```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns

Customers = pd.read_excel('C:\\Users\\hp\\OneDrive\\Desktop\\UNIFIED
PROJECTS\\Budget Sales data\\AdventureWorks_Database.xlsx',
                              'Customers',
                              dtype={'CustomerKey':str},

parse_dates=['BirthDate','DateFirstPurchase']
                              )

Product = pd.read_excel('C:\\Users\\hp\\OneDrive\\Desktop\\UNIFIED
PROJECTS\\Budget Sales data\\AdventureWorks_Database.xlsx',
                              'Product',
                              dtype={'ProductKey':str},
                              parse_dates=['StartDate']
                              )

Sales = pd.read_excel('C:\\Users\\hp\\OneDrive\\Desktop\\UNIFIED
PROJECTS\\Budget Sales data\\AdventureWorks_Database.xlsx',
                              'Sales',
                              dtype={'ProductKey':str,
                                     'CustomerKey':str,
                                     'PromotionKey':str,
                                     'SalesTerritoryKey':str},
                              parse_dates=['OrderDate', 'ShipDate']
                              )
Sales['DateKey'] = Sales['OrderDate'].astype(str)

Territory = pd.read_excel('C:\\Users\\hp\\OneDrive\\Desktop\\UNIFIED
PROJECTS\\Budget Sales data\\AdventureWorks_Database.xlsx',
                              'Territory',
                              dtype={'SalesTerritoryKey':str}
                              )

temp_data = pd.merge(Sales, Product, on='ProductKey', how='inner')
df = pd.merge(temp_data, Customers, on='CustomerKey', how='inner')
df = pd.merge(df, Territory, on='SalesTerritoryKey', how='inner')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 26.2+ MB
```

```
df.describe().transpose()
```

```
                        count        mean         std        min  \
SalesOrderLineNumber  58189.0    1.887453    1.018829     1.0000

OrderQuantity         58189.0    1.569386    1.047532     1.0000

UnitPrice             58189.0  413.888218  833.052938     0.5725

TotalProductCost      58189.0  296.539185  560.171436     0.8565

SalesAmount           58189.0  503.666270  941.462817     2.2900

TaxAmt                58189.0   40.293303   75.317027     0.1832

Unnamed: 13               0.0         NaN         NaN        NaN

Unnamed: 14               0.0         NaN         NaN        NaN

Unnamed: 15           58189.0  503.666269  941.462815     2.2900

Unnamed: 16           58189.0    0.000001    0.000014     0.0000

Unnamed: 17               0.0         NaN         NaN        NaN

Unnamed: 18           58189.0   38.398254  667.349417 -5106.9068

Unnamed: 19               0.0         NaN         NaN        NaN

StandardCost_x        58189.0  296.539185  560.171436     0.8565

List Price            58189.0  503.666270  941.462817     2.2900

Unnamed: 22               0.0         NaN         NaN        NaN

diif std cost         58189.0    0.000000    0.000000     0.0000

diff list price       58189.0    0.000000    0.000000     0.0000
```

```
StandardCost_y       58189.0    296.539185    560.171436       0.8565
ListPrice            58189.0    503.666270    941.462817       2.2900
DaysToManufacture    58189.0      1.045215      1.757395       0.0000
YearlyIncome         58189.0  59769.887779  33128.041818   10000.0000
TotalChildren        58189.0      1.838921      1.614467       0.0000
NumberChildrenAtHome 58189.0      1.073502      1.580055       0.0000
HouseOwnerFlag       58189.0      0.690560      0.462267       0.0000
NumberCarsOwned      58189.0      1.502466      1.155496       0.0000

                           25%         50%         75%          max
SalesOrderLineNumber    1.0000      2.0000      2.0000       8.0000
OrderQuantity           1.0000      1.0000      2.0000       4.0000
UnitPrice               4.9900     24.4900    269.9950    3578.2700
TotalProductCost        3.3623     12.1924    343.6496    2171.2942
SalesAmount             8.9900     32.6000    539.9900    3578.2700
TaxAmt                  0.7192      2.6080     43.1992     286.2616
Unnamed: 13                NaN         NaN         NaN          NaN
Unnamed: 14                NaN         NaN         NaN          NaN
Unnamed: 15             8.9900     32.6000    539.9900    3578.2700
Unnamed: 16             0.0000      0.0000      0.0000       0.0003
Unnamed: 17                NaN         NaN         NaN          NaN
Unnamed: 18             1.4335      6.2537     21.9037    1487.8356
Unnamed: 19                NaN         NaN         NaN          NaN
StandardCost_x          3.3623     12.1924    343.6496    2171.2942
List Price              8.9900     32.6000    539.9900    3578.2700
Unnamed: 22                NaN         NaN         NaN          NaN
diif std cost           0.0000      0.0000      0.0000       0.0000
```

```
diff list price           0.0000      0.0000      0.0000       0.0000
StandardCost_y            3.3623     12.1924    343.6496    2171.2942
ListPrice                 8.9900     32.6000    539.9900    3578.2700
DaysToManufacture         0.0000      0.0000      4.0000       4.0000
YearlyIncome          30000.0000  60000.0000  80000.0000  170000.0000
TotalChildren             0.0000      2.0000      3.0000       5.0000
NumberChildrenAtHome      0.0000      0.0000      2.0000       5.0000
HouseOwnerFlag            0.0000      1.0000      1.0000       1.0000
NumberCarsOwned           1.0000      2.0000      2.0000       4.0000
```

Standard deviation is square root of variance.

```
df.duplicated().sum()
```

```
0
```

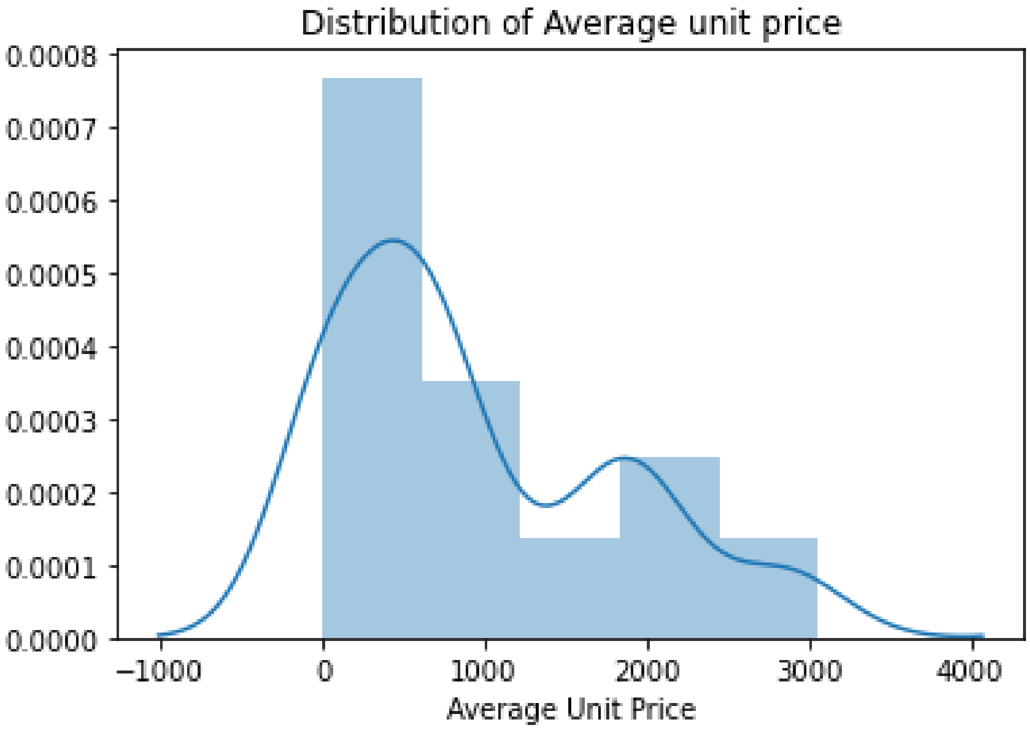
```
def missing_pct(df):
    # Calculate missing value and their percentage for each column
    missing_count_percent = df.isnull().sum() * 100 / df.shape[0]
    df_missing_count_percent = 
pd.DataFrame(missing_count_percent).round(2)
    df_missing_count_percent = 
df_missing_count_percent.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_Percentage (%)'
                    }
                )
    df_missing_value = df.isnull().sum()
    df_missing_value = df_missing_value.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_value_count'
                    }
                )
    # Sort the data frame
    #df_missing = df_missing.sort_values('Missing_Percentage (%)', 
ascending=False)
    Final = df_missing_value.merge(df_missing_count_percent, how = 
'inner', left_on = 'Column', right_on = 'Column')
    Final = Final.sort_values(by = 'Missing_Percentage (%)',ascending
= False)
    return Final

missing_pct(df)
```

```
                  Column  Missing_value_count  Missing_Percentage (%)
22           Unnamed: 22                58189                  100.00
19           Unnamed: 19                58189                  100.00
14           Unnamed: 14                58189                  100.00
13           Unnamed: 13                58189                  100.00
17           Unnamed: 17                58189                  100.00
30                 Color                27442                   47.16
0             ProductKey                    0                    0.00
42         MaritalStatus                    0                    0.00
41             BirthDate                    0                    0.00
39              LastName                    0                    0.00
40              FullName                    0                    0.00
38             FirstName                    0                    0.00
37             StartDate                    0                    0.00
36    ProductDescription                    0                    0.00
35                 Photo                    0                    0.00
34             ModelName                    0                    0.00
43                Gender                    0                    0.00
44          YearlyIncome                    0                    0.00
32     DaysToManufacture                    0                    0.00
45         TotalChildren                    0                    0.00
46  NumberChildrenAtHome                    0                    0.00
47             Education                    0                    0.00
48            Occupation                    0                    0.00
49        HouseOwnerFlag                    0                    0.00
50       NumberCarsOwned                    0                    0.00
51          AddressLine1                    0                    0.00
52     DateFirstPurchase                    0                    0.00
53       CommuteDistance                    0                    0.00
54                Region                    0                    0.00
55               Country                    0                    0.00
56                 Group                    0                    0.00
33           ProductLine                    0                    0.00
29         StandardCost_y                    0                    0.00
31             ListPrice                    0                    0.00
12                TaxAmt                    0                    0.00
2               ShipDate                    0                    0.00
3            CustomerKey                    0                    0.00
4           PromotionKey                    0                    0.00
5      SalesTerritoryKey                    0                    0.00
6       SalesOrderNumber                    0                    0.00
7   SalesOrderLineNumber                    0                    0.00
8          OrderQuantity                    0                    0.00
9              UnitPrice                    0                    0.00
10      TotalProductCost                    0                    0.00
```

```
11           SalesAmount                    0                    0.00
15          Unnamed: 15                     0                    0.00
1              OrderDate                    0                    0.00
16          Unnamed: 16                     0                    0.00
18          Unnamed: 18                     0                    0.00
20        StandardCost_x                    0                    0.00
21            List Price                    0                    0.00
23         diif std cost                    0                    0.00
24       diff list price                    0                    0.00
25               DateKey                    0                    0.00
26           ProductName                    0                    0.00
27           SubCategory                    0                    0.00
28              Category                    0                    0.00
57           RegionImage                    0                    0.00
```

```
df= df.dropna(axis=1)

# Extracting Year from OrderDate
df['sale_year'] = df['OrderDate'].dt.year

# Extracting Month from OrderDate
df['sale_month'] = df['OrderDate'].dt.month

# Extracting day from OrderDate
df['sale_day'] = df['OrderDate'].dt.day

# Extracting dayofweek from OrderDate
df['sale_week'] = df['OrderDate'].dt.dayofweek

# Extracting day_name from OrderDate
df['sale_day_name'] = df['OrderDate'].dt.day_name()

# Extracting Month Year from OrderDate
df['year_month'] = df['OrderDate'].apply(lambda x:x.strftime('%Y-%m'))

# Calculate Total Invoice Amount
df['total_Invoice_amount'] = df['SalesAmount'] + df['TaxAmt']

# Considering only salesamount and total_sales_amount to calculate
profit
df['profit'] = (df['UnitPrice']*df['OrderQuantity']) -
df['TotalProductCost']

# Removing extra character from the string
df['ProductName'] = df['ProductName'].str.replace(',','-')

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

df['Category'].unique().tolist()
df['SubCategory'].unique().tolist()
```

```
['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']

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

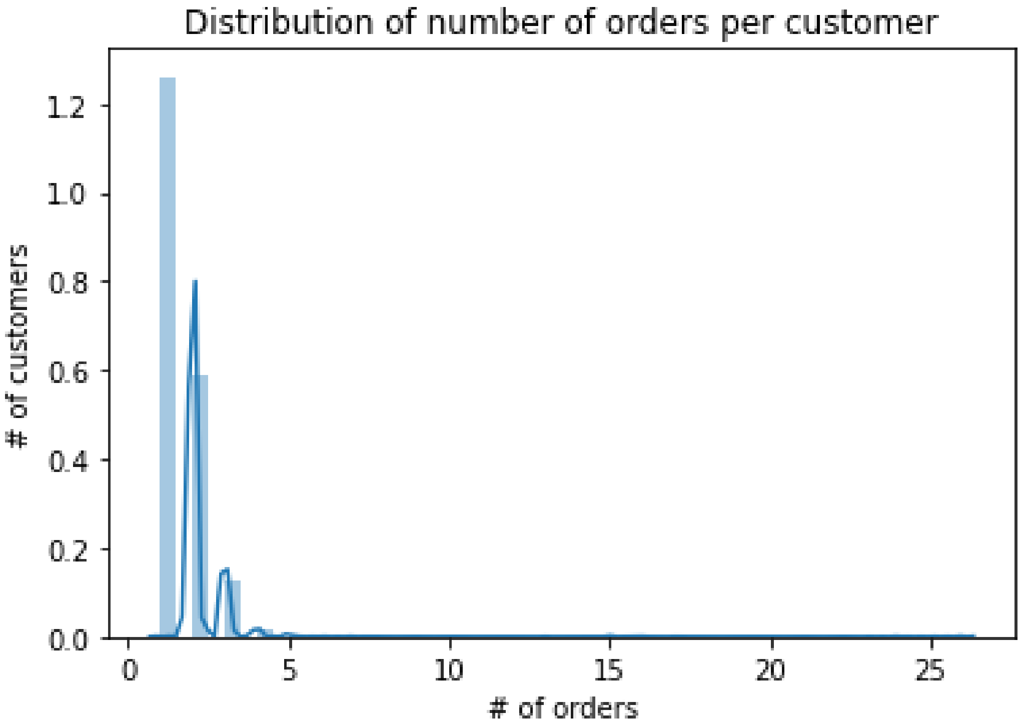


Unit product price is maximum between $0 $ to1000.

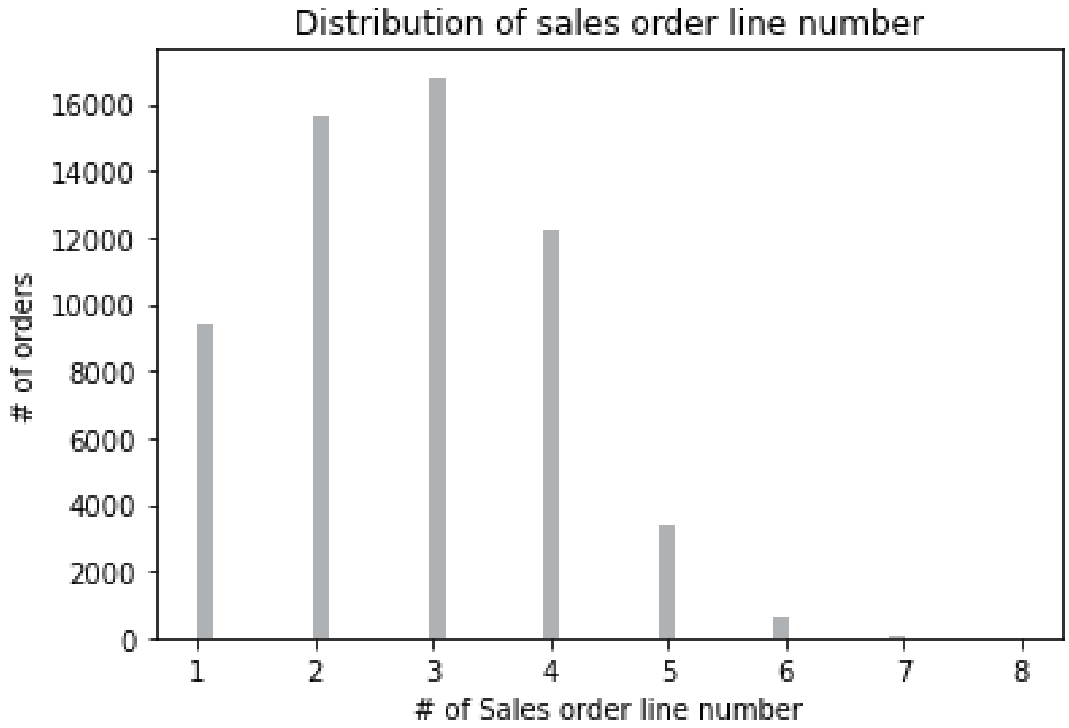
```
n_orders = df.groupby(['CustomerKey'])['SalesOrderNumber'].nunique()
multi_orders_perc = np.sum(n_orders > 1)/df['CustomerKey'].nunique()
print(f"{100*multi_orders_perc:.2f}% of customers ordered more than
once.")
```

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

```
ax = sns.distplot(n_orders)
ax.set(title='Distribution of number of orders per customer',
       xlabel='# of orders',
       ylabel='# of customers');
```

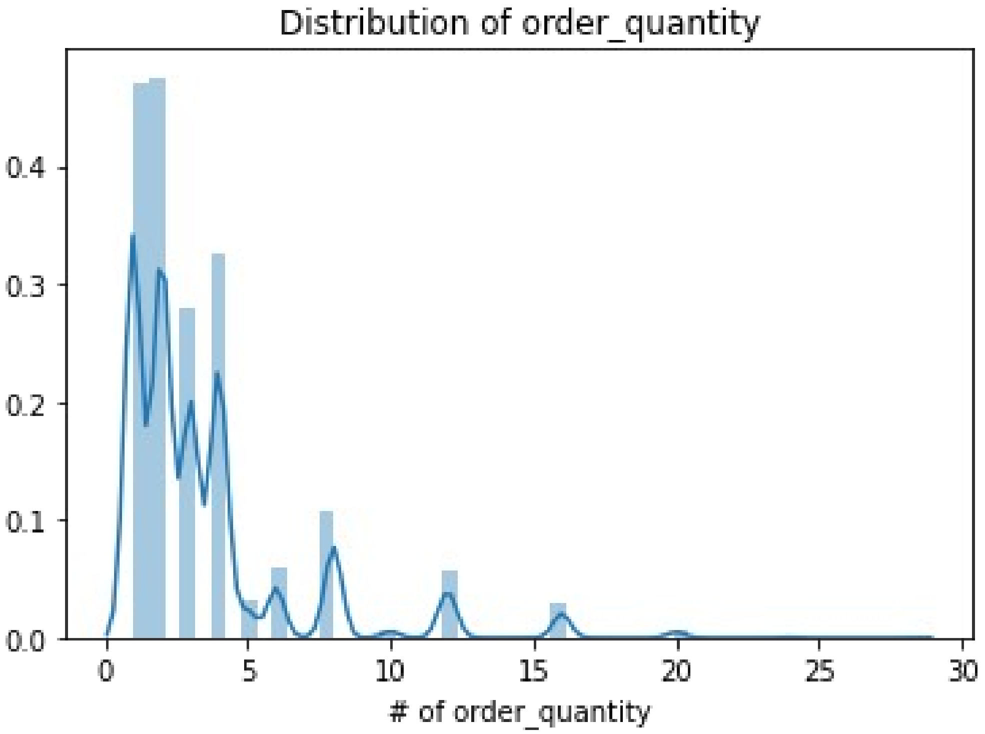


```
n_salesordernumber = df.groupby(['SalesOrderNumber'])
['SalesOrderLineNumber'].transform('max')
ax = sns.distplot(n_salesordernumber, kde=False, color='#374045')
ax.set(title='Distribution of sales order line number',
       xlabel='# of Sales order line number',
       ylabel='# of orders');
```



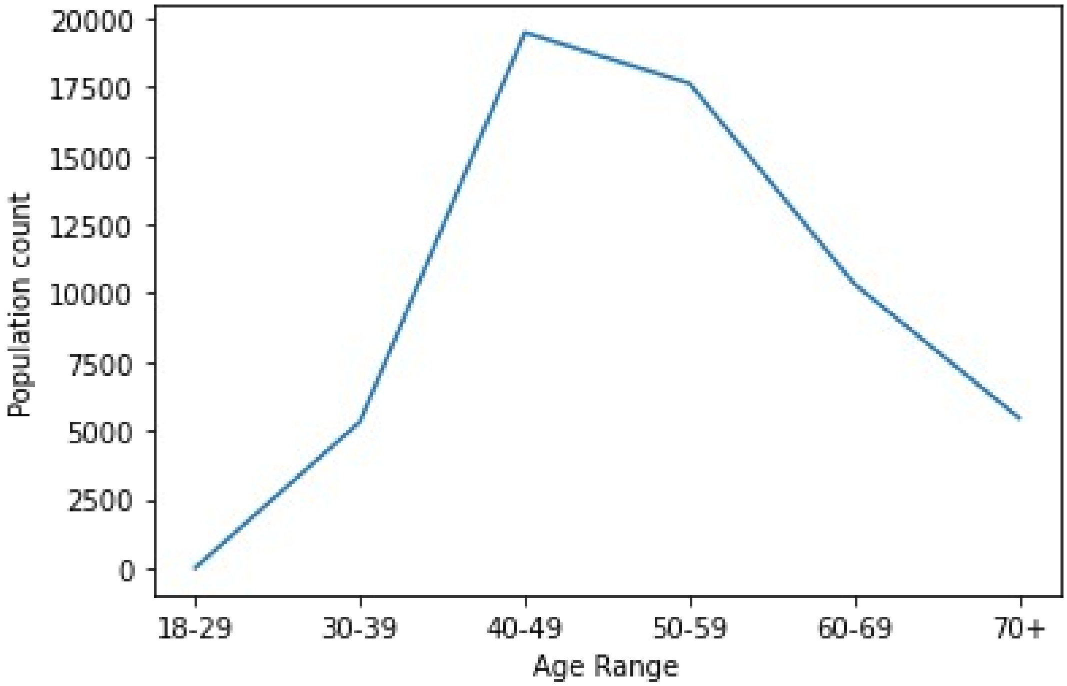
Three to two products are ordered in a single order most of the time.

```
n_order_quantity = df.groupby(['SalesOrderNumber'])
['OrderQuantity'].sum()
ax = sns.distplot(n_order_quantity, kde=True, hist=True)
ax.set(title='Distribution of order_quantity',
       xlabel='# of order_quantity',
       );
```



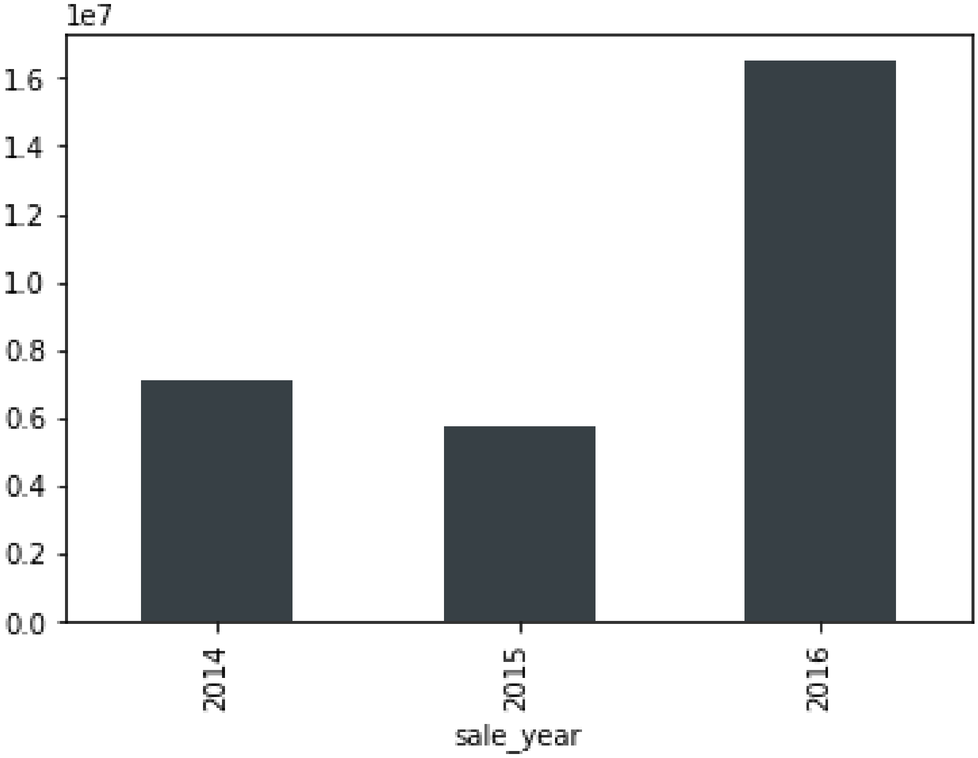
Maximum quantity ordered for a product is below 5

```
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']

sns.lineplot( x='Age Range', y='Population
count',data=age_distribution)
plt.show()
```



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

```
df.groupby('sale_year')['SalesAmount'].sum().plot(kind='bar',
color='#374045');
```

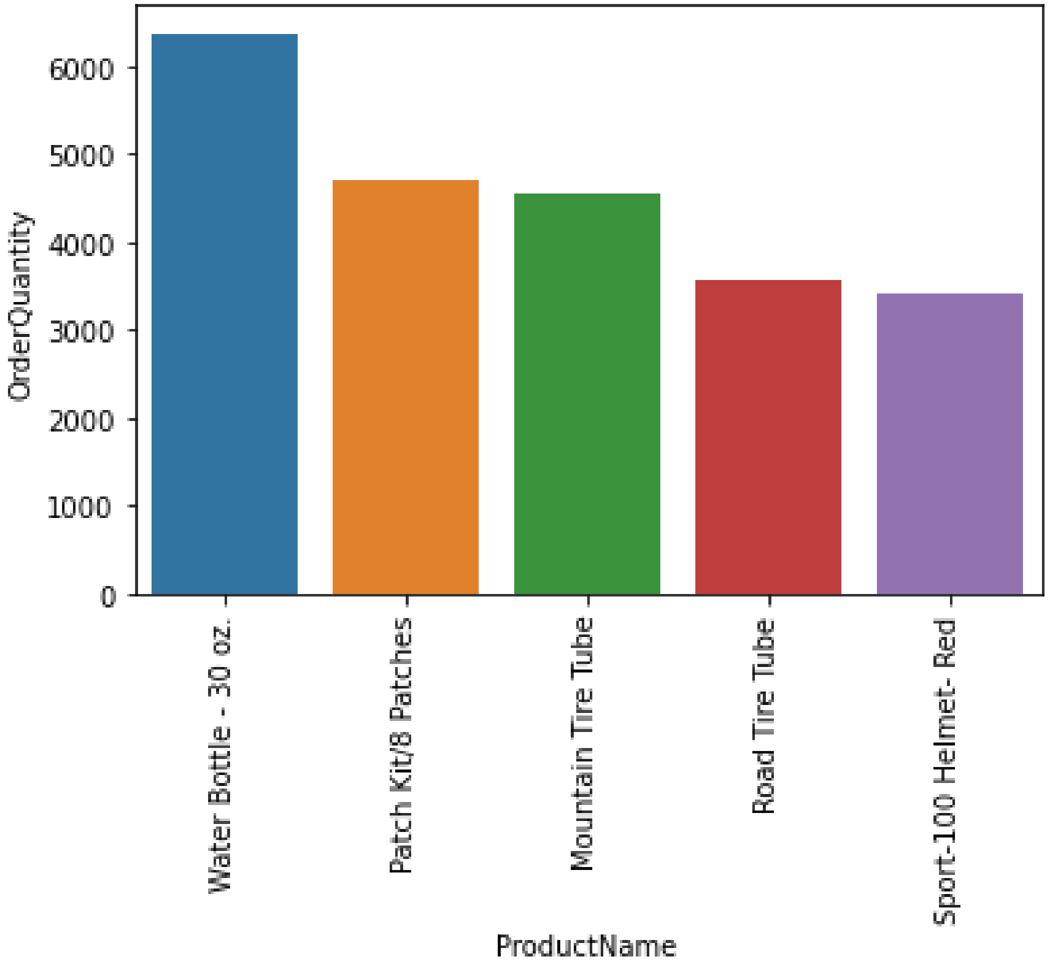


The year 2016 saw an exponential surge in sales.

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

```
                                                        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
```

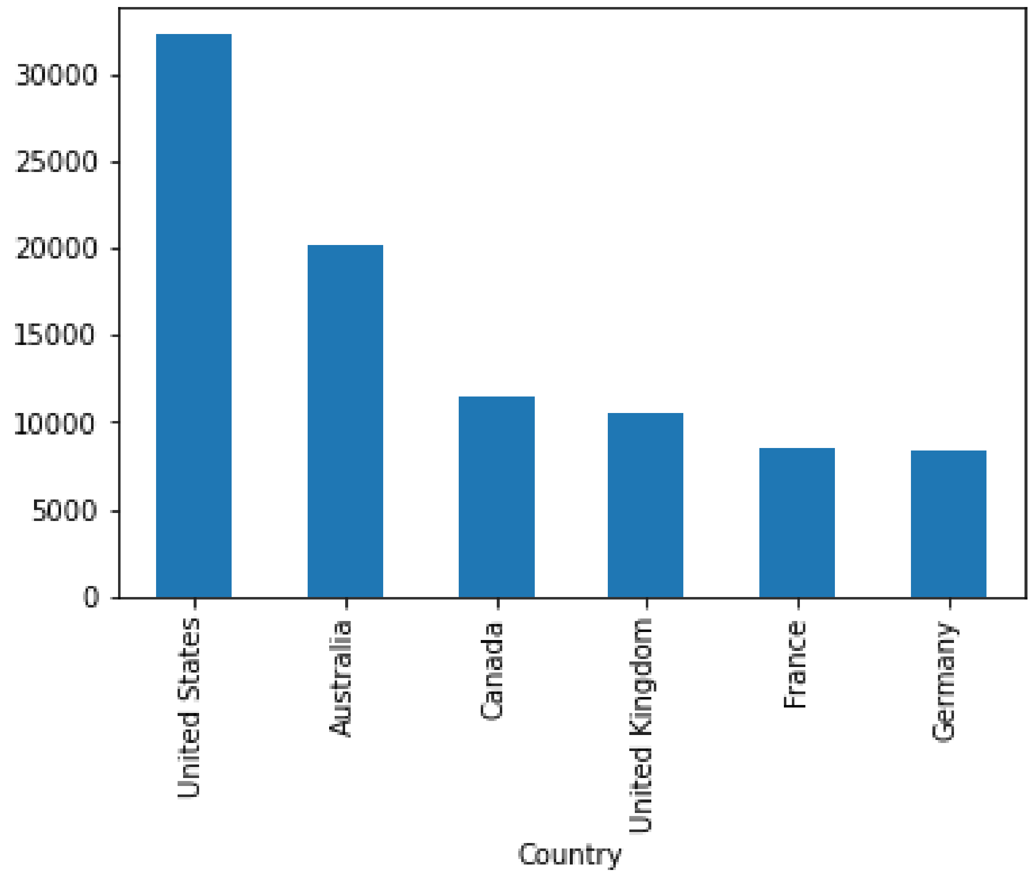
```
top_selling_product.reset_index(inplace=True)
sns.barplot(x='ProductName',
y='OrderQuantity',data=top_selling_product)
plt.xticks(rotation=90)
plt.show()
```



```
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'])
```

```
<pandas.io.formats.style.Styler at 0x2024bb47e50>
```

```
country_qty_sales = df.groupby('Country')
['OrderQuantity'].sum().sort_values(ascending=False)
country_qty_sales.plot(kind='bar');
```



High quantity of products is ordered from Australia and United States

```
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'])
```

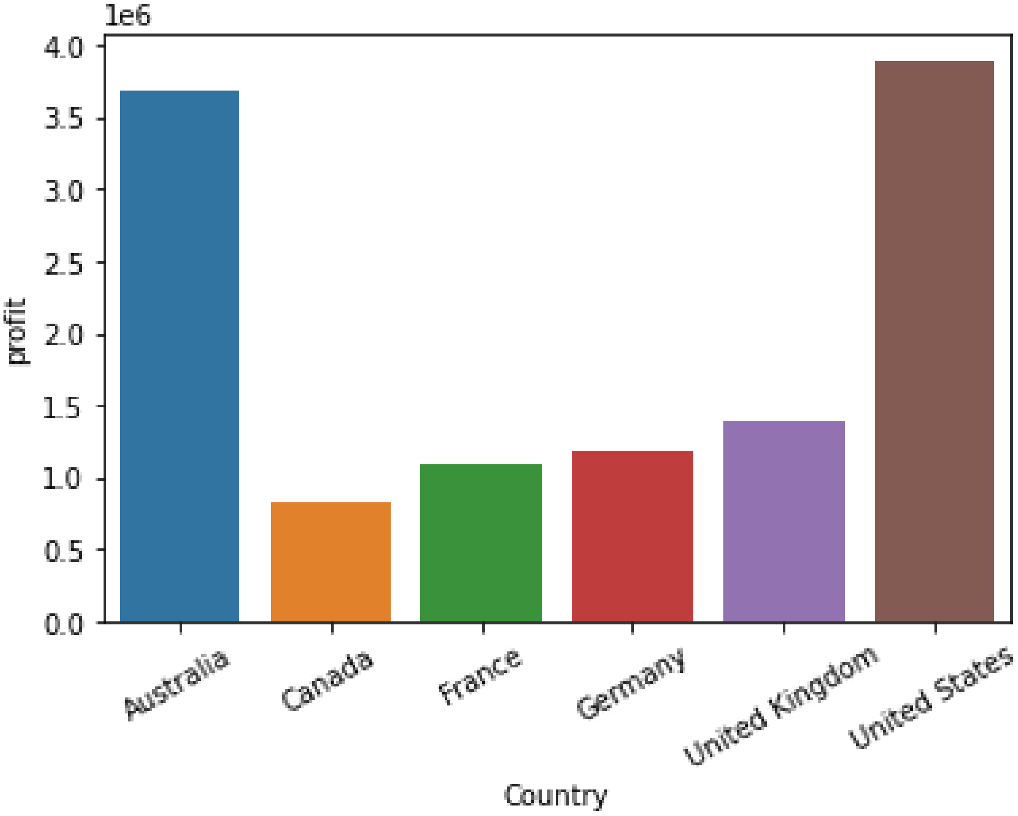
```
<pandas.io.formats.style.Styler at 0x202470c9a00>
```

Major Profit is contributed by the Bike Category

```
df.groupby(['Category', 'SubCategory','ProductName'])
['profit'].sum().nsmallest(10).to_frame()
```

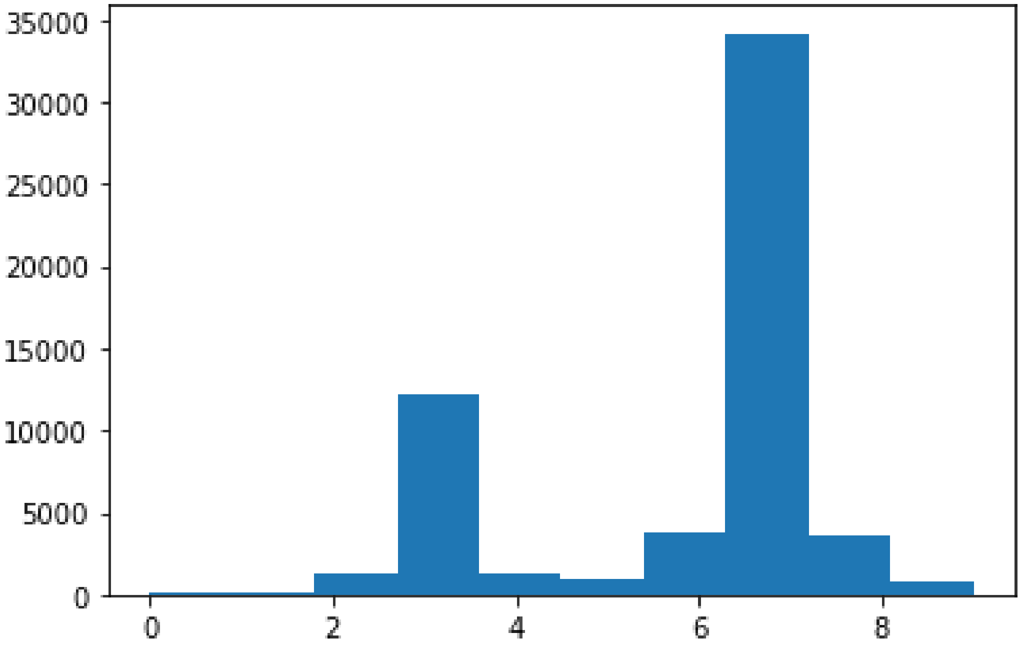
```
                                                                   profit
Category    SubCategory     ProductName
Clothing    Socks           Racing Socks- L                     1474.4574
                            Racing Socks- M                     1581.3837
Accessories Cleaners        Bike Wash - Dissolver               4299.8688
            Tires and Tubes Patch Kit/8 Patches                 4314.8350
Clothing    Caps            AWC Logo Cap                        4331.8315
Accessories Tires and Tubes Touring Tire Tube                   4363.8089
Clothing    Jerseys         Long-Sleeve Logo Jersey- XL         4495.6007
                            Short-Sleeve Classic Jersey- L      4544.8782
                            Long-Sleeve Logo Jersey- S          4610.5777
                            Short-Sleeve Classic Jersey- M      4793.2322
```

```
country_sales = pd.DataFrame(df.groupby('Country').sum()
[['SalesAmount', 'profit']])
country_sales.reset_index(inplace=True)

sns.barplot(data=country_sales, x='Country', y='profit')
plt.xticks(rotation=30)
plt.show()
```



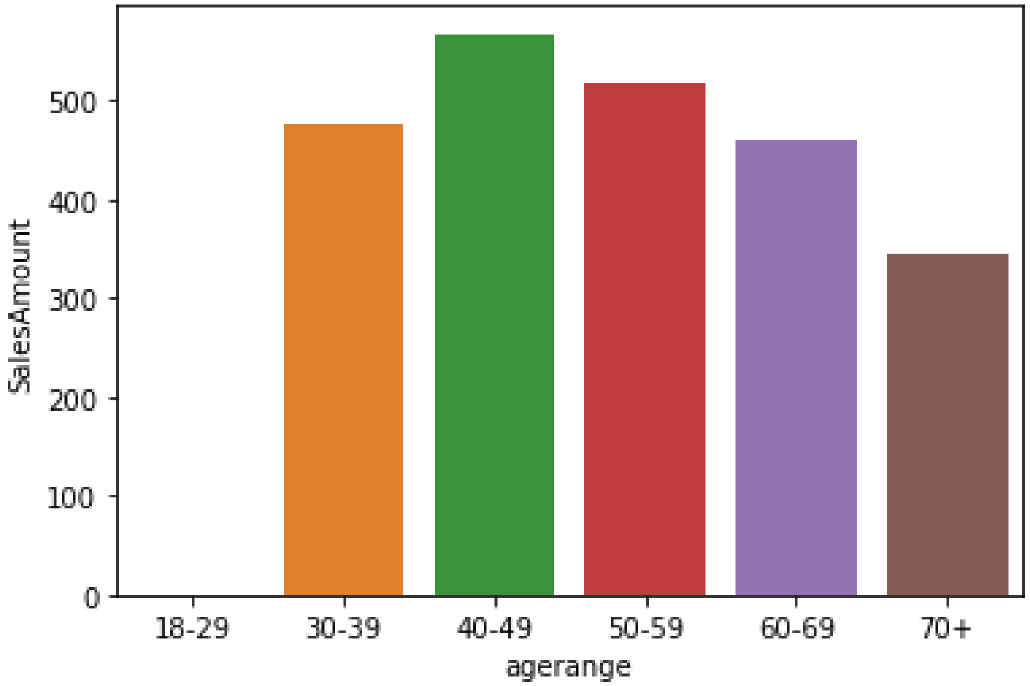
High volume of profit is earned from Australia and United States

```
df['OrderreadyDate'] = df['OrderDate'] +
pd.to_timedelta(df['DaysToManufacture'], unit='D')

# Check the delay between order shipment date and order ready to
supply
df['shipping_efficiency'] = (df['ShipDate'] -
df['OrderreadyDate']).dt.days

plt.hist(data=df, x="shipping_efficiency",)
plt.show()
```



The average order has a gap of 7 days between the day the order is ready for export from the factory and the date it was shipped. Management must work to reduce this gap toward 3 days.

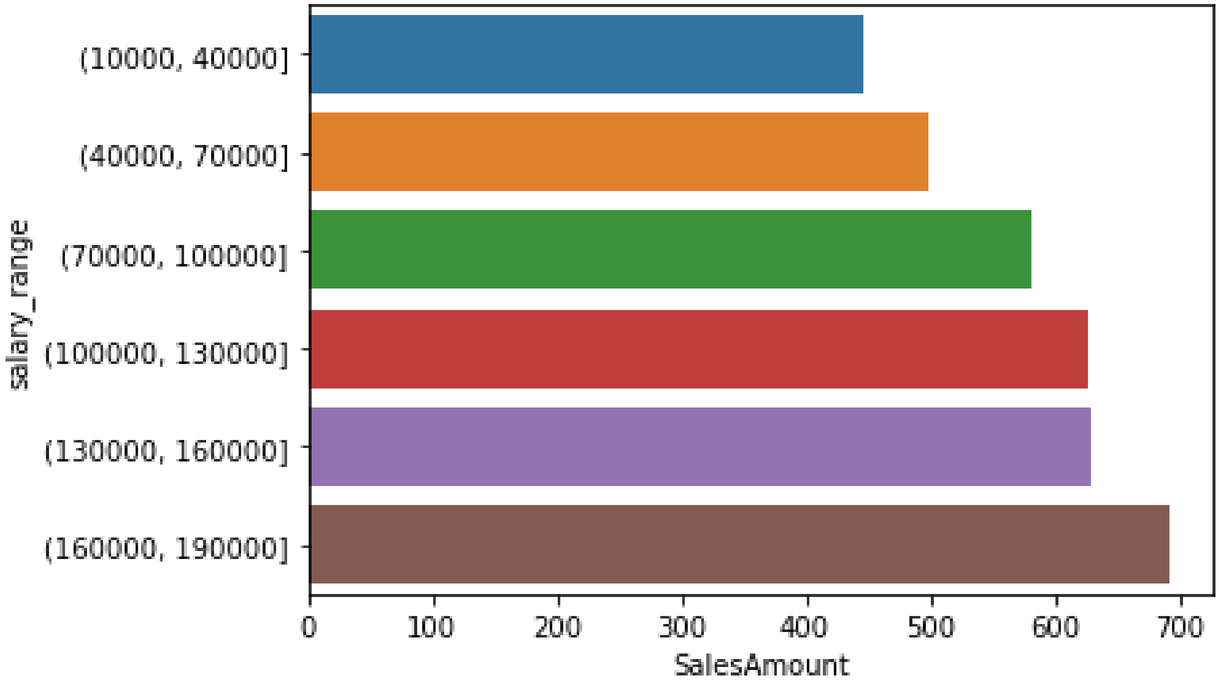
```
dj = df.groupby('agerange')['SalesAmount'].mean().to_frame().dropna()
dj.reset_index(inplace=True)
sns.barplot(data=dj, x='agerange', y='SalesAmount')
plt.show()
```



40-49 age group has produced most revenue.

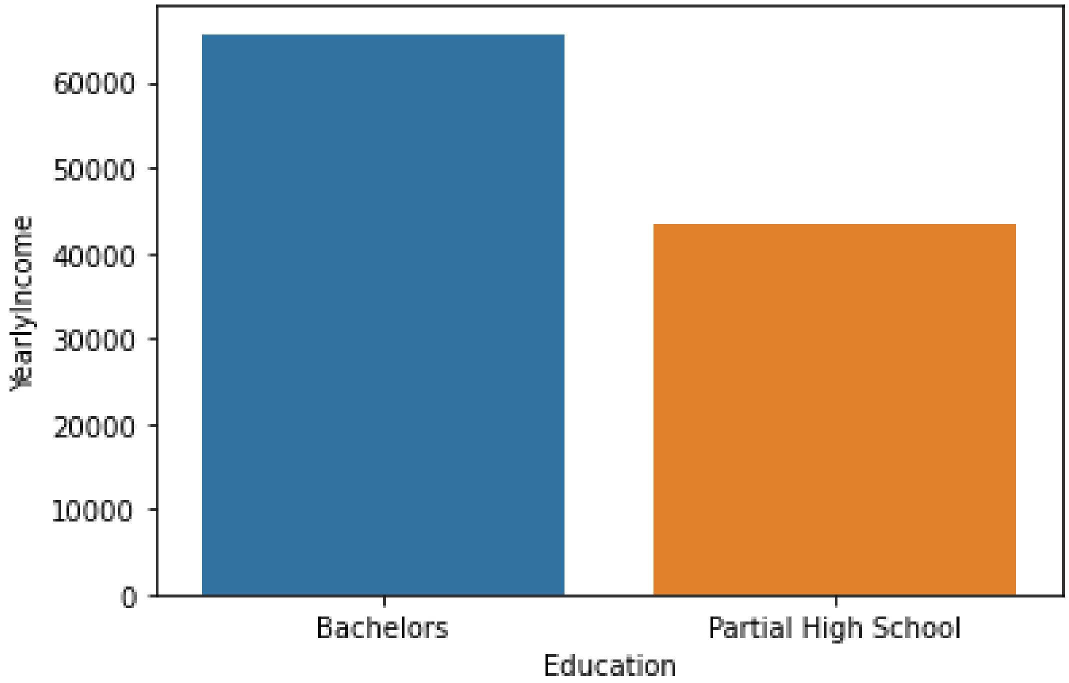
```python
def create_bins(lower_bound, width, quantity):
    """ create_bins returns an equal-width (distance) partitioning.
        It returns an ascending list of tuples, representing the
intervals.
        A tuple bins[i], i.e. (bins[i][0], bins[i][1])  with i > 0
        and i < quantity, satisfies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

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

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)
plt.show()
```

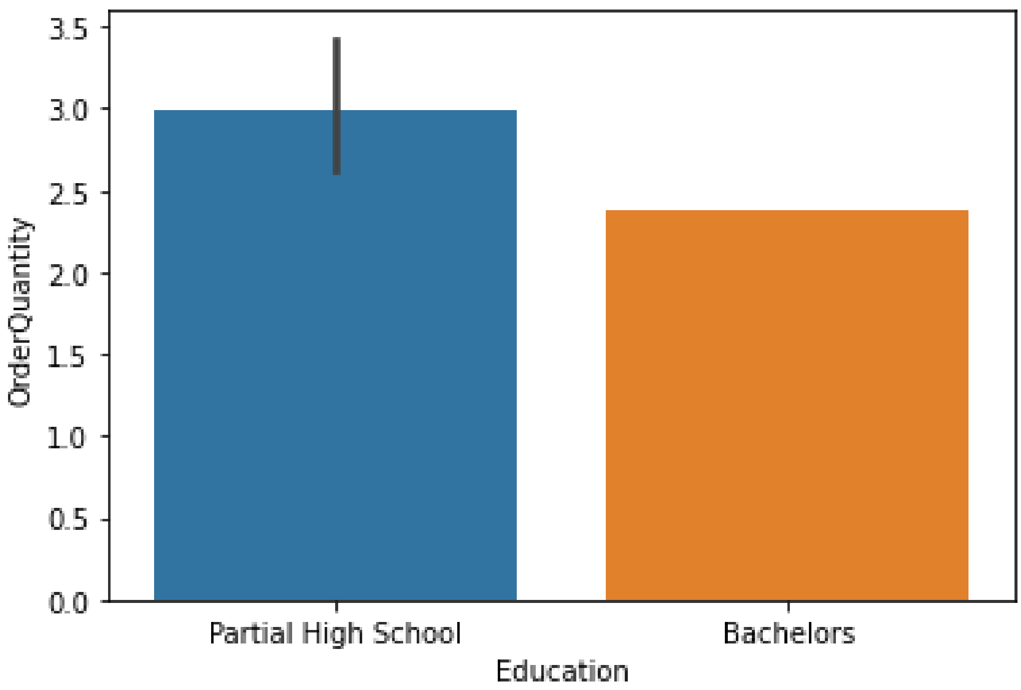


High salary range leads to increase in purchase

```
df_6 = df[(df['Education']=='Partial High School')|
(df['Education']=='Bachelors')].groupby('Education')
['YearlyIncome'].mean().to_frame()
df_6.reset_index(inplace=True)
sns.barplot(data=df_6, x='Education', y='YearlyIncome')
plt.show()
```



```
df_7 = df[(df['Education']=='Partial High School')|
(df['Education']=='Bachelors')]
df_7 = df_7.groupby(['Education','ProductName'])
['OrderQuantity'].mean().to_frame().sort_values('OrderQuantity',
ascending=False)[:10]
df_7.reset_index(inplace=True)
sns.barplot(data=df_7, x="Education",
             y="OrderQuantity")
plt.show()
```



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