RMOTR Data Science Curriculum

FreeCodeCamp - Pandas Real Life Example

cloned from rmotr-curriculum/ds-new-class-1-intro-to-data-science

Last updated: June 6th, 2020

Bike store sales

In this class we'll be analyzing sales made on bike stores.

Follow this data in a Google Spreadsheet

Hands on!

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

Loading our data:

```
!head data/sales_data.csv
```

OUTPUT

```
Date,Day,Month,Year,Customer_Age,Age_Group,Customer_Gender,Country,State,Product_Category
2013-11-26,26,November,2013,19,Youth (<25),M,Canada,British Columbia,Accessories,Bike Rac
2015-11-26,26,November,2015,19,Youth (<25),M,Canada,British Columbia,Accessories,Bike Rac
2014-03-23,23,March,2014,49,Adults (35-64),M,Australia,New South Wales,Accessories,Bike R
2016-03-23,23,March,2016,49,Adults (35-64),M,Australia,New South Wales,Accessories,Bike R
```

```
2014-02-22,22,February,2014,35,Adults (35-64),M,Australia,Victoria,Accessories,Bike Racks
```

```
sales = pd.read_csv(
    'data/sales_data.csv',
    parse_dates=['Date'])
```

The data at a glance:

```
sales.head()
```

OUTPUT

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_
0	2013-11-26	26	November	2013	19	Youth (<25)	M	Canada	British Columbia	Accessori
1	2015-11-26	26	November	2015	19	Youth (<25)	M	Canada	British Columbia	Accessori
2	2014-03-23	23	March	2014	49	Adults (35-64)	M	Australia	New South Wales	Accessori
3	2016-03-23	23	March	2016	49	Adults (35-64)	M	Australia	New South Wales	Accessori
4	2014-05-15	15	May	2014	47	Adults (35-64)	F	Australia	New South Wales	Accessori

```
sales.shape
```

OUTPUT

```
(113036, 18)
```

```
sales.info()
```

```
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   Date              113036 non-null  datetime64[ns]
 1   Day               113036 non-null  int64         
 2   Month             113036 non-null  object        
 3   Year              113036 non-null  int64         
 4   Customer_Age      113036 non-null  int64         
 5   Age_Group         113036 non-null  object        
 6   Customer_Gender   113036 non-null  object        
 7   Country           113036 non-null  object        
 8   State             113036 non-null  object        
 9   Product_Category  113036 non-null  object        
 10  Sub_Category      113036 non-null  object        
 11  Product           113036 non-null  object        
 12  Order_Quantity    113036 non-null  int64         
 13  Unit_Cost         113036 non-null  int64         
 14  Unit_Price        113036 non-null  int64         
 15  Profit            113036 non-null  int64         
 16  Cost              113036 non-null  int64         
 17  Revenue           113036 non-null  int64         
dtypes: datetime64[ns](1), int64(9), object(8)
memory usage: 15.5+ MB
```

```
sales.describe()
```

OUTPUT

	Day	Year	Customer_Age	Order_Quantity	Unit_Cost	Unit_Price
count	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000
mean	15.665753	2014.401739	35.919212	11.901660	267.296366	452.938427
std	8.781567	1.272510	11.021936	9.561857	549.835483	922.071219
min	1.000000	2011.000000	17.000000	1.000000	1.000000	2.000000
25%	8.000000	2013.000000	28.000000	2.000000	2.000000	5.000000
50%	16.000000	2014.000000	35.000000	10.000000	9.000000	24.000000
75%	23.000000	2016.000000	43.000000	20.000000	42.000000	70.000000
max	31.000000	2016.000000	87.000000	32.000000	2171.000000	3578.000000

Numerical analysis and visualization

We'll analyze the `Unit_Cost` column:

```
sales['Unit_Cost'].describe()
```

```
std          549.835483
min            1.000000
25%            2.000000
50%            9.000000
75%           42.000000
max         2171.000000
Name: Unit_Cost, dtype: float64
```

```
sales['Unit_Cost'].mean()
```

OUTPUT

```
267.296365759581
```

```
sales['Unit_Cost'].median()
```

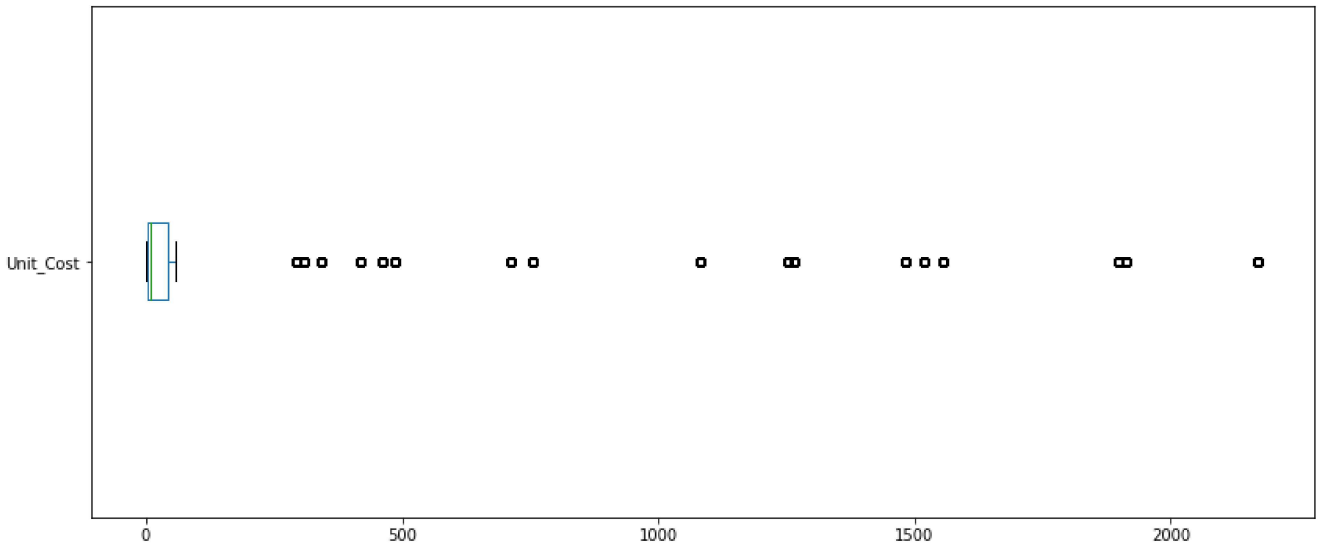
OUTPUT

```
9.0
```

```
sales['Unit_Cost'].plot(kind='box', vert=False, figsize=(14,6))
```

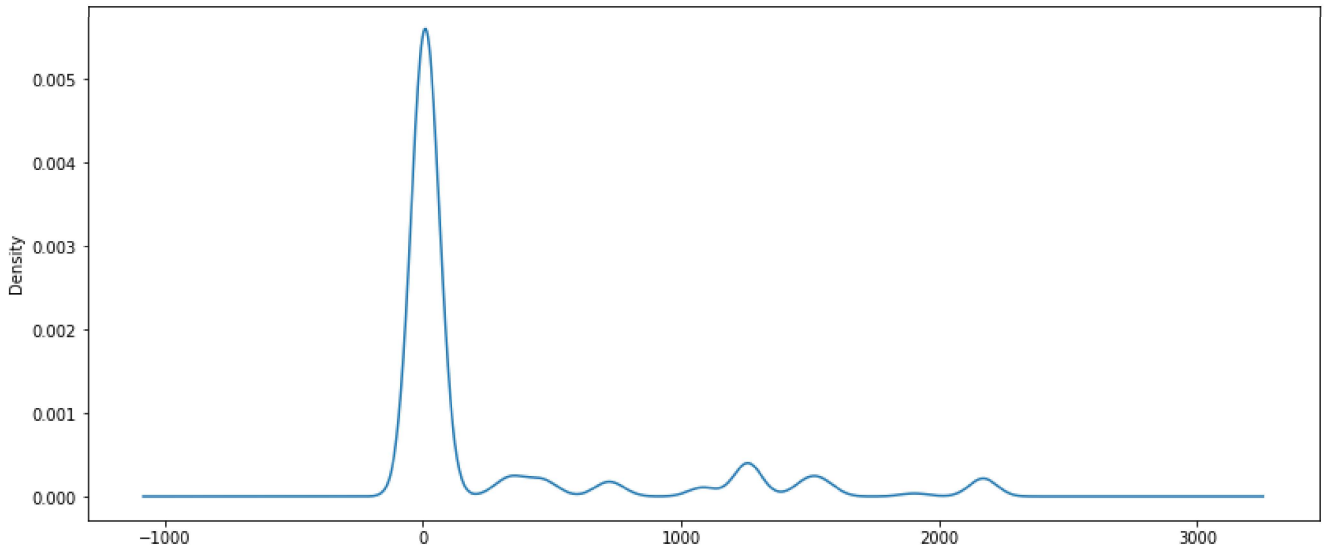
OUTPUT

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8af096b700>
```




```
sales['Unit_Cost'].plot(kind='density', figsize=(14,6)) # kde
```

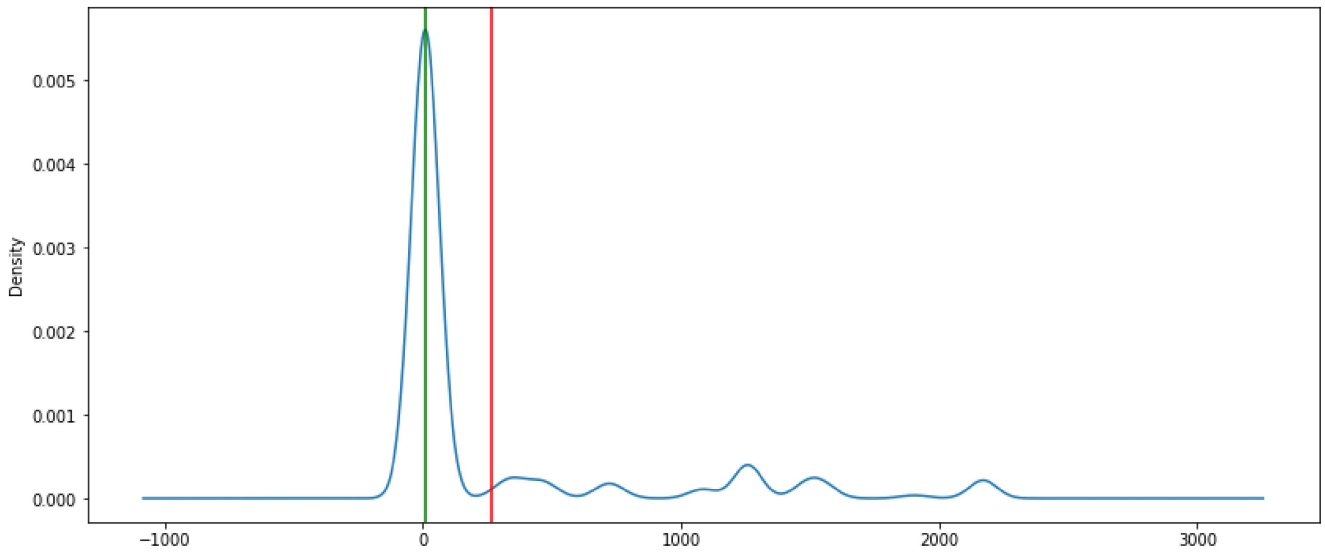
OUTPUT




```
ax = sales['Unit_Cost'].plot(kind='density', figsize=(14,6)) # kde
ax.axvline(sales['Unit_Cost'].mean(), color='red')
ax.axvline(sales['Unit_Cost'].median(), color='green')
```

OUTPUT

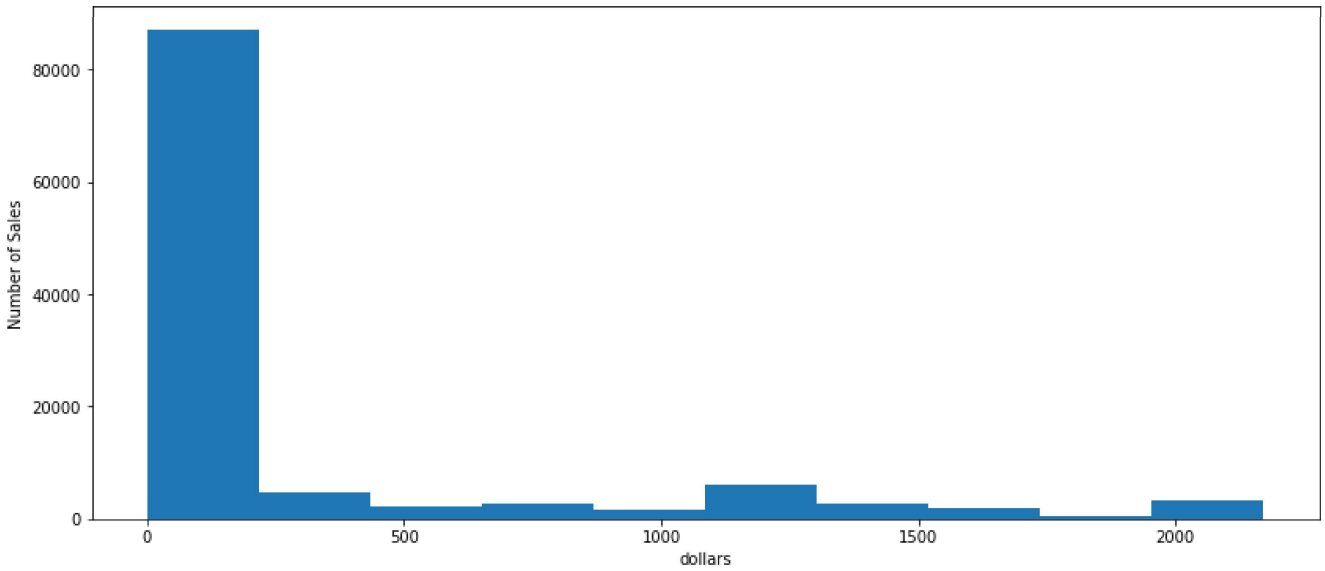
```
<matplotlib.lines.Line2D at 0x7f8b17499250>
```




```
ax = sales['Unit_Cost'].plot(kind='hist', figsize=(14,6))
ax.set_ylabel('Number of Sales')
ax.set_xlabel('dollars')
```

OUTPUT

```
Text(0.5, 0, 'dollars')
```



Categorical analysis and visualization

We'll analyze the `Age_Group` column:

```
sales.head()
```

OUTPUT

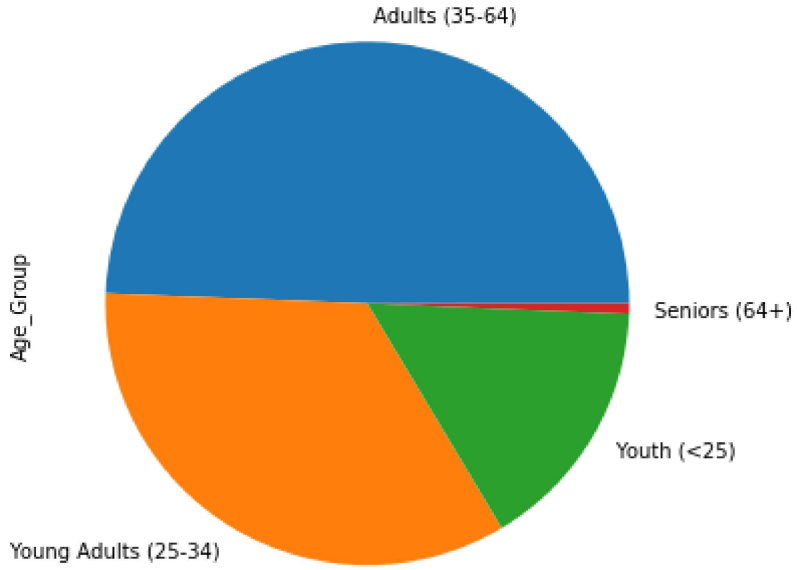
	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_
0	2013-11-26	26	November	2013	19	Youth (<25)	M	Canada	British Columbia	Accessori
1	2015-11-26	26	November	2015	19	Youth (<25)	M	Canada	British Columbia	Accessori
2	2014-03-23	23	March	2014	49	Adults (35-64)	M	Australia	New South Wales	Accessori
3	2016-03-23	23	March	2016	49	Adults (35-64)	M	Australia	New South Wales	Accessori
4	2014-05-15	15	May	2014	47	Adults (35-64)	F	Australia	New South Wales	Accessori

```
Adults (35-64)          55824
Young Adults (25-34)    38654
Youth (<25)             17828
Seniors (64+)             730
Name: Age_Group, dtype: int64
```

```
sales['Age_Group'].value_counts().plot(kind='pie', figsize=(6,6))
```

OUTPUT

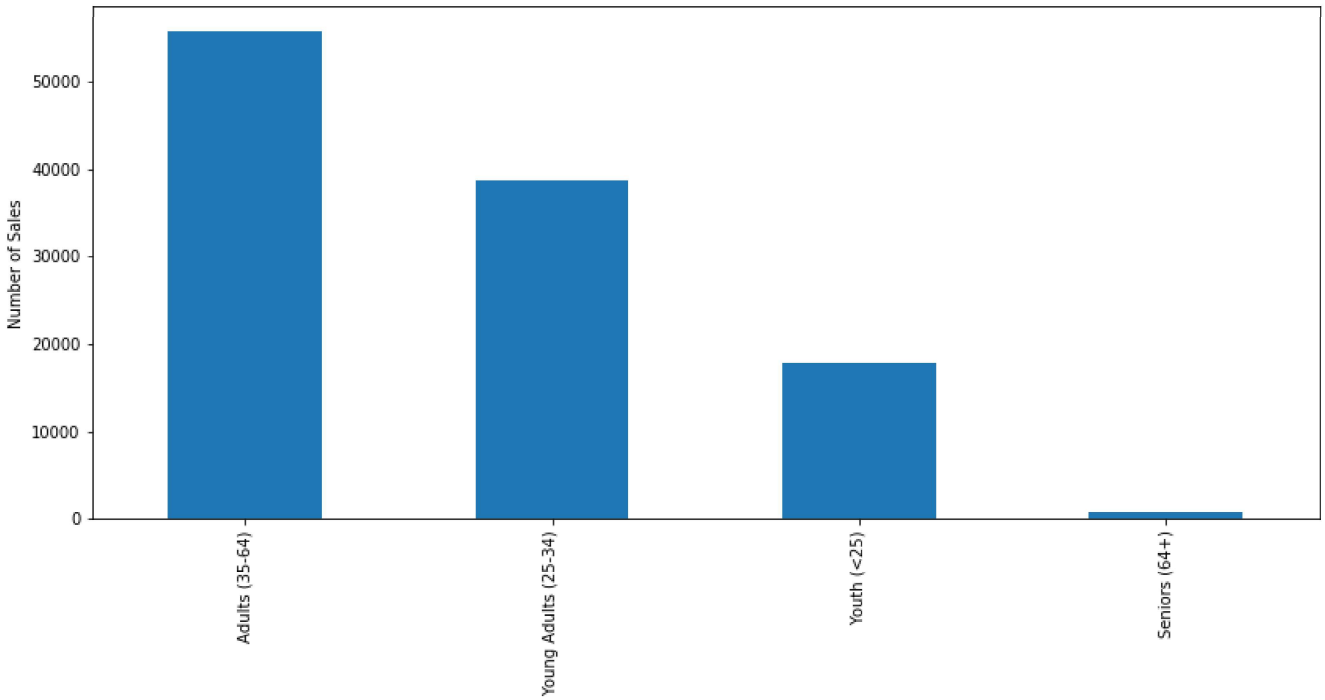
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade8bc2e0>
```




```
ax = sales['Age_Group'].value_counts().plot(kind='bar', figsize=(14,6))
ax.set_ylabel('Number of Sales')
```

OUTPUT

```
Text(0, 0.5, 'Number of Sales')
```



Relationship between the columns?

Can we find any significant relationship?

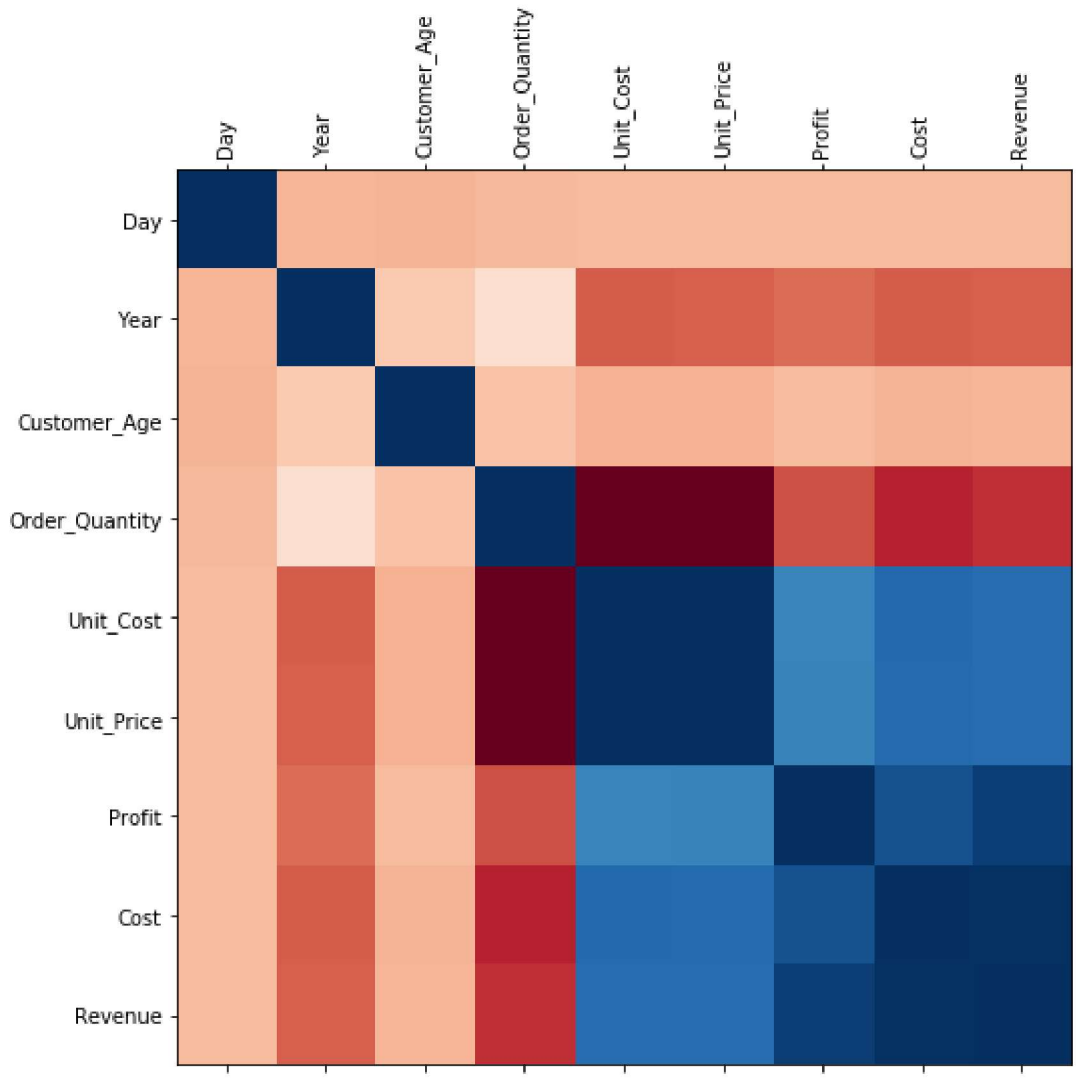
```
corr = sales.corr()

corr
```

OUTPUT

	Day	Year	Customer_Age	Order_Quantity	Unit_Cost	Unit_Price	Profit	
Day	1.000000	-0.007635	-0.014296	-0.002412	0.003133	0.003207	0.004623	0.
Year	-0.007635	1.000000	0.040994	0.123169	-0.217575	-0.213673	-0.181525	-0
Customer_Age	-0.014296	0.040994	1.000000	0.026887	-0.021374	-0.020262	0.004319	-0
Order_Quantity	-0.002412	0.123169	0.026887	1.000000	-0.515835	-0.515925	-0.238863	-0
Unit_Cost	0.003133	-0.217575	-0.021374	-0.515835	1.000000	0.997894	0.741020	0.
Unit_Price	0.003207	-0.213673	-0.020262	-0.515925	0.997894	1.000000	0.749870	0.
Profit	0.004623	-0.181525	0.004319	-0.238863	0.741020	0.749870	1.000000	0.
Cost	0.003329	-0.215604	-0.016013	-0.340382	0.829869	0.826301	0.902233	1.
Revenue	0.003853	-0.208673	-0.009326	-0.312895	0.817865	0.818522	0.956572	0.

```
plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical');
plt.yticks(range(len(corr.columns)), corr.columns);
```

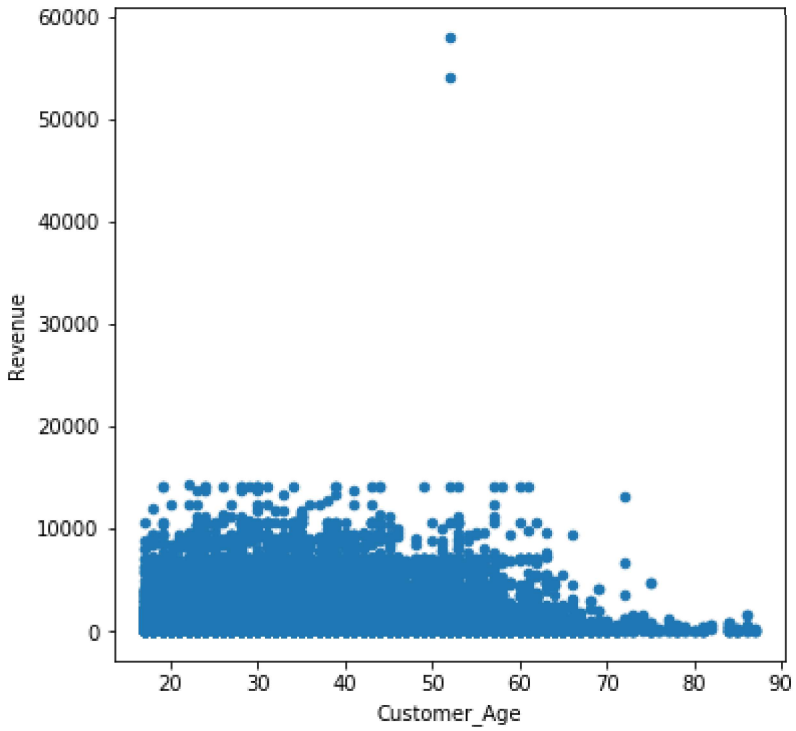
OUTPUT




```
sales.plot(kind='scatter', x='Customer_Age', y='Revenue', figsize=(6,6))
```

OUTPUT

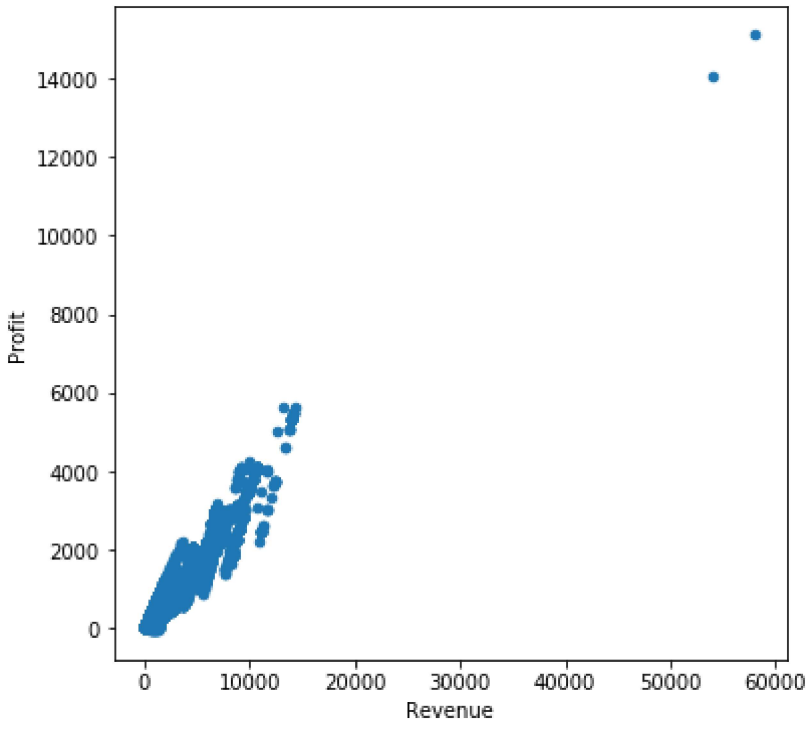
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade7c34f0>
```




```
sales.plot(kind='scatter', x='Revenue', y='Profit', figsize=(6,6))
```

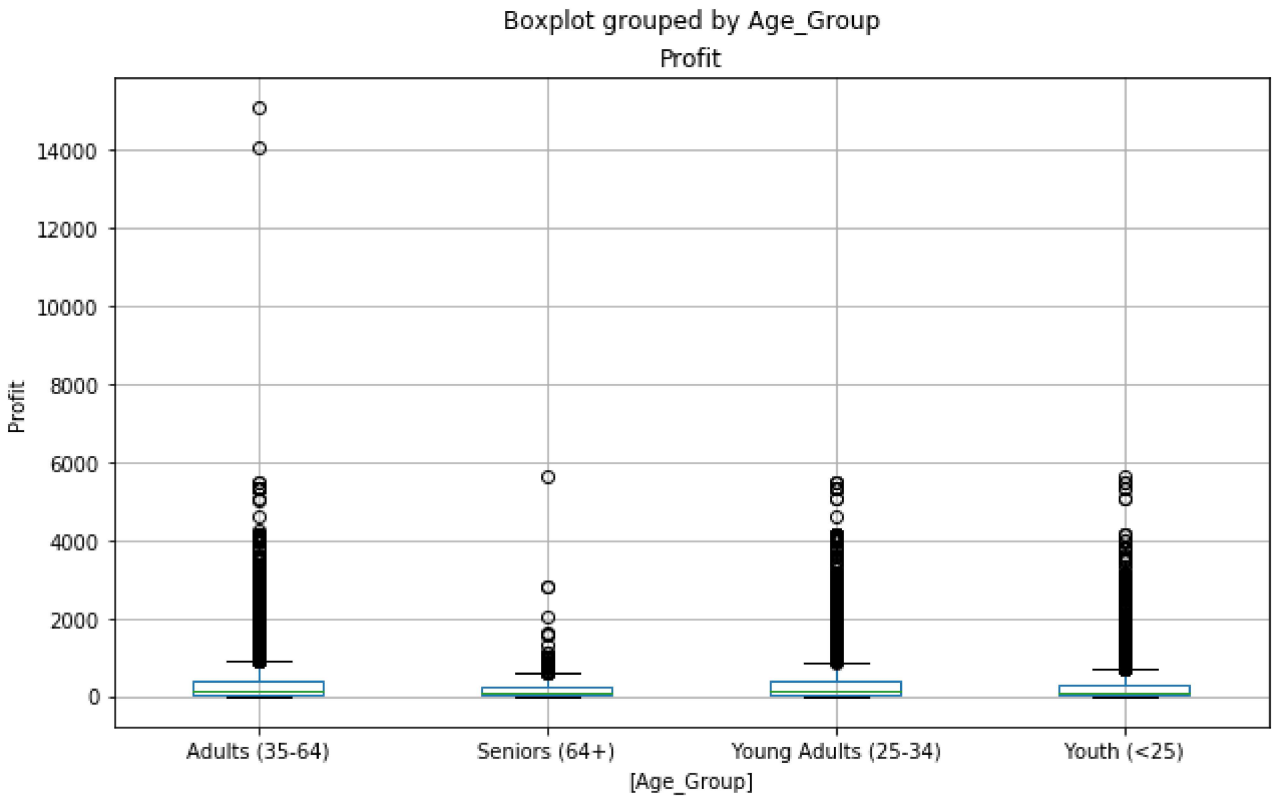
OUTPUT

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade7a70a0>
```



OUTPUT

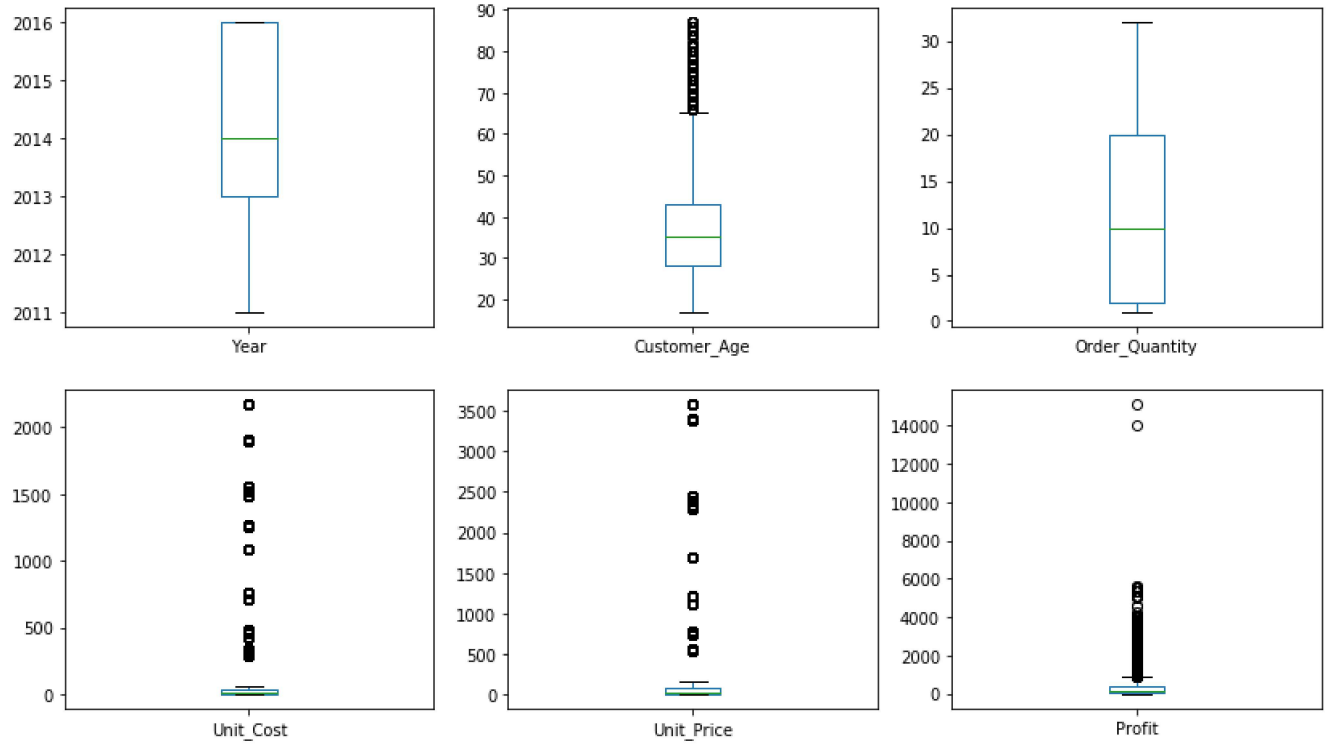
```
Text(0, 0.5, 'Profit')
```




```
boxplot_cols = ['Year', 'Customer_Age', 'Order_Quantity', 'Unit_Cost', 'Unit_Price', 'P

sales[boxplot_cols].plot(kind='box', subplots=True, layout=(2,3), figsize=(14,8))
```

OUTPUT

```
Year                AxesSubplot(0.125,0.536818;0.227941x0.343182)
Customer_Age      AxesSubplot(0.398529,0.536818;0.227941x0.343182)
Order_Quantity    AxesSubplot(0.672059,0.536818;0.227941x0.343182)
Unit_Cost               AxesSubplot(0.125,0.125;0.227941x0.343182)
Unit_Price           AxesSubplot(0.398529,0.125;0.227941x0.343182)
Profit               AxesSubplot(0.672059,0.125;0.227941x0.343182)
dtype: object
```



Column wrangling

We can also create new columns or modify existing ones.

Add and calculate a new `Revenue_per_Age` column

```
sales['Revenue_per_Age'] = sales['Revenue'] / sales['Customer_Age']
```

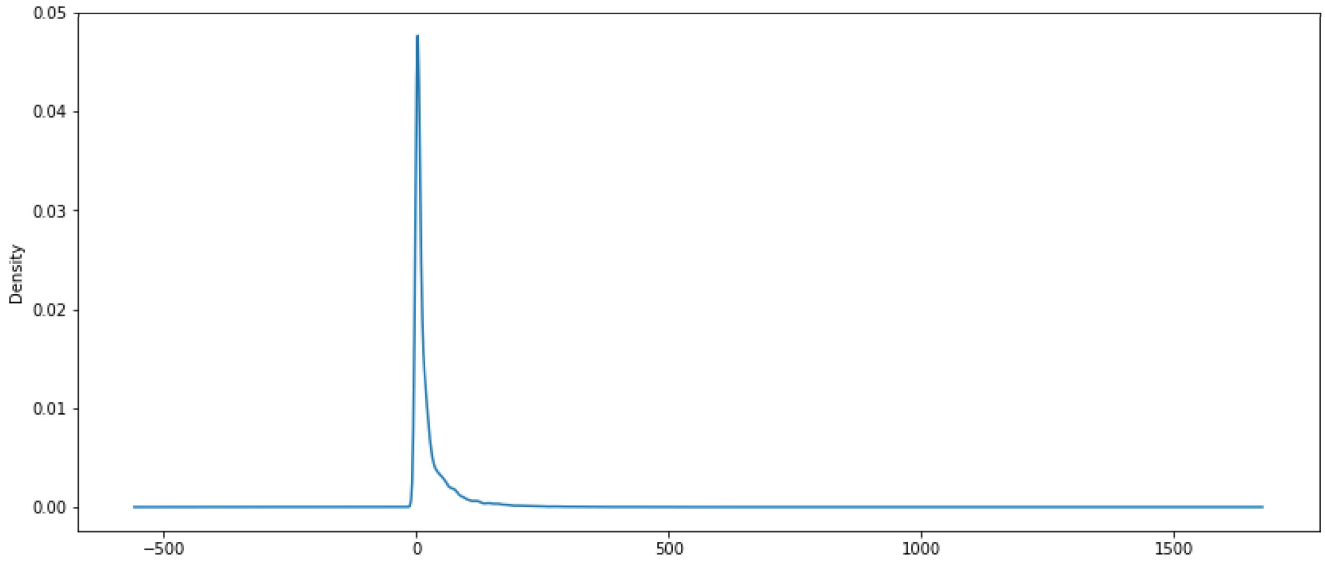
```
sales['Revenue_per_Age'].head()
```

OUTPUT

```
0    50.000000
1    50.000000
2    49.000000
3    42.612245
4     8.893617
Name: Revenue_per_Age, dtype: float64
```

```
sales['Revenue_per_Age'].plot(kind='density', figsize=(14,6))
```

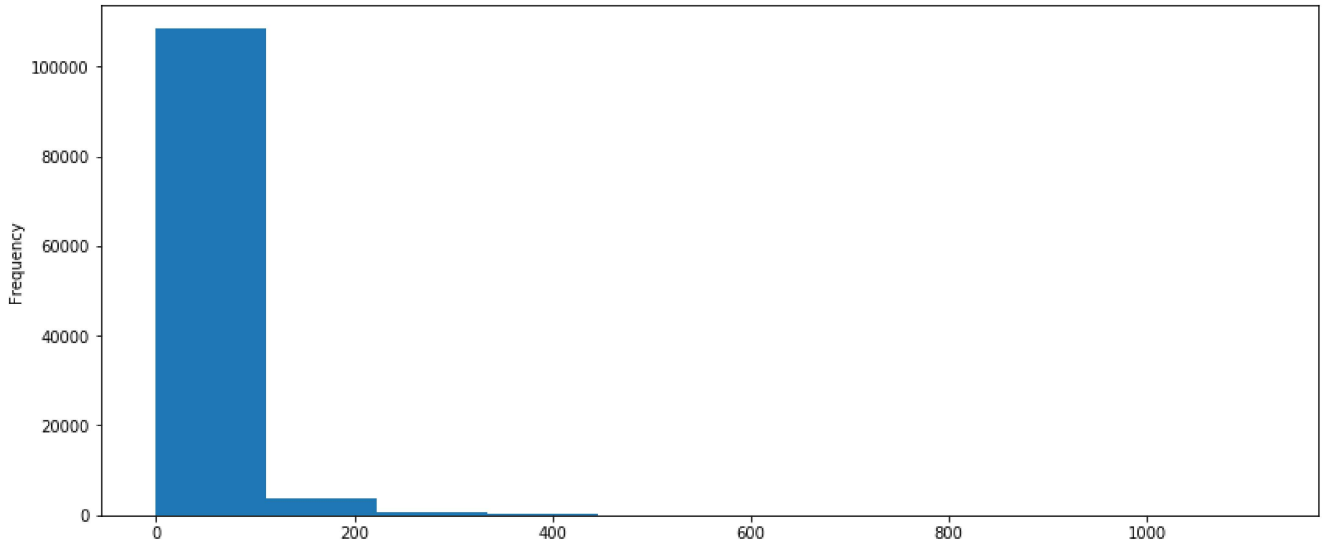
OUTPUT




```
sales['Revenue_per_Age'].plot(kind='hist', figsize=(14,6))
```

OUTPUT

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade3f9100>
```



Add and calculate a new `Calculated_Cost` column

Use this formula

$$Calculated\_Cost = Order\_Quantity * Unit\_Cost$$

```
sales['Calculated_Cost'] = sales['Order_Quantity'] * sales['Unit_Cost']
```

OUTPUT

```
0     360
1     360
2    1035
3     900
4     180
Name: Calculated_Cost, dtype: int64
```

```
(sales['Calculated_Cost'] != sales['Cost']).sum()
```

OUTPUT

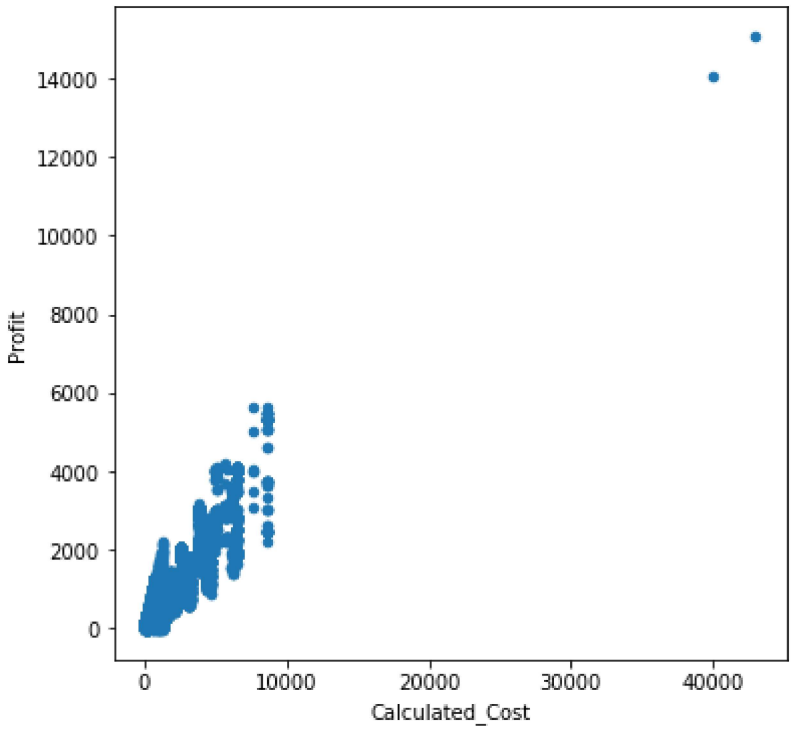
```
0
```

We can see the relationship between `Cost` and `Profit` using a scatter plot:

```
sales.plot(kind='scatter', x='Calculated_Cost', y='Profit', figsize=(6,6))
```

OUTPUT

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade3df790>
```



Add and calculate a new `Calculated_Revenue` column

```
sales['Calculated_Revenue'] = sales['Cost'] + sales['Profit']

sales['Calculated_Revenue'].head()
```

OUTPUT

```
0     950
1     950
2    2401
3    2088
4     418
Name: Calculated_Revenue, dtype: int64
```

```
(sales['Calculated_Revenue'] != sales['Revenue']).sum()
```

OUTPUT

```
0
```

```
sales.head()
```

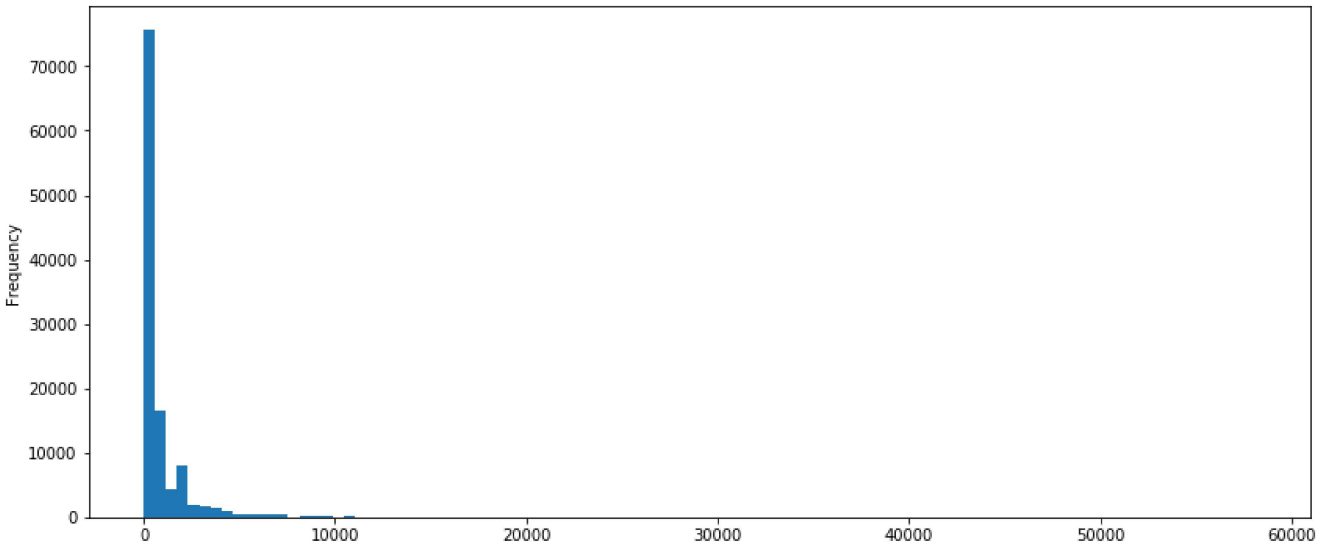
OUTPUT

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_
0	2013-11-26	26	November	2013	19	Youth (<25)	M	Canada	British Columbia	Accessori
1	2015-11-26	26	November	2015	19	Youth (<25)	M	Canada	British Columbia	Accessori
2	2014-03-23	23	March	2014	49	Adults (35-64)	M	Australia	New South Wales	Accessori
3	2016-03-23	23	March	2016	49	Adults (35-64)	M	Australia	New South Wales	Accessori
4	2014-05-15	15	May	2014	47	Adults (35-64)	F	Australia	New South Wales	Accessori

5 rows × 21 columns

OUTPUT

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ade39afa0>
```



Modify all `Unit_Price` values adding 3% tax to them

```
sales['Unit_Price'].head()
```

OUTPUT

```
0    120
1    120
2    120
3    120
4    120
Name: Unit_Price, dtype: int64
```

```
#sales['Unit_Price'] = sales['Unit_Price'] * 1.03

sales['Unit_Price'] *= 1.03


sales['Unit_Price'].head()
```

OUTPUT

```
0    123.6
1    123.6
2    123.6
3    123.6
4    123.6
Name: Unit_Price, dtype: float64
```

Selection & Indexing:

Get all the sales made in the state of `Kentucky`

```
sales.loc[sales['State'] == 'Kentucky']
```

OUTPUT

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Proc
156	2013-11-04	4	November	2013	40	Adults (35-64)	M	United States	Kentucky	Acce
157	2015-11-04	4	November	2015	40	Adults (35-64)	M	United States	Kentucky	Acce
23826	2014-04-16	16	April	2014	40	Adults (35-64)	M	United States	Kentucky	Acce
23827	2016-04-16	16	April	2016	40	Adults (35-64)	M	United States	Kentucky	Acce
31446	2014-04-16	16	April	2014	40	Adults (35-64)	M	United States	Kentucky	Acce
31447	2016-04-16	16	April	2016	40	Adults (35-64)	M	United States	Kentucky	Acce
79670	2014-04-16	16	April	2014	40	Adults (35-64)	M	United States	Kentucky	Acce
79671	2014-04-16	16	April	2014	40	Adults (35-64)	M	United States	Kentucky	Acce
79672	2016-04-16	16	April	2016	40	Adults (35-64)	M	United States	Kentucky	Acce
79673	2016-04-16	16	April	2016	40	Adults (35-64)	M	United States	Kentucky	Acce

10 rows × 21 columns

```
sales.loc[sales['Age_Group'] == 'Adults (35-64)', 'Revenue'].mean()
```

OUTPUT

```
762.8287654055604
```

How many records belong to Age Group `Youth (<25)` or `Adults (35-64)`?

```
sales.loc[(sales['Age_Group'] == 'Youth (<25)') | (sales['Age_Group'] == 'Adults (35-64
```

OUTPUT

```
73652
```

Get the mean revenue of the sales group `Adults (35-64)` in `United States`

```
sales.loc[(sales['Age_Group'] == 'Adults (35-64)') & (sales['Country'] == 'United State
```

OUTPUT

```
726.7260473588342
```

Increase the revenue by 10% to every sale made in France

```
sales.loc[sales['Country'] == 'France', 'Revenue'].head()
```

OUTPUT

```
50     787
51     787
52    2957
53    2851
60     626
Name: Revenue, dtype: int64
```

```
#sales.loc[sales['Country'] == 'France', 'Revenue'] = sales.loc[sales['Country'] == 'Fr
```

```
sales.loc[sales['Country'] == 'France', 'Revenue'] *= 1.1
```

OUTPUT

```
50     865.7
51     865.7
52    3252.7
53    3136.1
60     688.6
Name: Revenue, dtype: float64
```

We were unable to load Disqus. If you are a moderator please see our troubleshooting guide.

Make a fork of the project and work on it interactively on our free Jupyter online environment.

→ Start now