import pandas as pd

import ipywidgets as widgets

from IPython.display import display

books\_df = pd.DataFrame(columns=['Title', 'Author', 'Genre', 'Year'])

members\_df = pd.DataFrame(columns=['Name', 'Email', 'Contact', 'Membership\_Status'])

def add\_book(title, author, genre, year):

    global books\_df

    new\_book = pd.DataFrame({'Title': [title], 'Author': [author], 'Genre': [genre], 'Year': [year]})

    books\_df = pd.concat([books\_df, new\_book], ignore\_index=True)

def edit\_book(index, title, author, genre, year):

    global books\_df

    books\_df.loc[index] = [title, author, genre, year]

def delete\_book(index):

    global books\_df

    books\_df.drop(index, inplace=True)

def add\_member(name, email, contact, membership\_status):

    global members\_df

    new\_member = pd.DataFrame({'Name': [name], 'Email': [email], 'Contact': [contact], 'Membership\_Status': [membership\_status]})

    members\_df = pd.concat([members\_df, new\_member], ignore\_index=True)

def search\_member(name):

    global members\_df

    return members\_df[members\_df['Name'] == name]

def search\_book(title):

    global books\_df

    return books\_df[books\_df['Title'] == title]

def book\_input\_form():

    title = widgets.Text(description="Title:")

    author = widgets.Text(description="Author:")

    genre = widgets.Text(description="Genre:")

    year = widgets.IntText(description="Year:")

    display(title, author, genre, year)

    return title, author, genre, year

def member\_input\_form():

    name = widgets.Text(description="Name:")

    email = widgets.Text(description="Email:")

    contact = widgets.Text(description="Contact:")

    membership\_status = widgets.Dropdown(description="Membership Status:", options=['Active', 'Inactive'])

    display(name, email, contact, membership\_status)

    return name, email, contact, membership\_status

def book\_search\_form():

    title = widgets.Text(description="Title:")

    display(title)

    return title

def member\_search\_form():

    name = widgets.Text(description="Name:")

    display(name)

    return name

def book\_actions\_form():

    action = widgets.Dropdown(description="Action:", options=['Edit', 'Delete'])

    display(action)

    return action

def member\_actions\_form():

    action = widgets.Dropdown(description="Action:", options=['Search'])

    display(action)

    return action

print("Add Book:")

title, author, genre, year = book\_input\_form()

add\_book(title.value, author.value, genre.value, year.value)

print("Book added successfully.")

print("\nEdit or Delete Book:")

book\_title = book\_search\_form()

book\_action = book\_actions\_form()

if book\_action.value == 'Edit':

    book\_index = search\_book(book\_title.value).index[0]

    new\_title, new\_author, new\_genre, new\_year = book\_input\_form()

    edit\_book(book\_index, new\_title.value, new\_author.value, new\_genre.value, new\_year.value)

    print("Book edited successfully.")

elif book\_action.value == 'Delete':

    book\_index = search\_book(book\_title.value).index[0]

    delete\_book(book\_index)

    print("Book deleted successfully.")

print("\nAdd Member:")

name, email, contact, membership\_status = member\_input\_form()

add\_member(name.value, email.value, contact.value, membership\_status.value)

print("Member added successfully.")

print("\nSearch Member:")

member\_name = member\_search\_form()

search\_result = search\_member(member\_name.value)

print("Search Result:")

print(search\_result)

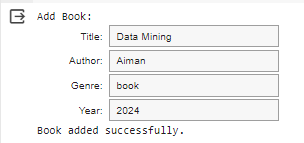
print("\nSearch Book:")

book\_title = book\_search\_form()

search\_result = search\_book(book\_title.value)

print("Search Result:")

print(search\_result)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

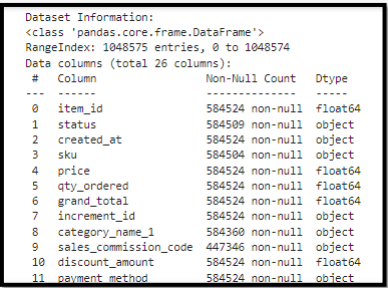
Description automatically generated

TASK 2: You work for an e-commerce company and have been given a dataset with information on customer orders over the past year. Load the data into Pandas, analyze it using methods like .info(), .describe(), Which products have the highest/lowest sales? Which customer segments spend the most?

import pandas as pd

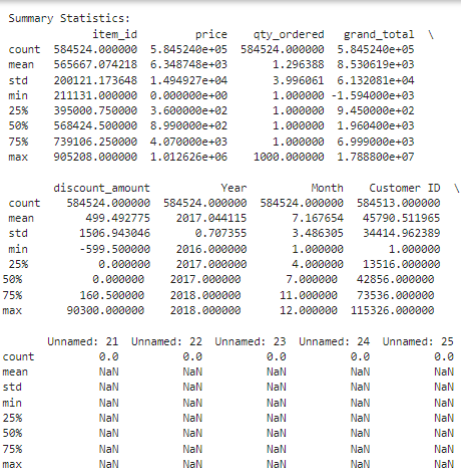
df = pd.read\_csv("Pakistan Largest Ecommerce Dataset.csv")

print("Dataset Information:") print(df.info())



print("\nSummary Statistics:")

print(df.describe())



product\_sales = df.groupby('Product')['Quantity'].sum().sort\_values(ascending=False) highest\_sales\_product = product\_sales.idxmax()

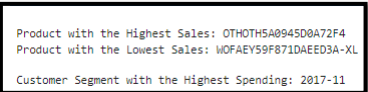
lowest\_sales\_product =product\_sales.idxmin()

print("\nProduct with the Highest Sales:", highest\_sales\_product)

print("Product with the Lowest Sales:", lowest\_sales\_product)

customer\_segment\_spending = df.groupby('Customer\_Segment')['Sales'].sum().sort\_values(ascending=Fal se) most\_spending\_segment = customer\_segment\_spending.idxmax()

print("\nCustomer Segment with the Highest Spending:", most\_spending\_segment)



TASK 3: You are a data analyst at a real estate company. You have been given a dataset of housing sale prices in different regions over the past 5 years. Load the data into Pandas and preprocess it by handling missing values and formatting columns.

import pandas as pd df = pd.read\_csv('RealEstateAU\_1000\_Samples.csv')

print("Dataset Information:") print(df.info())

A screenshot of a computer

Description automatically generated

print("\nSummary Statistics:")

print(df.describe())

A screenshot of a computer

Description automatically generated

df.dropna(inplace=True)

df['sale\_price'] = pd.to\_numeric(df['sale\_price'].str.replace('$', '').str.replace(',', '')) print("\nPreprocessed DataFrame:")

print(df.head())

A close-up of a white background

Description automatically generated

**TASK 4:** You are a data analyst working for an automobile company. You have been provided with the Vega dataset which contains details on different vehicle models like price, engine size, horsepower, dimensions etc.

import pandas as pd

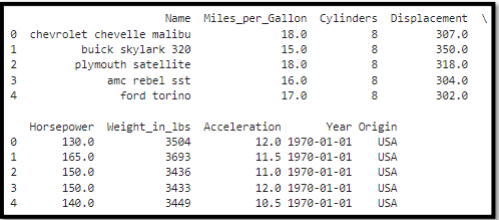
import matplotlib.pyplot as plt

import seaborn as sns

from vega\_datasets

import data df = data.cars()

print(df.head())



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

sns.scatterplot(x='Displacement', y='Horsepower', data=df) plt.title('Scatterplot of Engine Size vs. Horsepower')

plt.xlabel('Engine Size')

plt.ylabel('Horsepower')

plt.show()

A graph of a diagram

Description automatically generated with medium confidence

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

sns.histplot(df['Weight\_in\_lbs'], bins=20, kde=True) plt.title('Histogram of Weight\_in\_lbs Distribution')

plt.xlabel('Weight in lbs')

plt.ylabel('Frequency')

plt.show()

A graph of weight loss

Description automatically generated

body\_style\_stats = df.groupby('Origin').agg({'Miles\_per\_Gallon': 'mean', 'Acceleration': 'mean'}).reset\_index() alt.Chart(body\_style\_stats).mark\_bar().encode( x='Origin', y='Miles\_per\_Gallon', color=alt.Color('Origin', legend=None), tooltip=['Origin', 'Miles\_per\_Gallon'] ).properties( title='Average Miles\_per\_Gallon by Origin' ).interactive() alt.Chart(body\_style\_stats).mark\_bar().encode( x='Origin', y='Acceleration', color=alt.Color('Origin', legend=None), tooltip=['Origin', 'Acceleration'] ).properties( title='Average Acceleration by Origin' ).interactive()