zvpoou9il

November 19, 2024

[ ]:

```python
# import required modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.datasets import load_diabetes
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import scipy.stats as stats
import statsmodels.api as sm
```

[ ]:

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:

```python
data = pd.read_csv("/content/retailsales.csv")
print(data)
#pd.read_csv() is a function from the Pandas library used to read a CSV (Comma␣
  ↪Separated Values) file
# and load its data into a DataFrame
```

```
      Invoice ID Branch       City Customer type  Gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           …      …          …             …      …
995  233-67-5758      C  Naypyitaw        Normal    Male
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female
```

```
               Product line  Unit price  Quantity   Tax 5%      Total  \
0         Health and beauty       74.69         7  26.1415   548.9715
1    Electronic accessories       15.28         5   3.8200    80.2200
2        Home and lifestyle       46.33         7  16.2155   340.5255
3         Health and beauty       58.22         8  23.2880   489.0480
4         Sports and travel       86.31         7  30.2085   634.3785
..                      …         …       …      …        …
995       Health and beauty       40.35         1   2.0175    42.3675
996      Home and lifestyle       97.38        10  48.6900  1022.4900
997      Food and beverages       31.84         1   1.5920    33.4320
998      Home and lifestyle       65.82         1   3.2910    69.1110
999     Fashion accessories       88.34         7  30.9190   649.2990

          Date   Time      Payment    cogs  gross margin percentage  \
0     1/5/2019  13:08      Ewallet  522.83                 4.761905
1     3/8/2019  10:29         Cash   76.40                 4.761905
2     3/3/2019  13:23  Credit card  324.31                 4.761905
3    1/27/2019  20:33      Ewallet  465.76                 4.761905
4     2/8/2019  10:37      Ewallet  604.17                 4.761905
..         …      …           …      …                      …
995  1/29/2019  13:46      Ewallet   40.35                 4.761905
996   3/2/2019  17:16      Ewallet  973.80                 4.761905
997   2/9/2019  13:22         Cash   31.84                 4.761905
998  2/22/2019  15:33         Cash   65.82                 4.761905
999  2/18/2019  13:28         Cash  618.38                 4.761905

     gross income  Rating
0         26.1415     9.1
1          3.8200     9.6
2         16.2155     7.4
3         23.2880     8.4
4         30.2085     5.3
..             …      …
995        2.0175     6.2
996       48.6900     4.4
997        1.5920     7.7
998        3.2910     4.1
999       30.9190     6.6

[1000 rows x 17 columns]
```

```
[ ]: data.info()
     #data.info() method is used with a Pandas DataFrame to get a concise summary of␣
      ↪the DataFrame.
     # It provides essential information about the DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
```

```
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Invoice ID               1000 non-null   object
 1   Branch                   1000 non-null   object
 2   City                     1000 non-null   object
 3   Customer type            1000 non-null   object
 4   Gender                   1000 non-null   object
 5   Product line             1000 non-null   object
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object
 11  Time                     1000 non-null   object
 12  Payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

[ ]:

```
data.describe()
#data.describe() method is used with a Pandas DataFrame to generate descriptive␣
 ↪statistics of numeric columns
```

[ ]:

```
        Unit price     Quantity       Tax 5%        Total        cogs  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.00000
mean     55.672130     5.510000    15.379369   322.966749   307.58738
std      26.494628     2.923431    11.708825   245.885335   234.17651
min      10.080000     1.000000     0.508500    10.678500    10.17000
25%      32.875000     3.000000     5.924875   124.422375   118.49750
50%      55.230000     5.000000    12.088000   253.848000   241.76000
75%      77.935000     8.000000    22.445250   471.350250   448.90500
max      99.960000    10.000000    49.650000  1042.650000   993.00000

       gross margin percentage  gross income      Rating
count             1.000000e+03   1000.000000  1000.00000
mean              4.761905e+00     15.379369     6.97270
std               6.131498e-14     11.708825     1.71858
min               4.761905e+00      0.508500     4.00000
25%               4.761905e+00      5.924875     5.50000
50%               4.761905e+00     12.088000     7.00000
75%               4.761905e+00     22.445250     8.50000
max               4.761905e+00     49.650000    10.00000
```

```
[ ]: head= data.head()
     head
     #data.head() method is used with a Pandas DataFrame to display the first few␣
      ↪rows of the dataset.
```

```
[ ]:     Invoice ID Branch       City Customer type  Gender  \
     0  750-67-8428      A     Yangon        Member  Female   
     1  226-31-3081      C  Naypyitaw        Normal  Female   
     2  631-41-3108      A     Yangon        Normal    Male   
     3  123-19-1176      A     Yangon        Member    Male   
     4  373-73-7910      A     Yangon        Normal    Male   

                   Product line  Unit price  Quantity   Tax 5%     Total       Date  \
     0       Health and beauty       74.69         7  26.1415  548.9715   1/5/2019   
     1  Electronic accessories       15.28         5   3.8200   80.2200   3/8/2019   
     2      Home and lifestyle       46.33         7  16.2155  340.5255   3/3/2019   
     3       Health and beauty       58.22         8  23.2880  489.0480  1/27/2019   
     4       Sports and travel       86.31         7  30.2085  634.3785   2/8/2019   

         Time      Payment    cogs  gross margin percentage  gross income  Rating
     0  13:08      Ewallet  522.83                 4.761905       26.1415     9.1
     1  10:29         Cash   76.40                 4.761905        3.8200     9.6
     2  13:23  Credit card  324.31                 4.761905       16.2155     7.4
     3  20:33      Ewallet  465.76                 4.761905       23.2880     8.4
     4  10:37      Ewallet  604.17                 4.761905       30.2085     5.3
```

```
[ ]: data.tail()
     #data.tail() method is used with a Pandas DataFrame to display the last few␣
      ↪rows of the dataset.
```

```
[ ]:       Invoice ID Branch       City Customer type  Gender           Product line  \
     995  233-67-5758      C  Naypyitaw        Normal    Male      Health and beauty   
     996  303-96-2227      B   Mandalay        Normal  Female     Home and lifestyle   
     997  727-02-1313      A     Yangon        Member    Male     Food and beverages   
     998  347-56-2442      A     Yangon        Normal    Male     Home and lifestyle   
     999  849-09-3807      A     Yangon        Member  Female    Fashion accessories   

          Unit price  Quantity   Tax 5%      Total       Date   Time  Payment  \
     995       40.35         1   2.0175    42.3675  1/29/2019  13:46  Ewallet   
     996       97.38        10  48.6900  1022.4900   3/2/2019  17:16  Ewallet   
     997       31.84         1   1.5920    33.4320   2/9/2019  13:22     Cash   
     998       65.82         1   3.2910    69.1110  2/22/2019  15:33     Cash   
     999       88.34         7  30.9190   649.2990  2/18/2019  13:28     Cash   

            cogs  gross margin percentage  gross income  Rating
     995   40.35                 4.761905        2.0175     6.2
     996  973.80                 4.761905       48.6900     4.4
```

```
997   31.84                 4.761905        1.5920     7.7
998   65.82                 4.761905        3.2910     4.1
999  618.38                 4.761905       30.9190     6.6
```

```
[ ]: data.shape
     #data.shape attribute is used with a Pandas DataFrame (or Series) to get the␣
      ↪dimensions of the dataset.
```

```
[ ]: (1000, 17)
```

```
[ ]: data.size
     #data.size attribute is used with a Pandas DataFrame (or Series) to return the␣
      ↪total number of elements in the dataset.
```

```
[ ]: 17000
```

```
[ ]: data.duplicated()
     #data.duplicated() method is used with a Pandas DataFrame to identify duplicate␣
      ↪rows in the dataset
```

```
[ ]: 0      False
     1      False
     2      False
     3      False
     4      False
            …
     995    False
     996    False
     997    False
     998    False
     999    False
     Length: 1000, dtype: bool
```

```
[ ]: data.drop_duplicates()
     #drop_duplicates() method in Python's Pandas library is used to remove␣
      ↪duplicate rows from a DataFrame.
```

```
[ ]:       Invoice ID Branch       City Customer type  Gender  \
     0    750-67-8428      A     Yangon        Member  Female
     1    226-31-3081      C  Naypyitaw        Normal  Female
     2    631-41-3108      A     Yangon        Normal    Male
     3    123-19-1176      A     Yangon        Member    Male
     4    373-73-7910      A     Yangon        Normal    Male
     ..           …    …        …           …      …
     995  233-67-5758      C  Naypyitaw        Normal    Male
     996  303-96-2227      B   Mandalay        Normal  Female
     997  727-02-1313      A     Yangon        Member    Male
```

```
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

               Product line  Unit price  Quantity   Tax 5%      Total  \
0         Health and beauty       74.69         7  26.1415   548.9715
1    Electronic accessories       15.28         5   3.8200    80.2200
2        Home and lifestyle       46.33         7  16.2155   340.5255
3         Health and beauty       58.22         8  23.2880   489.0480
4         Sports and travel       86.31         7  30.2085   634.3785
..                      …         …       …      …        …
995       Health and beauty       40.35         1   2.0175    42.3675
996      Home and lifestyle       97.38        10  48.6900  1022.4900
997      Food and beverages       31.84         1   1.5920    33.4320
998      Home and lifestyle       65.82         1   3.2910    69.1110
999     Fashion accessories       88.34         7  30.9190   649.2990

          Date   Time      Payment    cogs  gross margin percentage  \
0     1/5/2019  13:08      Ewallet  522.83                 4.761905
1     3/8/2019  10:29         Cash   76.40                 4.761905
2     3/3/2019  13:23  Credit card  324.31                 4.761905
3    1/27/2019  20:33      Ewallet  465.76                 4.761905
4     2/8/2019  10:37      Ewallet  604.17                 4.761905
..         …    …          …     …                      …
995  1/29/2019  13:46      Ewallet   40.35                 4.761905
996   3/2/2019  17:16      Ewallet  973.80                 4.761905
997   2/9/2019  13:22         Cash   31.84                 4.761905
998  2/22/2019  15:33         Cash   65.82                 4.761905
999  2/18/2019  13:28         Cash  618.38                 4.761905

     gross income  Rating
0         26.1415     9.1
1          3.8200     9.6
2         16.2155     7.4
3         23.2880     8.4
4         30.2085     5.3
..            …     …
995        2.0175     6.2
996       48.6900     4.4
997        1.5920     7.7
998        3.2910     4.1
999       30.9190     6.6

[1000 rows x 17 columns]
```

[ ]:

```
#isnull() method in the Pandas library is used to detect missing or null values in a DataFrame or Series.
a = data.isnull()
```

```
a
```

[ ]:

```
     Invoice ID  Branch   City  Customer type  Gender  Product line  \
0         False   False  False          False   False         False
1         False   False  False          False   False         False
2         False   False  False          False   False         False
3         False   False  False          False   False         False
4         False   False  False          False   False         False
..          …     …    …            …     …           …
995       False   False  False          False   False         False
996       False   False  False          False   False         False
997       False   False  False          False   False         False
998       False   False  False          False   False         False
999       False   False  False          False   False         False

     Unit price  Quantity  Tax 5%  Total   Date   Time  Payment   cogs  \
0         False     False   False  False  False  False    False  False
1         False     False   False  False  False  False    False  False
2         False     False   False  False  False  False    False  False
3         False     False   False  False  False  False    False  False
4         False     False   False  False  False  False    False  False
..          …       …     …    …    …    …      …    …
995       False     False   False  False  False  False    False  False
996       False     False   False  False  False  False    False  False
997       False     False   False  False  False  False    False  False
998       False     False   False  False  False  False    False  False
999       False     False   False  False  False  False    False  False

     gross margin percentage  gross income  Rating
0                      False         False   False
1                      False         False   False
2                      False         False   False
3                      False         False   False
4                      False         False   False
..                       …           …     …
995                    False         False   False
996                    False         False   False
997                    False         False   False
998                    False         False   False
999                    False         False   False

[1000 rows x 17 columns]
```

[ ]:

```
#Replace Null Values with Mean (for categorical columns)
df = data['Rating']
df.fillna(df.mean(), inplace=True)
df
```

```
[ ]: 0      9.1
     1      9.6
     2      7.4
     3      8.4
     4      5.3
           …
     995    6.2
     996    4.4
     997    7.7
     998    4.1
     999    6.6
     Name: Rating, Length: 1000, dtype: float64
```

[ ]:

```
#Replace Null Values with Median (for categorical columns)
df = data['Rating']
df.fillna(df.median(), inplace=True)
df
```

```
[ ]: 0      9.1
     1      9.6
     2      7.4
     3      8.4
     4      5.3
           …
     995    6.2
     996    4.4
     997    7.7
     998    4.1
     999    6.6
     Name: Rating, Length: 1000, dtype: float64
```

[ ]:

```
#Replace Null Values with Mode (for categorical columns)
for column in data.select_dtypes(include=['object']).columns:
    mode = data['Payment'].mode()[0]
    data['Payment'].fillna(mode, inplace=True)
```

[ ]:

```
# Additional descriptive statistics: Mean, Median, Mode
print("\nAdditional Descriptive Statistics:")
for col in data.select_dtypes(include=['float64', 'int64']).columns:
    print(f"{col}:")
    print(f"  Mean: {data[col].mean()}")
    print(f"  Median: {data[col].median()}")
    print(f"  Mode: {data[col].mode()[0]}")
```

```

Additional Descriptive Statistics:
Unit price:
  Mean: 55.67213
  Median: 55.230000000000004
  Mode: 83.77
Quantity:
  Mean: 5.51
  Median: 5.0
  Mode: 10
Tax 5%:
  Mean: 15.379368999999999
  Median: 12.088000000000001
  Mode: 4.154
Total:
  Mean: 322.966749
  Median: 253.848
  Mode: 87.234
cogs:
  Mean: 307.58738
  Median: 241.76
  Mode: 83.08
gross margin percentage:
  Mean: 4.761904762
  Median: 4.761904762
  Mode: 4.761904762
gross income:
  Mean: 15.379368999999999
  Median: 12.088000000000001
  Mode: 4.154
Rating:
  Mean: 6.9727
  Median: 7.0
  Mode: 6.0
```

[ ]:

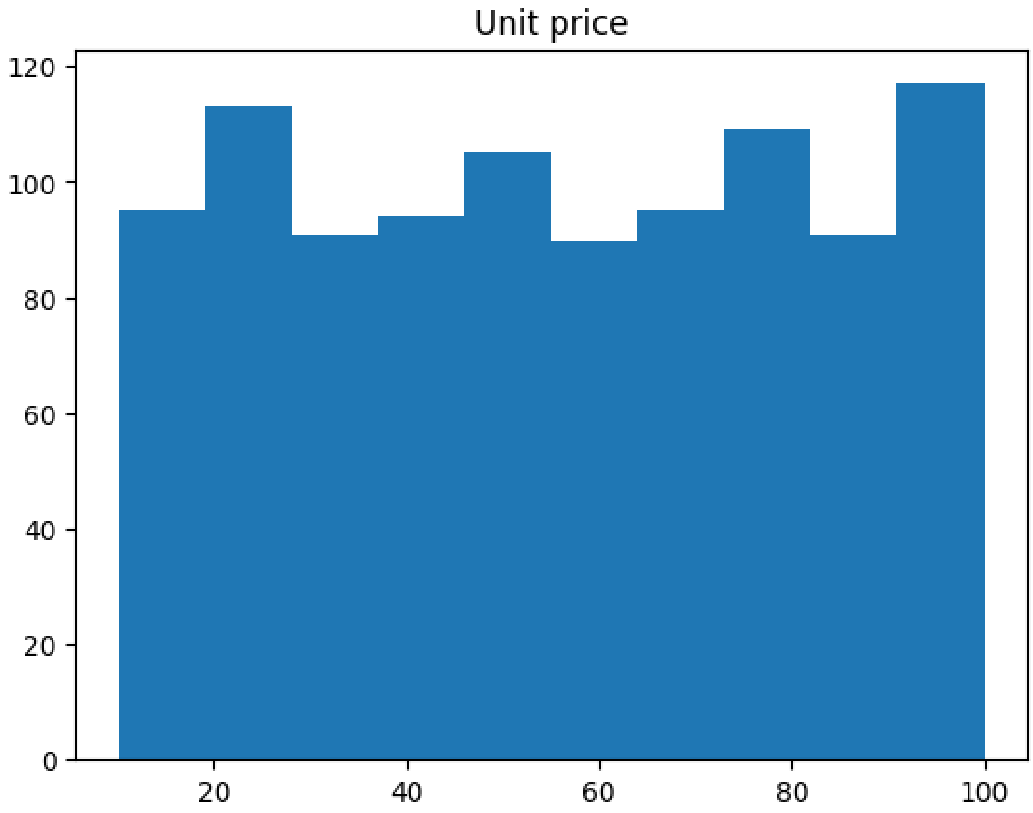
```python
#Outliers are values in a dataset that are significantly different from the majority of the data.
#They can be much larger or smaller than most other values, and they can arise due to variability in the data or errors during data collection.
outliners =[]
def detect_outliners(data):
  threshold = 3
  mean = np.mean(data)
  std = np.std(data)

  for i in data:
    z_score = (i-mean)/std
    if np.abs(z_score) > threshold:
      outliners.append(i)
  return outliners
```

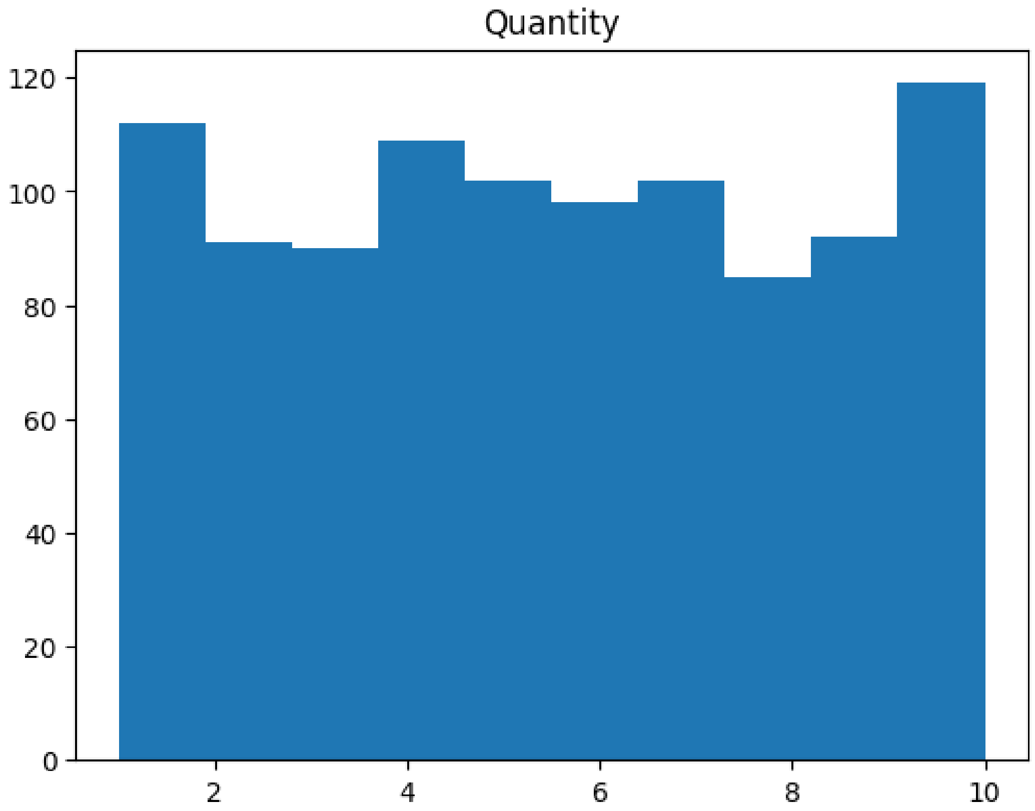
```
outliners
```

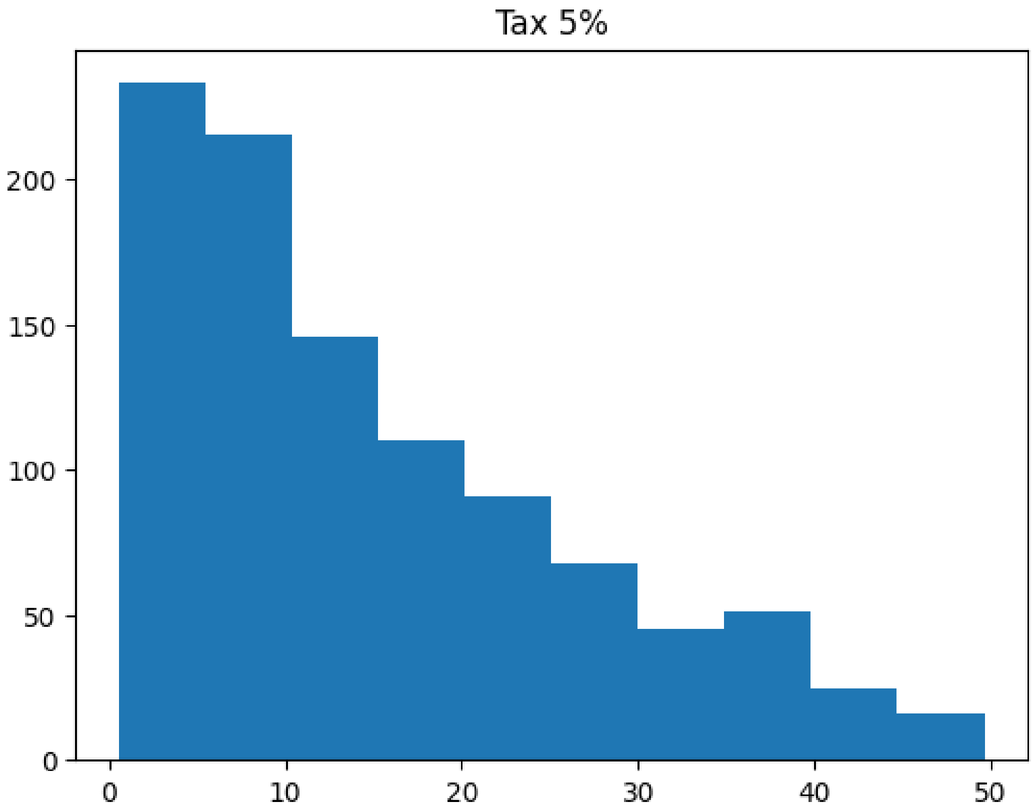
[ ]:
```
[]
```

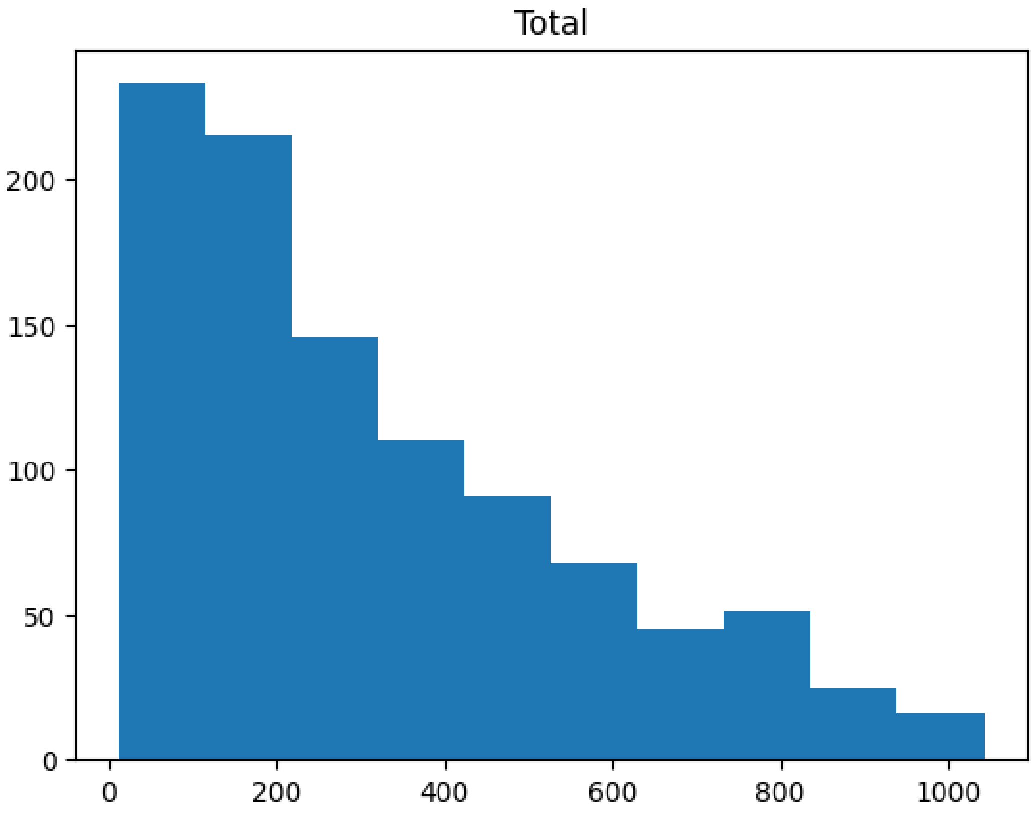
[ ]:
```
#Replace Outliers: Replace the values that fall outside the IQR bounds with the␣
 ↪Median, Mode, or Mean based on your choice.
def replace_outliers_with_median(err_arr):
    a = err_arr
    med = np.median(a)
    outlierConstant = 1.5
    upper_quartile = np.percentile(a, 80)
    lower_quartile = np.percentile(a, 20)
    IQR = (upper_quartile - lower_quartile) * outlierConstant
    quartileSet = (lower_quartile - IQR, upper_quartile + IQR)
    # Find the outliers with 80% interval and replace them with median value
    output = np.where((a >= quartileSet[0]) & (a <= quartileSet[1]), a, med)

    return output
```

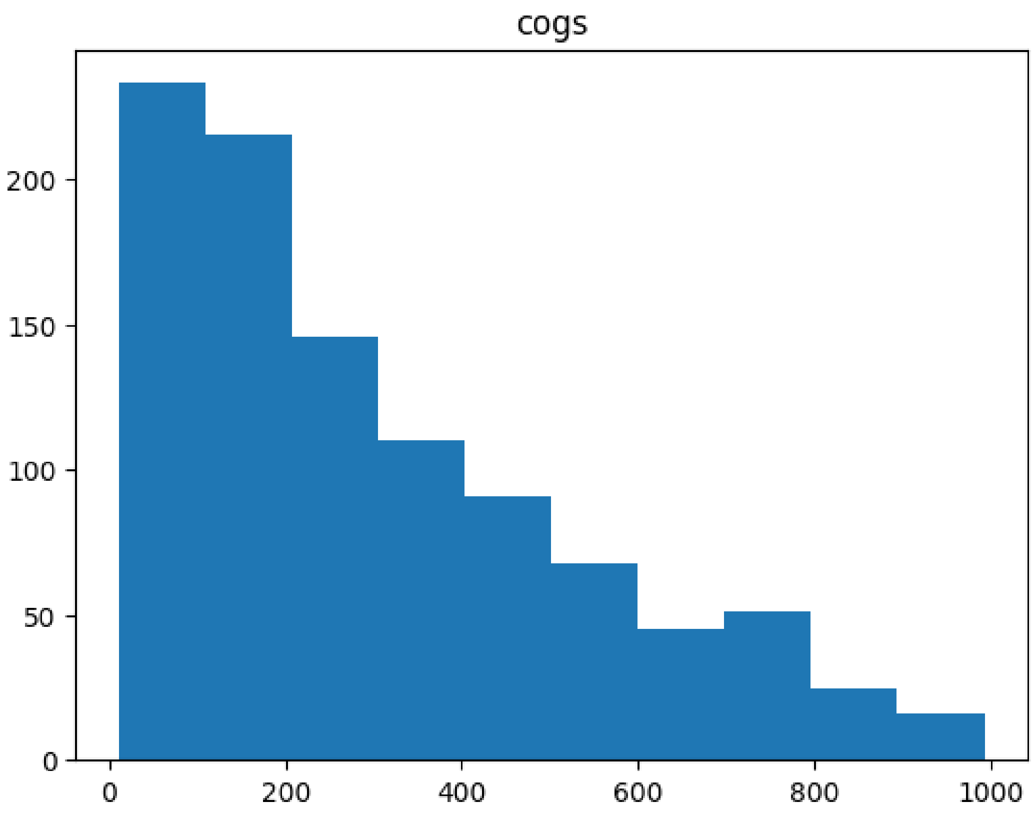
[ ]:
```
# Histogram  is a graphical representation of the distribution of numerical␣
 ↪data.
# It divides the data into bins or intervals and counts the frequency of data␣
 ↪points that fall into each bin.
# It is a univariate analysis plot.
for column in data.select_dtypes(include=[np.number]).columns:
    plt.hist(data[column])
    plt.title(column)
    plt.show()
```

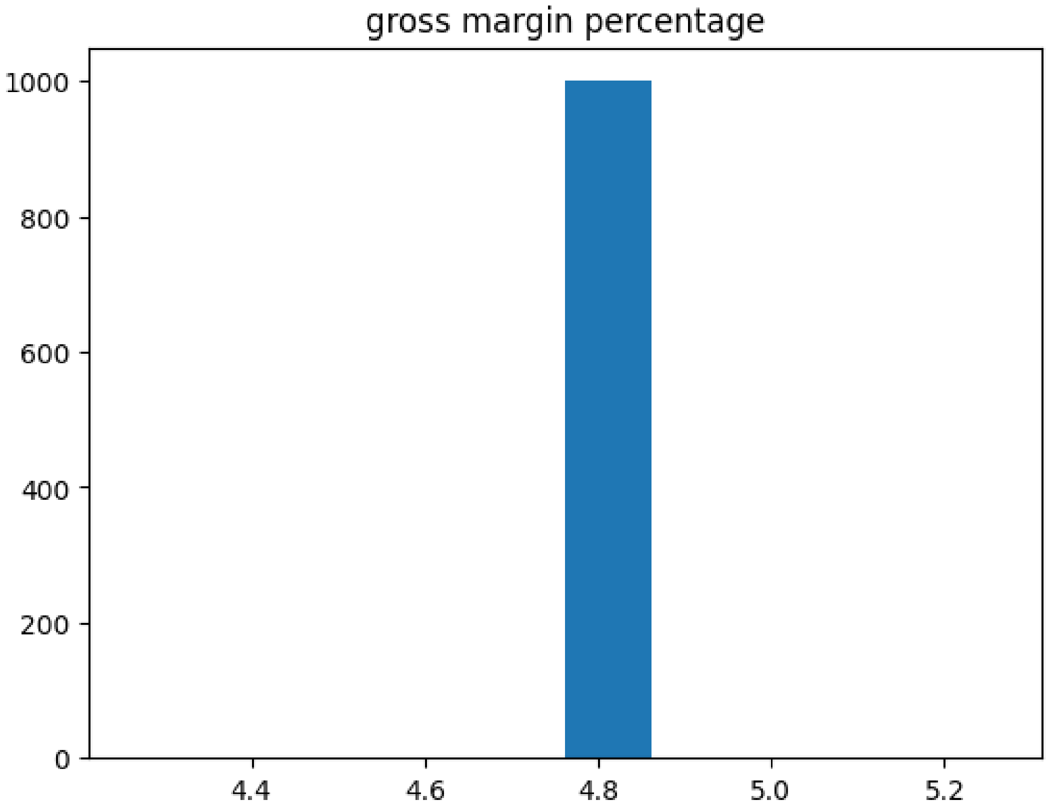

Unit price
0
20
40
60
80
100
120
20
40
60
80
100

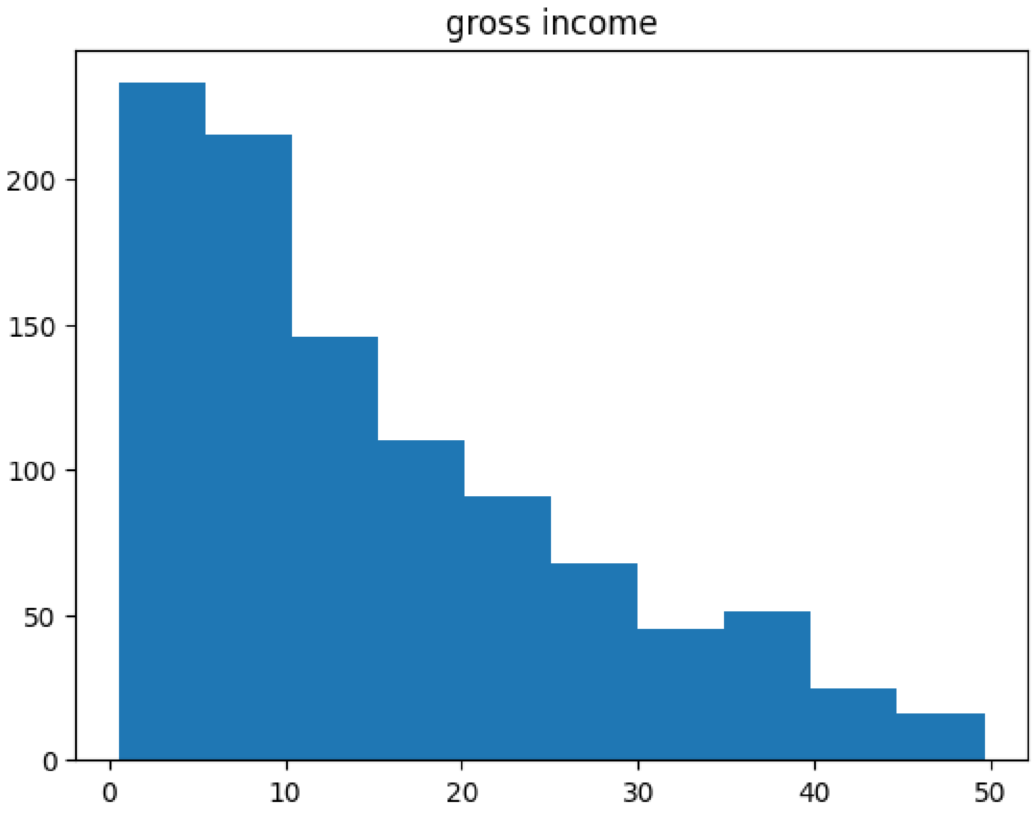

Quantity
120
100
80
60
40
20
0
2
4
6
8
10

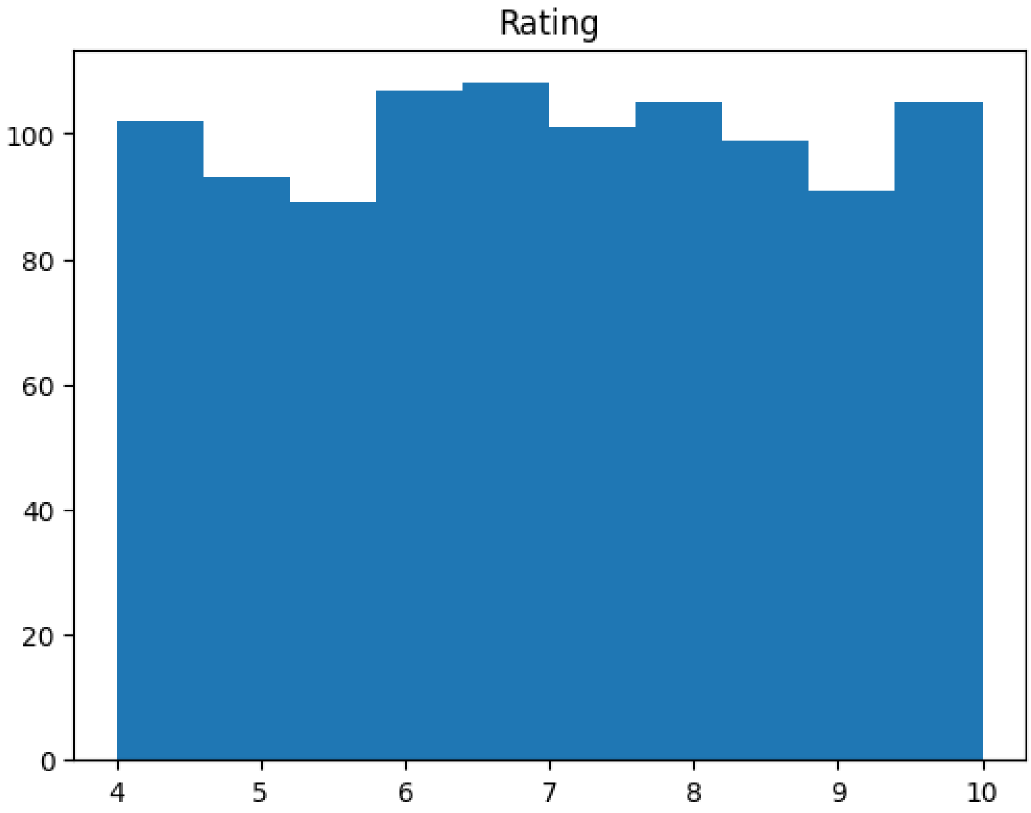

Tax 5%
0
50
100
150
200
0
10
20
30
40
50


Total
0
50
100
150
200
0
200
400
600
800
1000

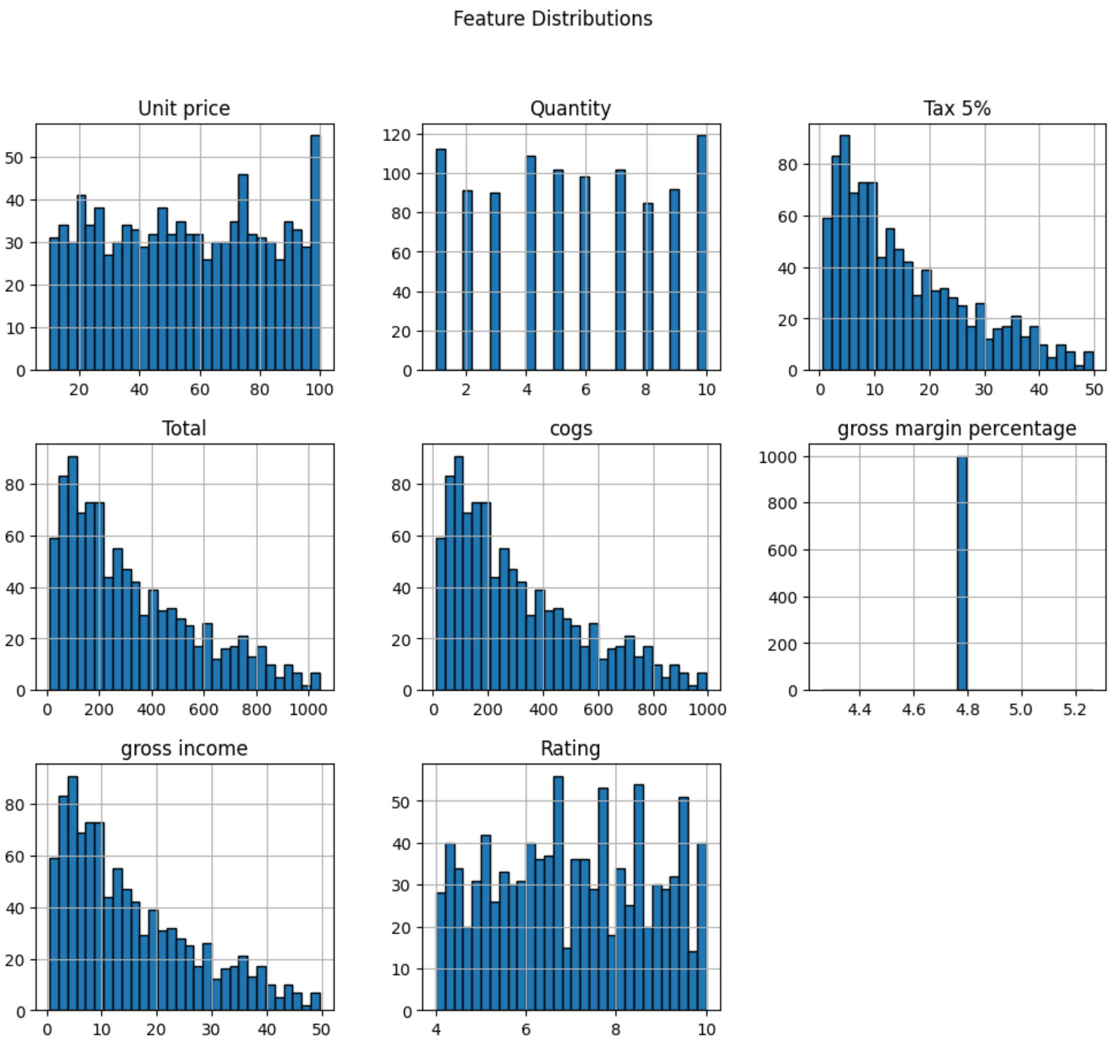

cogs
0
50
100
150
200
0
200
400
600
800
1000


gross margin percentage
1000
800
600
400
200
0
4.4
4.6
4.8
5.0
5.2

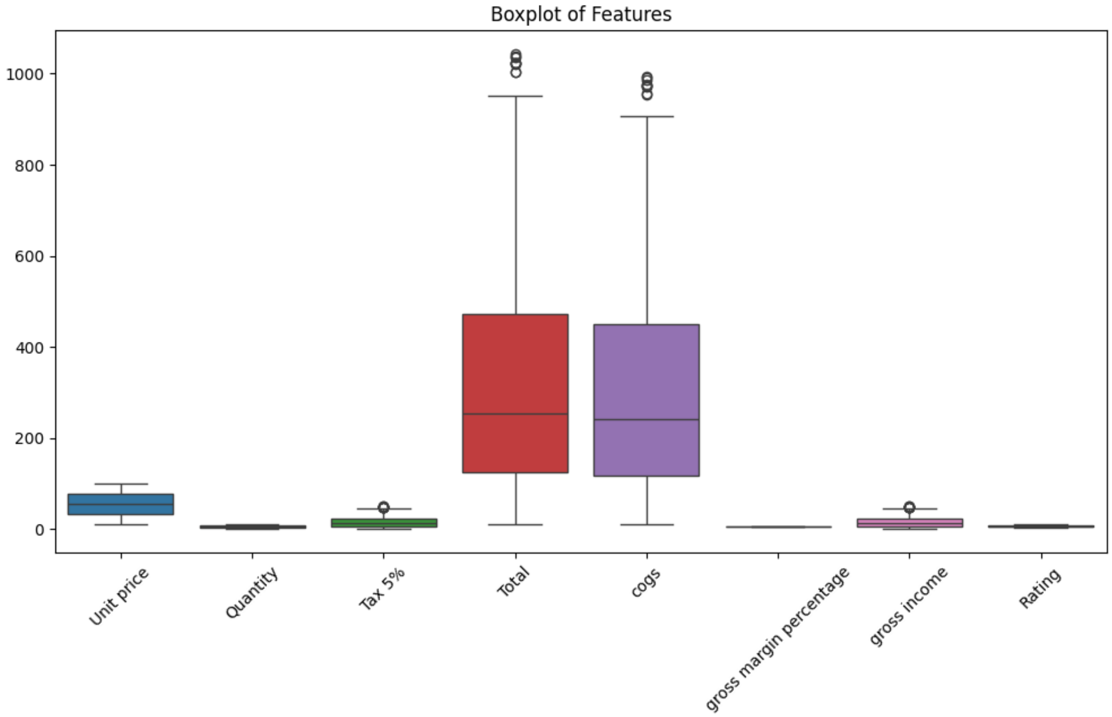

gross income
0
50
100
150
200
0
10
20
30
40
50



```
[ ]: # Histogram of each feature for distribution analysis
     # It is a univariate analysis plot.
     data.hist(figsize=(12, 10), bins=30, edgecolor='black')
     plt.suptitle('Feature Distributions')
     plt.show()
```

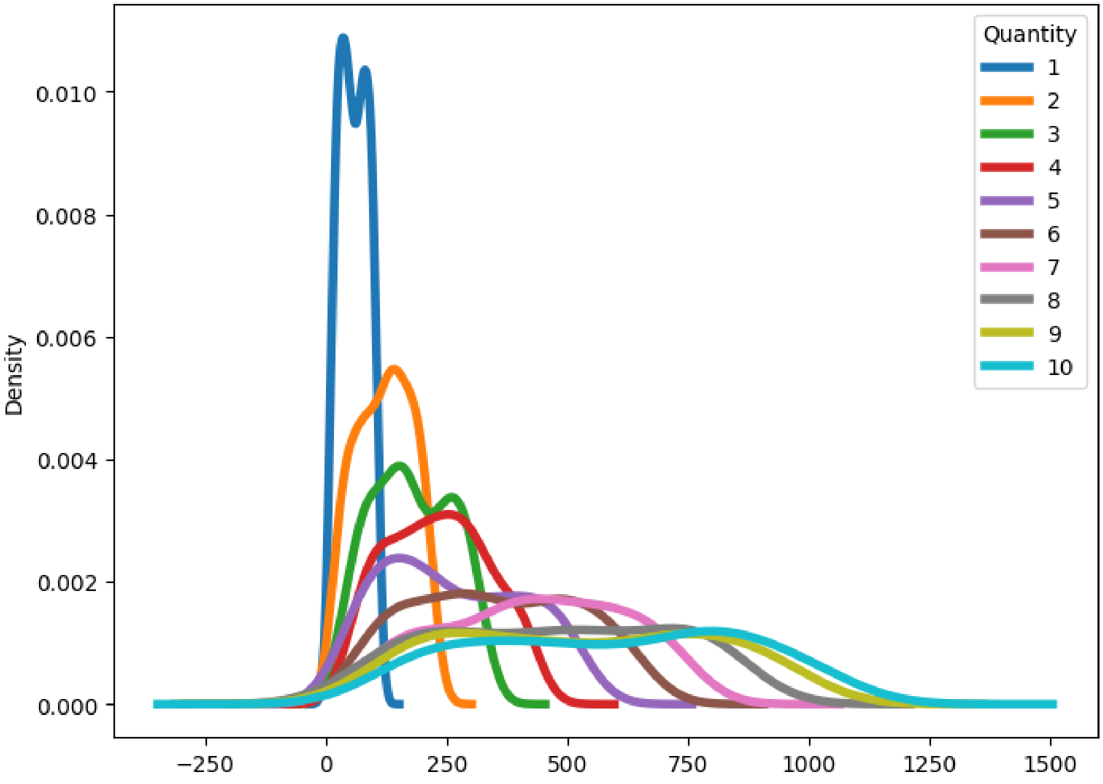


[ ]:
```python
# A box plot is a graphical representation of the distribution of a dataset.
# It shows the minimum, first quartile (Q1), median, third quartile (Q3), and maximum values, providing a summary of the data's spread and central tendency
# It is a univariate analysis plot.
plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.title('Boxplot of Features')
plt.xticks(rotation=45)
plt.show()
```



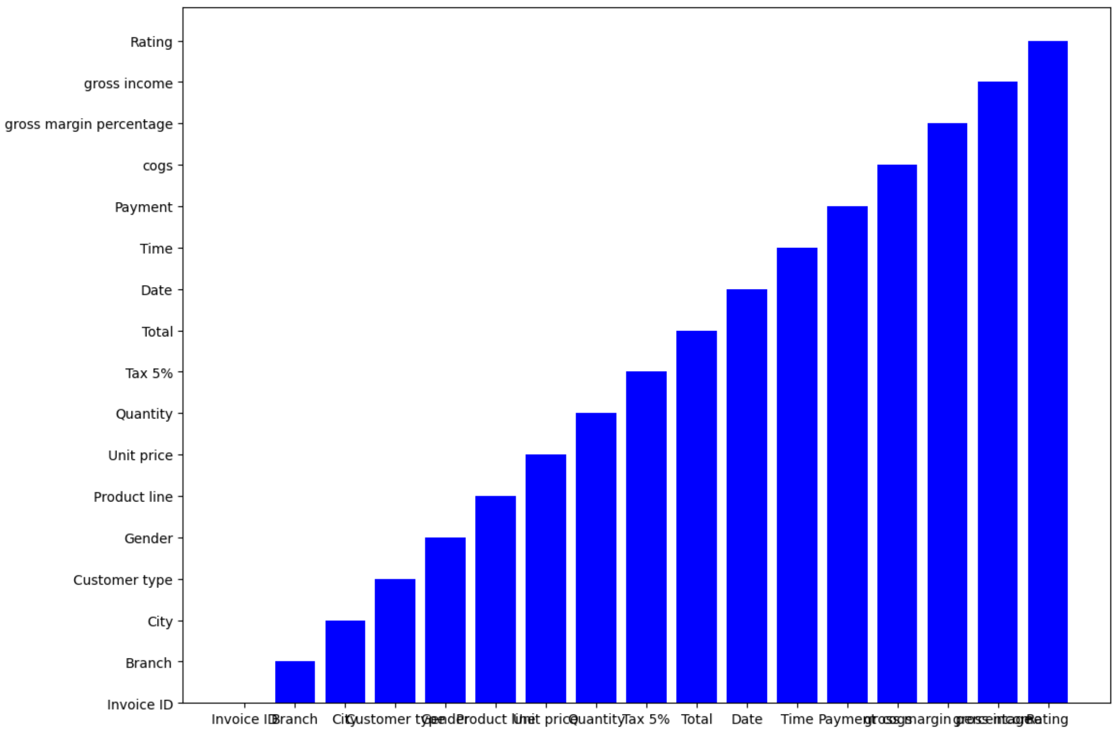
```
[ ]: #A density plot is a smoothed version of a histogram.
     #It estimates the probability density function of a continuous random variable
     # It is a univariate analysis plot.
     data_wide = data.pivot(columns = 'Quantity',
                          values = 'Total')

     # plotting multiple density plot
     data_wide.plot.kde(figsize = (8, 6),
                       linewidth = 4)
```

```
[ ]: <Axes: ylabel='Density'>
```

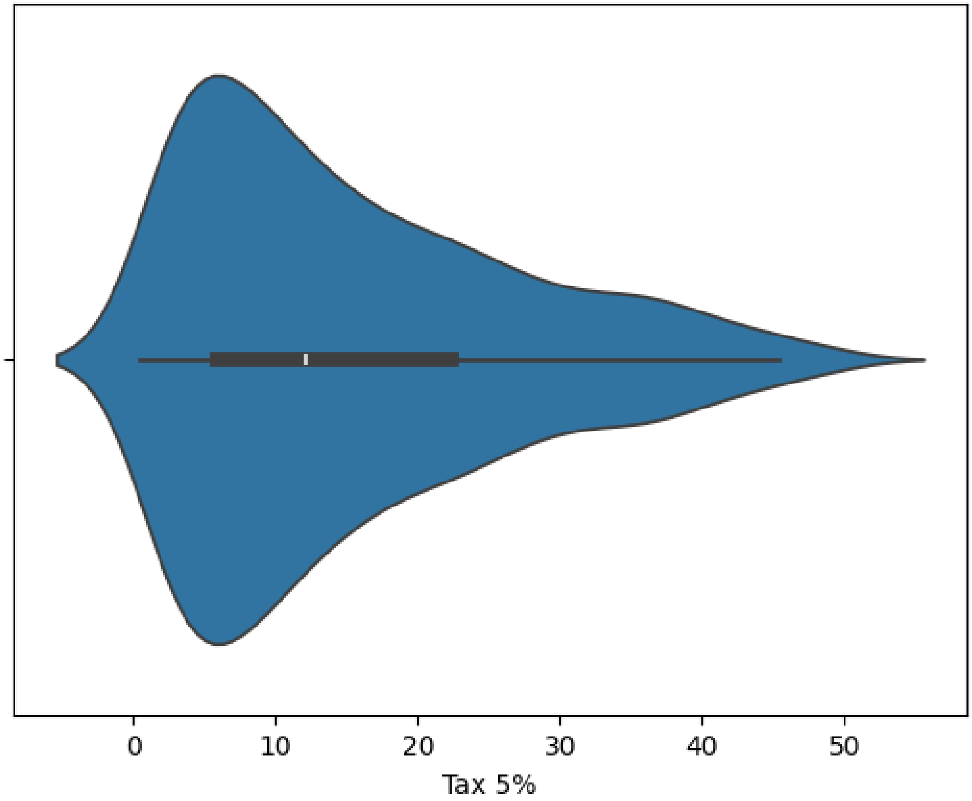


```
[ ]: # A bar plot (or bar chart) is a graphical representation of categorical data, where individual bars represent the frequency or value of categories.
     # It is a univariate analysis plot.
     fig = plt.figure()
     ax = fig.add_axes([1,1,1.5,1.5])
     x = list(data.iloc[:0])
     y = list(data.iloc[:1])
     ax.bar(x,y,color='b')
     plt.show
```

```
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>
```

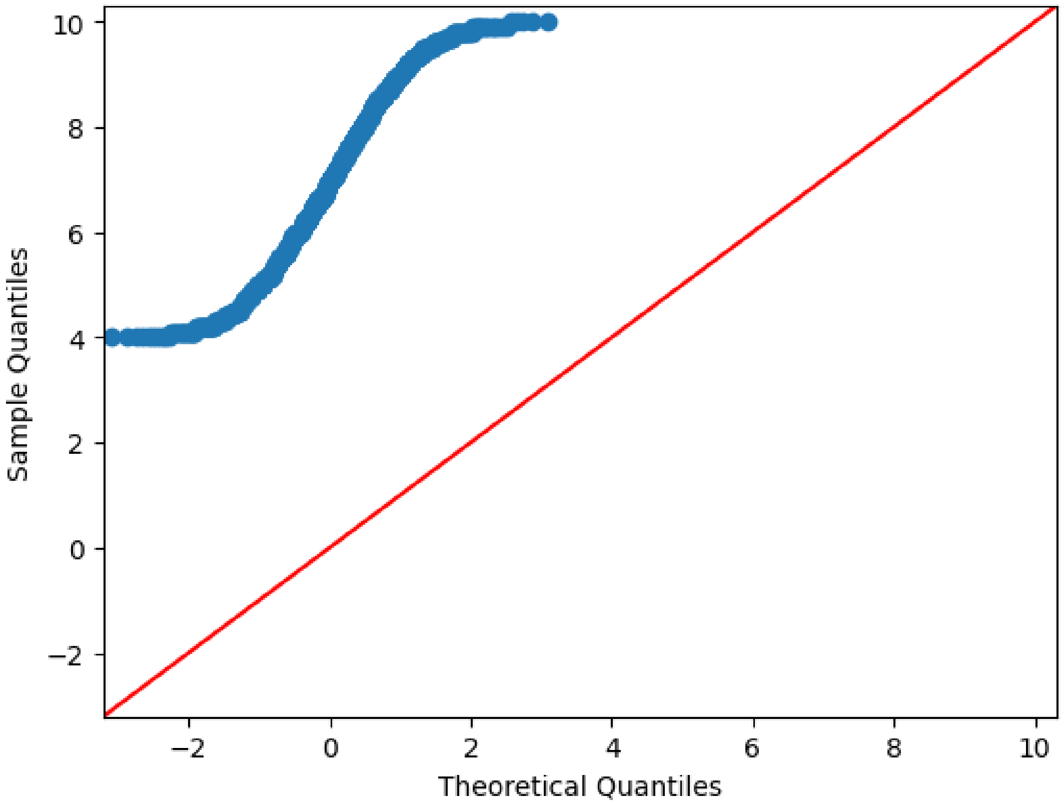


```
[ ]: #A violin plot is a combination of a box plot and a kernel density estimate (KDE).
     # It shows the distribution of a numerical variable for different categories, providing more detailed information
     # It is a univariate analysis plot.
     sns.violinplot(x='Tax 5%', data=data)
     plt.show()
```



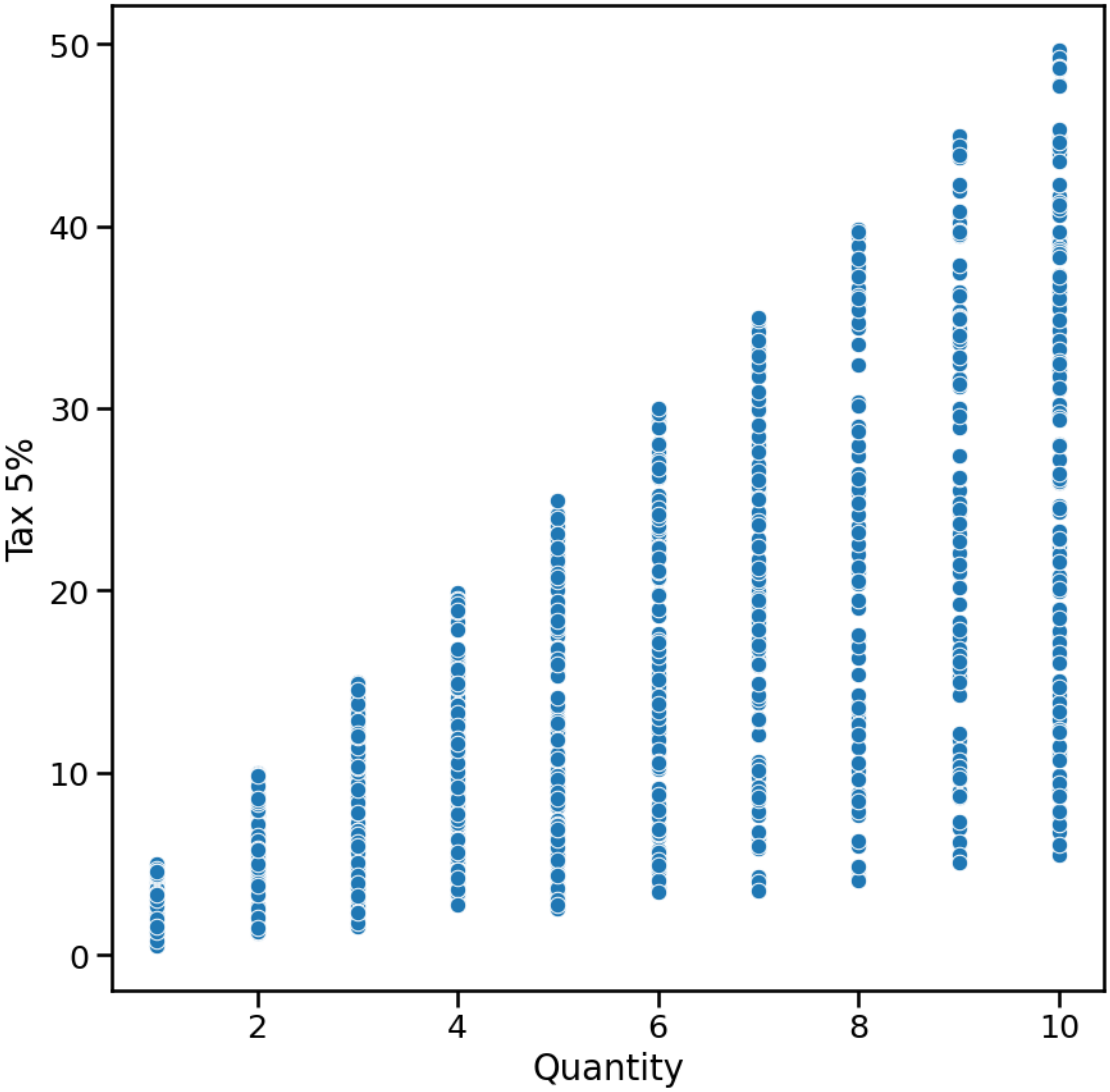
[ ]:

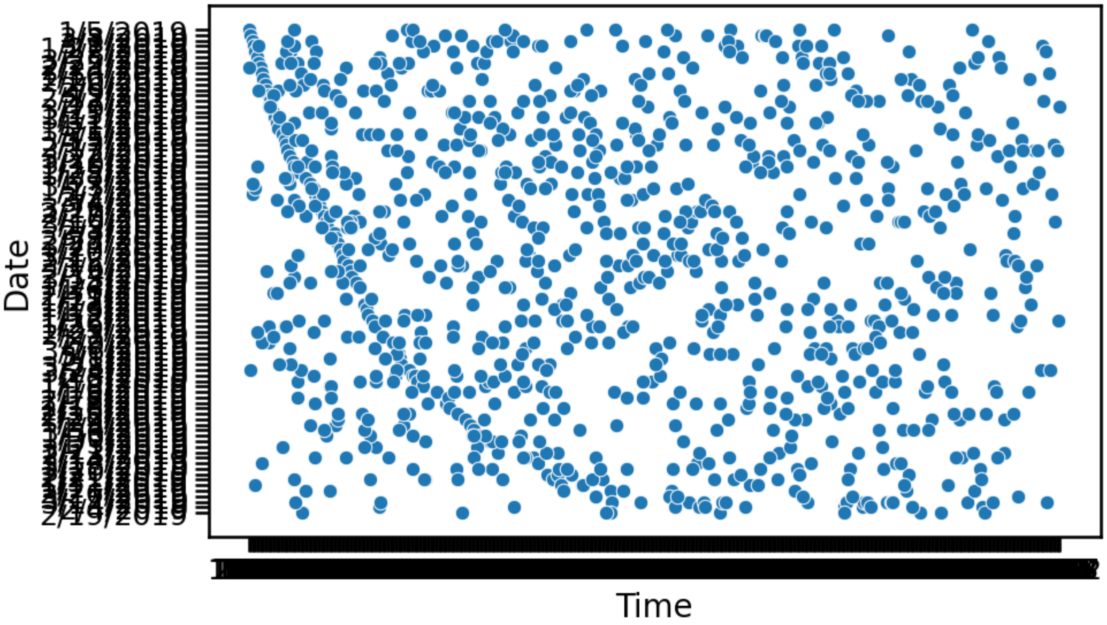
```
#A QQ plot is a graphical tool used to assess if a dataset follows a certain theoretical distribution, such as the normal distribution.
# It is a univariate analysis plot.
data1 = data['Rating']
data1
fig = sm.qqplot(data1, line='45')
plt.show()
```



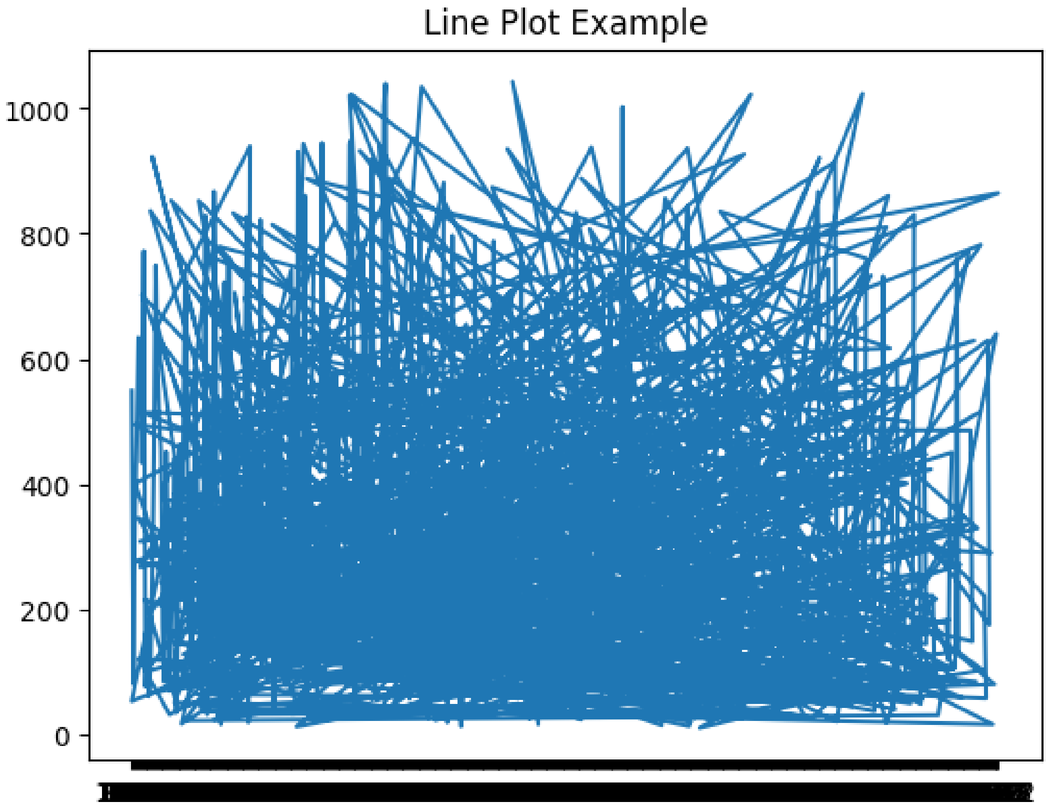
[ ]:

```
#A scatter plot is a type of data visualization that uses dots to represent␣
  ↪individual data points in two-dimensional space
# It is a bivariante analysis plot.
plt.figure(figsize=(10, 10))
sns.scatterplot(x='Quantity', y='Tax 5%', data=data)
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Time', y='Date', data=data)
plt.show()
```

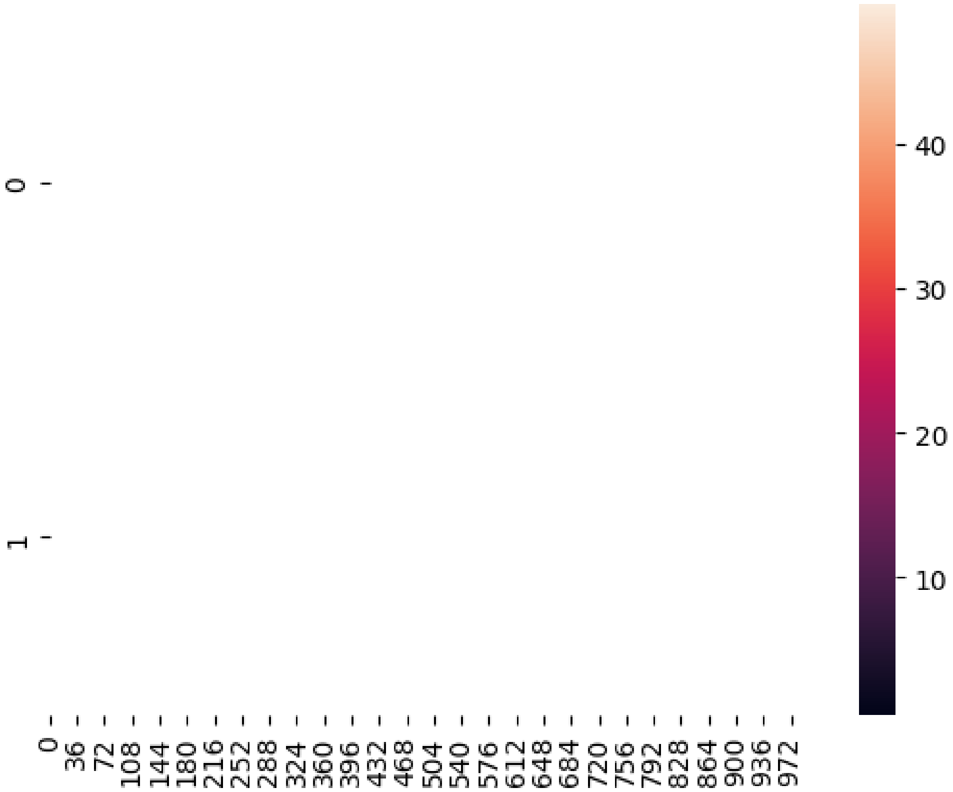

50
40
30
20
10
0
Tax 5%
2
4
6
8
10
Quantity



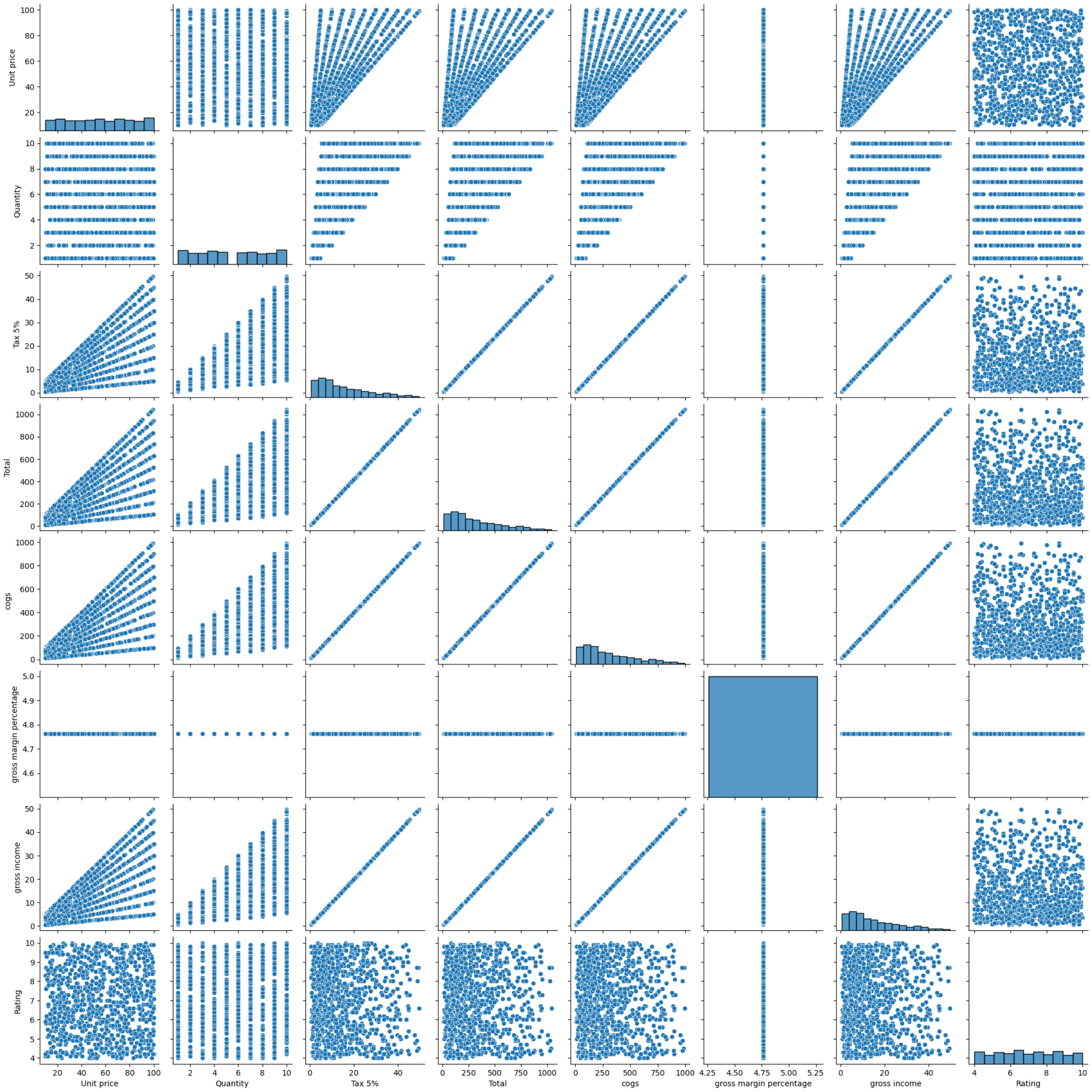
```
[ ]: #A line plot (or line chart) is a graphical representation of data points connected by straight lines.
     # It is a bivariante analysis plot.
     plt.plot(data['Time'], data['Total'])
     plt.title("Line Plot Example")
     plt.show()
```



```
[ ]: #A heatmap is a graphical representation of data where individual values are␣
      ↪represented by colors.
     # It is a bivariante analysis plot.
     data1 = data['Rating']
     data2 = data['gross income']
     b = np.array(data2)
     a = np.array(data1)
     c = np.array([b, a])
     sns.heatmap(c, linewidth=0.5)
     plt.show()
```

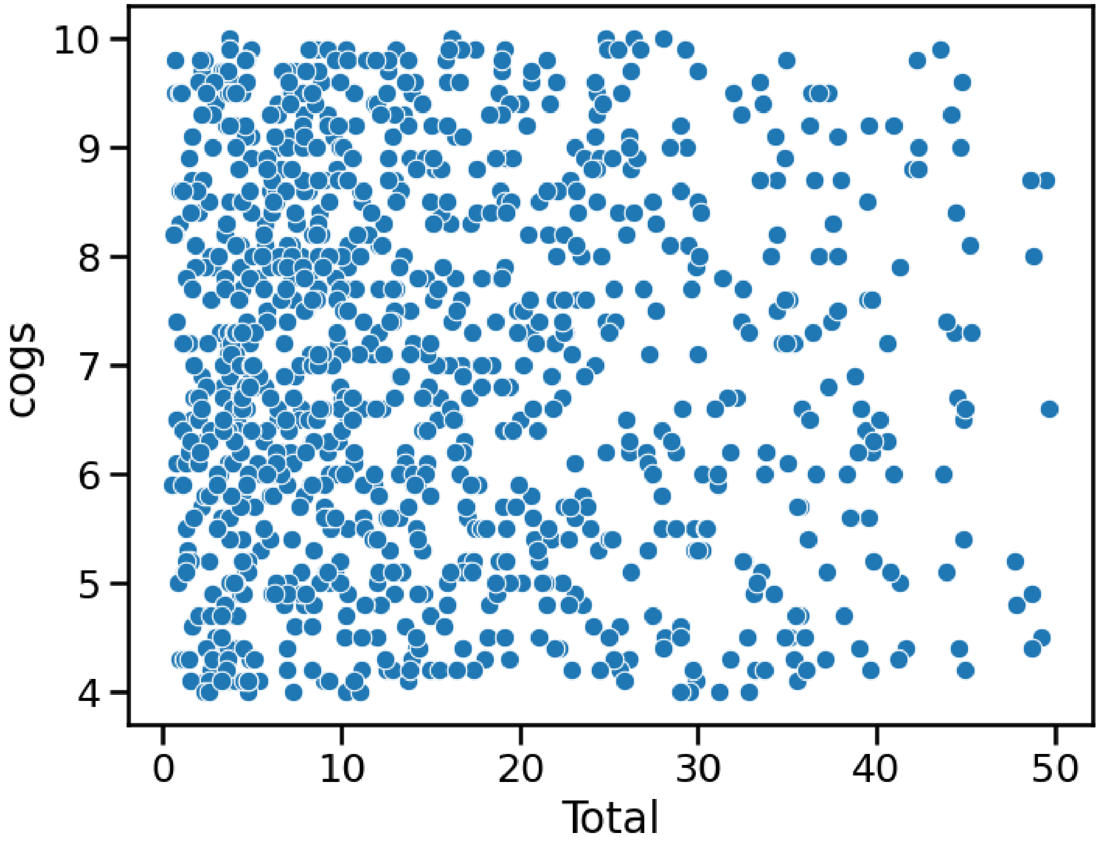


```
[ ]: #A pair plot  is a type of visualization that shows pairwise relationships␣
     ↪between variables in a dataset.
     # It is a bivariante analysis plot.
     sns.pairplot(data)
     plt.show()
```

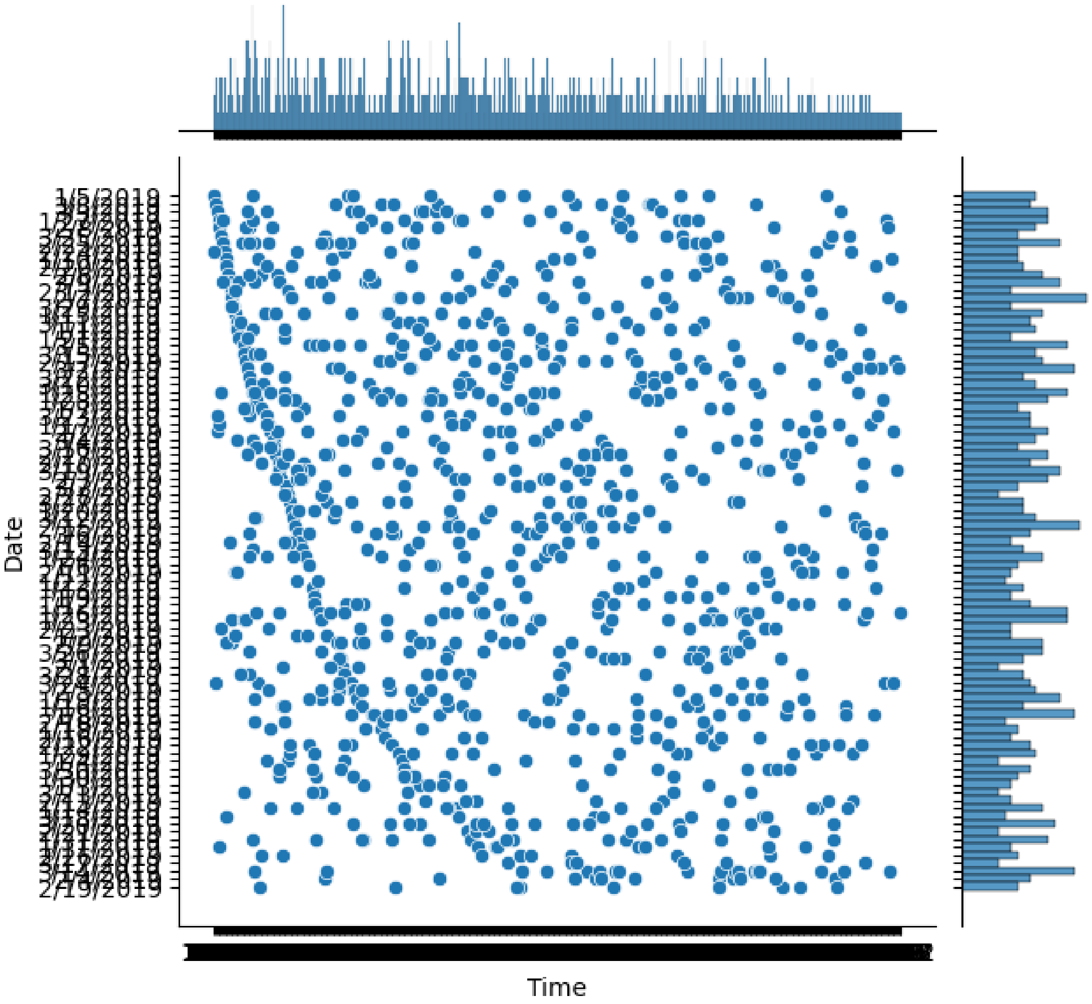


```
[ ]: #A bubble plot is a type of data visualization where each point in the plot is represented by a bubble,
     # and the size of the bubble corresponds to a third variable
     # It is a bivariante analysis plot.
     sns.set_context("talk", font_scale=1.1)
     plt.figure(figsize=(8, 6))
     sns.scatterplot(x="gross income",
                     y="Rating",
                     data=data)

     plt.xlabel("Total")
     plt.ylabel("cogs")
```

```
[ ]: Text(0, 0.5, 'cogs')
```

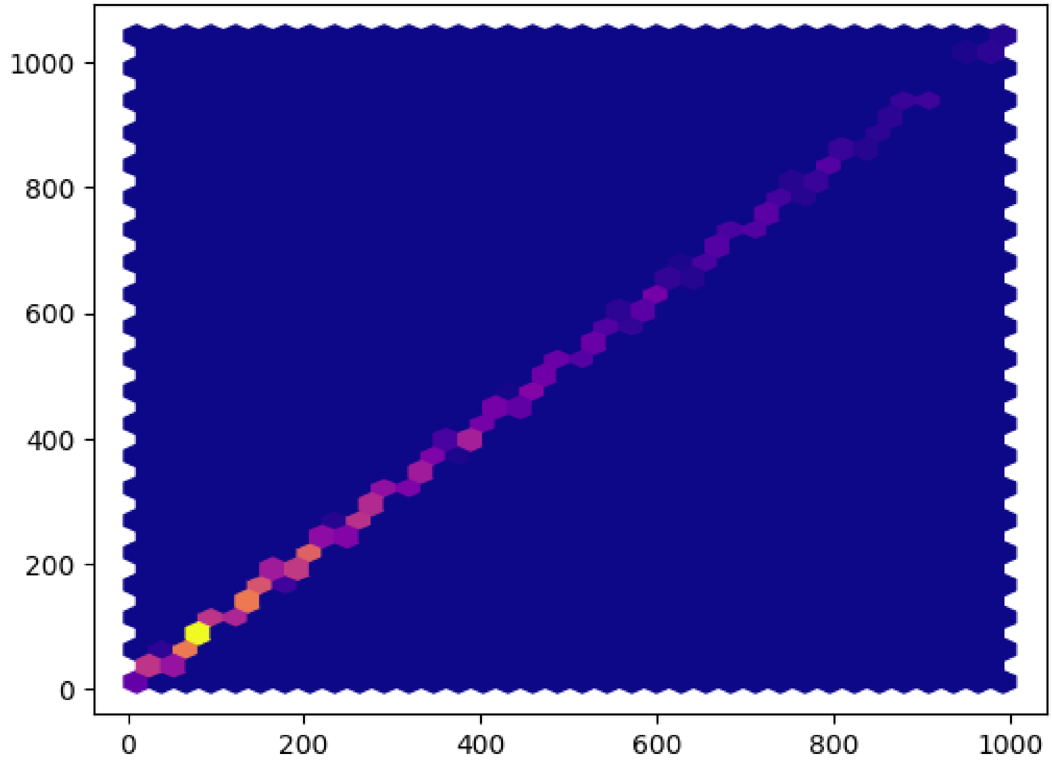


```
[ ]: #A joint plot is a type of data visualization in Python  that shows
     # the relationship between two variables while also displaying their individual␣
      ↪distributions.
     # It is a bivariante analysis plot.
     sns.jointplot(x='Time', y='Date', data=data, kind='scatter')
     plt.show()
```

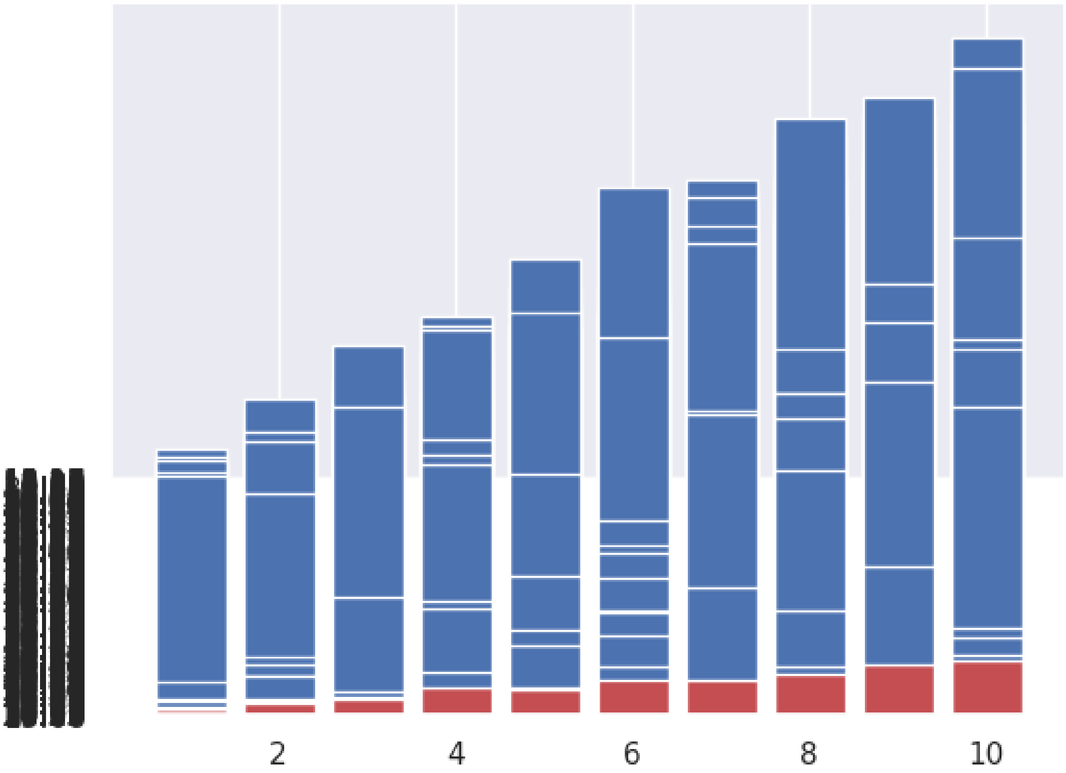


```
[ ]: #A Hexbin plot is a type of data visualization used to represent the density of␣
      ↪data points in a two-dimensional space.
     # It is a bivariante analysis plot.
     plt.hexbin(data['cogs'], data['Total'], gridsize=35, cmap="plasma")
     y
```

```
[ ]: array([[ 0.89918019,  0.85017771, -0.13406132, …,  0.33990667,
              0.94082558,  1.55526607]])
```



```
[ ]: # A stacked bar plot is a type of bar plot where each bar is divided into␣
      ↪multiple segments,
     # with each segment representing a different category within the bar.
     # It is a bivariante analysis plot.
     x = data['Quantity']
     y1 = data['Total']
     y2 = data['Time']
     plt.bar(x, y1, color='r')
     plt.bar(x, y2, bottom=y1, color='b')
     plt.show()
```


2
4
6
8
10