Data Analysis & Visualisation

Name	Avinash Gautam
Course	B.Sc(H) Computer Science
College Roll No.	20201407
Exam Roll No.	20020570009

Practicals

Q. 1)
Given below is a dictionary having two keys ‘Boys’ and ‘Girls’ and having two lists of heights of five Boys and five Girls respectively as values associated with these keys.

Original dictionary of lists:
{'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

From the given dictionary of lists create the following list of dictionaries:
[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {‘Boys’: 74, ‘Girls’: 61]

Code)

In [1]:

```
d1 = {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}
n = d1['Boys'].__len__()

list_d = [{k: a[i] for k, a in d1.items()}
          for i in range(n)]

print(f'''
\t\t\t Que. 1 Output \n
Original Dictionary    :
{d1}\n
Derived List of dicts :
{list_d}
''')
```

```
			Que. 1 Output

Original Dictionary   :
{'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

Derived List of dicts :
[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69},
{'Boys': 69, 'Girls': 62}, {'Boys': 74, 'Girls': 61}]
```

Q. 2)
Write programs in Python using NumPy library to do the following:

a) Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.

b) Get the indices of the sorted elements of a given array.
B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

c) Create a 2-dimensional array of size m x n integer elements, also print the shape, type and data type of the array and then reshape it into n x m array, n and m are user inputs given at the run time.

d) Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.

Code)

In [1]:

```python
import numpy as np

# a. Computing mean, sd, var along with axis 2
arr = np.random.randint(100, size=(2, 2))
print('\t\t\tQue 2 Output \n')

print(f'''--------------------------------------------------------------------
\nA.\n\nRandom 2D array of {arr.shape} diameter:\n{arr} \n
Array Stats regard axis 2 :
Mean \t\t\t: {arr.mean(1)}
Standard Deviation \t: {np.sqrt(arr.var(1))}
Variance \t\t: {arr.var(1)}
''')
```

```
                        Que 2 Output

--------------------------------------------------------------------

A.

Random 2D array of (2, 2) diameter:
[[ 7 78]
 [36 56]]

Array Stats regard axis 2 :
Mean                    : [42.5 46. ]
Standard Deviation      : [35.5 10. ]
Variance                : [1260.25  100.  ]
```

In [2]:

```
# b. Sorting array's indices
arr = np.array([56, 48, 22, 41, 78, 91, 24, 46, 8, 33])
index = arr.argsort()
print(f'''--------------------------------------------------------------------
\nB.\n\nGiven Numpy Integer Array    : {arr}
Indices of sorted elements    : {index}
Access Array using indices    : {arr[index[0::]]}
''')
```

```
--------------------------------------------------------------------

B.

Given Numpy Integer Array     : [56 48 22 41 78 91 24 46  8 33]
Indices of sorted elements    : [8 2 6 9 3 7 1 0 4 5]
Access Array using indices    : [ 8 22 24 33 41 46 48 56 78 91]
```

In [3]:

```
# c. Simple Matrix Simulation
print('--------------------------------------------------------------------')
print('\nC.\n\nEnter the parameters of matrix :')
m, n = [int(x) for x in input("rows & columns (use space) : ").split()]
arr = np.random.randint(10*m*n, size=(m, n))
print(f'''
Created Array : \n{arr}\n
Array Details :
    Shape     : {arr.shape}
    Data Type : {arr.dtype}
    Obj Type  : {type(arr)} \n
Reshaped Array into {n} x {m}: \n{arr.reshape(n,m)}
''')
```

```
--------------------------------------------------------------------

C.

Enter the parameters of matrix :

Created Array :
[[ 58 102  82 118]
 [ 64  74 114  98]
 [ 22  92  66  87]]

Array Details :
    Shape     : (3, 4)
    Data Type : int32
    Obj Type  : <class 'numpy.ndarray'>

Reshaped Array into 4 x 3:
[[ 58 102  82]
 [118  64  74]
 [114  98  22]
 [ 92  66  87]]
```

In [4]:

```python
# d. Checking that elements are zero, non-zero or null
def cmp_arr(arr: np.ndarray, cmp):
    return np.array([i for i in range(len(arr)) if cmp(arr[i])])


x = np.array([2, np.NaN, 0, 4, np.NaN, 5, 0, -7, np.NaN])

zero = cmp_arr(x, cmp=lambda a: a == 0)
nzero = cmp_arr(x, cmp=lambda a: a > 0 or a < 0)
nan = cmp_arr(x, cmp=lambda a: np.isnan(a))

print(f'''--------------------------------------------------------------------
\nD.\n\nGiven Array (x) : {x}
\nIndices of array x that are equal to :
Zero     : {zero}
Non-Zero : {nzero}
NaN      : {nan}
--------------------------------------------------------------------''')
```

```
--------------------------------------------------------------------

D.

Given Array (x) : [ 2. nan  0.  4. nan  5.  0. -7. nan]

Indices of array x that are equal to :
Zero     : [2 6]
Non-Zero : [0 3 5 7]
NaN      : [1 4 8]
--------------------------------------------------------------------
```

Q. 3)

Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random function. Replace 10% of the values by null values whose index positions are generated using random function.

Do the following:

---

a. Identify and count missing values in a dataframe.

b. Drop the column having more than 5 null values.

c. Identify the row label having maximum of the sum of all values in a row and drop that row.

d. Sort the dataframe on the basis of the first column.

e. Remove all duplicates from the first column.

f. Find the correlation between first and second column and covariance between second and third column.

g. Detect the outliers and remove the rows having outliers.

h. Discretize second column and create 5 bins.

Code)

In [4]:

```
import pandas as pd
from numpy import random

nrows = 50
# creating a dataframe
df = pd.DataFrame({'Age': random.randint(10, 90, nrows),
                   'Height': random.randint(150, 200, nrows),
                   'Weight': random.randint(50, 200, nrows),
                   })

# replacing 10% random values to null
ncols = len(df.columns)
while df.isnull().sum().sum() != (ncols * nrows // 10):
    df.iloc[random.randint(nrows), random.randint(ncols)] = None

print(f'''\t\t\tQ.3 Output \n
--------------------------------------------------------------------
\nGiven DataFrame's head  : \n
{df.head()}\n\nDetails : \n''')
df.info()
```

```
                         Q.3 Output

--------------------------------------------------------------------

Given DataFrame's head  :

    Age  Height  Weight
0  25.0   168.0   121.0
1  53.0   177.0   112.0
2  88.0   179.0    83.0
3   NaN   167.0     NaN
4  83.0   159.0    76.0

Details :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Age     45 non-null     float64
 1   Height  46 non-null     float64
 2   Weight  44 non-null     float64
dtypes: float64(3)
memory usage: 1.3 KB
```

In [5]:

```python
# A. Identifying & counting null values

var1 = df.isnull().sum()

print(f'''\n--------------------------------------------------------------------
\nA.\n\nTotal null values in Given DataFrame : {sum(var1)}\n
{var1}
''')
```

```
--------------------------------------------------------------------

A.

Total null values in Given DataFrame : 15

Age       5
Height    4
Weight    6
dtype: int64
```

In [12]:

```python
# B. Dropping cols with more than 5 null

var1 = [i for i in df if var1[i] > 5]

print(f'''--------------------------------------------------------------------
\nB.\n\nColumns with more than 5 null values : {var1}\n
DataFrame after dropping columns : \n
{df.drop(columns=var1).head()}
''')
# df.dropna(thresh = len(df) - 5, axis = 1)
```

```
--------------------------------------------------------------------

B.

Columns with more than 5 null values : ['Height']

DataFrame after dropping columns :

    Age  Weight
0  55.0   127.0
1  58.0     NaN
2  86.0   141.0
3  69.0   188.0
4  49.0   138.0
```

In [17]:

```python
# C. Dropping row with max row_sum value

var1 = df.sum(axis=1).idxmax()

print(f'''--------------------------------------------------------------------
\nC.\n\nRow with max sum value : {var1}\n
DataFrame after dropping row {var1} : \n
{df.drop(index=var1)[(var1 - 2) : (var1 + 2)]}
''')
```

```
--------------------------------------------------------------------

C.

Row with max sum value : 37

DataFrame after dropping row 37 :

     Age  Height  Weight
44  68.0   161.0   173.0
14  69.0   156.0    84.0
3   69.0     NaN   188.0
24  70.0   164.0   151.0
```

In [15]:

```python
# D. Sorting DataFrame according to 1st col

df = df.sort_values(by=df.columns[0])

print(f'''--------------------------------------------------------------------
\nD.\n\nSorted dataFarme on the basis of first column : \n
{df.head()}\n\n{df.shape}
''')
```

```
--------------------------------------------------------------------

D.

Sorted dataFarme on the basis of first column :

     Age  Height  Weight
47  10.0   191.0    79.0
10  10.0   177.0   153.0
22  11.0     NaN   168.0
35  14.0   157.0     NaN
15  15.0     NaN   106.0

(50, 3)
```

In [19]:

```
# E. Removing duplicates from the 1st col

df.drop_duplicates(df.columns[0], inplace=True)

print(f'''--------------------------------------------------------------------
\nE.\n\nDataFarme after removing duplicates from the first column :
\n {df.head()}
\n{df.shape}''')
```

```
--------------------------------------------------------------------

E.

DataFarme after removing duplicates from the first column :

      Age  Height  Weight
47  10.0   191.0    79.0
22  11.0     NaN   168.0
35  14.0   157.0     NaN
15  15.0     NaN   106.0
8   16.0   159.0   170.0

(41, 3)
```

In [79]:

```
# F. Calculating correlation & covariance

var1 = df.columns

print(f'''--------------------------------------------------------------------
\nF.\n\nCorrelation between first & second column : {df[var1[0]].corr(df[var1[1]])
\nCovariance between second & third column  : {df[var1[1]].cov(df[var1[2]])}
''')
```

```
--------------------------------------------------------------------

F.

Correlation between first & second column : -0.2497633957546939

Covariance between second & third column  : 51.89818548387097
```

In [22]:

```
# G. Detect & remove the row having outliers

q = df.quantile(q=[0.25, 0.75])
q.loc['IQR'] = q.iloc[1] - q.iloc[0]
```

```
q.loc['low'] = q.iloc[0] - 1.5 * q.iloc[2]
q.loc['high'] = q.iloc[1] + 1.5 * q.iloc[2]

df = df[~((df < (q.loc['low'])) | (df > (q.loc['high']))).any(axis=1)]

print(f'''---------------------------------------------------------------------
\nG.\n\nInter-Quartile Parameters for the outliers : \n\n{q}
\nDataFrame after removing outliers : \n\n{df.head()}\n\n{df.shape}
''')
```

```
--------------------------------------------------------------------

G.

Inter-Quartile Parameters for the outliers :

         Age  Height  Weight
0.25   28.75   158.0   82.75
0.75   67.25   184.0  169.25
IQR    38.50    26.0   86.50
low   -29.00   119.0  -47.00
high  125.00   223.0  299.00

DataFrame after removing outliers :

     Age  Height  Weight
47  10.0   191.0    79.0
22  11.0     NaN   168.0
35  14.0   157.0     NaN
15  15.0     NaN   106.0
8   16.0   159.0   170.0

(41, 3)
```

In [87]:

```
# H. Discretizing second column & creating 5 bins

df['binned'] = pd.qcut(df[df.columns[1]], q=5)
var1 = df.binned.unique()
print(f'''--------------------------------------------------------------------------
\nH.\n
Bins created for second column : \n
{var1.categories.values}

\nDataFrame after discretizing & creating 5 bins for second column : \n
{df.head()}\n\n{df.shape}
-----------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------
------------

H.

Bins created for second column :

<IntervalArray>
[(149.999, 157.0], (157.0, 161.0], (161.0, 177.4], (177.4, 185.8], (185.8, 199.0]]
Length: 5, dtype: interval[float64, right]


DataFrame after discretizing & creating 5 bins for second column :

     Age  Height  Weight            binned
47  10.0   191.0    79.0    (185.8, 199.0]
22  11.0     NaN   168.0               NaN
35  14.0   157.0     NaN  (149.999, 157.0]
15  15.0     NaN   106.0               NaN
8   16.0   159.0   170.0    (157.0, 161.0]

(41, 4)
----------------------------------------------------------------------------------
------------
```

Q.4.)
Consider two excel files having attendance of a workshop's participants for two days. Each file has three fields 'Name', 'Time of joining', duration (in minutes) where names are unique within a file.
Note that duration may take one of three values (30, 40, 50) only.

Import the data into two dataframes and do the following:

---

a. Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.

b. Find names of all students who have attended workshop on either of the days.

c. Merge two data frames row-wise and find the total number of records in the data frame.

d. Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.

Code)

In [4]:

```
import pandas as pd

df1 = pd.read_excel('Day1.xlsx')
df2 = pd.read_excel('Day2.xlsx')

print(f'''
\t\t\t Q.4 Output
--------------------------------------------------------------------

Day1 excel file : \n
{df1.head()}
\n{df1.shape}

--------------------------------------------------------------------

Day2 excel file : \n
{df2.head()}
\n{df2.shape}
''')
```

```
                         Q.4 Output
--------------------------------------------------------------------

Day1 excel file :

        Name Time of Joining  Duration
0  Abhimanyu        11:00:00        40
1   Abhishek        11:04:00        30
2      Aasif        11:08:00        30
3       Aman        11:01:00        40
4      Anand        11:12:00        50

(15, 3)

--------------------------------------------------------------------

Day2 excel file :

        Name Time of Joining  Duration
0  Abhimanyu        11:00:00        40
1   Abhishek        11:06:00        30
2  Deepanshu        11:10:00        40
3       Aman        11:09:00        40
4    Anubhav        11:10:00        50

(15, 3)
```

In [5]:

```
# A. Merge two dataframes & find the names of students
#           who had attended the workshop on both days

mdf = pd.merge(df1, df2, how = 'inner', on = 'Name')
print(f'''
----------------------------------------------------------------------------
\nA.\n
Merged DataFrame :\n
{mdf.head()} \n
{mdf.shape}

Name of the students who attended workshop on both days :\n
{mdf.Name}
''')
```

```
----------------------------------------------------------------------------

A.

Merged DataFrame :

        Name Time of Joining_x  Duration_x Time of Joining_y  Duration_y
0  Abhimanyu          11:00:00          40          11:00:00          40
1   Abhishek          11:04:00          30          11:06:00          30
2       Aman          11:01:00          40          11:09:00          40
3    Anubhav          11:10:00          30          11:10:00          50
4     Anurag          11:11:00          30          11:08:00          30

(10, 5)

Name of the students who attended workshop on both days :

0    Abhimanyu
1     Abhishek
2         Aman
3      Anubhav
4       Anurag
5        Arpit
6      Bhavana
7    Deepanshu
8       Ishant
9      Harshit
Name: Name, dtype: object
```

In [6]:

```python
# B. Find the names of students who had attended the workshop on either of days

mdf1 = pd.merge(df1, df2, how = 'outer', on = 'Name')
print(f'''
----------------------------------------------------------------------------
\nB.\n
Merged DataFrame :\n
{mdf1.head()} \n
{mdf1.shape}

Name of the students who attended workshop on either of days :\n
{mdf1.Name}
''')
```

```
----------------------------------------------------------------------------

B.

Merged DataFrame :

        Name Time of Joining_x  Duration_x Time of Joining_y  Duration_y
0  Abhimanyu          11:00:00        40.0          11:00:00        40.0
1   Abhishek          11:04:00        30.0          11:06:00        30.0
2      Aasif          11:08:00        30.0               NaN         NaN
3       Aman          11:01:00        40.0          11:09:00        40.0
4      Anand          11:12:00        50.0               NaN         NaN

(20, 5)

Name of the students who attended workshop on either of days :

0     Abhimanyu
1      Abhishek
2         Aasif
3          Aman
4         Anand
5       Anubhav
6        Anurag
7         Arpit
8      Akanksha
9       Bhavana
10    Deepanshu
11       Ishant
12       Gourav
13      Harshit
14     Kartikey
15       Bharat
16    Divyanshu
17       Deepak
18       Jayesh
19        Jeeva
Name: Name, dtype: object
```

In [7]:

```python
# C. Merge the DataFrame row-wise & find the total no. of records

print(f'''
----------------------------------------------------------------------------
\nC.\n
Row-wise Merged DataFrame :\n
{mdf1.head()} \n
{mdf1.shape}

Total no. of records : {len(mdf1)}
''')
```

```
----------------------------------------------------------------------------

C.

Row-wise Merged DataFrame :

        Name Time of Joining_x  Duration_x Time of Joining_y  Duration_y
0  Abhimanyu          11:00:00        40.0          11:00:00        40.0
1   Abhishek          11:04:00        30.0          11:06:00        30.0
2      Aasif          11:08:00        30.0               NaN         NaN
3       Aman          11:01:00        40.0          11:09:00        40.0
4      Anand          11:12:00        50.0               NaN         NaN

(20, 5)

Total no. of records : 20
```

In [10]:

```
# D. Merge the DataFrame with two columns Name & Duration as indices
#       Generate Descriptive Statistics

mdf2 = pd.merge(df1, df2, how = 'outer', on = ['Name', 'Duration'])
mdf2.set_index(['Name', 'Duration'], inplace = True)

cols = mdf2.columns
mdf2[cols[0]] = pd.to_datetime(mdf2[cols[0]], format='%H:%M:%S').dt.time
mdf2[cols[1]] = pd.to_datetime(mdf2[cols[1]], format='%H:%M:%S').dt.time
desc = mdf2.describe()

t1 = mdf2[cols[0]]
t2 = mdf2[cols[1]]
t1.dropna(inplace = True)
t2.dropna(inplace = True)

desc.loc['min'] = [t1.min(), t2.min()]
desc.loc['max'] = [t1.max(), t2.max()]

print(f'''
----------------------------------------------------------------------------
\nD.\n
Merged DataFrame :\n
{mdf2.head()} \n
{mdf2.shape}

Descriptive Statistics :\n\n {desc}
----------------------------------------------------------------------------''')
```

```
----------------------------------------------------------------------------

D.

Merged DataFrame :

                   Time of Joining_x Time of Joining_y
Name      Duration
Abhimanyu 40                11:00:00          11:00:00
Abhishek  30                11:04:00          11:06:00
Aasif     30                11:08:00               NaT
Aman      40                11:01:00          11:09:00
Anand     50                11:12:00               NaT

(21, 2)

Descriptive Statistics :

        Time of Joining_x Time of Joining_y
count                  15                15
unique                 14                 9
top              11:08:00          11:08:00
freq                    2                 3
min              11:00:00          11:00:00
max              11:19:00          11:14:00
----------------------------------------------------------------------------
```

Q.5)
Taking Iris data, plot the following with proper legend and axis labels:
(Download IRIS data from: https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn.datasets)

a. Plot bar chart to show the frequency of each class label in the data.

b. Draw a scatter plot for Petal width vs sepal width.

c. Plot density distribution for feature petal length.

d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.

Code)

In [39]:

```python
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# iris dataset
df = datasets.load_iris()
iris = pd.DataFrame(data = df.data, columns = df.feature_names)
t_names = {0:df.target_names[0], 1: df.target_names[1], 2: df.target_names[2]}
iris['type'] = df.target
iris['type'] = iris['type'].map(t_names)
color = ['r','g','b','y']
```

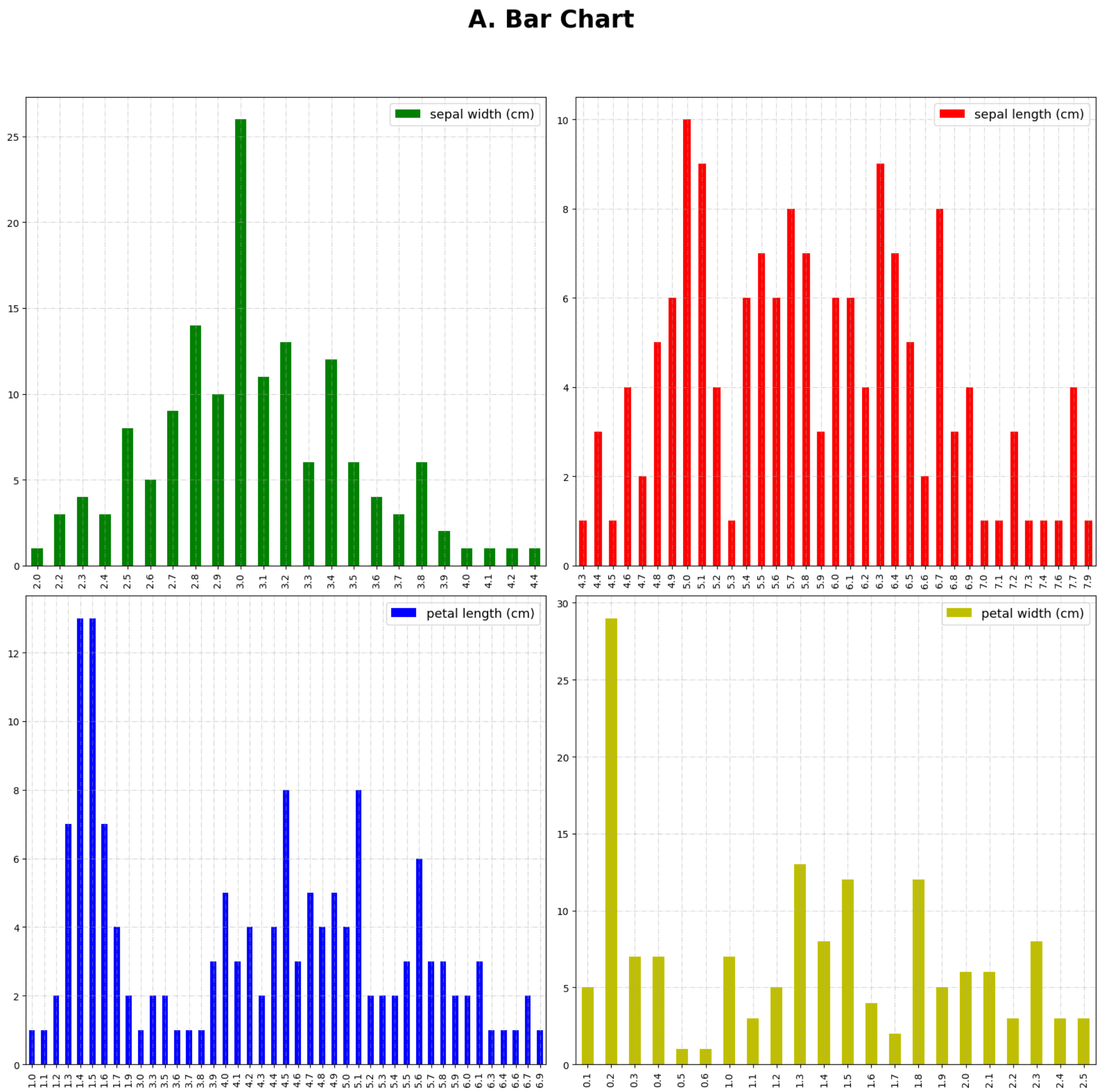
```
print(f'''
\t\t\t\t Q.5 Output
----------------------------------------------------------------------------------
\nIris Dataset : \n
{iris.head()} \n
{iris.shape}

\nDetails :
''')
iris.info()
```

```
                                 Q.5 Output
----------------------------------------------------------------------------------
-

Iris Dataset :

   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

     type
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa

(150, 5)


Details :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   type               150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

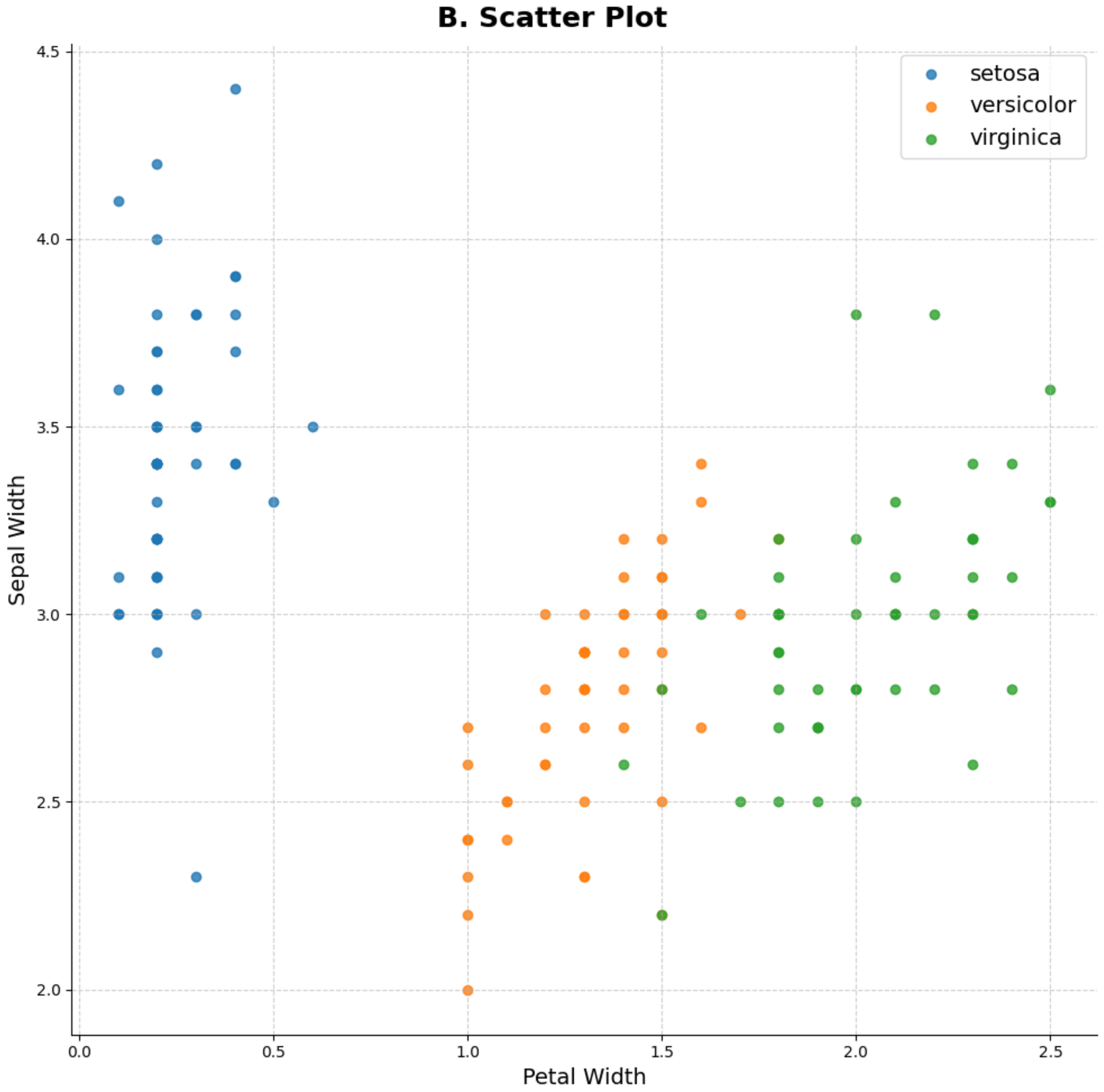
In [13]:

```
# A. Bar Chart

fig, ax = plt.subplots(2,2)
fig.set_figwidth(16)
fig.set_figheight(16)

for i in range(len(ax)):
    for j in range(len(ax[0])):
        iris.iloc[:,i+1^j].value_counts().sort_index().plot(
                                    kind = 'bar', ax = ax[i, j], color = color[i+1^j])
        ax[i,j].legend(fontsize = 13)
        ax[i,j].grid(alpha = 0.5, linestyle = '-.')
```

```
fig.suptitle('A. Bar Chart', fontweight = 'bold', fontsize = 25)
fig.tight_layout()
fig.subplots_adjust(top = 0.9)
```



In [42]:
```
# B. Scatter Plot

sc = sns.lmplot(x = 'petal width (cm)', y = 'sepal width (cm)' , data=iris, fit_reg
sc.fig.suptitle('B. Scatter Plot', fontsize = 18, fontweight = 'bold')
sc.ax.legend(loc = 'upper right',fontsize = 14)
sc.ax.set_xlabel('Petal Width',fontsize = 14)
sc.ax.set_ylabel('Sepal Width',fontsize = 14)
sc.ax.grid(linestyle = '--', alpha = 0.6)
sc.tight_layout()
plt.show()
```

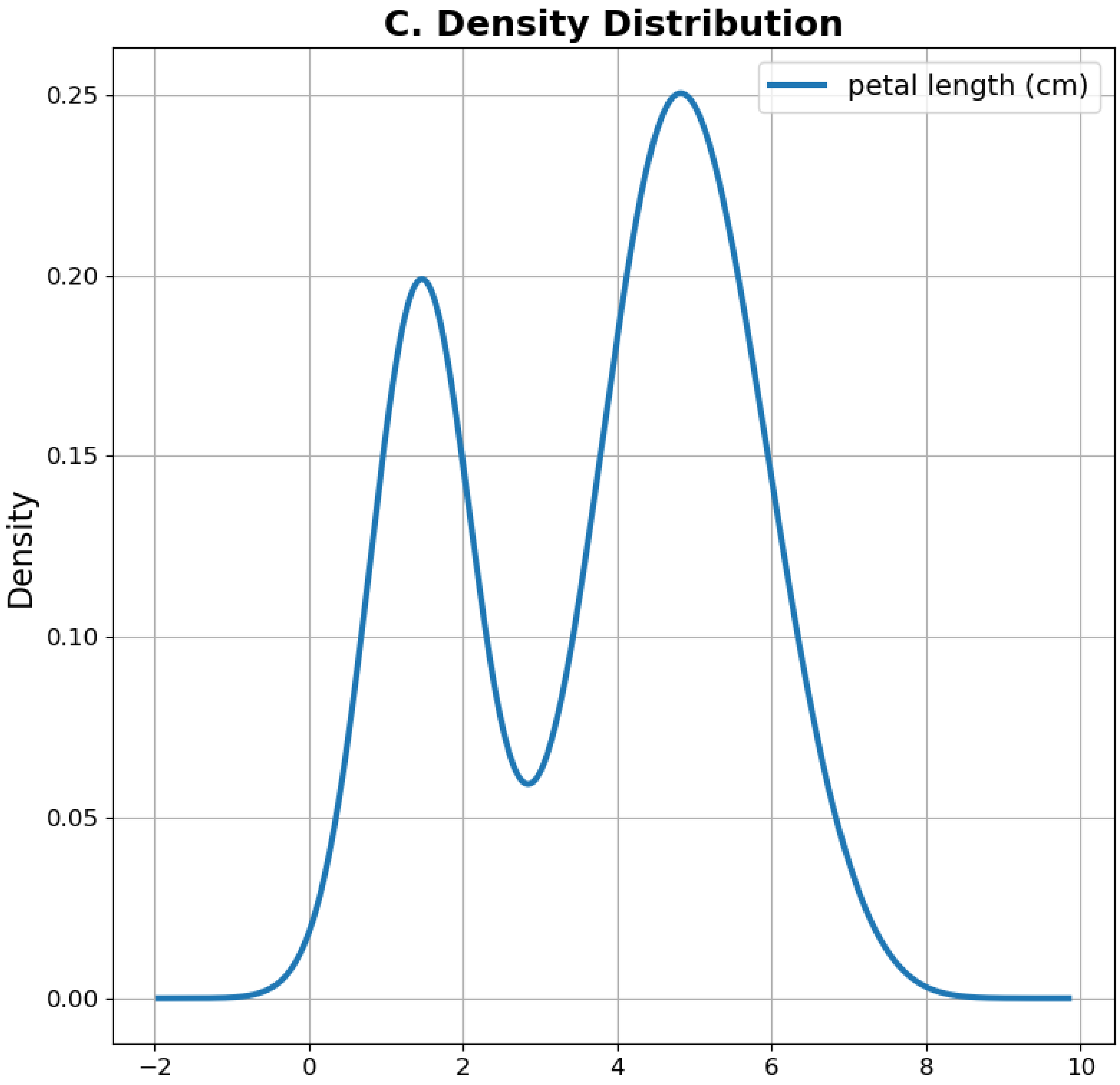


In [119…

```
# C. Density Plot
den = iris['petal length (cm)'].plot(kind = 'density', figsize = (9,9), linewidth =
den.legend(fontsize = 14)
den.set_ylabel('Density', fontsize = 16)
den.set_title('C. Density Distribution', fontsize = 18, fontweight = 'bold')
```

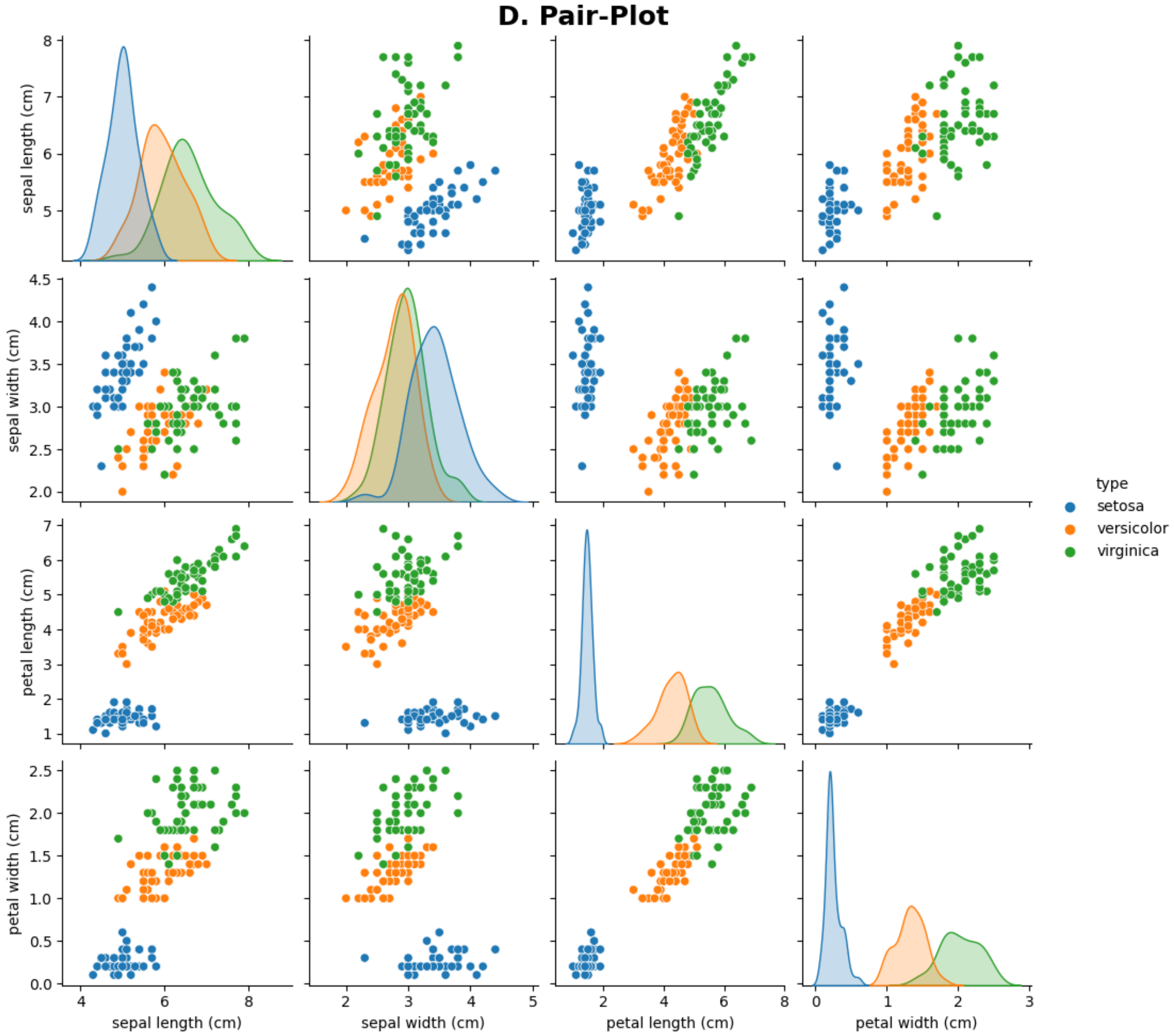
Out[119]:

```
Text(0.5, 1.0, 'C. Density Distribution')
```



In [43]:

```
# D. Pair Plot
pair = sns.pairplot(iris, hue = 'type')
pair.fig.suptitle('D. Pair-Plot', fontsize = 18, fontweight = 'bold')
pair.tight_layout()
plt.show()
```



Q.6)
Consider any sales training/ weather forecasting dataset :

---

a. Compute mean of a series grouped by another series.

b. Fill an intermittent time series to replace all missing dates with values of previous non-missing date.

c. Perform appropriate year-month string to dates conversion.

d. Split a dataset to group by two columns and then sort the aggregated results within the groups.

e. Split a given dataframe into groups with bin counts.

Code)

In [114…

```
import pandas as pd

df = pd.read_csv('weather.csv')

print(f'''
\t\t\t Q.6 Output
--------------------------------------------------------------------

Data file : \n
{df.head()}
\n{df.shape} \n
Details :
''')
df.info()
```

```
                         Q.6 Output
--------------------------------------------------------------------

Data file :

         date  temp_min  temp_max  wind_speed  humidity weather
0  31/10/2020        17        29           7        40   sunny
1   1/11/2020        18        30           8        39   sunny
2         NaN        19        30           9        49   sunny
3   3/11/2020        17        28          13        59   sunny
4   4/11/2020        20        29          18        89  cloudy

(22, 6)

Details :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        14 non-null     object
 1   temp_min    22 non-null     int64 
 2   temp_max    22 non-null     int64 
 3   wind_speed  22 non-null     int64 
 4   humidity    22 non-null     int64 
 5   weather     22 non-null     object
dtypes: int64(4), object(2)
memory usage: 1.2+ KB
```

In [95]:

```
print(f'''
--------------------------------------------------------------------
\n A.\n
Mean of the other series based on weather type : \n
{df.groupby(['weather']).mean(numeric_only = True)}
''')
```

```
--------------------------------------------------------------------

 A.

Mean of the other series based on weather type :

          temp_min   temp_max  wind_speed   humidity
weather                                             
cloudy   21.000000  30.000000   10.666667  79.333333
cold     18.000000  25.600000   12.200000  43.000000
sunny    17.833333  29.166667    9.833333  49.166667
```

In [115…

```
print(f'''
--------------------------------------------------------------------
\n B.\n
Total Null values in Date column : {df.date.isnull().sum()}\n
DataFrame before changing : \n
{df.head(10)}

Changing null Dates with previous non-null dates :
''')
df.fillna(method = 'ffill', inplace=True)
df.head(10)
```

```
--------------------------------------------------------------------

 B.

Total Null values in Date column : 8

DataFrame before changing :

         date  temp_min  temp_max  wind_speed  humidity weather
0  31/10/2020        17        29           7        40   sunny
1   1/11/2020        18        30           8        39   sunny
2         NaN        19        30           9        49   sunny
3   3/11/2020        17        28          13        59   sunny
4   4/11/2020        20        29          18        89  cloudy
5         NaN        20        30           8        69  cloudy
6   5/11/2020        23        31           6        80  cloudy
7   6/11/2020        20        28          10        35    cold
8         NaN        18        25          14        37    cold
9   8/11/2020        16        24          12        59    cold

Changing null Dates with previous non-null dates :
```

Out[115]:

	date	temp_min	temp_max	wind_speed	humidity	weather
0	31/10/2020	17	29	7	40	sunny
1	1/11/2020	18	30	8	39	sunny
2	1/11/2020	19	30	9	49	sunny
3	3/11/2020	17	28	13	59	sunny
4	4/11/2020	20	29	18	89	cloudy
5	4/11/2020	20	30	8	69	cloudy
6	5/11/2020	23	31	6	80	cloudy
7	6/11/2020	20	28	10	35	cold
8	6/11/2020	18	25	14	37	cold
9	8/11/2020	16	24	12	59	cold

In [121…

```
print(f'''
--------------------------------------------------------------------
\n C.\n
DataFrame before changing :
''')
df.info()
df.date = pd.to_datetime(df.date, dayfirst=True)
print('\n\nDataframe after converting string to dates :\n')
df.info()
```

```
--------------------------------------------------------------------

 C.

DataFrame before changing :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   date        22 non-null     datetime64[ns]
 1   temp_min    22 non-null     int64
 2   temp_max    22 non-null     int64
 3   wind_speed  22 non-null     int64
 4   humidity    22 non-null     int64
 5   weather     22 non-null     object
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 1.2+ KB


Dataframe after converting string to dates :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   date        22 non-null     datetime64[ns]
 1   temp_min    22 non-null     int64
 2   temp_max    22 non-null     int64
 3   wind_speed  22 non-null     int64
 4   humidity    22 non-null     int64
 5   weather     22 non-null     object
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 1.2+ KB
```

In [126…

```python
# d. Split a dataset to group by two columns and
#    sort the aggregated results within the groups.
df1 = df.groupby(["date", "weather"]).agg(lambda x: x.sort_values().head(3))
print(f'''
--------------------------------------------------------------------
\n D.\n
DataFrame grouped by weather & dates :\n
{df1}
''')
```

```
--------------------------------------------------------------------

 D.

DataFrame grouped by weather & dates :

                    temp_min  temp_max wind_speed  humidity
date       weather
2020-10-31 sunny          17        29          7        40
2020-11-01 sunny    [18, 19]  [30, 30]     [8, 9]  [39, 49]
2020-11-03 sunny          17        28         13        59
2020-11-04 cloudy   [20, 20]  [29, 30]    [8, 18]  [69, 89]
2020-11-05 cloudy         23        31          6        80
2020-11-06 cold     [18, 20]  [25, 28]   [10, 14]  [35, 37]
2020-11-08 cold           16        24         12        59
           sunny          19        30          9        49
2020-11-09 sunny          17        28         13        59
2020-11-10 cold     [18, 20]  [25, 28]   [10, 14]  [35, 37]
2020-11-12 cold     [16, 18]  [24, 25]   [12, 14]  [37, 59]
2020-11-14 cloudy   [20, 20]  [29, 30]    [8, 18]  [69, 89]
2020-11-15 cloudy         23        31          6        80
2020-11-16 cold     [18, 20]  [25, 28]   [10, 14]  [35, 37]
2020-11-18 cold           16        24         12        59
```

In [133…

```
print(f'''
--------------------------------------------------------------------
\n E.\n
DataFrame grouped by bin_counts :\n
{df.groupby(['wind_speed', pd.cut(df.humidity, 3)]).size().unstack()}
''')
```

```
--------------------------------------------------------------------

 E.

DataFrame grouped by bin_counts :

humidity    (34.946, 53.0]  (53.0, 71.0]  (71.0, 89.0]
wind_speed
6                        0             0             2
7                        1             0             0
8                        1             2             0
9                        2             0             0
10                       3             0             0
12                       0             3             0
13                       0             2             0
14                       4             0             0
18                       0             0             2
```

Q. 7)

Consider a data frame containing data about students i.e. name, gender and passing division:

S.N.	Name	Birth_Month	Gender	Pass_Division
0	Mudit Chauhan	December	M	III

S.N.	Name	Birth_Month	Gender	Pass_Division
1	Seema Chopra	January	F	II
2	Rani Gupta	March	F	I
3	Aditya Narayan	October	M	I
4	Sanjeev Sahni	February	M	II
5	Prakash Kumar	December	M	III
6	Ritu Agarwal	September	F	I
7	Akshay Goel	August	M	I
8	Meeta Kulkarni	July	F	II
9	Preeti Ahuja	November	F	II
10	Sunil Das Gupta	April	M	III
11	Sonali Sapre	January	F	I
12	Rashmi Talwar	June	F	III
13	Ashish Dubey	May	M	II
14	Kiran Sharma	February	F	II
15	Sameer Bansal	October	M	I

---

a. Perform one hot encoding of the last two columns of categorical data using the get_dummies() function.

b. Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to Categorical.

Code)

In [9]:
```
import pandas as pd
import numpy as np

bm = np.array(['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December'])

gen = np.array(['M', 'F'])
p_div = np.array(['I', 'II', 'III'])

df = pd.DataFrame({
    'Name' : ['Mudit Chauhan', 'Seema Chopra', 'Rani Gupta', 'Aditya Narayan',
                 'Sanjeev Sahni', 'Prakash Kumar', 'Ritu Agarwal', 'Akshay Goel',
                 'Meeta Kulkarni', 'Preeti Ahuja', 'Sunil Das Gupta', 'Sonali Sapre
                 'Rashmi Talwar', 'Ashish Dubey', 'Kiran Sharma', 'Sameer Bansal',
    'Birth_Month': bm[[11,0,2,9,1,11,8,7,6,10,3,0,5,4,1,9]],
    'Gender' : gen[[0,1,1,0,0,0,1,0,1,1,0,1,1,0,1,0]],
    'Pass_Division' : p_div[[2,1,0,0,1,2,0,0,1,1,2,0,2,1,1,0]]
})

print(f'''\t\t\t Q.7 Output
\n--------------------------------------------------------------------------\n
Given DataFrame : \n\n{df}''')
```

```
                         Q.7 Output

-------------------------------------------------------------------------

Given DataFrame :

               Name Birth_Month Gender Pass_Division
0     Mudit Chauhan    December      M           III
1      Seema Chopra     January      F            II
2        Rani Gupta       March      F             I
3    Aditya Narayan     October      M             I
4     Sanjeev Sahni    February      M            II
5     Prakash Kumar    December      M           III
6      Ritu Agarwal   September      F             I
7       Akshay Goel      August      M             I
8    Meeta Kulkarni        July      F            II
9      Preeti Ahuja    November      F            II
10  Sunil Das Gupta       April      M           III
11     Sonali Sapre     January      F             I
12    Rashmi Talwar        June      F           III
13     Ashish Dubey         May      M            II
14     Kiran Sharma    February      F            II
15    Sameer Bansal     October      M             I
```

In [11]:

```
print(f'''
\n-------------------------------------------------------------------------\n
A. \n
Performing one hot encoding on the last two columns : \n
{pd.get_dummies(df, columns=['Gender', 'Pass_Division'])}
''')
```

```
-------------------------------------------------------------------------

A.

Performing one hot encoding on the last two columns :

               Name Birth_Month  Gender_F  Gender_M  Pass_Division_I  \
0     Mudit Chauhan    December         0         1                0
1      Seema Chopra     January         1         0                0
2        Rani Gupta       March         1         0                1
3    Aditya Narayan     October         0         1                1
4     Sanjeev Sahni    February         0         1                0
5     Prakash Kumar    December         0         1                0
6      Ritu Agarwal   September         1         0                1
7       Akshay Goel      August         0         1                1
8    Meeta Kulkarni        July         1         0                0
9      Preeti Ahuja    November         1         0                0
10  Sunil Das Gupta       April         0         1                0
11     Sonali Sapre     January         1         0                1
12    Rashmi Talwar        June         1         0                0
13     Ashish Dubey         May         0         1                0
14     Kiran Sharma    February         1         0                0
15    Sameer Bansal     October         0         1                1

    Pass_Division_II  Pass_Division_III
0                  0                  1
1                  1                  0
2                  0                  0
3                  0                  0
4                  1                  0
5                  0                  1
6                  0                  0
7                  0                  0
8                  1                  0
9                  1                  0
10                 0                  1
11                 0                  0
12                 0                  1
13                 1                  0
14                 1                  0
15                 0                  0
```

In [3]:

```
df['Birth_Month'] = pd.Categorical(df['Birth_Month'], categories=bm)
print(f'''------------------------------------------------------------------------
B. \n
Sorting DataFrame by the Birth_Month : \n
{df.sort_values(by = 'Birth_Month')}
\n-------------------------------------------------------------------------
''')
```

```
-------------------------------------------------------------------------

B.

Sorting DataFrame by the Birth_Month :

               Name Birth_Month Gender Pass_Division
1      Seema Chopra     January      F            II
11     Sonali Sapre     January      F             I
4     Sanjeev Sahni    February      M            II
14     Kiran Sharma    February      F            II
2        Rani Gupta       March      F             I
10  Sunil Das Gupta       April      M           III
13     Ashish Dubey         May      M            II
12    Rashmi Talwar        June      F           III
8    Meeta Kulkarni        July      F            II
7       Akshay Goel      August      M             I
6      Ritu Agarwal   September      F             I
3    Aditya Narayan     October      M             I
15    Sameer Bansal     October      M             I
9      Preeti Ahuja    November      F            II
0     Mudit Chauhan    December      M           III
5     Prakash Kumar    December      M           III

-------------------------------------------------------------------------
```

Q. 8)
Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record.

Name	Gender	MonthlyIncome(Rs.)
Shah	**Male**	**114000.00**
Vats	**Male**	**65000.00**
Vats	**Female**	**43150.00**
Kumar	**Female**	**69500.00**
Vats	**Female**	**155000.00**
Kumar	**Male**	**103000.00**
Shah	**Male**	**55000.00**
Shah	**Female**	**112400.00**
Kumar	**Female**	**81030.00**
Vats	**Male**	**71900.00**

Write a program in Python using Pandas to perform the following:

a. Calculate and display familywise gross monthly income.

b. Calculate and display the member with the highest monthly income in a family.

c. Calculate and display monthly income of all members with income greater than Rs. 60000.00.

d. Calculate and display the average monthly income of the female members in the Shah family.

Code)

In [7]:

```
import pandas as pd
import numpy as np

name = np.array(['Shah', 'Vats', 'Kumar'])
gender = np.array(['Male', 'Female'])

f_inc = pd.DataFrame({
    'Name' : name[[0,1,1,2,1,2,0,0,2,1]],
    'Gender' : gender[[0,0,1,1,1,0,0,1,1,0]],
    'MonthlyIncome' : [114000, 65000, 43150, 69500, 155000, 103000, 55000, 112400,
})

print(f'''\t\t\t Q.8 Output
\n-------------------------------------------------------------------------\n
Given DataFrame : \n\n{f_inc}
\n-------------------------------------------------------------------------\n
A. \n
Calculating Familywise Gross Monthly Income : \n
{f_inc.groupby(by = 'Name')['MonthlyIncome'].sum()}
''')
```

```
                    Q.8 Output

-------------------------------------------------------------------------

Given DataFrame :

    Name  Gender  MonthlyIncome
0   Shah    Male         114000
1   Vats    Male          65000
2   Vats  Female          43150
3  Kumar  Female          69500
4   Vats  Female         155000
5  Kumar    Male         103000
6   Shah    Male          55000
7   Shah  Female         112400
8  Kumar  Female          81030
9   Vats    Male          71900

-------------------------------------------------------------------------

A.

Calculating Familywise Gross Monthly Income :

Name
Kumar    253530
Shah     281400
Vats     335050
Name: MonthlyIncome, dtype: int64
```

In [8]:

```python
print(f'''
\n---------------------------------------------------------------------------\n
B. \n
Calculating Familywise Highesh Monthly Income : \n
{f_inc.groupby(by = ['Name', 'Gender'])['MonthlyIncome'].max()}
\n---------------------------------------------------------------------------\n
C. \n
Calculating Members\' Monthly Income > 60000  : \n
{f_inc[f_inc.MonthlyIncome > 60000]}
\n---------------------------------------------------------------------------\n
D. \n
Calculating average salary of female Shah member :  \n
{f_inc[(f_inc.Name == 'Shah') & (f_inc.Gender == 'Female')]['MonthlyIncome'].mean()
\n---------------------------------------------------------------------------\n
''')
```

```
-------------------------------------------------------------------------

B.

Calculating Familywise Highesh Monthly Income :

Name   Gender
Kumar  Female     81030
       Male      103000
Shah   Female    112400
       Male      114000
Vats   Female    155000
       Male       71900
Name: MonthlyIncome, dtype: int64

-------------------------------------------------------------------------

C.

Calculating Members' Monthly Income > 60000  :

    Name  Gender  MonthlyIncome
0   Shah    Male         114000
1   Vats    Male          65000
3  Kumar  Female          69500
4   Vats  Female         155000
5  Kumar    Male         103000
7   Shah  Female         112400
8  Kumar  Female          81030
9   Vats    Male          71900

-------------------------------------------------------------------------

D.

Calculating average salary of female Shah member :

112400.0 Rs.

-------------------------------------------------------------------------
```