In [73]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import plotly.figure_factory as ff
import seaborn as sns
import datetime as dt
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df=pd.read_excel(r"C:\Users\kruna\OneDrive\Desktop\internship\badget sales\AdventureWorks_Database.xlsx")
df.head()
```

Out[4]:

	Date	DateKey	Year	Quarter	MonthNum	Month	FiscalYear	FiscalQuarter	FiscalMonthNum	FiscalMonth	MonthYear	Month
0	2016-04-03	20160403	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	
1	2016-04-04	20160404	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	
2	2016-04-05	20160405	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	
3	2016-04-06	20160406	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	
4	2016-04-07	20160407	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	

In excel there are multiple sheets available. So we will execute it one by one

In [7]:
```
Customers_data = pd.read_excel(r"C:\Users\kruna\OneDrive\Desktop\internship\badget sales\AdventureWorks_Databas
                               'Customers',
                               dtype={'CustomerKey':str},
                               parse_dates=['BirthDate','DateFirstPurchase'])
```

In [9]:
```
Product_data = pd.read_excel(r"C:\Users\kruna\OneDrive\Desktop\internship\badget sales\AdventureWorks_Database.
                             'Product',
                             dtype={'ProductKey':str},
                             parse_dates=['StartDate'])
```

In [11]:
```
Sales_data = pd.read_excel(r"C:\Users\kruna\OneDrive\Desktop\internship\badget sales\AdventureWorks_Database.xl
                           'Sales',
                              dtype={'ProductKey':str,
                                     'CustomerKey':str,
                                     'PromotionKey':str,
                                     'SalesTerritoryKey':str},
                              parse_dates=['OrderDate', 'ShipDate']
                              )
Sales_data['DateKey'] = Sales_data['OrderDate'].astype(str)
```

In [12]:
```
Territory_data = pd.read_excel(r"C:\Users\kruna\OneDrive\Desktop\internship\badget sales\AdventureWorks_Databas
                               'Territory',
                               dtype={'SalesTerritoryKey':str}
                               )
```

In [14]:
```
Customers_data.head()
```

Out[14]:

	CustomerKey	FirstName	LastName	FullName	BirthDate	MaritalStatus	Gender	YearlyIncome	TotalChildren	NumberChildrenA
0	11000	Jon	Yang	Yang, Jon	1966-04-08	M	M	90000	2	
1	11001	Eugene	Huang	Huang, Eugene	1965-05-14	S	M	60000	3	
2	11002	Ruben	Torres	Torres, Ruben	1965-08-12	M	M	60000	3	
3	11003	Christy	Zhu	Zhu, Christy	1968-02-15	S	F	70000	0	
4	11004	Elizabeth	Johnson	Johnson, Elizabeth	1968-08-08	S	F	80000	5	

In [15]:
```
Product_data.head()
```

Out[15]:

	ProductKey	ProductName	SubCategory	Category	StandardCost	Color	ListPrice	DaysToManufacture	ProductLine	ModelNam
0	1	Adjustable Race	NaN	NaN	NaN	NaN	NaN	0	NaN	Na
1	2	Bearing Ball	NaN	NaN	NaN	NaN	NaN	0	NaN	Na
2	3	BB Ball Bearing	NaN	NaN	NaN	NaN	NaN	1	NaN	Na
3	4	Headset Ball Bearings	NaN	NaN	NaN	NaN	NaN	0	NaN	Na
4	5	Blade	NaN	NaN	NaN	NaN	NaN	1	NaN	Na

In [16]:
```python
Sales_data.head()
```

Out[16]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1
1	346	2014-01-01	2014-01-08	28389	1	7	SO43698	1
2	346	2014-01-01	2014-01-08	25863	1	1	SO43699	1
3	336	2014-01-01	2014-01-08	14501	1	4	SO43700	1
4	346	2014-01-01	2014-01-08	11003	1	9	SO43701	1

5 rows × 26 columns

In [17]:
```python
Territory_data.head()
```

Out[17]:

	SalesTerritoryKey	Region	Country	Group	RegionImage
0	1	Northwest	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
1	2	Northeast	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
2	3	Central	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
3	4	Southwest	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
4	5	Southeast	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...

In [18]:
```python
temp_data = pd.merge(Sales_data, Product_data, on='ProductKey', how='inner')
df = pd.merge(temp_data, Customers_data, on='CustomerKey', how='inner')
df = pd.merge(df, Territory_data, on='SalesTerritoryKey', how='inner')
```

In [19]:
```python
df.head()
```

Out[19]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3

5 rows × 58 columns

In [20]:
```python
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
```

Removing Columns which are not required for further analysis

In [21]:
```
columns_to_drop = ['Unnamed: 13','Unnamed: 14','Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18', 'Unna
df = df.drop(columns_to_drop, axis=1)
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 46 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  DateKey               58189 non-null  object        
 14  ProductName           58189 non-null  object        
 15  SubCategory           58189 non-null  object        
 16  Category              58189 non-null  object        
 17  StandardCost_y        58189 non-null  float64       
 18  Color                 30747 non-null  object        
 19  ListPrice             58189 non-null  float64       
 20  DaysToManufacture     58189 non-null  int64         
 21  ProductLine           58189 non-null  object        
 22  ModelName             58189 non-null  object        
 23  Photo                 58189 non-null  object        
 24  ProductDescription    58189 non-null  object        
 25  StartDate             58189 non-null  datetime64[ns]
 26  FirstName             58189 non-null  object        
 27  LastName              58189 non-null  object        
 28  FullName              58189 non-null  object        
 29  BirthDate             58189 non-null  datetime64[ns]
 30  MaritalStatus         58189 non-null  object        
 31  Gender                58189 non-null  object        
 32  YearlyIncome          58189 non-null  int64         
 33  TotalChildren         58189 non-null  int64         
 34  NumberChildrenAtHome  58189 non-null  int64         
 35  Education             58189 non-null  object        
 36  Occupation            58189 non-null  object        
 37  HouseOwnerFlag        58189 non-null  int64         
 38  NumberCarsOwned       58189 non-null  int64         
 39  AddressLine1          58189 non-null  object        
 40  DateFirstPurchase     58189 non-null  datetime64[ns]
 41  CommuteDistance       58189 non-null  object        
 42  Region                58189 non-null  object        
 43  Country               58189 non-null  object        
 44  Group                 58189 non-null  object        
 45  RegionImage           58189 non-null  object        
dtypes: datetime64[ns](5), float64(6), int64(8), object(27)
memory usage: 20.4+ MB
```

In [26]:
```
df.describe().transpose()
```

Out[26]:

	count	mean	min	25%	50%	75%	max	std
OrderDate	58189	2016-06-03 03:56:09.605939200	2014-01-01 00:00:00	2016-04-01 00:00:00	2016-07-07 00:00:00	2016-10-10 00:00:00	2016-12-30 00:00:00	NaN
ShipDate	58189	2016-06-10 04:03:24.657237760	2014-01-08 00:00:00	2016-04-08 00:00:00	2016-07-14 00:00:00	2016-10-17 00:00:00	2017-01-07 00:00:00	NaN
SalesOrderLineNumber	58189.0	1.887453	1.0	1.0	2.0	2.0	8.0	1.018829
OrderQuantity	58189.0	1.569386	1.0	1.0	1.0	2.0	4.0	1.047532
UnitPrice	58189.0	413.888218	0.5725	4.99	24.49	269.995	3578.27	833.052938
TotalProductCost	58189.0	296.539185	0.8565	3.3623	12.1924	343.6496	2171.2942	560.171436
SalesAmount	58189.0	503.66627	2.29	8.99	32.6	539.99	3578.27	941.462817
TaxAmt	58189.0	40.293303	0.1832	0.7192	2.608	43.1992	286.2616	75.317027
StandardCost_y	58189.0	296.539185	0.8565	3.3623	12.1924	343.6496	2171.2942	560.171436
ListPrice	58189.0	503.66627	2.29	8.99	32.6	539.99	3578.27	941.462817
DaysToManufacture	58189.0	1.045215	0.0	0.0	0.0	4.0	4.0	1.757395
StartDate	58189	2007-05-14 02:44:51.848974848	2005-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	NaN
BirthDate	58189	1962-03-02 12:33:19.305710720	1910-08-13 00:00:00	1954-12-20 00:00:00	1963-09-19 00:00:00	1970-07-08 00:00:00	1980-12-26 00:00:00	NaN
YearlyIncome	58189.0	59769.887779	10000.0	30000.0	60000.0	80000.0	170000.0	33128.041818
TotalChildren	58189.0	1.838921	0.0	0.0	2.0	3.0	5.0	1.614467
NumberChildrenAtHome	58189.0	1.073502	0.0	0.0	0.0	2.0	5.0	1.580055
HouseOwnerFlag	58189.0	0.69056	0.0	0.0	1.0	1.0	1.0	0.462267
NumberCarsOwned	58189.0	1.502466	0.0	1.0	2.0	2.0	4.0	1.155496
DateFirstPurchase	58189	2015-12-23 02:50:33.356820224	2014-01-01 00:00:00	2015-06-21 00:00:00	2016-03-12 00:00:00	2016-07-26 00:00:00	2016-12-30 00:00:00	NaN

Duplicate data

```
In [27]: df.duplicated().sum()
```

```
Out[27]: 0
```

Missing data

```
In [28]: df.isna().sum()
```

```
Out[28]: 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
         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
```

Drop Column in which data is missing

```
In [29]: df = df.dropna(axis=1)
```

```
In [30]: df.isna().sum()
```

```
Out[30]: 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
         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
```

Adding Columns for Better Analysis

```
In [31]: 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
```

List of product's category

```
In [32]: df['Category'].unique().tolist()
```

```
Out[32]: ['Bikes', 'Accessories', 'Clothing']
```

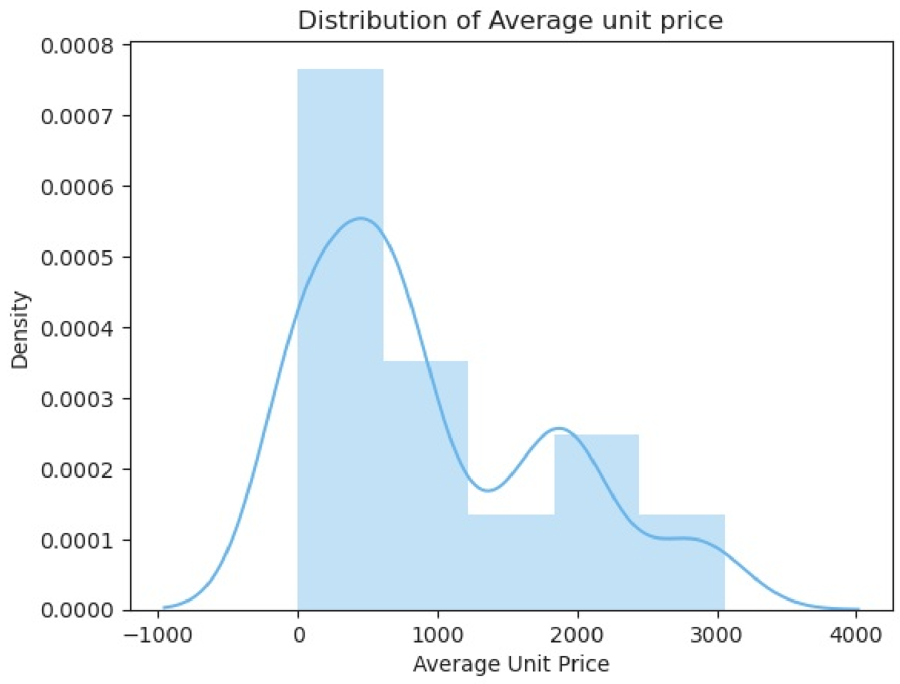
Product's subcategory

```
In [33]: df['SubCategory'].unique().tolist()
```

```
Out[33]: ['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']
```

Analysing Unit Price

```
In [38]: Avg_unit_price = df.groupby(['ProductKey'])['UnitPrice'].mean()
         ax = sns.distplot(Avg_unit_price, kde=True, hist=True, color='#53aae9')
         ax.set(title='Distribution of Average unit price',
                xlabel='Average Unit Price');
```

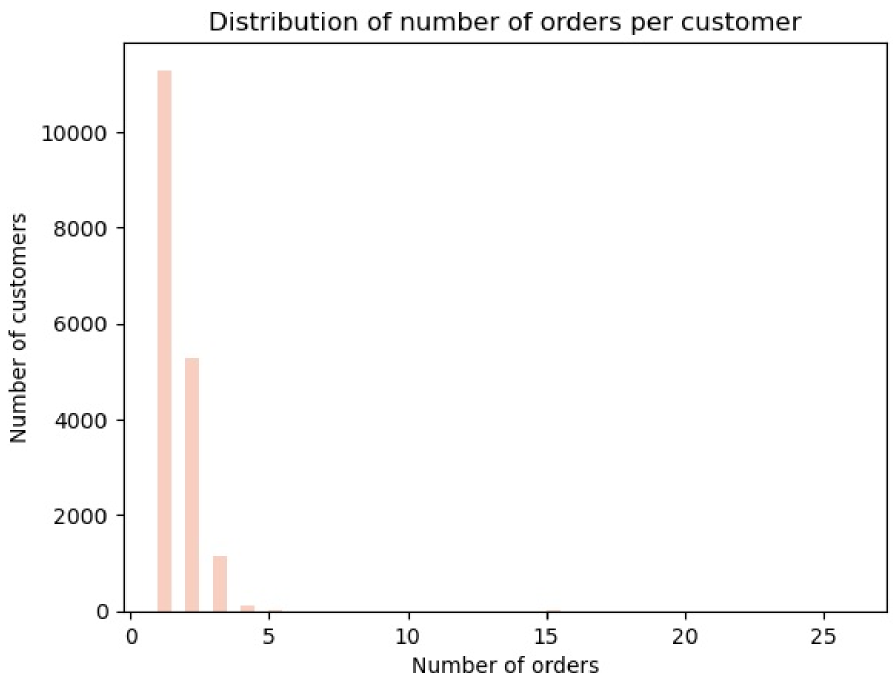


Maximum product unit price is below $1000Sales order number distribution

```
In [39]: n_orders = df.groupby(['CustomerKey'])['SalesOrderNumber'].nunique()
         multi_orders_perc = np.sum(n_orders > 1)/df['CustomerKey'].nunique()
         print(f"{100*multi_orders_perc:.2f}% of customers ordered more than once.")
```

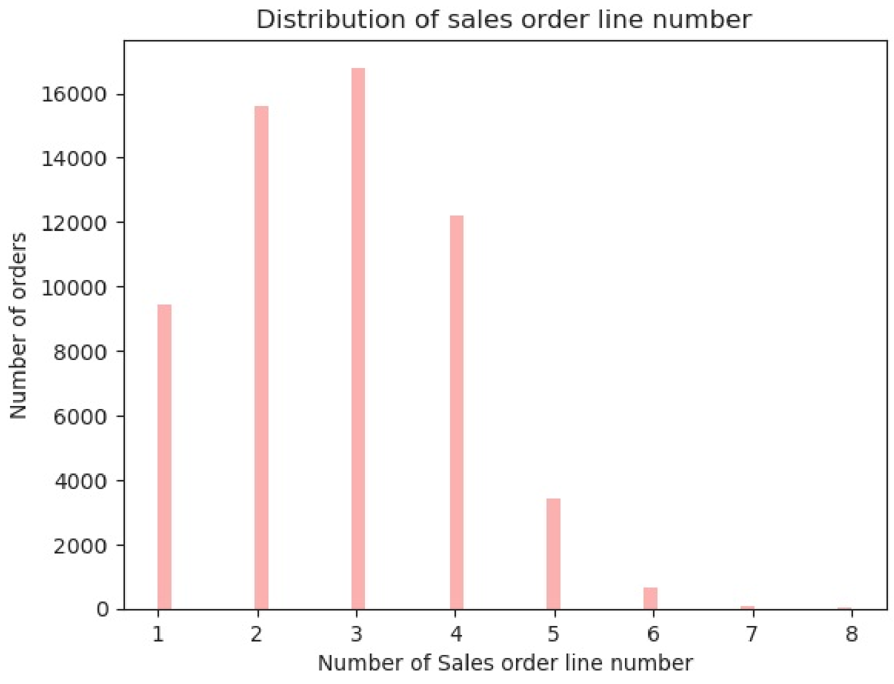
```
36.97% of customers ordered more than once.
```

```
In [40]: ax = sns.distplot(n_orders, kde=False, color='#ec8660')
         ax.set(title='Distribution of number of orders per customer',
                xlabel='Number of orders',
                ylabel='Number of customers');
```



Sales order line number distribution

```
In [42]: n_salesordernumber = df.groupby(['SalesOrderNumber'])['SalesOrderLineNumber'].transform('max')
         ax = sns.distplot(n_salesordernumber, kde=False, color='#f91e1e')
         ax.set(title='Distribution of sales order line number',
                xlabel='Number of Sales order line number',
                ylabel='Number of orders');
```

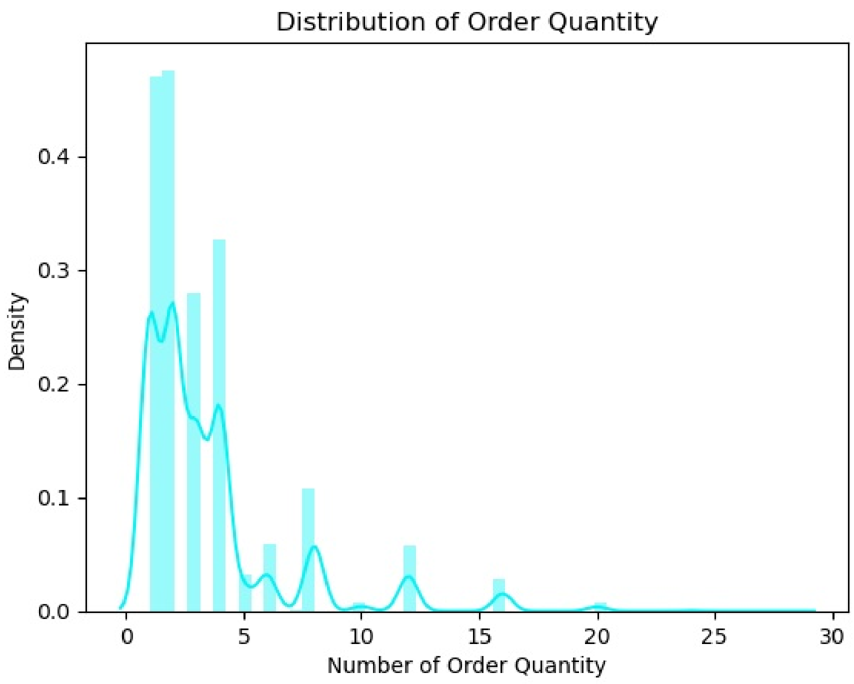


Most of the time two - three products are ordered in a single order

```
In [ ]:
```

Sales Order Quantity distribution

```
In [46]: n_order_quantity = df.groupby(['SalesOrderNumber'])['OrderQuantity'].sum()
         ax = sns.distplot(n_order_quantity, kde=True, hist=True,color='#02f4f7')
         ax.set(title='Distribution of Order Quantity',
                xlabel='Number of Order Quantity',
                );
```



Maximum quantity ordered for a product is below 5

```
In [ ]:
```

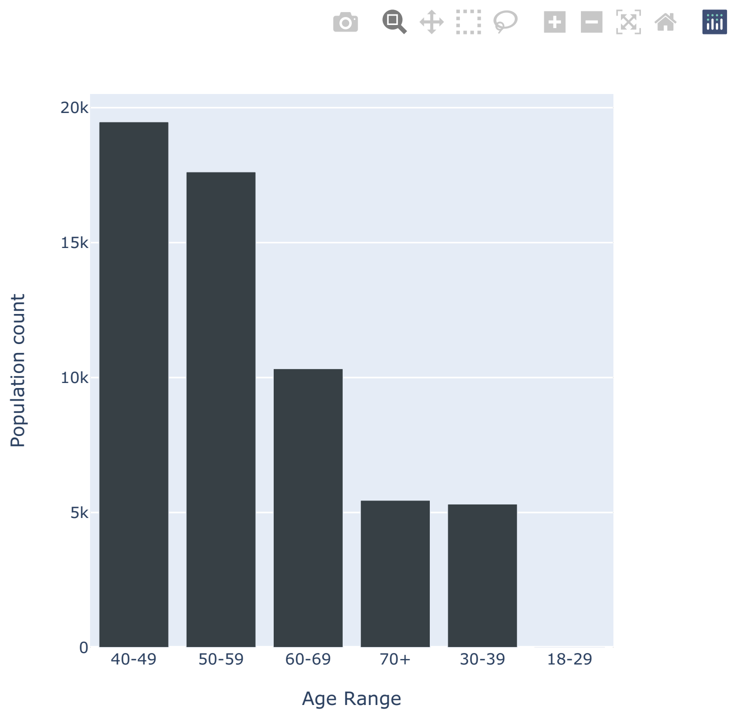
Age Distribution

```
In [48]: 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']

         fig = px.bar(age_distribution, x='Age Range', y='Population count', color_discrete_sequence=['#374045'])
         fig.update_layout(
             autosize=True,
             width=500,
             height=500,
             font=dict(size=10))
         fig.show()
```

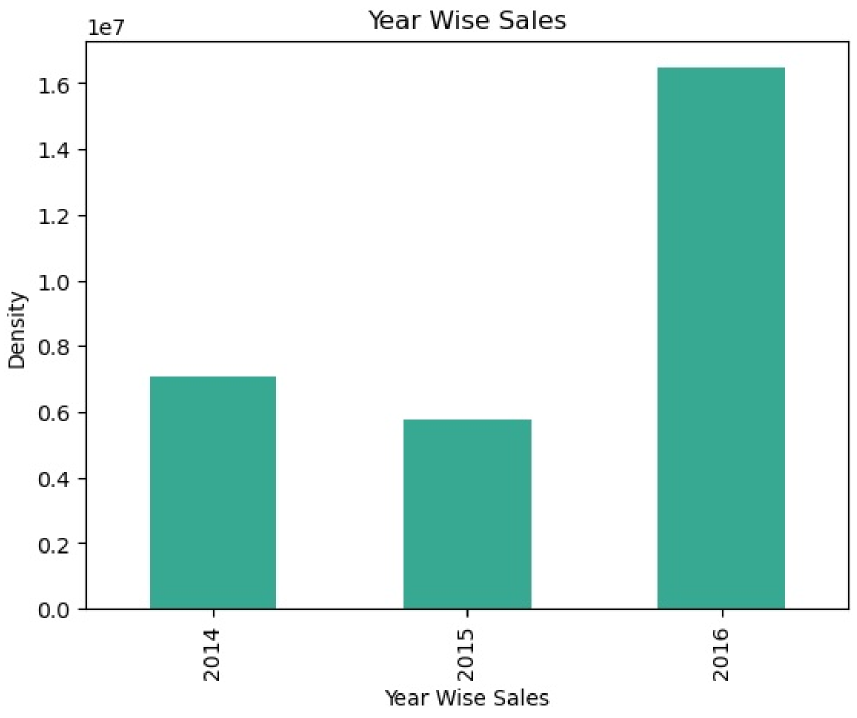


sizable portion of the clientele is made up of people between the ages of 40 and 59.

```
In [ ]:
```

Year Wise Sales

```
In [53]: df.groupby('sale_year')['SalesAmount'].sum().plot(kind='bar',
                                    color='#37a892',
                                    title = 'Year Wise Sales',
                                    xlabel= 'Year Wise Sales',
                                   ylabel= 'Density');
```



In [ ]:
```
The year 2016 saw an exponential rise in sales
```

In [ ]:
```

```

Top 5 Selling Product

In [57]:
```
top_selling_product = df.groupby(['Category', 'SubCategory', 'ProductName'])['OrderQuantity'].sum().nlargest(5)
top_selling_product
```

Out[57]:

			OrderQuantity
Category	**SubCategory**	**ProductName**	
Accessories	**Bottles and Cages**	**Water Bottle - 30 oz.**	6370
	Tires and Tubes	**Patch Kit/8 Patches**	4705
		Mountain Tire Tube	4551
		Road Tire Tube	3544
	Helmets	**Sport-100 Helmet- Red**	3398

In [ ]:
```

```

Quantity ordered based on category and subcategory from 2014 to 2016

In [58]:
```
cat_subcat_qty = df.groupby(['sale_year','Category', 'SubCategory'])['OrderQuantity'].sum().to_frame()
cat_subcat_qty = cat_subcat_qty.sort_values(['sale_year', 'Category'], ascending=True)
cat_subcat_qty.style.bar(subset=['OrderQuantity'], color='#D9B300')
```

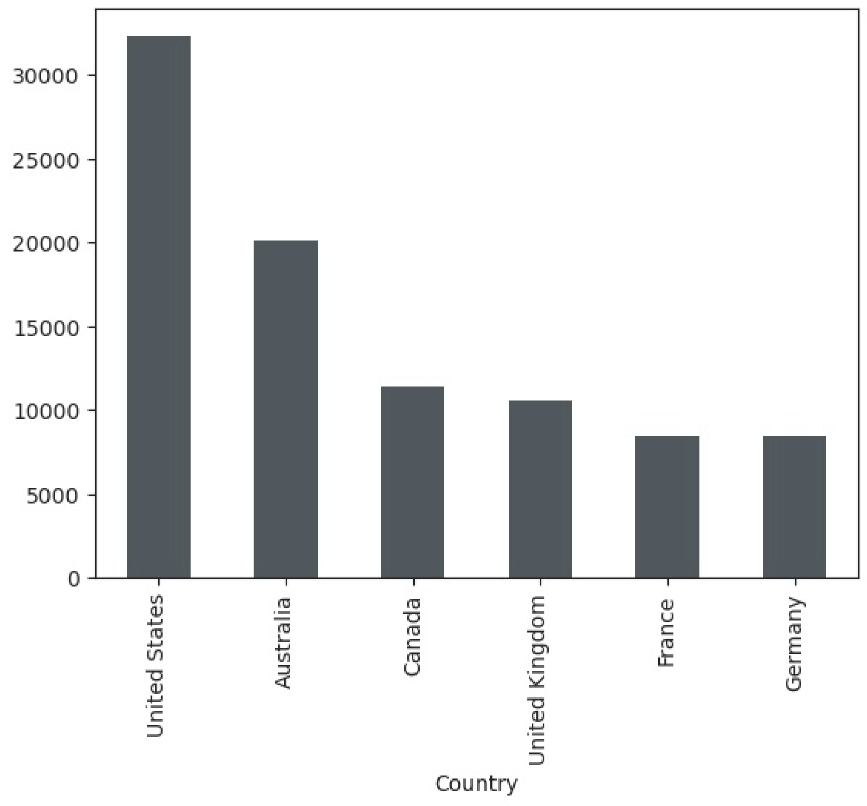
Out[58]:

sale_year	Category	SubCategory	OrderQuantity
2014	Bikes	Mountain Bikes	616
		Road Bikes	
2015	Bikes	Mountain Bikes	
		Road Bikes	
2016	Accessories	Bike Racks	
		Bike Stands	
		Bottles and Cages	
		Cleaners	
		Fenders	
		Helmets	
		Hydration Packs	
		Tires and Tubes	25518
	Bikes	Mountain Bikes	
		Road Bikes	
		Touring Bikes	
	Clothing	Caps	
		Gloves	
		Jerseys	
		Shorts	
		Socks	
		Vests	

In [ ]:

Country wise quantity ordered

In [63]:
```
country_qty_sales = df.groupby('Country')['OrderQuantity'].sum().sort_values(ascending=False)
country_qty_sales.plot(kind='bar', color='#374045');
```



In [ ]:

Overall profit based on order year, category and subcategory

In [68]:
```
cat_subcat_profit = df.groupby(['sale_year','Category', 'SubCategory'])['profit'].sum().to_frame()

#Sorting the results
```

```
cat_subcat_profit = cat_subcat_profit.sort_values(['sale_year', 'Category'], ascending=True)
cat_subcat_profit.style.bar(subset=['profit'], color='#62dee7')
```

Out[68]:

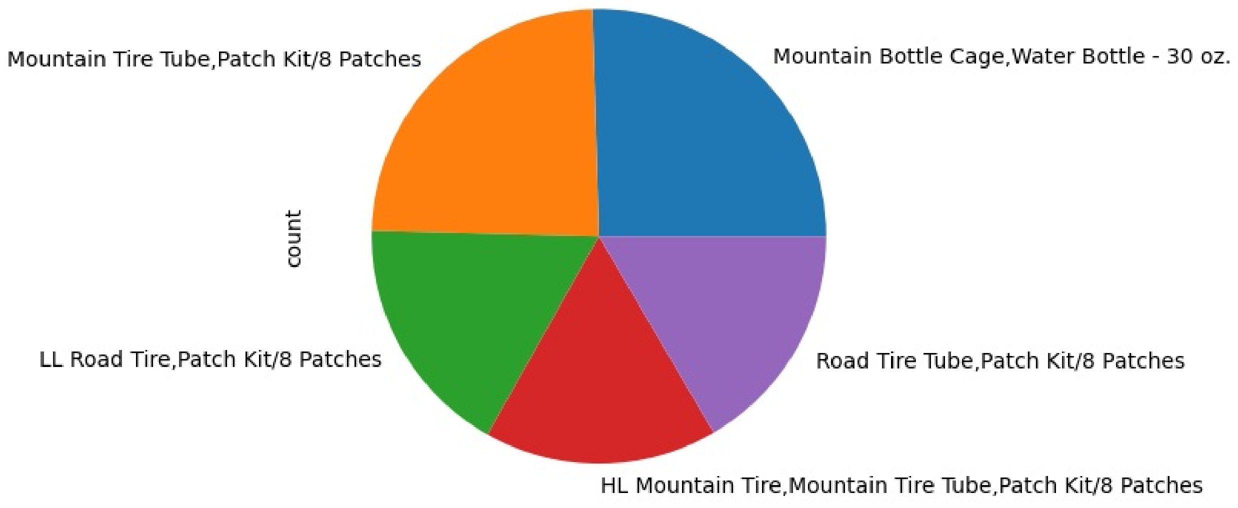
			profit
sale_year	**Category**	**SubCategory**	
2014	**Bikes**	**Mountain Bikes**	586874.557600
		Road Bikes	2256280.998
2015	**Bikes**	**Mountain Bikes**	10
		Road Bikes	1375
	Accessories	**Bike Racks**	
		Bike Stands	
		Bottles and Cages	
		Cleaners	
		Fenders	
		Helmets	
		Hydration Packs	
		Tires and Tubes	
2016	**Bikes**	**Mountain Bikes**	2907361.198000
		Road Bikes	1905953.
		Touring Bikes	14548
	Clothing	**Caps**	
		Gloves	
		Jerseys	
		Shorts	
		Socks	
		Vests	

In [ ]:

In [78]:
```
dup_order = df[df['SalesOrderNumber'].duplicated(keep=False)]
```

In [79]:
```
dup_order['grouped'] = df.groupby('SalesOrderNumber')['ProductName'].transform(lambda x: ','.join(x))
dup_order = dup_order[['SalesOrderNumber', 'grouped']].drop_duplicates()
```

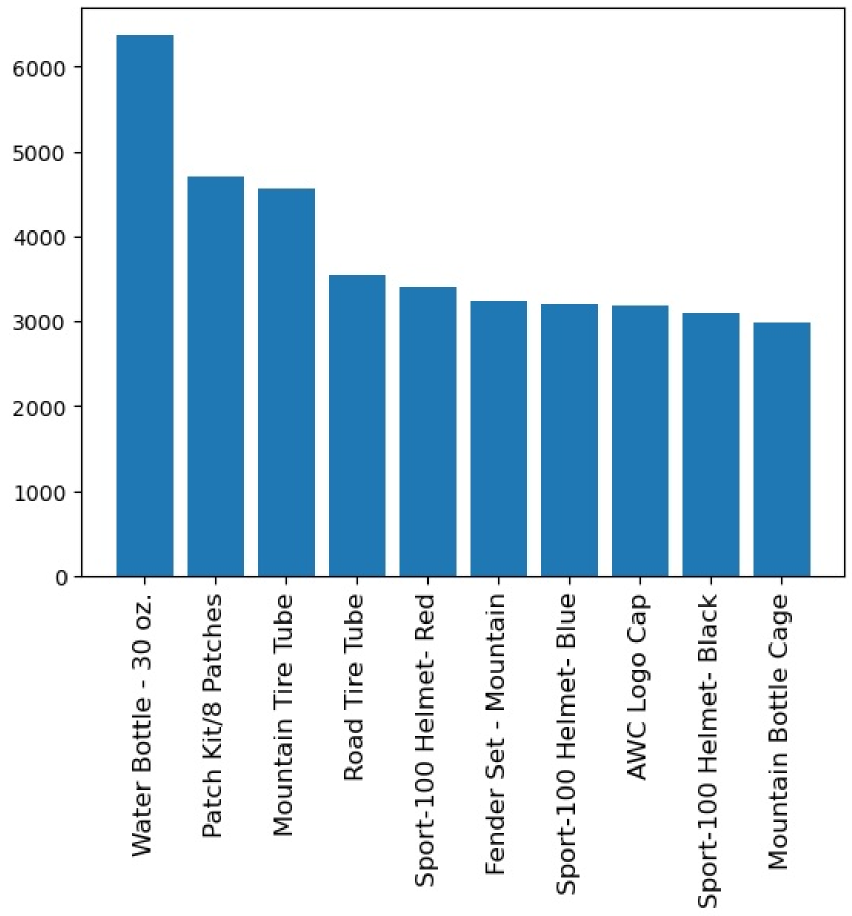
In [82]:
```
count = dup_order['grouped'].value_counts()[0:5].plot.pie()
```



sold the most

In [85]:
```
product_group = df.groupby('ProductName')
quantity_ordered = product_group['OrderQuantity'].sum().sort_values(ascending=False)[:10]
products = quantity_ordered.index.tolist()

plt.bar(products, quantity_ordered, )
plt.xticks(products, rotation='vertical', size=12)
plt.show()
```



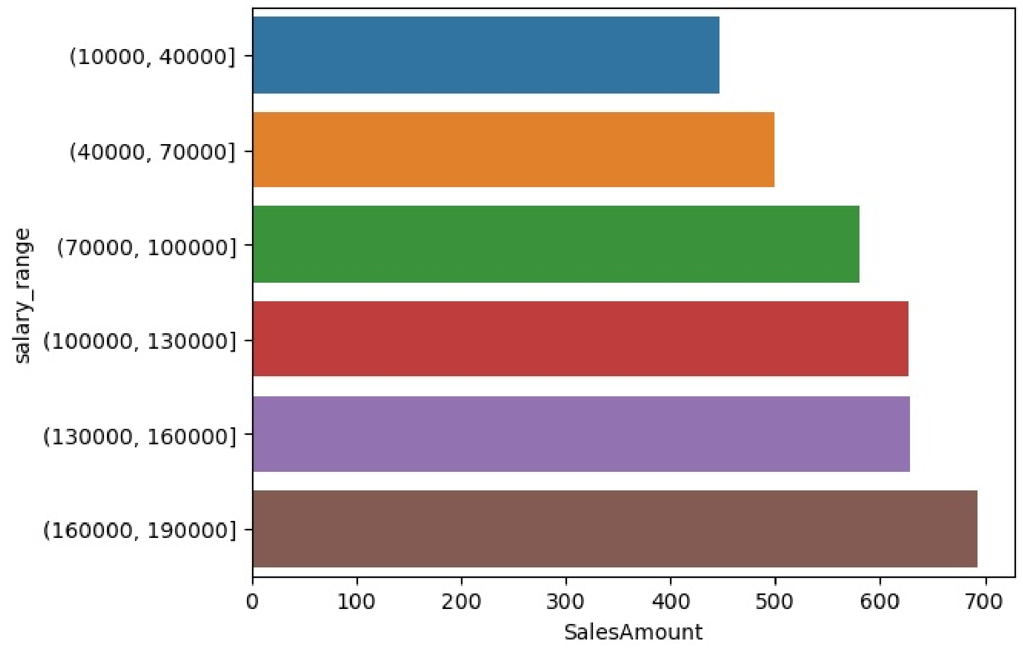
Yearly income range and purchase correlation

In [94]:
```
def create_bins(lower_bound, width, quantity):
    """ create_bins returns an equal-width (distance) partitioning.
        It returns an ascending list of tuples, representing the intervals.
        A tuple bins[i], i.e. (bins[i][0], bins[i][1])  with i > 0
        and i < quantity, satisfies the following conditions:
            (1) bins[i][0] + width == bins[i][1]
            (2) bins[i-1][0] + width == bins[i][0] and
                bins[i-1][1] + width == bins[i][1]
    """


    bins = []
    for low in range(lower_bound,
                     lower_bound + quantity*width + 1, width):
        bins.append((low, low+width))
    return bins
```

In [95]:
```
bins = create_bins(lower_bound=10000,
                   width=30000,
                   quantity=5)
bins2 = pd.IntervalIndex.from_tuples(bins)
df['salary_range'] = pd.cut(df['YearlyIncome'], bins2)
```

In [96]:
```
df_4 = df.groupby('salary_range')['SalesAmount'].mean().to_frame()
df_4.reset_index(inplace=True)
sns.barplot(x="SalesAmount", y="salary_range", data=df_4);
```



In [ ]:

In [1]:
```
conda install nbconvert
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

```

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.7.1

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=24.7.1

```

In [ ]: