```
In [1]: #importing the packages

import pandas as pd
import numpy as np
import seaborn as sns
```

```
In [ ]:
```

```
In [2]: import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [3]: #import Data

sales_df=pd.read_csv("supermarket_sales - Sheet12.csv")
```

```
In [19]: sales_df.head(20)
```

Out[19]:

	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	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	Ewallet	597.73	4.761905	29.8865	4.1
6	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewallet	413.04	4.761905	20.6520	5.8
7	315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.7800	772.3800	2/24/2019	11:38	Ewallet	735.60	4.761905	36.7800	8.0
8	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.6260	76.1460	1/10/2019	17:15	Credit card	72.52	4.761905	3.6260	7.2
9	692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.2260	172.7460	2/20/2019	13:27	Credit card	164.52	4.761905	8.2260	5.9
10	351-62-0822	B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.8960	60.8160	2/6/2019	18:07	Ewallet	57.92	4.761905	2.8960	4.5
11	529-56-3974	B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.1020	107.1420	3/9/2019	17:03	Cash	102.04	4.761905	5.1020	6.8
12	365-64-0515	A	Yangon	Normal	Female	Electronic accessories	46.95	5	11.7375	246.4875	2/12/2019	10:25	Ewallet	234.75	4.761905	11.7375	7.1
13	252-56-2699	A	Yangon	Normal	Male	Food and beverages	43.19	10	21.5950	453.4950	2/7/2019	16:48	Ewallet	431.90	4.761905	21.5950	8.2
14	829-34-3910	A	Yangon	Normal	Female	Health and beauty	71.38	10	35.6900	749.4900	3/29/2019	19:21	Cash	713.80	4.761905	35.6900	5.7
15	299-46-1805	B	Mandalay	Member	Female	Sports and travel	93.72	6	28.1160	590.4360	1/15/2019	16:19	Cash	562.32	4.761905	28.1160	4.5
16	656-95-9349	A	Yangon	Member	Female	Health and beauty	68.93	7	24.1255	506.6355	3/11/2019	11:03	Credit card	482.51	4.761905	24.1255	4.6
17	765-26-6951	A	Yangon	Normal	Male	Sports and travel	72.61	6	21.7830	457.4430	1/1/2019	10:39	Credit card	435.66	4.761905	21.7830	6.9
18	329-62-1586	A	Yangon	Normal	Male	Food and beverages	54.67	3	8.2005	172.2105	1/21/2019	18:00	Credit card	164.01	4.761905	8.2005	8.6
19	319-50-3348	B	Mandalay	Normal	Female	Home and lifestyle	40.30	2	4.0300	84.6300	3/11/2019	15:30	Ewallet	80.60	4.761905	4.0300	4.4

```
In [5]: # using the describe function for the statistiscal summary of the dataset

sales_df.describe()
```

Out[5]:

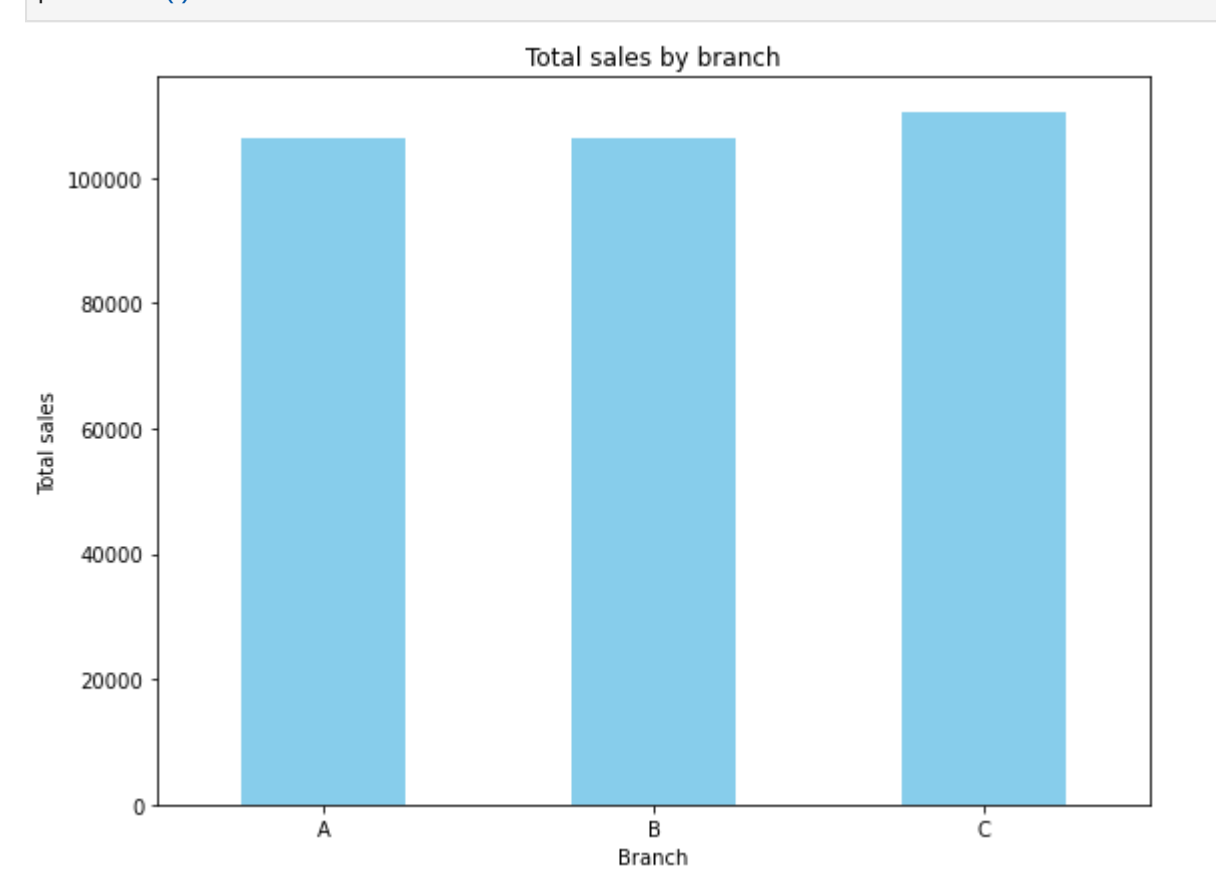
	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000

```
In [7]: # using the describe function for the statistiscal summary of the dataset

sales_df.describe()
```

Out[7]:

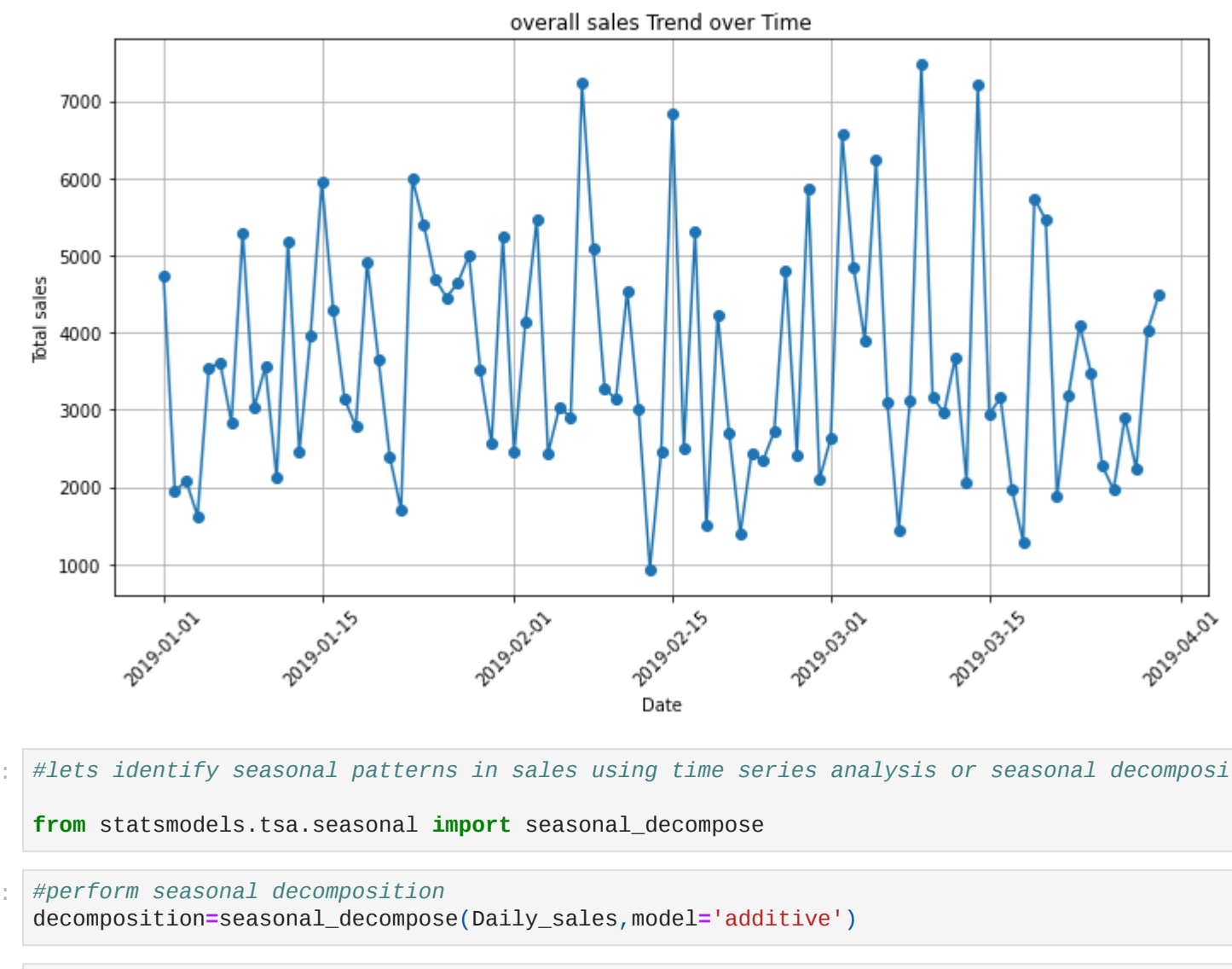
	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1.000000e+03	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000

```
In [8]: #we want to analyze the overall sales trend overtime. but before we do, we will covert the "date" column to datetime

sales_df["Date"]=pd.to_datetime(sales_df["Date"])

#plot the sales by branch
branch_sales=sales_df.groupby('Branch')['Total'].sum()
plt.figure(figsize=(8,6))
branch_sales.plot(kind='bar',color='skyblue')
plt.title('Total sales by branch')
plt.xlabel('Branch')
plt.ylabel('Total sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```




```
In [9]: #Group by date and sum the sales
Daily_sales=sales_df.groupby("Date")["Total"].sum()
```

```
In [10]: #plot the overall sales trend
plt.figure(figsize=(10,6))
plt.plot(Daily_sales.index,Daily_sales.values,marker='o',linestyle='-')
plt.title('overall sales Trend over Time')
plt.xlabel('Date')
plt.ylabel('Total sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

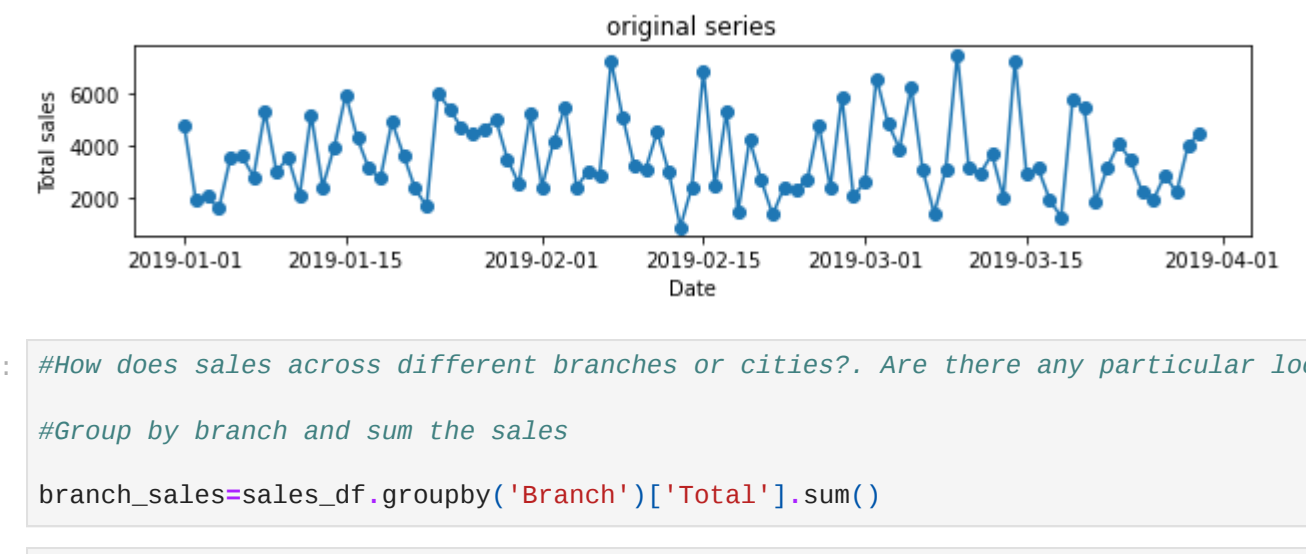



```
In [11]: #lets identify seasonal patterns in sales using time series analysis or seasonal decomposition

from statsmodels.tsa.seasonal import seasonal_decompose
```

```
In [12]: #perform seasonal decomposition
decomposition=seasonal_decompose(Daily_sales,model='additive')
```

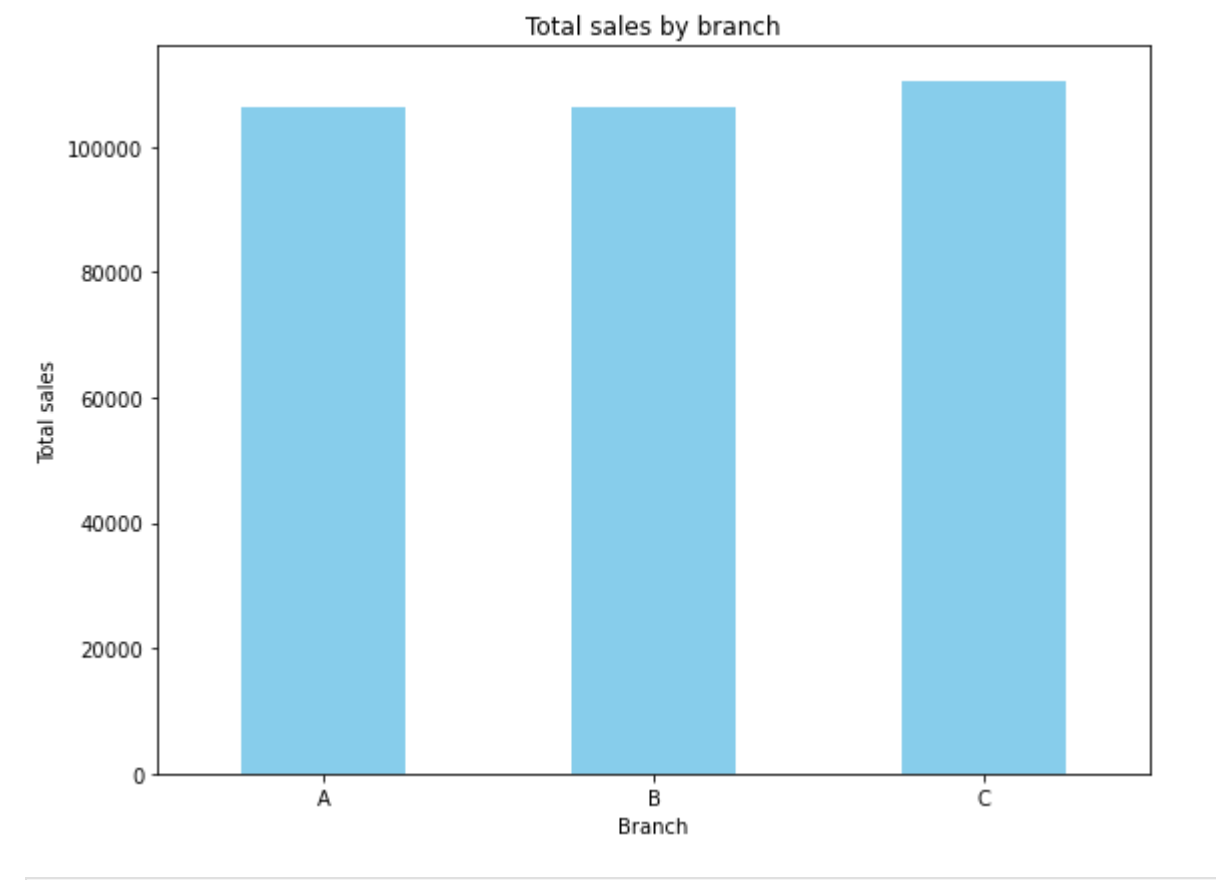
```
In [13]: #plot the seasonal decomposition
plt.figure(figsize=(10,8))

#original series
plt.subplot(4,1,1)
plt.plot(Daily_sales,marker='o',linestyle='-')
plt.title('original series')
plt.xlabel('Date')
plt.ylabel('Total sales')
```

```
Out[13]: Text(0, 0.5, 'Total sales')
```

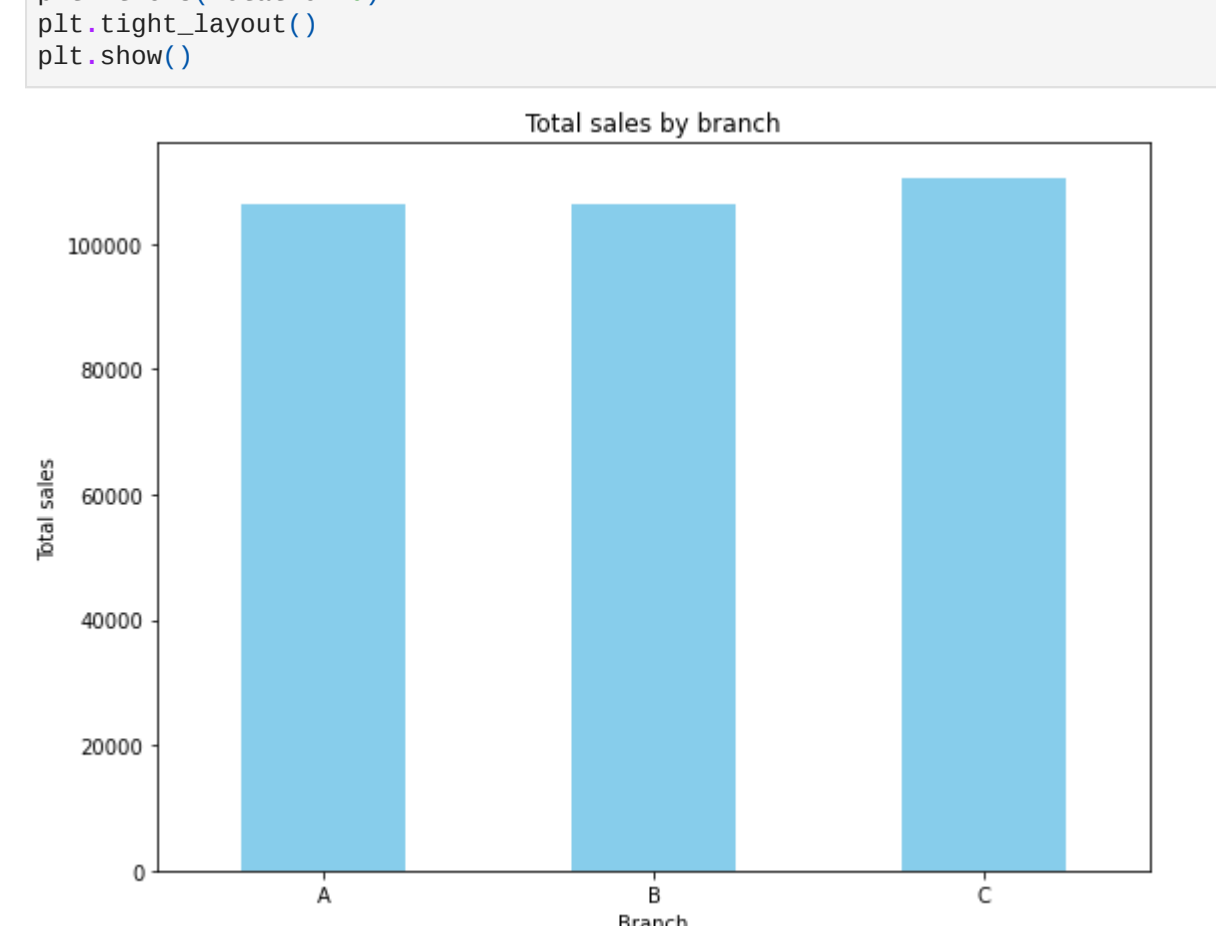



```
In [14]: #How does sales across different branches or cities?. Are there any particular location that perform better than others?

#Group by branch and sum the sales

branch_sales=sales_df.groupby('Branch')['Total'].sum()
```

```
In [15]: #plot the sales by branch

plt.figure(figsize=(8,6))
branch_sales.plot(kind='bar',color='skyblue')
plt.title('Total sales by branch')
plt.xlabel('Branch')
plt.ylabel('Total sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

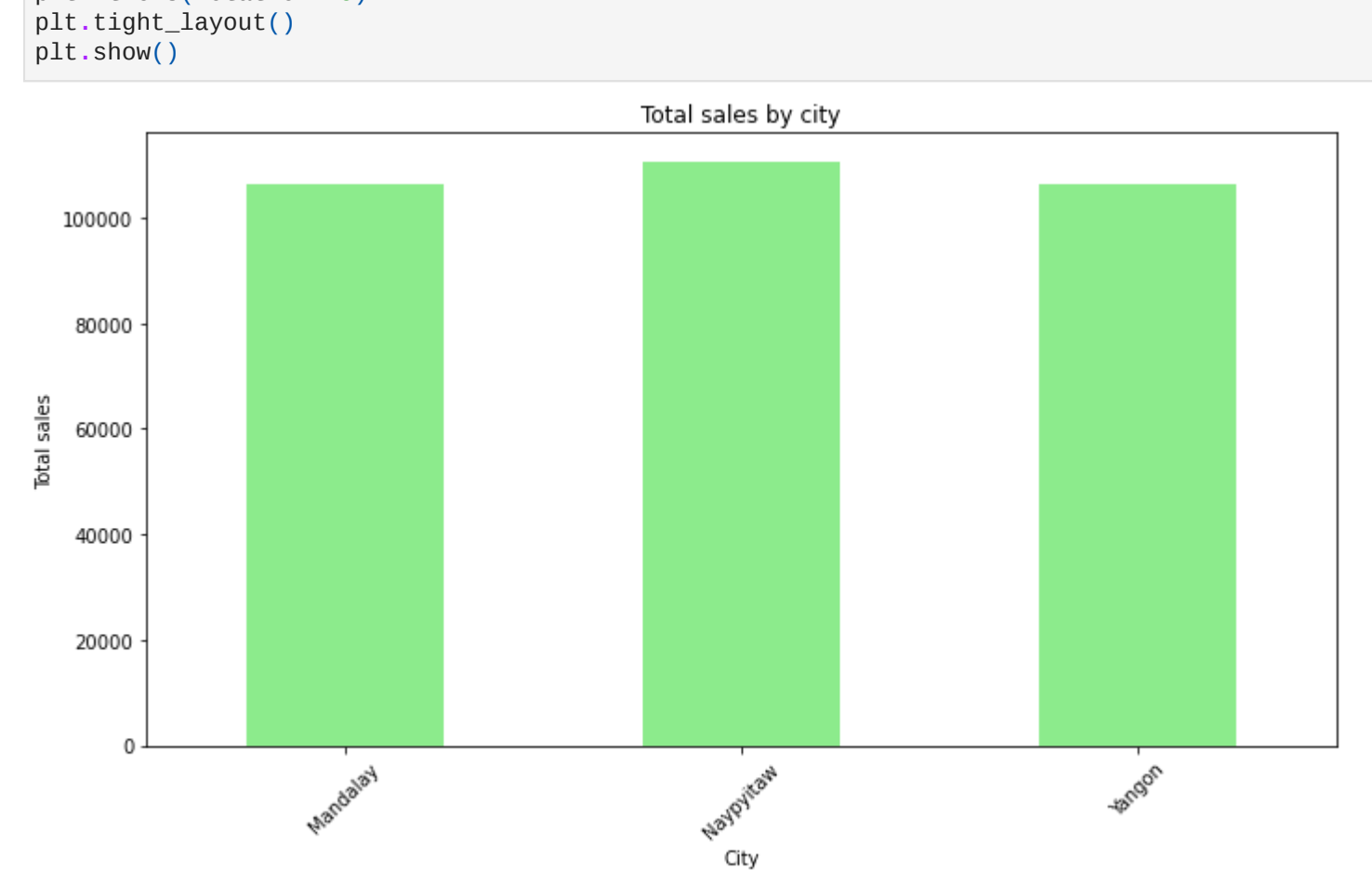



```
In [16]: #plot the sales by branch

plt.figure(figsize=(8,6))
branch_sales.plot(kind='bar',color='skyblue')
plt.title('Total sales by branch')
plt.xlabel('Branch')
plt.ylabel('Total sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```




```
In [18]: #Group by city and sum the sales


city_sales=sales_df.groupby('City')['Total'].sum()
```

```
In [19]: #plot the sales by city

plt.figure(figsize=(10,6))
city_sales.plot(kind='bar',color='lightgreen')
plt.title('Total sales by city')
plt.xlabel('City')
plt.ylabel('Total sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



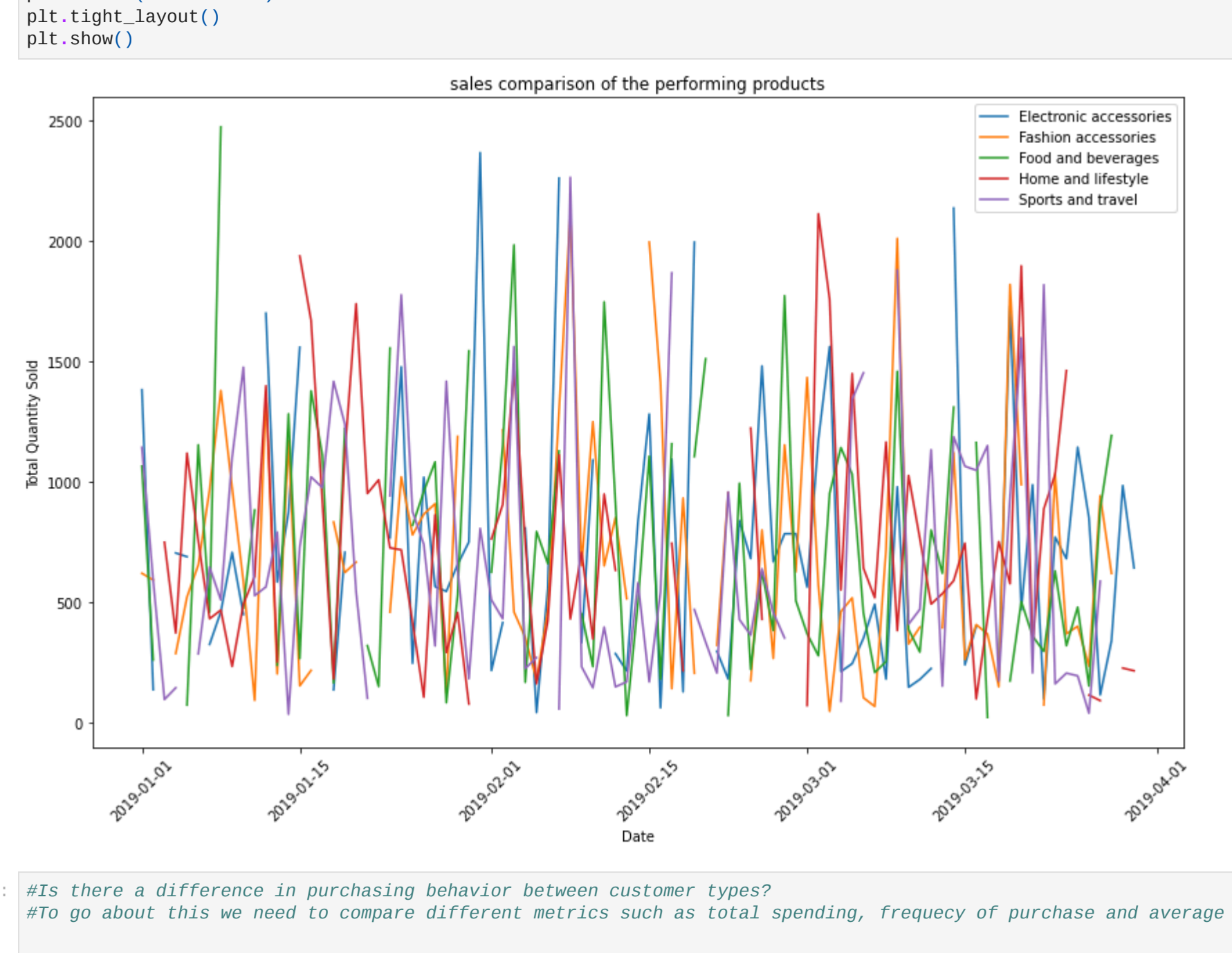

```
In [20]: #What are top selling product lines and how do their sales compare?

product_sales=sales_df.groupby('Product line')['Quantity'].sum().sort_values(ascending=False)
```

```
In [21]: #What are top selling product lines and how do their sales compare?

product_sales=sales_df.groupby('Product line')['Quantity'].sum().sort_values(ascending=False)
```

plot the top performing products

top_products=product_sales.head(10) plt.figure(figsize=(10,6)) top_products.plot(kind='bar',color='orange') plt.title('Top performing products') plt.xlabel('product line') plt.ylabel('Total Quantity sold') plt.xticks(rotation=45) plt.tight_layout() plt.show()

```
In [22]: #compare sales of top performing products
product_sales=sales_df.groupby('Product line')['Total'].sum().sort_values(ascending=False)

#select the top five products for comparion
top_products=product_sales.head(5)

#filter data for top products
top_products_data=sales_df[sales_df['Product line'].isin(top_products.index)]

#groupby product and date,sum the sales
product_date_sales=top_products_data.groupby(['Product line','Date']).agg({'Total':'sum'}).reset_index()

#pivot the data for plotting
pivot_data=product_date_sales.pivot(index='Date',columns='Product line',values='Total')

#plotting the sales of the top products over time
plt.figure(figsize=(12,8))
for product in pivot_data.columns:
    plt.plot(pivot_data.index,pivot_data[product], label=product)
plt.title('sales comparison of the performing products')
plt.xlabel('Date')
plt.ylabel('Total Quantity Sold')
plt.legend(loc='upper right')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
In [23]: #Is there a difference in purchasing behavior between customer types?
#To go about this we need to compare different metrics such as total spending, frequecy of purchase and average transaction value


#Group by customer type and calculate different metrics such as total spending, frequecy of purchase and average transaction value

customer_type_analysis=sales_df.groupby('Customer type').agg({'Invoice ID':'count',
                                                              'Total':['sum','mean'],
                                                              'Quantity':'sum',
                                                              'Product line':'nunique'
                                                              }).reset_index()

#Rename columns for clarity
customer_type_analysis.columns=['Customer type','Total Invoices','Total spending','Average spending per transaction','Total quantity purchased','unique product

#plotting
plt.figure(figsize=(12,8))

#Total spending comparison

plt.subplot(2,2,1)
plt.bar(customer_type_analysis['Customer type'],customer_type_analysis['Total spending'],color=['skyblue','lightgreen'])
plt.title('Total spending by customer Type')
plt.xlabel('Customer Type')
plt.ylabel('Total spending')

#Total purchase comparison
plt.subplot(2,2,2)
plt.bar(customer_type_analysis['Customer type'],customer_type_analysis['Total quantity purchased'],color=['skyblue','lightgreen'])
plt.title('Total purchase by customer Type')
plt.xlabel('Customer Type')
plt.ylabel('Total purchase')

#Average Transaction value comparion
plt.subplot(1,3,3)
plt.bar(customer_type_analysis['Customer type'],customer_type_analysis['Average spending per transaction'],color=['skyblue','lightgreen'])
plt.title('Average spending per transaction by customer type')
plt.xlabel('Customer Type')
plt.ylabel('Average spending per transaction')
plt.tight_layout()
plt.show()
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_11364\2769887947.py:40: UserWarning: tight_layout not applied: number of columns in subplot specifications must be multiples of one another.
  plt.tight_layout()
```


```
In [23]: #To datermine the gender with the highest purchasing power
gender_gross_income=sales_df.groupby('Gender')['gross income'].sum()

#Determine the gender with the highest purchasing power
highest_purchasing_gender=gender_gross_income.idxmax()

#Get the total gross income for the gender with the highest purchasing power
highest_purchasing_power=gender_gross_income.max()

print("Gender with the highest purchasing power:",highest_purchasing_gender)
print("Total gross income for the gender with the highest purchasing power:",highest_purchasing_gender)
```

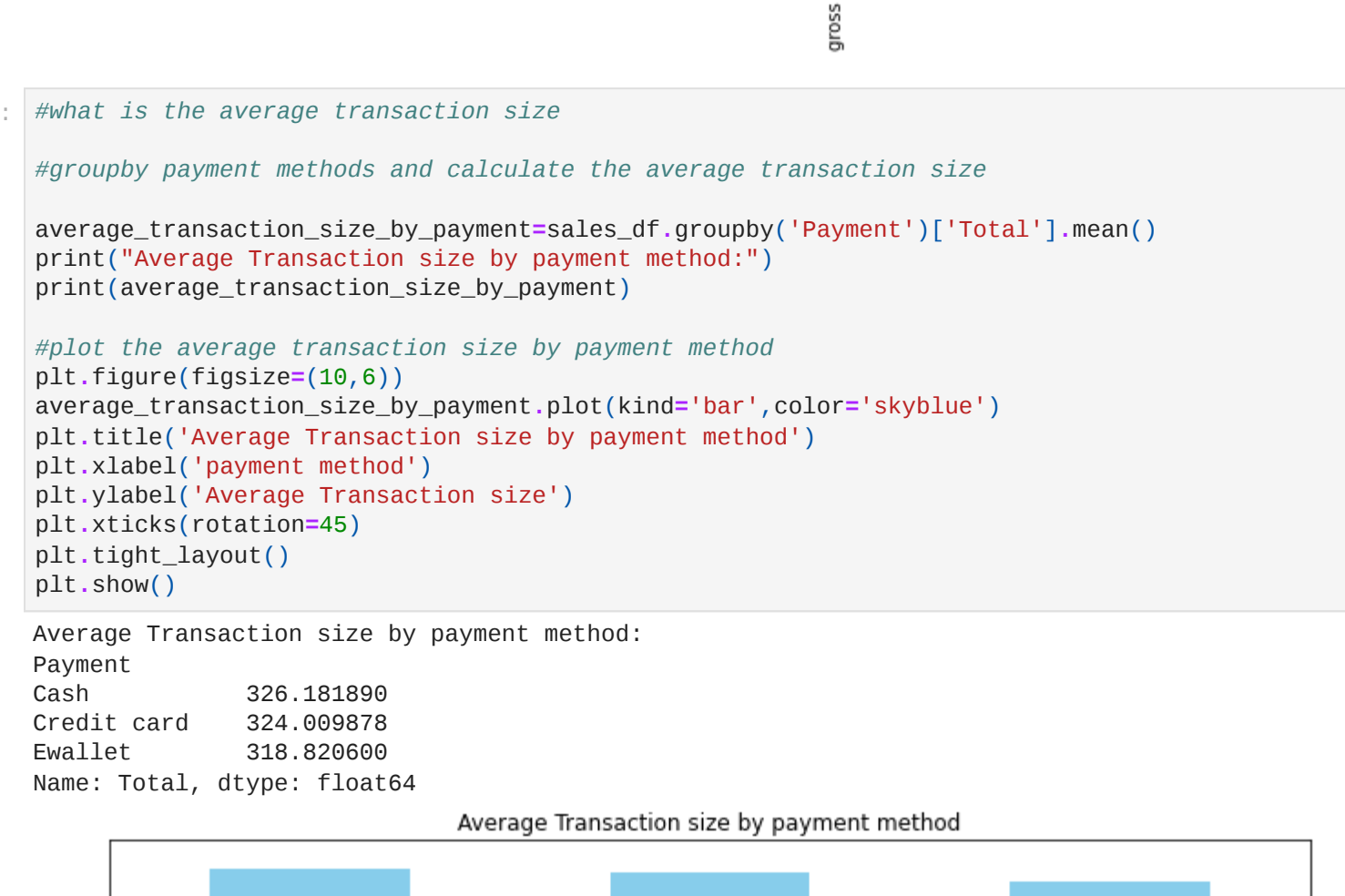
```
Gender with the highest purchasing power: Female
Total gross income for the gender with the highest purchasing power: Female
```

```
In [24]:  #To check for correlation between gender and purchasing behaviour

#using correlation analysis or hypothesis testing

#correlation analysis
correlation_matrix=sales_df.corr()
print(correlation_matrix)

#plot correlation matrix
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm',fmt="2f")
plt.title('correlation matrix')
plt.show()
```

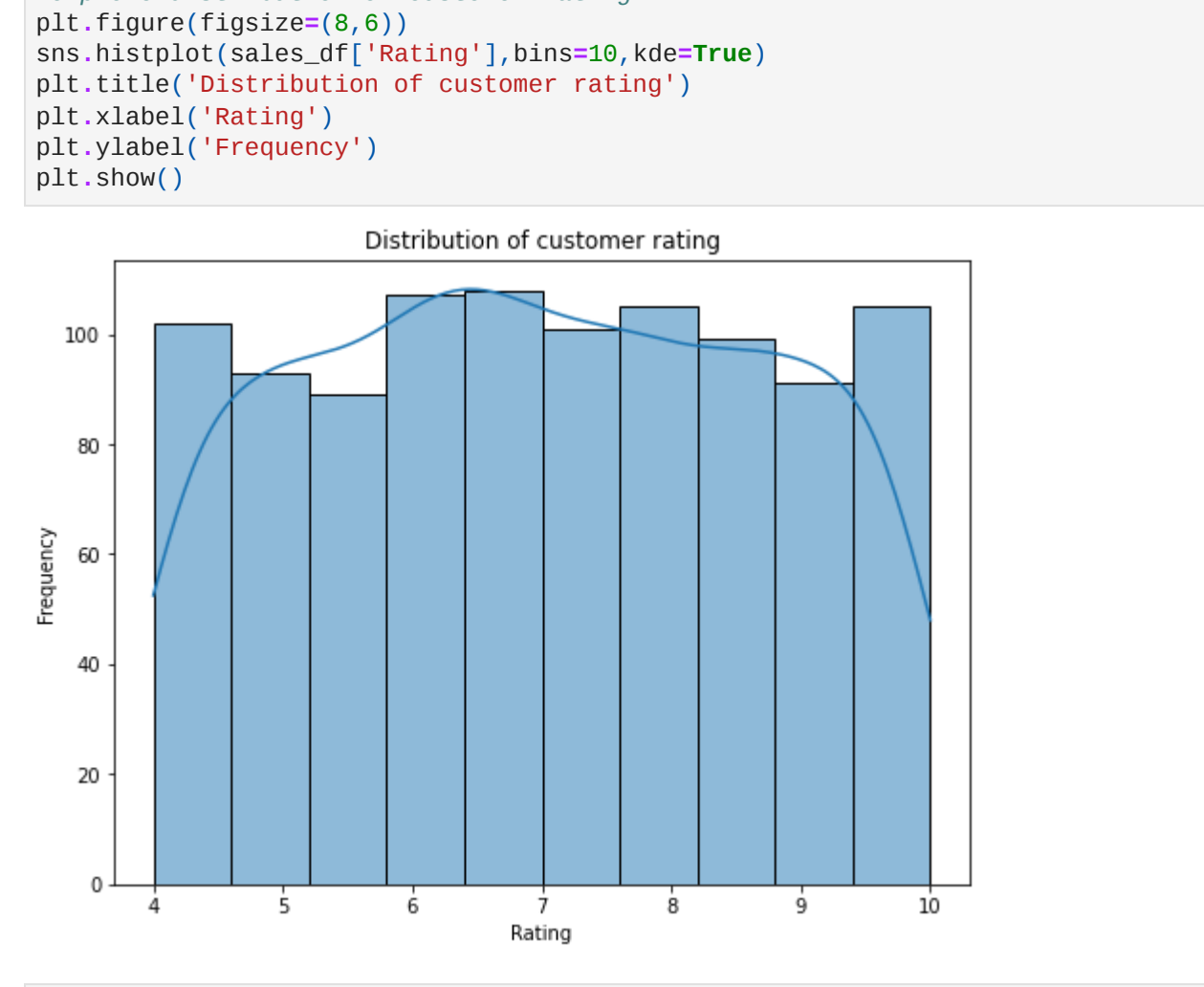
```
                         Unit price  Quantity    Tax 5%     Total      cogs  \
Unit price                 1.000000  0.010778  0.633962  0.633962  0.633962
Quantity                   0.010778  1.000000  0.705510  0.705510  0.705510
Tax 5%                     0.633962  0.705510  1.000000  1.000000  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
```


```
In [26]: #what is the average transaction size

#groupby payment methods and calculate the average transaction size

average_transaction_size_by_payment=sales_df.groupby('Payment')['Total'].mean()
print("Average Transaction size by payment method:")
print(average_transaction_size_by_payment)

#plot the average transaction size by payment method
plt.figure(figsize=(10,6))
average_transaction_size_by_payment.plot(kind='bar',color='skyblue')
plt.title('Average Transaction size by payment method')
plt.xlabel('payment method')
plt.ylabel('Average Transaction size')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Average Transaction size by payment method:
Payment
Cash           326.181890
Credit card    324.009878
Ewallet        318.820600
Name: Total, dtype: float64
```




```
In [27]: #How can we identify the patterns in customer rating based on factors such as product line, branch location or time of purchase

#explore distribution of customer rating
plt.figure(figsize=(8,6))
sns.histplot(sales_df['Rating'],bins=10,kde=True)
plt.title('Distribution of customer rating')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```




```
In [ ]:
```

```
In [ ]:
```