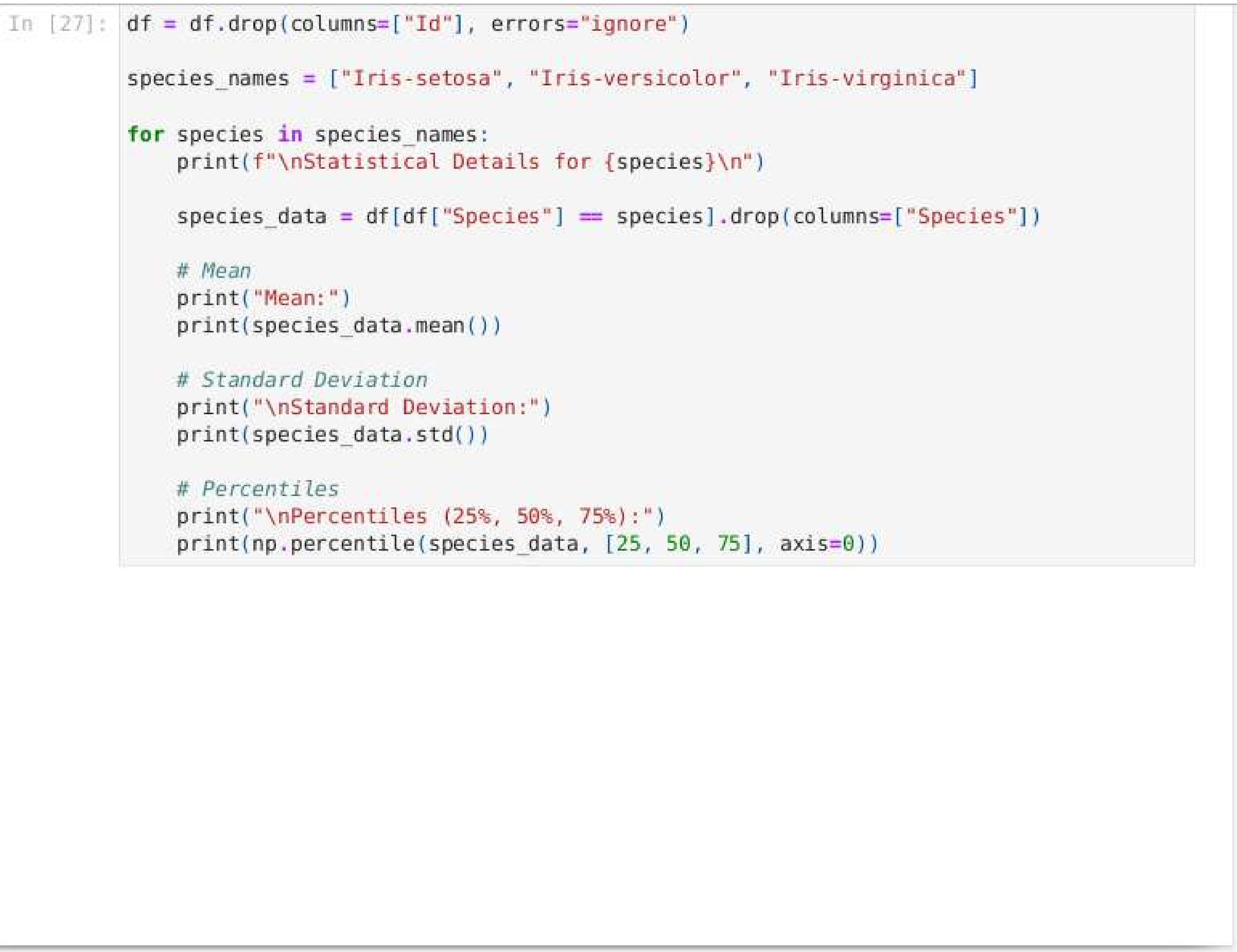
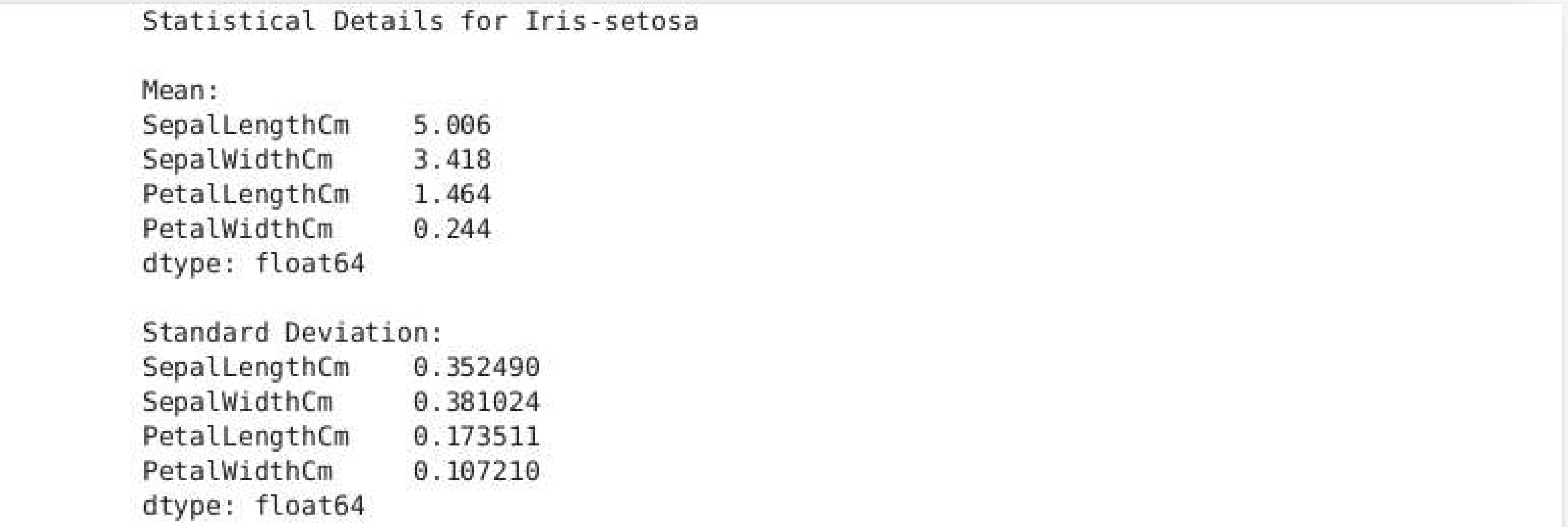
```
Statistical Details for Iris-versicolor

Mean:
SepalLengthCm    5.936
SepalWidthCm     2.770
PetalLengthCm    4.260
PetalWidthCm     1.326
dtype: float64

Standard Deviation:
SepalLengthCm    0.516171
SepalWidthCm     0.313798
PetalLengthCm    0.469911
PetalWidthCm     0.197753
dtype: float64

Percentiles (25%, 50%, 75%):
[[5.6   2.525 4.    1.2  ]
 [5.9   2.8   4.35  1.3  ]
 [6.3   3.    4.6   1.5  ]]

Statistical Details for Iris-virginica

Mean:
SepalLengthCm    6.588
SepalWidthCm     2.974
PetalLengthCm    5.552
PetalWidthCm     2.026
dtype: float64

Standard Deviation:
SepalLengthCm    0.635880
SepalWidthCm     0.322497
PetalLengthCm    0.551895
PetalWidthCm     0.274650
dtype: float64

Percentiles (25%, 50%, 75%):
[[6.225 2.8   5.1   1.8  ]
 [6.5   3.    5.55  2.   ]
 [6.9   3.175 5.875 2.3  ]]
```



```
In [27]: df = df.drop(columns=["Id"], errors="ignore")

         species_names = ["Iris-setosa", "Iris-versicolor", "Iris-virginica"]

         for species in species_names:
             print(f"\nStatistical Details for {species}\n")

             species_data = df[df["Species"] == species].drop(columns=["Species"])

             # Mean
             print("Mean:")
             print(species_data.mean())

             # Standard Deviation
             print("\nStandard Deviation:")
             print(species_data.std())

             # Percentiles
             print("\nPercentiles (25%, 50%, 75%):")
             print(np.percentile(species_data, [25, 50, 75], axis=0))
```



```
Statistical Details for Iris-setosa

Mean:
SepalLengthCm    5.006
SepalWidthCm     3.418
PetalLengthCm    1.464
PetalWidthCm     0.244
dtype: float64

Standard Deviation:
SepalLengthCm    0.352490
SepalWidthCm     0.381024
PetalLengthCm    0.173511
PetalWidthCm     0.107210
dtype: float64
```

In [21]:

```
summary_statistics = df.groupby("education")["age"].agg(
    Mean="mean",
    Median="median",
    Minimum="min",
    Maximum="max",
    Standard_Deviation="std"
)

print("\nSummary Statistics of Age grouped by Education:\n")
summary_statistics
```

```
Summary Statistics of Age grouped by Education:
```

Out[21]:

	Mean	Median	Minimum	Maximum	Standard_Deviation
education					
10th	37.429796	34.0	17	90	16.720713
11th	32.355745	28.0	17	90	15.545485
12th	32.000000	28.0	17	79	14.334625
1st-4th	46.142857	46.0	19	90	15.615625
5th-6th	42.885886	42.0	17	84	15.557285
7th-8th	48.445820	50.0	17	90	16.092350
9th	41.060311	39.0	17	90	15.946862
Assoc-acdm	37.381443	36.0	19	90	11.095177
Assoc-voc	38.553546	37.0	19	84	11.631300
Bachelors	38.904949	37.0	19	90	11.912210
Doctorate	47.702179	47.0	24	80	11.784716
HS-grad	38.974479	37.0	17	90	13.541524
Masters	44.049913	43.0	18	90	11.068935
Preschool	42.764706	41.0	19	75	15.126914
Prof-school	44.746528	43.0	25	90	11.962477
Some-college	35.756275	34.0	17	90	13.474051

In [23]:

```
df = pd.read_csv("Iris.csv")
df
```

In [17]:

```python
import pandas as pd
import numpy as np

df = pd.read_csv("adult.csv")
df
```

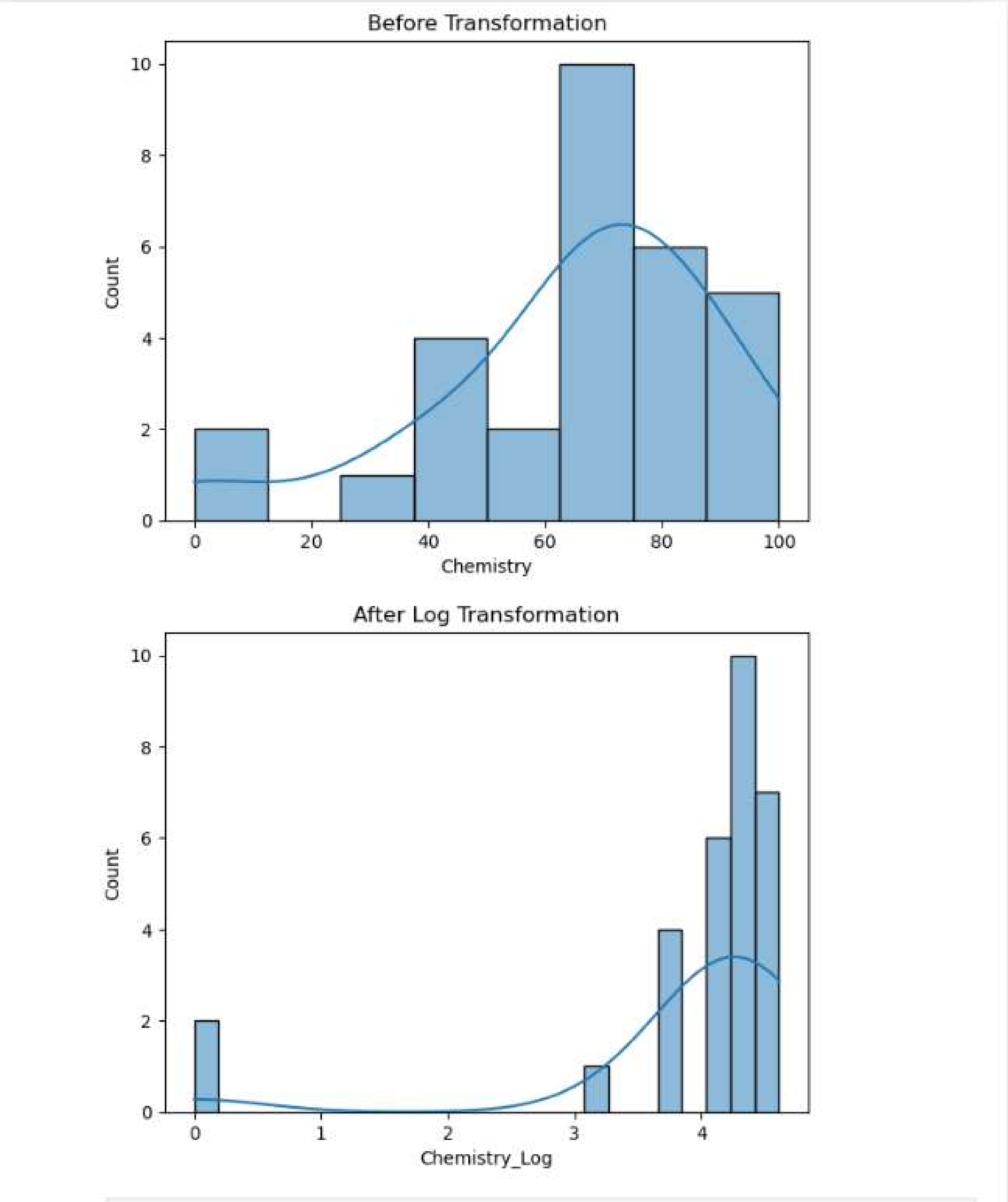
Out[17]:

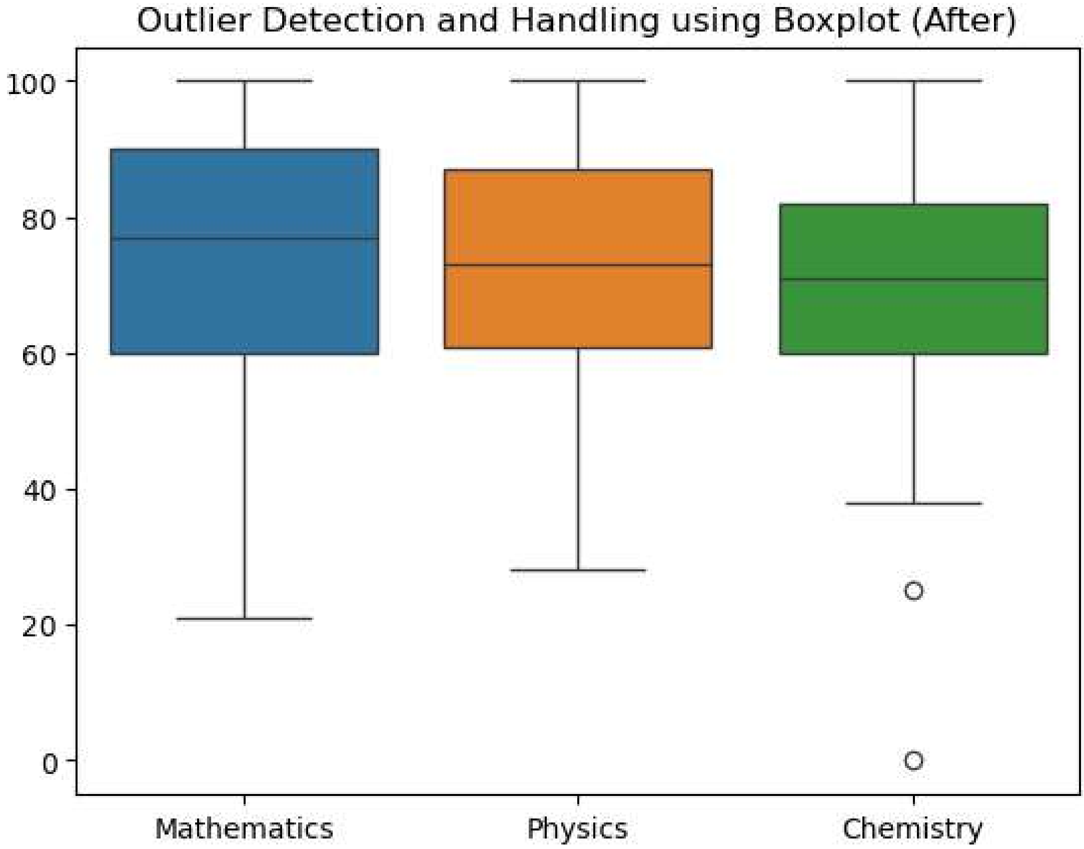
	age	workclass	fnlwgt	education	education.num	marital.status	occupation
0	90	?	77053	HS-grad	9	Widowed	?
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial
2	66	?	186061	Some-college	10	Widowed	?
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct
4	41	Private	264663	Some-college	10	Separated	Prof-specialty
...	...	...	...	...	...	...	...
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical

32561 rows × 15 columns

In [19]:

```python
df = df[["education", "age"]]
age_list_by_education = df.groupby("education")["age"].apply(list)
print("List of Age values grouped by Education:\n")
age_list_by_education
```


Before Transformation
Count
0
2
4
6
8
10
0
20
40
60
80
100
Chemistry
After Log Transformation
Count
0
2
4
6
8
10
0
1
2
3
4
Chemistry_Log

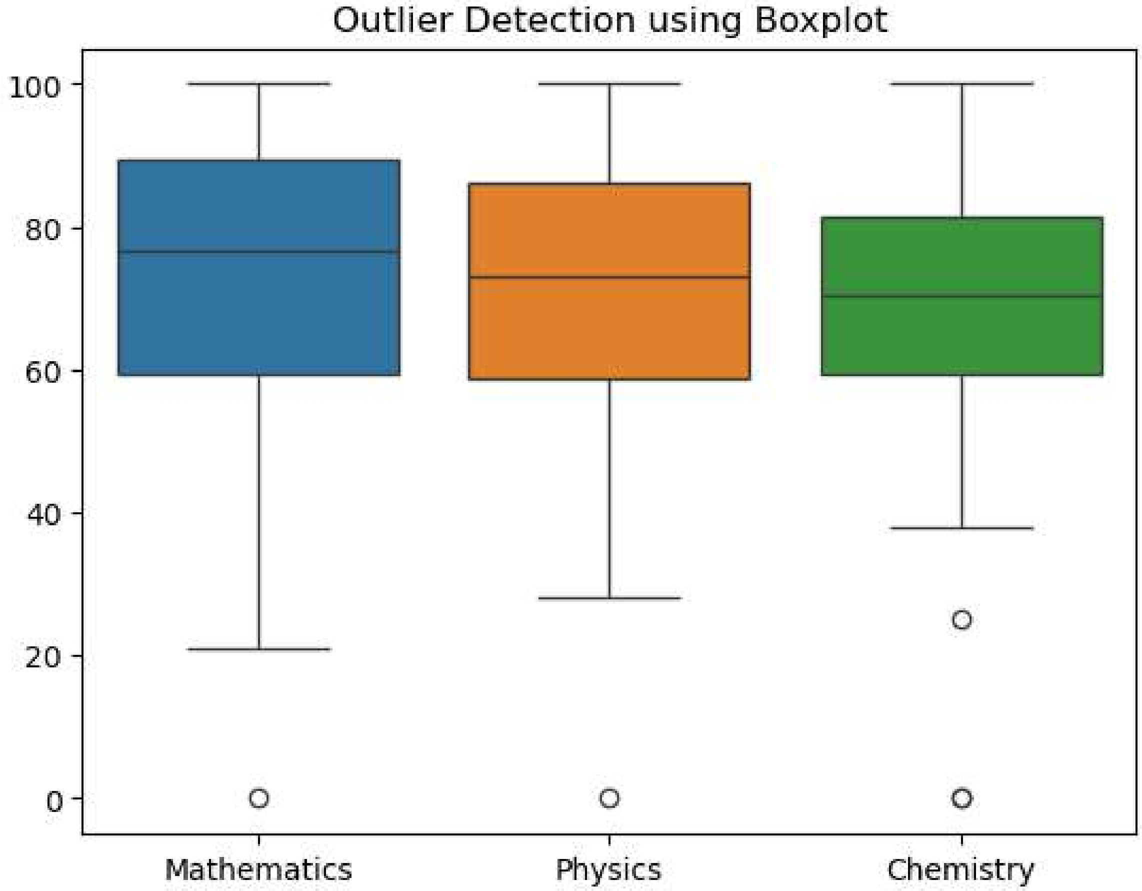


```
In [58]: df['Chemistry'].skew()
```

```
Out[58]: -1.1507378639285284
```

```
In [60]: df['Chemistry_Log'] = np.log(df['Chemistry'] + 1)
```

```
In [62]: sns.histplot(df['Chemistry'], kde=True)
         plt.title("Before Transformation")
         plt.show()

         sns.histplot(df['Chemistry_Log'], kde=True)
         plt.title("After Log Transformation")
         plt.show()
```



[40]:
```
Q1 = df['Mathematics'].quantile(0.25)
Q3 = df['Mathematics'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df['Mathematics'] = np.where(df['Mathematics'] > upper, upper, np.where(df['Mathemati
```

[71]:
```
z_scores = np.abs(stats.zscore(df['Physics']))
df = df[z_scores < 2.5]
```

[73]:
```
sns.boxplot(data=df[['Mathematics','Physics','Chemistry']])
plt.title("Outlier Detection and Handling using Boxplot (After)")
plt.show()
```

In [22]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

In [30]:

```python
data = {
    'StudentId': range(1, 31),
    'Name': [
        'Abhishek Podi','Abhishek Bachhan','Bikash Pandey','Anish Joshi','Abhay Sharm
        'Ashish Song','Ajay Cinthol','Ashutosh Song','Ayush Roll17','Abhishek Poda',
        'Rohit Kumar','Neha Singh','Pooja Patil','Suresh Mehta','Kunal Shah',
        'Aman Verma','Riya Desai','Vikas Rao','Sneha Nair','Mohit Jain',
        'Kriti Malhotra','Arjun Singh','Priya Kapoor','Nitin Yadav','Shreya Bose',
        'Deepak Gupta','Simran Kaur','Rahul Mishra','Tina Roy','Manish Tiwari'
    ],
    'Mathematics': [
        98, 85, 76, 23, np.nan, 100, 125, 55, 21, 88,
        92, 81, 77, 66, 59, 101, 34, 48, 72, 83,
        150, 95, 67, 29, 74, -20, 90, 86, 60, 79
    ],
    'Physics': [
        78, 55, 96, 150, 46, 66, np.nan, 35, 81, 78,
        88, 72, 69, 58, 61, 120, 42, 39, 75, 84,
        160, 91, 70, 28, 73, -10, 89, 87, 62, 80
    ],
    'Chemistry': [
        -95, 45, np.nan, 73, 76, 79, 86, 65, 41, np.nan,
        88, 74, 71, 60, 63, 110, 38, 44, 70, 82,
        145, 90, 68, 25, 72, -30, 91, 85, 59, 77
    ]
}

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

In [53]:

```
# Concatenate along columns (axis=1)
concatenated_axis1 = pd.concat([df1, df2], axis=1)
print("\nConcatenation along columns (axis=1):\n")
concatenated_axis1.head()
```

```
Concatenation along columns (axis=1):
```

Out[53]:

	PassengerId	Name	Age	Survived	PassengerId	Pclass	Sex	Fare
0	1	Braund, Mr. Owen Harris	22.0	0	1	3	male	7.2500
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	2	1	female	71.2833
2	3	Heikkinen, Miss. Laina	26.0	1	3	3	female	7.9250
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	4	1	female	53.1000
4	5	Allen, Mr. William Henry	35.0	0	5	3	male	8.0500

In [73]:

```
# Calculate mean, median, and mode for numerical columns
print("\n----------- Mean -----------\n", titanic[['age', 'fare']].mean())
print("\n----------- Median -----------\n", titanic[['age', 'fare']].median())
print("\n----------- Mode -----------\n", titanic[['age', 'fare']].mode())
```

```
----------- Mean -----------
 age     29.699118
fare    32.204208
dtype: float64

----------- Median -----------
 age     28.0000
fare    14.4542
dtype: float64

----------- Mode -----------
     age  fare
0  24.0  8.05
```

In [67]:

```
# Calculate mid-range (average of max and min for each column)
mid_range = titanic[['age', 'fare']].apply(lambda x: (x.max() + x.min()) / 2)
print("\n----------- Mid-Range -----------\n", mid_range)
```

```
----------- Mid-Range -----------
 age      40.2100
fare    256.1646
dtype: float64
```

In [43]:

```python
# Left Join on 'PassengerId'
merged_left = pd.merge(df1, df2, on='PassengerId', how='left')
print("\nLeft Join on 'PassengerId':\n")
merged_left.head()
```

```
Left Join on 'PassengerId':
```

Out[43]:

	PassengerId	Name	Age	Survived	Pclass	Sex	Fare
0	1	Braund, Mr. Owen Harris	22.0	0	3	male	7.2500
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	1	female	71.2833
2	3	Heikkinen, Miss. Laina	26.0	1	3	female	7.9250
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	1	female	53.1000
4	5	Allen, Mr. William Henry	35.0	0	3	male	8.0500

In [45]:

```python
# Right Join on 'passengPassengerIder_id'
merged_right = pd.merge(df1, df2, on='PassengerId', how='right')

print("\nRight Join on 'PassengerId':\n")
merged_right.head()
```

```
Right Join on 'PassengerId':
```

Out[45]:

	PassengerId	Name	Age	Survived	Pclass	Sex	Fare
0	1	Braund, Mr. Owen Harris	22.0	0	3	male	7.2500
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	1	female	71.2833
2	3	Heikkinen, Miss. Laina	26.0	1	3	female	7.9250
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	1	female	53.1000
4	5	Allen, Mr. William Henry	35.0	0	3	male	8.0500

In [47]:

```python
# Outer Join on 'passenger_id'
merged_outer = pd.merge(df1, df2, on='PassengerId', how='outer')
print("\nOuter Join on 'PassengerId':\n")
merged_outer.head()
```

```
In [31]: # 1. Create df1: Identification and Outcome
         # Selecting specifically from the columns you listed
         df1 = df[['PassengerId', 'Name', 'Age', 'Survived']].copy()

         # 2. Create df2: Socio-Economic and Travel Data
         df2 = df[['PassengerId', 'Pclass', 'Sex', 'Fare']].copy()

         print("DataFrames successfully created!")
         display(df1.head())
         display(df2.head())
```

```
DataFrames successfully created!
```

	PassengerId	Name	Age	Survived
0	1	Braund, Mr. Owen Harris	22.0	0
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1
2	3	Heikkinen, Miss. Laina	26.0	1
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1
4	5	Allen, Mr. William Henry	35.0	0

	PassengerId	Pclass	Sex	Fare
0	1	3	male	7.2500
1	2	1	female	71.2833
2	3	3	female	7.9250
3	4	1	female	53.1000
4	5	3	male	8.0500

```
In [41]: # Inner Join on 'PassengerId'
         merged_inner = pd.merge(df1, df2, on='PassengerId', how='inner')
         print("\nInner Join on 'PassengerId':\n")
         merged_inner.head()
```

```
Inner Join on 'PassengerId':
```

Out[41]:

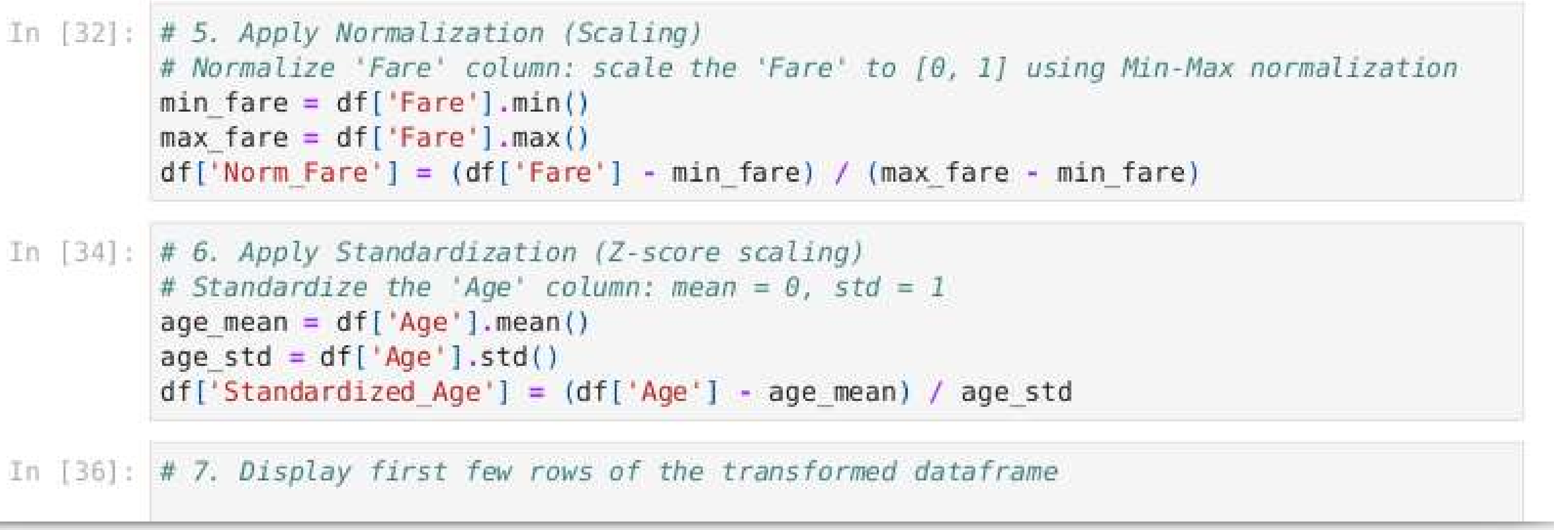
	PassengerId	Name	Age	Survived	Pclass	Sex	Fare
0	1	Braund, Mr. Owen Harris	22.0	0	3	male	7.2500
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	1	female	71.2833
2	3	Heikkinen, Miss. Laina	26.0	1	3	female	7.9250
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	1	female	53.1000
4	5	Allen, Mr. William Henry	35.0	0	3	male	8.0500

In [29]:
```
import pandas as pd
import seaborn as sns

# Read the Data with Pandas
df = pd.read_csv("train.csv")
df
```

Out[29]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7

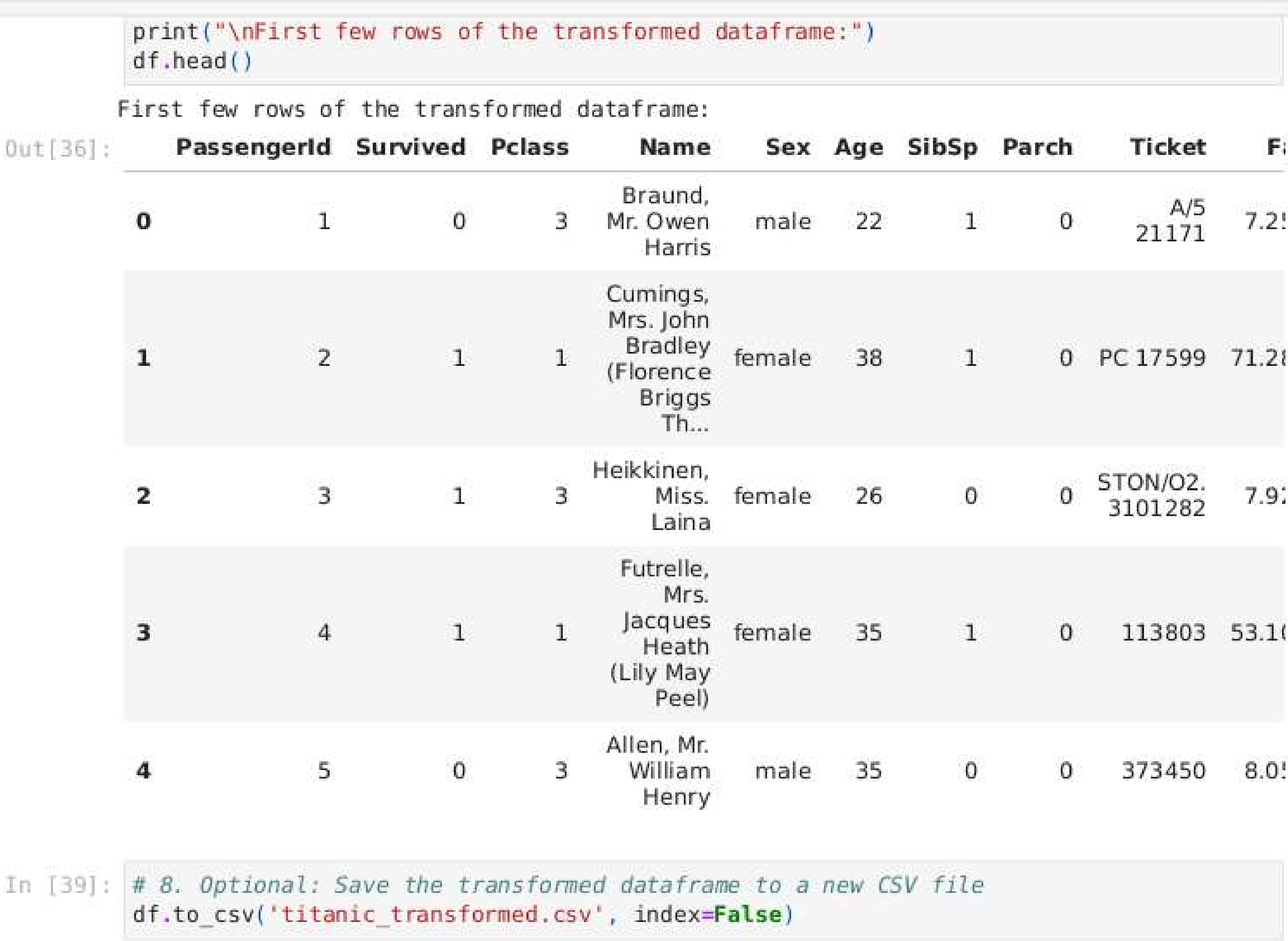
891 rows × 12 columns



```
In [32]: # 5. Apply Normalization (Scaling)
         # Normalize 'Fare' column: scale the 'Fare' to [0, 1] using Min-Max normalization
         min_fare = df['Fare'].min()
         max_fare = df['Fare'].max()
         df['Norm_Fare'] = (df['Fare'] - min_fare) / (max_fare - min_fare)
```

```
In [34]: # 6. Apply Standardization (Z-score scaling)
         # Standardize the 'Age' column: mean = 0, std = 1
         age_mean = df['Age'].mean()
         age_std = df['Age'].std()
         df['Standardized_Age'] = (df['Age'] - age_mean) / age_std
```

```
In [36]: # 7. Display first few rows of the transformed dataframe
```



```
         print("\nFirst few rows of the transformed dataframe:")
         df.head()
```

```
First few rows of the transformed dataframe:
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0

```
In [39]: # 8. Optional: Save the transformed dataframe to a new CSV file
         df.to_csv('titanic_transformed.csv', index=False)
```

In [10]:
```
# 1. Handling missing values

# Convert to numeric first (results in float64 due to NaNs and decimals)
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# To convert to 'Int64', we must first handle the decimals.
# We'll round them to the nearest whole number, then cast.
df['Age'] = df['Age'].round(0).astype('Int64')

# Fill missing 'Embarked' with the most frequent value
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

# Fill missing 'Fare' with the median value
df['Fare'] = df['Fare'].fillna(df['Fare'].median())
```

In [12]:
```
# 2. Convert non-numeric columns to category types to save memory
df['Pclass'] = df['Pclass'].astype('category')
df['Survived'] = df['Survived'].astype('category')
df['Sex'] = df['Sex'].astype('category')
df['Embarked'] = df['Embarked'].astype('category')
```

In [28]:
```
# 3. Check the data types after the conversion and filling missing values
print("Data types after conversion:")
df.dtypes
```

```
Data types after conversion:
```

Out[28]:
```
PassengerId            int64
Survived            category
Pclass              category
Name                  object
Sex                 category
Age                    Int64
SibSp                  int64
Parch                  int64
Ticket                object
Fare                 float64
Cabin                 object
Embarked            category
Norm_Fare            float64
Standardized_Age     Float64
dtype: object
```

In [30]:
```
# 4. Apply Aggregation: Example for 'Age' and 'Fare'
age_stats = df['Age'].agg(['mean', 'std', 'min', 'max'])
fare_stats = df['Fare'].agg(['mean', 'std', 'min', 'max'])

# Display the summary statistics
print("\nAge Stats:")
age_stats
print("\nFare Stats:")
```

In [2]:

```python
import pandas as pd
import numpy as np

# Load Titanic dataset
df = pd.read_csv("train.csv")

# Display the first few rows and info
display(df.head())
df.info()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
```

In [102…

```
csv_data.isnull()
```

Out[102…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	False	False	False	False	False	False	False	False	False	False	T
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	T
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	T
...	...	...	...	...	...	...	...	...	...	...	
886	False	False	False	False	False	False	False	False	False	False	T
887	False	False	False	False	False	False	False	False	False	False	Fa
888	False	False	False	False	False	True	False	False	False	False	T
889	False	False	False	False	False	False	False	False	False	False	Fa
890	False	False	False	False	False	False	False	False	False	False	T

891 rows × 12 columns

In [104…

```
csv_data.isna()
```

Out[104…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	False	False	False	False	False	False	False	False	False	False	T
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	T
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	T
...	...	...	...	...	...	...	...	...	...	...	
886	False	False	False	False	False	False	False	False	False	False	T
887	False	False	False	False	False	False	False	False	False	False	Fa
888	False	False	False	False	False	True	False	False	False	False	T
889	False	False	False	False	False	False	False	False	False	False	Fa
890	False	False	False	False	False	False	False	False	False	False	T

891 rows × 12 columns

In [97]:

```
# For numeric data, the result's index will include count, mean, std, min, max as wel
# For object data (e.g. strings or timestamps), the result's index will include count
# The top is the most common value. The freq is the most common value's frequency. Ti
csv_data.describe(include="all")
```

Out[97]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	89
unique	NaN	NaN	NaN	891	2	NaN	NaN	
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	
freq	NaN	NaN	NaN	1	577	NaN	NaN	
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	

In [98]:

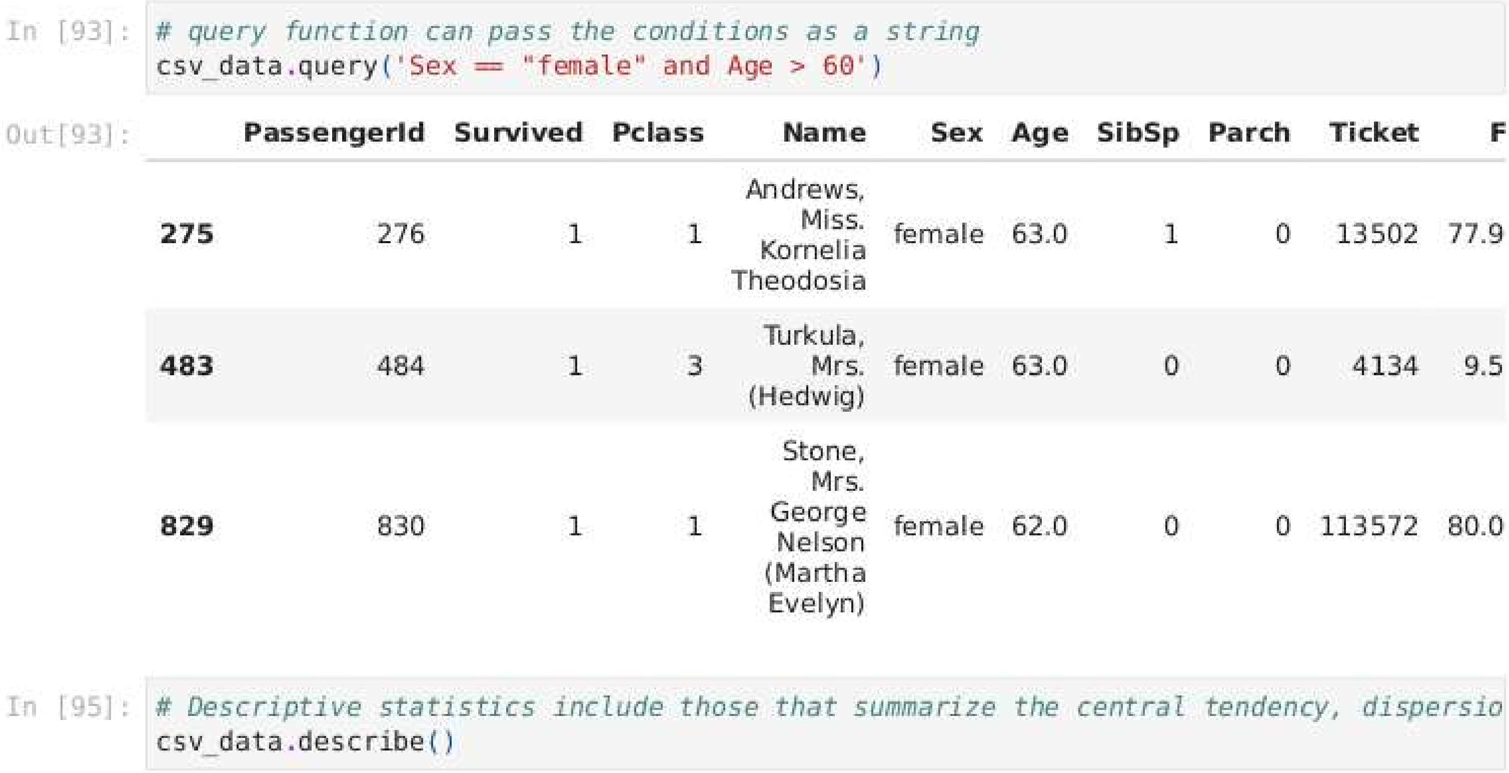
```
# Including only numeric columns
csv_data.describe(include=[np.number])
```

Out[98]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32

In [100…

```
# ignore non-numeric data for processing
csv_data.describe(exclude=["O"])
```



In [93]:

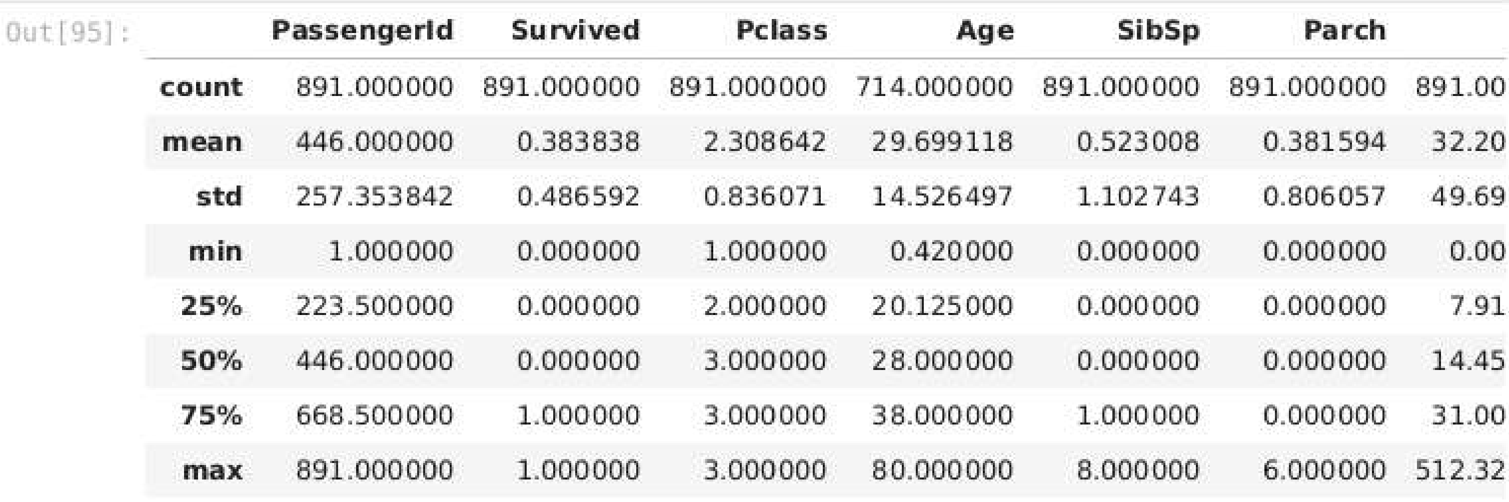
```
# query function can pass the conditions as a string
csv_data.query('Sex == "female" and Age > 60')
```

Out[93]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
275	276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9
483	484	1	3	Turkula, Mrs. (Hedwig)	female	63.0	0	0	4134	9.5
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0

In [95]:

```
# Descriptive statistics include those that summarize the central tendency, dispersio
csv_data.describe()
```



Out[95]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32

In [40]:

```
import pandas as pd
import numpy as np
```

In [42]:

```
# Read the Data with Pandas
csv_data = pd.read_csv("train.csv")
csv_data
```

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7