```
from google.colab import files
uploaded = files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 61194 bytes, last modified: 12/11/2019 - 100% done

```
Saving train.csv to train.csv
```

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

sns.set_style('whitegrid')
%matplotlib inline
```

```
RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
```

```
df = pd.read_csv('train.csv')
df_original = df.copy()
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily	female	35.0	1	0	113803	53.1000	C123	S

Next steps: Generate code with df View recommended plots New interactive sheet

```
print("Shape:", df.shape)
df.info()
df.describe(include='all').T
```

```
Shape: (891, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
PassengerId	891.0	NaN	NaN	NaN	446.0	257.353842	1.0	223.5	446.0	668.5	891.0
Survived	891.0	NaN	NaN	NaN	0.383838	0.486592	0.0	0.0	0.0	1.0	1.0
Pclass	891.0	NaN	NaN	NaN	2.308642	0.836071	1.0	2.0	3.0	3.0	3.0
Name	891	891	Dooley, Mr. Patrick	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Sex	891	2	male	577	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Age	714.0	NaN	NaN	NaN	29.699118	14.526497	0.42	20.125	28.0	38.0	80.0
SibSp	891.0	NaN	NaN	NaN	0.523008	1.102743	0.0	0.0	0.0	1.0	8.0
Parch	891.0	NaN	NaN	NaN	0.381594	0.806057	0.0	0.0	0.0	0.0	6.0
Ticket	891	681	347082	7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fare	891.0	NaN	NaN	NaN	32.204208	49.693429	0.0	7.9104	14.4542	31.0	512.3292
Cabin	204	147	G6	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Embarked	889	3	S	644	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
missing = df.isnull().sum().sort_values(ascending=False)
print("Missing values:\n", missing)

print("\nDuplicate rows:", df.duplicated().sum())
```

```
Missing values:
 Cabin          687
Age            177
Embarked         2
PassengerId      0
Name             0
Pclass           0
Survived         0
Sex              0
Parch            0
SibSp            0
Fare             0
Ticket           0
dtype: int64

Duplicate rows: 0
```

```
for col in ['Survived', 'Pclass', 'Sex', 'Embarked']:
    print(f"=== {col} ===")
    print(df[col].value_counts(dropna=False))
    print()
```

```
=== Survived ===
Survived
0    549
1    342
Name: count, dtype: int64

=== Pclass ===
Pclass
```

```
3    491
1    216
2    184
Name: count, dtype: int64

=== Sex ===
Sex
male      577
female    314
Name: count, dtype: int64

=== Embarked ===
Embarked
S      644
C      168
Q       77
NaN      2
Name: count, dtype: int64
```

```python
# Fill missing Embarked with mode
df.loc[:, 'Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

# Fill missing Fare with median (if any)
if df['Fare'].isnull().sum() > 0:
    df.loc[:, 'Fare'] = df['Fare'].fillna(df['Fare'].median())

# Fill missing Age with median
df.loc[:, 'Age'] = df['Age'].fillna(df['Age'].median())
```

```python
# Extract title from Name
df['Title'] = df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)

# Group rare titles
rare_titles = ['Lady','Countess','Capt','Col','Don','Dr','Major','Rev','Sir','Jonkheer','Dona']
df['Title'] = df['Title'].replace(rare_titles, 'Rare')
df['Title'] = df['Title'].replace({'Mlle':'Miss','Ms':'Miss','Mme':'Mrs'})

# Family size & IsAlone
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
df['IsAlone'] = (df['FamilySize'] == 1).astype(int)

# Age bands
df['AgeBand'] = pd.cut(df['Age'], bins=[0,12,20,40,60,120], labels=['Child','Teen','Adult','MidAge','Senior'])

# Fare bands (quartiles)
df['FareBand'] = pd.qcut(df['Fare'], 4, labels=[1,2,3,4])

df.head()
```

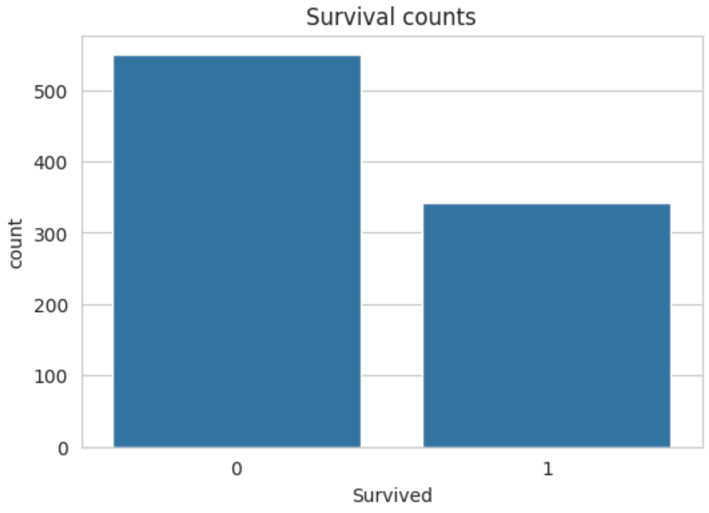
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	FamilySize	IsAlone	Ag
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Mr	2	0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	Mrs	2	0	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Miss	1	1	
3	4	1	1	Futrelle, Mrs. Jacques	female	35.0	1	0	113803	53.1000	C123	S	Mrs	2	0	

Next steps: Generate code with df View recommended plots New interactive sheet

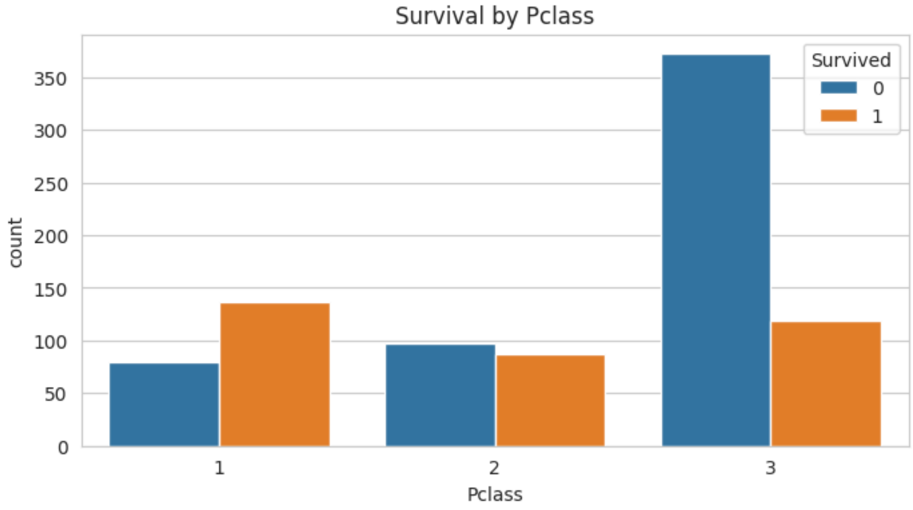
```python
# Encode Sex to numeric
df['SexCode'] = df['Sex'].map({'male':0, 'female':1})
```

```
# One-hot encode Embarked
df = pd.get_dummies(df, columns=['Embarked'], prefix='Emb', drop_first=True)

# Title to numeric
df['TitleCode'] = df['Title'].map({t:i for i,t in enumerate(df['Title'].unique())})
```

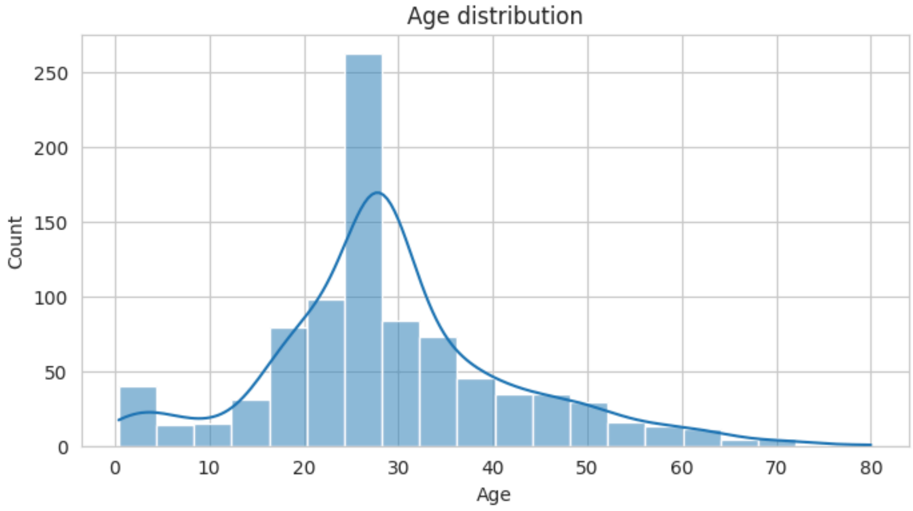
```
plt.figure(figsize=(6,4))
sns.countplot(x='Survived', data=df)
plt.title('Survival counts')
plt.show()

plt.figure(figsize=(8,4))
sns.countplot(x='Pclass', hue='Survived', data=df)
plt.title('Survival by Pclass')
plt.show()

plt.figure(figsize=(8,4))
sns.histplot(df['Age'], bins=20, kde=True)
plt.title('Age distribution')
plt.show()

plt.figure(figsize=(8,4))
sns.boxplot(x='Pclass', y='Fare', data=df)
plt.title('Fare distribution by Pclass')
plt.show()
```



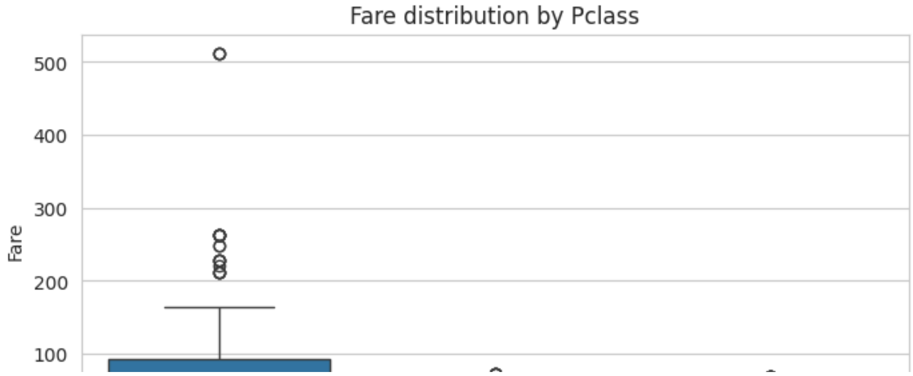
Survival counts



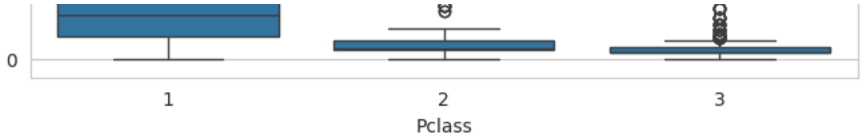
Survival by Pclass



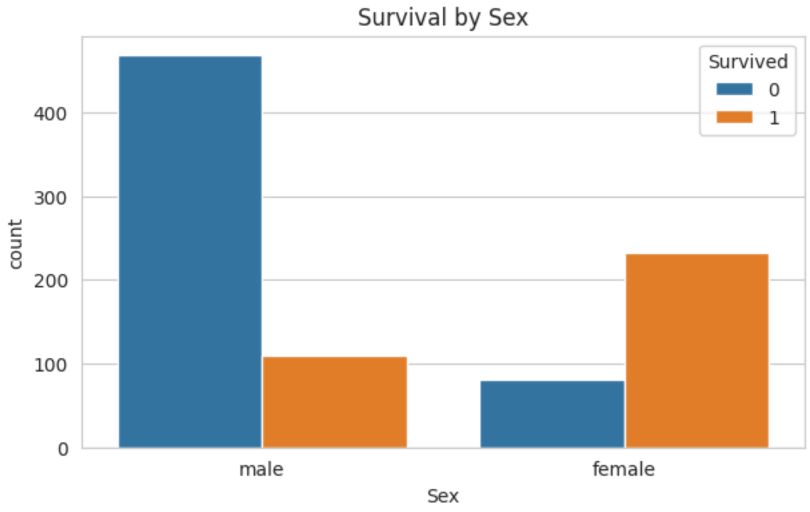
Age distribution



Fare distribution by Pclass

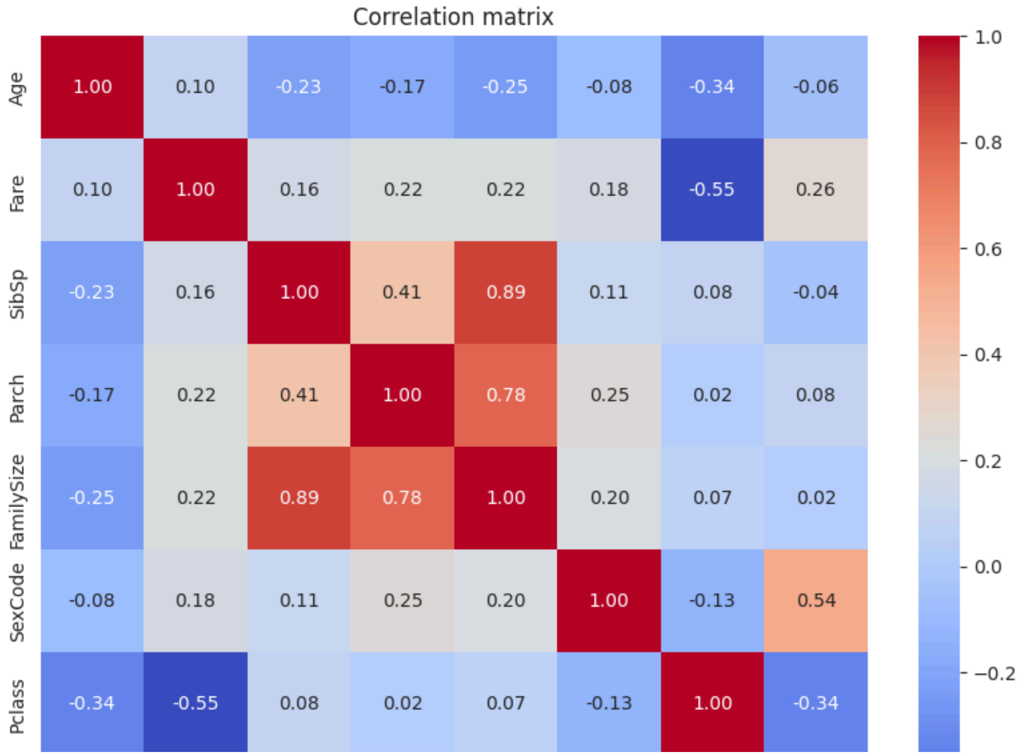


```
plt.figure(figsize=(7,4))
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title('Survival by Sex')
plt.show()

# Grouped survival rates
print("Survival rate by sex:\n", df.groupby('Sex')['Survived'].mean())
print("\nSurvival rate by Pclass:\n", df.groupby('Pclass')['Survived'].mean())
print("\nSurvival rate by AgeBand:\n", df.groupby('AgeBand')['Survived'].mean())
```



```
Survival rate by sex:
 Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64

Survival rate by Pclass:
 Pclass
1    0.629630
2    0.472826
3    0.242363
Name: Survived, dtype: float64

Survival rate by AgeBand:
 AgeBand
Child     0.579710
Teen      0.381818
Adult     0.364769
MidAge    0.390625
Senior    0.227273
Name: Survived, dtype: float64
/tmp/ipython-input-3120970124.py:9: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future v
  print("\nSurvival rate by AgeBand:\n", df.groupby('AgeBand')['Survived'].mean())
```

```
numeric_cols = ['Age','Fare','SibSp','Parch','FamilySize','SexCode','Pclass','Survived']
plt.figure(figsize=(10,8))
corr = df[numeric_cols].corr()
sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation matrix')
plt.show()
```



```
from scipy.stats import chi2_contingency, ttest_ind

# Chi-square test: Sex vs Survived
ct = pd.crosstab(df['Sex'], df['Survived'])
chi2, p, dof, exp = chi2_contingency(ct)
print("Chi2 test (Sex vs Survived): chi2=%.3f p=%.5f" % (chi2, p))

# t-test: Age difference between survived & not
survived_age = df[df['Survived']==1]['Age']
dead_age = df[df['Survived']==0]['Age']
tstat, pval = ttest_ind(survived_age, dead_age, nan_policy='omit')
print("t-test Age: t=%.3f p=%.5f" % (tstat, pval))
```

```
Chi2 test (Sex vs Survived): chi2=260.717 p=0.00000
t-test Age: t=-1.939 p=0.05276
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```