Data Visualization

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

seaborn:- efficient with large dataset seaborn all graphs High quality images, built on matplotlib and pandas

```
==================================
scatter plot
Box plot
Heatmap
Histogram
Density plot
Factor plot
Swarm plot
Violin plot
overlaying plots
Joint distribution plot
Bar plot


====================================

Matplotlib:- simple graph
seaborn:- high end graph and simple code

Dependency of seaborn:- numpy, pandas, scipy, matplotlib

pip install seaborn
conda install seaborn

========================================
```

In [ ]:
```

```

In [2]:
```
# !pip3 install seaborn
```

In [3]:
```
1 print(dir(sns))
```

```
['FacetGrid', 'JointGrid', 'PairGrid', '__builtins__', '__cached__', '__d
oc__', '__file__', '__loader__', '__name__', '__package__', '__path__',
'__spec__', '__version__', '_compat', '_core', '_decorators', '_docstring
s', '_oldcore', '_orig_rc_params', '_statistics', '_stats', 'algorithms',
'axes_style', 'axisgrid', 'barplot', 'blend_palette', 'boxenplot', 'boxpl
ot', 'categorical', 'catplot', 'choose_colorbrewer_palette', 'choose_cube
helix_palette', 'choose_dark_palette', 'choose_diverging_palette', 'choos
e_light_palette', 'clustermap', 'cm', 'color_palette', 'colors', 'countpl
ot', 'crayon_palette', 'crayons', 'cubehelix_palette', 'dark_palette', 'd
esaturate', 'despine', 'displot', 'distplot', 'distributions', 'diverging
_palette', 'dogplot', 'ecdfplot', 'external', 'get_data_home', 'get_datas
et_names', 'heatmap', 'histplot', 'hls_palette', 'husl_palette', 'jointpl
ot', 'kdeplot', 'light_palette', 'lineplot', 'lmplot', 'load_dataset', 'm
atrix', 'miscplot', 'move_legend', 'mpl', 'mpl_palette', 'pairplot', 'pal
ettes', 'palplot', 'plotting_context', 'pointplot', 'rcmod', 'regplot',
'regression', 'relational', 'relplot', 'reset_defaults', 'reset_orig', 'r
esidplot', 'rugplot', 'saturate', 'scatterplot', 'set', 'set_color_code
s', 'set_context', 'set_hls_values', 'set_palette', 'set_style', 'set_the
me', 'stripplot', 'swarmplot', 'utils', 'violinplot', 'widgets', 'xkcd_pa
lette', 'xkcd_rgb']
```

In [4]:
```
1 sns.get_dataset_names()
```

Out[4]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [5]:
```
1 sns.load_dataset('glue')
```

Out[5]:

	Model	Year	Encoder	Task	Score
0	ERNIE	2019	Transformer	CoLA	75.5
1	T5	2019	Transformer	CoLA	71.6
2	RoBERTa	2019	Transformer	CoLA	67.8
3	BERT	2018	Transformer	CoLA	60.5
4	BiLSTM+ELMo	2018	LSTM	CoLA	32.1
...	...	...	...	...	...
59	BERT	2018	Transformer	RTE	70.1
60	BiLSTM+ELMo	2018	LSTM	RTE	57.4
61	BiLSTM+CoVe	2017	LSTM	RTE	52.7
62	BiLSTM+Attn	2017	LSTM	RTE	58.4
63	BiLSTM	2017	LSTM	RTE	57.4

64 rows × 5 columns

In [19]:
```
1 data = pd.read_csv('https://github.com/mwaskom/seaborn-data/blob/maste
2 data
```

Out[19]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [ ]:
```
1
```

In [8]:
```
1 data = pd.read_csv('https://github.com/mwaskom/seaborn-data/blob/maste
```

In [9]:
```
1 data.sample(5)
```

Out[9]:

	total_bill	tip	sex	smoker	day	time	size
219	30.14	3.09	Female	Yes	Sat	Dinner	4
102	44.30	2.50	Female	Yes	Sat	Dinner	3
187	30.46	2.00	Male	Yes	Sun	Dinner	5
236	12.60	1.00	Male	Yes	Sat	Dinner	2
209	12.76	2.23	Female	Yes	Sat	Dinner	2

In [10]:
```
1 data['time'].unique()
```

Out[10]:
```
array(['Dinner', 'Lunch'], dtype=object)
```

In [11]:
```
1 # sns.lineplot()
```

In [ ]:
```
1
```

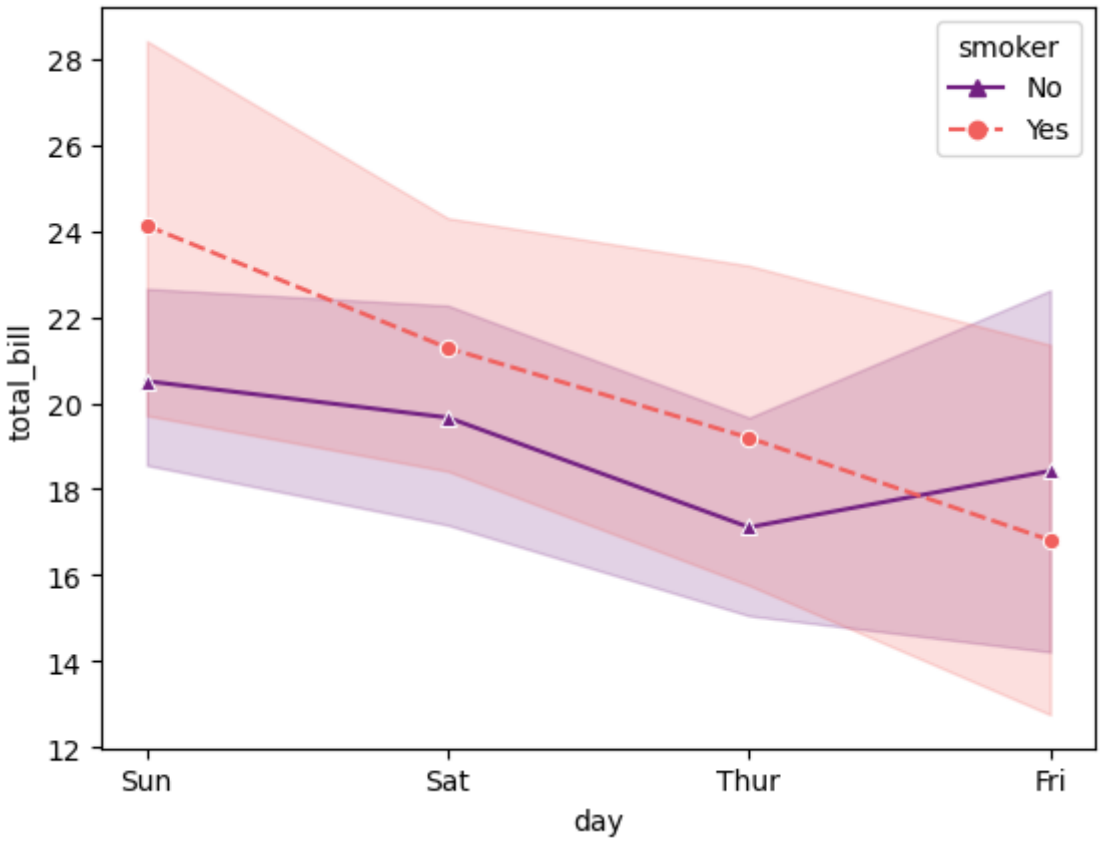
In [ ]:
```
1
```

In [8]:
```
1  # sns.lineplot(x='key name'
2  #               y = 'key name'
3  #               data = 'dataset name'
4  #               hue = 'dataset col name',
5  #              size = int
6  #              style = 'hue col name'
7  #              palette = 'color combination name ex:- Blues',
8  #              markers = list or single value ex:- ['o', ' *'],
9  #              dashes = bool,
10 #              legend = bool or ['auto','brief','full'],
11 #              )
```

In [12]:
```
sns.lineplot(data = data,
             x = 'day',
             y = 'total_bill',
             hue = 'smoker',
             # size = 'size',
             style = 'smoker',
             palette='magma',
             markers=['^','o'],
             dashes=True,
             legend='brief')

```

Out[12]: <Axes: xlabel='day', ylabel='total_bill'>



In [ ]:
```

```

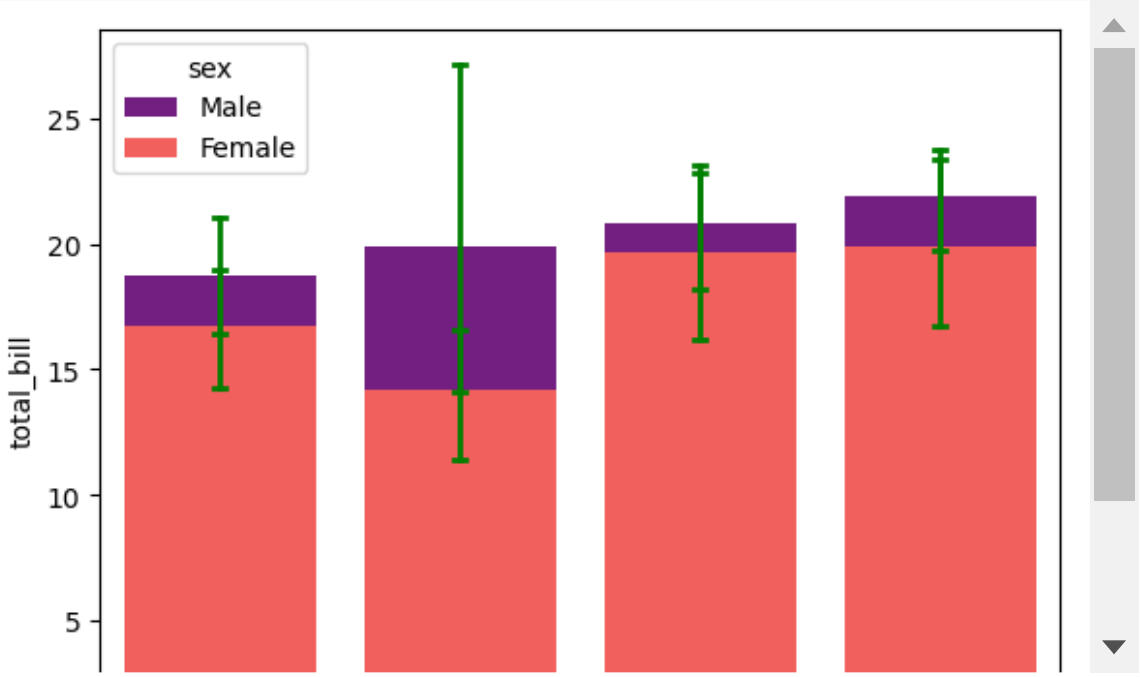
In [ ]:
```

```

Bar Plot

In [86]:

```
# sns.set(style = 'darkgrid')
# column name, dataset.colname,dataset['colname']
# :confidence interval = int(0 to 100)
# \'v or h', x and y must be numerical
# single or list value
# color bar accent or more
# same as alpha', int or float value
# 'error bar color','yellow or y'
# 'error bar capping' float value
sns.barplot(y = data.total_bill,
            x = data.day,
            hue = 'sex',
            data = data,
            order = ['Thur','Fri','Sat','Sun'],
            hue_order = ['Male','Female'],
#             ci = 95,
            n_boot = 95,
            orient = 'v',
#             color = 'y',
            palette = 'magma',
            saturation = 90,
            errcolor = 'g',
            errwidth = 2,
            capsize = 0.05,
            dodge = False,
            alpha = 1
            )
plt.show()
```



In [34]:
```
1 data[(data['sex']=='Male') & (data['day']=='Fri')].describe()
```

Out[34]:

	total_bill	tip	size
count	10.000000	10.000000	10.000000
mean	19.857000	2.693000	2.100000
std	10.015847	1.136428	0.737865
min	8.580000	1.500000	1.000000
25%	12.235000	1.665000	2.000000
50%	17.215000	2.600000	2.000000
75%	26.082500	3.375000	2.000000
max	40.170000	4.730000	4.000000

In [67]:
```
1 data.describe()
```

Out[67]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

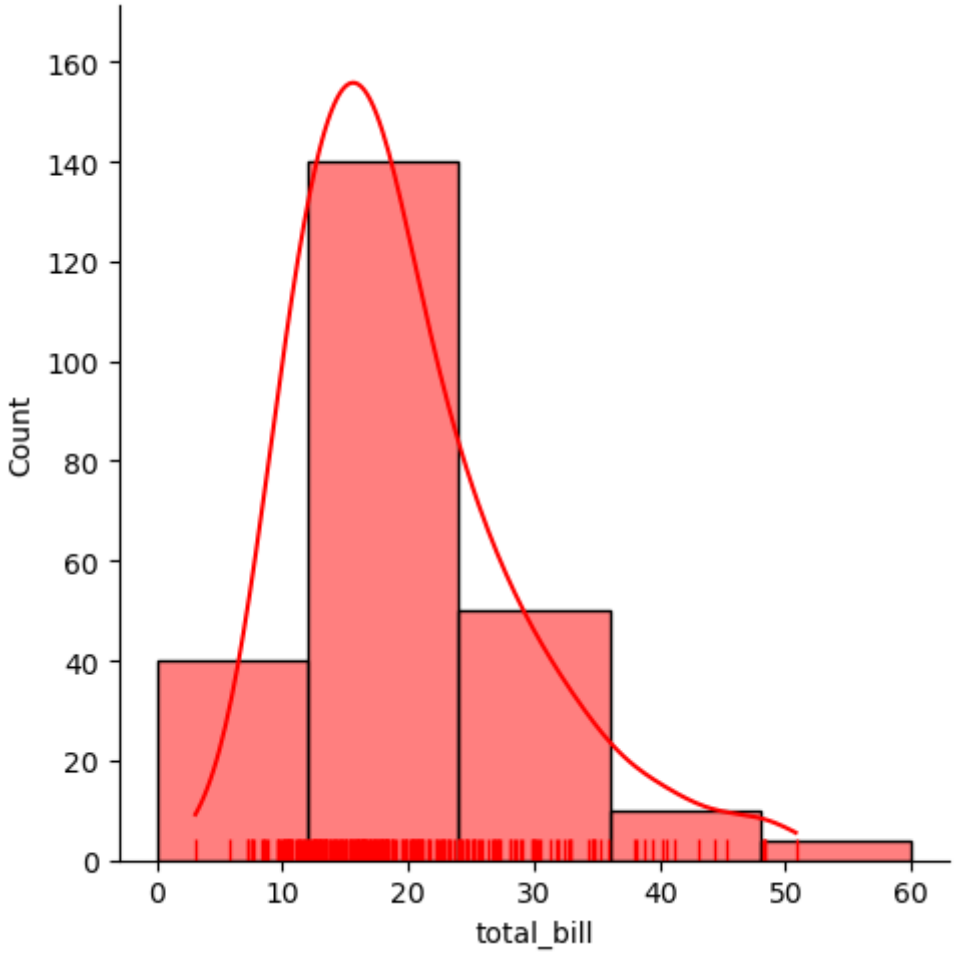
In [ ]:
```
1
```

In [ ]:
```
1
```

Histogram or dis plot

In [109]:
```
#  kernal desnsity estimator,
#  list,ex:- [120,140,170,190,210]
sns.displot(data['total_bill'],
            bins = [0,12,24,36,48,60],
            kde=True,
            rug=True,
            color='r',
#             log_scale =False,
#              hue = 'sex'
            )
```

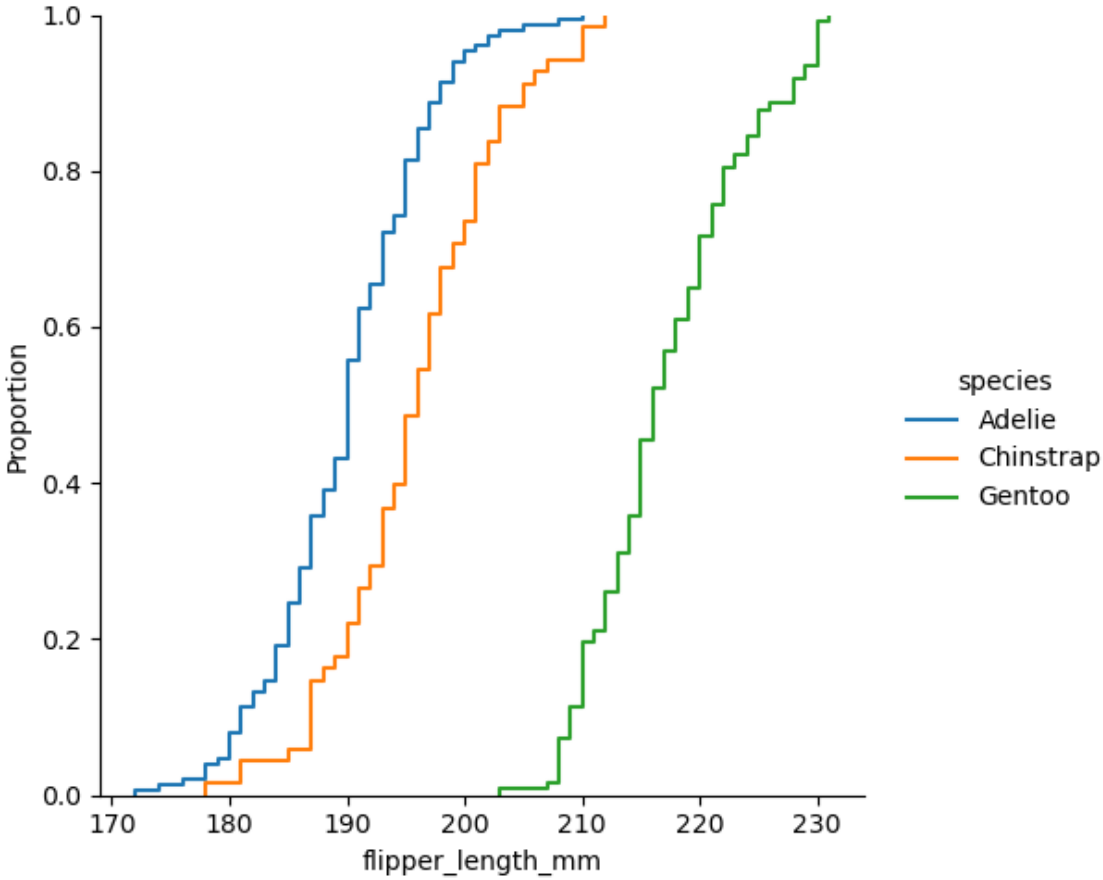
Out[109]: <seaborn.axisgrid.FacetGrid at 0x25e30e2c550>



In [111]:
```
data = sns.load_dataset('penguins')
```

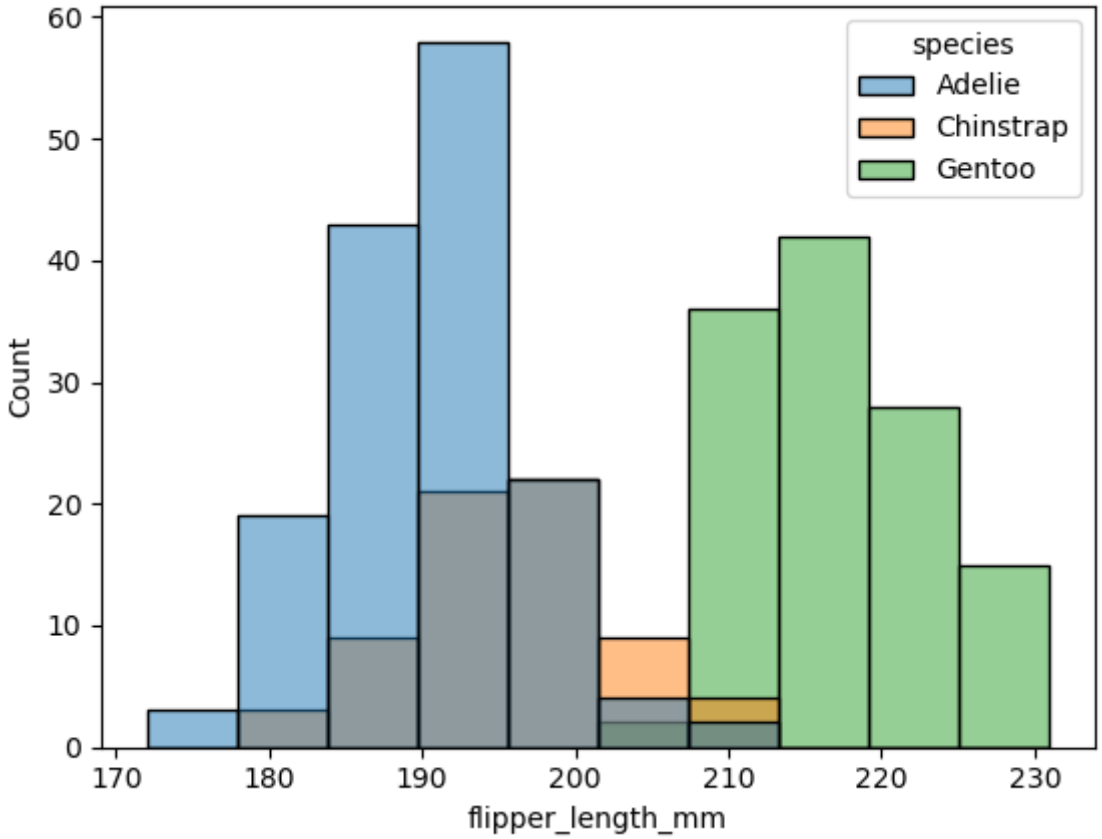
In [119]:
```
1 sns.displot(data=data, x="flipper_length_mm", hue="species",kind='ecdf
```

Out[119]: <seaborn.axisgrid.FacetGrid at 0x25e325cef90>



In [120]:
```
1 sns.histplot(data=data, x="flipper_length_mm", hue="species")
```

Out[120]:
```
<Axes: xlabel='flipper_length_mm', ylabel='Count'>
```



In [ ]:
```
1
```

In [1]:
```
1 #
2 import matplotlib.pyplot as plt
3 import seaborn as sns
```

Scatter plot

In [16]:
```
1 from sklearn.datasets import load_iris
2 data = load_iris()
3 data = data['data']
4 data[:,0]
```

Out[16]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8,
       4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. ,
       5. , 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4,
       5.1, 5. , 4.5, 4.4, 5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. , 7. , 6.4,
       6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5. , 5.9, 6. , 6.1, 5.6,
       6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7,
       6. , 5.7, 5.5, 5.5, 5.8, 6. , 5.4, 6. , 6.7, 6.3, 5.6, 5.5, 5.5,
       6.1, 5.8, 5. , 5.6, 5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8, 7.1, 6.3,
       6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5,
       7.7, 7.7, 6. , 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2,
       7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6. , 6.9, 6.7, 6.9, 5.8,
       6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9])
```

In [5]:
```
1 data['feature_names']
```

Out[5]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [32]:
```
1 import pandas as pd
2
3 df = pd.DataFrame(data)
4 df.columns = load_iris()['feature_names']
5 df['target'] = load_iris()['target']
6 df
```

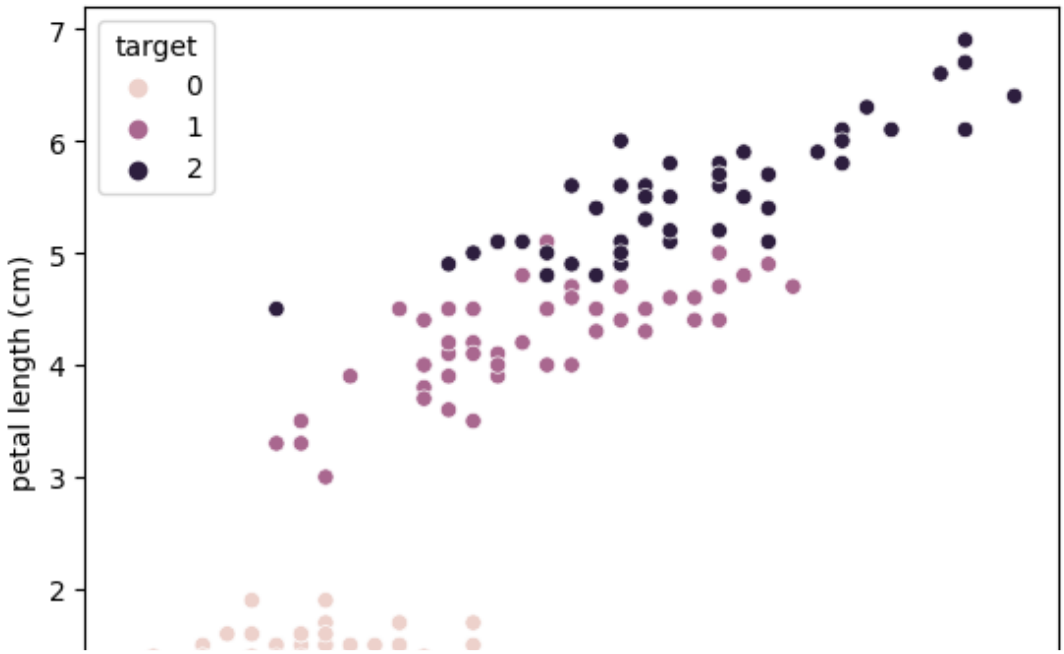
Out[32]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

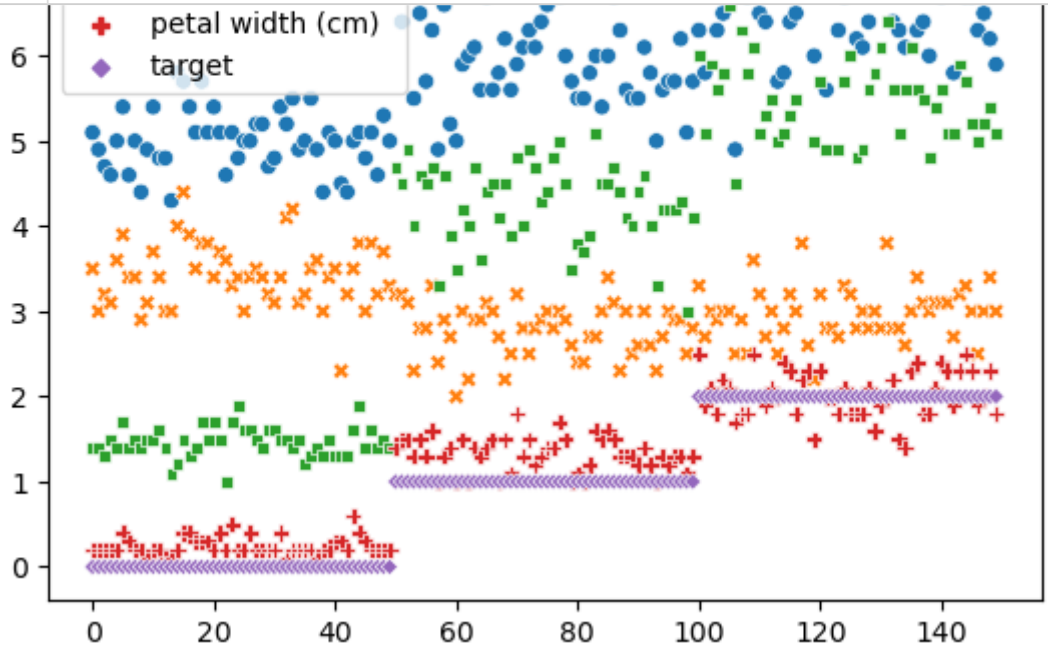
In [38]:

```
# # numerical vs numerical analysis
sns.scatterplot(x = 'sepal length (cm)',
                y = 'petal length (cm)',
                data = df,
                hue = 'target'
#                 style = 'col name, mostly hue col name'
#                 size = 'col name, to control scatter size'
#                 sizes = tuple(int,int), to provide size manually
#                 palette = 'color gradient, ex:- blues'
#                 alpha = float,
#                 markers = {'colname':'o','colname':'*'}
                )
plt.xlabel('sepal length (cm)')
plt.ylabel('petal length (cm)')
plt.show()
```



In [35]:

```
sns.scatterplot(df)
```



In [ ]:

```
1
```

In [40]:

```
1 tips = sns.load_dataset('tips')
2 tips
```

Out[40]:

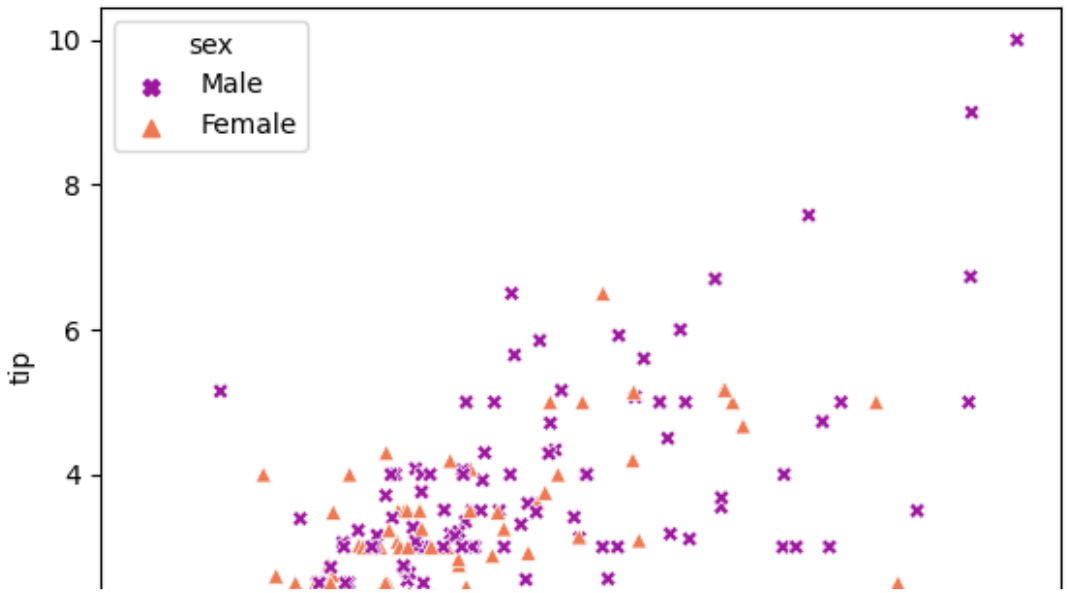
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [73]:
```
style = 'col name, mostly hue col name'
#                 size = 'col name, to control scatter size'
#                 sizes = tuple(int,int), to provide size manually
#                 palette = 'color gradient, ex:- blues'
#                 alpha = float,
#                 markers = {'colname':'o','colname':'*'
# {"Female": "s", "Male": "X"}


sns.scatterplot(data = tips,
                x = 'total_bill',
                y = 'tip',
                hue = 'sex',
                 style = 'sex',
                markers = ['X','^'],
                palette='plasma')
```

Out[73]:
```
<Axes: xlabel='total_bill', ylabel='tip'>
```



In [ ]:
```

```

Heat map

In [75]:
```
df.corr()
```

Out[75]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
sepal length (cm)	1.000000	-0.117570	0.871754	0.817941	0.782561
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.366126	-0.426658
petal length (cm)	0.871754	-0.428440	1.000000	0.962865	0.949035
petal width (cm)	0.817941	-0.366126	0.962865	1.000000	0.956547
target	0.782561	-0.426658	0.949035	0.956547	1.000000

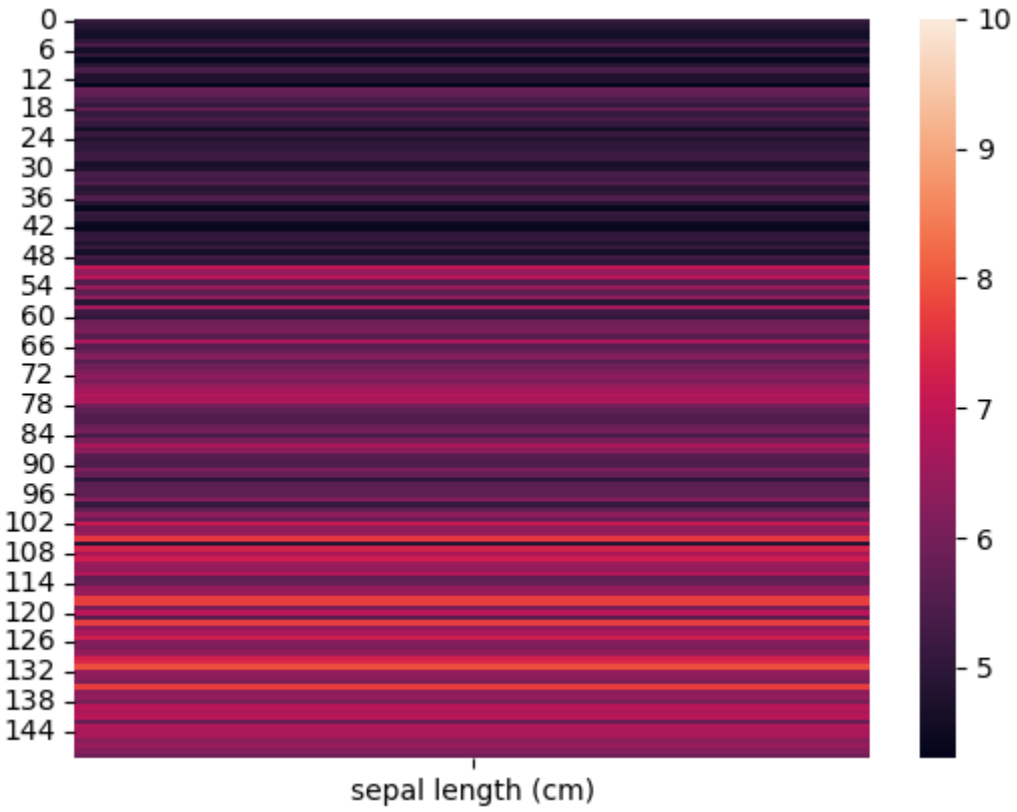
In [ ]:

```
1
```

In [86]:

```
# # corr, table, or pivot
# table,corr,pivot,pivot table
# int, color bar minimum value
# int, color bar max value,
graph = sns.heatmap(df[['sepal length (cm)']],
#             vmin = 0,
            vmax = 10
#             cmap: 'string ex:- rainbow',
#             annot: bool
#             annot_kws = dict(),{'fontsize':10,'color':'g'}
#             linewidth: int:- to make line gap
#             linecolor = 'y',
#             fmt = 's': to show string value in place of annot
#             cbar:- bool>> to show to hide colorbar
#             xticklabels:bool , to show or hide
#             yticklabels:bool, to show or hide
                    )

# graph.set(xlabel = 'str',
#           ylabel = 'str')
# sns.set(font_scale : int)
```



In [13]:

```
1 path = r"C:\Users\Lenovo\Downloads\T20-GL-gainers-NIFTY-10-Jan-2024.cs
```

In [14]:

```
df = pd.read_csv(path)
df
```

Out[14]:

	Symbol	Open	High	Low	Prev. Close	LTP	%chng	Volume	Va
0	HCLTECH	1474.90	1508.00	1466.05	1460.40	1494.45	2.33	2748207	4.093894e
1	ADANIENT	3024.00	3112.00	3020.05	3014.60	3084.70	2.33	2375345	7.295658e
2	CIPLA	1296.00	1319.55	1294.30	1289.35	1316.00	2.07	1901033	2.490809e
3	ADANIPORTS	1220.00	1220.00	1197.00	1197.10	1210.90	1.15	4819089	5.818231e
4	ICICIBANK	974.30	992.30	974.30	979.75	990.60	1.11	4442522	4.385969e
5	INDUSINDBK	1620.00	1640.00	1620.00	1625.10	1636.80	0.72	565074	9.225116e
6	RELIANCE	2577.00	2596.50	2575.05	2580.50	2594.05	0.53	1892649	4.899046e
7	SBILIFE	1454.95	1495.35	1451.65	1449.35	1456.95	0.52	1862104	2.756119e
8	TATASTEEL	133.80	134.35	132.10	133.65	134.35	0.52	14788964	1.972552e
9	TECHM	1236.50	1246.45	1228.30	1235.65	1241.50	0.47	328945	4.075398e
10	ASIANPAINT	3292.00	3292.00	3258.20	3267.50	3282.75	0.47	381310	1.247440e
11	TCS	3690.00	3729.25	3688.00	3689.90	3706.65	0.45	677855	2.515249e
12	HDFCLIFE	642.00	650.35	642.00	640.90	643.00	0.33	2303332	1.488482e
13	TATAMOTORS	800.00	804.45	792.65	799.80	801.90	0.26	3827504	3.055114e
14	JSWSTEEL	823.85	825.40	811.20	821.40	823.40	0.24	661030	5.416744e
15	HDFCBANK	1643.00	1656.00	1641.40	1650.50	1654.25	0.23	4153853	6.856391e
16	TITAN	3708.00	3730.00	3690.00	3694.00	3700.70	0.18	280159	1.039297e
17	BHARTIARTL	1066.95	1073.30	1055.30	1064.50	1064.95	0.04	1496632	1.590770e

	Symbol	Open	High	Low	Prev. Close	LTP	%chng	Volume	Va
18	NESTLEIND	2590.00	2622.40	2583.65	2592.60	2593.10	0.02	406467	1.056863e

In [15]:
```
fdf = df[['Symbol','%chng']]
fdf['%chng']
```

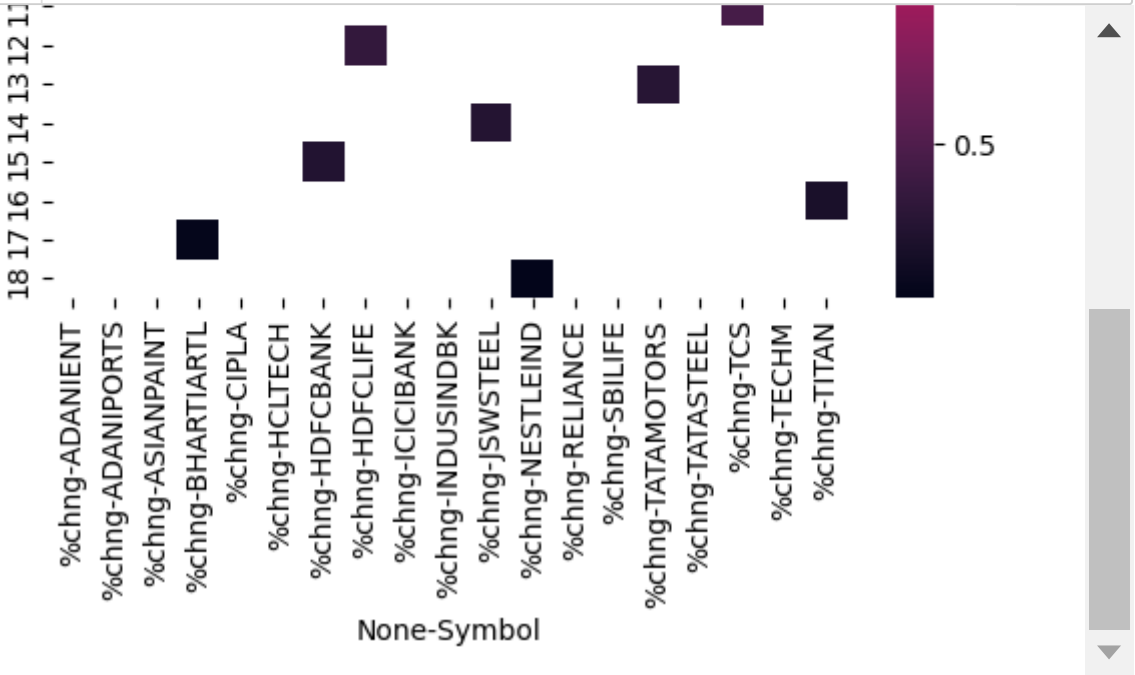
Out[15]:
```
0     2.33
1     2.33
2     2.07
3     1.15
4     1.11
5     0.72
6     0.53
7     0.52
8     0.52
9     0.47
10    0.47
11    0.45
12    0.33
13    0.26
14    0.24
15    0.23
16    0.18
17    0.04
18    0.02
Name: %chng, dtype: float64
```

In [16]:
```
1 fdf.pivot(columns='Symbol')
```

Out[16]:

Symbol	ADANIENT	ADANIPORTS	ASIANPAINT	BHARTIARTL	CIPLA	HCLTECH	HDFCBAN
0	NaN	NaN	NaN	NaN	NaN	2.33	Na
1	2.33	NaN	NaN	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	2.07	NaN	Na
3	NaN	1.15	NaN	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	Na
5	NaN	NaN	NaN	NaN	NaN	NaN	Na
6	NaN	NaN	NaN	NaN	NaN	NaN	Na
7	NaN	NaN	NaN	NaN	NaN	NaN	Na
8	NaN	NaN	NaN	NaN	NaN	NaN	Na
9	NaN	NaN	NaN	NaN	NaN	NaN	Na
10	NaN	NaN	0.47	NaN	NaN	NaN	Na
11	NaN	NaN	NaN	NaN	NaN	NaN	Na
12	NaN	NaN	NaN	NaN	NaN	NaN	Na
13	NaN	NaN	NaN	NaN	NaN	NaN	Na
14	NaN	NaN	NaN	NaN	NaN	NaN	Na
15	NaN	NaN	NaN	NaN	NaN	NaN	0.2
16	NaN	NaN	NaN	NaN	NaN	NaN	Na
17	NaN	NaN	NaN	0.04	NaN	NaN	Na
18	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [17]:
```
1 sns.heatmap(fdf.pivot(columns='Symbol'))
```



In [ ]:
```
1
```

In [18]:
```
1 fdf
```

Out[18]:

	Symbol	%chng
0	HCLTECH	2.33
1	ADANIENT	2.33
2	CIPLA	2.07
3	ADANIPORTS	1.15
4	ICICIBANK	1.11
5	INDUSINDBK	0.72
6	RELIANCE	0.53
7	SBILIFE	0.52
8	TATASTEEL	0.52
9	TECHM	0.47
10	ASIANPAINT	0.47
11	TCS	0.45
12	HDFCLIFE	0.33
13	TATAMOTORS	0.26
14	JSWSTEEL	0.24
15	HDFCBANK	0.23
16	TITAN	0.18
17	BHARTIARTL	0.04
18	NESTLEIND	0.02

In [21]:
```
1 sns.heatmap(fdf[['%chng']],annot=fdf[['Symbol']],fmt='s')
2 plt.show()
```

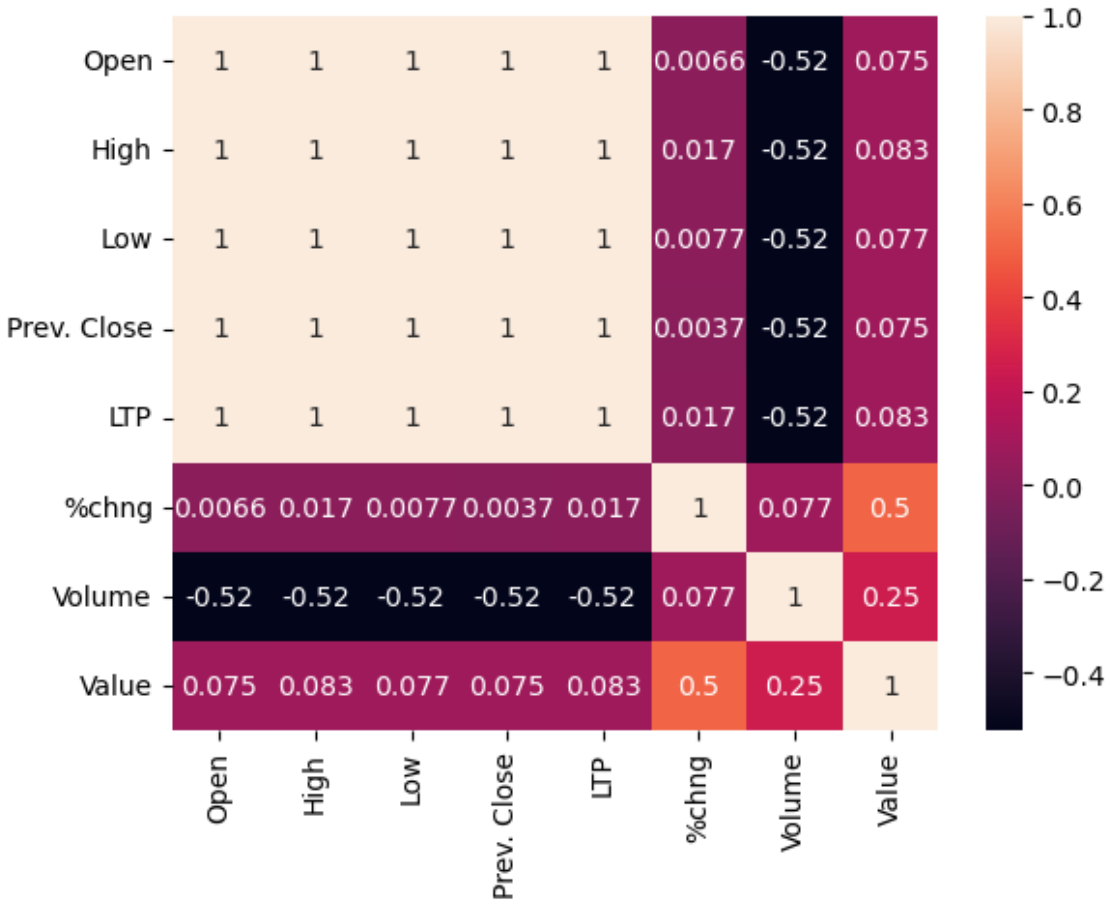
In [32]:
```
df.iloc[:,1:-1].corr()
```

Out[32]:

	Open	High	Low	Prev. Close	LTP	%chng	Volume	Value
Open	1.000000	0.999856	0.999968	0.999967	0.999895	0.006646	-0.523727	0.075163
High	0.999856	1.000000	0.999904	0.999847	0.999960	0.017192	-0.523202	0.082604
Low	0.999968	0.999904	1.000000	0.999987	0.999936	0.007666	-0.523231	0.076789
Prev. Close	0.999967	0.999847	0.999987	1.000000	0.999897	0.003712	-0.524097	0.074967
LTP	0.999895	0.999960	0.999936	0.999897	1.000000	0.016869	-0.522772	0.083115
%chng	0.006646	0.017192	0.007666	0.003712	0.016869	1.000000	0.076622	0.502129
Volume	-0.523727	-0.523202	-0.523231	-0.524097	-0.522772	0.076622	1.000000	0.248373
Value	0.075163	0.082604	0.076789	0.074967	0.083115	0.502129	0.248373	1.000000

In [34]:
```
sns.heatmap(df.iloc[:,1:-1].corr(),annot=True)
```

Out[34]: <Axes: >

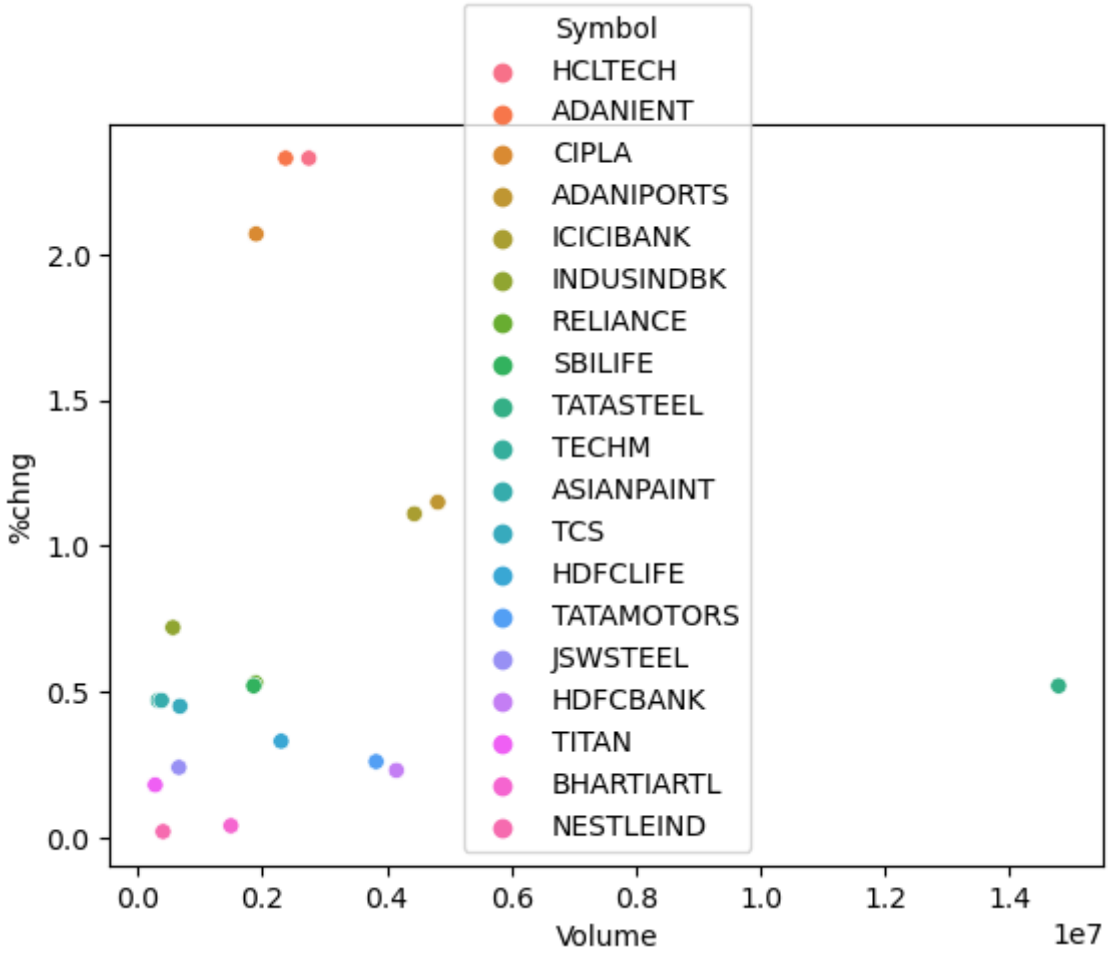


In [ ]:
```
import requests as r
```

In [20]:
```
# url = '''https://www.nseindia.com/market-data/top-gainers-losers'''
# r.get(url)
```

In [40]:
```
1 sns.scatterplot(y=df['%chng'],x=df['Volume'],hue='Symbol',data=df)
```

Out[40]:
```
<Axes: xlabel='Volume', ylabel='%chng'>
```



In [99]:
```
1 options = pd.read_csv(r"C:\Users\Lenovo\Downloads\MW-FO-nifty_bank_opt
```

In [100]:
```
1 options.columns = [i.strip() for i in options.columns]
2 options.columns
```

Out[100]:
```
Index(['INSTRUMENT TYPE', 'SYMBOL', 'EXPIRY DATE', 'OPTION TYPE', 'STRIK
E',
       'LTP', 'CHNG', '%CHNG', 'OPEN', 'HIGH', 'LOW', 'VOLUME \n(Contract
s)',
       'VALUE \n (₹ Crores)', 'OPEN INTEREST', 'UNDERLYING VALUE'],
      dtype='object')
```

In [101]:
```
1 options
```

Out[101]:

	INSTRUMENT TYPE	SYMBOL	EXPIRY DATE	OPTION TYPE	STRIKE	LTP	CHNG	%CHNG	OPE
0	Index Options	BANKNIFTY	10-Jan-2024	Call	47,300.00	50.90	-66.85	-56.77	59.0
1	Index Options	BANKNIFTY	10-Jan-2024	Put	47,200.00	38.10	-130.80	-77.44	200.7
2	Index Options	BANKNIFTY	10-Jan-2024	Put	47,000.00	17.00	-68.20	-80.05	106.3
3	Index Options	BANKNIFTY	10-Jan-2024	Call	47,400.00	23.65	-59.20	-71.45	38.1
4	Index Options	BANKNIFTY	10-Jan-2024	Call	47,500.00	13.20	-43.60	-76.76	24.0
...	...	...	...	...	...	...	...	...	.
1409	Index Options	BANKNIFTY	28-Mar-2024	Call	47,800.00	-	-	-	
1410	Index Options	BANKNIFTY	29-Feb-2024	Put	43,900.00	-	-	-	
1411	Index Options	BANKNIFTY	07-Feb-2024	Put	39,500.00	-	-	-	
1412	Index Options	BANKNIFTY	26-Dec-2024	Put	49,500.00	-	-	-	
1413	Index Options	BANKNIFTY	07-Feb-2024	Call	44,600.00	-	-	-	

1414 rows × 15 columns

In [102]:
```
1 options = options[options['EXPIRY DATE'] =='10-Jan-2024']
```

In [103]:
```
options['STRIKE'] = options['STRIKE'].apply(lambda x: str(x))

options
```

2	Index Options	BANKNIFTY	10-Jan-2024	Put	47,000.00	17.00	-68.20	-80.05	1
3	Index Options	BANKNIFTY	10-Jan-2024	Call	47,400.00	23.65	-59.20	-71.45	
4	Index Options	BANKNIFTY	10-Jan-2024	Call	47,500.00	13.20	-43.60	-76.76	
...	...	...	...	...	...	...	...	...	
1152	Index Options	BANKNIFTY	10-Jan-2024	Put	53,000.00	-	-	-	
1281	Index Options	BANKNIFTY	10-Jan-2024	Put	54,000.00	-	-	-	
1301	Index Options	BANKNIFTY	10-Jan-2024	Put	52,000.00	-	-	-	
1374	Index Options	BANKNIFTY	10-Jan-2024	Call	40,500.00	-	-	-	
1378	Index Options	BANKNIFTY	10-Jan-	Put	55,000.00				

In [104]:
```
options['STRIKE'] = options['STRIKE'].astype('str')
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14900\3193261658.py:1: Setti
ngWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  options['STRIKE'] = options['STRIKE'].astype('str')
```

In [105]:
```
options['STRIKE']
```

Out[105]:
```
0       47,300.00
1       47,200.00
2       47,000.00
3       47,400.00
4       47,500.00
          ...
1152    53,000.00
1281    54,000.00
1301    52,000.00
1374    40,500.00
1378    55,000.00
Name: STRIKE, Length: 180, dtype: object
```

In [106]:
```
options['VALUE \n (₹ Crores)'] = options['VALUE \n (₹ Crores)'].apply(
options['VALUE \n (₹ Crores)']
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_14900\3480862592.py:1: Se
ttingWithCopyWarning:
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
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  options['VALUE \n (₹ Crores)'] = options['VALUE \n (₹ Crores)'].appl
y(lambda x: float(x.replace(',','')) if '-' not in x else 0)
```

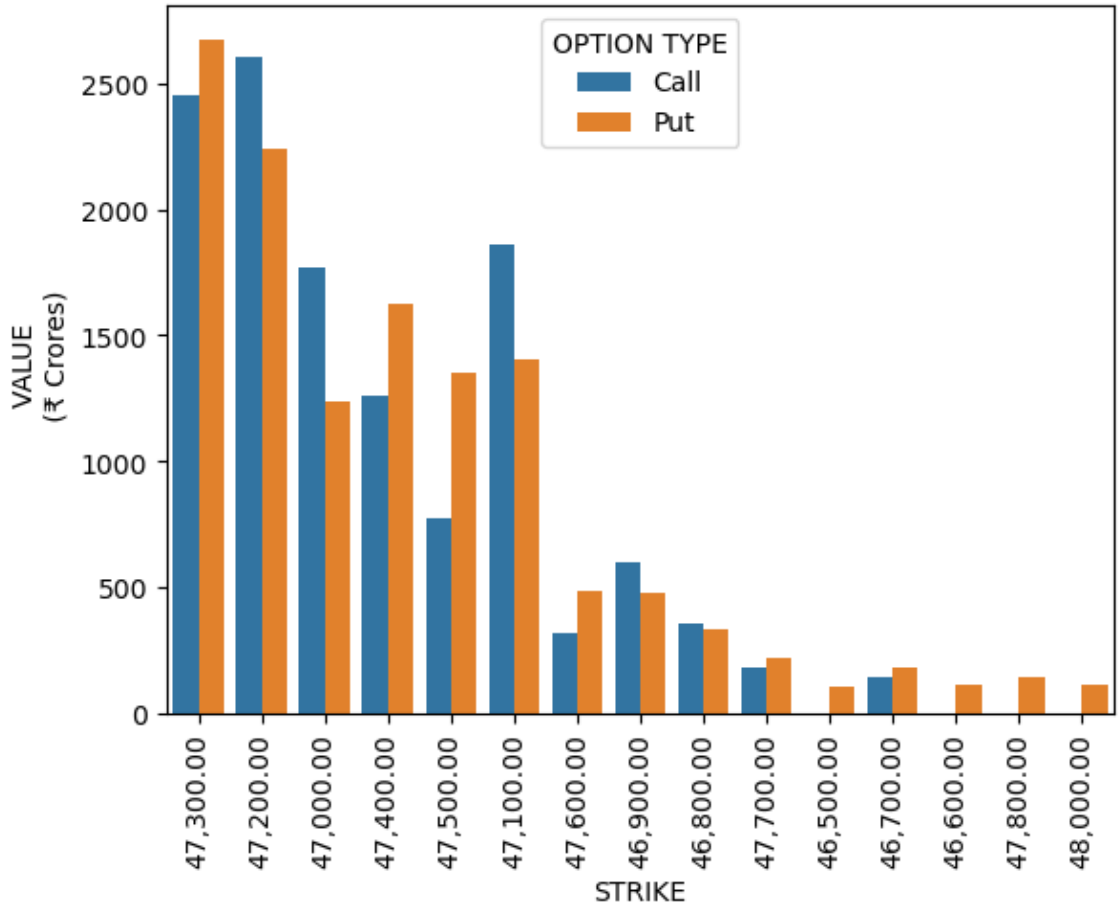
Out[106]:
```
0       2456.98
1       2244.43
2       1234.47
3       1257.50
4        773.35
         ...
1152       0.00
1281       0.00
```

In [107]:
```
options[options['VALUE \n (₹ Crores)']>0]['VALUE \n (₹ Crores)']
```

Out[107]:
```
0      2456.98
1      2244.43
2      1234.47
3      1257.50
4       773.35
        ...
730       0.03
749       0.01
750       0.01
784       0.01
829       0.01
Name: VALUE \n (₹ Crores), Length: 169, dtype: float64
```

In [113]:

```
sns.barplot(x = options[options['VALUE \n (₹ Crores)']>100]['STRIKE'],
            y = options[options['VALUE \n (₹ Crores)']>100]['VALUE \n
           hue='OPTION TYPE',
           data = options)
plt.xticks(rotation=90)
plt.show()
```



In [109]:

```
options[options['STRIKE'] =='47,300.00']
```

Out[109]:

	INSTRUMENT TYPE	SYMBOL	EXPIRY DATE	OPTION TYPE	STRIKE	LTP	CHNG	%CHNG	OPEN
0	Index Options	BANKNIFTY	10-Jan-2024	Call	47,300.00	50.90	-66.85	-56.77	59.05
6	Index Options	BANKNIFTY	10-Jan-2024	Put	47,300.00	70.40	-155.00	-68.77	202.00

count plot

In [114]:
```
# sns.countplot(x = 'col name'
#               data = dataset
#               hue = 'col name'
#               palette = 'color name or r'
#               color = 'r'
#               saturation:float 0<to>1
#               )
```

In [ ]:
```

```

violin Plot

In [115]:
```
# # similar as box and whisker plot
# # shows distribution of quantitavie data across severals levels

# # tips.csv
# # more the width more the distribuition
# sns.violinplot(x = 'day'
#                y = 'total_bill'
#                data = dataset
#                hue = 'time'
#                linewidth:int
#                palette = 'blues')

```

In [116]:
```
# sns.violinplot(x = 'time'
#                y = 'total_bill'
#                data = dataset
#                order = ['Dinner','Lunch']
#                saturation:float <0 to 1>
#                color = 'r')
```

In [117]:
```
# sns.violinplot(x = 'time'
#                y = 'total_bill'
#              hue = 'sex'
#                split:bool True
#                scale = 'count','area','width'
#                #to make horizontal swap x and y
#               inner: 'quart',{'box','quartile','point,'stick},'None' #
#                )
```

In [118]:
```
# single violin plot
# sns.violinplot(x = dataset['colname'])
```

In [119]:
```
# single violin plot
# sns.violinplot(y = dataset['colname'])
```

In [ ]:
```

```