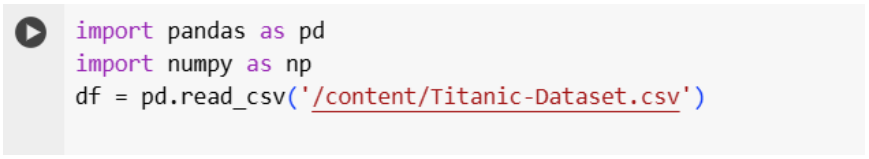
PYTHON Coding Assessment

1. Loading Data in Pandas DataFrame

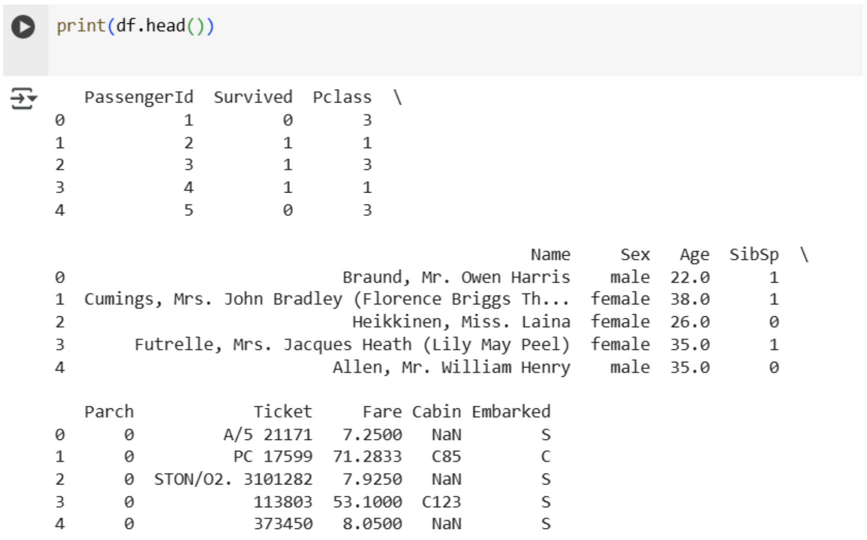


```python
import pandas as pd
import numpy as np
df = pd.read_csv('/content/Titanic-Dataset.csv')
```

Importing the python library and Loading the CSV file

2. Printing Rows of the Data

Printing 1st five rows



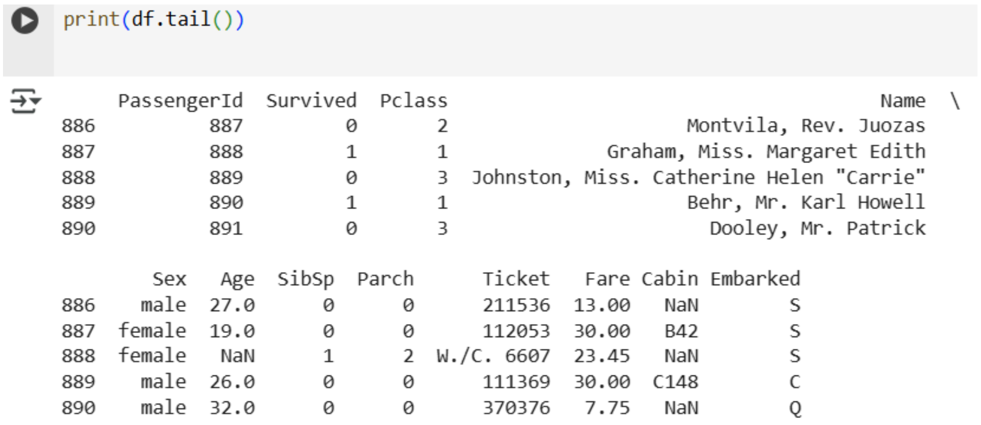
```python
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3   
1            2         1       1   
2            3         1       3   
3            4         1       1   
4            5         0       3   

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1   
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1   
2                             Heikkinen, Miss. Laina  female  26.0      0   
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1   
4                           Allen, Mr. William Henry    male  35.0      0   

   Parch            Ticket     Fare Cabin Embarked  
0      0         A/5 21171   7.2500   NaN        S  
1      0          PC 17599  71.2833   C85        C  
2      0  STON/O2. 3101282   7.9250   NaN        S  
3      0            113803  53.1000  C123        S  
4      0            373450   8.0500   NaN        S  
```

Printing last five rows

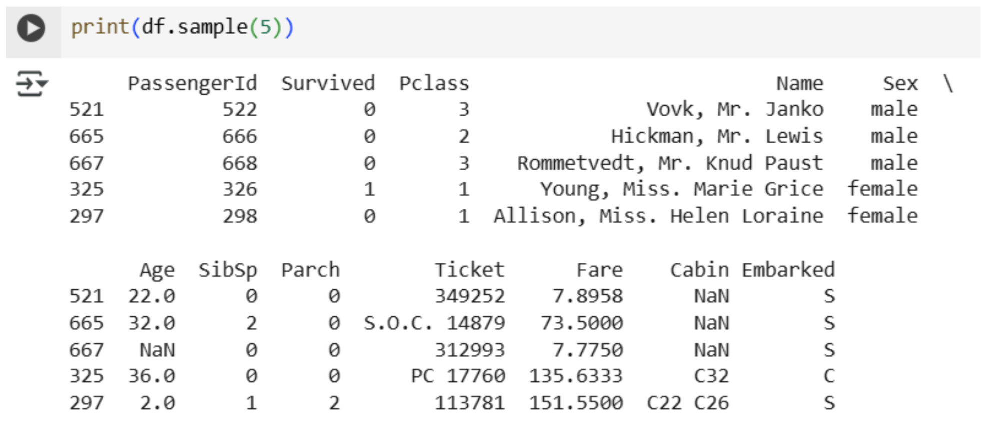


```python
print(df.tail())
```

```
     PassengerId  Survived  Pclass                                      Name  \
886          887         0       2                     Montvila, Rev. Juozas   
887          888         1       1              Graham, Miss. Margaret Edith   
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"   
889          890         1       1                     Behr, Mr. Karl Howell   
890          891         0       3                       Dooley, Mr. Patrick   

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked  
886    male  27.0      0      0      211536  13.00   NaN        S  
887  female  19.0      0      0      112053  30.00   B42        S  
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S  
889    male  26.0      0      0      111369  30.00  C148        C  
890    male  32.0      0      0      370376   7.75   NaN        Q  
```

Printing random 5 rows

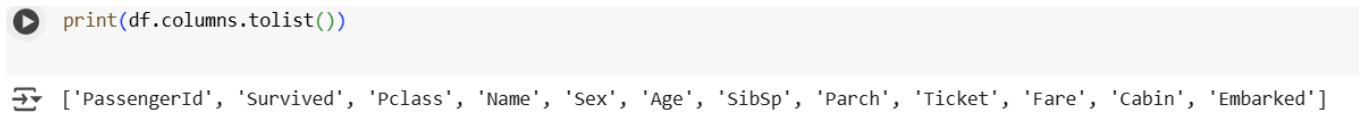


```
print(df.sample(5))
```

```
     PassengerId  Survived  Pclass                            Name     Sex  \
521          522         0       3                 Vovk, Mr. Janko    male   
665          666         0       2               Hickman, Mr. Lewis    male   
667          668         0       3       Rommetvedt, Mr. Knud Paust    male   
325          326         1       1        Young, Miss. Marie Grice  female   
297          298         0       1    Allison, Miss. Helen Loraine  female   

      Age  SibSp  Parch        Ticket      Fare    Cabin Embarked  
521  22.0      0      0        349252    7.8958      NaN        S  
665  32.0      2      0  S.O.C. 14879   73.5000      NaN        S  
667   NaN      0      0        312993    7.7750      NaN        S  
325  36.0      0      0      PC 17760  135.6333      C32        C  
297   2.0      1      2        113781  151.5500  C22 C26        S  
```

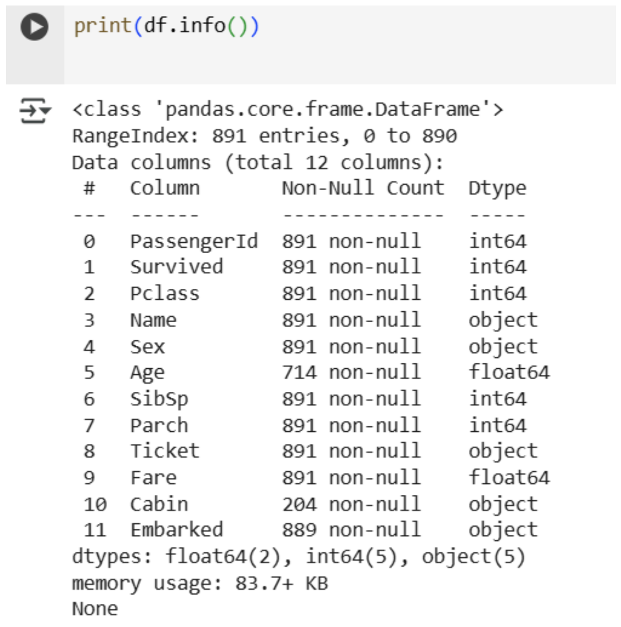
3. Printing Column Names



```
print(df.columns.tolist())
```

```
['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
```

4. Summary of DataFrame



```
print(df.info())
```

```
<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
None
```

5. Descriptive Statistical Measures

Only for numeric columns

```
print(df.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000   
mean    446.000000    0.383838    2.308642   29.699118    0.523008   
std     257.353842    0.486592    0.836071   14.526497    1.102743   
min       1.000000    0.000000    1.000000    0.420000    0.000000   
25%     223.500000    0.000000    2.000000   20.125000    0.000000   
50%     446.000000    0.000000    3.000000   28.000000    0.000000   
75%     668.500000    1.000000    3.000000   38.000000    1.000000   
max     891.000000    1.000000    3.000000   80.000000    8.000000   

            Parch        Fare  
count  891.000000  891.000000  
mean     0.381594   32.204208  
std      0.806057   49.693429  
min      0.000000    0.000000  
25%      0.000000    7.910400  
50%      0.000000   14.454200  
75%      0.000000   31.000000  
max      6.000000  512.329200  
```

For all column

```
print(df.describe(include='all'))
```

```
        PassengerId    Survived      Pclass                   Name   Sex  \
count    891.000000  891.000000  891.000000                    891   891   
unique          NaN         NaN         NaN                    891     2   
top             NaN         NaN         NaN  Dooley, Mr. Patrick  male   
freq            NaN         NaN         NaN                      1   577   
mean     446.000000    0.383838    2.308642                    NaN   NaN   
std      257.353842    0.486592    0.836071                    NaN   NaN   
min        1.000000    0.000000    1.000000                    NaN   NaN   
25%      223.500000    0.000000    2.000000                    NaN   NaN   
50%      446.000000    0.000000    3.000000                    NaN   NaN   
75%      668.500000    1.000000    3.000000                    NaN   NaN   
max      891.000000    1.000000    3.000000                    NaN   NaN   

               Age       SibSp       Parch  Ticket        Fare Cabin Embarked  
count   714.000000  891.000000  891.000000     891  891.000000   204      889  
unique         NaN         NaN         NaN     681         NaN   147        3  
top            NaN         NaN         NaN  347082         NaN    G6        S  
freq           NaN         NaN         NaN       7         NaN     4      644  
mean     29.699118    0.523008    0.381594     NaN   32.204208   NaN      NaN  
std      14.526497    1.102743    0.806057     NaN   49.693429   NaN      NaN  
min       0.420000    0.000000    0.000000     NaN    0.000000   NaN      NaN  
25%      20.125000    0.000000    0.000000     NaN    7.910400   NaN      NaN  
50%      28.000000    0.000000    0.000000     NaN   14.454200   NaN      NaN  
75%      38.000000    1.000000    0.000000     NaN   31.000000   NaN      NaN  
max      80.000000    8.000000    6.000000     NaN  512.329200   NaN      NaN  
```

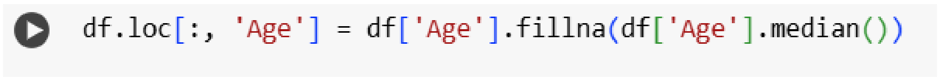
6. Missing Data Handling

Check missing values

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

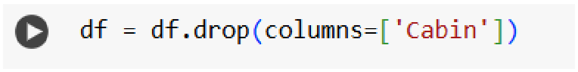
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

Fill missing values with median



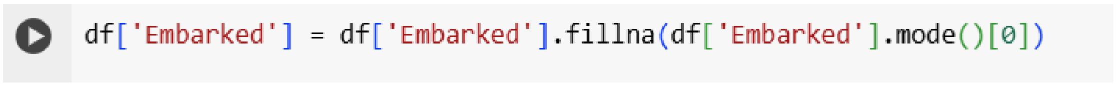
```
df.loc[:, 'Age'] = df['Age'].fillna(df['Age'].median())
```

Drop missing rows



```
df = df.drop(columns=['Cabin'])
```

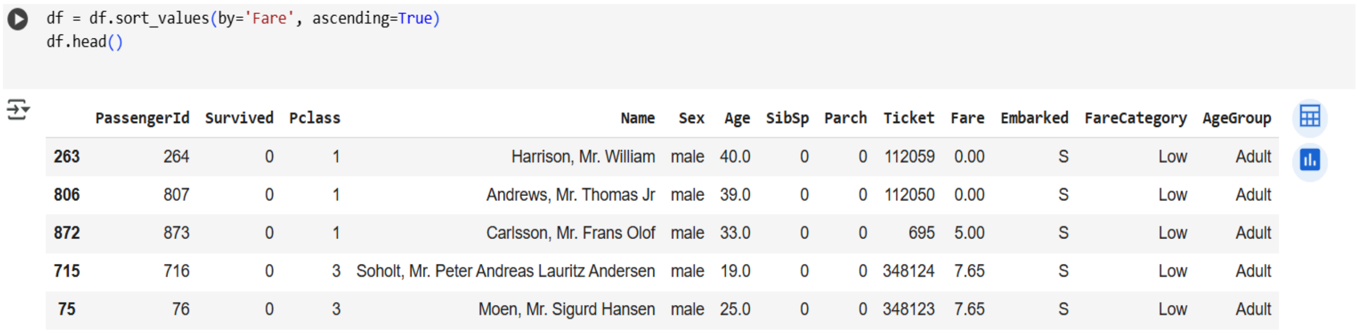
Fill categorical column with mode



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

7. Sorting DataFrame Values

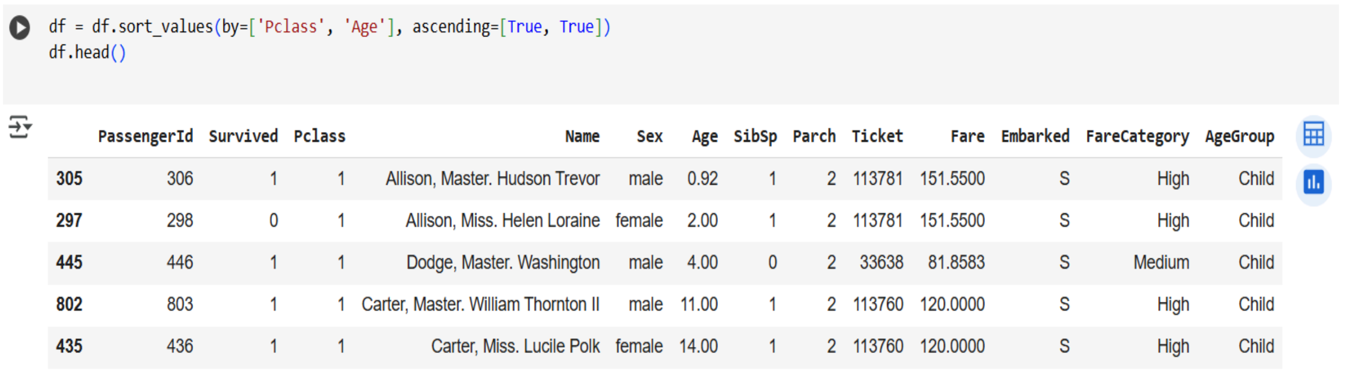
Sort by Fare (lowest to highest) - sorting by single column



```
df = df.sort_values(by='Fare', ascending=True)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	FareCategory	AgeGroup
263	264	0	1	Harrison, Mr. William	male	40.0	0	0	112059	0.00	S	Low	Adult
806	807	0	1	Andrews, Mr. Thomas Jr	male	39.0	0	0	112050	0.00	S	Low	Adult
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.00	S	Low	Adult
715	716	0	3	Soholt, Mr. Peter Andreas Lauritz Andersen	male	19.0	0	0	348124	7.65	S	Low	Adult
75	76	0	3	Moen, Mr. Sigurd Hansen	male	25.0	0	0	348123	7.65	S	Low	Adult

Sorting by multiple column



```
df = df.sort_values(by=['Pclass', 'Age'], ascending=[True, True])
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	FareCategory	AgeGroup
305	306	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	S	High	Child
297	298	0	1	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	S	High	Child
445	446	1	1	Dodge, Master. Washington	male	4.00	0	2	33638	81.8583	S	Medium	Child
802	803	1	1	Carter, Master. William Thornton II	male	11.00	1	2	113760	120.0000	S	High	Child
435	436	1	1	Carter, Miss. Lucile Polk	female	14.00	1	2	113760	120.0000	S	High	Child

8. Merge DataFrames

```
ticket_info = pd.DataFrame({
    'Ticket': ['A/5 21171', 'PC 17599', 'STON/O2. 3101282'],
    'Discount': [5, 10, 0]
})
merged_df = pd.merge(df, ticket_info, on='Ticket', how='left')
print(merged_df[['Name', 'Ticket', 'Fare', 'Discount']].head())
```

```
                               Name    Ticket      Fare  Discount
0  Cardeza, Mr. Thomas Drake Martinez  PC 17755  512.3292       NaN
1              Lesurer, Mr. Gustave J  PC 17755  512.3292       NaN
2                   Fortune, Mr. Mark     19950  263.0000       NaN
3      Fortune, Miss. Alice Elizabeth     19950  263.0000       NaN
4          Fortune, Miss. Mabel Helen     19950  263.0000       NaN
```

9. Apply Function

```
def fare_category(fare):
    if fare >= 100:
        return 'High'
    elif fare >= 50:
        return 'Medium'
    else:
        return 'Low'

df['FareCategory'] = df['Fare'].apply(fare_category)
print(df[['Fare', 'FareCategory']].head())
```

```
         Fare FareCategory
679  512.3292         High
737  512.3292         High
438  263.0000         High
341  263.0000         High
88   263.0000         High
```

10. Using Lambda Function

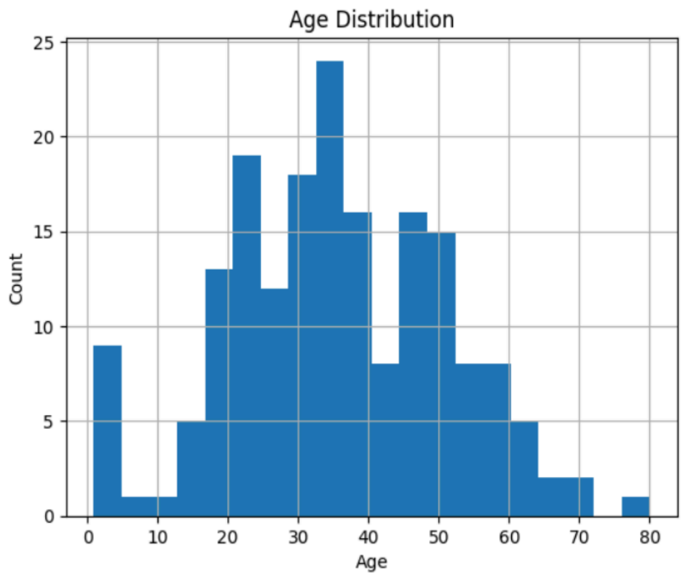
```
df['AgeGroup'] = df['Age'].apply(lambda age: 'Child' if age < 18 else 'Adult')
print(df[['Age', 'AgeGroup']].head())
```

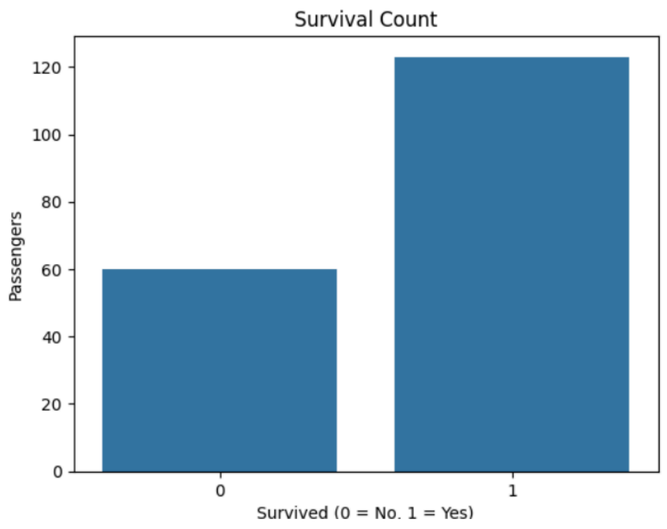
```
      Age AgeGroup
679  36.0    Adult
737  35.0    Adult
438  64.0    Adult
341  24.0    Adult
88   23.0    Adult
```

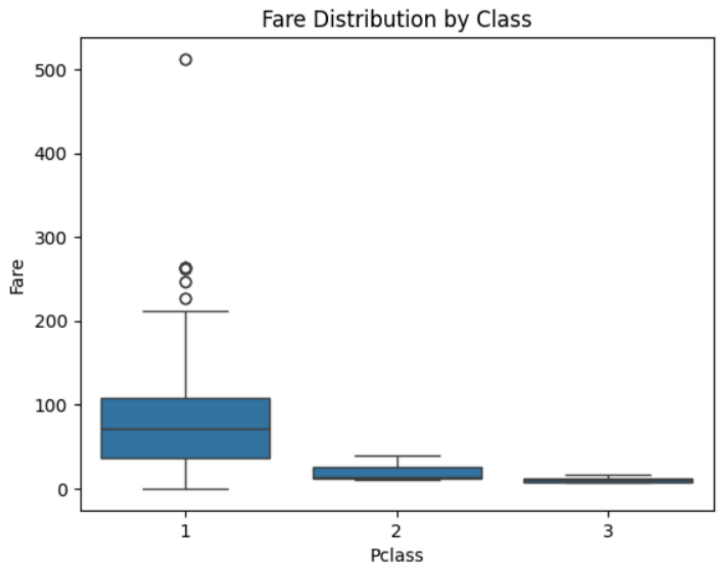
11. Visualizing the DataFrame

```
import matplotlib.pyplot as plt
import seaborn as sns
df['Age'].hist(bins=20)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
sns.countplot(x='Survived', data=df)
plt.title("Survival Count")
plt.xlabel("Survived (0 = No, 1 = Yes)")
plt.ylabel("Passengers")
plt.show()
sns.boxplot(x='Pclass', y='Fare', data=df)
plt.title("Fare Distribution by Class")
plt.show()
```

Output:







Part 2: Performing Joins in Pandas

Two Dataframes :
df1 has info about people.
df2 has their scores, but notice: id=1 is missing in df2 and id=4 is missing in df1.

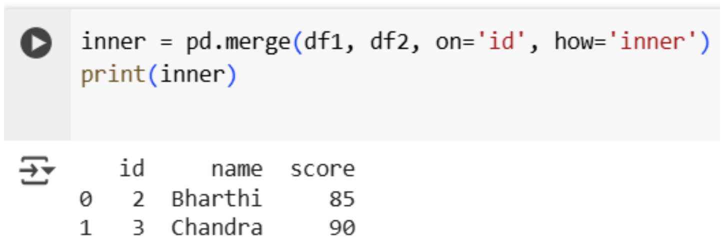
```python
import pandas as pd

df1 = pd.DataFrame({
    'id': [1, 2, 3],
    'name': ['Aruna', 'Bharthi', 'Chandra']
})

df2 = pd.DataFrame({
    'id': [2, 3, 4],
    'score': [85, 90, 75]
})
```

1. INNER JOIN

Takes only matching keys from both DataFrames

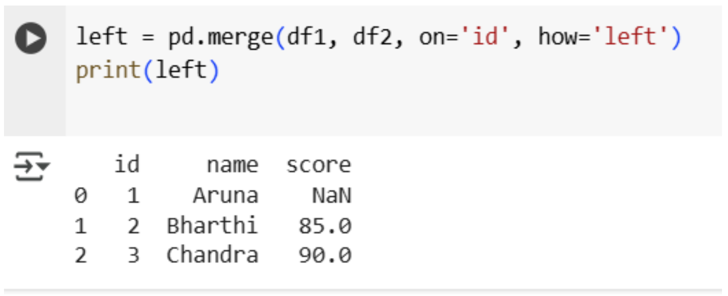


```python
inner = pd.merge(df1, df2, on='id', how='inner')
print(inner)
```

```
   id     name  score
0   2  Bharthi     85
1   3  Chandra     90
```

2. LEFT JOIN

It takes all records from left, match from right

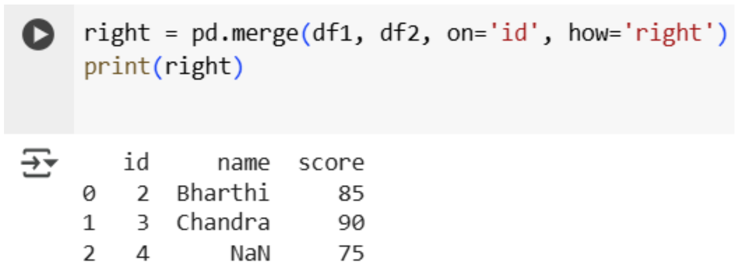


```python
left = pd.merge(df1, df2, on='id', how='left')
print(left)
```

```
   id     name  score
0   1    Aruna    NaN
1   2  Bharthi   85.0
2   3  Chandra   90.0
```

3. RIGHT JOIN

It takes all records from right, match from left



```python
right = pd.merge(df1, df2, on='id', how='right')
print(right)
```

```
   id     name  score
0   2  Bharthi     85
1   3  Chandra     90
2   4      NaN     75
```

4. OUTER JOIN

It takes all records from both, fill with NaN where no match

```python
outer = pd.merge(df1, df2, on='id', how='outer')
print(outer)
```

```
   id     name  score
0   1    Aruna    NaN
1   2  Bharthi   85.0
2   3  Chandra   90.0
3   4      NaN   75.0
```

5. JOIN on Columns with Different Names

```python
df1 = pd.DataFrame({'student_id': [1, 2, 3], 'name': ['A', 'B', 'C']})
df2 = pd.DataFrame({'exam_id': [2, 3, 4], 'score': [80, 90, 70]})

merged = pd.merge(df1, df2, left_on='student_id', right_on='exam_id', how='inner')
print(merged)
```

```
   student_id name  exam_id  score
0           2    B        2     80
1           3    C        3     90
```

6. USING suffixes to Handle Duplicate Column Names

```python
df1 = pd.DataFrame({'id': [1, 2], 'value': ['X', 'Y']})
df2 = pd.DataFrame({'id': [1, 2], 'value': ['A', 'B']})

merged = pd.merge(df1, df2, on='id', suffixes=('_df1', '_df2'))
print(merged)
```

```
   id value_df1 value_df2
0   1         X         A
1   2         Y         B
```

7. Joining on Index

```python
df1 = df1.set_index('id')
df2 = df2.set_index('id')

joined = df1.join(df2, how='inner')
print(joined)
```

```
        name  score
id
2   Bharathi     85
3    Chandra     90
```