

Data and Artificial Intelligence

Cyber Shujaa Program

Week 3 Assignment

Exploratory Data Analysis

Student Name: Faith Jeptoo

Student ID: CS-DA02-25005

Table of Contents

1. **Cover Page**
2. **Introduction**
3. **Importing Libraries and Loading the Dataset**
4. **Initial Data Exploration**
 - 4.1. Dataset Overview
 - 4.2. Feature Profiling
5. **Checking for Missing Values and Duplicates**
6. **Univariate Analysis**
 - 6.1. Age Distribution
 - 6.2. Fare Distribution
 - 6.3. Passenger Class, Gender, and Embarked
7. **Bivariate Analysis**
 - 7.1. Fare vs Passenger Class
 - 7.2. Survival by Gender and Embarked
 - 7.3. Age and Survival
8. **Multivariate Analysis**
 - 8.1. Class, Gender, and Survival
 - 8.2. Age, Fare, and Survival
9. **Outlier Detection and Handling**
 - 9.1. Age Outliers
 - 9.2. Fare Outliers
10. **Target Variable (Survived) Analysis**
11. **Conclusion**
12. **Notebook Link**

Introduction

This assignment focuses on performing **Exploratory Data Analysis (EDA)** on the famous **Titanic dataset** from Kaggle.

The goal is to gain hands-on experience in exploring, cleaning, and visualizing data to uncover meaningful patterns and insights.

EDA involves:

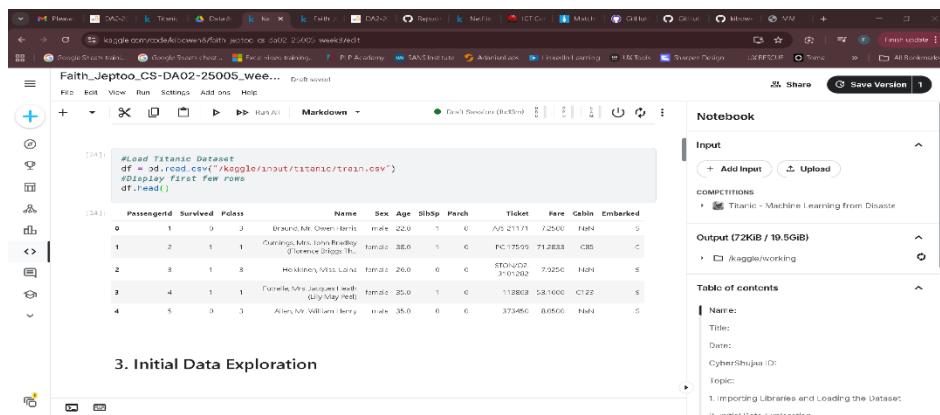
- Understanding the structure and quality of the data
- Handling missing values and outliers
- Performing univariate, bivariate, and multivariate analyses
- Exploring the target variable (**Survived**) to identify influencing factors

Tasks Completed

Step 1: Importing Libraries and Loading the Dataset

```
import pandas as pd      # Data manipulation and analysis
import numpy as np       # Numerical computations
import matplotlib.pyplot as plt # Data visualization
import seaborn as sns    # Advanced plots
sns.set(style="whitegrid")

# Load the Titanic dataset
df = pd.read_csv("/kaggle/input/titanic/train.csv")# Display first few rows
df.head()
```



The screenshot shows a Jupyter Notebook interface with the following content:

```
# Load the Titanic dataset
df = pd.read_csv("/kaggle/input/titanic/train.csv")
# Display first few rows
df.head()
```

Below the code cell, the resulting DataFrame is displayed as a table:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Brundt, Mr. Green Berth	male	22.0	1	0	AVG21173	72.000	NAN	S
1	2	1	1	O'Grady, Mrs. John Bradley	female	38.0	1	0	PC17593	71.2833	C85	C
2	3	1	3	Hockley, Miss. Linda	female	26.0	0	0	STON/O23	7.9250	NAN	S
3	4	1	1	Forde, Mrs. Jacques Heath (Daisy May Forde)	female	35.0	1	0	113803	53.1000	C122	S
4	5	0	3	Allen, Mr. William Henry	male	38.0	0	0	373460	8.0500	NAN	S

Below the table, a section titled "3. Initial Data Exploration" is visible.

Step 2: Initial Data Exploration

#Shapes of the Dataset (rows,columns)

```
print("Dataset Shape:", df.shape)
```

#Column names

```
print("\ncolumns:\n", df.columns)
```

#Info: data types + non-null values

```
print("\nDataset info:")
df.info()
```

Column	Data Type	Non-Null Count
PassengerId	object	891
Survived	object	891
Pclass	int64	891
Name	object	891
Sex	object	891
Age	float64	891
SibSp	int64	891
Parch	int64	891
Ticket	object	891
Fare	float64	891
Cabin	object	891
Embarked	object	891

#summary statistics for numeric and categorical columns

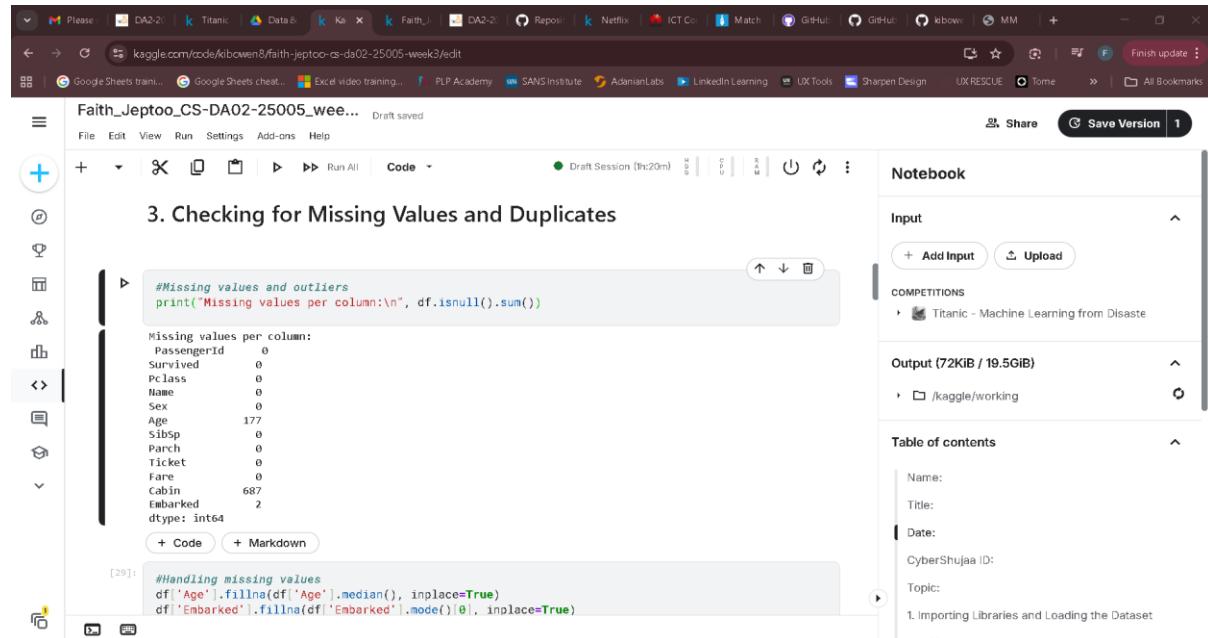
```
df.describe(include="all")
```

Count	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891.000000	891.000000	891.000000	891	891.000000	204	891
unique	Nan	Nan	Nan	2	Nan	Nan	Nan	891	Nan	147	3
top	Nan	Nan	Nan	Male	Nan	Nan	Nan	347682	Nan	898	Nan
freq	Nan	Nan	Nan	1	577	Nan	Nan	7	Nan	4	644
mean	446.000000	0.388088	2.308642	Nan	29.699118	0.528008	0.381594	Nan	32.204208	Nan	Nan
std	25.7318942	0.4866599	0.8386071	Nan	Nan	145.26497	1.107483	0.806057	Nan	49.693429	Nan
min	1.000000	0.000000	1.000000	Nan	Nan	0.420000	0.000000	Nan	0.000000	Nan	Nan

Step 3: Checking for Missing Values and Duplicates

#Missing values and outliers

```
print("Missing values per column:\n", df.isnull().sum())
```



The screenshot shows a Jupyter Notebook interface with the following code in a cell:

```
#Missing values and outliers
print("Missing values per column:\n", df.isnull().sum())
```

Output:

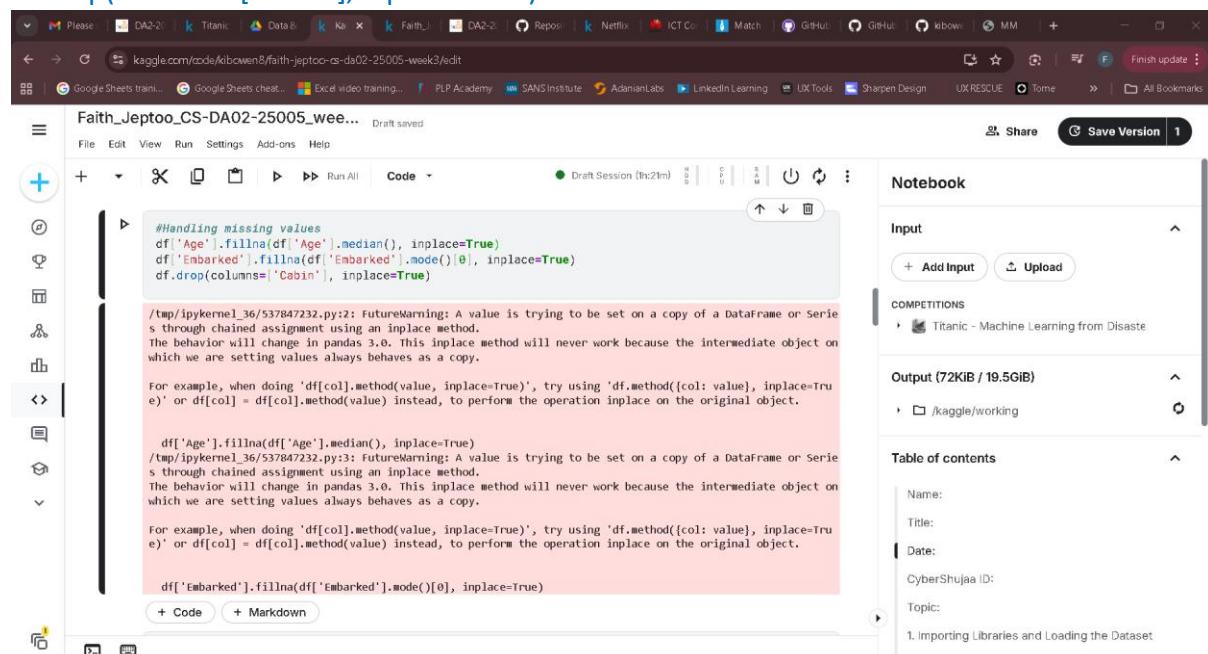
```
Missing values per column:
PassengerId      0
survived         0
pclass           0
Name             0
Sex              0
Age            177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin          687
Embarked        2
dtype: int64
```

Below the code cell, there is another cell with the following code:

```
[20]: #Handling missing values
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

#Handling missing values

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df.drop(columns=['Cabin'], inplace=True)
```



The screenshot shows a Jupyter Notebook interface with the following code in a cell:

```
#Handling missing values
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df.drop(columns=['Cabin'], inplace=True)
```

Output:

```
/tmp/ipykernel_36/537847232.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method([col: value], inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

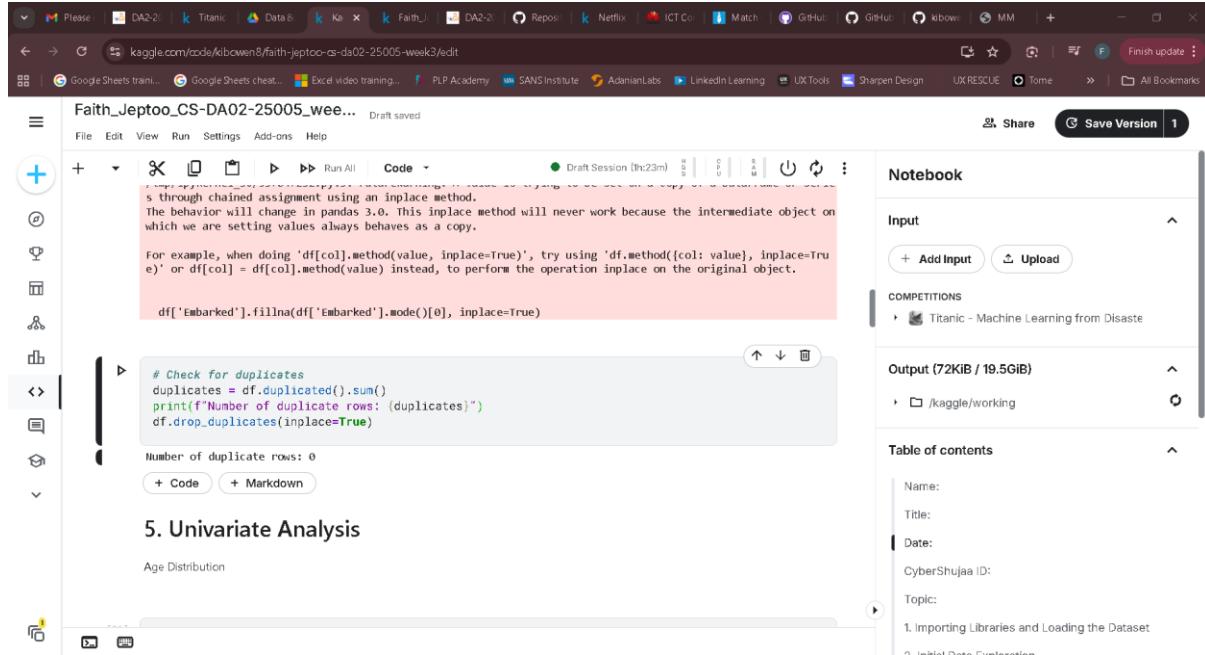
df['Age'].fillna(df['Age'].median(), inplace=True)
/tmp/ipykernel_36/537847232.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method([col: value], inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.
```

Below the code cell, there is another cell with the following code:

```
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

```
# Check for duplicates
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
df.drop_duplicates(inplace=True)
```



The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Faith_Jeptoo_CS-DA02-25005_week3
- Code Cell Content:**

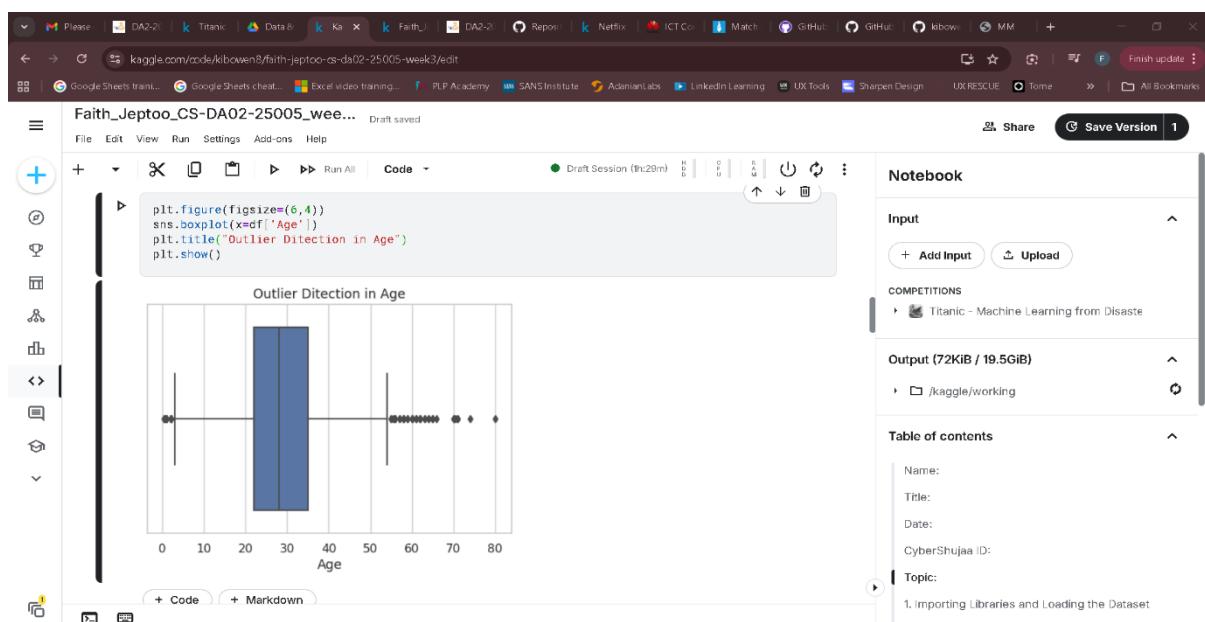
```
# Check for duplicates
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
df.drop_duplicates(inplace=True)
```
- Output:**

Number of duplicate rows: 0
- Section Header:** 5. Univariate Analysis
- Figure:** Age Distribution
- Right Panel:**
 - Notebook:** Shows the current notebook structure.
 - Input:** Buttons for Add Input and Upload.
 - Competitions:** Shows the competition "Titanic - Machine Learning from Disaster".
 - Output:** Shows the output size (72KiB / 19.5GiB) and the path /kaggle/working.
 - Table of contents:** Fields for Name, Title, Date, CyberShujaa ID, and Topic, with entries for Step 4: Univariate Analysis, 1. Importing Libraries and Loading the Dataset, and 2. Initial Data Exploration.

Step 4: Univariate Analysis

Age Distribution

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df['Age'])
plt.title("Outlier Detection in Age")
plt.show()
```



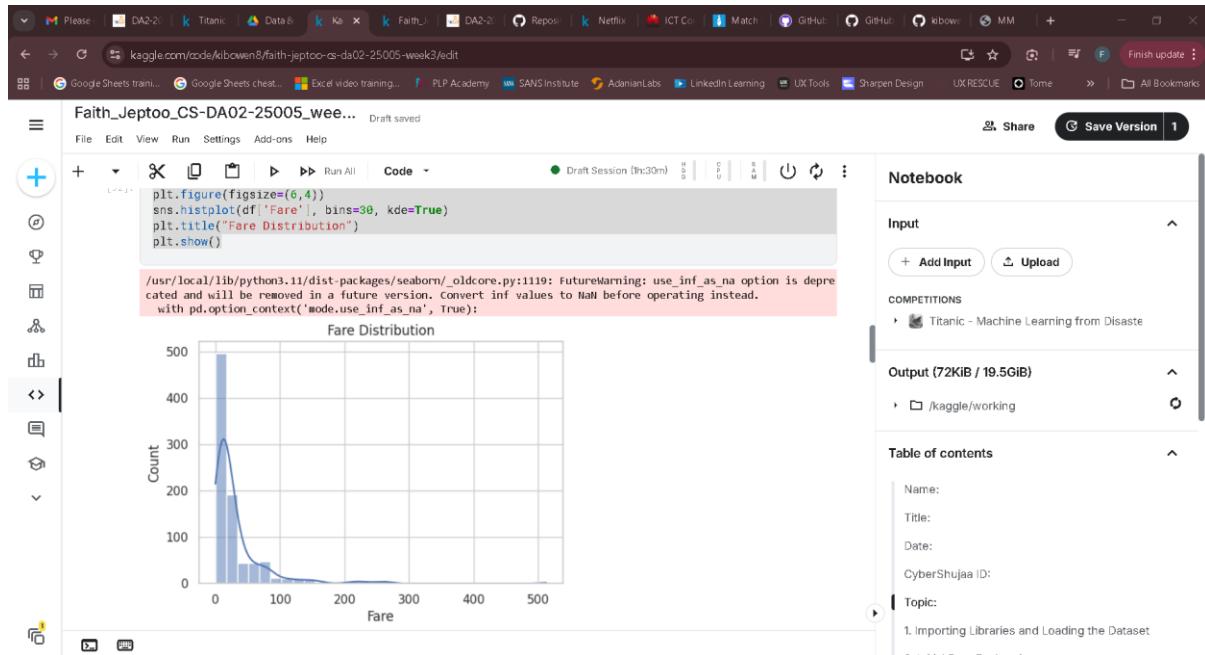
The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Faith_Jeptoo_CS-DA02-25005_week3
- Code Cell Content:**

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df['Age'])
plt.title("Outlier Detection in Age")
plt.show()
```
- Figure:** A boxplot titled "Outlier Detection in Age" showing the distribution of age. The x-axis ranges from 0 to 80. The plot shows a dense cluster of data points between ages 20 and 60, with several outliers extending beyond the whiskers at both ends.
- Right Panel:**
 - Notebook:** Shows the current notebook structure.
 - Input:** Buttons for Add Input and Upload.
 - Competitions:** Shows the competition "Titanic - Machine Learning from Disaster".
 - Output:** Shows the output size (72KiB / 19.5GiB) and the path /kaggle/working.
 - Table of contents:** Fields for Name, Title, Date, CyberShujaa ID, and Topic, with entries for Step 4: Univariate Analysis, 1. Importing Libraries and Loading the Dataset, and 2. Initial Data Exploration.

Fare Distribution

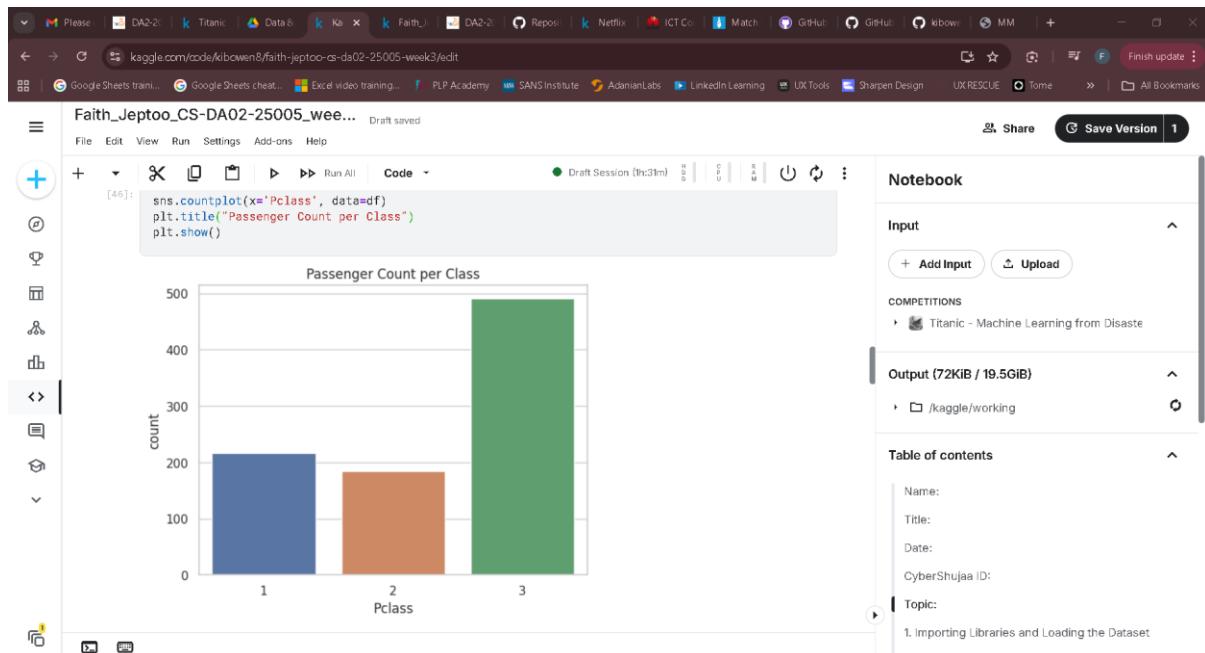
```
plt.figure(figsize=(6,4))
sns.histplot(df['Fare'], bins=30, kde=True)
plt.title("Fare Distribution")
plt.show()
```



Passenger Class, Gender, and Embarked

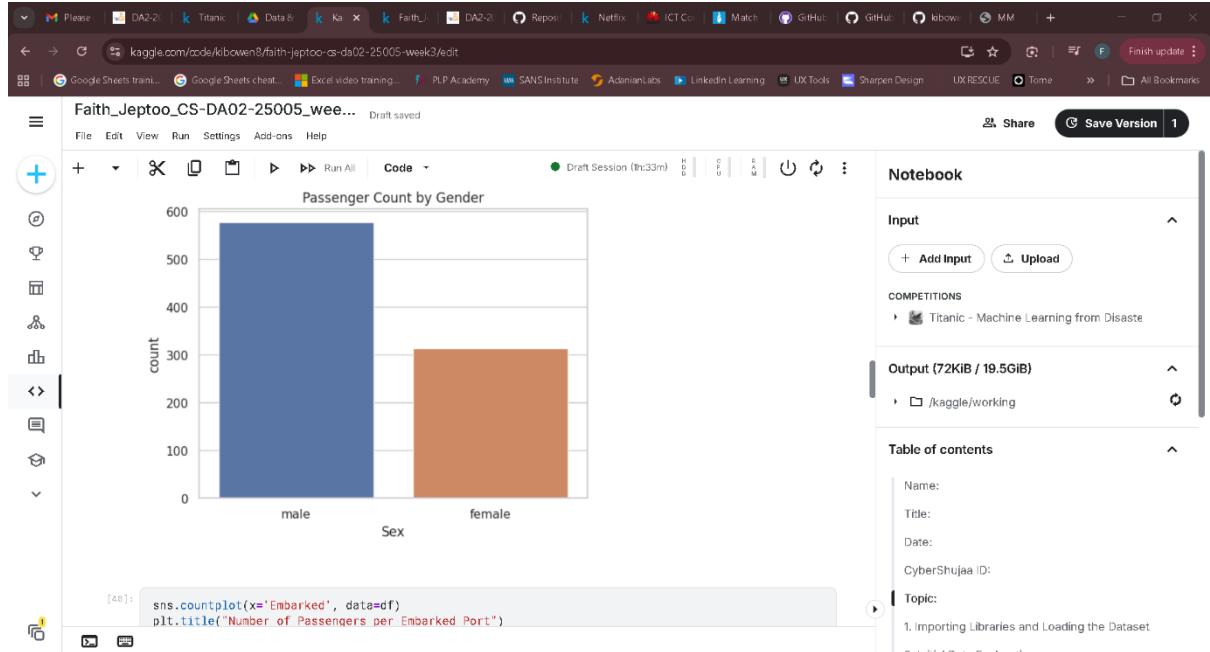
Class

```
sns.countplot(x='Pclass', data=df)
plt.title("Passenger Count per Class")
plt.show()
```



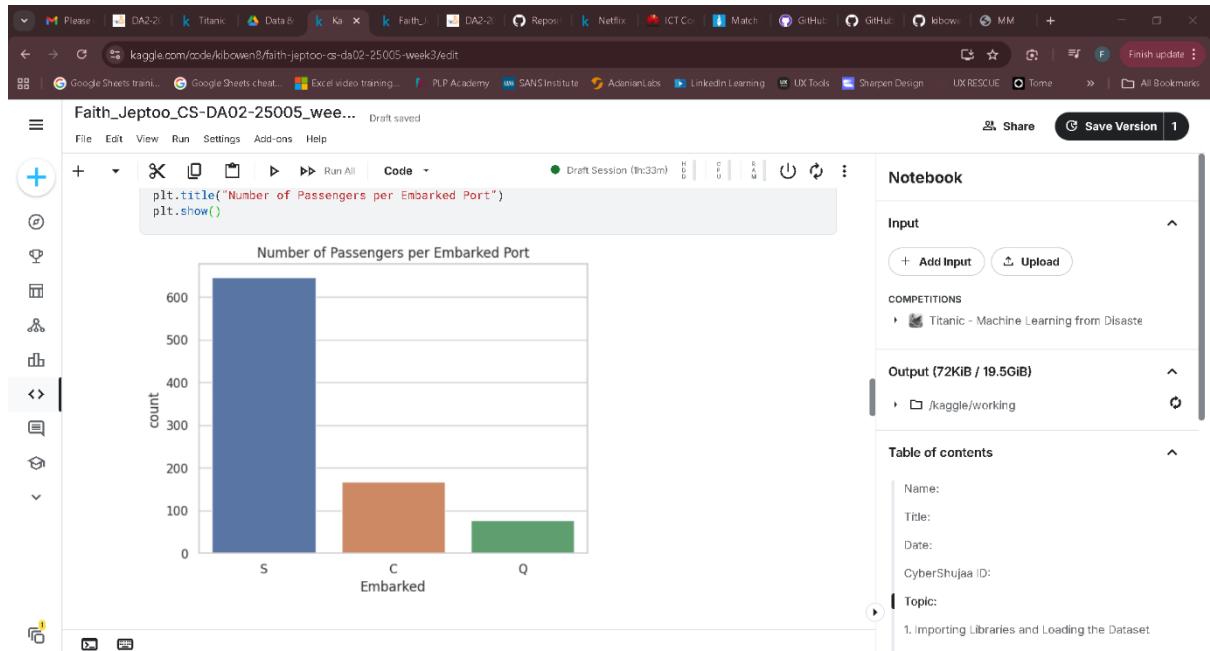
Gender

```
sns.countplot(x='Sex', data=df)
plt.title("Passenger Count by Gender")
plt.show()
```



Embarked

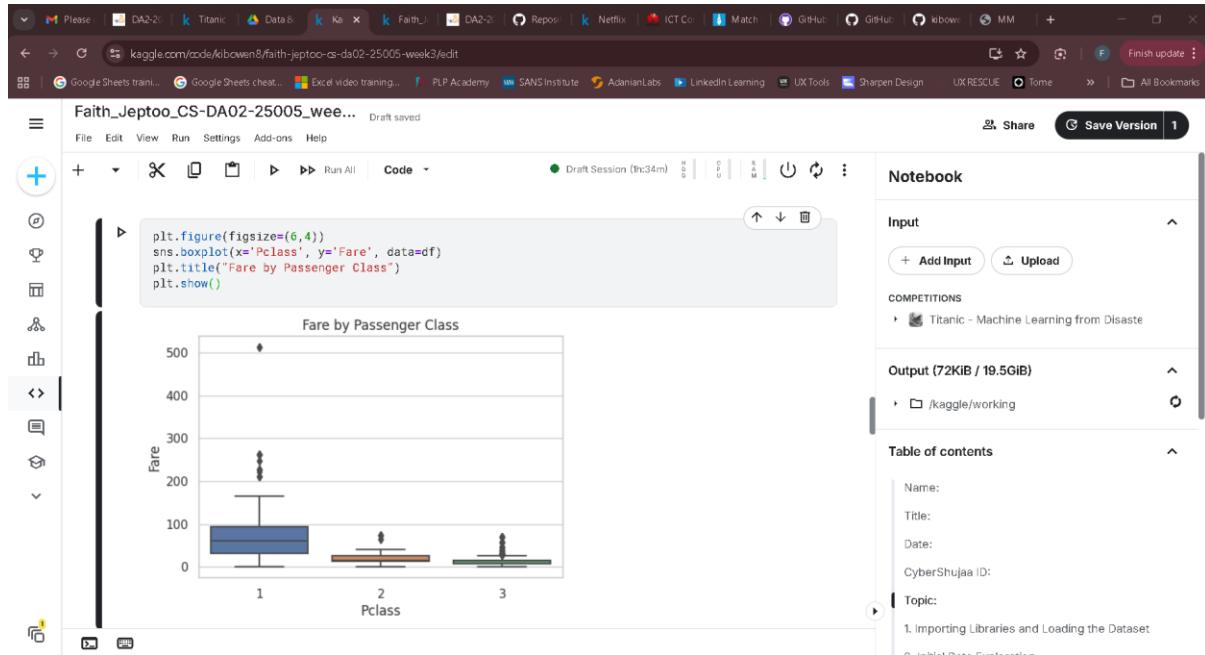
```
sns.countplot(x='Embarked', data=df)
plt.title("Number of Passengers per Embarked Port")
plt.show()
```



Step 5: Bivariate Analysis

Fare vs Pclass

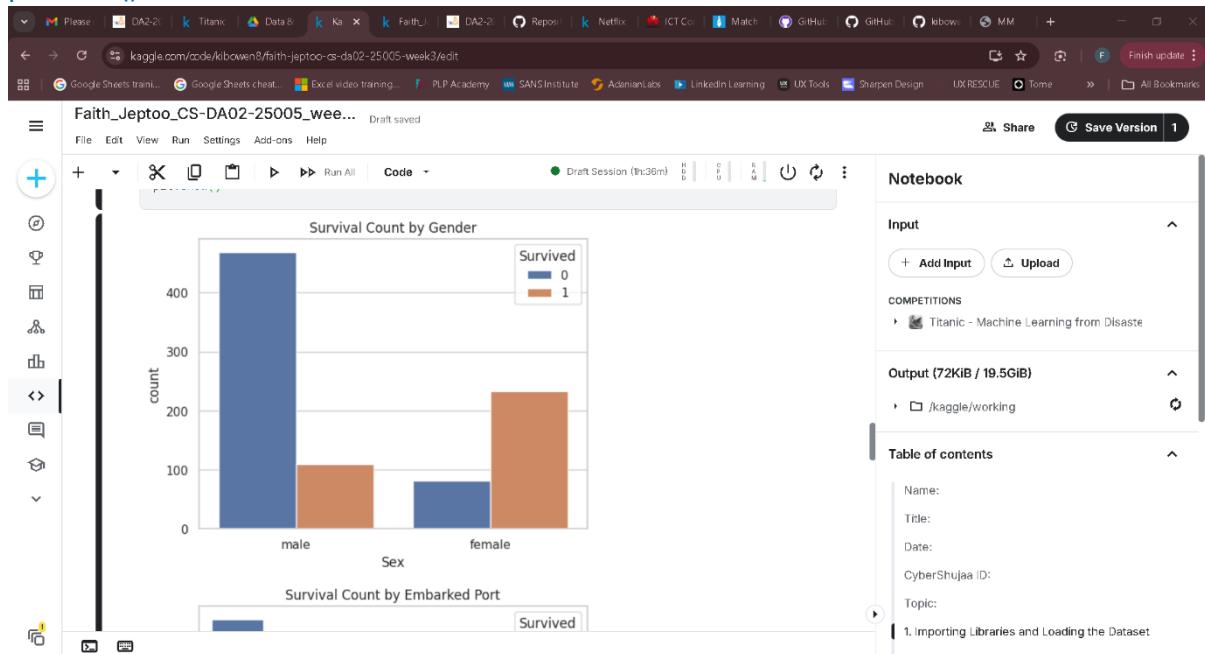
```
plt.figure(figsize=(6,4))
sns.boxplot(x='Pclass', y='Fare', data=df)
plt.title("Fare by Passenger Class")
plt.show()
```



Survival by Gender and Embarked

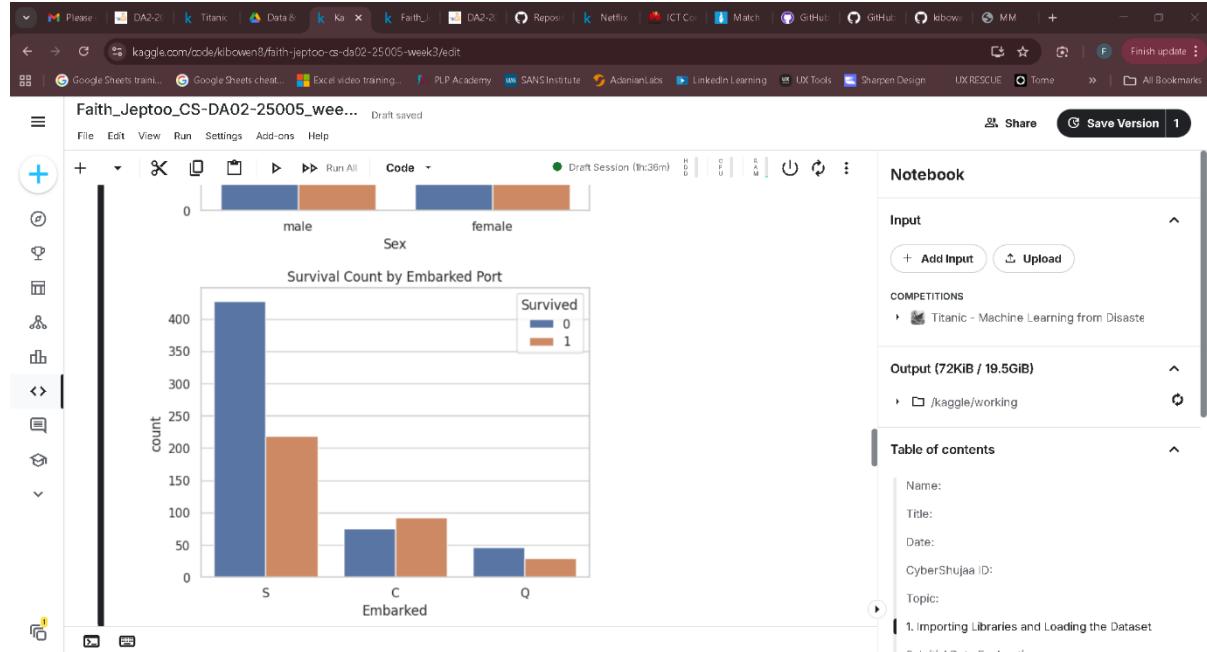
Gender

```
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title("Survival Count by Gender")
plt.show()
```



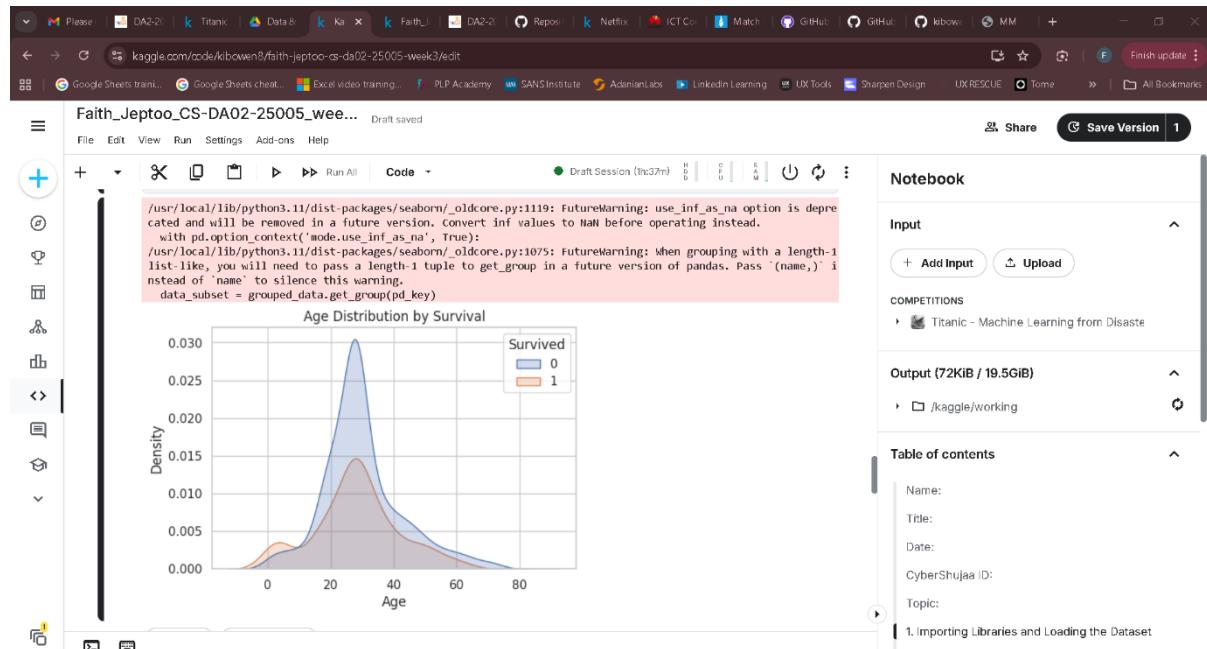
Emberked

```
sns.countplot(x='Embarked', hue='Survived', data=df)
plt.title("Survival Count by Embarked Port")
plt.show()
```



Age and Survival

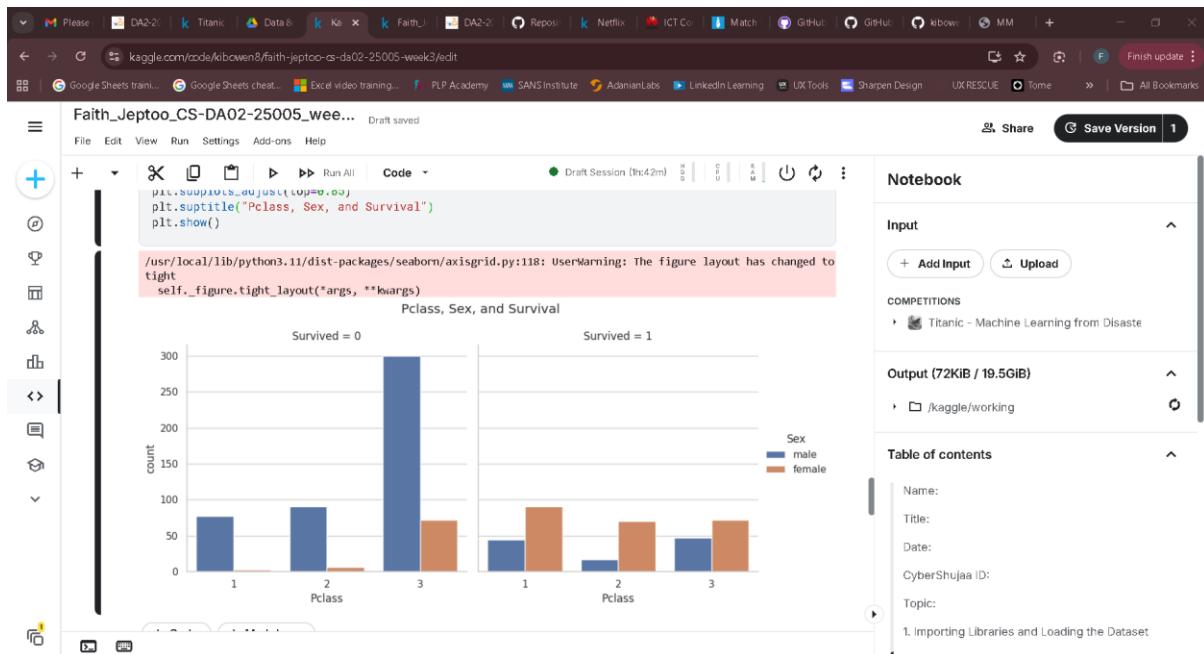
```
plt.figure(figsize=(6,4))
sns.kdeplot(data=df, x='Age', hue='Survived', fill=True)
plt.title("Age Distribution by Survival")
plt.show()
```



Step 6: Multivariate Analysis

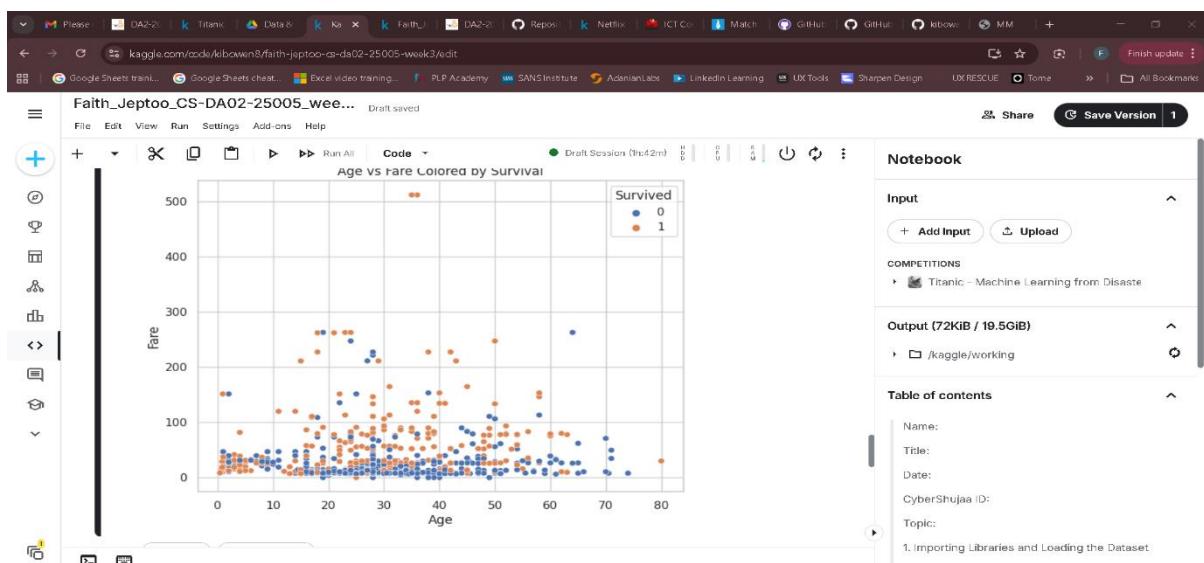
Class, Gender, and Survival

```
sns.catplot(x='Pclass', hue='Sex', col='Survived', data=df, kind='count')
plt.subplots_adjust(top=0.85)
plt.suptitle("Pclass, Sex, and Survival")
plt.show()
```



Age, Fare, and Survival

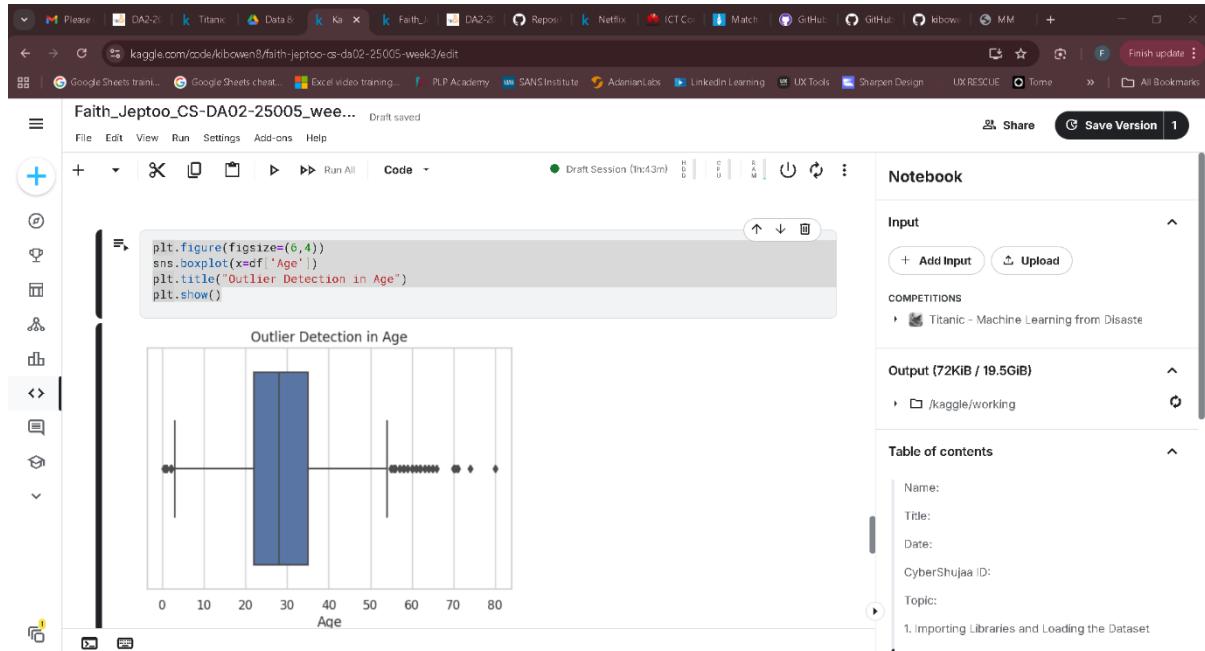
```
plt.figure(figsize=(8,6))
sns.scatterplot(x='Age', y='Fare', hue='Survived', data=df)
plt.title("Age vs Fare Colored by Survival")
plt.show()
```



Step 7: Outlier Detection and Handling

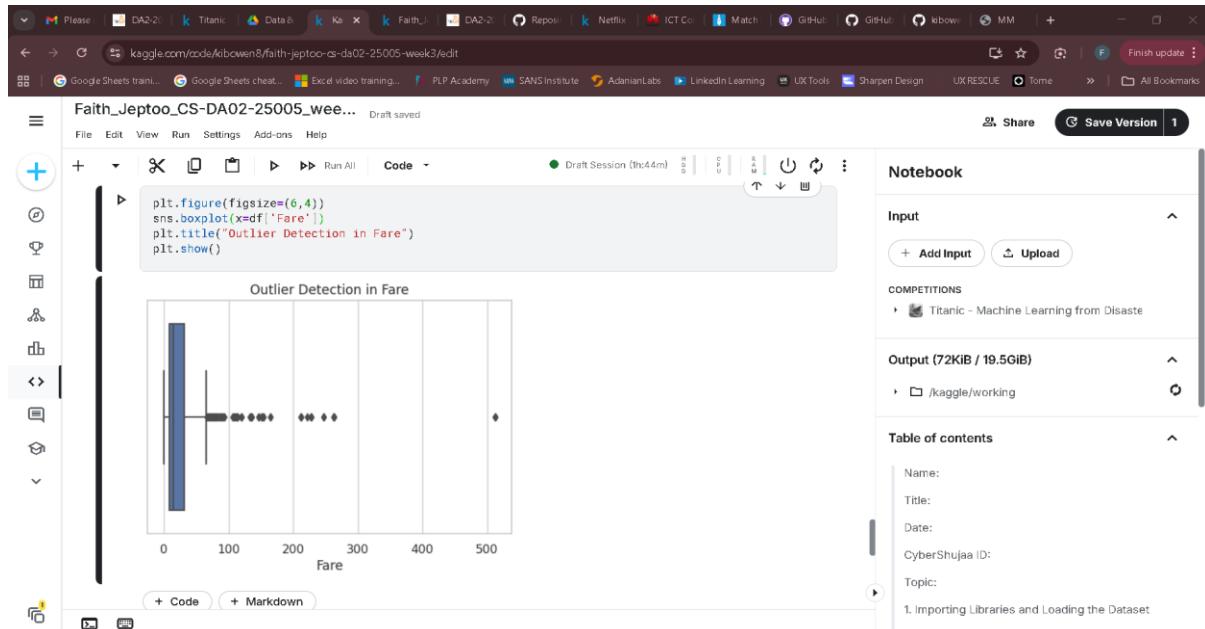
Age Outliers

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df['Age'])
plt.title("Outlier Detection in Age")
plt.show()
```



Fare Outliers

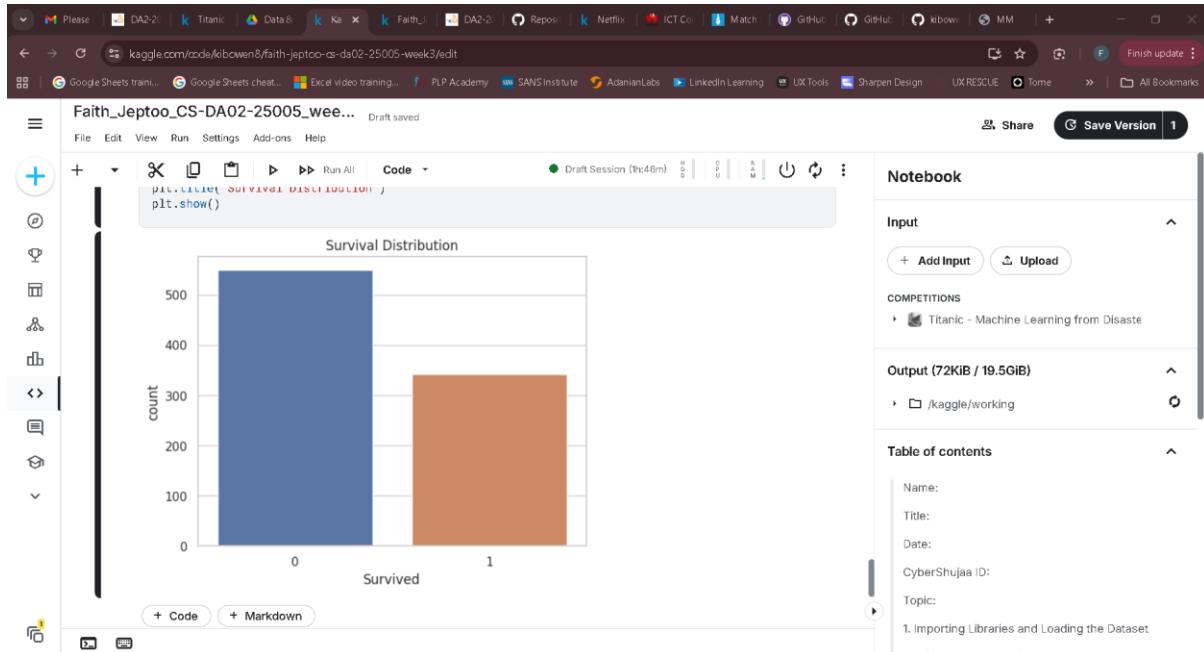
```
plt.figure(figsize=(6,4))
sns.boxplot(x=df['Fare'])
plt.title("Outlier Detection in Fare")
plt.show()
```



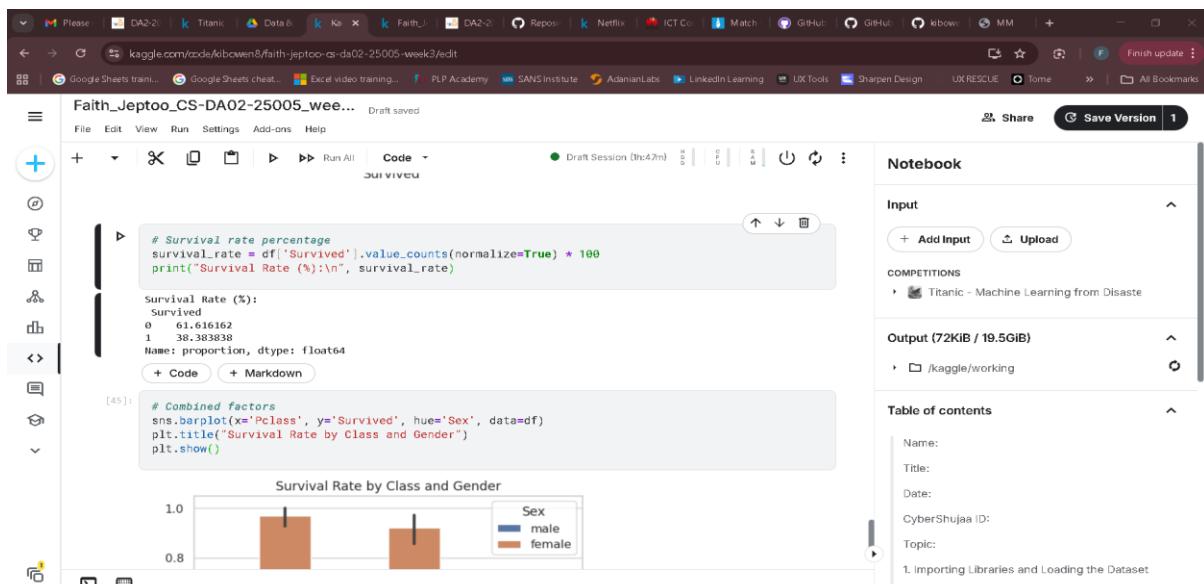
```
# Cap extreme fares at 99th percentile
fare_cap = df['Fare'].quantile(0.99)
df['Fare'] = np.where(df['Fare'] > fare_cap, fare_cap, df['Fare'])
```

Step 8: Target Variable (Survived) Analysis

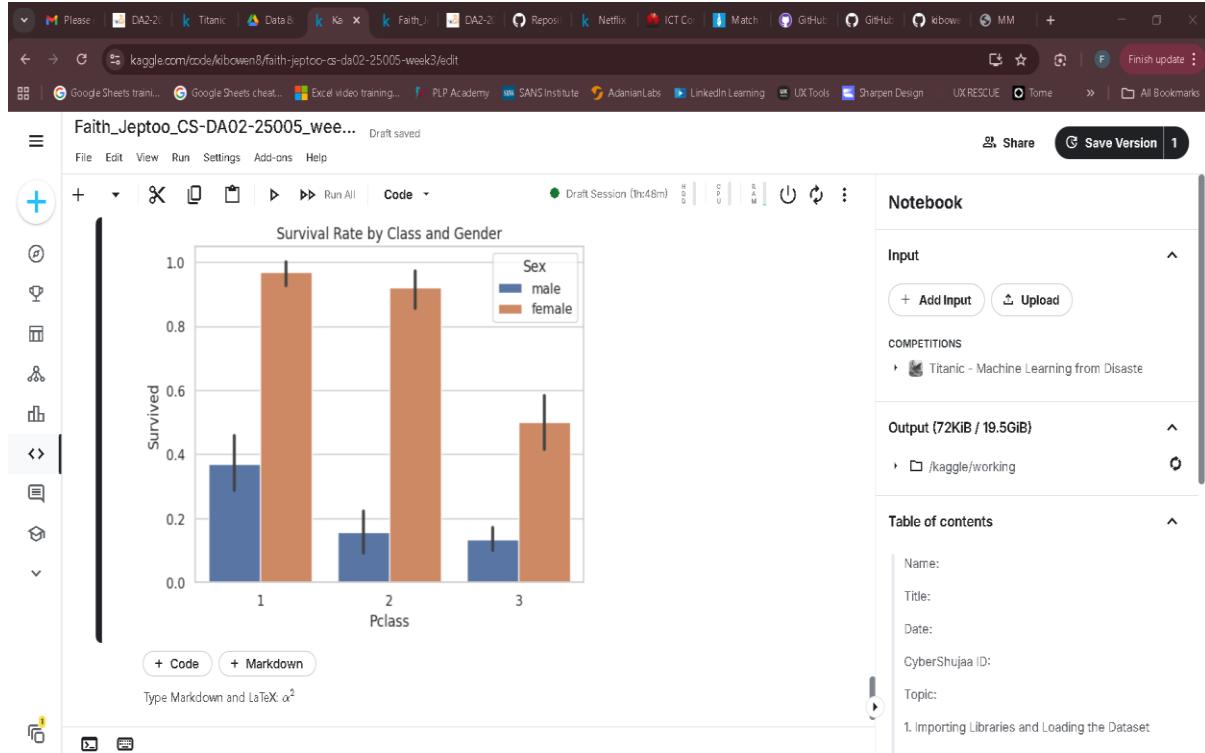
```
sns.countplot(x='Survived', data=df)
plt.title("Survival Distribution")
plt.show()
```



```
# Survival rate percentage
survival_rate = df['Survived'].value_counts(normalize=True) * 100
print("Survival Rate (%):\n", survival_rate)
```



```
# Combined factors
sns.barplot(x='Pclass', y='Survived', hue='Sex', data=df)
plt.title("Survival Rate by Class and Gender")
plt.show()
```



Conclusion

The Exploratory Data Analysis on the Titanic dataset revealed the following insights:

- The dataset contained missing values in *Age*, *Cabin*, and *Embarked* which were handled using imputation and removal.
- Most passengers were male and from 3rd class.
- Fare and Age were both right-skewed distributions.
- Survival was highly influenced by **Gender**, **Class**, and **Fare**.
- Outliers in *Fare* were capped to minimize skewness.
- The target variable (*Survived*) showed moderate imbalance, which is important for modeling

Link To Notebook

<https://www.kaggle.com/code/kibowen8/faith-jeptoo-cs-da02-25005-week-3>