St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2021-2022
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()
```

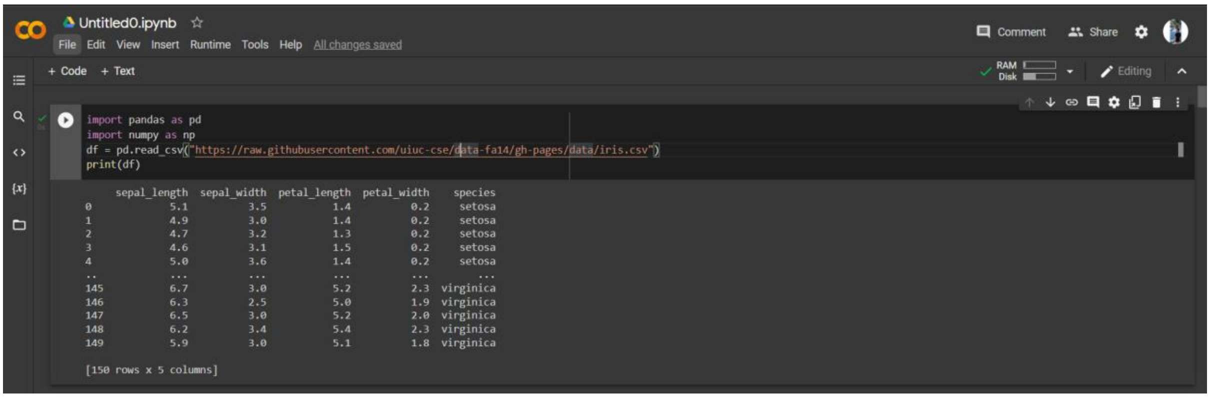
Employee Dataset:

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())
```

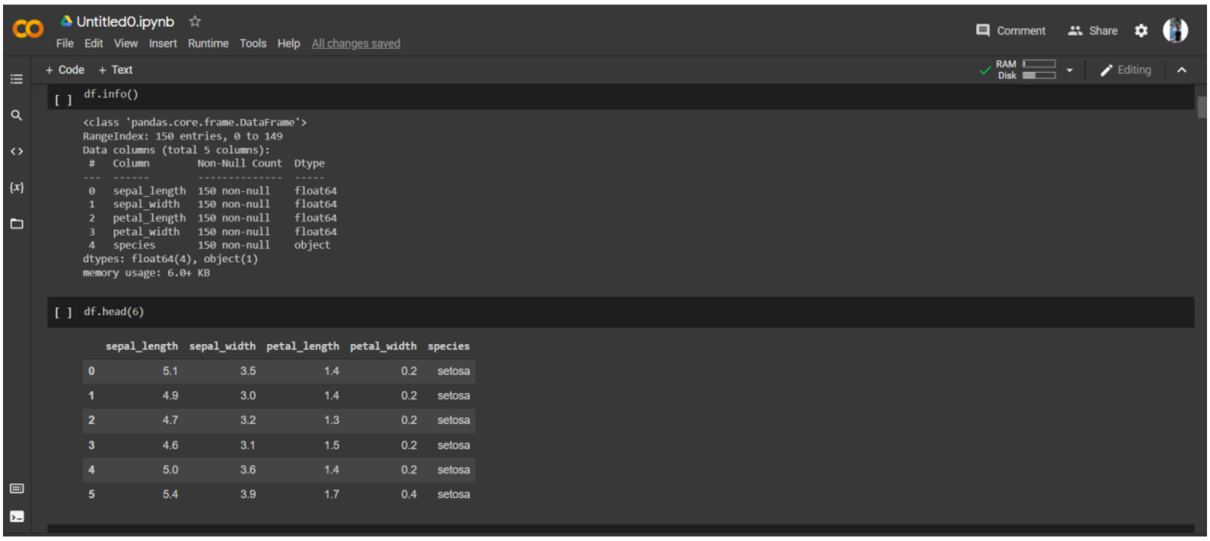
Paste Screenshots of above commands.

Iris Dataset:



```
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")
print(df)
```

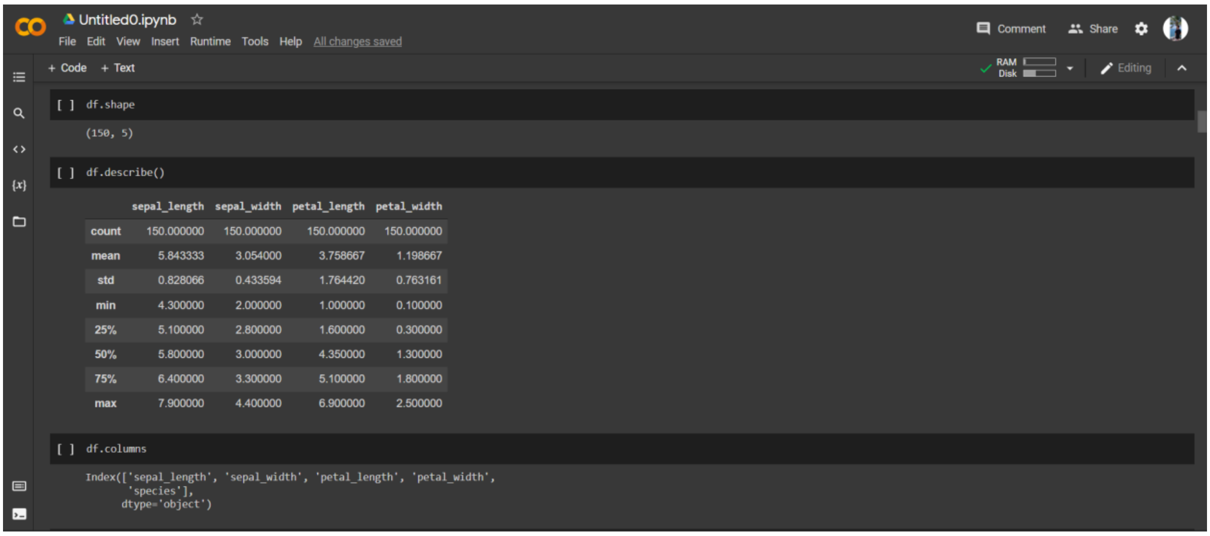
```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          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 x 5 columns]
```



```
[ ] 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
```

```
[ ] df.head(6)
```

	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
5	5.4	3.9	1.7	0.4	setosa

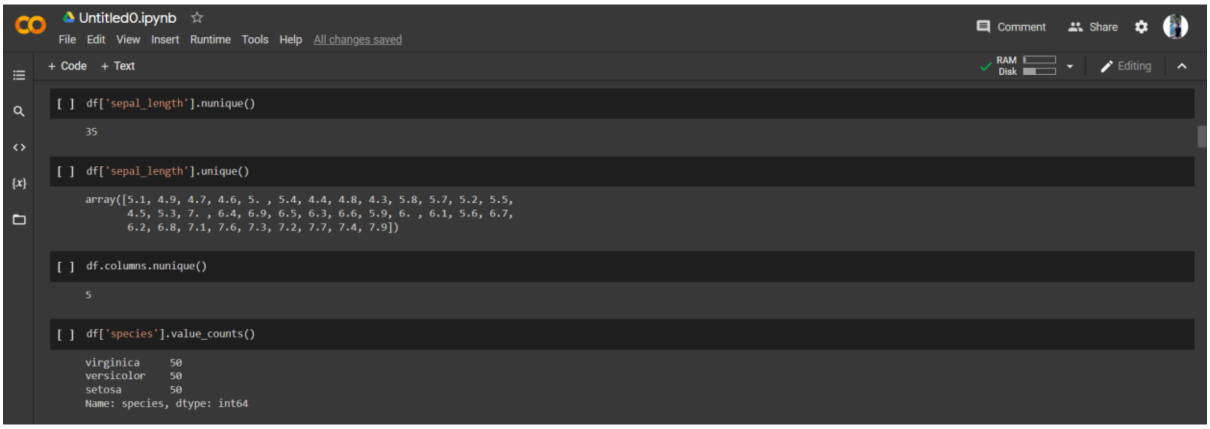


```
[ ] df.shape

(150, 5)
```

```
[ ] df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
[ ] df.columns

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

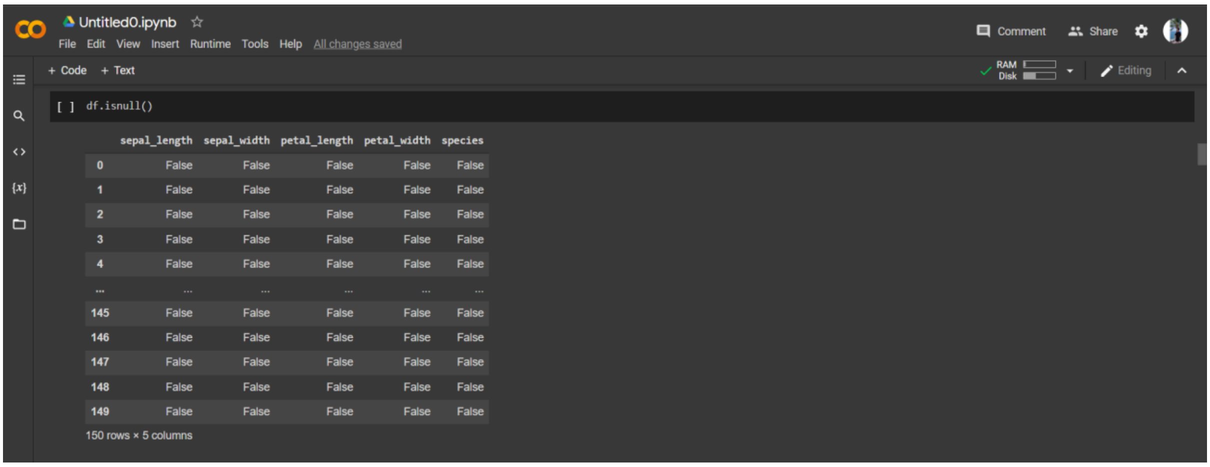


```
[ ] df['sepal_length'].nunique()

35
```

```
[ ] df['sepal_length'].unique()

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])
```

```
[ ] df.columns.nunique()

5
```

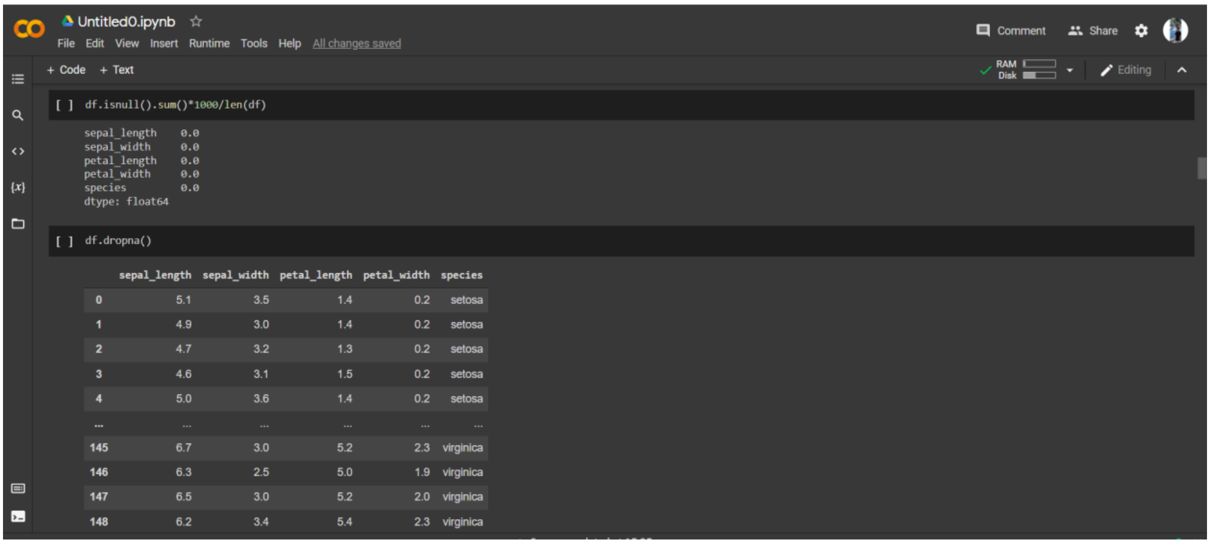
```
[ ] df['species'].value_counts()

virginica     50
versicolor    50
setosa        50
Name: species, dtype: int64
```



```
[ ] df.isnull()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

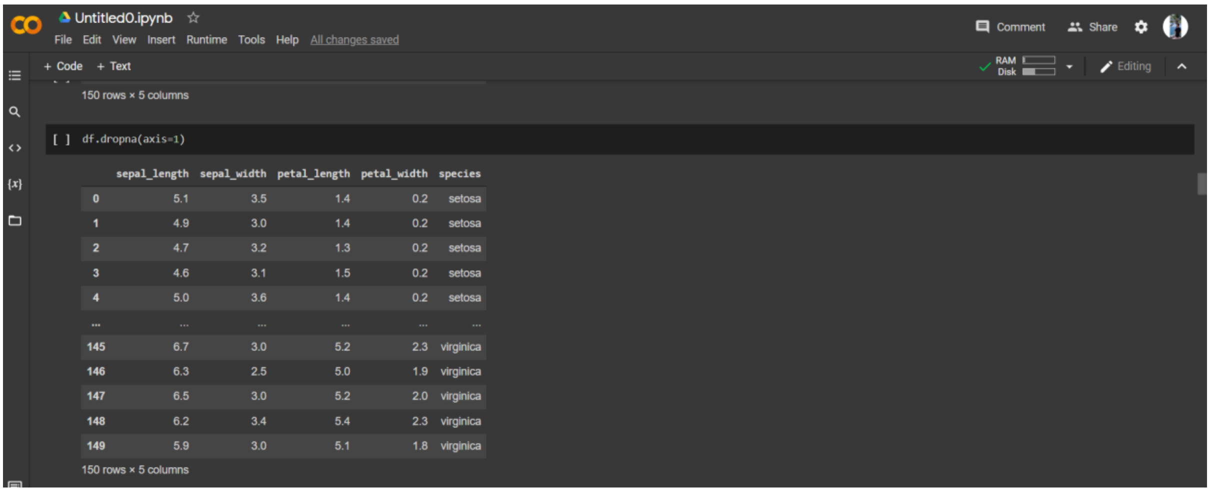
150 rows × 5 columns



```
[ ] df.isnull().sum()*1000/len(df)

sepal_length    0.0
sepal_width     0.0
petal_length    0.0
petal_width     0.0
species         0.0
dtype: float64
```

```
[ ] df.dropna()
```

	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

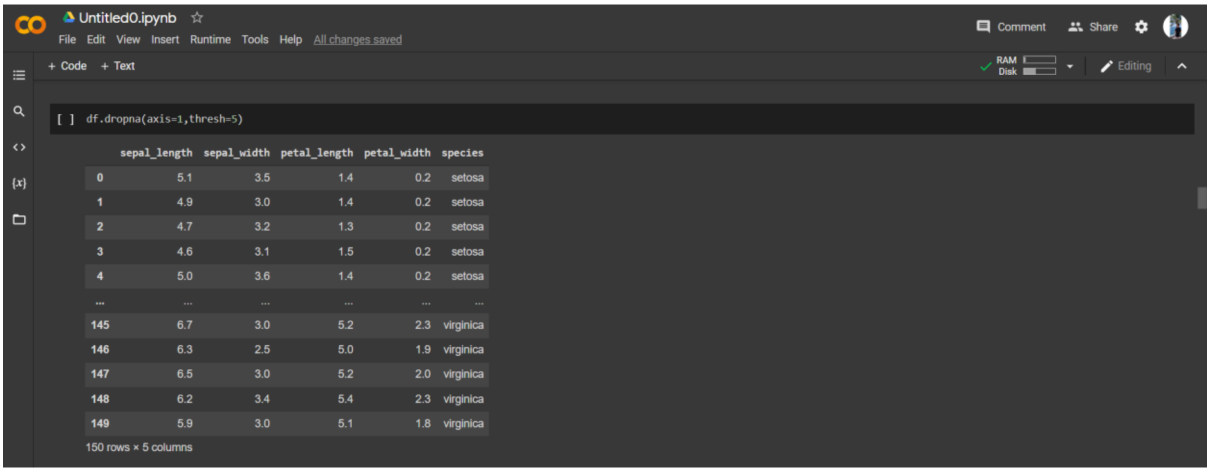


150 rows × 5 columns

```
[ ] df.dropna(axis=1)
```

	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

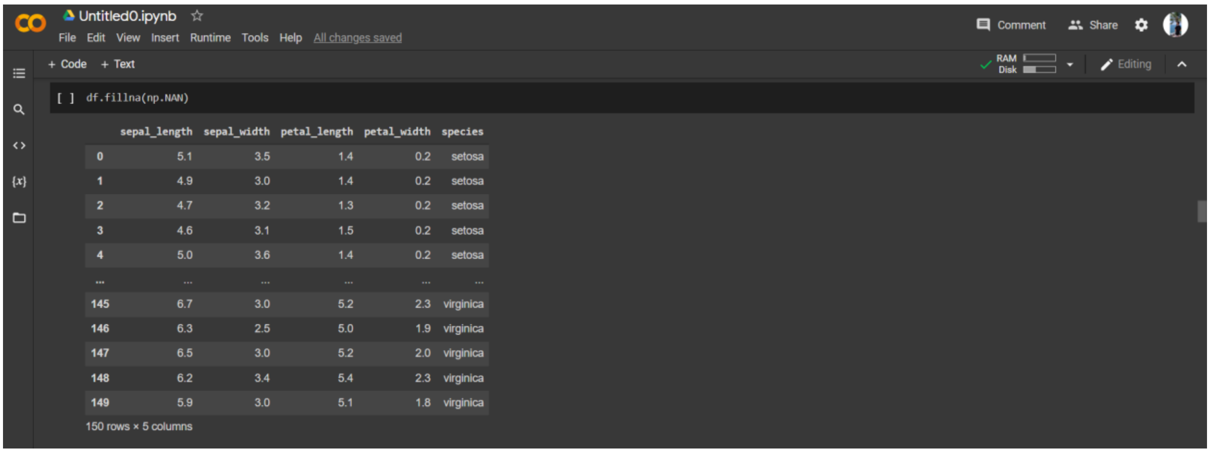
150 rows × 5 columns



```
[ ] df.dropna(axis=1,thresh=5)
```

	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

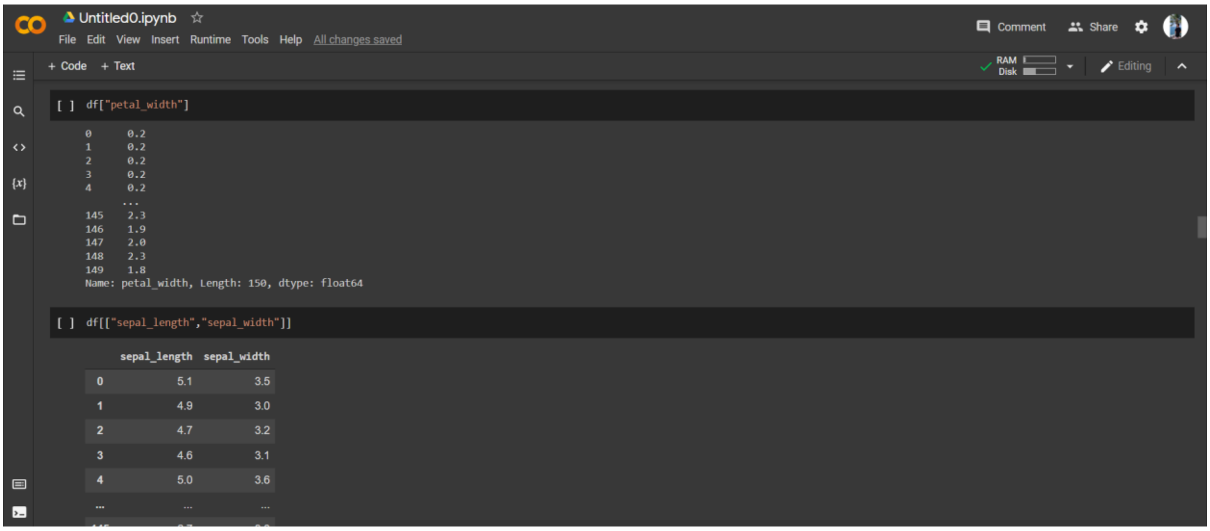
150 rows × 5 columns



```
[ ] df.fillna(np.NAN)
```

	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

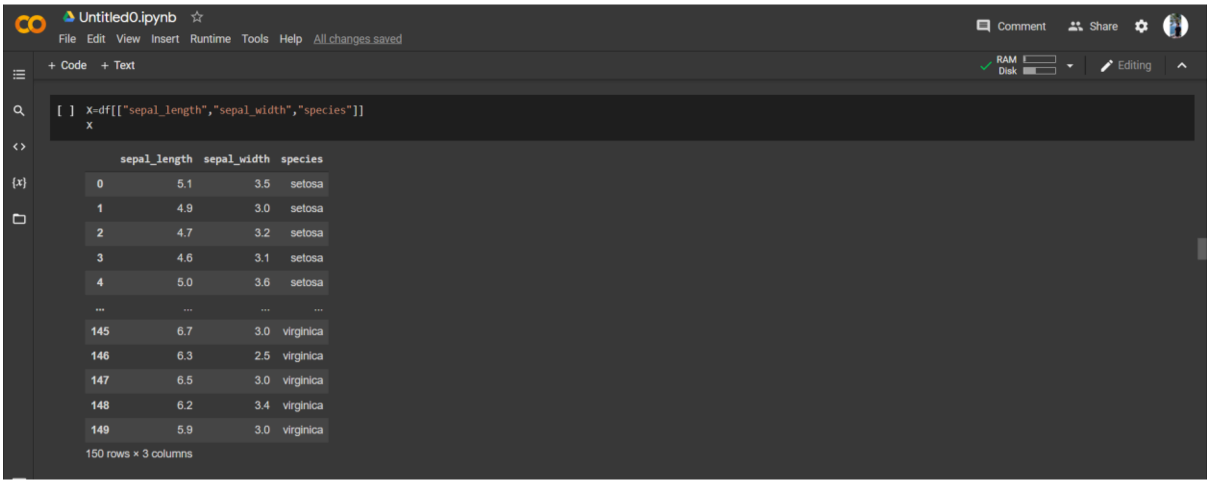
150 rows × 5 columns



```
[ ] df["petal_width"]

0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petal_width, Length: 150, dtype: float64
```

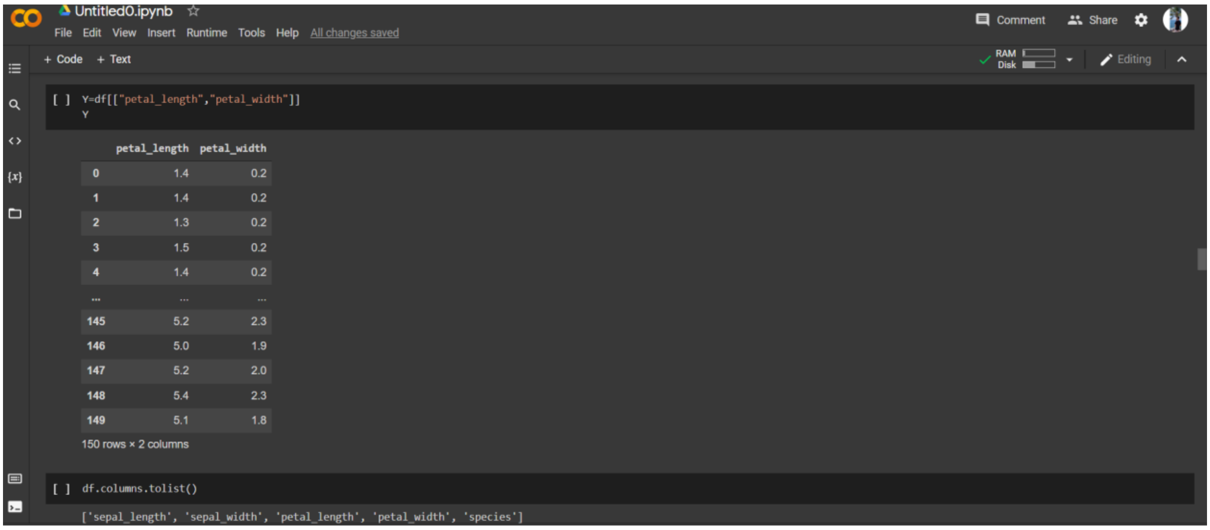
```
[ ] df[["sepal_length","sepal_width"]]
```

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6
...	...	...



```
[ ] X=df[["sepal_length","sepal_width","species"]]
    X
```

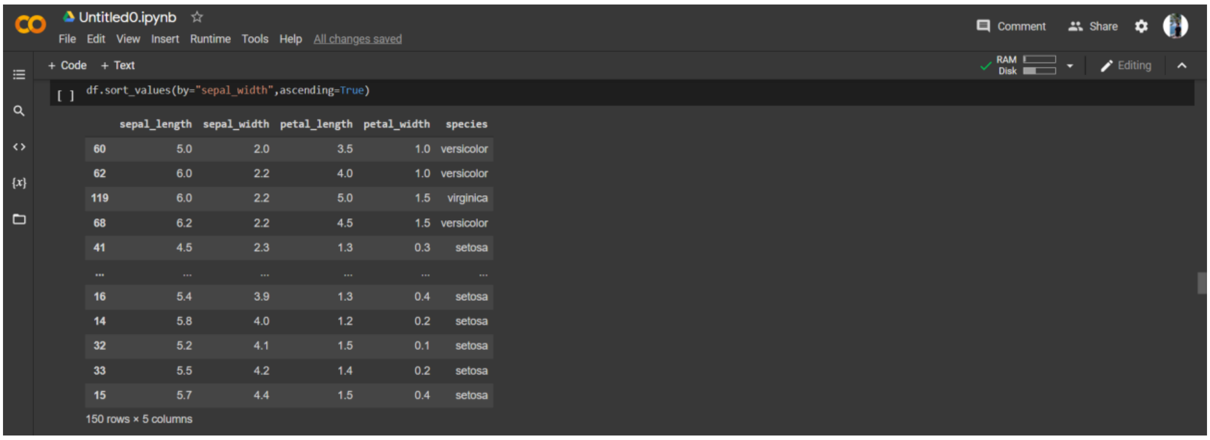
```
     sepal_length  sepal_width    species
0             5.1          3.5     setosa
1             4.9          3.0     setosa
2             4.7          3.2     setosa
3             4.6          3.1     setosa
4             5.0          3.6     setosa
...           ...          ...        ...
145           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
```



```
[ ] Y=df[["petal_length","petal_width"]]
    Y
```

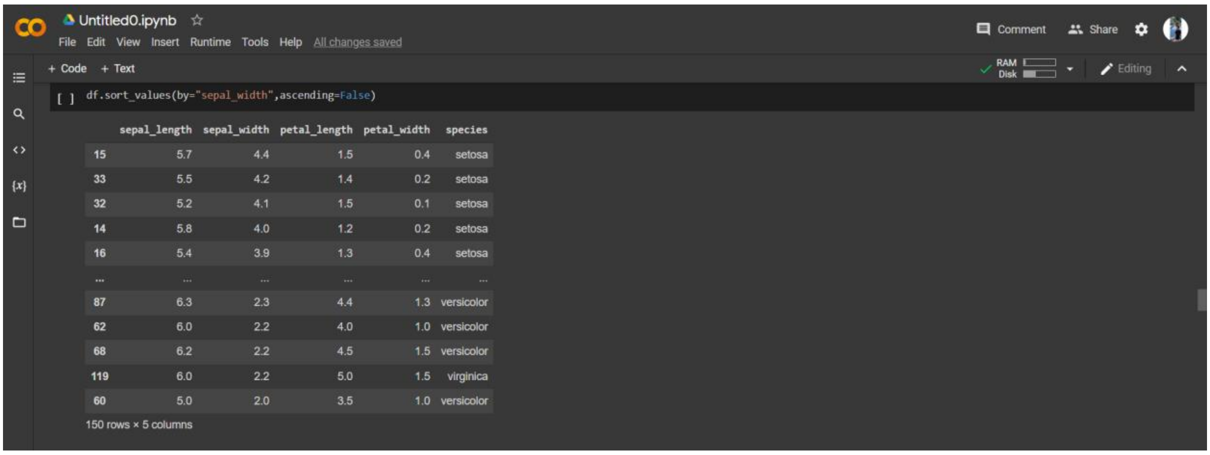
```
     petal_length  petal_width
0             1.4          0.2
1             1.4          0.2
2             1.3          0.2
3             1.5          0.2
4             1.4          0.2
...           ...          ...
145           5.2          2.3
146           5.0          1.9
147           5.2          2.0
148           5.4          2.3
149           5.1          1.8
150 rows × 2 columns
```

```
[ ] df.columns.tolist()

    ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```



```
[ ] df.sort_values(by="sepal_width",ascending=True)
```

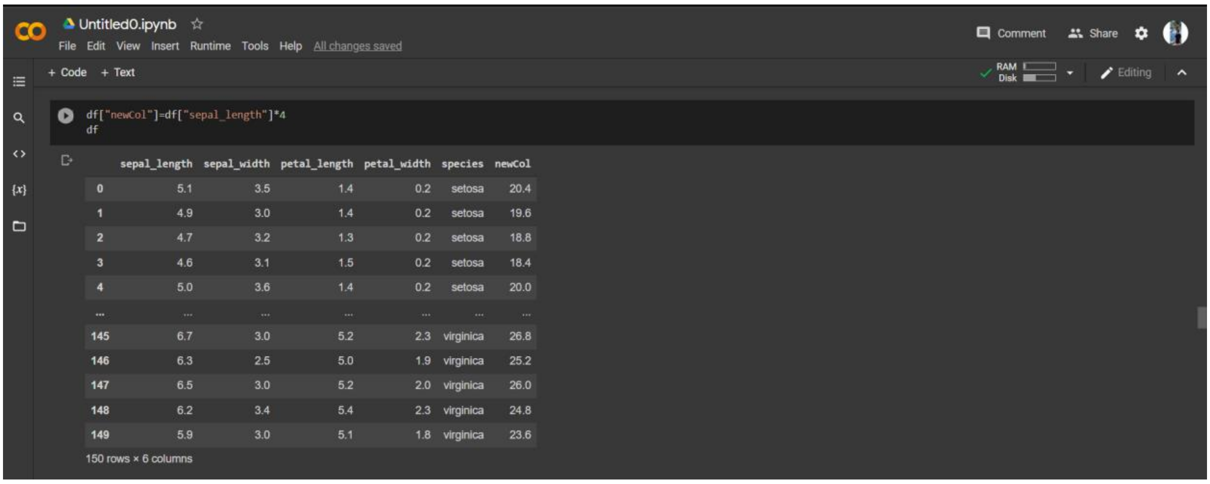
```
     sepal_length  sepal_width  petal_length  petal_width     species
60            5.0          2.0           3.5          1.0  versicolor
62            6.0          2.2           4.0          1.0  versicolor
119           6.0          2.2           5.0          1.5   virginica
68            6.2          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
```



```
df.sort_values(by="sepal_width",ascending=False)
```

	sepal_length	sepal_width	petal_length	petal_width	species
15	5.7	4.4	1.5	0.4	setosa
33	5.5	4.2	1.4	0.2	setosa
32	5.2	4.1	1.5	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa
16	5.4	3.9	1.3	0.4	setosa
...	...	...	...	...	...
87	6.3	2.3	4.4	1.3	versicolor
62	6.0	2.2	4.0	1.0	versicolor
68	6.2	2.2	4.5	1.5	versicolor
119	6.0	2.2	5.0	1.5	virginica
60	5.0	2.0	3.5	1.0	versicolor

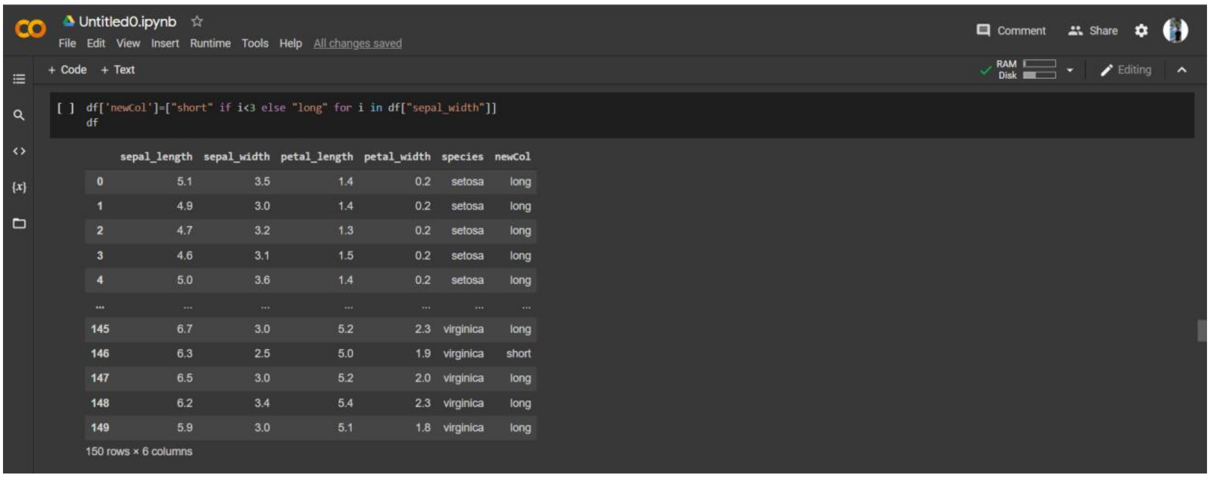
150 rows × 5 columns



```
df["newCol"]=df["sepal_length"]*4
df
```

	sepal_length	sepal_width	petal_length	petal_width	species	newCol
0	5.1	3.5	1.4	0.2	setosa	20.4
1	4.9	3.0	1.4	0.2	setosa	19.6
2	4.7	3.2	1.3	0.2	setosa	18.8
3	4.6	3.1	1.5	0.2	setosa	18.4
4	5.0	3.6	1.4	0.2	setosa	20.0
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	26.8
146	6.3	2.5	5.0	1.9	virginica	25.2
147	6.5	3.0	5.2	2.0	virginica	26.0
148	6.2	3.4	5.4	2.3	virginica	24.8
149	5.9	3.0	5.1	1.8	virginica	23.6

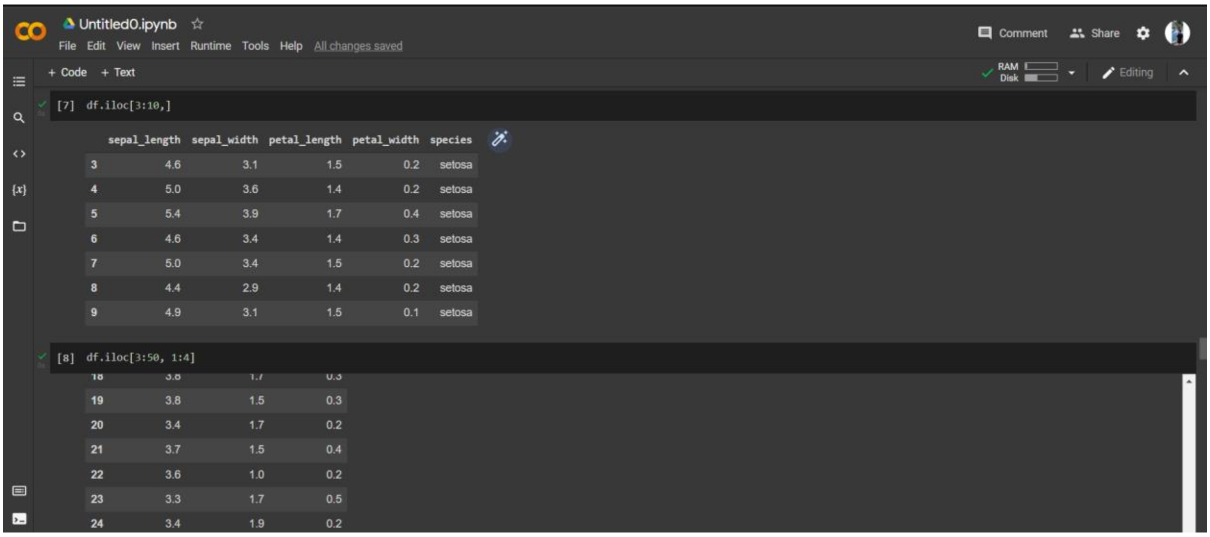
150 rows × 6 columns



```
df['newCol']=["short" if i<3 else "long" for i in df["sepal_width"]]
df
```

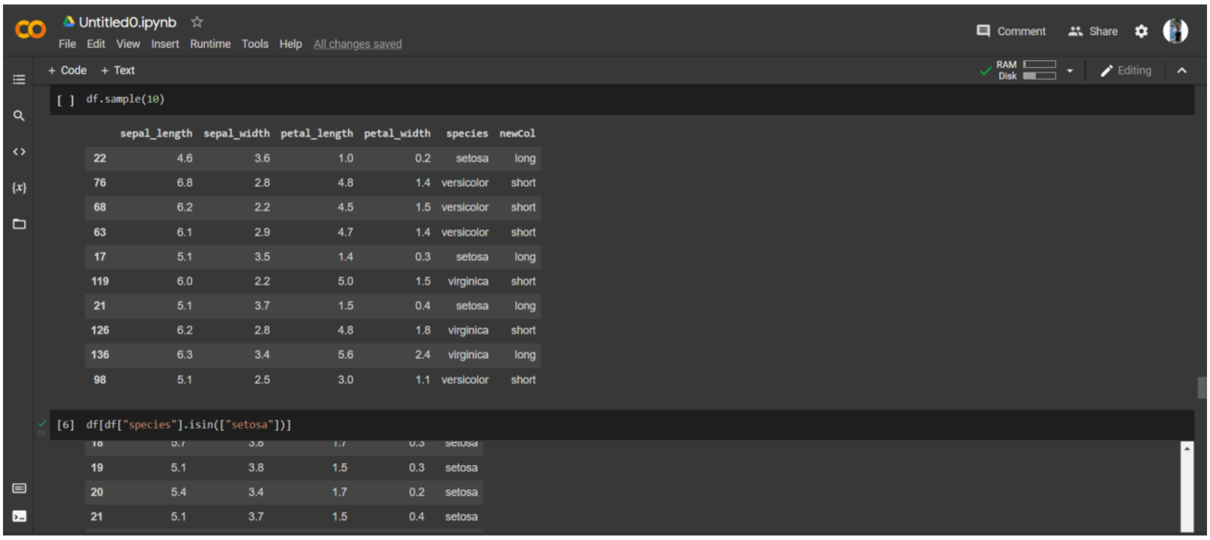
	sepal_length	sepal_width	petal_length	petal_width	species	newCol
0	5.1	3.5	1.4	0.2	setosa	long
1	4.9	3.0	1.4	0.2	setosa	long
2	4.7	3.2	1.3	0.2	setosa	long
3	4.6	3.1	1.5	0.2	setosa	long
4	5.0	3.6	1.4	0.2	setosa	long
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica	long
146	6.3	2.5	5.0	1.9	virginica	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



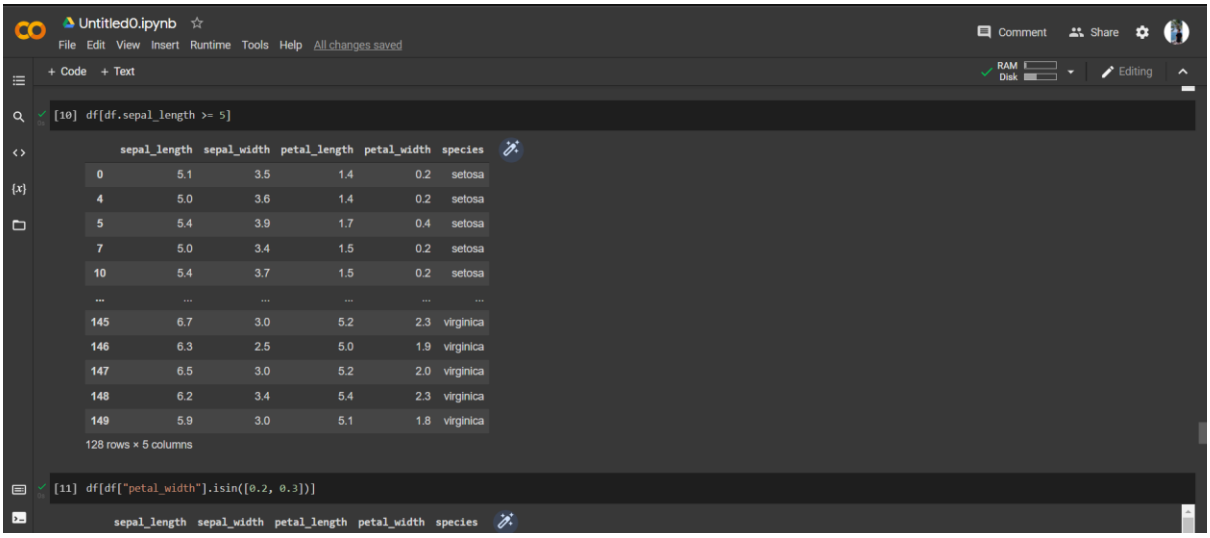
```
[7] df.iloc[3:10,]

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

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

18   3.8   1.7   0.3
19   3.8   1.5   0.3
20   3.4   1.7   0.2
21   3.7   1.5   0.4
22   3.6   1.0   0.2
23   3.3   1.7   0.5
24   3.4   1.9   0.2
```



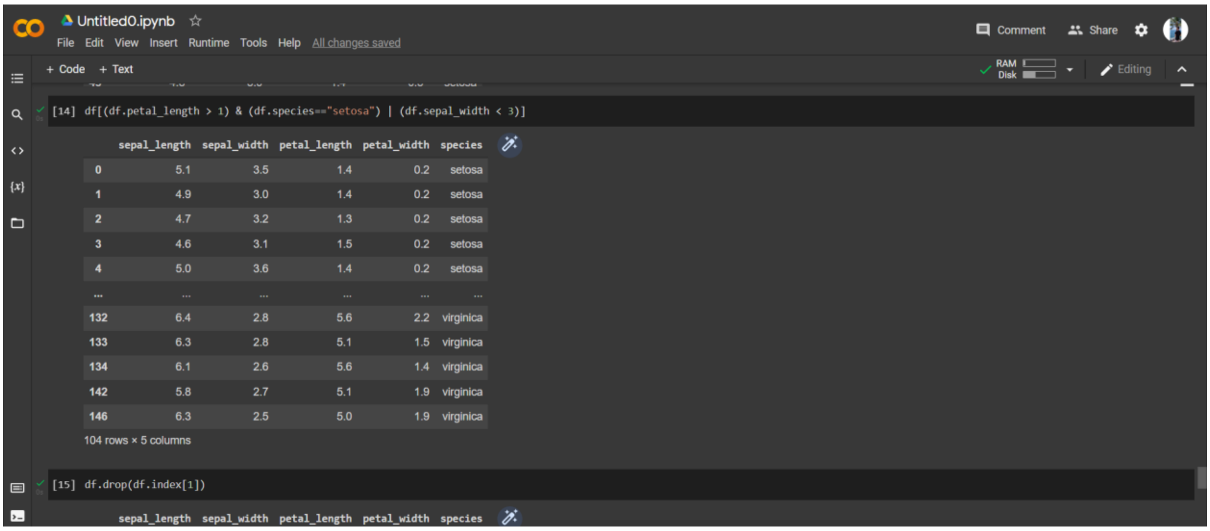
```
[ ] df.sample(10)

     sepal_length  sepal_width  petal_length  petal_width     species  newCol
22            4.6          3.6           1.0          0.2      setosa    long
76            6.8          2.8           4.8          1.4  versicolor   short
68            6.2          2.2           4.5          1.5  versicolor   short
63            6.1          2.9           4.7          1.4  versicolor   short
17            5.1          3.5           1.4          0.3      setosa    long
119           6.0          2.2           5.0          1.5   virginica   short
21            5.1          3.7           1.5          0.4      setosa    long
126           6.2          2.8           4.8          1.8   virginica   short
136           6.3          3.4           5.6          2.4   virginica    long
98            5.1          2.5           3.0          1.1  versicolor   short
```

```
[6] df[df["species"].isin(["setosa"])]

18   5.7   3.8   1.7   0.3   setosa
19   5.1   3.8   1.5   0.3   setosa
20   5.4   3.4   1.7   0.2   setosa
21   5.1   3.7   1.5   0.4   setosa
```



```
[10] df[df.sepal_length >= 5]

     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
5             5.4          3.9           1.7          0.4     setosa
7             5.0          3.4           1.5          0.2     setosa
10            5.4          3.7           1.5          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

128 rows × 5 columns
```

```
[11] df[df["petal_width"].isin([0.2, 0.3])]

   sepal_length  sepal_width  petal_length  petal_width  species
```



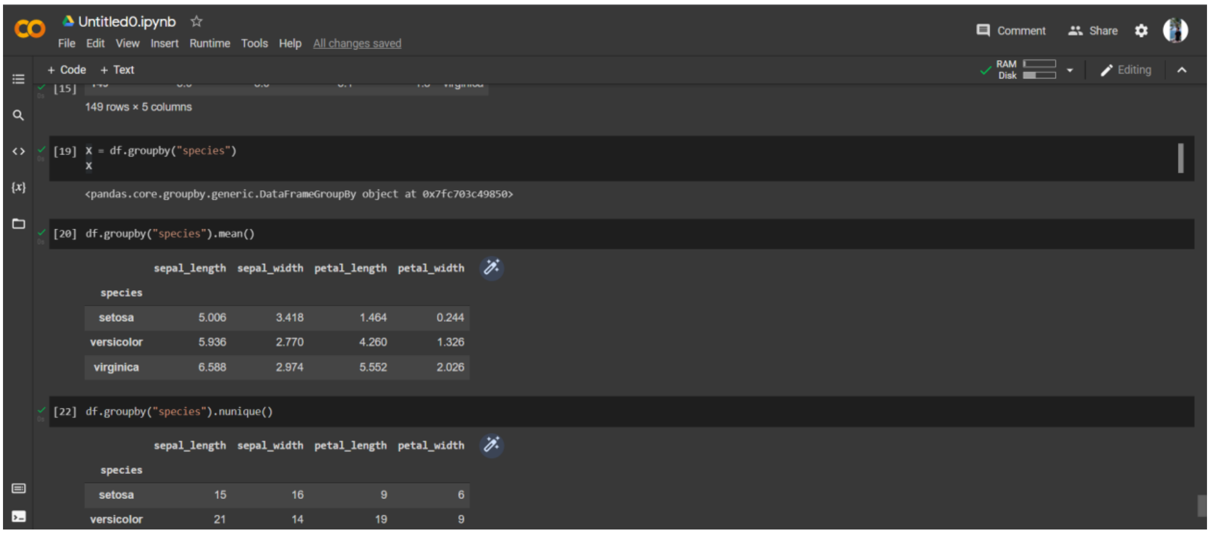
```
[14] df[(df.petal_length > 1) & (df.species=="setosa") | (df.sepal_width < 3)]
```

	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
...	...	...	...	...	...
132	6.4	2.8	5.6	2.2	virginica
133	6.3	2.8	5.1	1.5	virginica
134	6.1	2.6	5.6	1.4	virginica
142	5.8	2.7	5.1	1.9	virginica
146	6.3	2.5	5.0	1.9	virginica

104 rows × 5 columns

```
[15] df.drop(df.index[1])
```

sepal_length	sepal_width	petal_length	petal_width	species



149 rows × 5 columns

```
[19] X = df.groupby("species")
     X
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fc703c49850>
```

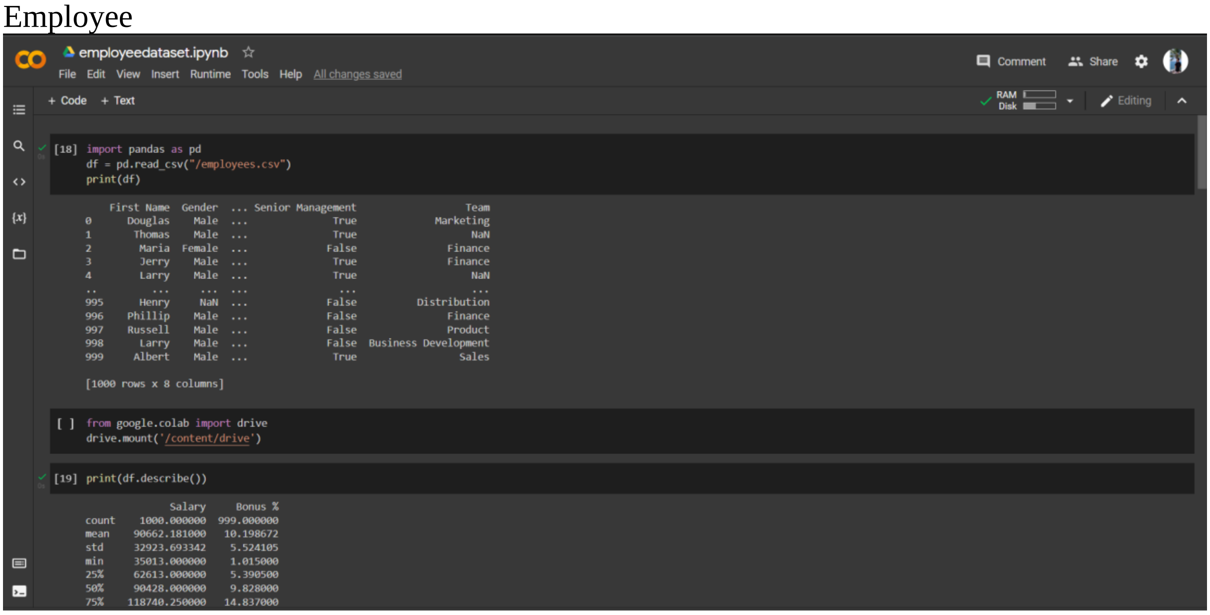
```
[20] df.groupby("species").mean()
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	5.006	3.418	1.464	0.244
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

```
[22] df.groupby("species").nunique()
```

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	15	16	9	6
versicolor	21	14	19	9

Employee



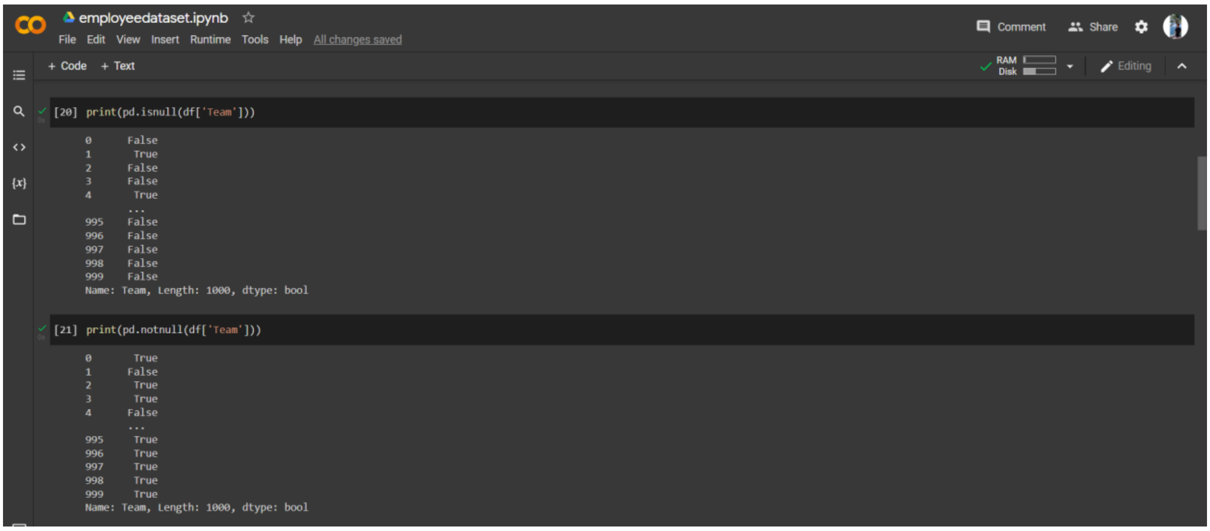
```
[18] import pandas as pd
     df = pd.read_csv("/employees.csv")
     print(df)
```

```
    First Name  Gender  ... Senior Management                 Team
0      Douglas    Male  ...              True            Marketing
1       Thomas    Male  ...              True                  NaN
2        Maria  Female  ...             False              Finance
3        Jerry    Male  ...              True              Finance
4        Larry    Male  ...              True                  NaN
..         ...     ...  ...               ...                  ...
995      Henry     NaN  ...             False         Distribution
996    Phillip    Male  ...             False              Finance
997    Russell    Male  ...             False              Product
998      Larry    Male  ...             False Business Development
999     Albert    Male  ...              True                Sales

[1000 rows x 8 columns]
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

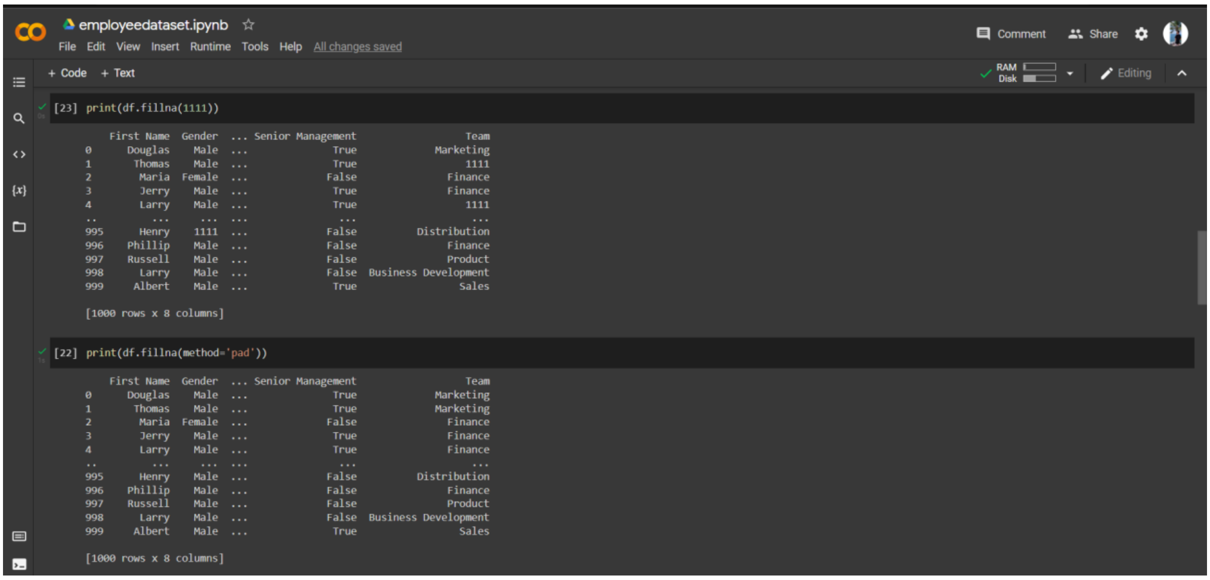
```
[19] print(df.describe())
```

```
              Salary     Bonus %
count    1000.000000  999.000000
mean    90662.181000   10.198672
std     32923.693342    5.524105
min     35013.000000    1.015000
25%     62613.000000    5.390500
50%     90428.000000    9.828000
75%    118740.250000   14.837000
```



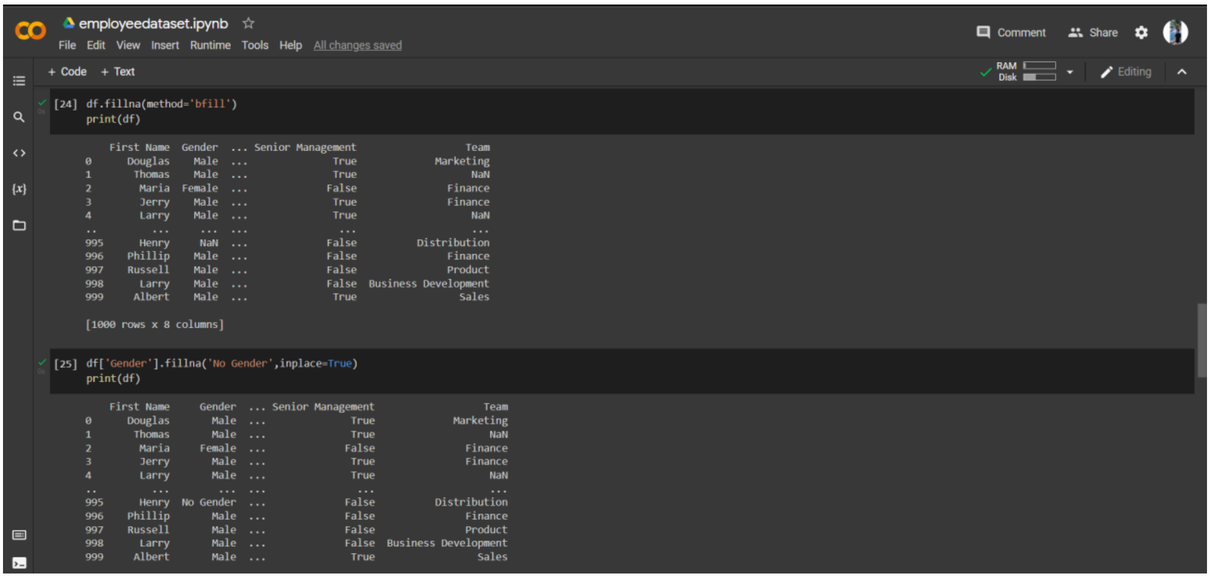
```
[20] print(pd.isnull(df['Team']))

     0      False
     1       True
     2      False
     3      False
     4       True
            ...
     995    False
     996    False
     997    False
     998    False
     999    False
     Name: Team, Length: 1000, dtype: bool
```

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

     0       True
     1      False
     2       True
     3       True
     4      False
            ...
     995     True
     996     True
     997     True
     998     True
     999     True
     Name: Team, Length: 1000, dtype: bool
```



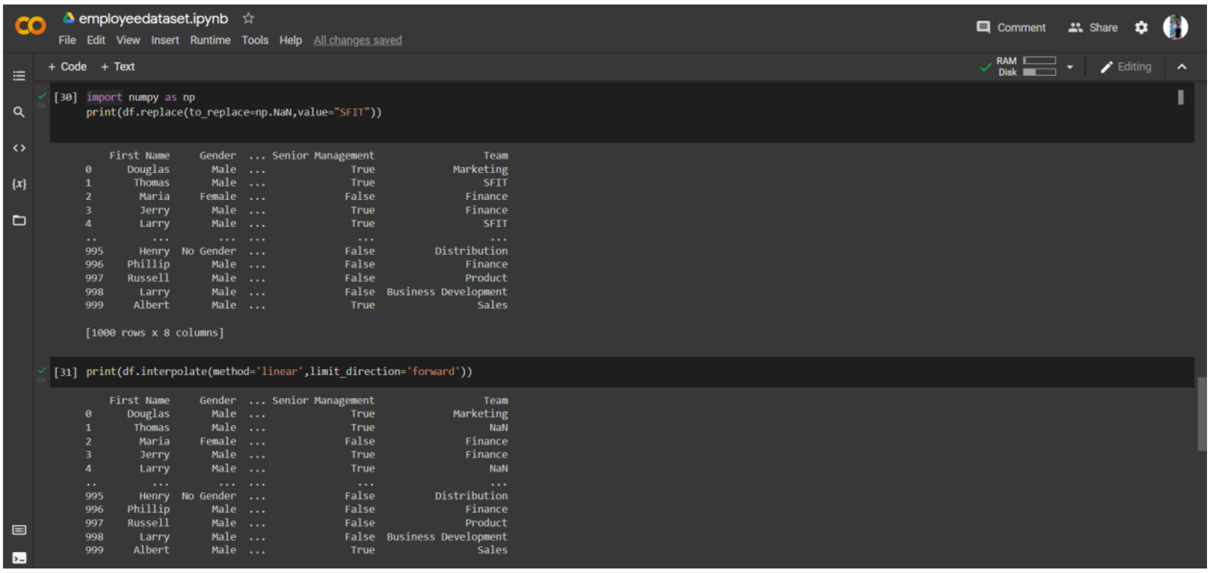
```
[23] print(df.fillna(1111))

        First Name  Gender  ... Senior Management                 Team
     0     Douglas    Male  ...              True            Marketing
     1      Thomas    Male  ...              True                 1111
     2       Maria  Female  ...             False              Finance
     3       Jerry    Male  ...              True              Finance
     4       Larry    Male  ...              True                 1111
     ..        ...     ...  ...               ...                  ...
     995     Henry    1111  ...             False         Distribution
     996   Phillip    Male  ...             False              Finance
     997   Russell    Male  ...             False              Product
     998     Larry    Male  ...             False  Business Development
     999    Albert    Male  ...              True                Sales

     [1000 rows x 8 columns]
```

```
[22] print(df.fillna(method='pad'))

        First Name  Gender  ... Senior Management                 Team
     0     Douglas    Male  ...              True            Marketing
     1      Thomas    Male  ...              True            Marketing
     2       Maria  Female  ...             False              Finance
     3       Jerry    Male  ...              True              Finance
     4       Larry    Male  ...              True              Finance
     ..        ...     ...  ...               ...                  ...
     995     Henry    Male  ...             False         Distribution
     996   Phillip    Male  ...             False              Finance
     997   Russell    Male  ...             False              Product
     998     Larry    Male  ...             False  Business Development
     999    Albert    Male  ...              True                Sales

     [1000 rows x 8 columns]
```



```
[24] df.fillna(method='bfill')
     print(df)

        First Name  Gender  ... Senior Management                 Team
     0     Douglas    Male  ...              True            Marketing
     1      Thomas    Male  ...              True                  NaN
     2       Maria  Female  ...             False              Finance
     3       Jerry    Male  ...              True              Finance
     4       Larry    Male  ...              True                  NaN
     ..        ...     ...  ...               ...                  ...
     995     Henry     NaN  ...             False         Distribution
     996   Phillip    Male  ...             False              Finance
     997   Russell    Male  ...             False              Product
     998     Larry    Male  ...             False  Business Development
     999    Albert    Male  ...              True                Sales

     [1000 rows x 8 columns]
```

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

        First Name     Gender  ... Senior Management                 Team
     0     Douglas       Male  ...              True            Marketing
     1      Thomas       Male  ...              True                  NaN
     2       Maria     Female  ...             False              Finance
     3       Jerry       Male  ...              True              Finance
     4       Larry       Male  ...              True                  NaN
     ..        ...        ...  ...               ...                  ...
     995     Henry  No Gender  ...             False         Distribution
     996   Phillip       Male  ...             False              Finance
     997   Russell       Male  ...             False              Product
     998     Larry       Male  ...             False  Business Development
     999    Albert       Male  ...              True                Sales
```

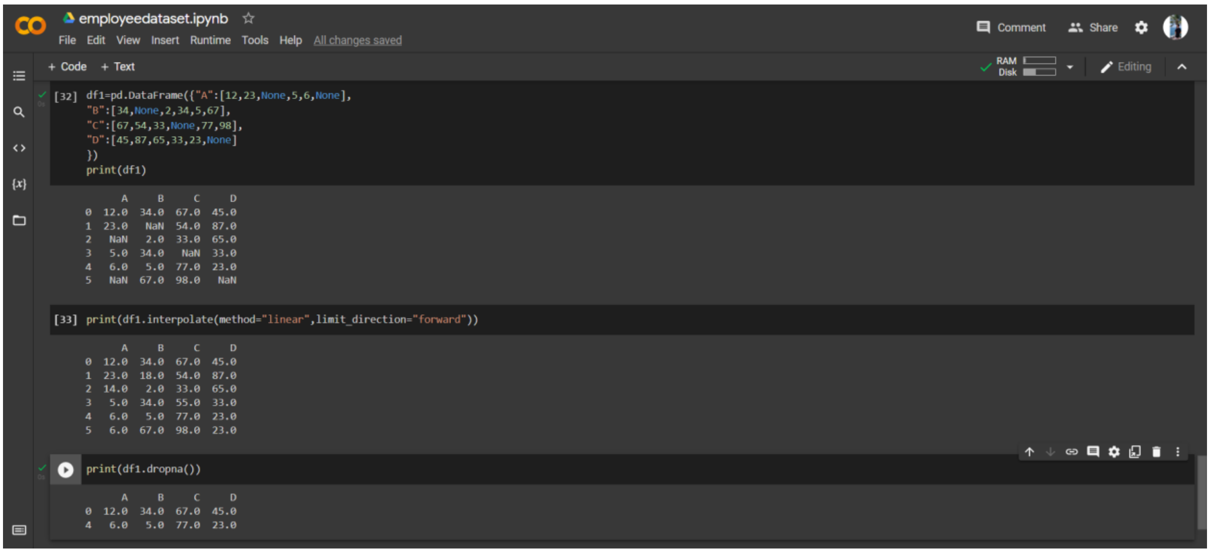


```
[30] import numpy as np
     print(df.replace(to_replace=np.NaN,value="SFIT"))
```

```
      First Name     Gender  ... Senior Management                 Team
0        Douglas       Male  ...              True            Marketing
1         Thomas       Male  ...              True                 SFIT
2          Maria     Female  ...             False              Finance
3          Jerry       Male  ...              True              Finance
4          Larry       Male  ...              True                 SFIT
..           ...        ...  ...               ...                  ...
995        Henry  No Gender  ...             False         Distribution
996      Phillip       Male  ...             False              Finance
997      Russell       Male  ...             False              Product
998        Larry       Male  ...             False  Business Development
999       Albert       Male  ...              True                Sales

[1000 rows x 8 columns]
```

```
[31] print(df.interpolate(method='linear',limit_direction='forward'))
```

```
      First Name     Gender  ... Senior Management                 Team
0        Douglas       Male  ...              True            Marketing
1         Thomas       Male  ...              True                  NaN
2          Maria     Female  ...             False              Finance
3          Jerry       Male  ...              True              Finance
4          Larry       Male  ...              True                  NaN
..           ...        ...  ...               ...                  ...
995        Henry  No Gender  ...             False         Distribution
996      Phillip       Male  ...             False              Finance
997      Russell       Male  ...             False              Product
998        Larry       Male  ...             False  Business Development
999       Albert       Male  ...              True                Sales
```



```
[32] 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)
```

```
      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
```

```
[33] print(df1.interpolate(method="linear",limit_direction="forward"))
```

```
      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
```

```
print(df1.dropna())
```

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

8. Post-Experiments Exercise

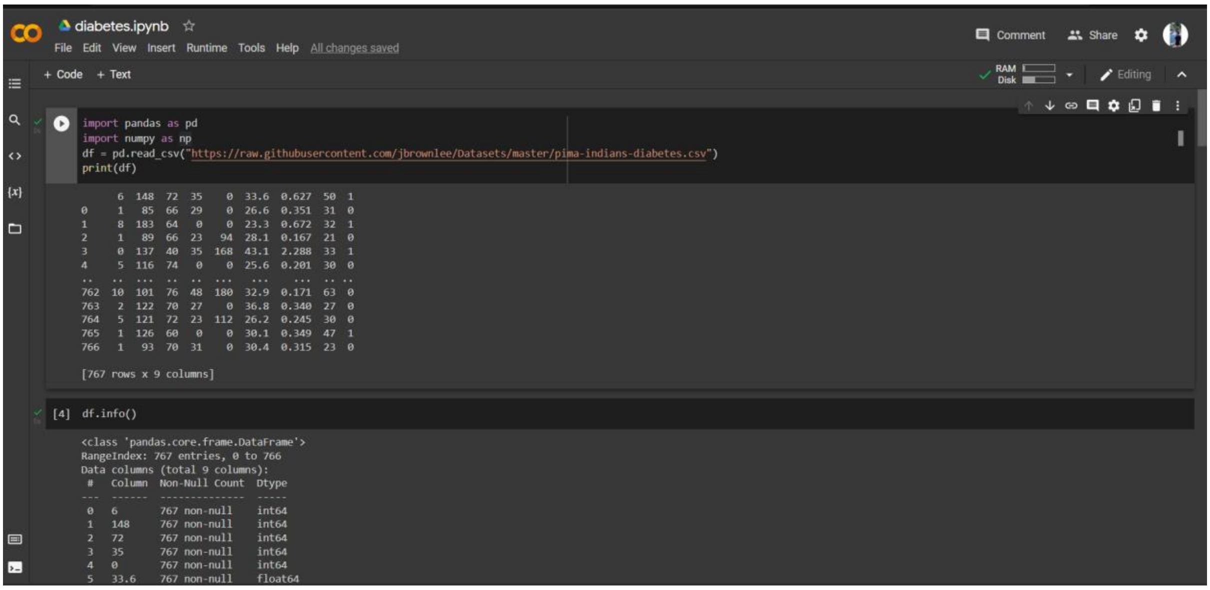
A. Extended Theory: (Soft Copy)

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

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting,
removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace
- interpolate()

Diabetes dataset eg

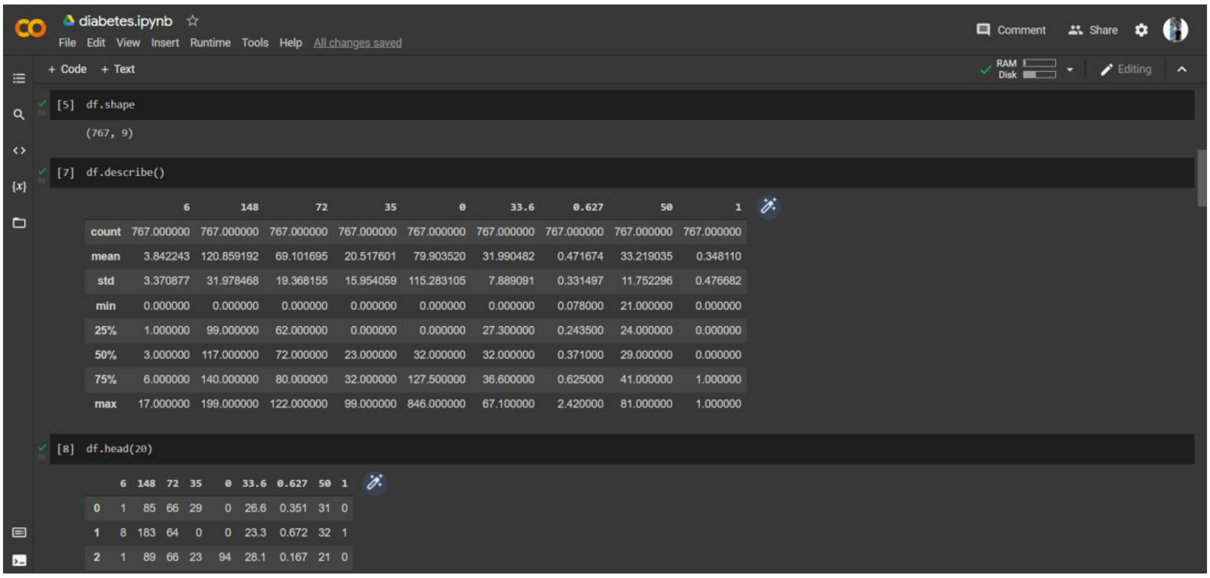


```
import pandas as pd
import numpy as np
df = pd.read_csv("https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv")
print(df)
```

```
       6  148  72  35    0  33.6  0.627  50  1
0      1   85  66  29    0  26.6  0.351  31  0
1      8  183  64   0    0  23.3  0.672  32  1
2      1   89  66  23   94  28.1  0.167  21  0
3      0  137  40  35  168  43.1  2.288  33  1
4      5  116  74   0    0  25.6  0.201  30  0
..    ..  ...  ..  ..  ...   ...    ...  .. ..
762   10  101  76  48  180  32.9  0.171  63  0
763    2  122  70  27    0  36.8  0.340  27  0
764    5  121  72  23  112  26.2  0.245  30  0
765    1  126  60   0    0  30.1  0.349  47  1
766    1   93  70  31    0  30.4  0.315  23  0

[767 rows x 9 columns]
```

```
[4] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 767 entries, 0 to 766
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   6       767 non-null    int64
 1   148     767 non-null    int64
 2   72      767 non-null    int64
 3   35      767 non-null    int64
 4   0       767 non-null    int64
 5   33.6    767 non-null    float64
```



```
[5] df.shape
```

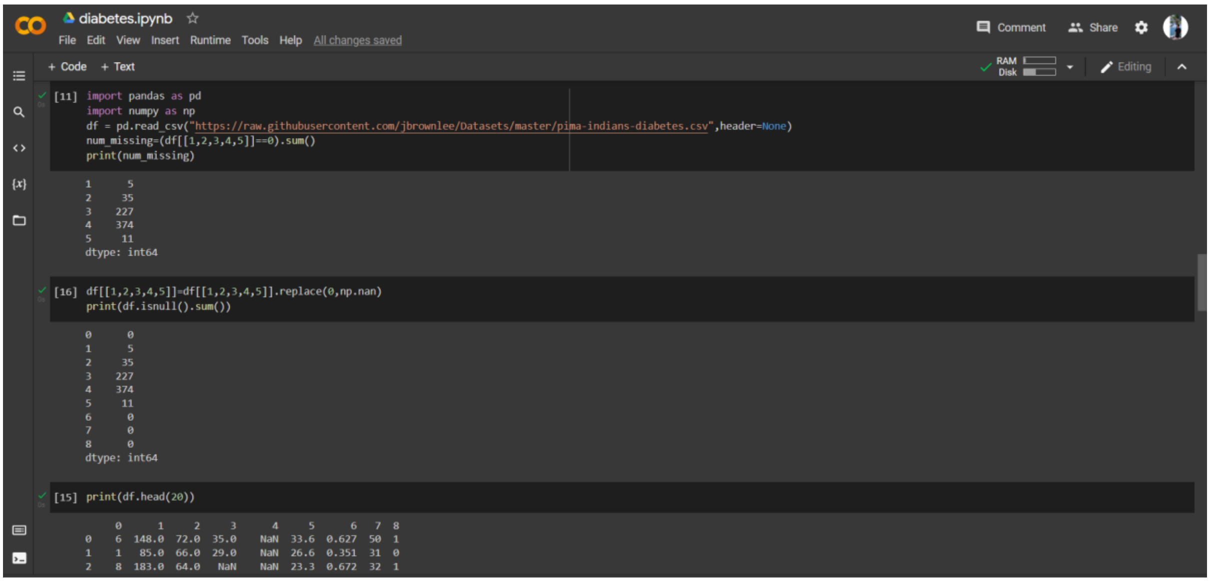
```
(767, 9)
```

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

	6	148	72	35	0	33.6	0.627	50	1
count	767.000000	767.000000	767.000000	767.000000	767.000000	767.000000	767.000000	767.000000	767.000000
mean	3.842243	120.859192	69.101695	20.517601	79.903520	31.990482	0.471674	33.219035	0.348110
std	3.370877	31.978468	19.368155	15.954059	115.283105	7.889091	0.331497	11.752296	0.476682
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243500	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	32.000000	32.000000	0.371000	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	127.500000	36.600000	0.625000	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
[8] df.head(20)
```

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0



```
[11] import pandas as pd
     import numpy as np
     df = pd.read_csv("https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv",header=None)
     num_missing=(df[[1,2,3,4,5]]==0).sum()
     print(num_missing)
```

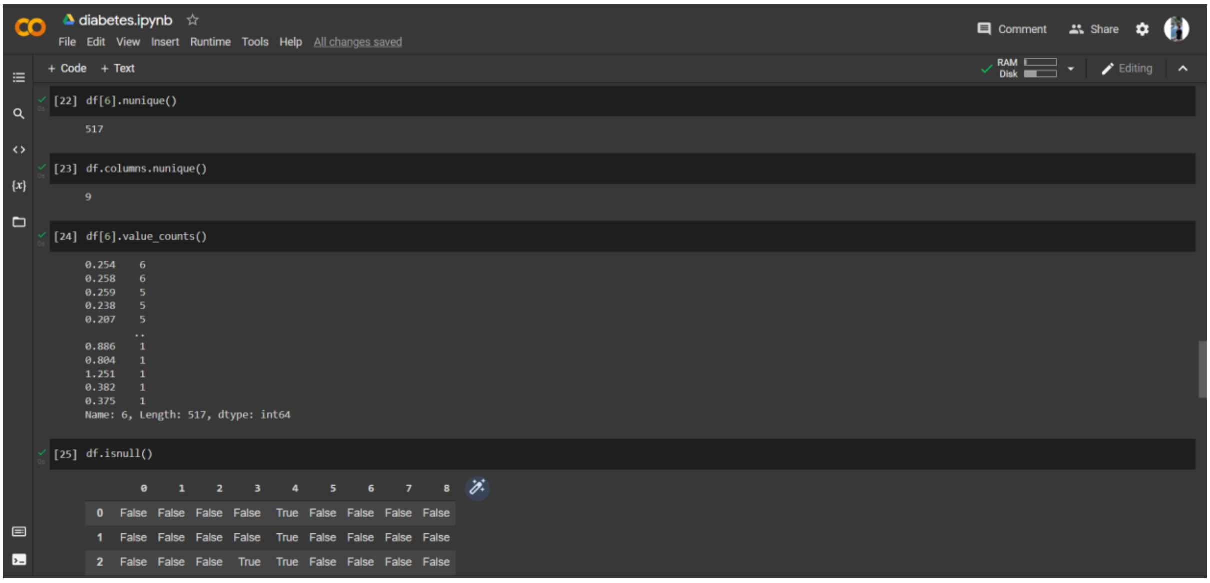
```
1      5
2     35
3    227
4    374
5     11
dtype: int64
```

```
[16] df[[1,2,3,4,5]]=df[[1,2,3,4,5]].replace(0,np.nan)
     print(df.isnull().sum())
```

```
0      0
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8      0
dtype: int64
```

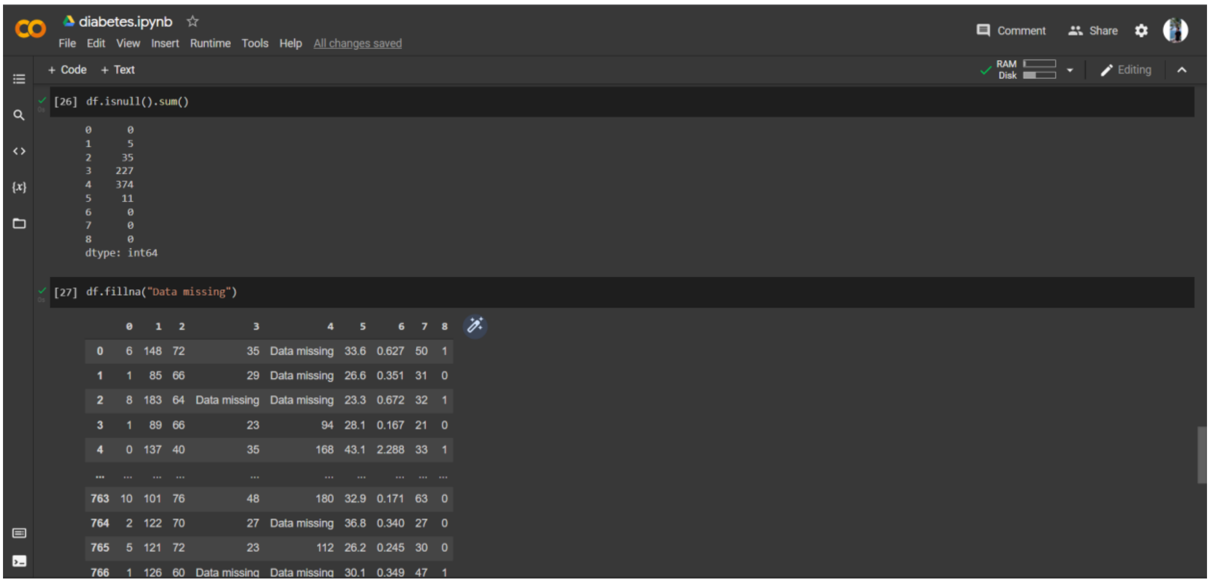
```
[15] print(df.head(20))
```

```
    0      1     2     3      4     5      6   7  8
0   6  148.0  72.0  35.0    NaN  33.6  0.627  50  1
1   1   85.0  66.0  29.0    NaN  26.6  0.351  31  0
2   8  183.0  64.0   NaN    NaN  23.3  0.672  32  1
```



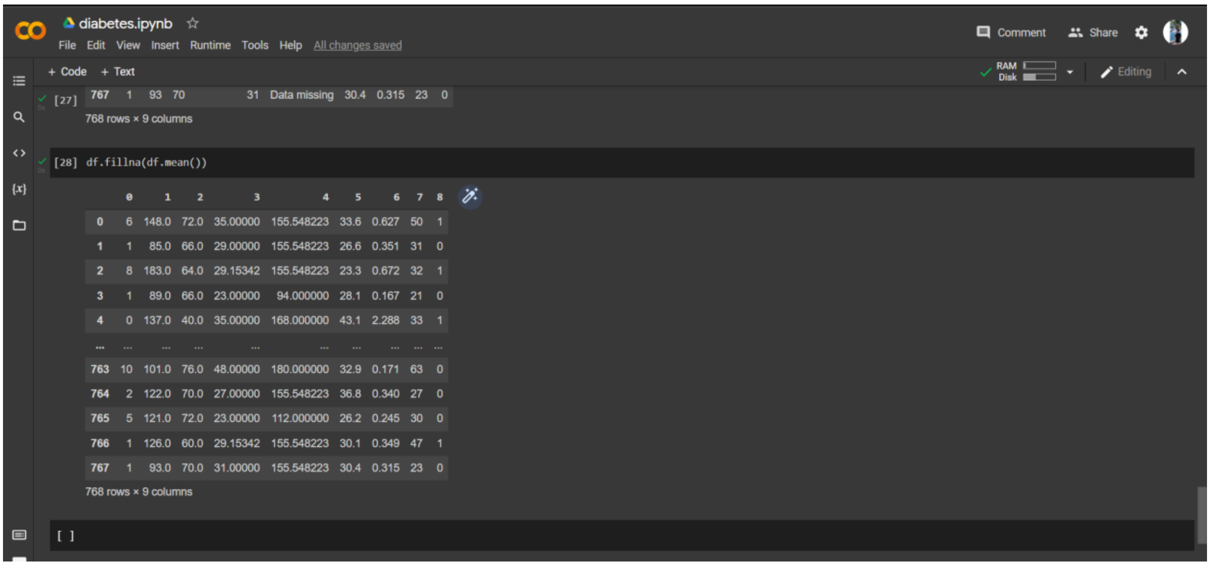
```
[22] df[6].nunique()

     517

[23] df.columns.nunique()

     9

[24] df[6].value_counts()

     0.254    6
     0.258    6
     0.259    5
     0.238    5
     0.207    5
             ..
     0.886    1
     0.804    1
     1.251    1
     0.382    1
     0.375    1
     Name: 6, Length: 517, dtype: int64

[25] df.isnull()
```

	0	1	2	3	4	5	6	7	8
0	False	False	False	False	True	False	False	False	False
1	False	False	False	False	True	False	False	False	False
2	False	False	False	True	True	False	False	False	False



```
[26] df.isnull().sum()

     0      0
     1      5
     2     35
     3    227
     4    374
     5     11
     6      0
     7      0
     8      0
     dtype: int64

[27] df.fillna("Data missing")
```

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	Data missing	33.6	0.627	50	1
1	1	85	66	29	Data missing	26.6	0.351	31	0
2	8	183	64	Data missing	Data missing	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	Data missing	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	Data missing	Data missing	30.1	0.349	47	1



[27]

767	1	93	70	31	Data missing	30.4	0.315	23	0

768 rows × 9 columns

```
[28] df.fillna(df.mean())
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.00000	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.00000	155.548223	36.8	0.340	27	0
765	5	121.0	72.0	23.00000	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	29.15342	155.548223	30.1	0.349	47	1
767	1	93.0	70.0	31.00000	155.548223	30.4	0.315	23	0

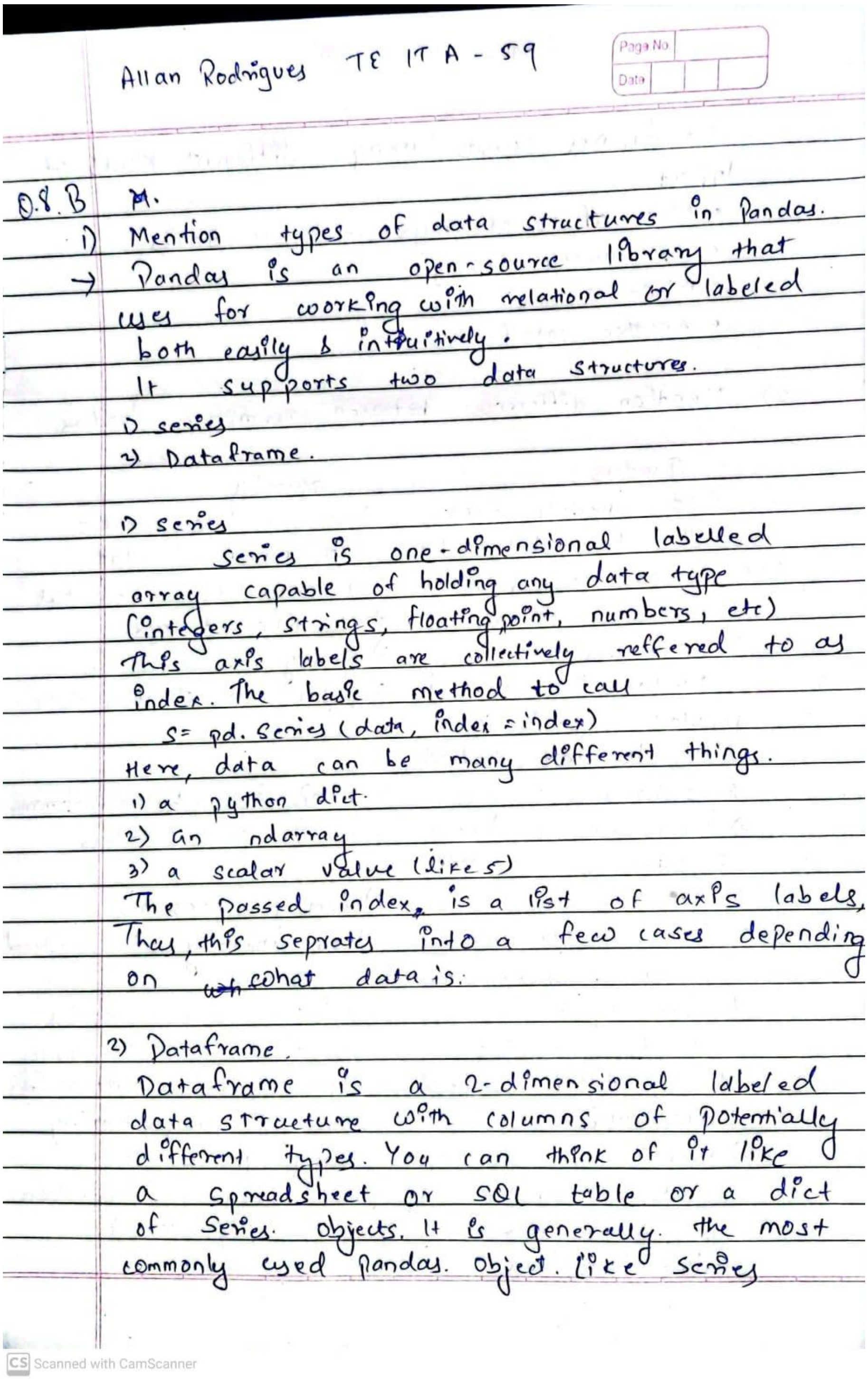
768 rows × 9 columns

```
[ ]
```

B. Questions:

Mention types of data structures in Pandas.

Mention difference between Numpy and Pandas.



Allan Rodrigues TE IT A - 59

Page No.
Date

Q.8.B A.

1) Mention types of data structures in Pandas.

→ Pandas is an open-source library that uses for working with relational or labeled both easily & intuitively.

It supports two data structures.

1) series

2) Dataframe.

1) series

series is one-dimensional labelled array capable of holding any data type (integers, strings, floating point, numbers, etc) This axis labels are collectively reffered to as index. The basic method to call

s= pd.Series(data, index =index)

Here, data can be many different things.

1) a python dict.

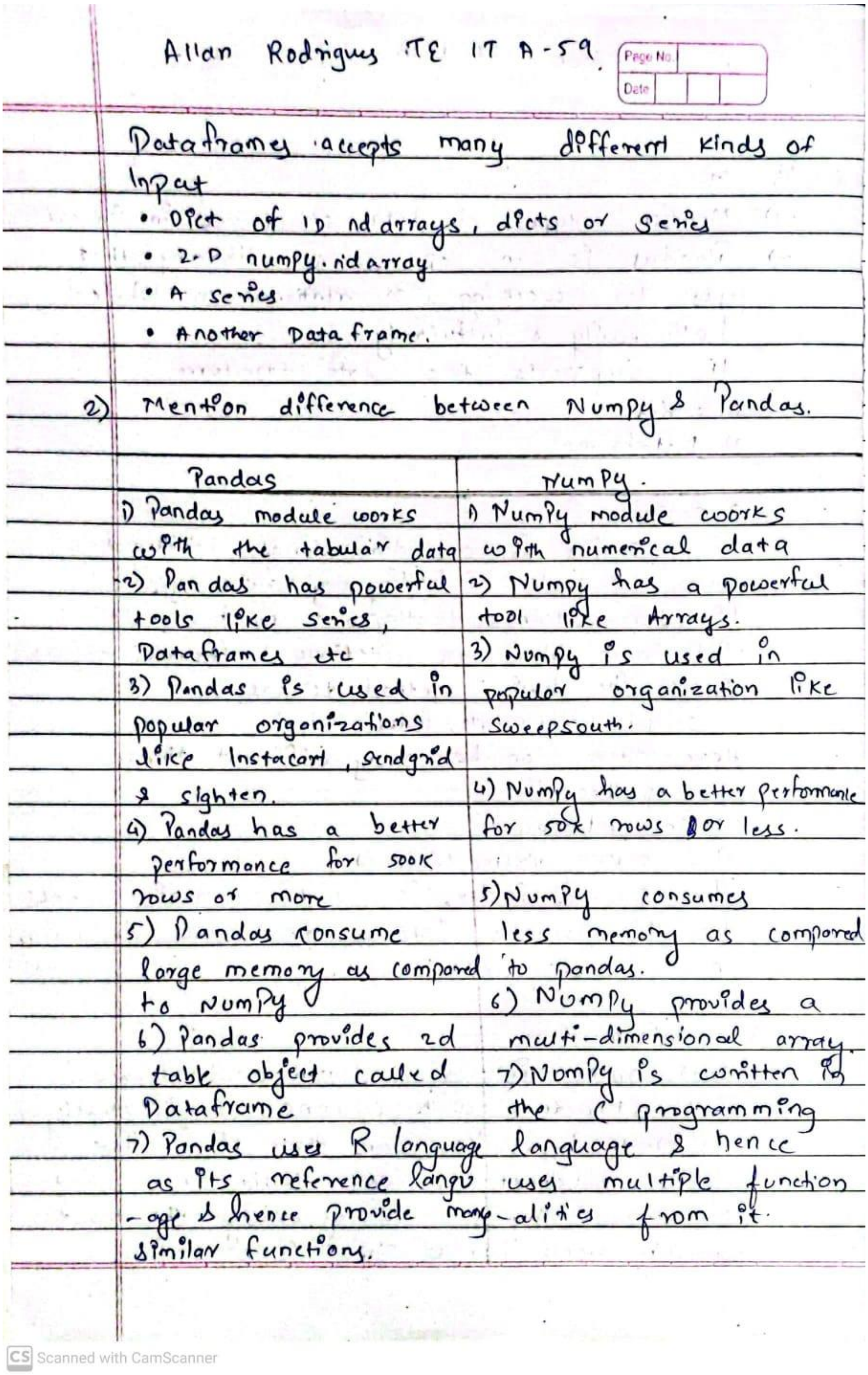
2) an ndarray

3) a scalar value (like 5)

The passed index, is a list of axis labels, Thus, this seprates into a few cases depending on ~~wh~~ what data is.

2) Dataframe.

Dataframe is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table or a dict of Series objects. It is generally the most commonly used pandas object. like series



Allan Rodrigues TE IT A-59.

Page No.

Date

Dataframes accepts many different kinds of Input

- Dict of 1D nd arrays, dicts or Series
- 2-D numpy.ndarray
- A series
- Another Data frame.

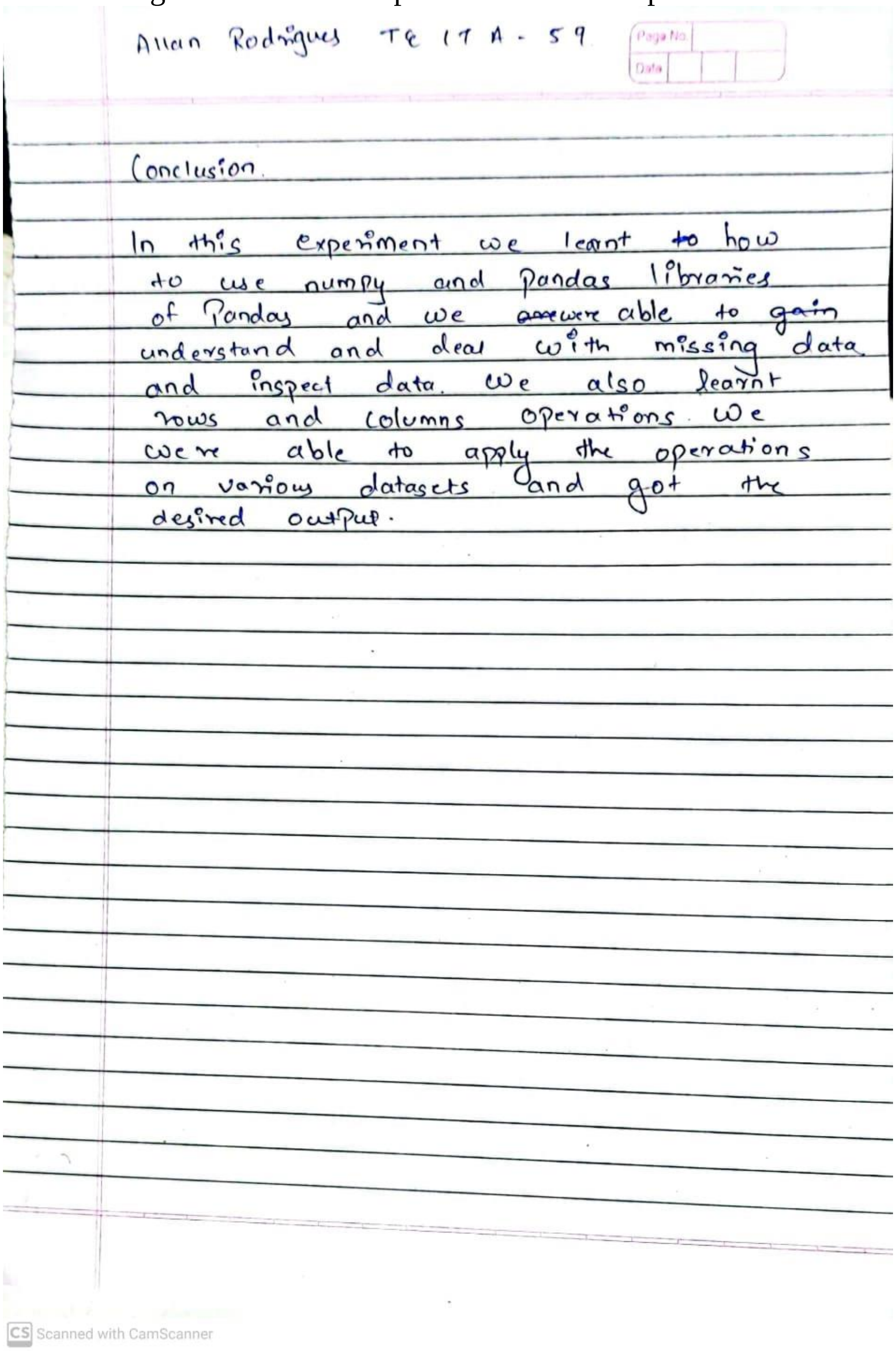
2) Mention difference between Numpy & Pandas.

Pandas	Numpy.
1) Pandas module works with the tabular data	1) Numpy module works with numerical data
2) Pandas has powerful tools like Series, Dataframes etc	2) Numpy has a powerful tool like Arrays.
3) Pandas is used in popular organizations like Instacart, Sendgrid & sighten.	3) Numpy is used in popular organization like SweepSouth.
4) Pandas has a better performance for 500K rows or more	4) Numpy has a better performance for 50K rows or less.
5) Pandas consume large memory as compared to Numpy	5) Numpy consumes less memory as compared to pandas.
6) Pandas provides 2d table object called Dataframe	6) Numpy provides a multi-dimensional array.
7) Pandas uses R language as its reference language & hence provide many similar functions.	7) Numpy is written in the C programming language & hence uses multiple functionalities from it.

Scanned with CamScanner

C. Conclusion:

Write the significance of the topic studied in the experiment.



Allan Rodrigues TE IT A - 59

Page No.
Date

Conclusion.

In this experiment we learnt ~~to~~ how to use numpy and Pandas libraries of Pandas and we ~~are~~ were able to ~~gain~~ understand and deal with missing data and inspect data. We also learnt rows and columns operations. We were able to apply the operations on various datasets and got the desired output.

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/
