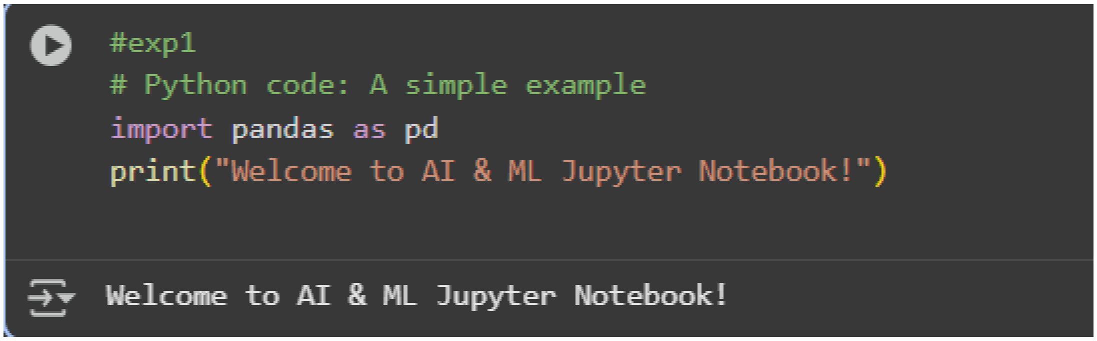
Name: JERON J

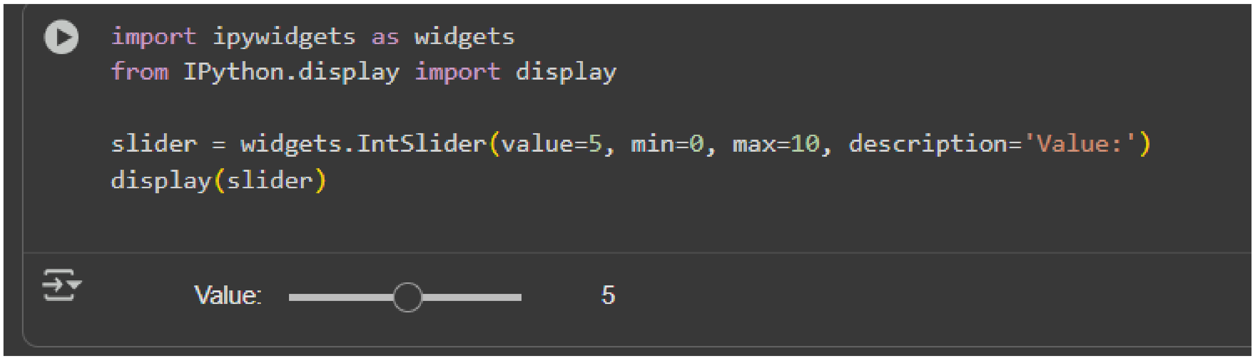
Roll no: 231501068

Exp: 1



```
#exp1
# Python code: A simple example
import pandas as pd
print("Welcome to AI & ML Jupyter Notebook!")
```

```
Welcome to AI & ML Jupyter Notebook!
```

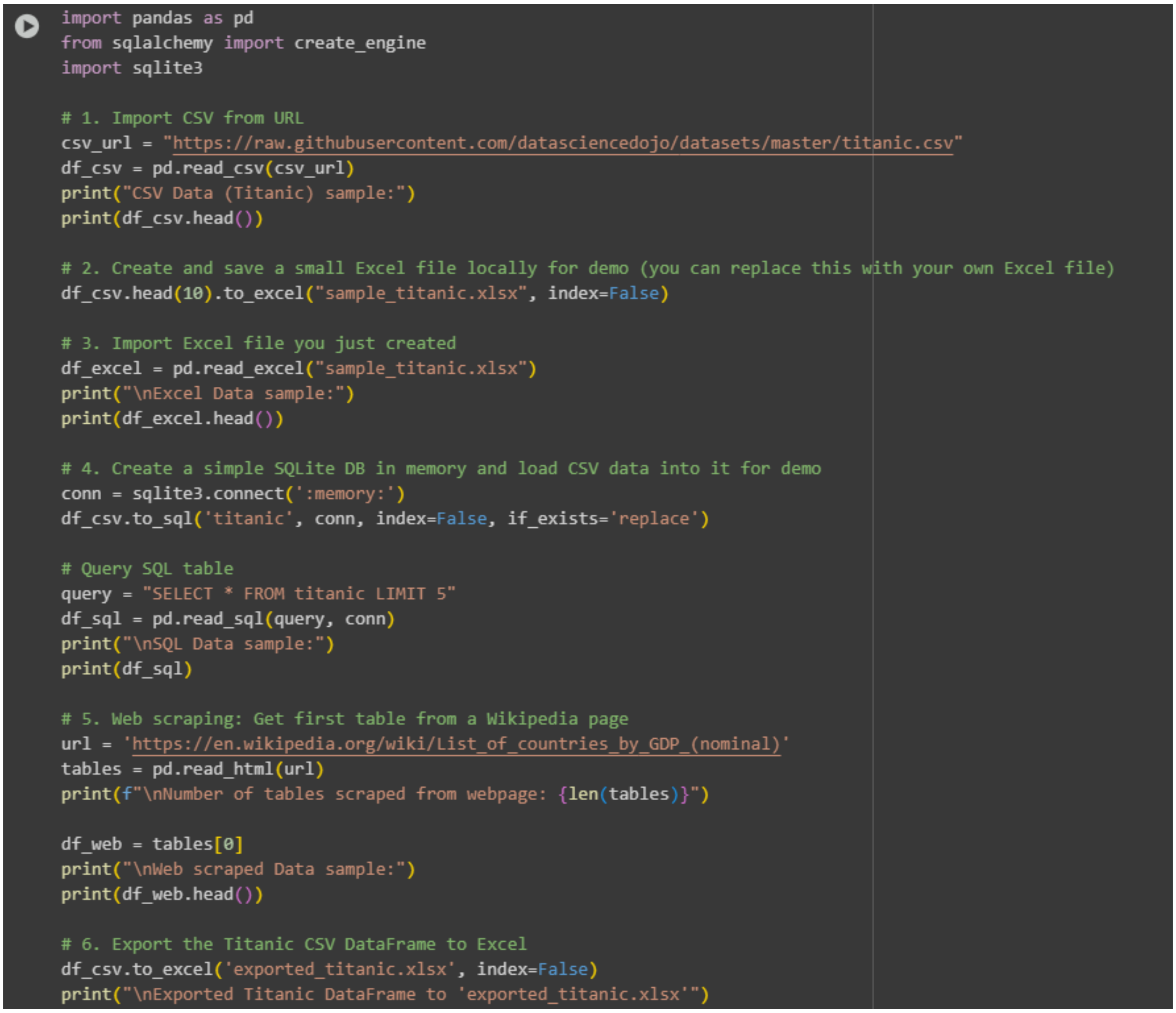


```
import ipywidgets as widgets
from IPython.display import display

slider = widgets.IntSlider(value=5, min=0, max=10, description='Value:')
display(slider)
```

```
Value:        5
```

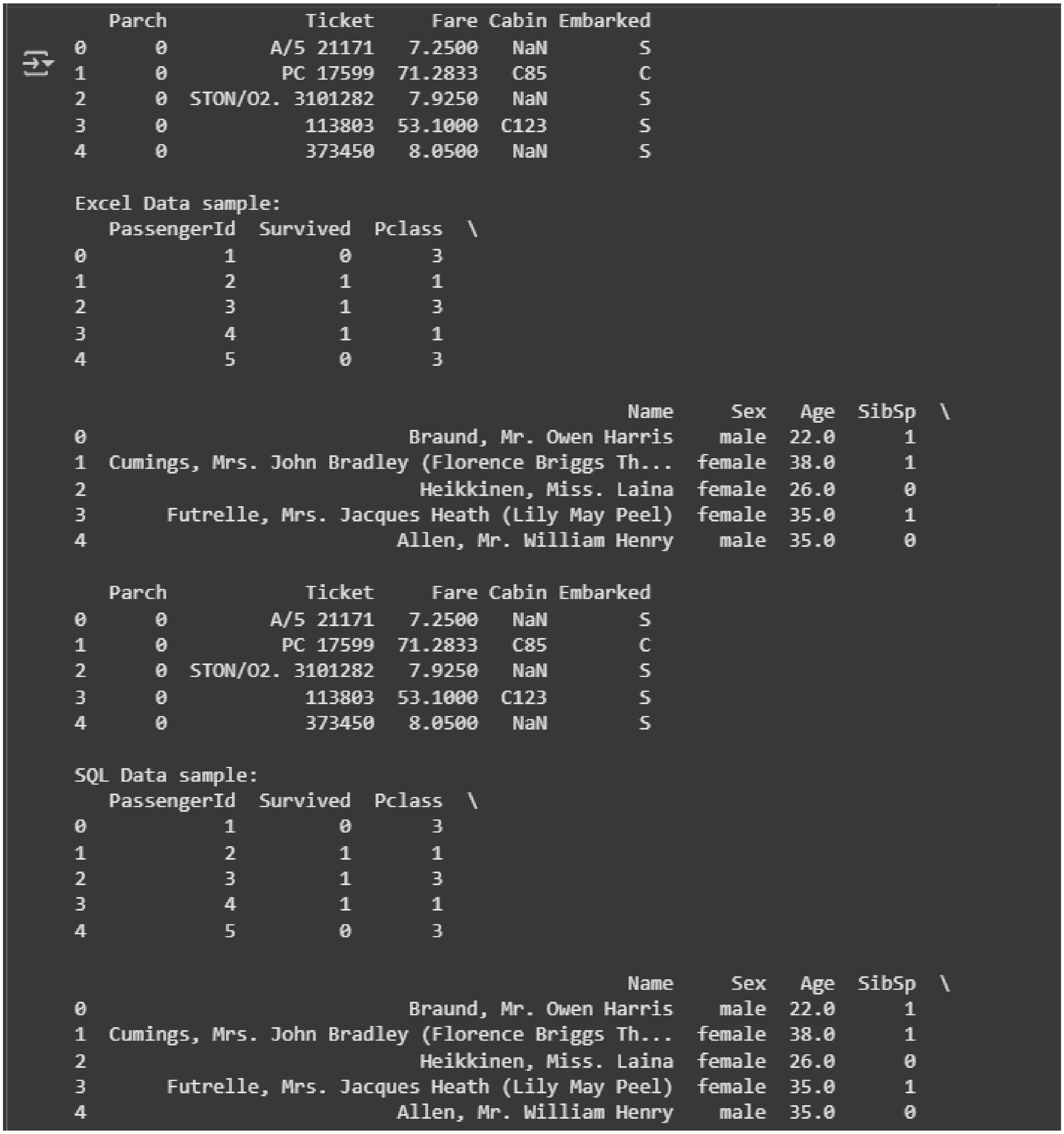
Exp: 2

Code:

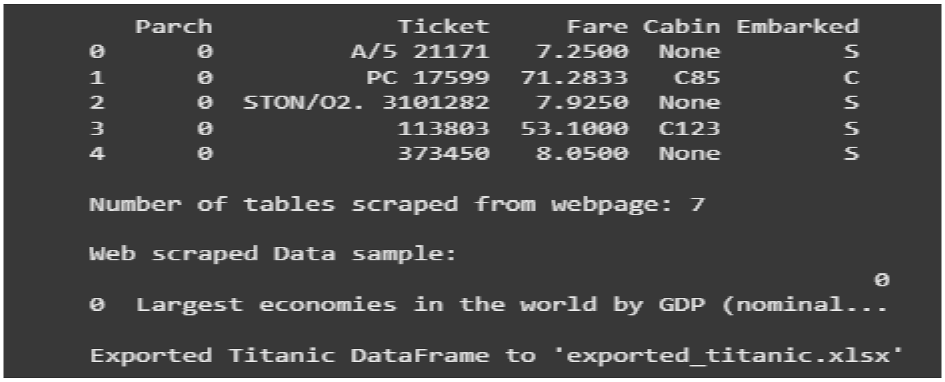


```python
import pandas as pd
from sqlalchemy import create_engine
import sqlite3

# 1. Import CSV from URL
csv_url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df_csv = pd.read_csv(csv_url)
print("CSV Data (Titanic) sample:")
print(df_csv.head())

# 2. Create and save a small Excel file locally for demo (you can replace this with your own Excel file)
df_csv.head(10).to_excel("sample_titanic.xlsx", index=False)

# 3. Import Excel file you just created
df_excel = pd.read_excel("sample_titanic.xlsx")
print("\nExcel Data sample:")
print(df_excel.head())

# 4. Create a simple SQLite DB in memory and load CSV data into it for demo
conn = sqlite3.connect(':memory:')
df_csv.to_sql('titanic', conn, index=False, if_exists='replace')

# Query SQL table
query = "SELECT * FROM titanic LIMIT 5"
df_sql = pd.read_sql(query, conn)
print("\nSQL Data sample:")
print(df_sql)

# 5. Web scraping: Get first table from a Wikipedia page
url = 'https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)'
tables = pd.read_html(url)
print(f"\nNumber of tables scraped from webpage: {len(tables)}")

df_web = tables[0]
print("\nWeb scraped Data sample:")
print(df_web.head())

# 6. Export the Titanic CSV DataFrame to Excel
df_csv.to_excel('exported_titanic.xlsx', index=False)
print("\nExported Titanic DataFrame to 'exported_titanic.xlsx'")
```

Output:



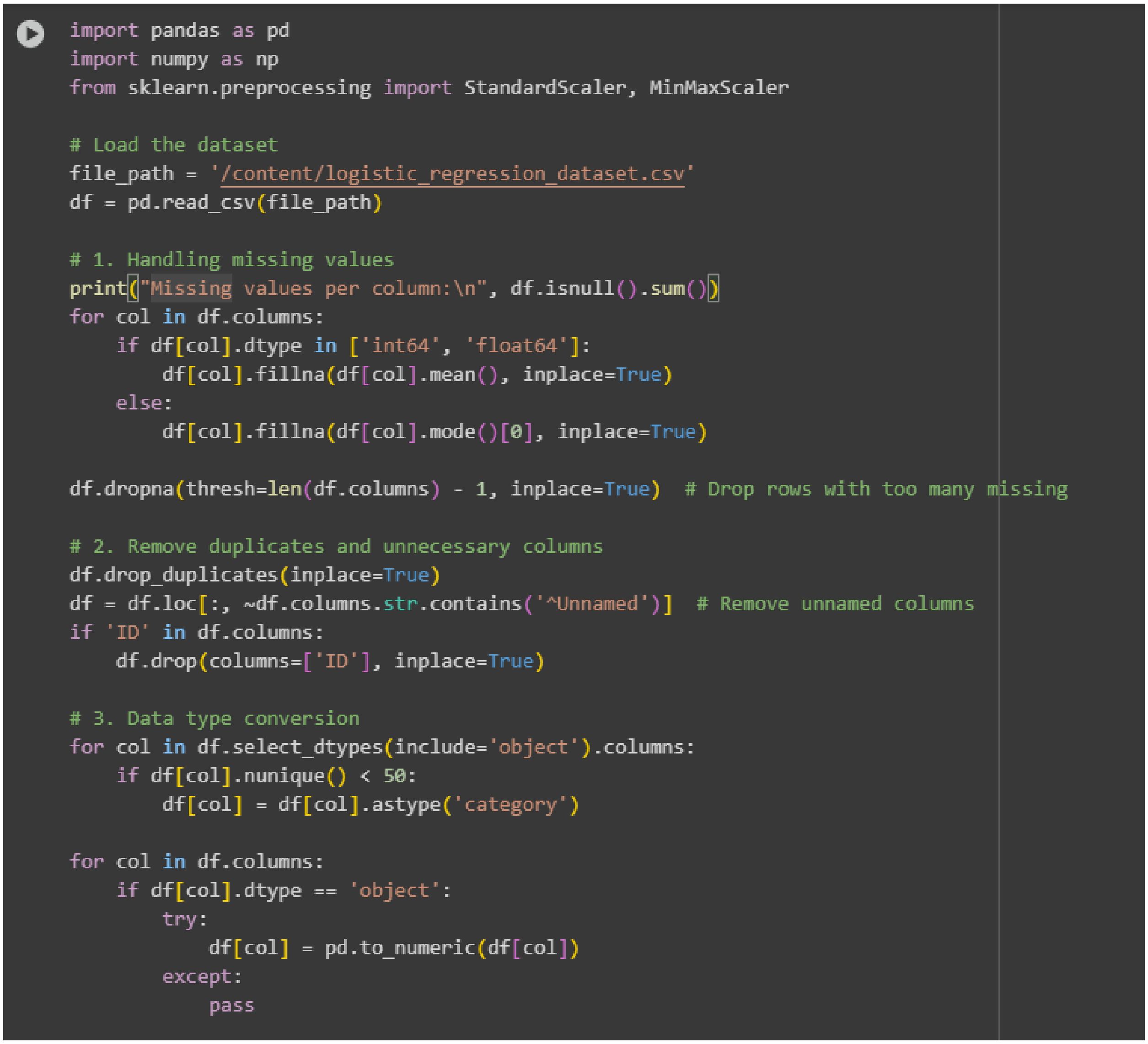
```
   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Excel Data sample:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

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

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

SQL Data sample:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

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



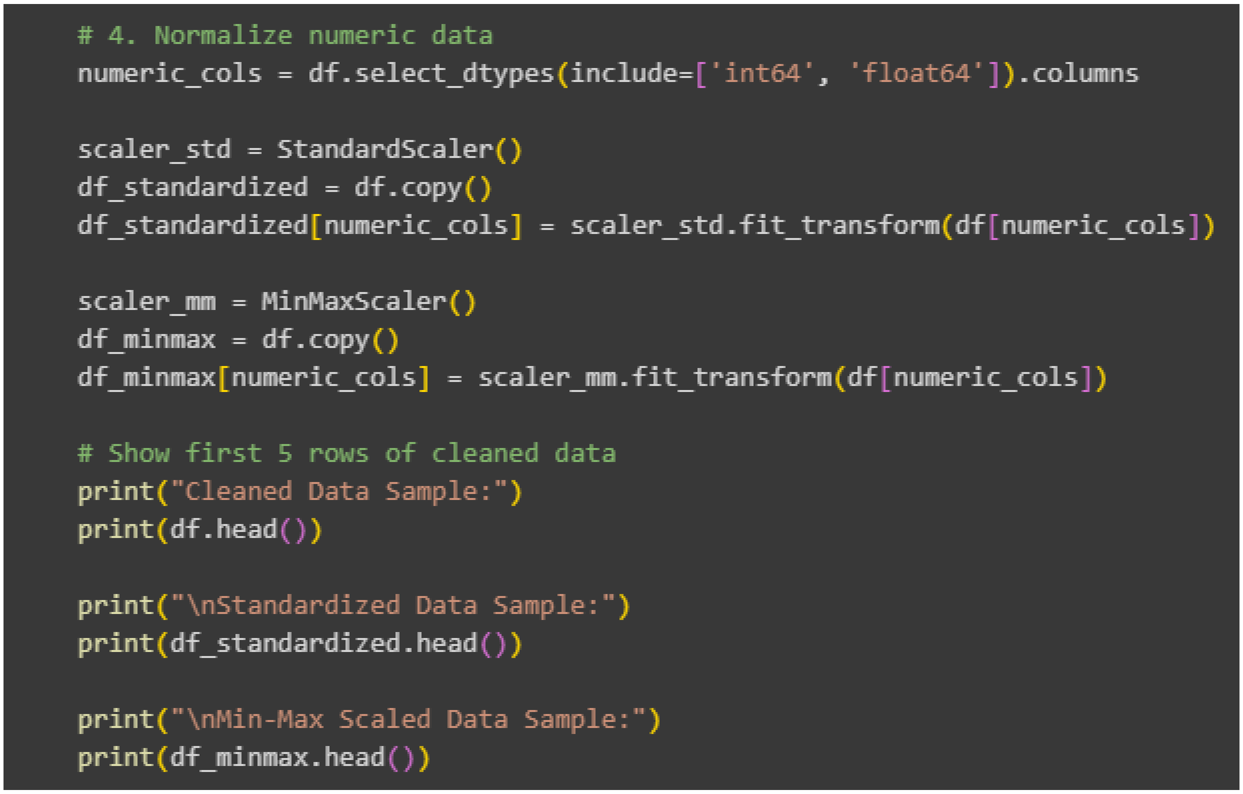
```
   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500  None        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250  None        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500  None        S

Number of tables scraped from webpage: 7

Web scraped Data sample:
                                                   0
0  Largest economies in the world by GDP (nominal...

Exported Titanic DataFrame to 'exported_titanic.xlsx'
```

Exp:3

Code:

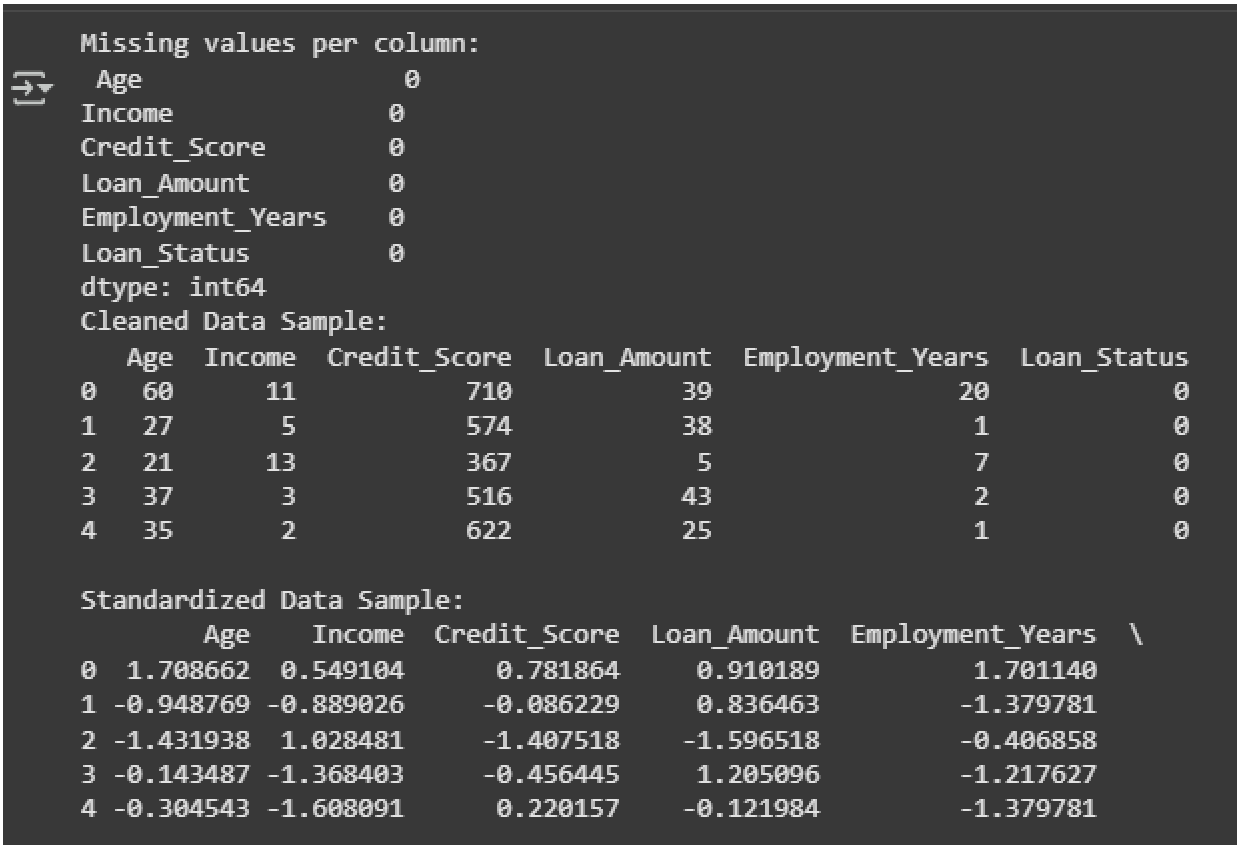


```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Load the dataset
file_path = '/content/logistic_regression_dataset.csv'
df = pd.read_csv(file_path)

# 1. Handling missing values
print("Missing values per column:\n", df.isnull().sum())
for col in df.columns:
    if df[col].dtype in ['int64', 'float64']:
        df[col].fillna(df[col].mean(), inplace=True)
    else:
        df[col].fillna(df[col].mode()[0], inplace=True)

df.dropna(thresh=len(df.columns) - 1, inplace=True)  # Drop rows with too many missing

# 2. Remove duplicates and unnecessary columns
df.drop_duplicates(inplace=True)
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]  # Remove unnamed columns
if 'ID' in df.columns:
    df.drop(columns=['ID'], inplace=True)

# 3. Data type conversion
for col in df.select_dtypes(include='object').columns:
    if df[col].nunique() < 50:
        df[col] = df[col].astype('category')

for col in df.columns:
    if df[col].dtype == 'object':
        try:
            df[col] = pd.to_numeric(df[col])
        except:
            pass
```

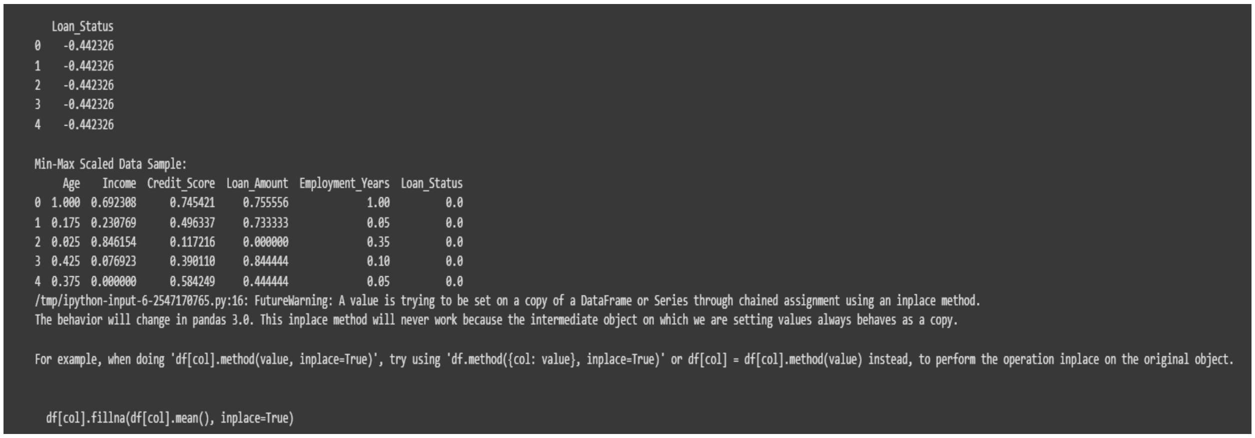


```
# 4. Normalize numeric data
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

scaler_std = StandardScaler()
df_standardized = df.copy()
df_standardized[numeric_cols] = scaler_std.fit_transform(df[numeric_cols])

scaler_mm = MinMaxScaler()
df_minmax = df.copy()
df_minmax[numeric_cols] = scaler_mm.fit_transform(df[numeric_cols])

# Show first 5 rows of cleaned data
print("Cleaned Data Sample:")
print(df.head())

print("\nStandardized Data Sample:")
print(df_standardized.head())

print("\nMin-Max Scaled Data Sample:")
print(df_minmax.head())
```

Output:



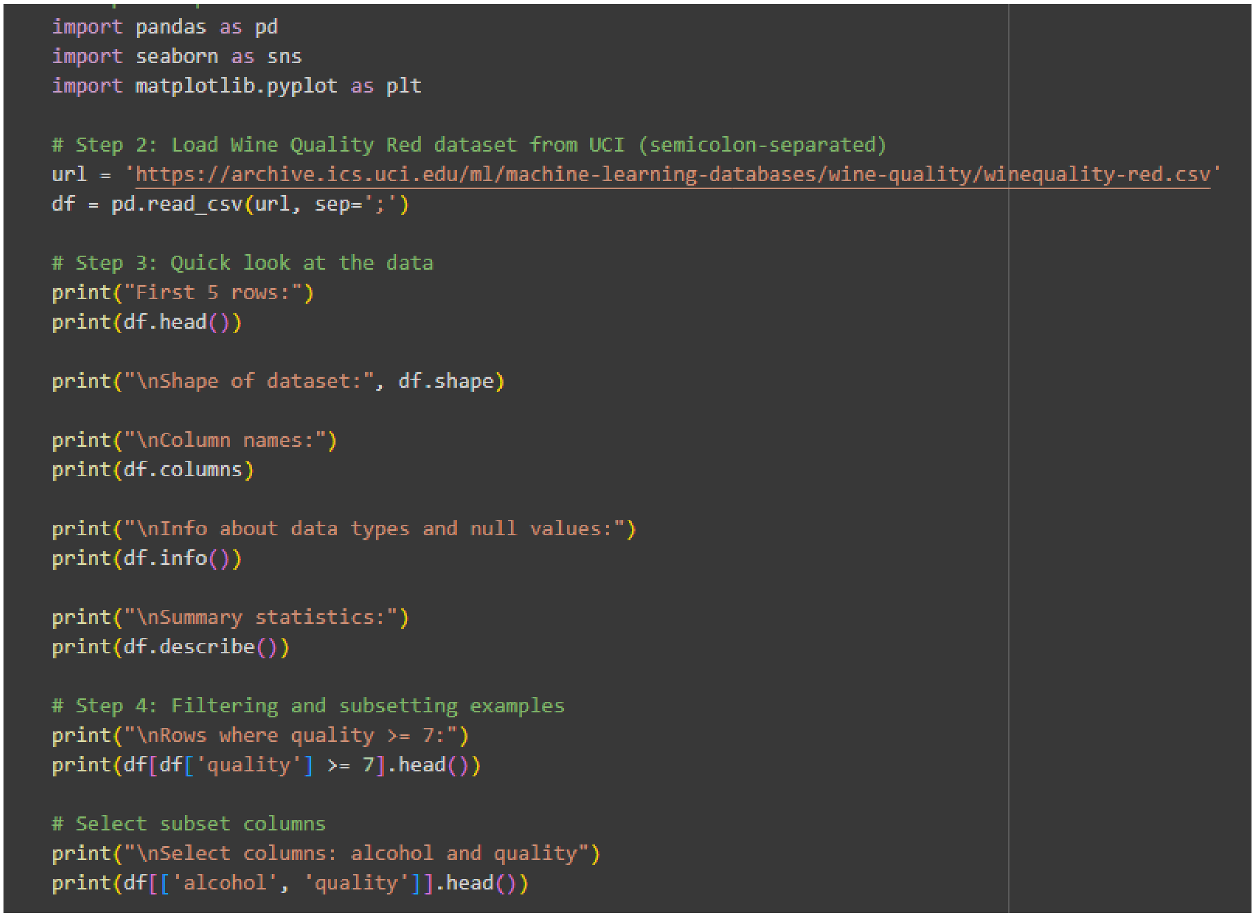
```
Missing values per column:
 Age                 0
Income              0
Credit_Score        0
Loan_Amount         0
Employment_Years    0
Loan_Status         0
dtype: int64
Cleaned Data Sample:
   Age  Income  Credit_Score  Loan_Amount  Employment_Years  Loan_Status
0   60      11           710           39                20            0
1   27       5           574           38                 1            0
2   21      13           367            5                 7            0
3   37       3           516           43                 2            0
4   35       2           622           25                 1            0

Standardized Data Sample:
        Age    Income  Credit_Score  Loan_Amount  Employment_Years  \
0  1.708662  0.549104      0.781864     0.910189          1.701140   
1 -0.948769 -0.889026     -0.086229     0.836463         -1.379781   
2 -1.431938  1.028481     -1.407518    -1.596518         -0.406858   
3 -0.143487 -1.368403     -0.456445     1.205096         -1.217627   
4 -0.304543 -1.608091      0.220157    -0.121984         -1.379781   
```



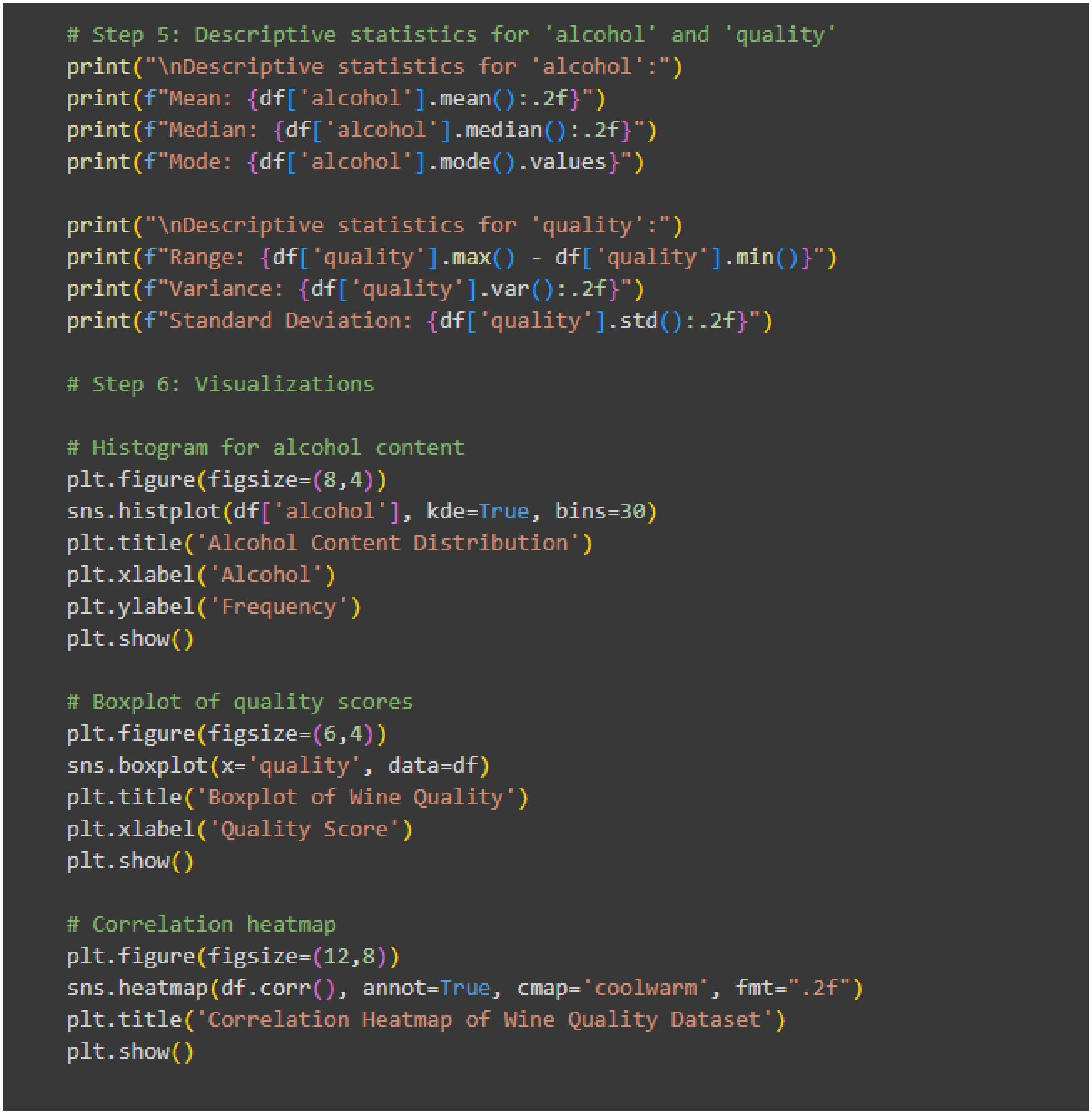
```
   Loan_Status
0    -0.442326
1    -0.442326
2    -0.442326
3    -0.442326
4    -0.442326

Min-Max Scaled Data Sample:
     Age    Income  Credit_Score  Loan_Amount  Employment_Years  Loan_Status
0  1.000  0.692308      0.745421     0.755556              1.00          0.0
1  0.175  0.230769      0.496337     0.733333              0.05          0.0
2  0.025  0.846154      0.117216     0.000000              0.35          0.0
3  0.425  0.076923      0.390110     0.844444              0.10          0.0
4  0.375  0.000000      0.584249     0.444444              0.05          0.0
/tmp/ipython-input-6-2547170765.py:16: 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 operation inplace on the original object.


  df[col].fillna(df[col].mean(), inplace=True)
```

Exp: 4

Code:

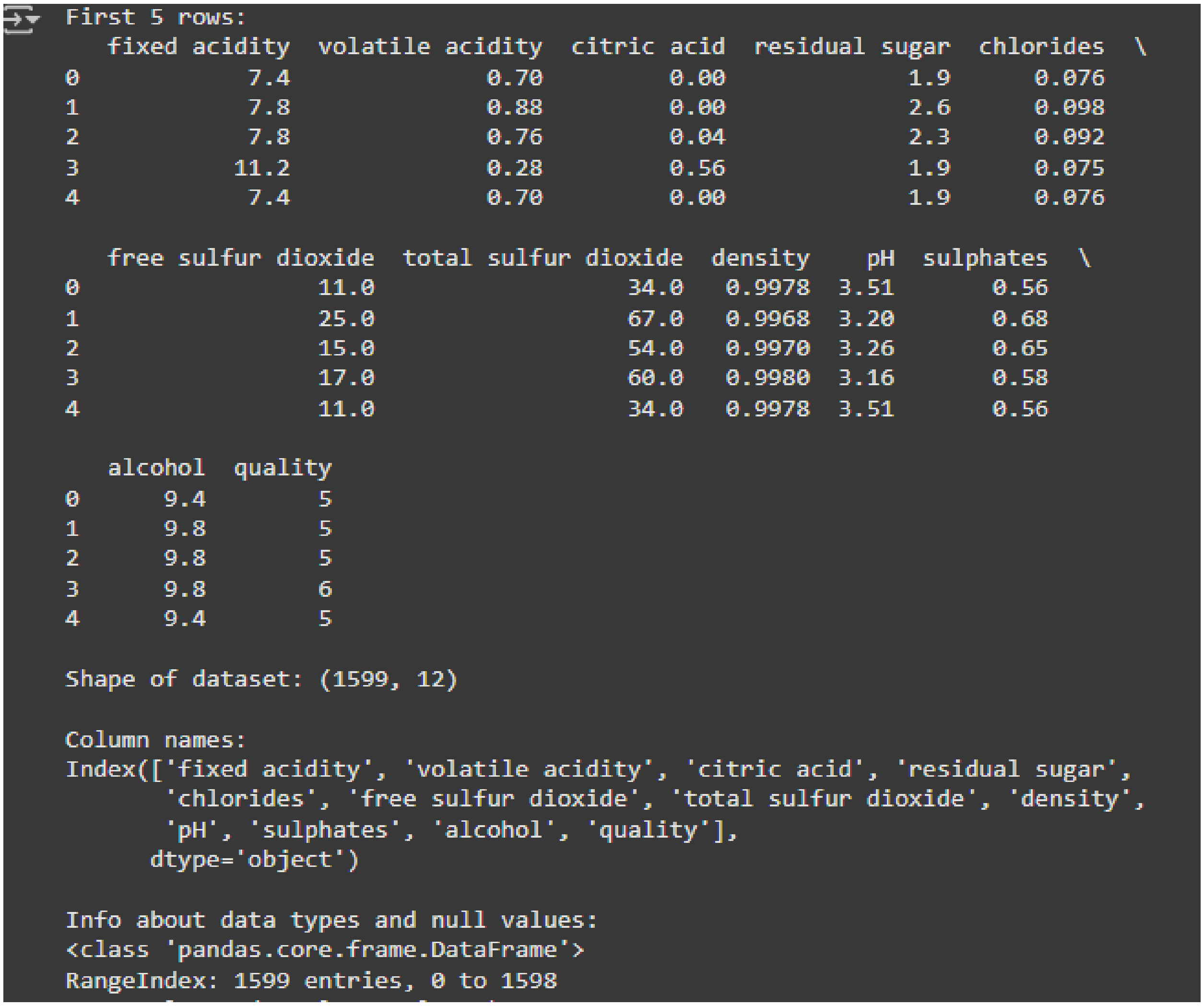


```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Step 2: Load Wine Quality Red dataset from UCI (semicolon-separated)
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'
df = pd.read_csv(url, sep=';')

# Step 3: Quick look at the data
print("First 5 rows:")
print(df.head())

print("\nShape of dataset:", df.shape)

print("\nColumn names:")
print(df.columns)

print("\nInfo about data types and null values:")
print(df.info())

print("\nSummary statistics:")
print(df.describe())

# Step 4: Filtering and subsetting examples
print("\nRows where quality >= 7:")
print(df[df['quality'] >= 7].head())

# Select subset columns
print("\nSelect columns: alcohol and quality")
print(df[['alcohol', 'quality']].head())
```

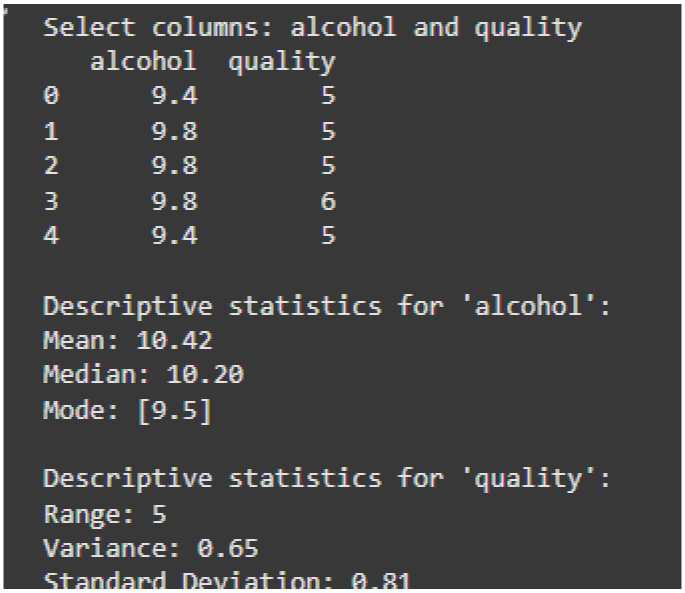


```
# Step 5: Descriptive statistics for 'alcohol' and 'quality'
print("\nDescriptive statistics for 'alcohol':")
print(f"Mean: {df['alcohol'].mean():.2f}")
print(f"Median: {df['alcohol'].median():.2f}")
print(f"Mode: {df['alcohol'].mode().values}")

print("\nDescriptive statistics for 'quality':")
print(f"Range: {df['quality'].max() - df['quality'].min()}")
print(f"Variance: {df['quality'].var():.2f}")
print(f"Standard Deviation: {df['quality'].std():.2f}")

# Step 6: Visualizations

# Histogram for alcohol content
plt.figure(figsize=(8,4))
sns.histplot(df['alcohol'], kde=True, bins=30)
plt.title('Alcohol Content Distribution')
plt.xlabel('Alcohol')
plt.ylabel('Frequency')
plt.show()

# Boxplot of quality scores
plt.figure(figsize=(6,4))
sns.boxplot(x='quality', data=df)
plt.title('Boxplot of Wine Quality')
plt.xlabel('Quality Score')
plt.show()

# Correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Wine Quality Dataset')
plt.show()
```

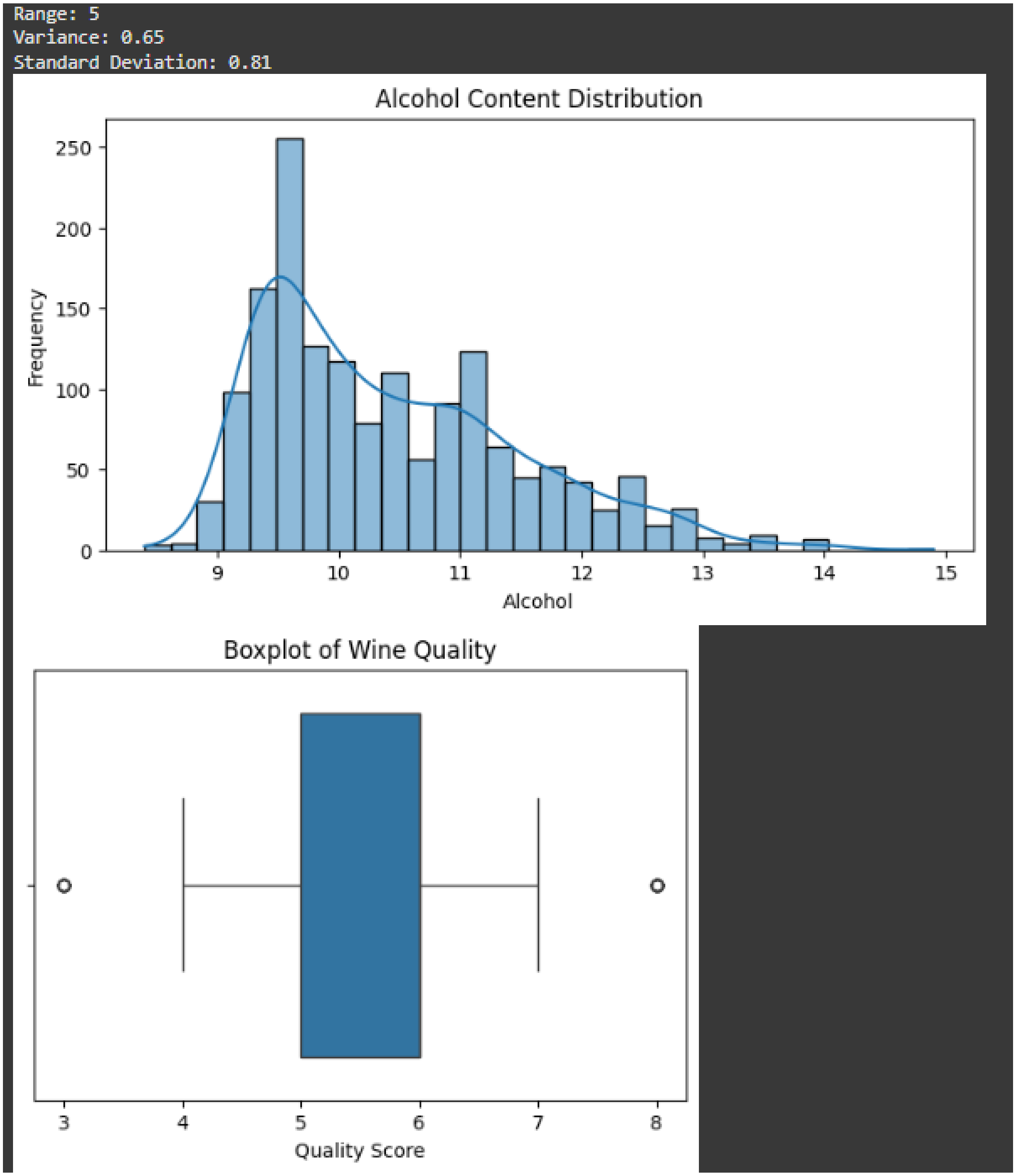
Output:



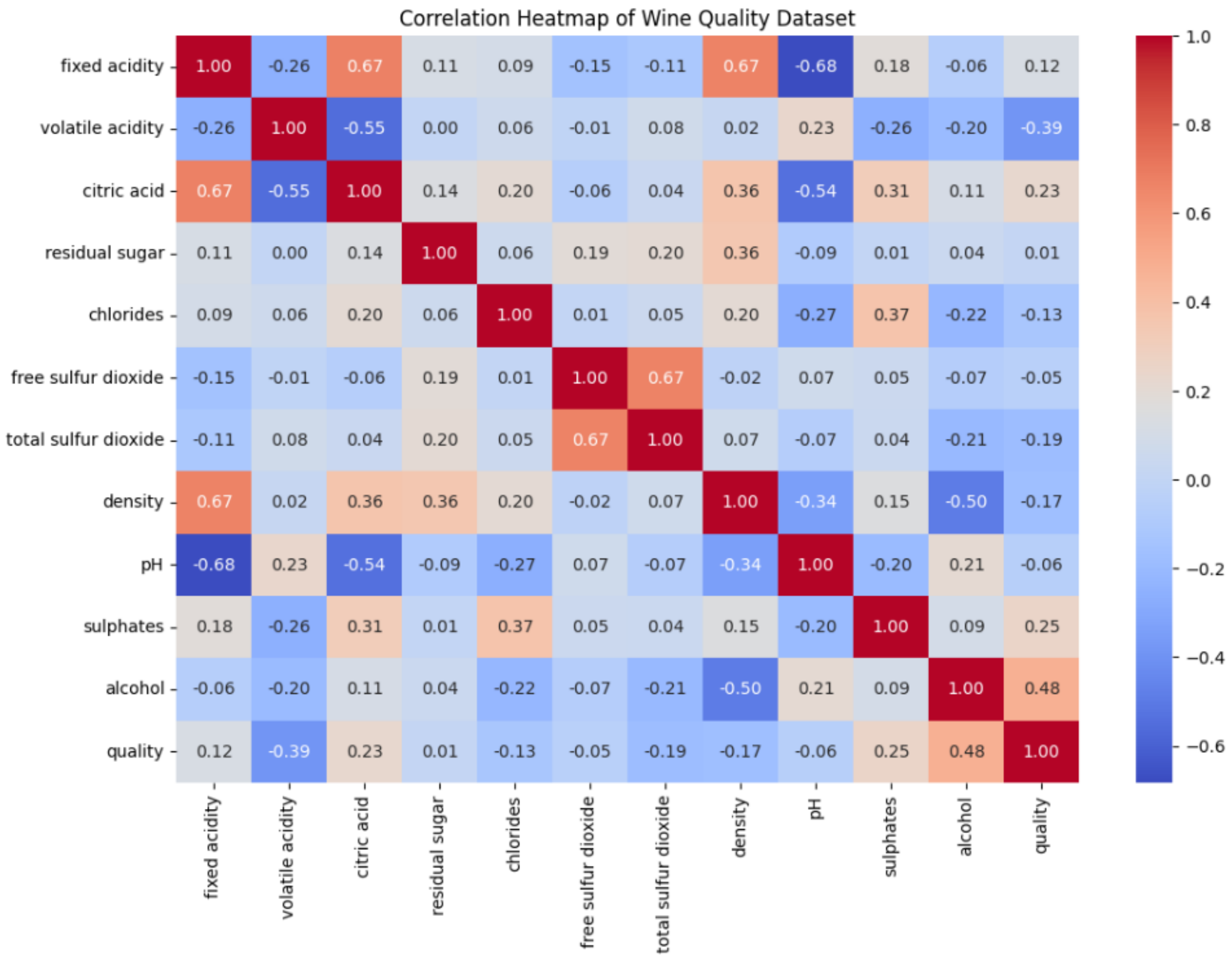
```
First 5 rows:
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076   
1            7.8              0.88         0.00             2.6      0.098   
2            7.8              0.76         0.04             2.3      0.092   
3           11.2              0.28         0.56             1.9      0.075   
4            7.4              0.70         0.00             1.9      0.076   

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56   
1                 25.0                  67.0   0.9968  3.20       0.68   
2                 15.0                  54.0   0.9970  3.26       0.65   
3                 17.0                  60.0   0.9980  3.16       0.58   
4                 11.0                  34.0   0.9978  3.51       0.56   

   alcohol  quality  
0      9.4        5  
1      9.8        5  
2      9.8        5  
3      9.8        6  
4      9.4        5  

Shape of dataset: (1599, 12)

Column names:
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')

Info about data types and null values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
```



```
Select columns: alcohol and quality
   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5

Descriptive statistics for 'alcohol':
Mean: 10.42
Median: 10.20
Mode: [9.5]

Descriptive statistics for 'quality':
Range: 5
Variance: 0.65
Standard Deviation: 0.81
```



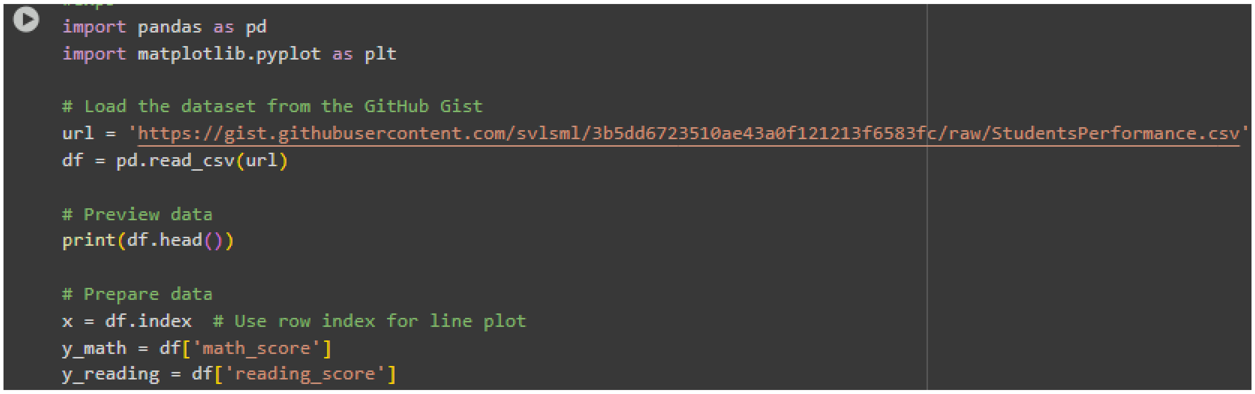
```
Range: 5
Variance: 0.65
Standard Deviation: 0.81
```



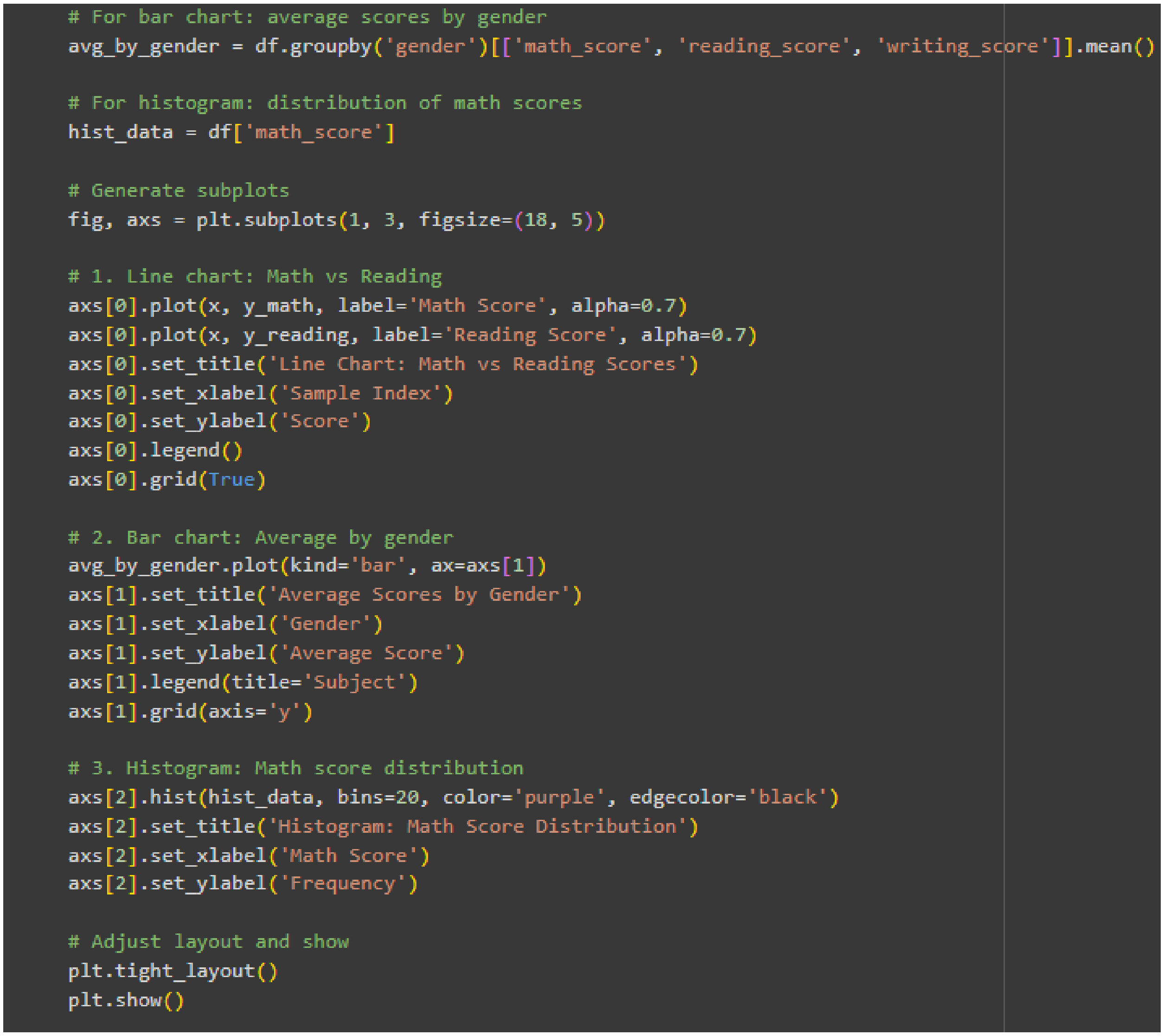
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.00	-0.26	0.67	0.11	0.09	-0.15	-0.11	0.67	-0.68	0.18	-0.06	0.12
volatile acidity	-0.26	1.00	-0.55	0.00	0.06	-0.01	0.08	0.02	0.23	-0.26	-0.20	-0.39
citric acid	0.67	-0.55	1.00	0.14	0.20	-0.06	0.04	0.36	-0.54	0.31	0.11	0.23
residual sugar	0.11	0.00	0.14	1.00	0.06	0.19	0.20	0.36	-0.09	0.01	0.04	0.01
chlorides	0.09	0.06	0.20	0.06	1.00	0.01	0.05	0.20	-0.27	0.37	-0.22	-0.13
free sulfur dioxide	-0.15	-0.01	-0.06	0.19	0.01	1.00	0.67	-0.02	0.07	0.05	-0.07	-0.05
total sulfur dioxide	-0.11	0.08	0.04	0.20	0.05	0.67	1.00	0.07	-0.07	0.04	-0.21	-0.19
density	0.67	0.02	0.36	0.36	0.20	-0.02	0.07	1.00	-0.34	0.15	-0.50	-0.17
pH	-0.68	0.23	-0.54	-0.09	-0.27	0.07	-0.07	-0.34	1.00	-0.20	0.21	-0.06
sulphates	0.18	-0.26	0.31	0.01	0.37	0.05	0.04	0.15	-0.20	1.00	0.09	0.25
alcohol	-0.06	-0.20	0.11	0.04	-0.22	-0.07	-0.21	-0.50	0.21	0.09	1.00	0.48
quality	0.12	-0.39	0.23	0.01	-0.13	-0.05	-0.19	-0.17	-0.06	0.25	0.48	1.00

Exp: 5

Code:

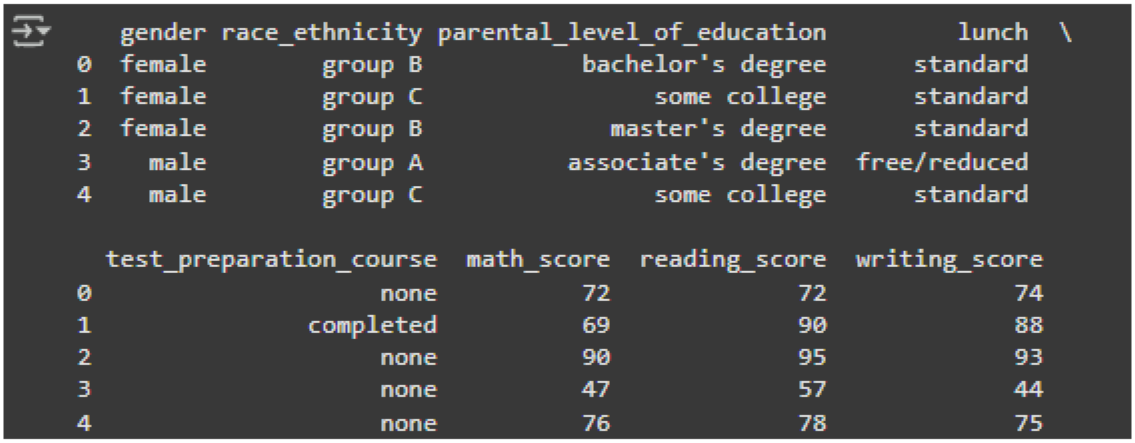


```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset from the GitHub Gist
url = 'https://gist.githubusercontent.com/svlsml/3b5dd6723510ae43a0f121213f6583fc/raw/StudentsPerformance.csv'
df = pd.read_csv(url)

# Preview data
print(df.head())

# Prepare data
x = df.index  # Use row index for line plot
y_math = df['math_score']
y_reading = df['reading_score']
```

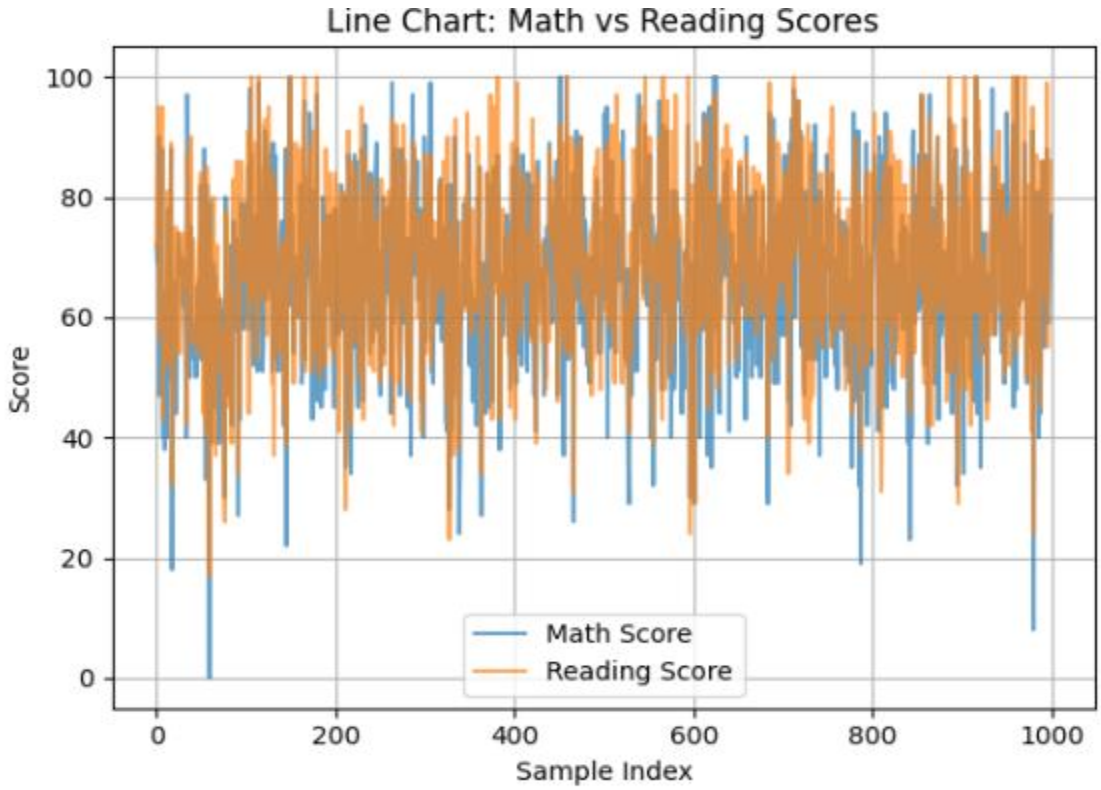


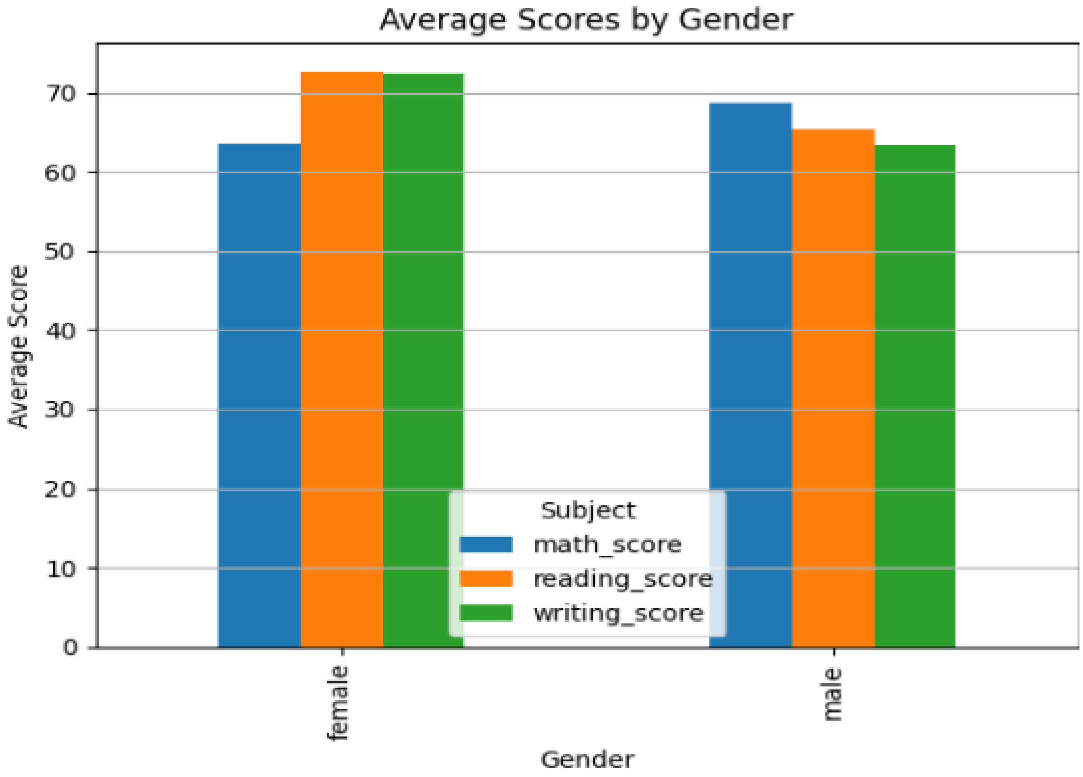
```
    # For bar chart: average scores by gender
    avg_by_gender = df.groupby('gender')[['math_score', 'reading_score', 'writing_score']].mean()

    # For histogram: distribution of math scores
    hist_data = df['math_score']

    # Generate subplots
    fig, axs = plt.subplots(1, 3, figsize=(18, 5))

    # 1. Line chart: Math vs Reading
    axs[0].plot(x, y_math, label='Math Score', alpha=0.7)
    axs[0].plot(x, y_reading, label='Reading Score', alpha=0.7)
    axs[0].set_title('Line Chart: Math vs Reading Scores')
    axs[0].set_xlabel('Sample Index')
    axs[0].set_ylabel('Score')
    axs[0].legend()
    axs[0].grid(True)

    # 2. Bar chart: Average by gender
    avg_by_gender.plot(kind='bar', ax=axs[1])
    axs[1].set_title('Average Scores by Gender')
    axs[1].set_xlabel('Gender')
    axs[1].set_ylabel('Average Score')
    axs[1].legend(title='Subject')
    axs[1].grid(axis='y')

    # 3. Histogram: Math score distribution
    axs[2].hist(hist_data, bins=20, color='purple', edgecolor='black')
    axs[2].set_title('Histogram: Math Score Distribution')
    axs[2].set_xlabel('Math Score')
    axs[2].set_ylabel('Frequency')

    # Adjust layout and show
    plt.tight_layout()
    plt.show()
```

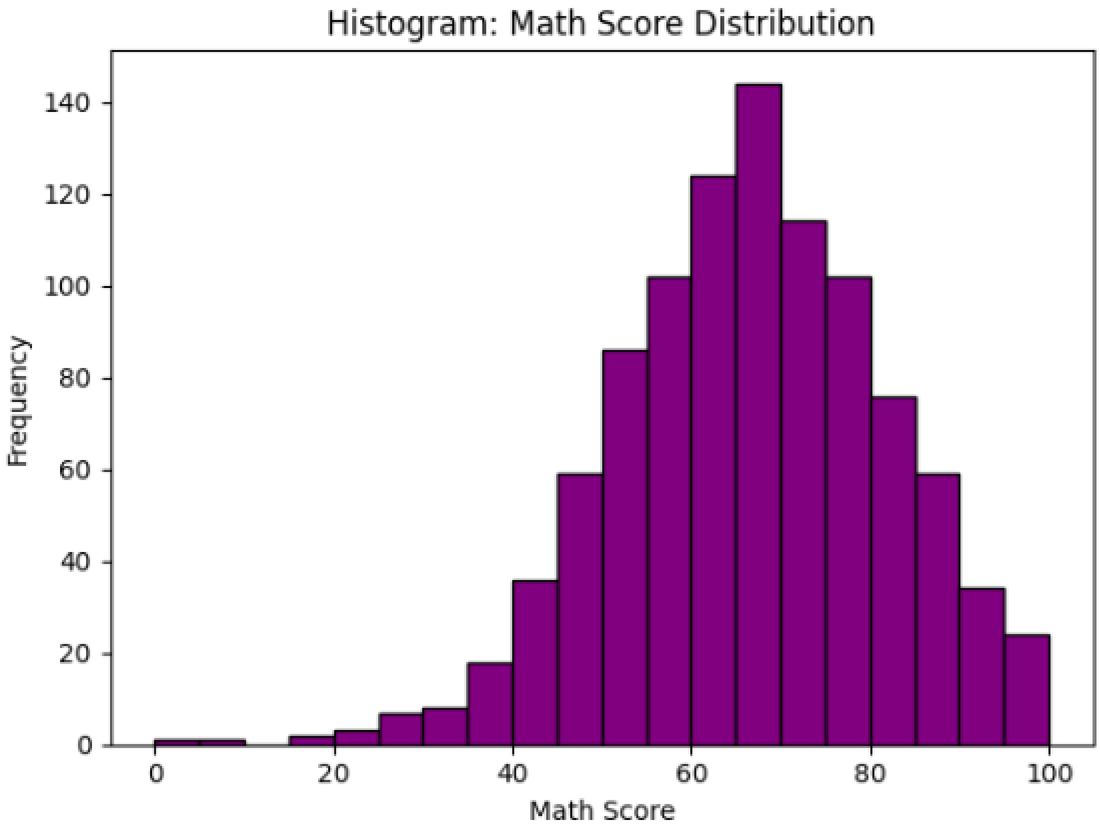
Output:



```
   gender race_ethnicity parental_level_of_education         lunch  \
0  female        group B           bachelor's degree      standard   
1  female        group C                some college      standard   
2  female        group B             master's degree      standard   
3    male        group A          associate's degree  free/reduced   
4    male        group C                some college      standard   

  test_preparation_course  math_score  reading_score  writing_score  
0                    none          72             72             74  
1               completed          69             90             88  
2                    none          90             95             93  
3                    none          47             57             44  
4                    none          76             78             75  
```


Line Chart: Math vs Reading Scores
Score
0
20
40
60
80
100
Sample Index
0
200
400
600
800
1000
Math Score
Reading Score


Average Scores by Gender
Average Score
0
10
20
30
40
50
60
70
Subject
math_score
reading_score
writing_score
female
male
Gender


Histogram: Math Score Distribution
Frequency
0
20
40
60
80
100
120
140
0
20
40
60
80
100
Math Score