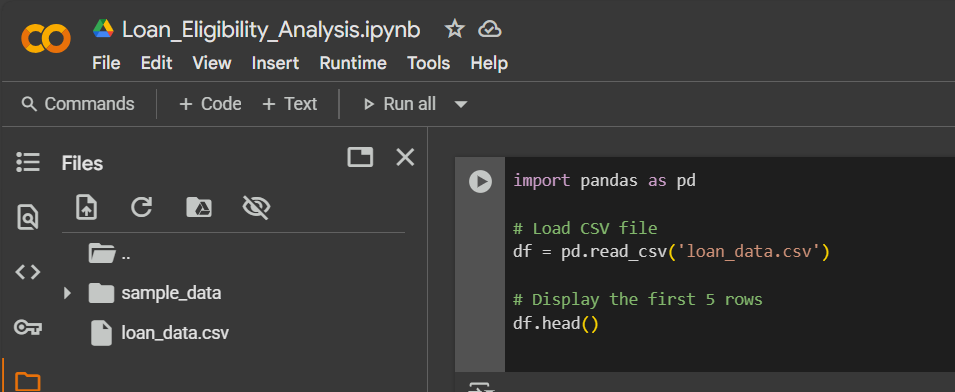
**Coding Assessment – Data Cleaning and Pandas**

1. **Loading Data in Pandas DataFrame**

import pandas as pd

# Load the dataset

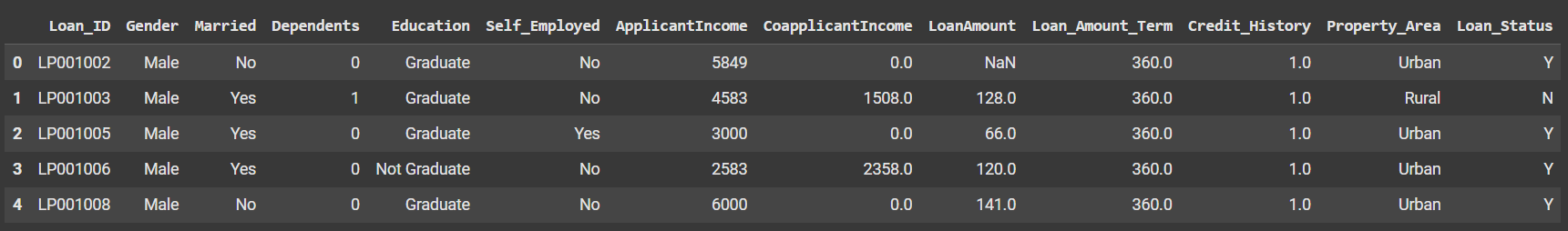
df = pd.read\_csv('loan\_data.csv')



1. **Printing rows of the Data**

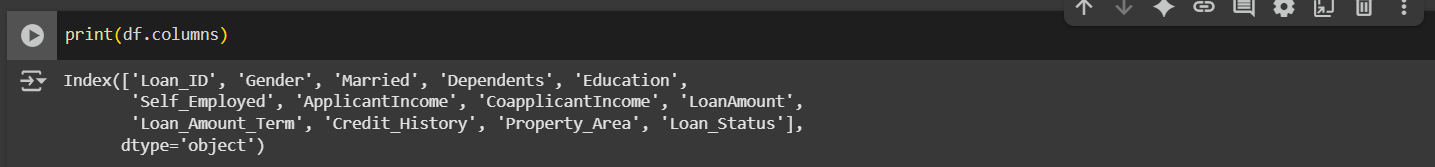
# Show first 5 rows

df.head()



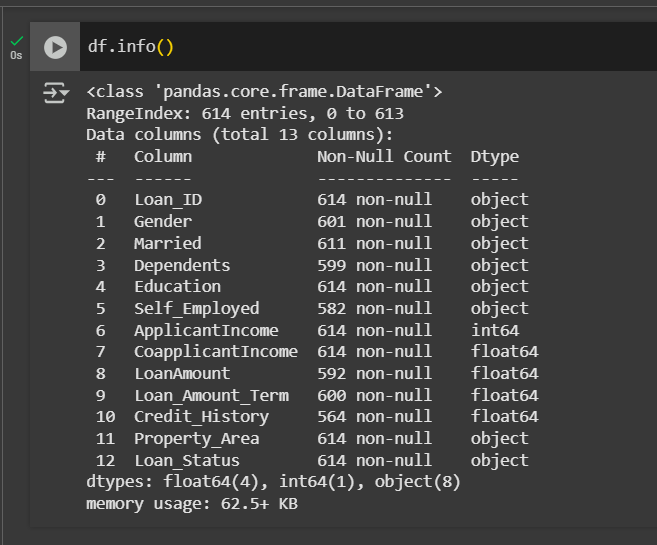
1. **Printing the column names of the DataFrame**

* print(df.columns)



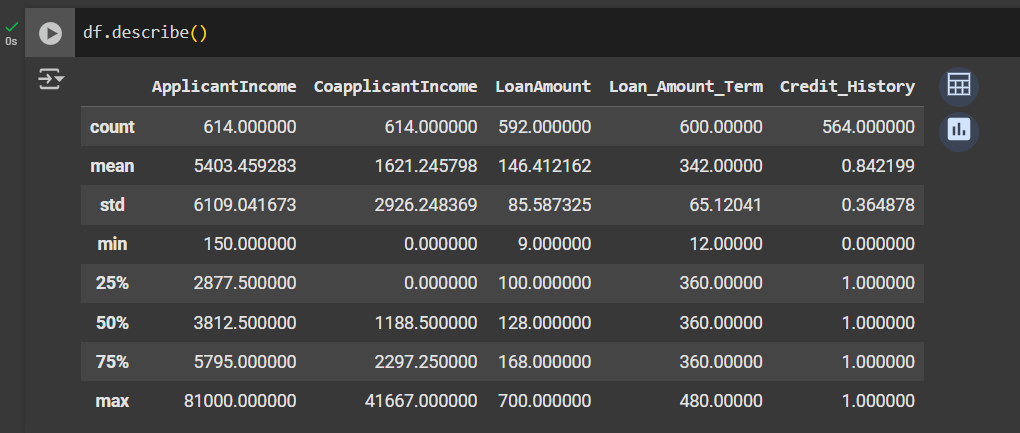
1. **Summary of Data Frame**

* df.info()



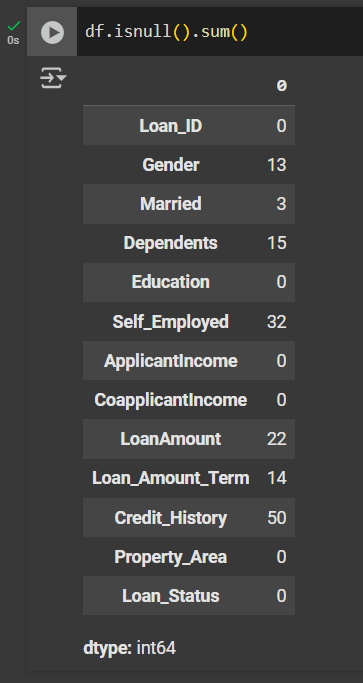
1. **Descriptive Statistical Measures of a DataFrame**

* df.describe()



1. **Missing Data Handing**

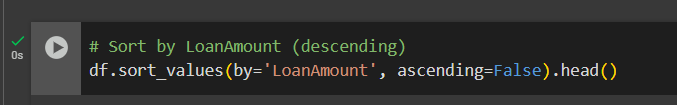
* df.isnull( ).sum( )

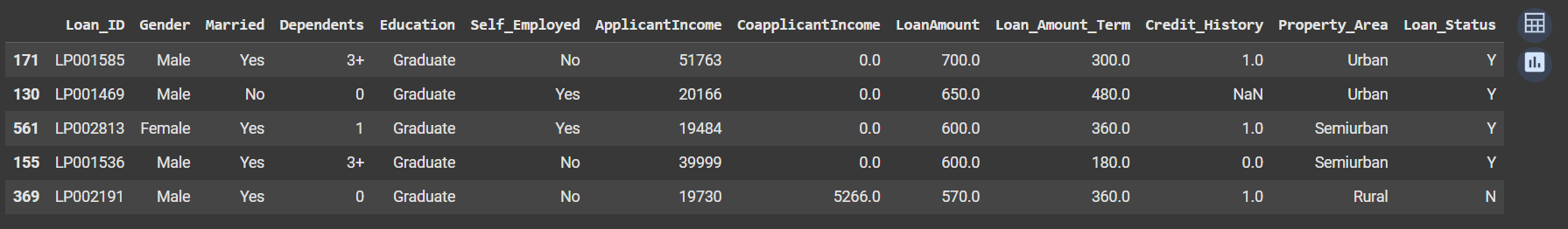


1. **Sorting DataFrame values**

# Sort by LoanAmount (descending)

df.sort\_values(by='LoanAmount', ascending=False).head()





1. **Merge Data Frames**

df1 = pd.DataFrame({

'Loan\_ID': ['LP001002', 'LP001003'],

'Loan\_Status': ['Y', 'N']})

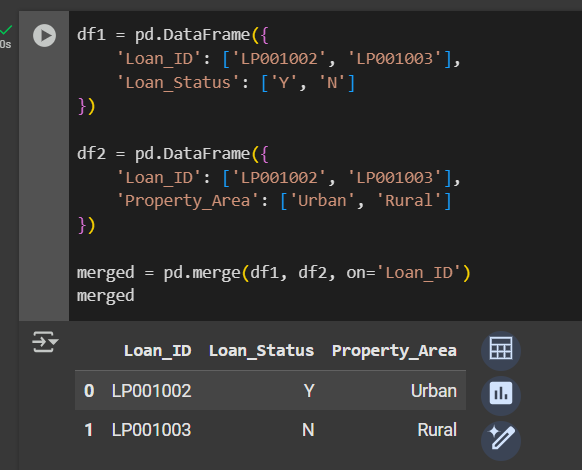
df2 = pd.DataFrame({

'Loan\_ID': ['LP001002', 'LP001003'],

'Property\_Area': ['Urban', 'Rural']})

merged = pd.merge(df1, df2, on='Loan\_ID')

merged

****

1. **Apply Function**

# Create a new column 'TotalIncome'

df['TotalIncome']=df['ApplicantIncome']+ df['CoapplicantIncome']

def income\_category(income):

if income < 2500:

return 'Low'

elif income < 6000:

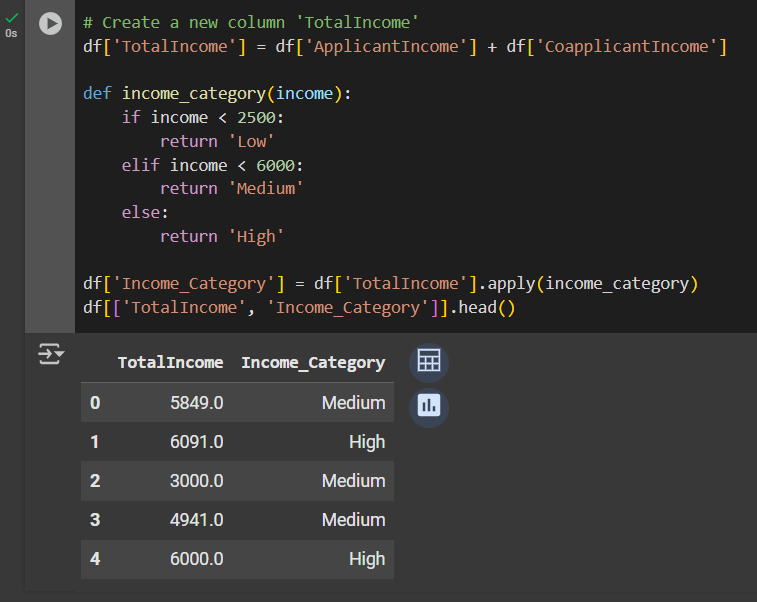
return 'Medium'

else:

return 'High'

df['Income\_Category'] = df['TotalIncome'].apply(income\_category)

df[['TotalIncome', 'Income\_Category']].head()

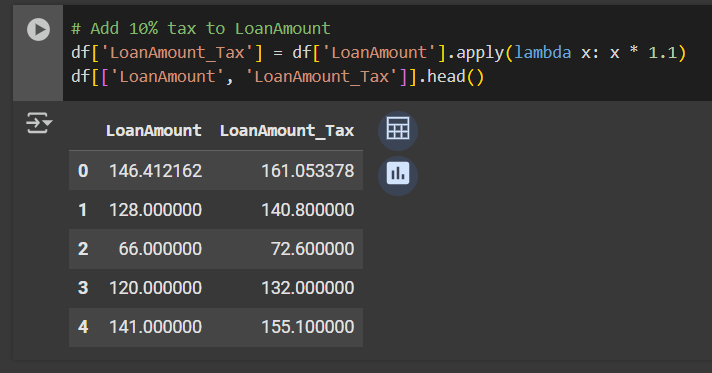
****

1. **By using the lambda operator**

# Add 10% tax to LoanAmount (demo)

df['LoanAmount\_Tax'] = df['LoanAmount'].apply(lambda x: x \* 1.1)

df[['LoanAmount', 'LoanAmount\_Tax']].head()

****

1. **Visualizing DataFrame**

import matplotlib.pyplot as plt

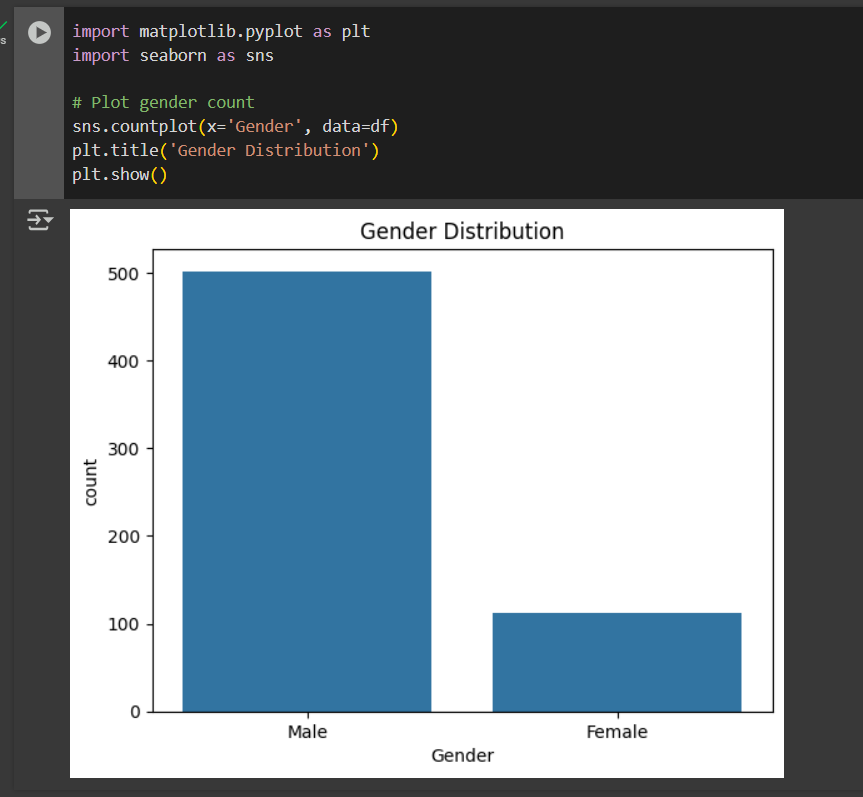
import seaborn as sns

# Plot gender count

sns.countplot(x='Gender', data=df)

plt.title('Gender Distribution')

plt.show()

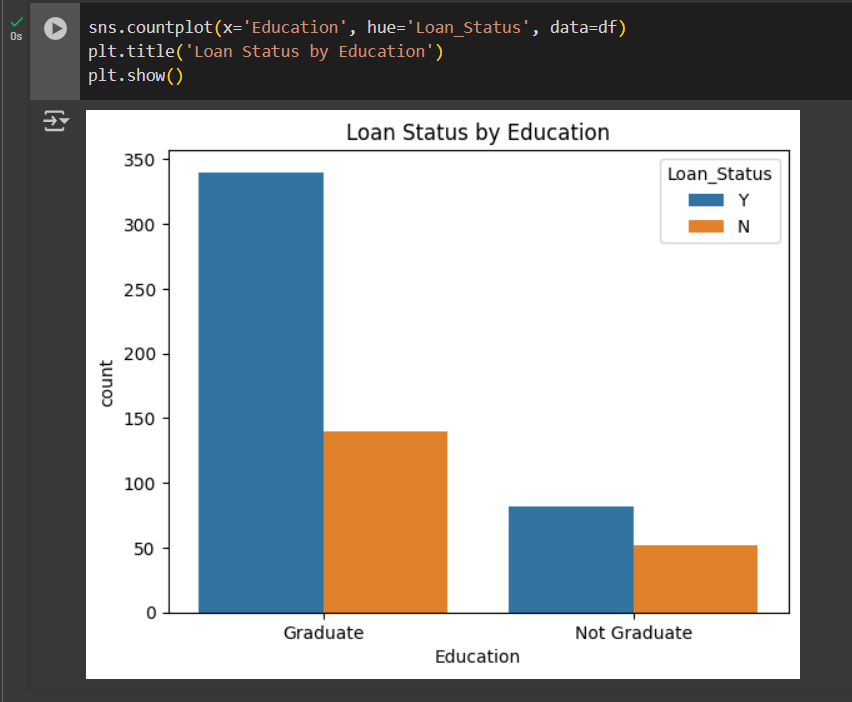
****

**Loan status by education**

sns.countplot(x='Education', hue='Loan\_Status', data=df)

plt.title('Loan Status by Education')

plt.show()

****

**Pandas Joins**

import pandas as pd

# Loan application data

loan\_df = pd.DataFrame({

'Loan\_ID': ['LP001002', 'LP001003', 'LP001004', 'LP001005'],

'Gender': ['Male', 'Male', 'Male', 'Male'],

'Married': ['No', 'Yes', 'Yes', 'Yes'],

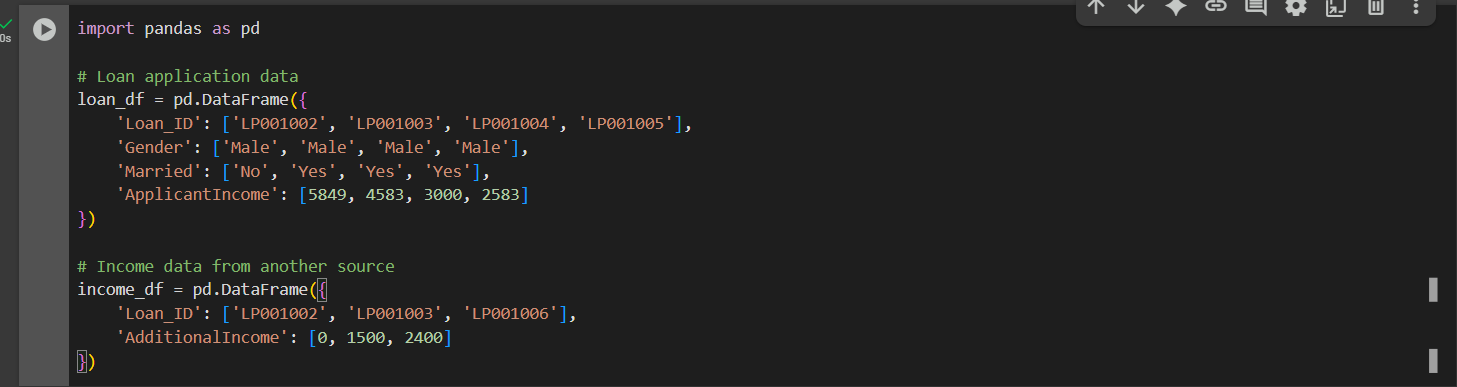
'ApplicantIncome': [5849, 4583, 3000, 2583]})

# Income data from another source

income\_df = pd.DataFrame({

'Loan\_ID': ['LP001002', 'LP001003', 'LP001006'],

'AdditionalIncome': [0, 1500, 2400]})



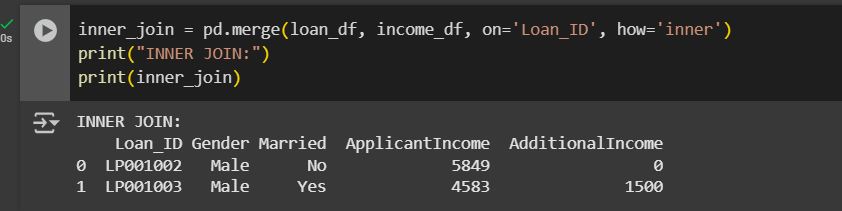
1. **Inner Join**

Returns only rows with matching Loan\_ID in both DataFrames.

inner\_join = pd.merge(loan\_df, income\_df, on='Loan\_ID', how='inner')

print(" INNER JOIN:")

print(inner\_join)



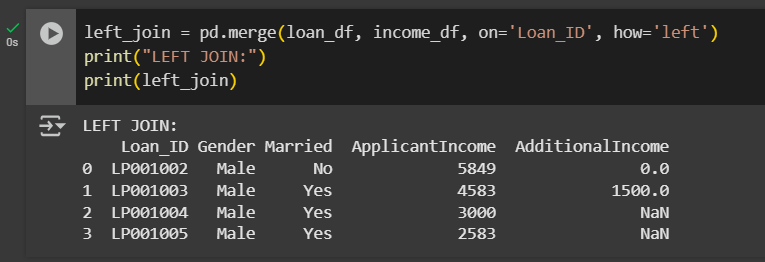
1. **Left Join**

Returns all records from loan\_df and matching rows from income\_df.

left\_join = pd.merge(loan\_df, income\_df, on='Loan\_ID', how='left')

print(" LEFT JOIN:")

print(left\_join)



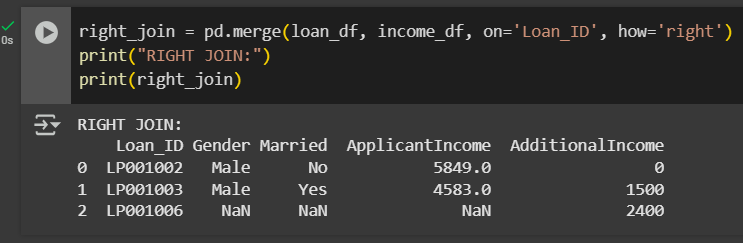
1. **Right Join**

Returns all records from income\_df and matching ones from loan\_df.

right\_join = pd.merge(loan\_df, income\_df, on='Loan\_ID', how='right')

print(" RIGHT JOIN:")

print(right\_join)



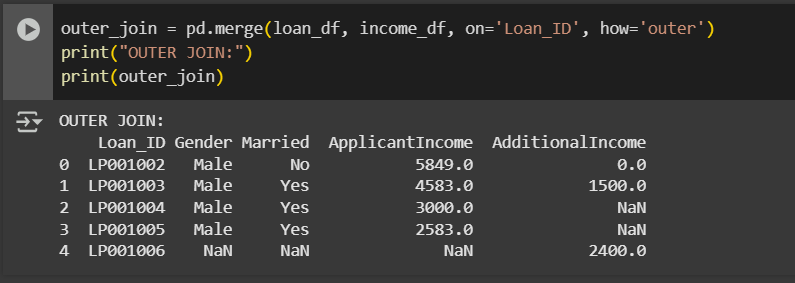
1. **Outer Join**

Returns all rows from both DataFrames. Non-matching fields are filled with NaN.

outer\_join = pd.merge(loan\_df, income\_df, on='Loan\_ID', how='outer')

print(" OUTER JOIN:")

print(outer\_join)



**Submitted by**

Siva Balan T