Data Toolkit Assignment

Question_1st:-Demonstrate three different methods for creating identical 2D arrays in NumPy. Provide the code for eachmethod and the final output after each method.

In [4]:
```
# Method 1: Using np.array():-You can create a 2D array directly by passing a list of lists to the

import numpy as np

array1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Array created using np.array():\n", array1)

print("========================")

# Method 2: Using np.full():-This method creates a 2D array filled with a specified value. You speci

array2 = np.full((3, 3), [[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Array created using np.full():\n", array2)

print("========================")

# Method 3: Using np.zeros() with slicing:-First, create a 2D array of zeros with the desired shape,

array3 = np.zeros((3, 3), dtype=int)
array3[:] = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print("Array created using np.zeros() with slicing:\n", array3)
```

```
Array created using np.array():
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
========================
Array created using np.full():
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
========================
Array created using np.zeros() with slicing:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Question_2nd:- Using the Numpy function, generate an array of 100 evenly spaced numPers Petween 1 and 10 and Reshape that wD array into a 2D array

In [7]:
```
import numpy as np

# Step 1: Generate 100 evenly spaced numbers between 1 and 10
array_1d = np.linspace(1, 10, 100)

# Step 2: Reshape the 1D array into a 2D array (10x10)
array_2d = array_1d.reshape(10, 10)

print("1D Array of 100 evenly spaced numbers between 1 and 10:\n", array_1d)
print("\nReshaped 2D Array (10x10):\n", array_2d)
```

```
1D Array of 100 evenly spaced numbers between 1 and 10:
 [ 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.        ]

Reshaped 2D Array (10x10):
 [[ 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.        ]]
```

Question_3rd:- Explain the following terms:

The difference in nparray, npasarray and npasanyarrayX

The difference between Deep copy and shallow copyX

In [4]:

```
# 1st:- np.array():-np.array(): This function always creates a new array. If you pass an existing ar

import numpy as np

list_data = [1, 2, 3]
array1 = np.array(list_data)
print("Using np.array():", array1)

print("==========================")

# 2nd:-npasarray():-This function converts the input to an array, but it does not create a copy if t

array2 = np.asarray(array1)
print("Using np.asarray():", array2)

print("==========================")

# 3rd:-np.asanarray():- This function is similar to np.asarray(), but it preserves subclasses. For e

matrix_data = np.matrix([[1, 2], [3, 4]])
array3 = np.asanyarray(matrix_data)
print("Using np.asanyarray():\n", array3)
```

```
Using np.array(): [1 2 3]
==========================
Using np.asarray(): [1 2 3]
==========================
Using np.asanyarray():
 [[1 2]
 [3 4]]
```

Question_4th:-Generate a 3x3 array with random floating-point numPers Petween 5 and 20 9hen, round each numPer inthe array to 2 decimal places

In [6]:

```
import numpy as np

# Step 1: Generate a 3x3 array with random floating-point numbers between 5 and 20
random_array = np.random.uniform(5, 20, (3, 3))

# Step 2: Round each number in the array to 2 decimal places
rounded_array = np.round(random_array, 2)
```

```
print("3x3 Array with random floating-point numbers between 5 and 20:\n", random_array)
print("\nRounded 3x3 Array to 2 decimal places:\n", rounded_array)
```

```
3x3 Array with random floating-point numbers between 5 and 20:
 [[ 6.64242178 15.00523942  5.48201415]
 [ 5.60472063 17.454048   17.30607245]
 [ 5.92602993 18.60470873  7.27726084]]

Rounded 3x3 Array to 2 decimal places:
 [[ 6.64 15.01  5.48]
 [ 5.6  17.45 17.31]
 [ 5.93 18.6   7.28]]
```

Question_5th:- Create a NumPy array with random integers Petween 1 and 10 of shape (5,6)) After creating the array perform the following operations:

a)Extract all even integers from array.

b)Extract all odd integers from array

In [11]:
```
import numpy as np

# Create a random integer array with sahpe 5 row and 6 col
array=np.random.randint(1,10,size=(5,6))
print(array)
print("====================================")

even_integers = array[array % 2 == 0]
print("Even integers:", even_integers)

print("====================================")

odd_integers = array[array % 2 != 0]
print("odd integers:", odd_integers)
```

```
[[7 3 1 9 9 9]
 [9 2 2 8 1 1]
 [3 9 8 1 1 1]
 [7 5 2 3 4 3]
 [5 9 1 4 6 2]]
==================================
Even integers: [2 2 8 8 2 4 4 6 2]
==================================
odd integers: [7 3 1 9 9 9 9 1 1 3 9 1 1 1 7 5 3 3 5 9 1]
```

Question_6th:- Create a D NumPy array of shape (3, 3, 3) containing random integers Petween 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 arrayX

In [23]:
```
import numpy as np

# Create a 3D NumPy array with random integers between 1 and 10
array = np.random.randint(1, 10, size=(3, 3, 3))
print("3D Array:\n", array)
print("====================")



# Find indices of the maximum values along the third axis
max_indices = np.argmax(array, axis=2)
print("Indices of maximum values along each depth level:\n", max_indices)

print("Another 3D array")
```

```
# Another 3D array
array2=np.random.randint(1,10,size=(3,3,3))
print(array2)

print(" multhiplication of both array")
# multhiplication of both array

array_multiplication=array1*array2
print(array_multiplication)
```

```
3D Array:
 [[[7 9 9]
  [3 5 2]
  [4 2 6]]

 [[3 7 9]
  [8 3 7]
  [9 3 1]]

 [[3 3 1]
  [2 7 7]
  [3 4 5]]]
====================
Indices of maximum values along each depth level:
 [[1 1 2]
 [2 0 0]
 [0 1 2]]
Another 3D array
[[[5 5 3]
  [9 5 9]
  [8 3 5]]

 [[8 3 7]
  [9 9 3]
  [2 3 3]]

 [[4 3 2]
  [9 9 8]
  [7 7 3]]]
 multhiplication of both array
[[[5 5 3]
  [0 5 9]
  [0 0 5]]

 [[8 3 7]
  [0 9 3]
  [0 0 3]]

 [[4 3 2]
  [0 9 8]
  [0 0 3]]]
```

Question_7th:-□ 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 taPle attriPutes and data types of each column□

In [44]:

```
import pandas as pd

# Display the original DataFrame
df=pd.read_csv("People Data.csv")
df
```

Out [44]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation offi
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	Dancer
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	(599)782-0605	25-11-2013	Copy
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Counselling psychologist
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biomedical er
...	...	...	...	...	...	...	...	...	...

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel offi
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/r surveyor
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-1971	Ambulance pe
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learnin disability

1000 rows × 10 columns

In [45]:
```
# Clean the 'Phone' column by removing non-numeric characters
df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True)
df
```

Out [45]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	8571398239	27-01-2014	Probation offic
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	Dancer
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	5997820605	25-11-2013	Copy
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Counselling psychologist
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	39041716353010	15-04-1923	Biomedical eng
...	...	...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	0217752933	05-01-1959	Personnel offic
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	0011497107799721	06-10-2001	Education administrator
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	17507744128332 65	13-05-1918	Commercial/re surveyor
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	9152922254	31-08-1971	Ambulance per
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079752542467259	24-01-1947	Nurse, learning disability

1000 rows × 10 columns

In [46]:
```
# Convert the 'Phone' column to a numeric data type
df['Phone'] = pd.to_numeric(df['Phone'])
df
```

Out [46]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job T
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	8.571398e+09	27-01-2014	Probation officer
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-	Dancer

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job T
								1931	
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	5.997821e+09	25-11-2013	Copy
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Counselling psychologist
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	3.904172e+13	15-04-1923	Biomedical engineer
...	...	...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	2.177529e+08	05-01-1959	Personnel officer
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	1.149711e+13	06-10-2001	Education administrator
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	1.750774e+15	13-05-1918	Commercial/residen surveyor
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	9.152922e+09	31-08-1971	Ambulance person
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	7.975254e+13	24-01-1947	Nurse, learning disability

1000 rows × 10 columns

In [47]:
```
# Display table attributes and data types of each column
print("\nTable attributes and data types:")
print(df.info())
```

```
Table attributes and data types:
<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          979 non-null    float64
 7   Date of birth  1000 non-null   object 
 8   Job Title      1000 non-null   object 
 9   Salary         1000 non-null   int64  
dtypes: float64(1), int64(2), object(7)
memory usage: 78.2+ KB
None
```

Question_8th:- Perform the following terms using people dataset:

a) Read the 'dataYcsv' file using pandas, skipping the first 50 rows.

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

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

d) Extract the ‘Salary" column as a Series and display its last 5 valuesX

In [54]:
```
# a) Read the 'dataYcsv' file using pandas, skipping the first 50 rows.

import pandas as pd

# Read the CSV file, skipping the first 50 rows
df1 = pd.read_csv("People Data.csv",skiprows=50)
df1
```

Out [54]:

	50	afF3018e9cdd1dA	George	Mercer	Female	douglascontreras@example.net	+1-326-669-0118x4341	11-09-1941	Human r
0	51	CccE5DAb6E288e5	Jo	Zavala	Male	pamela64@example.net	001-859-448-9935x54536	23-11-1992	Nurse, adult
1	52	DfBDc3621D4bcec	Joshua	Carey	Female	dianashepherd@example.net	001-274-739-8470x814	07-01-1915	Seismic interp
2	53	f55b0A249f5E44D	Rickey	Hobbs	Female	ingramtiffany@example.org	241.179.9509x498	01-07-1910	Barrister
3	54	Ed71DcfaBFd0beE	Robyn	Reilly	Male	carriecrawford@example.org	207.797.8345x6177	27-07-1982	Engineer, struc
4	55	FDaFD0c3f5387EC	Christina	Conrad	Male	fuentesclaudia@example.net	001-599-042-7428x143	06-01-1998	Producer, radio
...	...	...	...	...	...	...	...	...	...
945	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel offic
946	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator
947	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/re surveyor
948	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-1971	Ambulance pe
949	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learning disability

950 rows × 10 columns

In [71]:
```
# b. Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the file.

df1=pd.read_csv("People Data.csv",usecols=["Last Name","Gender","Email","Phone","Salary"])
df1
```

Out [71]:

	Last Name	Gender	Email	Phone	Salary
0	Mahoney	Male	pwarner@example.org	857.139.8239	90000
1	Rivers	Female	fergusonkatherine@example.net	NaN	80000
2	Lowery	Female	fhoward@example.org	(599)782-0605	50000
3	Hooper	Male	zjohnston@example.com	NaN	65000
4	Rice	Female	elin@example.net	(390)417-1635x3010	100000
...	...	...	...	...	...
995	Bryant	Female	lyonsdaisy@example.net	021.775.2933	90000
996	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	50000
997	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	60000
998	Phelps	Male	wanda04@example.net	(915)292-2254	100000
999	Tran	Female	deannablack@example.org	079.752.5424x67259	90000

1000 rows × 5 columns

In [76]:
```
# c) Display the first 10 rows of the filtered dataset.
df=pd.read_csv("People Data.csv")

df2=df[(df['Salary'] > 50000)]

df2.head()
```

Out [76]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Sal
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation officer	9000

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Sal
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	fergusonkatherine@example.net	NaN	26-07-1931	Dancer	8000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	zjohnston@example.com	NaN	17-11-2012	Counselling psychologist	6500
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biomedical engineer	1000
6	7	efeb05c7Cc94EA3	Ernest	Hoffman	Male	jeffharvey@example.com	093.655.7480x7895	22-12-1984	Health visitor	6000

In [77]:
```
# d. Extract the 'Salary'' column as a Series and display its last 5 valuesX
import pandas as pd

# Read the CSV file
df = pd.read_csv("People Data.csv")

# Extract the 'Salary' column as a Series
salary_series = df['Salary']

# Display the last 5 values of the 'Salary' Series
print(salary_series.tail(5))
```

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

Question_9th:- 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 Pe less than 85000

In [81]:
```
import pandas as pd

# Read the CSV file
df = pd.read_csv("People Data.csv")

# Filter rows based on the conditions
filtered_df = df[(df['Last Name']=='Duke') & (df['Gender']=='Female') & (df['Salary'] < 80000)]

# Display the filtered DataFrame
filtered_df
```

Out [81]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job Title	Salary
45	46	99A502C175C4EBd	Olivia	Duke	Female	diana26@example.net	001-366-475-8607x04350	13-10-1934	Dentist	60000
210	211	DF17975CC0a0373	Katrina	Duke	Female	robin78@example.com	740.434.0212	21-09-1935	Producer, radio	50000
457	458	dcE1B7DE83c1076	Traci	Duke	Female	perryhoffman@example.org	+1-903-596-0995x489	11-02-1997	Herbalist	50000
729	730	c9b482D7aa3e682	Lonnie	Duke	Female	kevinkramer@example.net	982.692.6257	12-05-2015	Nurse, adult	70000

Question_10th:-Create a 7*5. Dataframe in Pandas using a series

generated from 35. random integers Petween 1 to 6)?

In [90]:
```
import pandas as pd
import numpy as np

# Create a 2D numpy array with random integers between 1 and 6, shape 7x5
array = np.random.randint(1, 6, size=(7, 5))
print("Array:\n", array)

# Create a DataFrame from the 2D numpy array
df = pd.DataFrame(array, columns=['Column1', 'Column2', 'Column3', 'Column4', 'Column5'])
print("\nDataFrame:\n", df)

# Flatten the DataFrame to a 1D array and then create a Series
flattened_series = pd.Series(df.values.flatten())
print("\nFlattened Series:\n", flattened_series)
```

```
Array:
 [[4 5 1 2 4]
 [1 3 3 2 4]
 [2 5 4 5 3]
 [2 4 2 4 2]
 [1 2 3 2 5]
 [1 4 5 3 4]
 [4 3 3 2 2]]

DataFrame:
    Column1  Column2  Column3  Column4  Column5
0        4        5        1        2        4
1        1        3        3        2        4
2        2        5        4        5        3
3        2        4        2        4        2
4        1        2        3        2        5
5        1        4        5        3        4
6        4        3        3        2        2

Flattened Series:
 0     4
1     5
2     1
3     2
4     4
5     1
6     3
7     3
8     2
9     4
10    2
11    5
12    4
13    5
14    3
15    2
16    4
17    2
18    4
19    2
20    1
21    2
22    3
23    2
24    5
25    1
26    4
27    5
28    3
29    4
30    4
31    3
32    3
33    2
34    2
dtype: int32
```

Question_11th:-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

In [96]:
```
# a) The first Series should contain random numbers ranging from 10 to 50.

import pandas as pd
import numpy as np

# Create a Series with random integers between 10 and 50
random_series_1st = pd.Series(np.random.randint(10, 50, size=10))

# Display the Series
print("Random Series_1st:\n", random_series_1st)
```

```
Random Series_1st:
 0    32
1    30
2    33
3    37
4    47
5    40
6    18
7    20
8    11
9    37
dtype: int32
```

In [97]:
```
# b. The second Series should contain random numbers ranging from 100 to 1000.

import pandas as pd
import numpy as np

# Create a Series with random integers between 10 and 50
random_series_2nd = pd.Series(np.random.randint(100, 1000, size=10))

# Display the Series
print("Random Series_2nd:\n", random_series_2nd)
```

```
Random Series_2nd:
 0    778
1    259
2    348
3    773
4    636
5    661
6    526
7    303
8    164
9    996
dtype: int32
```

In [98]:
```
# Create a DataFrame by 'joining these Series by column, and, change the names of the columns to 'co

# Combine the Series into a DataFrame
df = pd.DataFrame({
    'col1': random_series_1st,
    'col2': random_series_2nd,
})



# Display the DataFrame
print("DataFrame:\n", df)
```

```
DataFrame:
    col1  col2
0    32   778
1    30   259
2    33   348
3    37   773
4    47   636
5    40   661
6    18   526
7    20   303
8    11   164
9    37   996
```

Question_12th:-g 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.

In [101]:
```
# a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

# a) Delete the 'Email', 'Phone', and 'Date of birth' columns
df=pd.read_csv("People Data.csv")
df = df.drop(columns=['Email', 'Phone', 'Date of birth'])
df
```

Out [101]:

	Index	User Id	First Name	Last Name	Gender	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	Dancer	80000
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	Copy	50000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	Counselling psychologist	65000
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	Biomedical engineer	100000
...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	Personnel officer	90000
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	Education administrator	50000
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	Commercial/residential surveyor	60000
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	Ambulance person	100000
999	1000	8b756f6231DDC6e	Lee	Tran	Female	Nurse, learning disability	90000

1000 rows × 7 columns

In [102]:
```
# b) Delete the rows containing any missing values.

import pandas as pd

# Load the dataset
df = pd.read_csv("People Data.csv")


# Delete the rows containing any missing values
df = df.dropna()
df
```

Out [102]:

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	pwarner@example.org	857.139.8239	27-01-2014	Probation officer
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	fhoward@example.org	(599)782-0605	25-11-2013	Copy
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	elin@example.net	(390)417-1635x3010	15-04-1923	Biomedical enginee
5	6	aF75e6dDEBC5b66	Sherry	Caldwell	Male	kaitlin13@example.net	8537800927	06-08-1917	Higher education lecturer
6	7	efeb05c7Cc94EA3	Ernest	Hoffman	Male	jeffharvey@example.com	093.655.7480x7895	22-12-1984	Health visitor
...	...	...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	lyonsdaisy@example.net	021.775.2933	05-01-1959	Personnel officer
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	dariusbryan@example.com	001-149-710-7799x721	06-10-2001	Education administrator
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	georgechan@example.org	+1-750-774-4128x33265	13-05-1918	Commercial/reside surveyor
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	wanda04@example.net	(915)292-2254	31-08-	Ambulance person

	Index	User Id	First Name	Last Name	Gender	Email	Phone	Date of birth	Job
								1971	
999	1000	8b756f6231DDC6e	Lee	Tran	Female	deannablack@example.org	079.752.5424x67259	24-01-1947	Nurse, learning disability

979 rows × 10 columns

In [104]:
```
# d) Print the final output also.

import pandas as pd

# Load the dataset
df = pd.read_csv("People Data.csv")

# a) Delete the 'Email', 'Phone', and 'Date of birth' columns
df = df.drop(columns=['Email', 'Phone', 'Date of birth'])

# b) Delete the rows containing any missing values
df = df.dropna()

# d) Print the final output
df
```

Out [104]:

	Index	User Id	First Name	Last Name	Gender	Job Title	Salary
0	1	8717bbf45cCDbEe	Shelia	Mahoney	Male	Probation officer	90000
1	2	3d5AD30A4cD38ed	Jo	Rivers	Female	Dancer	80000
2	3	810Ce0F276Badec	Sheryl	Lowery	Female	Copy	50000
3	4	BF2a889C00f0cE1	Whitney	Hooper	Male	Counselling psychologist	65000
4	5	9afFEafAe1CBBB9	Lindsey	Rice	Female	Biomedical engineer	100000
...	...	...	...	...	...	...	...
995	996	fedF4c7Fd9e7cFa	Kurt	Bryant	Female	Personnel officer	90000
996	997	ECddaFEDdEc4FAB	Donna	Barry	Female	Education administrator	50000
997	998	2adde51d8B8979E	Cathy	Mckinney	Female	Commercial/residential surveyor	60000
998	999	Fb2FE369D1E171A	Jermaine	Phelps	Male	Ambulance person	100000
999	1000	8b756f6231DDC6e	Lee	Tran	Female	Nurse, learning disability	90000

1000 rows × 7 columns

Question_13th:-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'.

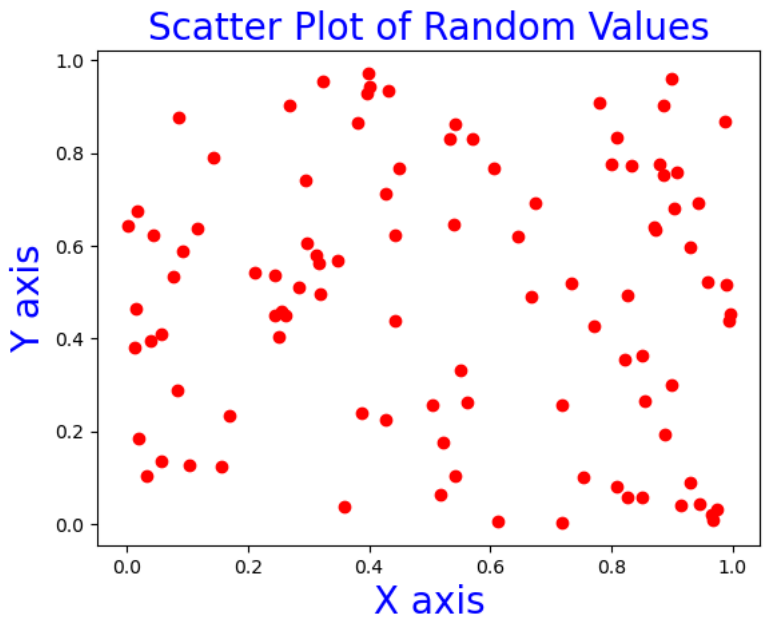
c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x = 0.5'.

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'.

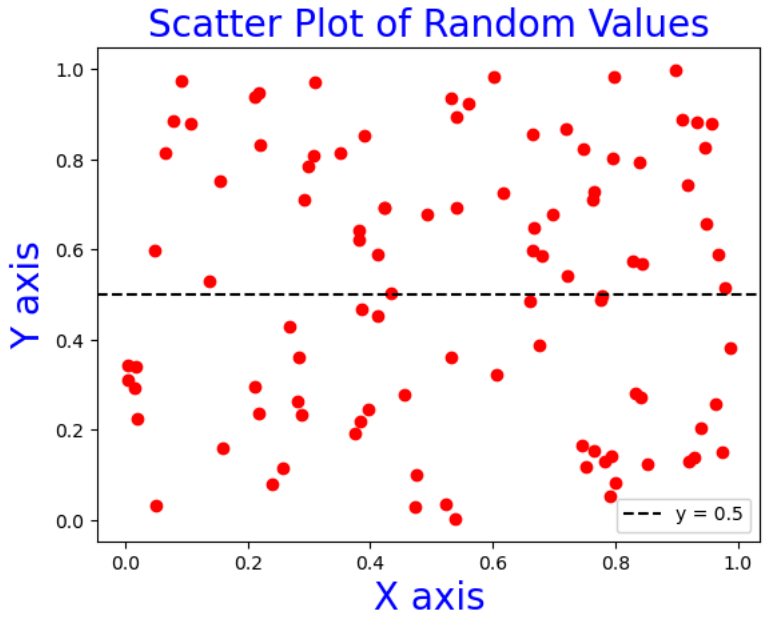
f) Display a legend for the scatter plot, the horizontal line, and the vertical line.

In [110]:
```
# a) Create a scatter plot using x and y, setting the color of the points to red and the marker styl

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

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



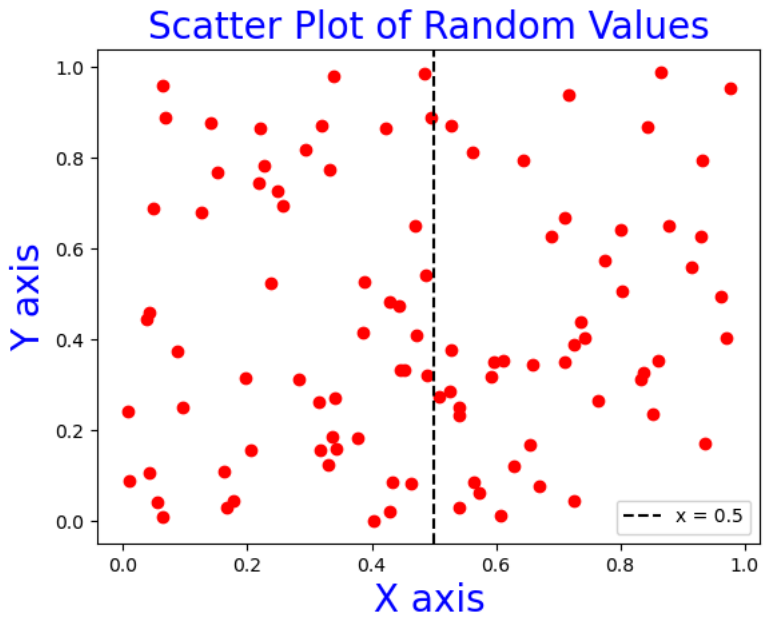
In [115]:
```
# b) Add a horizontal line at y = 0.5 using a dashed line style and label it as 'y = 0.5'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='black', linestyle='--', label='y = 0.5')

# Add a legend to show the label for the horizontal line
plt.legend()
```

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

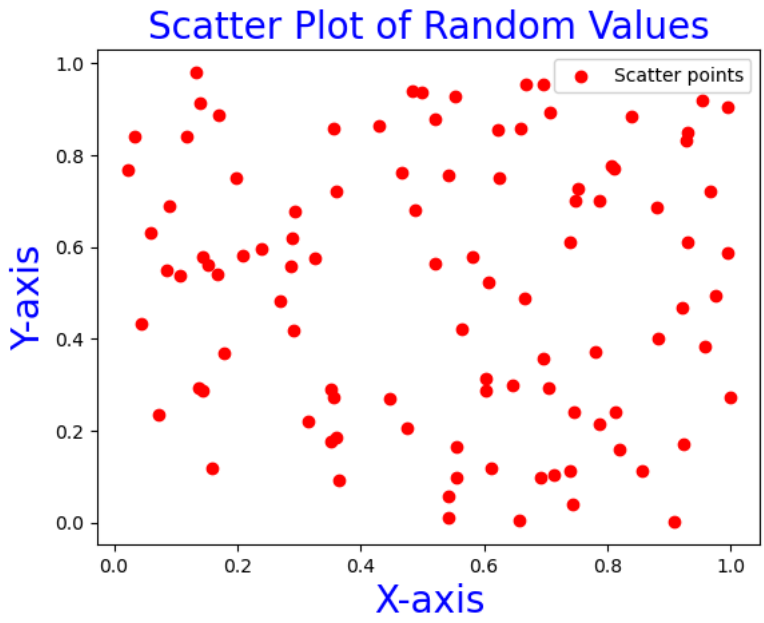


In [114]:

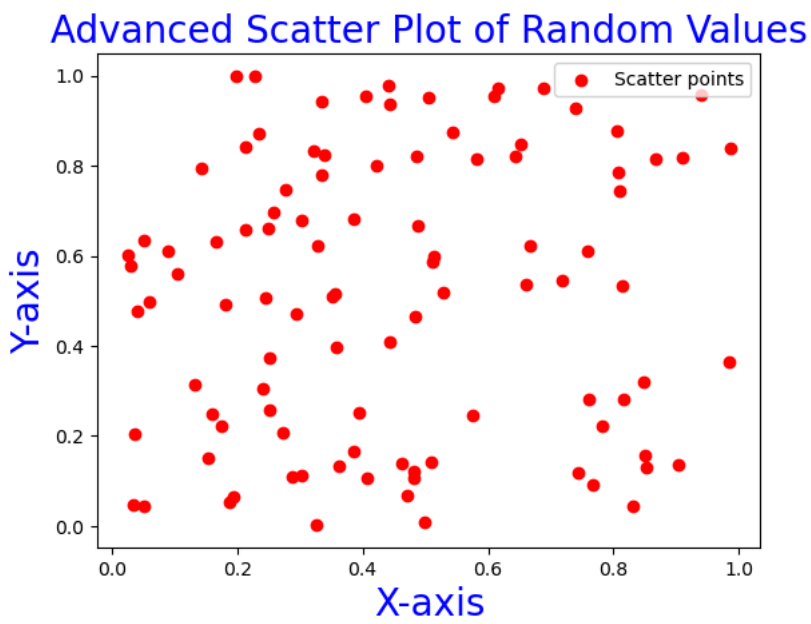
```
# c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x = 0.5'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y, color='red', marker='o')

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X axis', color='blue', fontsize=20)
plt.ylabel('Y axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axvline(x=0.5, color='black', linestyle='--', label='x = 0.5')

# Add a legend to show the label for the horizontal line
plt.legend()

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



In [120]:
```
# d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)
plt.title('Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a legend to show the label for the scatter points
plt.legend()

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

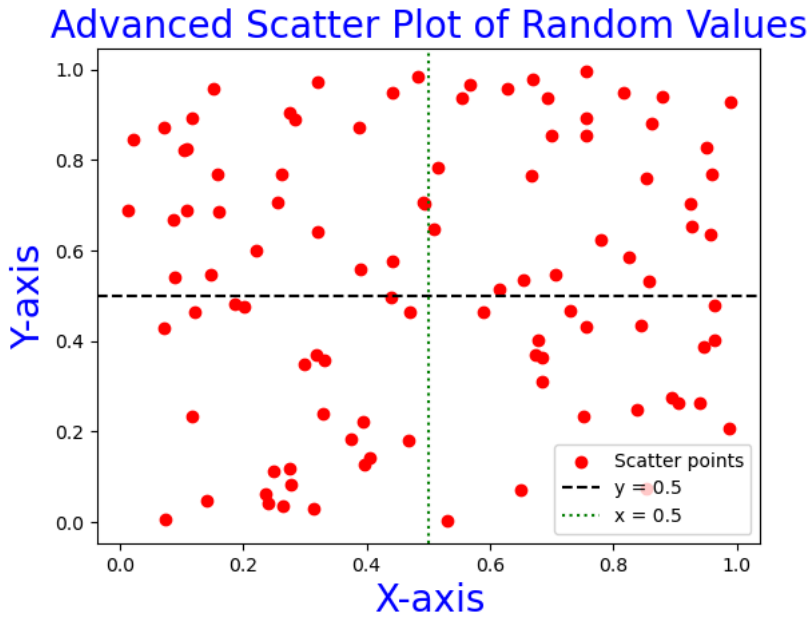


In [122]:
```
# e) Set the title of the plot as 'Advanced Scatter Plot of Random Values'.

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a legend to show the label for the scatter points
plt.legend()

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



In [123]:

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

import numpy as np
import matplotlib.pyplot as plt

# Create two NumPy arrays with 100 random float values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot with a single label
plt.scatter(x, y, color='red', marker='o', label="Scatter points")

# Add labels and title for clarity with specified colors and sizes
plt.xlabel('X-axis', color='blue', fontsize=20)
plt.ylabel('Y-axis', color='blue', fontsize=20)

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values', color='blue', fontsize=20)

# Add a horizontal line at y = 0.5 with dashed line style
plt.axhline(y=0.5, color='black', linestyle='--', label='y = 0.5')

# Add a vertical line at x = 0.5 with dotted line style
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

# Add a legend to show the labels for the scatter plot, horizontal line, and vertical line
plt.legend()

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



Question_14th:-Create a time-series dataset in a Pandas DataFrame with columns: 'Date', 'Temperature', 'Humidity' and Perform the following tasks using Matplotlib:

a) Plot the 'Temperature' and 'Humidity' on the same plot with different y-axes (left y-axis for 'Temperature' and right y-axis for 'Humidity').

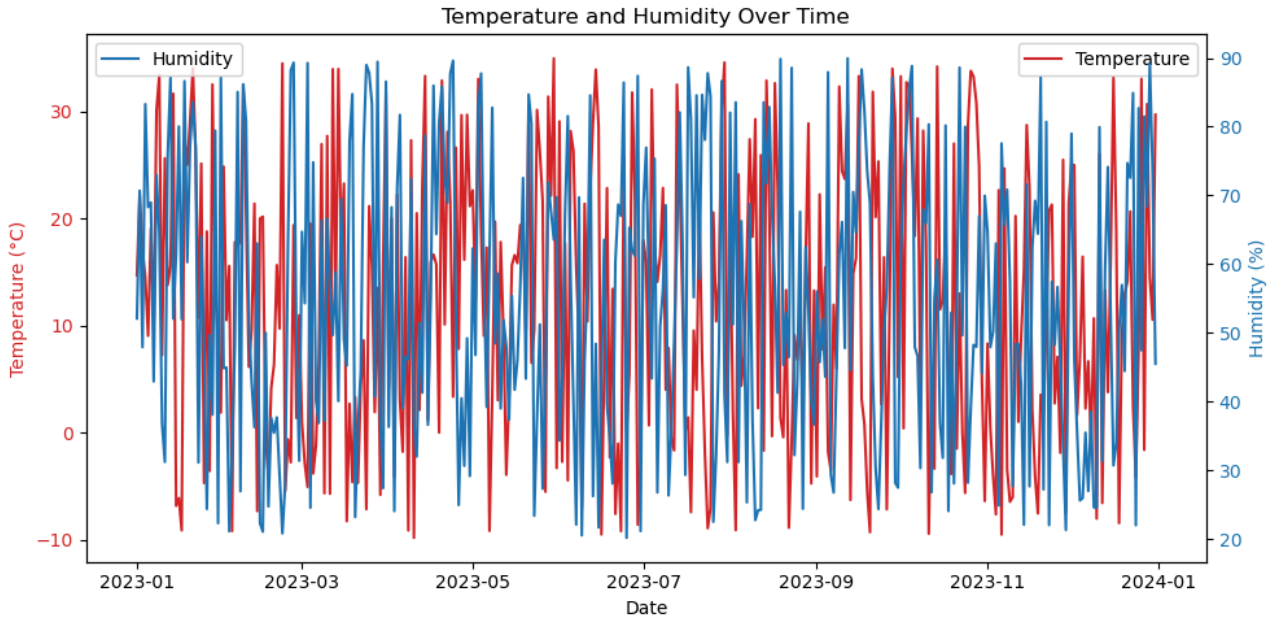
b) Label the x-axis as 'Date'.

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

In [13]:

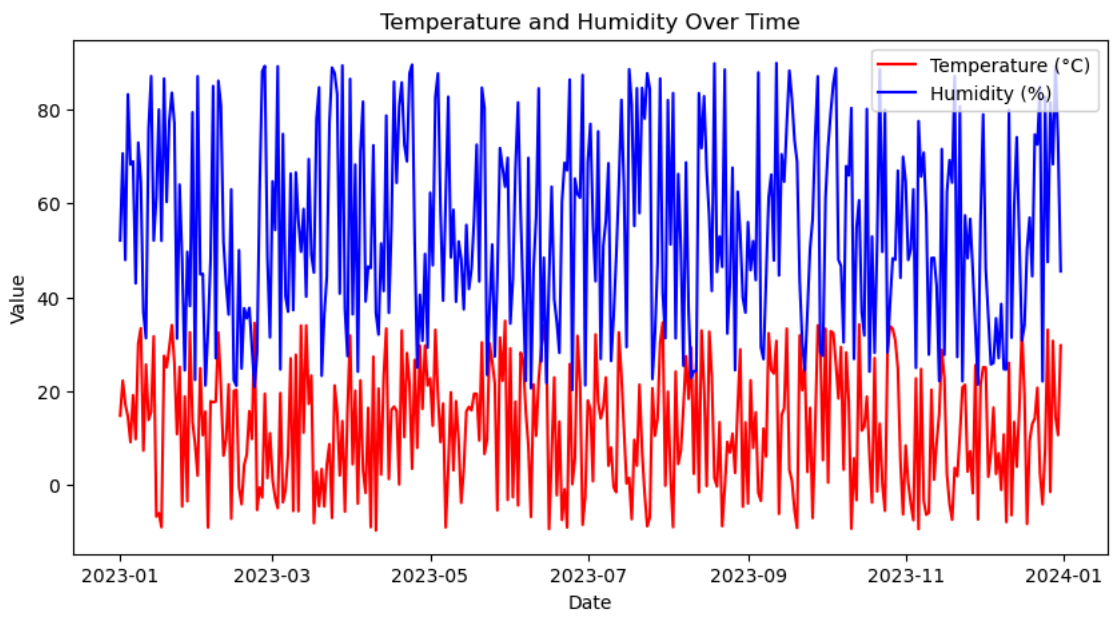
```
# a) Plot the 'Temperature' and 'Humidity' on the same plot with different y-axes (left y-axis for '

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a date range
date_range = pd.date_range(start='2023-01-01', end='2023-12-31', freq='D')

# Generate random data for temperature and humidity
np.random.seed(0)  # for reproducibility
temperature = np.random.uniform(-10,35, size=len(date_range))
humidity = np.random.uniform(20,90, size=len(date_range))

# Create a DataFrame
df = pd.DataFrame({
    'Date': date_range,
    'Temperature': temperature,
    'Humidity': humidity
})

fig, ax1 = plt.subplots(figsize=(10, 5))

# Plot Temperature with the primary y-axis
ax1.plot(df['Date'], df['Temperature'], color='tab:red', label='Temperature')
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature (°C)', color='tab:red')
ax1.tick_params(axis='y', labelcolor='tab:red')
ax1.legend()

# Create a secondary y-axis for the Humidity
ax2 = ax1.twinx()
ax2.plot(df['Date'], df['Humidity'], color='tab:blue', label='Humidity')
```

```
ax2.set_ylabel('Humidity (%)', color='tab:blue')
ax2.tick_params(axis='y', labelcolor='tab:blue')
ax2.legend()
# Add a title
plt.title('Temperature and Humidity Over Time')

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



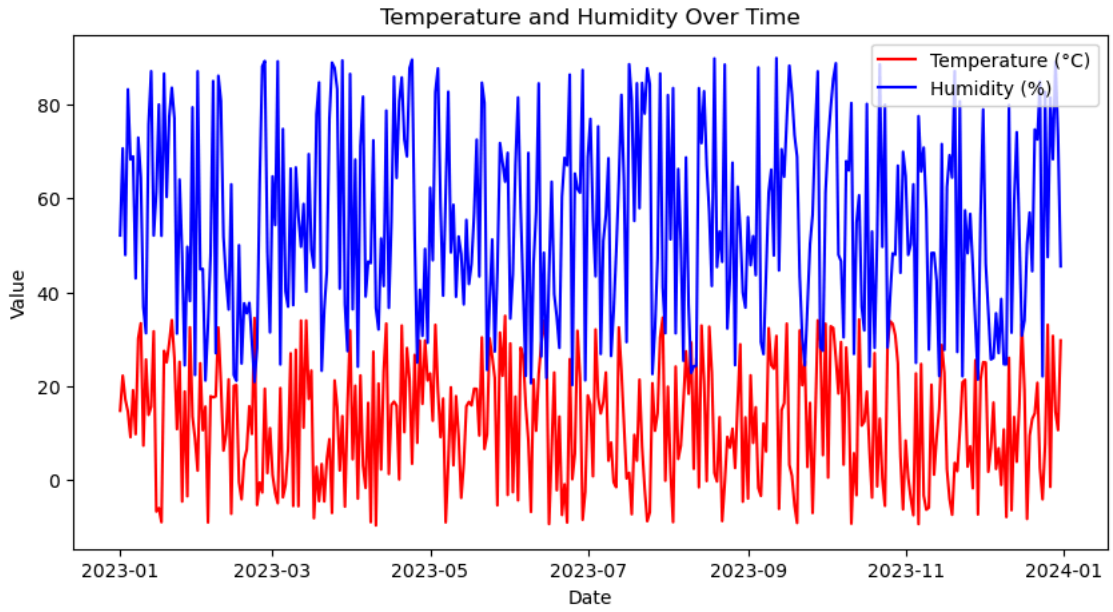
In [8]:

```
# b) Label the x-axis as 'Date'.


import matplotlib.pyplot as plt

# Plot Temperature and Humidity over time
plt.figure(figsize=(10, 5))
plt.plot(df['Date'], df['Temperature'], label='Temperature (°C)', color='red')
plt.plot(df['Date'], df['Humidity'], label='Humidity (%)', color='blue')

# Label the axes
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Temperature and Humidity Over Time')

# Show legend
plt.legend()

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



In [9]:

```
# C. Set the title of the plot as 'Temperature and Humidity Over Time'.

import matplotlib.pyplot as plt

# Plot Temperature and Humidity over time
plt.figure(figsize=(10, 5))
plt.plot(df['Date'], df['Temperature'], label='Temperature (°C)', color='red')
plt.plot(df['Date'], df['Humidity'], label='Humidity (%)', color='blue')

# Label the axes
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Temperature and Humidity Over Time')

# Show legend
plt.legend()

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



Question_15th:- Create a NumPy array data containing 1000 samples from a normal distribution. Perform the following tasks using Matplotlib:

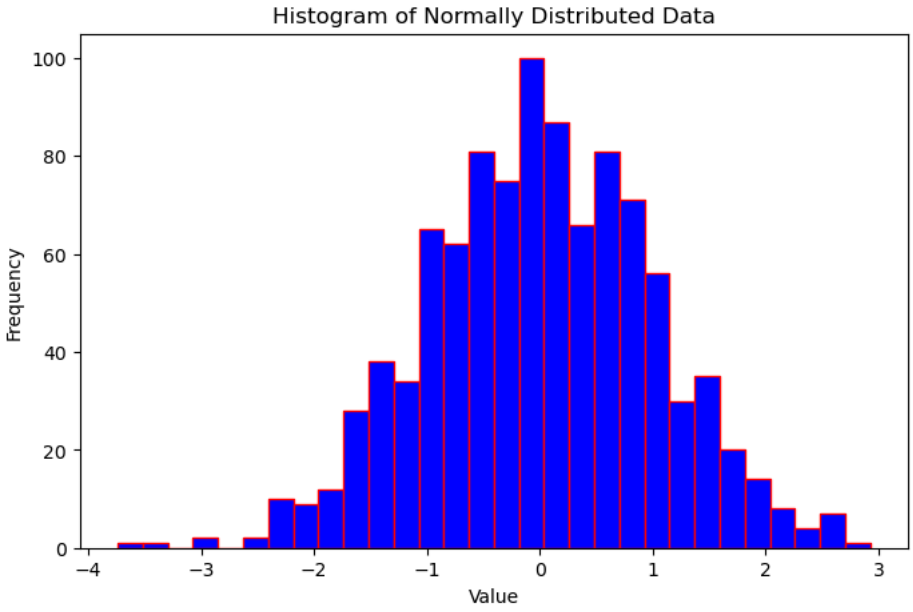
a) Plot a histogram of the data with 30 bins.

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'.

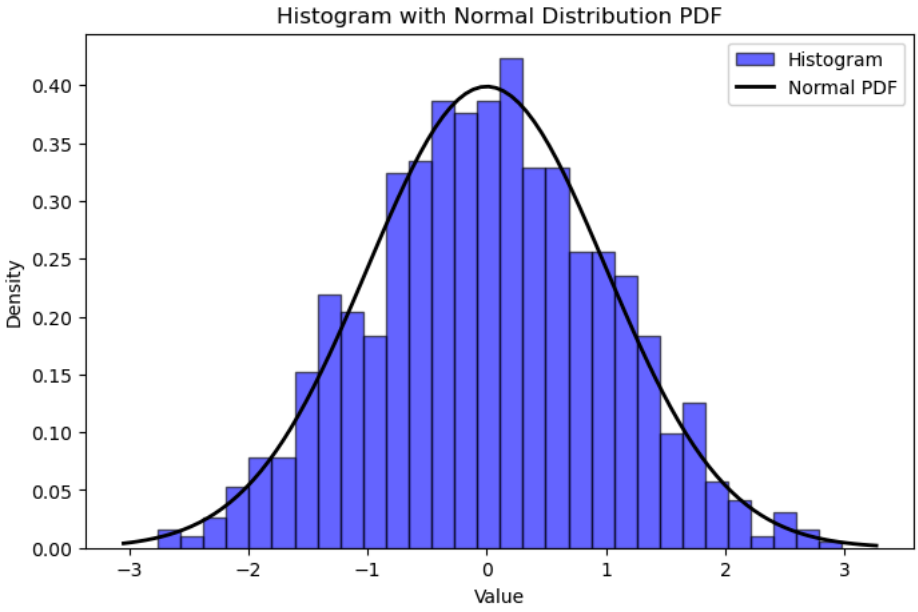
d) Set the title of the plot as 'Histogram with PDF Overlay'.

In [17]:
```
# a) Plot a histogram of the data with 30 bins.

import numpy as np

# Generate 1000 samples from a normal distribution
data = np.random.normal(0,1, size=1000)

import matplotlib.pyplot as plt

# Plot the histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, color='blue', edgecolor='red')

# Label the axes
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Normally Distributed Data')

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



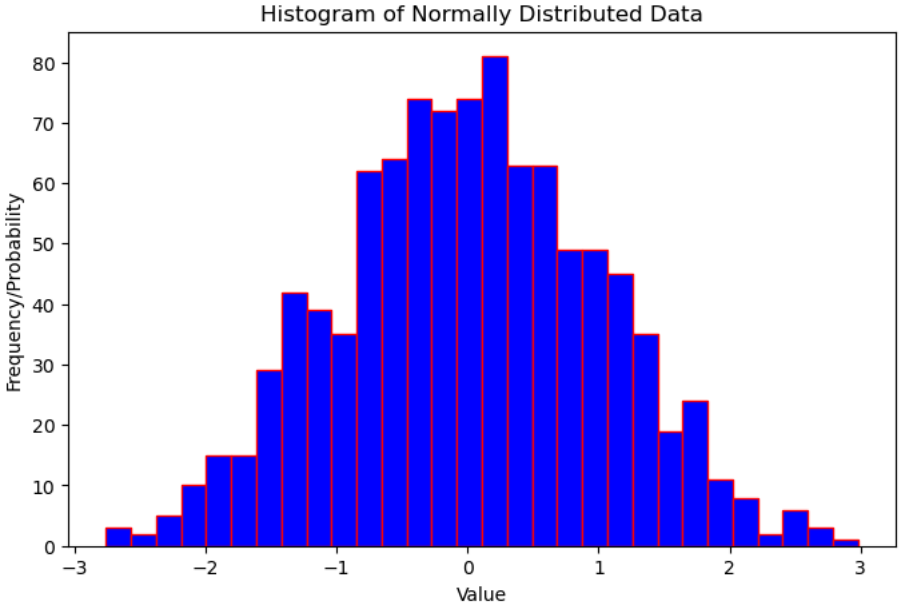
In [19]:
```
# b. Overlay a line plot representing the normal distribution's probability density function (PDF).
import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black', alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')
```

```
# Show legend
plt.legend()

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

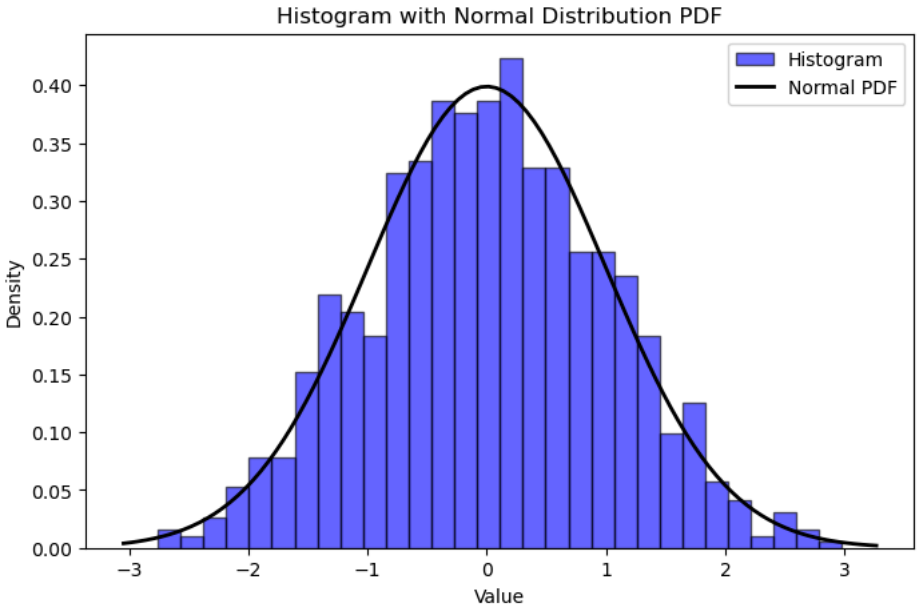


In [18]:

```
# c) Label the x-axis as 'Value' and the y-axis as 'Frequency/Probability'.

import numpy as np

# Generate 1000 samples from a normal distribution
data = np.random.normal(0,1, size=1000)

import matplotlib.pyplot as plt

# Plot the histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, color='blue', edgecolor='red')

# Label the axes
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram of Normally Distributed Data')

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



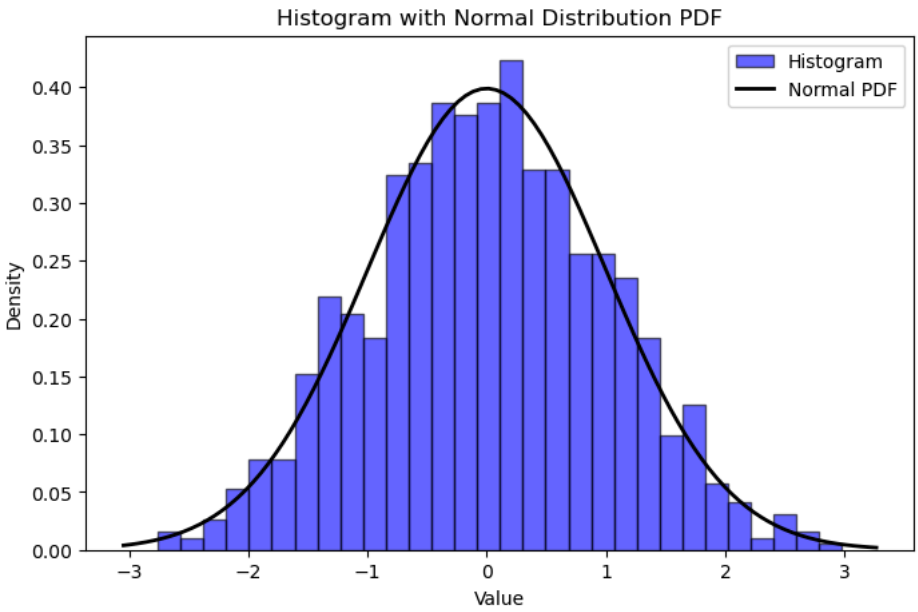
In [20]:
```
# d) Set the title of the plot as 'Histogram with PDF Overlay'.

import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black', alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')

# Show legend
plt.legend()

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



Question_16th:- Set the title of the plot as 'Histogram with PDF Overlay'.

In [21]:
```
import matplotlib.pyplot as plt
from scipy.stats import norm

# Create histogram
plt.figure(figsize=(8, 5))
plt.hist(data, bins=30, density=True, color='blue', edgecolor='black', alpha=0.6, label='Histogram')

# Overlay the PDF
xmin, xmax = plt.xlim()  # Get the limits of the x-axis
x = np.linspace(xmin, xmax, 100)  # Create 100 points between xmin and xmax
p = norm.pdf(x, loc=0, scale=1)  # Calculate the PDF of the normal distribution
plt.plot(x, p, 'k', linewidth=2, label='Normal PDF')  # Plot the PDF

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram with Normal Distribution PDF')
```

```
# Show legend
plt.legend()

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

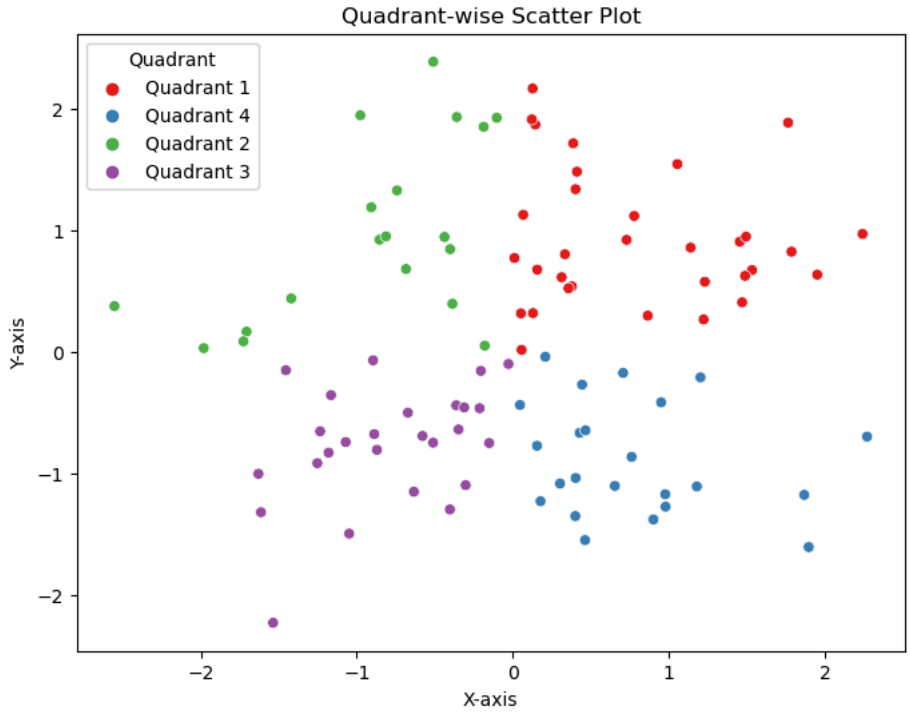


Question_17th:-Create a Seaborn scatter plot of two random arrays, color points based on their position relative to the origin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise

In [22]:
```
import numpy as np
import pandas as pd

# Generate random data for x and y
np.random.seed(0)  # For reproducibility
x = np.random.randn(100)
y = np.random.randn(100)

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

# Define the quadrant based on the position relative to the origin
def determine_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(determine_quadrant, axis=1)
```

In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='x', y='y', hue='Quadrant', palette='Set1', legend='full')

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
```

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



Question_18th:- 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.

In [26]:
```
from bokeh.plotting import figure, show, output_notebook
import numpy as np

# Prepare data
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# Create a Bokeh figure
p = figure(title="Sine Wave Function", x_axis_label='X-axis', y_axis_label='Y-axis')

# Add a line renderer
p.line(x, y, line_width=2, color='blue', legend_label='Sine Wave')

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

# Show the plot in a Jupyter notebook
output_notebook()
show(p)
```

Loading BokehJS ...

Question_19th:-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.

In [27]:
```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd
import numpy as np
```

```
# Generate random categorical data
np.random.seed(0)  # For reproducibility
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(1, 100, size=len(categories))

# Create a DataFrame
df = pd.DataFrame({'Category': categories, 'Value': values})

# Create a ColumnDataSource
source = ColumnDataSource(df)

# Create a Bokeh figure
p = figure(x_range=df['Category'], title="Random Categorical Bar Chart",
           x_axis_label='Category', y_axis_label='Value',
           toolbar_location=None, tools='')

# Add bars with color based on values
p.vbar(x='Category', top='Value', width=0.5, source=source,
       legend_field='Category', color='blue', line_color='white')

# Add hover tooltips
hover = HoverTool()
hover.tooltips = [("Category", "@Category"), ("Value", "@Value")]
p.add_tools(hover)

# Customize grid lines and axis ticks
p.grid.grid_line_color = 'gray'
p.grid.grid_line_alpha = 0.5

# Show the plot in a Jupyter notebook
output_notebook()
show(p)
```

Loading BokehJS ...

Question_20:-Using Plotly, create a basic line plot of a randomly generated dataset, label the axes, and set the title as'Simple Line Plot.

In [28]:

```
import plotly.graph_objects as go
import numpy as np

# Generate random data
np.random.seed(0)  # For reproducibility
x = np.linspace(0, 10, 100)
y = np.random.randn(100)

# Create a line plot
fig = go.Figure()

# Add a line trace
fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Random Data'))

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

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

Question_21th:-Using Plotly, create an interactive pie chart of randomly generated data, add labels and percentages, set the title as 'Interactive Pie Chart'.

In [29]:
```
import plotly.graph_objects as go
import numpy as np

# Generate random data
np.random.seed(0)  # For reproducibility
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(10, 100, size=len(categories))

# Create the pie chart
fig = go.Figure(data=[go.Pie(
    labels=categories,
    values=values,
    textinfo='label+percent',  # Show labels and percentages
    hoverinfo='label+value+percent',  # Show additional info on hover
    hole=0.3  # Create a donut chart (set to 0 for a standard pie chart)
)])

# Update layout with a title
fig.update_layout(title='Interactive Pie Chart')

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

In [ ]: