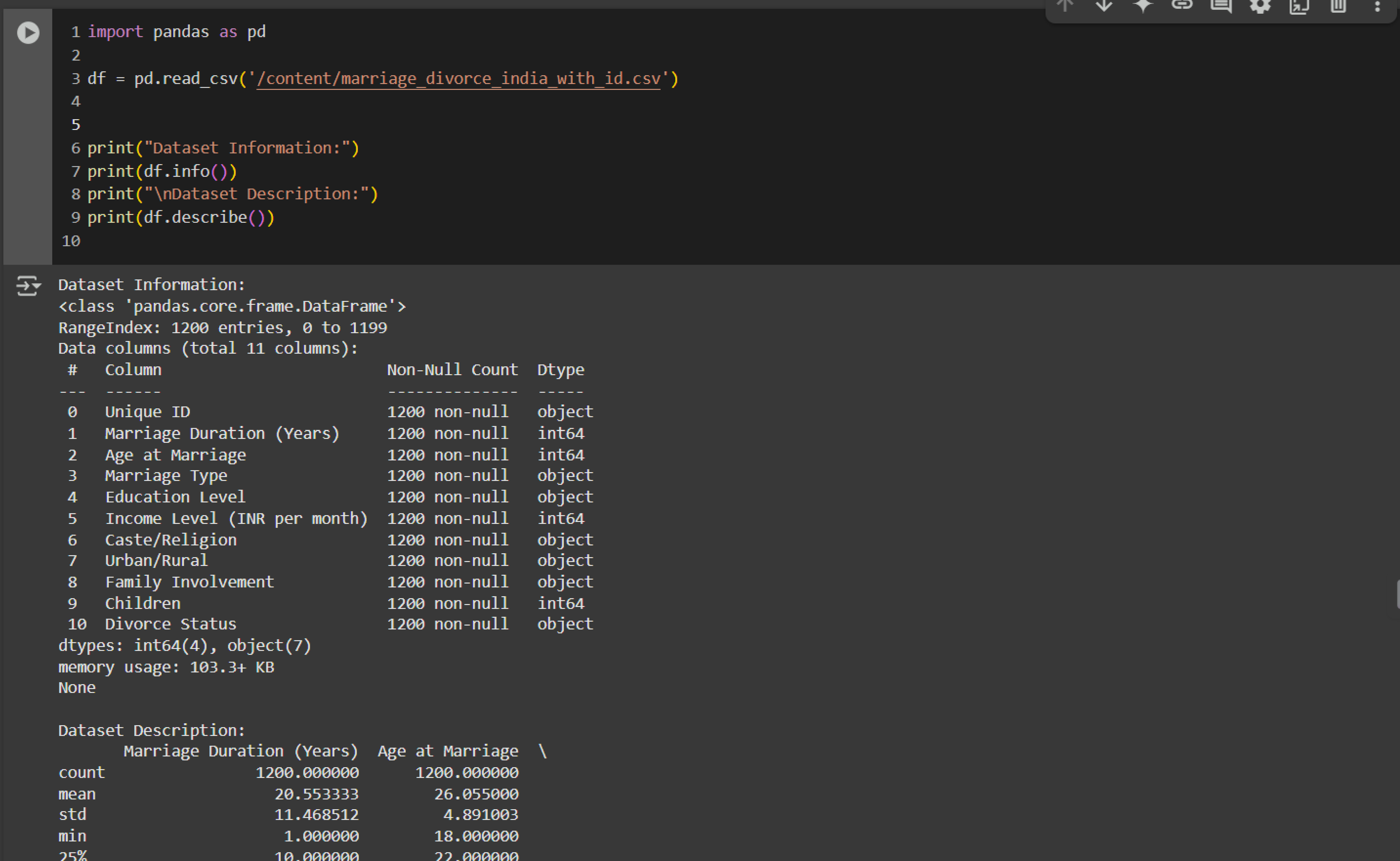
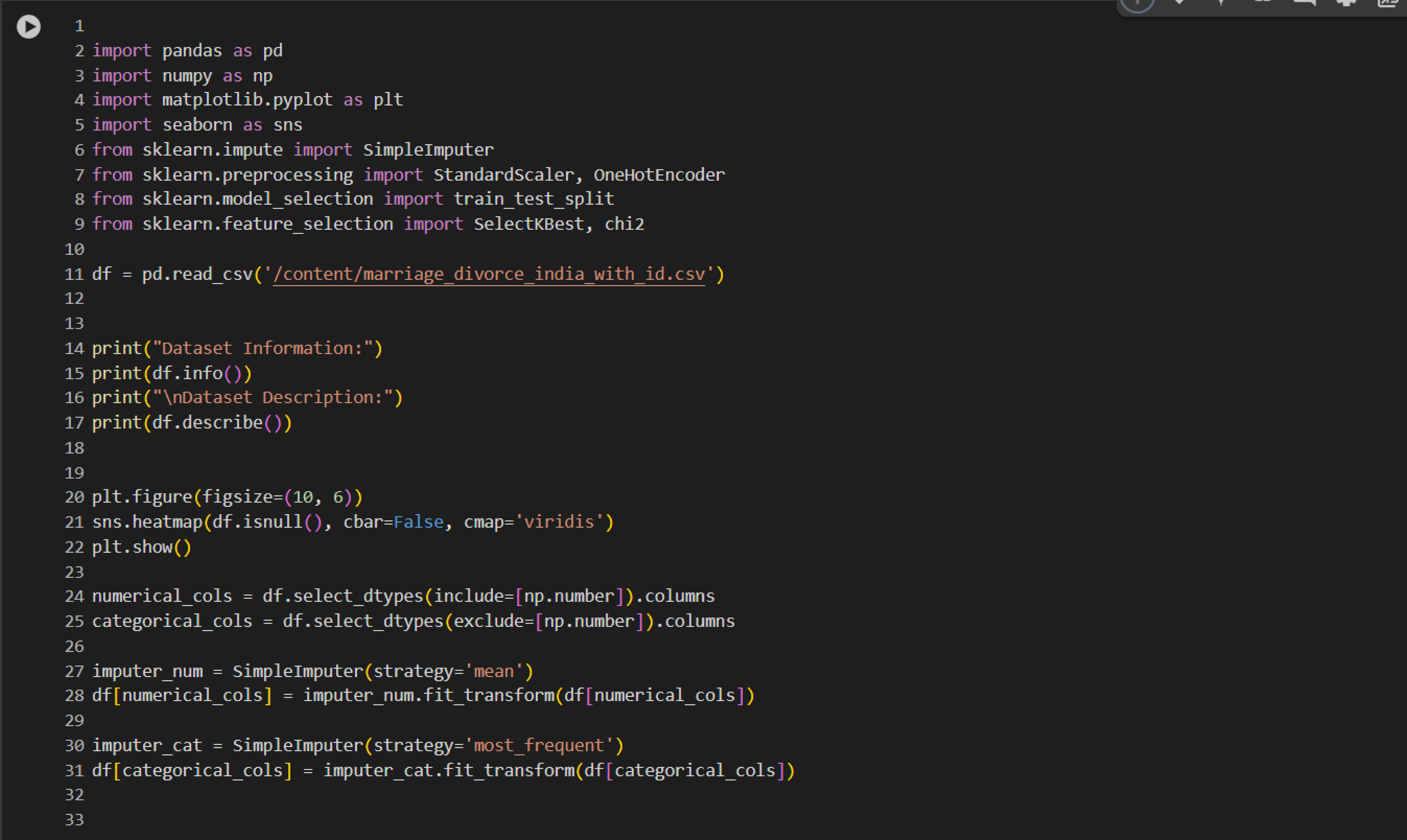
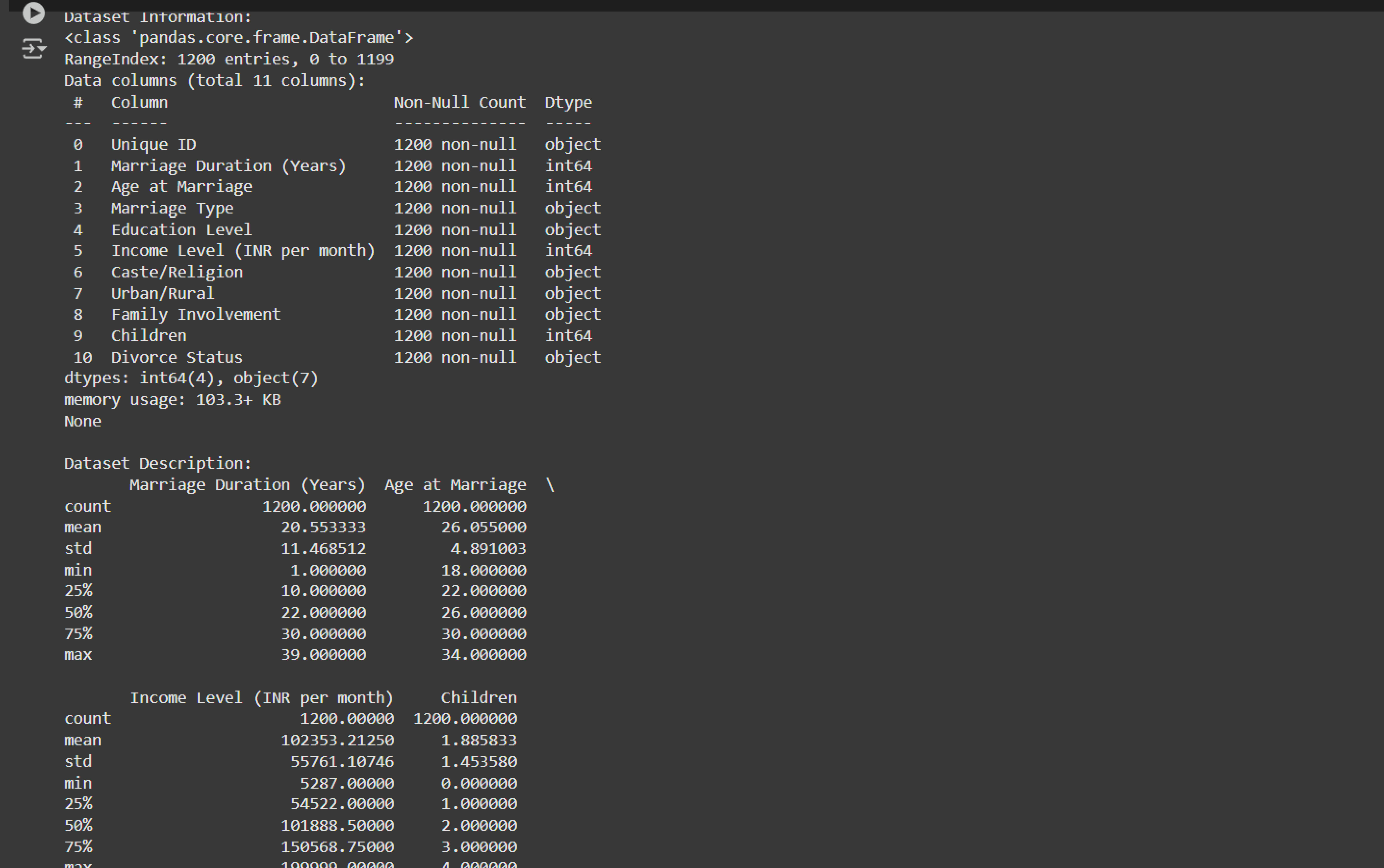


```python
import pandas as pd

df = pd.read_csv('/content/marriage_divorce_india_with_id.csv')


print("Dataset Information:")
print(df.info())
print("\nDataset Description:")
print(df.describe())

```

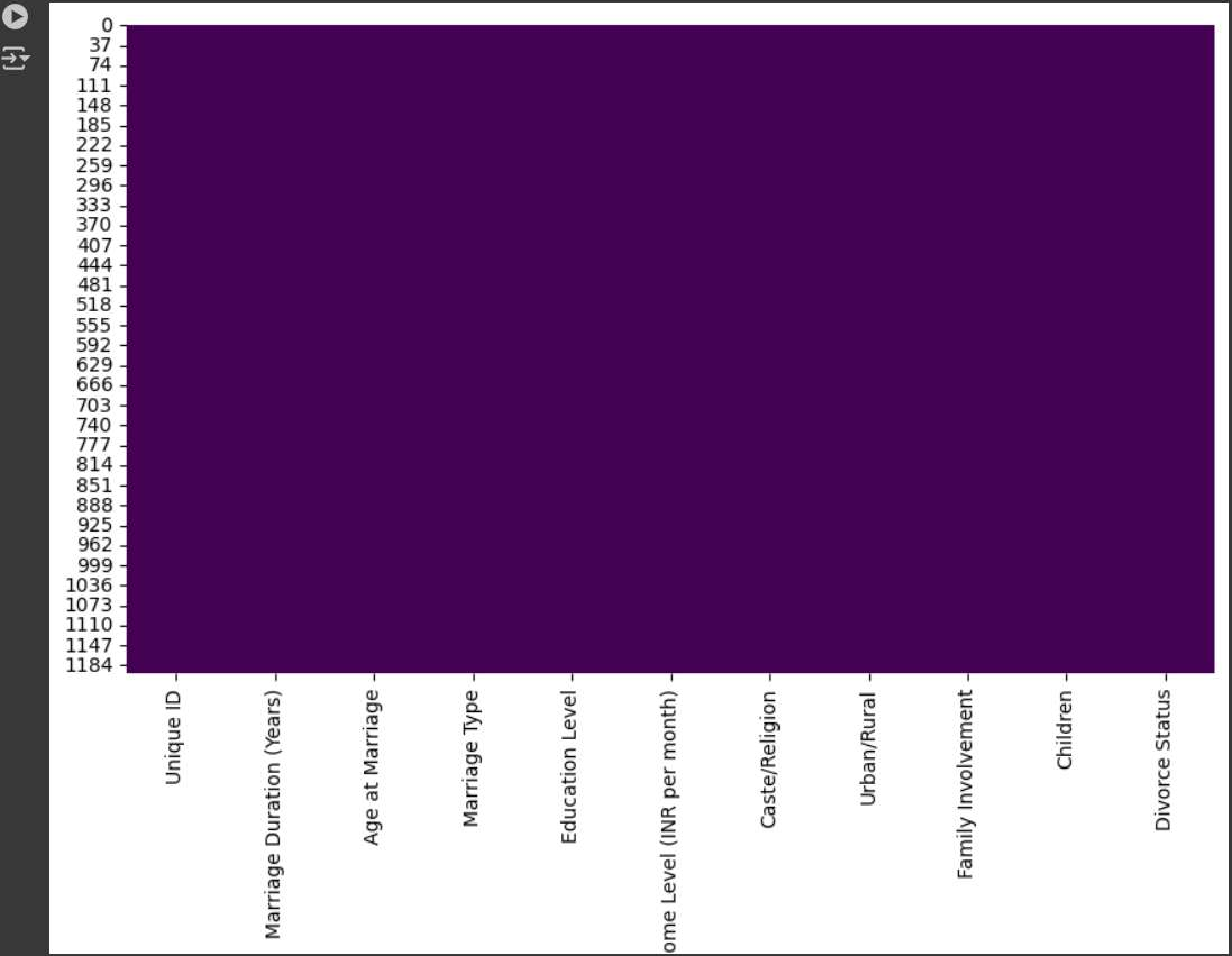
```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Unique ID                  1200 non-null   object
 1   Marriage Duration (Years)  1200 non-null   int64 
 2   Age at Marriage            1200 non-null   int64 
 3   Marriage Type              1200 non-null   object
 4   Education Level            1200 non-null   object
 5   Income Level (INR per month)  1200 non-null   int64 
 6   Caste/Religion             1200 non-null   object
 7   Urban/Rural                1200 non-null   object
 8   Family Involvement         1200 non-null   object
 9   Children                   1200 non-null   int64 
 10  Divorce Status             1200 non-null   object
dtypes: int64(4), object(7)
memory usage: 103.3+ KB
None

Dataset Description:
       Marriage Duration (Years)  Age at Marriage  \
count                1200.000000      1200.000000   
mean                   20.553333        26.055000   
std                    11.468512         4.891003   
min                     1.000000        18.000000   
25%                    10.000000        22.000000   
```

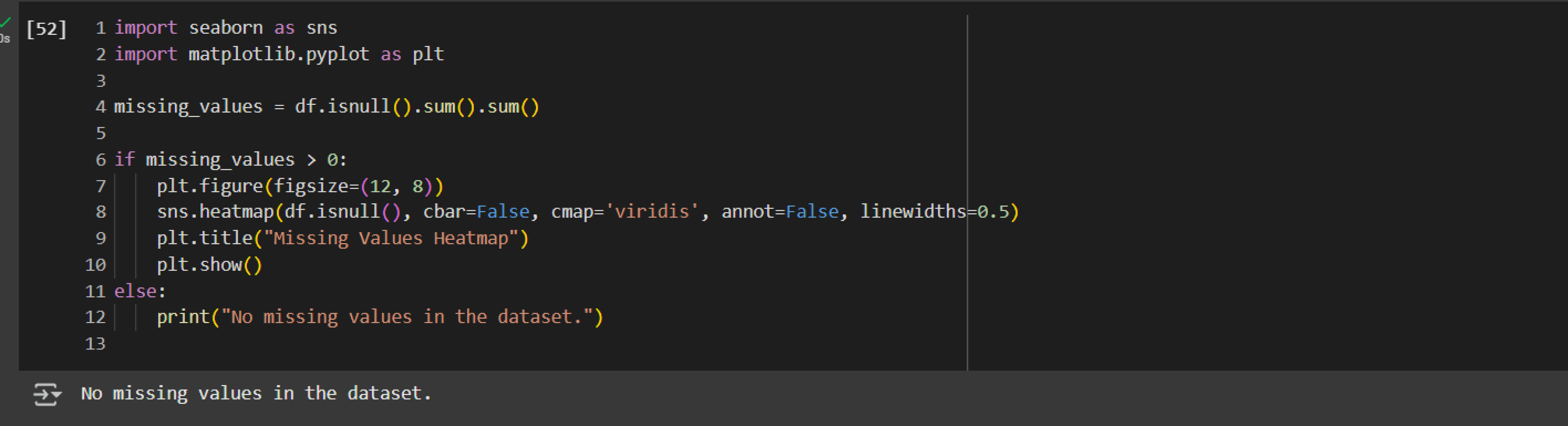


```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, chi2

df = pd.read_csv('/content/marriage_divorce_india_with_id.csv')


print("Dataset Information:")
print(df.info())
print("\nDataset Description:")
print(df.describe())


plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.show()

numerical_cols = df.select_dtypes(include=[np.number]).columns
categorical_cols = df.select_dtypes(exclude=[np.number]).columns

imputer_num = SimpleImputer(strategy='mean')
df[numerical_cols] = imputer_num.fit_transform(df[numerical_cols])

imputer_cat = SimpleImputer(strategy='most_frequent')
df[categorical_cols] = imputer_cat.fit_transform(df[categorical_cols])


```



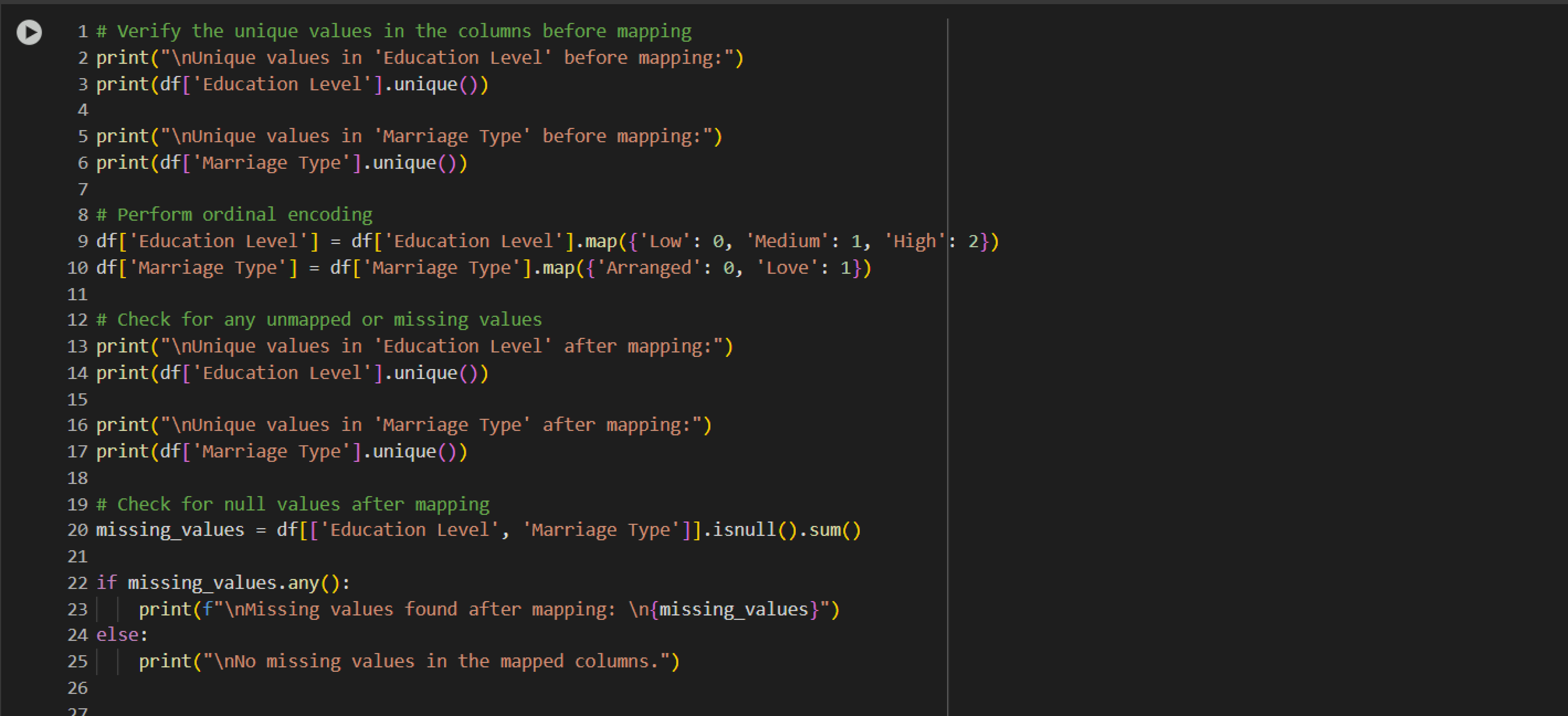
```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   Unique ID                     1200 non-null   object
 1   Marriage Duration (Years)     1200 non-null   int64 
 2   Age at Marriage               1200 non-null   int64 
 3   Marriage Type                 1200 non-null   object
 4   Education Level               1200 non-null   object
 5   Income Level (INR per month)  1200 non-null   int64 
 6   Caste/Religion                1200 non-null   object
 7   Urban/Rural                   1200 non-null   object
 8   Family Involvement            1200 non-null   object
 9   Children                      1200 non-null   int64 
 10  Divorce Status                1200 non-null   object
dtypes: int64(4), object(7)
memory usage: 103.3+ KB
None

Dataset Description:
       Marriage Duration (Years)  Age at Marriage  \
count                1200.000000      1200.000000   
mean                   20.553333        26.055000   
std                    11.468512         4.891003   
min                     1.000000        18.000000   
25%                    10.000000        22.000000   
50%                    22.000000        26.000000   
75%                    30.000000        30.000000   
max                    39.000000        34.000000   

       Income Level (INR per month)     Children
count                    1200.00000  1200.000000
mean                   102353.21250     1.885833
std                     55761.10746     1.453580
min                      5287.00000     0.000000
25%                     54522.00000     1.000000
50%                    101888.50000     2.000000
75%                    150568.75000     3.000000
max                    199999.00000     4.000000
```


0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
Unique ID
Marriage Duration (Years)
Age at Marriage
Marriage Type
Education Level
ome Level (INR per month)
Caste/Religion
Urban/Rural
Family Involvement
Children
Divorce Status

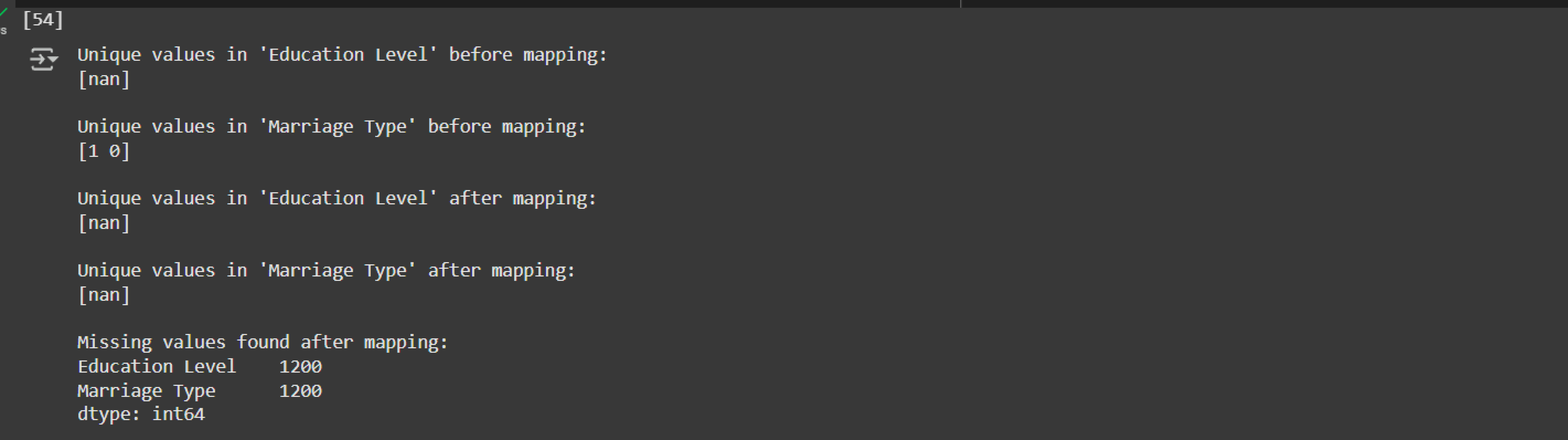


```python
import seaborn as sns
import matplotlib.pyplot as plt

missing_values = df.isnull().sum().sum()

if missing_values > 0:
    plt.figure(figsize=(12, 8))
    sns.heatmap(df.isnull(), cbar=False, cmap='viridis', annot=False, linewidths=0.5)
    plt.title("Missing Values Heatmap")
    plt.show()
else:
    print("No missing values in the dataset.")

```

```
No missing values in the dataset.
```

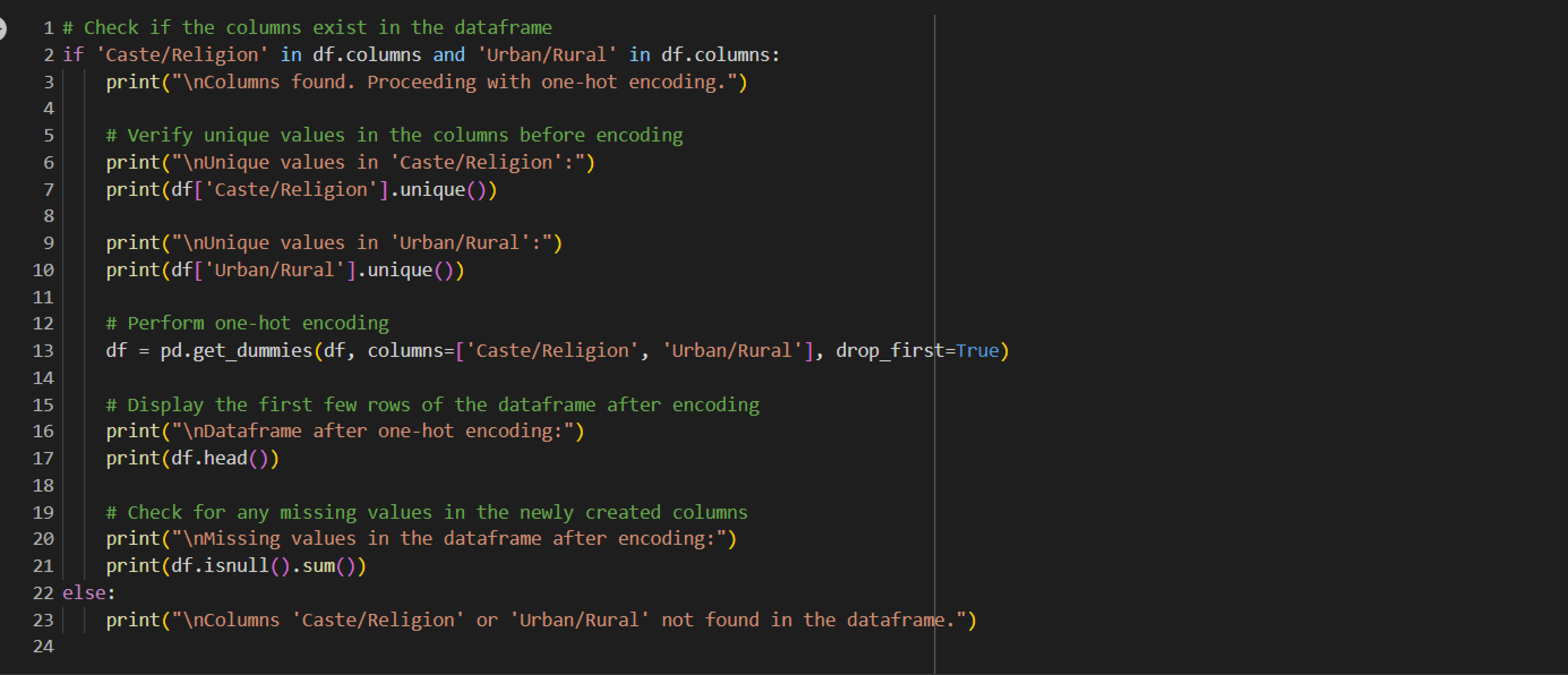


```python
# Verify the unique values in the columns before mapping
print("\nUnique values in 'Education Level' before mapping:")
print(df['Education Level'].unique())

print("\nUnique values in 'Marriage Type' before mapping:")
print(df['Marriage Type'].unique())

# Perform ordinal encoding
df['Education Level'] = df['Education Level'].map({'Low': 0, 'Medium': 1, 'High': 2})
df['Marriage Type'] = df['Marriage Type'].map({'Arranged': 0, 'Love': 1})

# Check for any unmapped or missing values
print("\nUnique values in 'Education Level' after mapping:")
print(df['Education Level'].unique())

print("\nUnique values in 'Marriage Type' after mapping:")
print(df['Marriage Type'].unique())

# Check for null values after mapping
missing_values = df[['Education Level', 'Marriage Type']].isnull().sum()

if missing_values.any():
    print(f"\nMissing values found after mapping: \n{missing_values}")
else:
    print("\nNo missing values in the mapped columns.")
```

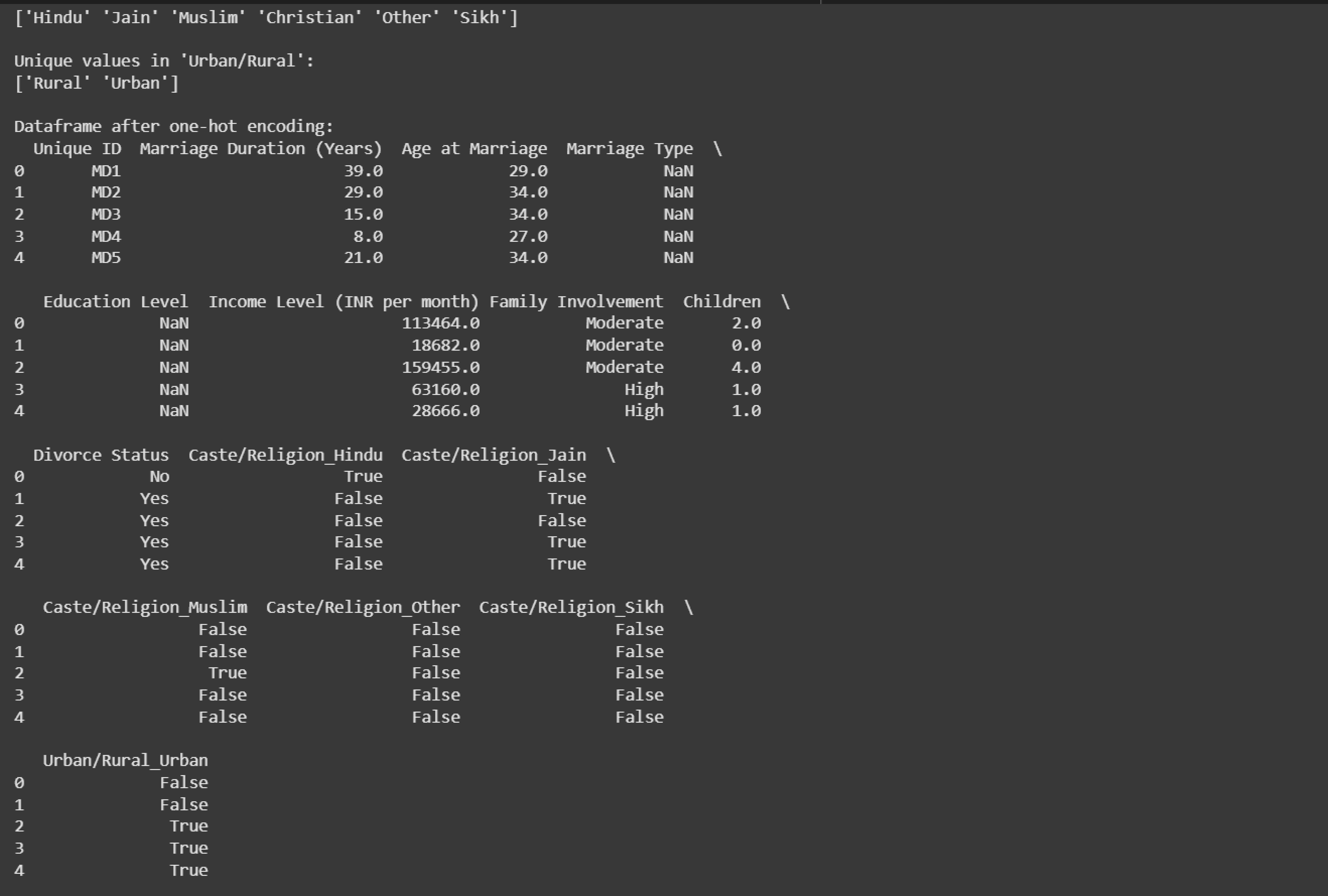


[54]

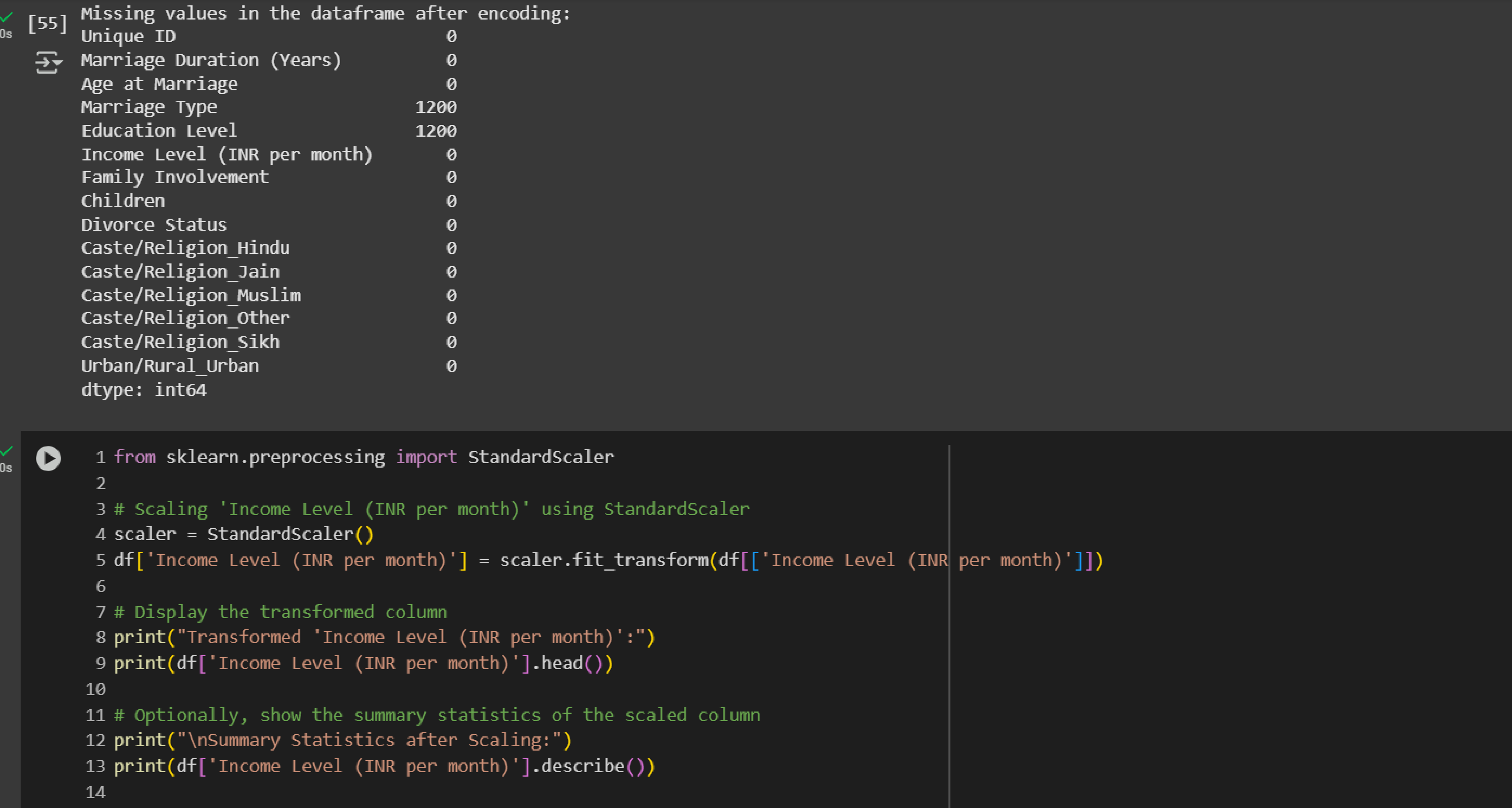
```
Unique values in 'Education Level' before mapping:
[nan]

Unique values in 'Marriage Type' before mapping:
[1 0]

Unique values in 'Education Level' after mapping:
[nan]

Unique values in 'Marriage Type' after mapping:
[nan]

Missing values found after mapping:
Education Level    1200
Marriage Type      1200
dtype: int64
```



```
# Check if the columns exist in the dataframe
if 'Caste/Religion' in df.columns and 'Urban/Rural' in df.columns:
    print("\nColumns found. Proceeding with one-hot encoding.")

    # Verify unique values in the columns before encoding
    print("\nUnique values in 'Caste/Religion':")
    print(df['Caste/Religion'].unique())

    print("\nUnique values in 'Urban/Rural':")
    print(df['Urban/Rural'].unique())

    # Perform one-hot encoding
    df = pd.get_dummies(df, columns=['Caste/Religion', 'Urban/Rural'], drop_first=True)

    # Display the first few rows of the dataframe after encoding
    print("\nDataframe after one-hot encoding:")
    print(df.head())

    # Check for any missing values in the newly created columns
    print("\nMissing values in the dataframe after encoding:")
    print(df.isnull().sum())
else:
    print("\nColumns 'Caste/Religion' or 'Urban/Rural' not found in the dataframe.")

```

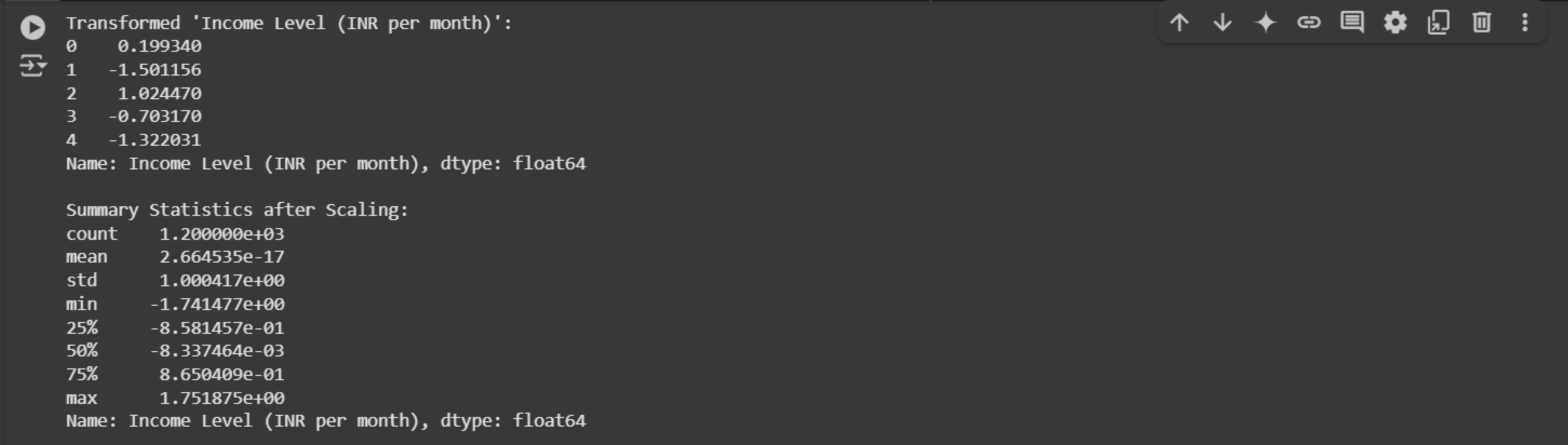


```
['Hindu' 'Jain' 'Muslim' 'Christian' 'Other' 'Sikh']

Unique values in 'Urban/Rural':
['Rural' 'Urban']

Dataframe after one-hot encoding:
  Unique ID  Marriage Duration (Years)  Age at Marriage  Marriage Type  \
0       MD1                       39.0             29.0            NaN   
1       MD2                       29.0             34.0            NaN   
2       MD3                       15.0             34.0            NaN   
3       MD4                        8.0             27.0            NaN   
4       MD5                       21.0             34.0            NaN   

   Education Level  Income Level (INR per month) Family Involvement  Children  \
0              NaN                      113464.0           Moderate       2.0   
1              NaN                       18682.0           Moderate       0.0   
2              NaN                      159455.0           Moderate       4.0   
3              NaN                       63160.0               High       1.0   
4              NaN                       28666.0               High       1.0   

  Divorce Status  Caste/Religion_Hindu  Caste/Religion_Jain  \
0             No                  True                False   
1            Yes                 False                 True   
2            Yes                 False                False   
3            Yes                 False                 True   
4            Yes                 False                 True   

   Caste/Religion_Muslim  Caste/Religion_Other  Caste/Religion_Sikh  \
0                  False                 False                False   
1                  False                 False                False   
2                   True                 False                False   
3                  False                 False                False   
4                  False                 False                False   

   Urban/Rural_Urban  
0              False  
1              False  
2               True  
3               True  
4               True  
```

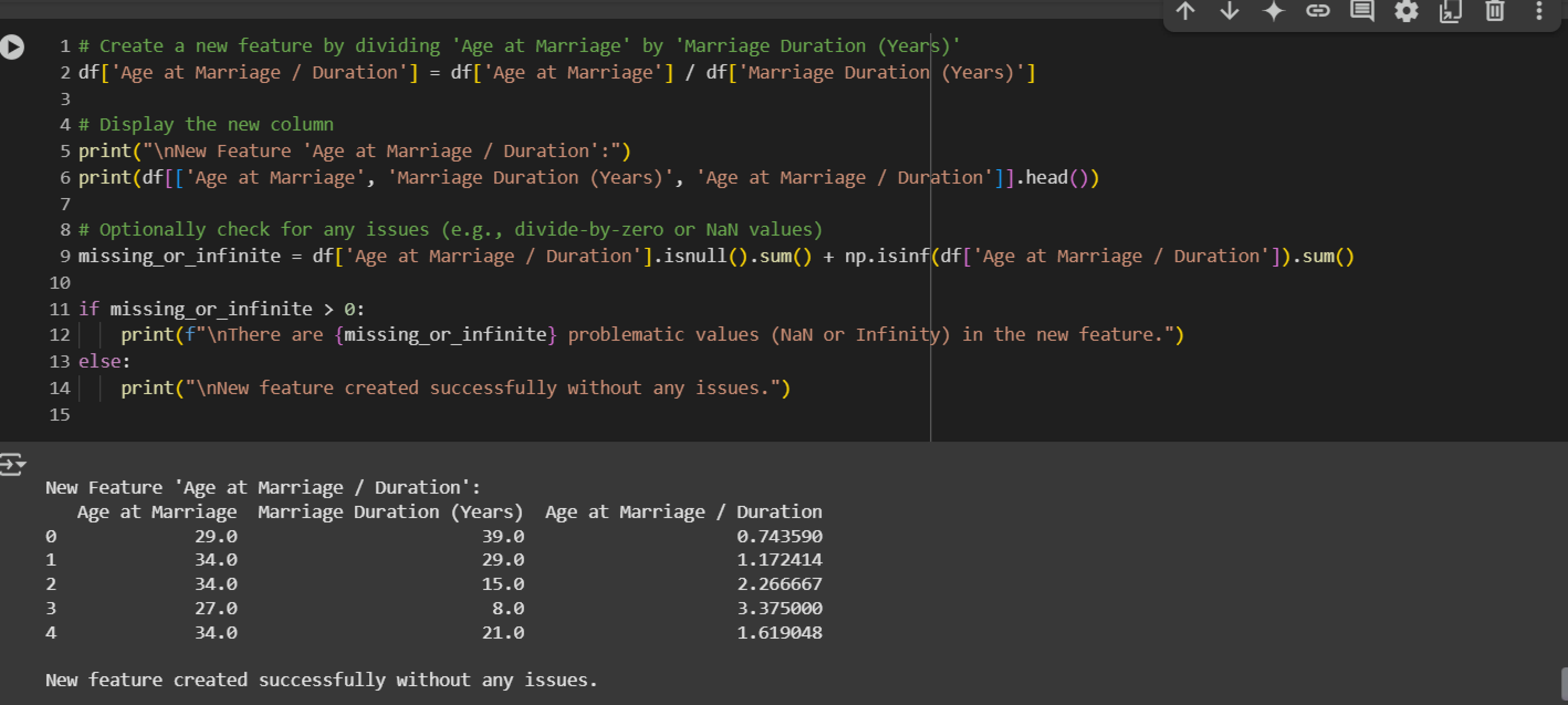


```
Missing values in the dataframe after encoding:
Unique ID                          0
Marriage Duration (Years)          0
Age at Marriage                    0
Marriage Type                   1200
Education Level                 1200
Income Level (INR per month)       0
Family Involvement                 0
Children                           0
Divorce Status                     0
Caste/Religion_Hindu               0
Caste/Religion_Jain                0
Caste/Religion_Muslim              0
Caste/Religion_Other               0
Caste/Religion_Sikh                0
Urban/Rural_Urban                  0
dtype: int64
```

```python
from sklearn.preprocessing import StandardScaler

# Scaling 'Income Level (INR per month)' using StandardScaler
scaler = StandardScaler()
df['Income Level (INR per month)'] = scaler.fit_transform(df[['Income Level (INR per month)']])

# Display the transformed column
print("Transformed 'Income Level (INR per month)':")
print(df['Income Level (INR per month)'].head())

# Optionally, show the summary statistics of the scaled column
print("\nSummary Statistics after Scaling:")
print(df['Income Level (INR per month)'].describe())

```

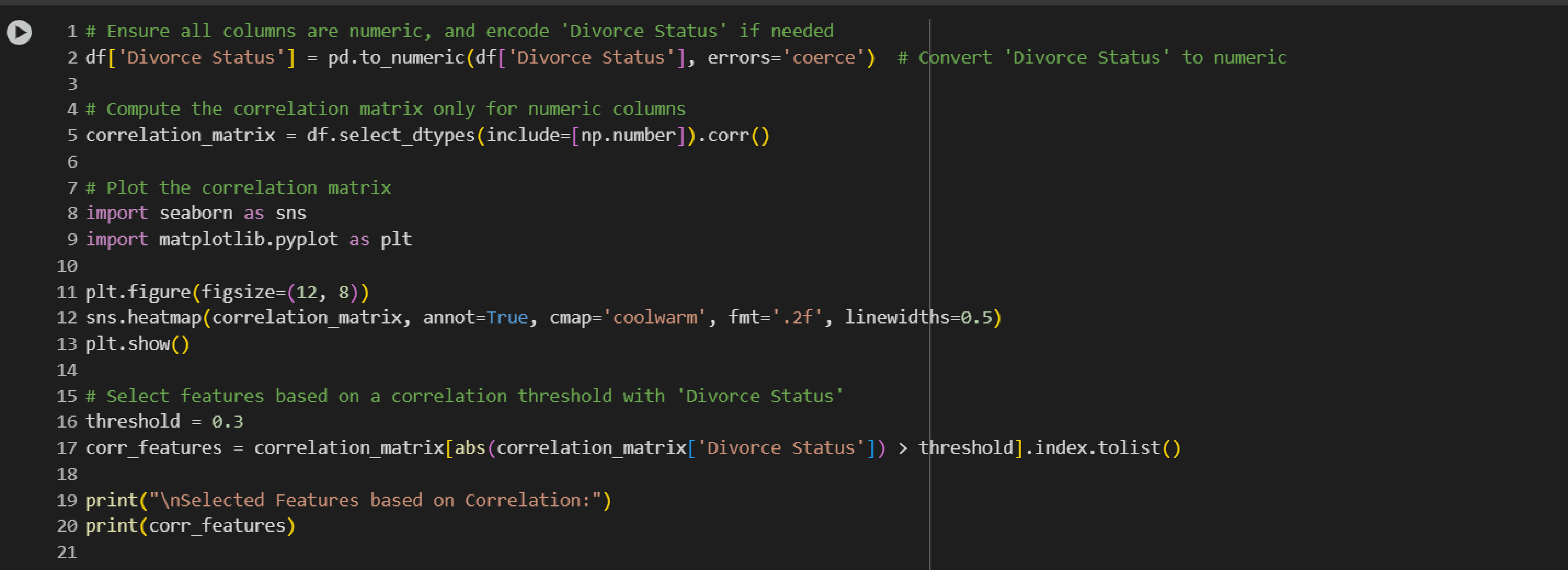


```
Transformed 'Income Level (INR per month)':
0    0.199340
1   -1.501156
2    1.024470
3   -0.703170
4   -1.322031
Name: Income Level (INR per month), dtype: float64

Summary Statistics after Scaling:
count    1.200000e+03
mean     2.664535e-17
std      1.000417e+00
min     -1.741477e+00
25%     -8.581457e-01
50%     -8.337464e-03
75%      8.650409e-01
max      1.751875e+00
Name: Income Level (INR per month), dtype: float64
```



```python
# Create a new feature by dividing 'Age at Marriage' by 'Marriage Duration (Years)'
df['Age at Marriage / Duration'] = df['Age at Marriage'] / df['Marriage Duration (Years)']

# Display the new column
print("\nNew Feature 'Age at Marriage / Duration':")
print(df[['Age at Marriage', 'Marriage Duration (Years)', 'Age at Marriage / Duration']].head())

# Optionally check for any issues (e.g., divide-by-zero or NaN values)
missing_or_infinite = df['Age at Marriage / Duration'].isnull().sum() + np.isinf(df['Age at Marriage / Duration']).sum()

if missing_or_infinite > 0:
    print(f"\nThere are {missing_or_infinite} problematic values (NaN or Infinity) in the new feature.")
else:
    print("\nNew feature created successfully without any issues.")

```

```

New Feature 'Age at Marriage / Duration':
   Age at Marriage  Marriage Duration (Years)  Age at Marriage / Duration
0             29.0                       39.0                    0.743590
1             34.0                       29.0                    1.172414
2             34.0                       15.0                    2.266667
3             27.0                        8.0                    3.375000
4             34.0                       21.0                    1.619048

New feature created successfully without any issues.
```



```python
# Ensure all columns are numeric, and encode 'Divorce Status' if needed
df['Divorce Status'] = pd.to_numeric(df['Divorce Status'], errors='coerce')  # Convert 'Divorce Status' to numeric

# Compute the correlation matrix only for numeric columns
correlation_matrix = df.select_dtypes(include=[np.number]).corr()

# Plot the correlation matrix
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.show()

# Select features based on a correlation threshold with 'Divorce Status'
threshold = 0.3
corr_features = correlation_matrix[abs(correlation_matrix['Divorce Status']) > threshold].index.tolist()

print("\nSelected Features based on Correlation:")
print(corr_features)
```