FDA Lab-9

KHAN MOHD OWAIS RAZA
20BCD7138

Q1] Consider the following pandas DataFrame:

```
import pandas as pd
data = {'Name': ['John', 'Emily', 'Michael', 'Jessica'],
        'Age': [25, 28, 32, 30],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston'],
        'Salary': [50000, 60000, 70000, 55000]}
df = pd.DataFrame(data)
```

1. Introducing Pandas Objects:
a) Print the entire DataFrame `df` and observe its structure and contents.
b) Determine and print the shape of the DataFrame `df`.
c) Identify the data types of each column in the DataFrame `df` and print the results.

2. Data Indexing and Selection:
a) Access and print the 'Name' column of the DataFrame `df`.
b) Access and print the second row of the DataFrame `df`.
c) Access and print the salary of the employee with the name 'Michael'.
d) Create a new DataFrame `df_filtered` that includes only the rows where the age is greater than 27. Print the new DataFrame.

```
>>
= RESTART: C:/Users/asus/Documents/fda_lab11_q1.py
DataFrame df:
      Name  Age         City  Salary
0     John   25     New York   50000
1    Emily   28  Los Angeles   60000
2  Michael   32      Chicago   70000
3  Jessica   30      Houston   55000
Shape of df: (4, 4)
Data types of df columns:
Name      object
Age        int64
City      object
Salary     int64
dtype: object
Name column:
0       John
1      Emily
2    Michael
3    Jessica
Name: Name, dtype: object
Second row:
Name            Emily
Age                28
City      Los Angeles
Salary          60000
Name: 1, dtype: object
Salary of Michael:
2    70000
Name: Salary, dtype: int64
Filtered DataFrame df_filtered:
      Name  Age         City  Salary
1    Emily   28  Los Angeles   60000
2  Michael   32      Chicago   70000
3  Jessica   30      Houston   55000
>>
```

Q2] Consider the following pandas DataFrame:
import pandas as pd
data = {'Name': ['John', 'Emily', 'Michael', 'Jessica'],
'Age': [25, 28, 32, 30],
'City': ['New York', 'Los Angeles', 'Chicago', 'Houston'],
'Salary': [50000, 60000, 70000, 55000]}
df = pd.DataFrame(data)

1. Operating on Data in Pandas:
a) Calculate and print the mean age of the employees in the DataFrame `df`.
b) Calculate and print the total salary of all employees in the DataFrame `df`.
c) Calculate and print the maximum salary among all employees in the DataFrame `df`.

2. Data Manipulation:
a) Add a new column named 'Bonus' to the DataFrame `df` with random bonus values between 1000 and 5000 for each employee. Print the modified DataFrame.
b) Increase the salary of all employees in the DataFrame `df` by 10%. Print the updated salary column.

3. Data Aggregation:
a) Group the DataFrame `df` by the 'City' column and calculate the average age for each city. Print the resulting DataFrame.
b) Group the DataFrame `df` by the 'City' column and calculate the total salary for each city. Print the resulting DataFrame.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd
import numpy as np

data = {
    'Name': ['John', 'Emily', 'Michael', 'Jessica'],
    'Age': [25, 28, 32, 30],
    'City': ['New York', 'Los Angeles', 'Chicago',
'Houston'],
    'Salary': [50000, 60000, 70000, 55000]
}

df = pd.DataFrame(data)

# 1. Operating on Data in Pandas
# a) Calculate and print the mean age of the employees in
the DataFrame df
mean_age = df['Age'].mean()
print("Mean Age:", mean_age)
```

```
# b) Calculate and print the total salary of all
employees in the DataFrame df
total_salary = df['Salary'].sum()
print("Total Salary:", total_salary)

# c) Calculate and print the maximum salary among all
employees in the DataFrame df
max_salary = df['Salary'].max()
print("Maximum Salary:", max_salary)

# 2. Data Manipulation
# a) Add a new column named 'Bonus' to the DataFrame df
with random bonus values between 1000 and 5000 for each
employee
df['Bonus'] = np.random.randint(1000, 5001, len(df))
print("DataFrame with 'Bonus' column:")
print(df)

# b) Increase the salary of all employees in the
DataFrame df by 10%
df['Salary'] = df['Salary'] * 1.1
print("Updated Salary Column:")
print(df['Salary'])

# 3. Data Aggregation
# a) Group the DataFrame df by the 'City' column and
calculate the average age for each city
average_age_by_city = df.groupby('City')['Age'].mean()
print("Average Age by City:")
print(average_age_by_city)

# b) Group the DataFrame df by the 'City' column and
calculate the total salary for each city
total_salary_by_city = df.groupby('City')['Salary'].sum()
print("Total Salary by City:")
print(total_salary_by_city)
```

```
-----------------------------------------
Mean Age: 28.75
Total Salary: 235000
Maximum Salary: 70000
DataFrame with 'Bonus' column:
      Name  Age         City  Salary  Bonus
0     John   25     New York   50000   1045
1    Emily   28  Los Angeles   60000   2217
2  Michael   32      Chicago   70000   3979
3  Jessica   30      Houston   55000   2382
Updated Salary Column:
0    55000.0
1    66000.0
2    77000.0
3    60500.0
Name: Salary, dtype: float64
Average Age by City:
City
Chicago        32.0
Houston        30.0
Los Angeles    28.0
New York       25.0
Name: Age, dtype: float64
Total Salary by City:
City
Chicago        77000.0
Houston        60500.0
Los Angeles    66000.0
New York       55000.0
Name: Salary, dtype: float64
```

Q3] Consider the following pandas DataFrame:

```
import pandas as pd
import numpy as np
data = {'Name': ['John', 'Emily', np.nan, 'Jessica'],
        'Age': [25, np.nan, 32, 30],
        'City': ['New York', 'Los Angeles', 'Chicago', np.nan],
        'Salary': [50000, 60000, np.nan, 55000]}
df = pd.DataFrame(data)
```

1. Detection of Missing Data:
a) Identify and print the total number of missing values in each column of the DataFrame `df`.
b) Determine and print the percentage of missing values in the 'Age' column.

2. Handling of Missing Data:
a) Remove any rows that contain missing values from the DataFrame `df` and assign the result to a new DataFrame `df_clean`. Print the new DataFrame.
b) Fill the missing values in the 'Salary' column of the DataFrame `df` with the mean salary value. Print the modified DataFrame.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd
import numpy as np

data = {
    'Name': ['John', 'Emily', np.nan, 'Jessica'],
```

```
    'Age': [25, np.nan, 32, 30],
    'City': ['New York', 'Los Angeles', 'Chicago',
np.nan],
    'Salary': [50000, 60000, np.nan, 55000]
}

df = pd.DataFrame(data)

# 1. Detection of Missing Data
# a) Identify and print the total number of missing
values in each column of the DataFrame df
missing_values_count = df.isnull().sum()
print("Total number of missing values in each column:")
print(missing_values_count)

# b) Determine and print the percentage of missing values
in the 'Age' column
age_missing_percentage = df['Age'].isnull().sum() /
len(df) * 100
print("Percentage of missing values in the 'Age'
column:", age_missing_percentage)

# 2. Handling of Missing Data
# a) Remove any rows that contain missing values from the
DataFrame df and assign the result to a new DataFrame
df_clean
df_clean = df.dropna()
print("DataFrame after removing rows with missing
values:")
print(df_clean)

# b) Fill the missing values in the 'Salary' column of
the DataFrame df with the mean salary value
mean_salary = df['Salary'].mean()
df['Salary'].fillna(mean_salary, inplace=True)
print("DataFrame after filling missing values in 'Salary'
column:")
print(df)
```

```
============================================================
Total number of missing values in each column:
Name      1
Age       1
City      1
Salary    1
dtype: int64
Percentage of missing values in the 'Age' column: 25.0
DataFrame after removing rows with missing values:
   Name   Age      City   Salary
0  John  25.0  New York  50000.0
DataFrame after filling missing values in 'Salary' column:
      Name   Age         City   Salary
0     John  25.0     New York  50000.0
1    Emily   NaN  Los Angeles  60000.0
2      NaN  32.0      Chicago  55000.0
3  Jessica  30.0          NaN  55000.0
```

Q4] Consider the following pandas DataFrame:
import pandas as pd
import numpy as np
data = {'Name': ['John', 'Emily', np.nan, 'Jessica'],
'Age': [25, np.nan, 32, 30],
'City': ['New York', 'Los Angeles', 'Chicago', np.nan],
'Salary': [50000, 60000, np.nan, 55000]}
df = pd.DataFrame(data)

1. Detection of Missing Data:
a) Identify and print the total number of missing values in each column of the DataFrame `df`.
b) Determine and print the percentage of missing values in the 'Age' column.

2. Handling of Missing Data:
a) Remove any rows that contain missing values from the DataFrame `df` and assign the result to a new DataFrame `df_clean`. Print the new DataFrame.
b) Fill the missing values in the 'Salary' column of the DataFrame `df` with the mean salary value. Print the modified DataFrame.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd
import numpy as np
data = {
    'Name': ['John', 'Emily', np.nan, 'Jessica'],
    'Age': [25, np.nan, 32, 30],
    'City': ['New York', 'Los Angeles', 'Chicago',
np.nan],
    'Salary': [50000, 60000, np.nan, 55000]
}
df = pd.DataFrame(data)
# 1. Detection of Missing Data
# a) Identify and print the total number of missing
values in each column of the DataFrame df
missing_values_count = df.isnull().sum()
print("Total number of missing values in each column:")
print(missing_values_count)
# b) Determine and print the percentage of missing values
in the 'Age' column
age_missing_percentage = df['Age'].isnull().sum() /
len(df) * 100
print("Percentage of missing values in the 'Age'
column:", age_missing_percentage)
# 2. Handling of Missing Data
```

```
# a) Remove any rows that contain missing values from the
DataFrame df and assign the result to a new DataFrame
df_clean
df_clean = df.dropna()
print("DataFrame after removing rows with missing
values:")
print(df_clean)
# b) Fill the missing values in the 'Salary' column of
the DataFrame df with the mean salary value
mean_salary = df['Salary'].mean()
df['Salary'].fillna(mean_salary, inplace=True)
print("DataFrame after filling missing values in 'Salary'
column:")
print(df)
```

```
============================================================
Total number of missing values in each column:
Name      1
Age       1
City      1
Salary    1
dtype: int64
Percentage of missing values in the 'Age' column: 25.0
DataFrame after removing rows with missing values:
   Name   Age      City   Salary
0  John  25.0  New York  50000.0
DataFrame after filling missing values in 'Salary' column:
      Name   Age         City   Salary
0     John  25.0     New York  50000.0
1    Emily   NaN  Los Angeles  60000.0
2      NaN  32.0      Chicago  55000.0
3  Jessica  30.0          NaN  55000.0
```

Q5] Consider the following pandas DataFrame:

```
import pandas as pd
data = {'Group': ['A', 'A', 'B', 'B', 'C', 'C'],
        'Category': ['X', 'Y', 'X', 'Y', 'X', 'Y'],
        'Value': [10, 20, 30, 40, 50, 60]}
df = pd.DataFrame(data)
df = df.set_index(['Group', 'Category'])
```

1. Hierarchical Indexing:

a) Print the DataFrame `df` and observe its structure with the hierarchical index.

b) Access and print the value corresponding to Group 'A' and Category 'X'.

c) Access and print the sub-dataframe corresponding to all rows with Group 'B'.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd

data = {
```

```
    'Group': ['A', 'A', 'B', 'B', 'C', 'C'],
    'Category': ['X', 'Y', 'X', 'Y', 'X', 'Y'],
    'Value': [10, 20, 30, 40, 50, 60]
}

df = pd.DataFrame(data)
df = df.set_index(['Group', 'Category'])

# 1. Hierarchical Indexing
# a) Print the DataFrame df and observe its structure
with the hierarchical index.
print("DataFrame with hierarchical index:")
print(df)

# b) Access and print the value corresponding to Group
'A' and Category 'X'.
value_A_X = df.loc[('A', 'X'), 'Value']
print("Value corresponding to Group 'A' and Category
'X':", value_A_X)

# c) Access and print the sub-dataframe corresponding to
all rows with Group 'B'.
sub_df_B = df.loc['B']
print("Sub-dataframe with Group 'B':")
print(sub_df_B)
```

```
==========================================================
DataFrame with hierarchical index:
                Value
Group Category
A     X            10
      Y            20
B     X            30
      Y            40
C     X            50
      Y            60
Value corresponding to Group 'A' and Category 'X': 10
Sub-dataframe with Group 'B':
          Value
Category
X            30
Y            40
```

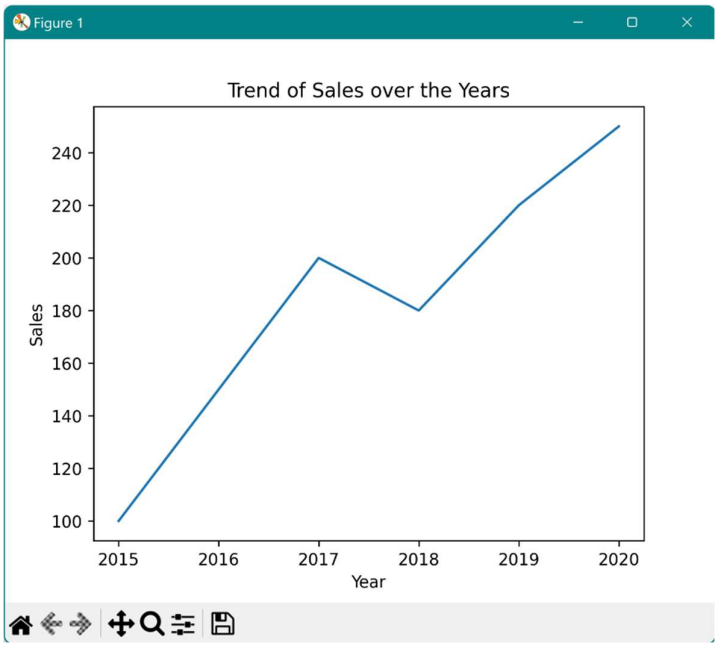
Q6] Consider the following example dataset:
import pandas as pd
data = {'Year': [2015, 2016, 2017, 2018, 2019, 2020],
'Sales': [100, 150, 200, 180, 220, 250],
'Profit': [10, 20, 25, 15, 30, 35]}
df = pd.DataFrame(data)

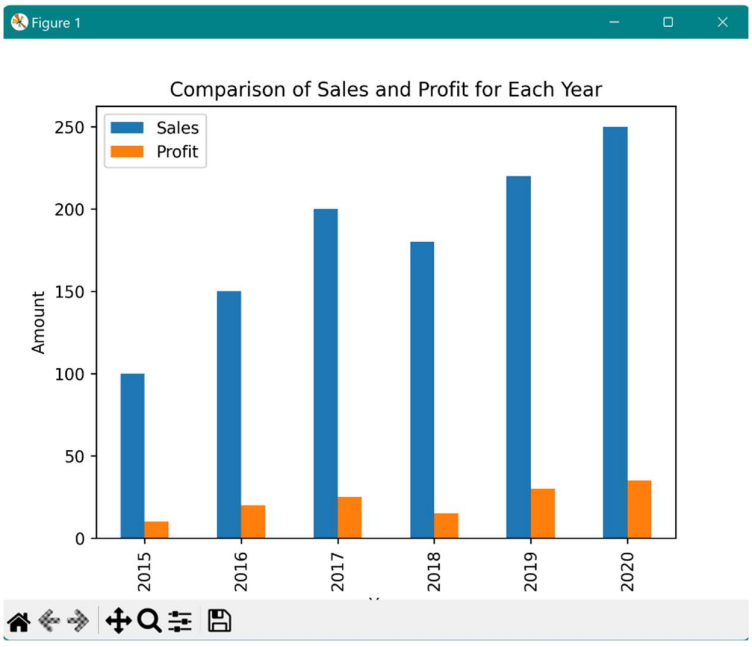
1. Visualization with Matplotlib:

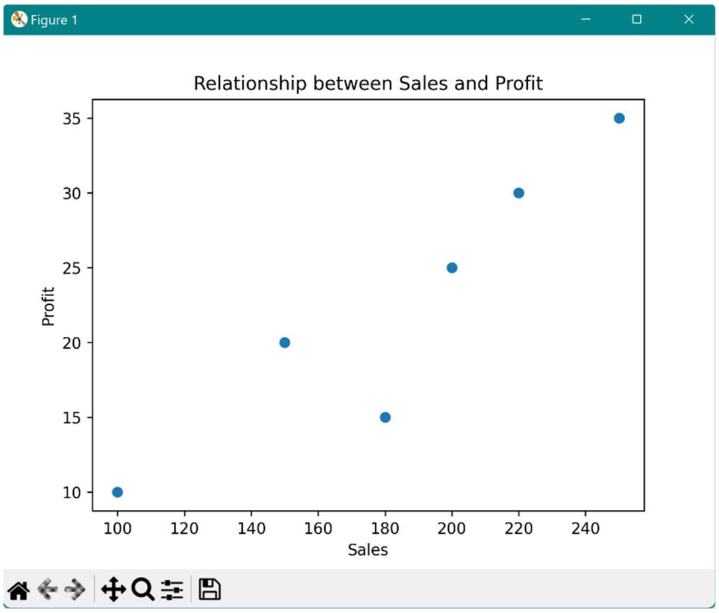
a) Create a line plot to visualize the trend of sales over the years. Add appropriate labels and title to the plot.
b) Create a bar plot to compare the sales and profit for each year. Add appropriate labels and title to the plot.
c) Create a scatter plot to show the relationship between sales and profit. Add appropriate labels and title to the plot.
d) Create a box plot to display the distribution of sales and profit. Add appropriate labels and title to the plot.
e) Create a histogram to visualize the distribution of sales. Add appropriate labels and title to the plot.
f) Create a pie chart to represent the proportion of sales for each year. Add appropriate labels and title to the plot.

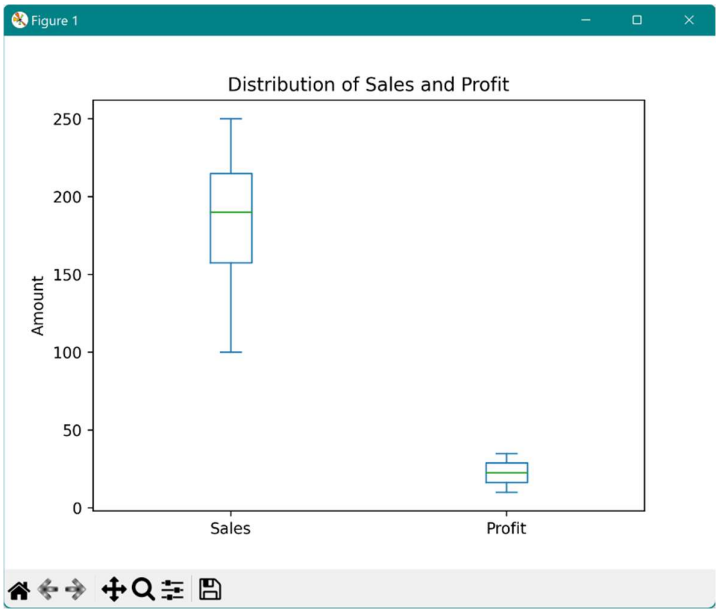
```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Year': [2015, 2016, 2017, 2018, 2019, 2020],
    'Sales': [100, 150, 200, 180, 220, 250],
    'Profit': [10, 20, 25, 15, 30, 35]
}

df = pd.DataFrame(data)

# 1. Visualization with Matplotlib

# a) Line plot to visualize the trend of sales over the
years
plt.plot(df['Year'], df['Sales'])
plt.xlabel('Year')
plt.ylabel('Sales')
plt.title('Trend of Sales over the Years')
plt.show()

# b) Bar plot to compare the sales and profit for each
year
df.plot(x='Year', y=['Sales', 'Profit'], kind='bar')
plt.xlabel('Year')
plt.ylabel('Amount')
plt.title('Comparison of Sales and Profit for Each Year')
plt.legend(['Sales', 'Profit'])
plt.show()

# c) Scatter plot to show the relationship between sales
and profit
```

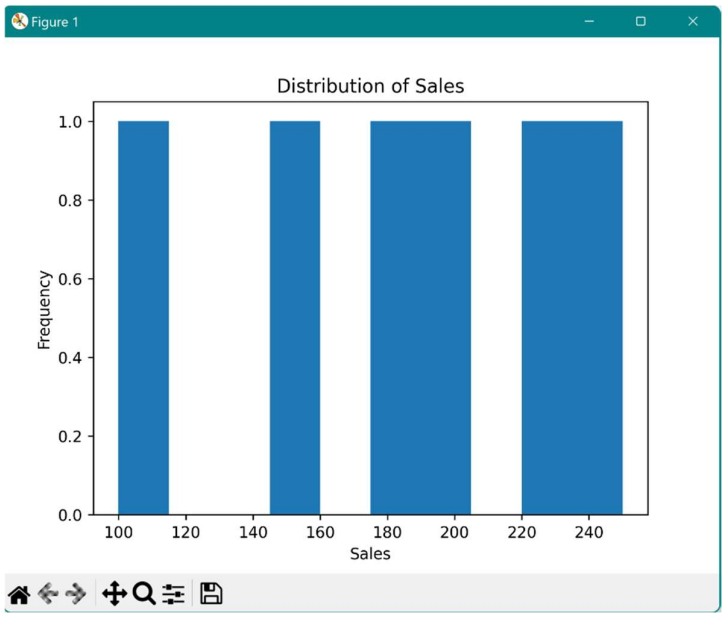
```
plt.scatter(df['Sales'], df['Profit'])
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.title('Relationship between Sales and Profit')
plt.show()

# d) Box plot to display the distribution of sales and
profit
df[['Sales', 'Profit']].plot(kind='box')
plt.ylabel('Amount')
plt.title('Distribution of Sales and Profit')
plt.show()

# e) Histogram to visualize the distribution of sales
df['Sales'].plot(kind='hist')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.title('Distribution of Sales')
plt.show()

# f) Pie chart to represent the proportion of sales for
each year
df.plot(x='Year', y='Sales', kind='pie',
labels=df['Year'], autopct='%1.1f%%')
plt.ylabel('Sales')
plt.title('Proportion of Sales for Each Year')
plt.show()
```

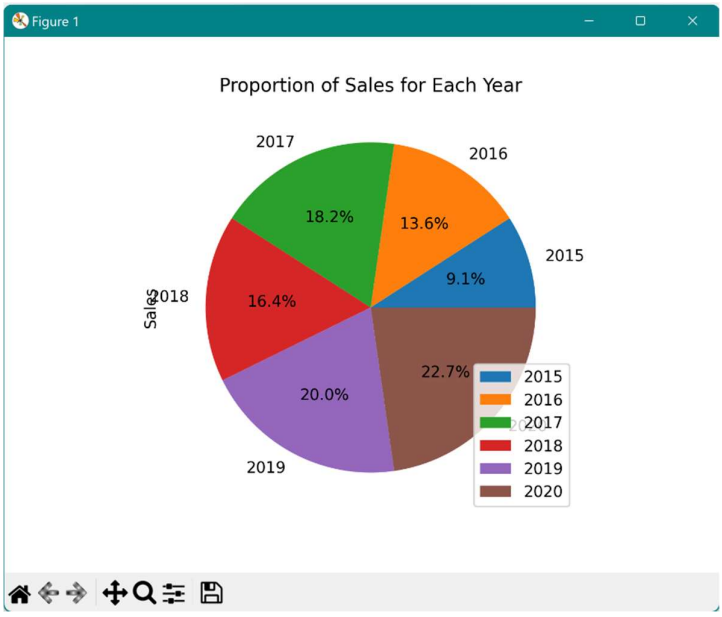












Q7] Consider the following pandas Series:
import pandas as pd
series = pd.Series(['apple', 'banana', 'cherry', 'durian', 'elderberry'])

1. Vectorized String Operations:
a) Convert all the strings in the Series `series` to uppercase. Print the modified Series.
b) Determine the length of each string in the Series `series`. Print the resulting Series.
c) Check if each string in the Series `series` contains the letter 'a'. Print the resulting Series of boolean values.
d) Replace all occurrences of the letter 'a' in each string of the Series `series` with the letter 'x'. Print the modified Series.
e) Extract the substring consisting of the first three characters from each string in the Series `series`. Print the resulting Series.
f) Count the number of occurrences of the letter 'r' in each string of the Series `series`. Print the resulting Series.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import pandas as pd

series = pd.Series(['apple', 'banana', 'cherry',
'durian', 'elderberry'])

# 1. Vectorized String Operations

# a) Convert all the strings in the Series `series` to
uppercase
series_upper = series.str.upper()
print(series_upper)

# b) Determine the length of each string in the Series
`series`
series_length = series.str.len()
print(series_length)

# c) Check if each string in the Series `series` contains
the letter 'a'
series_contains_a = series.str.contains('a')
print(series_contains_a)

# d) Replace all occurrences of the letter 'a' in each
string of the Series `series` with the letter 'x'
series_replace_a = series.str.replace('a', 'x')
print(series_replace_a)

# e) Extract the substring consisting of the first three
characters from each string in the Series `series`
series_substring = series.str[:3]
print(series_substring)

# f) Count the number of occurrences of the letter 'r' in
each string of the Series `series`
series_count_r = series.str.count('r')
print(series_count_r)
```

```
==========================================
0         APPLE
1        BANANA
2        CHERRY
3        DURIAN
4    ELDERBERRY
dtype: object
0     5
1     6
2     6
3     6
4    10
dtype: int64
0     True
1     True
2    False
3     True
4    False
dtype: bool
0         xpple
1        bxnxnx
2        cherry
3        durixn
4    elderberry
dtype: object
0    app
1    ban
2    che
3    dur
4    eld
dtype: object
0    0
1    0
2    2
3    1
4    3
dtype: int64
```