St. Francis Institute of Technology, Mumbai-400 103

Department of Information Technology

A.Y. 2024-2025

Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 1: To implement Data Preparation using Numpy and Pandas.

1. **Aim:** To implement Data Preparation using Numpy and Pandas.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand Numpy concepts
   - Understand Pandas concepts
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand data preparation, Numpy and Pandas.
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

   Brief Theory:

   Basic Concepts of Pandas and Numpy.
7. **Laboratory Exercise**

   A. Procedure:

Software Installation:

1. Python 3.6
   - This setup requires that your machine has python 3.6 installed on it. you can refer to this url https://www.python.org/downloads/ to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/.
   - Setting up PATH variable is optional as you can also run program without it.
2. Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/
3. You will also need to download and install below 2 packages after you install either python or anaconda from the steps above
   - Pandas
   - Numpy

- if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages

```
pip install pandas
pip install numpy
```

- if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages

  conda install -c anaconda numpy
  conda install -c anaconda pandas

4. Use Google Colab

Dataset Used:

Iris Dataset:

Iris Dataset is considered as the Hello World for data science. It contains five columns namely – Petal Length, Petal Width, Sepal Length, Sepal Width, and Species Type. Iris is a flowering plant, the researchers have measured various features of the different iris flowers and recorded them digitally.

```
import pandas as pd
import numpy as np
df = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv")

df.info()

"""**Data Inspection**"""

df.head(5) # head

df.shape

df.columns

df["sepal_length"].nunique()

df["sepal_length"].unique()

# number of unique values alltogether
df.columns.nunique()

# value counts
df['species'].value_counts()

"""**Dealing with NA values**"""

# show null/NA values per column
df.isnull().sum()

# show NA values as % of total observations per column
df.isnull().sum()*100/len(df)

# drop all rows containing null
df.dropna()
```

```
# drop all columns containing null
df.dropna(axis=1)

# drop columns with less than 5 NA values
df.dropna(axis=1, thresh=5)

# replace all na values with -9999
df.fillna(-9999)

# fill na values with NaN
df.fillna(np.NaN)

# fill na values with strings
df.fillna("data missing")

# fill missing values with mean column values
df.fillna(df.mean())

"""**Column Operation**"""

# select a column
df["sepal_length"]

# select multiple columns and create a new dataframe X
X = df[["sepal_length", "sepal_width", "species"]]
X

# select a column by column number
df.iloc[:, [1,3,4]]

# save all columns to a list
df.columns.tolist()

# sorting values by column "sepalW" in ascending order
df.sort_values(by = "sepal_width", ascending = True)

# add new calculated column
df['newcol'] = df["sepal_length"]*2
df

# create a conditional calculated column
df['newcol'] = ["short" if i<3 else "long" for i in df["sepal_width"]]
df

"""**Row Operation (Sort, Filter, Slice)**"""

# select rows 3 to 10
df.iloc[3:10,]

# select rows 3 to 49 and columns 1 to 3
```

```
df.iloc[3:50, 1:4]

# randomly select 10 rows
df.sample(10)

# find rows with specific strings
df[df["species"].isin(["setosa"])]

# conditional filtering
df[df.sepal_length >= 5]

# filtering rows with multiple values e.g. 0.2, 0.3
df[df["petal_width"].isin([0.2, 0.3])]

# multi-conditional filtering
df[(df.petal_length > 1) & (df.species=="setosa") | (df.sepal_width < 3)]

# drop rows
df.drop(df.index[1]) # 1 is row index to be deleted

"""**Grouping**"""

# data grouped by column "species"
X = df.groupby("species")
X

# return mean values of a column ("sepal_length" ) grouped by "species" column
df.groupby("species")["sepal_length"].mean()

# return mean values of ALL columns grouped by "species" category
df.groupby("species").mean()

# get counts in different categories
df.groupby("species").nunique()
```

Employee Dataset:

Employee dataset contains columns such as first name, gender, start date, last login, salary bonus, senior management and team. Some of the fields are null in the dataset.

```
import pandas as pd
df = pd.read_csv("C:/Users/Vaishali/Desktop/AI-DS/employees.csv")
print(df)

print(df.describe())

#print(df.isNull())

print(pd.isnull(df['Team']))
```

```
print(pd.notnull(df['Team']))

print(df.fillna(1111))


print(df.fillna(method='pad'))

#import pandas as pd
#df=pd.read_csv("employees.csv")
print(df)

df.fillna(method='bfill') # check the output
print(df)

df['Gender'].fillna("No Gender",inplace=True)
print(df)

import numpy as np
print(df.replace(to_replace=np.NaN,value="SFIT"))


print(df)

print(df.interpolate(method='linear',limit_direction='forward'))

df1=pd.DataFrame({"A":[12,23,None,5,6,None],
                  "B":[34,None,2,34,5,67],
                  "C":[67,54,33,None,77,98],
                  "D":[45,87,65,33,23,None]

})
print(df1)

print(df1.interpolate(method='linear',limit_direction='forward'))

print(df1.dropna())
```

Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

How to handle missing data in dataset? (Use Diabetes dataset & reference link)

B. Questions:

Mention types of data structures in Pandas.

Mention difference between Numpy and Pandas.

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. How to Handle Missing Data with Python (machinelearningmastery.com)
2. https://www.w3schools.com/python/pandas
3. https://www.geeksforgeeks.org/difference-between-pandas-vs-numpy/

IRIS DATASET

In [1]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Data Inspection

In [2]:
```
df.head(5)
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
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 [3]:
```
df.shape
```

Out[3]:
```
(150, 5)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [5]:
```
df["sepal_length"].nunique()
```

Out[5]:
```
35
```

In [6]:
```
df["sepal_length"].unique()
```

Out[6]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [7]:
```
# number of unique values alltogether
df.columns.nunique()
```

Out[7]:
```
5
```

In [8]:
```
# value counts
df['species'].value_counts()
```

Out[8]:

species	count
setosa	50
versicolor	50
virginica	50

dtype: int64

Dealing with NA values

In [9]:
```
# show null/NA values per column
df.isnull().sum()
```

Out[9]:

	0
sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

In [10]:
```
df.isnull().sum()*100/len(df)
```

Out[10]:

	0
sepal_length	0.0
sepal_width	0.0
petal_length	0.0
petal_width	0.0
species	0.0

dtype: float64

In [11]:
```
# drop all rows containing null
df.dropna()
```

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [12]:
```
# drop all columns containing null
df.dropna(axis=1)
```

Out[12]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [13]:
```
# drop columns with less than 5 NA values
df.dropna(axis=1, thresh=5)
```

Out[13]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [14]:
```
# replace all na values with -9999
df.fillna(-9999)
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [15]:
```
# fill na values with NaN
df.fillna(np.NaN)
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [16]:
```
# fill na values with strings
df.fillna("data missing")
```

Out[16]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [17]:
```
# fill missing values with mean column values
df.fillna(df.mean)
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Column Operation

In [18]:
```
# select a column
df["sepal_length"]
```

Out[18]:

	sepal_length
0	5.1
1	4.9
2	4.7
3	4.6
4	5.0
...	...
145	6.7
146	6.3
147	6.5
148	6.2
149	5.9

150 rows × 1 columns

dtype: float64

In [19]:
```
# select multiple columns and create a new dataframe X
X = df[["sepal_length", "sepal_width", "species"]]
X
```

Out[19]:

	sepal_length	sepal_width	species
0	5.1	3.5	setosa
1	4.9	3.0	setosa
2	4.7	3.2	setosa
3	4.6	3.1	setosa
4	5.0	3.6	setosa
...	...	...	...
145	6.7	3.0	virginica
146	6.3	2.5	virginica
147	6.5	3.0	virginica
148	6.2	3.4	virginica
149	5.9	3.0	virginica

150 rows × 3 columns

In [20]:
```
# select a column by column number
df.iloc[:, [1,3,4]]
```

Out[20]:

	sepal_width	petal_width	species
0	3.5	0.2	setosa
1	3.0	0.2	setosa
2	3.2	0.2	setosa
3	3.1	0.2	setosa
4	3.6	0.2	setosa
...	...	...	...
145	3.0	2.3	virginica
146	2.5	1.9	virginica
147	3.0	2.0	virginica
148	3.4	2.3	virginica
149	3.0	1.8	virginica

150 rows × 3 columns

In [21]:
```
# save all columns to a list
df.columns.tolist()
```

Out[21]:
```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

In [22]:
```
# sorting values by column "sepalW" in ascending order
df.sort_values(by = "sepal_width", ascending = True)
```

Out[22]:

	sepal_length	sepal_width	petal_length	petal_width	species
60	5.0	2.0	3.5	1.0	versicolor
62	6.0	2.2	4.0	1.0	versicolor
119	6.0	2.2	5.0	1.5	virginica
68	6.2	2.2	4.5	1.5	versicolor
41	4.5	2.3	1.3	0.3	setosa
...	...	...	...	...	...
16	5.4	3.9	1.3	0.4	setosa
14	5.8	4.0	1.2	0.2	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa

150 rows × 5 columns

In [23]:
```
# add new calculated column
df['newcol'] = df["sepal_length"]*2
df
```

Out[23]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	10.2
1	4.9	3.0	1.4	0.2	setosa	9.8
2	4.7	3.2	1.3	0.2	setosa	9.4
3	4.6	3.1	1.5	0.2	setosa	9.2
4	5.0	3.6	1.4	0.2	setosa	10.0
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	13.4
146	6.3	2.5	5.0	1.9	virginica	12.6
147	6.5	3.0	5.2	2.0	virginica	13.0
148	6.2	3.4	5.4	2.3	virginica	12.4
149	5.9	3.0	5.1	1.8	virginica	11.8

150 rows × 6 columns

In [24]:
```
# create a conditional calculated column
df['newcol'] = ["short" if i<3 else "long" for i in df["sepal_width"]]
df
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
1	4.9	3.0	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	long
146	6.3	2.5	5.0	1.9	virginica	short
147	6.5	3.0	5.2	2.0	virginica	long
148	6.2	3.4	5.4	2.3	virginica	long
149	5.9	3.0	5.1	1.8	virginica	long

150 rows × 6 columns

Row Operation (Sort, Filter, Slice)

In [25]:
```
# select rows 3 to 10
df.iloc[3:10,]
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
5	5.4	3.9	1.7	0.4	setosa	long
6	4.6	3.4	1.4	0.3	setosa	long
7	5.0	3.4	1.5	0.2	setosa	long
8	4.4	2.9	1.4	0.2	setosa	short
9	4.9	3.1	1.5	0.1	setosa	long

In [37]:
```
# select rows 3 to 15 and columns 1 to 3
df.iloc[3:15, 1:4]
```

Out[37]:

	sepal_width	petal_length	petal_width
3	3.1	1.5	0.2
4	3.6	1.4	0.2
5	3.9	1.7	0.4
6	3.4	1.4	0.3
7	3.4	1.5	0.2
8	2.9	1.4	0.2
9	3.1	1.5	0.1
10	3.7	1.5	0.2
11	3.4	1.6	0.2
12	3.0	1.4	0.1
13	3.0	1.1	0.1
14	4.0	1.2	0.2

In [27]:
```
# randomly select 10 rows
df.sample(10)
```

Out[27]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
53	5.5	2.3	4.0	1.3	versicolor	short
56	6.3	3.3	4.7	1.6	versicolor	long
77	6.7	3.0	5.0	1.7	versicolor	long
142	5.8	2.7	5.1	1.9	virginica	short
38	4.4	3.0	1.3	0.2	setosa	long
8	4.4	2.9	1.4	0.2	setosa	short
144	6.7	3.3	5.7	2.5	virginica	long
47	4.6	3.2	1.4	0.2	setosa	long
106	4.9	2.5	4.5	1.7	virginica	short
46	5.1	3.8	1.6	0.2	setosa	long

In [28]:
```
# find rows with specific strings
df[df["species"].isin(["setosa"])]
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
1	4.9	3.0	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
5	5.4	3.9	1.7	0.4	setosa	long
6	4.6	3.4	1.4	0.3	setosa	long
7	5.0	3.4	1.5	0.2	setosa	long
8	4.4	2.9	1.4	0.2	setosa	short
9	4.9	3.1	1.5	0.1	setosa	long
10	5.4	3.7	1.5	0.2	setosa	long
11	4.8	3.4	1.6	0.2	setosa	long
12	4.8	3.0	1.4	0.1	setosa	long
13	4.3	3.0	1.1	0.1	setosa	long
14	5.8	4.0	1.2	0.2	setosa	long
15	5.7	4.4	1.5	0.4	setosa	long
16	5.4	3.9	1.3	0.4	setosa	long
17	5.1	3.5	1.4	0.3	setosa	long
18	5.7	3.8	1.7	0.3	setosa	long
19	5.1	3.8	1.5	0.3	setosa	long
20	5.4	3.4	1.7	0.2	setosa	long
21	5.1	3.7	1.5	0.4	setosa	long
22	4.6	3.6	1.0	0.2	setosa	long
23	5.1	3.3	1.7	0.5	setosa	long
24	4.8	3.4	1.9	0.2	setosa	long
25	5.0	3.0	1.6	0.2	setosa	long
26	5.0	3.4	1.6	0.4	setosa	long
27	5.2	3.5	1.5	0.2	setosa	long
28	5.2	3.4	1.4	0.2	setosa	long
29	4.7	3.2	1.6	0.2	setosa	long
30	4.8	3.1	1.6	0.2	setosa	long
31	5.4	3.4	1.5	0.4	setosa	long
32	5.2	4.1	1.5	0.1	setosa	long
33	5.5	4.2	1.4	0.2	setosa	long
34	4.9	3.1	1.5	0.1	setosa	long
35	5.0	3.2	1.2	0.2	setosa	long
36	5.5	3.5	1.3	0.2	setosa	long
37	4.9	3.1	1.5	0.1	setosa	long
38	4.4	3.0	1.3	0.2	setosa	long
39	5.1	3.4	1.5	0.2	setosa	long
40	5.0	3.5	1.3	0.3	setosa	long
41	4.5	2.3	1.3	0.3	setosa	short
42	4.4	3.2	1.3	0.2	setosa	long
43	5.0	3.5	1.6	0.6	setosa	long
44	5.1	3.8	1.9	0.4	setosa	long
45	4.8	3.0	1.4	0.3	setosa	long
46	5.1	3.8	1.6	0.2	setosa	long
47	4.6	3.2	1.4	0.2	setosa	long
48	5.3	3.7	1.5	0.2	setosa	long
49	5.0	3.3	1.4	0.2	setosa	long

In [29]:
```
# conditional filtering
df[df.sepal_length >= 5]
```

Out[29]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
5	5.4	3.9	1.7	0.4	setosa	long
7	5.0	3.4	1.5	0.2	setosa	long
10	5.4	3.7	1.5	0.2	setosa	long
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	long
146	6.3	2.5	5.0	1.9	virginica	short
147	6.5	3.0	5.2	2.0	virginica	long
148	6.2	3.4	5.4	2.3	virginica	long
149	5.9	3.0	5.1	1.8	virginica	long

128 rows × 6 columns

In [30]:
```
df[df["petal_width"].isin([0.2])]
```

Out[30]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
1	4.9	3.0	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
6	4.6	3.4	1.4	0.3	setosa	long
7	5.0	3.4	1.5	0.2	setosa	long
8	4.4	2.9	1.4	0.2	setosa	short
10	5.4	3.7	1.5	0.2	setosa	long
11	4.8	3.4	1.6	0.2	setosa	long
14	5.8	4.0	1.2	0.2	setosa	long
17	5.1	3.5	1.4	0.3	setosa	long
18	5.7	3.8	1.7	0.3	setosa	long
19	5.1	3.8	1.5	0.3	setosa	long
20	5.4	3.4	1.7	0.2	setosa	long
22	4.6	3.6	1.0	0.2	setosa	long
24	4.8	3.4	1.9	0.2	setosa	long
25	5.0	3.0	1.6	0.2	setosa	long
27	5.2	3.5	1.5	0.2	setosa	long
28	5.2	3.4	1.4	0.2	setosa	long
29	4.7	3.2	1.6	0.2	setosa	long
30	4.8	3.1	1.6	0.2	setosa	long
33	5.5	4.2	1.4	0.2	setosa	long
35	5.0	3.2	1.2	0.2	setosa	long
36	5.5	3.5	1.3	0.2	setosa	long
38	4.4	3.0	1.3	0.2	setosa	long
39	5.1	3.4	1.5	0.2	setosa	long
40	5.0	3.5	1.3	0.3	setosa	long
41	4.5	2.3	1.3	0.3	setosa	short
42	4.4	3.2	1.3	0.2	setosa	long
45	4.8	3.0	1.4	0.3	setosa	long
46	5.1	3.8	1.6	0.2	setosa	long
47	4.6	3.2	1.4	0.2	setosa	long
48	5.3	3.7	1.5	0.2	setosa	long
49	5.0	3.3	1.4	0.2	setosa	long

In [31]:
```
# multi-conditional filtering
df[(df.petal_length > 1) & (df.species=="setosa") | (df.sepal_width < 3)]
```

Out[31]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
1	4.9	3.0	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
...	...	...	...	...	...	...
132	6.4	2.8	5.6	2.2	virginica	short
133	6.3	2.8	5.1	1.5	virginica	short
134	6.1	2.6	5.6	1.4	virginica	short
142	5.8	2.7	5.1	1.9	virginica	short
146	6.3	2.5	5.0	1.9	virginica	short

104 rows × 6 columns

In [32]:
```
# drop rows
df.drop(df.index[1]) # 1 is row index to be deleted
```

Out[32]:

	sepal_length	sepal_width	petal_length	petal_width	species	newcol
0	5.1	3.5	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
5	5.4	3.9	1.7	0.4	setosa	long
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	long
146	6.3	2.5	5.0	1.9	virginica	short
147	6.5	3.0	5.2	2.0	virginica	long
148	6.2	3.4	5.4	2.3	virginica	long
149	5.9	3.0	5.1	1.8	virginica	long

149 rows × 6 columns

Grouping

In [33]:
```
# data grouped by column "species"
X = df.groupby("species")
X
```

Out[33]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7963977f5b10>
```

In [34]:
```
# return mean values of a column ("sepal_length" ) grouped by "species" column
df.groupby("species")["sepal_length"].mean()
```

Out[34]:

species	sepal_length
setosa	5.006
versicolor	5.936
virginica	6.588

dtype: float64

In [38]:
```
# get counts in different categories
df.groupby("species").nunique()
```

Out[38]:

species	sepal_length	sepal_width	petal_length	petal_width	newcol
setosa	15	16	9	6	2
versicolor	21	14	19	9	2
virginica	21	13	20	12	2

Employee Dataset

In [5]:
```
import pandas as pd
import numpy as np
df = pd.read_excel("/content/employees.xlsx")

df
```

Out[5]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	NaN
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	NaN
...	...	...	...	...	...	...	...	...
995	Henry	NaN	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [4]:
```
df.describe()
```

Out[4]:

	Start Date	Salary	Bonus %	Senior Management
count	1000	1000.000000	999.000000	933.000000
mean	1998-12-12 21:50:24	90662.181000	10.198672	0.501608
min	1980-01-19 00:00:00	35013.000000	1.015000	0.000000
25%	1990-07-21 12:00:00	62613.000000	5.390500	0.000000
50%	1999-07-16 00:00:00	90428.000000	9.828000	1.000000
75%	2007-09-30 12:00:00	118740.250000	14.837000	1.000000
max	2016-07-15 00:00:00	149908.000000	19.944000	1.000000
std	NaN	32923.693342	5.524105	0.500266

In [6]:
```
pd.isnull(df['Team'])
```

Out[6]:

	Team
0	False
1	True
2	False
3	False
4	True
...	...
995	False
996	False
997	False
998	False
999	False

1000 rows × 1 columns

dtype: bool

In [7]:
```
pd.notnull(df['Team'])
```

Out[7]:

	Team
0	True
1	False
2	True
3	True
4	False
...	...
995	True
996	True
997	True
998	True
999	True

1000 rows × 1 columns

dtype: bool

In [8]:
```
df.fillna(1111)
```

Out[8]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	1111
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	1111
...	...	...	...	...	...	...	...	...
995	Henry	1111	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [9]:
```
df.fillna(method="pad")
```

```
<ipython-input-9-e5963cb4ed9a>:1: FutureWarning: DataFrame.fillna with 'method' is deprecated an
d will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method="pad")
```

Out[9]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	Marketing
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	Finance
...	...	...	...	...	...	...	...	...
995	Henry	Male	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [10]:
```
df.fillna(method='bfill')
```

```
<ipython-input-10-b823574c06e2>:1: FutureWarning: DataFrame.fillna with 'method' is deprecated a
nd will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='bfill')
```

Out[10]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	Finance
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	0
...	...	...	...	...	...	...	...	...
995	Henry	Male	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [12]:
```
df["Gender"].fillna('No Gender',inplace=True)
df
```

Out[12]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	NaN
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	NaN
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [13]:
```
df.replace(to_replace=np.nan,value="SFIT")
```

Out[13]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.17	1.0	SFIT
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.34	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	SFIT
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [14]:
```
df.interpolate(method="linear",limit_direction="forward")
```

```
<ipython-input-14-738f9fadd668>:1: FutureWarning: DataFrame.interpolate with object dtype is dep
recated and will raise in a future version. Call obj.infer_objects(copy=False) before interpolat
ing instead.
  df.interpolate(method="linear",limit_direction="forward")
```

Out[14]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	1993-08-06	12:42:00	97308	6.945	1.0	Marketing
1	Thomas	Male	1996-03-31	06:53:00	61933	4.170	1.0	NaN
2	Maria	Female	1993-04-23	11:17:00	130590	11.858	0.0	Finance
3	Jerry	Male	2005-03-04	13:00:00	138705	9.340	1.0	Finance
4	Larry	Male	1998-01-24	16:47:00	101004	1.389	1.0	NaN
...	...	...	...	...	...	...	...	...
995	Henry	No Gender	2014-11-23	06:09:00	132483	16.655	0.0	Distribution
996	Phillip	Male	1984-01-31	06:30:00	42392	19.675	0.0	Finance
997	Russell	Male	2013-05-20	12:39:00	96914	1.421	0.0	Product
998	Larry	Male	2013-04-20	16:45:00	60500	11.985	0.0	Business Development
999	Albert	Male	2012-05-15	18:24:00	129949	10.169	1.0	Sales

1000 rows × 8 columns

In [16]:
```
df1=pd.DataFrame({"A":[12,23,None,5,6,None],
"B":[34,None,2,34,5,67],
"C":[67,54,33,None,77,98],
"D":[45,87,65,33,23,None]
})
df1
```

Out[16]:

	A	B	C	D
0	12.0	34.0	67.0	45.0
1	23.0	NaN	54.0	87.0
2	NaN	2.0	33.0	65.0
3	5.0	34.0	NaN	33.0
4	6.0	5.0	77.0	23.0
5	NaN	67.0	98.0	NaN

In [17]:
```
df1.interpolate(method="linear",limit_direction="forward")
```

Out[17]:

	A	B	C	D
0	12.0	34.0	67.0	45.0
1	23.0	18.0	54.0	87.0
2	14.0	2.0	33.0	65.0
3	5.0	34.0	55.0	33.0
4	6.0	5.0	77.0	23.0
5	6.0	67.0	98.0	23.0

In [18]:
```
df1.dropna()
```

Out[18]:

	A	B	C	D
0	12.0	34.0	67.0	45.0
4	6.0	5.0	77.0	23.0

Handling Missing Value on Diabetes Dataset

In [13]:
```
# Import necessary libraries, loading the diabetes dataset and displaying info about dataset
import pandas as pd
import numpy as np
df = pd.read_csv("/content/diabetes.csv")
df.info()
```

```
<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
```

In [14]:
```
# Displaying the first five rows of the dataset
df.head()
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [15]:
```
# Replacing 0 values with NaN in specific columns
df_with_nan = df.iloc[:,[0,1,2,3,4,5,6,7,8]].replace(0,np.nan)
df_with_nan.head()
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	NaN	33.6	0.627	50	1.0
1	1.0	85.0	66.0	29.0	NaN	26.6	0.351	31	NaN
2	8.0	183.0	64.0	NaN	NaN	23.3	0.672	32	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	NaN
4	NaN	137.0	40.0	35.0	168.0	43.1	2.288	33	1.0

In [16]:
```
# Counting the number of missing values in each column
df_with_nan.isnull().sum()
```

Out[16]:

	0
Pregnancies	111
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	500

dtype: int64

In [17]:
```
# Calculating the percentage of missing values in each column
df_with_nan.isnull().sum()*100/len(df)
```

Out[17]:

	0
Pregnancies	14.453125
Glucose	0.651042
BloodPressure	4.557292
SkinThickness	29.557292
Insulin	48.697917
BMI	1.432292
DiabetesPedigreeFunction	0.000000
Age	0.000000
Outcome	65.104167

dtype: float64

In [18]:
```
# Filling missing values with -999
df_with_nan.fillna(-999)
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	-999.0	33.6	0.627	50	1.0
1	1.0	85.0	66.0	29.0	-999.0	26.6	0.351	31	-999.0
2	8.0	183.0	64.0	-999.0	-999.0	23.3	0.672	32	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	-999.0
4	-999.0	137.0	40.0	35.0	168.0	43.1	2.288	33	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63	-999.0
764	2.0	122.0	70.0	27.0	-999.0	36.8	0.340	27	-999.0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30	-999.0
766	1.0	126.0	60.0	-999.0	-999.0	30.1	0.349	47	1.0
767	1.0	93.0	70.0	31.0	-999.0	30.4	0.315	23	-999.0

768 rows × 9 columns

In [19]:
```
# Filling missing values with NaN
df_with_nan.fillna(np.nan)
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	NaN	33.6	0.627	50	1.0
1	1.0	85.0	66.0	29.0	NaN	26.6	0.351	31	NaN
2	8.0	183.0	64.0	NaN	NaN	23.3	0.672	32	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	NaN
4	NaN	137.0	40.0	35.0	168.0	43.1	2.288	33	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63	NaN
764	2.0	122.0	70.0	27.0	NaN	36.8	0.340	27	NaN
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30	NaN
766	1.0	126.0	60.0	NaN	NaN	30.1	0.349	47	1.0
767	1.0	93.0	70.0	31.0	NaN	30.4	0.315	23	NaN

768 rows × 9 columns

In [20]:
```
# Filling missing values with "Data Missing"
df_with_nan.fillna("Data Missing")
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	Data Missing	33.6	0.627	50	1.0
1	1.0	85.0	66.0	29.0	Data Missing	26.6	0.351	31	Data Missing
2	8.0	183.0	64.0	Data Missing	Data Missing	23.3	0.672	32	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	Data Missing
4	Data Missing	137.0	40.0	35.0	168.0	43.1	2.288	33	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63	Data Missing
764	2.0	122.0	70.0	27.0	Data Missing	36.8	0.340	27	Data Missing
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30	Data Missing
766	1.0	126.0	60.0	Data Missing	Data Missing	30.1	0.349	47	1.0
767	1.0	93.0	70.0	31.0	Data Missing	30.4	0.315	23	Data Missing

768 rows × 9 columns

In [21]:
```
# Dropping rows with missing values
df_with_nan.dropna(axis=0)
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	3.0	78.0	50.0	32.0	88.0	31.0	0.248	26	1.0
8	2.0	197.0	70.0	45.0	543.0	30.5	0.158	53	1.0
13	1.0	189.0	60.0	23.0	846.0	30.1	0.398	59	1.0
14	5.0	166.0	72.0	19.0	175.0	25.8	0.587	51	1.0
19	1.0	115.0	70.0	30.0	96.0	34.6	0.529	32	1.0
...	...	...	...	...	...	...	...	...	...
730	3.0	130.0	78.0	23.0	79.0	28.4	0.323	34	1.0
732	2.0	174.0	88.0	37.0	120.0	44.5	0.646	24	1.0
740	11.0	120.0	80.0	37.0	150.0	42.3	0.785	48	1.0
748	3.0	187.0	70.0	22.0	200.0	36.4	0.408	36	1.0
755	1.0	128.0	88.0	39.0	110.0	36.5	1.057	37	1.0

111 rows × 9 columns

In [22]:
```
# Dropping columns with missing values
df_with_nan.dropna(axis=1)
```

Out[22]:

	DiabetesPedigreeFunction	Age
0	0.627	50
1	0.351	31
2	0.672	32
3	0.167	21
4	2.288	33
...	...	...
763	0.171	63
764	0.340	27
765	0.245	30
766	0.349	47
767	0.315	23

768 rows × 2 columns

In [24]:
```
# Dropping columns with less than 300 non-missing values
df_with_nan.dropna(axis=1, thresh =300)
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6.0	148.0	72.0	35.0	NaN	33.6	0.627	50
1	1.0	85.0	66.0	29.0	NaN	26.6	0.351	31
2	8.0	183.0	64.0	NaN	NaN	23.3	0.672	32
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21
4	NaN	137.0	40.0	35.0	168.0	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63
764	2.0	122.0	70.0	27.0	NaN	36.8	0.340	27
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30
766	1.0	126.0	60.0	NaN	NaN	30.1	0.349	47
767	1.0	93.0	70.0	31.0	NaN	30.4	0.315	23

768 rows × 8 columns

In [25]:
```
# Filling missing values with the mean of each column
df_with_nan.fillna(df_with_nan.mean())
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.000000	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1.0
1	1.000000	85.0	66.0	29.00000	155.548223	26.6	0.351	31	1.0
2	8.000000	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1.0
3	1.000000	89.0	66.0	23.00000	94.000000	28.1	0.167	21	1.0
4	4.494673	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1.0
...	...	...	...	...	...	...	...	...	...
763	10.000000	101.0	76.0	48.00000	180.000000	32.9	0.171	63	1.0
764	2.000000	122.0	70.0	27.00000	155.548223	36.8	0.340	27	1.0
765	5.000000	121.0	72.0	23.00000	112.000000	26.2	0.245	30	1.0
766	1.000000	126.0	60.0	29.15342	155.548223	30.1	0.349	47	1.0
767	1.000000	93.0	70.0	31.00000	155.548223	30.4	0.315	23	1.0

768 rows × 9 columns