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

#1. mount Google drive
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

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

```
import·io
df·=·pd.read_csv(io.BytesIO(uploaded['train.csv']))
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1

```
df.isnull().sum()
```

```
PassengerId          0
Survived             0
Pclass               0
Name                 0
Sex                  0
Age                  0
SibSp                0
Parch                0
Ticket               0
Fare                 0
Cabin              687
Embarked             0
HasCabin             0
FamilySize           0
IsAlone              0
CategoricalFare      0
CategoricalAge       0
SexNumerical         0
dtype: int64
```

```
df.isnull().sum()
```

```
PassengerId          0
Survived             0
Pclass               0
Name                 0
Sex                  0
Age                  0
SibSp                0
Parch                0
Ticket               0
Fare                 0
Cabin              687
Embarked             0
HasCabin             0
FamilySize           0
IsAlone              0
CategoricalFare      0
CategoricalAge       0
SexNumerical         0
dtype: int64
```

```
def fill_na_age(df, colname):
    mean = df['Age'].mean()
    sd = df['Age'].std()
    def fill_empty(x):
        if np.isnan(x):
            return np.random.randint(mean-sd, mean+sd, ())
        return x
    return df[colname].apply(fill_empty).astype(int)
df['Age'] = fill_na_age(df, 'Age')
```

```
def create_feat_familly_size(df):
    return df['SibSp'] + df['Parch'] + 1

df['FamilySize'] = create_feat_familly_size(df)


def create_feat_isalone(df, colname):
    def _is_alone(x):
        if x==1:
            return 1
        return 0

    return df[colname].apply(_is_alone)

df['IsAlone'] = create_feat_isalone(df, 'FamilySize')


def create_feat_categoricalFare(df, colname):
    return pd.qcut(df[colname], 4, labels = [0, 1, 2, 3]).astype(int)
df['CategoricalFare'] = create_feat_categoricalFare(df, 'Fare')


def create_feat_categoricalAge(df, colname):
    return pd.qcut(df[colname], 5, labels = [0, 1, 2, 3, 4]).astype(int)
df['CategoricalAge'] = create_feat_categoricalAge(df, 'Age')


def create_feat_categoricalAge(df, colname):
    return pd.qcut(df[colname], 5, labels = [0, 1, 2, 3, 4]).astype(int)
df['CategoricalAge'] = create_feat_categoricalAge(df, 'Age')


import re
def create_feat_title(df, colname):
    def find_title(x):
        title_search = re.search(' ([A-Za-z]+)\.', x)
        if title_search:
            title = title_search.group(1)
            if title in ['Mlle', 'Ms']:
                return 'Miss'
            elif title in ['Mme', 'Mrs']:
                return 'Mrs'
            elif title=='Mr':
                return 'Mr'
            else:
                return 'Rare'
        return ""

    return_title= df[colname].apply(find_title)
    dict_title = {'Miss': 1, 'Mrs':2, 'Mr':3, 'Rare':4}
    return return_title.replace(dict_title)
df['Title'] = create_feat_title(df, 'Name')
```

```
def create_feat_sex(df, colname):
    def sex(x):
        if x=='male':
            return 1
        return 0

    return df[colname].apply(sex)

df['SexNumerical'] = create_feat_sex(df, 'Sex')
df['Embarked'] = df.Embarked.replace({'S': 0, 'C' : 1, 'Q' : 2})
```

```
df.isna().sum()
```

```
PassengerId          0
Survived             0
Pclass               0
Name                 0
Sex                  0
Age                  0
SibSp                0
Parch                0
Ticket               0
Fare                 0
Cabin              687
Embarked             0
HasCabin             0
FamilySize           0
IsAlone              0
CategoricalFare      0
CategoricalAge       0
SexNumerical         0
Title                0
dtype: int64
```

```
drop_list = ['PassengerId', 'Cabin', 'Ticket', 'SibSp', 'Name']
titanic = df.drop(drop_list, axis=1)
```

```
corrmat = titanic.corr()
corrmat
```

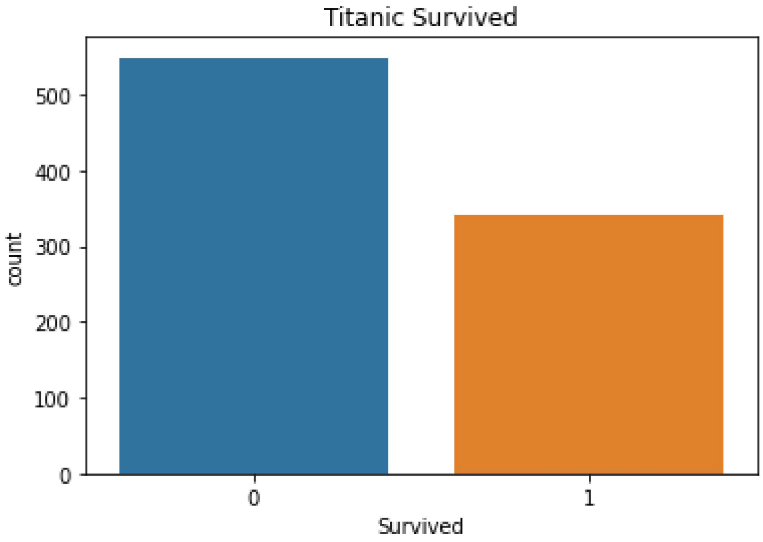
	Survived	Pclass	Age	Parch	Fare	Embarked	HasCabin
Survived	1.000000	-0.338481	-0.055717	0.081629	0.257307	0.106811	0.316912
Pclass	-0.338481	1.000000	-0.322743	0.018443	-0.549500	0.045702	-0.725541
Age	-0.055717	-0.322743	1.000000	-0.177178	0.093048	-0.007816	0.228420
Parch	0.081629	0.018443	-0.177178	1.000000	0.216225	-0.078665	0.036987
Fare	0.257307	-0.549500	0.093048	0.216225	1.000000	0.062142	0.482075
Embarked	0.106811	0.045702	-0.007816	-0.078665	0.062142	1.000000	0.013774
HasCabin	0.316912	-0.725541	0.228420	0.036987	0.482075	0.013774	1.000000
FamilySize	0.016639	0.065997	-0.241633	0.783111	0.217138	-0.080281	-0.009175
IsAlone	-0.203367	0.135207	0.174809	-0.583398	-0.271832	0.017807	-0.158029
CategoricalFare	0.299357	-0.634271	0.077588	0.393881	0.579345	-0.098161	0.500936

```
titanic['Survived'].value_counts()
```

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

```
sns.countplot('Survived', data=titanic)
plt.title("Titanic Survived")
plt.show()
```

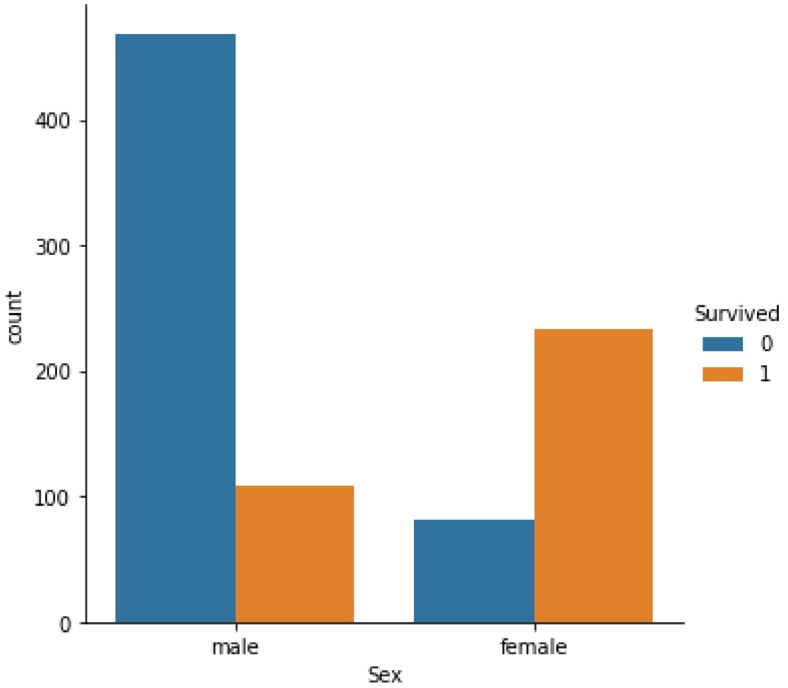
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
```



```
import seaborn as sns
import matplotlib.pyplot as plt
# Countplot
```

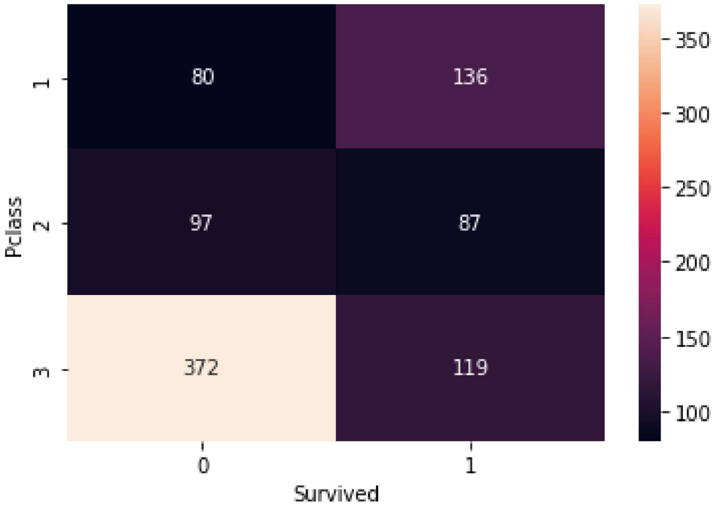
```
sns.catplot(x ="Sex", hue ="Survived",
kind ="count", data = titanic)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f797181e810>
```



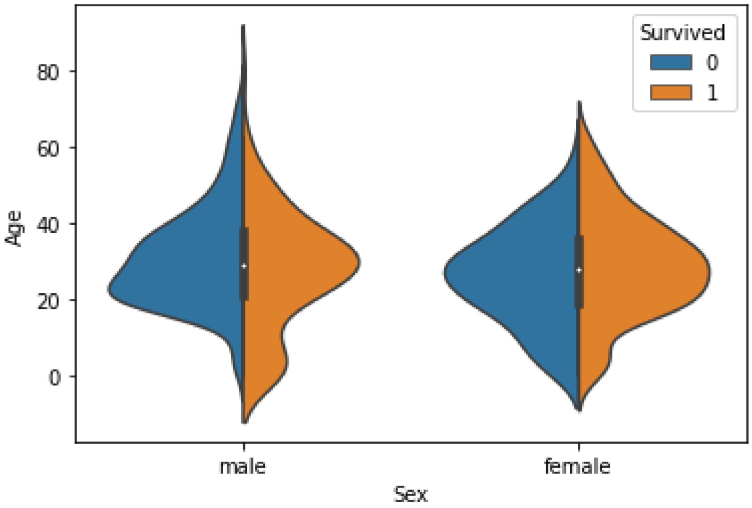
```
# Group the dataset by Pclass and Survived and then unstack them
group = titanic.groupby(['Pclass', 'Survived'])
pclass_survived = group.size().unstack()
# Heatmap - Color encoded 2D representation of data.
sns.heatmap(pclass_survived, annot = True, fmt ="d")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79717f5a10>
```



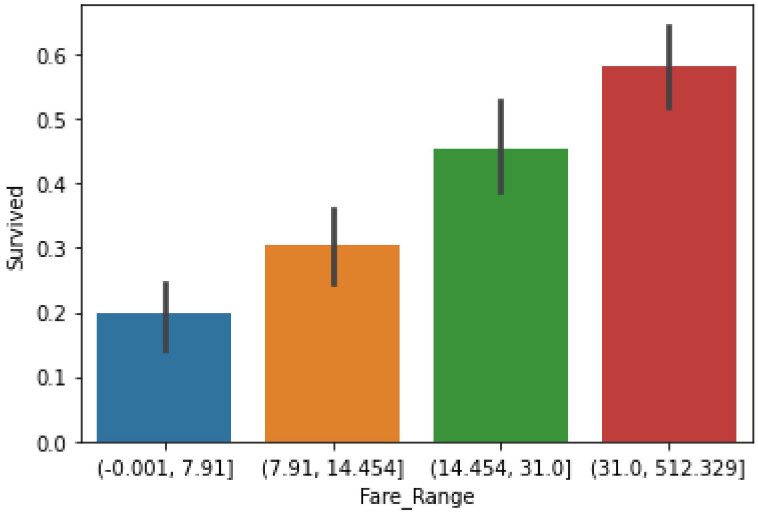
```
# Violinplot Displays distribution of data
# across all levels of a category.
sns.violinplot(x ="Sex", y ="Age", hue ="Survived",
data = titanic, split = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79716e4250>
```



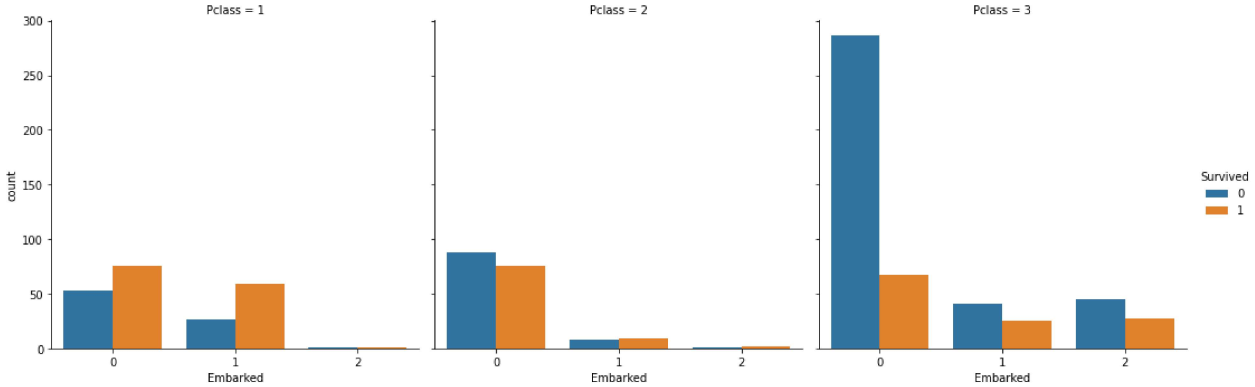
```
## Divide Fare into 4 bins
titanic['Fare_Range'] = pd.qcut(titanic['Fare'], 4)
# Barplot - Shows approximate values based
# on the height of bars.
sns.barplot(x ='Fare_Range', y ='Survived',
data = titanic)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7971677390>
```



```
# Countplot
sns.catplot(x ='Embarked', hue ='Survived',
kind ='count', col ='Pclass', data = titanic)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f79715d6e50>
```



✓ 0s completed at 10:57 PM