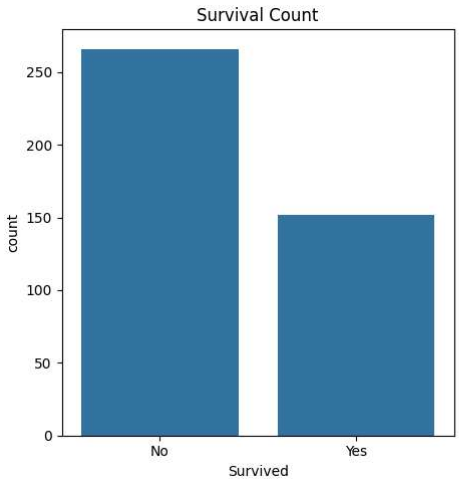
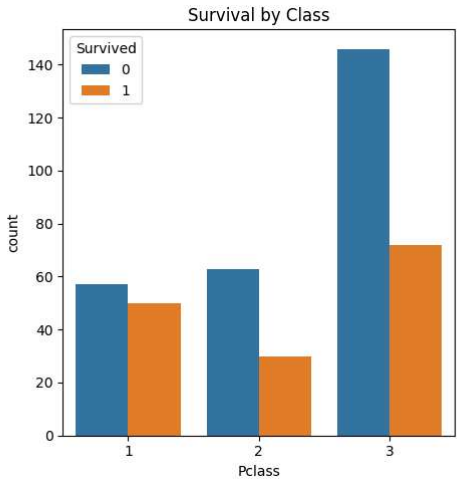
+ Code + Text

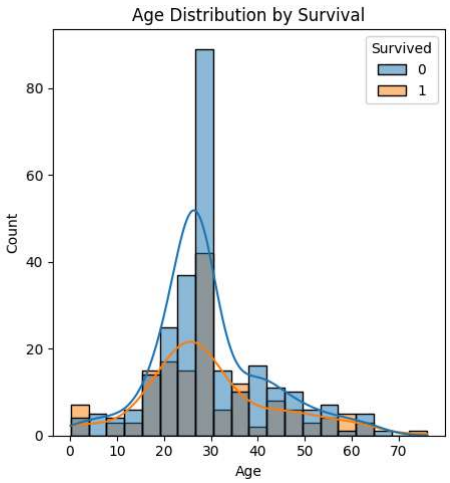
```
# Titanic Dataset Analysis & Prediction and import reqiured libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder


# Load dataset
df = pd.read_csv("/content/titanic.csv")


# Step 1: Data Cleaning
df_clean = df.copy()
df_clean['Age'].fillna(df_clean['Age'].median(), inplace=True)
df_clean['Fare'].fillna(df_clean['Fare'].median(), inplace=True)
df_clean.drop('Cabin', axis=1, inplace=True)
```

```
<ipython-input-3-1052877781>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df_clean['Age'].fillna(df_clean['Age'].median(), inplace=True)
<ipython-input-3-1052877781>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df_clean['Fare'].fillna(df_clean['Fare'].median(), inplace=True)
```

```
# Encode categorical columns
le_sex = LabelEncoder()
le_embarked = LabelEncoder()
df_clean['Sex'] = le_sex.fit_transform(df_clean['Sex'])
df_clean['Embarked'] = le_embarked.fit_transform(df_clean['Embarked'])


# Step 2: EDA
plt.figure(figsize=(14, 5))

plt.subplot(1, 3, 1)
sns.countplot(data=df_clean, x='Survived')
plt.title("Survival Count")
plt.xticks([0, 1], ['No', 'Yes'])

plt.subplot(1, 3, 2)
sns.countplot(data=df_clean, x='Pclass', hue='Survived')
plt.title("Survival by Class")

plt.subplot(1, 3, 3)
sns.histplot(data=df_clean, x='Age', hue='Survived', bins=20, kde=True)
plt.title("Age Distribution by Survival")

plt.tight_layout()
plt.show()
```








```
# Step 3: Model Training
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = df_clean[features]
y = df_clean['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Model Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        34

    accuracy                           1.00        84
   macro avg       1.00      1.00      1.00        84
weighted avg       1.00      1.00      1.00        84
```

```
# Step 4: Ask User to Enter Ticket Number and Check Survival
def check_ticket_survival(ticket_number):
    match = df[df['Ticket'] == ticket_number]
    if not match.empty:
        name = match.iloc[0]['Name']
        survived = match.iloc[0]['Survived']
        status = "Survived 🟢" if survived == 1 else "Did NOT Survive 🔴"
        return f"\nPassenger: {name}\nTicket: {ticket_number}\nStatus: {status}"
    else:
        return "\n❌ Ticket number not found in the dataset."


# Run user input section
user_ticket = input("\nEnter the ticket number to check survival status: ")
print(check_ticket_survival(user_ticket))
```

```
Enter the ticket number to check survival status: 363272

Passenger: Wilkes, Mrs. James (Ellen Needs)
Ticket: 363272
Status: Survived 🟢
```