RAJALAKSHMI ENGINEERING COLLEGE

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

Department of ComputerScience and Engineering

CS23334 Fundamentals of Data Science

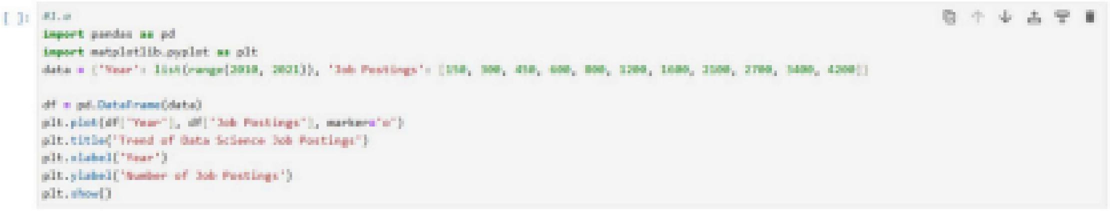
Lab III semesterIIYear (2023R)

Name of the Student: GOWTHAM S

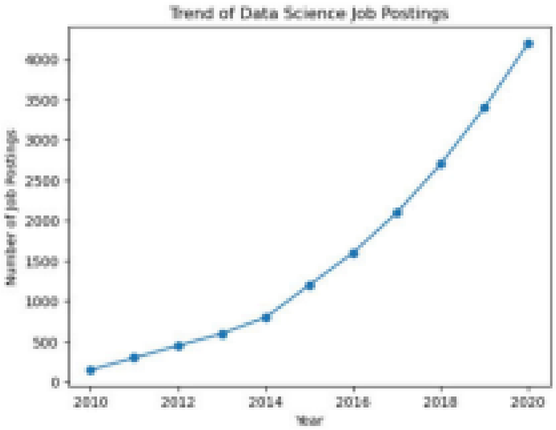
Register Number:2116240701157

Exp No:1. a Analyze the trend of data science job postings over the last decade

Description: Use web scraping (e.g., BeautifulSoup) or APIs (e.g., LinkedIn API) to gather data on the number of data science job postings each year. Use pandas for data manipulation and matplotlib/seaborn for visualization.

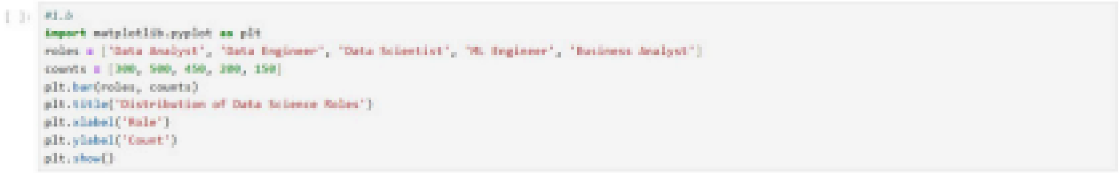


```
[ ]: #1.a
     import pandas as pd
     import matplotlib.pyplot as plt
     data = {'Year': list(range(2010, 2021)), 'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}

     df = pd.DataFrame(data)
     plt.plot(df['Year'], df['Job Postings'], marker='o')
     plt.title('Trend of Data Science Job Postings')
     plt.xlabel('Year')
     plt.ylabel('Number of Job Postings')
     plt.show()
```

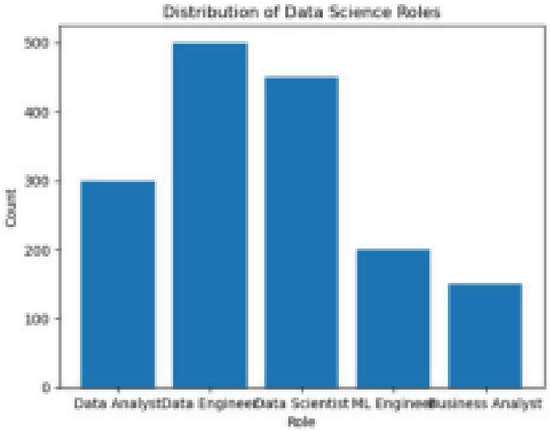


Exp No:1. b Analyze and visualize the distribution of various data science roles (Data Analyst, Data Engineer, Data Scientist, etc.) from a dataset.

Description: Use a dataset of job postings and categorize them into different roles. Visualize the distribution using pie charts or bar plots.

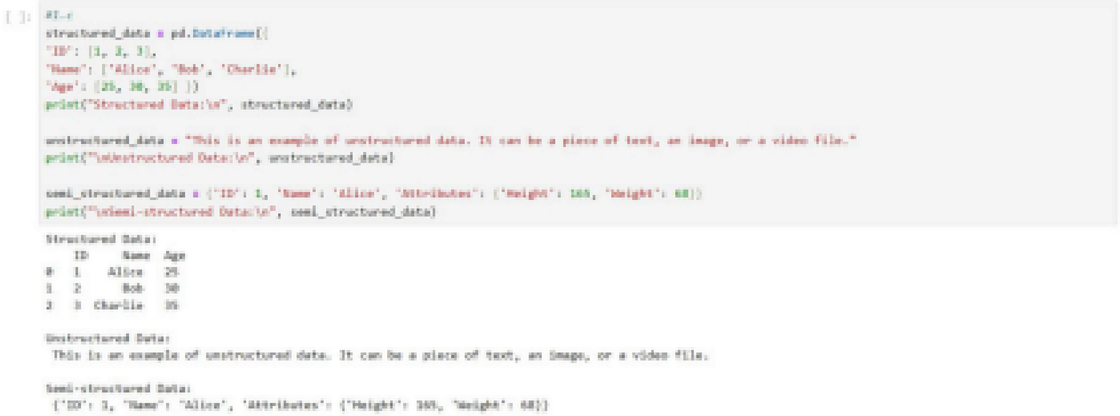


```
[ ]: #1.b
     import matplotlib.pyplot as plt
     roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer', 'Business Analyst']
     counts = [300, 500, 450, 200, 150]
     plt.bar(roles, counts)
     plt.title('Distribution of Data Science Roles')
     plt.xlabel('Role')
     plt.ylabel('Count')
     plt.show()
```



Exp No:1. c Conduct an Experiment to differentiate Structured, Un-structured and Semi structured data based on data sets given.

Description: Create small datasets for each type and Explain their characteristics.



```
#1.c
structured_data = pd.DataFrame({
'ID': [1, 2, 3],
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35] })
print("Structured Data:\n", structured_data)

unstructured_data = "This is an example of unstructured data. It can be a piece of text, an image, or a video file."
print("\nUnstructured Data:\n", unstructured_data)

semi_structured_data = {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
print("\nSemi-structured Data:\n", semi_structured_data)
```

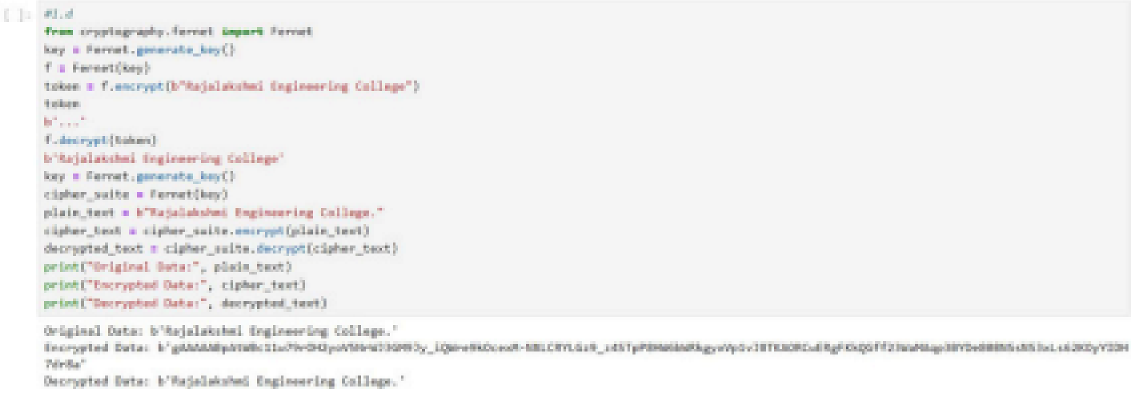
```
Structured Data:
    ID     Name  Age
0   1    Alice   25
1   2      Bob   30
2   3  Charlie   35

Unstructured Data:
 This is an example of unstructured data. It can be a piece of text, an image, or a video file.

Semi-structured Data:
 {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
```

Exp No:1. d

Conduct an Experiment to encrypt and decrypt given sensitive data.

Description:

Use the cryptography library to encrypt and decrypt a piece of data.

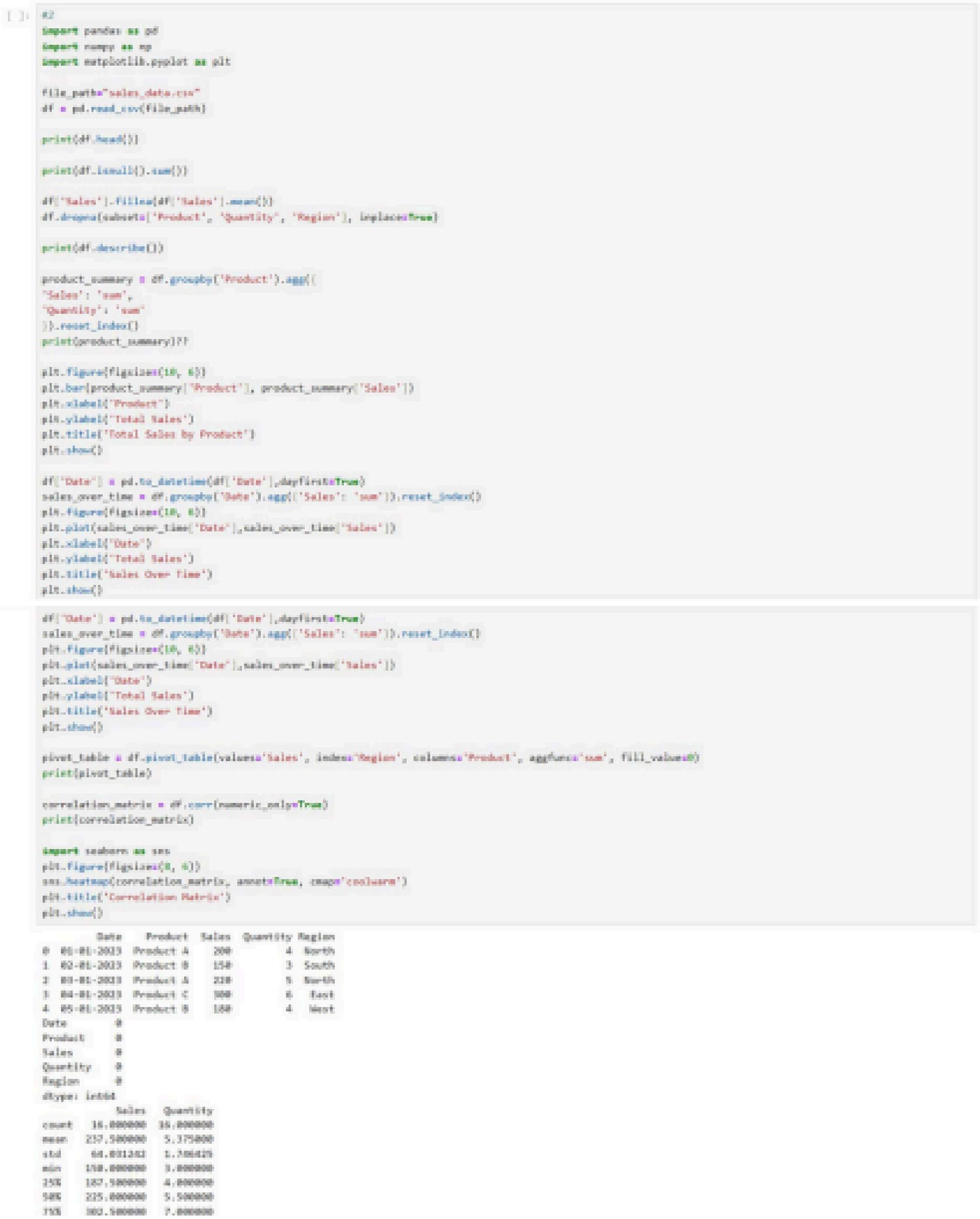


```
#1.d
from cryptography.fernet import Fernet
key = Fernet.generate_key()
f = Fernet(key)
token = f.encrypt(b"Rajalakshmi Engineering College")
token
b'...'
f.decrypt(token)
b'Rajalakshmi Engineering College'
key = Fernet.generate_key()
cipher_suite = Fernet(key)
plain_text = b"Rajalakshmi Engineering College."
cipher_text = cipher_suite.encrypt(plain_text)
decrypted_text = cipher_suite.decrypt(cipher_text)
print("Original Data:", plain_text)
print("Encrypted Data:", cipher_text)
print("Decrypted Data:", decrypted_text)
```

```
Original Data: b'Rajalakshmi Engineering College.'
Encrypted Data: b'gAAAAAB[illegible]
7drBa'
Decrypted Data: b'Rajalakshmi Engineering College.'
```

Exp No:2 Upload and Analyze the data set given in csv format and perform data preprocessing and visualization.

Description: Use sample data set sales-data. csv.

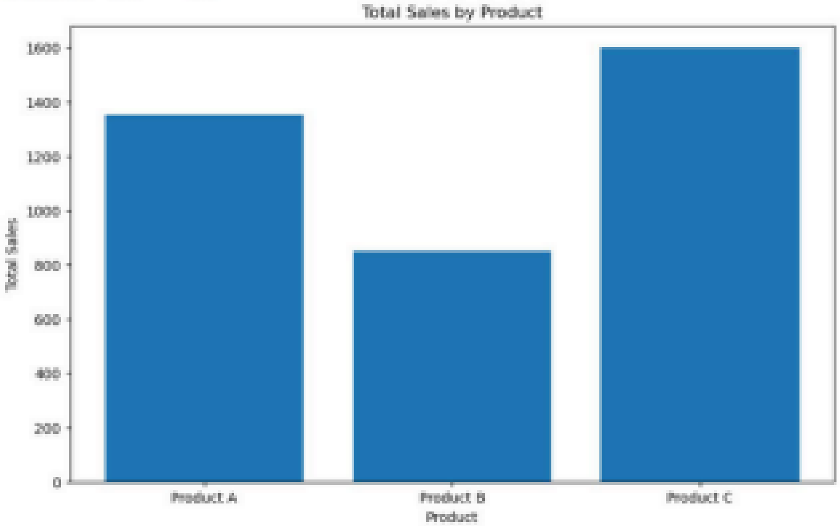


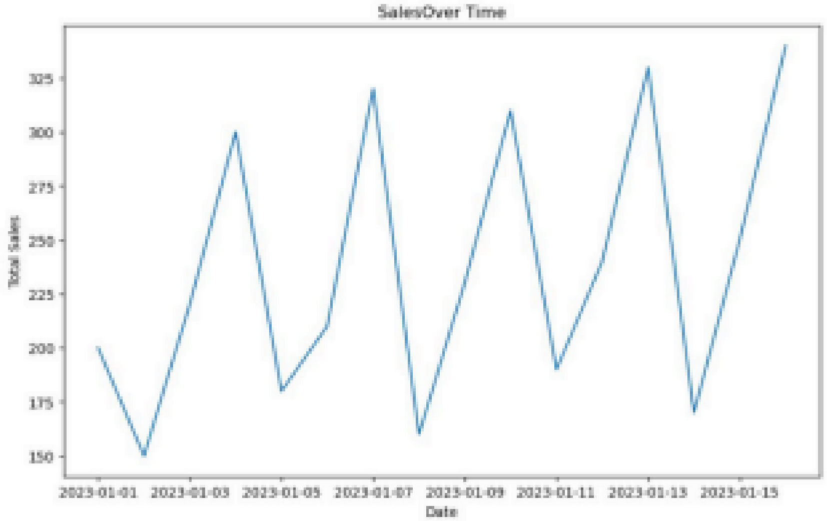
```
[ ]: #2
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     file_path="sales_data.csv"
     df = pd.read_csv(file_path)

     print(df.head())

     print(df.isnull().sum())

     df['Sales'].fillna(df['Sales'].mean())
     df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)

     print(df.describe())

     product_summary = df.groupby('Product').agg({
     'Sales': 'sum',
     'Quantity': 'sum'
     }).reset_index()
     print(product_summary)??

     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()

     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('Sales Over Time')
     plt.show()
```

```
     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('Sales Over Time')
     plt.show()

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

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

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

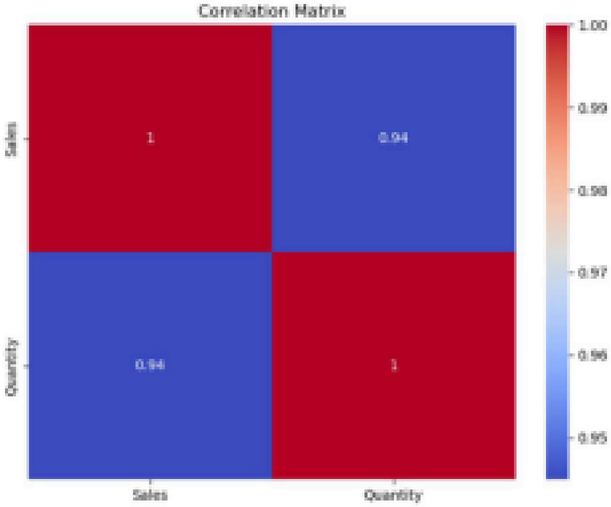
```
         Date    Product  Sales  Quantity Region
0  01-01-2023  Product A    200         4  North
1  02-01-2023  Product B    150         3  South
2  03-01-2023  Product A    220         5  North
3  04-01-2023  Product C    300         6   East
4  05-01-2023  Product B    180         4   West
Date        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
```

```
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
```





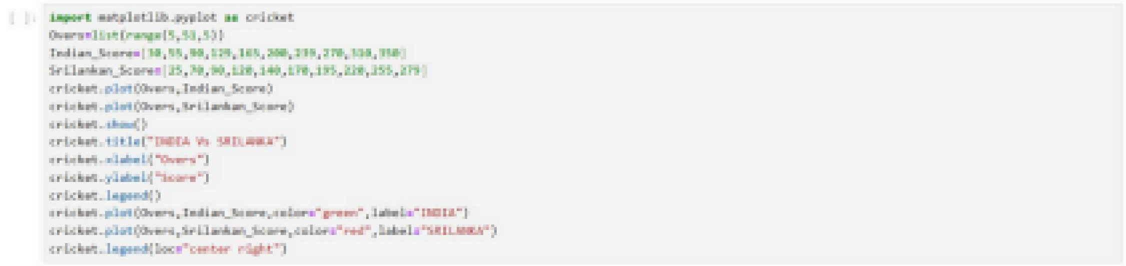
```
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0
             Sales  Quantity
Sales     1.000000  0.944922
Quantity  0.944922  1.000000
```



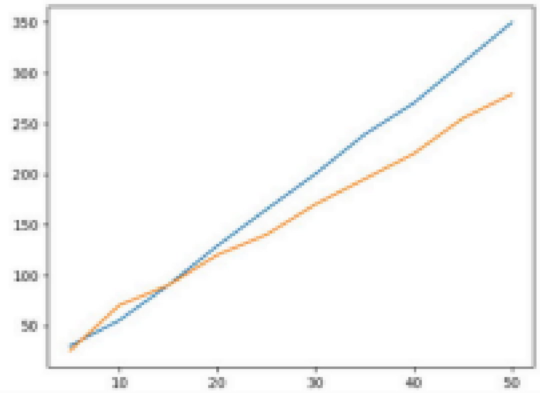
Exp No:3. a

Conduct an Experiment to show data visualization using line plot

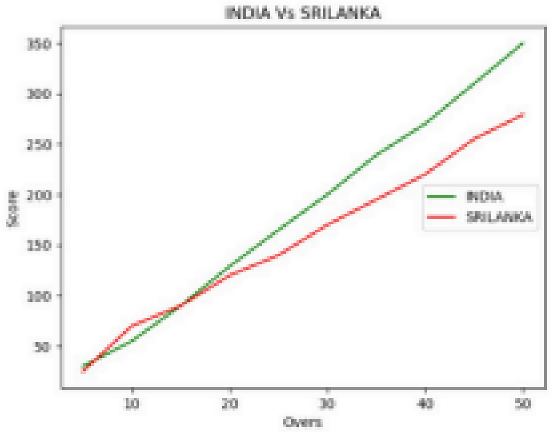
Description: Take any sample data either through csv file or data fetched directly through code.



```
[ ]: import matplotlib.pyplot as cricket
     Overs=list(range(5,51,5))
     Indian_Score=[30,55,90,129,165,200,239,270,310,350]
     Srilankan_Score=[25,70,90,120,140,170,195,220,255,279]
     cricket.plot(Overs,Indian_Score)
     cricket.plot(Overs,Srilankan_Score)
     cricket.show()
     cricket.title("INDIA Vs SRILANKA")
     cricket.xlabel("Overs")
     cricket.ylabel("Score")
     cricket.legend()
     cricket.plot(Overs,Indian_Score,color="green",label="INDIA")
     cricket.plot(Overs,Srilankan_Score,color="red",label="SRILANKA")
     cricket.legend(loc="center right")
```



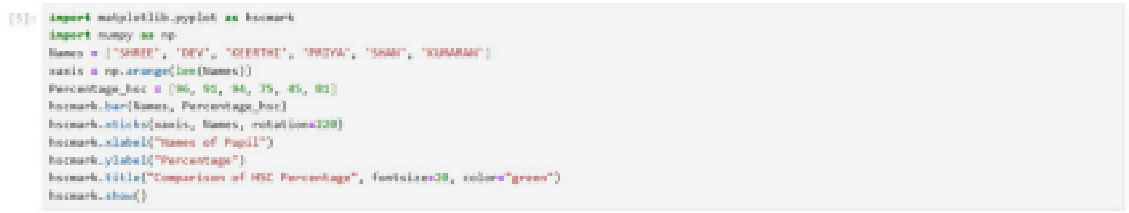
```
[19]: <matplotlib.legend.Legend at [illegible]>
```



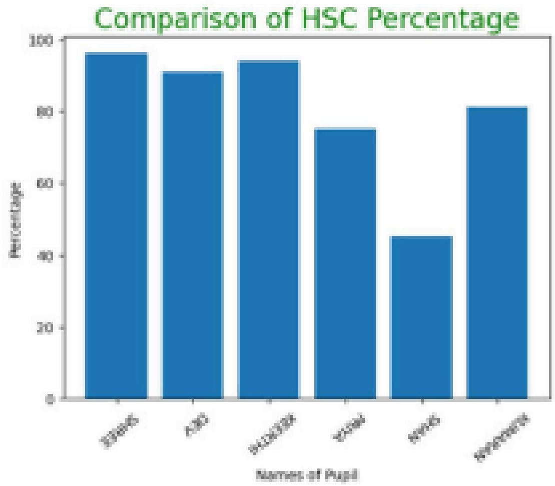
Exp No:3. b

Conduct an Experiment to show data visualization using bar chart.

Description: Take any sample data either through csv file or data fetched directly through code.



```
[5]: import matplotlib.pyplot as hscmark
     import numpy as np
     Names = ["SHREE", 'DEV', 'KEERTHI', 'PRIYA', 'SHAN', 'KUMARAN']
     xaxis = np.arange(len(Names))
     Percentage_hsc = [96, 91, 94, 75, 45, 81]
     hscmark.bar(Names, Percentage_hsc)
     hscmark.xticks(xaxis, Names, rotation=120)
     hscmark.xlabel("Names of Pupil")
     hscmark.ylabel("Percentage")
     hscmark.title("Comparison of HSC Percentage", fontsize=20, color="green")
     hscmark.show()
```

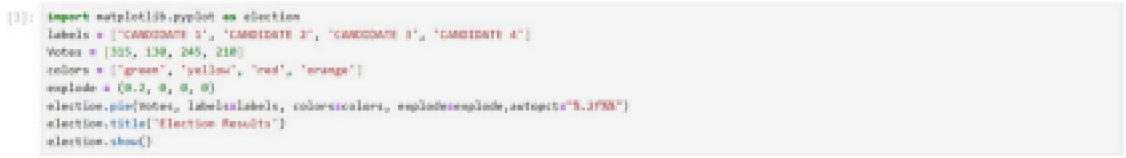


Exp No:3. c

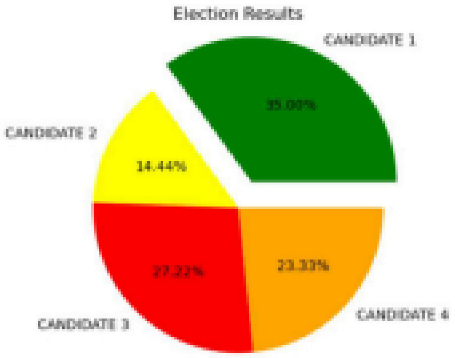
Conduct an Experiment to show data visualization using pie chart.

Description: code.

Take any sample data either through csv file
or data fetched directly through

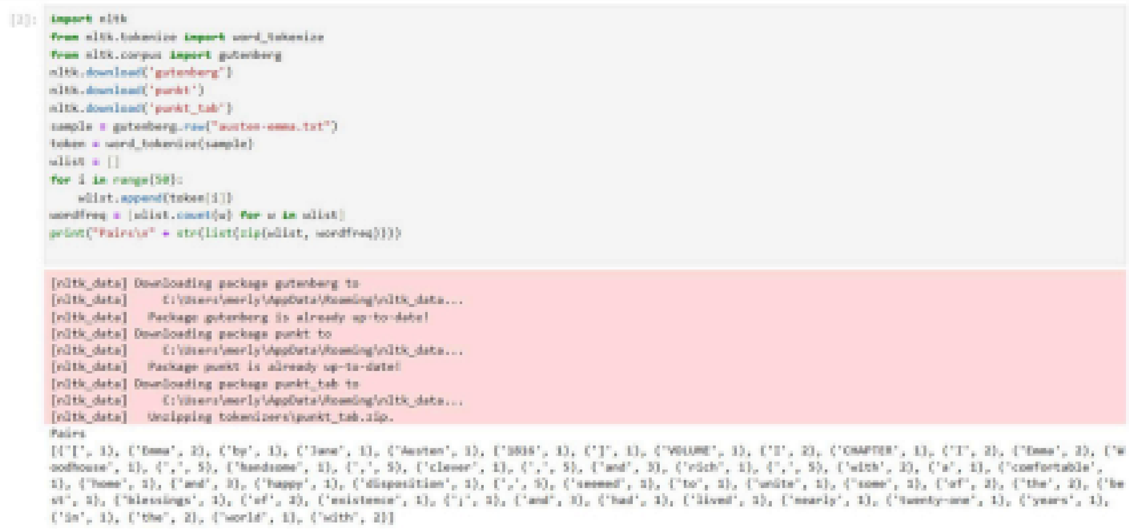


```
[3]: import matplotlib.pyplot as election
     labels = ['CANDIDATE 1', 'CANDIDATE 2', 'CANDIDATE 3', 'CANDIDATE 4']
     Votes = [315, 130, 245, 210]
     colors = ['green', 'yellow', 'red', 'orange']
     explode = (0.2, 0, 0, 0)
     election.pie(Votes, labels=labels, colors=colors, explode=explode,autopct="%.2f%%")
     election.title('Election Results')
     election.show()
```



Exp No:4 To Count the frequency of occurrence of a word in a body of text is often needed during text processing.

Description: Import the word_tokenize function and gutenberg.



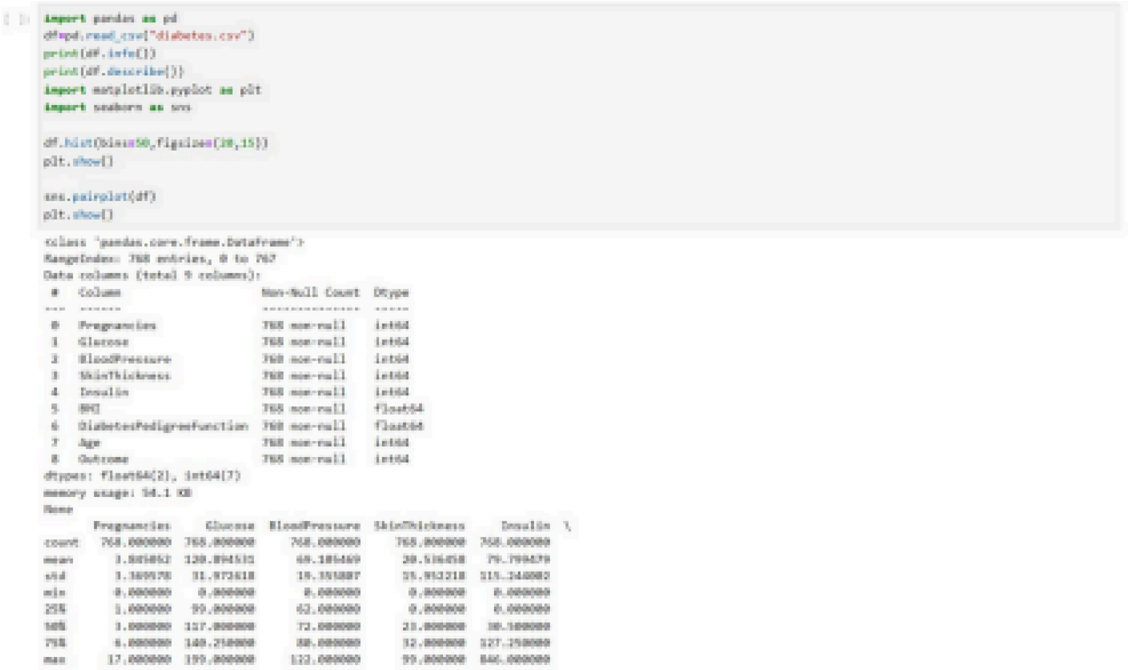
```
[2]: import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg
nltk.download('gutenberg')
nltk.download('punkt')
nltk.download('punkt_tab')
sample = gutenberg.raw("austen-emma.txt")
token = word_tokenize(sample)
wlist = []
for i in range(50):
    wlist.append(token[i])
wordfreq = [wlist.count(w) for w in wlist]
print("Pairs\n" + str(list(zip(wlist, wordfreq))))
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\merly\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\merly\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\merly\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
Pairs
[('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1), (']', 1), ('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('W
oodhouse', 1), (',', 5), ('handsome', 1), (',', 5), ('clever', 1), (',', 5), ('and', 3), ('rich', 1), (',', 5), ('with', 2), ('a', 1), ('comfortable',
1), ('home', 1), ('and', 3), ('happy', 1), ('disposition', 1), (',', 5), ('seemed', 1), ('to', 1), ('unite', 1), ('some', 1), ('of', 2), ('the', 2), ('be
st', 1), ('blessings', 1), ('of', 2), ('existence', 1), (';', 1), ('and', 3), ('had', 1), ('lived', 1), ('nearly', 1), ('twenty-one', 1), ('years', 1),
('in', 1), ('the', 2), ('world', 1), ('with', 2)]
```

Exp No:5 Initial Exploration
Data Collection and

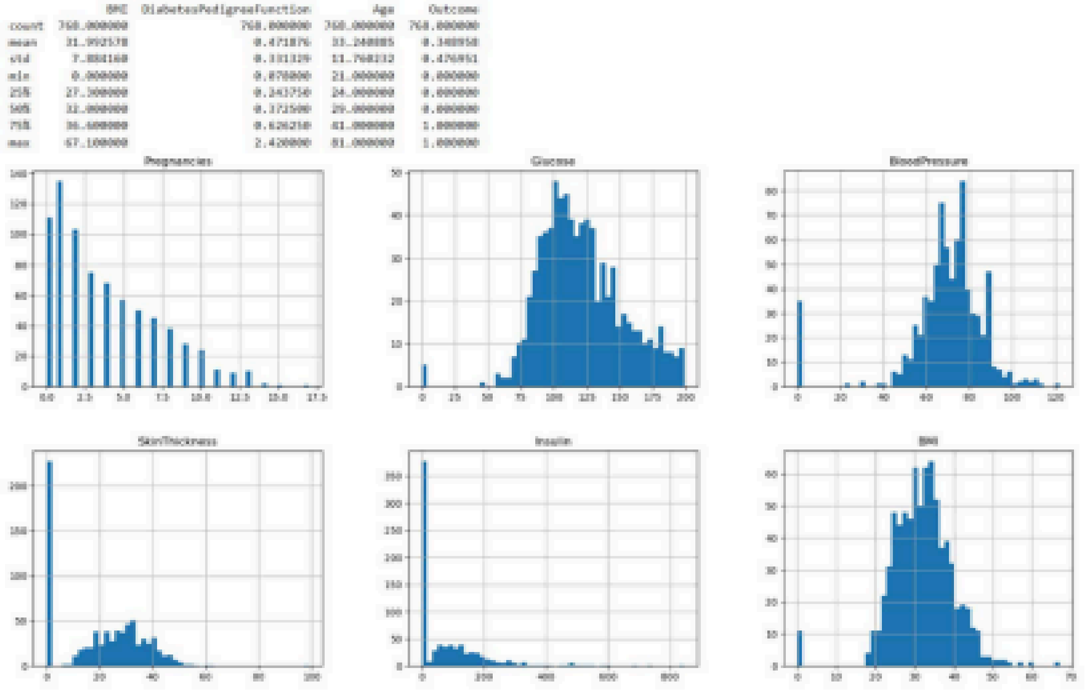
Exploration of the diabetes dataset.

Objective:
To collect, load, and perform initial

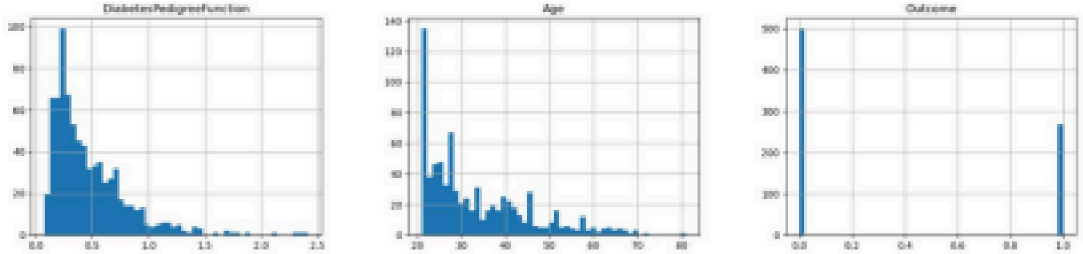


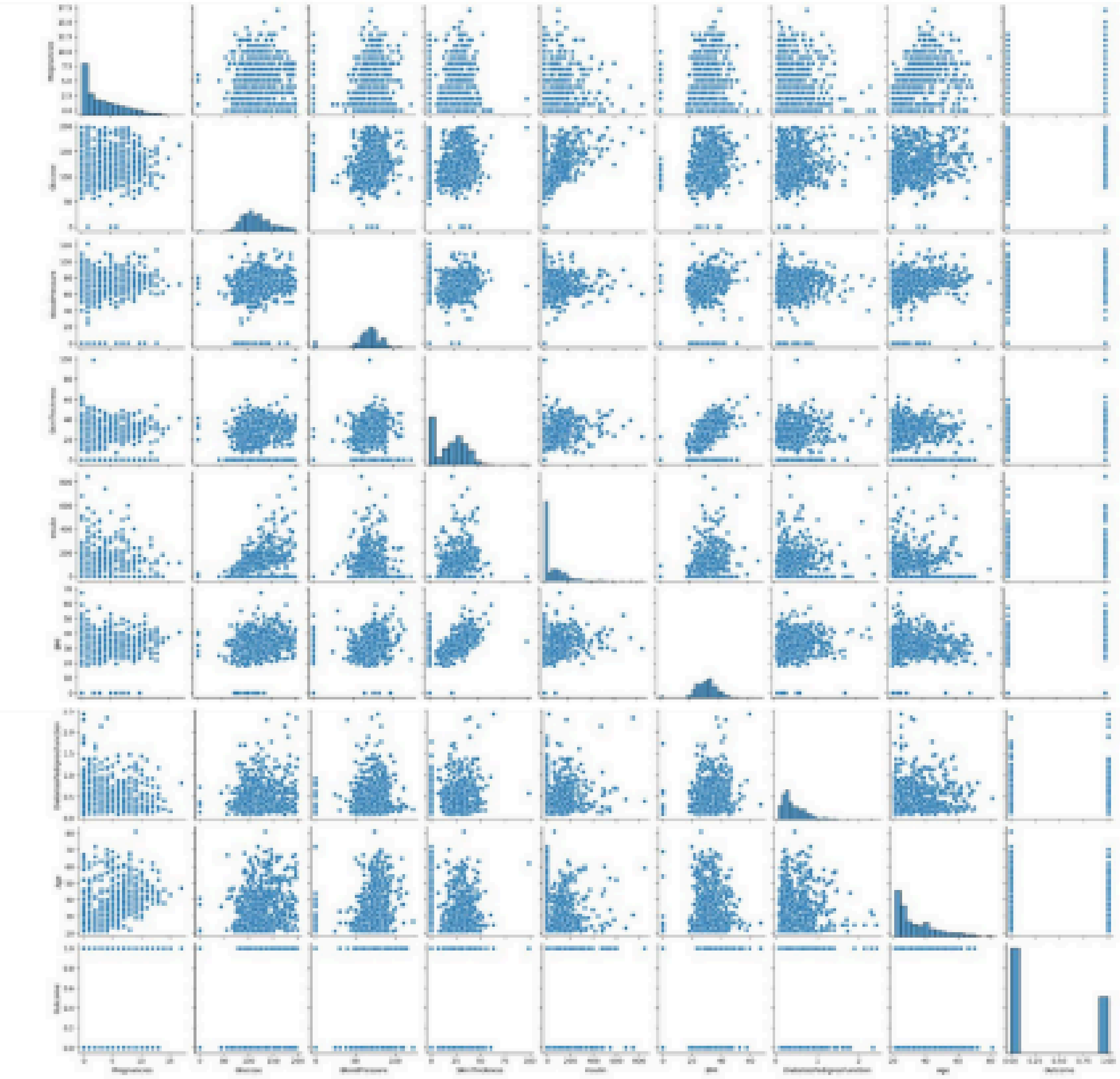
```
import pandas as pd
df=pd.read_csv("diabetes.csv")
print(df.info())
print(df.describe())
import matplotlib.pyplot as plt
import seaborn as sns

df.hist(bins=50,figsize=(20,15))
plt.show()

sns.pairplot(df)
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000   
mean      3.845052  120.894531      69.105469      20.536458   79.799479   
std       3.369578   31.972618      19.355807      15.952218  115.244002   
min       0.000000    0.000000       0.000000       0.000000    0.000000   
25%       1.000000   99.000000      62.000000       0.000000    0.000000   
50%       3.000000  117.000000      72.000000      23.000000   30.500000   
75%       6.000000  140.250000      80.000000      32.000000  127.250000   
max      17.000000  199.000000     122.000000      99.000000  846.000000   
```



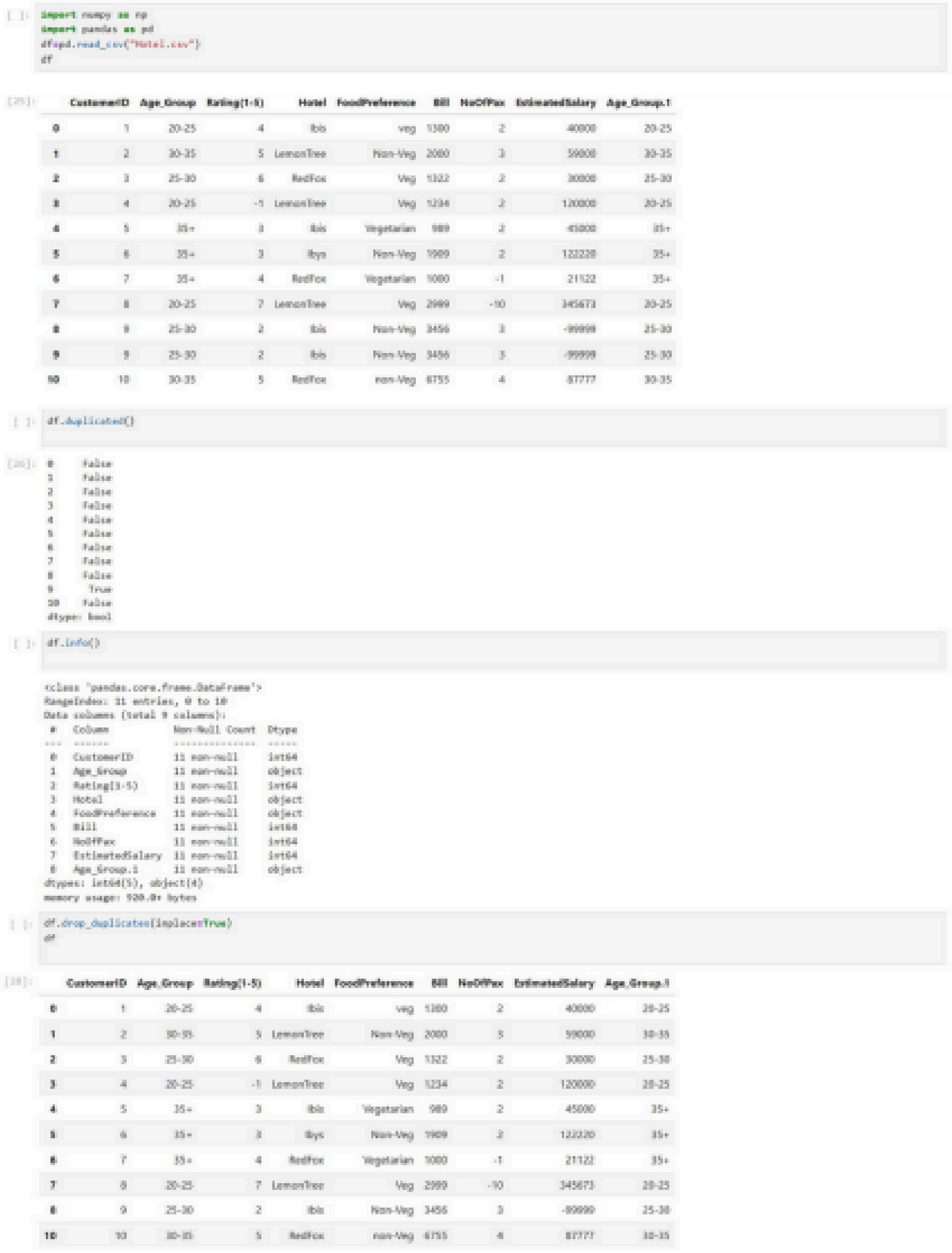
```
              BMI  DiabetesPedigreeFunction         Age     Outcome  
count  768.000000                768.000000  768.000000  768.000000  
mean    31.992578                  0.471876   33.240885    0.348958  
std      7.884160                  0.331329   11.760232    0.476951  
min      0.000000                  0.078000   21.000000    0.000000  
25%     27.300000                  0.243750   24.000000    0.000000  
50%     32.000000                  0.372500   29.000000    0.000000  
75%     36.600000                  0.626250   41.000000    1.000000  
max     67.100000                  2.420000   81.000000    1.000000  
```





Exp:6 Handling Missing and Inappropriate Data in a Dataset

Aim: Demonstrate an Experiment to handle missing data and inappropriate data in a Data set using Python Pandas Library for Data Preprocessing.



[ ]:

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

[25]:

	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	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	6755	4	87777	30-35

[ ]:

```
df.duplicated()
```

[26]:

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

[ ]:

```
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
```

[ ]:

```
df.drop_duplicates(inplace=True)
df
```

[28]:

	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
10	10	30-35	5	RedFox	non-Veg	6755	4	87777	30-35

Exp:7 Experiment to detect outliers in a given data set.

Description:

Understand the procedure to identify the outliers in a given dataset

Exp:8. a Experiment to understand feature scaling.	Understand the importance of feature scaling

Description:

Exp:8. b
Experiment to understand the data preprocessing in Data science

Understand the importance of Data preprocessing in data science

Description:

Exp No:9
Experiment to understand EDA-Quantitative and Qualitative analysis.

Understand the importance of EDA-Quantitative and Qualitative analysis.

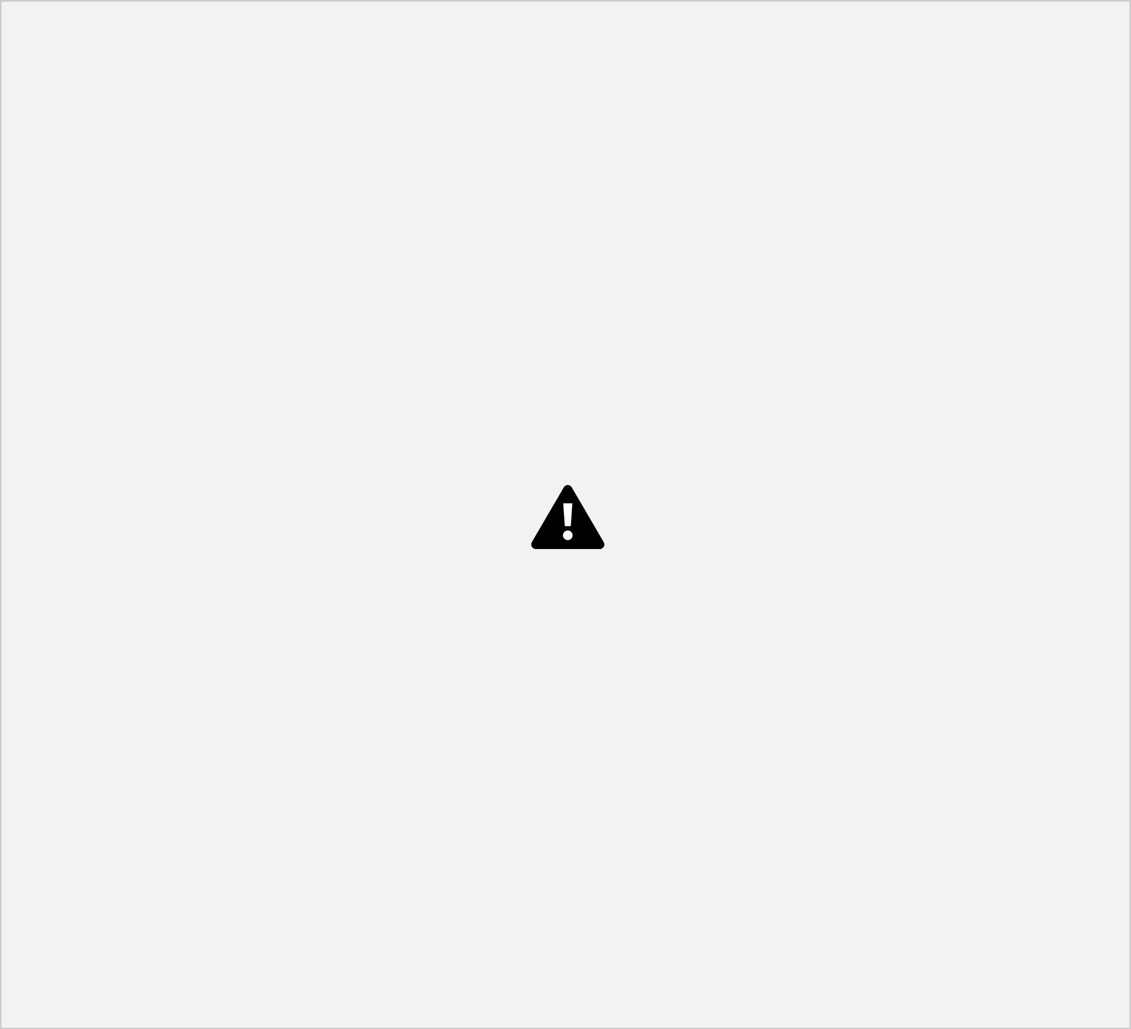
Description:

Exp:10 Regression

Exp:11 Logistic Regression

Exp:13. aKNN

Exp:13. bK-Means



Exp:14 Testing