**COVER PAGE**

**CYBER SHUJAA PROGRAM**

**WEEK 3 ASSIGNMENT: EXPLORATORY DATA ANALYSIS**

**STUDENT NAME: DOROTHY AWINO ONGONGA**

**STUDENT ID: CS-DA01-25075**

**KAGGLE JUPYTER NOTEBOOK LINK:** [**https://www.kaggle.com/code/awino614/titanic-exploratory-data-analysis**](https://www.kaggle.com/code/awino614/titanic-exploratory-data-analysis)

**GOOGLE COLAB LINK:** [**https://colab.research.google.com/#fileId=https%3A//storage.googleapis.com/kaggle-colab-exported-notebooks/awino614/titanic-exploratory-data-analysis.b0c8e25d-7092-4516-a000-3f119e6eeeb4.ipynb%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-161607.iam.gserviceaccount.com/20250613/auto/storage/goog4\_request%26X-Goog-Date%3D20250613T083624Z%26X-Goog-Expires%3D259200%26X-Goog-SignedHeaders%3Dhost%26X-Goog-Signature%**](https://colab.research.google.com/#fileId=https%3A//storage.googleapis.com/kaggle-colab-exported-notebooks/awino614/titanic-exploratory-data-analysis.b0c8e25d-7092-4516-a000-3f119e6eeeb4.ipynb%3FX-Goog-Algorithm%3DGOOG4-RSA-SHA256%26X-Goog-Credential%3Dgcp-kaggle-com%2540kaggle-161607.iam.)

**GITHUB LINK:** <https://github.com/Awino614/DATA-WRANGLING>

**DATE: 13/6/2025**

TABLE OF CONTENTS

[INTRODUCTION 2](#_Toc28667)

[1. Initial Exploration 2](#_Toc19043)

[2. Descriptive Statistics 2](#_Toc1796)

[3. Missing Value Analysis 2](#_Toc32583)

[4. Outlier Detection 2](#_Toc5884)

[5. Data Visualization 2](#_Toc28051)

[TASKS COMPLETION GUIDE 3](#_Toc29468)

[2. INITIAL DATA EXPLORATION 5](#_Toc24391)

[STEP 3.Handling Missing Values¶ 13](#_Toc6759)

[. Bivariate Analysis¶ 25](#_Toc18512)

[9. Key Insights and Reporting¶ 44](#_Toc17959)

[1. Feature Selection and Engineering: 45](#_Toc1441)

[2. Handling Missing Data: 45](#_Toc27939)

[3. Handling Imbalanced Classes: 45](#_Toc4603)

[4. Outlier Treatment: 45](#_Toc27088)

[5. Feature Scaling: 45](#_Toc17895)

[ CONCLUSION 45](#_Toc8911)

**INTRODUCTION**

The project’s main objective is to develop hands-on experience on Exploratory Data Analysis using the Kaggle Data Set and publishing on Kaggle.

Link: <https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda>

**Exploratory Data Analysis (EDA)** is the process of examining and visualizing data sets to summarize their main characteristics, often before applying any modeling techniques. It helps analysts and data scientists understand the structure, patterns, relationships, and anomalies in the data.

### 🔍 ****Key Purposes of EDA****

* To **gain insights** into the data
* To **check data quality** (missing values, duplicates, data types)
* To **identify patterns**, trends, and outliers
* To **form hypotheses** or choose the right analytical models

### 🛠️ ****Common EDA Tasks****

1. **Initial Exploration**
   * Previewing the data (df.head(), df.info())
   * Checking data types and dimensions
2. **Descriptive Statistics**
   * Mean, median, standard deviation, min, max
   * Value counts for categorical variables
3. **Missing Value Analysis**
   * Locating and deciding how to handle missing data
4. **Outlier Detection**
   * Identifying unusually high or low values
5. **Data Visualization**
   * **Univariate analysis** (e.g., histograms for single variables)
   * **Bivariate analysis** (e.g., scatter plots, boxplots)
   * **Multivariate analysis** (e.g., heatmaps, pairplots)

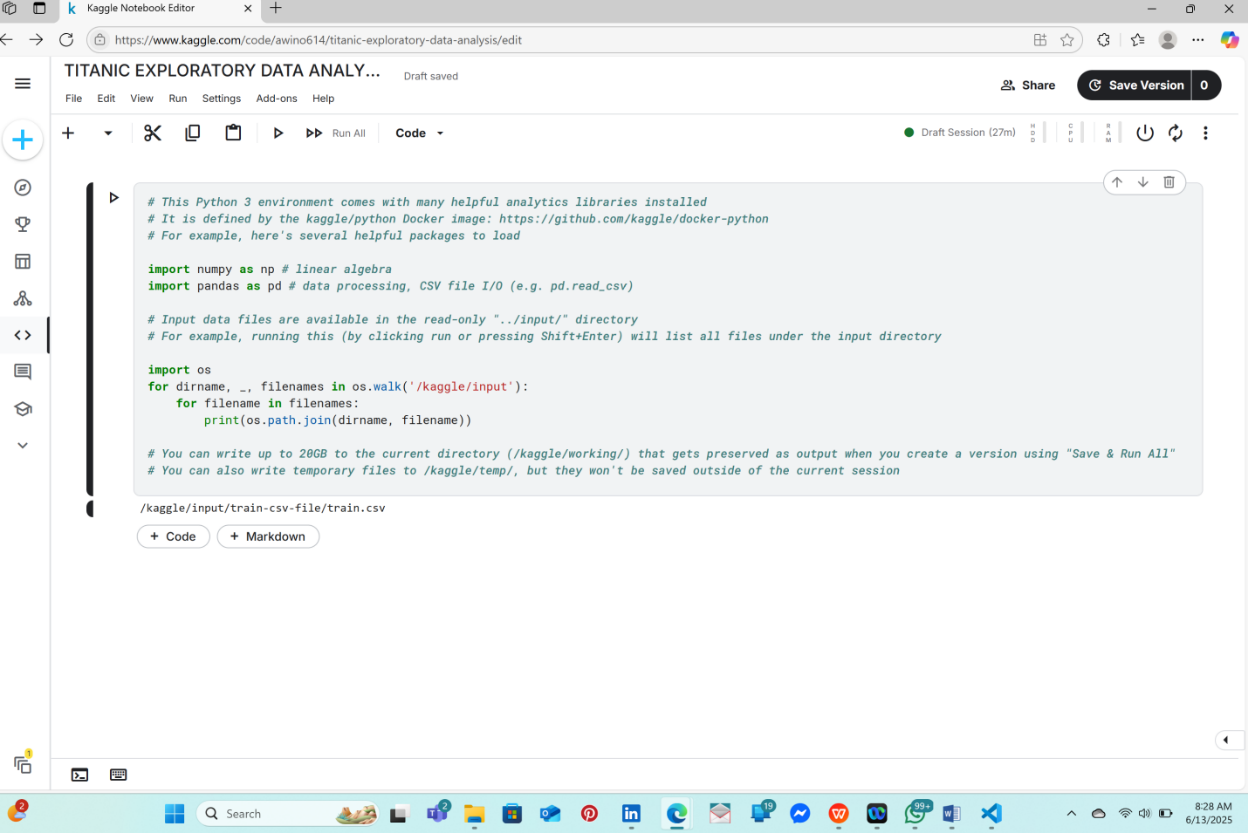
📈 **Why EDA is Important**

* Helps you **clean** and **prepare** data properly
* Prevents costly mistakes by uncovering hidden data issues
* Provides a foundation for building accurate models
* Enables better communication of findings through visualization.

**TASKS COMPLETION GUIDE**

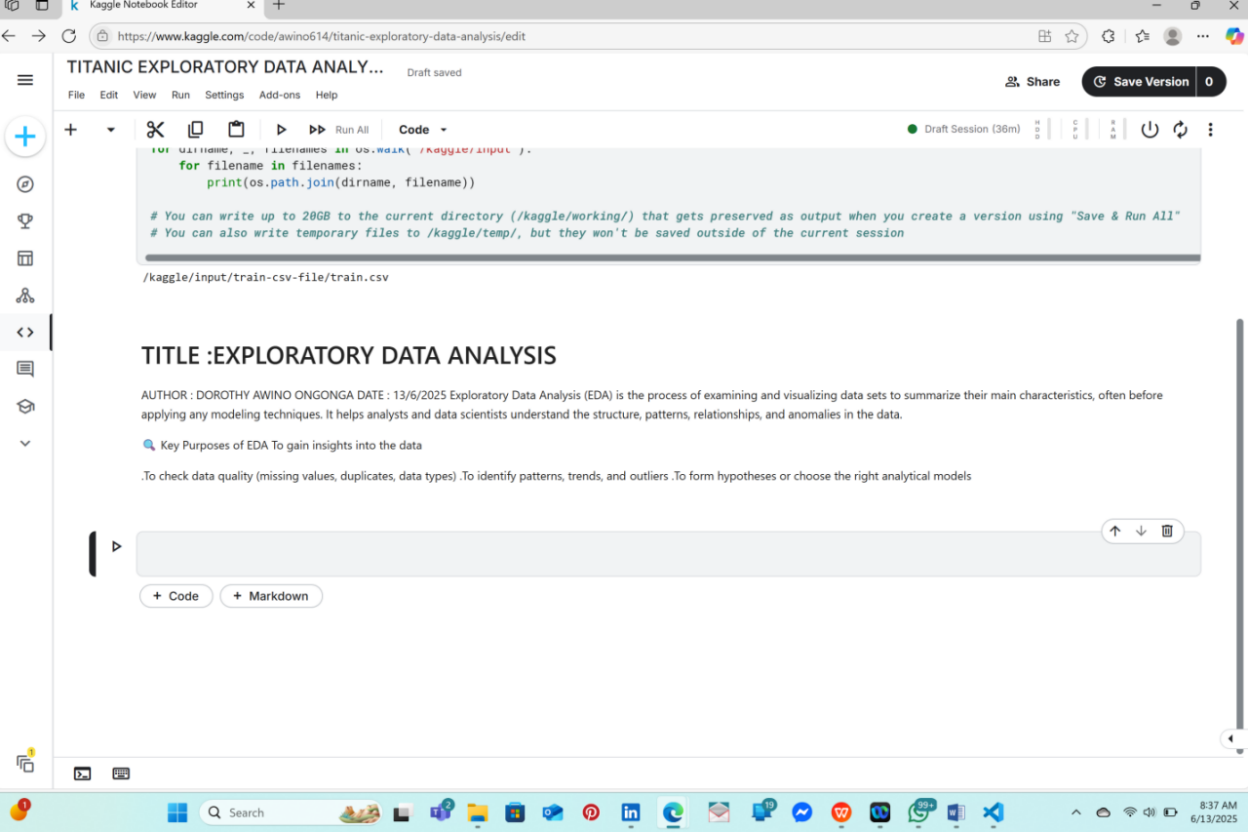
**DOWNLOADING AND UPLOADING FILE**

I made an initial step of downloading & uploading the train.csv dataset to my notebook to enable me to perform the EDA tasks.



**INTRODUCTORY COMMENT SECTION**

To give the project an introductory guide, I introduced a short comment section using the markdown feature

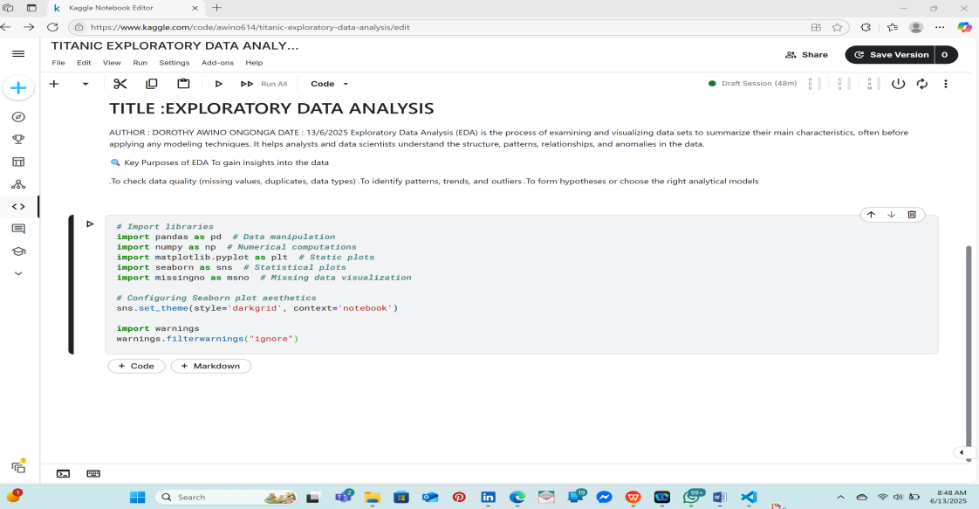


**IMPORTING PYTHON LIBRARIES**

## **Tools and Libraries for EDA**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#%F0%9F%9B%A0-Tools-and-Libraries-for-EDA)

To perform EDA efficiently, I used several Python libraries:

* **pandas**: For data manipulation and inspection.
* **numpy**: For numerical computations.
* **matplotlib**: For static visualizations (e.g., histograms, scatter plots).
* **seaborn**: For statistical plots and heatmaps.
* **missingno**: For visualizing missing data.
* **plotly/altair**: For interactive visualizations.



# **2. INITIAL DATA EXPLORATION**

**📋 Why Do We Perform an Initial Data Exploration?**

Before diving into complex analyses, the first thing I did with the dataset is **take a quick tour of its structure**. Think of it as **getting the lay of the land**—you want to understand:

* How many rows and columns are in the dataset?
* What types of data does each column contain (numerical, categorical, text, datetime)?
* Are there any missing values?
* Do the values in each column make sense (e.g., no negative ages for people)?

### ****Key Questions to Ask:****

1. **What does the dataset look like?**
2. **How many features (columns) and records (rows) are there?**
3. **What data types do the columns have?**
4. **Are there any missing values or duplicates?**
5. **Are there any obvious errors (like typos, outliers, or negative values)?**

## **🛠 Key Functions for Initial Exploration**

Here are some essential pandas functions used for initial exploration:

* **df.head()**: Displays the first few rows of the dataset to give you a quick preview.
* **df.shape**: Returns the number of rows and columns in the dataset.
* **df.info()**: Provides details about the columns, their data types, and the number of non-null (non-missing) values.
* **df.describe()**: Provides summary statistics (mean, median, min, max, etc.) for numerical columns.
* **df.columns**: Lists the names of all columns in the dataset.
* **df.nunique()**: Returns the number of unique values in each column.
* **df.duplicated()**: Checks for duplicate rows.

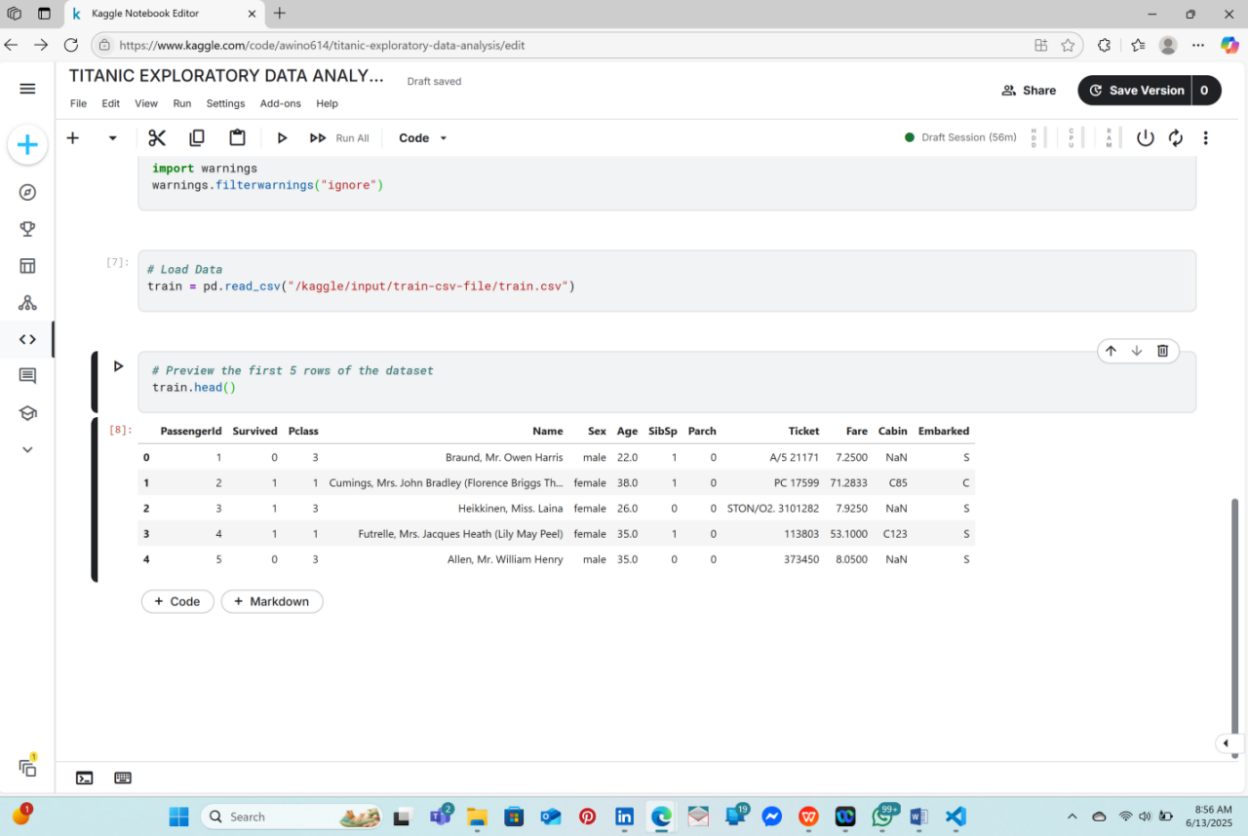
add Codeadd Markdown

## **Step 2.1 Previewing the Dataset**

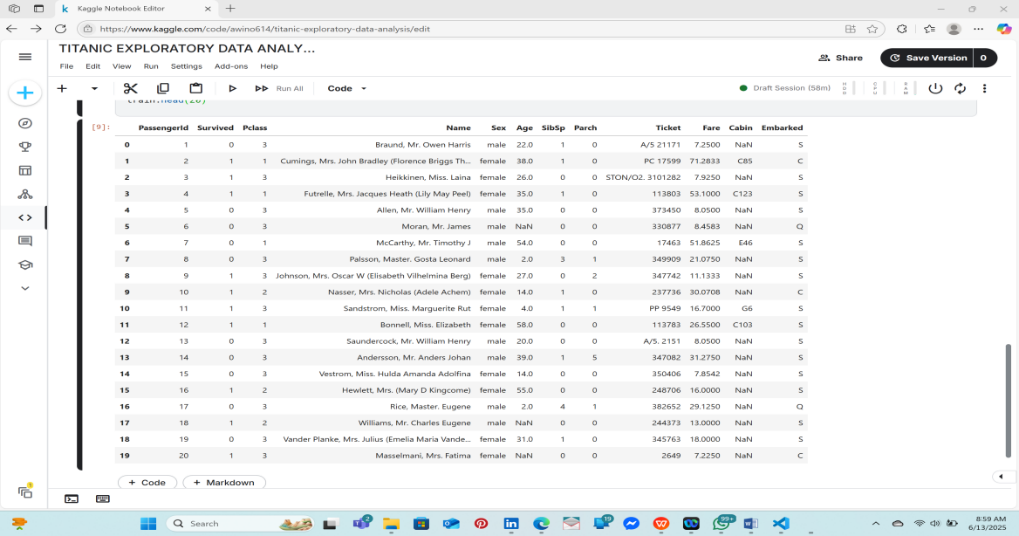
The first step is to get a **quick look at the dataset’s contents** using df.head(). This function shows the first 5 rows by default:

### ****Why Use .head()?****

It provides a quick overview of the dataset's structure and values. You can spot-check for potential anomalies or unexpected values (e.g., nulls, typos, negative values).



**# Preview the first 20 rows of the dataset**



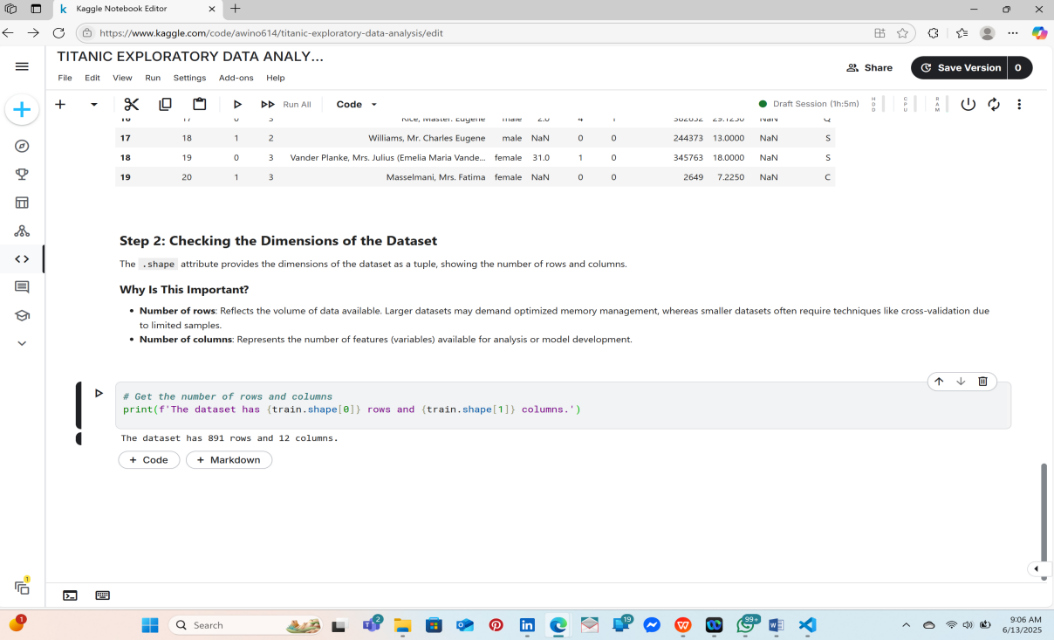
**Step 2.2: Checking the Dimensions of the Dataset**

The .shape attribute provides the dimensions of the dataset as a tuple, showing the number of rows and columns.

#### **Why Is This Important?**

* **Number of rows**: Reflects the volume of data available. Larger datasets may demand optimized memory management, whereas smaller datasets often require techniques like cross-validation due to limited samples.
* **Number of columns**: Represents the number of features (variables) available for analysis or model development.

💡 In the Titanic dataset, there are **891 rows** and **12 columns**, indicating a moderately sized dataset that is manageable for exploratory data analysis



🧭 **Step 3: Overview of Columns and Data Types**

The .info() function provides a concise summary of the dataset's structure.

#### **What Does the Output Reveal?**

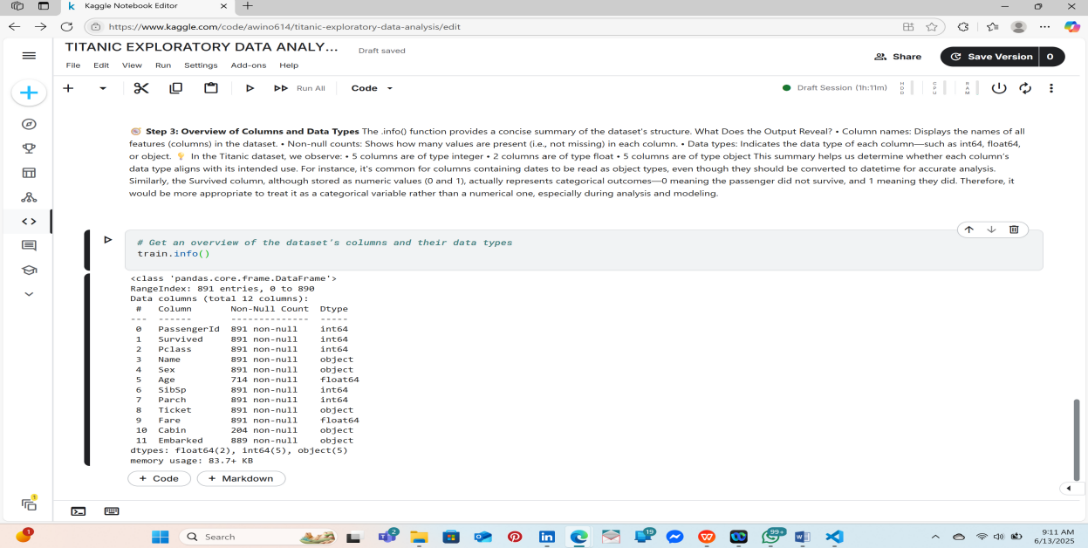
* **Column names**: Displays the names of all features (columns) in the dataset.
* **Non-null counts**: Shows how many values are present (i.e., not missing) in each column.
* **Data types**: Indicates the data type of each column—such as int64, float64, or object.

💡 In the Titanic dataset, we observe:

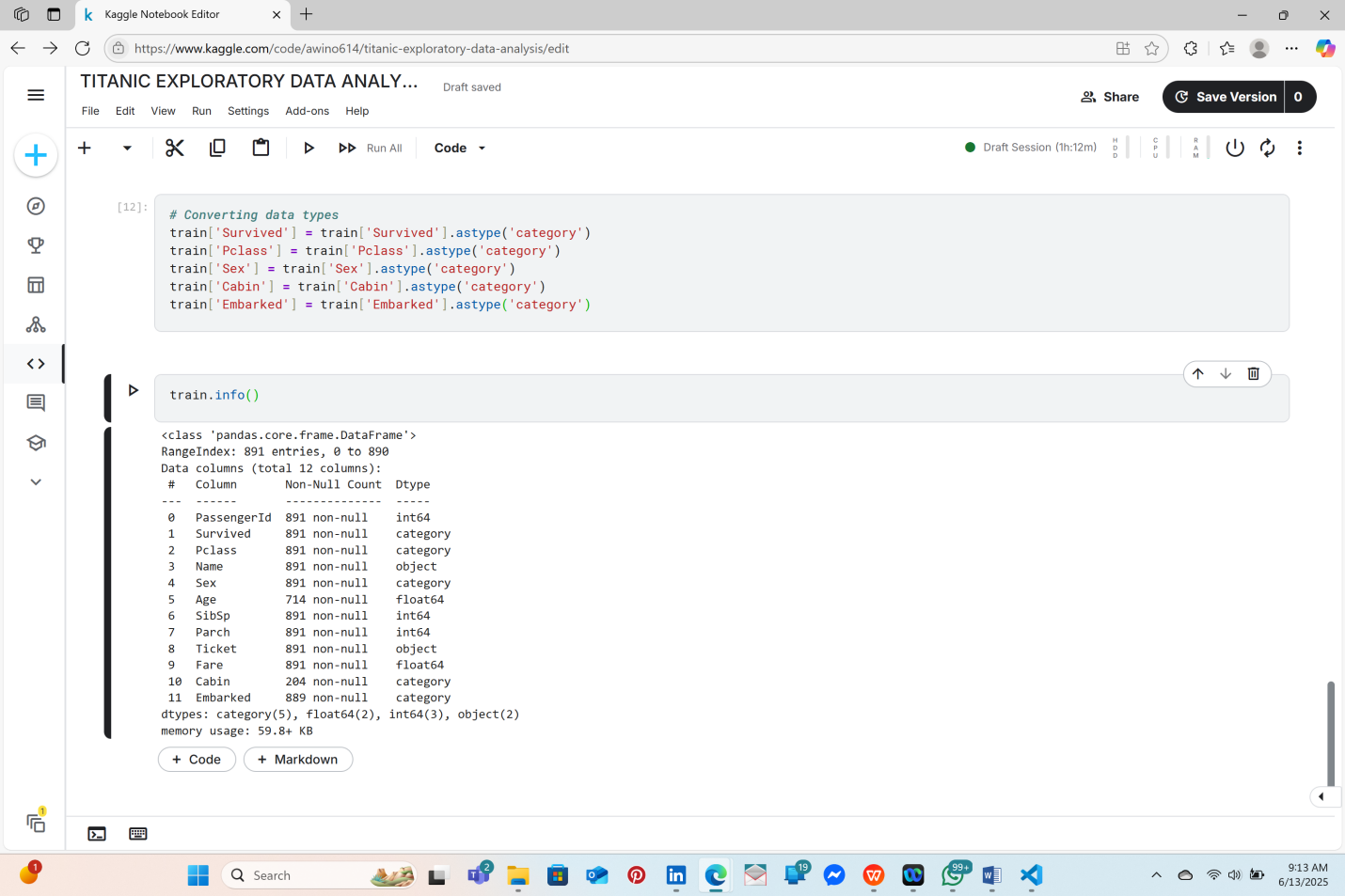
* 5 columns are of type **integer**
* 2 columns are of type **float**
* 5 columns are of type **object**

This summary helps us determine whether each column’s data type aligns with its intended use. For instance, it's common for columns containing dates to be read as object types, even though they should be converted to datetime for accurate analysis.

Similarly, the Survived column, although stored as numeric values (0 and 1), actually represents categorical outcomes—**0** meaning the passenger did not survive, and **1** meaning they did. Therefore, it would be more appropriate to treat it as a **categorical** variable rather than a numerical one, especially during analysis and modeling.



**CONVERTING DATA TYPES**



**Step 4: Statistical Summary of Numerical Features**

The .describe() function provides a quick overview of summary statistics for all numerical columns in the dataset.

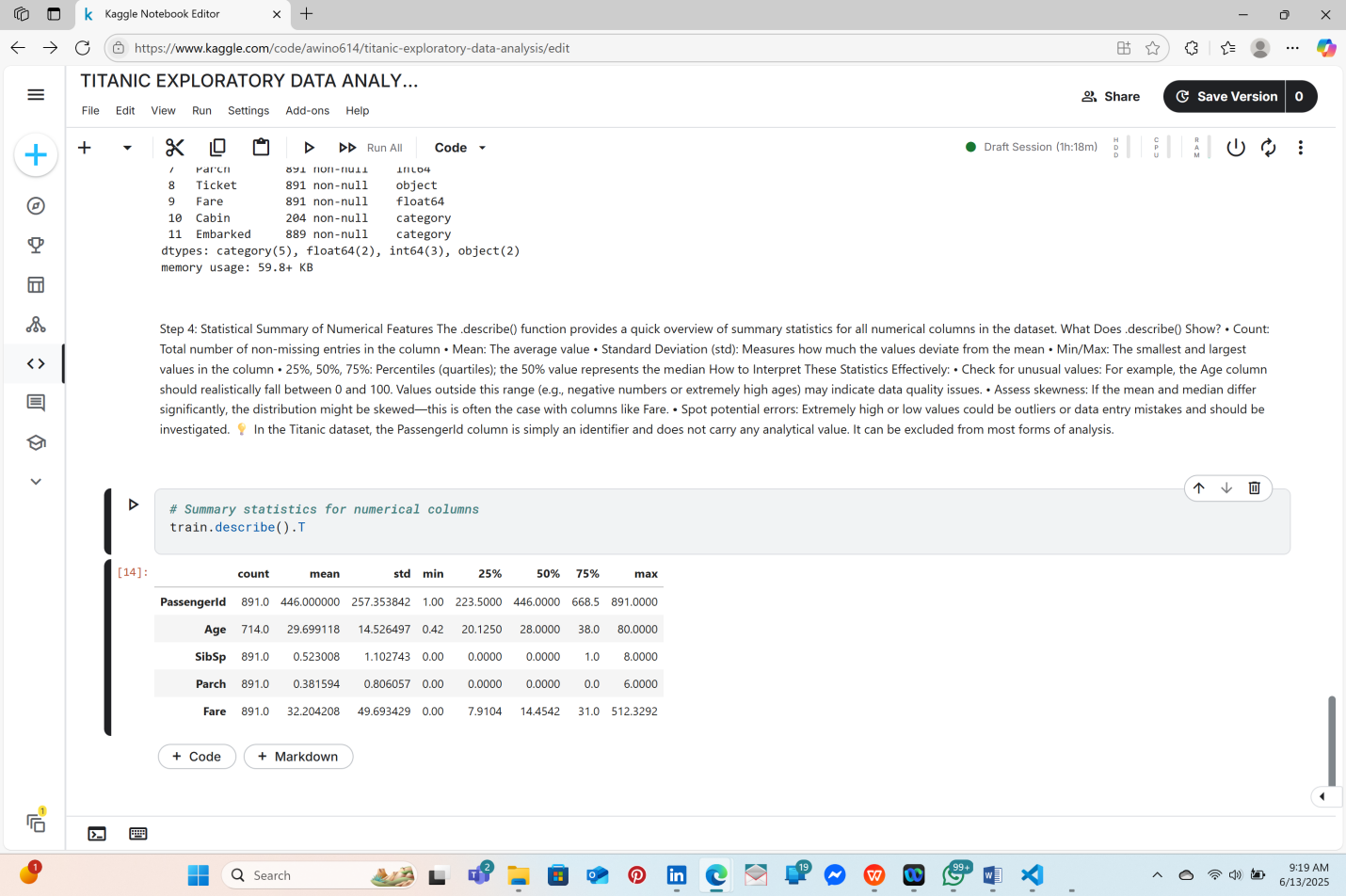
#### **What Does** .describe() **Show?**

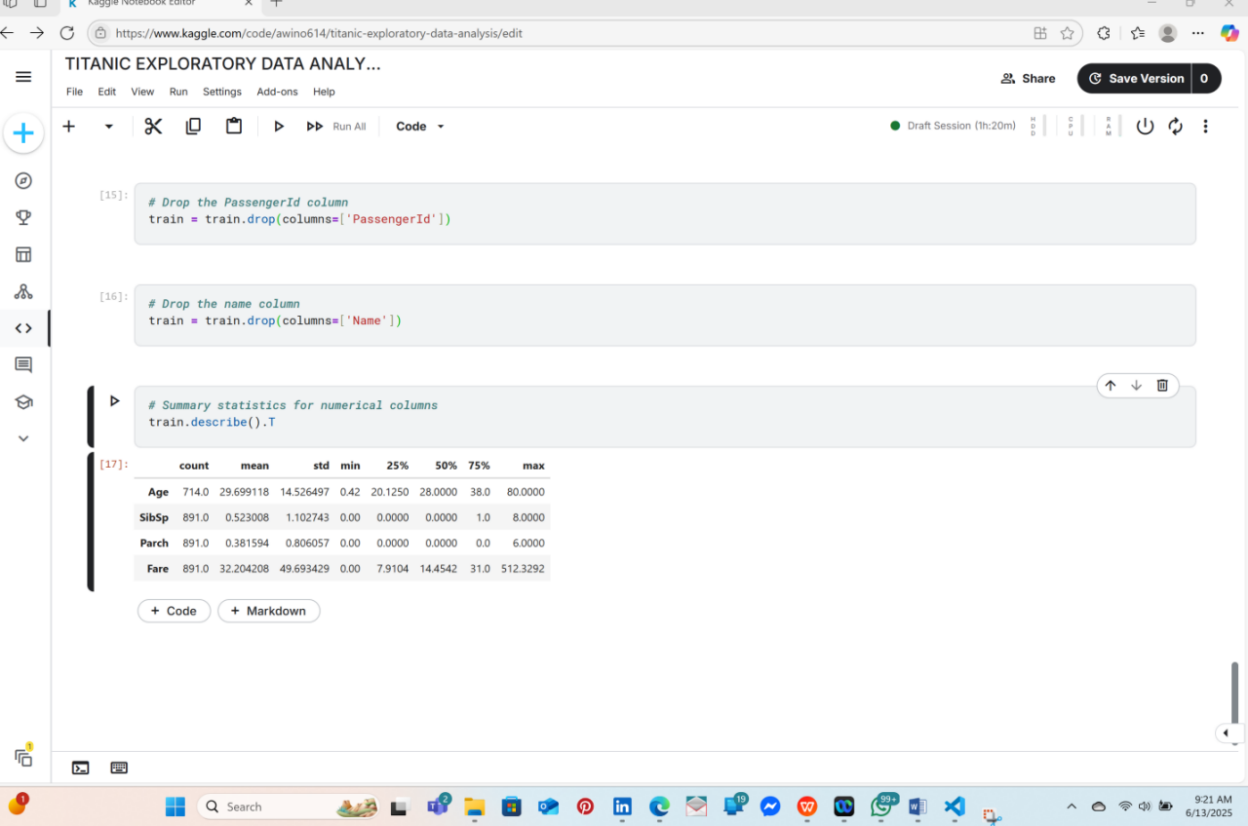
* **Count**: Total number of non-missing entries in the column
* **Mean**: The average value
* **Standard Deviation (std)**: Measures how much the values deviate from the mean
* **Min/Max**: The smallest and largest values in the column
* **25%, 50%, 75%**: Percentiles (quartiles); the 50% value represents the median

#### **How to Interpret These Statistics Effectively:**

* **Check for unusual values**: For example, the Age column should realistically fall between 0 and 100. Values outside this range (e.g., negative numbers or extremely high ages) may indicate data quality issues.
* **Assess skewness**: If the mean and median differ significantly, the distribution might be skewed—this is often the case with columns like Fare.
* **Spot potential errors**: Extremely high or low values could be outliers or data entry mistakes and should be investigated.

💡 In the Titanic dataset, the PassengerId column is simply an identifier and does not carry any analytical value. It can be excluded from most forms of analysis.





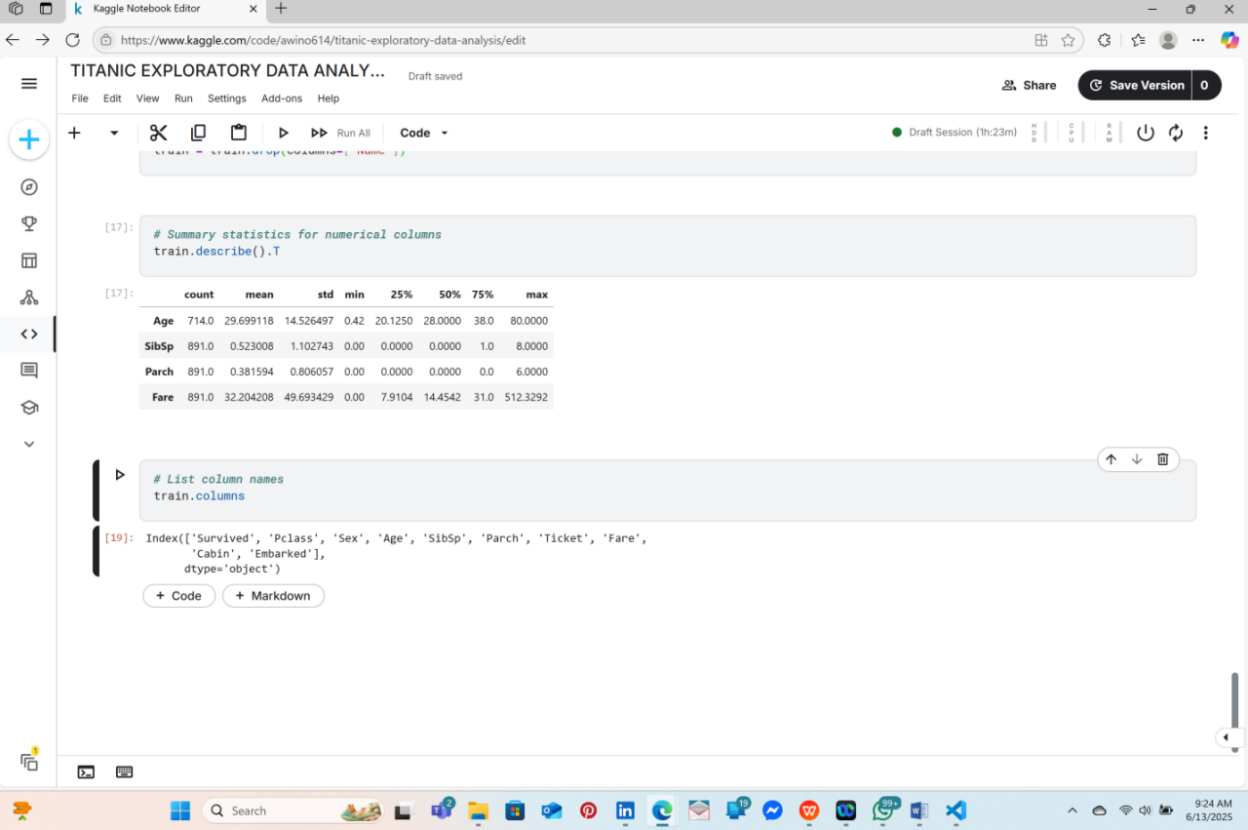
🧭 **Step 5: Displaying Column Names**

You can use the .columns attribute to view all the column names in the dataset.

#### **Why Is This Important?**

* Allows for a quick overview of all available features.
* Helps identify columns that may need to be renamed, cleaned, or removed.
* Useful for spotting inconsistencies or irrelevant features.

💡 In the Titanic dataset, the PassengerId column serves only as a unique identifier and does not contribute to the analysis—so it can be excluded from further exploration.



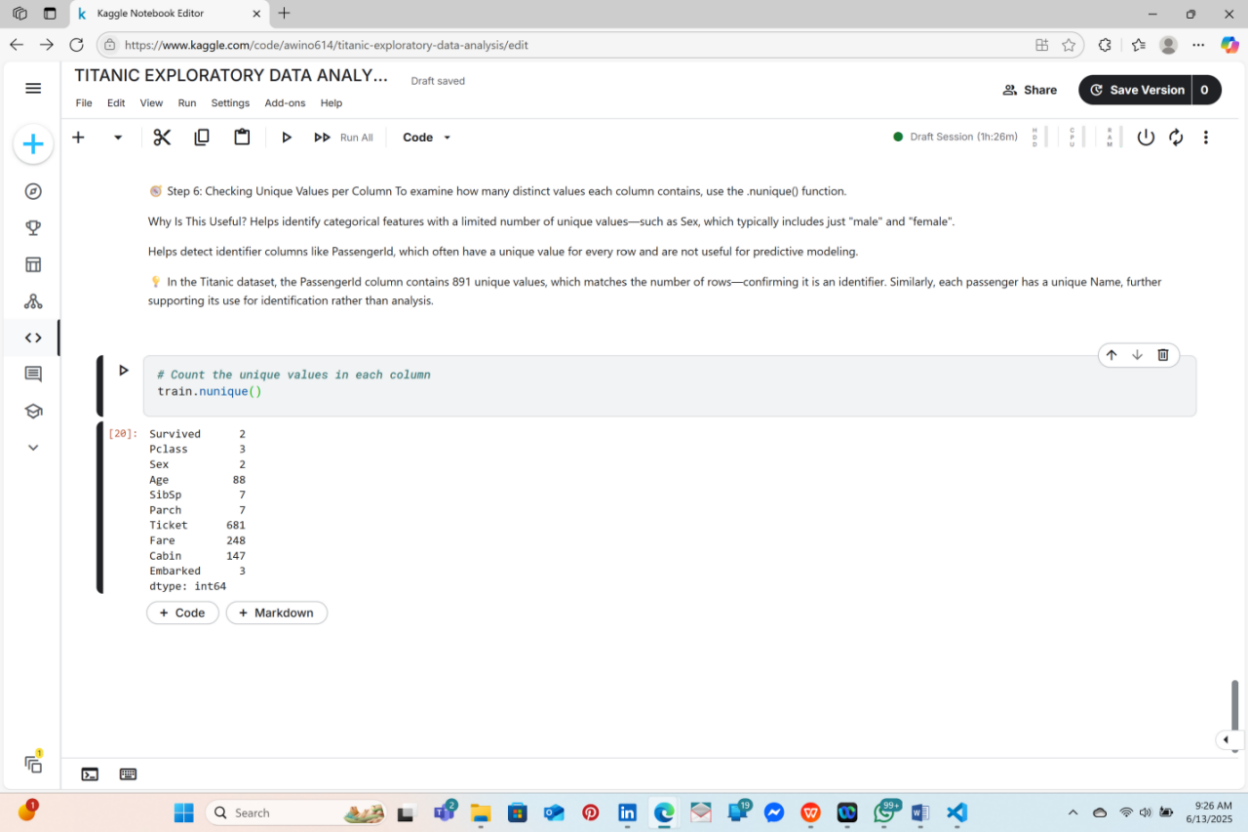
### 🧭 ****Step 6: Checking Unique Values per Column****

To examine how many distinct values each column contains, use the .nunique() function.

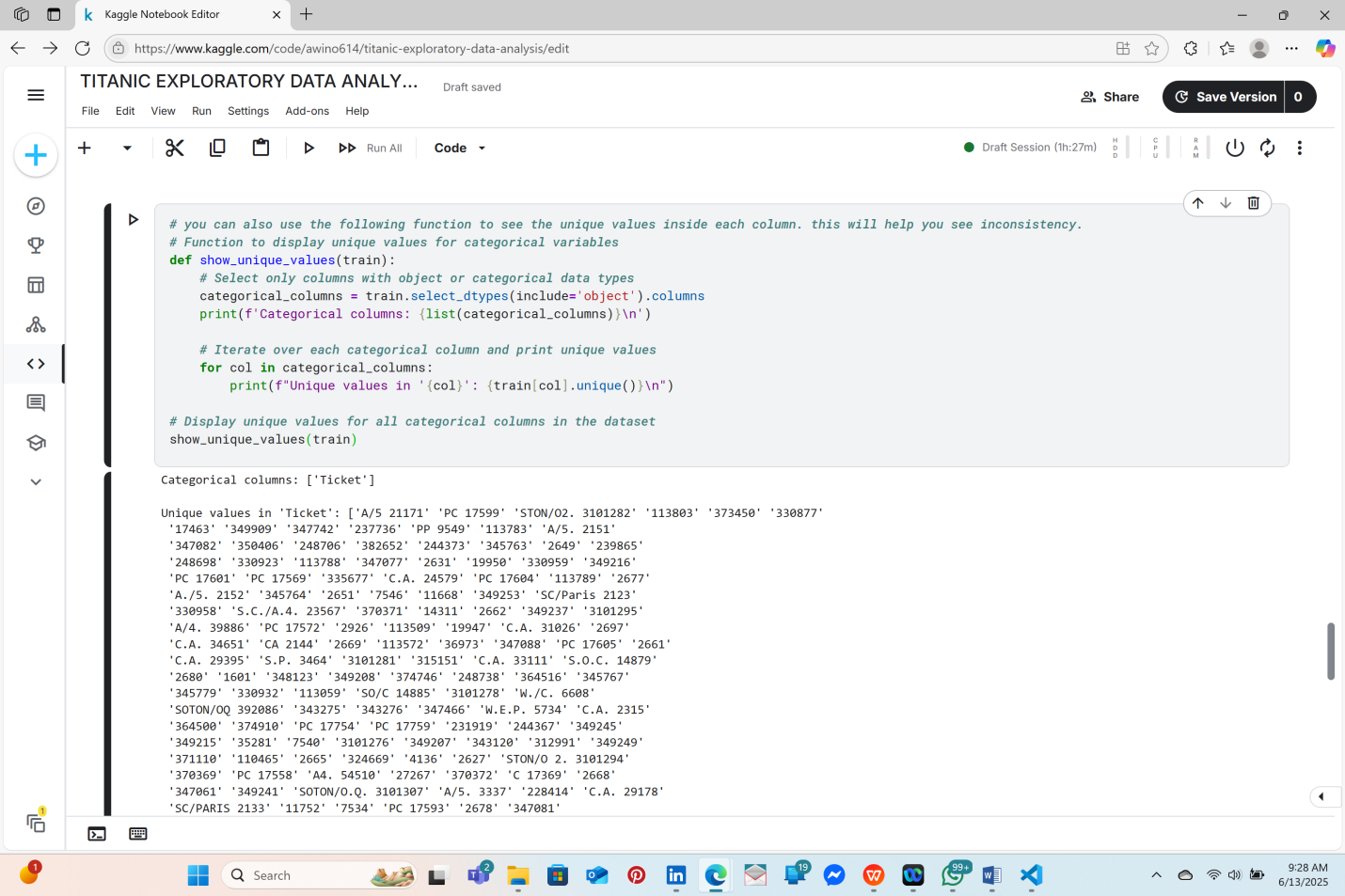
#### **Why Is This Useful?**

* Helps identify **categorical features** with a limited number of unique values—such as Sex, which typically includes just "male" and "female".
* Helps detect **identifier columns** like PassengerId, which often have a unique value for every row and are not useful for predictive modeling.

💡 In the Titanic dataset, the PassengerId column contains **891 unique values**, which matches the number of rows—confirming it is an identifier. Similarly, each passenger has a unique Name, further supporting its use for identification rather than analysis.



**CATEGORICAL COLUMNS**



You can view the full columns using the notebook link indicated at the cover page of this report.

## **Common Issues You Might Spot in Initial Data Exploration**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#%F0%9F%9A%A9-Common-Issues-You-Might-Spot-in-Initial-Data-Exploration)

* **Missing Values:** Some columns may have missing data (e.g., Cabin in Titanic).
* **Outliers:** Some columns may contain unusually high or low values.
* **Irrelevant Features:** Columns like PassengerId may not contribute to the analysis.
* **Data Type Mismatches:** Columns with dates may be stored as strings, or numerical data may be stored as text.

linkcode

## **📝 Summary of Initial Data Exploration**

### ****In this section, we have:****

* Previewed the dataset using .head() to check the first few rows.
* Checked the size of the dataset using .shape().
* Viewed column names and data types using .info().
* Generated summary statistics for numerical columns using .describe().
* Displayed unique values for categorical columns using a custom function.
* Checked for duplicate rows and learned how to remove them if needed.

# **STEP 3.Handling Missing Values**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#3.-Handling-Missing-Values)

## **🚩 Why Are Missing Values Important?**

Missing values are one of the most common issues in real-world datasets. They occur due to:

* **Human errors** during data collection or entry.
* **Technical issues** (e.g., data corruption or transmission errors).
* **Incomplete information** (e.g., passengers on the Titanic not providing their cabin details).

Ignoring missing values can lead to:

1. **Bias in analysis**: Missing data may skew averages or distributions.
2. **Errors in computations**: Some machine learning algorithms cannot handle missing values and may crash.
3. **Loss of valuable information**: Dropping rows or columns without thought can reduce the size of your dataset unnecessarily.

## **🛠 Common Methods to Handle Missing Values**

There are three main approaches to handling missing values:

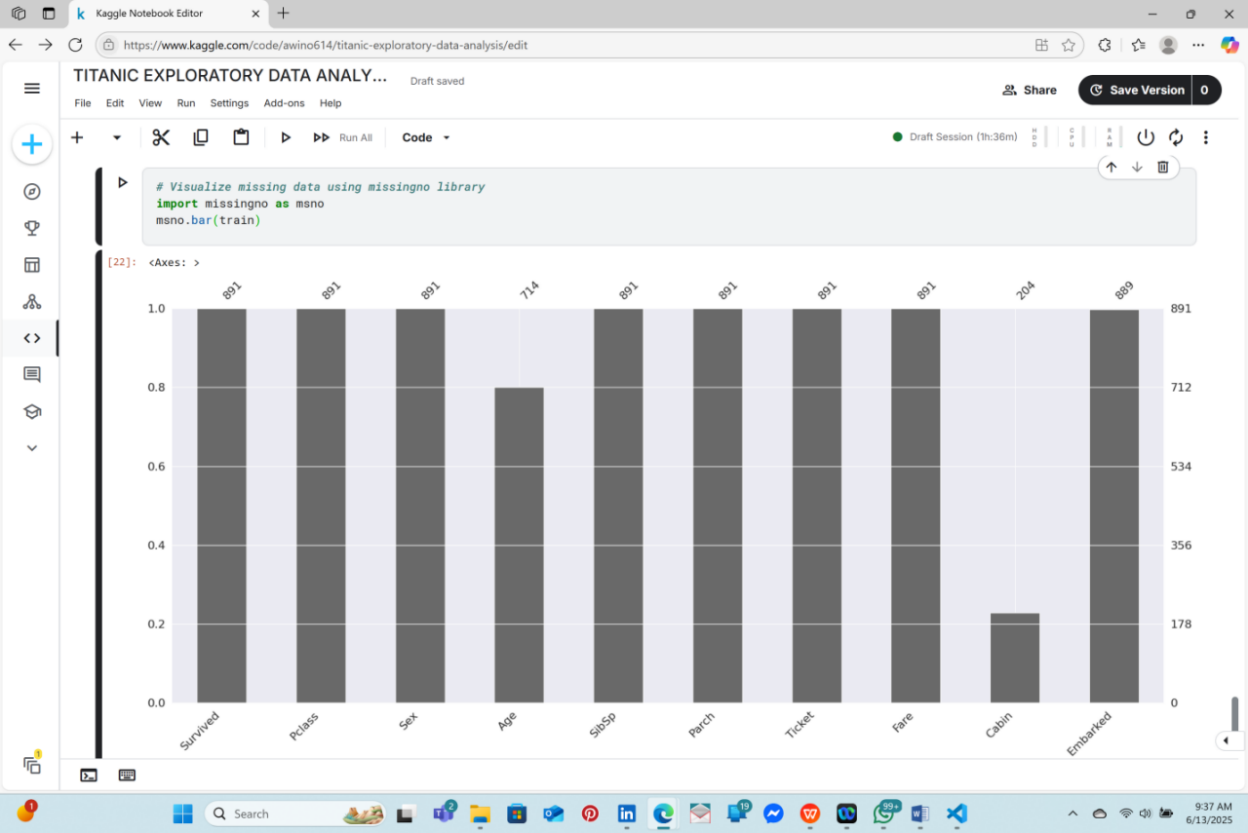
| **Approach** | **Description** | **When to Use** |
| --- | --- | --- |
| **Drop Data** | Remove rows or columns with missing values. | When the missing values are minimal and the affected data is not critical. |
| **Impute Data** | Fill in missing values with an estimate (mean, median, mode, or other methods). | When the column has enough data to reasonably estimate the missing values. |
| **Flag Missing** | Create a new column indicating where values were missing. | When missing values themselves might indicate an important pattern. |

## **🧭 Step 3. 1: Visualizing Missing Data**

Before deciding how to handle missing values, it's helpful to **visualize the missing data**. You can use **msno.bar(df)** which shows the missing values of each column in a bar chart.

### ****What the Plot Shows?****

* Each white line in the plot represents a missing value.
* Columns with many white gaps indicate high percentages of missing data.
* 💡 In the titanic dataset, we can quickly see that there are missing values in cabin and age and few in embarked.

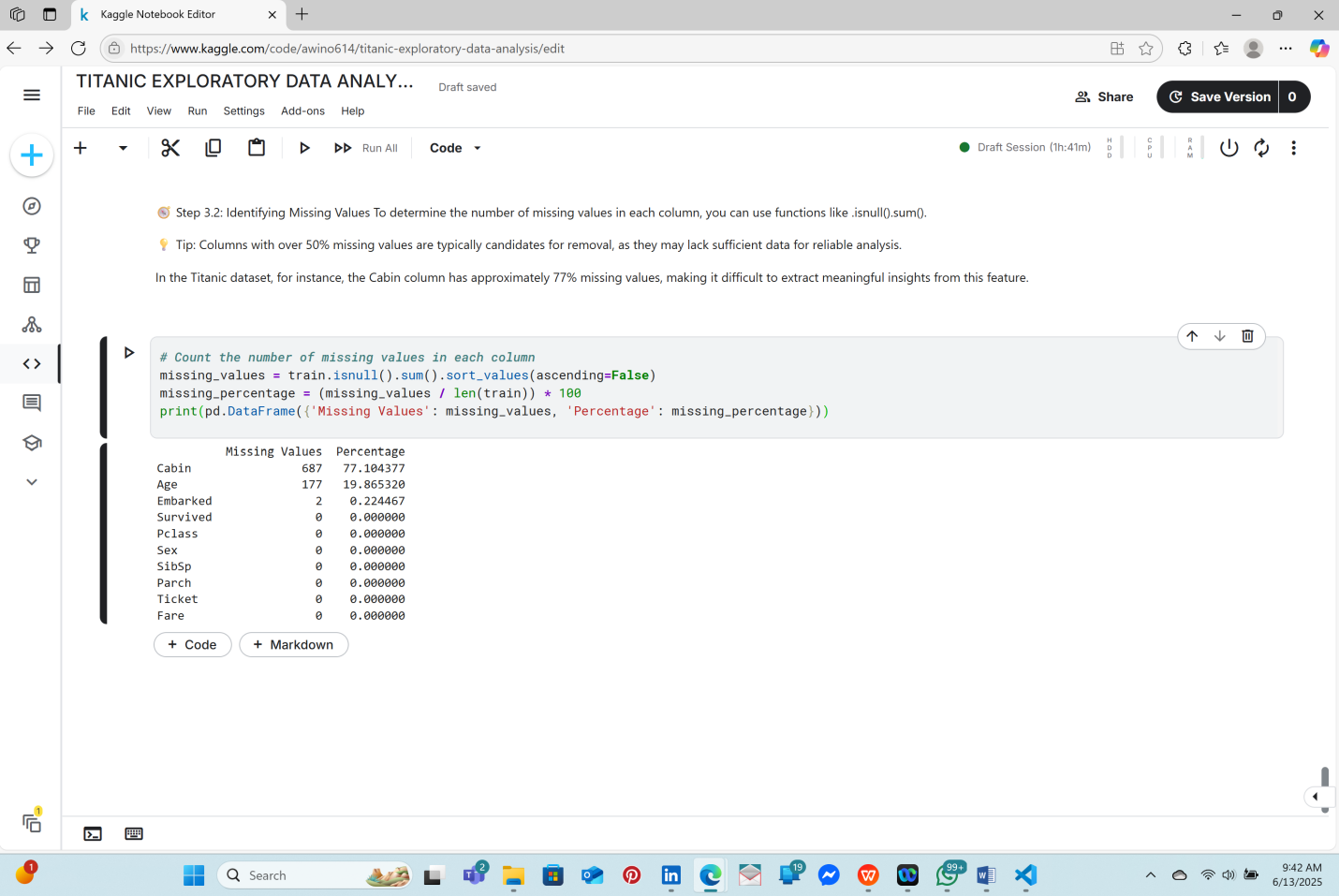


### 🧭 ****Step 3.2: Identifying Missing Values****

To determine the number of missing values in each column, you can use functions like .isnull().sum().

💡 **Tip**: Columns with over **50% missing values** are typically candidates for removal, as they may lack sufficient data for reliable analysis.

In the **Titanic dataset**, for instance, the Cabin column has approximately **77% missing values**, making it difficult to extract meaningful insights from this feature.

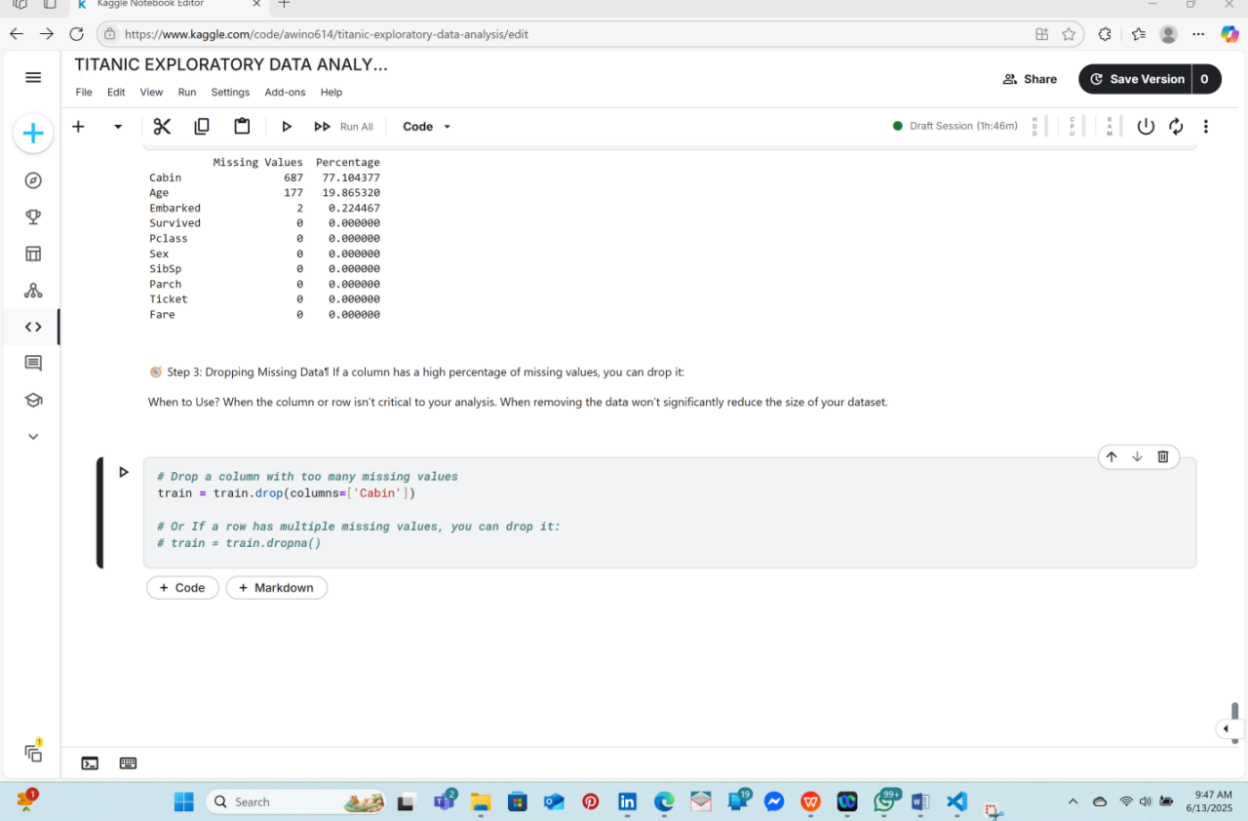


## **🧭 Step 3:3 Dropping Missing Data**[¶](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#%F0%9F%A7%AD-Step-3:-Dropping-Missing-Data)

If a column has a high percentage of missing values, you can drop it:

### ****When to Use?****

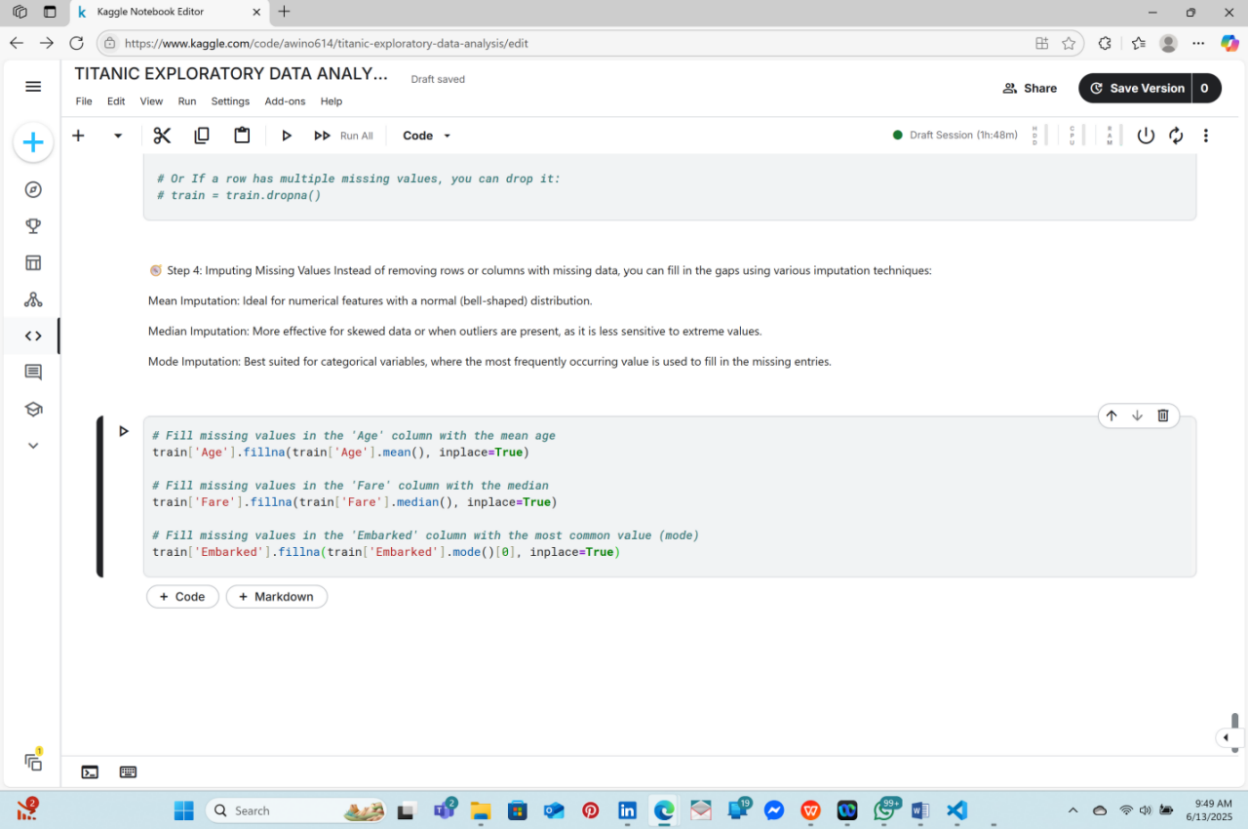
* When the column or row isn’t critical to your analysis.
* When removing the data won’t significantly reduce the size of your dataset.



### 🧭 ****Step 4: Imputing Missing Values****

Instead of removing rows or columns with missing data, you can fill in the gaps using various imputation techniques:

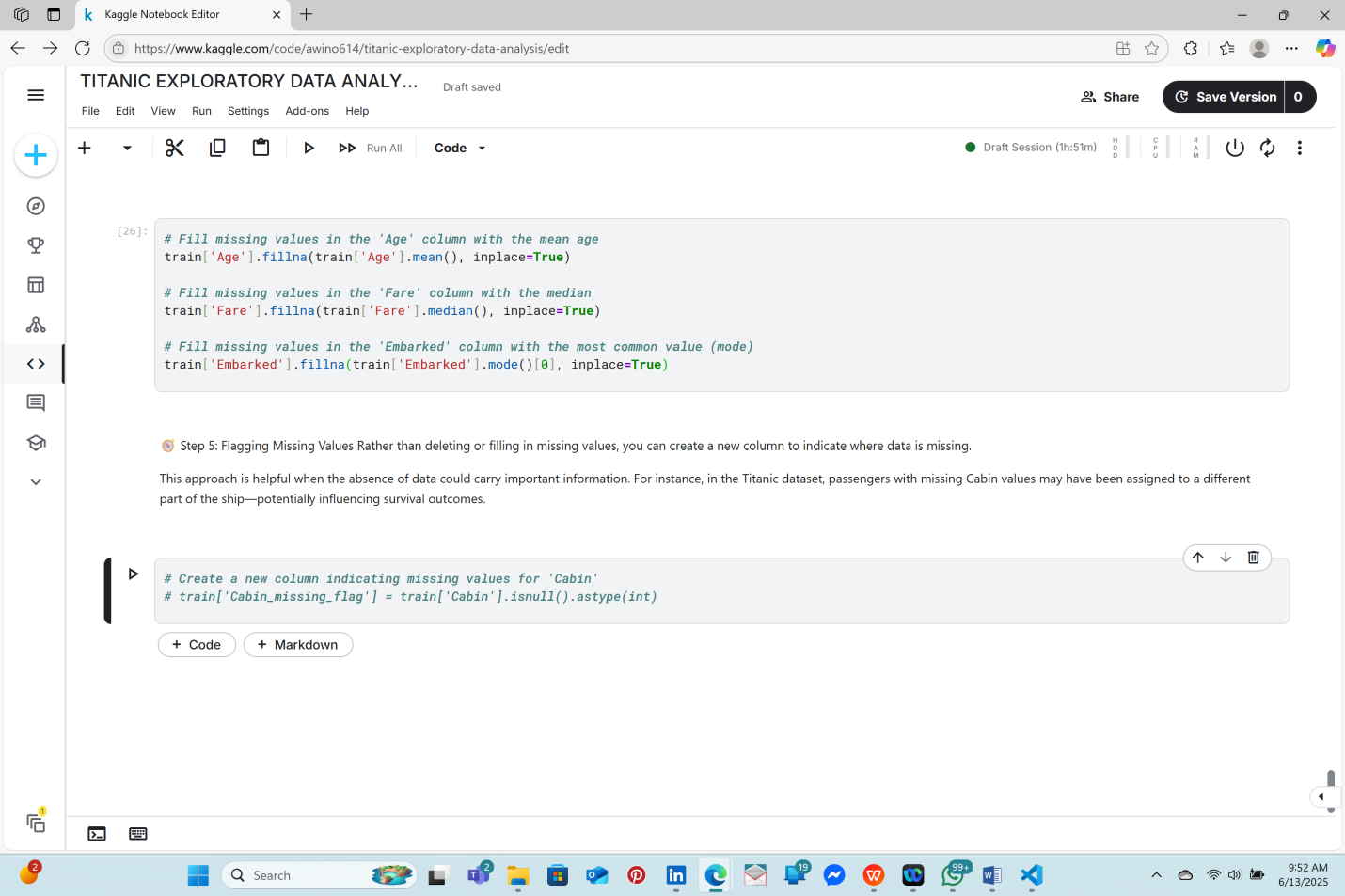
* **Mean Imputation**: Ideal for numerical features with a **normal (bell-shaped) distribution**.
* **Median Imputation**: More effective for **skewed data** or when **outliers** are present, as it is less sensitive to extreme values.
* **Mode Imputation**: Best suited for **categorical variables**, where the most frequently occurring value is used to fill in the missing entries.

****

### 🧭 ****Step 5: Flagging Missing Values****

Rather than deleting or filling in missing values, you can create a **new column** to indicate where data is missing.

This approach is helpful when the **absence of data could carry important information**. For instance, in the Titanic dataset, passengers with missing Cabin values may have been assigned to a different part of the ship—potentially influencing survival outcomes.

****

## **📝 Summary of Missing Values Handling**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#%F0%9F%93%9D-Summary-of-Missing-Values-Handling)

In this section, we have:

* Visualized missing values using missingno.
* Counted the number of missing values in each column.
* Learned how to drop rows and columns with missing data.
* Used different strategies to fill missing values (mean, median, mode).
* Created a flag column to indicate missing data instead of dropping or imputing it.

### **STEP 2: UNIVARIATE ANALYSIS**

#### 🎯 **What Is Univariate Analysis?**

Univariate analysis focuses on examining a **single variable** at a time to understand its characteristics, including:

* **Distribution** (e.g., normal, skewed)
* **Central tendency** (mean, median, mode)
* **Spread or variability** (range, variance, standard deviation)

This type of analysis helps answer questions such as:

* What is the age distribution of passengers?
* How are passengers distributed across embarkation points?
* Are ticket prices evenly distributed or skewed?

### 🧭 ****Step 2.1: Analyzing Numerical Features****

When working with numerical columns, the goal is to explore each one individually to understand its shape and patterns.

#### **Common Plots for Numerical Data:**

1. **Histogram** – Shows how frequently values occur across intervals.
2. **KDE Plot (Kernel Density Estimate)** – A smooth curve that estimates the probability distribution of the data.
3. **Boxplot** – Displays the range, median, and identifies outliers.

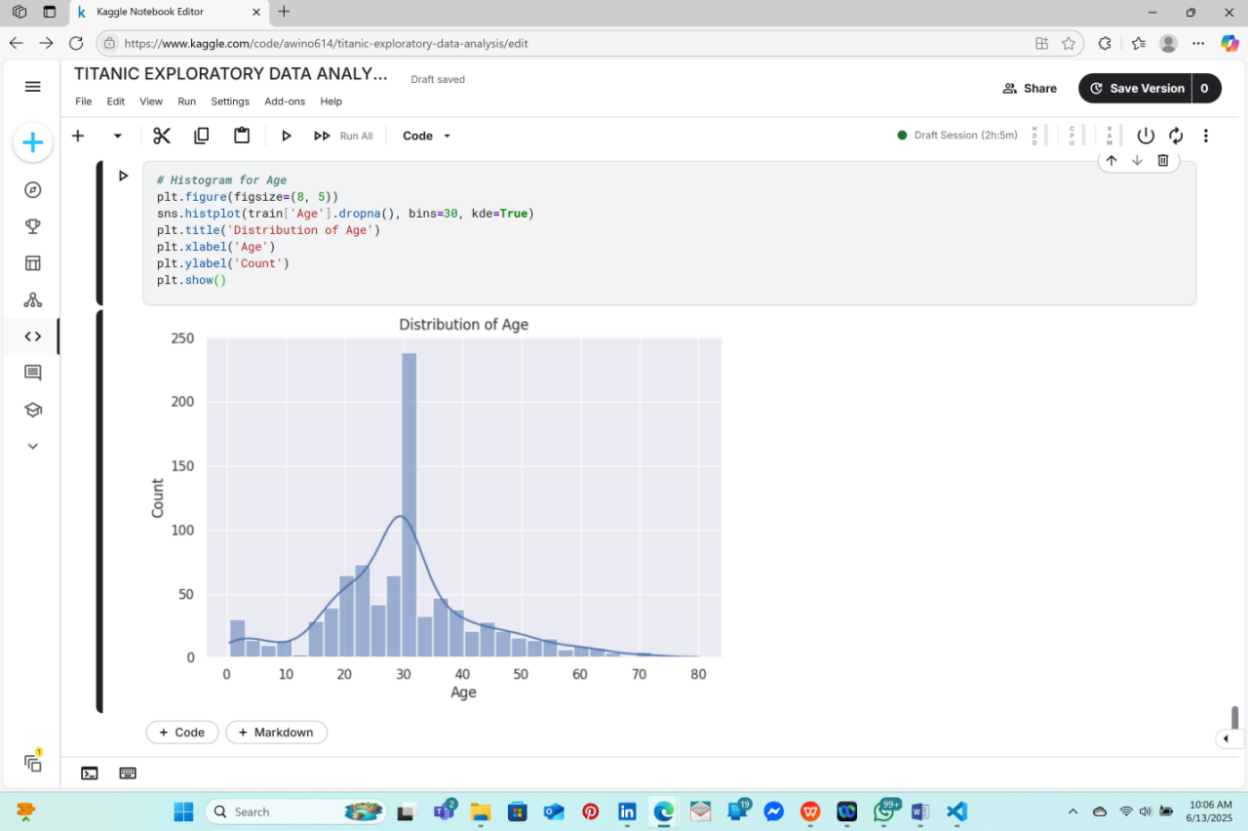
#### **Using Histograms**

Histograms are ideal for visualizing the distribution of numeric variables such as Age, Fare, etc.

**What to Look For:**

* **Peaks** indicate the most common value ranges (e.g., age groups).
* **Gaps** suggest missing or infrequent values.
* **Skewness** helps determine whether the data is symmetrical or biased to one side.

💡 **Tip**: You can enhance a histogram by adding a density curve using sns.histplot(data, kde=True) to better visualize the distribution.



### ****2. KDE Plot (Smoothed Distribution)****

The KDE plot shows the **probability density function** of a numerical column:

### ****When to Use KDE Plots?:****

* When you want to visualize the shape of the data distribution more smoothly than a histogram.
* Great for detecting skewness and multimodal distributions (multiple peaks).



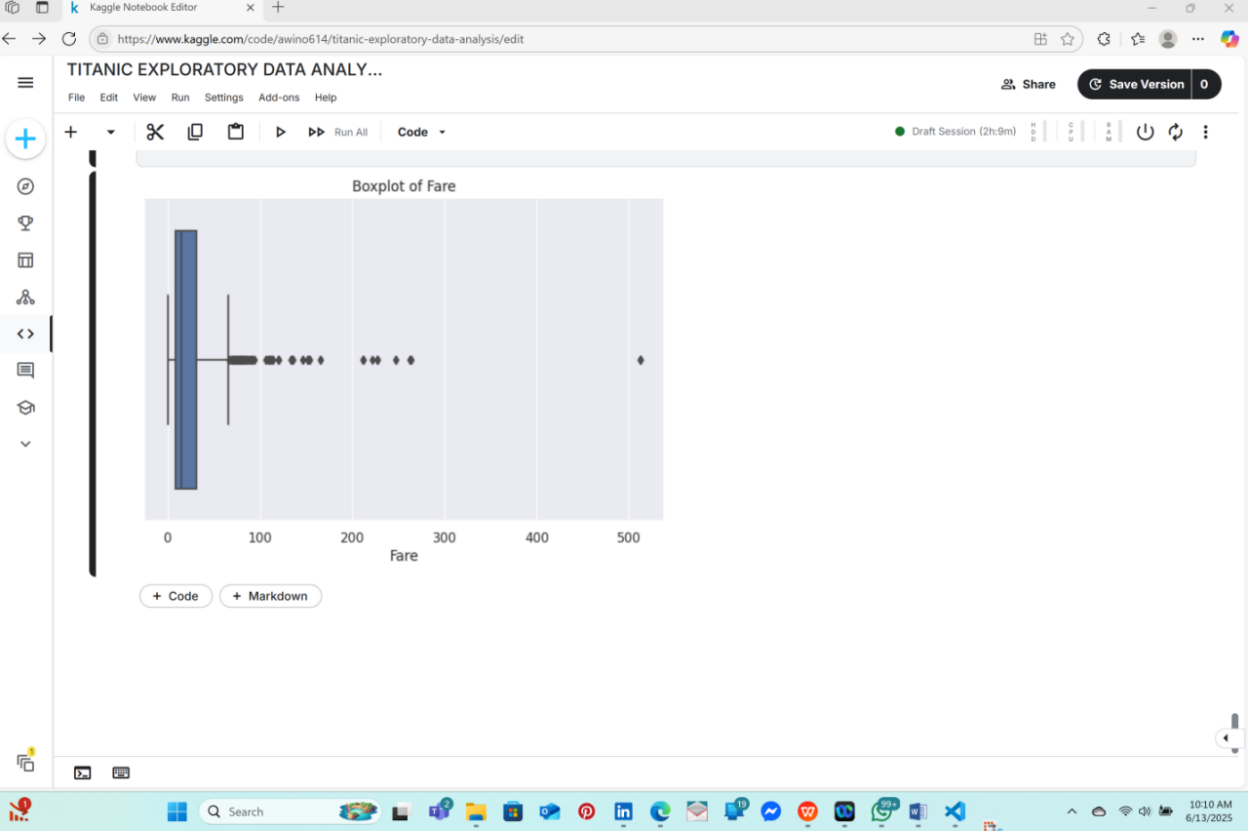
### ****3. Boxplot (Detecting Outliers)****

Boxplots visualize the minimum, lower quartile (25%), median, upper quartile (75%), and maximum values:

### ****Interpretation:****

* **Line in the middle:** Median (50% value).
* **Box edges:** 25th percentile (Q1) and 75th percentile (Q3).
* **Whiskers:** Minimum and maximum values (excluding outliers).
* **Dots outside the whiskers:** Outliers.

💡 Tip: If the boxplot has a long "tail" or many outliers, the data might be skewed.



## **🧭 Step 2: Univariate Analysis for Categorical Columns**

There are common visualizations used when we are dealing with categorical columns:

* **Countplot:** Shows the frequency count of each category.
* **Pie Chart:** Shows proportions of categories in a pie format (less commonly used in data science).

linkcode

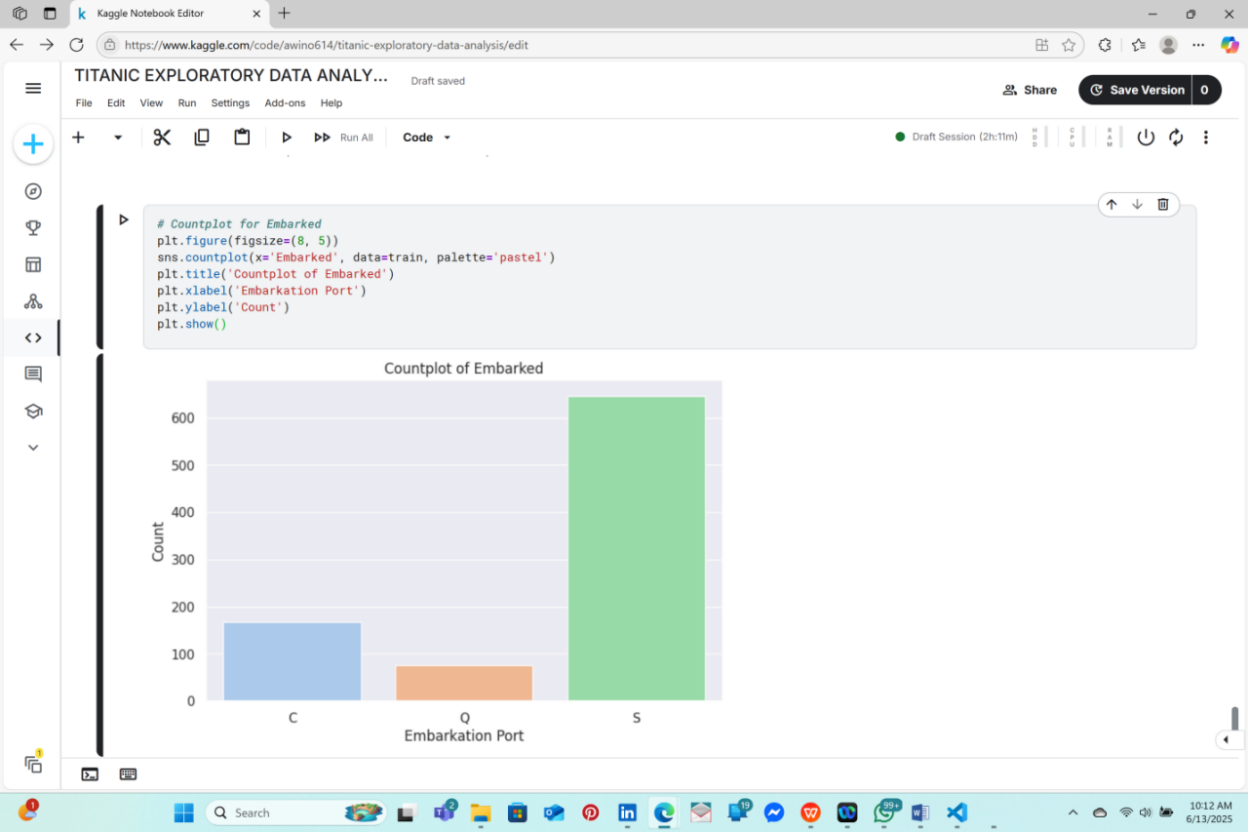
### ****1. Countplot****

Countplots are used to count the frequency of each category in a column:

### ****Interpretation:****

* **Bars represent categories:** Higher bars indicate more frequent categories.
* **Detect class imbalance** (e.g., if most passengers embarked from "S", your data is imbalanced).

💡 Tip: You can add hue='Survived' to compare survival rates across embarkation points.



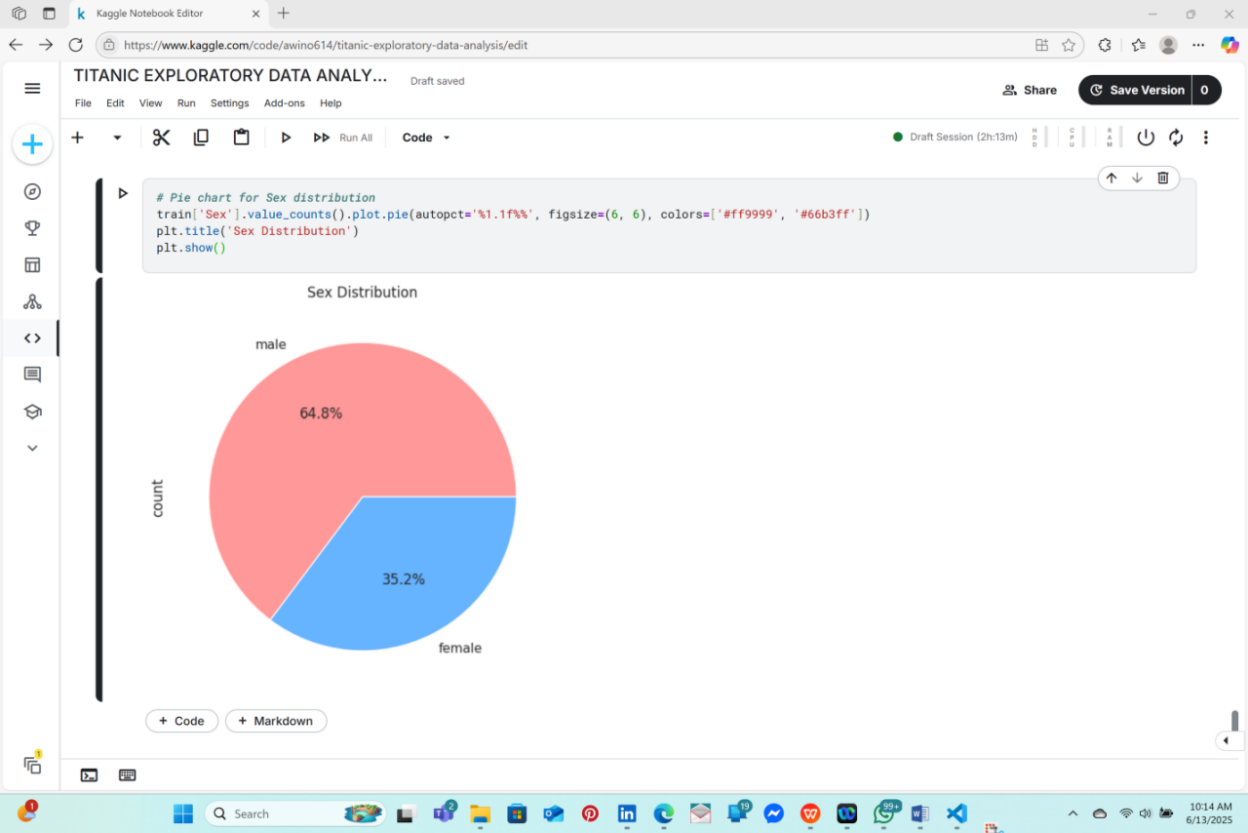
### ****2. Pie Chart****

While pie charts are visually appealing, they’re generally less informative than bar charts for categorical data.

### ****Why Use a Pie Chart?****

* Useful for displaying proportions (e.g., percentage of males vs. females).
* Avoid using them when you have more than 3-4 categories.

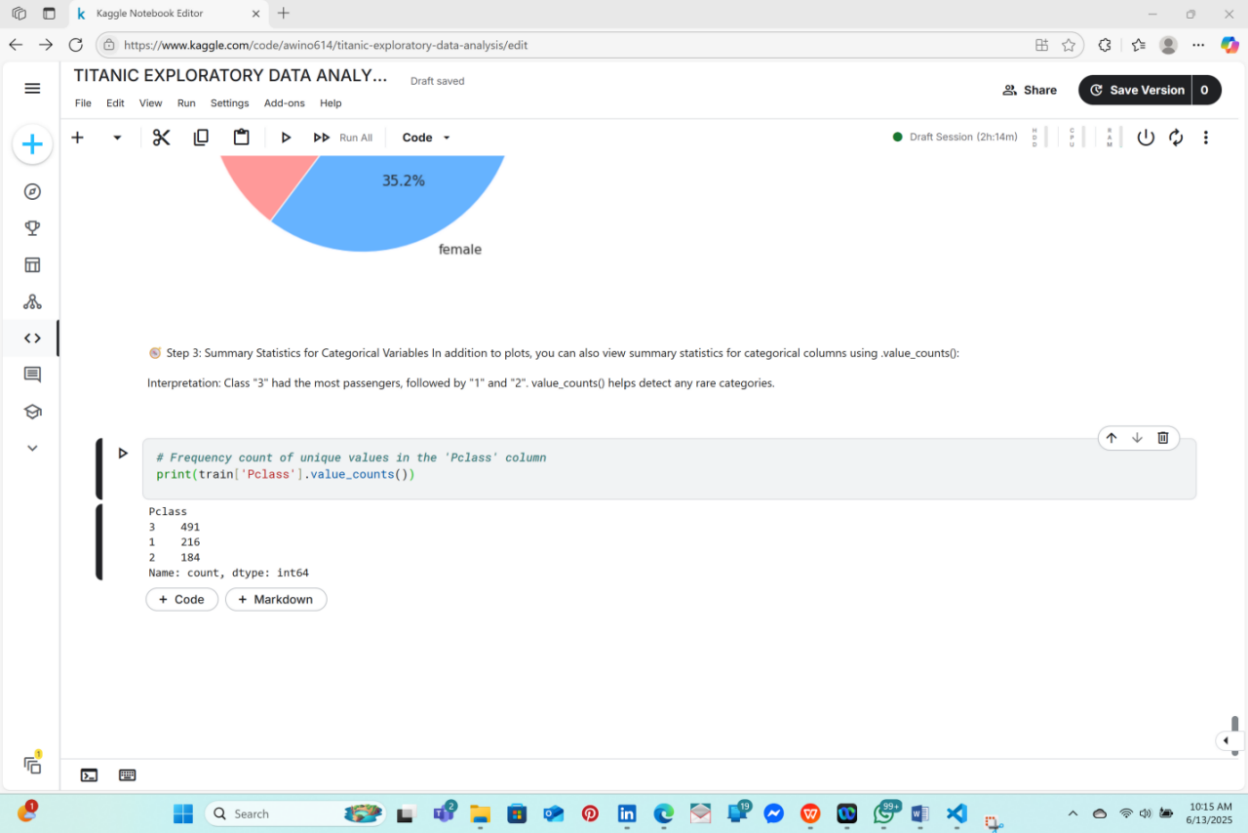
💡 Tip: You can add hue='Survived' to compare survival rates across embarkation points.



## **🧭 Step 3: Summary Statistics for Categorical Variables**

In addition to plots, you can also view summary statistics for categorical columns using **.value\_counts()**:

### ****Interpretation:****

* Class "3" had the most passengers, followed by "1" and "2".
* value\_counts() helps detect any rare categories.

## **🚩 Common Insights During Univariate Analysis**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#%F0%9F%9A%A9-Common-Insights-During-Univariate-Analysis)

**Numerical Columns:**

* Look for outliers (e.g., extremely high Fare values).
* Identify skewness (e.g., Fare might be right-skewed).

**Categorical Columns:**

* Detect imbalances (e.g., most passengers embarked from port "S").
* Spot categories with low frequencies (e.g., rare ticket classes).

linkcode

## **📝 Summary of Univariate Analysis**

In this section, we have:

* Used histograms, KDE plots, and boxplots for numerical columns to understand distributions and outliers.
* Used countplots and pie charts to explore categorical columns.
* Used .value\_counts() to summarize the frequency of categories.

# **. Bivariate Analysis**[**¶**](https://www.kaggle.com/code/mariyamalshatta/masterclass-1-a-comprehensive-guide-for-eda#5.-Bivariate-Analysis)