LAB # 1

Bahria University,

Karachi Campus

COURSE

DATA MINING

Term: Spring 2024

Class: BSE- 6(B)

Submitted By:

NAME: AIMAN ZIA SATTI

ENROLLNMENT: 02-131212-028

Submitted To:

ENGR. HAMZA/ DR.HINA SHAKIR

Signed ______________________ Remarks: ______________________

INDEX

SNO	DATE	LAB NO	LAB OBJECTIVE	SIGN
01	14-2-24	01	GUI IN PYTHON USING GOOGLE COLAB & data mining libraries	
02				
03				
04				
05				
06				
07				
08				
09				
10				
11				
12				
131				
14				

LAB # 1

Bahria University,

Karachi Campus

LAB NO. 01

LIST OF TASKS

TASK NO	OBJECTIVE
01	▪ 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. ▪ Requirements: • Load the Vega dataset into a Pandas data frame. • Using plotting libraries like Matplotlib and Seaborn, Altair create visualizations to understand relationships between different vehicle features. Some examples: • Scatterplot of engine size vs. horsepower • Histogram of price distribution • Grouping by body style and analyzing statistics
02	▪ 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?
03	▪ 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.
04	▪ 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. ▪ Requirements: • Load the Vega dataset into a Pandas data frame. • Using plotting libraries like Matplotlib and Seaborn, Altair create visualizations to understand relationships between different vehicle features. Some examples: • Scatterplot of engine size vs. horsepower • Histogram of price distribution • Grouping by body style and analyzing statistics

Submitted On:

17th February , 2024

(Date: DD/MM/YY)

AIMAN ZIA SATTI BSE-6B 02-131212-028

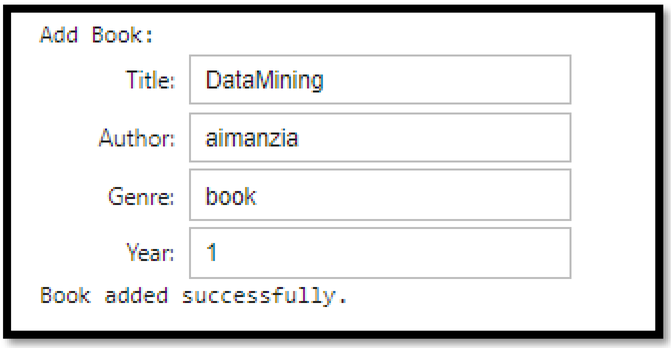
TASK #1: You work at a public library that wants to digitize its book catalog and membership management. Your task is to create a Library Management System using Google Colab to allow librarians to add/edit book records and manage member information.

- Create a Colab notebook with a simple frontend interface using Colab forms, input boxes, dropdowns, etc.
- Allow librarians to add, edit and delete book records through the interface, with details like title, author, genre, publishing year, etc.
- Records should be saved in a Pandas data frame or dictionary.
- Allow librarians to add new members, search for existing members, and edit their details like name, email, contact number, membership status etc.
- Create functionality for librarians to search books, take loans, return loans, collect fines etc. Integrate this with the member records.
- Add input validation and error handling to prevent crashes and bad data.

SOLUTION:

```
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)
```

OUTPUT:



```
Add Book:
```

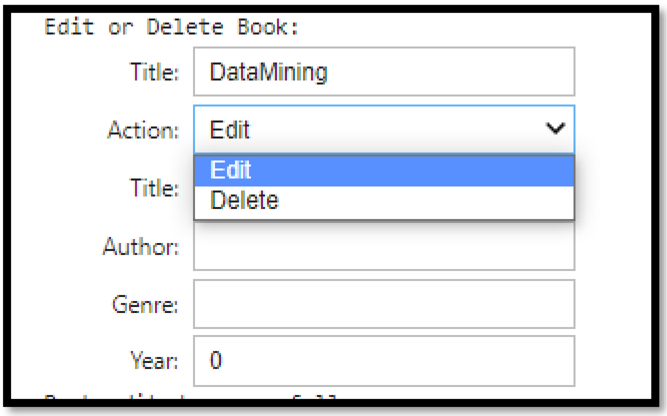
Title: DataMining

Author: aimanzia

Genre: book

Year: 1

```
Book added successfully.
```



```
Edit or Delete Book:
```

Title: DataMining

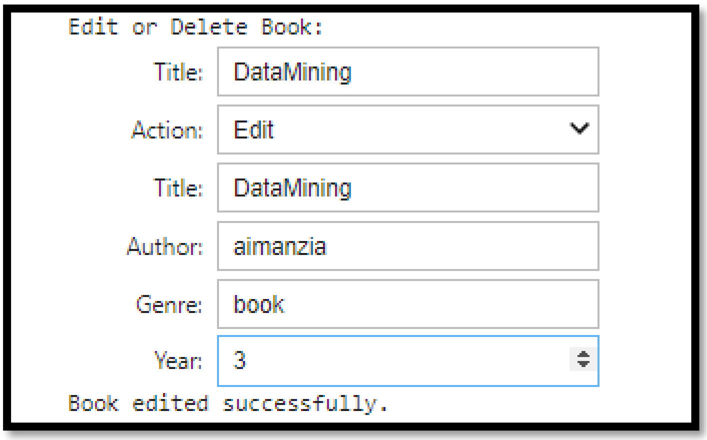
Action: Edit

Title:

Author:

Genre:

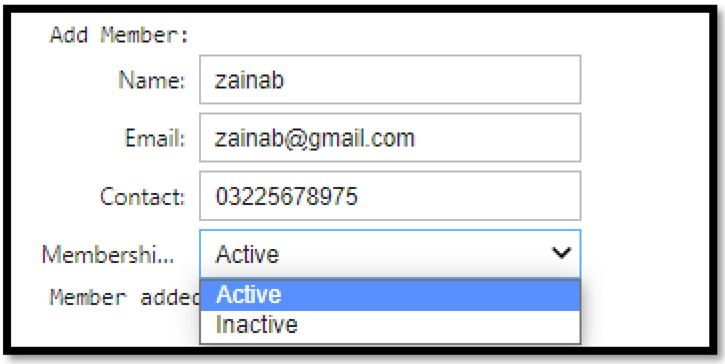
Year: 0



Edit or Delete Book:

Title:	DataMining
Action:	Edit
Title:	DataMining
Author:	aimanzia
Genre:	book
Year:	3

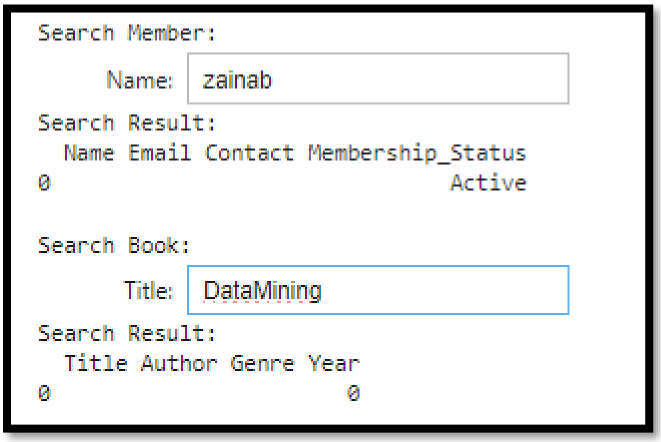
```
Book edited successfully.
```



Add Member:

Name:	zainab
Email:	zainab@gmail.com
Contact:	03225678975
Membershi...	Active

```
Member added
```



Search Member:

Name: zainab

```
Search Result:
  Name Email Contact Membership_Status
0                               Active
```

Search Book:

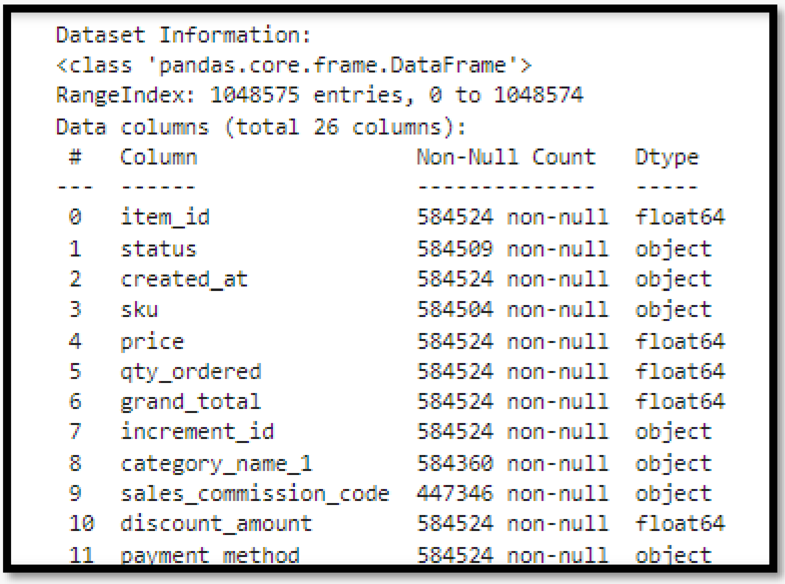
Title: DataMining

```
Search Result:
  Title Author Genre Year
0                       0
```

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?

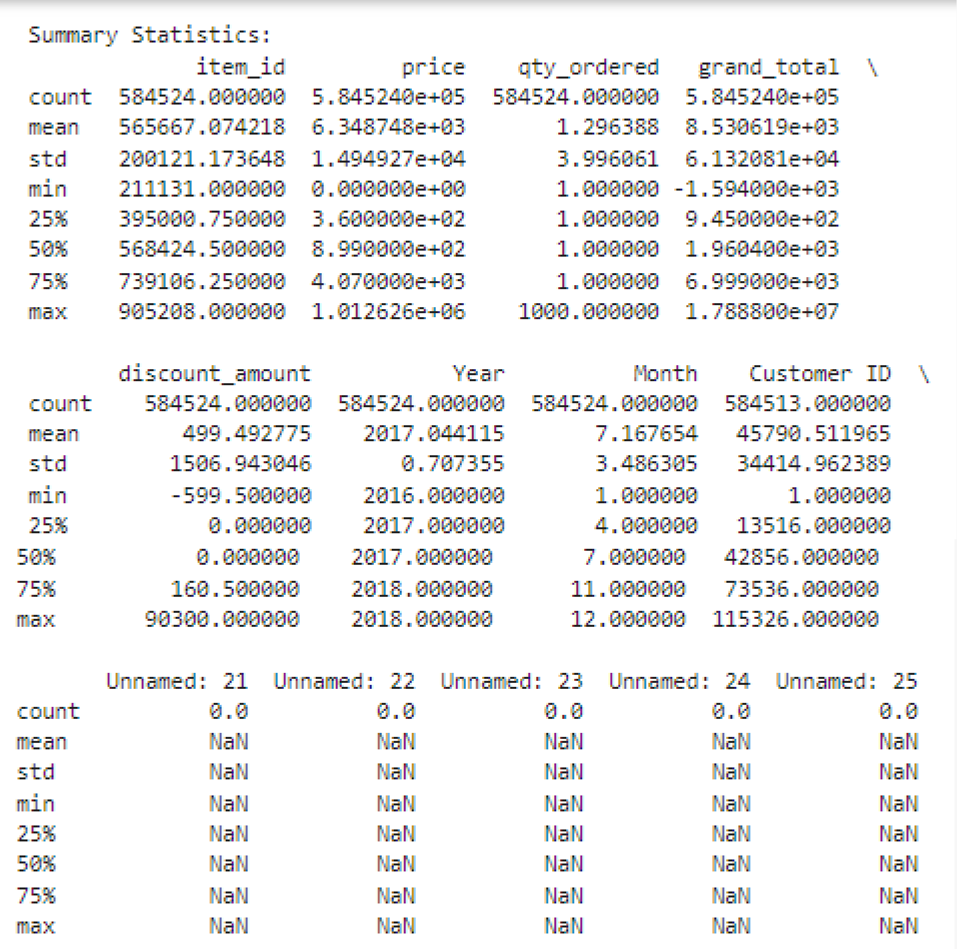
SOLUTION:

```
import pandas as pd

df = pd.read_csv("Pakistan Largest Ecommerce Dataset.csv")
print("Dataset Information:")
print(df.info())
```



```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 26 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   item_id                584524 non-null  float64
 1   status                 584509 non-null  object
 2   created_at             584524 non-null  object
 3   sku                    584504 non-null  object
 4   price                  584524 non-null  float64
 5   qty_ordered            584524 non-null  float64
 6   grand_total            584524 non-null  float64
 7   increment_id           584524 non-null  object
 8   category_name_1        584360 non-null  object
 9   sales_commission_code  447346 non-null  object
 10  discount_amount        584524 non-null  float64
 11  payment_method         584524 non-null  object
```

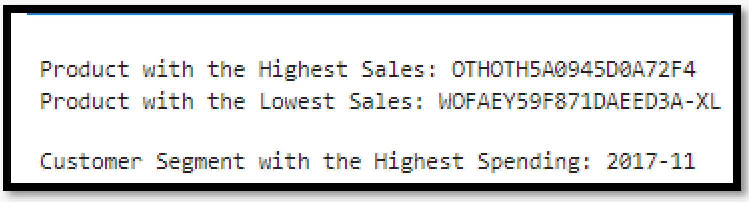
```python
print("\nSummary Statistics:")
print(df.describe())
```



```
Summary Statistics:
             item_id         price    qty_ordered   grand_total  \
count  584524.000000  5.845240e+05  584524.000000  5.845240e+05
mean   565667.074218  6.348748e+03       1.296388  8.530619e+03
std    200121.173648  1.494927e+04       3.996061  6.132081e+04
min    211131.000000  0.000000e+00       1.000000 -1.594000e+03
25%    395000.750000  3.600000e+02       1.000000  9.450000e+02
50%    568424.500000  8.990000e+02       1.000000  1.960400e+03
75%    739106.250000  4.070000e+03       1.000000  6.999000e+03
max    905208.000000  1.012626e+06    1000.000000  1.788800e+07

       discount_amount           Year          Month    Customer ID  \
count    584524.000000  584524.000000  584524.000000  584513.000000
mean        499.492775    2017.044115       7.167654   45790.511965
std        1506.943046       0.707355       3.486305   34414.962389
min        -599.500000    2016.000000       1.000000       1.000000
25%           0.000000    2017.000000       4.000000   13516.000000
50%           0.000000    2017.000000       7.000000   42856.000000
75%         160.500000    2018.000000      11.000000   73536.000000
max       90300.000000    2018.000000      12.000000  115326.000000

       Unnamed: 21  Unnamed: 22  Unnamed: 23  Unnamed: 24  Unnamed: 25
count          0.0          0.0          0.0          0.0          0.0
mean           NaN          NaN          NaN          NaN          NaN
std            NaN          NaN          NaN          NaN          NaN
min            NaN          NaN          NaN          NaN          NaN
25%            NaN          NaN          NaN          NaN          NaN
50%            NaN          NaN          NaN          NaN          NaN
75%            NaN          NaN          NaN          NaN          NaN
max            NaN          NaN          NaN          NaN          NaN
```

```python
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=False)
most_spending_segment = customer_segment_spending.idxmax()
print("\nCustomer Segment with the Highest Spending:",
most_spending_segment)
```

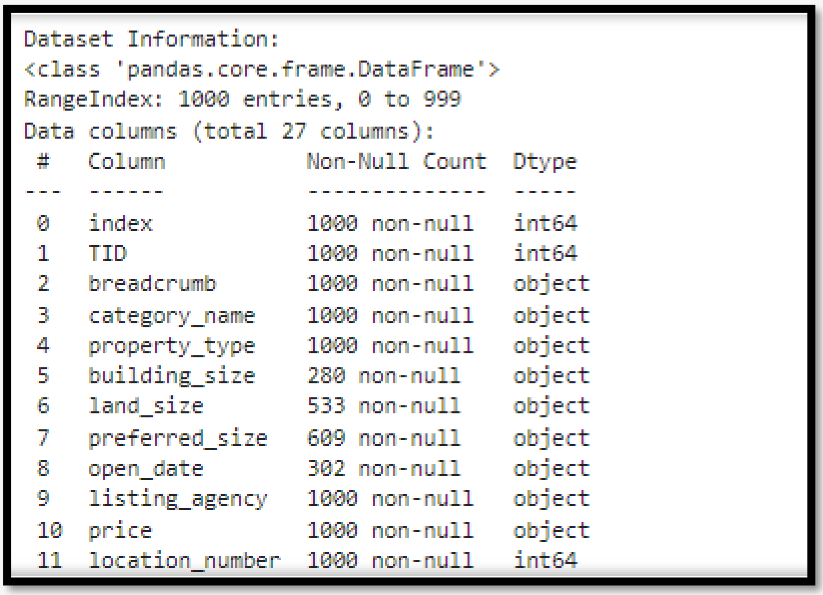


```
Product with the Highest Sales: OTHOTH5A0945D0A72F4
Product with the Lowest Sales: WOFAEY59F871DAEED3A-XL

Customer Segment with the Highest Spending: 2017-11
```

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.

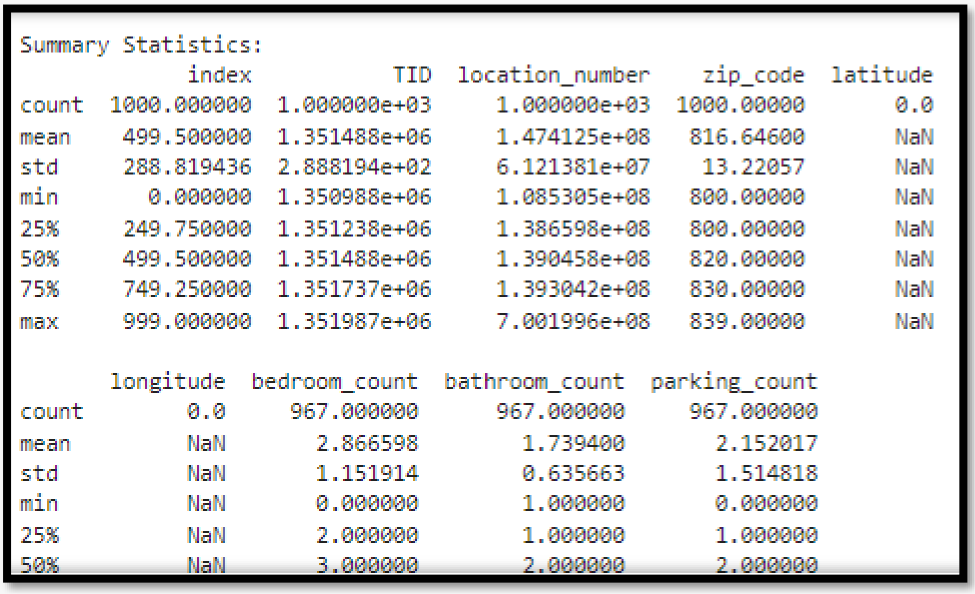
SOLUTION:

```
import pandas as pd
df = pd.read_csv('RealEstateAU_1000_Samples.csv')

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



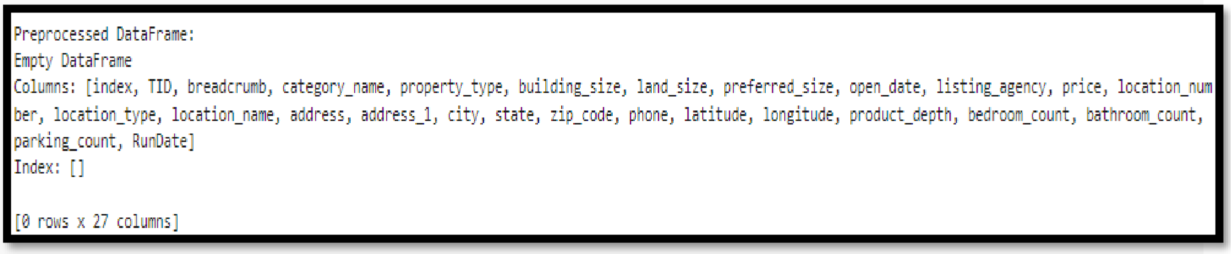
```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 27 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   index            1000 non-null   int64
 1   TID              1000 non-null   int64
 2   breadcrumb       1000 non-null   object
 3   category_name    1000 non-null   object
 4   property_type    1000 non-null   object
 5   building_size    280 non-null    object
 6   land_size        533 non-null    object
 7   preferred_size   609 non-null    object
 8   open_date        302 non-null    object
 9   listing_agency   1000 non-null   object
 10  price            1000 non-null   object
 11  location_number  1000 non-null   int64
```

```
print("\nSummary Statistics:")
print(df.describe())
```



```
Summary Statistics:
             index           TID  location_number    zip_code  latitude
count  1000.000000  1.000000e+03     1.000000e+03  1000.00000       0.0
mean    499.500000  1.351488e+06     1.474125e+08   816.64600       NaN
std     288.819436  2.888194e+02     6.121381e+07    13.22057       NaN
min       0.000000  1.350988e+06     1.085305e+08   800.00000       NaN
25%     249.750000  1.351238e+06     1.386598e+08   800.00000       NaN
50%     499.500000  1.351488e+06     1.390458e+08   820.00000       NaN
75%     749.250000  1.351737e+06     1.393042e+08   830.00000       NaN
max     999.000000  1.351987e+06     7.001996e+08   839.00000       NaN

       longitude  bedroom_count  bathroom_count  parking_count
count        0.0     967.000000      967.000000     967.000000
mean         NaN       2.866598        1.739400       2.152017
std          NaN       1.151914        0.635663       1.514818
min          NaN       0.000000        1.000000       0.000000
25%          NaN       2.000000        1.000000       1.000000
50%          NaN       3.000000        2.000000       2.000000
```

```
df.dropna(inplace=True)

df['sale_price'] = pd.to_numeric(df['sale_price'].str.replace('$',
'').str.replace(',', ''))

print("\nPreprocessed DataFrame:")
print(df.head())
```



```
Preprocessed DataFrame:
Empty DataFrame
Columns: [index, TID, breadcrumb, category_name, property_type, building_size, land_size, preferred_size, open_date, listing_agency, price, location_num
ber, location_type, location_name, address, address_1, city, state, zip_code, phone, latitude, longitude, product_depth, bedroom_count, bathroom_count,
parking_count, RunDate]
Index: []

[0 rows x 27 columns]
```

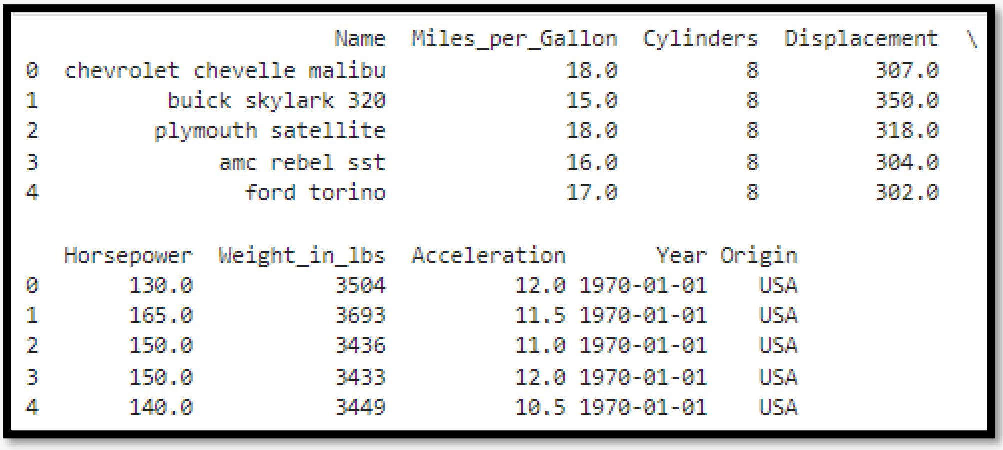
TASK 4: You are a data analyst working for an automobile company. You have been provided with the Vega dataset which contains details on different vehicle models like price, engine size, horsepower, dimensions etc.

- Requirements:
- Load the Vega dataset into a Pandas data frame.
- Using plotting libraries like Matplotlib and Seaborn, Altair create visualizations to understand relationships between different vehicle features. Some examples:
  - Scatterplot of engine size vs. horsepower
  - Histogram of price distribution
  - Grouping by body style and analyzing statistics

SOLUTION:

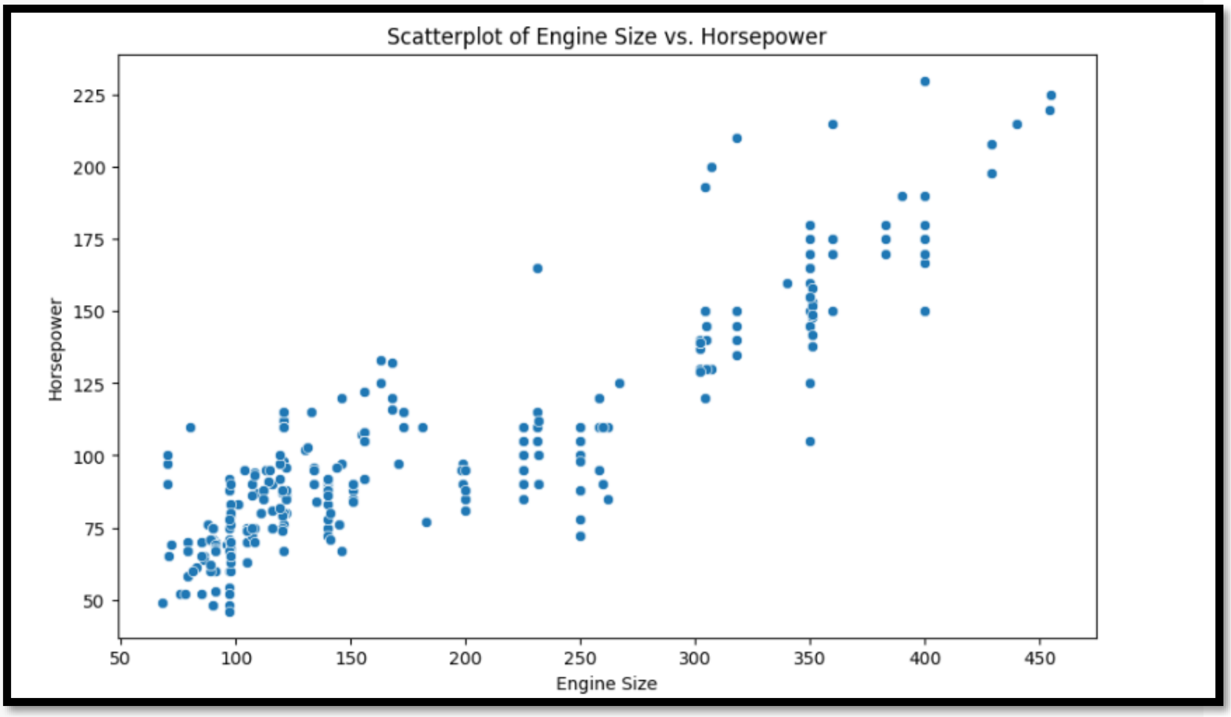
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
from vega_datasets import data
df = data.cars()

print(df.head())
```

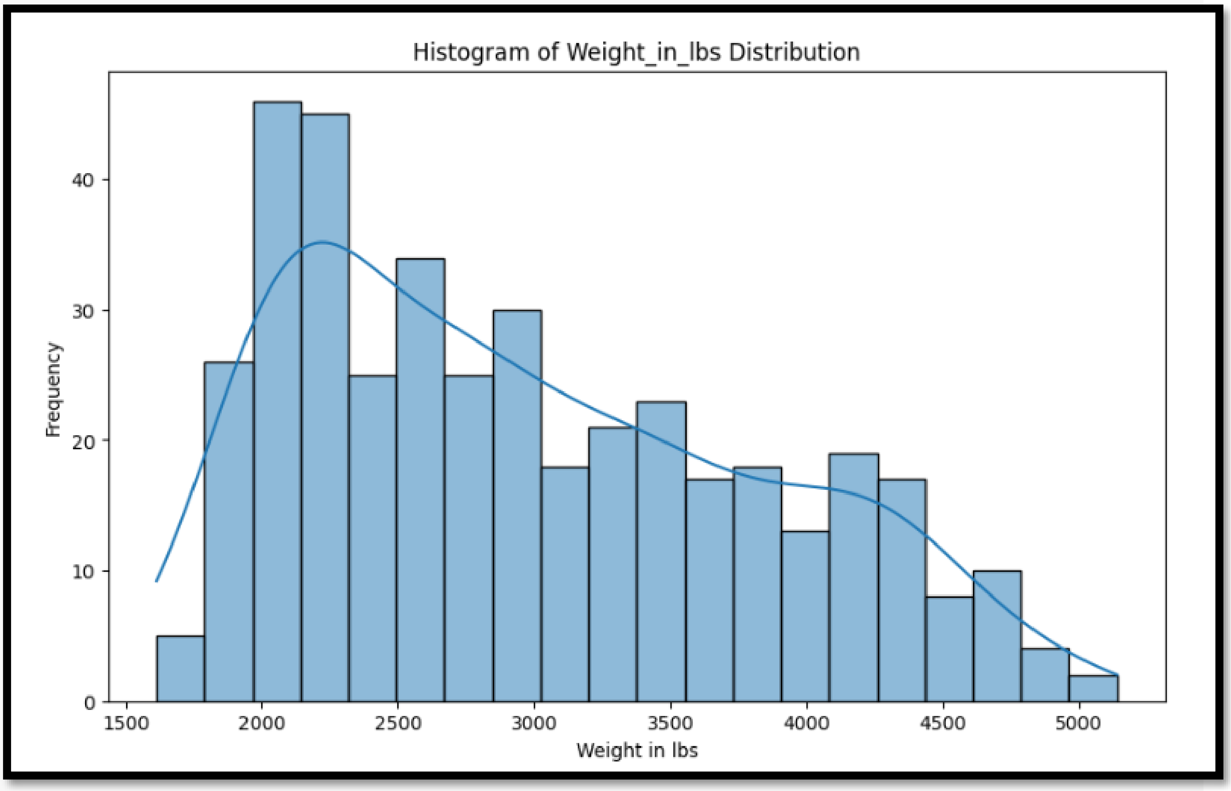


```
                        Name  Miles_per_Gallon  Cylinders  Displacement  \
0  chevrolet chevelle malibu              18.0          8         307.0
1          buick skylark 320              15.0          8         350.0
2         plymouth satellite              18.0          8         318.0
3              amc rebel sst              16.0          8         304.0
4                ford torino              17.0          8         302.0

   Horsepower  Weight_in_lbs  Acceleration       Year Origin
0       130.0           3504          12.0 1970-01-01    USA
1       165.0           3693          11.5 1970-01-01    USA
2       150.0           3436          11.0 1970-01-01    USA
3       150.0           3433          12.0 1970-01-01    USA
4       140.0           3449          10.5 1970-01-01    USA
```

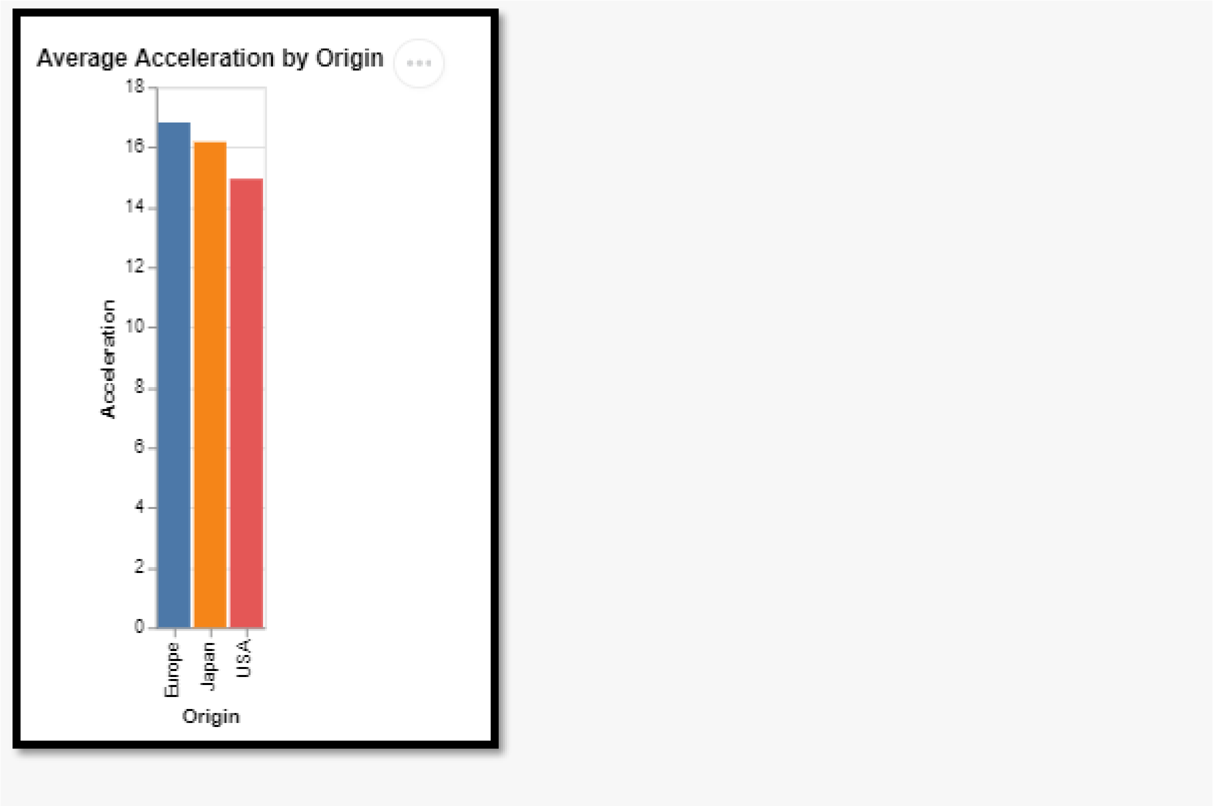
```python
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Displacement', y='Horsepower', data=df)
plt.title('Scatterplot of Engine Size vs. Horsepower')
plt.xlabel('Engine Size')
plt.ylabel('Horsepower')
plt.show()
```



```python
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()
```



```
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()
```


Average Acceleration by Origin
Acceleration
0
2
4
6
8
10
12
14
16
18
Europe
Japan
USA
Origin