```
#%%----------------Q2----------------
import pandas as pd
from pandas_datareader import data
import matplotlib.pyplot as plt
import yfinance as yf

from Home_Work_3_Q1 import datapreprocessing
apple_stock = yf.download('AAPL',start="2000-01-01",end="2022-09-25")
df = datapreprocessing(apple_stock)
df.Show_original()
df.Show_normalized()
df.Show_standardized()
df.Show_IQR()

#%%---------------Q3----------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
r_values = [0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6]
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'lime', 'navy', 'pink',
'purple', 'orange']
legend_handles = []
x = np.linspace(-1,1,400)
y = np.linspace(-1,1,400)
X,Y = np.meshgrid(x,y)
plt.figure(figsize=(8,8))
for r,color in zip(r_values, colors):
    Z=(np.abs(X)**r+np.abs(Y)**r)**(1/r)
    plt.contour(X,Y,Z,levels=[1],colors=[color])
    legend_handles.append(plt.Line2D([0], [0], linestyle='-', color=color,
label=f'$L_{{{r}}}$ norm'))
plt.title('Minkowski Distance for Lr Norms with Unity Distance (distances = 1)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(handles=legend_handles)
plt.grid(True)
plt.show()

#%%----------------Q6---------------
import numpy as np
from prettytable import PrettyTable
import matplotlib.pyplot as plt
import pandas as pd
np.random.seed(5808)
x = np.random.normal(1,np.sqrt(2),1000)
epsilon= np.random.normal(2,np.sqrt(3),1000)
y = x+ epsilon
X = np.vstack((x,y)).T
n = X.shape[0]
X_centered = X- X.mean(axis=0)
cov_matrix = (1/(n-1))*(np.dot(X_centered.T,X_centered))
cov_matrix_table = PrettyTable()
cov_matrix_table.field_names =["Covariance Matrix Element","Value"]
for i in range(cov_matrix.shape[0]):
    for j in range(cov_matrix.shape[1]):
        element_name = f'Cov(x{i+1},x{j+1})'
        value = "{:.2f}".format(cov_matrix[i,j])
        cov_matrix_table.add_row([element_name,value])
print("Estimated Covariance Matrix:")
```

```
print(cov_matrix_table)

eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)
eigen_table = PrettyTable()
eigen_table.field_names = ["Eigen value","Eigen vector"]
for i in range(len(eigen_values)):
    eigenvalue_formatted = "{:.2f}".format(eigen_values[i])
    eigenvector_formatted =
np.array2string(eigen_vectors[:,i],formatter={'float_kind' : lambda x:
"{:.2f}".format(x)})
    eigen_table.add_row([eigenvalue_formatted,eigenvector_formatted])
print("Eigen values & Eigen Vectors: ")
print(eigen_table)
plt.scatter(x, y, label='Data', alpha=0.5,color='yellow')
for i in range(len(eigen_values)):
    plt.quiver(
  X.mean(axis=0)[0],
        X.mean(axis=0)[1],
        eigen_vectors[0, i] *np.sqrt(eigen_values[i]),
        eigen_vectors[1, i] *np.sqrt(eigen_values[i]),
        angles='xy',
        scale_units='xy', scale=1,
        label=f'Eigenvector of {i+1}',color=f'C{i}')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter Plot with Eigenvectors')
plt.legend()
plt.grid(True)
plt.show()

eigen_values_scaled = eigen_values * (n - 1)
sqrt_eigenvalues_scaled = np.sqrt(eigen_values_scaled)
singular_values = np.linalg.svd(X_centered,compute_uv=False)
singular_values_table=PrettyTable()
singular_values_table.field_names = ["Singular Value"]
for i in range(len(singular_values)):
    singular_value_formatted = "{:.2f}".format(singular_values[i])
    singular_values_table.add_row([singular_value_formatted])
print("Singular Values Table:")
print(singular_values_table)
print("Scaled eigenvalues of X^T X:")
print(eigen_values_scaled)
print("Square roots of the scaled eigenvalues of X^T X:")
print(sqrt_eigenvalues_scaled)




df = pd.DataFrame({'x': x, 'y': y})
correlation_matrix = df.corr()
correlation_matrix = correlation_matrix.round(2)
print("Correlation Matrix:")
print(correlation_matrix)
mean_x = np.mean(x)
mean_y = np.mean(y)
numerator_x_y = np.sum((x - mean_x) * (y - mean_y))
denominator_x = np.sqrt(np.sum((x - mean_x) ** 2))
denominator_y = np.sqrt(np.sum((y - mean_y) ** 2))
correlation_coefficient = numerator_x_y / (denominator_x * denominator_y)
```

```
print(f'Sample Pearson Correlation Coefficient between x and y (r):
{ correlation_coefficient:.2f}')

mean_x=1
variance_x=2
sample_size=1000
np.random.seed(5808)
x = np.random.normal(mean_x,np.sqrt(variance_x),sample_size)
mean_epsilon = 2
variance_epsilon = 3
epsilon = np.random.normal(mean_epsilon,np.sqrt(variance_epsilon),sample_size)
y = x + epsilon
variance_y = np.var(y)
print(f"Variance of y: {variance_y:.2f}")

#%%--------------------Q7----------------
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from prettytable import PrettyTable
x_t = np.arange(-4,5,1)
y_t = x_t ** 3

def first_order_diff(y):
    return np.diff(y,n=1)
def second_order_diff(y):
    return np.diff(y,n=2)
def third_order_diff(y):
    return np.diff(y,n=3)

delta_y_t = first_order_diff(y_t)
delta2_y_t = second_order_diff(y_t)
delta3_y_t = third_order_diff(y_t)

first_order_diff = np.pad(delta_y_t.astype(float),
(1,0),mode='constant',constant_values=(np.nan,))
second_order_diff = np.pad(delta2_y_t.astype(float),
(2,0),mode='constant',constant_values=(np.nan,))
third_order_diff = np.pad(delta3_y_t.astype(float),
(3,0),mode='constant',constant_values=(np.nan,))

data = {'x(t)': x_t, 'y(t)': y_t, 'Δy(t)': first_order_diff,
'Δ2y(t)':second_order_diff, 'Δ3y(t)': third_order_diff}
df = pd.DataFrame(data)
print(df)
plt.figure(figsize=(10, 6))
plt.plot(x_t, y_t, label='Original', marker='o')
plt.plot(x_t, first_order_diff, label='1st Order Diff', marker='o')
plt.plot(x_t, second_order_diff, label='2nd Order Diff', marker='o')
plt.plot(x_t, third_order_diff, label='3rd Order Diff', marker='o')
plt.xlabel('x(t)')
plt.ylabel('y(t) / Δy(t) / Δ2y(t) / Δ3y(t)')
plt.title('Differencing of y(t) = x(t)^3')
plt.legend()
plt.grid(True)
plt.show()
```