```
Start coding or generate with AI.
```

Write a python program to compute Mean, Median, Mode, Variance, Standard Deviation using the Iris dataset. Also, demonstrate various data pre-processing techniques for a random dataset, including reshaping, filtering, merging, handling missing values, and Min-max normalization.

Load the iris dataset

Load the Iris dataset using a suitable library like scikit-learn or pandas.

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()

df_iris = pd.DataFrame(data=iris.data, columns=iris.feature_names)

display(df_iris.head())
display(df_iris.info())
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
None
```

Compute descriptive statistics

Calculate the Mean, Median, Mode, Variance, and Standard Deviation for each feature in the Iris dataset.

```
import numpy as np

mean_values = df_iris.mean()

median_values = df_iris.median()

mode_values = df_iris.mode().iloc[0]


variance_values = df_iris.var()

std_dev_values = df_iris.std()

statistics = {
    'Mean': mean_values,
    'Median': median_values,
    'Mode': mode_values,
    'Variance': variance_values,
    'Standard Deviation': std_dev_values
}

stats_df = pd.DataFrame(statistics)

display(stats_df)
```

	Mean	Median	Mode	Variance	Standard Deviation
sepal length (cm)	5.843333	5.80	5.0	0.685694	0.828066
sepal width (cm)	3.057333	3.00	3.0	0.189979	0.435866
petal length (cm)	3.758000	4.35	1.4	3.116278	1.765298
petal width (cm)	1.199333	1.30	0.2	0.581006	0.762238

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

Display results

Present the computed statistics in a clear format.

```
display(stats_df)
```

	Mean	Median	Mode	Variance	Standard Deviation
sepal length (cm)	5.843333	5.80	5.0	0.685694	0.828066
sepal width (cm)	3.057333	3.00	3.0	0.189979	0.435866
petal length (cm)	3.758000	4.35	1.4	3.116278	1.765298
petal width (cm)	1.199333	1.30	0.2	0.581006	0.762238

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

Create a random dataset

Generate a random dataset for demonstration purposes.

```
import numpy as np
import pandas as pd

data = {
    'Column A': np.random.rand(15),
    'Column B': np.random.rand(15),
    'Column C': np.random.rand(15)
}

df_random = pd.DataFrame(data)

display(df_random.head())
```

	Column A	Column B	Column C
0	0.465538	0.445022	0.759710
1	0.135850	0.271254	0.252382
2	0.467719	0.046016	0.628122
3	0.174865	0.394590	0.240067
4	0.581747	0.375265	0.485304

Reshape the data

Demonstrate reshaping the data using appropriate techniques (e.g., melt, pivot).

```
df_melted = pd.melt(df_random, var_name='variable', value_name='value')

display(df_melted.head())
```

```
     variable     value
0    Column A  0.465538
1    Column A  0.135850
2    Column A  0.467719
3    Column A  0.174865
4    Column A  0.581747
```

Filter the data

Show how to filter the data based on specific conditions.

```
df_filtered = df_random[df_random['Column A'] > 0.5]

display(df_filtered.head())
```

```
    Column A  Column B  Column C
4   0.581747  0.375265  0.485304
9   0.731315  0.702102  0.119362
10  0.993226  0.782158  0.621174
11  0.959279  0.774722  0.030088
12  0.556527  0.852705  0.899067
```

Merge the data

Create another random dataset and demonstrate merging it with the first dataset.

```
data2 = {
    'Column D': np.random.rand(len(df_random))
}

df_random2 = pd.DataFrame(data2)

df_merged = pd.concat([df_random, df_random2], axis=1)

display(df_merged.head())
```

```
   Column A  Column B  Column C  Column D
0  0.465538  0.445022  0.759710  0.010975
1  0.135850  0.271254  0.252382  0.343383
2  0.467719  0.046016  0.628122  0.632905
3  0.174865  0.394590  0.240067  0.125320
4  0.581747  0.375265  0.485304  0.779490
```

Handle missing values

Introduce missing values into the dataset and demonstrate techniques for handling them (e.g., imputation, dropping).

```
import numpy as np

np.random.seed(42)
mask = np.random.choice([True, False], size=df_merged.shape, p=[0.1, 0.9])
df_merged_with_missing = df_merged.mask(mask)

display("DataFrame with missing values:")
display(df_merged_with_missing)
```

```
'DataFrame with missing values:'
     Column A  Column B  Column C  Column D
0    0.465538  0.445022  0.759710  0.010975
1    0.135850  0.271254       NaN  0.343383
2    0.467719  0.046016       NaN  0.632905
3    0.174865  0.394590  0.240067  0.125320
4    0.581747  0.375265  0.485304  0.779490
5    0.117306  0.770830  0.030528  0.020289
6    0.303839  0.100768  0.303682  0.067747
7    0.324920       NaN  0.489434  0.435235
8         NaN  0.980674  0.688351  0.462965
9    0.731315       NaN  0.119362  0.583026
10   0.993226  0.782158       NaN  0.433757
11   0.959279  0.774722  0.030088  0.219863
12   0.556527  0.852705  0.899067  0.047146
13   0.073683  0.234956  0.381687  0.999222
14        NaN  0.537056       NaN  0.049887
```

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

```
df_dropped = df_merged_with_missing.dropna()

display("DataFrame after dropping rows with missing values (head):")
display(df_dropped.head())

df_imputed = df_merged_with_missing.fillna(df_merged_with_missing.mean())

display("DataFrame after imputing missing values with mean (head):")
display(df_imputed.head())
```

```
'DataFrame after dropping rows with missing values (head):'
    Column A  Column B  Column C  Column D
0   0.465538  0.445022  0.759710  0.010975
3   0.174865  0.394590  0.240067  0.125320
4   0.581747  0.375265  0.485304  0.779490
5   0.117306  0.770830  0.030528  0.020289
6   0.303839  0.100768  0.303682  0.067747
'DataFrame after imputing missing values with mean (head):'
    Column A  Column B  Column C  Column D
0   0.465538  0.445022  0.759710  0.010975
1   0.135850  0.271254  0.402480  0.343383
2   0.467719  0.046016  0.402480  0.632905
3   0.174865  0.394590  0.240067  0.125320
4   0.581747  0.375265  0.485304  0.779490
```

Perform min-max normalization

Apply Min-max normalization to a numerical feature in the dataset.

```
column_to_normalize = 'Column A'

min_val = df_imputed[column_to_normalize].min()
```

```
max_val = df_imputed[column_to_normalize].max()

df_imputed[f'{column_to_normalize}_normalized'] = (df_imputed[column_to_normalize] - min_val) / (max_val - min_val)

display(df_imputed.head())
```

	Column A	Column B	Column C	Column D	Column A_normalized
0	0.465538	0.445022	0.759710	0.010975	0.426141
1	0.135850	0.271254	0.402480	0.343383	0.067606
2	0.467719	0.046016	0.402480	0.632905	0.428513
3	0.174865	0.394590	0.240067	0.125320	0.110034
4	0.581747	0.375265	0.485304	0.779490	0.552517

Display results

Show the results of each pre-processing step.

```
display("Original Random DataFrame:")
display(df_random)

display("Melted DataFrame:")
display(df_melted)

display("Filtered DataFrame:")
display(df_filtered)

display("Merged DataFrame:")
display(df_merged)

display("DataFrame with Missing Values:")
display(df_merged_with_missing)

display("DataFrame after Dropping Missing Values:")
display(df_dropped)

display("DataFrame after Imputing Missing Values and Normalization:")
display(df_imputed)
```

```
'Original Random DataFrame:'
    Column A  Column B  Column C
0   0.465538  0.445022  0.759710
1   0.135850  0.271254  0.252382
2   0.467719  0.046016  0.628122
3   0.174865  0.394590  0.240067
4   0.581747  0.375265  0.485304
5   0.117306  0.770830  0.030528
6   0.303839  0.100768  0.303682
7   0.324920  0.558833  0.489434
8   0.497950  0.980674  0.688351
9   0.731315  0.702102  0.119362
10  0.993226  0.782158  0.621174
11  0.959279  0.774722  0.030088
12  0.556527  0.852705  0.899067
13  0.073683  0.234956  0.381687
14  0.661941  0.537056  0.756287
'Melted DataFrame:'
    variable     value
0   Column A  0.465538
1   Column A  0.135850
2   Column A  0.467719
3   Column A  0.174865
4   Column A  0.581747
5   Column A  0.117306
6   Column A  0.303839
7   Column A  0.324920
8   Column A  0.497950
9   Column A  0.731315
10  Column A  0.993226
11  Column A  0.959279
12  Column A  0.556527
13  Column A  0.073683
14  Column A  0.661941
15  Column B  0.445022
16  Column B  0.271254
17  Column B  0.046016
18  Column B  0.394590
19  Column B  0.375265
20  Column B  0.770830
21  Column B  0.100768
22  Column B  0.558833
23  Column B  0.980674
24  Column B  0.702102
25  Column B  0.782158
26  Column B  0.774722
27  Column B  0.852705
28  Column B  0.234956
```

28	Column B	0.234956
29	Column B	0.537056
30	Column C	0.759710
31	Column C	0.252382
32	Column C	0.628122
33	Column C	0.240067
34	Column C	0.485304
35	Column C	0.030528
36	Column C	0.303682
37	Column C	0.489434
38	Column C	0.688351
39	Column C	0.119362
40	Column C	0.621174
41	Column C	0.030088
42	Column C	0.899067
43	Column C	0.381687
44	Column C	0.756287

```
'Filtered DataFrame:'
```

	Column A	Column B	Column C
4	0.581747	0.375265	0.485304
9	0.731315	0.702102	0.119362
10	0.993226	0.782158	0.621174
11	0.959279	0.774722	0.030088
12	0.556527	0.852705	0.899067
14	0.661941	0.537056	0.756287

```
'Merged DataFrame:'
```

	Column A	Column B	Column C	Column D
0	0.465538	0.445022	0.759710	0.010975
1	0.135850	0.271254	0.252382	0.343383
2	0.467719	0.046016	0.628122	0.632905
3	0.174865	0.394590	0.240067	0.125320
4	0.581747	0.375265	0.485304	0.779490
5	0.117306	0.770830	0.030528	0.020289
6	0.303839	0.100768	0.303682	0.067747
7	0.324920	0.558833	0.489434	0.435235
8	0.497950	0.980674	0.688351	0.462965
9	0.731315	0.702102	0.119362	0.583026
10	0.993226	0.782158	0.621174	0.433757
11	0.959279	0.774722	0.030088	0.219863
12	0.556527	0.852705	0.899067	0.047146
13	0.073683	0.234956	0.381687	0.999222
14	0.661941	0.537056	0.756287	0.049887

```
'DataFrame with Missing Values:'
```

	Column A	Column B	Column C	Column D
0	0.465538	0.445022	0.759710	0.010975
1	0.135850	0.271254	NaN	0.343383
2	0.467719	0.046016	NaN	0.632905
3	0.174865	0.394590	0.240067	0.125320
4	0.581747	0.375265	0.485304	0.779490