In [53]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as mtick
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.preprocessing import PolynomialFeatures
```

In [54]:

```
supermarket = pd.read_csv('/home/amybirdee/hobby_projects/supermarket_sales/supermarket_sales.csv', delimiter = ',')
```

In [55]:

```
supermarket.head()
```

Out[55]:

	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	548
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634

In [56]:

```
#renaming columns
supermarket = supermarket.rename(columns = {'Invoice ID': 'invoice_id', 'Branch': 'branch', 'City': 'city', 'Customer type':
                                            'customer_type', 'Gender': 'gender', 'Product line': 'product_line', 'Unit price':
                                            'unit_price', 'Quantity': 'quantity', 'Tax 5%': 'tax_5%', 'Total': 'total_price',
                                            'Date': 'date', 'Time': 'time', 'Payment': 'payment_method', 'cogs':
                                            'cost_of_goods_sold', 'gross margin percentage': 'gross_margin_percentage',
                                            'gross income': 'gross_income', 'Rating': 'customer_rating'})
```

In [57]:

```
supermarket.head()
```

Out[57]:

	invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7

In [58]:

```
#datatypes all seem fine apart from date and time which are objects but we won't need t
hese fields in this analysis
#no null values to deal with
supermarket.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_price             1000 non-null   float64
 10  date                    1000 non-null   object
 11  time                    1000 non-null   object
 12  payment_method          1000 non-null   object
 13  cost_of_goods_sold      1000 non-null   float64
 14  gross_margin_percentage 1000 non-null   float64
 15  gross_income            1000 non-null   float64
 16  customer_rating         1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [59]:

```
#getting an overview of the data - invoice_id is a unique column so can be dropped. The
re are three branches in three
#cities so city column can also be dropped
supermarket.describe(include = 'all')
```

Out[59]:

	invoice_id	branch	city	customer_type	gender	product_line	unit_price	c
count	1000	1000	1000	1000	1000	1000	1000.000000	1000
unique	1000	3	3	2	2	6	NaN	
top	288-62-1085	A	Yangon	Member	Female	Fashion accessories	NaN	
freq	1	340	340	501	501	178	NaN	
mean	NaN	NaN	NaN	NaN	NaN	NaN	55.672130	5
std	NaN	NaN	NaN	NaN	NaN	NaN	26.494628	2
min	NaN	NaN	NaN	NaN	NaN	NaN	10.080000	1
25%	NaN	NaN	NaN	NaN	NaN	NaN	32.875000	3
50%	NaN	NaN	NaN	NaN	NaN	NaN	55.230000	5
75%	NaN	NaN	NaN	NaN	NaN	NaN	77.935000	8
max	NaN	NaN	NaN	NaN	NaN	NaN	99.960000	10

In [60]:

```
#dropping unnecessary columns
supermarket = supermarket.drop(['invoice_id', 'city', 'date', 'time'], axis = 1)
```

In [61]:

```
supermarket.head()
```

Out[61]:

	branch	customer_type	gender	product_line	unit_price	quantity	tax_5%	total_price	pa
0	A	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	
1	C	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	
2	A	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	
3	A	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	
4	A	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	

In [62]:

```
#grouping by branch and customer type to see which branch/type generated most sales
sales_customer_type = supermarket.groupby(['branch', 'customer_type']).total_price.sum
().round().to_frame().reset_index()
sales_customer_type
```

Out[62]:

	branch	customer_type	total_price
0	A	Member	53637.0
1	A	Normal	52563.0
2	B	Member	53705.0
3	B	Normal	52493.0
4	C	Member	56881.0
5	C	Normal	53687.0

In [63]:

```
#converting values to integer to remove the 0 decimal - this will be needed for chart l
abels
sales_customer_type['total_price'] = sales_customer_type['total_price'].apply(int)
```

Exploratory data analysis

In [64]:

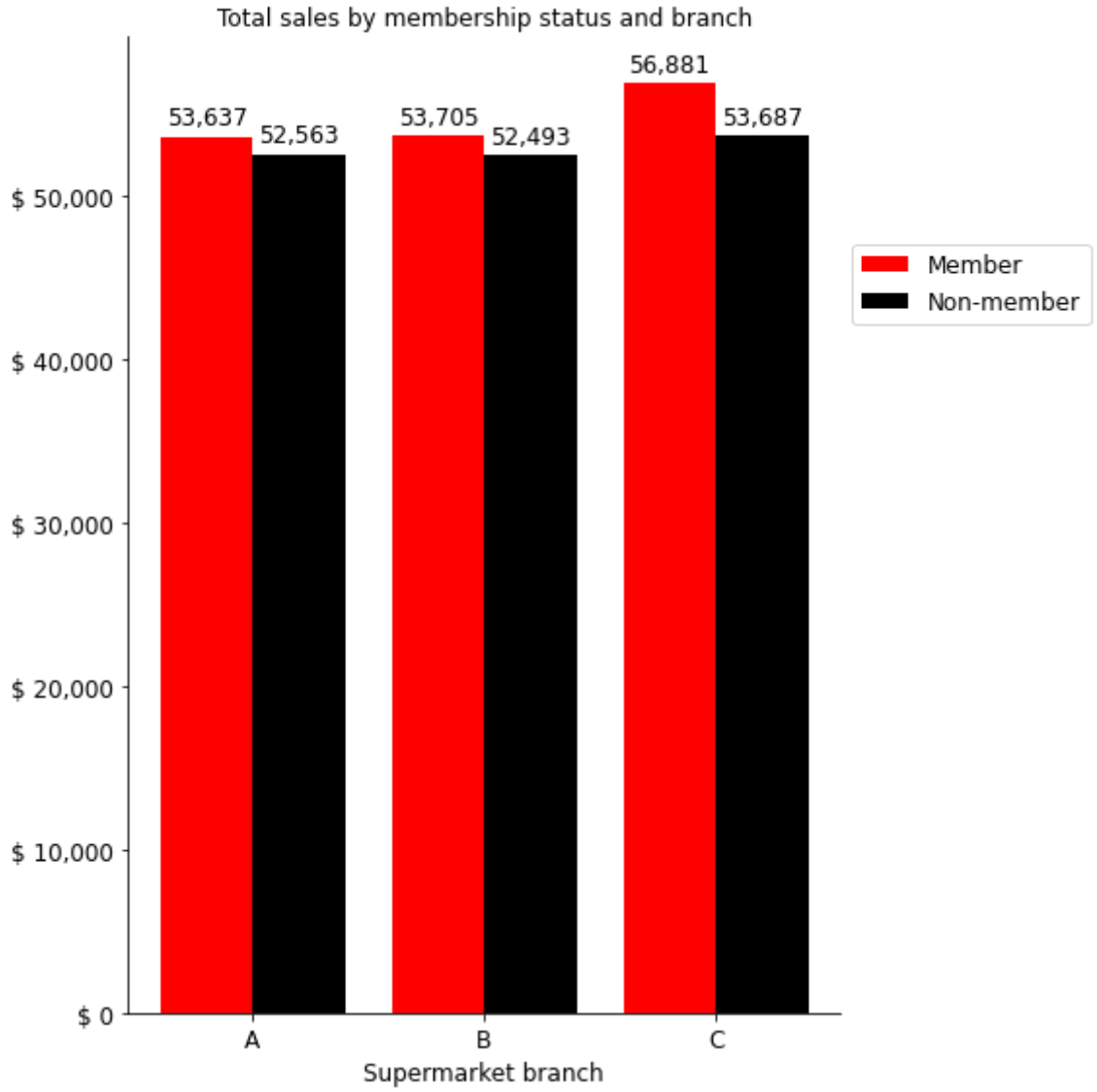
```
#creating pivot table showing supermarket members and non-members
customer_type_pivot = sales_customer_type.pivot_table(columns = 'customer_type',
                                                      index = 'branch',
                                                      values = 'total_price').reset_index()
customer_type_pivot
```

Out[64]:

customer_type	branch	Member	Normal
0	A	53637	52563
1	B	53705	52493
2	C	56881	53687

In [65]:

```
#plotting data on barchart. Majority sales by members in all cases with largest sales v
olume at branch C
num_bars = 3
width = 0.4

branch_bars = np.arange(num_bars)

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot()

bar_1 = ax.bar(branch_bars, customer_type_pivot.Member, width, color = 'red', label =
'Member')
bar_2 = ax.bar(branch_bars + width, customer_type_pivot.Normal, width, color = 'black',
label = 'Non-member')

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

#setting y_tick values to have commas to separate 1000s and also a $ sign in front
value_format = '${x: ,.0f}'
tick = mtick.StrMethodFormatter(value_format)
ax.yaxis.set_major_formatter(tick)
plt.yticks(fontsize = 12)

#setting x_ticks
ax.set_xticks(branch_bars + width / 2)
ax.set_xticklabels(customer_type_pivot.branch, fontsize = 12)
ax.set_xlabel('Supermarket branch', fontsize = 12)


#adding a label to each bar
def autolabel(bars):
    for bar in bars:
        height = bar.get_height()
        #the '{:,}' command adds a thousand separator to the labels
        ax.annotate('{:,}'.format(height),\
                    xy = (bar.get_x() + bar.get_width() / 2, height), \
                    #shows label position on x and y axis
                    xytext = (0, 3), \
                    textcoords = 'offset points', ha = 'center', va = 'bottom', fontsize
= 12)


autolabel(bar_1)
autolabel(bar_2)

#bbox_to_anchor shifts the legend along the x and y axis
ax.legend((bar_1, bar_2), ('Member', 'Non-member'), bbox_to_anchor = (1.0, 0.8), fontsi
ze = 12)
plt.title('Total sales by membership status and branch', fontsize = 12, y = 1.0)

plt.tight_layout()

plt.savefig('sales_customer_type')
```


Total sales by membership status and branch
53,637
52,563
53,705
52,493
56,881
53,687
$ 50,000
$ 40,000
$ 30,000
$ 20,000
$ 10,000
$ 0
A
B
C
Supermarket branch
Member
Non-member

In [66]:

```
#grouping by branch and gender to see which branch/gender generated most sales
sales_gender = supermarket.groupby(['branch', 'gender']).total_price.sum().round().to_f
rame().reset_index()
sales_gender
```

Out[66]:

	branch	gender	total_price
0	A	Female	53269.0
1	A	Male	52931.0
2	B	Female	52928.0
3	B	Male	53269.0
4	C	Female	61685.0
5	C	Male	48883.0

In [67]:

```
#converting values to integer to remove the 0 decimal - this will be needed for chart l
abels
sales_gender['total_price'] = sales_gender['total_price'].apply(int)
```

In [68]:

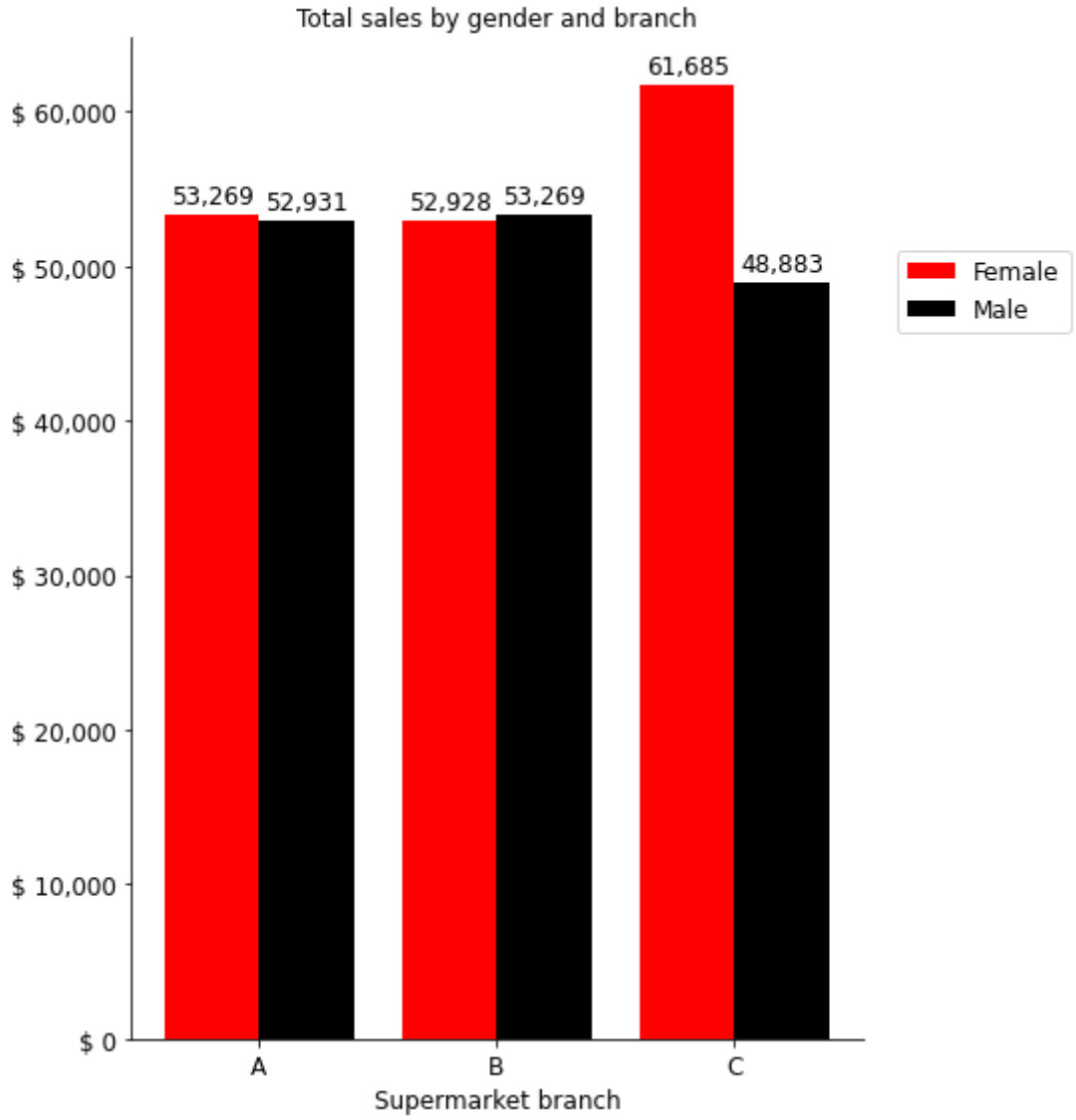
```
#creating pivot table showing males/female breakdown
gender_pivot = sales_gender.pivot_table(columns = 'gender',
                                        index = 'branch',
                                        values = 'total_price').reset_index()
gender_pivot
```

Out[68]:

gender	branch	Female	Male
0	A	53269	52931
1	B	52928	53269
2	C	61685	48883

In [69]:

```
#plotting data on barchart. Females are by far the biggest spenders in Branch C
num_bars = 3
width = 0.4

branch_bars = np.arange(num_bars)

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot()

bar_1 = ax.bar(branch_bars, gender_pivot.Female, width, color = 'red', label = 'Female'
)
bar_2 = ax.bar(branch_bars + width, gender_pivot.Male, width, color = 'black', label =
'Male')

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

#setting y_tick values to have commas to separate 1000s and also a $ sign in front
value_format = '${x: ,.0f}'
tick = mtick.StrMethodFormatter(value_format)
ax.yaxis.set_major_formatter(tick)
plt.yticks(fontsize = 12)

#setting x_ticks
ax.set_xticks(branch_bars + width / 2)
ax.set_xticklabels(gender_pivot.branch, fontsize = 12)
ax.set_xlabel('Supermarket branch', fontsize = 12)

#using the function created in first chart to add labels to each bar
autolabel(bar_1)
autolabel(bar_2)

#bbox_to_anchor shifts the legend along the x and y axis
ax.legend((bar_1, bar_2), ('Female', 'Male'), bbox_to_anchor = (1.3, 0.8), fontsize = 1
2)
plt.title('Total sales by gender and branch', fontsize = 12, y = 1.0)

plt.tight_layout()

plt.savefig('sales_gender')
```


Total sales by gender and branch
$ 60,000
$ 50,000
$ 40,000
$ 30,000
$ 20,000
$ 10,000
$ 0
53,269
52,931
52,928
53,269
61,685
48,883
A
B
C
Supermarket branch
Female
Male

In [70]:

```
#grouping by branch and product line to see which branch/product generated most sales
sales_product = supermarket.groupby(['branch', 'product_line']).total_price.sum().round
().to_frame().reset_index()
sales_product
```

Out[70]:

	branch	product_line	total_price
0	A	Electronic accessories	18317.0
1	A	Fashion accessories	16333.0
2	A	Food and beverages	17163.0
3	A	Health and beauty	12598.0
4	A	Home and lifestyle	22417.0
5	A	Sports and travel	19373.0
6	B	Electronic accessories	17051.0
7	B	Fashion accessories	16413.0
8	B	Food and beverages	15215.0
9	B	Health and beauty	19981.0
10	B	Home and lifestyle	17549.0
11	B	Sports and travel	19988.0
12	C	Electronic accessories	18969.0
13	C	Fashion accessories	21560.0
14	C	Food and beverages	23767.0
15	C	Health and beauty	16615.0
16	C	Home and lifestyle	13896.0
17	C	Sports and travel	15762.0

In [71]:

```
#converting values to integer to remove the 0 decimal - this will be needed for chart l
abels if added
sales_product['total_price'] = sales_product['total_price'].apply(int)
```

In [72]:

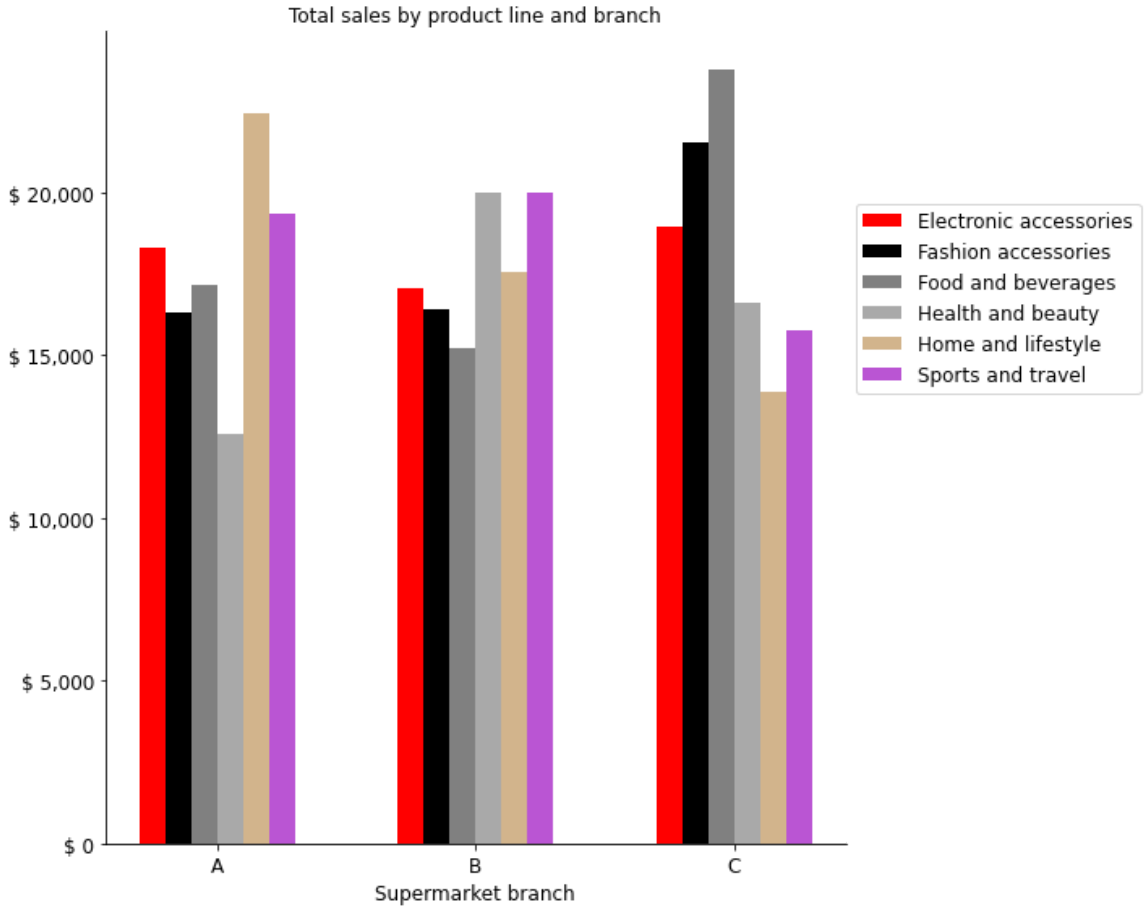
```
#creating pivot table showing product breakdown
product_pivot = sales_product.pivot_table(columns = 'product_line',
                                          index = 'branch',
                                          values = 'total_price').reset_index()
product_pivot
```

Out[72]:

product_line	branch	Electronic accessories	Fashion accessories	Food and beverages	Health and beauty	Home and lifestyle	Sports and travel
0	A	18317	16333	17163	12598	22417	19373
1	B	17051	16413	15215	19981	17549	19988
2	C	18969	21560	23767	16615	13896	15762

In [73]:

```
#plotting data on barchart. Food is biggest seller at Branch C and home and lifestyle i
n branch A
num_bars = 3
width = 0.1

branch_bars = np.arange(num_bars)

fig = plt.figure(figsize = (10,8))
ax = fig.add_subplot()

bar_1 = ax.bar(branch_bars, product_pivot['Electronic accessories'], width, color = 're
d')
bar_2 = ax.bar(branch_bars + 0.1, product_pivot['Fashion accessories'], width, color =
'black')
bar_3 = ax.bar(branch_bars + 0.2, product_pivot['Food and beverages'], width, color =
'grey')
bar_4 = ax.bar(branch_bars + 0.3, product_pivot['Health and beauty'], width, color = 'd
arkgrey')
bar_5 = ax.bar(branch_bars + 0.4, product_pivot['Home and lifestyle'], width, color =
'tan')
bar_6 = ax.bar(branch_bars + 0.5, product_pivot['Sports and travel'], width, color = 'm
ediumorchid')

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

#setting y_tick values to have commas to separate 1000s and also a $ sign in front
value_format = '${x: ,.0f}'
tick = mtick.StrMethodFormatter(value_format)
ax.yaxis.set_major_formatter(tick)
plt.yticks(fontsize = 12)

#setting x_ticks
ax.set_xticks(branch_bars + (width * 5) / 2)
ax.set_xticklabels(product_pivot.branch, fontsize = 12)
ax.set_xlabel('Supermarket branch', fontsize = 12)


#bbox_to_anchor shifts the legend along the x and y axis
ax.legend((bar_1, bar_2, bar_3, bar_4, bar_5, bar_6), \
          ('Electronic accessories', 'Fashion accessories', 'Food and beverages', 'Heal
th and beauty', \
          'Home and lifestyle', 'Sports and travel'), bbox_to_anchor = (1.0, 0.8), font
size = 12)
plt.title('Total sales by product line and branch', fontsize = 12, y = 1.0)

plt.tight_layout()
plt.savefig('sales_product')
```


Total sales by product line and branch
$ 20,000
$ 15,000
$ 10,000
$ 5,000
$ 0
A
B
C
Supermarket branch
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel

In [74]:

```
#grouping by branch and product line to see which branch/product generated most sale co
unts
sales_product_count = supermarket.groupby(['branch', 'product_line']).quantity.sum().ro
und().to_frame().reset_index()
sales_product_count
```

Out[74]:

	branch	product_line	quantity
0	A	Electronic accessories	322
1	A	Fashion accessories	263
2	A	Food and beverages	313
3	A	Health and beauty	257
4	A	Home and lifestyle	371
5	A	Sports and travel	333
6	B	Electronic accessories	316
7	B	Fashion accessories	297
8	B	Food and beverages	270
9	B	Health and beauty	320
10	B	Home and lifestyle	295
11	B	Sports and travel	322
12	C	Electronic accessories	333
13	C	Fashion accessories	342
14	C	Food and beverages	369
15	C	Health and beauty	277
16	C	Home and lifestyle	245
17	C	Sports and travel	265

In [75]:

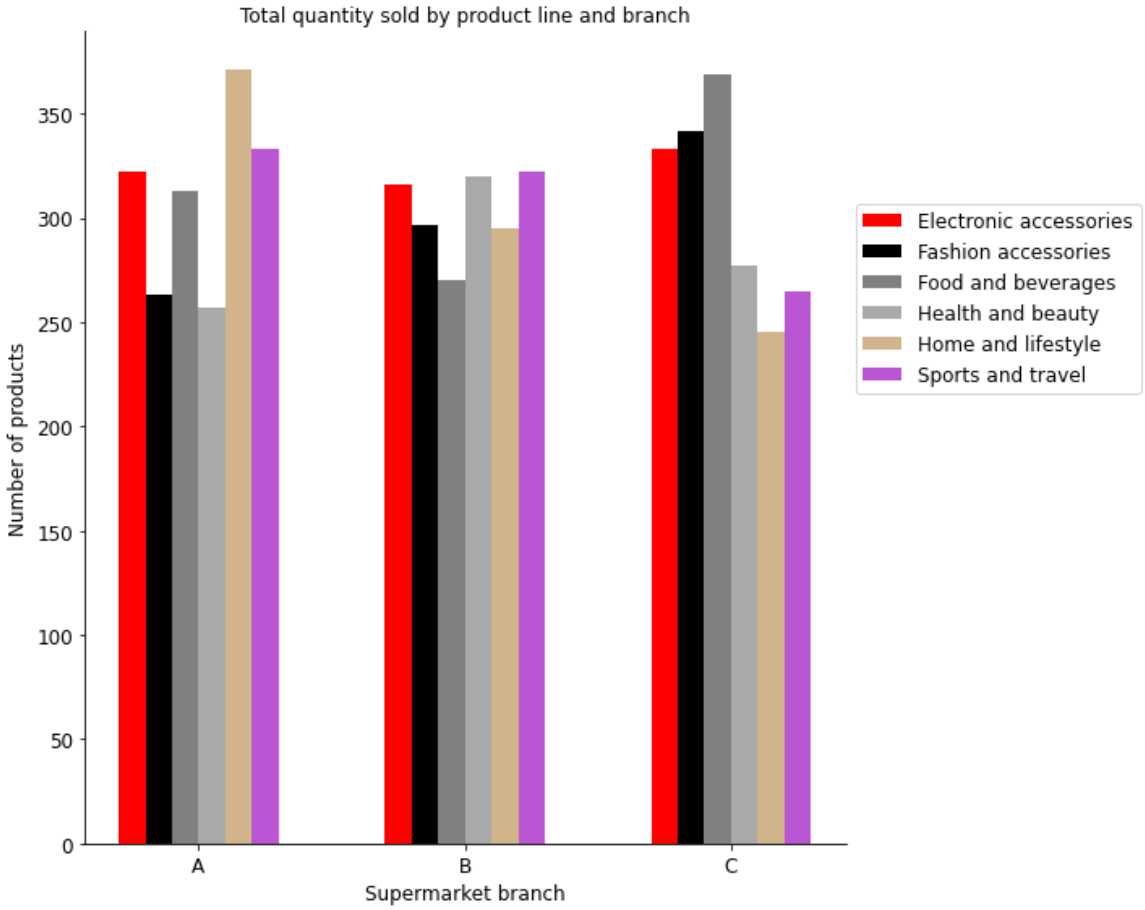
```
#creating pivot table showing product count breakdown
product_pivot_count = sales_product_count.pivot_table(columns = 'product_line',
                                                      index = 'branch',
                                                      values = 'quantity').reset_index()
product_pivot_count
```

Out[75]:

product_line	branch	Electronic accessories	Fashion accessories	Food and beverages	Health and beauty	Home and lifestyle	Sports and travel
0	A	322	263	313	257	371	333
1	B	316	297	270	320	295	322
2	C	333	342	369	277	245	265

In [76]:

```
#plotting data on barchart. Food is biggest seller at Branch C and home and lifestyle i
n branch A - same as sales volume
#chart

num_bars = 3
width = 0.1

branch_bars = np.arange(num_bars)

fig = plt.figure(figsize = (10,8))
ax = fig.add_subplot()

bar_1 = ax.bar(branch_bars, product_pivot_count['Electronic accessories'], width, color
= 'red')
bar_2 = ax.bar(branch_bars + 0.1, product_pivot_count['Fashion accessories'], width, co
lor = 'black')
bar_3 = ax.bar(branch_bars + 0.2, product_pivot_count['Food and beverages'], width, col
or = 'grey')
bar_4 = ax.bar(branch_bars + 0.3, product_pivot_count['Health and beauty'], width, colo
r = 'darkgrey')
bar_5 = ax.bar(branch_bars + 0.4, product_pivot_count['Home and lifestyle'], width, col
or = 'tan')
bar_6 = ax.bar(branch_bars + 0.5, product_pivot_count['Sports and travel'], width, colo
r = 'mediumorchid')

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.yticks(fontsize = 12)
plt.ylabel('Number of products', fontsize = 12)

#setting x_ticks
ax.set_xticks(branch_bars + (width * 5) / 2)
ax.set_xticklabels(product_pivot_count.branch, fontsize = 12)
ax.set_xlabel('Supermarket branch', fontsize = 12)


#bbox_to_anchor shifts the legend along the x and y axis
ax.legend((bar_1, bar_2, bar_3, bar_4, bar_5, bar_6), \
          ('Electronic accessories', 'Fashion accessories', 'Food and beverages', 'Heal
th and beauty', \
          'Home and lifestyle', 'Sports and travel'), bbox_to_anchor = (1.0, 0.8), font
size = 12)
plt.title('Total quantity sold by product line and branch', fontsize = 12, y = 1.0)

plt.tight_layout()
plt.savefig('sales_product_count')
```


Total quantity sold by product line and branch
Number of products
0
50
100
150
200
250
300
350
A
B
C
Supermarket branch
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel

In [77]:

```
#grouping by branch and payment method to see which branch/method generated most sales
sales_payment_method = supermarket.groupby(['branch', 'payment_method']).total_price.su
m().round().to_frame().reset_index()
sales_payment_method
```

Out[77]:

	branch	payment_method	total_price
0	A	Cash	33781.0
1	A	Credit card	33095.0
2	A	Ewallet	39324.0
3	B	Cash	35339.0
4	B	Credit card	37345.0
5	B	Ewallet	33513.0
6	C	Cash	43086.0
7	C	Credit card	30327.0
8	C	Ewallet	37155.0

In [78]:

```
#converting values to integer to remove the 0 decimal - this will be needed for chart l
abels
sales_payment_method['total_price'] = sales_payment_method['total_price'].apply(int)
```

In [79]:

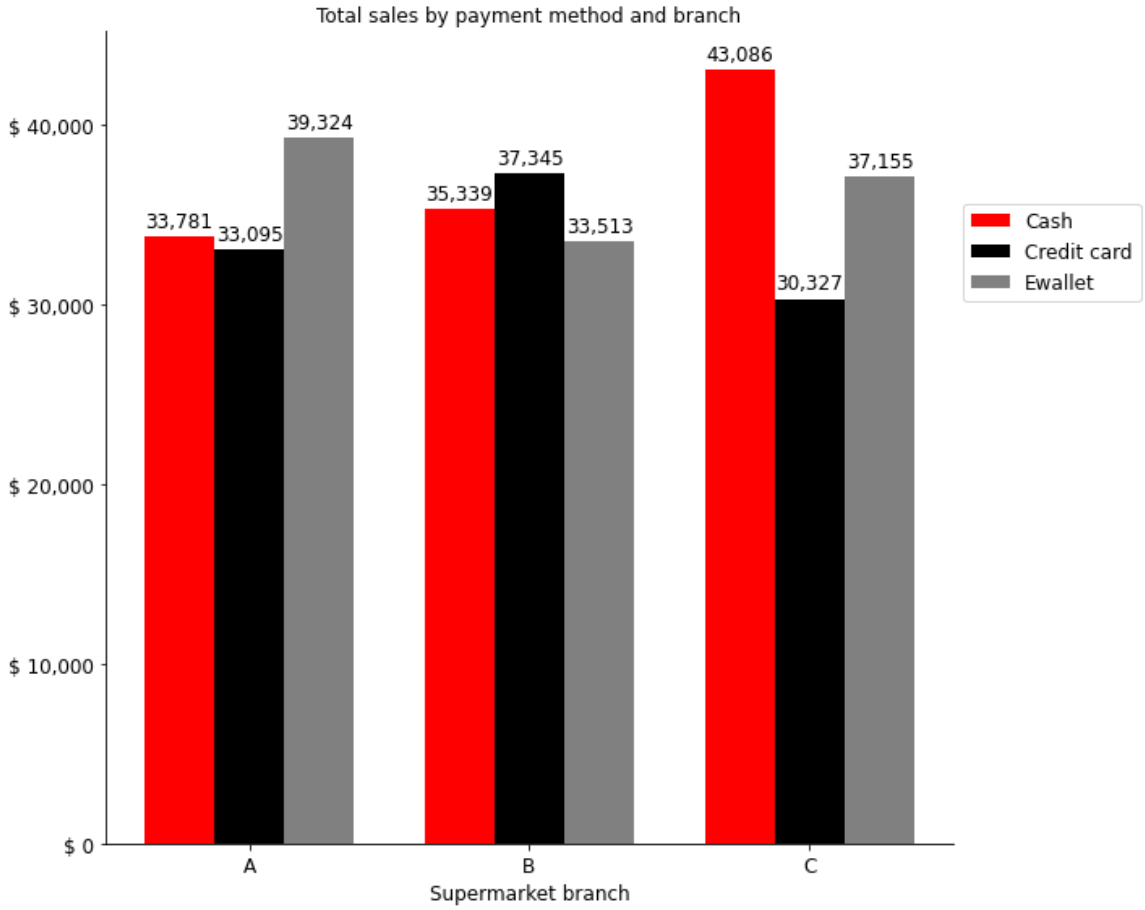
```
#creating pivot table showing payment breakdown
payment_method_pivot = sales_payment_method.pivot_table(columns = 'payment_method',
                                                        index = 'branch',
                                                        values = 'total_price').reset_index()
payment_method_pivot
```

Out[79]:

payment_method	branch	Cash	Credit card	Ewallet
0	A	33781	33095	39324
1	B	35339	37345	33513
2	C	43086	30327	37155

In [80]:

```
#plotting data on barchart. Cash mainly used in branch C with Ewallets close behind. Ew
allets a popular choice at Branch A
num_bars = 3
width = 0.25

branch_bars = np.arange(num_bars)

fig = plt.figure(figsize = (10,8))
ax = fig.add_subplot()

bar_1 = ax.bar(branch_bars, payment_method_pivot['Cash'], width, color = 'red')
bar_2 = ax.bar(branch_bars + 0.25, payment_method_pivot['Credit card'], width, color =
'black')
bar_3 = ax.bar(branch_bars + 0.50, payment_method_pivot['Ewallet'], width, color = 'gre
y')

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

#setting y_tick values to have commas to separate 1000s and also a $ sign in front
value_format = '${x: ,.0f}'
tick = mtick.StrMethodFormatter(value_format)
ax.yaxis.set_major_formatter(tick)
plt.yticks(fontsize = 12)

#setting x_ticks
ax.set_xticks(branch_bars + (width * 2) / 2)
ax.set_xticklabels(payment_method_pivot.branch, fontsize = 12)
ax.set_xlabel('Supermarket branch', fontsize = 12)

#using the function created in first chart to add labels to each bar
autolabel(bar_1)
autolabel(bar_2)
autolabel(bar_3)

#bbox_to_anchor shifts the legend along the x and y axis
ax.legend((bar_1, bar_2, bar_3), ('Cash', 'Credit card', 'Ewallet'), bbox_to_anchor = (
1.0, 0.8), fontsize = 12)

plt.title('Total sales by payment method and branch', fontsize = 12, y = 1.0)
plt.tight_layout()

plt.savefig('sales_payment_method')
```


Total sales by payment method and branch
$ 40,000
$ 30,000
$ 20,000
$ 10,000
$ 0
33,781
33,095
39,324
35,339
37,345
33,513
43,086
30,327
37,155
A
B
C
Supermarket branch
Cash
Credit card
Ewallet

In [81]:

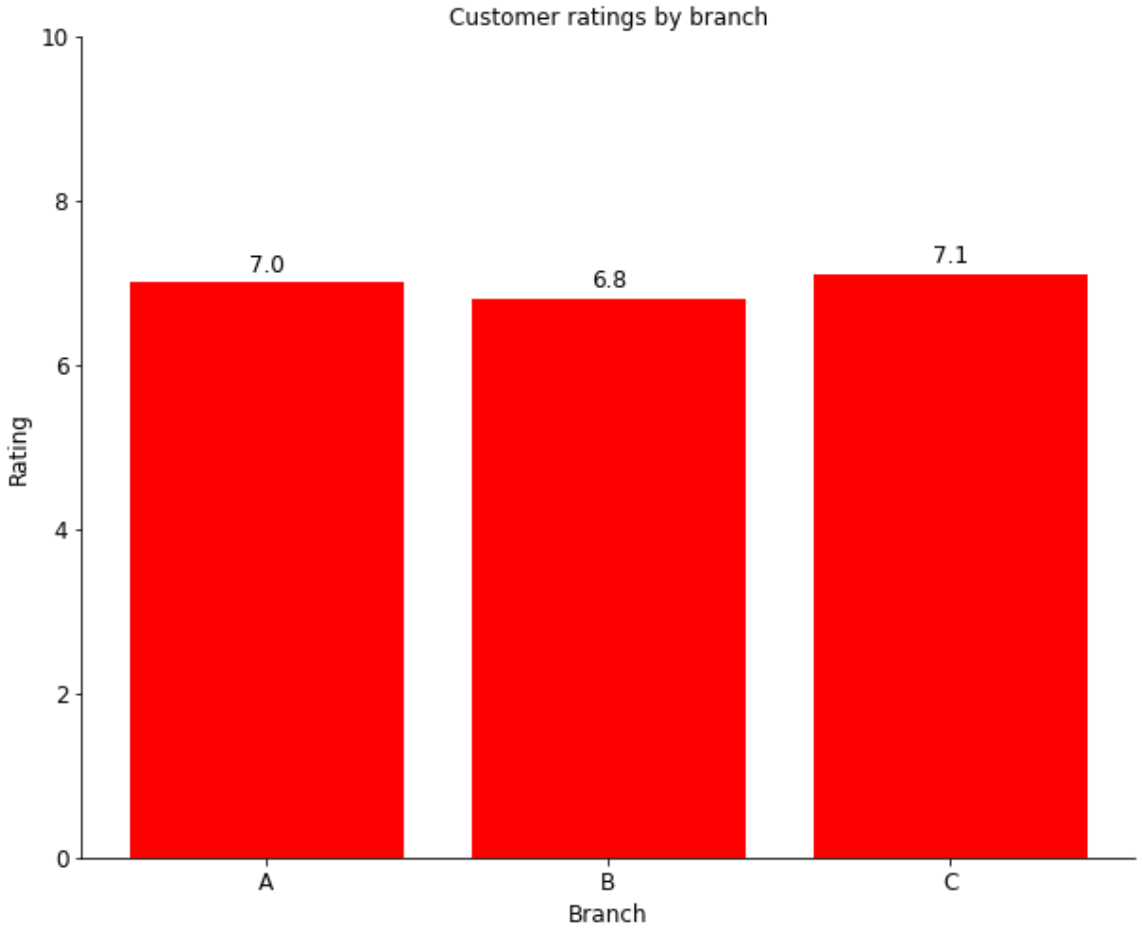
```
#grouping by branch to see which branch had the highest average customer rating. Branch
C had the highest rating by a small
#margin and this branch also generated the most sales
customer_rating = supermarket.groupby('branch').customer_rating.mean().round(1).to_fram
e().reset_index()
customer_rating
```

Out[81]:

	branch	customer_rating
0	A	7.0
1	B	6.8
2	C	7.1

In [82]:

```
#creating chart
plt.figure(figsize = (10, 8))
ax = plt.subplot()

bar = plt.bar(customer_rating.branch, customer_rating.customer_rating, color = 'red')
plt.xticks(fontsize = 12)
plt.xlabel('Branch', fontsize = 12)
plt.yticks(fontsize =12)
plt.ylabel('Rating', fontsize = 12)
ax.set_ylim(0, 10)

#removing chart borders
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

#adding labels
autolabel(bar)

plt.title('Customer ratings by branch', fontsize = 12)
plt.savefig('customer_ratings')
```



Preparing data for model

In [83]:

```
#preparing data for model - converting categorical variables to dummy data
supermarket_dummy = pd.get_dummies(supermarket)
supermarket_dummy.head()
```

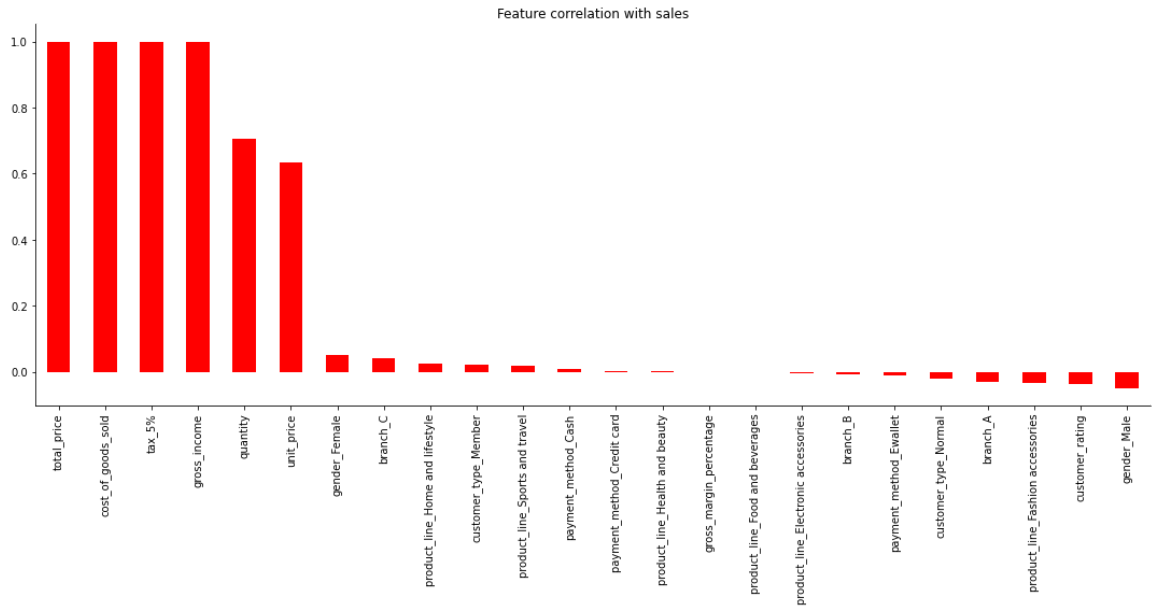
Out[83]:

	unit_price	quantity	tax_5%	total_price	cost_of_goods_sold	gross_margin_percentage	g
0	74.69	7	26.1415	548.9715	522.83	4.761905	
1	15.28	5	3.8200	80.2200	76.40	4.761905	
2	46.33	7	16.2155	340.5255	324.31	4.761905	
3	58.22	8	23.2880	489.0480	465.76	4.761905	
4	86.31	7	30.2085	634.3785	604.17	4.761905	

5 rows × 24 columns

In [84]:

```
#plot correlations between target feature (sales) and all other variables
fig = plt.figure(figsize = (15,8))
ax = plt.subplot()
supermarket_dummy.corr()['total_price'].sort_values(ascending = False).plot(kind = 'ba
r', color = 'red')
plt.title('Feature correlation with sales', fontsize = 12)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.tight_layout()
plt.savefig('correlation_1')
```



In [85]:

```
#cost of good sold, tax and gross income have a perfect positive correlation with sales
- removing these from data to
#avoid multicollinearity
supermarket_dummy = supermarket_dummy.drop(['cost_of_goods_sold', 'tax_5%', 'gross_inco
me'], axis = 1)
supermarket_dummy.head()
```

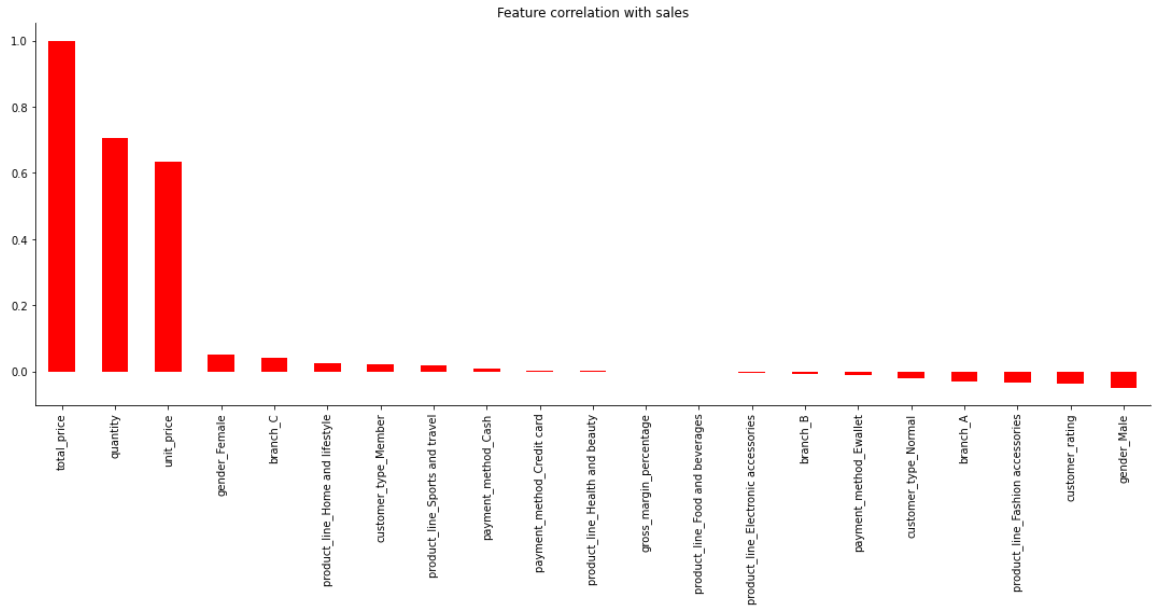
Out[85]:

	unit_price	quantity	total_price	gross_margin_percentage	customer_rating	branch_A	bra
0	74.69	7	548.9715	4.761905	9.1	1	
1	15.28	5	80.2200	4.761905	9.6	0	
2	46.33	7	340.5255	4.761905	7.4	1	
3	58.22	8	489.0480	4.761905	8.4	1	
4	86.31	7	634.3785	4.761905	5.3	1	

5 rows × 21 columns

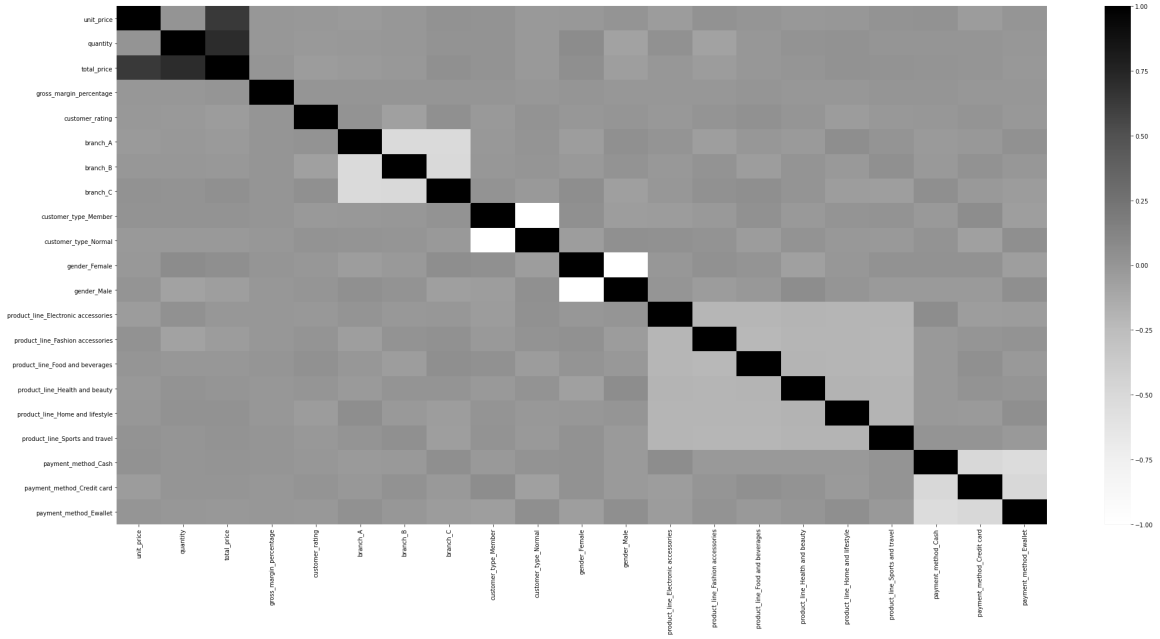
In [86]:

```
#re-plotting correlations between target feature (sales) and all other variables - quan
tity and unit price have the
#highest correlation with sales, male gender and rating are least correlated
fig = plt.figure(figsize = (15,8))
ax = plt.subplot()
supermarket_dummy.corr()['total_price'].sort_values(ascending = False).plot(kind = 'ba
r', color = 'red')
plt.title('Feature correlation with sales', fontsize = 12)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.tight_layout()
plt.savefig('sales_correlation')
```



In [87]:

```
#plotting correlations on a heatmap
fig, ax = plt.subplots(figsize = (30,15))
sns.heatmap(supermarket_dummy.corr(), cmap = 'Greys')
plt.tight_layout()
plt.savefig('heatmap')
```



In [88]:

```
#saving the column names to a separate list for reassigning after scaling the data
supermarket_columns = supermarket_dummy.columns
```

In [89]:

```
#scaling the numerical data

#instantiate the MinMaxScaler
scaler = MinMaxScaler()

#fit the scaler to data to transform the data. Converting to dataframe as well - otherw
ise it would be a NumPy array
supermarket_dummy = pd.DataFrame(scaler.fit_transform(supermarket_dummy))

#reassign the column names
supermarket_dummy.columns = supermarket_columns

supermarket_dummy.head()
```

Out[89]:

	unit_price	quantity	total_price	gross_margin_percentage	customer_rating	branch_A	br
0	0.718847	0.666667	0.521616	0.0	0.850000	1.0	
1	0.057855	0.444444	0.067387	0.0	0.933333	0.0	
2	0.403316	0.666667	0.319628	0.0	0.566667	1.0	
3	0.535603	0.777778	0.463549	0.0	0.733333	1.0	
4	0.848131	0.666667	0.604377	0.0	0.216667	1.0	

5 rows × 21 columns

In [90]:

```
#splitting the data into independent and dependent variables

y = supermarket_dummy.total_price

X = supermarket_dummy.drop('total_price', axis = 1)
```

In [91]:

```
#splitting the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 42)
```

In [92]:

```
#checking shape of training and testing sets - all 100 records are there
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)
```

```
X_train shape: (800, 20)
X_test shape: (200, 20)
y_train shape: (800,)
y_test shape: (200,)
```

Linear regression

In [93]:

```
#Initialise the linear regression model
lr = LinearRegression()

#fit the model
lr.fit(X_train, y_train)

#make predictions
prediction = lr.predict(X_test)
```

In [94]:

```
#evaluating model - r squared of 90% - very good fit
r2 = format(r2_score(y_test, prediction), '.3f')
rmse = format(np.sqrt(mean_squared_error(y_test, prediction)), '.3f')
mae = format(mean_absolute_error(y_test, prediction), '.3f')

print('r2:', r2)
print('root mean squared error', rmse)
print('mean absolute error', mae)
```

```
r2: 0.902
root mean squared error 0.077
mean absolute error 0.057
```

In [95]:

```
#trying polynomial regression which is more complex and can fit non linear trends. Fitt
ing with degree 2
polyfeat = PolynomialFeatures(degree = 2)
X_trainpoly = polyfeat.fit_transform(X_train) #this transforms data into an array
X_testpoly = polyfeat.fit_transform(X_test)

poly = LinearRegression().fit(X_trainpoly, y_train)

pred = poly.predict(X_testpoly)
```

In [96]:

```
#evaluating polynomial model - model has a prefect score!
r2 = format(r2_score(y_test, pred), '.3f')
rmse = format(np.sqrt(mean_squared_error(y_test, pred)), '.3f')
mae = format(mean_absolute_error(y_test, pred), '.3f')

print('r2:', r2)
print('root mean squared error', rmse)
print('mean absolute error', mae)
```

```
r2: 1.000
root mean squared error 0.000
mean absolute error 0.000
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```