Prompt:

# prompt: A excel sheet contains columns labeled with Date, Gauge, water level, Discharge, Observed\_Computed. Do a summary statics analysis and print the results. Plot time series trend, and plots associated with. The environment is Google colab. Upload the excel sheet from computer and download plots to a folder.

Code:

import pandas as pd

import matplotlib.pyplot as plt

import os

# Create a directory to save plots if it doesn't exist

plots\_folder = "analysis\_plots"

if not os.path.exists(plots\_folder):

    os.makedirs(plots\_folder)

# Upload the Excel file (replace 'your\_file.xlsx' with the actual filename)

from google.colab import files

uploaded = files.upload()

excel\_file = list(uploaded.keys())[0]  # Get the filename

# Read the Excel file into a Pandas DataFrame

df = pd.read\_excel(excel\_file)

# Summary Statistics

summary\_stats = df.describe()

print("Summary Statistics:")

print(summary\_stats)

# Time Series Analysis and Plots

# Convert 'Date' column to datetime objects if it's not already

df['Date'] = pd.to\_datetime(df['Date'])  # Important for time series plotting

df = df.set\_index('Date')  # Set 'Date' as the index

# 1. Time Series Trend for Water Level

plt.figure(figsize=(12, 6))

plt.plot(df.index, df['water level'], label='Water Level')

plt.title('Time Series Trend of Water Level')

plt.xlabel('Date')

plt.ylabel('Water Level')

plt.legend()

plt.grid(True)

plt.savefig(os.path.join(plots\_folder, 'water\_level\_time\_series.png')) #Save the plot

plt.show()

# 2. Time Series Trend for Discharge

plt.figure(figsize=(12, 6))

plt.plot(df.index, df['Discharge'], label='Discharge', color='orange')  # Different color

plt.title('Time Series Trend of Discharge')

plt.xlabel('Date')

plt.ylabel('Discharge')

plt.legend()

plt.grid(True)

plt.savefig(os.path.join(plots\_folder, 'discharge\_time\_series.png')) #Save the plot

plt.show()

# 3. Scatter Plot: Water Level vs. Discharge

plt.figure(figsize=(8, 6))

plt.scatter(df['water level'], df['Discharge'], alpha=0.7)  # Alpha for transparency

plt.title('Scatter Plot of Water Level vs. Discharge')

plt.xlabel('Water Level')

plt.ylabel('Discharge')

plt.grid(True)

plt.savefig(os.path.join(plots\_folder, 'water\_level\_vs\_discharge.png')) #Save the plot

plt.show()

# 4. Box Plot for Water Level and Discharge (to see distribution)

plt.figure(figsize=(8, 6))

df[['water level', 'Discharge']].boxplot()

plt.title('Box Plot of Water Level and Discharge')

plt.ylabel('Value')

plt.savefig(os.path.join(plots\_folder, 'box\_plot\_water\_discharge.png')) #Save the plot

plt.show()

#5. Histogram for Water Level and Discharge (to see distribution)

plt.figure(figsize=(8, 6))

plt.hist(df['water level'], alpha=0.5, label='Water Level')

plt.hist(df['Discharge'], alpha=0.5, label='Discharge')

plt.title('Histogram of Water Level and Discharge')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.legend(loc='upper right')

plt.savefig(os.path.join(plots\_folder, 'histogram\_water\_discharge.png')) #Save the plot

plt.show()

# Download the plots folder as a zip archive

!zip -r analysis\_plots.zip $plots\_folder

files.download('analysis\_plots.zip')

print(f"Plots saved to '{plots\_folder}' folder and downloaded as 'analysis\_plots.zip'")

# prompt: in continuation with the above, check for duplicates and give statistics

import pandas as pd

import matplotlib.pyplot as plt

import os

# ... (Previous code for folder creation and file upload remains the same)

# Read the Excel file

df = pd.read\_excel(excel\_file)

# --- Duplicate Check and Statistics ---

duplicates = df.duplicated()

num\_duplicates = duplicates.sum()

print("\nDuplicate Analysis:")

print(f"Number of duplicate rows: {num\_duplicates}")

if num\_duplicates > 0:

    print("Indices of duplicate rows:")

    print(df.loc[duplicates].index)  # Print the indices of duplicate rows

    # Statistics on the duplicate rows (if any)

    print("\nStatistics of Duplicate Rows:")

    print(df.loc[duplicates].describe())

    #Option to remove duplicates

    #df = df.drop\_duplicates()

    #print("\nDataFrame after removing duplicates:")

    #print(df)

else:

    print("No duplicate rows found.")

# --- Summary Statistics (including duplicates or after removal) ---

summary\_stats = df.describe()

print("\nSummary Statistics:")

print(summary\_stats)

# --- Time Series Analysis and Plots (using the DataFrame after duplicate check) ---

# Convert 'Date' column to datetime objects if it's not already

df['Date'] = pd.to\_datetime(df['Date'])  # Important for time series plotting

df = df.set\_index('Date')  # Set 'Date' as the index

# ... (The rest of the plotting code from the previous response remains the same)

# ... (Zip and download code remains the same)

# prompt: Read a excel file, do stage discharge function and bring out evaluation metrics

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import scipy.optimize as opt

from google.colab import files

# Step 1: Upload the Excel file

uploaded = files.upload()

# Step 2: Read the Excel file

file\_name = list(uploaded.keys())[0]

df = pd.read\_excel(file\_name)

# Step 3: Ensure required columns exist

required\_columns = ['Date', 'Water\_Level', 'Discharge']

if not all(col in df.columns for col in required\_columns):

    raise ValueError("❌ The dataset must contain 'Date', 'Water\_Level', and 'Discharge' columns.")

# Step 4: Drop missing values

df = df.dropna(subset=['Water\_Level', 'Discharge'])

# Step 5: Define the stage-discharge function

def rating\_curve(H, C, H0, n):

    return C \* (H - H0) \*\* n

# Step 6: Estimate initial parameters (H0 as minimum stage)

H0\_initial = df['Water\_Level'].min() - 0.1  # Slightly below min stage

p0 = [1, H0\_initial, 2]  # Initial guesses for C, H0, n

# Step 7: Fit the rating curve using nonlinear regression

params, \_ = opt.curve\_fit(rating\_curve, df['Water\_Level'], df['Discharge'], p0=p0, maxfev=10000)

# Step 8: Extract the best-fit parameters

C\_opt, H0\_opt, n\_opt = params

print(f"\n📌 Rating Curve Equation: Q = {C\_opt:.3f} \* (H - {H0\_opt:.3f})^{n\_opt:.3f}")

# Step 9: Plot the Stage-Discharge Relationship

plt.figure(figsize=(10, 6))

plt.scatter(df['Water\_Level'], df['Discharge'], label="Observed Data", color="blue", alpha=0.6)

H\_fit = np.linspace(df['Water\_Level'].min(), df['Water\_Level'].max(), 100)

Q\_fit = rating\_curve(H\_fit, C\_opt, H0\_opt, n\_opt)

plt.plot(H\_fit, Q\_fit, label=f"Fitted Curve: Q = {C\_opt:.3f} \* (H - {H0\_opt:.3f})^{n\_opt:.3f}", color="red")

plt.xlabel("Water Level (m)")

plt.ylabel("Discharge (m³/s)")

plt.title("📈 Stage-Discharge Rating Curve")

plt.legend()

plt.grid()

plt.show()

#step 3: Compute Fitted Discharge Values

df['Fitted\_Discharge'] = df['Gauge'].apply(calculate\_discharge)

# Step 4: Compute Evaluation Metrics

# R² Score

r2 = r2\_score(df['Observed\_Discharge'], df['Fitted\_Discharge'])

# RMSE (Root Mean Squared Error)

rmse = np.sqrt(np.mean((df['Observed\_Discharge'] - df['Fitted\_Discharge']) \*\* 2))

# RSS (Residual Sum of Squares)

rss = np.sum((df['Observed\_Discharge'] - df['Fitted\_Discharge']) \*\* 2)

# Adjusted R² Calculation

n = len(df)  # Number of observations

p = 1  # Number of independent variables (Gauge)

adjusted\_r2 = 1 - ((1 - r2) \* (n - 1) / (n - p - 1))

# Standard Error of Estimate (SEE)

see = np.sqrt(rss / (n - 2))

# Step 5: Print the Metrics

print(f"R² Score: {r2:.4f}")

print(f"RMSE: {rmse:.4f}")

print(f"RSS: {rss:.4f}")

print(f"Adjusted R²: {adjusted\_r2:.4f}")

print(f"SEE: {see:.4f}")

# prompt: A sheet contains 2 columns, gauge and discharge, using the above equation calculate discharge for the same gauge. then put that values into third column named as predicted discharge, finally download that sheet. all calculate all the metrics needed for evaluation, plots etc., do a summary statics also

import pandas as pd

from google.colab import files

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.metrics import r2\_score

# Step 1: Upload the Excel file

uploaded = files.upload()

# Step 2: Read the Excel file

file\_name = list(uploaded.keys())[0]

df = pd.read\_excel(file\_name)

# Ensure 'Gauge' and 'Discharge' columns exist

if 'Gauge' not in df.columns or 'Discharge' not in df.columns:

    print("Error: 'Gauge' and 'Discharge' columns are required in the Excel file.")

    exit()

# Step 3: Define the Rating Curve Equation (replace with your actual equation)

def calculate\_discharge(H):

    return 133.966 \* (H - 2.400) \*\* 1.840

# Step 4: Calculate Predicted Discharge

df['Predicted\_Discharge'] = df['Gauge'].apply(calculate\_discharge)

# Step 5: Calculate evaluation metrics

# R-squared

r2 = r2\_score(df['Discharge'], df['Predicted\_Discharge'])

# RMSE (Root Mean Squared Error)

rmse = np.sqrt(np.mean((df['Discharge'] - df['Predicted\_Discharge'])\*\*2))

# MAE (Mean Absolute Error)

mae = np.mean(np.abs(df['Discharge'] - df['Predicted\_Discharge']))

# MAPE (Mean Absolute Percentage Error)

mape = np.mean(np.abs((df['Discharge'] - df['Predicted\_Discharge']) / df['Discharge'])) \* 100

# Step 6: Summary Statistics

summary\_stats = df[['Discharge', 'Predicted\_Discharge']].describe()

# Step 7: Create plots

plt.figure(figsize=(10, 6))

plt.scatter(df['Gauge'], df['Discharge'], label='Observed', color='blue')

plt.scatter(df['Gauge'], df['Predicted\_Discharge'], label='Predicted', color='red')

plt.xlabel('Gauge')

plt.ylabel('Discharge')

plt.title('Observed vs. Predicted Discharge')

plt.legend()

plt.grid(True)

plt.show()

#Step 8: Print Evaluation Metrics and Summary Statistics

print(f"R-squared: {r2}")

print(f"RMSE: {rmse}")

print(f"MAE: {mae}")

print(f"MAPE: {mape}")

print("\nSummary Statistics:")

print(summary\_stats)

# Step 9: Download the updated Excel file

df.to\_excel('discharge\_predictions.xlsx', index=False)

files.download('discharge\_predictions.xlsx')