In [5]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\swathi\appdata\local\programs\python\python311\lib\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\swathi\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\swathi\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.11.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\swathi\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (1.3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\swathi\appdata\local\programs\python\python311\lib\site-packages (from scikit-learn) (3.2.0)
```

In [148…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [196…
```
df = pd.read_csv('market1.csv')
print(df)
```

```
     Unit price  Quantity   Tax 5%      Total    cogs  \
0         74.69         7  26.1415   548.9715  522.83   
1         15.28         5   3.8200    80.2200   76.40   
2         46.33         7  16.2155   340.5255  324.31   
3         58.22         8  23.2880   489.0480  465.76   
4         86.31         7  30.2085   634.3785  604.17   
..          ...       ...      ...        ...     ...   
995       40.35         1   2.0175    42.3675   40.35   
996       97.38        10  48.6900  1022.4900  973.80   
997       31.84         1   1.5920    33.4320   31.84   
998       65.82         1   3.2910    69.1110   65.82   
999       88.34         7  30.9190   649.2990  618.38   

     gross margin percentage  gross income  Rating  
0                   4.761905       26.1415     9.1  
1                   4.761905        3.8200     9.6  
2                   4.761905       16.2155     7.4  
3                   4.761905       23.2880     8.4  
4                   4.761905       30.2085     5.3  
..                       ...           ...     ...  
995                 4.761905        2.0175     6.2  
996                 4.761905       48.6900     4.4  
997                 4.761905        1.5920     7.7  
998                 4.761905        3.2910     4.1  
999                 4.761905       30.9190     6.6  

[1000 rows x 8 columns]
```

In [197…
```
df.head()                ###for printing the first 5 rows using "head"
```

Out[197]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
0	74.69	7	26.1415	548.9715	522.83	4.761905	26.1415	9.1
1	15.28	5	3.8200	80.2200	76.40	4.761905	3.8200	9.6
2	46.33	7	16.2155	340.5255	324.31	4.761905	16.2155	7.4
3	58.22	8	23.2880	489.0480	465.76	4.761905	23.2880	8.4
4	86.31	7	30.2085	634.3785	604.17	4.761905	30.2085	5.3

In [198…
```
df.describe()                                    ##For getting description of the dataset i.e. Average,Maximum,M
```

Out[198]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905	49.650000	10.00000

In [199…
```
df.info()                                   ##For checking whether the dataset has null values or not and getting a
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Unit price               1000 non-null   float64
 1   Quantity                 1000 non-null   int64  
 2   Tax 5%                   1000 non-null   float64
 3   Total                    1000 non-null   float64
 4   cogs                     1000 non-null   float64
 5   gross margin percentage  1000 non-null   float64
 6   gross income             1000 non-null   float64
 7   Rating                   1000 non-null   float64
dtypes: float64(7), int64(1)
memory usage: 62.6 KB
```

In [129…
```
df.isnull().sum()                                   ##We can also check the null values by using "isn
```

Out[129]:
```
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
cogs                      0
gross margin percentage   0
gross income              0
Rating                    0
dtype: int64
```

In [130…
```
df= df.drop(['gross margin percentage'], axis=1)                    ##for deleting a column
```

In [131…
```
df.head()
```

Out[131]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income	Rating
0	74.69	7	26.1415	548.9715	522.83	26.1415	9.1
1	15.28	5	3.8200	80.2200	76.40	3.8200	9.6
2	46.33	7	16.2155	340.5255	324.31	16.2155	7.4
3	58.22	8	23.2880	489.0480	465.76	23.2880	8.4
4	86.31	7	30.2085	634.3785	604.17	30.2085	5.3

In [132…
```
df
```

Out[132]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income	Rating
0	74.69	7	26.1415	548.9715	522.83	26.1415	9.1
1	15.28	5	3.8200	80.2200	76.40	3.8200	9.6
2	46.33	7	16.2155	340.5255	324.31	16.2155	7.4
3	58.22	8	23.2880	489.0480	465.76	23.2880	8.4
4	86.31	7	30.2085	634.3785	604.17	30.2085	5.3
...	...	...	...	...	...	...	...
995	40.35	1	2.0175	42.3675	40.35	2.0175	6.2
996	97.38	10	48.6900	1022.4900	973.80	48.6900	4.4
997	31.84	1	1.5920	33.4320	31.84	1.5920	7.7
998	65.82	1	3.2910	69.1110	65.82	3.2910	4.1
999	88.34	7	30.9190	649.2990	618.38	30.9190	6.6

1000 rows × 7 columns

In [133…
```
y=df['Rating']
y
```

Out[133]:
```
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
```

In [160…
```
x=df.drop('Rating',axis=1)
x
```

Out[160]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income
0	74.69	7	26.1415	548.9715	522.83	26.1415
1	15.28	5	3.8200	80.2200	76.40	3.8200
2	46.33	7	16.2155	340.5255	324.31	16.2155
3	58.22	8	23.2880	489.0480	465.76	23.2880
4	86.31	7	30.2085	634.3785	604.17	30.2085
...	...	...	...	...	...	...
995	40.35	1	2.0175	42.3675	40.35	2.0175
996	97.38	10	48.6900	1022.4900	973.80	48.6900
997	31.84	1	1.5920	33.4320	31.84	1.5920
998	65.82	1	3.2910	69.1110	65.82	3.2910
999	88.34	7	30.9190	649.2990	618.38	30.9190

1000 rows × 6 columns

In [183…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=100)
```

In [184…
```
x_train
```

Out[184]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income
675	83.77	2	8.3770	175.9170	167.54	8.3770
358	27.50	3	4.1250	86.6250	82.50	4.1250
159	93.39	6	28.0170	588.3570	560.34	28.0170
533	16.31	9	7.3395	154.1295	146.79	7.3395
678	58.95	10	29.4750	618.9750	589.50	29.4750
...	...	...	...	...	...	...
855	36.51	9	16.4295	345.0195	328.59	16.4295
871	56.50	1	2.8250	59.3250	56.50	2.8250
835	52.38	1	2.6190	54.9990	52.38	2.6190
792	97.37	10	48.6850	1022.3850	973.70	48.6850
520	45.71	3	6.8565	143.9865	137.13	6.8565

800 rows × 6 columns

In [182…
```
x_test
```

Out[182]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross income
249	73.06	7	25.5710	536.9910	511.42	25.5710
353	27.00	9	12.1500	255.1500	243.00	12.1500
537	97.94	1	4.8970	102.8370	97.94	4.8970
424	16.28	1	0.8140	17.0940	16.28	0.8140
564	99.25	2	9.9250	208.4250	198.50	9.9250
...	...	...	...	...	...	...
684	23.08	6	6.9240	145.4040	138.48	6.9240
644	12.05	5	3.0125	63.2625	60.25	3.0125
110	16.49	2	1.6490	34.6290	32.98	1.6490
28	88.36	5	22.0900	463.8900	441.80	22.0900
804	75.59	9	34.0155	714.3255	680.31	34.0155

200 rows × 6 columns

In [181…
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
y_train_pred=lr.predict(x_train)
y_test_pred=lr.predict(x_test)
```

In [154…
```
print(x_train_pred)   ##Output
```

```
[7.15881107 6.94207906 6.86437554 7.1674297  6.83310564 6.97410457
 6.92331725 6.7632332  7.17933751 7.10123743 7.09801634 6.94440846
 7.01937665 6.50940801 7.1733222  7.02171042 6.95317914 6.98903483
 6.99872059 6.90839078 6.84437382 7.00857484 6.63639504 6.95480809
 7.10560566 7.12950012 6.97578595 7.03959137 6.85484794 6.90245155
 6.9563065  6.96722995 7.11938758 6.99002828 6.99024867 7.1695144
 7.09777017 6.69816535 7.05109625 7.06056885 6.97587606 7.03017787
 6.9487636  7.00470024 6.96495688 6.94329215 6.95947661 6.96676385
 7.03346436 6.85136705 6.81605607 6.68571682 6.95247831 6.91985711
 7.02722699 6.97847035 6.76028047 7.20003294 6.79236728 7.04025404
 6.74997652 7.17081686 7.00602181 7.12978801 6.86940187 6.96111948
 6.94538081 6.94643651 6.57883745 6.98584231 6.93026421 6.62967895
 6.96504137 6.97134068 6.91811714 6.93708506 7.06913318 7.08199276
 6.97508512 6.96102845 6.97561401 7.04802998 7.00613901 6.97113167
 6.99451957 6.80511943 6.85882118 6.98419497 6.97793126 7.11388205
 6.94664462 6.61836765 6.82181763 6.99905355 6.86182527 6.96426228
 6.86032346 7.16891321 7.12516698 6.99253618 6.9713928  6.72670871
 6.68834303 6.91032669 7.16969201 7.12881552 6.95952284 6.57047136
 6.87783966 7.11978381 6.98051081 6.8395192  6.82094362 7.25139157
 7.10806212 7.08048746 7.27554622 7.05159625 6.94965838 7.0074005
 6.95305519 6.91966729 7.0940609  7.19892    6.95974693 6.88824896
 6.73385956 7.02745526 6.91741187 7.1378791  6.89202811 7.01181869
 6.93732548 6.98536246 7.0434794  6.82727325 7.01649423 6.94889202
 6.92848952 6.72292667 6.69480731 7.00094579 7.16086575 7.18746751
 6.60617266 7.04516199 7.0125896  7.01430163 7.04939236 7.01878462
 7.01653606 7.15690207 7.00490048 6.93880437 6.95959984 7.03305387
 7.00132624 6.91118079 6.98176866 6.94347894 7.01624393 6.94384763
 6.88872942 7.00142178 6.99852298 7.16122176 6.92926385 7.05646269
 7.22314801 7.02725353 7.01868168 6.90738782 6.97500053 6.96575405
 6.98041504 7.03033064 6.66202221 7.01428065 6.74782903 7.03852711
 6.98598495 6.96246015 7.10567717 6.87399177 6.92663856 7.01696236
 6.61360023 6.87985702 6.98588701 6.62985569 7.18579511 6.71469046
 6.95255716 6.99527907 7.00084624 6.98243424 7.16892237 7.05875303
 7.01271975 7.05644789 6.71663458 6.98987266 6.97547941 6.86564842
 7.09184061 7.28028243 7.00470024 7.14547804 7.01096097 6.92707909
 6.97325742 6.89063349 7.03605743 6.49500377 6.99791484 6.5604856
 6.74459044 6.96696314 6.89742182 6.66450104 6.95037203 6.97229355
 6.64965852 6.99110996 6.9989334  7.24694021 7.02471477 6.8921745
 6.77760957 7.00731334 6.91545348 6.70479307 7.05090963 6.9728807
 6.96918191 6.95005043 7.05139353 6.99679264 6.86362373 6.99689803
 6.98293343 6.95449182 6.7912099  7.07153145 7.15701324 7.00576006
 7.18023007 6.86098439 7.00662211 7.03984198 6.9615625  6.98275834
 6.97046965 7.06843374 6.65239348 7.13938028 6.98853107 7.10531731
 7.22781383 7.0693472  6.94081144 6.99413772 6.7655933  6.72213308
 6.96260137 7.10667486 7.22177823 6.96477551 7.05607744 6.89222487
 6.96236003 6.98974356 6.8002572  7.23321638 7.07134993 6.97663023
 7.04514818 6.91002948 7.01645418 6.84507779 6.95842215 6.97048967
 6.9787795  6.88964896 6.70399774 7.01715501 6.94661831 7.14853771
 7.0329403  6.9451147  6.9653336  7.0211097  7.01920155 6.75473139
 7.00528093 6.96071808 6.93804916 7.151236   6.6771334  7.064192
 7.17550529 7.04594753 7.21442022 6.57754092 6.81181495 6.96485472
 7.14070427 7.22751419 7.02045893 6.96232625 6.95761833 6.98678763
```

```
7.05817147 6.83608545 6.80519373 6.98640855 6.60856215 6.82276487
7.11575845 6.89857122 6.95586935 6.9948786  6.9766798  6.84664764
6.92030518 6.89566101 7.01810801 7.12878075 6.98470653 7.00334864
6.78403144 6.98730684 7.08120792 6.7241338  6.96204966 7.11265335
7.14079982 6.66227801 6.98280428 6.99815265 6.98611821 6.98897159
6.88846374 6.78826847 6.87900733 6.98520712 6.85722078 6.48342736
6.71529726 6.77964111 7.18030958 6.75633772 6.856518   6.97290626
7.04307168 7.06024794 6.96843502 6.83348085 7.02792779 6.99163475
7.08105191 6.99529274 6.93365466 6.95297218 7.01040378 7.15697043
7.0444197  7.0134306  6.90073563 7.3161592  6.63504808 7.07290814
7.11711643 6.96198784 6.94349851 6.99093365 7.0138317  6.99178677
6.97753803 6.85471953 6.95873148 7.00160657 6.99677348 7.20363194
7.00118554 6.91657373 6.84762996 6.73774986 7.00730874 7.11009821
6.97179525 7.15584613 6.67959914 6.91173917 7.01620849 7.19917683
7.13428886 6.95047705 7.13762709 6.98330107 6.98438014 7.01391385
6.92257352 7.05173189 6.93193726 7.21364515 7.0986831  6.92430301
7.13248634 7.10179934 6.89384392 7.22285593 7.09142069 6.99665217
6.80456228 7.0045168  6.56578777 6.9563398  6.99645473 7.12509188
7.08442236 6.98731725 6.96846865 7.18700626 7.03275351 7.01451188
6.95515148 7.06117132 6.94616574 7.1963627  6.88629176 6.56703393
6.99628292 6.85116202 6.97100123 6.88167144 7.01185792 6.96473267
6.96988329 6.47936236 6.96220115 6.85416727 6.94419915 6.56638618
7.09603284 6.77577043 6.58509907 6.76665175 6.77224433 6.91987155
6.72774884 7.09821409 6.69056606 6.63252904 7.03262494 7.08124718
6.80666177 6.9057658  6.98736969 7.10684498 7.19948577 6.98126963
6.88754128 6.79671572 6.93479023 6.981733   6.85876004 7.13221677
6.97512823 7.02069799 6.99184045 7.22714544 6.86387994 6.8372812
6.99857406 6.95544129 7.25464771 7.11836253 6.89505269 6.58434538
6.98737011 7.12067227 7.28383459 7.02863327 6.98667237 7.00767846
6.98512719 6.97356332 7.05927919 6.92944793 6.57250385 7.03724997
7.10089861 7.04819593 6.99656278 6.78356701 6.99974517 6.90859273
6.94595504 7.13034723 6.67183123 6.94192786 7.21380163 7.03433539
6.96404523 7.04237077 6.51064518 6.49244105 6.83467394 6.97900413
6.89375095 6.96702556 6.87998613 6.98359339 7.0571992  7.07032008
6.87155185 6.96755619 7.19400686 7.15252326 6.98978826 7.13244875
6.6748937  6.86040551 7.31195581 7.08093641 6.8050925  7.02898145
7.1251447  7.02810928 6.94425662 6.62428841 6.76582237 6.94289923
6.94899273 7.08339242 7.18513497 6.89530505 6.95657317 6.96642544
6.86349184 6.81266595 6.92094497 6.95213942 7.04665781 7.00089183
7.15084922 6.56331343 6.87366726 6.87198824 7.03011942 6.97097024
6.86687195 7.04622192 7.27181646 7.07066951 7.02221647 6.99020329
7.01913737 7.02490969 7.21707131 6.89955084 6.69622122 6.9987632
6.92698029 7.10224829 6.98468394 6.99753749 7.09358621 6.93072272
6.95289794 7.17289962 6.94885384 7.21012737 6.96422224 6.94593485
6.97997091 6.99690099 6.98702354 7.06804892 7.01621518 6.98495683
7.10860675 6.9684434  6.97546024 6.90856907 6.74739737 6.82823089
6.76879659 6.94978443 6.96102845 6.97967437 6.97174678 7.10118983
6.99113648 6.82779992 6.91759781 6.77855212 6.93009915 6.98597044
7.09136699 6.64274915 7.02285177 7.16886317 6.96031025 6.8162228
6.77463424 6.7232623  7.17987485 7.04862925 7.14079982 7.04912116
7.02699297 7.03822    7.15696345 6.57056059 7.07497919 7.32119142
6.90624099 7.03958169 6.99045335 7.03443551 7.01366087 6.71763144
6.95110511 7.10860001 6.94306405 7.23474522 6.99007559 6.93254282
6.72609347 6.81227692 6.89048605 6.8953178  7.02914879 6.80524091
6.97381994 6.85476156 7.27335573 7.00081365 6.89098483 6.87630332
6.93243903 6.55297419 6.90173791 6.98823071 7.29431344 6.99966147
6.98850759 7.00140634 6.87102048 7.1741914  6.97779718 7.04903714
6.97812572 6.9747579  6.69045491 7.0102137  6.56537856 7.01223918
6.97702742 6.95574218 6.99269983 6.91556051 7.0708454  6.88765867
6.68506406 7.00747353 7.047554   6.9672938  7.03495613 7.10961394
6.98728872 7.00432538 6.78554287 6.92729014 6.86057224 7.12381
6.99910402 7.11877619 6.66361286 7.04775132 6.93186325 7.07626205
6.82811335 7.06078607 7.30621316 6.9379708  6.96320738 6.86340908
7.26234405 7.03035449 6.9103681  7.01056719 7.04134081 6.96792592
6.99826832 6.89459983 7.09237864 6.87424011 6.95498985 6.79365905
7.09729931 6.83922319 7.20308226 7.03731318 6.94893685 6.8125548
7.19823511 6.86724606 7.22738577 6.98203683 6.91720199 6.97237604
6.85373631 7.16471899 6.85886873 7.04158733 6.71957659 6.86693
6.65795721 6.62640928 6.91171639 6.9478067  6.85266562 6.94954342
7.03598867 6.85172313 6.66705752 6.99970021 6.65289729 6.84997736
7.09971464 7.19909122 6.96633114 6.96664363 6.98993273 7.25381888
7.1300759  6.99946796 6.91097658 6.84970397 7.00179019 7.21433185
6.98252848 7.05031798 6.95177818 6.67430557 6.96484748 6.7349461
7.00108338 6.95212297 6.91227918 6.96015534 6.86517653 6.98544807
6.62384656 7.10771491 6.91950189 6.97957872 6.95295169 6.8082105
7.11563754 7.00230245 6.85724479 6.55429973 6.96944581 7.29999689
6.96209176 7.04092639 6.95667328 6.9705849  6.95905073 6.82160116
7.13541785 6.88565386 6.69806878 6.9555262  6.92512462 6.85290279
6.87050388 6.91089029 6.82678219 7.20010436 6.88233686 6.9779385
6.7067386  7.00302826 6.82235045 6.7124629  6.98504418 6.47653454
6.6063494  7.10687076 6.99489628 7.02204496 7.06549546 7.041104
6.49358986 6.99016944]
```

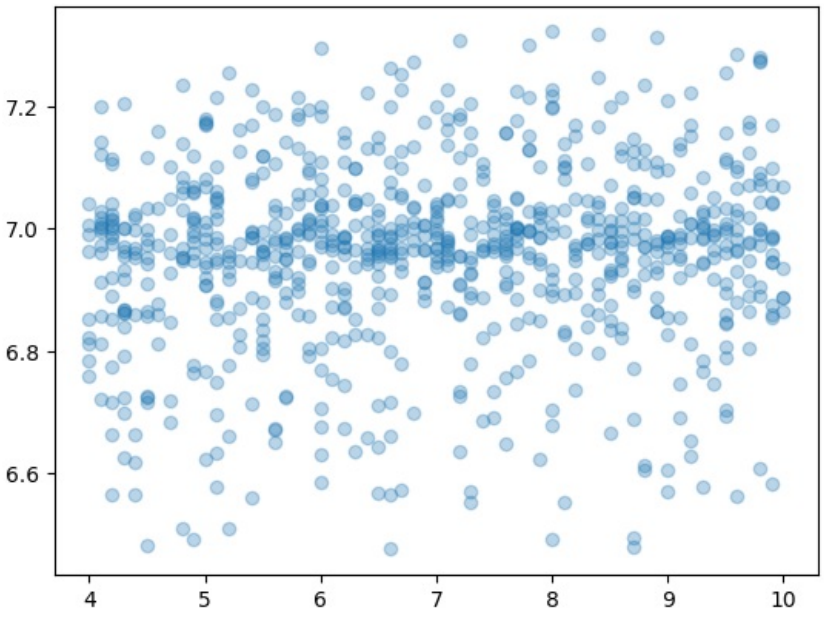
In [155…

```
print(y_test_pred)        ##Output
```

```
[6.86012246 7.09049095 7.31083096 6.82738267 7.22507427 6.90175386
 7.07487228 7.02653733 6.58322778 7.09583096 7.13275043 6.80993437
 7.0876983  7.06332175 6.73775491 6.958425   7.12633332 6.82063019
 7.22558793 6.80257679 6.92142721 6.99209326 6.84949894 6.86602022
 6.99390022 7.00788973 7.14834341 7.0340098  6.97983777 7.03355446
 7.06870337 6.90882309 6.8807242  6.97419406 6.99882308 6.83478466
 6.99049061 6.900987   7.00144962 6.99279729 6.95679682 6.94898154
 6.90314082 6.7257336  7.11219569 6.86312872 6.94133961 7.25719108
 7.14784322 6.84895691 6.95664236 6.78008392 7.0357771  6.89504002
 7.00036825 6.66705752 6.64865091 6.95888591 6.95845719 6.6939006
 6.95017229 7.03437431 6.93370031 7.10351591 6.84614383 6.81376608
 7.12477498 6.97580505 6.8204688  6.89126226 6.98222647 7.03571656
 6.9332681  7.02088285 6.94999004 7.24440567 7.15751024 7.24174155
 6.72393429 6.86349625 7.0792105  7.09443494 7.0219392  6.87680418
 6.81166894 6.98411583 6.84907403 6.66864992 6.97273478 6.72346689
 6.91413381 6.70335011 7.04079882 6.97943813 6.76564888 7.06070728
 6.89191889 6.71947198 6.76084329 6.96195014 6.7868     7.04987062
 7.26198883 6.96877513 7.27152045 7.00950594 6.99624021 7.09592953
 6.89346806 6.70850459 6.91103592 6.877908   7.11906408 7.07773712
 6.9683344  7.09699967 7.04041056 6.99645423 6.95077573 6.94250714
 6.94201139 6.89530505 7.02368959 7.17484263 6.9983427  7.19964769
 7.0351912  6.98874657 7.08799244 6.99424785 7.03506533 7.11493525
 6.97485235 6.93559186 7.05335893 6.90693219 7.15881107 7.12726329
 6.9376952  6.93616858 7.02729945 7.25559495 6.86491711 7.10192579
 7.01734407 7.04126166 7.01475907 6.94234275 7.03997874 7.07964238
 6.92457004 7.1497328  6.49350149 6.86653389 6.99872059 6.77204844
 7.15901395 6.91221997 6.91718405 6.77929652 6.79085428 6.57473501
 6.96437745 7.036538   6.86639107 6.96507181 6.75799153 7.14634171
 6.958743   6.9619996  6.6524783  6.98023122 6.8109922  7.04877249
 7.16466817 6.99759031 6.99204446 6.9677859  6.98982657 6.71089056
 6.89393366 6.98948178 6.93938536 6.98929795 6.70565324 7.00876122
 7.03421525 6.90419835 7.21356112 7.15826591 7.00175394 7.0335086
 6.94627671 7.07431652 6.95678341 7.0245553  7.00021501 6.87081446
 6.95149086 6.74077589]
```

In [179…
```
plt.scatter(x=y_train,y=y_train_pred,alpha=0.3)
plt.plot()
```

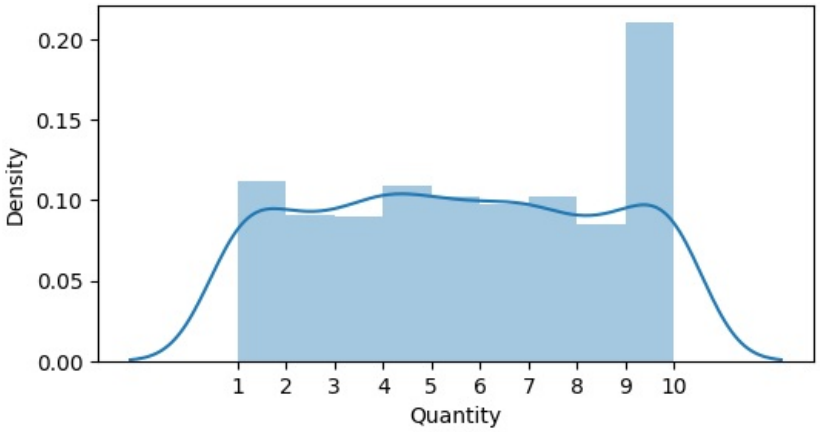
Out[179]: []



In [173…
```
xdata = [1,2,3,4,5,6,7,8,9,10]
plt.figure(figsize=(6,3))
sns.distplot(df['Quantity'])
plt.xticks(xdata)                          #visualization of how many customers buy the most
```

```
C:\Users\SWATHI\AppData\Local\Temp\ipykernel_8304\1367647900.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Quantity'])
```

Out[173]:
```
([<matplotlib.axis.XTick at 0x1def03b8f90>,
  <matplotlib.axis.XTick at 0x1def036d6d0>,
  <matplotlib.axis.XTick at 0x1def03e4d90>,
  <matplotlib.axis.XTick at 0x1def042f750>,
  <matplotlib.axis.XTick at 0x1def0439950>,
  <matplotlib.axis.XTick at 0x1def043bc90>,
  <matplotlib.axis.XTick at 0x1def043ded0>,
  <matplotlib.axis.XTick at 0x1def043e110>,
  <matplotlib.axis.XTick at 0x1def0444c90>,
  <matplotlib.axis.XTick at 0x1def0363250>],
 [Text(1, 0, '1'),
  Text(2, 0, '2'),
  Text(3, 0, '3'),
  Text(4, 0, '4'),
  Text(5, 0, '5'),
  Text(6, 0, '6'),
  Text(7, 0, '7'),
  Text(8, 0, '8'),
  Text(9, 0, '9'),
  Text(10, 0, '10')])
```



In [175…
```
quantity=pd.DataFrame(df['Quantity'].value_counts())   #for getting how much quantity sold the most
quantity
```

Out[175]:

Quantity	count
10	119
1	112
4	109
7	102
5	102
6	98
9	92
2	91
3	90
8	85

In [ ]: