Experiment 1: Write a program to import CSV file data and read it using python in data Wrangling.

Aim: Write a program in python to import CSV file data and apply some function.

Procedure:

Step1: import CSV file data for any website.

Step2: import pandas as pd for create a dataframe.

Step3: Use function .read_csv() for read a CSV file.

Step4: Print the CSV file data in any text editor.

Program: import csv with open

```
('hrdata.csv','r') as csvfile:

    reader = csv.reader(csvfile)


    for row in reader:

        print (row[0],end = '\t\t')

print (row[1],end = '\t\t')

print (row[2],end = '\t\t')

print (row[3],end = '\n')


import pandas as pd

dataset1 = pd.read_csv('hrdata.csv')

columns1 = list(dataset1.columns)


dataset2 = pd.read_csv('hrdata.csv',header = 0, names =
['Name','HiringDate','wage','paid_leaves_remaining','sick_leaves_remaining','address','working_h
ours
_left','performance_Score'])

columns2 = list(dataset2.columns)
```

Output:

```
Name of Employee                Hire Date               Salary          Paid leaves Remaining
Blake           03/15/14                50000.00                10
Clark           05/12/14                45000.00                10
Jones           11/01/13                70000.00                3
Martin          08/12/14                48000.00                7
Allen           05/23/13                66000.00                8
Ward            01/01/10                50000.00                5
Ford            05/06/09                75000.00                7
Scott           12/25/10                65000.00                6
Adams           07/26/08                100000.00               10
Miller          01/16/08                80000.00                2
```

Result:

Thus the program in python for import CSV file are executed successfully.

Experiment 2: Write a Program in python for data exploration.

Aim: Write a program in python for data exploration and using some function for checking null value.

Procedure:

Step1 : Import CSV file form any website.

Step2 : Using pandas for create a DataFrame.

Step3 : Using python function .head() for find starting five record.

Step4 : Using .describe() function in python for summary.

Step5 : Using .isnull() for finding any null value in datasets.

Program: import pandas as

pd df =

pd.read_csv('hrdata.csv')

// this is use for print starting five

record df2 = df.head() print(df2)

Output :

```
  Name of Employee Hire Date   Salary  Paid leaves Remaining  \
0            Blake  03/15/14  50000.0                     10
1            Clark  05/12/14  45000.0                     10
2            Jones  11/01/13  70000.0                      3
3           Martin  08/12/14  48000.0                      7
4            Allen  05/23/13  66000.0                      8

   Sick Leaves Remaining                        Address of Employee  \
0                      6  4150 Sydney Place Washington DC 20521-4150
1                      3    3290 Hermosillo Place Newyork 35201-0288
2                     10      799 E DRAGRAM SUITE 5A TUCSON AZ 85705
3                      7         300 BOYLSTON AVE E SEATTLE WA 98102
4                      9   11080 CIRCLE POINT RD STE 180 WESTMINSTER

   Due Working Hours  Performance Score
0                 70                  9
1                 55                  8
2                 78                  9
3                 60                  6
4                 40                  4
```

// this function is use for find any null value in our

datasets df3 = df.isnull() print(df3) **Output:**

```
   Name of Employee  Hire Date  Salary  Paid leaves Remaining  \
0             False      False   False                  False
1             False      False   False                  False
2             False      False   False                  False
3             False      False   False                  False
4             False      False   False                  False
5             False      False   False                  False
6             False      False   False                  False
7             False      False   False                  False
8             False      False   False                  False
9             False      False   False                  False

   Sick Leaves Remaining  Address of Employee  Due Working Hours  \
0                  False                False              False
1                  False                False              False
2                  False                False              False
3                  False                False              False
4                  False                False              False
5                  False                False              False
6                  False                False              False
7                  False                False              False
8                  False                False              False
9                  False                False              False
```

```
   Performance Score
0              False
1              False
2              False
3              False
4              False
5              False
6              False
7              False
8              False
9              False
```

df4 = df.isnull().sum() print(df4) **Output :**

```
Name of Employee          0
Hire Date                 0
Salary                    0
Paid leaves Remaining     0
Sick Leaves Remaining     0
Address of Employee       0
Due Working Hours         0
Performance Score         0
dtype: int64
```

df5 = df.isnull.sum().sum()

print(df5)

```
0
```

// this function is use for summary in

datasets df6 = df.describe() print(df6)

Output:

```
              Salary  Paid leaves Remaining  Sick Leaves Remaining  \
count      10.000000              10.000000              10.00000
mean    64900.000000               6.800000               4.90000
std     17368.234094               2.859681               3.28126
min     45000.000000               2.000000               0.00000
25%     50000.000000               5.250000               3.00000
50%     65500.000000               7.000000               5.00000
75%     73750.000000               9.500000               6.75000
max    100000.000000              10.000000              10.00000

       Due Working Hours  Performance Score
count          10.000000          10.000000
mean           72.800000           7.300000
std            17.750117           2.626785
min            40.000000           2.000000
25%            62.500000           6.250000
50%            74.000000           8.000000
75%            83.750000           9.000000
max           100.000000          10.000000
```

Result:

Thus the program in python for data exploration runs successfully.

Experiment 3: How to deal with missing values of data in python.

Aim: Write a program and using some function to deal with the missing value in datasets.

Procedure:

Step1 : Import CSV file for any website.

Step2 : Using numpy for replace value.

Step3 : Using pandas for create a file into DataFrame.

Step4 : Using isnull() function for finding a null value.

Step5 : Using some functions in python for replace and drop missing value.

Program

```
import pandas as pd

import numpy as np

df =

pd.read_csv('cricket_batting.csv')

df2 = df.isnull().sum() df2
```

Output:

	0
Winner	3
Margin	0
Target	1
Overs	2
Opposition	1
Ground	1
Match Date	0
Scorecard	0

dtype: int64

```
df3 = df.fillna(value = 1)

df3
```

Output:

	Winner	Margin	Target	Overs	Opposition	Ground	Match Date	Scorecard
0	India	10 wickets	121.0	29.5	v East Africa	Leeds	11-06-1975	ODI # 24
1	1	10 wickets	113.0	29.0	v India	Melbourne	10-01-1981	ODI # 105
2	West Indies	10 wickets	1.0	45.1	v Zimbabwe	Birmingham	20-06-1983	ODI # 220
3	1	10 wickets	97.0	1.0	v Sri Lanka	Sharjah	08-04-1984	ODI # 260
4	West Indies	10 wickets	117.0	24.2	v New Zealand	1	17-04-1985	ODI # 327
5	Pakistan	10 wickets	65.0	1.0	v New Zealand	Sharjah	15-04-1986	ODI # 384
6	West Indies	10 wickets	192.0	39.2	v New Zealand	Christchurch	28-03-1987	ODI # 441
7	West Indies	10 wickets	221.0	46.5	v Pakistan	Melbourne	23-02-1992	ODI # 717
8	West Indies	10 wickets	153.0	25.5	v South Africa	Port of Spain	11-04-1992	ODI # 754
9	India	10 wickets	113.0	23.1	v West Indies	Port of Spain	27-04-1997	ODI # 1201
10	West Indies	10 wickets	200.0	44.4	v India	Bridgetown	03-05-1997	ODI # 1203
11	India	10 wickets	197.0	30.0	v Zimbabwe	Sharjah	13-11-1998	ODI # 1374
12	1	10 wickets	165.0	29.2	v India	Sharjah	22-03-2000	ODI # 1577

df4 = df.dropna()

df4

Output:

	Winner	Margin	Target	Overs	Opposition	Ground	Match Date	Scorecard
0	India	10 wickets	121.0	29.5	v East Africa	Leeds	11-06-1975	ODI # 24
6	West Indies	10 wickets	192.0	39.2	v New Zealand	Christchurch	28-03-1987	ODI # 441
7	West Indies	10 wickets	221.0	46.5	v Pakistan	Melbourne	23-02-1992	ODI # 717
8	West Indies	10 wickets	153.0	25.5	v South Africa	Port of Spain	11-04-1992	ODI # 754
9	India	10 wickets	113.0	23.1	v West Indies	Port of Spain	27-04-1997	ODI # 1201
10	West Indies	10 wickets	200.0	44.4	v India	Bridgetown	03-05-1997	ODI # 1203
11	India	10 wickets	197.0	30.0	v Zimbabwe	Sharjah	13-11-1998	ODI # 1374

df5 = df.replace(to_replace = np.nan ,value

=12) df5

Output:

	Winner	Margin	Target	Overs	Opposition	Ground	Match Date	Scorecard
0	India	10 wickets	121.0	29.5	v East Africa	Leeds	11-06-1975	ODI # 24
1	12	10 wickets	113.0	29.0	v India	Melbourne	10-01-1981	ODI # 105
2	West Indies	10 wickets	12.0	45.1	v Zimbabwe	Birmingham	20-06-1983	ODI # 220
3	12	10 wickets	97.0	12.0	v Sri Lanka	Sharjah	08-04-1984	ODI # 260
4	West Indies	10 wickets	117.0	24.2	v New Zealand	12	17-04-1985	ODI # 327
5	Pakistan	10 wickets	65.0	12.0	v New Zealand	Sharjah	15-04-1986	ODI # 384
6	West Indies	10 wickets	192.0	39.2	v New Zealand	Christchurch	28-03-1987	ODI # 441
7	West Indies	10 wickets	221.0	46.5	v Pakistan	Melbourne	23-02-1992	ODI # 717
8	West Indies	10 wickets	153.0	25.5	v South Africa	Port of Spain	11-04-1992	ODI # 754
9	India	10 wickets	113.0	23.1	v West Indies	Port of Spain	27-04-1997	ODI # 1201
10	West Indies	10 wickets	200.0	44.4	v India	Bridgetown	03-05-1997	ODI # 1203
11	India	10 wickets	197.0	30.0	v Zimbabwe	Sharjah	13-11-1998	ODI # 1374

Result:

Thus the program to deal with the missing value runs successfully.

Experiment 4 : Perform data replacement, data filtering in python.

Aim: Write a program in python to filtering in python.

Procedure:

Step1 : Import CSV file for any website.

Step2 : Using pandas for create a file into DataFrame.

Step3 : Using method to filter data in datasets

Step4 : Print the datasets.

Program:

```
import pandas as pd df =

pd.read_csv('hrdata.csv')

df2 =

df[df['Salary']>50000]

print(df2)
```

```
   Name of Employee Hire Date    Salary  Paid leaves Remaining  \
2             Jones  11/01/13   70000.0                      3
4             Allen  05/23/13   66000.0                      8
6              Ford  05/06/09   75000.0                      7
7             Scott  12/25/10   65000.0                      6
8             Adams  07/26/08  100000.0                     10
9            Miller  01/16/08   80000.0                      2

   Sick Leaves Remaining                        Address of Employee  \
2                     10     799 E DRAGRAM SUITE 5A TUCSON AZ 85705
4                      9  11080 CIRCLE POINT RD STE 180 WESTMINSTER
6                      0    4343 E HIGHWAY 30 KIMBALL NE 69145 2076
7                      4                       Minnetonka MN  55305
8                      3       5420 E 108TH ST TULSA OK  74137 7219
9                      6                          ST JACOB IL 62281

   Due Working Hours  Performance Score
2                 78                  9
4                 40                  4
6                 80                 10
7                 85                  8
8                 90                  7
9                100                  2
```

```
df3 = df[df['Paid leaves Remaining']>4]

print(df3)
```

```
  Name of Employee Hire Date    Salary  Paid leaves Remaining  \
0            Blake  03/15/14   50000.0                     10
1            Clark  05/12/14   45000.0                     10
3           Martin  08/12/14   48000.0                      7
4            Allen  05/23/13   66000.0                      8
5             Ward  01/01/10   50000.0                      5
6             Ford  05/06/09   75000.0                      7
7            Scott  12/25/10   65000.0                      6
8            Adams  07/26/08  100000.0                     10

   Sick Leaves Remaining                              Address of Employee  \
0                      6         4150 Sydney Place Washington DC 20521-4150
1                      3           3290 Hermosillo Place Newyork 35201-0288
3                      7                300 BOYLSTON AVE E SEATTLE WA 98102
4                      9          11080 CIRCLE POINT RD STE 180 WESTMINSTER
5                      1  112 WEST 17TH ST PO BOX 7  FALLS CITY NE  6835...
6                      0            4343 E HIGHWAY 30 KIMBALL NE 69145 2076
7                      4                                Minnetonka MN  55305
8                      3               5420 E 108TH ST TULSA OK  74137 7219

   Due Working Hours  Performance Score
0                 70                  9
1                 55                  8
3                 60                  6
4                 40                  4
5                 70                 10
6                 80                 10
7                 85                  8
8                 90                  7
```

Result:

Thus the program data filtering in python runs successfully.

Experiment 5 : Write a program for removing duplicates from data.

Aim: Write a program in python for removing duplicates from data using some functions.

Procedure:

Step1 : Import CSV file for any website.

Step2 : Using pandas for create a file into DataFrame.

Step3 : Using function to remove duplicate value in python.

Step4 : Print the datasets.

Program:

```
import pandas as pd df =

pd.read_csv('hrdata.csv')

df2 = df.drop_duplicates()

df2
```

	Name of Employee	Hire Date	Salary	Paid leaves Remaining	Sick Leaves Remaining	Address of Employee	Due Working Hours	Performance Score
0	Blake	03/15/14	50000.0	10	6	4150 Sydney Place Washington DC 20521-4150	70	9
1	Clark	05/12/14	45000.0	10	3	3290 Hermosillo Place Newyork 35201-0288	55	8
2	Jones	11/01/13	70000.0	3	10	799 E DRAGRAM SUITE 5A TUCSON AZ 85705	78	9
3	Martin	08/12/14	48000.0	7	7	300 BOYLSTON AVE E SEATTLE WA 98102	60	6
4	Allen	05/23/13	66000.0	8	9	11080 CIRCLE POINT RD STE 180 WESTMINSTER	40	4
5	Ward	01/01/10	50000.0	5	1	112 WEST 17TH ST PO BOX 7 FALLS CITY NE 6835...	70	10
6	Ford	05/06/09	75000.0	7	0	4343 E HIGHWAY 30 KIMBALL NE 69145 2076	80	10
7	Scott	12/25/10	65000.0	6	4	Minnetonka MN 55305	85	8
8	Adams	07/26/08	100000.0	10	3	5420 E 108TH ST TULSA OK 74137 7219	90	7
9	Miller	01/16/08	80000.0	2	6	ST JACOB IL 62281	100	2

Result:

Removing duplicates in datasets and execute query successfully.

Experiment 6: Write a program to calculate basic descriptive statistics using numpy and pandas in python.

Aim: Write a program to calculate basic descriptive statistics.

Procedure:

Step1: Import libraries.

Step2: Prepare data.

Step3: Calculate statistics using numpy and pandas.

Step4: Display Result.

Program:

```
import numpy as np

import pandas as pd


# Sample data

data = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]


# Convert to a NumPy array

data_np = np.array(data)


# Convert to a Pandas Series

data_pd = pd.Series(data)


# NumPy statistics

mean_np = np.mean(data_np)

median_np = np.median(data_np)

std_dev_np = np.std(data_np)

variance_np = np.var(data_np)

min_np = np.min(data_np)

max_np = np.max(data_np)

quantiles_np = np.percentile(data_np, [25, 50, 75])
```

```
# Pandas statistics
mean_pd = data_pd.mean()
median_pd = data_pd.median()
std_dev_pd = data_pd.std()
variance_pd = data_pd.var()
min_pd = data_pd.min()
max_pd = data_pd.max()
quantiles_pd = data_pd.quantile([0.25, 0.5, 0.75])

# Display results
print("NumPy Statistics:")
print(f"Mean: {mean_np}")
print(f"Median: {median_np}")
print(f"Standard Deviation: {std_dev_np}")
print(f"Variance: {variance_np}")
print(f"Minimum: {min_np}")
print(f"Maximum: {max_np}")
print(f"25th Percentile: {quantiles_np[0]}")
print(f"50th Percentile (Median): {quantiles_np[1]}")
print(f"75th Percentile: {quantiles_np[2]}")

print("\nPandas Statistics:")
print(f"Mean: {mean_pd}")
print(f"Median: {median_pd}")
print(f"Standard Deviation: {std_dev_pd}")
print(f"Variance: {variance_pd}")
print(f"Minimum: {min_pd}")
print(f"Maximum: {max_pd}")
```

```
print(f"25th Percentile: {quantiles_pd[0.25]}")
print(f"50th Percentile (Median): {quantiles_pd[0.5]}")
print(f"75th Percentile: {quantiles_pd[0.75]}")
```

Output:

NumPy Statistics:

Mean: 55.0

Median: 55.0

Standard Deviation: 28.722813232690143

Variance: 825.0

Minimum: 10

Maximum: 100

25th Percentile: 32.5

50th Percentile (Median): 55.0

75th Percentile: 77.5

Pandas Statistics:

Mean: 55.0

Median: 55.0

Standard Deviation: 28.722813232690143

Variance: 825.0

Minimum: 10

Maximum: 100

25th Percentile: 32.5

50th Percentile (Median): 55.0

75th Percentile: 77.5

Result:

Thus the program in python are executed successfully.

Experiment 7: Write a python code for performing web scraping and parse all the HTML tags present on the webpage.

Aim: Python code for performing web scraping and parse all the HTML tags present on the webpage.

Procedure:

Step1: **Install Required Libraries**:

- `requests` for sending HTTP requests.
- `beautifulsoup4` for parsing HTML

Step2: Fetch the HTML content of a webpage.

Step3: Parse the HTML using BeautifulSoup.

Step4: Extract and print all unique HTML tags from the parsed HTML.

Step5: Execute the code.

Program:

```
import requests from bs4

import BeautifulSoup


# URL of the webpage to scrape url =

'http://example.com'  # Replace with your desired URL


# Send a GET request to fetch the webpage content

response = requests.get(url)


# Check if the request was successful

if response.status_code == 200:

    # Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')


    # Extract all HTML tags     tags = set(tag.name

for tag in soup.find_all(True))     # Print the
```

unique HTML tags print("Unique HTML tags

found on the webpage:") for tag in sorted(tags):

print(tag)

else:

print(f"Failed to retrieve the webpage. Status code: {response.status_code}")

Output:

Unique HTML tags found on the webpage:

a

html

head

body

title

Result:

Thus the program in python are executed successfully.

Experiment 8: Write a python code for performing web scraping using beautiful soup library, Parse all the tables data present on that webpage.

Aim: Python code for performing web scraping using beautiful soup library, Parse all the tables data present on that webpage.

Procedure:

Step1: Install Required Libraries:

- `requests` for sending HTTP requests.
- `beautifulsoup4` for parsing HTML.

Step2: Fetch the HTML content of a webpage.

Step3: Parse the HTML using BeautifulSoup.

Step4: Extract and print all table data from the parsed HTML.

Step5: Execute the programme.

Program:

```
import requests from bs4

import BeautifulSoup


# URL of the webpage to scrape url =

'https://www.example.com'  # Replace with your desired URL


# Send a GET request to fetch the webpage content

response = requests.get(url)


# Check if the request was successful

if response.status_code == 200:

    # Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')


    # Find all tables on the webpage

tables = soup.find_all('table')
```

```
    # Iterate over each table     for table_idx, table
in enumerate(tables, start=1):
        print(f"Table {table_idx}:")

        # Extract table headers         headers = [header.get_text(strip=True)
for header in table.find_all('th')]         print(f"Headers: {headers}")

        # Extract table rows         rows =
table.find_all('tr')         for row_idx, row
in enumerate(rows):
            cells = row.find_all('td')             row_data =
[cell.get_text(strip=True) for cell in cells]             if
row_data:
 print(f"Row {row_idx}: {row_data}")

      print("\n" + "-"*50 + "\n")
else:
    print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
```

Output:

```
Table 1:
Headers: ['Name', 'Age', 'Location']
Row 0: ['John Doe', '30', 'New York']
Row 1: ['Jane Smith', '25', 'Los Angeles']

--------------------------------------------------
Table 2:
Headers: ['Product', 'Price', 'Quantity']
Row 0: ['Laptop', '$1000', '10']
Row 1: ['Mouse', '$50', '100']
```

Result:

Thus the program in python are executed successfully

Experiment 9: Write a program to use Regex Expressions in python.

Aim: Write a program to use Regex Expressions.

Procedure:

Step1: Import the 're' module.

Step2: Define a sample text.

Step3: Compile a regex pattern.

Step4: Search for the pattern.

Step5: Extract information.

Step6: Find all occurrence.

Step7: Replace text.

Step8: Display results.

Program:

```
import re

# Sample text

text = "The email addresses are example1@example.com and example2@example.com. The
phone number is 123-456-7890."


# Define regex patterns email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-

Za-z0-9.-]+\.[A-Z|a-z]{2,}\b' phone_pattern = r'\b\d{3}-\d{3}-\d{4}\b'


# Compile regex patterns email_regex =

re.compile(email_pattern) phone_regex

= re.compile(phone_pattern)


# Search for the first occurrence of the email

pattern email_match = email_regex.search(text) if

email_match:

print("First email found:", email_match.group())
```

```
# Find all occurrences of the email pattern
all_emails = email_regex.findall(text)
print("All email addresses found:",
all_emails)

# Find all occurrences of the phone pattern
all_phones = phone_regex.findall(text)
print("All phone numbers found:", all_phones)

# Replace email addresses with a placeholder
text_with_placeholder = email_regex.sub('REDACTED', text)
print("Text with emails replaced:", text_with_placeholder)

# Replace phone numbers with a placeholder text_with_placeholder_phones
= phone_regex.sub('PHONE_REDACTED', text) print("Text with phone
numbers replaced:", text_with_placeholder_phones)
```

Output:

First email found: example1@example.com

All email addresses found: ['example1@example.com', 'example2@example.com']

All phone numbers found: ['123-456-7890']

Text with emails replaced: The email addresses are REDACTED and REDACTED. The phone number is 123-456-7890.

Text with phone numbers replaced: The email addresses are example1@example.com and example2@example.com. The phone number is PHONE_REDACTED.

Result:

Thus the program in python are executed successfully.

Experiment 10 : Write a program to perform Outliers detections in python.

Aim : In python write a program to perform Outliers detections in python.

Procedure :

Step1 : import pandas as pd for dataframe

Step2 : import numpy as np

Step3 : from scipy import stats using some scientific operation in python and detect outliers

Program :

```
import pandas as pd
import numpy as np
from scipy import stats

# Sample DataFrame
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Age': [25, 28, 29, 24, 27, 22, 26, 23, 100, 26],  # 100 is an outlier
    'Salary': [50000, 52000, 51000, 49000, 70000, 50000, 50500, 49500, 50500, 49500]
}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)

# Step 1: Detect outliers using Z-Score
df['Z_Score_Age'] = np.abs(stats.zscore(df['Age']))

# Set a threshold for Z-Score (commonly 3)
outliers_zscore = df[df['Z_Score_Age'] > 3]

print("\nOutliers detected using Z-Score Method (Age):")
```

```
print(outliers_zscore)

# Step 2: Detect outliers using Interquartile Range (IQR)
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1

# Define the bounds for detecting outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers_iqr = df[(df['Age'] < lower_bound) | (df['Age'] > upper_bound)]

print("\nOutliers detected using IQR Method (Age):")
print(outliers_iqr)
```

Output :

```
Original DataFrame:
   ID  Age  Salary
0   1   25   50000
1   2   28   52000
2   3   29   51000
3   4   24   49000
4   5   27   70000
5   6   22   50000
6   7   26   50500
7   8   23   49500
8   9  100   50500
9  10   26   49500

Outliers detected using Z-Score Method (Age):
Empty DataFrame
Columns: [ID, Age, Salary, Z_Score_Age]
Index: []

Outliers detected using IQR Method (Age):
   ID  Age  Salary  Z_Score_Age
8   9  100   50500     2.987382
```

Result :

Run program successfully and detect outliers are done.

Experiment 11: Write a program to read any tabular dataset and perform data cleaning aspects.

Aim : In python write a program to read any tabular dataset and perform data cleaning aspects using some python library.

Procedure :

Step1 : Import dataset and read in python by using pd.read_csv()

Step2 : Using some python operation and filled the missing value with the mean and categorical columns are filled with the mode.

Step3 : Duplicate rows in the dataset are removed.

Step4 : Outliers are identified as values that are beyond standard deviation from the mean.

Program :

```
import pandas as pd

import numpy as np


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


print('Original Data : ')

print(df.head())


df.fillna(df.mean(numeric_only=True) , inplace=True)


print("\n Data after handling missing value : ")

print(df.head())


df_no_duplicate = df.drop_duplicates()


print("\n Data after removing duplicates : ")

print(df_no_duplicate)


if 'Due Working Hours' in df.columns:
```

```
    df_no_outliers  = df_no_duplicate[np.abs(df_no_duplicate['Due Working Hours'] -
df_no_duplicate['Due Working Hours'].mean()) <= (3 * df_no_duplicate['Due Working
Hours'].std())]

    print("\n Data after removing outliers in 'age' column: ")

    print(df_no_duplicate.head())

else:

    df_no_outliers = df_no_duplicate


df_corrected = df_no_outliers.copy()

for col in df_corrected.select_dtypes(include=['object']):

    try:

        df_corrected[col]= pd.to_numeric(df_corrected[col], errors = 'coerce')


    except ValueError:

        pass


print("\n Data after correcting data types: ")

print(df_corrected.head())
```

Output:

```
 Data after handling missing value :
  Name of Employee Hire Date   Salary  ...                        Address of Employee  Due Working Hours Performance Score
0           Blake  03/15/14  50000.0  ...  4150 Sydney Place Washington DC 20521-4150                70                 9
1           Clark  05/12/14  45000.0  ...    3290 Hermosillo Place Newyork 35201-0288                55                 8
2           Jones  11/01/13  70000.0  ...      799 E DRAGRAM SUITE 5A TUCSON AZ 85705                78                 9
3          Martin  08/12/14  48000.0  ...         300 BOYLSTON AVE E SEATTLE WA 98102                60                 6
4           Allen  05/23/13  66000.0  ...   11080 CIRCLE POINT RD STE 180 WESTMINSTER                40                 4

[5 rows x 8 columns]
```

```
 Data after removing duplicates :
  Name of Employee Hire Date    Salary  ...                              Address of Employee  Due Working Hours Performance Score
0            Blake  03/15/14   50000.0  ...         4150 Sydney Place Washington DC 20521-4150                 70                 9
1            Clark  05/12/14   45000.0  ...           3290 Hermosillo Place Newyork 35201-0288                 55                 8
2            Jones  11/01/13   70000.0  ...             799 E DRAGRAM SUITE 5A TUCSON AZ 85705                 78                 9
3           Martin  08/12/14   48000.0  ...                300 BOYLSTON AVE E SEATTLE WA 98102                 60                 6
4            Allen  05/23/13   66000.0  ...          11080 CIRCLE POINT RD STE 180 WESTMINSTER                 40                 4
5             Ward  01/01/10   50000.0  ...  112 WEST 17TH ST PO BOX 7  FALLS CITY NE  6835...                 70                10
6             Ford  05/06/09   75000.0  ...            4343 E HIGHWAY 30 KIMBALL NE 69145 2076                 80                10
7            Scott  12/25/10   65000.0  ...                              Minnetonka MN  55305                 85                 8
8            Adams  07/26/08  100000.0  ...               5420 E 108TH ST TULSA OK  74137 7219                 90                 7
9           Miller  01/16/08   80000.0  ...                                 ST JACOB IL 62281                100                 2

[10 rows x 8 columns]

 Data after removing outliers in 'age' column:
  Name of Employee Hire Date   Salary  ...                      Address of Employee  Due Working Hours Performance Score
0            Blake  03/15/14  50000.0  ...  4150 Sydney Place Washington DC 20521-4150                 70                 9
1            Clark  05/12/14  45000.0  ...    3290 Hermosillo Place Newyork 35201-0288                 55                 8
2            Jones  11/01/13  70000.0  ...      799 E DRAGRAM SUITE 5A TUCSON AZ 85705                 78                 9
3           Martin  08/12/14  48000.0  ...         300 BOYLSTON AVE E SEATTLE WA 98102                 60                 6
4            Allen  05/23/13  66000.0  ...   11080 CIRCLE POINT RD STE 180 WESTMINSTER                 40                 4

[5 rows x 8 columns]

 Data after correcting data types:
   Name of Employee  Hire Date   Salary  Paid leaves Remaining  Sick Leaves Remaining  Address of Employee  Due Working Hours  Performance Score
0               NaN        NaN  50000.0                     10                      6                  NaN                 70                  9
1               NaN        NaN  45000.0                     10                      3                  NaN                 55                  8
2               NaN        NaN  70000.0                      3                     10                  NaN                 78                  9
3               NaN        NaN  48000.0                      7                      7                  NaN                 60                  6
4               NaN        NaN  66000.0                      8                      9                  NaN                 40                  4
PS C:\Users\vg270\Downloads\Data Wrangling>
```

Result :

Run program successfully handle and clean dataset by using python.

Experiment 12: Write a program to merge and combine the data in pandas objects.

Aim : In python create a dataset or load and merge the dataset by using some python operations.

Procedure:

Step1 : Import the pandas library for create a datasets or read a file.

Step2 : Import or create a datasets in python make sure create a two datasets.

Step3 : This function is used to combine dataframes in a single column (‘inner’,’outer’)

Program :

```
import pandas as pd

# Sample Data

data1 = {'ID': [1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21]}

data2 = {'ID':
[3,4,5,6],'Salary':[70000,80000,90000,100000],'Department':['HR','FINANCE','ENGINEERING','
MARKETING']}


df1 = pd.DataFrame(data1)

df2 = pd.DataFrame(data2)

print("DataFrame 1 : ")

print(df1)


print("DataFrame 2 : ")

print(df2)


df_inner_merge = pd.merge(df1,df2,on='ID',how='inner')


print("\n Inner Merge (intersection of DataFrame) : ")

print(df_inner_merge)


df_outer_merge = pd.merge(df1,df2,on='ID', how = 'outer')

print("\n Inner Merge (intersection of DataFrame) : ")

print(df_outer_merge)
```

Output:

```
DataFrame 1 :
   ID    Name  Age
0   1  Prerna   18
1   2   Tanya   19
2   3    Isha   20
3   4  Vishal   21
DataFrame 2 :
   ID  Salary   Department
0   3   70000           HR
1   4   80000      FINANCE
2   5   90000  ENGINEERING
3   6  100000    MARKETING

 Inner Merge (intersection of DataFrame) :
   ID    Name  Age  Salary Department
0   3    Isha   20   70000         HR
1   4  Vishal   21   80000    FINANCE

 Inner Merge (intersection of DataFrame) :
   ID    Name   Age    Salary   Department
0   1  Prerna  18.0       NaN          NaN
1   2   Tanya  19.0       NaN          NaN
2   3    Isha  20.0   70000.0           HR
3   4  Vishal  21.0   80000.0      FINANCE
4   5     NaN   NaN   90000.0  ENGINEERING
5   6     NaN   NaN  100000.0    MARKETING
```

Result :

Run Program Successfully and merge two datasets are done.

Experiment 13 : Perform concatenation in the pandas library.

Aim : In python use concatenation operation and merge two datasets.

Procedure:

Step1 : Import the pandas library for create a datasets or read a file.

Step2 : Import or create a datasets in python make sure create a two datasets.

Step3 : This function is used to combine dataframes in a single column concat().

Program :

```
import pandas as pd

# Sample Data

data1 = {'ID': [1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21]}

data2 = {'ID':
[3,4,5,6],'Salary':[70000,80000,90000,100000],'Department':['HR','FINANCE','ENGINEERING','
MARKETING']}


df1 = pd.DataFrame(data1)

df2 = pd.DataFrame(data2)


print("DataFrame 1 : ")

print(df1)


print("DataFrame 2 : ")

print(df2)


df_concat = pd.concat([df1,df2] ,ignore_index=True ,sort = False)


print("\nConcatenated DataFrames : ")

print(df_concat)
```

Output:

```
DataFrame 1 :
   ID    Name  Age
0   1  Prerna   18
1   2   Tanya   19
2   3    Isha   20
3   4  Vishal   21
DataFrame 2 :
   ID  Salary   Department
0   3   70000           HR
1   4   80000      FINANCE
2   5   90000  ENGINEERING
3   6  100000    MARKETING

Concatenated DataFrames :
   ID    Name   Age    Salary   Department
0   1  Prerna  18.0       NaN          NaN
1   2   Tanya  19.0       NaN          NaN
2   3    Isha  20.0       NaN          NaN
3   4  Vishal  21.0       NaN          NaN
4   3     NaN   NaN   70000.0           HR
5   4     NaN   NaN   80000.0      FINANCE
6   5     NaN   NaN   90000.0  ENGINEERING
7   6     NaN   NaN  100000.0    MARKETING
```

Result:

Run program successfully and use concat() function to merge the datasets are done.

Experiment 14 : Write a program to implement Reshaping and Pivoting using Pandas object.

Aim : In python write a program to implement reshaping and pivoting.

Procedure:

Step1 : Import the pandas library for create a datasets or read a file.

Step2 : Reshape the data to use pivot() operation.

Program :

```
import pandas as pd

# Sample Data

data = {'ID':
[1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21],'Salary':[70000,80000,90000,
100000],'Department':['HR','FINANCE','ENGINEERING','MARKETING']}

df = pd.DataFrame(data)


print("Original DataFrame  : ")

print(df)


df_pivot = df.pivot(index='ID', columns='Name',values='Age')


print("\n Pivoted DataFrame (ID as index ,name as columns , age as values) : ")

print(df_pivot)


df_pivot_table  = df.pivot_table(index='ID',columns='Name',values='Age',aggfunc='sum')


print("\n Pivot table(Total Sum Of Data)")

print(df_pivot_table)
```

Output :

```
Original DataFrame  :
   ID    Name  Age
0   1  Prerna   18
1   2   Tanya   19
2   3    Isha   20
3   4  Vishal   21

 Pivoted DataFrame (ID as index ,name as columns , age as values) :
Name  Isha  Prerna  Tanya  Vishal
ID
1      NaN    18.0    NaN     NaN
2      NaN     NaN   19.0     NaN
3     20.0     NaN    NaN     NaN
4      NaN     NaN    NaN    21.0

 Pivot table(Total Sum Of Data)
Name  Isha  Prerna  Tanya  Vishal
ID
1      NaN    18.0    NaN     NaN
2      NaN     NaN   19.0     NaN
3     20.0     NaN    NaN     NaN
4      NaN     NaN    NaN    21.0
```

Result :

Run program successfully and using pivot operation to reshape the data are done.

Experiment 15 : Write a program to perform Groupby operations using Pandas.

Aim : In python write a program to perform Groupby operations

Procedure :

Step1 : Import the pandas library for create a datasets or read a file.

Step2 : Using the groupby clause in python.

Program :

```
import pandas as pd

# Sample Data

data = {'ID':
[1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21],'Salary':[70000,80000,90000,
100000],'Department':['HR','FINANCE','ENGINEERING','MARKETING']}

df = pd.DataFrame(data)


print("Original DataFrame  : ")

print(df)


grouped_salary = df.groupby('Department')['Salary'].sum()


print("\n Total Salary by Department : ")

print(grouped_salary)


grouped_age = df.groupby('Department')['Age'].mean()


print("\n Average age by Department : ")

print(grouped_age)
```

Output :

```
Original DataFrame  :
   ID    Name  Age  Salary   Department
0   1  Prerna   18   70000           HR
1   2   Tanya   19   80000      FINANCE
2   3    Isha   20   90000  ENGINEERING
3   4  Vishal   21  100000    MARKETING

 Total Salary by Department :
Department
ENGINEERING     90000
FINANCE         80000
HR              70000
MARKETING      100000
Name: Salary, dtype: int64

 Average age by Department :
Department
ENGINEERING    20.0
FINANCE        19.0
HR             18.0
MARKETING      21.0
Name: Age, dtype: float64
```

Result :

Run program successfully and using groupby operation are done.

Experiment 16 : Write a program to perform data visualization using matplotlib.

Aim : In python write a program to perform data visualization using line plot , scatterplot using in matplotlib.

Procedure :

Step1 : Import matplotlib as plt for further using.

Step2 : Import numpy as np.

Step3 : Import pandas as pd for dataframe.

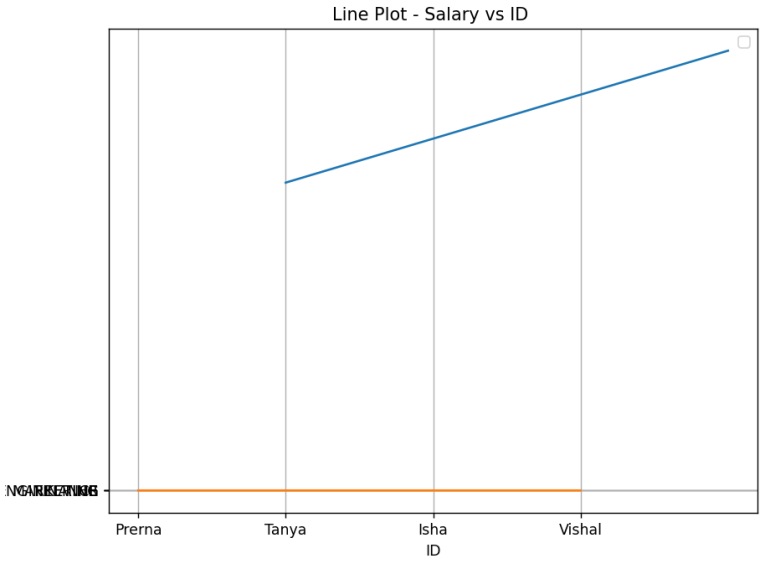
Step4 : Create a datasets for further apply matplotlib operations.

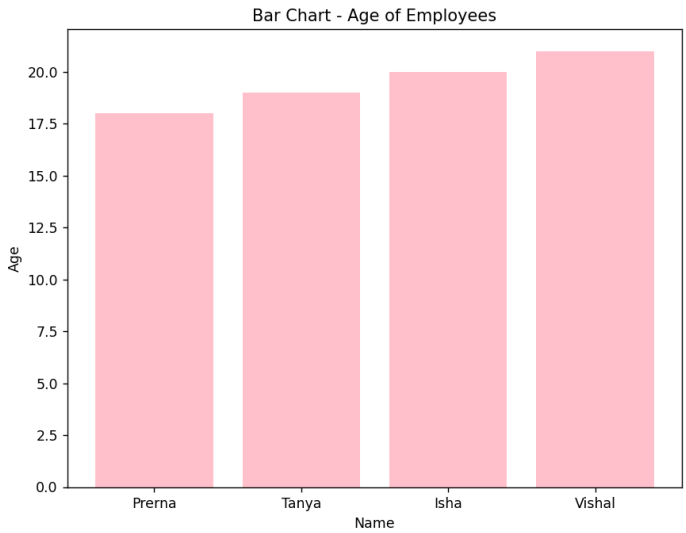
Program :

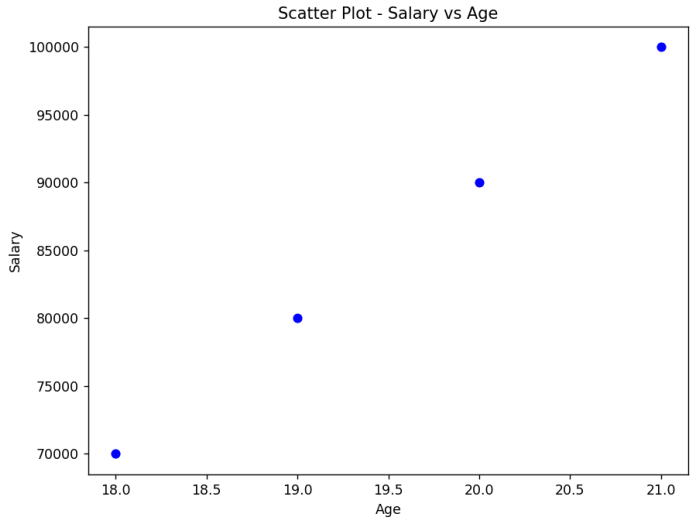
```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd


data = {'ID':
[1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21],'Salary':[70000,80000,90000,
100000],'Department':['HR','FINANCE','ENGINEERING','MARKETING']}

df = pd.DataFrame(data)


print("Original DataFrame  : ")

print(df)


# Line Plot

plt.figure(figsize=(8,6))

plt.plot(df['ID'],df['Salary'],df['Name'],df['Department'])

plt.title("Line Plot - Salary vs ID ")

plt.xlabel("ID")

plt.ylabel("Salary")

plt.legend()

plt.grid(True)

plt.show()
```

```
# Bar Chart
plt.figure(figsize=(8,6))
plt.bar(df['Name'],df['Age'],color='pink')
plt.title("Bar Chart - Age of Employees")
plt.xlabel("Name")
plt.ylabel("Age")
plt.show()

# Scatter Plot
plt.figure(figsize=(8,6))
plt.scatter(df['Age'],df['Salary'],color='blue')
plt.title("Scatter Plot - Salary vs Age")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.show()
```

Output :







Result :

Run program successfully and using matplotlib and create line plot and scatter plot are done.

Experiment 17 : Write a program to Perform aggregation operation on a dataframe.

Aim : In python write a program to Perform aggregation operation on a dataframe.

Procedure :

Step1 : import numpy as np

Step2 : import pandas as pd for dataframe.

Step3 : Use python operation to Perform aggregation operation on a dataframe.

Program :

```
import numpy as np

import pandas as pd


data = {'ID':
[1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21],'Salary':[70000,80000,90000,
100000],'Department':['HR','FINANCE','ENGINEERING','MARKETING']}

df = pd.DataFrame(data)


print("Original DataFrame  : ")

print(df)


total_salary = df['Salary'].sum()

print("\n Total Salary" ,total_salary)


salary_aggregates = df['Salary'].agg(['mean','sum','max','min'])

print("\n Salary Aggreates (mean,sum,max,min) : ")

print(salary_aggregates)


grouped_aggregates = df.groupby('Department').agg({'Salary' : ['mean','sum'],'Age' : ['mean'
,'max']})


print("\n Group-wise Aggregates (Department) : ")

print(grouped_aggregates)
```

Output :

```
Original DataFrame  :
   ID    Name  Age  Salary   Department
0   1  Prerna   18   70000           HR
1   2   Tanya   19   80000      FINANCE
2   3    Isha   20   90000  ENGINEERING
3   4  Vishal   21  100000    MARKETING

 Total Salary 340000

 Salary Aggreates (mean,sum,max,min) :
mean     85000.0
sum     340000.0
max     100000.0
min      70000.0
Name: Salary, dtype: float64

 Group-wise Aggregates (Department) :
               Salary           Age
                 mean     sum  mean max
Department
ENGINEERING   90000.0   90000  20.0  20
FINANCE       80000.0   80000  19.0  19
HR            70000.0   70000  18.0  18
MARKETING    100000.0  100000  21.0  21
```

Result :

Run program successfully and apply aggregation operation on a dataframe are done.

Experiment 18 : Write a program to perform Cross Tab analysis in python.

Aim : In python write a program to perfrom cross tab analysis.

Procedure:

Step1 : import numpy as np

Step2 : import pandas as pd for dataframe.

Step3 : Use python operation to Perform tab analysis on a dataframe.

Program :

```
import numpy as np

import pandas as pd


data = {'ID':
[1,2,3,4],'Name':['Prerna','Tanya','Isha','Vishal'],'Age':[18,19,20,21],'Salary':[70000,80000,90000,
100000],'Department':['HR','FINANCE','ENGINEERING','MARKETING']}

df = pd.DataFrame(data)


print("Original DataFrame  : ")

print(df)


cross_tab = pd.crosstab(df['Department'],df['Name'])


print("\n Cross Tabulation between Department and Age")

print(cross_tab)


cross_tab_margins = pd.crosstab(df['Department'],df['Name'],margins=True)


print("\n Cross Tabulation with Totals (Margins) : ")

print(cross_tab_margins)
```

Output :

```
Original DataFrame  :
   ID    Name  Age  Salary   Department
0   1  Prerna   18   70000           HR
1   2   Tanya   19   80000      FINANCE
2   3    Isha   20   90000  ENGINEERING
3   4  Vishal   21  100000    MARKETING

 Cross Tabulation between Department and Age
Name         Isha  Prerna  Tanya  Vishal
Department
ENGINEERING     1       0      0       0
FINANCE         0       0      1       0
HR              0       1      0       0
MARKETING       0       0      0       1

 Cross Tabulation with Totals (Margins) :
Name         Isha  Prerna  Tanya  Vishal  All
Department
ENGINEERING     1       0      0       0    1
FINANCE         0       0      1       0    1
HR              0       1      0       0    1
MARKETING       0       0      0       1    1
All             1       1      1       1    4
```

Result :

Run program successfully and perform cross tab analysis on dataframe are done.