Participant

Title= "PhD Student"

Name= "Munazza Usmani"

email = "munazza.usmani@unitn.it"

whatsapp = "00923316777441"

Python ka chilla with baba Ammar

Topics (Covered till now)

1-lecture & practice

- Basis python
- Indexing
- Basic Structure

2- Plotting

- Line and bar plot
- Box plot
- plot customization
- relplot
- scatter plot
- violin plot
- Linera regression plot
- swarmplot
- strip plot
- Ploty (animation plot & heat map etc.)

3- Jupyter Notebook in VS code

4- Markdown

5- Github in VS code

6- Numpy

7- Pandas

8- Exploratory data analysis

9- Many more (Github tools 101, Machine learning 101, Twitter & Linkdln, Data statistics, FAO data analysis)

In [ ]:
```
#Basic Python
print(2+3)
print("hello world")
print("we are learning python")
```

```
5
hello world
we are learning python
```

In [ ]:
```
#Operators
print(2+3)
print(6//3)
print(13%2)
print(2**3)
```

```
5
2
1
8
```

In [ ]:
```
#Strings
print("hello world")
print("we are learning python")
print('Test for single quotes')
print("Test for double quotes")
print('''Test for Triple quotes''')
print ("what's up")
```

```
hello world
we are learning python
Test for single quotes
Test for double quotes
Test for Triple quotes
what's up
```

Indexing

make a string

In [ ]:
```
a= "samosa pakora"
a[0]
```

Out[ ]:
```
's'
```

In [ ]:
```
#length of string
len(a)
```

Out[ ]:
```
13
```

In [ ]:
```
a[0:6] #last number exclusive
```

Out[ ]:
```
'samosa'
```

In [ ]:
```
a[-2]
```

Out[ ]:
```
'r'
```

In [ ]:
```
a.upper() #upper case
```

Out[ ]:
```
'SAMOSA PAKORA'
```

In [ ]:
```
a.replace("s","sh") #replacing words
a.count("a")
```

Out[ ]:
```
4
```

In [ ]:
```
a.find("r")
```

Out[ ]:
```
11
```

IRIS dataset line/bar plotting

import libraries

import dataset

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

In [ ]:
```
Iris= sns.load_dataset('iris')
Iris
```

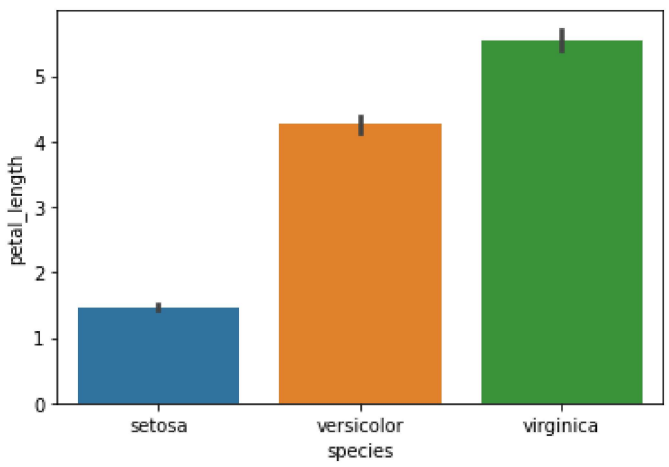
Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

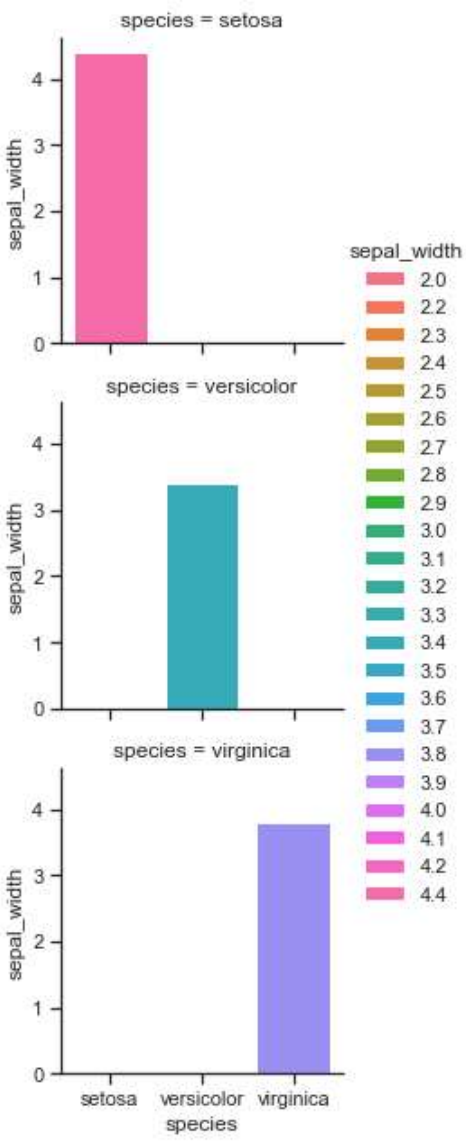
Iris Bar plot

In [ ]:
```
sns.barplot(x='species', y='petal_length', data=Iris)
plt.show()
```



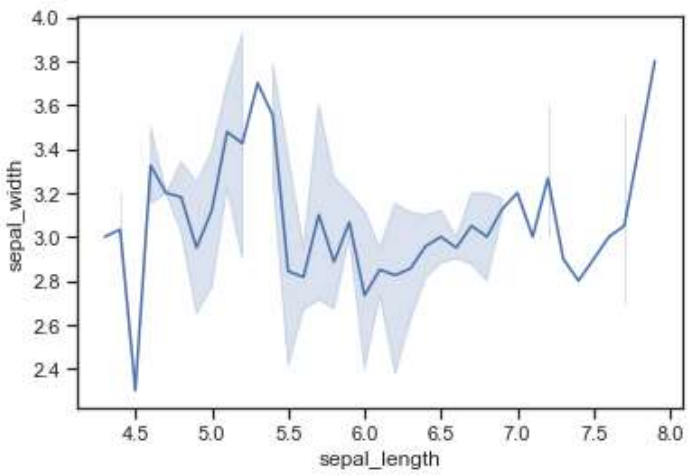
In [ ]:

```
sns.set_theme(style="ticks", color_codes=True)
g=sns.FacetGrid(Iris, row="species", hue="sepal_width")
g=(g.map(plt.bar,"species","sepal_width").add_legend())
plt.show()
```



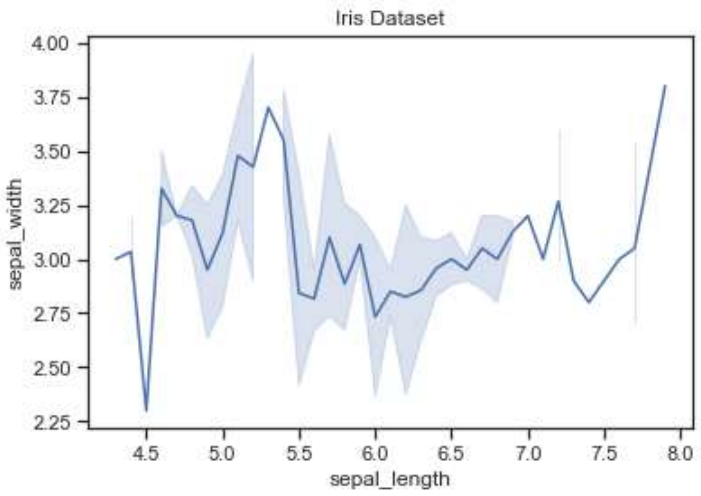
In [ ]:

```
#Lineplot Simple
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.show()
```

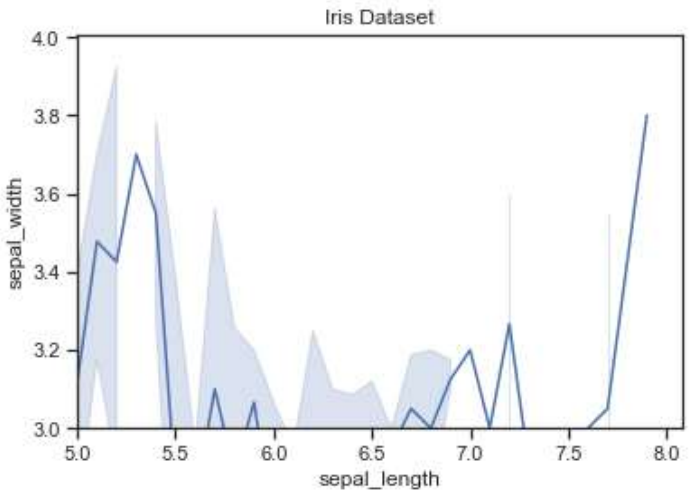


In [ ]:

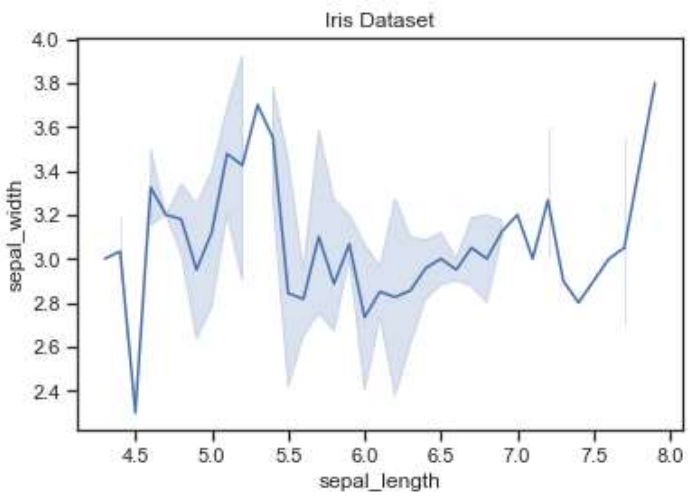
```
#Title
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.title('Iris Dataset')
plt.show()
```



In [ ]:
```
#Adding Limits
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.title('Iris Dataset')
plt.xlim(5)
plt.ylim(3)
plt.show()
```



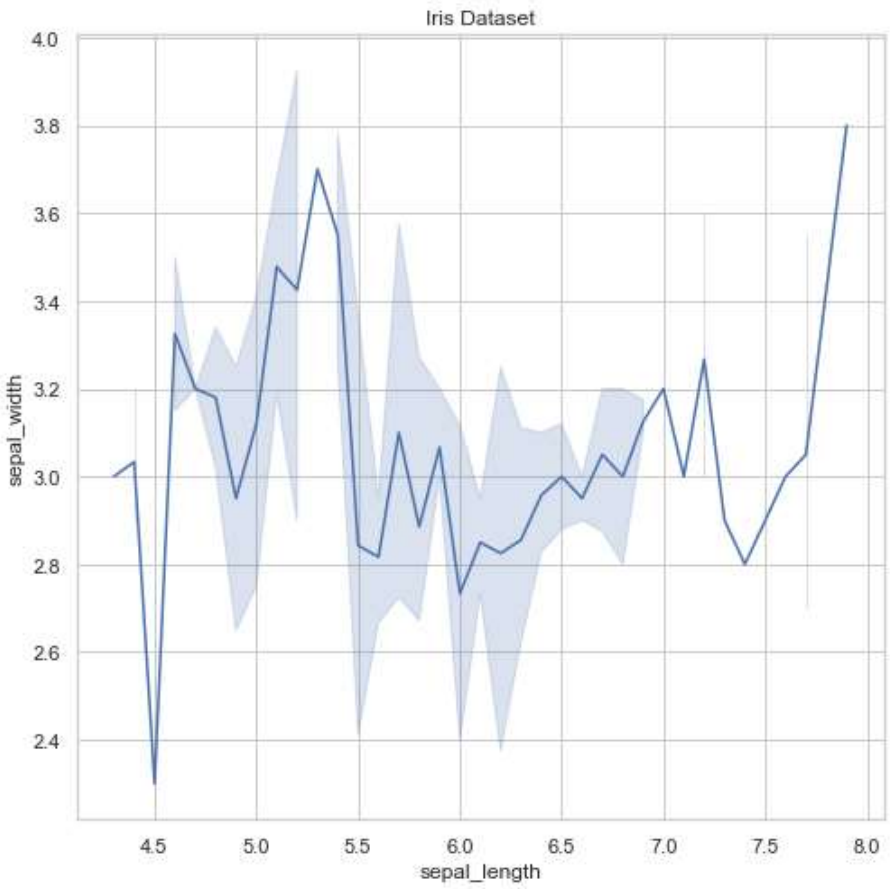
In [ ]:
```
#Style_Set
sns.set_style(style=None, rc=None)
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.title('Iris Dataset')
sns.set_style('darkgrid')
plt.show()
```



In [ ]:
```
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.title('Iris Dataset')
sns.set_style('whitegrid')
plt.show()
```

In [ ]:
```
#size of figure
Iris= sns.load_dataset('iris')
plt.figure(figsize=(8,8))
sns.lineplot(x='sepal_length', y='sepal_width', data=Iris)
plt.title('Iris Dataset')
```

```
sns.set_style('whitegrid')
plt.show()
```



In [ ]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\anaconda\lib\site-packages (5.5.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\anaconda\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six in c:\anaconda\lib\site-packages (from plotly) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
import seaborn as sns
#canvas style
sns.set(style='whitegrid')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Titanic data plotting (bar plot)

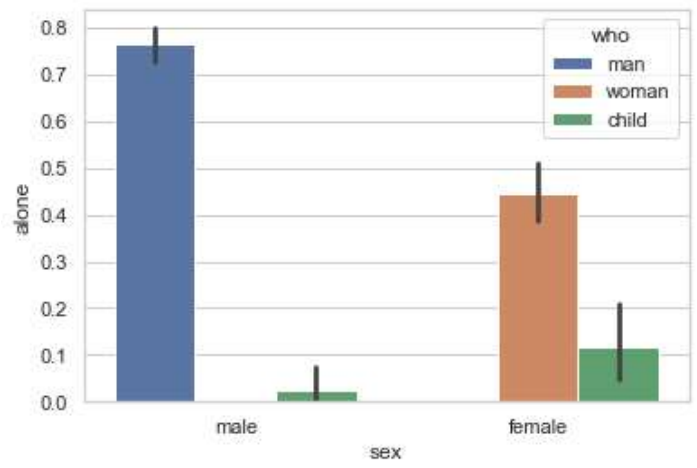
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
titanic=sns.load_dataset("titanic")
titanic
```

Out[ ]:

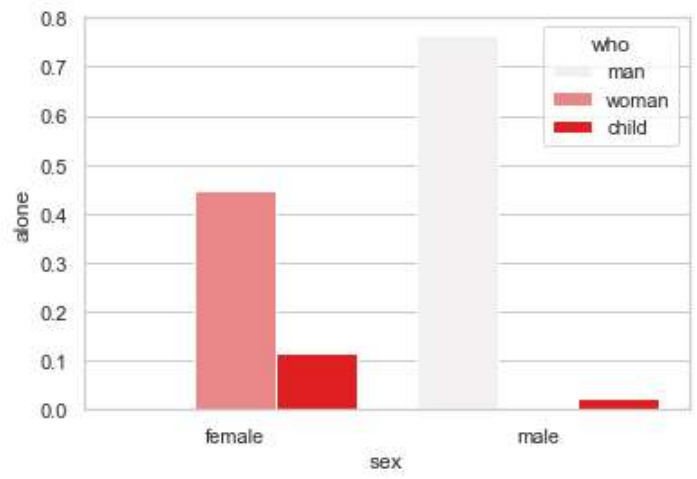
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

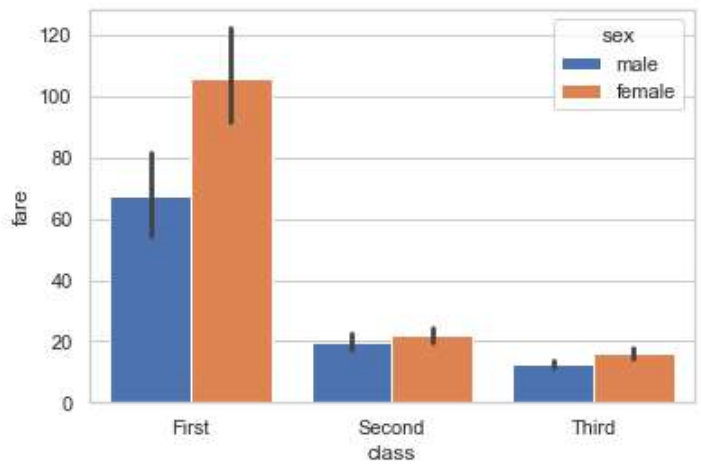
In [ ]:
```
#barplot
sns.barplot(x='sex',y='alone', hue='who',data=titanic)
plt.show()
```



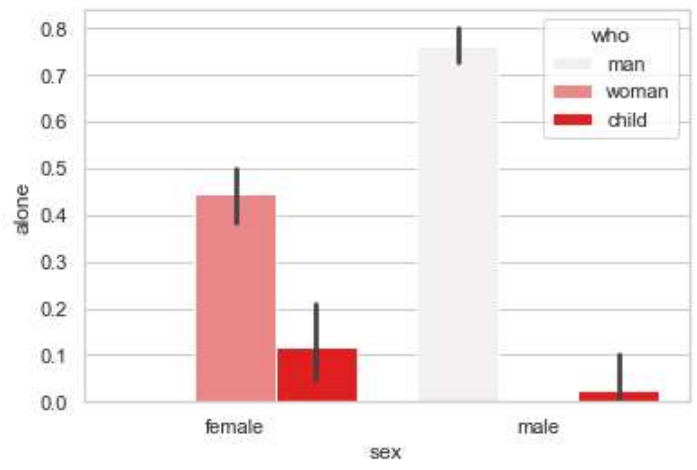
In [ ]:
```
#color and order
sns.barplot(x='sex',y='alone', hue='who',data=titanic, order=["female","male"], color="red", ci=None)
plt.show()
```



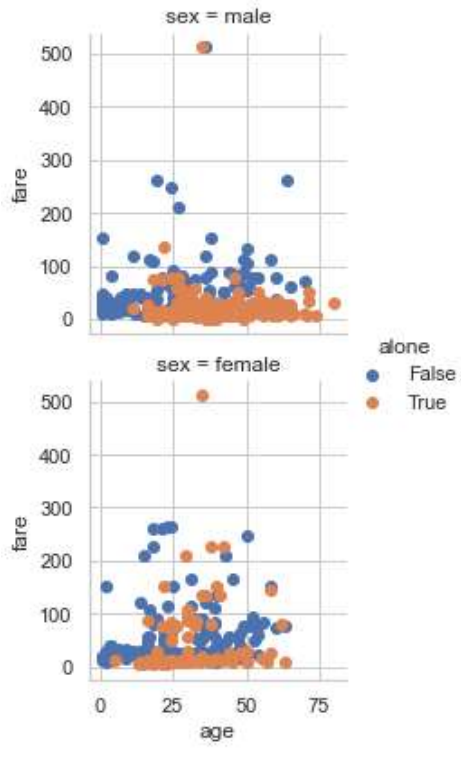
In [ ]:
```
#mean and median
import seaborn as sns
import matplotlib.pyplot as plt
from numpy import mean
sns.barplot(x='class',y='fare', hue='sex',data=titanic, estimator=mean, saturation= 1)
plt.show()
```



In [ ]:
```
sns.barplot(x='sex',y='alone', hue='who',data=titanic, order=["female","male"], color="red")
plt.show()
```



In [ ]:
```
g=sns.FacetGrid(titanic, row="sex", hue="alone")
g=(g.map(plt.scatter,"age","fare").add_legend())
plt.show()
```



Chilla_data2 plotting

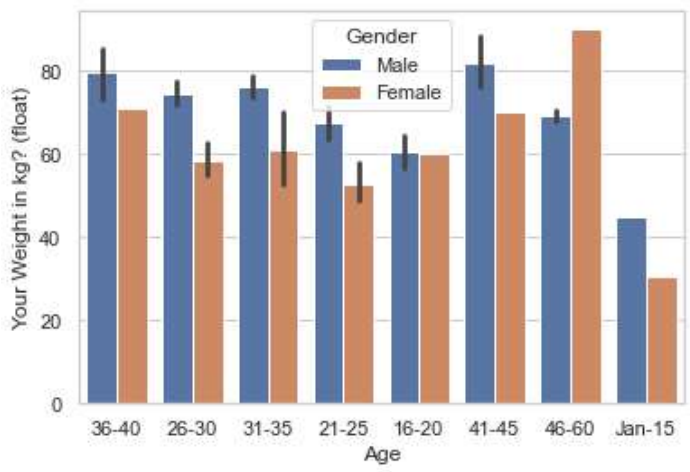
Line and bar plotting

importing data and libraries

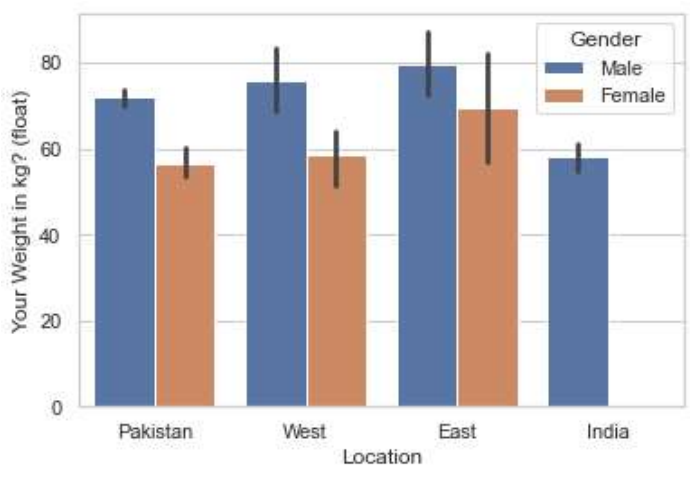
In [ ]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
Chilla_Class=pd.read_csv("Chilla_data2.csv")
#Chilla_Class
table = pd.DataFrame(Chilla_Class, columns = ['Age', 'Blood group',
                                              'Marital Status?','Age (years)-Float/Int', 'Gender','Are you Vaccinated?', 'Your Weight in kg? (floa)'

print("Data Types of The Columns in Data Frame")
display(table.dtypes)
```

```
Data Types of The Columns in Data Frame
Age                           object
Blood group                  float64
Marital Status?               object
Age (years)-Float/Int        float64
Gender                        object
Are you Vaccinated?           object
Your Weight in kg? (floa)    float64
Location                      object
dtype: object
```

In [ ]:
```
sns.barplot(data=Chilla_Class, x="Age", y="Your Weight in kg? (float)", hue="Gender")
plt.show()
```

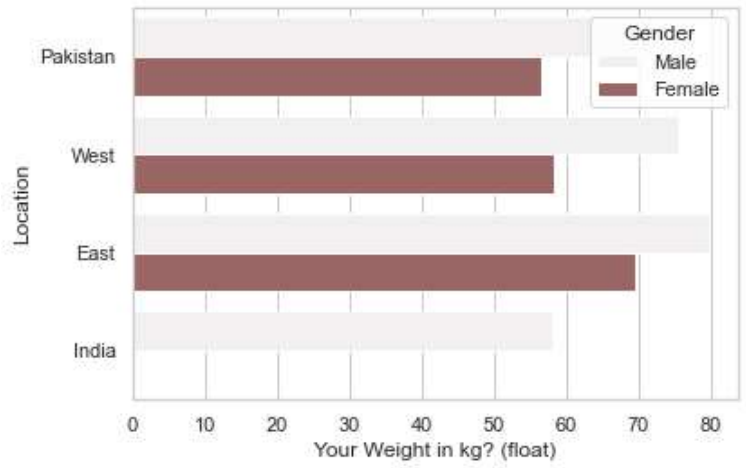


In [ ]:
```
sns.barplot(data=Chilla_Class, x="Location", y="Your Weight in kg? (float)", hue="Gender")
plt.show()
```



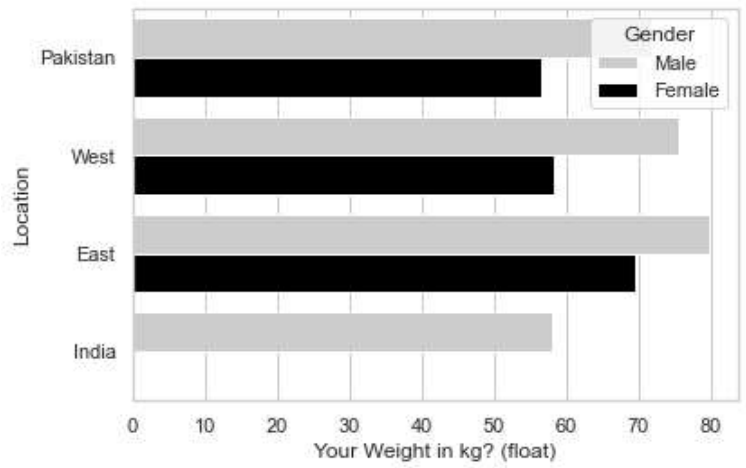
Horizonal Plot examples

In [ ]:
```
sns.barplot(data=Chilla_Class, x="Your Weight in kg? (float)", y="Location", hue="Gender", color="red", ci=None, saturation=0.2)
plt.show()
```



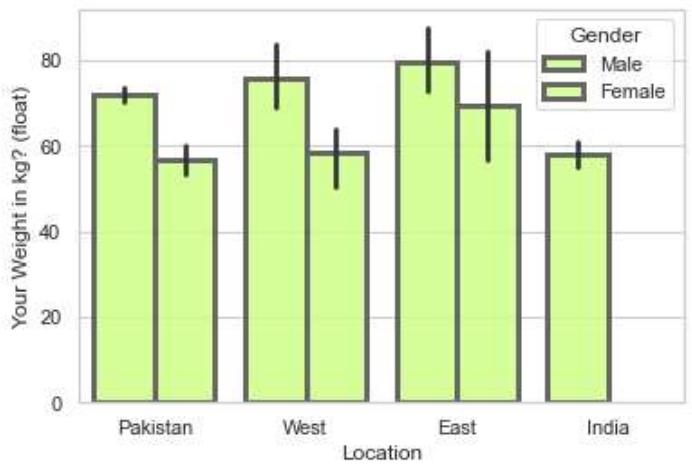
With indiviual palette color example

In [ ]:
```
sns.barplot(data=Chilla_Class, x="Your Weight in kg? (float)", y="Location", hue="Gender", color="red", ci=None, saturation=0.2,
            palette={"Male":"0.8","Female":"0"})
plt.show()
```



barplot with more cutomization in desgin (linewidth,edgecolour,facecolor (RGBalpha) etc)

In [ ]:
```
sns.barplot(data=Chilla_Class, x="Location", y="Your Weight in kg? (float)", hue="Gender",
            linewidth=3, facecolor=(0.8,1,0.5,0.8), errcolor=".2", edgecolor= ".4")
plt.show()
```



Python_Chilla_with_AMMAR

Plotting on Chilla_Class dataset

- import libraries
- import dataset
- datatypes
- clean dataset
- relplot
- scatter plot
- violin plot
- inera regression plot
- swarmplot
- strip plot

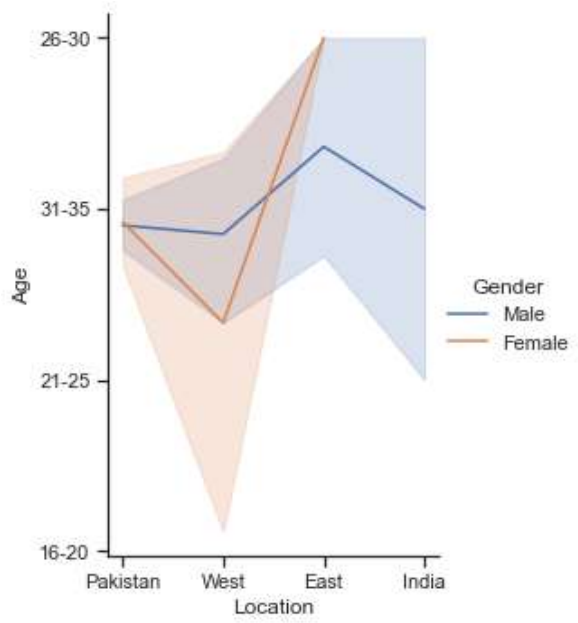
In [ ]:
```
#import libraries
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
##importing dataset
Chilla_Class=pd.read_csv("Chilla_data2.csv")
#Chilla_Class.head()
```

relplot

In [ ]:
```
import seaborn as sns
sns.set_theme(style="ticks")
# Define the palette as a list to specify exact values
palette = sns.color_palette("rocket_r")

# Plot the lines on two facets
sns.relplot(
    data=Chilla_Class,
    x="Location", y="Age",
    hue="Gender",
    kind="line", size_order=["T1", "T2"],
    height=5, aspect=.75, facet_kws=dict(sharex=False),
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x299fde13e50>
```



scatter plot

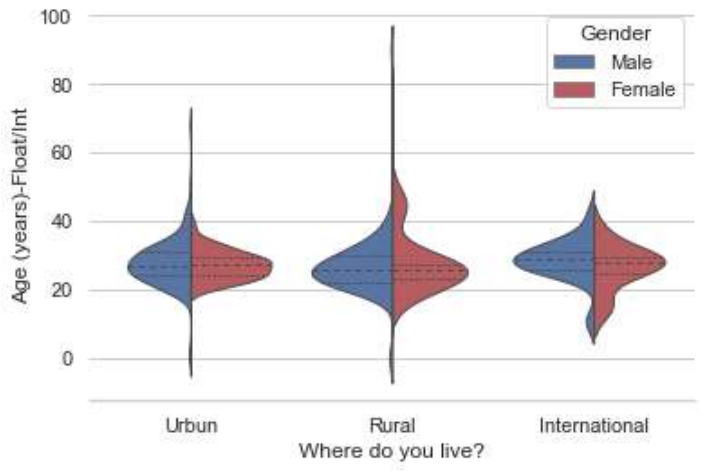
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")

# Load the example diamonds dataset
#Chilla_Class

# Draw a scatter plot while assigning point colors and sizes to different
# variables in the dataset
f, ax = plt.subplots(figsize=(6.5, 6.5))
sns.despine(f, right=True, bottom=False)
sns.scatterplot(x="Age (years)-Float/Int", y="Location",
                hue="Gender", size="Your favorite programming language?",
                palette="ch:r=8,d=0_r",
                sizes=(4, 10), linewidth=0,
                data=Chilla_Class)
```

Out[ ]:
```
<AxesSubplot:xlabel='Age (years)-Float/Int', ylabel='Location'>
```


ViolinPlot

In [ ]:
```
import seaborn as sns
sns.set_theme(style="whitegrid")

# Experiment on Chilla_Class dataset
```

```
# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(data=Chilla_Class, x="Where do you live?", y="Age (years)-Float/Int", hue="Gender",
               split=True, inner="quart", linewidth=1,
               palette={"Male": "b", "Female": "r"})
sns.despine(left=True)
```



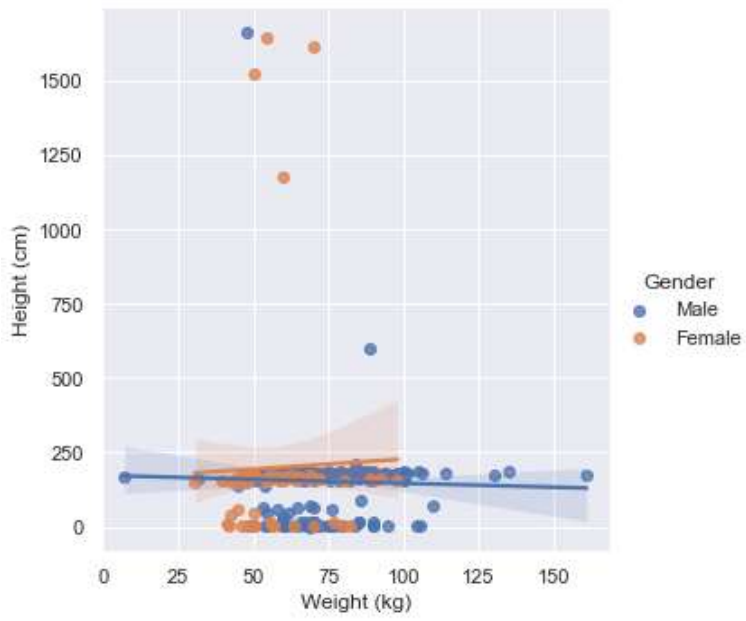
muliple linear regression

In [ ]:

```
import seaborn as sns
sns.set_theme()

# Plot sepal width as a function of sepal_length across days
g = sns.lmplot(
    data=Chilla_Class,
    x="Your Weight in kg? (float)", y="Height in cm? Freelancer- (Float)", hue="Gender",
    height=5
)

# Use more informative axis labels than are provided by default
g.set_axis_labels("Weight (kg)", "Height (cm)")
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x299824488b0>
```



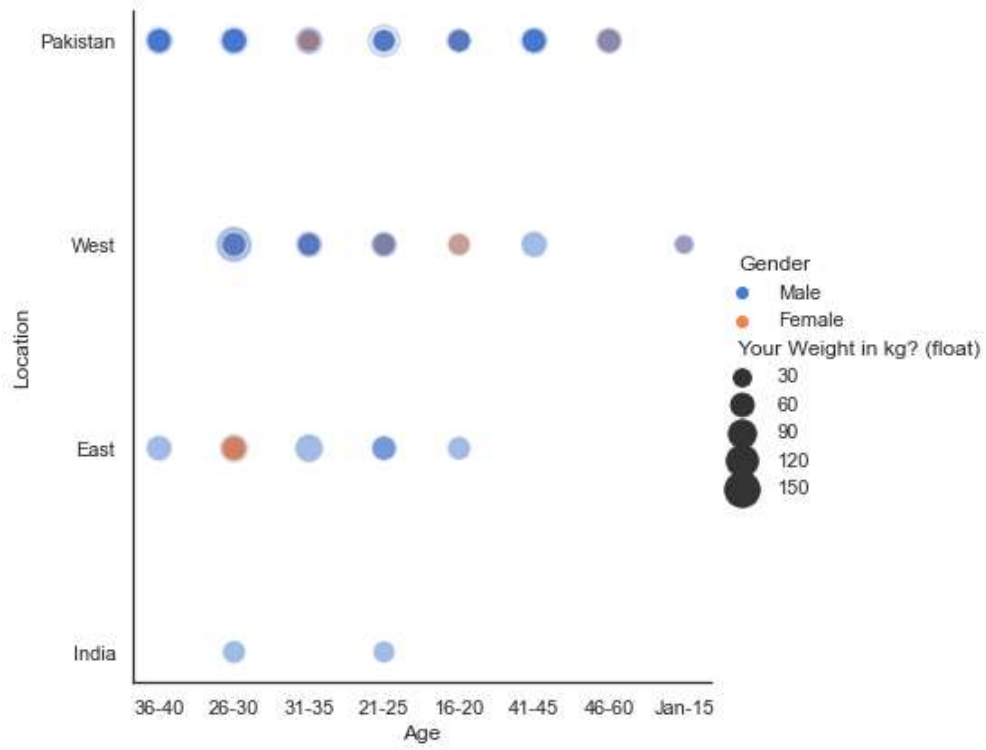
Scatterplot with varying point sizes and hues

In [ ]:

```
import seaborn as sns
sns.set_theme(style="white")
#Chilla_Class

# Plot miles per gallon against horsepower with other semantics
sns.relplot(x="Age", y="Location", hue="Gender", size="Your Weight in kg? (float)",
            sizes=(40, 400), alpha=.5, palette="muted",
            height=6, data=Chilla_Class)
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0x299822fbe80>
```

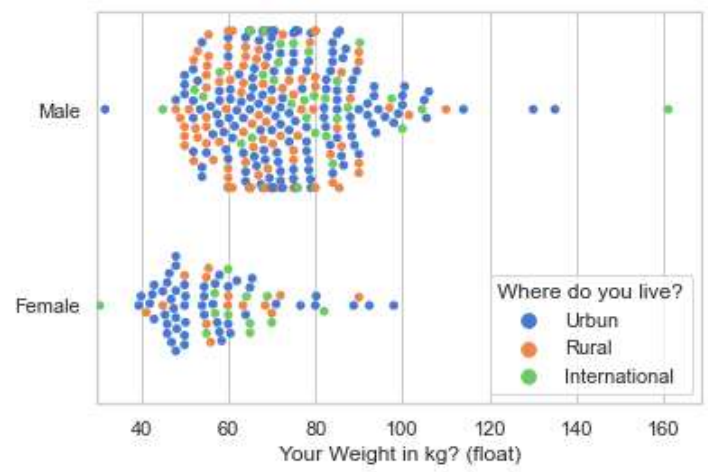


swarmplot

In [ ]:
```
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")

# Draw a categorical scatterplot to show each observation
ax = sns.swarmplot(data=Chilla_Class, x="Your Weight in kg? (float)", y="Gender", hue="Where do you live?")
ax.set(ylabel="")
plt.xlim(30)
```

```
C:\Anaconda\lib\site-packages\seaborn\categorical.py:1296: UserWarning:

17.8% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
```

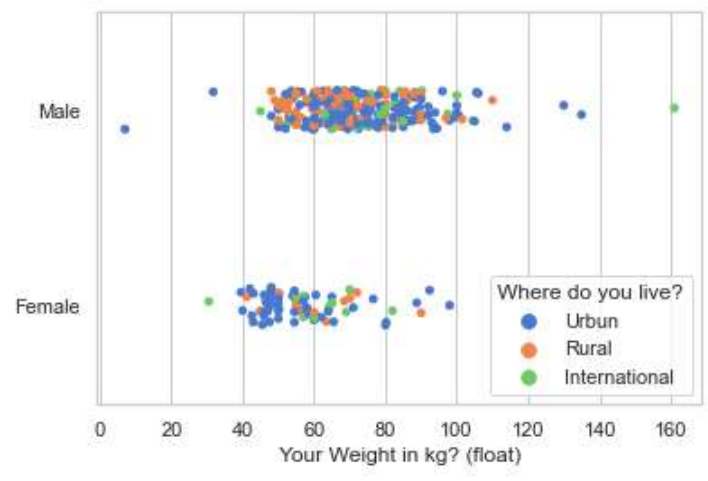
Out[ ]:
```
(30.0, 168.7)
```



Strip plot as suggested by previous output

In [ ]:
```
import seaborn as sns
sns.set_theme(style="whitegrid", palette="muted")

# Draw a categorical scatterplot to show each observation
ax = sns.stripplot(data=Chilla_Class, x="Your Weight in kg? (float)", y="Gender", hue="Where do you live?")
ax.set(ylabel="")
#plt.xlim(30)
```

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



Python_Chilla_with_Ammar

Plotly practice

import plotly library

- Chilla_Class dataset
- Different plots using plotly

In [ ]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\anaconda\lib\site-packages (5.5.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\anaconda\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six in c:\anaconda\lib\site-packages (from plotly) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

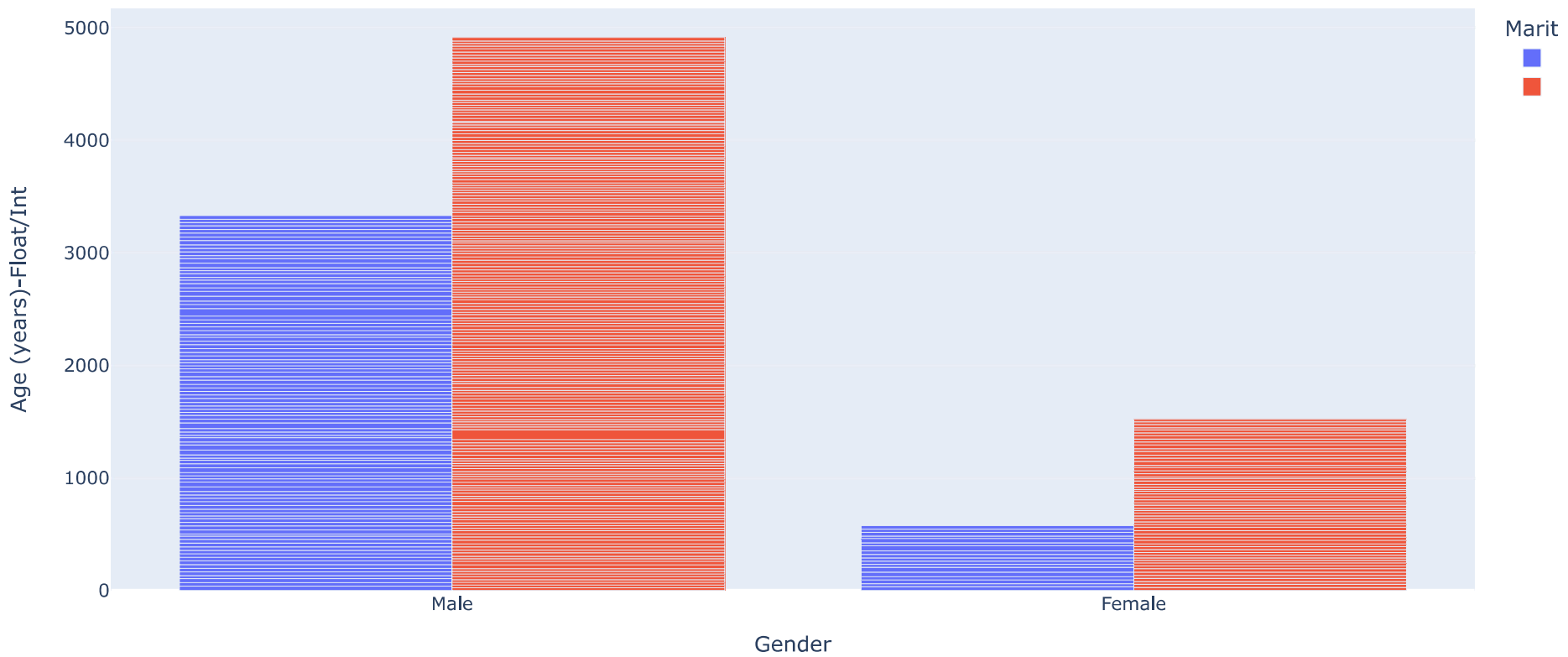
In [ ]:
```
import seaborn as sns
#canvas style
sns.set(style='whitegrid')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
##importing dataset
Chilla_Class=pd.read_csv("Chilla_data2.csv")
#Chilla_Class
```

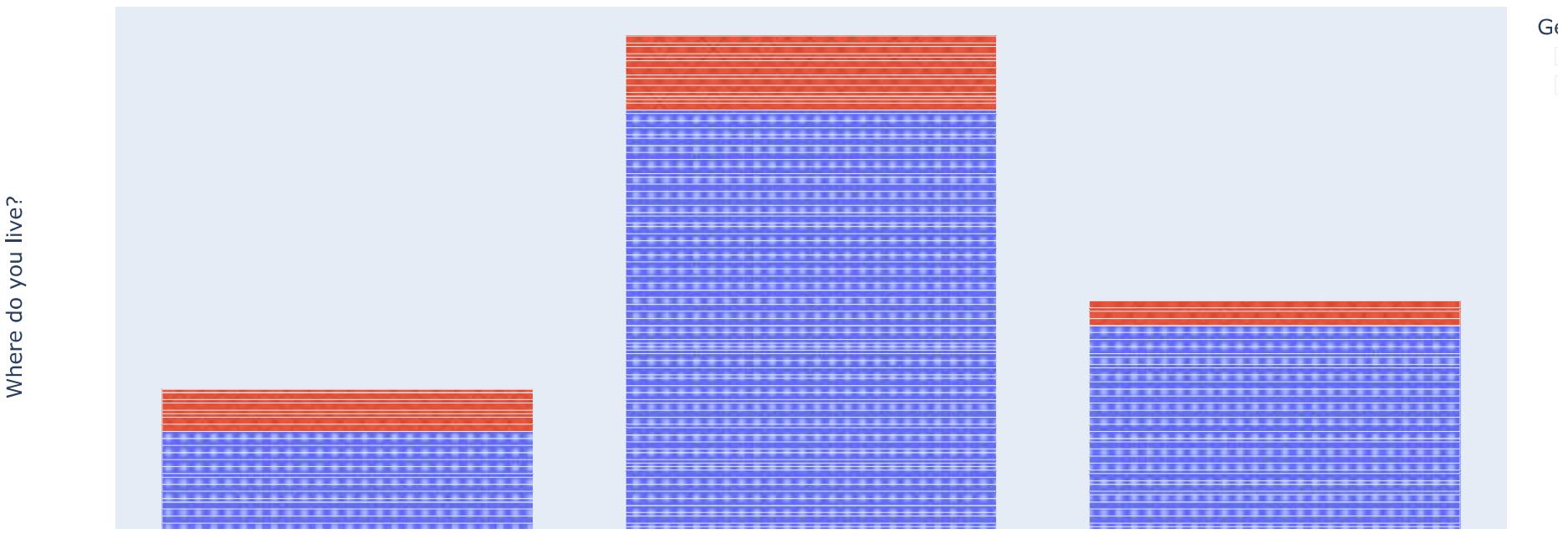
Import plotly.offline as py to make compatible in VS code

In [ ]:
```
import plotly.offline as py
py.init_notebook_mode(connected=True)
```

In [ ]:
```
#barplot with more details and animations
import plotly.express as px
fig = px.bar(Chilla_Class, x="Gender", y="Age (years)-Float/Int", color="Marital Status?", barmode="group")
fig.show()
```



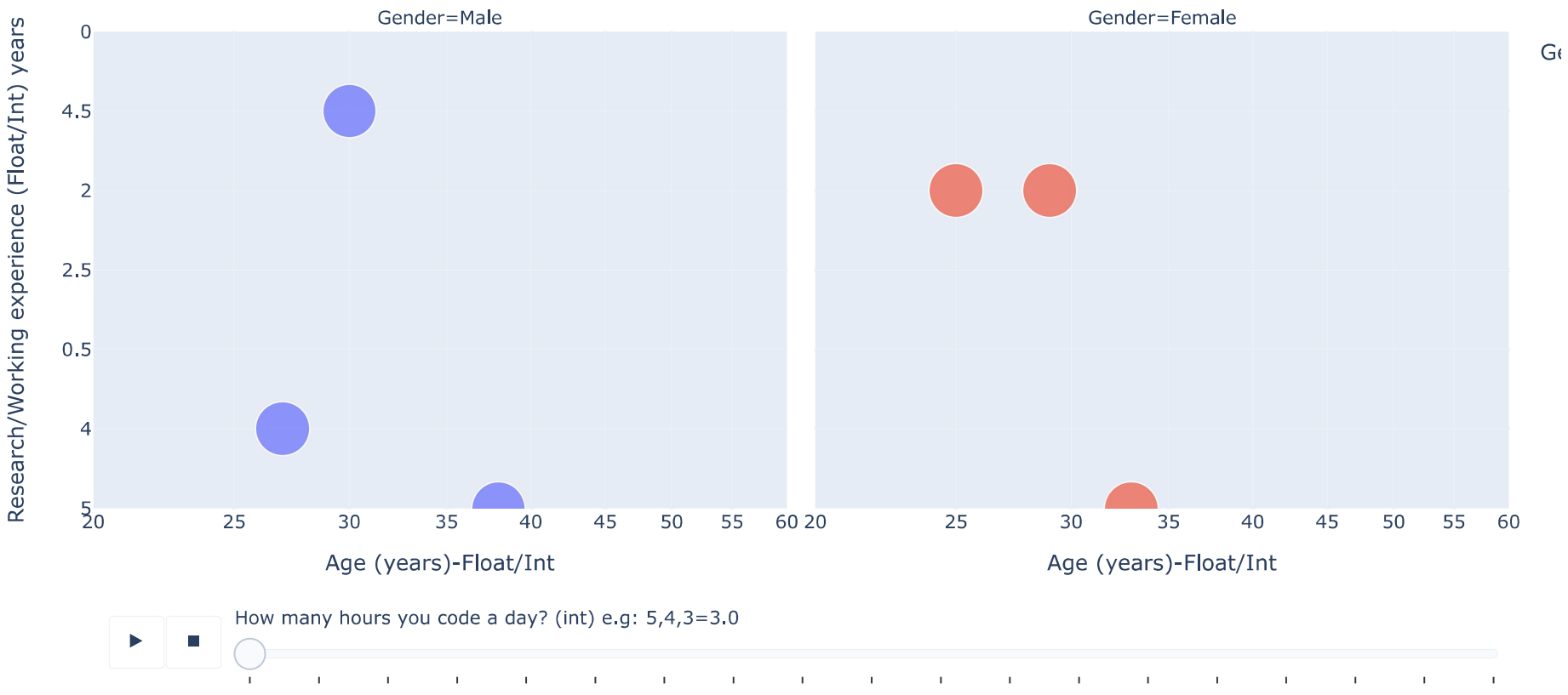
In [ ]:
```
fig = px.bar(Chilla_Class, x="What are you?", y="Where do you live?", color="Gender",
             pattern_shape="Gender", pattern_shape_sequence=[".", "x", "+"])
fig.show()
```



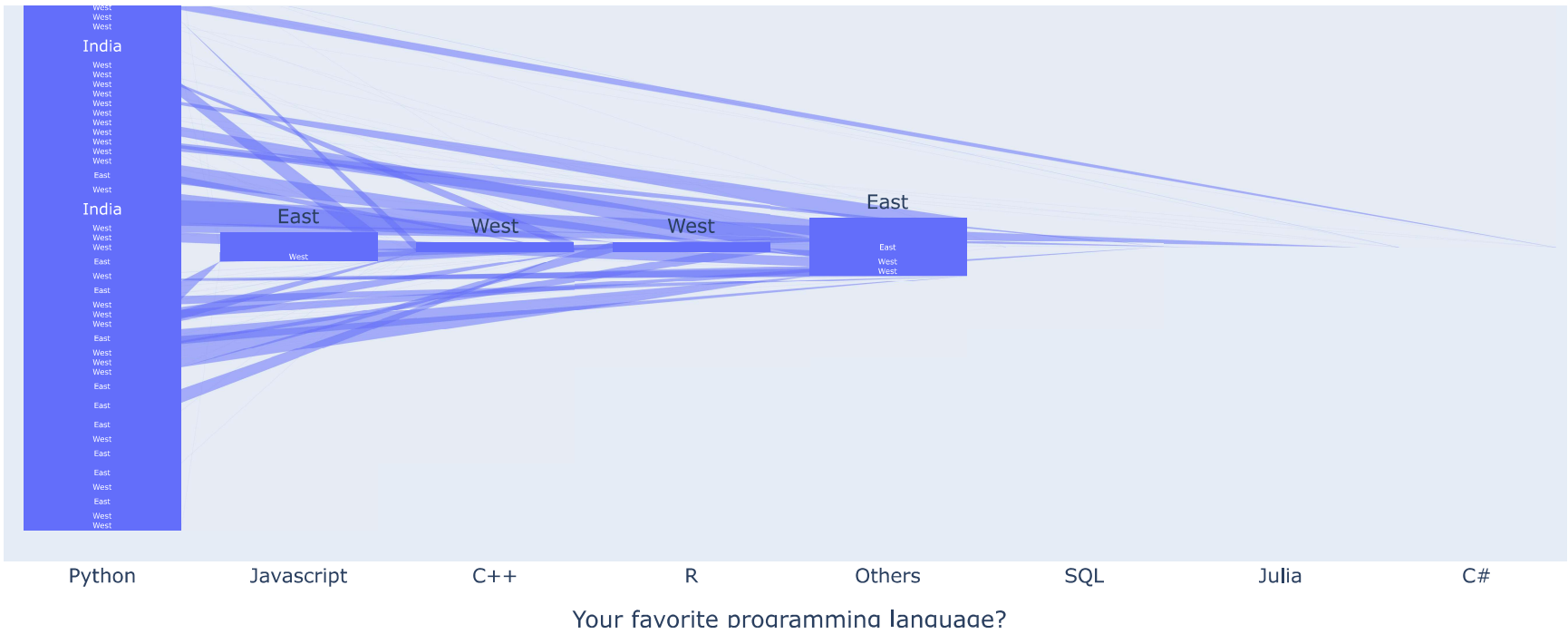
In [ ]:
```
fig = px.bar(Chilla_Class, x="Which mobile sim do you use", y="Age", color="Gender",
             pattern_shape="Gender", pattern_shape_sequence=[".", "x", "+"])
fig.show()
```


Animation plot

In [ ]:
```
import plotly.express as px
#df = px.data.gapminder()
fig = px.scatter(Chilla_Class, x="Age (years)-Float/Int", y="Research/Working experience (Float/Int) years", animation_frame="How many hours you c
           size="How many hours you code a day? (int) e.g: 5,4,3", color="Gender", hover_name="Location", facet_col="Gender",
           log_x=True, size_max=60, range_x=[20,60], range_y=[0,6])
fig.show()
```



In [ ]:
```
#Funnel plot
fig = px.funnel(Chilla_Class, x='Your favorite programming language?', y='Location')
fig.show()
```



Importing image from website

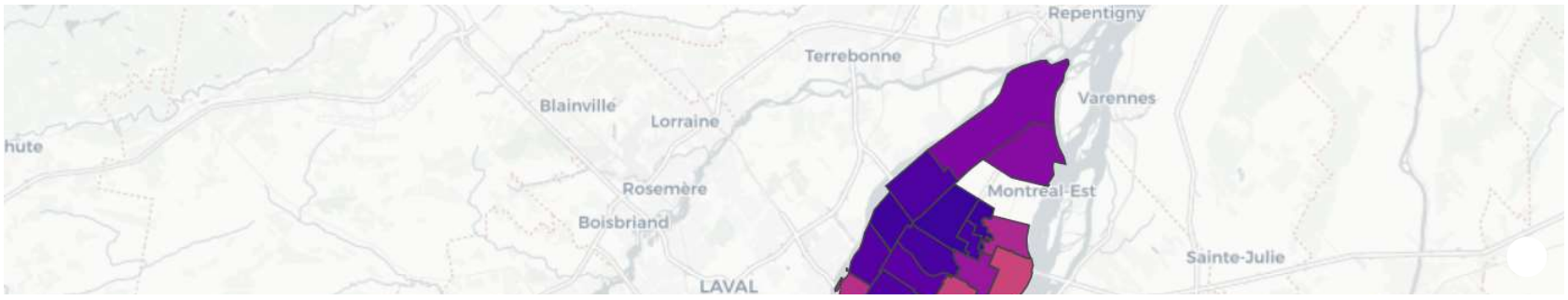
In [ ]:

```
import plotly.express as px
from skimage import io
img = io.imread('https://media-exp1.licdn.com/dms/image/C4E22AQGZvqJb7RRG6w/feedshare-shrink_2048_1536/0/1640175625963?e=1643241600&v=beta&t=H8s0z
fig = px.imshow(img)
fig.show()
```

Tilemap and Heat map

In [ ]:

```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_mapbox(df, geojson=geojson, color="Bergeron",
                           locations="district", featureidkey="properties.district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           mapbox_style="carto-positron", zoom=9)
fig.show()
```



Markdown Practice

1-Content

2-Headings

How to give headings in Markdown file

Heading 1

Heading 2

Heading 3

Heading 4

3-Block of words

This a block of text normal in Markdown.

> This is a block of special text.
>
> This is second line of special text.

4-Lines breaks

This is a 40 days data science with python course with baba ammar and also known as Python ka chilla.

This is second line and show double enter and back/ can be used to line break.

5-Combine two things

block of words and headings.

> ## heading 2

6-Face of text

Bold

italic

Both

or we can use other symbols like:

bold

italic

7-Bullet points/Lists

- Day-1
- Day-2
  - 2a
  - 2b
    - Sub-headings 2b.a
- Day-3
- Day-4
- Day-5
- Day-6
- Day-7
- Day-8

> Numbering of numbers
>
> 1. Day-1
> 2. Day-2
> 3. Day-3
> 4. Day-4
> 5. Day-5

we can also use * or + signs to make lists

- Class day 1
- Class day 2
- **Class day 3**
- *Class day 4*

8-Lines break and page break

This is one page and this is a guidline foe markdown language.

This class is taken by baba ammar and it is 40 days chilla.

9-Links and Hyperlinks

[Link] (https://www.youtube.com/watch?v=qJqAXjz-Rh4)

> this is an example of hyperlink.

All videos avaiable here

10-Images and figure with links

To join this course, please scan join telegram grop for class updates. I am enrolled student in this course. Let me introsuce myself. My name is munazza, first year PhD student in Italy.

Online course details can be show on this picture: 

11-Adding code line or code block

To print a string use `print (Codanics)`

```
Code details can be given like:

x=5+6
y=5-1
z=x+y
print(z)
```

> This code will show according to R or Python
>
> ```
> x=5+6
>  y=5-1
>  z=x+y
>  print(z)
> ```

12- Adding Tables

Species	Petal_Length	Sepal_Length	
Verginica	10.2	11.3	9
Setosa	10.2	11.3	9
Versicolor	10.2	11.3	9
Verginica	10.2	11.3	9
Setosa	10.2	11.3	9

13- Insatall Extention

To use these extensions more smart way is just click on text and there are a lot of options (like using toggle)

Image

Numpy Leacture and Practice

In [ ]:
```
import seaborn as sns
#canvas style
sns.set(style='whitegrid')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
a= np.array([1,2,3,4,5])
a
```

Out[ ]:
```
array([1, 2, 3, 4, 5])
```

In [ ]:
```
a= np.array([[1,2,3,4,5],[1,1,1,1,1],[0,0,0,0,0]])
a
```

Out[ ]:
```
array([[1, 2, 3, 4, 5],
       [1, 1, 1, 1, 1],
       [0, 0, 0, 0, 0]])
```

In [ ]:
```
type(a)
```

Out[ ]:
```
numpy.ndarray
```

In [ ]:

```
len(a)
```

Out[ ]:
```
3
```

In [ ]:
```
b= np.zeros(5)
b
```

Out[ ]:
```
array([0., 0., 0., 0., 0.])
```

In [ ]:
```
c=np.ones(7)
c
```

Out[ ]:
```
array([1., 1., 1., 1., 1., 1., 1.])
```

In [ ]:
```
d=np.arange(5)
d
```

Out[ ]:
```
array([0, 1, 2, 3, 4])
```

In [ ]:
```
e=np.arange(2,20,2)
e
```

Out[ ]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [ ]:
```
h=np.linspace(0,10, num=5)
h
```

Out[ ]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [ ]:
```
#Specific data type array
i=np.ones(5, dtype=np.int8)
i
```

Out[ ]:
```
array([1, 1, 1, 1, 1], dtype=int8)
```

In [ ]:
```
#2D Array
a=np.zeros((2,3))
a
```

Out[ ]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [ ]:
```
b=np.ones ((3,4), dtype=np.int8)
b
```

Out[ ]:
```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]], dtype=int8)
```

3-D array

In [ ]:
```
#making 3D array
c=np.arange(24).reshape(2,3,4)
c
```

Out[ ]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

In [ ]:
```
#ID array
food=np.array (['Pakora','Samosa','Cake'])
food
```

Out[ ]:
```
array(['Pakora', 'Samosa', 'Cake'], dtype='<U6')
```

In [ ]:
```
price=np.array([5,5,5])
price
```

Out[ ]:
```
array([5, 5, 5])
```

In [ ]:
```
len(food)
```

Out[ ]:
```
3
```

In [ ]:
```
food[1]
```

Out[ ]:
```
'Samosa'
```

In [ ]:
```
#mean function
```

```
price.mean()
```

Out[ ]:
```
5.0
```

In [ ]:
```
#range method
#zeros method
np.zeros(6)
```

Out[ ]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [ ]:
```
#empty
np.ones(5)
```

Out[ ]:
```
array([1., 1., 1., 1., 1.])
```

In [ ]:
```
#empty
np.empty(5)
```

Out[ ]:
```
array([1., 1., 1., 1., 1.])
```

In [ ]:
```
#2D array with zero
np.zeros([3,2])
```

Out[ ]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [ ]:
```
#range
np.arange(2,20)
```

Out[ ]:
```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19])
```

In [ ]:
```
#with specific range
np.arange(2,20,4)
```

Out[ ]:
```
array([ 2,  6, 10, 14, 18])
```

In [ ]:
```
#table
np.arange(0,55,5)
```

Out[ ]:
```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50])
```

In [ ]:
```
#linspace (same difference or same intverval)
np.linspace(1,100, num=10)
```

Out[ ]:
```
array([  1.,  12.,  23.,  34.,  45.,  56.,  67.,  78.,  89., 100.])
```

In [ ]:
```
#specify your datatype
np.ones(20, dtype=np.int64)
```

Out[ ]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      dtype=int64)
```

In [ ]:
```
#specify your datatype
np.ones(20, dtype=np.float64)
```

Out[ ]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1.])
```

array functions

In [ ]:
```
a=np.array([10,12,15,2,6,100,320,0.5,10.3])
a
```

Out[ ]:
```
array([ 10. ,  12. ,  15. ,   2. ,   6. , 100. , 320. ,   0.5,  10.3])
```

In [ ]:
```
a.sort()
a
```

Out[ ]:
```
array([  0.5,   2. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. , 320. ])
```

In [ ]:
```
a.max()
```

Out[ ]:
```
320.0
```

In [ ]:
```
b=np.array([10.2,53.6,4,2,40,11.5,7,12])
b
```

Out[ ]:
```
array([10.2, 53.6,  4. ,  2. , 40. , 11.5,  7. , 12. ])
```

In [ ]:
```
#concatenate
c=np.concatenate((a,b))
```

```
c
```

Out[ ]:
```
array([  0.5,   2. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. , 320. ,
        10.2,  53.6,   4. ,   2. ,  40. ,  11.5,   7. ,  12. ])
```

2D Array

In [ ]:
```
a=np.array([[1,2,3],[4,5,6]])
a
```

Out[ ]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [ ]:
```
b=np.array([[7,8,6],[9,10,11]])
b
```

Out[ ]:
```
array([[ 7,  8,  6],
       [ 9, 10, 11]])
```

In [ ]:
```
c=np.concatenate((a,b))
c
```

Out[ ]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  6],
       [ 9, 10, 11]])
```

In [ ]:
```
c=np.concatenate((a,b), axis=1)
c
```

Out[ ]:
```
array([[ 1,  2,  3,  7,  8,  6],
       [ 4,  5,  6,  9, 10, 11]])
```

In [ ]:
```
c=np.concatenate((a,b), axis=0)
c
```

Out[ ]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  6],
       [ 9, 10, 11]])
```

In [ ]:
```
b=np.array([[1,2,3],[4,4,4],[5,6,8]])
b
```

Out[ ]:
```
array([[1, 2, 3],
       [4, 4, 4],
       [5, 6, 8]])
```

In [ ]:
```
b.ndim
```

Out[ ]:
```
2
```

In [ ]:
```
c = np.array([[[  0,  1,  2],  # a 3D array (two stacked 2D arrays)
               [ 10, 12, 13]],
              [[100, 101, 102],
               [110, 112, 113]]])
c
```

Out[ ]:
```
array([[[  0,   1,   2],
        [ 10,  12,  13]],

       [[100, 101, 102],
        [110, 112, 113]]])
```

In [ ]:
```
b=np.array([[[1,2,3],
             [4,4,4]],
            [[5,6,8],
          [1,1,1]]])
b
```

Out[ ]:
```
array([[[1, 2, 3],
        [4, 4, 4]],

       [[5, 6, 8],
        [1, 1, 1]]])
```

In [ ]:
```
a=np.arange(9)
a
```

Out[ ]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [ ]:
```
#reshape
b=a.reshape(3,3)
b
```

Out[ ]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [ ]:
```
#reshape
np.reshape(a, newshape=(1,9), order='C')
```

Out[ ]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8]])
```

In [ ]:
```
#convert 1D into 2D
a= np.array([1,2,3,4,5,6])
a
```

Out[ ]:
```
array([1, 2, 3, 4, 5, 6])
```

In [ ]:
```
a.shape
```

Out[ ]:
```
(6,)
```

In [ ]:
```
b= a[np.newaxis, :]
b
```

Out[ ]:
```
array([[1, 2, 3, 4, 5, 6]])
```

In [ ]:
```
b.shape
```

Out[ ]:
```
(1, 6)
```

In [ ]:
```
#columnwise conversion
c= a[:, np.newaxis]
c
```

Out[ ]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6]])
```

In [ ]:
```
#operators with array #scaler multiplication or addition
a*6
```

Out[ ]:
```
array([ 6, 12, 18, 24, 30, 36])
```

In [ ]:
```
a+6
```

Out[ ]:
```
array([ 7,  8,  9, 10, 11, 12])
```

In [ ]:
```
a.sum()
```

Out[ ]:
```
21
```

In [ ]:
```
#add the arrays together with the plus sign
data = np.array([1, 2])
ones = np.ones(2, dtype=int)
data + ones
```

Out[ ]:
```
array([2, 3])
```

In [ ]:
```
ones #1+1 and 2+1
```

Out[ ]:
```
array([1, 1])
```

In [ ]:
```
#more math operations in array
data - ones
```

Out[ ]:
```
array([0, 1])
```

In [ ]:
```
data * data
```

Out[ ]:
```
array([1, 4])
```

In [ ]:
```
data / data
```

Out[ ]:
```
array([1., 1.])
```

In [ ]:
```
b = np.array([[1, 1], [2, 2]])
b
```

Out[ ]:
```
array([[1, 1],
       [2, 2]])
```

In [ ]:
```
#addition with specific axis
b.sum(axis=0)
```

Out[ ]:
```
array([3, 3])
```

In [ ]:
```
b.sum(axis=1)
```

Out[ ]:
```
array([2, 4])
```

In [ ]:
```
#min and max according to axis
data = np.array([[1, 2], [5, 3], [4, 6]])
data
```

Out[ ]:
```
array([[1, 2],
       [5, 3],
       [4, 6]])
```

In [ ]:
```
data.max(axis=0)
```

Out[ ]:
```
array([5, 6])
```

In [ ]:
```
data.max(axis=1)
```

Out[ ]:
```
array([2, 5, 6])
```

In [ ]:
```
data.min(axis=0)
```

Out[ ]:
```
array([1, 2])
```

In [ ]:
```
data.min(axis=1)
```

Out[ ]:
```
array([1, 3, 4])
```

In [ ]:
```
#random data generation
a = np.random.default_rng(0)
a.random(3)
```

Out[ ]:
```
array([0.63696169, 0.26978671, 0.04097352])
```

In [ ]:
```
#getting unique or remove duplicate data
a = np.array([11, 11, 12, 13, 14, 15, 16, 17, 12, 13, 11, 14, 18, 19, 20])
a
```

Out[ ]:
```
array([11, 11, 12, 13, 14, 15, 16, 17, 12, 13, 11, 14, 18, 19, 20])
```

In [ ]:
```
b = np.unique(a)
b
```

Out[ ]:
```
array([11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

In [ ]:
```
#reversing function 1D
c = np.array([1, 2, 3, 4, 5, 6, 7, 8])
c
```

Out[ ]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [ ]:
```
d = np.flip(c)
d
```

Out[ ]:
```
array([8, 7, 6, 5, 4, 3, 2, 1])
```

In [ ]:
```
#reversing function 2D
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
a
```

Out[ ]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

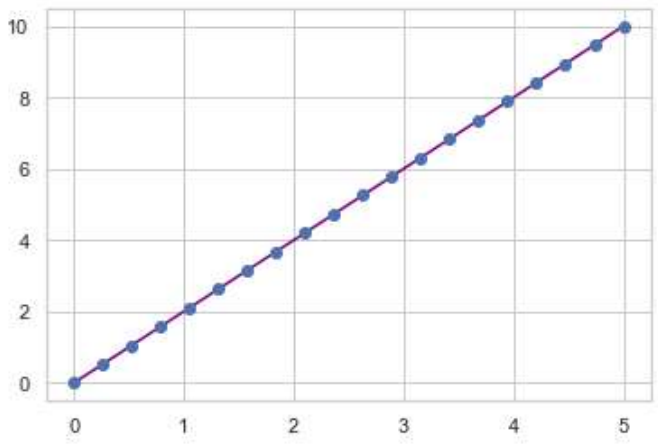
In [ ]:
```
b=np.flip(a)
b
```

Out[ ]:
```
array([[12, 11, 10,  9],
       [ 8,  7,  6,  5],
       [ 4,  3,  2,  1]])
```

plotting arrays

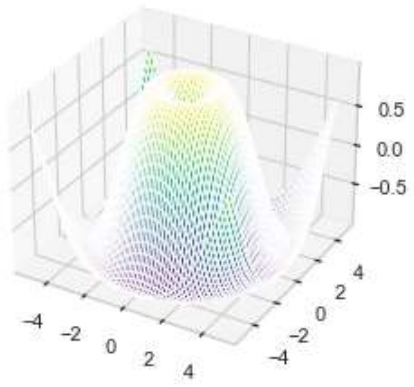
In [ ]:
```
x = np.linspace(0, 5, 20)
y = np.linspace(0, 10, 20)
plt.plot(x, y, 'purple') # line
plt.plot(x, y, 'o')      # dots
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x29983466910>]
```



In [ ]:
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
X = np.arange(-5, 5, 0.15)
Y = np.arange(-5, 5, 0.15)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis')
```

Out[ ]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x2998345c5e0>
```



Pandas tutorial (Day-11)

This notebook explains pandas implementation

In [ ]:
```
#import libraries
import pandas as pd
import numpy as np
```

In [ ]:
```
#object creation (like make a series and understanding of pandas defined functions)
s= pd.Series([1,2,3,4,np.nan,5]) #nan represet empty cell
s
```

Out[ ]:
```
0    1.0
1    2.0
2    3.0
3    4.0
4    NaN
5    5.0
dtype: float64
```

In [ ]:
```
date= pd.date_range("20150101", periods=6) #how to create date series using pd.date
date
```

Out[ ]:
```
DatetimeIndex(['2015-01-01', '2015-01-02', '2015-01-03', '2015-01-04',
               '2015-01-05', '2015-01-06'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:
```
date= pd.date_range("20150101", periods=6) #dataframe mean like an excelsheet creation
df=pd.DataFrame(np.random.randn(6,4), index=date, columns=list('ABCD'))
df
```

Out[ ]:

	A	B	C	D
2015-01-01	0.908978	-0.683433	-0.118416	-0.625582
2015-01-02	-0.545005	-0.001416	1.169408	0.454385
2015-01-03	0.330998	-1.757281	-0.898525	-1.612164
2015-01-04	0.630375	-0.199042	-0.806920	0.015171
2015-01-05	-0.800132	0.551789	0.329205	0.399188
2015-01-06	1.033763	-1.393580	0.392546	0.493160

In [ ]:
```
#how to make like dictionary  #data fram dictionary
df2= pd.DataFrame(
    {
        "a":1.0,
        "b":pd.Timestamp("20150102"),
        "c":pd.Series(1, index=list(range(5)), dtype="float32"),
```

```
        "d":np.array([3]*5, dtype="int32"),
        "e":pd.Categorical(["girl","women","men","boy","child"]),
        "f":"Gender",

    }

)
```

In [ ]:
```
df2.dtypes
```

Out[ ]:
```
a           float64
b    datetime64[ns]
c           float32
d             int32
e          category
f            object
dtype: object
```

In [ ]:
```
df.head(2) #showing start of data
```

Out[ ]:

	A	B	C	D
2015-01-01	0.908978	-0.683433	-0.118416	-0.625582
2015-01-02	-0.545005	-0.001416	1.169408	0.454385

In [ ]:
```
df.tail(2) #showing last part of data
```

Out[ ]:

	A	B	C	D
2015-01-05	-0.800132	0.551789	0.329205	0.399188
2015-01-06	1.033763	-1.393580	0.392546	0.493160

In [ ]:
```
df2.index #index
```

Out[ ]:
```
Int64Index([0, 1, 2, 3, 4], dtype='int64')
```

In [ ]:
```
df2.to_numpy() #converting into array
```

Out[ ]:
```
array([[1.0, Timestamp('2015-01-02 00:00:00'), 1.0, 3, 'girl', 'Gender'],
       [1.0, Timestamp('2015-01-02 00:00:00'), 1.0, 3, 'women', 'Gender'],
       [1.0, Timestamp('2015-01-02 00:00:00'), 1.0, 3, 'men', 'Gender'],
       [1.0, Timestamp('2015-01-02 00:00:00'), 1.0, 3, 'boy', 'Gender'],
       [1.0, Timestamp('2015-01-02 00:00:00'), 1.0, 3, 'child', 'Gender']],
      dtype=object)
```

In [ ]:
```
df2.describe() #calculatinf statitics
```

Out[ ]:

	a	c	d
count	5.0	5.0	5.0
mean	1.0	1.0	3.0
std	0.0	0.0	0.0
min	1.0	1.0	3.0
25%	1.0	1.0	3.0
50%	1.0	1.0	3.0
75%	1.0	1.0	3.0
max	1.0	1.0	3.0

In [ ]:
```
df2.T #transpose
```

Out[ ]:

	0	1	2	3	4
a	1.0	1.0	1.0	1.0	1.0
b	2015-01-02 00:00:00	2015-01-02 00:00:00	2015-01-02 00:00:00	2015-01-02 00:00:00	2015-01-02 00:00:00
c	1.0	1.0	1.0	1.0	1.0
d	3	3	3	3	3
e	girl	women	men	boy	child
f	Gender	Gender	Gender	Gender	Gender

In [ ]:
```
df2.sort_index(axis=0, ascending=True) #ascending and decsing order
```

Out[ ]:

	a	b	c	d	e	f
0	1.0	2015-01-02	1.0	3	girl	Gender
1	1.0	2015-01-02	1.0	3	women	Gender
2	1.0	2015-01-02	1.0	3	men	Gender
3	1.0	2015-01-02	1.0	3	boy	Gender

	a	b	c	d	e	f
4	1.0	2015-01-02	1.0	3	child	Gender

In [ ]:
```
##importing dataset
data=pd.read_csv("Chilla_data2.csv")
data.head()
```

In [ ]:
```
data.sort_index(axis=1, ascending=False)
```

Out[ ]:

	field_of_study	Your favorite programming language?	Your Weight in kg? (float)	Which mobile sim do you use	Which PC are you using?	Where do you live?	When did you write your first line of code?	What are you?	Research/Working experience (Float/Int) years	Qualification_completed	...	Location	Light kitnider bandhtihy? int	How many hours you code a day? (int) e.g: 5,4,3	Height in cm? Freelancer- (Float)
0	Natural Sciences	Python	77.0	U-fone	Apna Laptop	Urbun	>3 years ago	Unemplyed	5	Masters	...	Pakistan	2.0	3.0	179.000
1	CS/IT	Python	53.6	U-fone	Apna Laptop	Urbun	>3 years ago	Student	1	Bachelors	...	Pakistan	6.0	2.0	178.000
2	Enginnering	Python	93.0	Zong	Apna Laptop	Urbun	>3 years ago	Employed	5.5	Masters	...	Pakistan	0.0	2.0	173.000
3	CS/IT	Python	60.0	U-fone	Apna Laptop	Urbun	<3 years ago	Employed	5	Masters	...	Pakistan	24.0	3.0	157.000
4	Enginnering	Javascript	59.9	Mobilink	Apna Laptop	Rural	In this chilla	Student	3.5	Masters	...	Pakistan	12.0	6.0	164.544
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
370	Enginnering	R	70.5	Telenor	Apna Laptop	Rural	In this chilla	Employed	7	Masters	...	Pakistan	3.0	4.0	178.500
371	Enginnering	Python	83.4	Zong	Apna Laptop	Urbun	>3 years ago	Employed	5	Bachelors	...	Pakistan	1.0	1.0	172.700
372	CS/IT	Python	60.0	Mobilink	Apna Laptop	Urbun	>3 years ago	Employed	0	Bachelors	...	Pakistan	0.0	0.0	1.680
373	Enginnering	Python	86.0	Mobilink	Apna Laptop	Urbun	<3 years ago	Employed	2	Masters	...	Pakistan	1.0	2.0	180.000
374	Mathematics	Python	54.5	Telenor	Apna Laptop	Urbun	In this chilla	Unemplyed	3	Masters	...	Pakistan	0.0	3.0	161.544

375 rows × 23 columns

In [ ]:
```
data.T #transpose on chilla_class data
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	365	366
Gender	Male	Male	Male	Female	Female	Male	Male	Male	Male	Male	...	Male	Female
Location	Pakistan	Pakistan	Pakistan	Pakistan	Pakistan	Pakistan	Pakistan	West	Pakistan	West	...	West	Pakistan
Age	36-40	26-30	31-35	31-35	26-30	36-40	21-25	26-30	21-25	21-25	...	26-30	21-25
Qualification_completed	Masters	Bachelors	Masters	Masters	Masters	Masters	Masters	Bachelors	Bachelors	Bachelors	...	Masters	Masters
field_of_study	Natural Sciences	CS/IT	Enginnering	CS/IT	Enginnering	Others	Enginnering	CS/IT	CS/IT	Natural Sciences	...	Enginnering	Others
Purpose_for_chilla	to boost my skill set	to boost my skill set	Switch my field of study	to boost my skill set	to boost my skill set	Switch my field of study	to boost my skill set	to boost my skill set	to boost my skill set	to boost my skill set	...	to boost my skill set	Switch my field of study
What are you?	Unemplyed	Student	Employed	Employed	Student	Employed	Student	Student	Unemplyed	Student	...	Student	Unemplyed
Blood group	B+	B+	B+	O+	A-	B+	O+	AB+	O+	B+	...	B+	B+
Which mobile sim do you use	U-fone	U-fone	Zong	U-fone	Mobilink	Mobilink	Mobilink	International	Zong	Mobilink	...	International	U-fone
Prepaid or Postpaid	Prepaid	Prepaid	Prepaid	Postpaid	Prepaid	Prepaid	Prepaid	Prepaid	Prepaid	Prepaid	...	Prepaid	Prepaid
Which PC are you using?	Apna Laptop	Apna Laptop	Apna Laptop	Apna Laptop	Apna Laptop	Apna Laptop	Kisi ka laptop	Apna Laptop	Apna Laptop	Apna Laptop	...	Apna Laptop	Kisi ka laptop
Do you wear glasses?	No	No	Yes	No	Yes	Yes	Yes	No	No	No	...	No	No
When did you write your first line of code?	>3 years ago	>3 years ago	>3 years ago	<3 years ago	In this chilla	<3 years ago	In this chilla	>3 years ago	>3 years ago	<3 years ago	...	>3 years ago	In this chilla
Your favorite programming language?	Python	Python	Python	Python	Javascript	Javascript	Python	Python	Python	Python	...	Python	Python
Marital Status?	Yes	No	Yes	Yes	No	Yes	No	Yes	No	No	...	No	No

	0	1	2	3	4	5	6	7	8	9	...	365	366
Are you Vaccinated?	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	...	Yes	Yes
Where do you live?	Urbun	Urbun	Urbun	Urbun	Rural	Urbun	Urbun	International	Rural	International	...	International	Urbun
Research/Working experience (Float/Int) years	5	1	5.5	5	3.5	15	0	4	0.5	0.5	...	1.5	0
Age (years)-Float/Int	38.0	25.0	31.34	33.0	27.0	36.0	0.0	27.0	22.2	24.0	...	28.0	22.0
Your Weight in kg? (float)	77.0	53.6	93.0	60.0	59.9	99.0	58.0	100.0	76.8	84.0	...	85.0	48.0
Height in cm? Freelancer- (Float)	179.0	178.0	173.0	157.0	164.544	178.0	180.34	187.0	1.75	183.0	...	178.0	160.0
How many hours you code a day? (int) e.g: 5,4,3	3.0	2.0	2.0	3.0	6.0	1.0	1.0	3.0	3.0	3.0	...	4.0	3.0
Light kitni der band hti hy? int	2.0	6.0	0.0	24.0	12.0	0.0	2.0	0.0	0.0	0.0	...	0.0	6.0

23 rows × 375 columns

In [ ]:
```
#row wise selection
data[0:4]
```

Out[ ]:

	Gender	Location	Age	Qualification_completed	field_of_study	Purpose_for_chilla	What are you?	Blood group	Which mobile sim do you use	Prepaid or Postpaid	...	Your favorite programming language?	Marital Status?	Are you Vaccinated?	Where do you live?
0	Male	Pakistan	36-40	Masters	Natural Sciences	to boost my skill set	Unemplyed	B+	U-fone	Prepaid	...	Python	Yes	Yes	Urbun
1	Male	Pakistan	26-30	Bachelors	CS/IT	to boost my skill set	Student	B+	U-fone	Prepaid	...	Python	No	Yes	Urbun
2	Male	Pakistan	31-35	Masters	Enginnering	Switch my field of study	Employed	B+	Zong	Prepaid	...	Python	Yes	Yes	Urbun
3	Female	Pakistan	31-35	Masters	CS/IT	to boost my skill set	Employed	O+	U-fone	Postpaid	...	Python	Yes	Yes	Urbun

4 rows × 23 columns

In [ ]:
```
data.loc [:,["Age", "Location"]] #columnwisw section
```

Out[ ]:

	Age	Location
0	36-40	Pakistan
1	26-30	Pakistan
2	31-35	Pakistan
3	31-35	Pakistan
4	26-30	Pakistan
...	...	...
370	26-30	Pakistan
371	31-35	Pakistan
372	21-25	Pakistan
373	26-30	Pakistan
374	31-35	Pakistan

375 rows × 2 columns

In [ ]:
```
##importing  FAO dataset
LU=pd.read_csv("FAO_LU_comparison.csv")
LU.head(4)
```

Out[ ]:

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
0	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1961	1961	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
1	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1962	1962	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
2	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1963	1963	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
3	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1964	1964	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...

In [ ]:
```
LU.dtypes
```

Out[ ]:
```
Domain Code          object
Domain               object
Area Code (FAO)       int64
Area                 object
Element Code          int64
Element              object
Item Code             int64
Item                 object
Year Code             int64
Year                  int64
Unit                 object
Value               float64
Flag                 object
Flag Description     object
dtype: object
```

In [ ]:
```
LU.loc [:,["Area", "Item"]]
```

Out[ ]:

	Area	Item
0	Cameroon	Country area
1	Cameroon	Country area
2	Cameroon	Country area
3	Cameroon	Country area
4	Cameroon	Country area
...	...	...
2152	Pakistan	Agriculture area actually irrigated
2153	Pakistan	Agriculture area actually irrigated
2154	Pakistan	Agriculture area actually irrigated
2155	Pakistan	Agriculture area actually irrigated
2156	Pakistan	Agriculture area actually irrigated

2157 rows × 2 columns

In [ ]:
```
LU.iloc[3] #specific position data
```

Out[ ]:
```
Domain Code                                                    RL
Domain                                                  Land Use
Area Code (FAO)                                               32
Area                                                    Cameroon
Element Code                                                5110
Element                                                     Area
Item Code                                                   6600
Item                                                Country area
Year Code                                                   1964
Year                                                        1964
Unit                                                     1000 ha
Value                                                    47544.0
Flag                                                           Q
Flag Description    Official data reported on FAO Questionnaires f...
Name: 3, dtype: object
```

In [ ]:
```
LU.iloc[0:5,0:3] #rows to column data filter
```

Out[ ]:

	Domain Code	Domain	Area Code (FAO)
0	RL	Land Use	32
1	RL	Land Use	32
2	RL	Land Use	32
3	RL	Land Use	32
4	RL	Land Use	32

In [ ]:
```
LU[LU["Year"]>2000] #Boolean operators in pandas
```

Out[ ]:

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
40	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	2001	2001	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
41	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	2002	2002	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
42	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	2003	2003	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
43	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	2004	2004	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
44	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	2005	2005	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2152	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2015	2015	1000 ha	18730.0	Q	Official data reported on FAO Questionnaires f...
2153	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2016	2016	1000 ha	18601.0	Q	Official data reported on FAO Questionnaires f...
2154	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2017	2017	1000 ha	18220.0	Q	Official data reported on FAO Questionnaires f...
2155	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2018	2018	1000 ha	19320.0	W	Data reported on country official publications...
2156	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2019	2019	1000 ha	19320.0	W	Data reported on country official publications...

825 rows × 14 columns

In [ ]:
```
LU_PAK=LU[LU["Area"]=="Pakistan"]
LU_PAK
```

Out[ ]:

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
1694	RL	Land Use	165	Pakistan	5110	Area	6600	Country area	1961	1961	1000 ha	79610.0	Q	Official data reported on FAO Questionnaires f...
1695	RL	Land Use	165	Pakistan	5110	Area	6600	Country area	1962	1962	1000 ha	79610.0	Q	Official data reported on FAO Questionnaires f...
1696	RL	Land Use	165	Pakistan	5110	Area	6600	Country area	1963	1963	1000 ha	79610.0	Q	Official data reported on FAO Questionnaires f...
1697	RL	Land Use	165	Pakistan	5110	Area	6600	Country area	1964	1964	1000 ha	79610.0	Q	Official data reported on FAO Questionnaires f...
1698	RL	Land Use	165	Pakistan	5110	Area	6600	Country area	1965	1965	1000 ha	79610.0	Q	Official data reported on FAO Questionnaires f...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2152	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2015	2015	1000 ha	18730.0	Q	Official data reported on FAO Questionnaires f...
2153	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2016	2016	1000 ha	18601.0	Q	Official data reported on FAO Questionnaires f...
2154	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2017	2017	1000 ha	18220.0	Q	Official data reported on FAO Questionnaires f...
2155	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2018	2018	1000 ha	19320.0	W	Data reported on country official publications...
2156	RL	Land Use	165	Pakistan	5110	Area	6611	Agriculture area actually irrigated	2019	2019	1000 ha	19320.0	W	Data reported on country official publications...

463 rows × 14 columns

In [ ]:
```
LU[LU["Year"]<2000]
```

Out[ ]:

	Domain Code	Domain	Area Code (FAO)	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
0	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1961	1961	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
1	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1962	1962	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
2	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1963	1963	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
3	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1964	1964	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
4	RL	Land Use	32	Cameroon	5110	Area	6600	Country area	1965	1965	1000 ha	47544.0	Q	Official data reported on FAO Questionnaires f...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2113	RL	Land Use	165	Pakistan	5110	Area	6680	Inland waters	1995	1995	1000 ha	2522.0	Fm	Manual Estimation
2114	RL	Land Use	165	Pakistan	5110	Area	6680	Inland waters	1996	1996	1000 ha	2522.0	Fm	Manual Estimation
2115	RL	Land Use	165	Pakistan	5110	Area	6680	Inland waters	1997	1997	1000 ha	2522.0	Fm	Manual Estimation
2116	RL	Land Use	165	Pakistan	5110	Area	6680	Inland waters	1998	1998	1000 ha	2522.0	Fm	Manual Estimation
2117	RL	Land Use	165	Pakistan	5110	Area	6680	Inland waters	1999	1999	1000 ha	2522.0	Fm	Manual Estimation

1290 rows × 14 columns

In [ ]:

```
mean1 = LU['Value'].mean() #mean function
mean1
```

Out[ ]:
```
19435.188111914635
```

In [ ]:
```
mean2 = LU_PAK['Value'].mean() #mean function
mean2
```

Out[ ]:
```
34583.710480561575
```

FAO data analysis

Land Use and Land Cover

LULC Analysis

1- Land Use and Land Cover:

The terms land use and land cover are often used interchangeably, but each term has its own unique meaning. Land cover refers to the surface cover on the ground like vegetation, urban infrastructure, water, bare soil etc. Identification of land cover establishes the baseline information for activities like thematic mapping and change detection analysis. Land use refers to the purpose the land serves, for example, recreation, wildlife habitat, or agriculture.

1.1- Importance

- LULC maps play a significant and prime role in planning, management and monitoring programmes at local, regional and national levels.
- LULC maps also help us to study the changes that are happening in our ecosystem and environment.
- Wildlife habitat protection and many more

In [ ]:
```
pip install plotly
```

In [ ]:
```
#import libraries
import seaborn as sns
#canvas style
sns.set(style='whitegrid')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
##importing dataset
## Importing LU information of 5 countries
LUG=pd.read_csv("FAOSTAT_forest_1-13-2022.csv")
LUG.head()
```

Out[ ]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Source Code	Source	Unit	Value	Flag	Flag Description	Note
0	GF	Forests	165	Pakistan	5110	Area	6751	Forestland	1990	1990	3050	FAO TIER 1	1000 ha	4986.7900	E	Expert sources from FAO (including other divis...	NaN
1	GF	Forests	165	Pakistan	72332	Net emissions/removals (CO2) (Forest land)	6751	Forestland	1990	1990	3050	FAO TIER 1	kilotonnes	0.0000	Fc	Calculated data	NaN
2	GF	Forests	351	China	5110	Area	6751	Forestland	1990	1990	3050	FAO TIER 1	1000 ha	157140.5900	A	Aggregate, may include official, semi-official...	NaN
3	GF	Forests	351	China	72332	Net emissions/removals (CO2) (Forest land)	6751	Forestland	1990	1990	3050	FAO TIER 1	kilotonnes	-350983.5047	A	Aggregate, may include official, semi-official...	NaN
4	GF	Forests	351	China	72332	Net emissions/removals (CO2) (Forest land)	6751	Forestland	1991	1991	3050	FAO TIER 1	kilotonnes	-350983.5047	A	Aggregate, may include official, semi-official...	NaN

In [ ]:
```
##importing dataset
## Importing LU information of 5 countries
LU=pd.read_csv("FAO_ALL_LU.csv")
#LU.head()
```

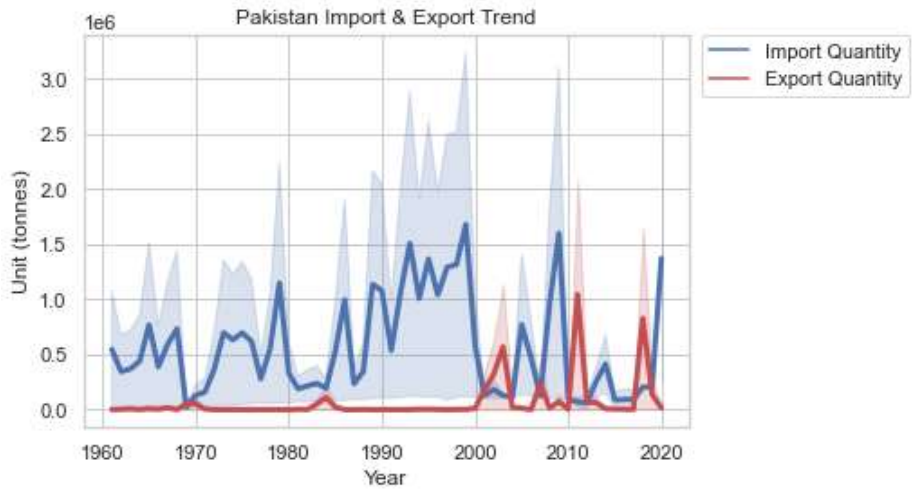
In [ ]:
```
IE=pd.read_csv("FAOSTAT_import_1-13-2022.csv")
IE.head()
```

Out[ ]:

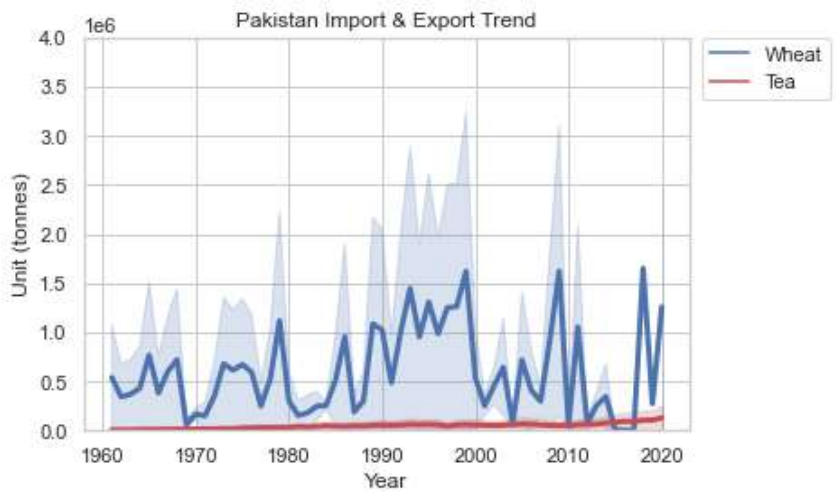
	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
0	TCL	Crops and livestock products	165	Pakistan	5610	Import Quantity	15	Wheat	1961	1961	tonnes	1078899	NaN	Official data
1	TCL	Crops and livestock products	165	Pakistan	5910	Export Quantity	15	Wheat	1961	1961	tonnes	432	NaN	Official data

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
2	TCL	Crops and livestock products	165	Pakistan	5610	Import Quantity	667	Tea	1961	1961	tonnes	16056	NaN	Official data
3	TCL	Crops and livestock products	165	Pakistan	5910	Export Quantity	667	Tea	1961	1961	tonnes	0	NaN	Official data
4	TCL	Crops and livestock products	165	Pakistan	5610	Import Quantity	15	Wheat	1962	1962	tonnes	677219	NaN	Official data

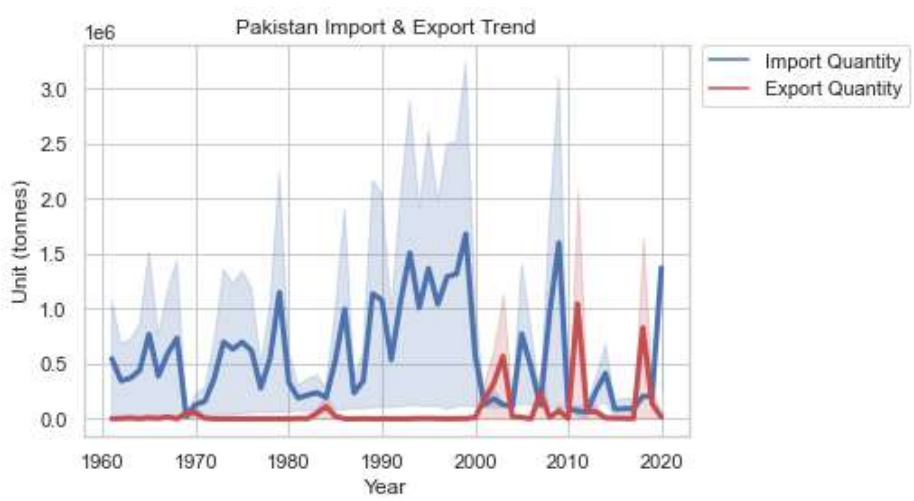
In [ ]:
```
sns.lineplot(data=IE, x="Year", y="Value", hue="Element", linewidth=3,
             palette={"Import Quantity": "b", "Export Quantity": "r"}
              ).set(title='Pakistan Import & Export Trend')
plt.xlabel("Year")
plt.ylabel("Unit (tonnes)")
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
plt.show()
```

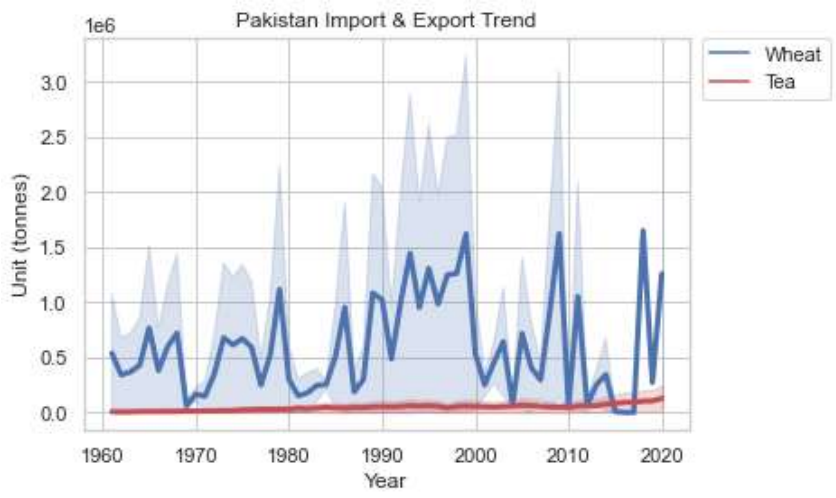


In [ ]:
```
sns.lineplot(data=IE, x="Year", y="Value", hue="Item", linewidth=3,
             palette={"Wheat": "b", "Tea": "r"}
              ).set(title='Pakistan Import & Export Trend')
plt.xlabel("Year")
plt.ylabel("Unit (tonnes)")
plt.ylim(0, 4000000)
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
plt.show()
```



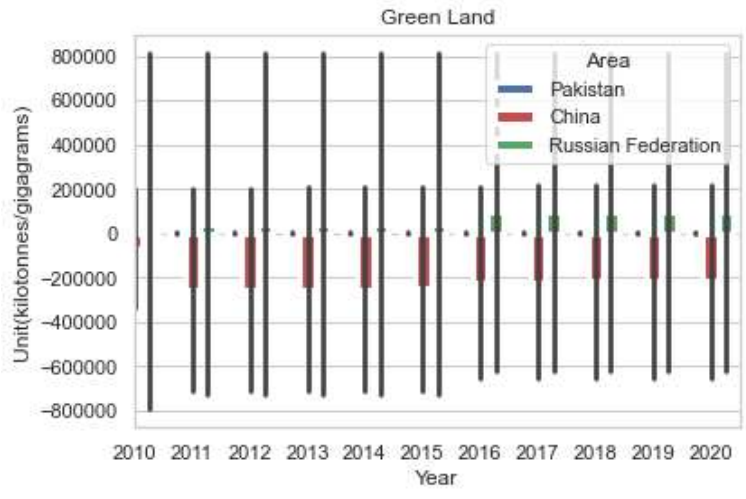
In [ ]:
```
sns.lineplot(data=IE, x="Year", y="Value", hue="Element", linewidth=3,
             palette={"Import Quantity": "b", "Export Quantity": "r"}
              ).set(title='Pakistan Import & Export Trend')
plt.xlabel("Year")
plt.ylabel("Unit (tonnes)")
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
fig, axs = plt.subplots(ncols=1)

sns.lineplot(data=IE, x="Year", y="Value", hue="Item", linewidth=3,
             palette={"Wheat": "b", "Tea": "r"}
              ).set(title='Pakistan Import & Export Trend')
plt.xlabel("Year")
plt.ylabel("Unit (tonnes)")
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
plt.show()
```





In [ ]:

```
sns.barplot(data=LUG, x="Year", y="Value",saturation=1, hue="Area",
            linewidth=3,
            palette={"Pakistan": "b", "China": "r","Russian Federation":"g"}
             ).set(title='Green Land')

plt.xlabel("Year")
plt.ylabel("Unit(kilotonnes/gigagrams)")
plt.xlim(20)
plt.show()
```



Pandas Library practice

In [ ]:

```
##importing dataset
## Importing LU information of 5 countries
LU=pd.read_csv("FAO_LU_comparison.csv")
#LU.head()
```

In [ ]:

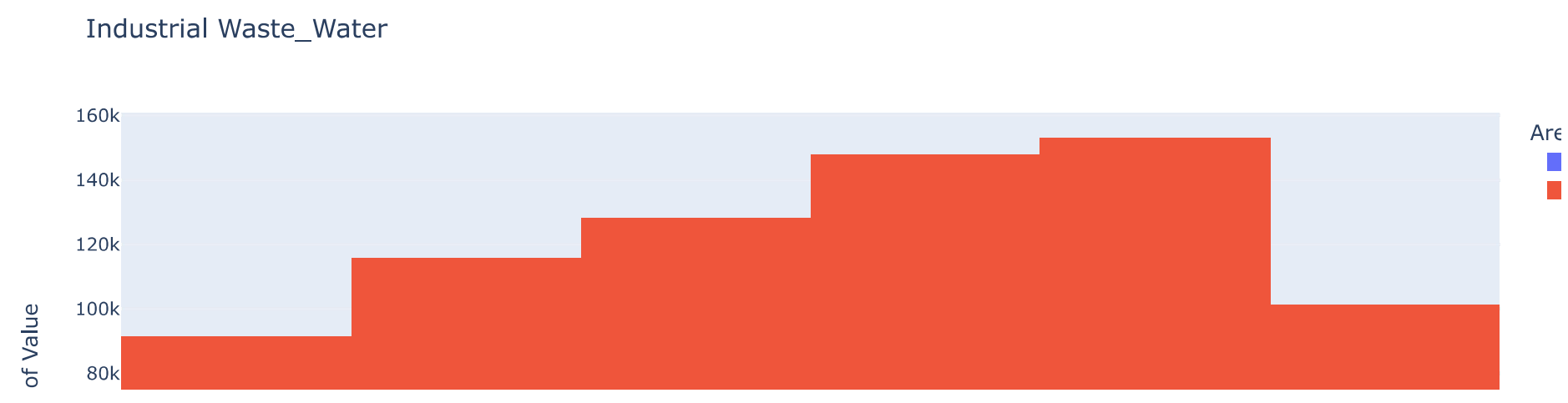
```
LUW=pd.read_csv("FAOSTAT_wastewater_1-13-2022.csv")
LUW.head()
```

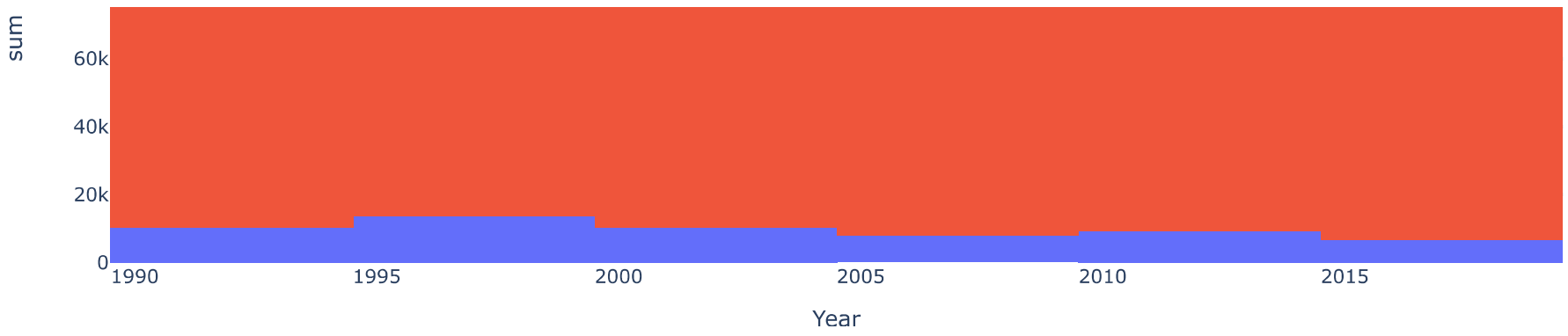
Out[ ]:

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value	Flag	Flag Description
0	GW	Waste Disposal	165	Pakistan	7225	Emissions (CH4)	6989	Industrial wastewater	1990	1990	kilotonnes	30.170089	Fc	Calculated data
1	GW	Waste Disposal	165	Pakistan	723112	Emissions (CO2eq) (SAR)	6989	Industrial wastewater	1990	1990	gigagrams	771.065186	Fc	Calculated data
2	GW	Waste Disposal	165	Pakistan	723114	Emissions (CO2eq) (AR4)	6989	Industrial wastewater	1990	1990	gigagrams	886.423221	Fc	Calculated data
3	GW	Waste Disposal	351	China	7225	Emissions (CH4)	6989	Industrial wastewater	1990	1990	kilotonnes	272.908754	A	Aggregate, may include official, semi-official...
4	GW	Waste Disposal	351	China	723112	Emissions (CO2eq) (SAR)	6989	Industrial wastewater	1990	1990	gigagrams	6687.476069	A	Aggregate, may include official, semi-official...

In [ ]:

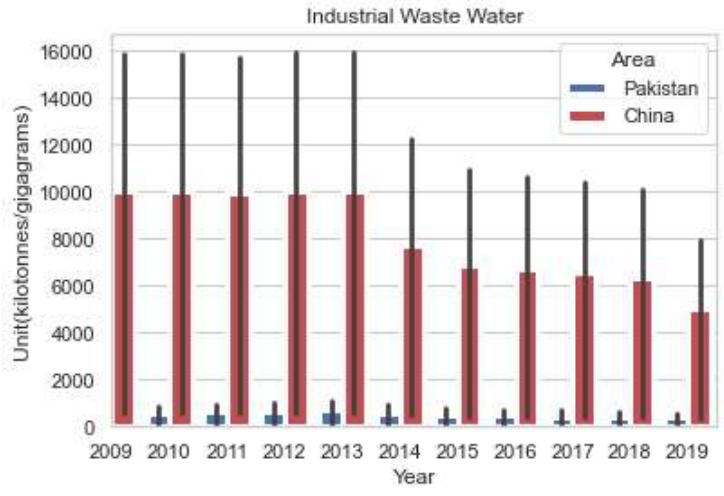
```
import plotly.express as px
#df = px.data.tips()
fig = px.histogram(LUW, x="Year", y="Value", color="Area",
                   title="Industrial Waste_Water")
fig.show()
```





In [ ]:

```
sns.barplot(data=LUW, x="Year", y="Value",saturation=1, hue="Area",
            linewidth=3,
            palette={"Pakistan": "b", "China": "r"}
             ).set(title='Industrial Waste Water')

plt.xlabel("Year")
plt.ylabel("Unit(kilotonnes/gigagrams)")
plt.xlim(19)
plt.show()
```

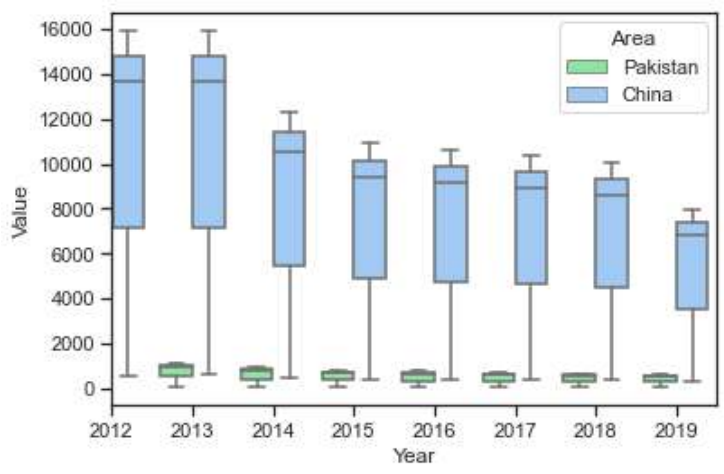


In [ ]:

```
import seaborn as sns
sns.set_theme(style="ticks", palette="pastel")

# Load the example tips datase

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="Year", y="Value",
            hue="Area", palette=["g","b"],saturation=1,
            data=LUW)
#sns.despine(offset=20, trim=True)
plt.xlim(22)
```

Out[ ]:

```
(22.0, 29.5)
```



In [ ]:

```
##importing dataset
# Pakistan LC data
LC=pd.read_csv("LC.csv")
#LC
```

In [ ]:

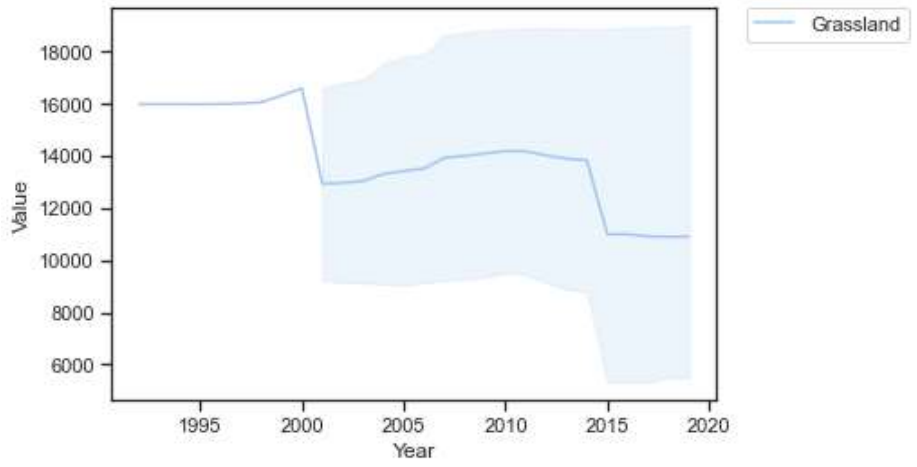
```
##importing grassland dataset of Pakistan
LG=pd.read_csv("Grassland_Pak.csv")
#LG.head()
```

Grassland of Pakistan area is getting decrease with the time

Timeseries map of grassland

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#plot multiple time series
sns.lineplot(x='Year', y='Value', hue='Item', data=LG)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```

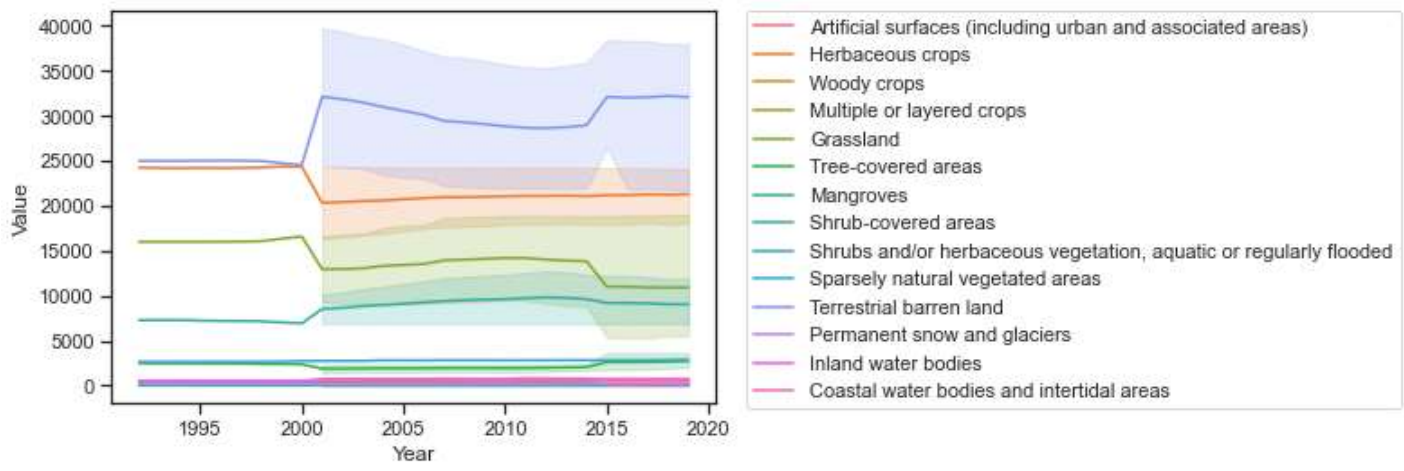


In [ ]:
```
##importing grassland dataset
LUA=pd.read_csv("FAO_ALL_LU.csv")
#LUA.head()
```

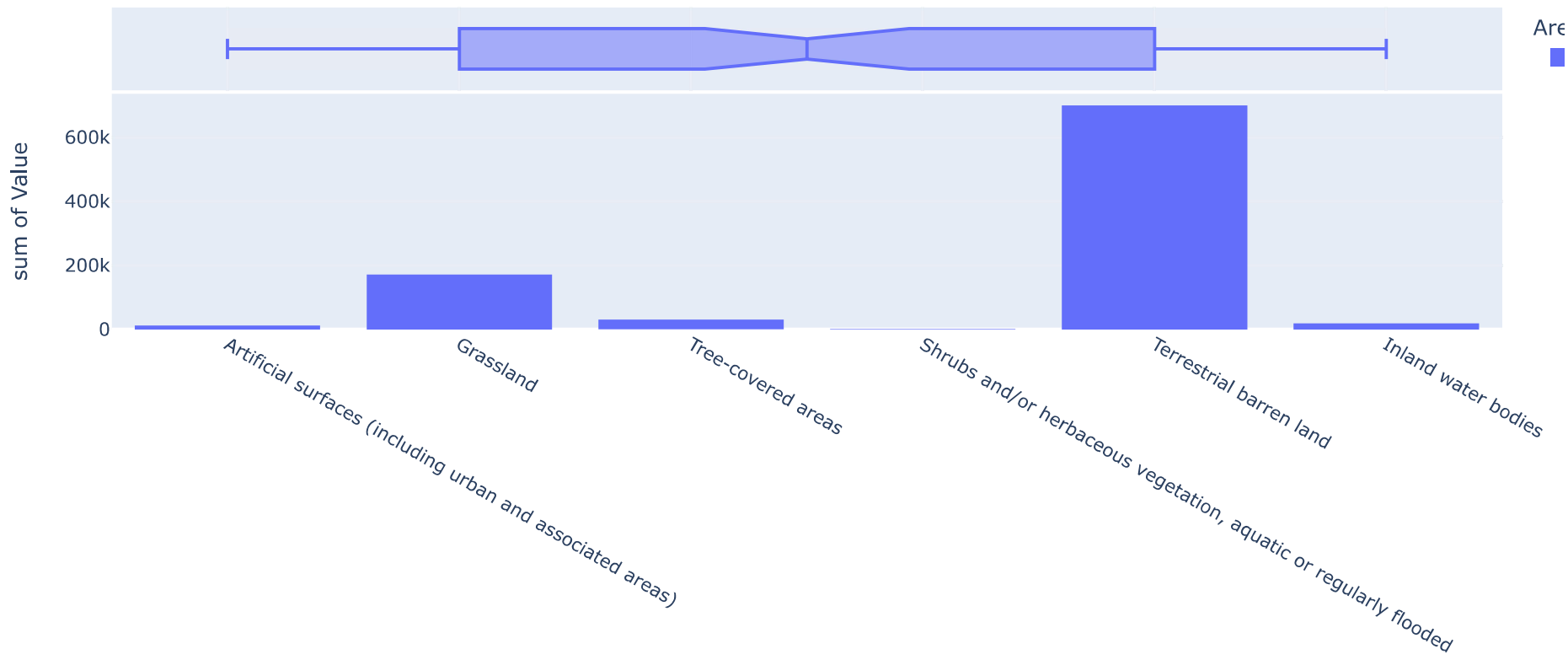
Land Use all calsses trend in Pakistan

* Time, types of classes and value analysis

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

#plot multiple time series
sns.lineplot(x='Year', y='Value', hue='Item', data=LUA)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
plt.show()
```



In [ ]:
```
import plotly.express as px #Some selected items
LC=pd.read_csv("LC.csv")
fig = px.histogram(LC, x="Item", y="Value", color="Area",
                   marginal="box", # or violin, rug
                   hover_data=LC.columns)
fig.show()
```



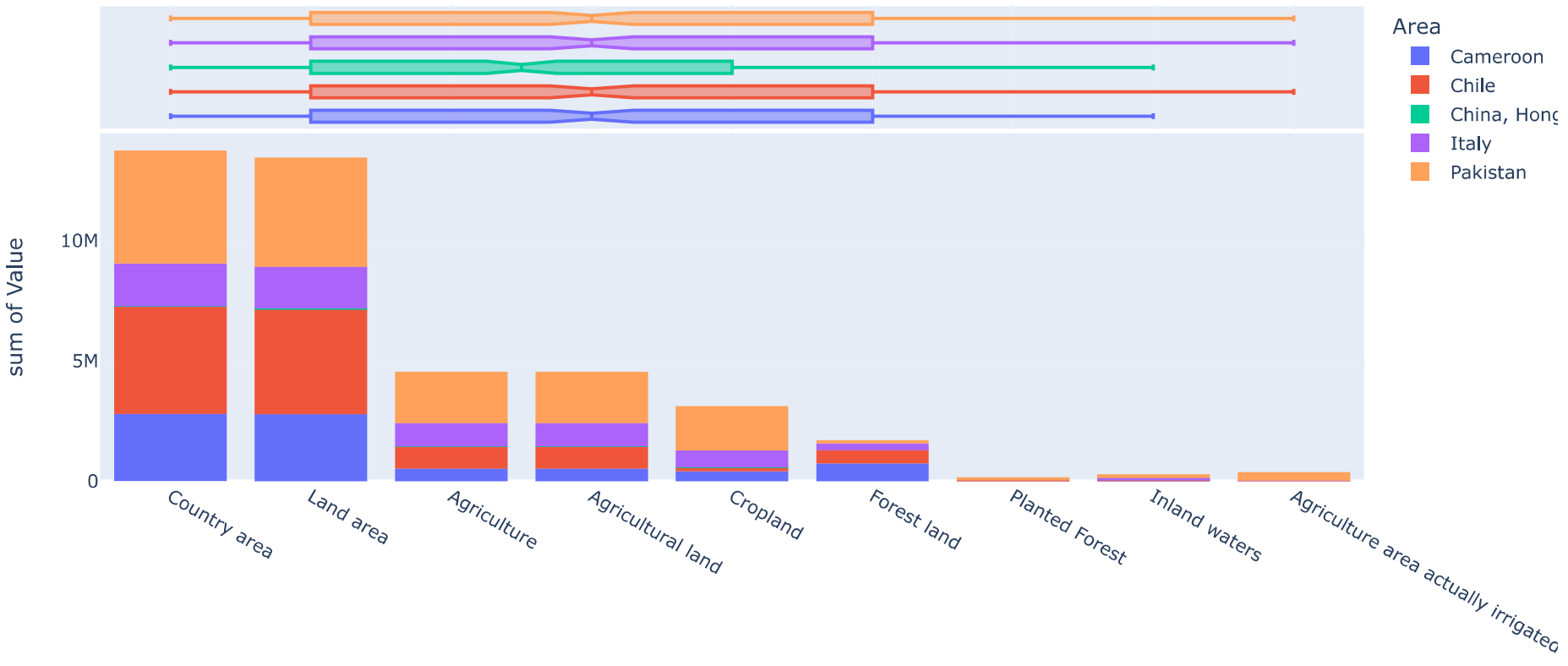
Compasrion with other countries

each country has specific value for every LU class

How to run plotly graph in VS code

In [ ]:
```
import plotly.io as pio
#pio.renderers.default = "vscode"
pio.renderers.default = "notebook_connected"
```

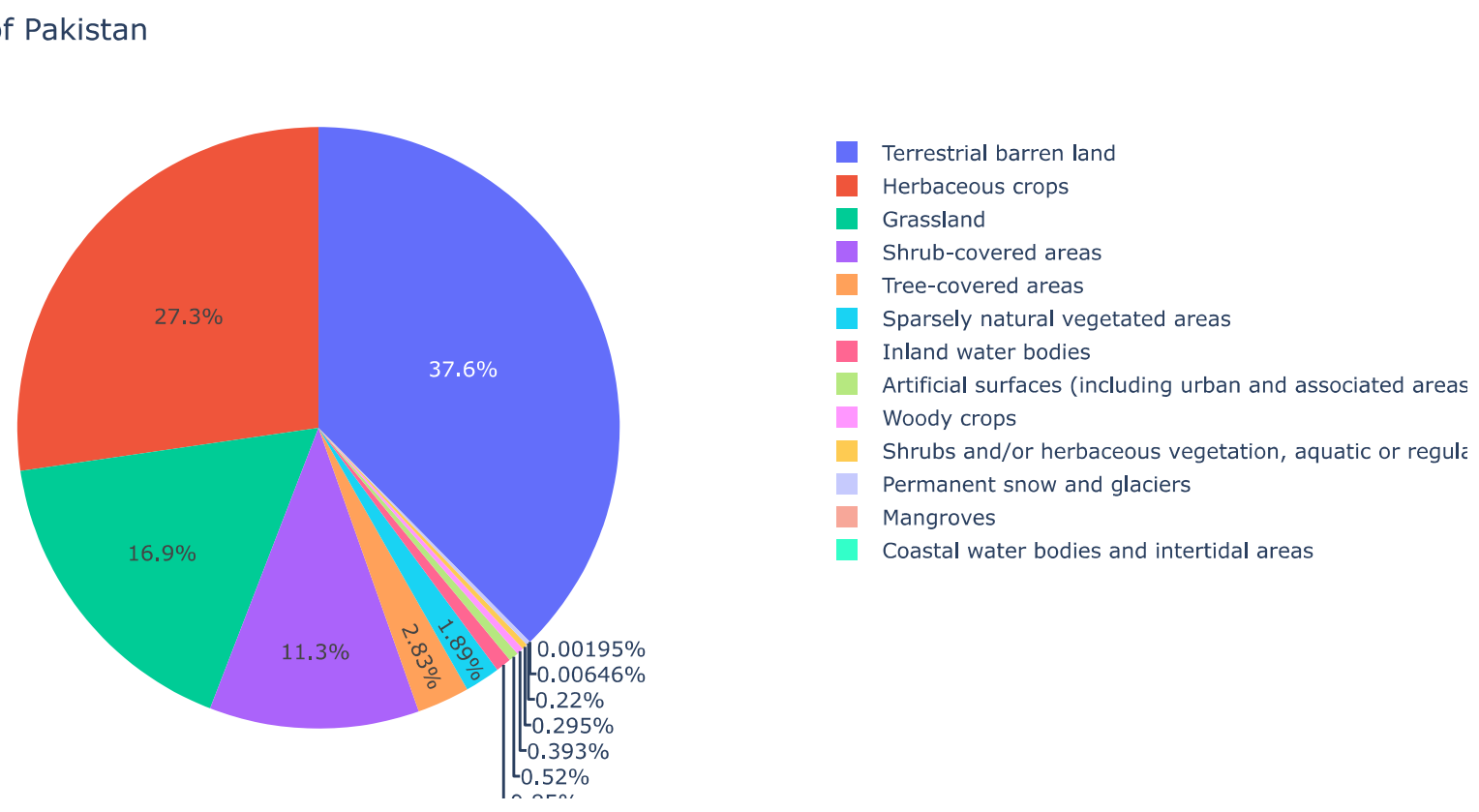
In [ ]:
```
import plotly.express as px
LU=pd.read_csv("FAO_LU_comparison.csv")
fig = px.histogram(LU, x="Item", y="Value", color="Area",
                   marginal="box", # or violin, rug
                   hover_data=LU.columns)
fig.show()
```



Pie chart analysis for each Land use class

- High value for Terrestrial barren land
- Less for Mangroves, woody crops, inland water bodies etc

In [ ]:
```
import plotly.express as px
#df = px.data.gapminder().query("year == 2007").query("Area == 'Pakistan','Italy','Chile','Cameroon'")
#df.loc[df['Value'] < 2.e6, 'country'] = 'Other countries' # Represent only large countries
fig = px.pie(LUA, values='Value', names='Item', title='Total Land Use of Pakistan')
fig.show()
```

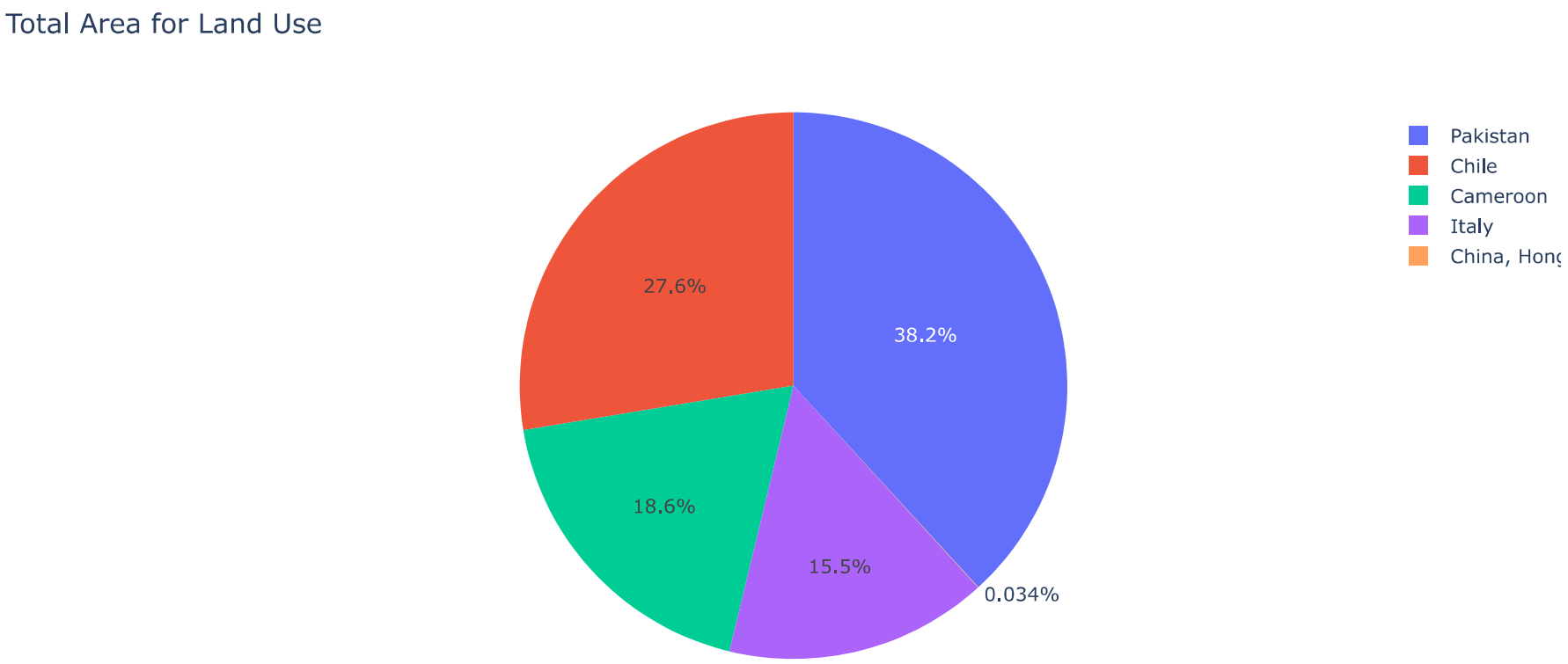


Other contries Land use

- Pakistan and Chile have more value

In [ ]:
```
```

```
import plotly.express as px
#df = px.data.gapminder().query("year == 2007").query("Area == 'Pakistan','Italy','Chile','Cameroon'")
#df.loc[df['Value'] < 2.e6, 'country'] = 'Other countries' # Represent only large countries
fig = px.pie(LU, values='Value', names='Area', title='Total Area for Land Use')
fig.show()
```



Some pandas practice on FAO data

In [ ]:
```
##importing dataset
LU=pd.read_csv("FAO_LU_comparison.csv")
#LU.head()
```

In [ ]:
```
mean1 = LU['Value'].mean() #mean function
mean1
```

Out[ ]:
```
19435.188111914635
```

In [ ]:
```
LU_PAK=LU[LU["Area"]=="Pakistan"]
#LU_PAK
```

In [ ]:
```
mean4 = LU_PAK['Value'].mean() #mean function
mean4
```

Out[ ]:
```
34583.710480561575
```

In [ ]:
```
LU_ITA=LU[LU["Area"]=="Italy"]
```

In [ ]:
```
mean2 = LU_ITA['Value'].mean() #mean function
mean2
```

Out[ ]:
```
14436.351932372472
```

In [ ]:
```
LU_Chile=LU[LU["Area"]=="Chile"]
```

In [ ]:
```
mean3 = LU_Chile['Value'].mean() #mean function
mean3
```

Out[ ]:
```
26012.186008539386
```

In [ ]:
```
LU_Ca=LU[LU["Area"]=="Cameroon"]
```

In [ ]:
```
mean5 = LU_Ca['Value'].mean() #mean function
mean5
```

Out[ ]:
```
17587.760381981974
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#create values for table
td=[
    ["Country", "Mean"],
    ["Pak", 34583.710480561575],
```

```
    ["Italy", 14436.351932372472],
    ["Chile", 26012.186008539386],
    ["Cameroon", 17587.760381981974]
]
td
#create table
#table = ax.table(cellText=table_data, loc='center')
```

Out[ ]:
```
[['Country', 'Mean'],
 ['Pak', 34583.710480561575],
 ['Italy', 14436.351932372472],
 ['Chile', 26012.186008539386],
 ['Cameroon', 17587.760381981974]]
```

In [ ]:
```
type(td)
```

Out[ ]:
```
list
```

In [ ]:
```
x=pd.DataFrame(td)
x
```

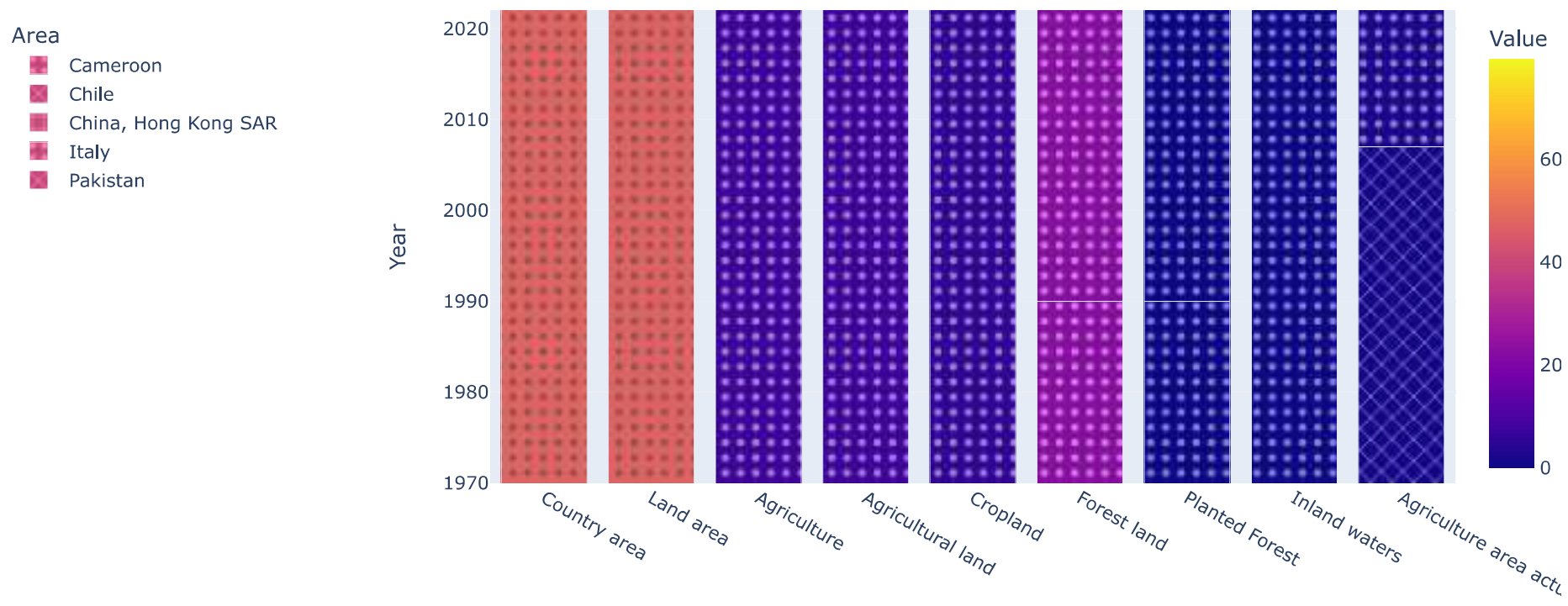
Out[ ]:

	0	1
0	Country	Mean
1	Pak	34583.710481
2	Italy	14436.351932
3	Chile	26012.186009
4	Cameroon	17587.760382

In [ ]:
```
type(x)
```

Out[ ]:
```
pandas.core.frame.DataFrame
```

In [ ]:
```
fig = px.bar(LU, x="Item", y="Year", color="Value", range_y=[1970, 2022],
             pattern_shape="Area", pattern_shape_sequence=[".", "x", "+"])
fig.update_layout(legend=dict(
    yanchor="top",
    y=0.99,
    xanchor="left",
    x=-0.5
))
fig.show()
```



Exploratory Data Analysis (EDA)

Three important steps

- Understand the data
- Clean the data
- Find a relationship between data

In [ ]:
```
# important libraries
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns
```

In [ ]:
```
kashti= sns.load_dataset('titanic')
```

In [ ]:
```
kashti.to_csv('kashti.csv')
```

In [ ]:
```
ks=kashti
```

In [ ]:
```
ks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:
```
#to look headings/type of data
ks.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [ ]:
```
#To look rows and column of data
ks.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
#give mean,std,median of numerical value
ks.describe()
```

In [ ]:
```
# to find unique value (type, categorial, numeric, etc)
ks.nunique()
```

Out[ ]:
```
survived       2
pclass         3
sex            2
age           88
sibsp          7
parch          7
fare         248
embarked       3
class          3
who            3
adult_male     2
deck           7
embark_town    3
alive          2
alone          2
dtype: int64
```

In [ ]:
```
#column names
ks.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [ ]:
```
#Unique value of specific value
ks['sex'].unique()
```

Out[ ]:
```
array(['male', 'female'], dtype=object)
```

In [ ]:
```
ks[["who","survived","age","fare"]].nunique()
```

Out[ ]:
```
who         3
survived    2
```

```
age          88
fare        248
dtype: int64
```

In [ ]:
```
pd.concat([ks['survived'],ks['age'],ks['class'],ks['survived']]).unique()
```

Out[ ]:
```
array([0, 1, 22.0, 38.0, 26.0, 35.0, nan, 54.0, 2.0, 27.0, 14.0, 4.0,
       58.0, 20.0, 39.0, 55.0, 31.0, 34.0, 15.0, 28.0, 8.0, 19.0, 40.0,
       66.0, 42.0, 21.0, 18.0, 3.0, 7.0, 49.0, 29.0, 65.0, 28.5, 5.0,
       11.0, 45.0, 17.0, 32.0, 16.0, 25.0, 0.83, 30.0, 33.0, 23.0, 24.0,
       46.0, 59.0, 71.0, 37.0, 47.0, 14.5, 70.5, 32.5, 12.0, 9.0, 36.5,
       51.0, 55.5, 40.5, 44.0, 61.0, 56.0, 50.0, 36.0, 45.5, 20.5, 62.0,
       41.0, 52.0, 63.0, 23.5, 0.92, 43.0, 60.0, 10.0, 64.0, 13.0, 48.0,
       0.75, 53.0, 57.0, 80.0, 70.0, 24.5, 6.0, 0.67, 30.5, 0.42, 34.5,
       74.0, 'Third', 'First', 'Second'], dtype=object)
```

Cleaning and filtering the data

In [ ]:
```
# find missing values inside
ks.isnull()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
887	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	True	False	False	False	False	False	False	False	True	False	False	False
889	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False

891 rows × 15 columns

In [ ]:
```
ks.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
# drop/remove whole missing value column
ks_clean= ks.drop(['deck'], axis=1)
ks_clean.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True

In [ ]:
```
ks_clean.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
ks_clean=ks_clean.dropna()
```

In [ ]:
```
ks_clean.isnull().sum()
```

Out[ ]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(712, 14)
```

In [ ]:
```
ks.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
ks_clean['sex'].value_counts()
```

Out[ ]:
```
male      453
female    259
Name: sex, dtype: int64
```

In [ ]:
```
#difference between both data before and after cleaning
ks.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

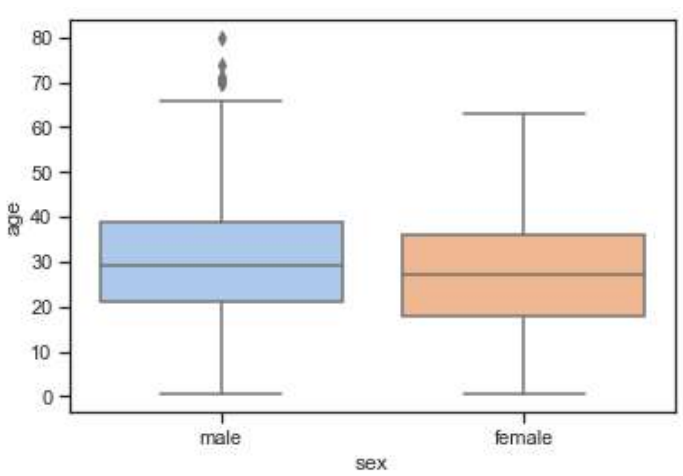
In [ ]:
```
ks_clean.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	712.000000	712.000000	712.000000	712.000000	712.000000	712.000000
mean	0.404494	2.240169	29.642093	0.514045	0.432584	34.567251
std	0.491139	0.836854	14.492933	0.930692	0.854181	52.938648
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	20.000000	0.000000	0.000000	8.050000
50%	0.000000	2.000000	28.000000	0.000000	0.000000	15.645850
75%	1.000000	3.000000	38.000000	1.000000	1.000000	33.000000
max	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

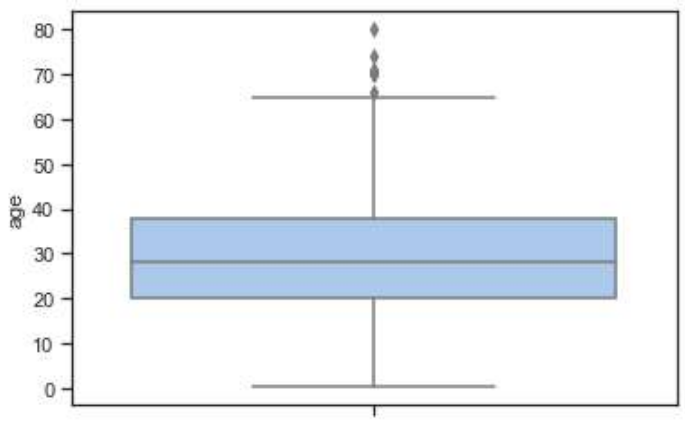
In [ ]:
```
#analysing outliers
sns.boxplot(x='sex', y='age', data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



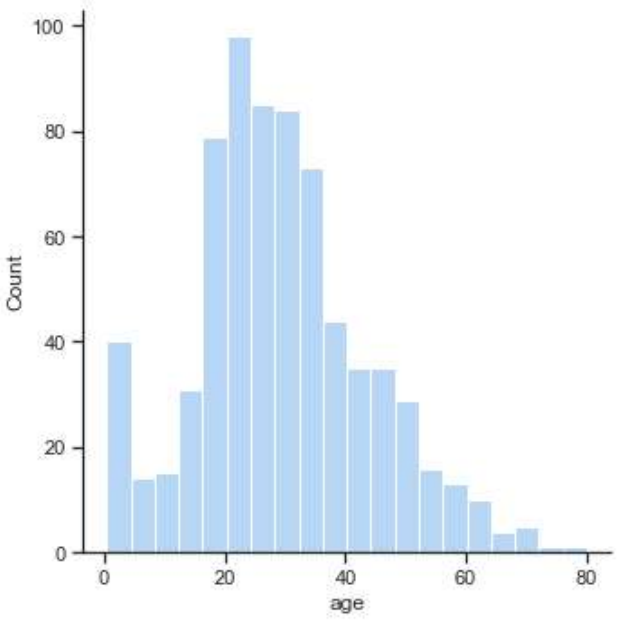
In [ ]:
```
sns.boxplot(y='age', data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:ylabel='age'>
```



In [ ]:
```
#displot or distplot to look bell curve and outlier effects
sns.displot(ks_clean['age'])
```

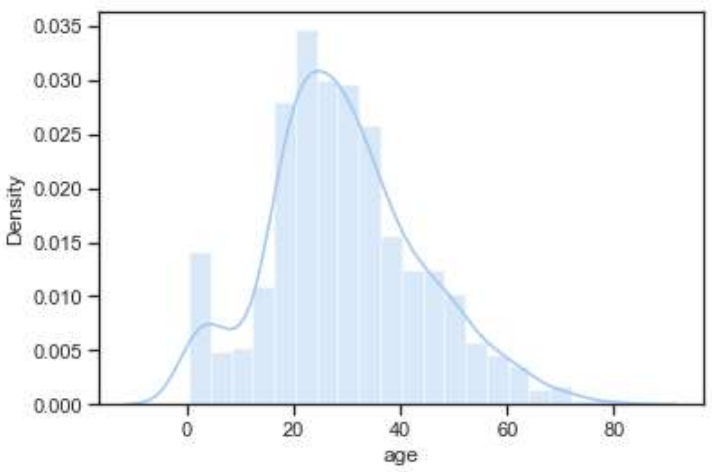
Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1bfdc4cf2e0>
```



In [ ]:
```
sns.distplot(ks_clean['age'])
```

```
C:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[ ]:
```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



In [ ]:
```
ks_clean['age'].mean()
```

Out[ ]:
```
29.64209269662921
```

In [ ]:
```
#remove outlier from specific column like giving range
ks1=ks_clean[ks_clean['age']<68]
ks1.head()
```

Out[ ]:

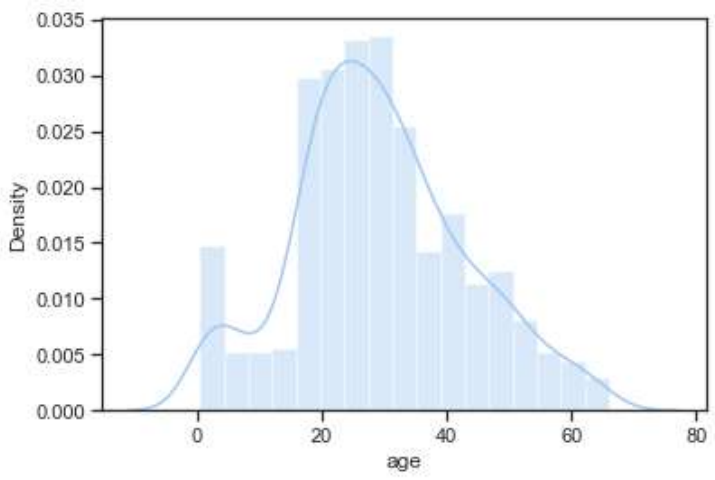
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True

In [ ]:
```
sns.distplot(ks1['age'])
```

```
C:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
```

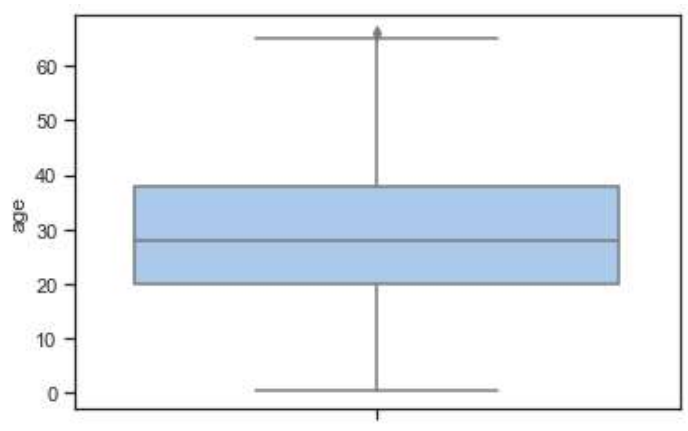
```
`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[ ]: <AxesSubplot:xlabel='age', ylabel='Density'>



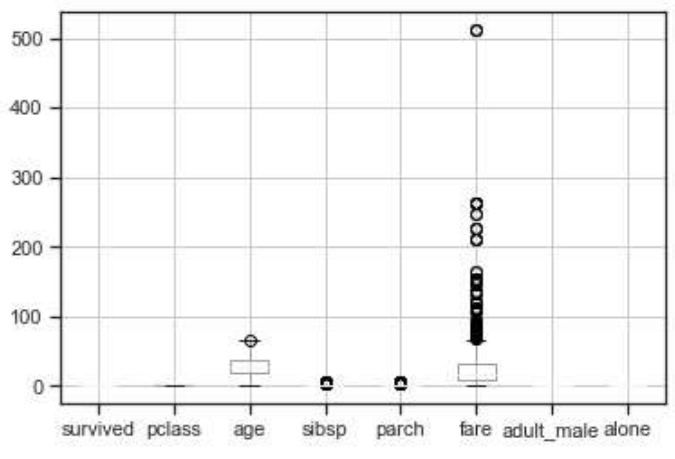
In [ ]:
```
sns.boxplot(y='age', data=ks1)
```

Out[ ]: <AxesSubplot:ylabel='age'>



In [ ]:
```
ks1.boxplot()
```

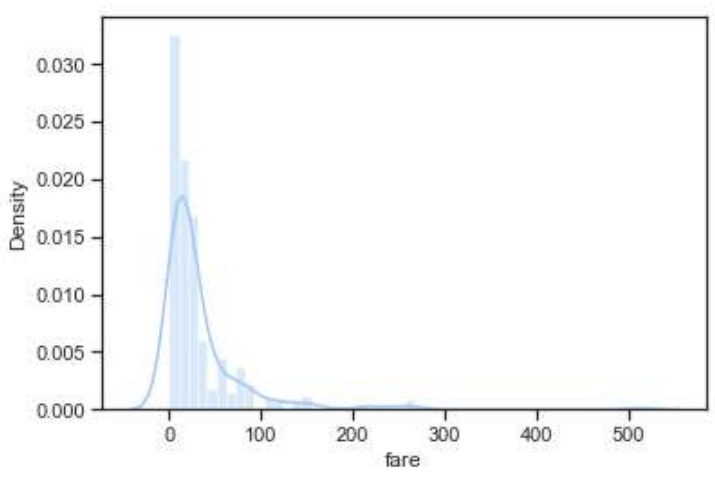
Out[ ]: <AxesSubplot:>



In [ ]:
```
sns.distplot(ks1['fare'])
```

```
C:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[ ]: <AxesSubplot:xlabel='fare', ylabel='Density'>



In [ ]:
```
#Log transformation
ks1['fare_log']=np.log(ks1['fare'])
```

```
C:\Anaconda\lib\site-packages\pandas\core\arraylike.py:364: RuntimeWarning:

divide by zero encountered in log

C:\Users\masha\AppData\Local\Temp/ipykernel_139256/4080481591.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
```

In [ ]:
```
ks1.head()
```

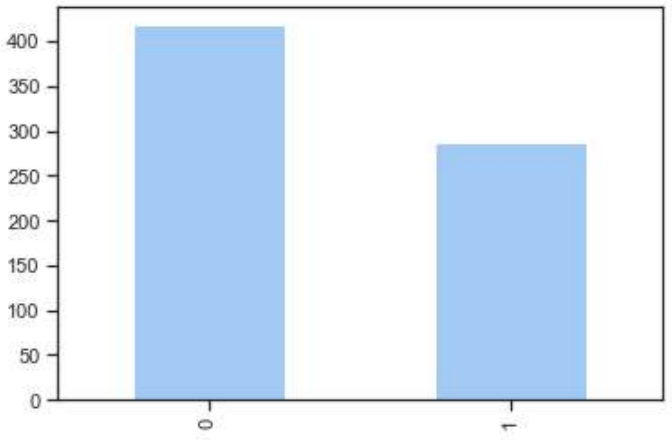
Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone	fare_log
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False	1.981001
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False	4.266662
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True	2.070022
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False	3.972177
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True	2.085672

In [ ]:
```
#histogram of whole dataset
ks1.hist()
```

In [ ]:
```
#using pandas function to draw specific bar plot
pd.value_counts(ks1['survived']).plot.bar()
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
#groupby function
ks1.groupby(['sex','class']).mean()
```

Out[ ]:

		survived	pclass	age	sibsp	parch	fare	adult_male	alone	fare_log
sex	**class**									
female	**First**	0.963855	1.0	34.240964	0.554217	0.506024	108.619680	0.000000	0.361446	4.482447
	Second	0.918919	2.0	28.722973	0.500000	0.621622	21.951070	0.000000	0.405405	2.985791
	Third	0.460784	3.0	21.750000	0.823529	0.950980	15.875369	0.000000	0.372549	2.617667
male	**First**	0.402062	1.0	39.973402	0.381443	0.340206	72.167655	0.969072	0.525773	NaN
	Second	0.153061	2.0	30.340102	0.377551	0.244898	21.221429	0.908163	0.632653	2.894890
	Third	0.151394	3.0	26.143108	0.494024	0.258964	12.197757	0.888446	0.737052	NaN

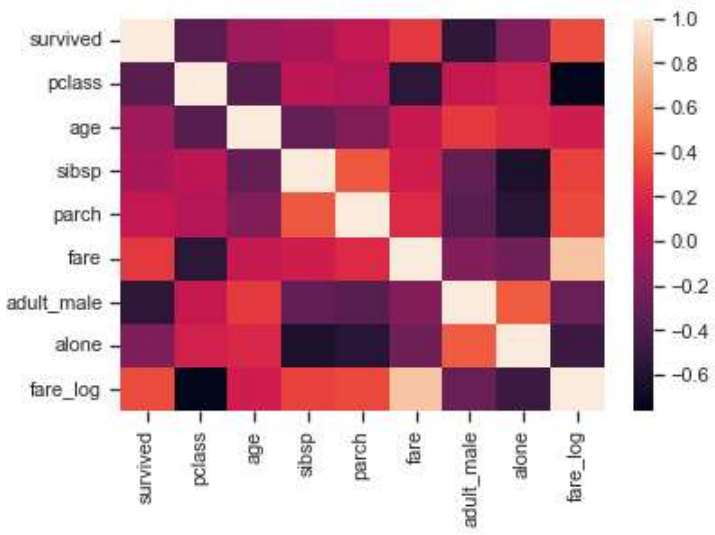
In [ ]:
```
# relationship in data
# finding co-relation in the data through matrix
ks2= ks1.corr()
ks2
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone	fare_log
survived	1.000000	-0.361441	-0.071804	-0.017289	0.094449	0.266954	-0.550780	-0.199052	0.343016
pclass	-0.361441	1.000000	-0.366032	0.064561	0.023157	-0.554566	0.101061	0.153622	-0.765251
age	-0.071804	-0.366032	1.000000	-0.309617	-0.186213	0.100263	0.274782	0.187088	0.134583
sibsp	-0.017289	0.064561	-0.309617	1.000000	0.381577	0.138697	-0.311226	-0.628019	0.304732
parch	0.094449	0.023157	-0.186213	0.381577	1.000000	0.205546	-0.364533	-0.575487	0.329916
fare	0.266954	-0.554566	0.100263	0.138697	0.205546	1.000000	-0.177542	-0.261454	0.816232
adult_male	-0.550780	0.101061	0.274782	-0.311226	-0.364533	-0.177542	1.000000	0.398833	-0.293023
alone	-0.199052	0.153622	0.187088	-0.628019	-0.575487	-0.261454	0.398833	1.000000	-0.480298
fare_log	0.343016	-0.765251	0.134583	0.304732	0.329916	0.816232	-0.293023	-0.480298	1.000000

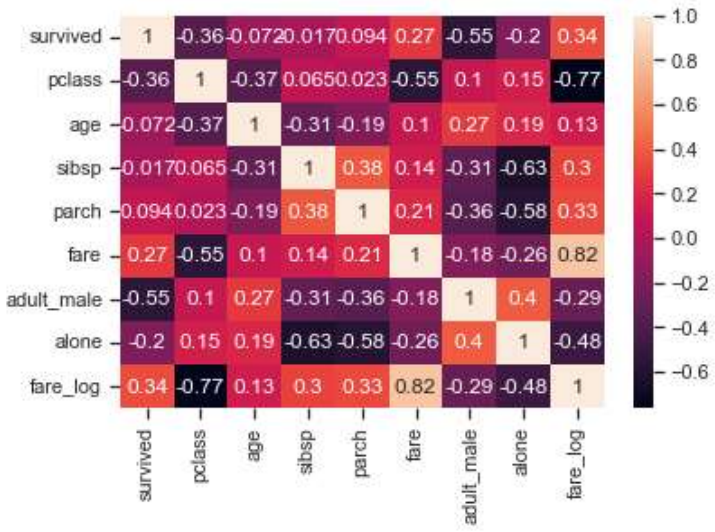
In [ ]:
```
# heatmap of correlation matrix
sns.heatmap(ks2)
```

Out[ ]:
```
<AxesSubplot:>
```



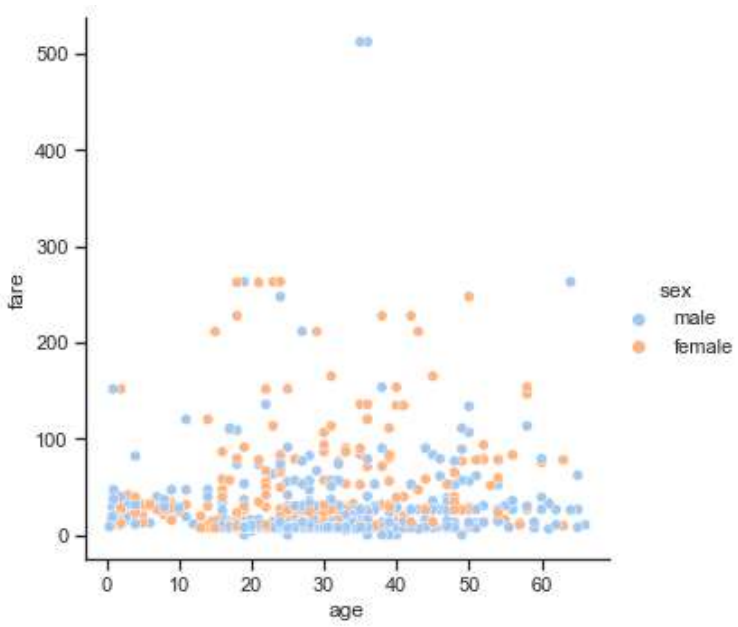
In [ ]:
```
#with annotation in the box
sns.heatmap(ks2, annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



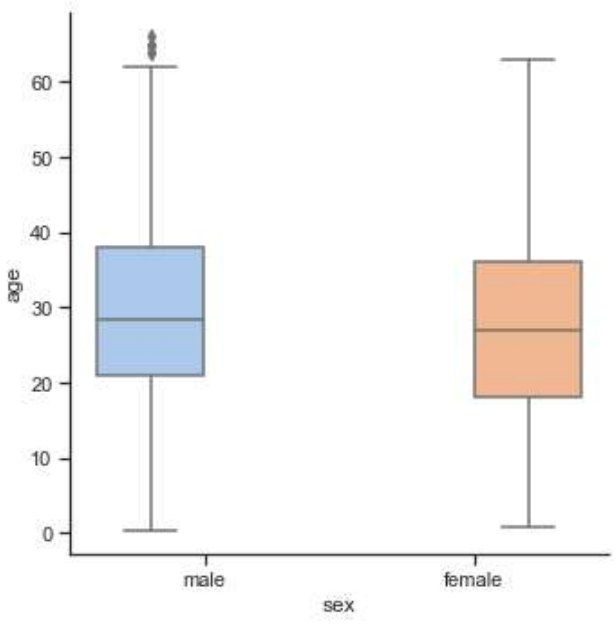
In [ ]:
```
sns.relplot(x='age',y='fare', hue='sex', data=ks1)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1bfdcb0eac0>
```



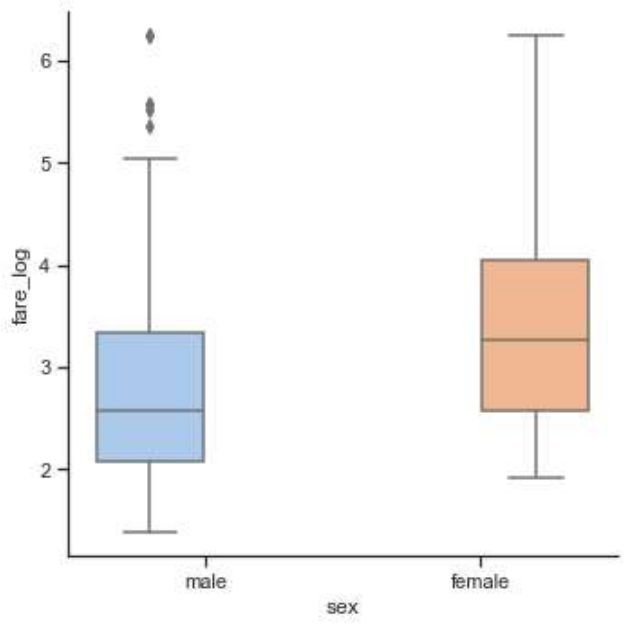
In [ ]:
```
# Catogery plot
sns.catplot(x='sex', y='age', hue='sex', data=ks1, kind='box')
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1bfdc2cf580>
```



In [ ]:
```
# after adding calculating and adding log fare, plot looks much better
sns.catplot(x='sex', y='fare_log', hue='sex', data=ks1, kind='box')
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1bfde058b20>



In [ ]: