Data_tool kit_ass

January 27, 2025

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

[154]:
```
#1)
array1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
array1
```

[154]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

[150]:
```
array2 = np.arange(1, 10).reshape(3, 3)
array2
```

[150]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

[156]:
```
array3=np.ones((3,3),dtype=int)
array3[0]=[1,2,3]
array3[1]=[4,5,6]
array3[2]=[7,8,9]
array3
```

[156]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

[5]:
```
#2
d=np.linspace(1,10,100)
```

[7]:
```
d
```

[7]:
```
array([ 1.        ,  1.09090909,  1.18181818,  1.27272727,  1.36363636,
        1.45454545,  1.54545455,  1.63636364,  1.72727273,  1.81818182,
        1.90909091,  2.        ,  2.09090909,  2.18181818,  2.27272727,
```

```
2.36363636,  2.45454545,  2.54545455,  2.63636364,  2.72727273,
2.81818182,  2.90909091,  3.        ,  3.09090909,  3.18181818,
3.27272727,  3.36363636,  3.45454545,  3.54545455,  3.63636364,
3.72727273,  3.81818182,  3.90909091,  4.        ,  4.09090909,
4.18181818,  4.27272727,  4.36363636,  4.45454545,  4.54545455,
4.63636364,  4.72727273,  4.81818182,  4.90909091,  5.        ,
5.09090909,  5.18181818,  5.27272727,  5.36363636,  5.45454545,
5.54545455,  5.63636364,  5.72727273,  5.81818182,  5.90909091,
6.        ,  6.09090909,  6.18181818,  6.27272727,  6.36363636,
6.45454545,  6.54545455,  6.63636364,  6.72727273,  6.81818182,
6.90909091,  7.        ,  7.09090909,  7.18181818,  7.27272727,
7.36363636,  7.45454545,  7.54545455,  7.63636364,  7.72727273,
7.81818182,  7.90909091,  8.        ,  8.09090909,  8.18181818,
8.27272727,  8.36363636,  8.45454545,  8.54545455,  8.63636364,
8.72727273,  8.81818182,  8.90909091,  9.        ,  9.09090909,
9.18181818,  9.27272727,  9.36363636,  9.45454545,  9.54545455,
9.63636364,  9.72727273,  9.81818182,  9.90909091, 10.        ])
```

[13]:
```
arr_2d=d.reshape(10,10)
```

[17]:
```
#the difference between np.array , np.asarray and np.asanyarray
#np.array creates ndarray
#np.asarray checks the input is an array
#there is no difference between np.array and np.asarray
#np.asanyarray preserves subclasses of ndarray
#this is used for ,atrix
```

[19]:
```
#sallow copy >> we give new pointer  for the given function ,so thagt both␣
 ↪pointers are pointing to the same location of the memory.when we change␣
 ↪values using the first pointer then the values also change in the other␣
 ↪pointer because as they are pointing to the function in a same memory␣
 ↪location
# deep copy >> providing pointer which copys the original pointer function and␣
 ↪the two pointer points two different memory location,so that what we change␣
 ↪in one pointer that values  or at the index values won't change in the other␣
 ↪pointer
```

[38]:
```
a=np.linspace(5,20,9)
a2=a.reshape(3,3)
a3=np.around(a2,decimals=2)
a2
```

[38]:
```
array([[ 5.   ,  6.875,  8.75 ],
       [10.625, 12.5  , 14.375],
       [16.25 , 18.125, 20.   ]])
```

[36]:
```
a3
```

[36]:
```
array([[ 5.  ,  6.88,  8.75],
       [10.62, 12.5 , 14.38],
       [16.25, 18.12, 20.  ]])
```

[50]:
```
s=np.linspace(1,10,30)
s1=s.reshape(5,6)
s2 = s1.astype(int)
s2
```

[50]:
```
array([[ 1,  1,  1,  1,  2,  2],
       [ 2,  3,  3,  3,  4,  4],
       [ 4,  5,  5,  5,  5,  6],
       [ 6,  6,  7,  7,  7,  8],
       [ 8,  8,  9,  9,  9, 10]])
```

[56]:
```
even_integers=s2[s2%2==0]
print(even_integers)
```

```
[ 2  2  2  4  4  4  6  6  6  8  8  8 10]
```

[58]:
```
odd_integers=s2[s2%2!=0]
print(odd_integers)
```

```
[1 1 1 1 3 3 3 5 5 5 5 7 7 7 9 9 9]
```

[60]:
```
arr1=np.random.randint(1,11,size= (3,3,3))
arr1
```

[60]:
```
array([[[ 9, 10,  4],
        [10,  6, 10],
        [10,  8,  8]],

       [[ 5,  6,  7],
        [ 8,  6, 10],
        [ 6,  1,  1]],

       [[ 7,  7,  7],
        [10,  6,  4],
        [ 2,  4,  7]]])
```

[62]:
```
max_indices = np.argmax(arr1, axis=2)
max_indices
```

[62]:
```
array([[1, 0, 0],
       [2, 2, 0],
       [0, 0, 2]], dtype=int64)
```

[66]:
```
result=arr1*arr1
result
```

[66]:
```
array([[[ 81, 100,  16],
        [100,  36, 100],
        [100,  64,  64]],

       [[ 25,  36,  49],
        [ 64,  36, 100],
        [ 36,   1,   1]],

       [[ 49,  49,  49],
        [100,  36,  16],
        [  4,  16,  49]]])
```

[68]:
```
#10
m=np.random.randint(1,7,size=35)
m1=m.reshape(7,5)
m2=pd.DataFrame(m1)
m2
```

[68]:

	0	1	2	3	4
0	5	1	4	2	1
1	2	5	3	6	6
2	3	2	1	6	1
3	6	5	3	1	6
4	4	3	6	4	1
5	2	3	4	6	1
6	4	6	6	3	6

[74]:
```
#11
series_1=pd.Series(np.arange(10,51))
series_1
```

[74]:
```
0     10
1     11
2     12
3     13
4     14
5     15
6     16
7     17
8     18
9     19
10    20
11    21
12    22
13    23
14    24
15    25
```

```
16    26
17    27
18    28
19    29
20    30
21    31
22    32
23    33
24    34
25    35
26    36
27    37
28    38
29    39
30    40
31    41
32    42
33    43
34    44
35    45
36    46
37    47
38    48
39    49
40    50
dtype: int32
```

[100]:
```
series_2=pd.Series(np.arange(100,1001,21))
series_2
```

[100]:
```
0     100
1     121
2     142
3     163
4     184
5     205
6     226
7     247
8     268
9     289
10    310
11    331
12    352
13    373
14    394
15    415
16    436
```

```
17    457
18    478
19    499
20    520
21    541
22    562
23    583
24    604
25    625
26    646
27    667
28    688
29    709
30    730
31    751
32    772
33    793
34    814
35    835
36    856
37    877
38    898
39    919
40    940
41    961
42    982
dtype: int32
```

[102]:
```
q=pd.DataFrame({"col1":series_1,"col2":series_2})
q
```

[102]:

	col1	col2
0	10.0	100
1	11.0	121
2	12.0	142
3	13.0	163
4	14.0	184
5	15.0	205
6	16.0	226
7	17.0	247
8	18.0	268
9	19.0	289
10	20.0	310
11	21.0	331
12	22.0	352
13	23.0	373
14	24.0	394

```
15  25.0   415
16  26.0   436
17  27.0   457
18  28.0   478
19  29.0   499
20  30.0   520
21  31.0   541
22  32.0   562
23  33.0   583
24  34.0   604
25  35.0   625
26  36.0   646
27  37.0   667
28  38.0   688
29  39.0   709
30  40.0   730
31  41.0   751
32  42.0   772
33  43.0   793
34  44.0   814
35  45.0   835
36  46.0   856
37  47.0   877
38  48.0   898
39  49.0   919
40  50.0   940
41   NaN   961
42   NaN   982
```

[122]:
```
#13
x=np.random.rand(100)
y=np.random.rand(100)
print(x)
```

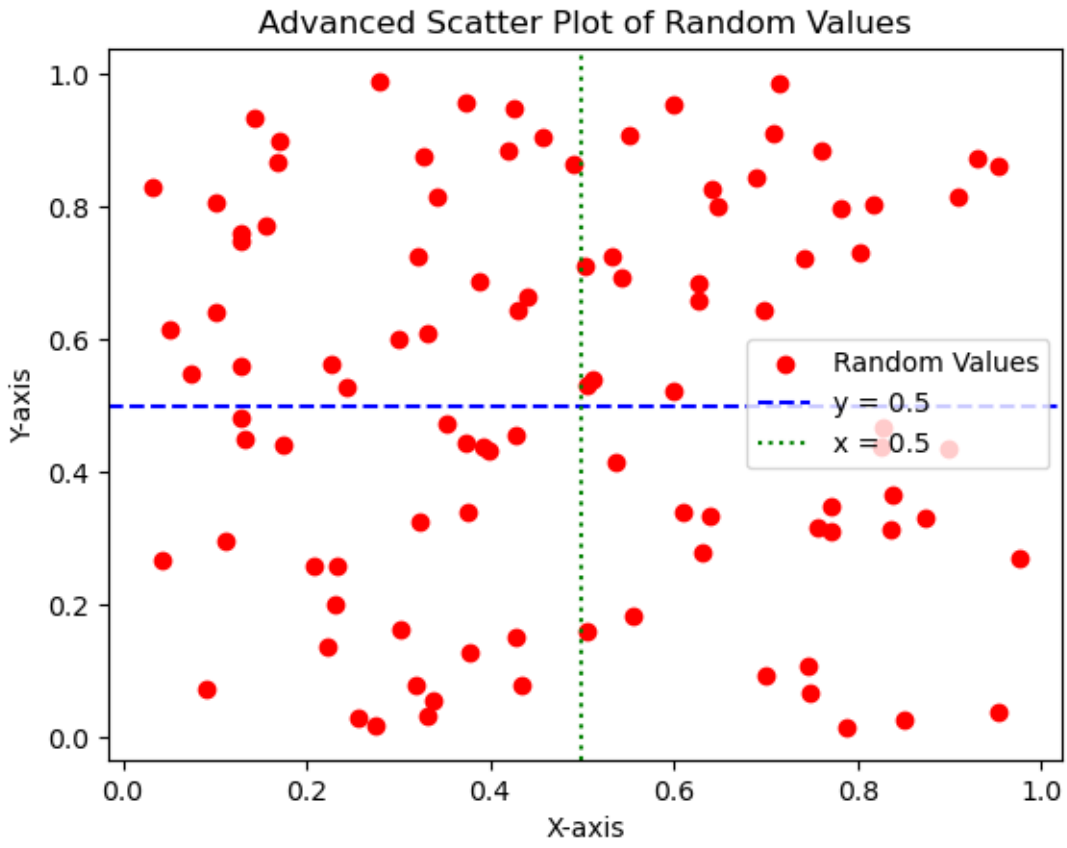
```
[0.33132143 0.91693252 0.14336721 0.43927739 0.55145939 0.67416113
 0.56398705 0.44024686 0.68103298 0.76313983 0.62255682 0.72921891
 0.91199283 0.10557072 0.22465849 0.29473779 0.29670279 0.74774113
 0.42328449 0.75020178 0.2350893  0.8537462  0.22370135 0.47153484
 0.06879057 0.61791118 0.72584633 0.69987185 0.6586736  0.63595945
 0.40969164 0.57827958 0.59768787 0.16395793 0.80580246 0.07488559
 0.52495036 0.69679656 0.53644565 0.75661209 0.35806963 0.84442199
 0.23666689 0.54168867 0.34922409 0.77988    0.4716724  0.06725361
 0.86203622 0.35137779 0.93006822 0.31807126 0.67930979 0.54939043
 0.26158348 0.46507891 0.50262179 0.37253134 0.0026915  0.10096374
 0.41272297 0.19697318 0.78834845 0.84931177 0.1939404  0.9681571
 0.84628643 0.30344496 0.17140337 0.97689171 0.6522909  0.56199217
 0.03927531 0.17360323 0.00419451 0.39695701 0.65461028 0.21811754
 0.40374066 0.60752634 0.35880888 0.55732055 0.81113529 0.93571497
```

```
 0.12720574 0.94084368 0.25296591 0.4651429  0.1532426  0.96485542
 0.05574891 0.26808723 0.97827875 0.07714719 0.3126161  0.38701105
 0.61397892 0.75544409 0.8095181  0.08732037]
```

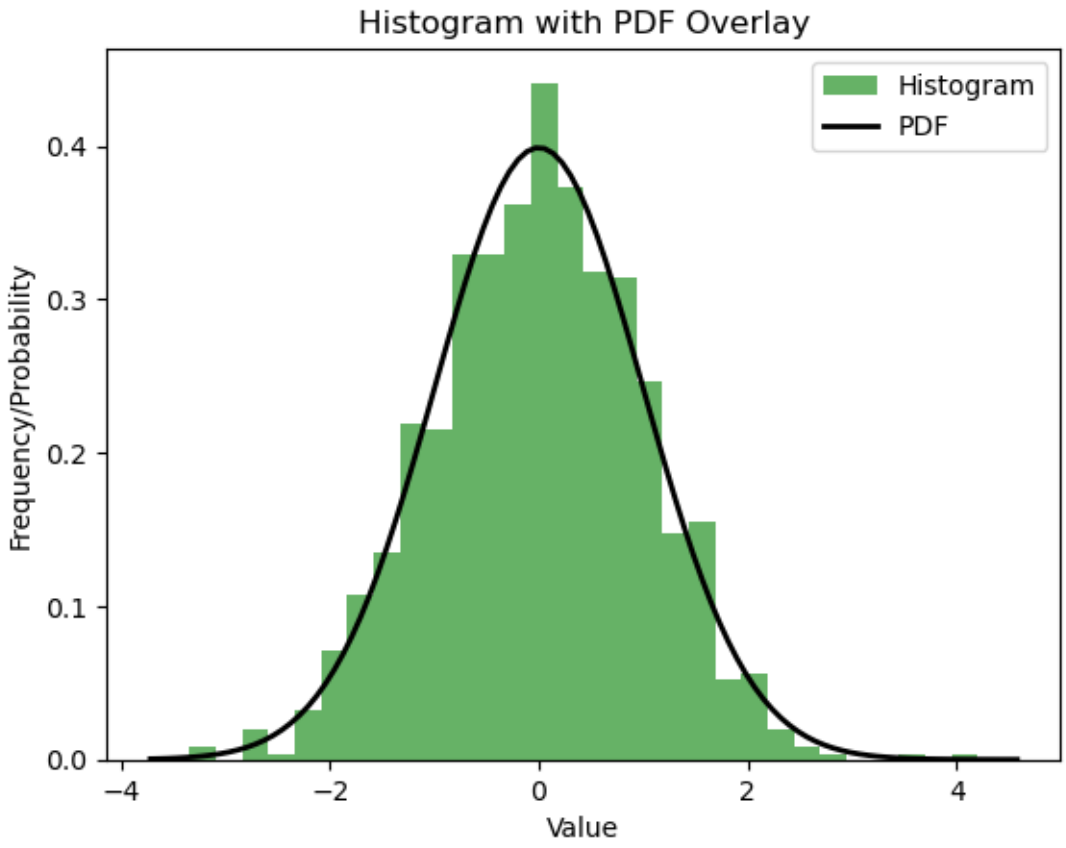
[124]:
```
print(y)
```

```
[4.01992477e-02 9.71243311e-01 8.67665027e-01 3.80003409e-01
 7.43600673e-01 2.34490169e-01 7.19430723e-01 8.41229297e-01
 7.21372301e-01 1.91511127e-01 4.91124185e-01 7.26154647e-01
 7.58569011e-01 5.10566545e-01 1.15615669e-01 2.63552135e-01
 1.14902237e-01 9.87416860e-01 3.65137046e-01 3.52923189e-01
 8.38175482e-01 8.04154158e-01 8.87389391e-01 2.13827918e-01
 4.45666312e-01 1.54608647e-01 8.48199488e-01 2.02706519e-01
 1.54747083e-01 2.40802159e-01 8.49713807e-01 8.61640444e-01
 3.92135889e-01 4.23192071e-01 2.32071643e-01 3.67640527e-01
 3.50729310e-01 7.23161368e-02 5.19023504e-01 5.74463450e-03
 5.41300665e-01 8.80036273e-01 4.54138571e-01 4.30783388e-01
 8.63607512e-01 6.62244324e-02 8.51895294e-01 6.86244730e-01
 7.74703721e-01 3.28115826e-01 2.28629425e-01 3.11020931e-01
 2.07061729e-01 2.57703672e-01 7.88248394e-01 3.03268435e-01
 8.44622625e-01 1.82675361e-01 8.80790206e-01 3.27800225e-01
 1.09073610e-02 1.59515322e-01 5.25915089e-01 5.18274643e-01
 9.96666430e-01 3.36881396e-01 8.42480748e-01 8.70632889e-01
 3.97096093e-01 4.62858742e-04 3.05102057e-01 7.21112881e-01
 2.61337828e-01 7.62474718e-01 7.83904799e-01 3.83167327e-01
 8.20184058e-01 2.41599903e-01 4.72214696e-01 8.49163815e-01
 8.71939541e-01 2.56860032e-01 1.09468506e-02 4.92827355e-01
 6.46658304e-01 8.88930933e-01 9.40801404e-01 5.20020622e-01
 1.84533470e-01 6.17783105e-01 6.83110159e-01 8.09902175e-01
 1.76835824e-01 6.21537602e-01 8.75786245e-01 3.55850775e-01
 9.44866523e-01 2.63347281e-01 1.42506941e-01 1.75460357e-01]
```

[142]:
```
plt.scatter(x, y, color='red', marker='o', label='Random Values')

plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Advanced Scatter Plot of Random Values')
plt.legend()
plt.show()
```



[146]:
```
#15,16
from scipy.stats import norm
data = np.random.normal(loc=0, scale=1, size=1000)
plt.hist(data, bins=30, density=True, alpha=0.6, color='g', label='Histogram')

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, 0, 1)
plt.plot(x, p, 'k', linewidth=2, label='PDF')
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')
plt.legend()

plt.show()
```



[49]:
```
df=pd.read_csv(r"C:\Users\jarup\Downloads\data science papers\EDA\People Data.
↪csv")
```

[51]:
```
df
```

[51]:

	Index	User Id	First Name	Last Name	Gender	\
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	
..	…	…	…	…	…	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	
999	1000	8b756f6231DDC6e	Lee	Tran	Female	

	Email	Phone	Date of birth	\
0	pwarner@example.org	857.139.8239	27-01-2014	

```
1    fergusonkatherine@example.net                   NaN    26-07-1931
2              fhoward@example.org          (599)782-0605    25-11-2013
3            zjohnston@example.com                   NaN    17-11-2012
4                 elin@example.net    (390)417-1635x3010    15-04-1923
..                             …                     …       …
995         lyonsdaisy@example.net           021.775.2933    05-01-1959
996        dariusbryan@example.com  001-149-710-7799x721    06-10-2001
997         georgechan@example.org  +1-750-774-4128x33265    13-05-1918
998            wanda04@example.net          (915)292-2254    31-08-1971
999        deannablack@example.org     079.752.5424x67259    24-01-1947

                          Job Title  Salary
0                 Probation officer   90000
1                           Dancer   80000
2                             Copy   50000
3          Counselling psychologist   65000
4               Biomedical engineer  100000
..                              …     …
995               Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                Ambulance person  100000
999      Nurse, learning disability   90000

[1000 rows x 10 columns]
```

[7]:
```
#7)
df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True)
```

[9]:
```
df['Phone'] = pd.to_numeric(df['Phone'], errors='coerce')
```

[15]:
```
df['Phone'] = df['Phone'].fillna(0).astype(int)
```

[17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Index          1000 non-null   int64 
 1   User Id        1000 non-null   object
 2   First Name     1000 non-null   object
 3   Last Name      1000 non-null   object
 4   Gender         1000 non-null   object
 5   Email          1000 non-null   object
 6   Phone          1000 non-null   int32 
 7   Date of birth  1000 non-null   object
```

```
 8   Job Title      1000 non-null   object
 9   Salary         1000 non-null   int64
dtypes: int32(1), int64(2), object(7)
memory usage: 74.3+ KB
```

[31]:
```
df.columns
```

[31]:
```
Index(['50', 'afF3018e9cdd1dA', 'George', 'Mercer', 'Female',
       'douglascontreras@example.net', '+1-326-669-0118x4341', '11-09-1941',
       'Human resources officer', '70000'],
      dtype='object')
```

[13]:
```
df = df.iloc[50:]
```

[15]:
```
df
```

[15]:

	Index	User Id	First Name	Last Name	Gender	\
50	51	CccE5DAb6E288e5	Jo	Zavala	Male	
51	52	DfBDc3621D4bcec	Joshua	Carey	Female	
52	53	f55b0A249f5E44D	Rickey	Hobbs	Female	
53	54	Ed71DcfaBFd0beE	Robyn	Reilly	Male	
54	55	FDaFD0c3f5387EC	Christina	Conrad	Male	
..	…	…	…	…	…	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	
999	1000	8b756f6231DDC6e	Lee	Tran	Female	

	Email	Phone	Date of birth	\
50	pamela64@example.net	001-859-448-9935x54536	23-11-1992	
51	dianashepherd@example.net	001-274-739-8470x814	07-01-1915	
52	ingramtiffany@example.org	241.179.9509x498	01-07-1910	
53	carriecrawford@example.org	207.797.8345x6177	27-07-1982	
54	fuentesclaudia@example.net	001-599-042-7428x143	06-01-1998	
..	…	…	…	
995	lyonsdaisy@example.net	021.775.2933	05-01-1959	
996	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	
997	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	
998	wanda04@example.net	(915)292-2254	31-08-1971	
999	deannablack@example.org	079.752.5424x67259	24-01-1947	

	Job Title	Salary
50	Nurse, adult	80000
51	Seismic interpreter	70000
52	Barrister	60000
53	Engineer, structural	100000
54	Producer, radio	50000

```
..                                 …     …
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                  Ambulance person  100000
999       Nurse, learning disability   90000

[950 rows x 10 columns]
```

[17]:
```
df = df[['Last Name', 'Gender', 'Email', 'Phone', 'Salary']]
```

[19]:
```
df
```

[19]:
```
    Last Name  Gender                        Email                    Phone  \
50     Zavala    Male        pamela64@example.net  001-859-448-9935x54536
51      Carey  Female   dianashepherd@example.net    001-274-739-8470x814
52      Hobbs  Female   ingramtiffany@example.org        241.179.9509x498
53     Reilly    Male  carriecrawford@example.org       207.797.8345x6177
54     Conrad    Male  fuentesclaudia@example.net    001-599-042-7428x143
..        …     …                           …                       …
995    Bryant  Female      lyonsdaisy@example.net            021.775.2933
996     Barry  Female     dariusbryan@example.com    001-149-710-7799x721
997  Mckinney  Female      georgechan@example.org   +1-750-774-4128x33265
998    Phelps    Male         wanda04@example.net           (915)292-2254
999      Tran  Female     deannablack@example.org      079.752.5424x67259

     Salary
50    80000
51    70000
52    60000
53   100000
54    50000
..      …
995   90000
996   50000
997   60000
998  100000
999   90000

[950 rows x 5 columns]
```

[21]:
```
df.head(10)
```

[21]:
```
    Last Name  Gender                        Email                    Phone  \
50     Zavala    Male        pamela64@example.net  001-859-448-9935x54536
51      Carey  Female   dianashepherd@example.net    001-274-739-8470x814
52      Hobbs  Female   ingramtiffany@example.org        241.179.9509x498
```

```
53    Reilly    Male  carriecrawford@example.org       207.797.8345x6177
54    Conrad    Male  fuentesclaudia@example.net    001-599-042-7428x143
55      Cole    Male      kaneaudrey@example.org            663-280-5834
56   Donovan    Male   rebekahsantos@example.net                     NaN
57    Little  Female         craig28@example.com       125.219.3673x0076
58    Dawson  Female  connercourtney@example.net      650-748-3069x64529
59      Page    Male  harrygallagher@example.com        849.500.6331x717

    Salary
50   80000
51   70000
52   60000
53  100000
54   50000
55   85000
56   65000
57   60000
58   60000
59   60000
```

[23]:
```
s=df['Salary']
print(s.tail())
```

```
995     90000
996     50000
997     60000
998    100000
999     90000
Name: Salary, dtype: int64
```

[27]:
```
#9
filtered_df = df[(df['Last Name'] == 'Duke') & (df['Gender'] == 'Female') &␣
 ↪(df['Salary'] < 85000)]
print(filtered_df)
```

```
    Last Name  Gender                    Email                Phone  Salary
210      Duke  Female       robin78@example.com         740.434.0212   50000
457      Duke  Female  perryhoffman@example.org  +1-903-596-0995x489   50000
729      Duke  Female   kevinkramer@example.net         982.692.6257   70000
```

[33]:
```
df
```

[33]:
```
     Index          User Id First Name Last Name  Gender  \
0        1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1        2  3d5AD30A4cD38ed         Jo    Rivers  Female
2        3  810Ce0F276Badec     Sheryl    Lowery  Female
3        4  BF2a889C00f0cE1    Whitney    Hooper    Male
4        5  9afFEafAe1CBBB9    Lindsey      Rice  Female
```

```
..     …                …         …        …      …
995    996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996    997  ECddaFEDdEc4FAB      Donna     Barry  Female
997    998  2adde51d8B8979E      Cathy  Mckinney  Female
998    999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999   1000  8b756f6231DDC6e        Lee      Tran  Female

                            Email                  Phone Date of birth  \
0              pwarner@example.org           857.139.8239    27-01-2014
1     fergusonkatherine@example.net                    NaN    26-07-1931
2              fhoward@example.org          (599)782-0605    25-11-2013
3            zjohnston@example.com                    NaN    17-11-2012
4                 elin@example.net     (390)417-1635x3010    15-04-1923
..                              …                    …           …
995         lyonsdaisy@example.net           021.775.2933    05-01-1959
996        dariusbryan@example.com   001-149-710-7799x721    06-10-2001
997         georgechan@example.org  +1-750-774-4128x33265    13-05-1918
998            wanda04@example.net          (915)292-2254    31-08-1971
999        deannablack@example.org     079.752.5424x67259    24-01-1947

                         Job Title  Salary
0                Probation officer   90000
1                          Dancer   80000
2                            Copy   50000
3          Counselling psychologist   65000
4               Biomedical engineer  100000
..                              …     …
995              Personnel officer   90000
996         Education administrator   50000
997  Commercial/residential surveyor   60000
998                Ambulance person  100000
999      Nurse, learning disability   90000

[1000 rows x 10 columns]
```

[53]:
```
#12
df=df.drop(columns=['Email','Phone','Date of birth'],axis =1)
df=df.dropna()
```

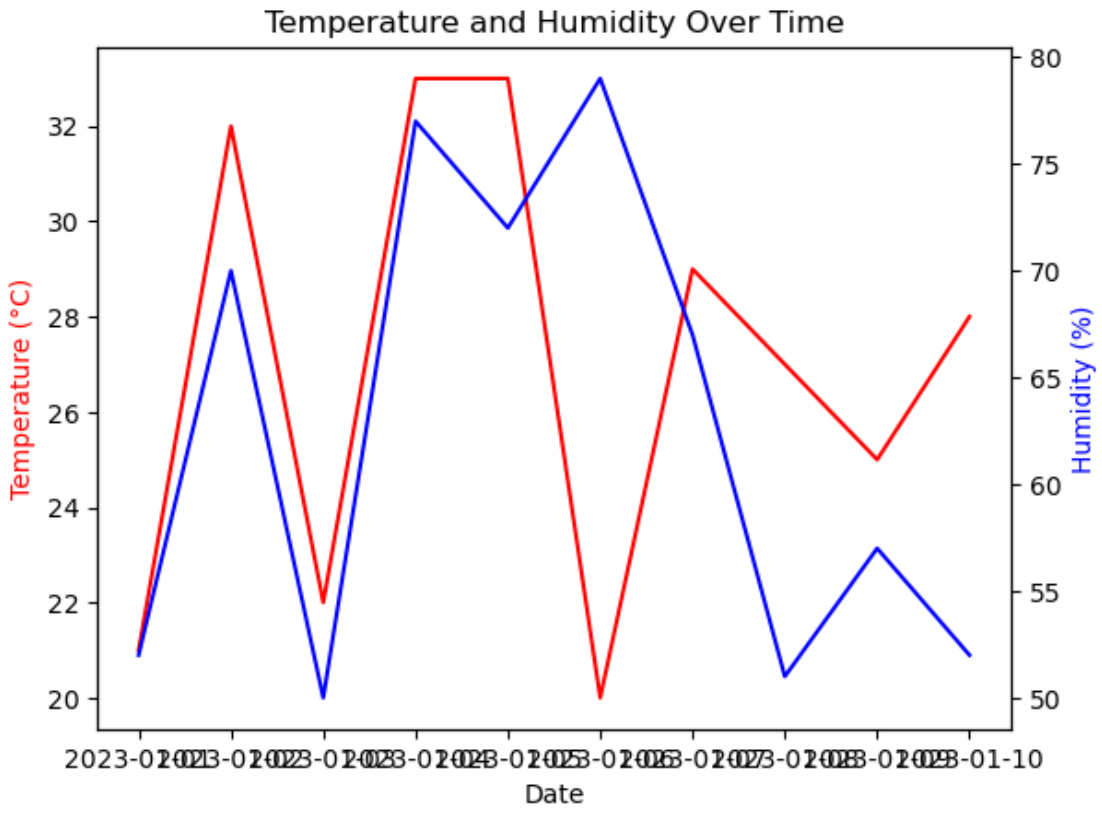
[39]:
```
df.columns
```

[39]:
```
Index(['Index', 'User Id', 'First Name', 'Last Name', 'Gender', 'Email',
       'Phone', 'Date of birth', 'Job Title', 'Salary'],
      dtype='object')
```

[43]:
```
df
```

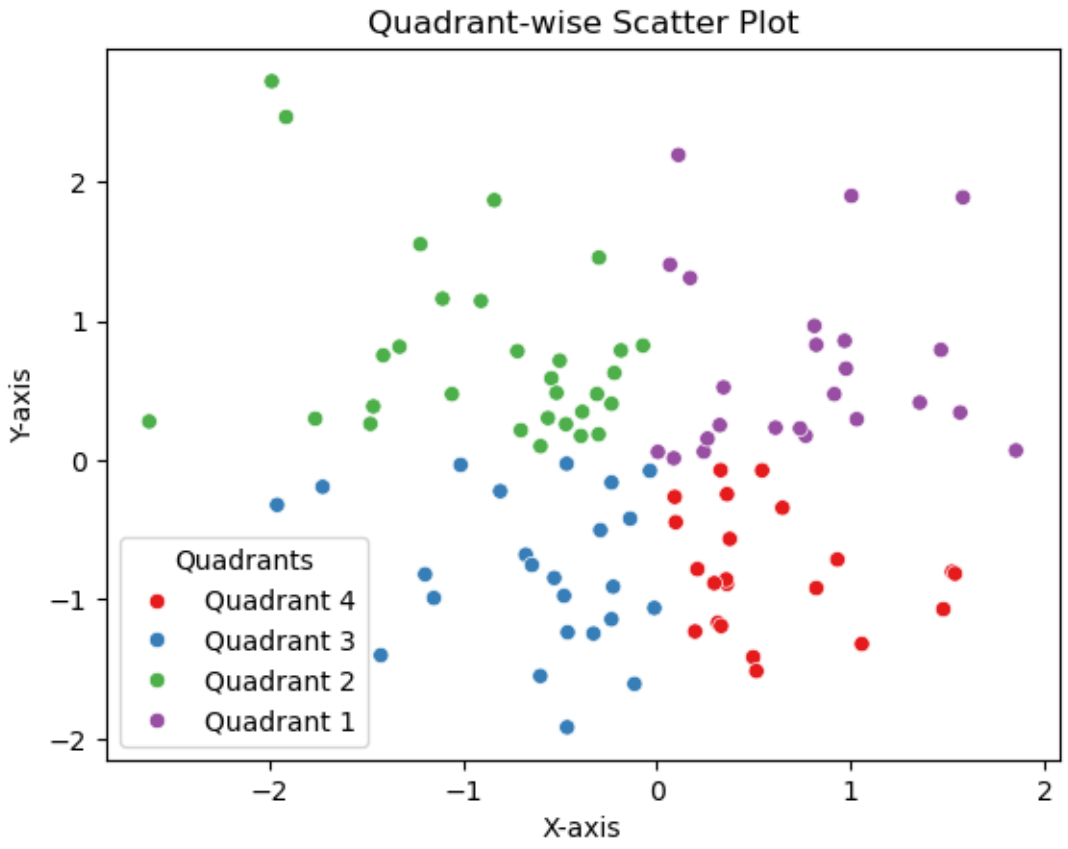
[43]:
```
      Index          User Id First Name Last Name  Gender  \
0         1  8717bbf45cCDbEe     Shelia   Mahoney    Male
1         2  3d5AD30A4cD38ed         Jo    Rivers  Female
2         3  810Ce0F276Badec     Sheryl    Lowery  Female
3         4  BF2a889C00f0cE1    Whitney    Hooper    Male
4         5  9afFEafAe1CBBB9    Lindsey      Rice  Female
..      ...              ...        ...       ...     ...
995     996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
996     997  ECddaFEDdEc4FAB      Donna     Barry  Female
997     998  2adde51d8B8979E      Cathy  Mckinney  Female
998     999  Fb2FE369D1E171A   Jermaine    Phelps    Male
999    1000  8b756f6231DDC6e        Lee      Tran  Female

                           Job Title  Salary
0                  Probation officer   90000
1                            Dancer   80000
2                              Copy   50000
3           Counselling psychologist   65000
4                Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996          Education administrator   50000
997  Commercial/residential surveyor   60000
998                  Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 7 columns]
```

[57]:
```python
#14
data = {
    'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'Temperature': np.random.randint(20, 35, size=10),
    'Humidity': np.random.randint(50, 80, size=10)
}
df = pd.DataFrame(data)
fig, ax1 = plt.subplots()
ax1.plot(df['Date'], df['Temperature'], color='red')
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature (°C)', color='red')
ax2 = ax1.twinx()
ax2.plot(df['Date'], df['Humidity'], color='blue')
ax2.set_ylabel('Humidity (%)', color='blue')
plt.title('Temperature and Humidity Over Time')

plt.show()
```



[61]:
```
#17
np.random.seed(42)
x = np.random.randn(100)
y = np.random.randn(100)
df = pd.DataFrame({'x': x, 'y': y})

def quadrant(row):
    if row['x'] >= 0 and row['y'] >= 0:
        return 'Quadrant 1'
    elif row['x'] < 0 and row['y'] >= 0:
        return 'Quadrant 2'
    elif row['x'] < 0 and row['y'] < 0:
        return 'Quadrant 3'
    else:
        return 'Quadrant 4'

df['Quadrant'] = df.apply(quadrant, axis=1)
sns.scatterplot(data=df, x='x', y='y', hue='Quadrant', palette='Set1')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
```

```
plt.title('Quadrant-wise Scatter Plot')


plt.legend(title='Quadrants')
plt.show()
```

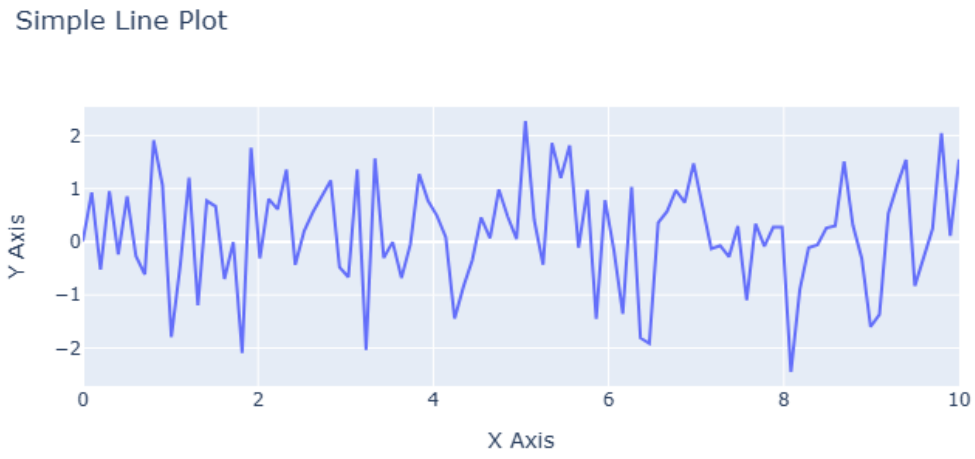


[65]:
```
from bokeh.plotting import figure, show
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)
p = figure(title="Sine Wave Function", x_axis_label='X', y_axis_label='Y')
p.line(x, y, legend_label="Sine Wave", line_width=2)
p.grid.visible = True
show(p)
```

[79]:
```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
categories = ['Category A', 'Category B', 'Category C', 'Category D', 'Category␣
 ↪E']
values = np.random.randint(10, 100, size=len(categories))
```

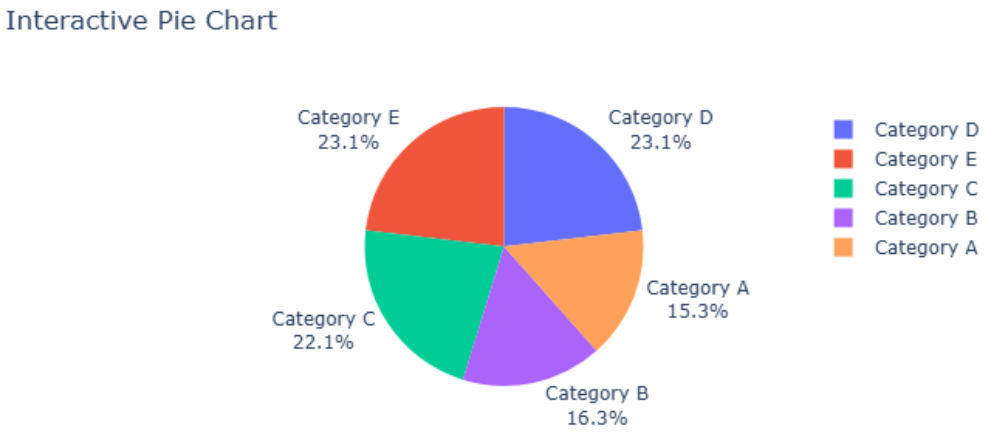
```
p = figure(x_range=categories, title="Random Categorical Bar␣
 ↪Chart",x_axis_label='Categories', y_axis_label='Values')

p.vbar(x=categories, top=values, width=0.6, color='skyblue')
show(p)
```

[73]:
```
#20

import plotly.graph_objects as go

x = np.linspace(0, 10, 100)
y = np.random.randn(100)

fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Random Data'))
fig.update_layout(
    title="Simple Line Plot",
    xaxis_title="X Axis",
    yaxis_title="Y Axis")


fig.show()
```



[81]:
```
import plotly.graph_objects as go
np.random.seed(0)
labels = ['Category A', 'Category B', 'Category C', 'Category D', 'Category E']
values = np.random.randint(1, 100, size=len(labels))
```

```
fig = go.Figure(data=[go.Pie(labels=labels, values=values,␣
↪textinfo='label+percent', insidetextorientation='radial')])
fig.update_layout(title='Interactive Pie Chart')
fig.show()
```



[ ]: