In [ ]:
'''Demonstrate three different methods for creating identical 2D arrays in NumPy Provide the code for each method and the final output after each method '''
In [1]:
```
# 1. First  Method

# Using np.array() with a list of lists
import numpy as np

# Define a list of lists containing the data
data = [[1, 2, 3], [4, 5, 6]]

# Convert the list of lists to a NumPy array
arr1 = np.array(data)

# Print the array
print("Array using np.array():")
print(arr1)
```
```
Array using np.array():
[[1 2 3]
 [4 5 6]]
```
In [2]:
```
# Second Method
#  Using np.full()
import numpy as np

# Define the shape of the array (number of rows and columns)
shape = (2, 3)

# Specify the fill value for all elements
fill_value = 7

# Create the array using np.full()
arr2 = np.full(shape, fill_value)

# Print the array
print("Array using np.full():")
print(arr2)
```
```
Array using np.full():
[[7 7 7]
 [7 7 7]]
```
In [3]:
```
# Method 3: Using np.ones() with a different data type

import numpy as np

# Define the shape of the array
shape = (2, 3)

# Set the data type to float
data_type = float

# Create the array using np.ones() with float data type
arr3 = np.ones(shape, dtype=data_type)

# Print the array
print("Array using np.ones() with float data type:")
print(arr3)
```
```
Array using np.ones() with float data type:
[[1. 1. 1.]
 [1. 1. 1.]]
```
In [ ]:
2 . Using the Numpy function, generate an array of wRR evenly spaced numPers Petween w **and** wR **and** Reshape that wD array into a 2D array .
In [5]:
```
import numpy as np

# Generate 1D array of 100 evenly spaced numbers between 1 and 10
arr1d = np.linspace(1, 10, 100)

# Reshape 1D array into 2D array with 10 rows and 10 columns
arr2d = arr1d.reshape((10, 10))

print(arr2d)
```

```
[[ 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.        ]]
In [7]:
'''
3.  Explain the following termsg
a. The difference in npYarray, npYasarray and npYasanyarrayX
 b. The difference between Deep copy and shallow copyX'''
```

np.array:

np.array **is** a function that creates a new NumPy array**.**
If the input **is** already an array, it makes a copy of the input array**.**
If the input **is** a sequence (like a list **or** a tuple), it creates a new array **and** fills it **with** the data **from** the sequence**.**
It has an optional parameter dtype to specify the data type of the elements **in** the array

```
import numpy as np

arr1 = np.array([1, 2, 3])  # Creates a new array from a list
arr2 = np.array(arr1)       # Creates a copy of arr1
```

np.asarray:

np.asarray **is** a function that converts input into an array **if** it's not already one.
If the input **is** already an array, it doesn't make a copy.
If the input **is** a sequence, it creates a new array **and** fills it **with** the data **from** the sequence**.**
It has an optional parameter dtype to specify the data type of the elements **in** the array

```
#Example

import numpy as np

arr1 = np.array([1, 2, 3])   # Creates a new array from a list
arr2 = np.asarray(arr1)      # Doesn't copy arr1, just returns arr1
```

np.asanyarray:

np.asanyarray **is** a function that converts input into an array **if** it's not already one, but it prefers not to copy if the input is already an array-li
If the input **is** already an array, it doesn't make a copy.
If the input **is** a sequence, it creates a new array **and** fills it **with** the data **from** the sequence**.**
It has an optional parameter dtype to specify the data type of the elements **in** the array

```
# Example
import numpy as np

arr1 = np.array([1, 2, 3])   # Creates a new array from a list
arr2 = np.asanyarray(arr1)   # Doesn't copy arr1, just returns arr1
```

```
  Cell In[7], line 22
    np.asarray is a function that converts input into an array if it's not already one.
                                                                ^
SyntaxError: unterminated string literal (detected at line 22)
```

In [ ]:

'''
b. The difference between Deep copy and shallow copy .
'''

Shallow Copy:

A shallow copy creates a new object but does **not** create copies of nested objects**.** Instead, it copies references to the nested objects**.** There
Shallow copy can be performed using the copy() method **or** the copy module **in** Python

Deep Copy:

A deep copy creates a new object **and** recursively copies all nested objects **as** well**.** Thus, changes made to the nested objects **in** the origin
Deep copy can be performed using the deepcopy() function **from** the copy module **in** Python**.**

In [8]:
'''
Generate a x array with random floating-point numPers Petween . and 2R 9hen, round each numPer in
the array to 2 decimal places .
'''

```
import numpy as np

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

# Round each number to 2 decimal places
rounded_array = np.round(random_array, 2)

print("Original Array:")
print(random_array)
print("\nRounded Array:")
print(rounded_array)
```

```
Original Array:
[[10.60608661  9.60632521 19.68892612]
 [ 5.37280754 10.2300815   7.26151256]
 [16.17658607 16.84754784 12.36496754]]

Rounded Array:
[[10.61  9.61 19.69]
 [ 5.37 10.23  7.26]
 [16.18 16.85 12.36]]
```

In [9]:
'''
Create a NumPy array with random integers Petween w and wR of shape (.,)) After creating the array
perform the following operations:

 a)Extract all even integers from array.

 b)Extract all odd integers from arrayX
'''

```
import numpy as np

# Create a NumPy array with random integers between 1 and 10 of shape (5, 6)
random_array = np.random.randint(1, 11, size=(5, 6))

print("Original Array:")
print(random_array)

# Extract all even integers from the array
even_numbers = random_array[random_array % 2 == 0]

# Extract all odd integers from the array
odd_numbers = random_array[random_array % 2 != 0]

print("\nEven Integers:")
print(even_numbers)

print("\nOdd Integers:")
print(odd_numbers)
```

```
Original Array:
[[ 9  7  8  5  4  9]
 [ 8  5  3  4  6 10]
 [10  2  1  5  4  3]
 [ 3  9  7  1  3  4]
 [ 7  5  1  1  4  1]]

Even Integers:
[ 8  4  8  4  6 10 10  2  4  4  4]

Odd Integers:
[9 7 5 9 5 3 1 5 3 3 9 7 1 3 7 5 1 1 1]
```

In [10]:

```
'''
Create a D NumPy array of shape (, , ) containing random integers Petween w and wR 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
'''

import numpy as np

# Create a 3D NumPy array of shape (3, 3, 3) containing random integers between 1 and 10
random_array = np.random.randint(1, 11, size=(3, 3, 3))

print("Original 3D Array:")
print(random_array)

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

print("\nIndices of Maximum Values along each Depth Level:")
print(max_indices)

# b) Perform element-wise multiplication of between both array
elementwise_multiplication = random_array * random_array

print("\nElement-wise Multiplication of the Array with Itself:")
print(elementwise_multiplication)
```

```
Original 3D Array:
[[[ 2  2  5]
  [ 8  9  9]
  [ 3  1  8]]

 [[10  9  9]
  [ 3  9  3]
  [ 2  6  9]]

 [[ 9  6  6]
  [ 9  3  4]
  [ 7  3  1]]]

Indices of Maximum Values along each Depth Level:
[[2 1 2]
 [0 1 2]
 [0 0 0]]

Element-wise Multiplication of the Array with Itself:
[[[  4   4  25]
  [ 64  81  81]
  [  9   1  64]]

 [[100  81  81]
  [  9  81   9]
  [  4  36  81]]

 [[ 81  36  36]
  [ 81   9  16]
  [ 49   9   1]]]
```

In [8]:

''' 7 . 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 '''

```
import pandas as pd

# Load the sample dataset (replace 'sample_dataset.csv' with your dataset path)
df = pd.read_csv(r"C:\Users\sudha\Downloads\People Data (1).csv")
print(df)
# Clean and transform the 'Phone' column
df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True)  # Remove non-numeric characters
df['Phone'] = pd.to_numeric(df['Phone'], errors='coerce')  # Convert to numeric, errors='coerce' to handle non-convertible values

# Display table attributes and data types
print("Table Attributes:")
print("Number of rows:", df.shape[0])
print("Number of columns:", df.shape[1])
print("Column Names:", df.columns.tolist())
print("\nData Types of Each Column:")
print(df.dtypes)

# Display the transformed dataset
print("\nTransformed Dataset:")
print(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          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]
Table Attributes:
Number of rows: 1000
Number of columns: 10
Column Names: ['Index', 'User Id', 'First Name', 'Last Name', 'Gender', 'Email', 'Phone', 'Date of birth', 'Job Title', 'Salary']

Data Types of Each Column:
Index             int64
User Id          object
First Name       object
Last Name        object
Gender           object
Email            object
Phone           float64
Date of birth    object
Job Title        object
Salary            int64
dtype: object

Transformed Dataset:
    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  8.571398e+09    27-01-2014
1    fergusonkatherine@example.net           NaN    26-07-1931
2           fhoward@example.org  5.997821e+09    25-11-2013
3          zjohnston@example.com           NaN    17-11-2012
4              elin@example.net  3.904172e+13    15-04-1923
..                          ...           ...           ...
995       lyonsdaisy@example.net  2.177529e+08    05-01-1959
996       dariusbryan@example.com  1.149711e+13    06-10-2001
997       georgechan@example.org  1.750774e+15    13-05-1918
998         wanda04@example.net  9.152922e+09    31-08-1971
999       deannablack@example.org  7.975254e+13    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]
```

In [11]:

```
# 8.  Perform the following tas\s using people dataset:

import pandas as pd

# a) Read the 'data.csv' file using pandas, skipping the first 50 rows.
# b) Only read the columns: 'Last Name', 'Gender', 'Email', 'Phone', and 'Salary' from the file.

read = ['Last Name','Gender', 'Email', 'Phone','Salary']
data = pd.read_csv(r'C:\Users\sudha\Downloads\People Data (1).csv', usecols=read)
print(data)

df = pd.read_csv(r'C:\Users\sudha\Downloads\People Data (1).csv',skiprows=50)
print(df)
# c) Display the first 10 rows of the filtered dataset.
print("First 10 rows of the filtered dataset:")
print(df.head(10))

# d) Extract the 'Salary' column as a Series and display its last 5 values.
salary_series = df['Salary']
print("\nLast 5 values of the 'Salary' column:")
print(salary_series.tail())
```

```
   Last Name  Gender                  Email                Phone  \
0    Mahoney    Male          pwarner@example.org         857.139.8239
1     Rivers  Female  fergusonkatherine@example.net                  NaN
2     Lowery  Female          fhoward@example.org         (599)782-0605
3     Hooper    Male         zjohnston@example.com                  NaN
4       Rice  Female             elin@example.net     (390)417-1635x3010
..       ...     ...                         ...                  ...
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
0    90000
1    80000
2    50000
3    65000
4   100000
..     ...
995  90000
996  50000
```

```
997   60000
998  100000
999   90000

[1000 rows x 5 columns]
      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]
First 10 rows of the filtered dataset:
   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
5  56  998C3Fda97EfAff     Shelby     Cole    Male
6  57  D7040faD2d368d8      Steve  Donovan    Male
7  58  3CEf7FDfACa48b7       Gina   Little  Female
8  59  239dbABfd1d1B1e     Connie   Dawson  Female
9  60  4e03dA0BCAc82e3      Aaron     Page    Male

  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
5     kaneaudrey@example.org             663-280-5834  18-08-1975
6   rebekahsantos@example.net                      NaN  14-04-1935
7        craig28@example.com       125.219.3673x0076  07-10-1954
8  connercourtney@example.net      650-748-3069x64529  21-07-1979
9  harrygallagher@example.com        849.500.6331x717  11-03-1981

         Human resources officer   70000
0                   Nurse, adult   80000
1              Seismic interpreter   70000
2                    Barrister   60000
3             Engineer, structural  100000
4                Producer, radio   50000
5           Therapist, nutritional   85000
6                Paediatric nurse   65000
7              Production engineer   60000
8  Accountant, chartered certified   60000
9                   Administrator   60000
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:3790, in Index.get_loc(self, key)
   3789 try:
-> 3790     return self._engine.get_loc(casted_key)
   3791 except KeyError as err:
```

```
File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Salary'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[11], line 17
     14 print(df.head(10))
     16 # d) Extract the 'Salary' column as a Series and display its last 5 values.
---> 17 salary_series = df['Salary']
     18 print("\nLast 5 values of the 'Salary' column:")
     19 print(salary_series.tail())

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\frame.py:3896, in DataFrame.__getitem__(self, key)
   3894 if self.columns.nlevels > 1:
   3895     return self._getitem_multilevel(key)
-> 3896 indexer = self.columns.get_loc(key)
   3897 if is_integer(indexer):
   3898     indexer = [indexer]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\pandas\core\indexes\base.py:3797, in Index.get_loc(self, key)
   3792     if isinstance(casted_key, slice) or (
   3793         isinstance(casted_key, abc.Iterable)
   3794         and any(isinstance(x, slice) for x in casted_key)
   3795     ):
   3796         raise InvalidIndexError(key)
-> 3797     raise KeyError(key) from err
   3798 except TypeError:
   3799     # If we have a listlike key, _check_indexing_error will raise
   3800     #  InvalidIndexError. Otherwise we fall through and re-raise
   3801     #  the TypeError.
   3802     self._check_indexing_error(key)

KeyError: 'Salary'
```

In [15]:

```
'''
 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 .
'''

import pandas as pd

# Filtering based on conditions
df = pd.read_csv(r"C:\Users\sudha\Downloads\People Data (1).csv")
print(df)
filter_data = df[(df['Last Name'].str.contains('Duke')) &
                 ( df['Gender'] == 'Female') &
                 (df['Salary'] < 85000)]

# Selecting specific columns
selected_columns = ['Last Name', 'Gender', 'Email', 'Phone', 'Salary']
filtered_and_selected_data = filter_data[selected_columns]

# Displaying the filtered and selected data
print(filtered_and_selected_data)
```

```
   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]
   Last Name  Gender                Email                Phone  \
45      Duke  Female      diana26@example.net  001-366-475-8607x04350
210     Duke  Female       robin78@example.com          740.434.0212
457     Duke  Female  perryhoffman@example.org    +1-903-596-0995x489
729     Duke  Female   kevinkramer@example.net          982.692.6257

    Salary
45   60000
210  50000
457  50000
729  70000
In [16]:
'''
 9 . Create a 7*5. Dataframe in Pandas using a series generated from 35. random integers Petween 1 to 6)?
'''

import pandas as pd
import numpy as np

# Generate 35 random integers between 1 and 6
random_integers = np.random.randint(1, 7, size=35)

# Reshape the array into a 7x5 matrix
matrix = random_integers.reshape(7, 5)

# Create a DataFrame from the matrix
df = pd.DataFrame(matrix)

# Display the DataFrame
print(df)
  0 1 2 3 4
0 4 3 2 5 2
1 2 2 4 4 6
2 3 5 1 5 2
3 2 5 4 3 1
4 4 3 6 5 6
5 3 4 2 1 6
6 1 5 3 6 1
In [ ]:
```

In [8]:

```
#Q NO-4
import numpy as np

# Generate random floating-point numbers between 5 and 20
random_array = np.random.uniform(5, 20, size=(3, 3))

# Round each number to 2 decimal places
rounded_array = np.round(random_array, 2)

print(rounded_array)
```

```
[[10.55 19.39 12.34]
 [ 9.79  7.09 15.66]
 [ 7.16  6.22 10.63]]
```

In [9]:

```
#Q.NO.-11
import pandas as pd
import numpy as np

# Creating the first Series with random numbers ranging from 10 to 50
series1 = pd.Series(np.random.randint(10, 51, size=50))

# Creating the second Series with random numbers ranging from 100 to 1000
series2 = pd.Series(np.random.randint(100, 1001, size=50))

# Creating a DataFrame by joining the two Series by column
df = pd.DataFrame({'col1': series1, 'col2': series2})

print(df)
```

	col1	col2
0	23	273
1	48	626
2	33	889
3	46	891
4	50	850
5	29	896
6	48	567
7	33	565
8	44	926
9	21	847
10	27	328
11	24	453
12	34	992
13	21	129
14	43	248
15	23	798
16	40	904
17	50	253
18	23	829
19	13	662
20	23	789
21	48	524
22	26	661
23	23	275
24	14	238
25	12	829
26	22	699
27	28	140
28	10	482
29	38	832
30	40	823
31	40	414
32	16	386
33	36	681
34	10	841
35	45	388
36	42	251
37	41	739
38	32	300
39	44	746
40	12	716
41	24	456
42	15	640
43	11	165
44	14	421
45	23	595
46	46	949
47	43	397
48	11	383
49	29	659

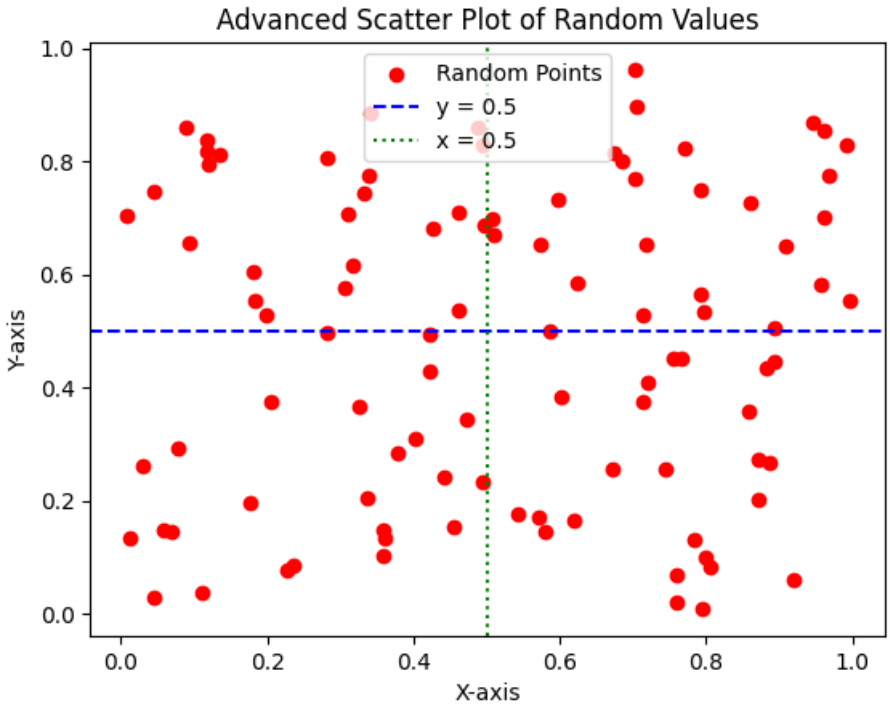
In [11]:

```
#Q NO.-12
import pandas as pd

# Assuming 'people' is your DataFrame containing the dataset
people=pd.read_csv(r"C:\Users\HP\Downloads\People Data.csv")

# a) Delete the 'Email', 'Phone', and 'Date of birth' columns
people.drop(['Email', 'Phone', 'Date of birth'], axis=1, inplace=True)

# b) Delete the rows containing any missing values
people.dropna(inplace=True)

# Printing the final output
print(people)
```

```
     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]
```

In [12]:

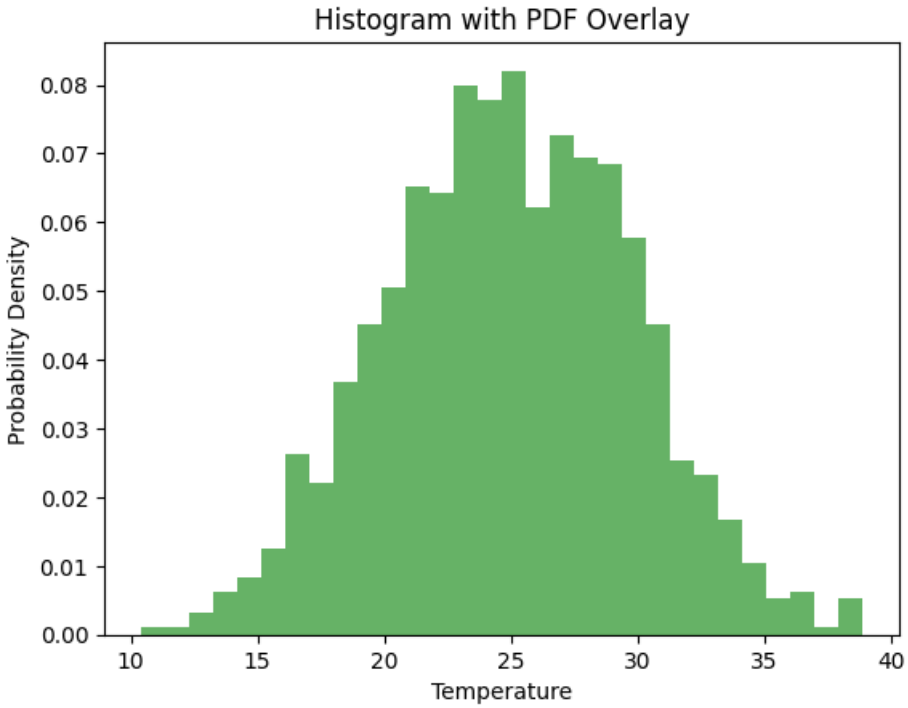
```
#Q NO-13
import numpy as np
import matplotlib.pyplot as plt

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

# Create a scatter plot using x and y
plt.scatter(x, y, color='red', marker='o', label='Random Points')

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

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

# Label the x-axis and y-axis
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values')

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

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


Advanced Scatter Plot of Random Values
Random Points
y = 0.5
x = 0.5
Y-axis
X-axis
0.0
0.2
0.4
0.6
0.8
1.0

In [13]:

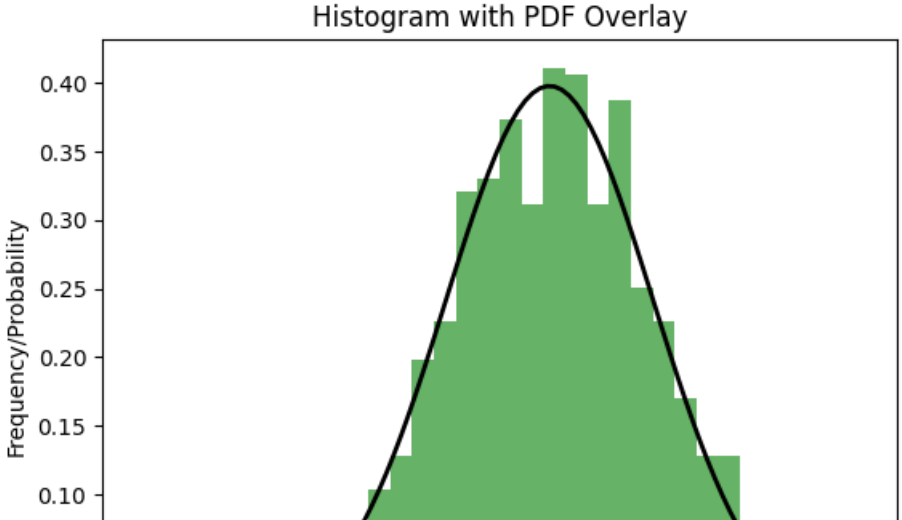
```
#Q NO-14
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Create time-series dataset
dates = pd.date_range(start='2024-01-01', periods=1000)
temperature = np.random.normal(loc=25, scale=5, size=1000)
humidity = np.random.normal(loc=50, scale=10, size=1000)

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

# Plot histogram with PDF overlay
plt.hist(temperature, bins=30, density=True, alpha=0.6, color='g')
plt.title('Histogram with PDF Overlay')
plt.xlabel('Temperature')
plt.ylabel('Probability Density')

# Add PDF overlay
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(temperature), np.std(temperature))
plt.plot(x, p, 'k', linewidth=2)

plt.show()

# Create Seaborn scatter plot
sns.scatterplot(x=np.random.randn(100), y=np.random.randn(100), hue=(np.random.randn(100) > 0), palette={True: 'blue', False: 'red'})
plt.legend(title='Quadrants', labels=['Quadrant I', 'Quadrant II'])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.show()

# Plot Temperature and Humidity over time with different y-axes
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature', color=color)
ax1.plot(df['Date'], df['Temperature'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Humidity', color=color)
ax2.plot(df['Date'], df['Humidity'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

plt.title('Temperature and Humidity Over Time')

fig.tight_layout()
plt.show()
```

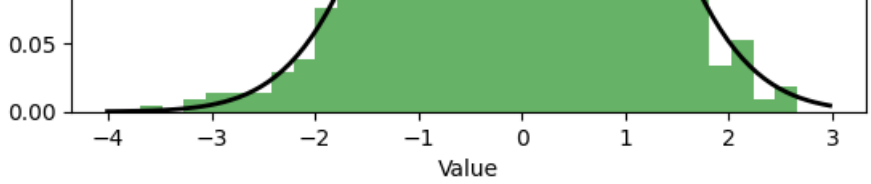
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 22
     20 xmin, xmax = plt.xlim()
     21 x = np.linspace(xmin, xmax, 100)
---> 22 p = norm.pdf(x, np.mean(temperature), np.std(temperature))
     23 plt.plot(x, p, 'k', linewidth=2)
     25 plt.show()

NameError: name 'norm' is not defined
```

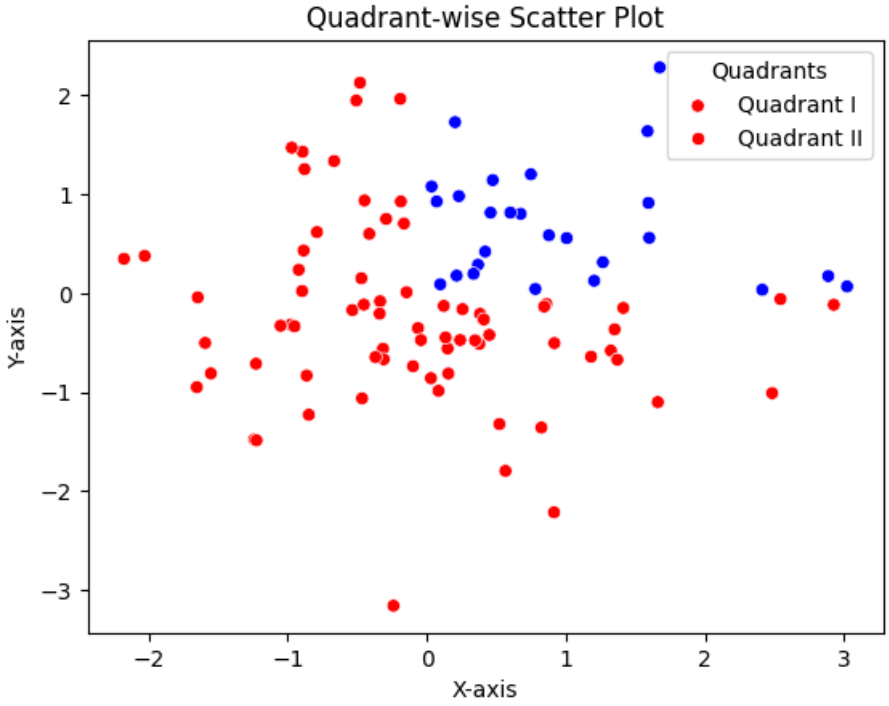


In [14]:

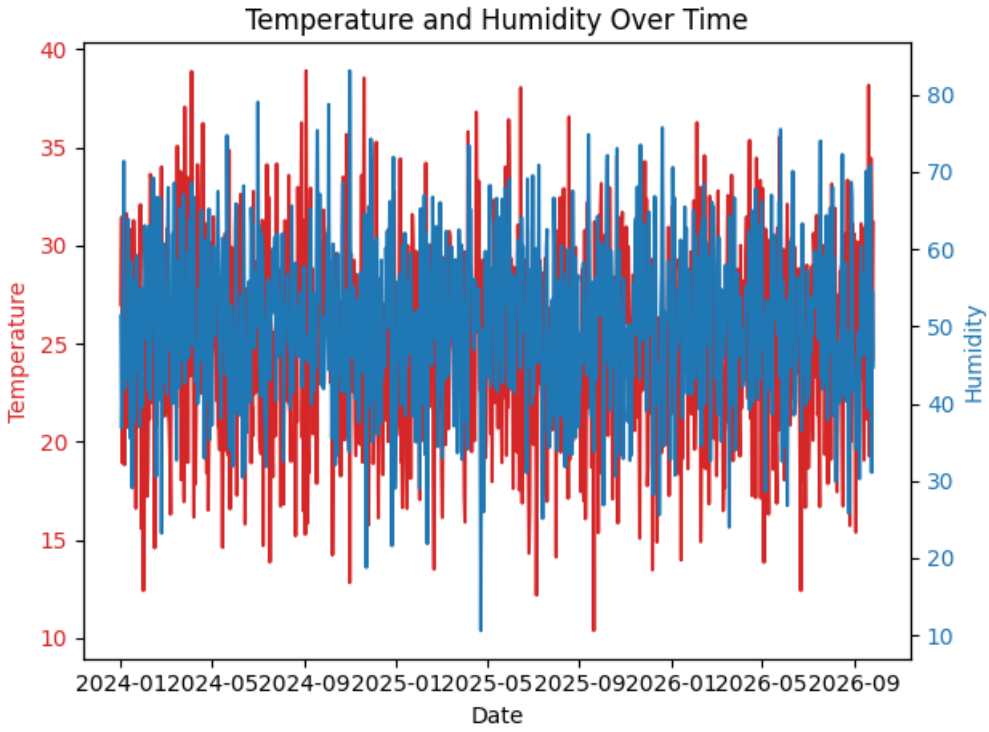
```
#Q NO-15
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

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

# Plot histogram with PDF overlay
plt.hist(data, bins=30, density=True, alpha=0.6, color='g')
plt.title('Histogram with PDF Overlay')
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

# Add PDF overlay
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(data), np.std(data))
plt.plot(x, p, 'k', linewidth=2)

plt.show()

# Create Seaborn scatter plot
x = np.random.randn(100)
y = np.random.randn(100)
hue = (x > 0) & (y > 0)
plt.figure()
sns.scatterplot(x=x, y=y, hue=hue, palette={True: 'blue', False: 'red'})
plt.legend(title='Quadrants', labels=['Quadrant I', 'Quadrant II'])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.show()

# Plot Temperature and Humidity over time with different y-axes
# Assuming 'df' is your DataFrame containing 'Temperature' and 'Humidity' columns
plt.figure()
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature', color=color)
ax1.plot(df['Date'], df['Temperature'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Humidity', color=color)
ax2.plot(df['Date'], df['Humidity'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

plt.title('Temperature and Humidity Over Time')

fig.tight_layout()
plt.show()
```





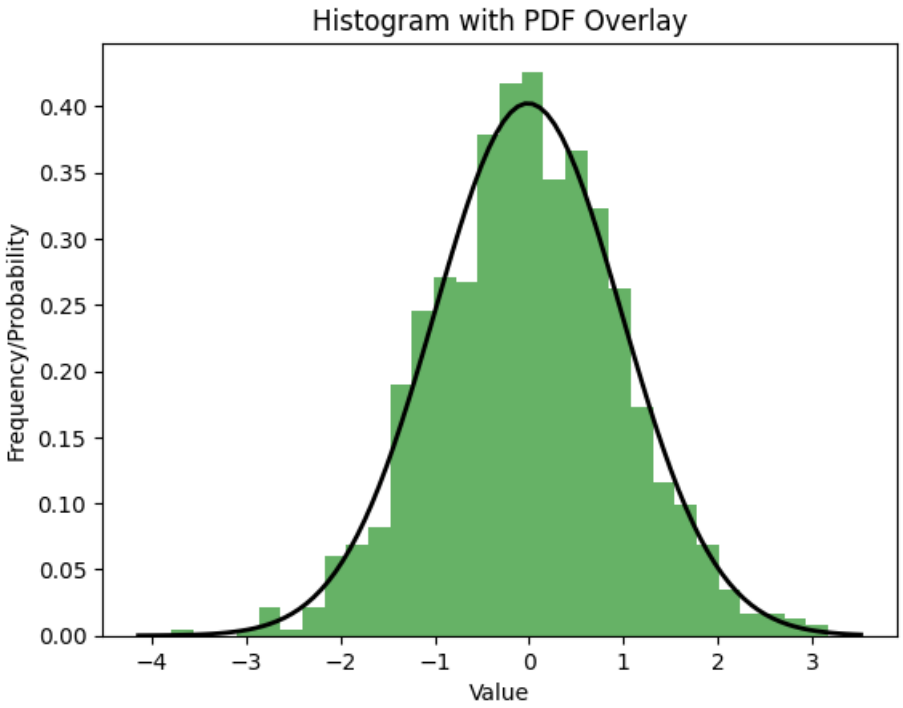


<Figure size 640x480 with 0 Axes>



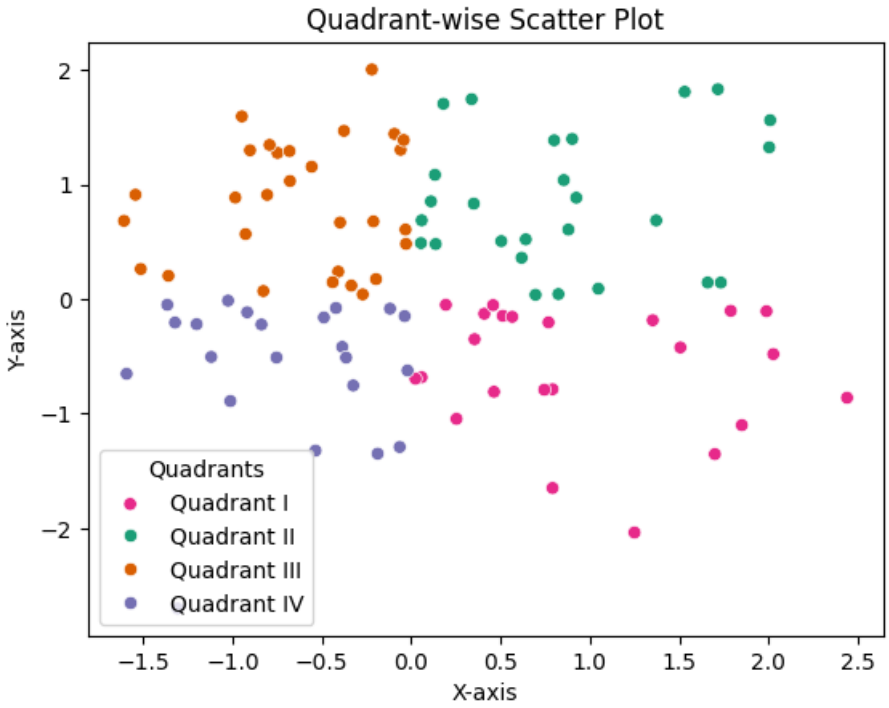
In [15]:

```
#Q NO-16
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

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

# Plot histogram with PDF overlay
plt.hist(data, bins=30, density=True, alpha=0.6, color='g')
plt.title('Histogram with PDF Overlay')
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')

# Add PDF overlay
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(data), np.std(data))
plt.plot(x, p, 'k', linewidth=2)

plt.show()
```



In [16]:

```
#Q NO-17
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

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

# Determine the quadrant for each point
quadrant = np.zeros_like(x, dtype=int)
quadrant[(x > 0) & (y > 0)] = 1  # Quadrant I
quadrant[(x < 0) & (y > 0)] = 2  # Quadrant II
quadrant[(x < 0) & (y < 0)] = 3  # Quadrant III
quadrant[(x > 0) & (y < 0)] = 4  # Quadrant IV

# Create a Seaborn scatter plot
sns.scatterplot(x=x, y=y, hue=quadrant, palette='Dark2')

# Add legend
plt.legend(title='Quadrants', labels=['Quadrant I', 'Quadrant II', 'Quadrant III', 'Quadrant IV'])

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

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

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



In [20]:

```
#Q NO-18

from bokeh.plotting import figure, show
import numpy as np

# Generate data for the sine wave
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

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

# Plot the sine wave
p.line(x, y, line_width=2)

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

# Show the plot
show(p)
```

In [27]:

```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (5.22.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (from plotly) (8.3.0)
Requirement already satisfied: packaging in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (from plotly) (24.0)
```

In [21]:

```
#Q NO-19
#from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool
import numpy as np

# Generate random categorical data and their corresponding values
categories = ['Category A', 'Category B', 'Category C', 'Category D']
values = np.random.randint(1, 10, size=len(categories))

# Create a ColumnDataSource
source = ColumnDataSource(data=dict(categories=categories, values=values, color=values))

# Create a Bokeh figure
p = figure(x_range=categories, plot_height=350, title='Random Categorical Bar Chart',
           toolbar_location=None, tools='')

# Plot the bars
p.vbar(x='categories', top='values', width=0.9, color='color', legend_field='categories', source=source)

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

# Label the axes
p.xaxis.axis_label = 'Categories'
p.yaxis.axis_label = 'Values'

# Show the plot
show(p)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[21], line 13
     10 source = ColumnDataSource(data=dict(categories=categories, values=values, color=values))
     12 # Create a Bokeh figure
---> 13 p = figure(x_range=categories, plot_height=350, title='Random Categorical Bar Chart',
     14            toolbar_location=None, tools='')
     16 # Plot the bars
     17 p.vbar(x='categories', top='values', width=0.9, color='color', legend_field='categories', source=source)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\bokeh\plotting\_figure.py:196, in figure.__init__(self, *arg, **kw)
    194 for name in kw.keys():
    195     if name not in names:
--> 196         self._raise_attribute_error_with_matches(name, names | opts.properties())
    198 super().__init__(*arg, **kw)
    200 self.x_range = get_range(opts.x_range)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\bokeh\core\has_props.py:379, in HasProps._raise_attribute_error_with_matches(self, name, properties)
    376 if not matches:
    377     matches, text = sorted(properties), "possible"
--> 379 raise AttributeError(f"unexpected attribute {name!r} to {self.__class__.__name__}, {text} attributes are {nice_join(matches)}")

AttributeError: unexpected attribute 'plot_height' to figure, similar attributes are outer_height, height or min_height
```

In [29]:

```
#Q NO-20
import plotly.graph_objs as go
import numpy as np

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

# Create a trace for the line plot
trace = go.Scatter(x=x, y=y, mode='lines', name='Random Data')

# Create layout
layout = go.Layout(title='Simple Line Plot', xaxis=dict(title='X-axis'), yaxis=dict(title='Y-axis'))

# Create figure
fig = go.Figure(data=[trace], layout=layout)

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

In [30]:

```
#Q NO-21
import plotly.graph_objs as go
import numpy as np

# Generate random data
labels = ['Category A', 'Category B', 'Category C', 'Category D']
values = np.random.randint(1, 10, size=len(labels))

# Create trace for the pie chart
trace = go.Pie(labels=labels, values=values, hoverinfo='label+percent', textinfo='value+percent', textfont_size=20)

# Create layout
layout = go.Layout(title='Interactive Pie Chart')

# Create figure
fig = go.Figure(data=[trace], layout=layout)

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

In [ ]: