RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

Department of Computer Science and Engineering

CS23334 Fundamentals of Data Science Lab

III semester II Year (2023R)

Name of the Student : Feminna G

Register Number : 240701137

Exercise-1
Matplotlib

Name: Feminna G

Roll no: 240701137

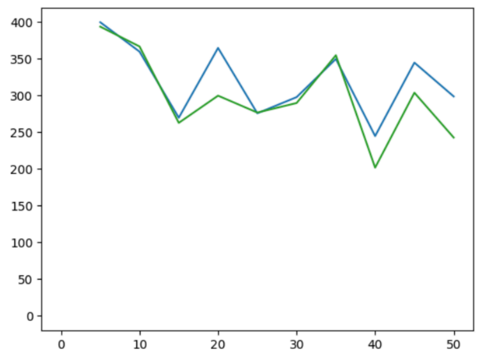
CSE Batch-2

[5]

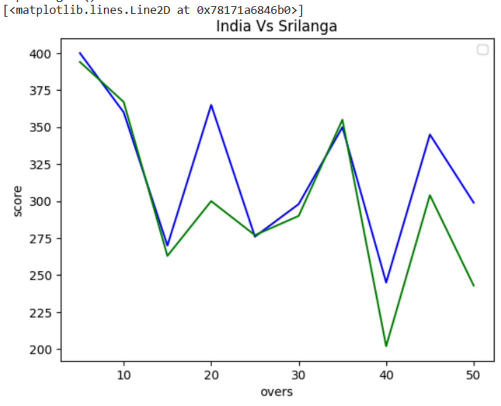
```
#Line plot
```

[9]

```
import matplotlib.pyplot as plt
overs=list(range(5,51,5))
India_Score=[400,360,270,365,276,298,350,245,345,299]
Srilanga_Score=[394,367,263,300,277,290,355,202,304,243]
plt.plot(overs,India_Score,'color'=='blue')
plt.plot(overs,Srilanga_Score)
plt.show()
plt.title("India Vs Srilanga")
plt.xlabel("overs")
plt.ylabel("score")
plt.legend()
plt.plot(overs,India_Score,color="blue",label="India")
plt.plot(overs,Srilanga_Score,color="green",label="Srilanga")
```



```
/tmp/ipython-input-1389799779.py:11: UserWarning: No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when lege
  plt.legend()
[<matplotlib.lines.Line2D at 0x78171a6846b0>]
```

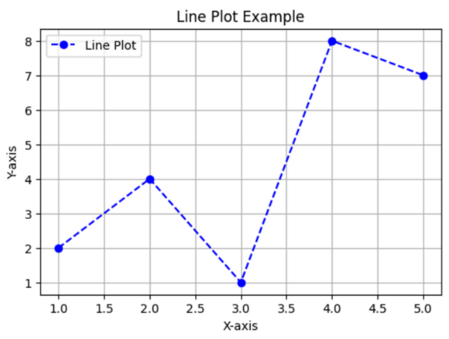


[10]
✓ 0s

```
# LINE PLOT
```

[11]
✓ 0s

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 8, 7]
plt.figure(figsize=(6, 4))  # Set the figure size
plt.plot(x, y, color='blue', marker='o', linestyle='--', label='Line Plot')
plt.title("Line Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.grid(True)
plt.show()
```

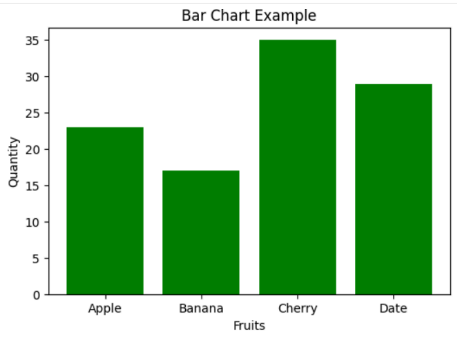


[12]
✓ 0s

```
# BAR CHART
```

[13]
✓ 0s

```
categories = ['Apple', 'Banana', 'Cherry', 'Date']
values = [23, 17, 35, 29]

plt.figure(figsize=(6, 4))
plt.bar(categories, values, color='green')
plt.title("Bar Chart Example")
plt.xlabel("Fruits")
plt.ylabel("Quantity")
plt.show()
```

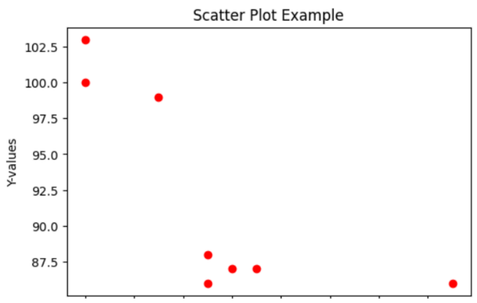


[14]
✓ 0s

```
# SCATTER PLOT
```

[15]
✓ 0s

```
x_scatter = [5, 7, 8, 7, 2, 17, 2, 9]
y_scatter = [99, 86, 87, 88, 100, 86, 103, 87]
plt.figure(figsize=(6, 4))
plt.scatter(x_scatter, y_scatter, color='red')
plt.title("Scatter Plot Example")
plt.xlabel("X-values")
plt.ylabel("Y-values")
plt.show()
```

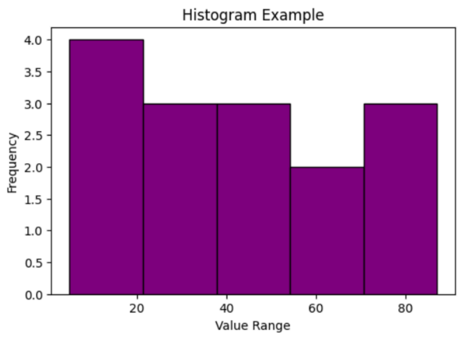


[16]
✓ 0s

```
# HISTOGRAM
```

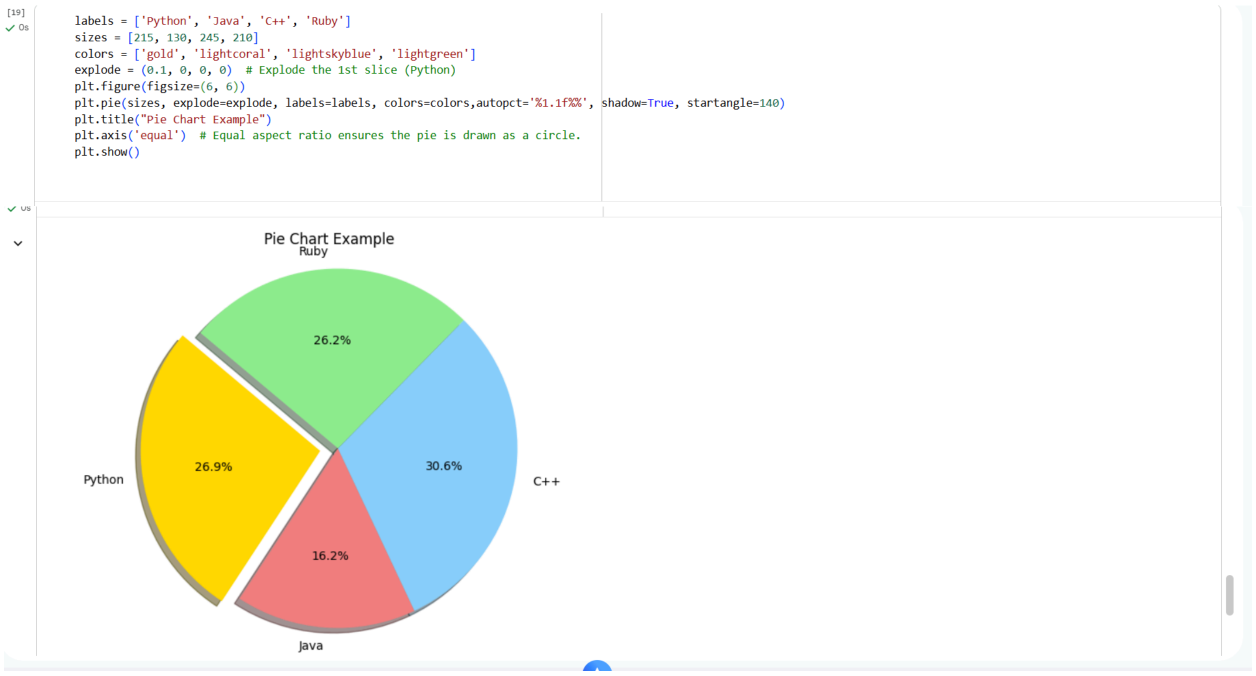
[17]
✓ 0s

```
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
plt.figure(figsize=(6, 4))
plt.hist(data, bins=5, color='purple', edgecolor='black')
plt.title("Histogram Example")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```



[18]
✓ 0s

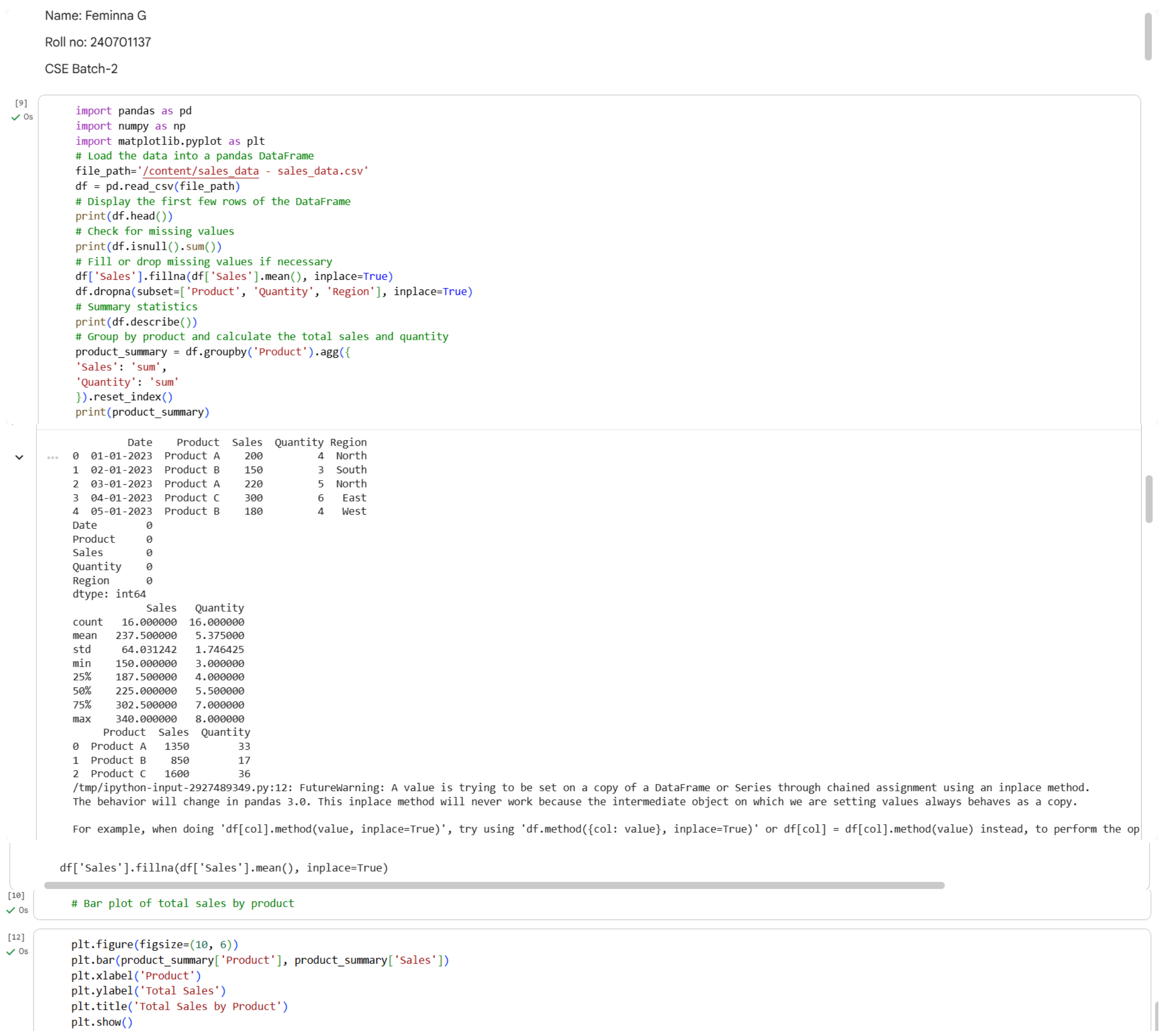
```
# PIE CHART
```



[19]
✓ 0s

```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [215, 130, 245, 210]
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']
explode = (0.1, 0, 0, 0)  # Explode the 1st slice (Python)
plt.figure(figsize=(6, 6))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("Pie Chart Example")
plt.axis('equal')  # Equal aspect ratio ensures the pie is drawn as a circle.
plt.show()
```

Exercise-2
data
preprocessing



Name: Feminna G

Roll no: 240701137

CSE Batch-2

[9]
✓ 0s

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Load the data into a pandas DataFrame
file_path='/content/sales_data - sales_data.csv'
df = pd.read_csv(file_path)
# Display the first few rows of the DataFrame
print(df.head())
# Check for missing values
print(df.isnull().sum())
# Fill or drop missing values if necessary
df['Sales'].fillna(df['Sales'].mean(), inplace=True)
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)
# Summary statistics
print(df.describe())
# Group by product and calculate the total sales and quantity
product_summary = df.groupby('Product').agg({
'Sales': 'sum',
'Quantity': 'sum'
}).reset_index()
print(product_summary)
```

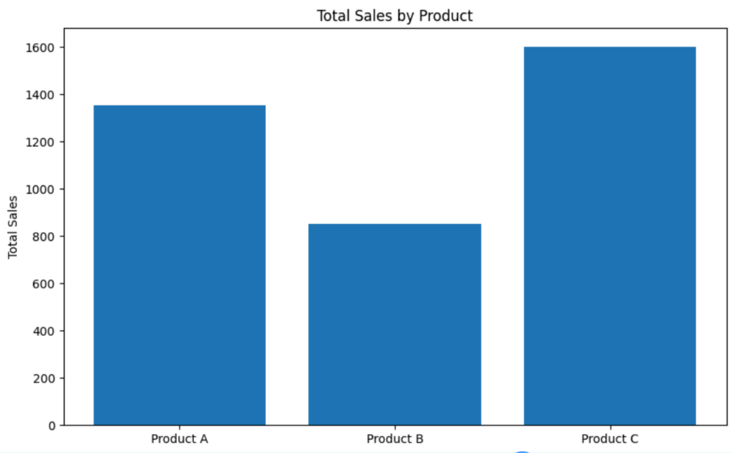
```
         Date    Product  Sales  Quantity Region
0  01-01-2023  Product A    200         4  North
1  02-01-2023  Product B    150         3  South
2  03-01-2023  Product A    220         5  North
3  04-01-2023  Product C    300         6   East
4  05-01-2023  Product B    180         4   West
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64
            Sales   Quantity
count   16.000000  16.000000
mean   237.500000   5.375000
std     64.031242   1.746425
min    150.000000   3.000000
25%    187.500000   4.000000
50%    225.000000   5.500000
75%    302.500000   7.000000
max    340.000000   8.000000
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
/tmp/ipython-input-2927489349.py:12: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the op

  df['Sales'].fillna(df['Sales'].mean(), inplace=True)
```

[10]
✓ 0s

```
# Bar plot of total sales by product
```

[12]
✓ 0s

```
plt.figure(figsize=(10, 6))
plt.bar(product_summary['Product'], product_summary['Sales'])
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.show()
```

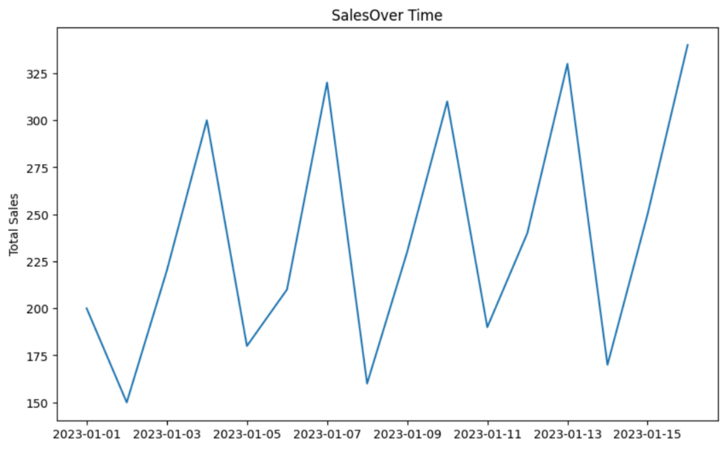


[14]
0s

```
# Line plot of sales over time
```

[18]
0s

```
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
sales_over_time = df.groupby('Date').agg({'Sales': 'sum'}).reset_index()
plt.figure(figsize=(10, 6))
plt.plot(sales_over_time['Date'],sales_over_time['Sales'])
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('SalesOver Time')
plt.show()
```



```
# Pivot table to analyze sales by region and product
```

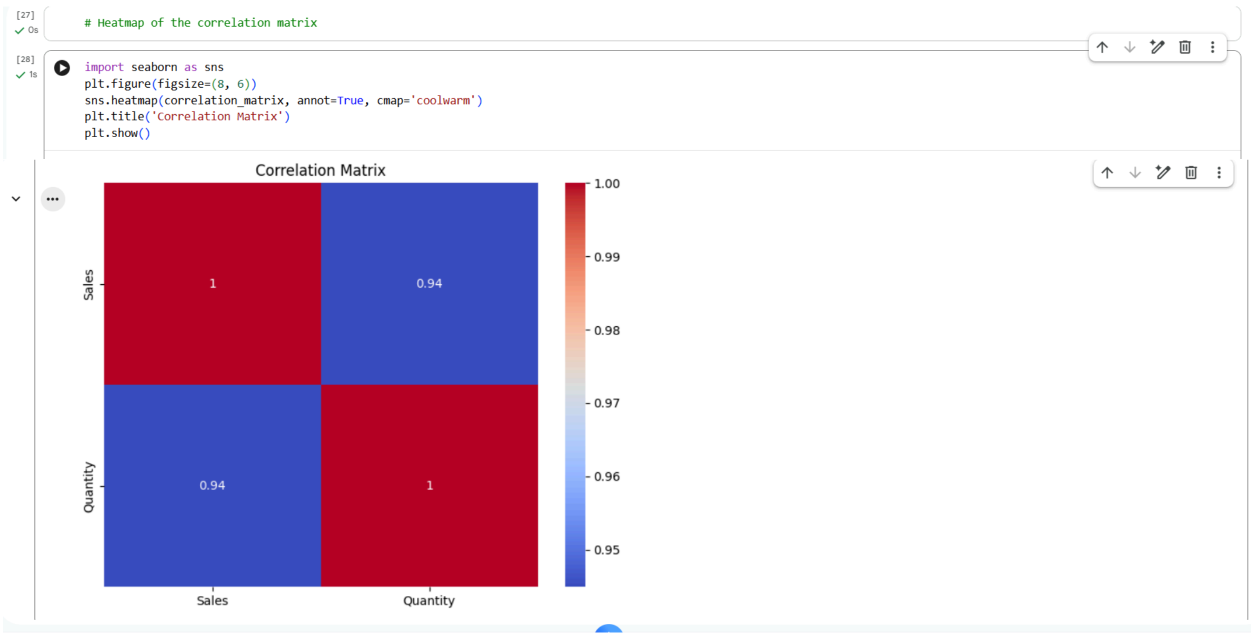
```
pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
aggfunc=np.sum, fill_value=0)
print(pivot_table)
```

```
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0
/tmp/ipython-input-4129461768.py:1: FutureWarning: The provided callable <function sum at 0x792017f3b100> is currently using DataFrameGroupBy.sum. In a future version of pand
  pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
```

```
# Correlation matrix
```

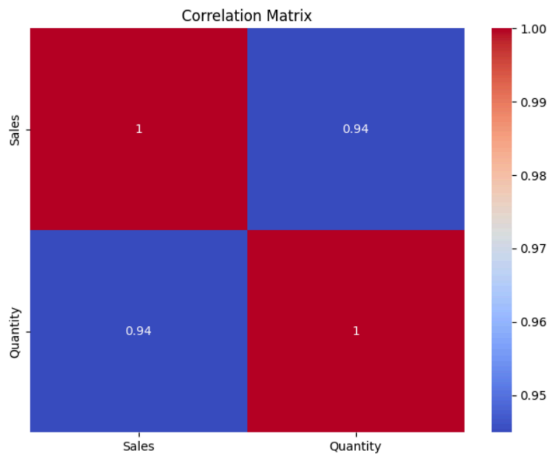
```
correlation_matrix = df.corr(numeric_only=True)
print(correlation_matrix)
```

```
             Sales  Quantity
Sales     1.000000  0.944922
Quantity  0.944922  1.000000
```



```
[27]
# Heatmap of the correlation matrix
```

```
[28]
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



Exercise-3

Handling Missing and Inappropriate Data in a Dataset

In [19]:
```
#Feminna G
#Roll no: 240701137
#cse-Batch 2

import numpy as np
import pandas as pd
df=pd.read_csv("pre_process_datasample.csv")
df
```

Out[19]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    9 non-null      object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

In [18]:
```
df.Country.mode()
```

Out[18]:
```
0    France
Name: Country, dtype: object
```

In [4]:
```
df.Country.mode()[0]
```

Out[4]:
```
'France'
```

In [5]:
```
type(df.Country.mode())
```

Out[5]:
```
pandas.core.series.Series
```

In [22]:
```
pd.get_dummies(df.Country)
```

Out[22]:

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0
6	0	0	1
7	1	0	0
8	1	0	0
9	1	0	0

In [23]:
```
updated_dataset=pd.concat([pd.get_dummies(df.Country),df.iloc[:,[1,2,3]]],axis=1
updated_dataset
```

Out[23]:

	France	Germany	Spain	Age	Salary	Purchased
0	1	0	0	44.0	72000.0	No
1	0	0	1	27.0	48000.0	Yes
2	0	1	0	30.0	54000.0	No
3	0	0	1	38.0	61000.0	No
4	0	1	0	40.0	63778.0	Yes
5	1	0	0	35.0	58000.0	Yes
6	0	0	1	38.0	52000.0	No
7	1	0	0	48.0	79000.0	Yes
8	1	0	0	50.0	83000.0	No
9	1	0	0	37.0	67000.0	Yes

```
In [24]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        10 non-null     float64
 2   Salary     10 non-null     float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

```
In [25]: updated_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)
         updated_dataset
```

Out[25]:

	France	Germany	Spain	Age	Salary	Purchased
0	1	0	0	44.0	72000.0	0
1	0	0	1	27.0	48000.0	1
2	0	1	0	30.0	54000.0	0
3	0	0	1	38.0	61000.0	0
4	0	1	0	40.0	63778.0	1
5	1	0	0	35.0	58000.0	1
6	0	0	1	38.0	52000.0	0
7	1	0	0	48.0	79000.0	1
8	1	0	0	50.0	83000.0	0
9	1	0	0	37.0	67000.0	1

Exercise-4
Handling Missing and Inappropriate Data in a Dataset

In [1]:

```
#Name: Feminna G
#Roll no: 240701137
#CSE Batch-2

import numpy as np
import pandas as pd
df=pd.read_csv("Hotel_Dataset.csv")
df
```

Out[1]:

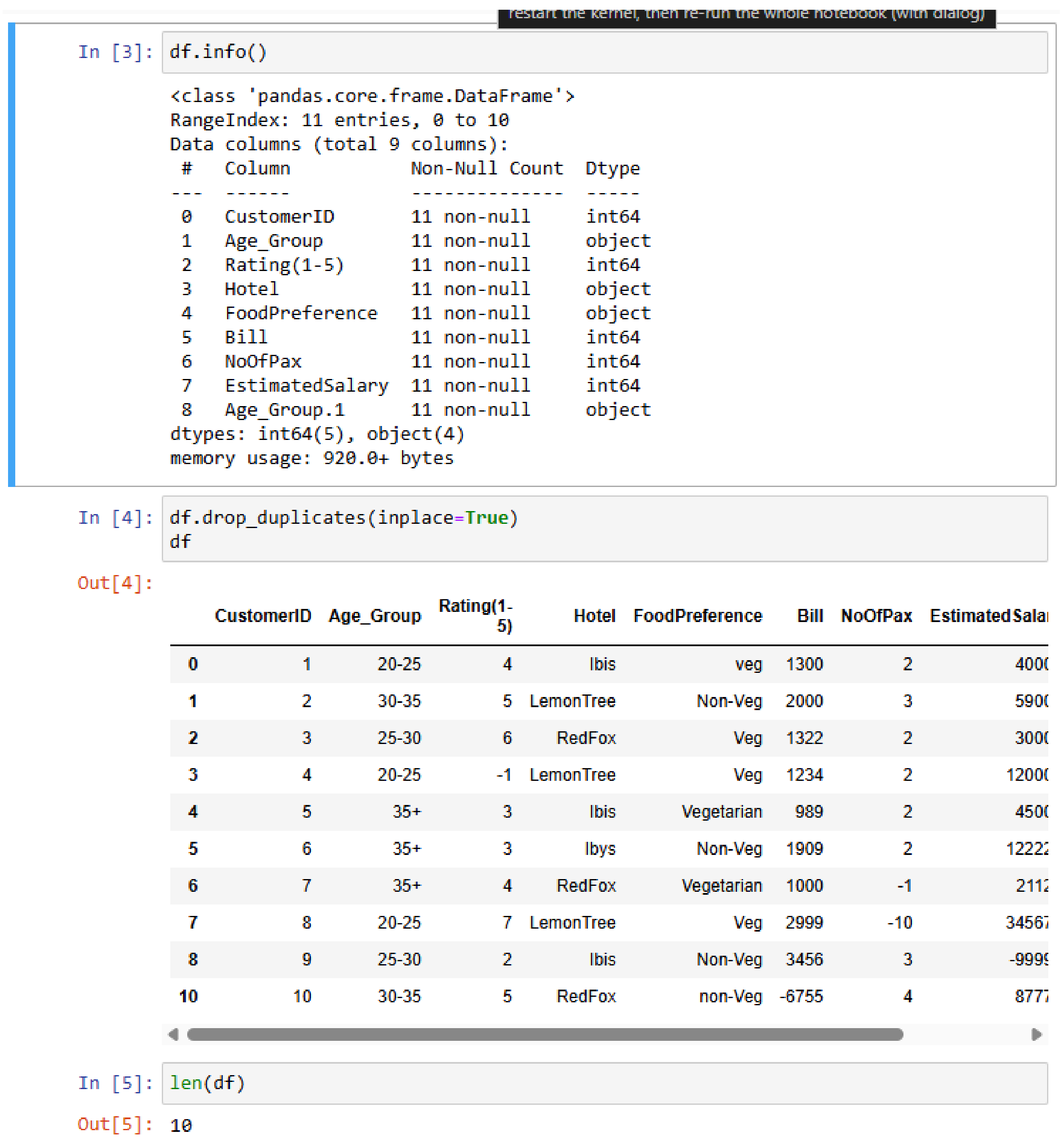
	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSala
0	1	20-25	4	Ibis	veg	1300	2	400
1	2	30-35	5	LemonTree	Non-Veg	2000	3	590
2	3	25-30	6	RedFox	Veg	1322	2	300
3	4	20-25	-1	LemonTree	Veg	1234	2	1200
4	5	35+	3	Ibis	Vegetarian	989	2	450
5	6	35+	3	Ibys	Non-Veg	1909	2	1222
6	7	35+	4	RedFox	Vegetarian	1000	-1	211
7	8	20-25	7	LemonTree	Veg	2999	-10	3456
8	9	25-30	2	Ibis	Non-Veg	3456	3	-999
9	9	25-30	2	Ibis	Non-Veg	3456	3	-999
10	10	30-35	5	RedFox	non-Veg	-6755	4	877

In [2]:

```
df.duplicated()
```

Out[2]:

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9      True
10    False
dtype: bool
```



restart the kernel, then re-run the whole notebook (with dialog)

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   CustomerID       11 non-null     int64 
 1   Age_Group        11 non-null     object
 2   Rating(1-5)      11 non-null     int64 
 3   Hotel            11 non-null     object
 4   FoodPreference   11 non-null     object
 5   Bill             11 non-null     int64 
 6   NoOfPax          11 non-null     int64 
 7   EstimatedSalary  11 non-null     int64 
 8   Age_Group.1      11 non-null     object
dtypes: int64(5), object(4)
memory usage: 920.0+ bytes
```

In [4]:
```
df.drop_duplicates(inplace=True)
df
```

Out[4]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalar
0	1	20-25	4	Ibis	veg	1300	2	4000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	5900
2	3	25-30	6	RedFox	Veg	1322	2	3000
3	4	20-25	-1	LemonTree	Veg	1234	2	12000
4	5	35+	3	Ibis	Vegetarian	989	2	4500
5	6	35+	3	Ibys	Non-Veg	1909	2	12222
6	7	35+	4	RedFox	Vegetarian	1000	-1	2112
7	8	20-25	7	LemonTree	Veg	2999	-10	34567
8	9	25-30	2	Ibis	Non-Veg	3456	3	-9999
10	10	30-35	5	RedFox	non-Veg	-6755	4	8777

In [5]:
```
len(df)
```

Out[5]:
```
10
```

In [6]:
```python
index=np.array(list(range(0,len(df))))
df.set_index(index,inplace=True)
index
```

Out[6]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [7]:
```python
df
```

Out[7]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

In [8]:
```python
df.drop(['Age_Group.1'],axis=1,inplace=True)
df
```

Out[8]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	Ibys	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

In [9]:
```
df.CustomerID.loc[df.CustomerID<0]=np.nan
df.Bill.loc[df.Bill<0]=np.nan
df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
df
```

```
C:\Users\HDC0422092\AppData\Local\Temp\ipykernel_8256\2080958306.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  df.CustomerID.loc[df.CustomerID<0]=np.nan
C:\Users\HDC0422092\AppData\Local\Temp\ipykernel_8256\2080958306.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  df.Bill.loc[df.Bill<0]=np.nan
C:\Users\HDC0422092\AppData\Local\Temp\ipykernel_8256\2080958306.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
```

Out[9]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2	122220.0
6	7.0	35+	4	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4	87777.0

```
In [10]: df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
         df
```

```
C:\Users\HDC0422092\AppData\Local\Temp\ipykernel_8256\2129877948.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

Out[10]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Vegetarian	989.0	2.0	45000.0
5	6.0	35+	3	Ibys	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4	RedFox	Vegetarian	1000.0	NaN	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	NaN	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	NaN
9	10.0	30-35	5	RedFox	non-Veg	NaN	4.0	87777.0

```
In [11]: df.Age_Group.unique()

Out[11]: array(['20-25', '30-35', '25-30', '35+'], dtype=object)

In [12]: df.Hotel.unique()

Out[12]: array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)

In [13]: df.Hotel.replace(['Ibys'],'Ibis',inplace=True)

In [14]: df.FoodPreference.unique

Out[14]: <bound method Series.unique of 0           veg
         1       Non-Veg
         2           Veg
         3           Veg
         4    Vegetarian
         5       Non-Veg
         6    Vegetarian
         7           Veg
         8       Non-Veg
         9       non-Veg
         Name: FoodPreference, dtype: object>
```

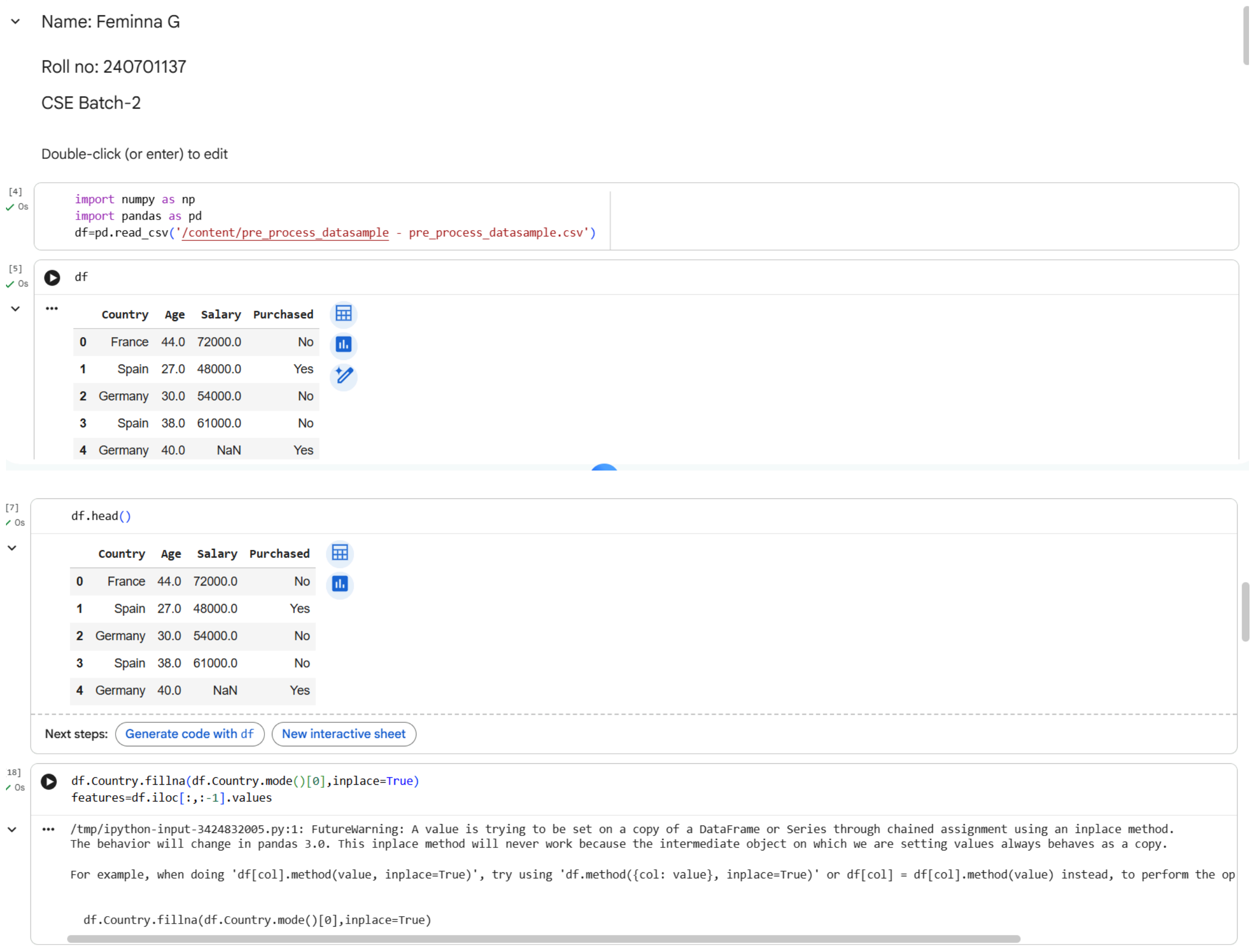
```
In [15]: df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)
         df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)
```

```
In [16]: df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)
         df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)
         df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)
         df.Bill.fillna(round(df.Bill.mean()),inplace=True)
         df
```

Out[16]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4	Ibis	Veg	1300.0	2.0	40000.0
1	2.0	30-35	5	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	6	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	-1	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3	Ibis	Veg	989.0	2.0	45000.0
5	6.0	35+	3	Ibis	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4	RedFox	Veg	1000.0	2.0	21122.0
7	8.0	20-25	7	LemonTree	Veg	2999.0	2.0	345673.0
8	9.0	25-30	2	Ibis	Non-Veg	3456.0	3.0	96755.0
9	10.0	30-35	5	RedFox	Non-Veg	1801.0	4.0	87777.0

Exercise-5
feature scaling



Name: Feminna G

Roll no: 240701137

CSE Batch-2

Double-click (or enter) to edit

[4]
```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/pre_process_datasample - pre_process_datasample.csv')
```

[5]
```
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

[7]
```
df.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

Next steps: Generate code with df New interactive sheet

[8]
```
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
```

```
/tmp/ipython-input-3424832005.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the op


  df.Country.fillna(df.Country.mode()[0],inplace=True)
```

[21] ✓ 0s

```
label=df.iloc[:,-1].values
```

[23] ✓ 2s

```
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
```

[24] ✓ 0s

```
age.fit(features[:,[1]])
```

```
▾ SimpleImputer
SimpleImputer()
```

[25] ✓ 0s

```
Salary.fit(features[:,[2]])
```

```
▾ SimpleImputer
SimpleImputer()
```

[32] ✓ 0s

```
SimpleImputer()
```

```
▾ SimpleImputer
SimpleImputer()
```

[27] ✓ 0s

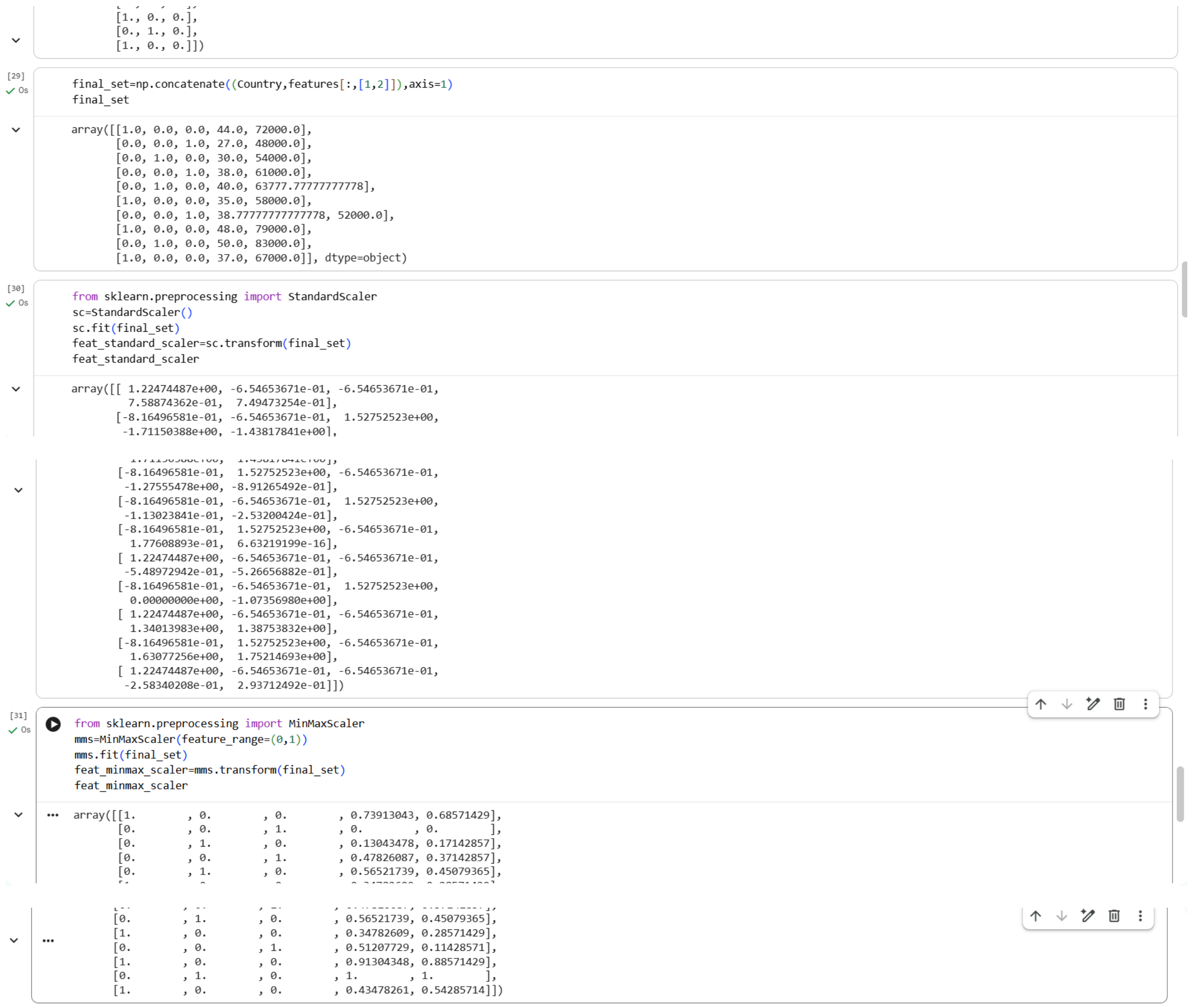
```
features[:,[1]]=age.transform(features[:,[1]])
features[:,[2]]=Salary.transform(features[:,[2]])
features
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

[28] ✓ 0s

```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)
Country=oh.fit_transform(features[:,[0]])
Country
```

```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
```



```
        [1., 0., 0.],
        [0., 1., 0.],
        [1., 0., 0.]])
```

[29]

```
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
final_set
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

[30]

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
feat_standard_scaler=sc.transform(final_set)
feat_standard_scaler
```

```
array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

[31]

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

```
array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
       [0.        , 0.        , 1.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
       [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
       [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
       [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
       [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
       [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
       [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
       [0.        , 1.        , 0.        , 1.        , 1.        ],
       [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

Exercise-10
K-Means Clusterin

In [6]:
```
#Name: Feminn G
#Roll no: 240701137
#CSE Batch-2

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [7]:
```
df=pd.read_csv('Mall_Customers.csv')
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

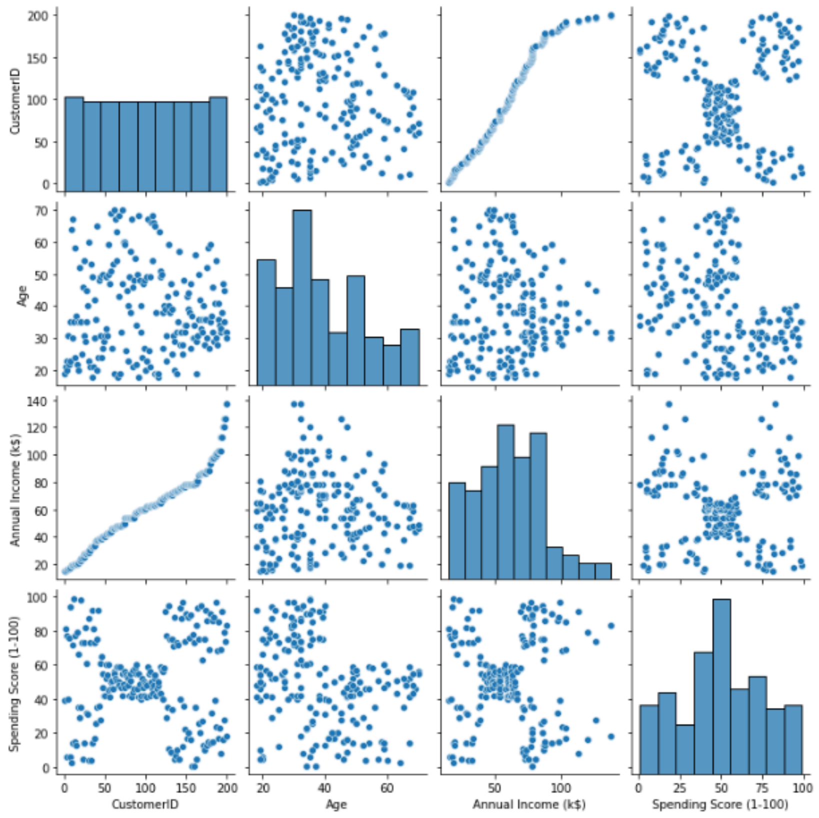
In [9]:
```
df.head()
```

Out[9]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
In [10]: sns.pairplot(df)
```

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x11719dee4c0>
```



```
In [11]: features=df.iloc[:,[3,4]].values
```

```
In [13]: from sklearn.cluster import KMeans
         model=KMeans(n_clusters=5)
         model.fit(features)
         KMeans(n_clusters=5)
```

```
Out[13]: KMeans(n_clusters=5)
```

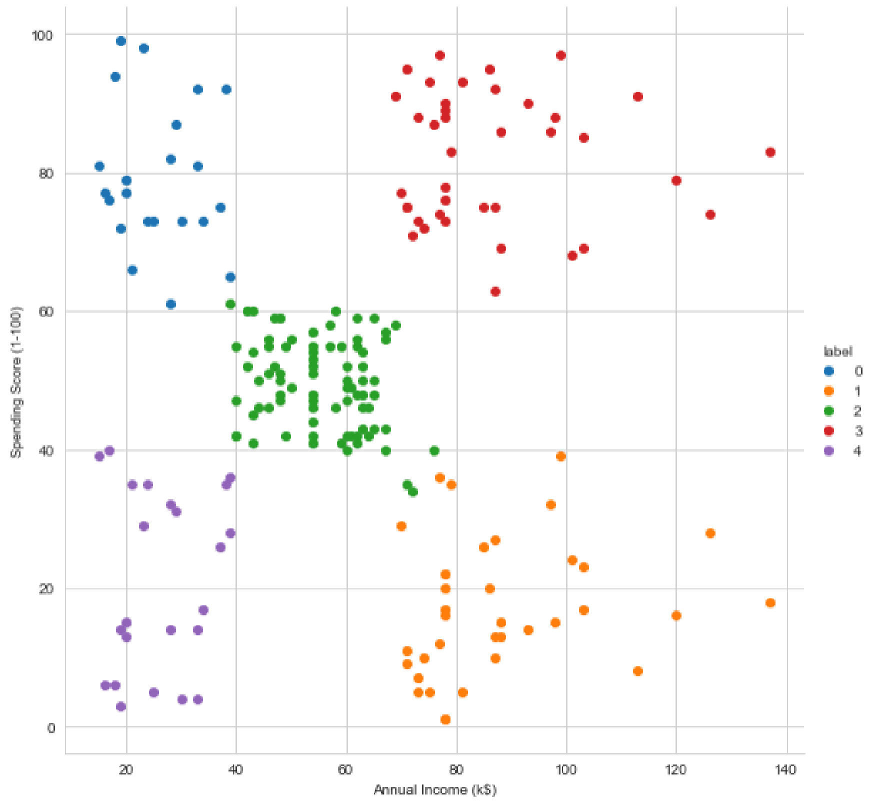
```
In [14]: Final=df.iloc[:,[3,4]]
         Final['label']=model.predict(features)
         Final.head()
```

```
C:\Users\HDC0422092\AppData\Local\Temp\ipykernel_12900\470183701.py:2: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  Final['label']=model.predict(features)
```

Out[14]:

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	4
1	15	81	0
2	16	6	4
3	16	77	0
4	17	40	4

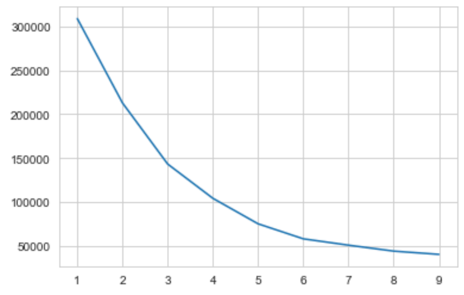
```
In [15]: sns.set_style("whitegrid")
         sns.FacetGrid(Final,hue="label",height=8) \
         .map(plt.scatter,"Annual Income (k$)", "Spending Score (1-100)") \
         .add_legend();
         plt.show()
```



```
In [17]: features_el=df.iloc[:,[2,3,4]].values
         from sklearn.cluster import KMeans
         wcss=[]
         for i in range(1,10):
             model=KMeans(n_clusters=i)
             model.fit(features_el)
             wcss.append(model.inertia_)
         plt.plot(range(1,10),wcss)
```

```
d:\Users\HDC0422092\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:103
6: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the e
nvironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

```
Out[17]: [<matplotlib.lines.Line2D at 0x1171e4855e0>]
```



Exercise-11
KNN

In [1]:
```
#Name: Feminna G
#Roll no: 240701137
#CSE Batch-2

import numpy as np
import pandas as pd
```

In [3]:
```
df=pd.read_csv('Iris.csv')
df.info()
```

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

In [4]:
```
df.variety.value_counts()
```

Out[4]:
```
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

In [5]:
```
df.head()
```

Out[5]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
In [6]: features=df.iloc[:,:-1].values
        label=df.iloc[:,4].values
```

```
In [7]: from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
```

```
In [9]: :test,ytrain,ytest=train_test_split(features,label,test_size=.2, random_state=42)
        IN=KNeighborsClassifier(n_neighbors=5)
        IN.fit(xtrain,ytrain)

Out[9]: KNeighborsClassifier()
```

```
In [10]: print(model_KNN.score(xtrain,ytrain))
         print(model_KNN.score(xtest,ytest))

0.9666666666666667
1.0
```

```
In [11]: from sklearn.metrics import confusion_matrix
         confusion_matrix(label,model_KNN.predict(features))

Out[11]: array([[50,  0,  0],
                [ 0, 47,  3],
                [ 0,  1, 49]], dtype=int64)
```

```
In [12]: from sklearn.metrics import classification_report
         print(classification_report(label,model_KNN.predict(features)))

              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        50
  Versicolor       0.98      0.94      0.96        50
   Virginica       0.94      0.98      0.96        50

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150
```

Exercise-12
T-test

A sample of 10 students scored the following marks in an exam:
[72, 68, 75, 70, 74, 69, 71, 73, 70, 72] We want to test whether the average mark = 70 ($\mu_0 = 70$) at 5% significance level using python

```
In [7]: #Name:Feminna G
        #Roll no: 240701137
        #cse-Batch 2

        import numpy as np
        from scipy import stats
        marks=np.array([72,68,75,70,74,69,71,73,70,72])
        mu_0=70
        t_stat, p_value=stats.ttest_1samp(marks,mu_0)
        print(f"T-statistic: {t_stat:.3f}")
        print(f"P-value: {p_value:.4f}")
        alpha=0.05
        if p_value<alpha:
            print("Reject Null Hypothesis->Mean is singnificantly different from 70.")
        else:
            print("Fail to Reject Null Hypothesis->No significant difference.")
```

```
T-statistic: 1.993
P-value: 0.0774
Fail to Reject Null Hypothesis->No significant difference.
```

Exercise-13
Z-test

A manufacturer claims that the average weight of packets is 50 g.A random sample of 36 packets has an average weight of 51.2 g with a known σ = 3 g.At a 5% significance level, test the claim.

In [10]:
```
#Name:Feminna G
#Roll no: 240701137
#cse-Batch 2

import numpy as np
from math import sqrt
from scipy.stats import norm
x_bar = 51.2
mu_0 = 50
sigma = 3
n = 36
z_stat = (x_bar - mu_0) / (sigma / sqrt(n))
p_value = 2 * (1 - norm.cdf(abs(z_stat)))
print(f"Z-statistic: {z_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis → Mean is significantly different from 50 g.")
else:
    print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
Z-statistic: 2.400
P-value: 0.0164
Reject Null Hypothesis → Mean is significantly different from 50 g.
```

Exercise-14
Anova test

Three fertilizers (A, B, C) were tested on crop yield (in kg). Is there a significant difference among fertilizers? (Use α = 0.05)

Fertilizers	Yields
A	20, 22, 23
B	19, 20, 18
C	25, 27, 26

In [2]:
```
#Name: Feminn G
#Roll no: 240701137
#CSE Batch-2

import numpy as np
from scipy import stats
A = [20, 22, 23]
B = [19, 20, 18]
C = [25, 27, 26]
f_stat, p_value = stats.f_oneway(A, B, C)
print(f"F-statistic: {f_stat:.3f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis → Means are significantly different.")
else:
    print("Fail to Reject Null Hypothesis → No significant difference.")
```

```
F-statistic: 25.923
P-value: 0.0011
Reject Null Hypothesis → Means are significantly different.
```