**I started my initial research using ChatGPT to get clarity on basic questions. Moreover, like project 2, it was used for rephrasing to write things precisly.**

**1.How is a data frame explored?**

***ChatGPT said:***

Exploring a **DataFrame**—typically in Python using **pandas**—means getting to know your data: its structure, types, content, and any patterns or issues. Here's a beginner-friendly guide with commonly used commands:

**🧰 1. Basic Info**

python

CopyEdit

import pandas as pd   
   
# Load DataFrame   
df = pd.read\_csv("your\_file.csv") 

**🧭 2. Understand the Structure**

|  |  |  |
| --- | --- | --- |
| **Task** | **Command** | **Description** |
| View top rows | df.head() | Shows first 5 rows |
| View bottom rows | df.tail() | Shows last 5 rows |
| Shape | df.shape | Rows and columns (e.g., (100, 5)) |
| Column names | df.columns | Lists all column names |
| Index info | df.index | Shows the index (e.g., range) |

**🧪 3. Data Types & Nulls**

python

CopyEdit

df.info() 

* Shows **column types**, **non-null counts**, and **memory usage**

python

CopyEdit

df.isnull().sum() 

* Tells how many missing values are in each column

**📊 4. Descriptive Stats**

python

CopyEdit

df.describe() 

* Summary stats for numeric columns: mean, std, min, quartiles, max

python

CopyEdit

df.describe(include='object') 

* Summary for non-numeric (categorical/text) columns

**🔎 5. Quick Checks**

python

CopyEdit

df['column\_name'].value\_counts() 

* Frequency of unique values in a column

python

CopyEdit

df['column\_name'].unique() 

* Shows all unique values

python

CopyEdit

df.duplicated().sum() 

* Number of duplicated rows

**📌 6. Filtering & Sorting**

python

CopyEdit

df[df['column'] > 100]         # filter   
df.sort\_values('column')       # sort   
df.sort\_values('column', ascending=False)  # sort descending 

**📈 7. Visual Exploration (Optional)**

python

CopyEdit

import matplotlib.pyplot as plt   
import seaborn as sns   
   
df.hist(figsize=(10, 6))       # histograms for numeric data   
sns.heatmap(df.corr(), annot=True)  # correlation heatmap

**2. What is df.shape for?**

***ChatGPT said:***

df.shape is used to check the **dimensions** of a DataFrame.

**📐 What it returns:**

It returns a **tuple** in the form:

python

CopyEdit

(rows, columns) 

* **rows**: Number of observations (i.e. entries, or records)
* **columns**: Number of features or variables

**✅ Example:**

python

CopyEdit

import pandas as pd   
   
df = pd.read\_csv("data.csv")   
print(df.shape)

**Output:**

scss

CopyEdit

(150, 5)

This means:

* The DataFrame has **150 rows** (entries),
* and **5 columns** (variables).

**3. What is the title extraction data frame?**

***ChatGPT said:***

In data science or natural language processing (NLP), a **"title extraction DataFrame"** typically refers to a **Pandas DataFrame** that contains extracted **titles** from a larger dataset, such as articles, webpages, or documents.

**🔍 What Is Title Extraction?**

**Title extraction** is the process of identifying and extracting the **title** (or headline) of a text — like from:

* News articles
* Web pages (e.g., HTML <title> tags)
* Academic papers
* Social media posts
* Product descriptions, etc.

**🧾 What Does a Title Extraction DataFrame Look Like?**

It's a table (using Pandas in Python) where **titles** have been extracted and stored, usually alongside other metadata. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **source\_url** | **title** | **date** |
| 1 | <https://www.aljazeera.com/news/gaza> | "Gaza faces humanitarian crisis" | 2024-10-01 |
| 2 | <https://bbc.com/news/article123> | "Ceasefire talks resume in Egypt" | 2024-10-02 |
| 3 | <https://nytimes.com/middleeast/gaza> | "Rebuilding Gaza: The next steps" | 2024-10-03 |

**🧠 Where Might You See It?**

* **Web scraping projects** (e.g., scraping news site article titles)
* **Text analysis or NLP pipelines**
* **Machine learning datasets** (e.g., classifying news headlines)

**4. what is th code for transforming separate year and month columns into a single, sortable, human-readable month value.**

***ChatGPT said:***

The simplest and most reliable **one-liner** for transforming separate year and month columns into a **single, sortable, human-readable month value** (like "2023-11") is:

python

CopyEdit

df['month'] = pd.to\_datetime(df[['year', 'month']]).dt.to\_period('M').astype(str) 

**✅ Why This Works:**

* pd.to\_datetime(df[['year', 'month']]): combines year and month into full dates (defaults to the 1st of the month).
* .dt.to\_period('M'): converts the date to **just the year and month** (Period object).
* .astype(str): makes it a clean, string label like "2023-11" — perfect for grouping and plotting.

**📌 Example:**

If you have:

python

CopyEdit

year  month   
2023     3   
2023    11 

You’ll get:

python

CopyEdit

month   
2023-03   
2023-11 

**🧠 Use Case:**

This one-liner is best when:

* You want chronological sorting (e.g., in line charts).
* You want simple grouping (e.g., df.groupby('month')).
* You want a clean label like "2023-11" instead of handling messy string formatting.