margin percentage	gross income	Rating
Unit price	NaN	0.633962	-0.008778
Quantity	NaN	0.705510	-0.015815
Tax 5%	NaN	1.000000	-0.036442
Total	NaN	1.000000	-0.036442
cogs	NaN	1.000000	-0.036442
gross margin percentage	NaN	NaN	NaN
gross income	NaN	1.000000	-0.036442
Rating	NaN	-0.036442	1.000000

[305]:
```
print('Correlation Heatmap : ')
corr_heatmap = sns.heatmap(corr_matrix)
```

```
Correlation Heatmap :
```



[306]:
```
print('Made by : Nicolas Mrynck')
```

```
Made by : Nicolas Mrynck
```