In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
```

In [11]:

```
d=pd.read_csv('supermarket_sales.csv')
print("Dataset contains {} row and {} colums".format(d.shape[0],d.shape[1]))
```

```
Dataset contains 1000 row and 17 colums
```

In [12]:

```
d.head(6)
```

Out[12]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	To
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.97
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.22
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.52
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.04
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.37
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.61

In [13]:

```
d.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
```

In [14]:

```
d.describe()
```

Out[14]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gr inc
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.220360e-14	11.708
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650

In [19]:

```
d.corr()
```

Out[19]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross marg percenta
Unit price	1.000000e+00	1.077756e-02	6.339621e-01	6.339621e-01	6.339621e-01	-6.998957
Quantity	1.077756e-02	1.000000e+00	7.055102e-01	7.055102e-01	7.055102e-01	-3.849075
Tax 5%	6.339621e-01	7.055102e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.461896e-
Total	6.339621e-01	7.055102e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.408632e-
cogs	6.339621e-01	7.055102e-01	1.000000e+00	1.000000e+00	1.000000e+00	1.439279e-
gross margin percentage	-6.998957e-16	-3.849075e-16	2.461896e-16	2.408632e-16	1.439279e-15	1.000000e+
gross income	6.339621e-01	7.055102e-01	1.000000e+00	1.000000e+00	1.000000e+00	2.461896e-
Rating	-8.777507e-03	-1.581490e-02	-3.644170e-02	-3.644170e-02	-3.644170e-02	2.042714e-

In [21]:

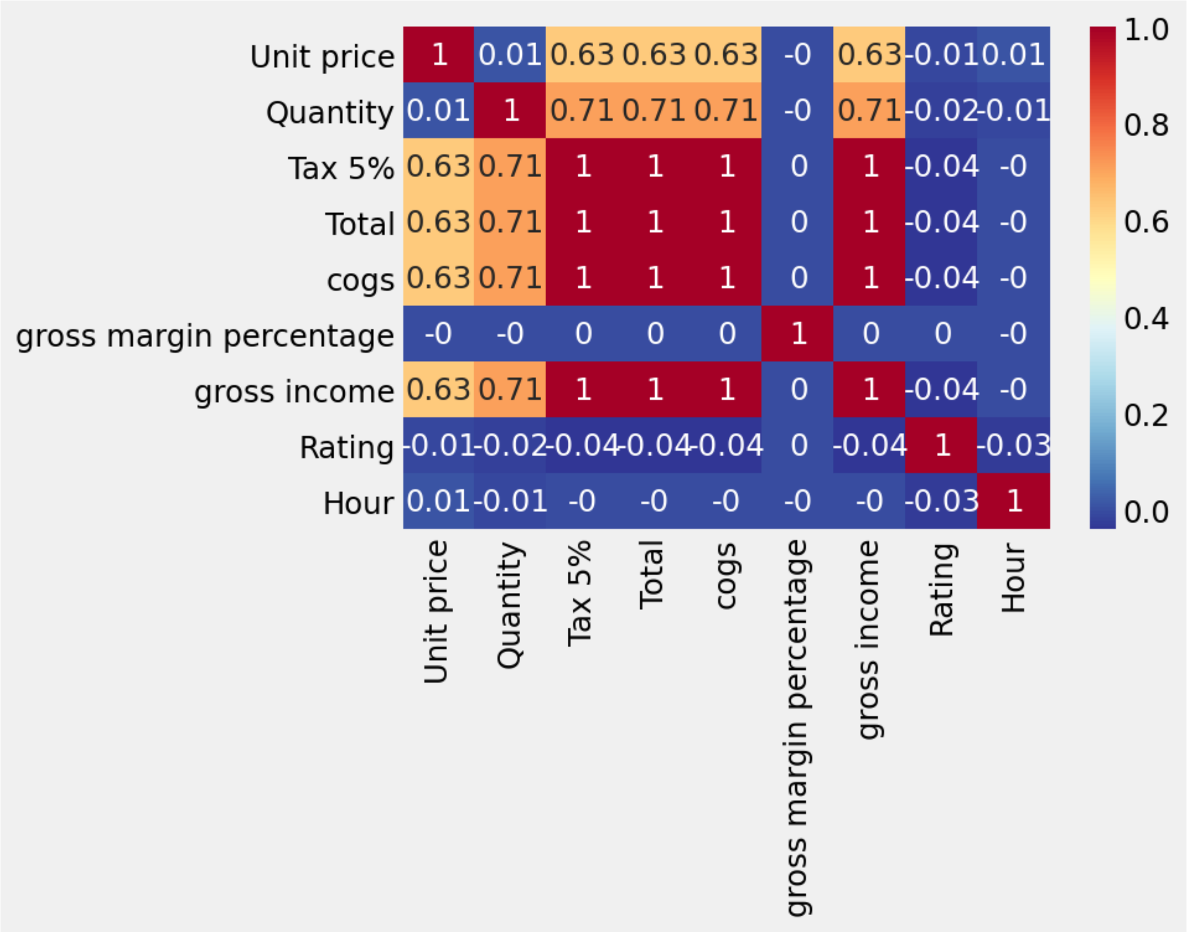
```
np.round(d.corr(),2)
```

Out[21]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
Unit price	1.00	0.01	0.63	0.63	0.63	-0.0	0.63	-0.01
Quantity	0.01	1.00	0.71	0.71	0.71	-0.0	0.71	-0.02
Tax 5%	0.63	0.71	1.00	1.00	1.00	0.0	1.00	-0.04
Total	0.63	0.71	1.00	1.00	1.00	0.0	1.00	-0.04
cogs	0.63	0.71	1.00	1.00	1.00	0.0	1.00	-0.04
gross margin percentage	-0.00	-0.00	0.00	0.00	0.00	1.0	0.00	0.00
gross income	0.63	0.71	1.00	1.00	1.00	0.0	1.00	-0.04
Rating	-0.01	-0.02	-0.04	-0.04	-0.04	0.0	-0.04	1.00

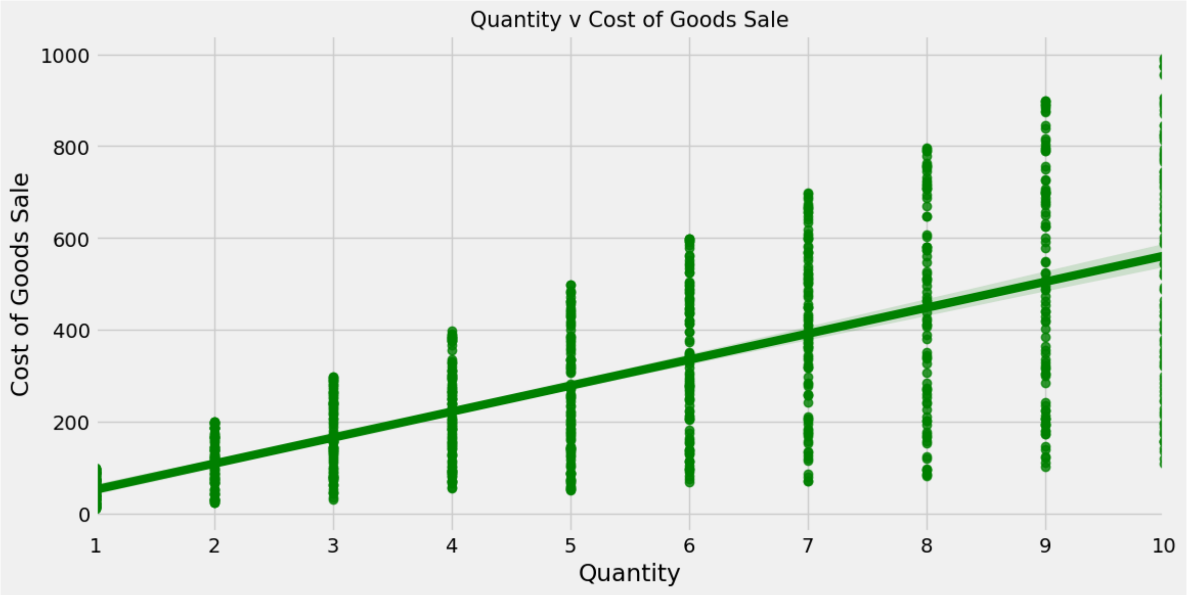
In [59]:

```
plt.figure(dpi=125)
sns.heatmap(np.round(d.corr(),2),annot=True,cmap='RdYlBu_r')
plt.show()
```



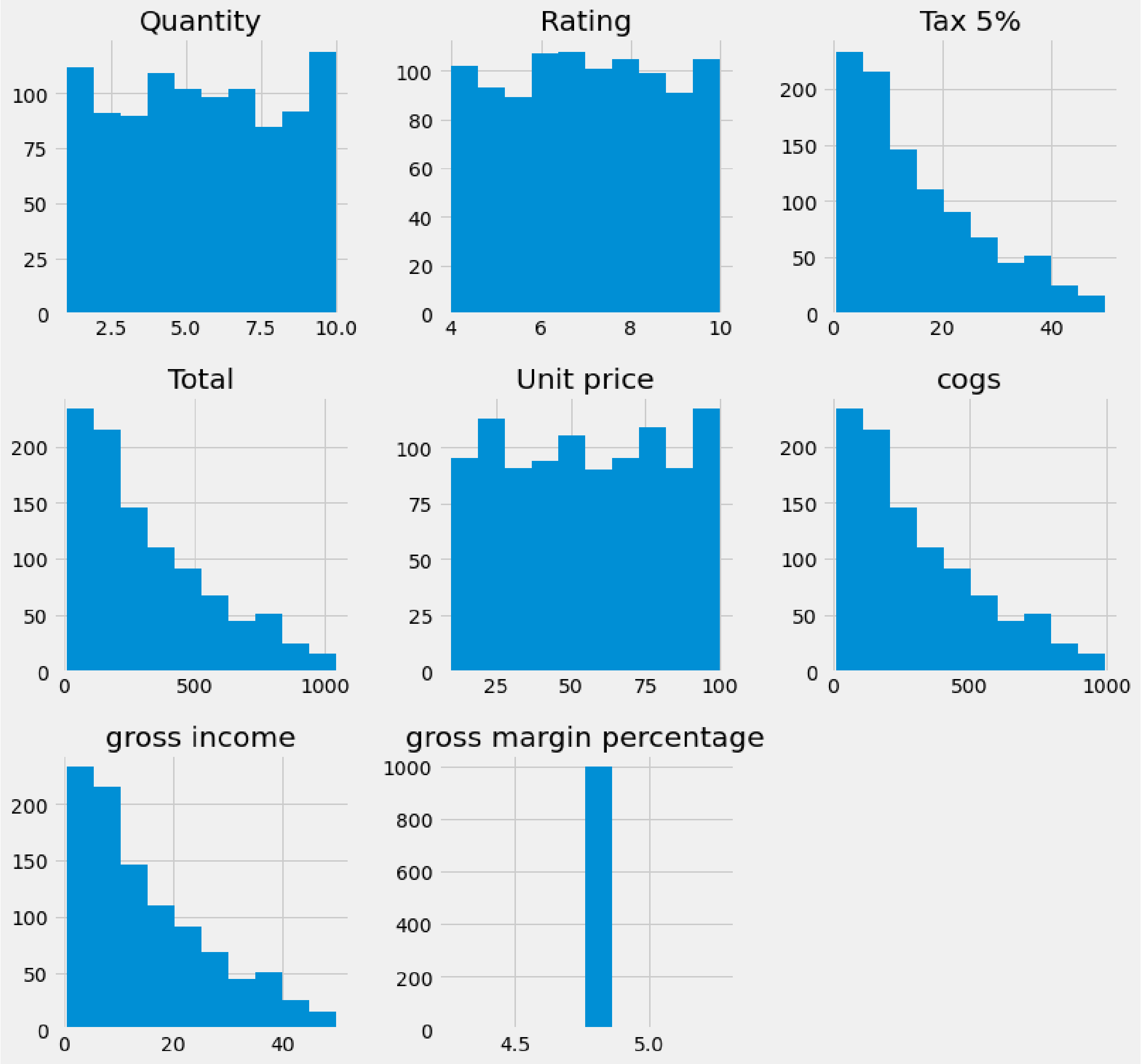
In [24]:

```
plt.figure(figsize=(12,6),dpi=100)
sns.regplot(x='Quantity',y='cogs',data=d,color='green')
plt.xlabel('Quantity')
plt.ylabel('Cost of Goods Sale')
plt.title('Quantity v Cost of Goods Sale',fontsize=15)
plt.show()
```



In [28]:

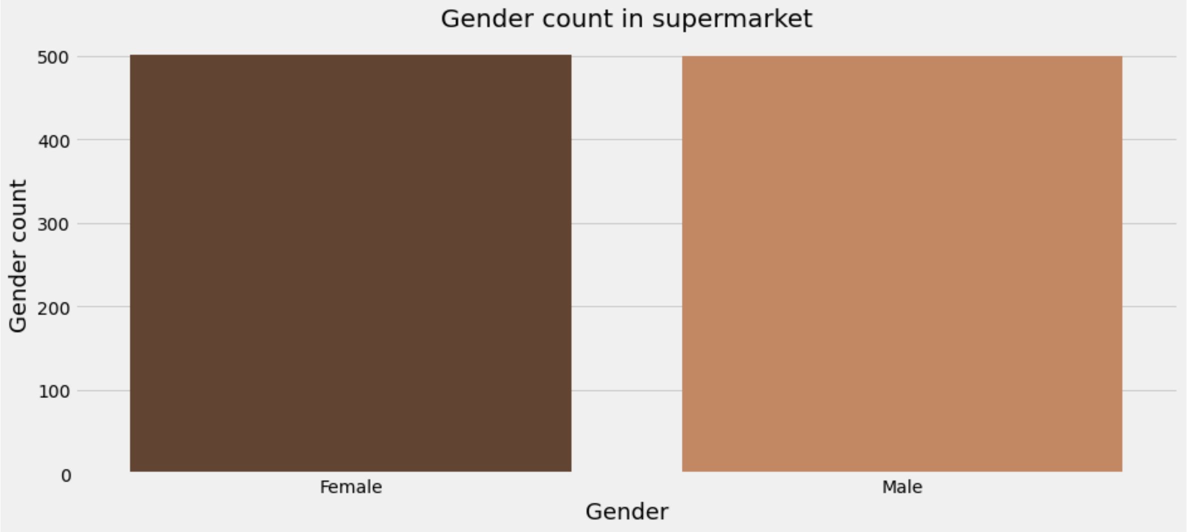
```
d.hist(figsize=(12,12))
plt.show()
```



gender

In [34]:

```
plt.figure(figsize=(14,6))
plt.style.use('fivethirtyeight')
ax= sns.countplot('Gender', data=d , palette = 'copper')
ax.set_xlabel(xlabel= "Gender",fontsize=18)
ax.set_ylabel(ylabel = "Gender count", fontsize = 18)
ax.set_title(label = "Gender count in supermarket", fontsize = 20)
plt.show()

d.groupby(['Gender']). agg({'Total':'sum'})
```

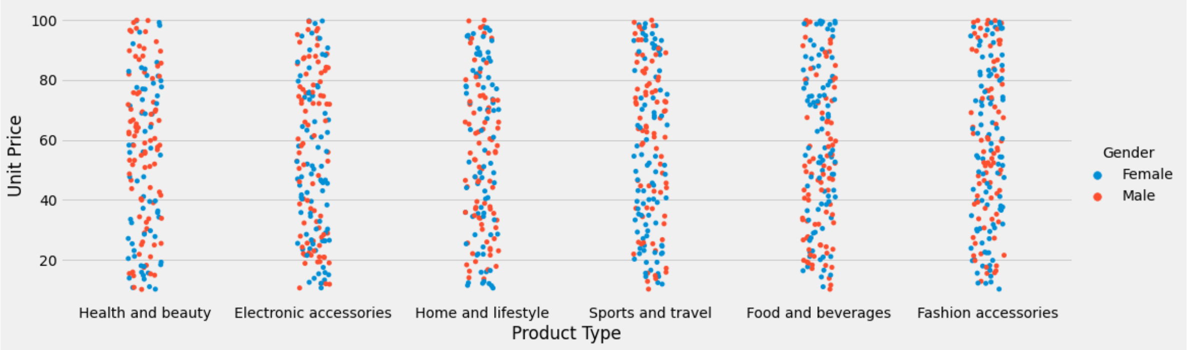


Out[34]:

	Total
Gender	
Female	167882.925
Male	155083.824

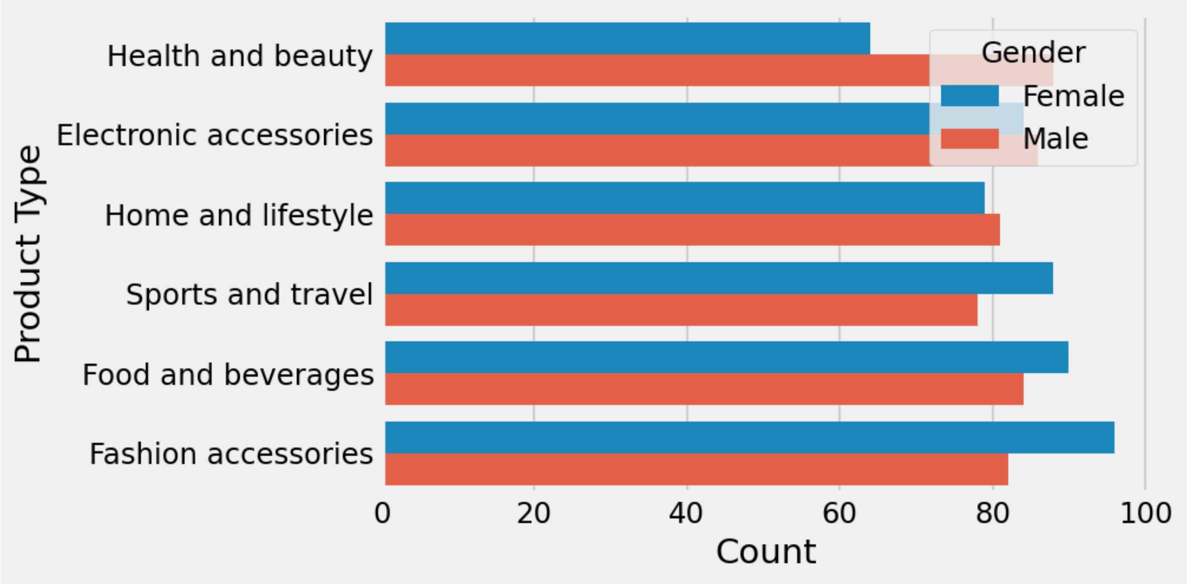
In [38]:

```
sns.catplot(x='Product line',y='Unit price',hue='Gender',data=d,aspect=3)
plt.xlabel('Product Type')
plt.ylabel('Unit Price')
plt.show()
```



In [36]:

```
plt.figure(dpi=125)
sns.countplot(y ='Product line', hue = "Gender", data = d)
plt.xlabel('Count')
plt.ylabel('Product Type')
plt.show()
```



Customer & Branches

In [41]:

```
d.groupby(['Customer type']). agg({'Total':'sum'})
```

Out[41]:

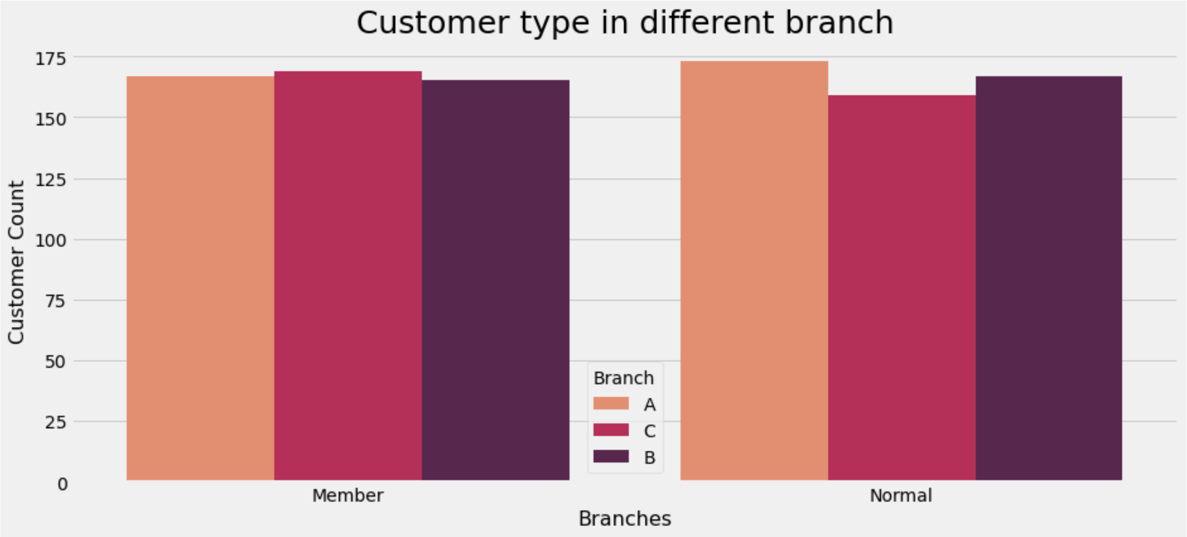
Customer type	Total
Member	164223.444
Normal	158743.305

In [39]:

```
plt.figure(figsize=(14,6))
ax = sns.countplot(x = "Customer type", hue = "Branch", data = d, palette= "rocket_r")
ax.set_title(label = "Customer type in different branch", fontsize = 25)
ax.set_xlabel(xlabel = "Branches", fontsize = 16)
ax.set_ylabel(ylabel = "Customer Count", fontsize = 16)
```

Out[39]:

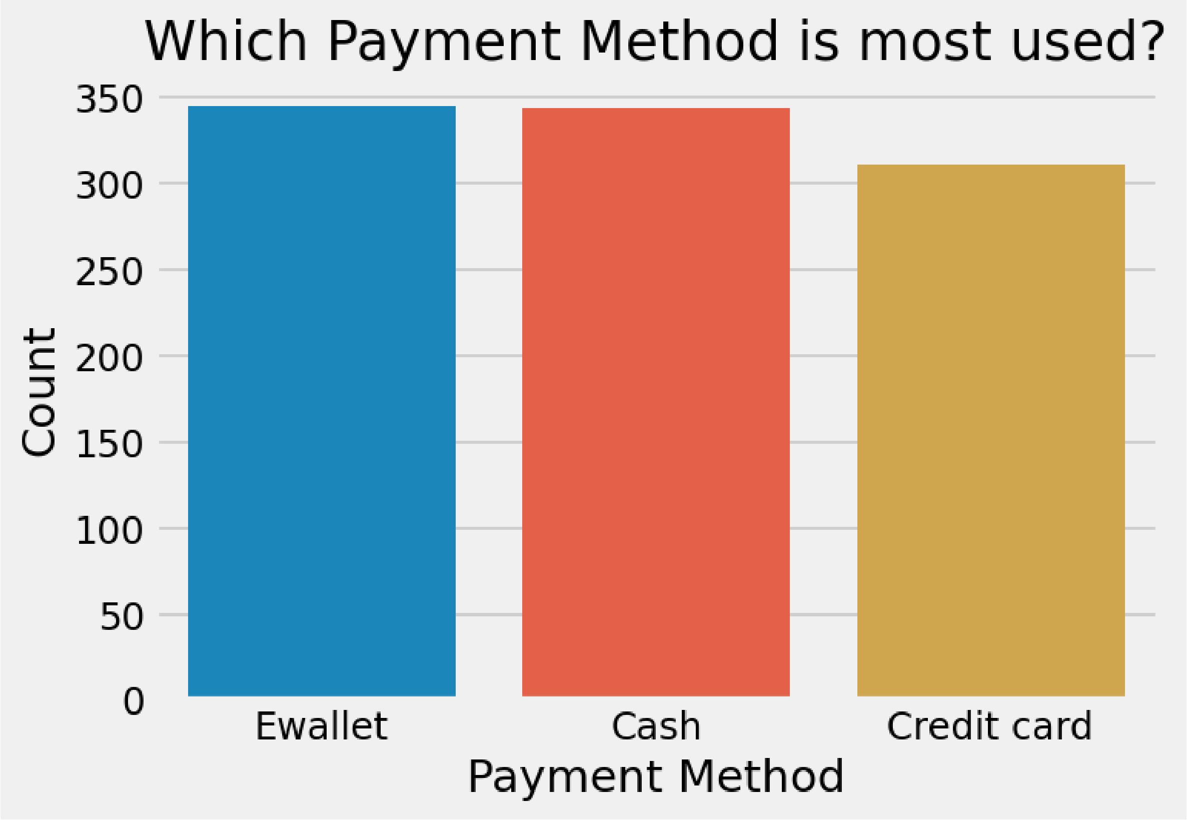
```
Text(0, 0.5, 'Customer Count')
```



In [42]:

```
plt.figure(dpi=125)
sns.countplot(d['Payment'])
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.title('Which Payment Method is most used?')
A,B,C =d.Payment.value_counts()

print('E-wallet -',A)
print('Cash -',B)
print('Credit Card -',C)
plt.show()
```

```
E-wallet - 345
Cash - 344
Credit Card - 311
```

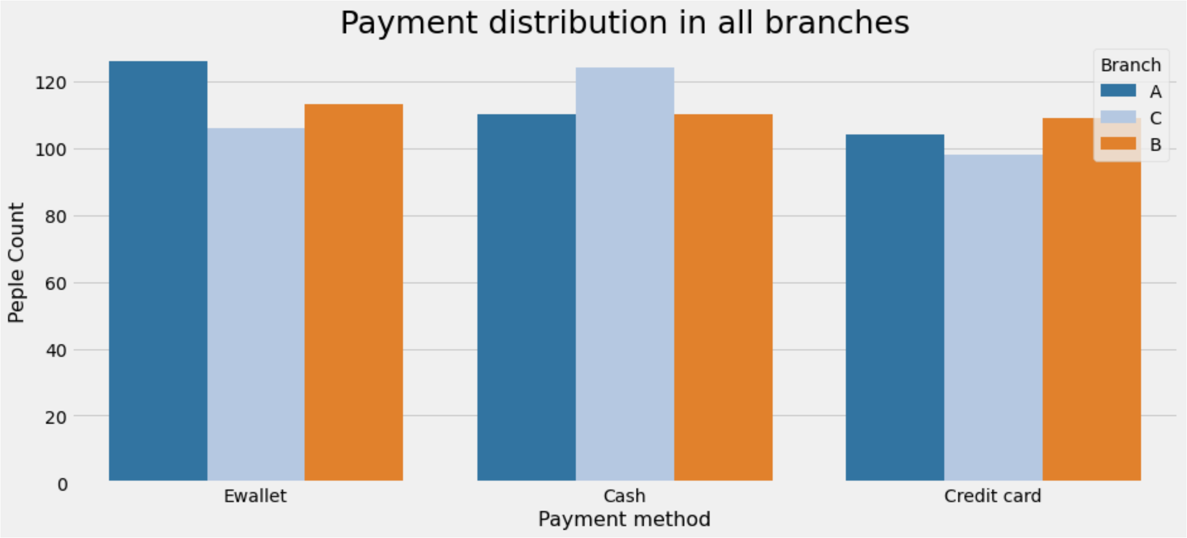


In [43]:

```
plt.figure(figsize = (14,6))
ax = sns.countplot(x="Payment", hue = "Branch", data = d, palette= "tab20")
ax.set_title(label = "Payment distribution in all branches", fontsize= 25)
ax.set_xlabel(xlabel = "Payment method", fontsize = 16)
ax.set_ylabel(ylabel = "Peple Count", fontsize = 16)
```

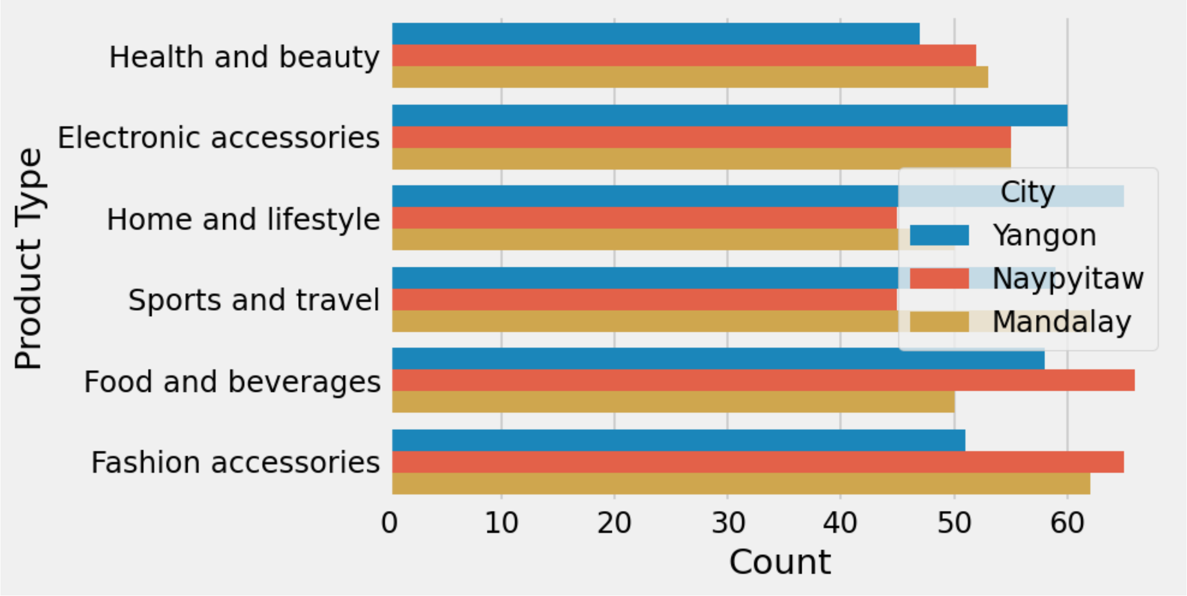
Out[43]:

```
Text(0, 0.5, 'Peple Count')
```



In [44]:

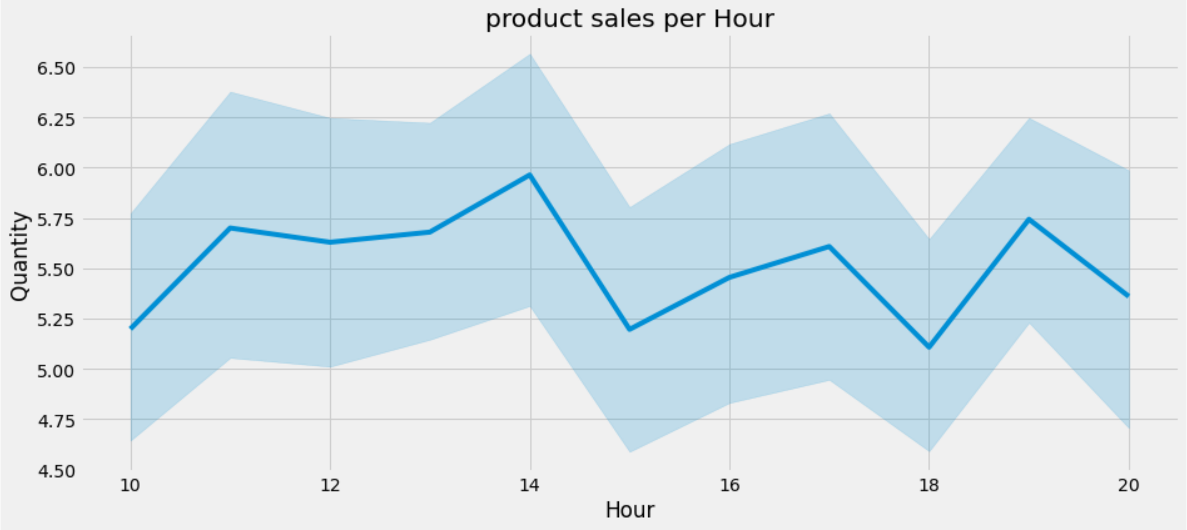
```
plt.figure(dpi=125)
sns.countplot(y ='Product line', hue = "City", data = d)
plt.xlabel('Count')
plt.ylabel('Product Type')
plt.show()
```



Sells time

In [45]:

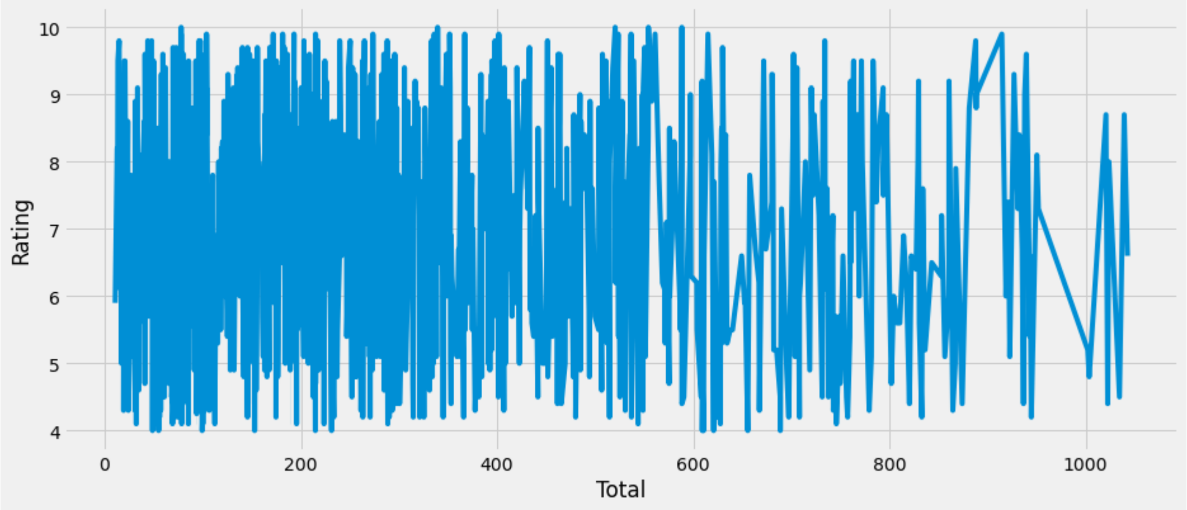
```
d["Time"]= pd.to_datetime(d["Time"])
d["Hour"]= (d["Time"]).dt.hour
plt.figure(figsize=(14,6))
SalesTime = sns.lineplot(x="Hour", y ="Quantity", data = d).set_title("product sales per Ho
```



Rating VS Sales

In [46]:

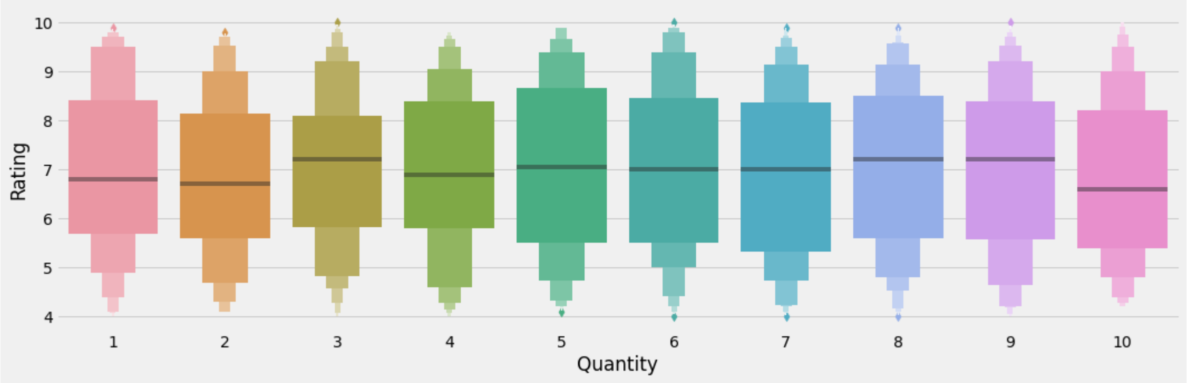
```
plt.figure(figsize=(14,6))
rating_vs_sales = sns.lineplot(x="Total", y= "Rating", data=d)
```



Rating VS Quantity

In [47]:

```
sns.catplot(y ='Rating',x='Quantity', data = d,kind='boxen',aspect=3)
plt.xlabel('Quantity')
plt.ylabel('Rating')
plt.show()
```

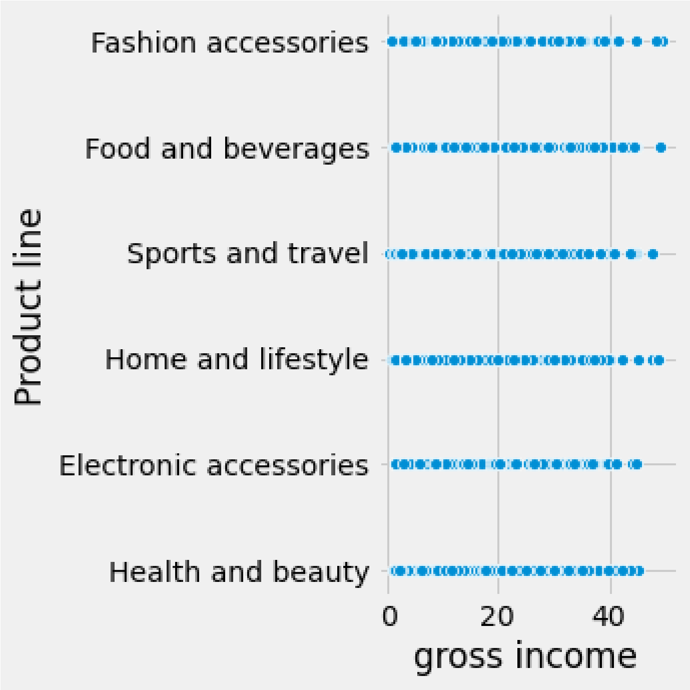


Product and gross income

In [67]:

```
plt.figure(figsize = (5,20),dpi="100")
ax = sns.relplot(y= "Product line", x = "gross income", data = d)
```

```
<Figure size 500x2000 with 0 Axes>
```



PREDICTION

In [51]:

```
x = d.iloc[:,6].values.reshape(-1,1)
y = d.iloc[:,-2].values
print("Display x")
print(x)
print("Display y")
print(y)
```

```
 [30.35]
 [88.67]
 [27.38]
 [62.13]
 [33.98]

 [81.97]
 [16.49]
 [98.21]
 [72.84]
 [58.07]
 [80.79]
 [27.02]
 [21.94]
 [51.36]
 [10.96]
 [53.44]
 [99.56]
 [57.12]
 [99.96]
```

In [52]:

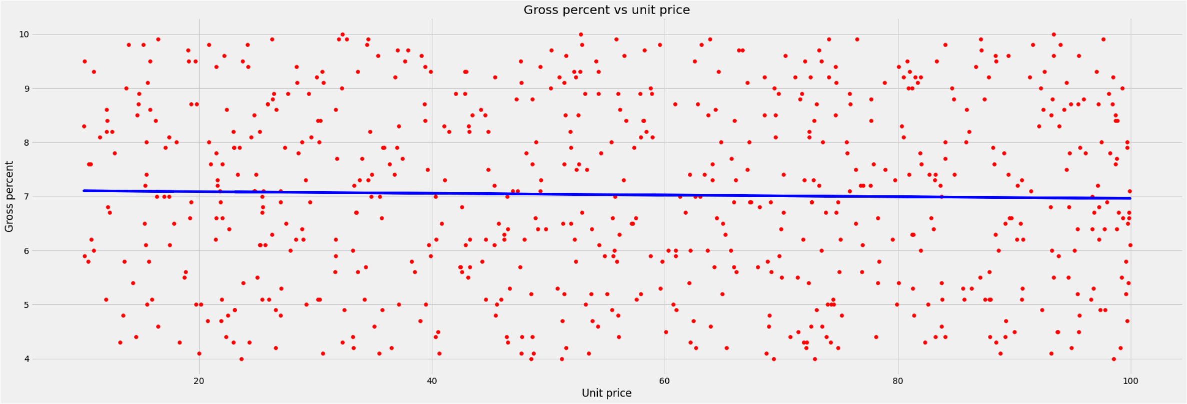
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[52]:

```
LinearRegression()
```

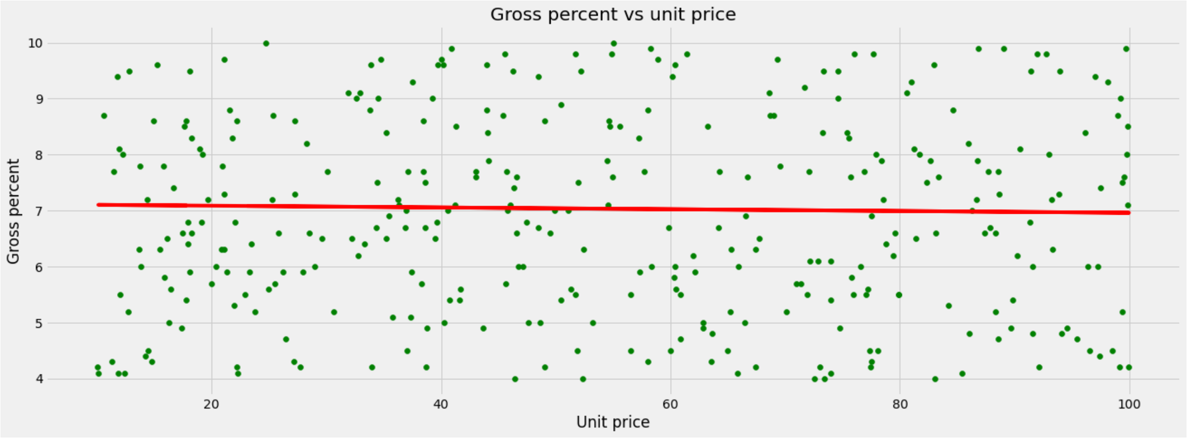
In [56]:

```
plt.figure(figsize=(30,10))
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Gross percent vs unit price')
plt.xlabel('Unit price')
plt.ylabel('Gross percent')
plt.show()
```



In [58]:

```
plt.figure(figsize=(20,7))
plt.scatter(X_test, y_test, color = 'green')
plt.plot(X_train, regressor.predict(X_train), color = 'red')
plt.title('Gross percent vs unit price')
plt.xlabel('Unit price')
plt.ylabel('Gross percent')
plt.show()
```



In [ ]: