

Importing Necessary Libraries

```
import pandas as pd
import numpy as np
```

C:\Users\aaakas\AppData\Local\Temp\ipykernel_16520\2162656668.py:1:

DeprecationWarning:

Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),

(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)

but was not found to be installed on your system.

If this would cause problems for you,

please provide us feedback at

<https://github.com/pandas-dev/pandas/issues/54466>

```
import pandas as pd
```

Reading Dataset

```
salary = pd.read_csv("assignment3-part-1-dataset1.csv")
```

```
#store = pd.read_csv("assignment3-part-1-dataset2.csv")
```

```
salary.head()
```

Unnamed: 0	Company Name	Job Title	Salaries
0	Mu Sigma	Data Scientist	105
1	IBM	Data Scientist	95
2	Tata Consultancy Services	Data Scientist	66
3	Impact Analytics	Data Scientist	40
4	Accenture	Data Scientist	32

	Location	Salary
0	Bangalore	648573.0
1	Bangalore	1191950.0
2	Bangalore	836874.0
3	Bangalore	669578.0
4	Bangalore	944110.0

Replacing ? with NaN

```
salary.replace("?", np.nan,inplace=True)
```

Checking Missing Values

```
salary.isnull().sum()
```

```
Unnamed: 0      0
Company Name    0
Job Title       0
Salaries Reported 0
Location        0
Salary          0
dtype: int64
```

Grouping dataset by Job Title

```
salary.groupby("Job Title")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001E13C1BB140>
```

```
salary.head()
```

	Unnamed: 0	Company Name	Job Title	Salaries Reported
0	0	Mu Sigma	Data Scientist	105
1	1	IBM	Data Scientist	95
2	2	Tata Consultancy Services	Data Scientist	66
3	3	Impact Analytics	Data Scientist	40
4	4	Accenture	Data Scientist	32

	Location	Salary
0	Bangalore	648573.0
1	Bangalore	1191950.0
2	Bangalore	836874.0
3	Bangalore	669578.0
4	Bangalore	944110.0

Finding Mean

```
mean = salary.groupby("Job Title")["Salary"].mean()
print(mean)
```

Job Title	
Associate Machine Learning Engineer	4.643720e+05
Data Analyst	6.164699e+05
Data Engineer	1.309051e+06
Data Science	3.649053e+05
Data Science Associate	1.203913e+06

Data Science Consultant	2.671464e+06
Data Science Lead	4.068310e+06
Data Science Manager	4.619021e+06
Data Scientist	1.411330e+06
Data Scientist - Trainee	6.105120e+05
Junior Data Scientist	5.963231e+05
Lead Data Scientist	1.852189e+06
Machine Learning Associate	2.951140e+05
Machine Learning Consultant	7.064010e+05
Machine Learning Data Analyst	3.613780e+05
Machine Learning Data Associate	2.758410e+05
Machine Learning Data Associate I	2.585960e+05
Machine Learning Data Associate II	3.832130e+05
Machine Learning Developer	5.811190e+05
Machine Learning Engineer	7.971884e+05
Machine Learning Scientist	1.701180e+05
Machine Learning Software Engineer	1.397347e+06
Senior Data Scientist	1.766130e+06
Senior Machine Learning Engineer	1.473436e+06
Software Engineer - Machine Learning	1.566780e+06

Name: Salary, dtype: float64

Finding median

```
median = salary.groupby("Job Title")["Salary"].median()
print(median)
```

Job Title	
Associate Machine Learning Engineer	464372.0
Data Analyst	508150.5
Data Engineer	792683.0
Data Science	240780.0
Data Science Associate	1203913.0
Data Science Consultant	2671464.0
Data Science Lead	4068310.0
Data Science Manager	4619021.0
Data Scientist	914480.0
Data Scientist - Trainee	610512.0
Junior Data Scientist	554963.0
Lead Data Scientist	1664364.0
Machine Learning Associate	295114.0
Machine Learning Consultant	706401.0
Machine Learning Data Analyst	361378.0
Machine Learning Data Associate	275841.0
Machine Learning Data Associate I	258596.0
Machine Learning Data Associate II	383213.0
Machine Learning Developer	581119.0
Machine Learning Engineer	627048.5
Machine Learning Scientist	170118.0
Machine Learning Software Engineer	1397347.0

Senior Data Scientist	1733388.0
Senior Machine Learning Engineer	1335445.0
Software Engineer - Machine Learning	1566780.0
Name: Salary, dtype: float64	

Finding mode

```
mode = salary.groupby("Job Title")["Salary"].apply(lambda
x:x.mode().iloc[0])
print(mode)
```

*# lambda x: This defines an anonymous function (lambda function) that takes one argument x.
In this context, x represents each group of salaries within each job title.*

*# x.mode(): Inside the lambda function, x is a Series object containing all the salaries within a specific job title group.
The mode() function is called on this Series object to compute the mode, i.e., the most frequently occurring value.*

*# .iloc[0]: After calculating the mode, .iloc[0] is used to retrieve the first value from the resulting Series.
This is necessary because the mode() function may return multiple values if there are ties for the most frequent value.
By selecting the first value, we ensure that only one mode value is returned.*

Job Title	
Associate Machine Learning Engineer	464372.0
Data Analyst	338792.0
Data Engineer	515940.0
Data Science	180000.0
Data Science Associate	1203913.0
Data Science Consultant	2671464.0
Data Science Lead	4068310.0
Data Science Manager	4619021.0
Data Scientist	600000.0
Data Scientist - Trainee	610512.0
Junior Data Scientist	616492.0
Lead Data Scientist	1520967.0
Machine Learning Associate	295114.0
Machine Learning Consultant	186475.0
Machine Learning Data Analyst	361378.0
Machine Learning Data Associate	275841.0
Machine Learning Data Associate I	258596.0
Machine Learning Data Associate II	383213.0
Machine Learning Developer	410952.0
Machine Learning Engineer	128988.0
Machine Learning Scientist	62160.0

Machine Learning Software Engineer	1397347.0
Senior Data Scientist	2474429.0
Senior Machine Learning Engineer	229416.0
Software Engineer - Machine Learning	1521236.0

Name: Salary, dtype: float64

Finding minimum value

```
minimum = salary.groupby("Job Title")["Salary"].min()
print(minimum)
```

Job Title	
Associate Machine Learning Engineer	464372.0
Data Analyst	10814.0
Data Engineer	33120.0
Data Science	60840.0
Data Science Associate	1203913.0
Data Science Consultant	2671464.0
Data Science Lead	4068310.0
Data Science Manager	4619021.0
Data Scientist	48000.0
Data Scientist - Trainee	610512.0
Junior Data Scientist	60840.0
Lead Data Scientist	1520967.0
Machine Learning Associate	295114.0
Machine Learning Consultant	186475.0
Machine Learning Data Analyst	361378.0
Machine Learning Data Associate	275841.0
Machine Learning Data Associate I	258596.0
Machine Learning Data Associate II	383213.0
Machine Learning Developer	410952.0
Machine Learning Engineer	21628.0
Machine Learning Scientist	62160.0
Machine Learning Software Engineer	1397347.0
Senior Data Scientist	324089.0
Senior Machine Learning Engineer	229416.0
Software Engineer - Machine Learning	1521236.0

Name: Salary, dtype: float64

Finding Maximum Values

```
maximum = salary.groupby("Job Title")["Salary"].max()
print(maximum)
```

Job Title	
Associate Machine Learning Engineer	4.643720e+05
Data Analyst	3.900962e+07
Data Engineer	1.190400e+08
Data Science	2.000000e+06
Data Science Associate	1.203913e+06

Data Science Consultant	2.671464e+06
Data Science Lead	4.068310e+06
Data Science Manager	4.619021e+06
Data Scientist	1.661404e+08
Data Scientist - Trainee	6.105120e+05
Junior Data Scientist	1.498750e+06
Lead Data Scientist	2.839138e+06
Machine Learning Associate	2.951140e+05
Machine Learning Consultant	1.226327e+06
Machine Learning Data Analyst	3.613780e+05
Machine Learning Data Associate	2.758410e+05
Machine Learning Data Associate I	2.585960e+05
Machine Learning Data Associate II	3.832130e+05
Machine Learning Developer	7.512860e+05
Machine Learning Engineer	6.518917e+06
Machine Learning Scientist	2.780760e+05
Machine Learning Software Engineer	1.397347e+06
Senior Data Scientist	3.654010e+06
Senior Machine Learning Engineer	3.110514e+06
Software Engineer - Machine Learning	1.612324e+06

Name: Salary, dtype: float64

Finding Standard Deviation

```
std = salary.groupby("Job Title")["Salary"].std()
std.replace(np.NaN, 0, inplace=True)
print(std)
```

Job Title	
Associate Machine Learning Engineer	0.000000e+00
Data Analyst	1.292116e+06
Data Engineer	6.009190e+06
Data Science	3.388020e+05
Data Science Associate	0.000000e+00
Data Science Consultant	0.000000e+00
Data Science Lead	0.000000e+00
Data Science Manager	0.000000e+00
Data Scientist	5.140558e+06
Data Scientist - Trainee	0.000000e+00
Junior Data Scientist	3.931792e+05
Lead Data Scientist	5.017356e+05
Machine Learning Associate	0.000000e+00
Machine Learning Consultant	7.352864e+05
Machine Learning Data Analyst	0.000000e+00
Machine Learning Data Associate	0.000000e+00
Machine Learning Data Associate I	0.000000e+00
Machine Learning Data Associate II	0.000000e+00
Machine Learning Developer	2.406525e+05
Machine Learning Engineer	7.047460e+05
Machine Learning Scientist	1.526757e+05

Machine Learning Software Engineer	0.000000e+00
Senior Data Scientist	7.833905e+05
Senior Machine Learning Engineer	9.506370e+05
Software Engineer - Machine Learning	6.440894e+04

Name: Salary, dtype: float64

Installing Libraries

```
%pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (3.8.3)Note: you may need to restart the kernel to use updated packages.
```

```
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (1.2.0)
```

```
Requirement already satisfied: cyclor>=0.10 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (0.12.1)
```

```
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (4.49.0)
```

```
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (1.4.5)
```

```
Requirement already satisfied: numpy<2,>=1.21 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (1.26.4)
```

```
Requirement already satisfied: packaging>=20.0 in c:\users\aakas\appdata\roaming\python\python312\site-packages (from matplotlib) (23.2)
```

```
Requirement already satisfied: pillow>=8 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (10.2.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (3.1.1)
```

```
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aakas\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (2.8.2)
```

```
Requirement already satisfied: six>=1.5 in c:\users\aakas\appdata\roaming\python\python312\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

Importing Libraries

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
C:\Users\aakas\AppData\Local\Temp\ipykernel_10312\3311980270.py:1:
```

```
DeprecationWarning:
```

```
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
```


(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries) but was not found to be installed on your system. If this would cause problems for you, please provide us feedback at <https://github.com/pandas-dev/pandas/issues/54466>

```
import pandas as pd

virus = pd.read_csv("assignment3-part2.csv")

virus.replace("?", np.nan, inplace=True)

virus.isna().sum()

Unnamed: 0      0
ANTI_A         0
ANTI_B         0
V1             0
dtype: int64
```

Binning

```
num_bin = 5

label = ["A", "B", "C", "D", "E"]

virus["E-W-Partition"] = pd.cut(virus["V1"], bins=num_bin,
labels=label)
virus["E-F-Partition"] = pd.qcut(virus["V1"], q = num_bin,
labels=label)

virus.iloc[:]
```

	Unnamed: 0	ANTI_A	ANTI_B	V1	E-W-Partition	E-F-Partition
0	0	2.77	98.1	6.0	A	A
1	1	9.79	16.8	6.7	A	A
2	2	9.29	13.6	7.5	A	A
3	3	3.41	1.6	7.6	A	A
4	4	9.83	55.3	8.4	A	A
5	5	5.76	94.4	9.3	A	A
6	6	2.73	67.2	9.4	A	A
7	7	0.17	41.9	9.6	A	A
8	8	7.50	22.7	10.6	A	B
9	9	0.21	39.6	11.2	A	B
10	10	0.66	61.6	11.3	A	B
11	11	2.84	78.9	11.5	A	B
12	12	7.90	15.5	11.5	A	B
13	13	2.00	93.3	11.9	A	B
14	14	5.07	87.9	13.0	A	B
15	15	2.14	85.0	13.8	A	B

16	16	9.94	16.4	14.1	A	C
17	17	7.21	20.6	14.9	A	C
18	18	4.41	4.8	15.2	A	C
19	19	8.70	48.3	15.8	A	C
20	20	3.05	86.3	16.5	A	C
21	21	9.34	52.6	17.7	A	C
22	22	7.36	13.0	21.4	A	C
23	23	7.79	61.4	55.5	C	C
24	24	4.24	53.3	60.0	D	D
25	25	0.63	11.3	61.4	D	D
26	26	1.43	26.2	62.1	D	D
27	27	1.44	32.9	64.9	D	D
28	28	7.34	83.4	64.9	D	D
29	29	1.74	12.7	69.0	D	D
30	30	6.12	35.4	70.1	D	D
31	31	7.59	76.7	73.4	D	D
32	32	7.31	58.4	74.9	E	E
33	33	7.17	86.9	76.8	E	E
34	34	2.99	20.6	77.0	E	E
35	35	2.84	33.6	77.9	E	E
36	36	0.67	9.2	80.1	E	E
37	37	8.62	78.9	81.2	E	E
38	38	7.44	98.8	83.8	E	E
39	39	7.21	88.0	90.7	E	E

Dividing Dataset with V1 less than and greater than 40

threshold = 40

```
virus_low = virus[virus["V1"] < threshold]
virus_high = virus[virus["V1"] > threshold]
```

```
virus_low.describe()
```

	Unnamed: 0	ANTI_A	ANTI_B	V1
count	23.000000	23.000000	23.000000	23.000000
mean	11.000000	5.307826	48.495652	11.952174
std	6.78233	3.363449	32.804302	3.830840
min	0.000000	0.170000	1.600000	6.000000
25%	5.500000	2.750000	16.600000	9.350000
50%	11.000000	5.070000	48.300000	11.500000
75%	16.500000	8.300000	81.950000	14.500000
max	22.000000	9.940000	98.100000	21.400000

```
virus_high.describe()
```

	Unnamed: 0	ANTI_A	ANTI_B	V1
count	17.000000	17.000000	17.000000	17.000000
mean	31.000000	4.857059	51.041176	71.982353
std	5.049752	2.937230	30.416526	9.593112

min	23.000000	0.630000	9.200000	55.500000
25%	27.000000	1.740000	26.200000	64.900000
50%	31.000000	6.120000	53.300000	73.400000
75%	35.000000	7.340000	78.900000	77.900000
max	39.000000	8.620000	98.800000	90.700000

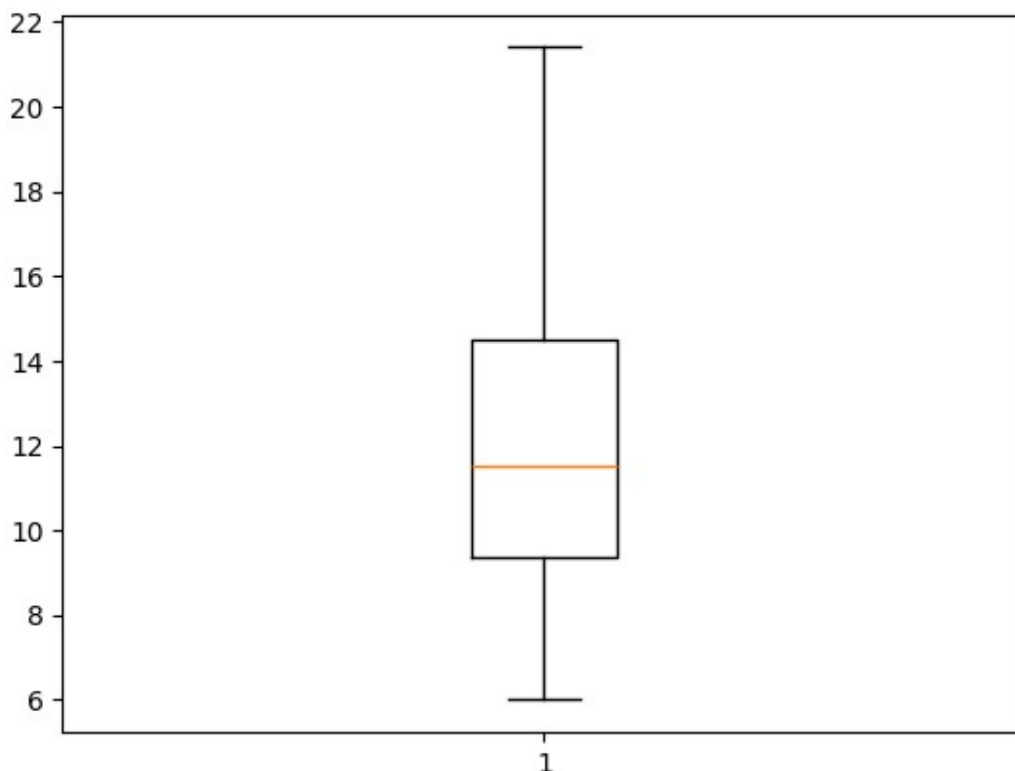
Plotting Boxplot from V1 for both low and high groups

```
fig = plt.figure(figsize= (10,5))
```

<Figure size 1000x500 with 0 Axes>

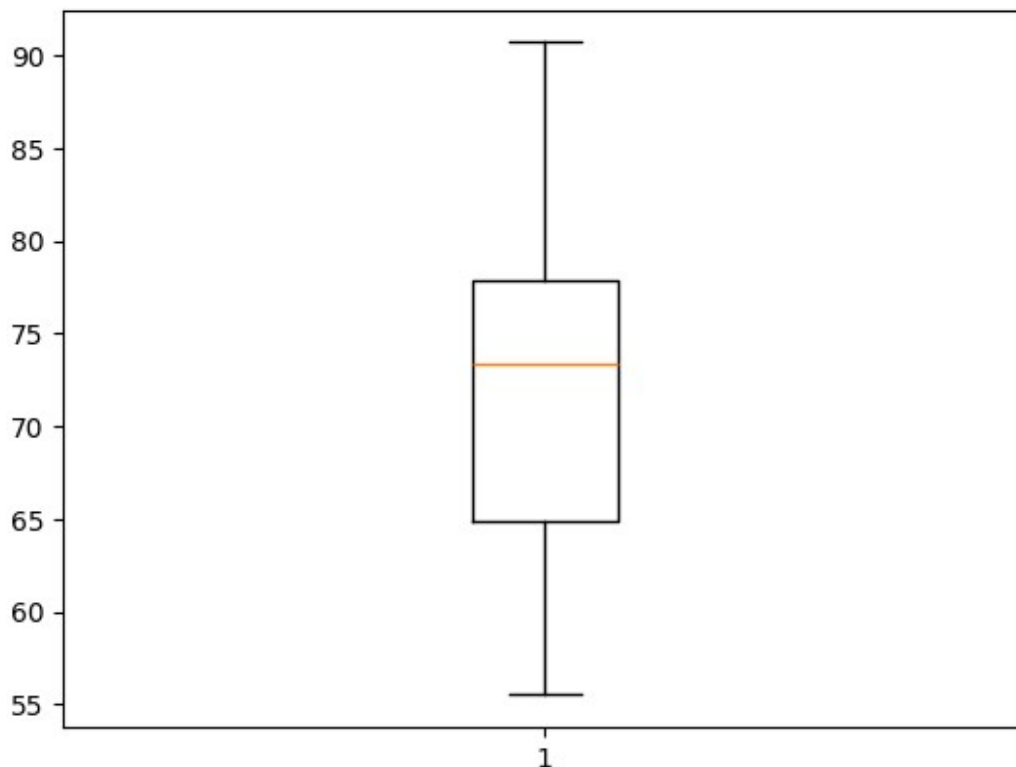
```
plt.boxplot(virus_low["V1"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1593850e540>,
<matplotlib.lines.Line2D at 0x159385cc440>],
'caps': [<matplotlib.lines.Line2D at 0x159385cc740>,
<matplotlib.lines.Line2D at 0x159385cca70>],
'boxes': [<matplotlib.lines.Line2D at 0x1593858ff80>],
'medians': [<matplotlib.lines.Line2D at 0x159385ccd40>],
'fliers': [<matplotlib.lines.Line2D at 0x159385cd040>],
'means': []}
```



```
plt.boxplot(virus_high["V1"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1593a8901a0>,
<matplotlib.lines.Line2D at 0x1593a8904d0>],
'caps': [<matplotlib.lines.Line2D at 0x1593a8907d0>,
<matplotlib.lines.Line2D at 0x1593a890ad0>],
'boxes': [<matplotlib.lines.Line2D at 0x1593a84bef0>],
'medians': [<matplotlib.lines.Line2D at 0x1593a890dd0>],
'fliers': [<matplotlib.lines.Line2D at 0x1593a891040>],
'means': []}
```



```
virus_low["ANTI_A"] = (virus_low["ANTI_A"] -
virus_low["ANTI_A"].min())/(virus_low["ANTI_A"].max() -
virus_low["ANTI_A"].min())
virus_low["ANTI_B"] = (virus_low["ANTI_B"] -
virus_low["ANTI_B"].min())/(virus_low["ANTI_B"].max() -
virus_low["ANTI_B"].min())
virus_low
```

C:\Users\aaakas\AppData\Local\Temp\ipykernel_10312\3142245604.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```

virus_low["ANTI_A"] = (virus_low["ANTI_A"] -
virus_low["ANTI_A"].min())/(virus_low["ANTI_A"].max() -
virus_low["ANTI_A"].min())
C:\Users\alakas\AppData\Local\Temp\ipykernel_10312\3142245604.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

```

See the caveats in the documentation:

https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```

virus_low["ANTI_B"] = (virus_low["ANTI_B"] -
virus_low["ANTI_B"].min())/(virus_low["ANTI_B"].max() -
virus_low["ANTI_B"].min())

```

	Unnamed: 0	ANTI_A	ANTI_B	V1	E-W-Partition	E-F-Partition
0	0	0.266121	1.000000	6.0	A	A
1	1	0.984647	0.157513	6.7	A	A
2	2	0.933470	0.124352	7.5	A	A
3	3	0.331627	0.000000	7.6	A	A
4	4	0.988741	0.556477	8.4	A	A
5	5	0.572160	0.961658	9.3	A	A
6	6	0.262027	0.679793	9.4	A	A
7	7	0.000000	0.417617	9.6	A	A
8	8	0.750256	0.218653	10.6	A	B
9	9	0.004094	0.393782	11.2	A	B
10	10	0.050154	0.621762	11.3	A	B
11	11	0.273286	0.801036	11.5	A	B
12	12	0.791198	0.144041	11.5	A	B
13	13	0.187308	0.950259	11.9	A	B
14	14	0.501535	0.894301	13.0	A	B
15	15	0.201638	0.864249	13.8	A	B
16	16	1.000000	0.153368	14.1	A	C
17	17	0.720573	0.196891	14.9	A	C
18	18	0.433982	0.033161	15.2	A	C
19	19	0.873081	0.483938	15.8	A	C
20	20	0.294780	0.877720	16.5	A	C
21	21	0.938588	0.528497	17.7	A	C
22	22	0.735926	0.118135	21.4	A	C

Min-Max Normalization

```

virus_high["ANTI_A"] = (virus_high["ANTI_A"] -
virus_high["ANTI_A"].min())/(virus_high["ANTI_A"].max() -
virus_high["ANTI_A"].min())
virus_high["ANTI_B"] = (virus_high["ANTI_B"] -
virus_high["ANTI_B"].min())/(virus_high["ANTI_B"].max() -
virus_high["ANTI_B"].min())
virus_high

```

```

C:\Users\akas\AppData\Local\Temp\ipykernel_10312\34499113.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
    virus_high["ANTI_A"] = (virus_high["ANTI_A"] -
virus_high["ANTI_A"].min())/(virus_high["ANTI_A"].max() -
virus_high["ANTI_A"].min())
C:\Users\akas\AppData\Local\Temp\ipykernel_10312\34499113.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
    virus_high["ANTI_B"] = (virus_high["ANTI_B"] -
virus_high["ANTI_B"].min())/(virus_high["ANTI_B"].max() -
virus_high["ANTI_B"].min())

```

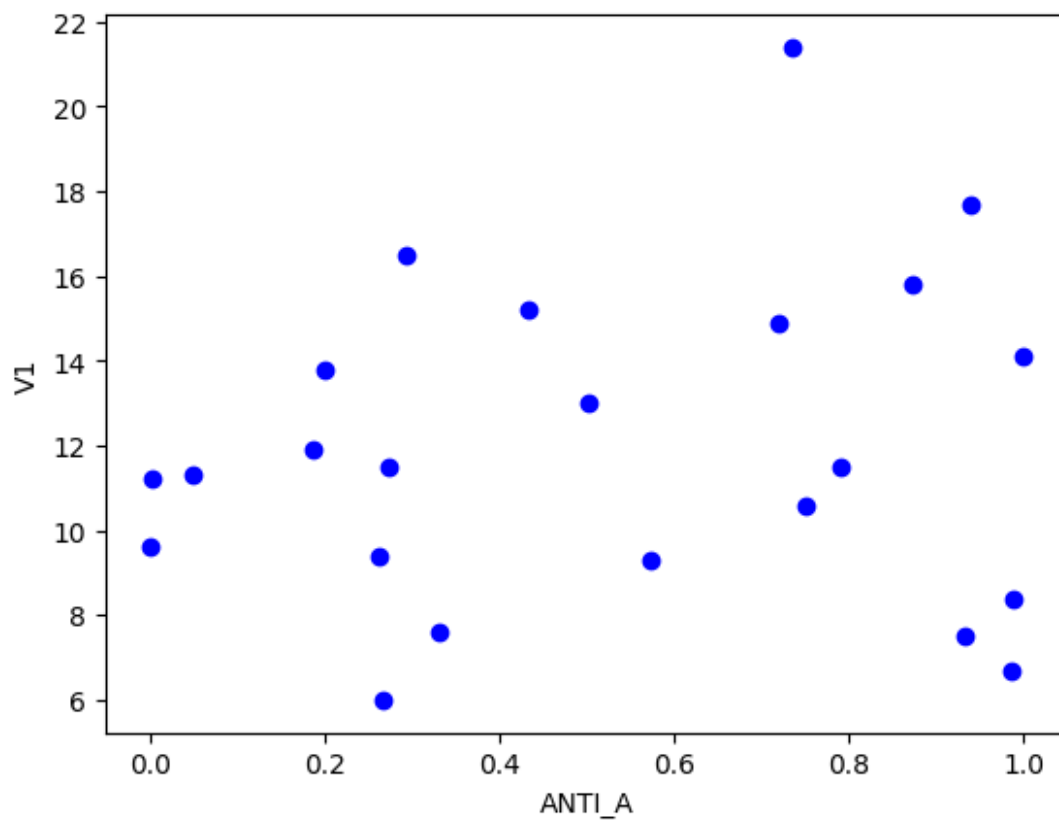
	Unnamed: 0	ANTI_A	ANTI_B	V1	E-W-Partition	E-F-Partition
23	23	0.896120	0.582589	55.5	C	C
24	24	0.451815	0.492187	60.0	D	D
25	25	0.000000	0.023438	61.4	D	D
26	26	0.100125	0.189732	62.1	D	D
27	27	0.101377	0.264509	64.9	D	D
28	28	0.839800	0.828125	64.9	D	D
29	29	0.138924	0.039062	69.0	D	D
30	30	0.687109	0.292411	70.1	D	D
31	31	0.871089	0.753348	73.4	D	D
32	32	0.836045	0.549107	74.9	E	E
33	33	0.818523	0.867188	76.8	E	E
34	34	0.295369	0.127232	77.0	E	E
35	35	0.276596	0.272321	77.9	E	E
36	36	0.005006	0.000000	80.1	E	E
37	37	1.000000	0.777902	81.2	E	E
38	38	0.852315	1.000000	83.8	E	E
39	39	0.823529	0.879464	90.7	E	E

Plotting Scatterplot

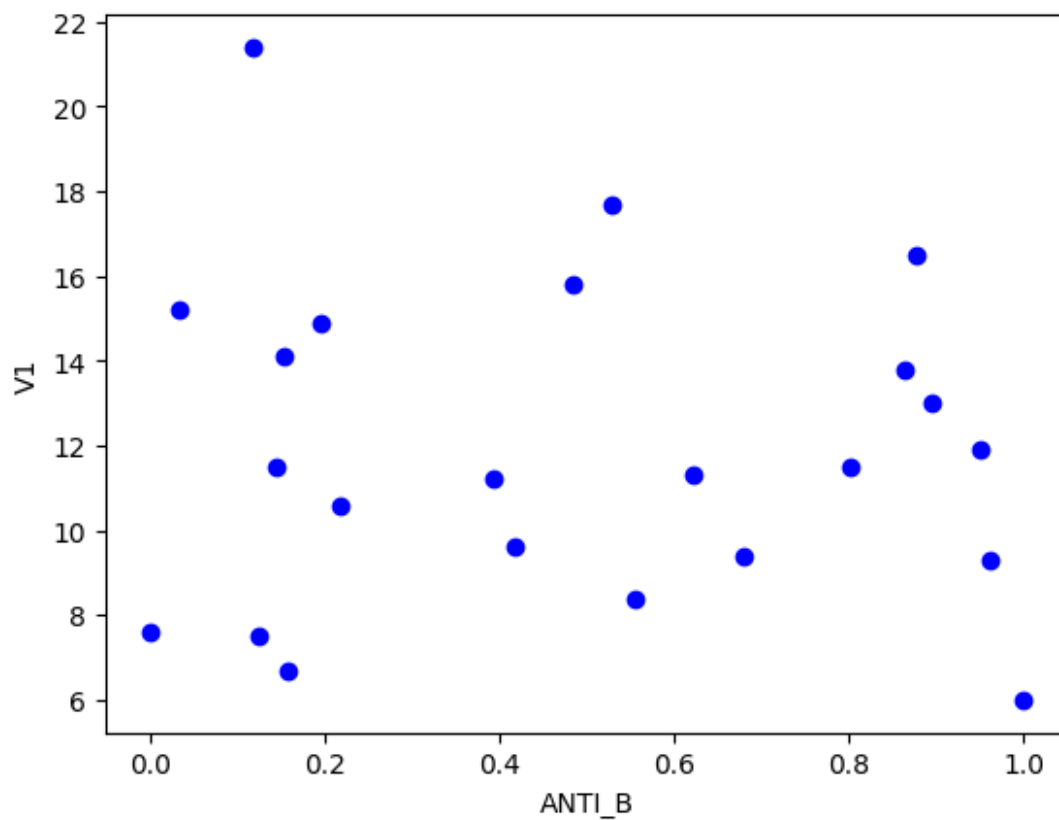
```

plt.scatter(virus_low["ANTI_A"], virus_low["V1"], c="blue")
plt.xlabel("ANTI_A")
plt.ylabel("V1")
plt.show()

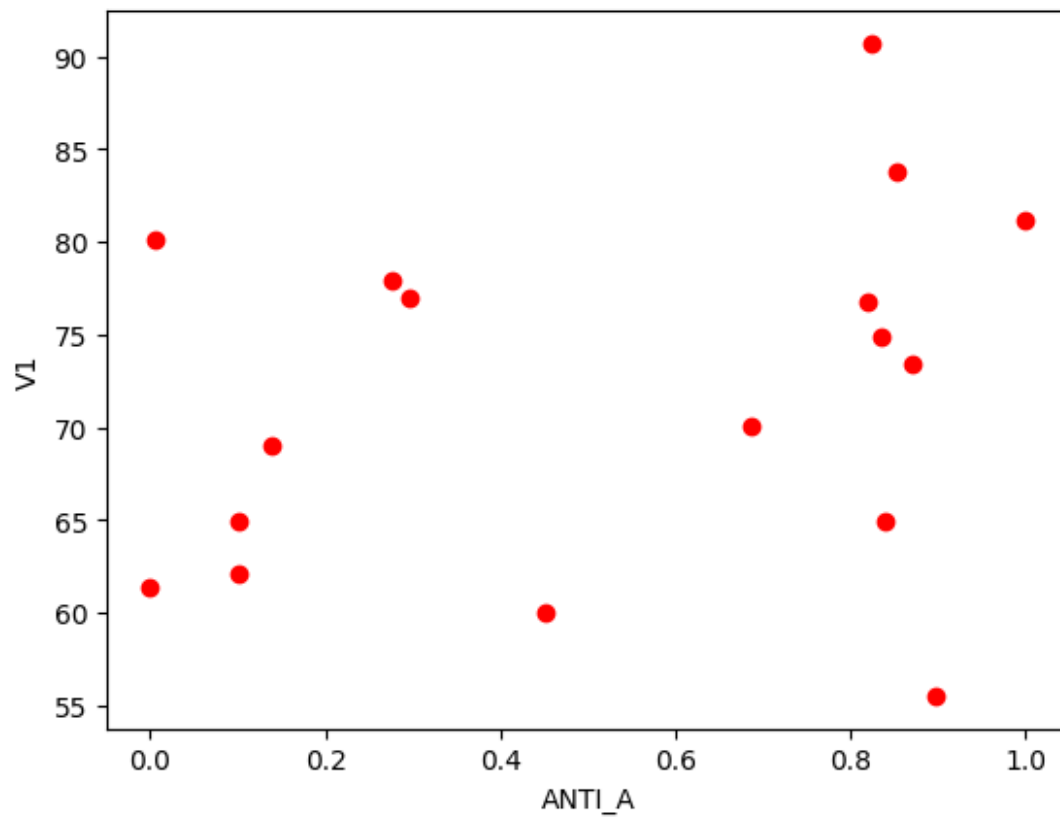
```



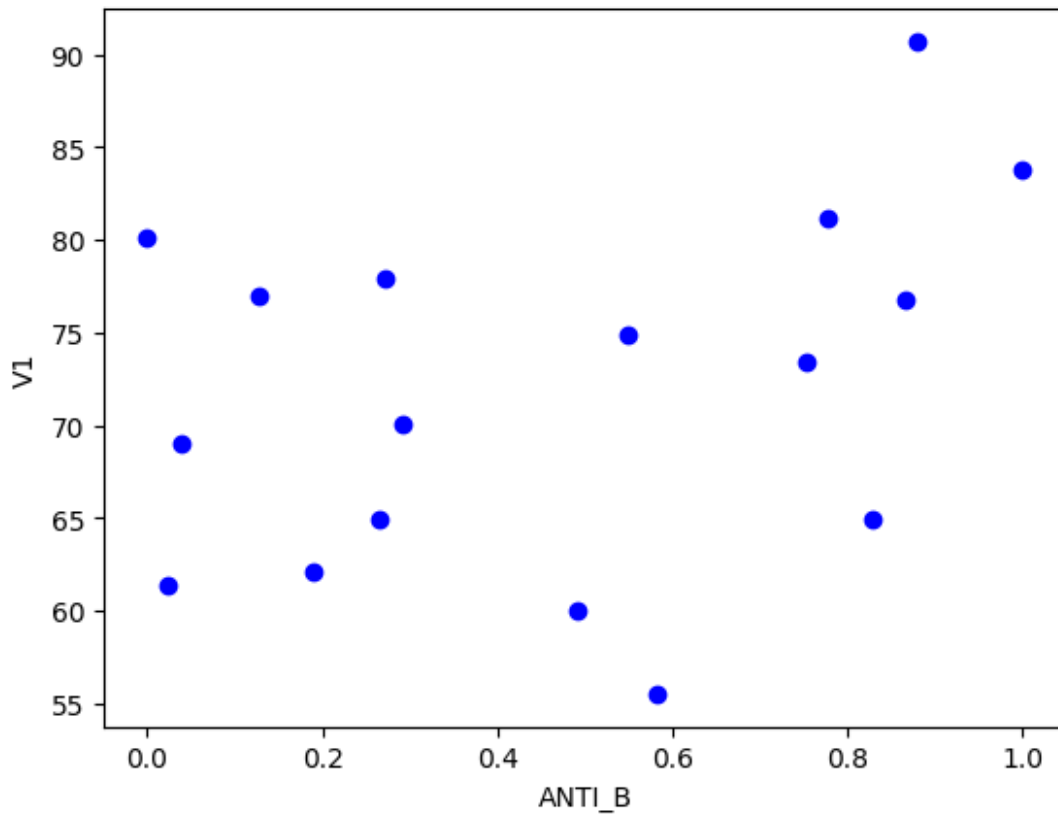
```
plt.scatter(virus_low["ANTI_B"], virus_low["V1"], c="blue")  
plt.xlabel("ANTI_B")  
plt.ylabel("V1")  
plt.show()
```



```
plt.scatter(virus_high["ANTI_A"], virus_high["V1"], c="red")
plt.xlabel("ANTI_A")
plt.ylabel("V1")
plt.show()
```

```
plt.scatter(virus_high["ANTI_B"], virus_high["V1"], c="blue")  
plt.xlabel("ANTI_B")  
plt.ylabel("V1")  
plt.show()
```



Corelation between ANTI_A and ANTI_B

```
virus_low["ANTI_A"].corr(virus_low["ANTI_B"])
```

```
-0.44962747254599306
```

```
virus_high["ANTI_A"].corr(virus_high["ANTI_B"])
```

```
0.8938242533758364
```

For low values, we must use only one of the medicines i.e ANTI_A or ANTI_B as the corelation is negative but for high values, we can use both ANTI_A and ANTI_B together as the corelation is positive