Data Toolkit Assignment

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import seaborn as sns
from bokeh.plotting import figure, show
import plotly.graph_objects as go
import plotly.express as px

#Q.1 Demonstrate three different methods for creating identical 2D
arrays in NumPy. Provide the code for each
#    method and the final output after each method.

#method1
arr1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
arr1

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

#method2

arr2 = np.zeros((3,3), dtype = int)

arr2[0,:] = [1,2,3]
arr2[1,:] = [4,5,6]
arr2[2,:] = [7,8,9]

arr2

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

#method3

arr3 =np.fromfunction(lambda i,j : i*3+j+1 , (3,3), dtype =int)
arr3

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



#Q2. Using the Numpy function, generate an array of 100 evenly spaced
numbers Between 1 and 10 and
```

```
#    Reshape that 1D array into a 2D array.

a = np.linspace(1,10,100)
a

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.        ])

b = a.reshape((20,5))

b
```

```
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.
]])
```

```
#Q4.Generate a 3*3 array with random floating-point numbers between 5
and 20 then, round each number in
#   the array to 2 decimal places.

rand_float = np.random.uniform(5,20, size = (3,3))
```

```
round_off = np.round(rand_float,2)

round_off
```

```
array([[13.53,  8.83, 12.67],
       [10.67, 16.95,  6.73],
       [18.07, 15.43, 15.06]])
```

```
#Q5.Create a NumPy array with random integers between 1 and 10 of
shape (5, 6). After creating the array
#perform the following operations:


# a)Extract all even integers from array.
rand_int = np.random.randint(1,10,(5,6))

even_int = rand_int[(rand_int % 2 == 0 ) & (rand_int ==
rand_int.astype(int))]
even_int
```

```
array([6, 2, 6, 8, 4, 8, 8, 4, 8, 4, 6, 4, 6, 2, 6, 4])
```

```
# b)Extract all odd integers from array.

odd_int = rand_int[(rand_int % 2 != 0 ) & (rand_int ==
rand_int.astype(int))]
odd_int
```

```
array([9, 3, 3, 9, 9, 1, 7, 9, 3, 3, 7, 1, 1, 5, 1, 9, 5])
```

```
#Q6. Create a 3D NumPy array of shape (3, 3, 3) containing random
integers between 1 and 10. Perform the
#    following operations:

#   a) Find the indices of the maximum values along each depth level
(third axis).
#   b) Perform element-wise multiplication of between both array.

# Create a 3D NumPy array of shape (3, 3, 3) with random integers
between 1 and 10
arr_3d = np.random.randint(1, 10, size=(3, 3, 3))

arr_3d
```

```
array([[[9, 5, 8],
        [9, 6, 6],
        [9, 3, 3]],
```

```
       [[3, 9, 3],
        [4, 5, 3],
        [2, 7, 5]],

       [[8, 7, 3],
        [1, 4, 8],
        [5, 2, 4]]])

# Find the indices of the maximum values along each depth level (third
axis)
max_indices = np.argmax(arr_3d, axis=2)

max_indices

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

# Perform element-wise multiplication of between both array.
elmnt_multiplication =(arr_3d * max_indices)
elmnt_multiplication

array([[[ 0,  0,  0],
        [ 9,  6,  6],
        [ 0,  6,  0]],

       [[ 0,  0,  0],
        [ 4,  5,  3],
        [ 0, 14,  0]],

       [[ 0,  0,  0],
        [ 1,  4,  8],
        [ 0,  4,  0]]], dtype=int64)



#Q7.Clean and transform the 'Phone' column in the sample dataset to
remove non-numeric characters and
#   convert it to a numeric data type. Also display the table
attributes and data types of each column

df = pd.read_csv("People Data.csv")

# Clean and transform the 'Phone' column to remove non-numeric
characters and convert it to a numeric data type
df['Phone'] =pd.to_numeric( df['Phone'].str.replace(r'\D', '',
regex=True)).fillna(0).astype(int)

# Display the first few rows to verify the changes
df.head()
```

```
   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

                           Email           Phone Date of birth  \
0            pwarner@example.org      8571398239    27-01-2014
1  fergusonkatherine@example.net               0    26-07-1931
2            fhoward@example.org      5997820605    25-11-2013
3          zjohnston@example.com               0    17-11-2012
4              elin@example.net  39041716353010    15-04-1923

                  Job Title  Salary
0        Probation officer   90000
1                   Dancer   80000
2                     Copy   50000
3  Counselling psychologist   65000
4       Biomedical engineer  100000
```

```
# display the table attributes and data types of each column

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   int64
 7   Date of birth  1000 non-null   object
 8   Job Title      1000 non-null   object
 9   Salary         1000 non-null   int64
dtypes: int64(3), object(7)
memory usage: 78.2+ KB
```

```
#Q8. Perform the following tasks using people dataset:

# a) Read the 'data.csv' file using pandas, skipping the first 50
rows.
```

```
people_data =pd.read_csv("People Data.csv", skiprows = 50)
people_data
```

```
       50  afF3018e9cdd1dA     George    Mercer  Female  \
0      51  CccE5DAb6E288e5         Jo    Zavala    Male
1      52  DfBDc3621D4bcec     Joshua     Carey  Female
2      53  f55b0A249f5E44D     Rickey     Hobbs  Female
3      54  Ed71DcfaBFd0beE      Robyn    Reilly    Male
4      55  FDaFD0c3f5387EC  Christina    Conrad    Male
..    ...              ...        ...       ...     ...
945   996  fedF4c7Fd9e7cFa       Kurt    Bryant  Female
946   997  ECddaFEDdEc4FAB      Donna     Barry  Female
947   998  2adde51d8B8979E      Cathy  Mckinney  Female
948   999  Fb2FE369D1E171A   Jermaine    Phelps    Male
949  1000  8b756f6231DDC6e        Lee      Tran  Female

    douglascontreras@example.net    +1-326-669-0118x4341  11-09-
1941  \
0           pamela64@example.net  001-859-448-9935x54536  23-11-1992

1      dianashepherd@example.net    001-274-739-8470x814  07-01-1915

2      ingramtiffany@example.org        241.179.9509x498  01-07-1910

3     carriecrawford@example.org       207.797.8345x6177  27-07-1982

4     fuentesclaudia@example.net    001-599-042-7428x143  06-01-1998

..                           ...                     ...         ...

945       lyonsdaisy@example.net             021.775.2933  05-01-1959

946      dariusbryan@example.com    001-149-710-7799x721  06-10-2001

947       georgechan@example.org   +1-750-774-4128x33265  13-05-1918

948          wanda04@example.net           (915)292-2254  31-08-1971

949      deannablack@example.org      079.752.5424x67259  24-01-1947


             Human resources officer   70000
0                       Nurse, adult   80000
1                Seismic interpreter   70000
2                          Barrister   60000
3               Engineer, structural  100000
4                    Producer, radio   50000
..                               ...     ...
945                Personnel officer   90000
946          Education administrator   50000
947  Commercial/residential surveyor   60000
```

```
948                  Ambulance person  100000
949       Nurse, learning disability   90000

[950 rows x 10 columns]
```

```
# b) Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and
‘Salary’ from the file.

df[["Last Name","Gender","Email","Phone","Salary"]]
```

```
    Last Name  Gender                          Email              Phone
Salary
0     Mahoney    Male            pwarner@example.org         8571398239
90000
1      Rivers  Female  fergusonkatherine@example.net                  0
80000
2      Lowery  Female            fhoward@example.org         5997820605
50000
3      Hooper    Male          zjohnston@example.com                  0
65000
4        Rice  Female               elin@example.net     39041716353010
100000
..        ...     ...                             ...                ...
...
995    Bryant  Female         lyonsdaisy@example.net          217752933
90000
996     Barry  Female        dariusbryan@example.com     11497107799721
50000
997  Mckinney  Female         georgechan@example.org  1750774412833265
60000
998    Phelps    Male            wanda04@example.net         9152922254
100000
999      Tran  Female        deannablack@example.org     79752542467259
90000

[1000 rows x 5 columns]
```

```
# c) Display the first 10 rows of the filtered dataset.

df.head(10)
```

```
   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
5      6  aF75e6dDEBC5b66     Sherry  Caldwell    Male
6      7  efeb05c7Cc94EA3     Ernest   Hoffman    Male
7      8  fb1BF3FED57E9d7      Doris  Andersen    Male
8      9  421fAB9a3b98F30     Cheryl      Mays    Male
```

```
9     10  4A42Fe10dB717CB      Harry  Mitchell    Male

                           Email           Phone Date of birth  \
0            pwarner@example.org      8571398239    27-01-2014
1  fergusonkatherine@example.net               0    26-07-1931
2            fhoward@example.org      5997820605    25-11-2013
3          zjohnston@example.com               0    17-11-2012
4              elin@example.net  39041716353010    15-04-1923
5          kaitlin13@example.net      8537800927    06-08-1917
6         jeffharvey@example.com   9365574807895    22-12-1984
7           alicia33@example.org      4709522945    02-12-2016
8             jake50@example.com       138204758    16-12-2012
9      lanechristina@example.net  56090350684985    29-06-1953

                   Job Title  Salary
0          Probation officer   90000
1                     Dancer   80000
2                       Copy   50000
3   Counselling psychologist   65000
4        Biomedical engineer  100000
5  Higher education lecturer   50000
6             Health visitor   60000
7                 Air broker   65000
8       Designer, multimedia   50000
9  Insurance account manager   50000
```

```
# d) Extract the ‘Salary’' column as a Series and display its last 5 values

df["Salary"].tail(5)
```

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

```
df
```

```
     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        8571398239    27-01-2014
1    fergusonkatherine@example.net                 0    26-07-1931
2               fhoward@example.org        5997820605    25-11-2013
3             zjohnston@example.com                 0    17-11-2012
4                 elin@example.net    39041716353010    15-04-1923
..                             ...               ...           ...
995         lyonsdaisy@example.net         217752933    05-01-1959
996        dariusbryan@example.com    11497107799721    06-10-2001
997         georgechan@example.org  1750774412833265    13-05-1918
998            wanda04@example.net        9152922254    31-08-1971
999        deannablack@example.org    79752542467259    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]

#Q9. Filter and select rows from the People_Dataset, where the “Last
Name' column contains the name 'Duke',
#   'Gender' column contains the word Female and ‘Salary’ should be
less than 85000.

df[(df["Last Name"] == "Duke") & (df["Gender"] == "Female") &
(df["Salary"] <85000)]

     Index          User Id First Name Last Name  Gender  \
45      46  99A502C175C4EBd     Olivia      Duke  Female
210    211  DF17975CC0a0373    Katrina      Duke  Female
457    458  dcE1B7DE83c1076      Traci      Duke  Female
729    730  c9b482D7aa3e682     Lonnie      Duke  Female

                        Email             Phone Date of birth  \
45        diana26@example.net  1366475860704350    13-10-1934
210       robin78@example.com        7404340212    21-09-1935
457  perryhoffman@example.org    19035960995489    11-02-1997
729   kevinkramer@example.net        9826926257    12-05-2015
```

```
           Job Title  Salary
45           Dentist   60000
210  Producer, radio   50000
457        Herbalist   50000
729     Nurse, adult   70000
```

```
#Q10. Create a 7*5 Dataframe in Pandas using a series generated from
35 random integers between 1 to 6?

data_frame = pd.DataFrame(np.random.randint(1,6, size = (7,5)))
data_frame
```

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

```
#Q11. Create two different Series, each of length 50, with the
following criteria:
# a) The first Series should contain random numbers ranging from 10 to
50.
# b) The second Series should contain random numbers ranging from 100
to 1000.
# c) Create a DataFrame by joining these Series by column, and,
#    change the names of the columns to 'col1', 'col2',etc.

# a)Create the first Series with random numbers ranging from 10 to 50
series_1 = pd.Series(np.random.randint(10,50 , size = 50))

# b)Create the second Series with random numbers ranging from 100 to
1000
series_2 = pd.Series(np.random.randint(100 , 1000 , size = 50))

# c)Create a DataFrame by joining these Series by column
data_f = pd.concat([series_1 , series_2], axis = 1)

data_f.columns = ['col1','col2']

data_f
```

```
    col1  col2
0     44   624
1     18   598
2     17   799
```

```
3     45   443
4     48   231
5     32   268
6     48   738
7     29   314
8     35   822
9     11   558
10    24   634
11    14   857
12    25   480
13    32   557
14    11   380
15    40   191
16    23   818
17    48   790
18    39   865
19    10   265
20    29   784
21    19   224
22    22   859
23    37   824
24    25   355
25    48   758
26    46   732
27    22   392
28    35   556
29    48   692
30    19   439
31    24   735
32    14   294
33    45   248
34    16   594
35    45   711
36    48   132
37    32   679
38    47   146
39    16   300
40    35   134
41    19   630
42    11   621
43    22   586
44    13   944
45    32   788
46    43   979
47    32   326
48    31   923
49    28   947

#Q12. Perform the following operations using people data set:
#     a) Delete the 'Email', 'Phone', and 'Date of birth' columns from
```

```
the dataset.
#     b) Delete the rows containing any missing values.
#     d) Print the final output also

df1 = pd.read_csv("People Data.csv")

df1
```

```
     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
```

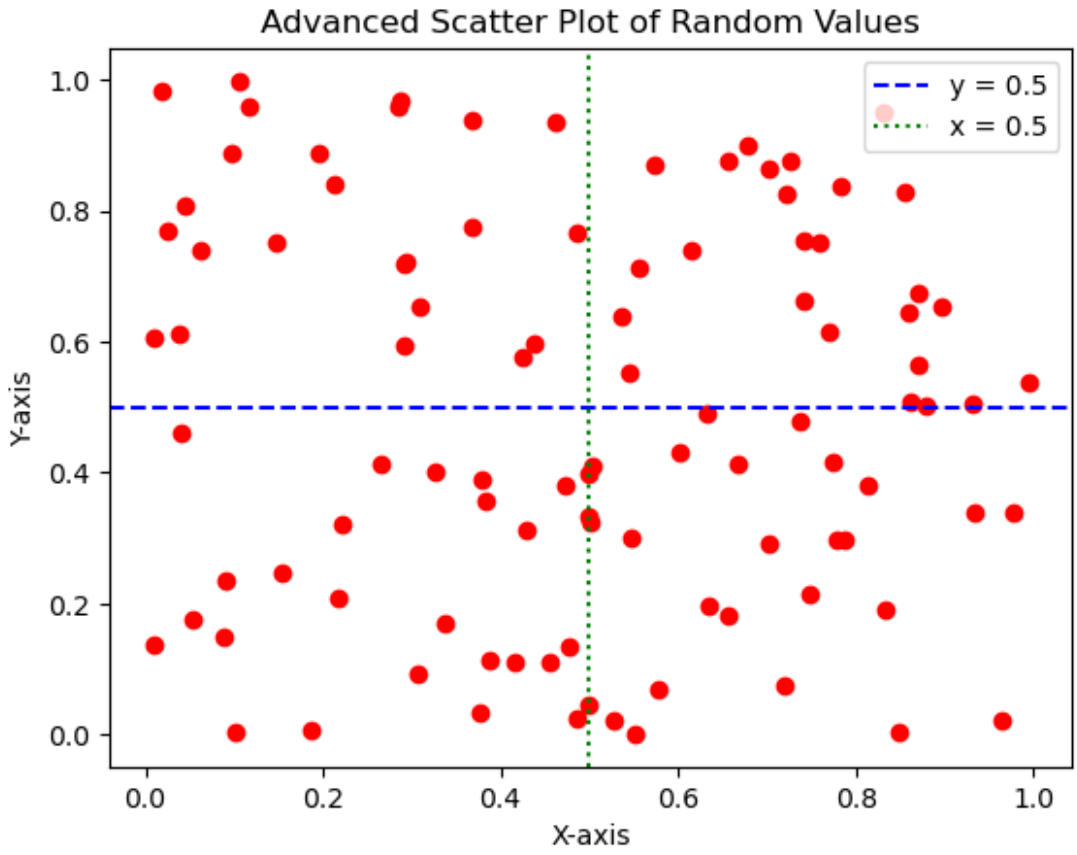
```
3           Counselling psychologist   65000
4                Biomedical engineer  100000
..                               ...     ...
995                Personnel officer   90000
996         Education administrator   50000
997  Commercial/residential surveyor   60000
998                Ambulance person  100000
999       Nurse, learning disability   90000

[1000 rows x 10 columns]
```

```
#a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

df1.drop(columns = ["Email","Phone","Date of birth"], inplace = True)

#b) Delete the rows containing any missing values.

df1.dropna(inplace = True)

#d) Print the final output also

df1
```

```
     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]
```

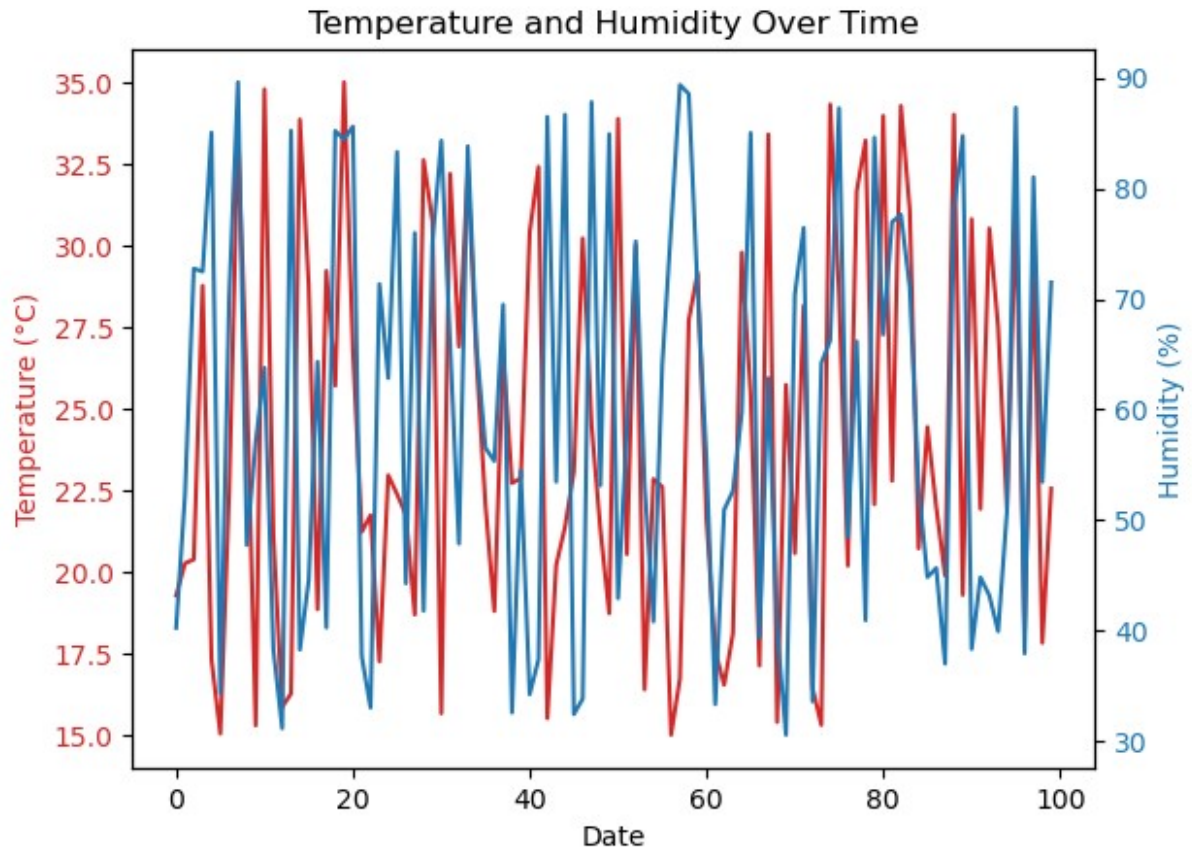
```
#Q13.Create two NumPy arrays, x and y, each containing 100 random
float values between 0 and 1. Perform the
#    following tasks using Matplotlib and NumPy:

#  a) Create a scatter plot using x and y, setting the color of the
points to red and the marker style to 'o'.
#  b) Add a horizontal line at y = 0.5 using a dashed line style and
label it as 'y = 0.5'.
#  c) Add a vertical line at x = 0.5 using a dotted line style and
label it as 'x = 0.5'.
#  d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.
#  e) Set the title of the plot as 'Advanced Scatter Plot of Random
Values'.
#  f) Display a legend for the scatter plot, the horizontal line, and
the vertical line.

x = np.random.random(100)
y = np.random.random(100)

#  a) Create a scatter plot using x and y, setting the color of the
points to red and the marker style to 'o'.

plt.scatter(x, y, color='red', marker='o')

#  b) Add a horizontal line at y = 0.5 using a dashed line style and
label it as 'y = 0.5'.

plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')

#  c) Add a vertical line at x = 0.5 using a dotted line style and
label it as 'x = 0.5'.

plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

#  d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

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

#  e) Set the title of the plot as 'Advanced Scatter Plot of Random
Values'.

plt.title('Advanced Scatter Plot of Random Values')

#  f) Display a legend for the scatter plot, the horizontal line, and
the vertical line.

plt.legend()
```

```
plt.show()
```



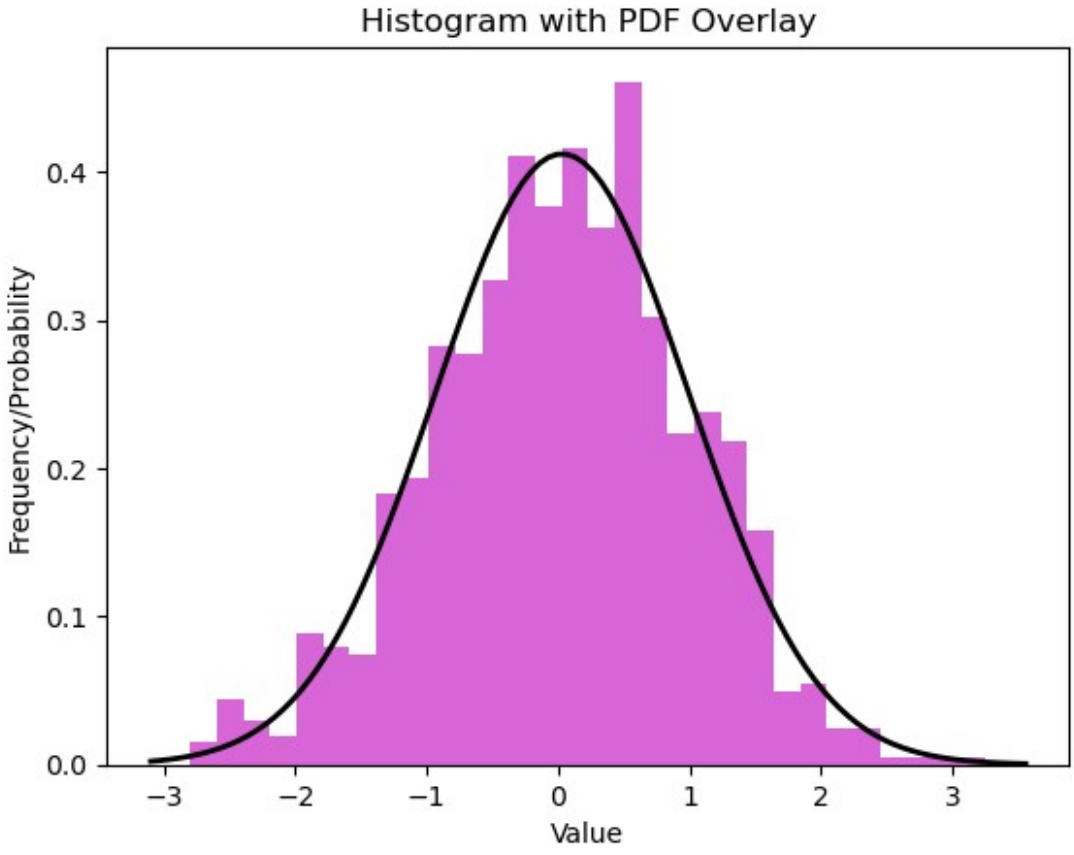

```
#Q14.Create a time -series dataset in a Pandas DataFrame with columns:
'Date', 'Tempreature','Humidity' and perform the following tasks using

#    Matplotlib:

#    a) Plot the 'Tempreature' and 'Humidity' on the same plot with
different y-axes(left y - axis for 'Tempreature' and right y-axis for
'Humidity')
#    b) Label the x-axis as 'Date'
#    c) Set the title of the plot as 'Tempreature and Humidity Over
Time'.


# Create a time-series dataset
dates = pd.date_range(start='2023-01-01', periods=100, freq='D')
temperature = np.random.uniform(low=15, high=35, size=100)
humidity = np.random.uniform(low=30, high=90, size=100)
```

```
data = pd.DataFrame({'Date': dates,'Temperature':
temperature,'Humidity': humidity})


# a) Plot the 'Temperature' and 'Humidity' on the same plot with
different y-axes
fig, ax1 = plt.subplots()

# Plot temperature with the left y-axis
ax1.plot(data.index, data['Temperature'], color='tab:red',
label='Temperature')
ax1.set_ylabel('Temperature (°C)', color='tab:red')
ax1.tick_params(axis='y', labelcolor='tab:red')

# Create a second y-axis to plot humidity
ax2 = ax1.twinx()
ax2.plot(data.index, data['Humidity'], color='tab:blue',
label='Humidity')
ax2.set_ylabel('Humidity (%)', color='tab:blue')
ax2.tick_params(axis='y', labelcolor='tab:blue')

# b) Label the x-axis as 'Date'
ax1.set_xlabel('Date')

# c) Set the title of the plot as 'Temperature and Humidity Over Time'
plt.title('Temperature and Humidity Over Time')

# Display the plot
plt.show()
```



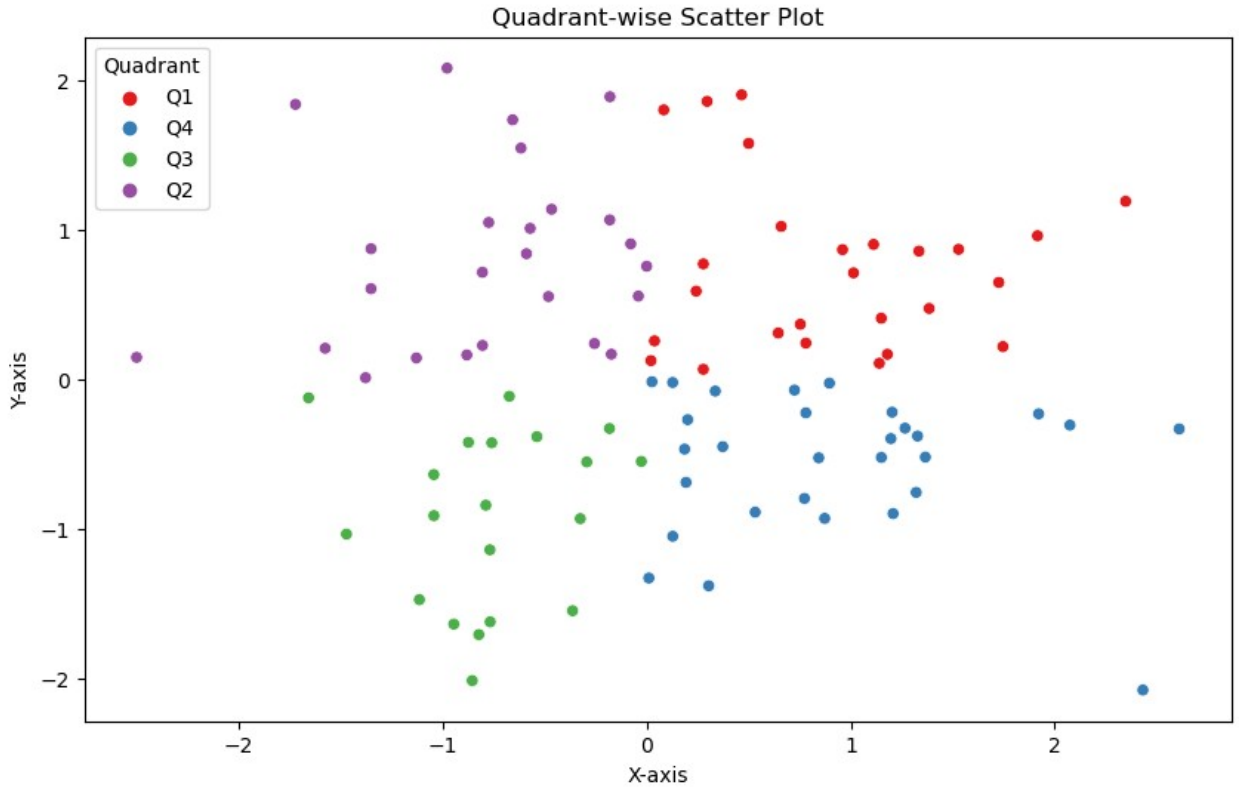

```
#Q15. Create a NumPy array data containing 1000 samples from a normal
distribution. Perform the following
#     tasks using Matplotlib:

#     a) Plot a histogram of the data with 30 bins.
#     b) Overlay a line plot representing the normal distribution's
probability density function (PDF).
#     c) Label the x-axis as 'Value' and the y-axis as
'Frequency/Probability'.
#     d) Set the title of the plot as 'Histogram with PDF Overlay'.


# Create a NumPy array containing 1000 samples from a normal
distribution
data = np.random.randn(1000)

# a) Plot a histogram of the data with 30 bins
count, bins, ignored = plt.hist(data, bins=30, density=True,
alpha=0.6, color='m')

# b) Overlay a line plot representing the normal distribution's
probability density function (PDF)
```

```
mu, std = norm.fit(data)  # Fit a normal distribution to the data
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)

# c) Label the x-axis as 'Value' and the y-axis as
'Frequency/Probability'
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

# d) Set the title of the plot as 'Histogram with PDF Overlay'
plt.title('Histogram with PDF Overlay')

# Display the plot
plt.show()
```



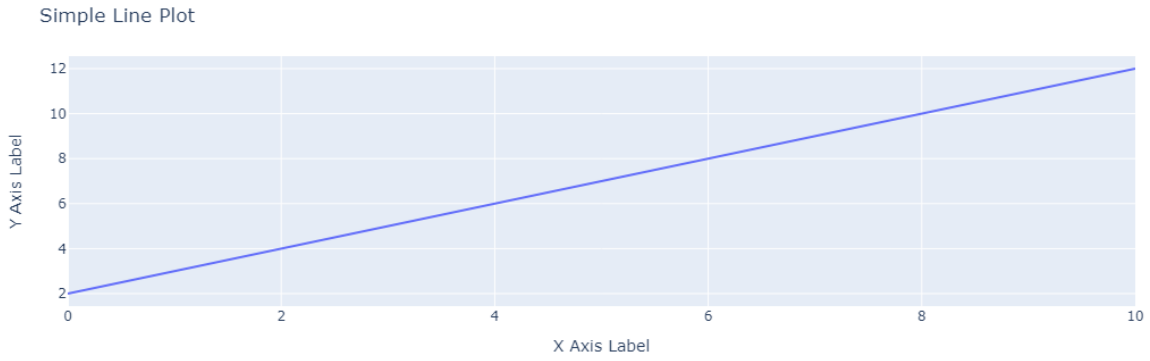

```
#Q16. Set the title of the plot as 'Histogram with PDF Overlay'.

plt.title('Histogram with PDF Overlay')
```

```
#see the above plot for the result


#Q17. Create a Seaborn scatter plot of two random arrays, color points
based on their position relative to the
#     origin (quadrants), add a legend, label the axes, and set the
title as 'Quadrant-wise Scatter Plot'.


# Create two random arrays
x = np.random.randn(100)
y = np.random.randn(100)

# Determine the quadrant for each point
def get_quadrant(x, y):
    if x > 0 and y > 0:
        return 'Q1'
    elif x < 0 and y > 0:
        return 'Q2'
    elif x < 0 and y < 0:
        return 'Q3'
    else:
        return 'Q4'

quadrants = [get_quadrant(xi, yi) for xi, yi in zip(x, y)]

# Create a DataFrame
df = pd.DataFrame({'x': x, 'y': y, 'Quadrant': quadrants})

# Create a Seaborn scatter plot
plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=df, x='x', y='y', hue='Quadrant',
palette='Set1')

# Add a legend
plt.legend(title='Quadrant')

# Label the axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set the title
plt.title('Quadrant-wise Scatter Plot')

# Show the plot
plt.show()
```




```
#Q18. With Bokeh, plot a line chart of a sine wave function, add grid
lines, label the axes, and set the title as 'Sine
#     Wave Function'


# Generate data
x = np.arange(0, 5*np.pi, 0.1)
y = np.sin(x)

# Create a new plot with title and axis labels
p = figure(title="Sine Wave Function", x_axis_label='x',
y_axis_label='sin(x)', width=800, height=400)

# Add a line renderer with legend and line thickness
p.line(x, y, legend_label="Sine Wave", line_width=2)

# Show grid lines
p.grid.grid_line_color = 'gray'
p.grid.grid_line_alpha = 0.5

# Show the plot
show(p)
```

```
#Q19.Using Bokeh, generate a bar chart of randomly generated
categorical data, color bars based on their
#    values, add hover tooltips to display exact values, label the
axes, and set the title as 'Random Categorical
#    Bar Chart'



#Q20. Using Plotly, create a basic line plot of a randomly generated
dataset, label the axes, and set the title as
#    'Simple Line Plot'


# Generate random data
x = np.linspace(0, 10, 100)
y = x+2

# Create a line plot
fig = px.line(x=x,y=y)

# Update layout with title and axis labels
fig.update_layout(title='Simple Line Plot',xaxis_title='X Axis
Label',yaxis_title='Y Axis Label')

# Show the plot
fig.show()
```




```
#Q21.Using Plotly, create an interactive pie chart of randomly
generated data, add labels and percentages, set
#   the title as 'Interactive Pie Chart'

# Generate random data
labels = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(1, 100, size=len(labels))
```

```
# Create the pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values ,hole=0.3)])

# Update layout for title and labels
fig.update_layout(title_text='Interactive Pie
Chart',annotations=[dict(text='Pie', x=0.5, y=0.5, font_size=20,
showarrow=False)])

# Show the figure
fig.show()
```