Practical 1

Aim: - Write A Python Script To Read Each Row From A Given CSV File And Print A List Of Strings.

Code

```
import pandas as pd
import csv
import numpy as np
with open ('/content/diabetes.csv')as obj:
  row_obj = csv.reader(obj)
  for row in row_obj:
    print(row)
```

Output

```
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
['6', '148', '72', '35', '0', '33.6', '0.627', '50', '1']
['1', '85', '66', '29', '0', '26.6', '0.351', '31', '0']
['8', '183', '64', '0', '0', '23.3', '0.672', '32', '1']
['1', '89', '66', '23', '94', '28.1', '0.167', '21', '0']
['0', '137', '40', '35', '168', '43.1', '2.288', '33', '1']
['5', '116', '74', '0', '0', '25.6', '0.201', '30', '0']
['3', '78', '50', '32', '88', '31', '0.248', '26', '1']
['10', '115', '0', '0', '0', '35.3', '0.134', '29', '0']
['2', '197', '70', '45', '543', '30.5', '0.158', '53', '1']
['8', '125', '96', '0', '0', '0', '0.232', '54', '1']
['4', '110', '92', '0', '0', '37.6', '0.191', '30', '0']
```

Practical 2

Aim: - Write a Python program to read a given CSV file as a dictionary.Code

```
import csv
with open ('/content/gender.csv','r')as file:
  reader = csv.DictReader( file,fieldnames =['Name','HEX','RGB'])
  for row in reader:
    print(row)
```

output

```
{'Name': ' Gender', 'HEX': ' Age', 'RGB': ' Height (cm)', None: [' Weight (kg)', ' Occupation', ' Education Level', ' Marital Stat
{'Name': ' male', 'HEX': '32', 'RGB': '175', None: ['70', ' Software Engineer', " Master's Degree", ' Married', '75000', ' Blue',
{'Name': ' male', 'HEX': '25', 'RGB': '182', None: ['85', ' Sales Representative', " Bachelor's Degree", ' Single', '45000', ' Gre
{'Name': ' female', 'HEX': '41', 'RGB': '160', None: ['62', ' Doctor', ' Doctorate Degree', ' Married', '120000', ' Purple', '']}
{'Name': ' male', 'HEX': '38', 'RGB': '178', None: ['79', ' Lawyer', " Bachelor's Degree", ' Single', '90000', ' Red', '']}
{'Name': ' female', 'HEX': '29', 'RGB': '165', None: ['58', ' Graphic Designer', " Associate's Degree", ' Single', '35000', ' Yell
{'Name': ' male', 'HEX': '45', 'RGB': '190', None: ['92', ' Business Consultant', " Master's Degree", ' Divorced', '110000', ' Bla
{'Name': ' female', 'HEX': '27', 'RGB': '163', None: ['55', ' Marketing Specialist', " Bachelor's Degree", ' Single', '50000', ' F
{'Name': ' male', 'HEX': '52', 'RGB': '179', None: ['83', ' CEO', ' Doctorate Degree', ' Married', '500000', ' Blue', '']}
{'Name': ' female', 'HEX': '31', 'RGB': '168', None: ['61', ' Project Manager', " Bachelor's Degree", ' Married', '80000', ' Greer
{'Name': ' male', 'HEX': '36', 'RGB': '177', None: ['76', ' Engineer', " Master's Degree", ' Married', '95000', ' Orange', '']}
{'Name': ' female', 'HEX': '24', 'RGB': '162', None: ['53', ' Accountant', " Bachelor's Degree", ' Single', '40000', ' Blue', '']}
{'Name': ' male', 'HEX': '44', 'RGB': '183', None: ['87', ' Architect', " Bachelor's Degree", ' Widowed', '120000', ' Grey', '']}
{'Name': ' female', 'HEX': '28', 'RGB': '166', None: ['60', ' Nurse', " Associate's Degree", ' Married', '55000', ' Purple', '']}
{'Name': ' male', 'HEX': '29', 'RGB': '181', None: ['84', ' Analyst', " Bachelor's Degree", ' Single', '60000', ' Red', '']}
{'Name': ' female', 'HEX': '33', 'RGB': '170', None: ['65', ' Teacher', " Master's Degree", ' Married', '65000', ' Yellow', '']}
{'Name': ' male', 'HEX': '37', 'RGB': '176', None: ['78', ' IT Manager', " Bachelor's Degree", ' Married', '85000', ' Green', '']}
```

Practical 3

Aim: - Write a Python program to convert Python dictionary object (sort by key) to JSON data. Print the object members with indent level 4

Code

```
import json
dictionary ={
    "id":"28",
    "name":"pooja",
    "development":"HR"
}
json_object=json.dumps(dictionary,indent=4)
print(json_object)
{
    "id":"28",
    "name":"pooja",
    "development":"HR"
}
```

Output

```
{
    "id": "28",
    "name": "pooja",
    "development": "HR"
}
{'id': '28', 'name': 'pooja', 'development': 'HR'}
```

```
 {
     "id": "28",
     "name": "pooja",
     "development": "HR"
 }
```

Practical 4

Aim :- Write the python script to Read the XML file

Code

```
import xml
data = pd.read_xml('/content/Demo_xml.xml')
data
```

Output

	id	author	title	genre	price	publish_date	description
0	bk101	Gambardella, Matthew	XML Developer's Guide	Computer	44.95	2000-10-01	An in-depth look at creating applications \n ...
1	bk102	Ralls, Kim	Midnight Rain	Fantasy	5.95	2000-12-16	A former architect battles corporate zombies, ...
2	bk103	Corets, Eva	Maeve Ascendant	Fantasy	5.95	2000-11-17	After the collapse of a nanotechnology \n ...
3	bk104	Corets, Eva	Oberon's Legacy	Fantasy	5.95	2001-03-10	In post-apocalypse England, the mysterious \n ...
4	bk105	Corets, Eva	The Sundered Grail	Fantasy	5.95	2001-09-10	The two daughters of Maeve, half-sisters, \n ...
5	bk106	Randall, Cynthia	Lover Birds	Romance	4.95	2000-09-02	When Carla meets Paul at an ornithology \n ...
6	bk107	Thurman, Paula	Splish Splash	Romance	4.95	2000-11-02	A deep sea diver finds true love twenty \n ...

Practical 5

Aim: - Write a Pandas program to import excel data (child labour and child marriage data.xlsx) into a Pandas data frame and process the following

a. Get the data types of the given excel data
b. Display the last ten rows.
c. Insert a column in the sixth position of the said excel sheet and fill it with NaN values

Code

```
import pandas as pd
import numpy as np
import openpyxl
excel_file = '/content/Child-marriage-database.csv'
df = pd.read_csv(excel_file)
df
data_types = df.dtypes
print("Data Type:")
print(data_types)
last_ten_row = df.tail(10)
print("\nLast Ten Rows:")
print(last_ten_row)
```

Output

```
Data Type:
Country                 object
Female Married by 15    float64
Female Married by 18    float64
Reference year          float64
Data source              object
Male Married by 18      float64
Male Reference year     float64
Data source.1            object
dtype: object
```

```
Last Ten Rows:
                            Country  Female Married by 15  \
192        United Republic of Tanzania                   5.0
193                       United States                   NaN
194                             Uruguay                   1.0
195                          Uzbekistan                   0.0
196                             Vanuatu                   3.0
197  Venezuela (Bolivarian Republic of)                   NaN
198                            Viet Nam                   1.0
199                               Yemen                   9.0
200                              Zambia                   5.0
201                            Zimbabwe                   5.0
```

	Female Married by 18	Reference year	Data source	Male Married by 18
192	31.0	2016.0	DHS 2015-16	4.0
193	NaN	NaN	NaN	NaN
194	25.0	2013.0	MICS 2013	NaN
195	7.0	2006.0	MICS 2006	1.0
196	21.0	2013.0	DHS 2013	5.0
197	NaN	NaN	NaN	NaN
198	11.0	2014.0	MICS 2014	3.0
199	32.0	2013.0	DHS 2013	NaN
200	29.0	2018.0	DHS 2018	3.0
201	34.0	2019.0	MICS 2019	2.0

	Male Reference year	Data source.1
192	2016.0	DHS 2015-16
193	NaN	NaN
194	NaN	NaN
195	2002.0	DHS 2002
196	2013.0	DHS 2013
197	NaN	NaN
198	2005.0	AIS 2005
199	NaN	NaN
200	2018.0	DHS 2018
201	2019.0	MICS 2019

Practical 6

Aim: -Develop the python script to parsing the PDF files using pdfminer

Code

```
!pip install pdfminer.six
from pdfminer.high_level import extract_text

def parse_pdf(file_path):
    text = extract_text(file_path)
    return text

pdf_file = '/content/drive/MyDrive/Pooja_ kadam .pdf'
parsed_text = parse_pdf(pdf_file)
print(parsed_text)
```

Output

```
methodologies, and IT governance standards. Passionate about proactive threat hunting and digital
asset protection.

Work Experience

Cyber Security Audit Assistant

📍 Mehta Sanghvi & Associates, Mumbai
 🗓 March 2024 - Present

• Conduct cybersecurity audits and ensure compliance with security frameworks.
• Identify vulnerabilities, implement remediation strategies, and mitigate threats.
• Develop and present detailed audit reports with actionable insights.
• Collaborate with teams to improve security policies and best practices.
• Stay updated on emerging cybersecurity threats to ensure proactive risk management.

Consultant Network Engineer

📍 Ebix Cash, Mumbai
```

Practical 7

Aim: - Extract the table from the child labour and child marriage data.xlsx using pdfables library

Code

```
import pandas as pd
excel_file="/content/Child-marriage-database.csv"
data_frame=pd.read_csv(excel_file)
tables=data_frame.values.tolist()
for table in tables:
  print(table)
```

Output

```
['Afghanistan', 4.0, 28.0, 2017.0, 'ALCS 2016-17', 7.0, 2015.0, 'DHS 201
['Albania', 1.0, 12.0, 2018.0, 'DHS 2017-18', 1.0, 2018.0, 'DHS 2017-18']
['Algeria', 0.0, 3.0, 2013.0, 'MICS 2012-13', nan, nan, nan]
['Andorra', nan, nan, nan, nan, nan, nan, nan]
['Angola', 8.0, 30.0, 2016.0, 'DHS 2015-16', 6.0, 2016.0, 'DHS 2015-16']
['Anguilla', nan, nan, nan, nan, nan, nan, nan]
['Antigua and Barbuda', nan, nan, nan, nan, nan, nan, nan]
['Argentina', nan, nan, nan, nan, nan, nan, nan]
['Armenia', 0.0, 5.0, 2016.0, 'DHS 2015-16', 0.0, 2016.0, 'DHS 2015-16']
['Australia', nan, nan, nan, nan, nan, nan, nan]
['Austria', nan, nan, nan, nan, nan, nan, nan]
['Azerbaijan', 2.0, 11.0, 2011.0, 'DHS 2011', 0.0, 2006.0, 'DHS 2006']
['Bahamas', nan, nan, nan, nan, nan, nan, nan]
['Bahrain', nan, nan, nan, nan, nan, nan, nan]
['Bangladesh', 22.0, 59.0, 2014.0, 'DHS 2014', 4.0, 2011.0, 'DHS 2011']
['Barbados', 8.0, 29.0, 2012.0, 'MICS 2012', nan, nan, nan]
['Belarus', 0.0, 5.0, 2019.0, 'MICS 2019', 1.0, 2012.0, 'MICS 2012']
['Belgium', nan, nan, nan, nan, nan, nan, nan]
['Belize', 6.0, 34.0, 2016.0, 'MICS 2015-16', 22.0, 2016.0, 'MICS 2015-16']
['Benin', 9.0, 31.0, 2018.0, 'DHS 2017-18', 5.0, 2018.0, 'DHS 2017-18']
['Bhutan', 6.0, 26.0, 2010.0, 'MICS 2010', nan, nan, nan]
['Bolivia (Plurinational State of)', 3.0, 20.0, 2016.0, 'DHS 2016', 8.0, 2008.0, 'DHS 2008']
['Bosnia and Herzegovina', 0.0, 4.0, 2012.0, 'MICS 2011-12', 0.0, 2012.0, 'MICS 2011-12']
['Botswana', nan, nan, nan, nan, nan, nan, nan]
```

Practical 8

Aim:- write a python Data Wrangling script to insert the data into SQLite Database

Code

```
import sqlite3
conn = sqlite3.connect('gajera1.db')
cursor = conn.cursor()
create_table_sql ='''
CREATE TABLE IF NOT EXISTS my_tables (
  id INTEGER PRIMARY KEY,
  name TEXT,
  age INTEGER
)
'''
cursor.execute(create_table_sql)
insert_data_sql = '''
INSERT INTO my_tables (name, age) VALUES
('pooja',25),
('krina',90),
('gajera',45),
('pooja',56) -- Removing the trailing comma
'''
cursor.execute(insert_data_sql)
conn.commit()
conn.close()
conn = sqlite3.connect('gajera1.db')
cursor = conn.cursor()
select_data_sql = 'SELECT * FROM my_tables'
cursor.execute(select_data_sql)
rows = cursor.fetchall()
for row in rows:
  print(row)
conn.close()
```

Output

```
(1, 'pooja', 25)
(2, 'krina', 90)
(3, 'gajera', 45)
(4, 'pooja', 56)
```

Practical 9

Aim: - Develop the Python Shell Script to do the basic data cleanup on child labour and child marriage data.xlsx

a. Check duplicates and missing data

b. Eliminate Mismatches

c. Cleans line breaks, spaces, and special characters

Code

```
import pandas as pd
# Read the Excel file
excel_file = "/content/Child-marriage-database.csv"
data_frame = pd.read_csv(excel_file)

# Remove duplicates if any
deduplicates = data_frame.duplicated()
if deduplicates.any():
    print("Duplicates found:")
    print(data_frame[deduplicates])
    data_frame = data_frame.drop_duplicates()
    print("Duplicates removed.")
# Check for missing data
missing_data = data_frame.isnull().sum()
if missing_data.any():
    print("Missing data found:")
    print(missing_data)
# Clean up whitespace and strip values in all string columns
data_frame = data_frame.replace(r'\s+', ' ', regex=True)  # Replace multiple spaces with one space
data_frame = data_frame.apply(lambda x: x.str.strip() if x.dtype == "object" else x)  # Strip
leading/trailing spaces
# Display the cleaned data table
print("\nCleaned Data:")
print(data_frame)
# Write the cleaned data to a new Excel file
data_frame.to_excel('output.xlsx', index=False)
# Print completion message
print("\nData cleaning completed and saved to 'output.xlsx'.")
```

Output

```
Missing data found:
Country                   0
Female Married by 15     74
Female Married by 18     74
Reference year           74
Data source              74
Male Married by 18      107
Male Reference year     107
Data source.1           107
dtype: int64
```

```
Cleaned Data:
                            Country  Female Married by 15  \
0                       Afghanistan                   4.0
1                           Albania                   1.0
2                           Algeria                   0.0
3                           Andorra                   NaN
4                            Angola                   8.0
..                              ...                   ...
197  Venezuela (Bolivarian Republic of)                   NaN
198                        Viet Nam                   1.0
199                           Yemen                   9.0
200                          Zambia                   5.0
201                        Zimbabwe                   5.0

     Female Married by 18  Reference year   Data source  Male Married by 18  \
0                    28.0          2017.0  ALCS 2016-17                 7.0
1                    12.0          2018.0   DHS 2017-18                 1.0
2                     3.0          2013.0  MICS 2012-13                 NaN
3                     NaN             NaN           NaN                 NaN
4                    30.0          2016.0   DHS 2015-16                 6.0
..                    ...             ...           ...                 ...
197                   NaN             NaN           NaN                 NaN
198                  11.0          2014.0     MICS 2014                 3.0
199                  32.0          2013.0      DHS 2013                 NaN
200                  29.0          2018.0      DHS 2018                 3.0
201                  34.0          2019.0     MICS 2019                 2.0

     Male Reference year Data source.1
0                 2015.0        DHS 2015
1                 2018.0     DHS 2017-18
2                    NaN             NaN
3                    NaN             NaN
4                 2016.0     DHS 2015-16
..                   ...             ...
197                  NaN             NaN
198               2005.0        AIS 2005
199                  NaN             NaN
200               2018.0        DHS 2018
201               2019.0       MICS 2019

[202 rows x 8 columns]

Data cleaning completed and saved to 'output.xlsx'.
```

Practical 10

Aim: - Import the data into ‘agate then explores the table using agate methods and perform statistical correlations.

Code

```
!pip install agate
import agate
table = agate.Table.from_csv('/content/drive/MyDrive/oral_cancer_data.csv')
print(table)
print("Column Names:", table.column_names)
print("Number of Rows:", len(table.rows))
age = table.aggregate(agate.Sum('Age'))
age = table.aggregate(agate.Mean('Age'))   gender_counts = table.group_by('Gender').aggregate([
   ('count', agate.Count())
])
print("gender_counts:", gender_counts)
print("age:", age)
print("age:", age)
from scipy.stats import pearsonr
import numpy as np
population = [float(value) for value in table.columns['Age']]
gender_mapping = {'Male': 0, 'Female': 1}
households = [float(gender_mapping.get(value, np.nan)) for value in
table.columns['Gender'correlation_coefficient, p_value = pearsonr(population, households)
print("Person correlation Coefficient (age vs gender):", correlation_coefficient)
print("p_value:", p_value)
```

OUTPUT

```
| column         | data_type |
| -------------- | --------- |
| Cancer Type    | Text      |
| Label          | Text      |
| Metastasis     | Boolean   |
| Age            | Number    |
| Gender         | Text      |
| Stage          | Number    |
| Size           | Number    |
| Spread         | Boolean   |
| Survival Rate  | Text      |
| Cause          | Text      |
| Affected Organ | Text      |
| Image Path     | Text      |

Column Names: ('Cancer Type', 'Label', 'Metastasis', 'Age', 'Gender', 'Stage', 'Size', 'Spread', 'Survival Rate', 'Cause', 'Affected
Number of Rows: 2156
gender_counts: | column | data_type |
| ------ | --------- |
| Gender | Text      |
| count  | Number    |

age: 49.87198515769944341372912801
age: 49.87198515769944341372912801
Person correlation Coefficient (age vs gender): 0.00528689746683999
p_value: 0.8061896808857851
```