```
#data set create

import pandas as pd

data = {
    'Name': ['Jai', 'Princi', 'Princi', 'Gaurav', 'Anuj', 'Ravi', 'Natasha', 'Riya'],
    'Age': [17, 17, 18, 17, 18, 17, 17, 18],
    'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F', 'F'],
    'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71, 80]
}

df = pd.DataFrame(data)
print(df)
```

```
      Name  Age Gender Marks
0      Jai   17      M    90
1   Princi   17      F    76
2   Princi   18      M   NaN
3   Gaurav   17      M    74
4     Anuj   18      M    65
5     Ravi   17      F   NaN
6  Natasha   17      F    71
7     Riya   18      F    80
```

To upload the file to your Colab environment, you can use the file upload feature in the left sidebar or run the following code:

After uploading the file, you can read it into a pandas DataFrame using the file name:

```
#upload csv file

import pandas as pd

# Assuming the uploaded file is named 'loan_data.csv'
df = pd.read_csv('loan_data.csv')
print(df.head())
```

Show hidden output

Next steps: Explain error

```
#Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using Datasets

# Import statistics library
import statistics

# Calculate average values
print('Calculate the average from a sample of data')
print(statistics.mean([1, 3, 5, 7, 9, 11, 13]))
print(statistics.mean([1, 3, 5, 7, 9, 11]))
print(statistics.mean([-11, 5.5, -3.4, 7.1, -9, 22]))
print('------------------------------------')
```

```
# Calculate middle values
print('Calculate the middle from a sample of data')
print(statistics.median([1, 3, 5, 7, 9, 11, 13]))
print(statistics.median([1, 3, 5, 7, 9, 11]))
print(statistics.median([-11, 5.5, -3.4, 7.1, -9, 22]))
print('------------------------------------')


# Calculate the mode
print('Calculate the mode from a sample of data')
print(statistics.mode([1, 3, 3, 3, 5, 7, 7, 9, 11]))
print(statistics.mode([1, 1, 3, -5, 7, -9, 11]))
print(statistics.mode(['red', 'green', 'blue', 'red']))
print('------------------------------------')


# Calculate the standard deviation from a sample of data
print('Calculate the standard deviation from a sample of data')
print(statistics.stdev([1, 3, 5, 7, 9, 11]))
print(statistics.stdev([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.stdev([-11, 5.5, -3.4, 7.1]))
print(statistics.stdev([1, 30, 50, 100]))
print('------------------------------------')


# Calculate the variance from a sample of data
print('Calculate the variance from a sample of data')
print(statistics.variance([1, 3, 5, 7, 9, 11]))
print(statistics.variance([2, 2.5, 1.25, 3.1, 1.75, 2.8]))
print(statistics.variance([-11, 5.5, -3.4, 7.1]))
print(statistics.variance([1, 30, 50, 100]))
print('------------------------------------')
```

```
Calculate the average from a sample of data
7
6
1.8666666666666667
------------------------------------
Calculate the middle from a sample of data
7
6.0
1.05
------------------------------------
Calculate the mode from a sample of data
3
1
red
------------------------------------
Calculate the standard deviation from a sample of data
3.7416573867739413
0.6925797186365383
8.414471660973927
41.67633221226007
------------------------------------
Calculate the variance from a sample of data
14
```

```
0.47966666666666663
70.80333333333333
1736.9166666666667
------------------------------------
```

```
#Demonstrate various data pre-processing techniques for a given dataset. Write a python
'''program to compute
i. Reshaping the data,
ii. Filtering the data,
iii. Merging the data
iv. Handling the missing values in datasets
v. Feature Normalization: Min-max normalization
'''
```

```
#1. reshaping the data

# Import pandas package
import pandas as pd

# Assign data
data = {'Name': ['Jai', 'Princi', 'Gaurav',
                 'Anuj', 'Ravi', 'Natasha', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```

	Name	Age	Gender	Marks
0	Jai	17	M	90
1	Princi	17	F	76
2	Gaurav	18	M	NaN
3	Anuj	17	M	74
4	Ravi	18	M	65
5	Natasha	17	F	NaN
6	Riya	17	F	71

Next steps: Generate code with df View recommended plots New interactive sheet

```
#2.filtering the data
```

```
df.filter(['Name'])
```

```
      Name
0      Jai
1   Princi
2   Gaurav
3     Anuj
4     Ravi
5  Natasha
6     Riya
```

```
#3.Merging the data

# import module
import pandas as pd

print('first table')

# creating DataFrame for Student Details
details = pd.DataFrame({
    'ID': [101, 102, 103, 104, 105, 106,
           107, 108, 109, 110],
    'NAME': ['Jagroop', 'Praveen', 'Harjot',
             'Pooja', 'Rahul', 'Nikita',
             'Saurabh', 'Ayush', 'Dolly', 'Mohit'],
    'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE',
               'CSE', 'CSE', 'CSE', 'CSE', 'CSE', 'CSE']
})

# printing details
print(details)

print('-------------------------------------------------')

print("second table")

# Creating DataFrame for Fees_Status
fees_status = pd.DataFrame(
    {'ID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],
     'PENDING': ['5000', '250', 'NIL', '9000', '15000', 'NIL',
                 '4500', '1800', '250', 'NIL']}
)

# Printing fees_status
print(fees_status)

print('-------------------------------------------------')

print('Merging the data base on ID ')
print(pd.merge(details, fees_status, on='ID'))
```

```
first table
    ID     NAME BRANCH
0  101  Jagroop    CSE
1  102  Praveen    CSE
2  103   Harjot    CSE
3  104    Pooja    CSE
4  105    Rahul    CSE
5  106   Nikita    CSE
6  107  Saurabh    CSE
7  108    Ayush    CSE
8  109    Dolly    CSE
9  110    Mohit    CSE
-------------------------------------------------
second table
    ID PENDING
0  101    5000
1  102     250
```

```
2  103     NIL
3  104    9000
4  105   15000
5  106     NIL
6  107    4500
7  108    1800
8  109     250
9  110     NIL
-------------------------------------------------
Merging the data base on ID
    ID     NAME BRANCH PENDING
0  101  Jagroop    CSE    5000
1  102  Praveen    CSE     250
2  103   Harjot    CSE     NIL
3  104    Pooja    CSE    9000
4  105    Rahul    CSE   15000
5  106   Nikita    CSE     NIL
6  107  Saurabh    CSE    4500
7  108    Ayush    CSE    1800
8  109    Dolly    CSE     250
9  110    Mohit    CSE     NIL
```

```
#iv. Handling the missing values in datasets

# Import module
import pandas as pd
import numpy as np

# Creating DataFrame for Fees_Status
fees_status = pd.DataFrame(
    {'ID': [101, 102, 103, 104, 105,
            106, 107, 108, 109, 110],
     'PENDING': [5000, 250, np.nan,
                 9000, 15000, np.nan,
                 4500, 1800, 250, np.nan]}
)

# Printing fees_status
fees_status
```

	ID	PENDING
0	101	5000.0
1	102	250.0
2	103	NaN
3	104	9000.0
4	105	15000.0
5	106	NaN
6	107	4500.0
7	108	1800.0
8	109	250.0
9	110	NaN

Next steps: Generate code with `fees_status` View recommended plots New interactive sheet

```
#handle missing values with mean

import numpy as np
import pandas as pd

# A dictionary with list as values
GFG_dict = {
    'G1': [10, 20, 30, 40],
    'G2': [25, np.nan, np.nan, 29],
    'G3': [15, 14, 17, 11],
    'G4': [21, 22, 23, 25]
}

# Create a DataFrame from dictionary
gfg = pd.DataFrame(GFG_dict)

# Finding the mean of the column having NaN
mean_value = gfg['G2'].mean()

# Replace NaNs in column G2 with the mean of values in the same column
gfg['G2'].fillna(value=mean_value, inplace=True)

print('Updated Dataframe:')
print(gfg)
```

```
Updated Dataframe:
   G1    G2  G3  G4
0  10  25.0  15  21
1  20  27.0  14  22
2  30  27.0  17  23
3  40  29.0  11  25
/tmp/ipython-input-17-1124811658.py:21: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

```
    For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the ope

      gfg['G2'].fillna(value=mean_value, inplace=True)
```

```
#5.Min-Max Normalization

# Import necessary libraries
```