In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
Customers = pd.read_excel('AdventureWorks_Database.xlsx','Customers',
               dtype = {'CustomerKey':str},
               parse_dates = ['BirthDate','DateFirstPurchase']
             )
```

In [3]:
```
Product = pd.read_excel('AdventureWorks_Database.xlsx','Product',
                         dtype={'ProductKey':str},
                         parse_dates=['StartDate']
                    )
```

In [4]:
```
Sales = pd.read_excel('AdventureWorks_Database.xlsx','Sales',
                        dtype={'ProductKey':str,
                               'CustomerKey':str,
                               'PromotionKey':str,
                               'SalesTerritoryKey':str},
                        parse_dates=['OrderDate', 'ShipDate']
                              )
Sales['DateKey'] = Sales['OrderDate'].astype(str)
```

In [5]:
```
Territory = pd.read_excel('AdventureWorks_Database.xlsx','Territory',
                              dtype={'SalesTerritoryKey':str}
                              )
```

In [6]:
```
temp  = pd.merge(Sales, Product, on='ProductKey', how='inner')
df = pd.merge(temp, Customers, on='CustomerKey', how='inner')
df = pd.merge(df, Territory, on='SalesTerritoryKey', how='inner')
```

In [7]:
```
df.describe()
```

Out[7]:

	OrderDate	ShipDate	SalesOrderLineNumber	OrderQuantity	UnitPrice	To
count	58189	58189	58189.000000	58189.000000	58189.000000	
mean	2016-06-03 03:56:09.605939200	2016-06-10 04:03:24.657237760	1.887453	1.569386	413.888218	
min	2014-01-01 00:00:00	2014-01-08 00:00:00	1.000000	1.000000	0.572500	
25%	2016-04-01 00:00:00	2016-04-08 00:00:00	1.000000	1.000000	4.990000	
50%	2016-07-07 00:00:00	2016-07-14 00:00:00	2.000000	1.000000	24.490000	
75%	2016-10-10 00:00:00	2016-10-17 00:00:00	2.000000	2.000000	269.995000	
max	2016-12-30 00:00:00	2017-01-07 00:00:00	8.000000	4.000000	3578.270000	
std	NaN	NaN	1.018829	1.047532	833.052938	

8 rows × 31 columns

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 58 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  object
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  object
 4   PromotionKey          58189 non-null  object
 5   SalesTerritoryKey     58189 non-null  object
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Unnamed: 13           0 non-null      float64
 14  Unnamed: 14           0 non-null      float64
 15  Unnamed: 15           58189 non-null  float64
 16  Unnamed: 16           58189 non-null  float64
 17  Unnamed: 17           0 non-null      float64
 18  Unnamed: 18           58189 non-null  float64
 19  Unnamed: 19           0 non-null      float64
 20  StandardCost_x        58189 non-null  float64
 21  List Price            58189 non-null  float64
 22  Unnamed: 22           0 non-null      float64
 23  diif std cost         58189 non-null  int64
 24  diff list price       58189 non-null  int64
 25  DateKey               58189 non-null  object
 26  ProductName           58189 non-null  object
 27  SubCategory           58189 non-null  object
 28  Category              58189 non-null  object
 29  StandardCost_y        58189 non-null  float64
 30  Color                 30747 non-null  object
 31  ListPrice             58189 non-null  float64
 32  DaysToManufacture     58189 non-null  int64
 33  ProductLine           58189 non-null  object
 34  ModelName             58189 non-null  object
 35  Photo                 58189 non-null  object
 36  ProductDescription    58189 non-null  object
 37  StartDate             58189 non-null  datetime64[ns]
 38  FirstName             58189 non-null  object
 39  LastName              58189 non-null  object
 40  FullName              58189 non-null  object
 41  BirthDate             58189 non-null  datetime64[ns]
 42  MaritalStatus         58189 non-null  object
 43  Gender                58189 non-null  object
 44  YearlyIncome          58189 non-null  int64
 45  TotalChildren         58189 non-null  int64
 46  NumberChildrenAtHome  58189 non-null  int64
 47  Education             58189 non-null  object
 48  Occupation            58189 non-null  object
 49  HouseOwnerFlag        58189 non-null  int64
 50  NumberCarsOwned       58189 non-null  int64
 51  AddressLine1          58189 non-null  object
 52  DateFirstPurchase     58189 non-null  datetime64[ns]
 53  CommuteDistance       58189 non-null  object
 54  Region                58189 non-null  object
 55  Country                58189 non-null  object
 56  Group                  58189 non-null  object
 57  RegionImage            58189 non-null  object
dtypes: datetime64[ns](5), float64(16), int64(10), object(27)
memory usage: 25.7+ MB
```

In [9]:
```
df.drop(df.columns[13:20].append(df.columns[21:23]), axis=1, inplace=True)
```

In [10]:
```
df.shape[0]
```

Out[10]:
```
58189
```

In [11]:
```
df.shape[1]
```

Out[11]:
```
49
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
ProductKey                 0
OrderDate                  0
ShipDate                   0
CustomerKey                0
PromotionKey               0
SalesTerritoryKey          0
SalesOrderNumber           0
SalesOrderLineNumber       0
OrderQuantity              0
UnitPrice                  0
TotalProductCost           0
SalesAmount                0
TaxAmt                     0
StandardCost_x             0
diif std cost              0
diff list price            0
DateKey                    0
ProductName                0
SubCategory                0
Category                   0
StandardCost_y             0
Color                  27442
ListPrice                  0
DaysToManufacture          0
ProductLine                0
ModelName                  0
Photo                      0
ProductDescription         0
StartDate                  0
FirstName                  0
LastName                   0
FullName                   0
BirthDate                  0
MaritalStatus              0
Gender                     0
YearlyIncome               0
TotalChildren              0
NumberChildrenAtHome       0
Education                  0
Occupation                 0
HouseOwnerFlag             0
NumberCarsOwned            0
AddressLine1               0
DateFirstPurchase          0
CommuteDistance            0
Region                     0
Country                    0
Group                      0
RegionImage                0
dtype: int64
```

In [14]:
```
df= df.dropna(axis=1)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
ProductKey              0
OrderDate               0
ShipDate                0
CustomerKey             0
PromotionKey            0
SalesTerritoryKey       0
SalesOrderNumber        0
SalesOrderLineNumber    0
OrderQuantity           0
UnitPrice               0
TotalProductCost        0
SalesAmount             0
TaxAmt                  0
StandardCost_x          0
diif std cost           0
diff list price         0
DateKey                 0
ProductName             0
SubCategory             0
Category                0
StandardCost_y          0
ListPrice               0
DaysToManufacture       0
ProductLine             0
ModelName               0
Photo                   0
ProductDescription      0
StartDate               0
FirstName               0
LastName                0
FullName                0
BirthDate               0
MaritalStatus           0
Gender                  0
YearlyIncome            0
TotalChildren           0
NumberChildrenAtHome    0
Education               0
Occupation              0
HouseOwnerFlag          0
NumberCarsOwned         0
AddressLine1            0
DateFirstPurchase       0
CommuteDistance         0
Region                  0
Country                 0
Group                   0
RegionImage             0
dtype: int64
```

In [16]:
```
df['ProductName'].nunique()
```

Out[16]:
```
130
```

In [17]:
```
df['ProductName'].unique().tolist()
```

```
Out[17]: ['Road-150 Red, 62',
 'Mountain-500 Black, 52',
 'Road Bottle Cage',
 'Water Bottle - 30 oz.',
 'Road-750 Black, 44',
 'Road-750 Black, 52',
 'Half-Finger Gloves, M',
 'Mountain Bottle Cage',
 'Patch Kit/8 Patches',
 'Mountain-500 Silver, 42',
 'Sport-100 Helmet, Red',
 'Touring-1000 Blue, 46',
 'Touring-1000 Blue, 60',
 'LL Mountain Tire',
 'Mountain-500 Black, 48',
 'Sport-100 Helmet, Black',
 'Long-Sleeve Logo Jersey, XL',
 'Road-250 Black, 52',
 'Bike Wash - Dissolver',
 'AWC Logo Cap',
 'Touring-1000 Yellow, 46',
 'Sport-100 Helmet, Blue',
 'Road-350-W Yellow, 48',
 'Road-350-W Yellow, 42',
 'Long-Sleeve Logo Jersey, L',
 'Touring-2000 Blue, 60',
 'Road-550-W Yellow, 38',
 'Touring-2000 Blue, 50',
 'Touring-3000 Yellow, 44',
 'Mountain-100 Silver, 44',
 'Mountain-200 Black, 38',
 'Road-150 Red, 44',
 'Hydration Pack - 70 oz.',
 'Mountain Tire Tube',
 'Mountain-500 Black, 42',
 'Racing Socks, M',
 'Touring-1000 Yellow, 54',
 'Touring-1000 Yellow, 50',
 'Touring-2000 Blue, 54',
 'Road Tire Tube',
 'ML Road Tire',
 'Touring-2000 Blue, 46',
 'Touring Tire',
 'Touring Tire Tube',
 'Touring-3000 Blue, 58',
 'LL Road Tire',
 'Mountain-100 Black, 48',
 'Road-550-W Yellow, 40',
 'Mountain-200 Silver, 46',
 'Mountain-100 Silver, 38',
 'Short-Sleeve Classic Jersey, L',
 'Road-550-W Yellow, 42',
 'Road-150 Red, 48',
 'Road-750 Black, 58',
 'Fender Set - Mountain',
 'Touring-1000 Blue, 50',
 'Half-Finger Gloves, L',
 'Touring-1000 Blue, 54',
 'Racing Socks, L',
 'Road-150 Red, 52',
 'Mountain-500 Silver, 44',
 'Mountain-500 Silver, 40',
 'Touring-1000 Yellow, 60',
 'HL Road Tire',
 'Road-250 Black, 48',
 'Classic Vest, S',
 'Touring-3000 Blue, 50',
 'Touring-3000 Yellow, 58',
 'Long-Sleeve Logo Jersey, S',
 'Touring-3000 Yellow, 50',
 'Touring-3000 Blue, 62',
 'Road-150 Red, 56',
 'Mountain-500 Black, 40',
 'Road-250 Black, 44',
 'Classic Vest, L',
 'Road-750 Black, 48',
 'Classic Vest, M',
 'Touring-3000 Yellow, 62',
 'Mountain-100 Black, 44',
 'Short-Sleeve Classic Jersey, M',
 'Mountain-200 Silver, 42',
 'Road-550-W Yellow, 48',
 'Mountain-100 Silver, 42',
 'HL Mountain Tire',
 'Mountain-200 Black, 46',
 'Road-650 Black, 52',
 'Road-650 Red, 48',
 'Road-650 Black, 44',
 'Mountain-100 Black, 38',
 'Road-550-W Yellow, 44',
 'Mountain-200 Silver, 38',
 'Hitch Rack - 4-Bike',
 'Mountain-100 Black, 42',
 'Mountain-200 Black, 42',
 'Road-650 Red, 62',
 'Road-650 Red, 60',
 'Road-650 Black, 60',
 'Road-650 Red, 58',
 'Road-250 Red, 58',
 'Road-650 Red, 52',
 'Long-Sleeve Logo Jersey, M',
 'All-Purpose Bike Stand',
 'Mountain-400-W Silver, 40',
 'ML Mountain Tire',
 'Mountain-500 Black, 44',
 'Road-250 Black, 58',
 'Road-250 Red, 48',
 'Road-250 Red, 44',
 'Road-250 Red, 52',
 'Road-650 Black, 62',
 'Short-Sleeve Classic Jersey, XL',
 'Half-Finger Gloves, S',
 'Mountain-400-W Silver, 42',
 "Women's Mountain Shorts, L",
 'Short-Sleeve Classic Jersey, S',
 'Road-650 Black, 58',
 'Road-650 Black, 48',
 'Road-650 Red, 44',
 "Women's Mountain Shorts, S",
 "Women's Mountain Shorts, M",
 'Touring-3000 Blue, 54',
 'Road-350-W Yellow, 44',
 'Road-350-W Yellow, 40',
 'Mountain-400-W Silver, 46',
 'Mountain-500 Silver, 52',
 'Mountain-400-W Silver, 38',
 'Touring-3000 Yellow, 54',
 'Mountain-500 Silver, 48',
 'Touring-3000 Blue, 44',
 'Mountain-100 Silver, 48']
```

In [18]:
```
print(df['Category'].unique())
```

```
['Bikes' 'Accessories' 'Clothing']
```

In [19]:
```
df['SubCategory'].unique().tolist()
```

Out[19]:
```
['Road Bikes',
 'Mountain Bikes',
 'Bottles and Cages',
 'Gloves',
 'Tires and Tubes',
 'Helmets',
 'Touring Bikes',
 'Jerseys',
 'Cleaners',
 'Caps',
 'Hydration Packs',
 'Socks',
 'Fenders',
 'Vests',
 'Bike Racks',
 'Bike Stands',
 'Shorts']
```

In [20]:
```
df['sale_year'] = df['OrderDate'].dt.year
df['sale_month'] = df['OrderDate'].dt.month
df['sale_day'] = df['OrderDate'].dt.day
df['sale_week'] = df['OrderDate'].dt.dayofweek



df['sale_day_name'] = df['OrderDate'].dt.day_name()
df['year_month'] = df['OrderDate'].apply(lambda x:x.strftime('%Y-%m'))


df['total_Invoice_amount'] = df['SalesAmount'] + df['TaxAmt']
df['profit'] = (df['UnitPrice']*df['OrderQuantity']) - df['TotalProductCost']


df['ProductName'] = df['ProductName'].str.replace(',','-')

df['Age'] = df['OrderDate'].dt.year - df['BirthDate'].dt.year
```

In [21]:
```
df
```

Out[21]:

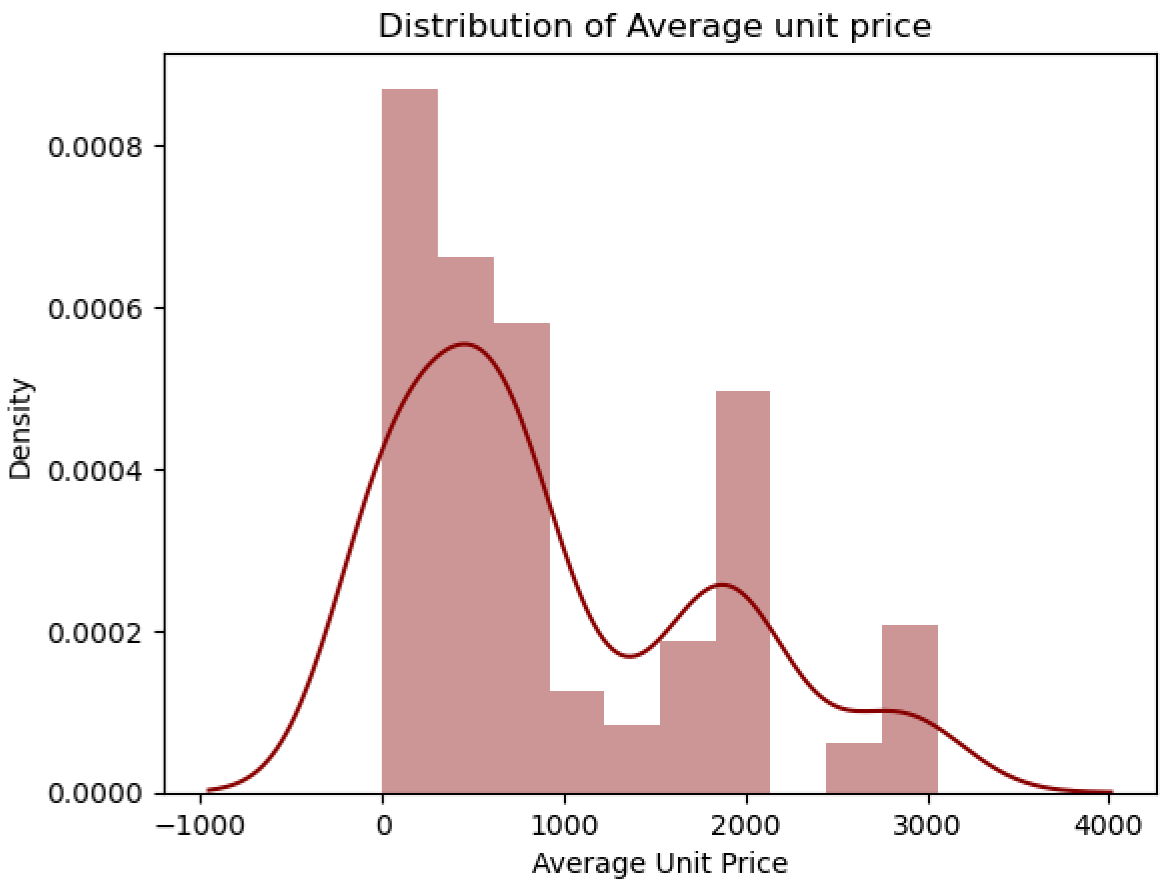
	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderN
0	310	2014-01-01	2014-01-08	21768	1	6	S
1	600	2016-04-16	2016-04-23	21768	1	6	S
2	310	2014-01-30	2014-02-06	21727	1	6	S
3	479	2016-11-29	2016-12-05	21727	1	6	S
4	477	2016-11-29	2016-12-05	21727	1	6	S
...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	S
58185	361	2016-11-07	2016-11-14	13145	1	2	S
58186	480	2016-11-07	2016-11-14	13145	1	2	S
58187	530	2016-02-06	2016-02-13	27040	1	2	S
58188	480	2016-02-06	2016-02-13	27040	2	2	S

58189 rows × 57 columns

Average Unit Price

In [22]:
```
AvgUnitPrice = df.groupby(['ProductKey'])['UnitPrice'].mean()
sns.distplot(AvgUnitPrice,bins=10 , kde = True, hist = True , color = 'darkred' )
plt.title('Distribution of Average unit price ')
plt.xlabel('Average Unit Price')
```

Out[22]:
```
Text(0.5, 0, 'Average Unit Price')
```

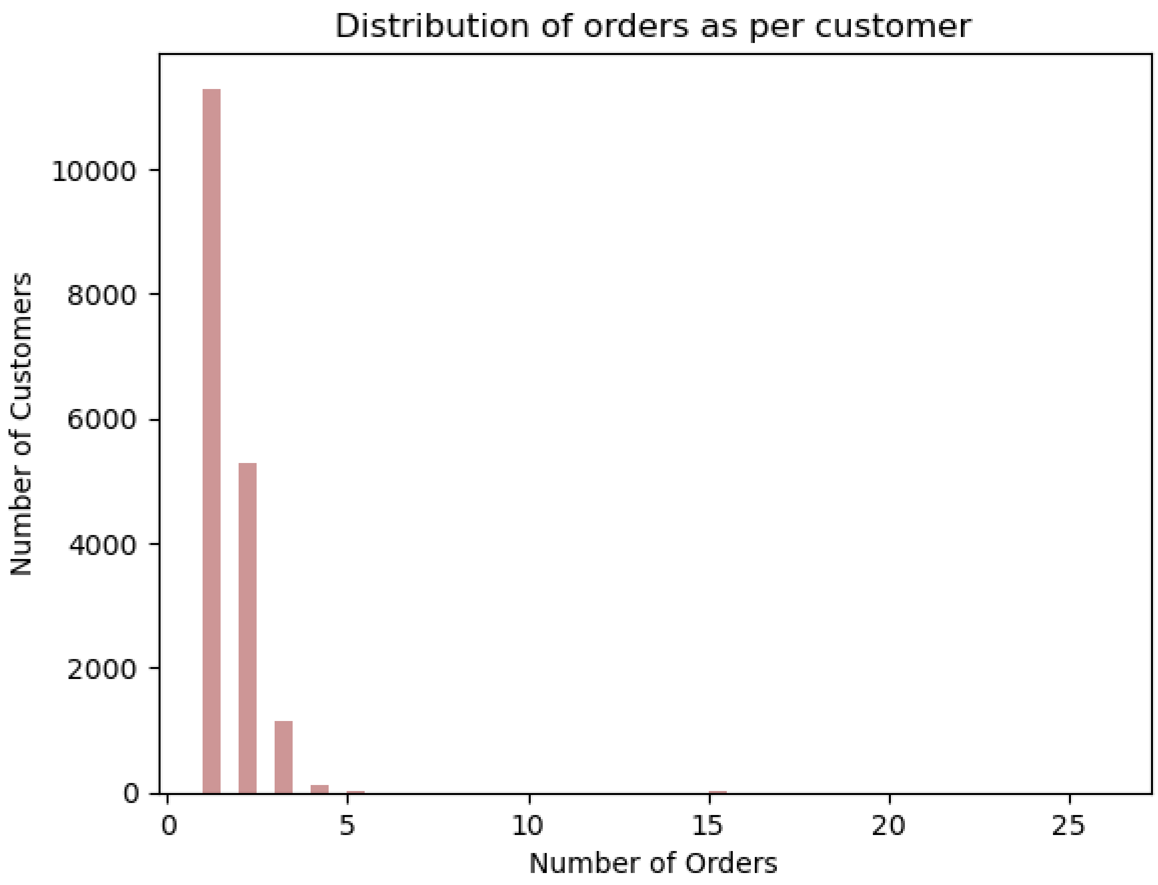


- Maximum of the product unit price is under $1000.

Sales order number distribution

In [23]:
```
orders = df.groupby(['CustomerKey'])['SalesOrderNumber'].nunique()
sns.distplot(orders, kde = False, color= 'darkred')
plt.title('Distribution of orders as per customer')
plt.xlabel('Number of Orders')
plt.ylabel('Number of Customers')
```

Out[23]:
```
Text(0, 0.5, 'Number of Customers')
```



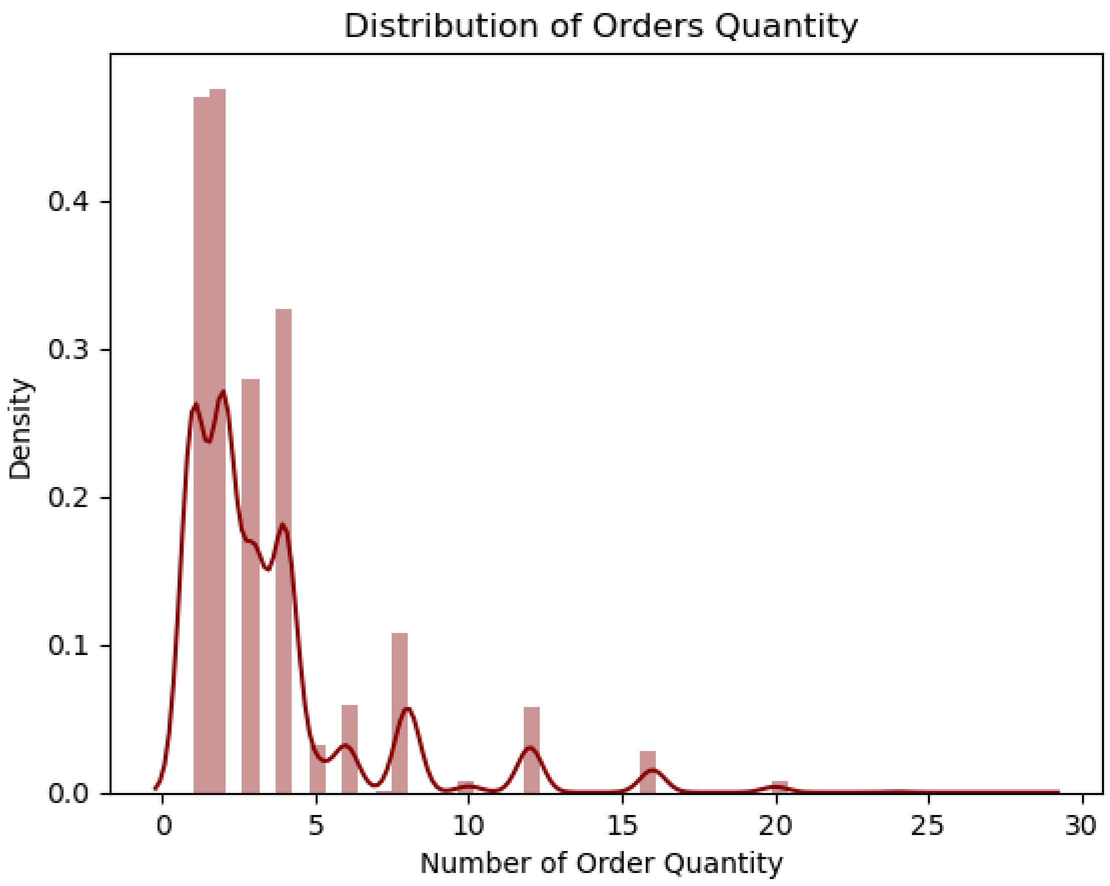
In [24]:
```
print(round((np.sum(orders>1)/df['CustomerKey'].nunique())*100,2),"% of Customers orde
```

```
36.97 % of Customers ordered more then once
```

Sales Order Quantity distribution

In [25]:
```
OrderQuantity = df.groupby(['SalesOrderNumber'])['OrderQuantity'].sum()
sns.distplot(OrderQuantity, kde = True,hist = True, color= 'darkred')
plt.title('Distribution of Orders Quantity ')
plt.xlabel('Number of Order Quantity ')
```

Out[25]:
```
Text(0.5, 0, 'Number of Order Quantity ')
```



- Maximum quantity ordered for a product is less then 5.

Age Distribution

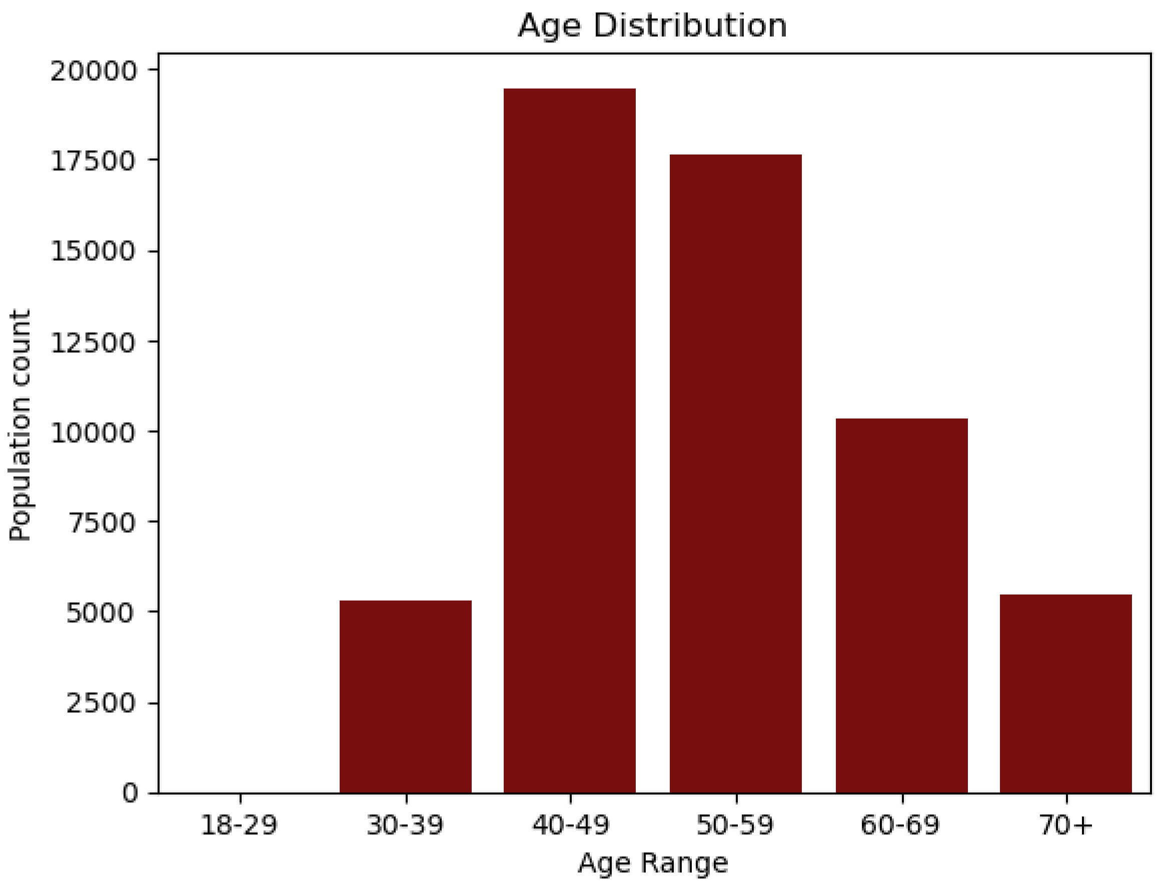
In [26]:
```
bins = [18, 30, 40, 50, 60, 70, 120]
labels = ['18-29', '30-39', '40-49', '50-59', '60-69', '70+']
df['agerange'] = pd.cut(df.Age, bins, labels = labels,include_lowest = True)


age_distribution = df['agerange'].value_counts().to_frame().reset_index()
age_distribution.columns = ['Age Range','Population count']
```

In [27]:
```
pd.cut(df.Age, bins, labels = labels,include_lowest = True)
```

Out[27]:
```
0        60-69
1        60-69
2        40-49
3        40-49
4        40-49
         ...
58184    30-39
58185    30-39
58186    30-39
58187    50-59
58188    50-59
Name: Age, Length: 58189, dtype: category
Categories (6, object): ['18-29' < '30-39' < '40-49' < '50-59' < '60-69' < '70+']
```

```
In [28]: sns.barplot(age_distribution,x = 'Age Range', y = 'Population count' , color= 'darkred
         plt.title('Age Distribution')
```

```
Out[28]: Text(0.5, 1.0, 'Age Distribution')
```



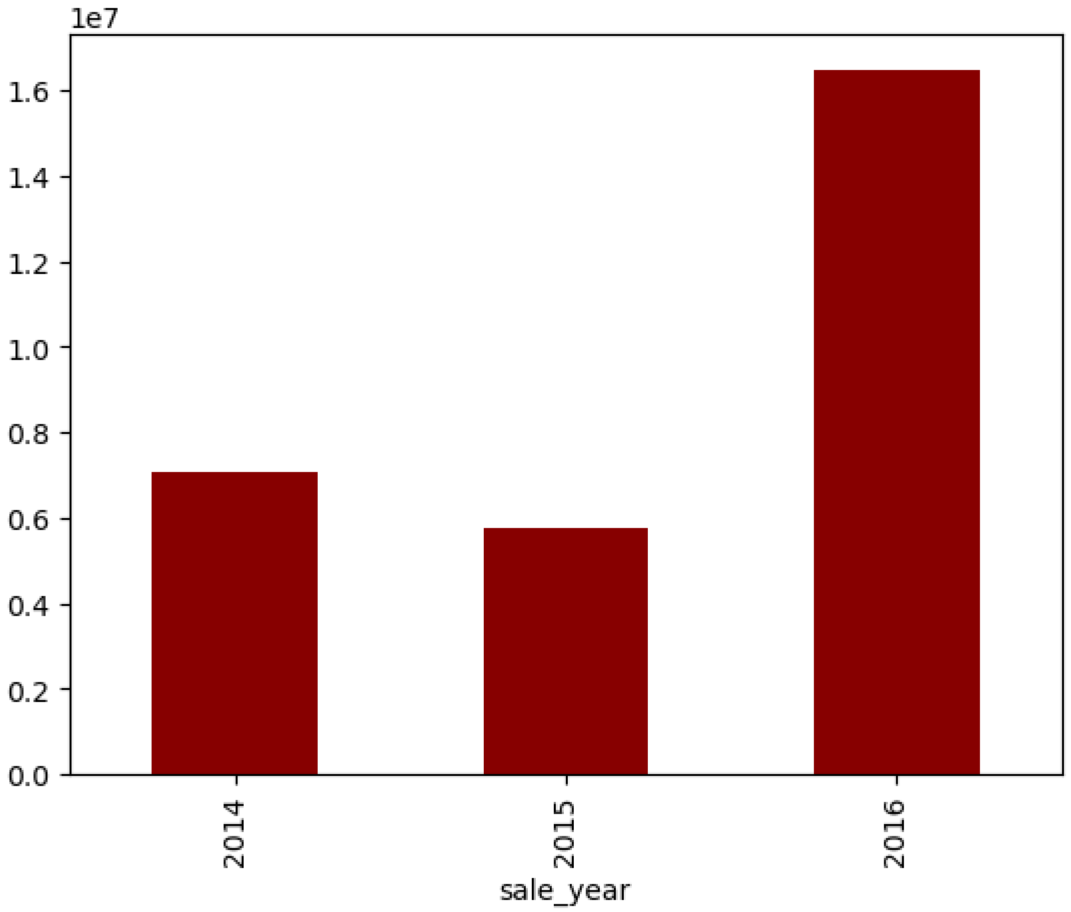
- Most of the clients are between the ages of 40 and 59

Sales

Year wise sales

```
In [29]: df.groupby('sale_year')['SalesAmount'].sum().plot(kind='bar', color='darkred')
```

```
Out[29]: <Axes: xlabel='sale_year'>
```



Quantity ordered according to category and subcategory between 2014-16

In [30]:
```
df.groupby(['sale_year','Category', 'SubCategory'])['OrderQuantity'].sum().to_frame()
```

Out[30]:

sale_year	Category	SubCategory	OrderQuantity
2014	Bikes	Mountain Bikes	616
		Road Bikes	2876
2015	Bikes	Mountain Bikes	1661
		Road Bikes	3284
2016	Accessories	Bike Racks	493
		Bike Stands	394
		Bottles and Cages	12055
		Cleaners	1381
		Fenders	3239
		Helmets	9685
		Hydration Packs	1124
		Tires and Tubes	25518
	Bikes	Mountain Bikes	5490
		Road Bikes	6535
		Touring Bikes	3410
	Clothing	Caps	3178
		Gloves	2143
		Jerseys	5068
		Shorts	1491
		Socks	856
		Vests	824

- In 2016, in the Accessories subcategory, Tires and Tubes had the maximum number of ordered quantities 25518.

Profit generated according to category and subcategory between 2014-16

In [31]:
```
df.groupby(['sale_year','Category', 'SubCategory'])['profit'].sum().to_frame().sort_va
```

Out[31]:

sale_year	Category	SubCategory	profit
2016	**Bikes**	**Mountain Bikes**	2.907361e+06
		Road Bikes	1.905954e+06
		Touring Bikes	1.454873e+06
	Accessories	**Tires and Tubes**	1.447931e+05
		Helmets	1.351677e+05
	Clothing	**Shorts**	4.197352e+04
		Jerseys	3.796523e+04
	Accessories	**Bottles and Cages**	3.444898e+04
		Fenders	2.771163e+04
		Hydration Packs	2.430313e+04
		Bike Stands	2.368909e+04
		Bike Racks	2.313696e+04
	Clothing	**Vests**	2.094878e+04
		Gloves	2.089574e+04
		Caps	4.331832e+03
	Accessories	**Cleaners**	4.299869e+03
	Clothing	**Socks**	3.055841e+03
2015	**Bikes**	**Road Bikes**	1.375065e+06
		Mountain Bikes	1.019388e+06
2014	**Bikes**	**Road Bikes**	2.256281e+06
		Mountain Bikes	5.868746e+05

- In 2016, within the Bikes subcategory, Mountain Bikes had generated the maximum amount of profit.

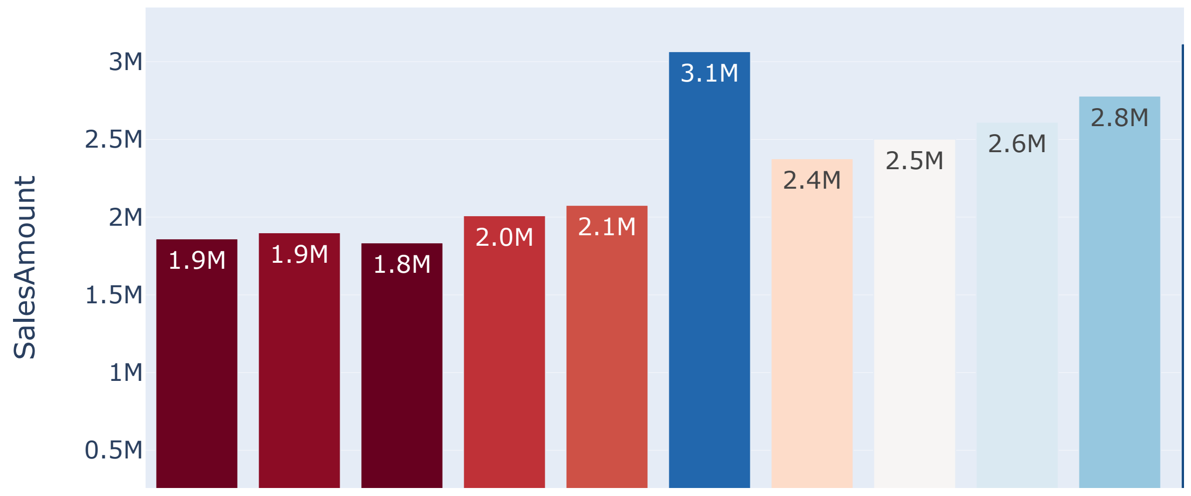
Monthly Sales & Profit

In [32]:
```
df.groupby('sale_month')[['SalesAmount', 'profit']].sum()
```

Out[32]:

sale_month	SalesAmount	profit
1	1.860422e+06	7.550957e+05
2	1.899607e+06	7.787224e+05
3	1.834668e+06	7.511616e+05
4	2.009169e+06	8.273750e+05
5	2.076070e+06	8.519208e+05
6	3.064630e+06	1.256211e+06
7	2.375857e+06	9.790081e+05
8	2.502387e+06	1.029611e+06
9	2.610615e+06	1.076202e+06
10	2.778842e+06	1.142254e+06
11	3.114646e+06	1.290573e+06
12	3.180924e+06	1.314384e+06

In [33]:
```python
salesmonth = df.groupby('sale_month')[['SalesAmount', 'profit']].sum()
salesmonth.reset_index(inplace=True)
px.bar(salesmonth, x='sale_month', y='SalesAmount',text_auto='.2s',
             hover_data=['sale_month', 'SalesAmount'], color='profit',
             color_continuous_scale='RdBu',
             height=400)
```

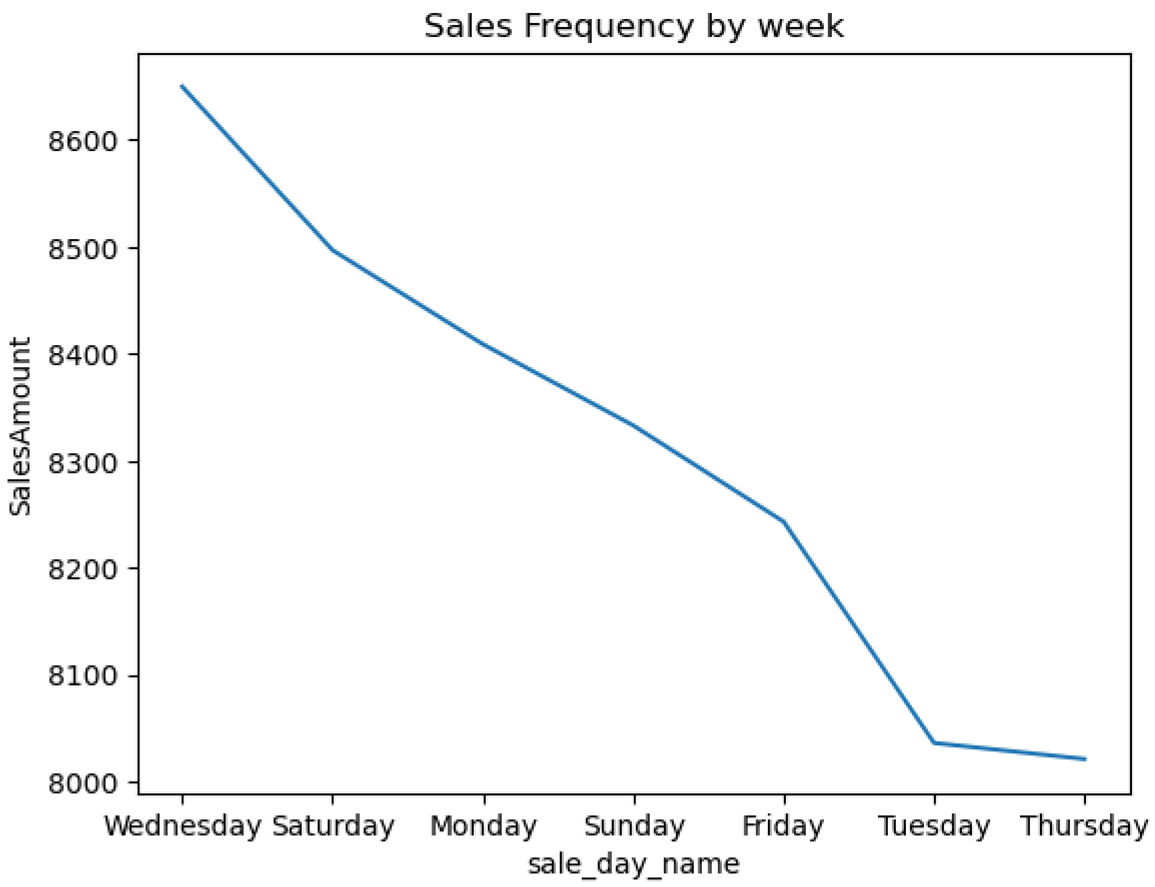


- In the months of june, november and december, Profit is more as compare other months.

Sales Freqcuency by week

```
In [34]: weeklysales = df.groupby(['sale_day_name']).count()['SalesAmount'].reset_index().sort_
         sns.lineplot(weeklysales, x = 'sale_day_name', y = 'SalesAmount')
         plt.title('Sales Frequency by week')
```

```
Out[34]: Text(0.5, 1.0, 'Sales Frequency by week')
```



- Here, Wednesday and Saturday had highest sales orders.

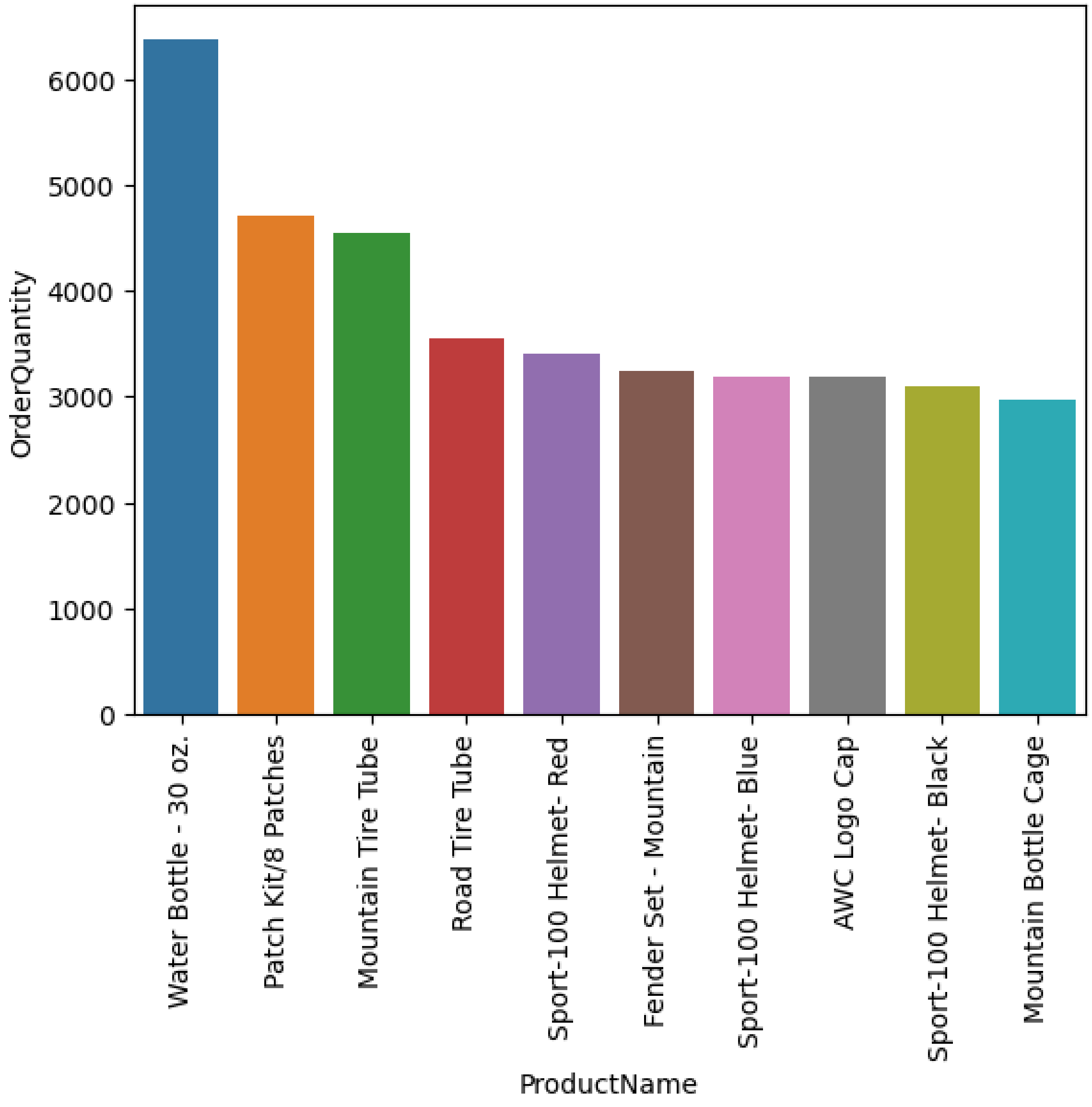
Top Selling Product based on Category and SubCategory

```
In [35]: topsellingproduct = df.groupby(['Category', 'SubCategory', 'ProductName'])['OrderQuant
         topsellingproduct
```

Out[35]:

	Category	SubCategory	ProductName	OrderQuantity
4	Accessories	Bottles and Cages	Water Bottle - 30 oz.	6370
18	Accessories	Tires and Tubes	Patch Kit/8 Patches	4705
17	Accessories	Tires and Tubes	Mountain Tire Tube	4551
19	Accessories	Tires and Tubes	Road Tire Tube	3544
9	Accessories	Helmets	Sport-100 Helmet- Red	3398
6	Accessories	Fenders	Fender Set - Mountain	3239
8	Accessories	Helmets	Sport-100 Helmet- Blue	3193
110	Clothing	Caps	AWC Logo Cap	3178
7	Accessories	Helmets	Sport-100 Helmet- Black	3094
2	Accessories	Bottles and Cages	Mountain Bottle Cage	2977

In [36]:
```
sns.barplot(topsellingproduct, x='ProductName', y = 'OrderQuantity')
plt.xticks(rotation=90)
plt.show()
```



Product with maximum and minimum profit

In [37]:
```
df.groupby(['ProductName'])['profit'].sum().reset_index().sort_values('profit', ascend
```

Out[37]:

	ProductName	profit
34	Mountain-200 Black- 46	626621.5699
33	Mountain-200 Black- 42	621759.6023
35	Mountain-200 Silver- 38	610864.4348
37	Mountain-200 Silver- 46	593490.4728
32	Mountain-200 Black- 38	590477.4544

In [38]:
```
df.groupby(['ProductName'])['profit'].sum().nsmallest(5).to_frame()
```

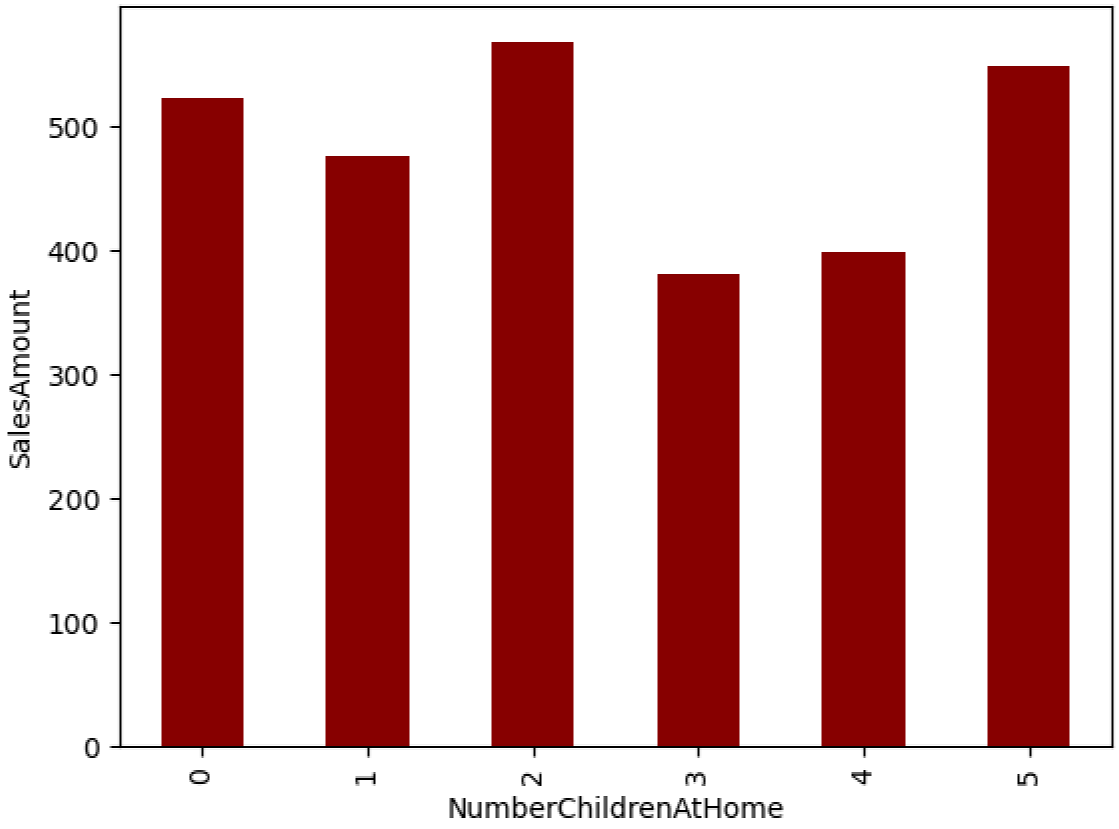
Out[38]:

ProductName	profit
Racing Socks- L	1474.4574
Racing Socks- M	1581.3837
Bike Wash - Dissolver	4299.8688
Patch Kit/8 Patches	4314.8350
AWC Logo Cap	4331.8315

Average Sales Amount Based on Number of Children at home

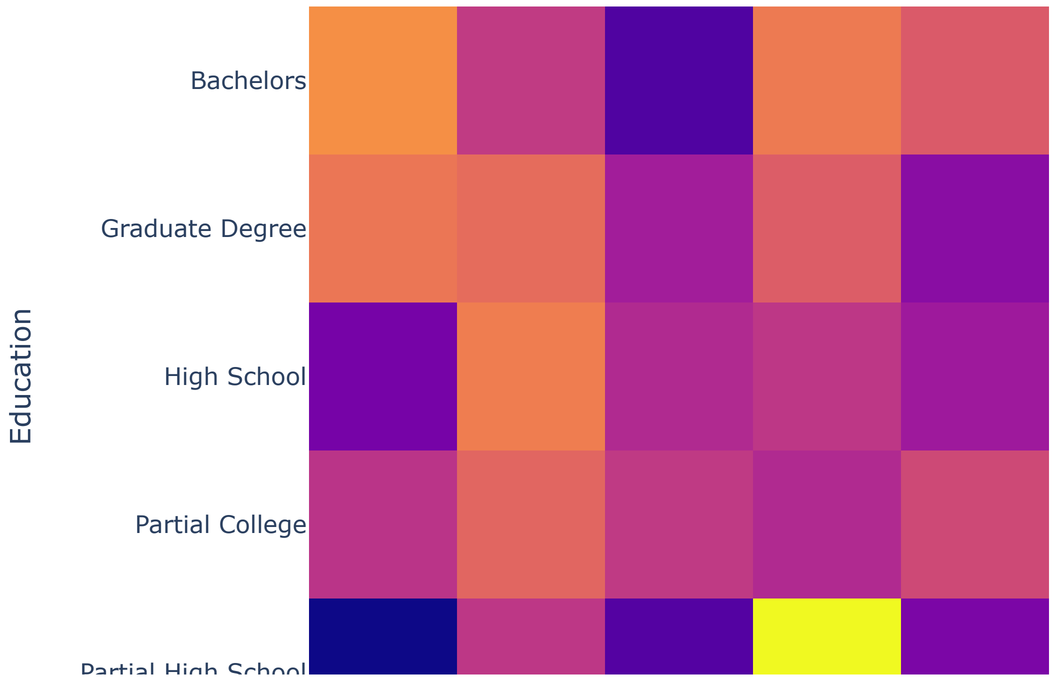
In [39]:
```
df.groupby(["NumberChildrenAtHome"])["SalesAmount"].mean().plot(kind = 'bar', color =
plt.ylabel('SalesAmount')
```

Out[39]:
```
Text(0, 0.5, 'SalesAmount')
```



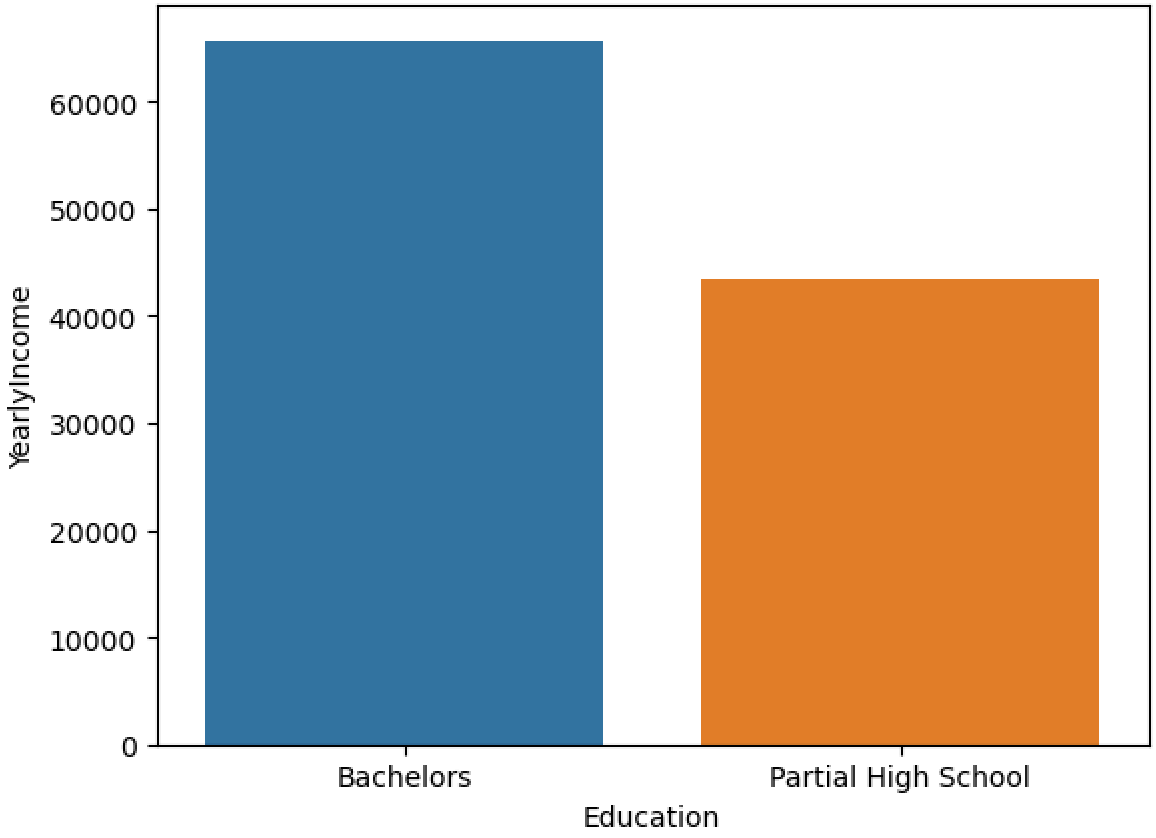
Education, Occupation and Purchase correlation

In [40]:
```
fig = px.imshow(df.groupby(["Education", "Occupation"])["SalesAmount"].mean().unstack(
                labels=dict(color="Average Purchase"))
fig.show()
```

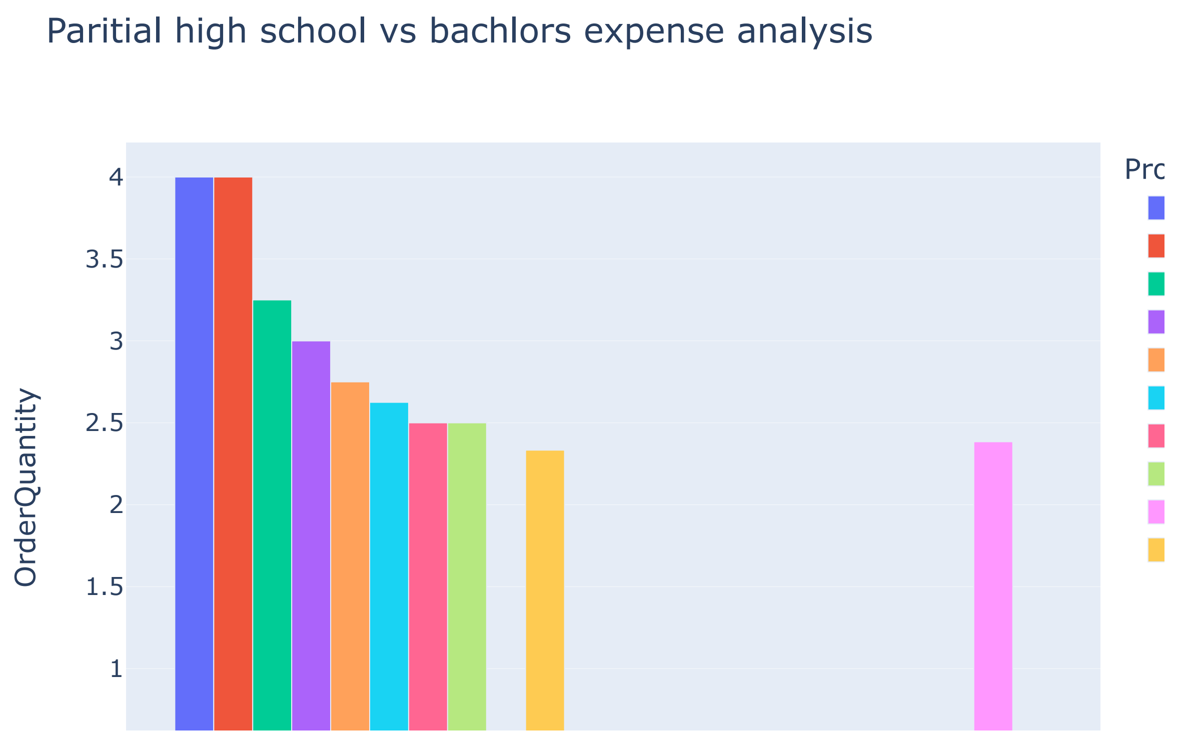


```
In [41]: df_6 = df[(df['Education']=='Partial High School')|(df['Education']=='Bachelors')].grc
         df_6.reset_index(inplace=True)
         sns.barplot(df_6, x='Education', y='YearlyIncome')
```

```
Out[41]: <Axes: xlabel='Education', ylabel='YearlyIncome'>
```



In [42]:
```
df_7 = df[(df['Education']=='Partial High School')|(df['Education']=='Bachelors')]
df_7 = df_7.groupby(['Education','ProductName'])['OrderQuantity'].mean().to_frame().sc
df_7.reset_index(inplace=True)
fig = px.bar(df_7, x="Education",
             y="OrderQuantity", color="ProductName",
             title="Paritial high school vs bachlors expense analysis",
             barmode="group")
fig.show()
```

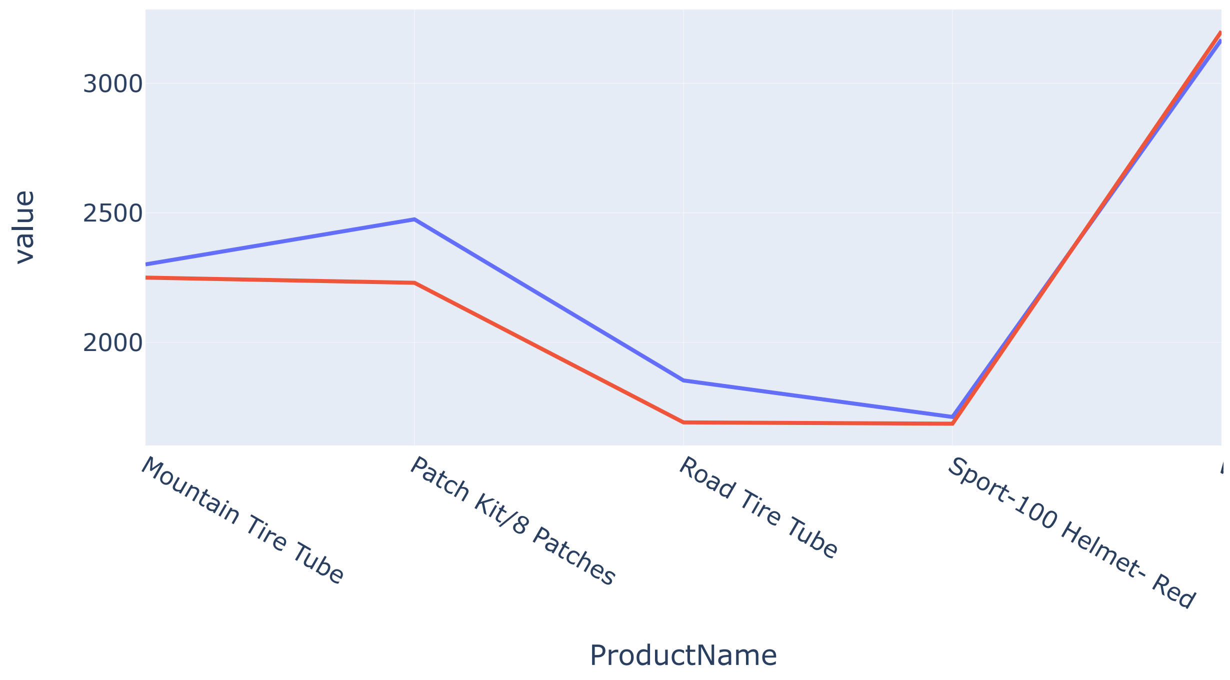


- Customers with a **high school diploma and modest annual income buy more products** than people with bachelor's degrees

Compare most ordered product by gender

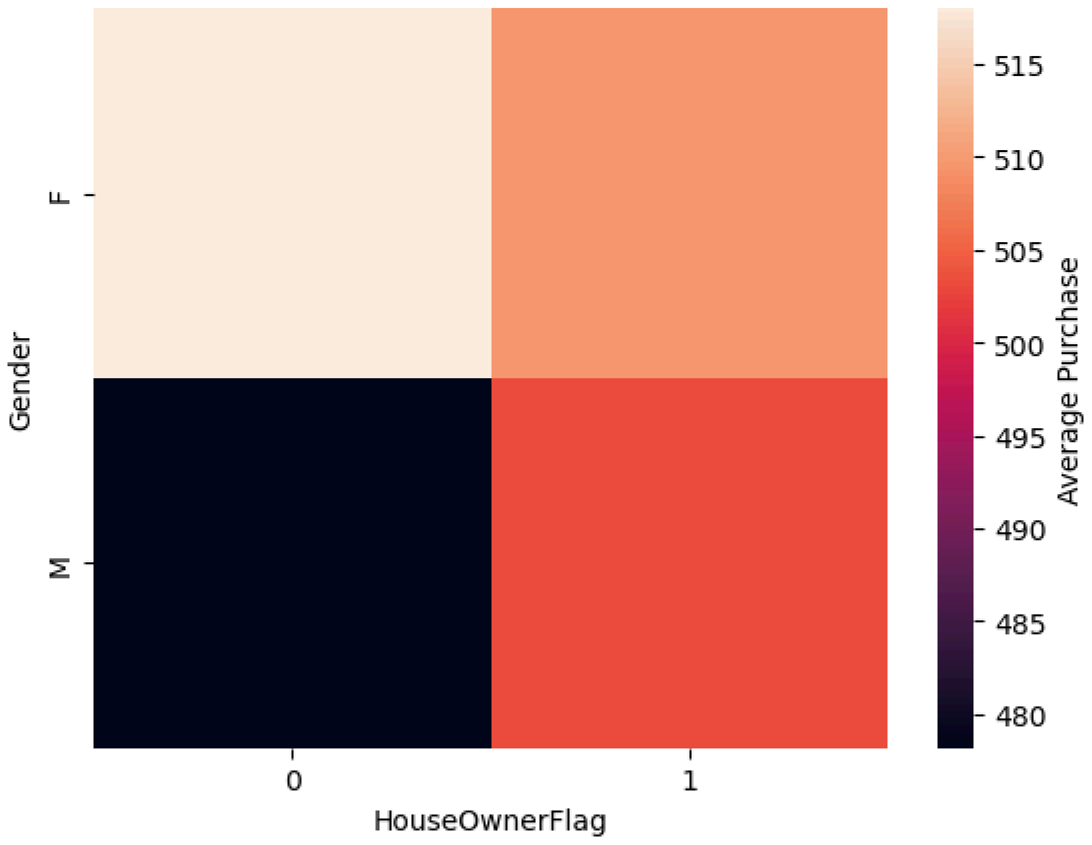
```
In [43]: male = df[df["Gender"]=="M"]
         female = df[df["Gender"]=="F"]


         male_ord_qty = male.groupby(['ProductName'],as_index=False)['OrderQuantity'].sum().nla
         male_ord_qty.columns=['ProductName','Order_Qty_Male']


         female_ord_qty = female.groupby(['ProductName'],as_index=False)['OrderQuantity'].sum()
         female_ord_qty.columns=['ProductName','Order_Qty_Female']


         df_merge = pd.merge(male_ord_qty, female_ord_qty, on='ProductName')
```

```
In [44]: px.line(df_merge, x="ProductName", y=["Order_Qty_Male","Order_Qty_Female"], width=800,
             height=400)
```



```
In [45]: sns.heatmap(df.groupby(["Gender", "HouseOwnerFlag"])["SalesAmount"].mean().unstack(),
                     cbar_kws={'label': 'Average Purchase'}
                 )
```

```
Out[45]: <Axes: xlabel='HouseOwnerFlag', ylabel='Gender'>
```



- It's interesting to note that the average amount spent by men without permanent addresses is low, whilst the average amount spent by women without permanent addresses is higher.

```
In [46]: df.to_csv('BudgetSalesData.csv')
```

```
In [ ]:
```