```
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('/content/sample_data/Titanic-Dataset.csv')

# 1. Display the first 5 rows of the dataset
print(data.head())

# 2. Display the last 5 rows of the dataset
print(data.tail())

# 3. Get the shape of the dataset
print(data.shape)

# 4. Get a summary of the dataset (info)
print(data.info())

# 5. Get basic statistics of the dataset
print(data.describe())

# 6. Check for missing values
print(data.isnull().sum())

# 7. Fill missing values in the 'Age' column with the median age
data['Age'].fillna(data['Age'].median(), inplace=True)

# 8. Fill missing values in the 'Embarked' column with the mode
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# 9. Drop the 'Cabin' column due to many missing values
data.drop('Cabin', axis=1, inplace=True)

# 10. Create a new column 'FamilySize' by adding 'SibSp' and 'Parch'
data['FamilySize'] = data['SibSp'] + data['Parch']
```

```
   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
     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
(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
None
       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
```

```
# 11. Create a new column 'IsAlone' (1 if alone, 0 otherwise)
data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)

# 12. Convert 'Sex' to numeric (male: 0, female: 1)
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# 13. One-hot encode the 'Embarked' column
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# 14. Get the mean age of passengers
print(data['Age'].mean())

# 15. Get the median fare of passengers
print(data['Fare'].median())

# 16. Get the number of passengers by class
print(data['Pclass'].value_counts())

# 17. Get the number of passengers by gender
print(data['Sex'].value_counts())

# 18. Get the number of passengers by survival status
print(data['Survived'].value_counts())

# 19. Calculate the survival rate
```

```
print(data['Survived'].mean())

# 20. Calculate the survival rate by gender
print(data.groupby('Sex')['Survived'].mean())
```

```
29.36158249158249
14.4542
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
Sex
0    577
1    314
Name: count, dtype: int64
Survived
0    549
1    342
Name: count, dtype: int64
0.3838383838383838
Sex
0    0.188908
1    0.742038
Name: Survived, dtype: float64
```

```
# 21. Calculate the survival rate by class
print(data.groupby('Pclass')['Survived'].mean())

# 22. Calculate the survival rate by embarked location
print(data.groupby('Embarked_S')['Survived'].mean())

# 23. Calculate the survival rate by family size
print(data.groupby('FamilySize')['Survived'].mean())

# 24. Calculate the survival rate by being alone
print(data.groupby('IsAlone')['Survived'].mean())

# 25. Get the average fare by class
print(data.groupby('Pclass')['Fare'].mean())

# 26. Get the average age by class
print(data.groupby('Pclass')['Age'].mean())

# 27. Get the average age by survival status
print(data.groupby('Survived')['Age'].mean())

# 28. Get the average fare by survival status
print(data.groupby('Survived')['Fare'].mean())

# 29. Get the number of survivors by class
print(data[data['Survived'] == 1]['Pclass'].value_counts())

# 30. Get the number of non-survivors by class
print(data[data['Survived'] == 0]['Pclass'].value_counts())
```

```
Pclass
1    0.629630
2    0.472826
```

```
3    0.242363
Name: Survived, dtype: float64
Embarked_S
False    0.502041
True     0.339009
Name: Survived, dtype: float64
FamilySize
0     0.303538
1     0.552795
2     0.578431
3     0.724138
4     0.200000
5     0.136364
6     0.333333
7     0.000000
10    0.000000
Name: Survived, dtype: float64
IsAlone
0    0.505650
1    0.303538
Name: Survived, dtype: float64
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Fare, dtype: float64
Pclass
1    36.812130
2    29.765380
3    25.932627
Name: Age, dtype: float64
Survived
0    30.028233
1    28.291433
Name: Age, dtype: float64
Survived
0    22.117887
1    48.395408
Name: Fare, dtype: float64
Pclass
1    136
3    119
2     87
Name: count, dtype: int64
Pclass
3    372
2     97
1     80
Name: count, dtype: int64
```

```
# 31. Get the number of survivors by gender
print(data[data['Survived'] == 1]['Sex'].value_counts())

# 32. Get the number of non-survivors by gender
print(data[data['Survived'] == 0]['Sex'].value_counts())

# 33. Get the number of survivors by embarked location
print(data[data['Survived'] == 1]['Embarked_S'].value_counts())

# 34. Get the number of non-survivors by embarked location
print(data[data['Survived'] == 0]['Embarked_S'].value_counts())
```

```python
# 35. Calculate the percentage of children (Age < 18) who survived
children = data[data['Age'] < 18]
print(children['Survived'].mean())

# 36. Calculate the percentage of adults (Age >= 18) who survived
adults = data[data['Age'] >= 18]
print(adults['Survived'].mean())

# 37. Get the median age of survivors
print(data[data['Survived'] == 1]['Age'].median())

# 38. Get the median age of non-survivors
print(data[data['Survived'] == 0]['Age'].median())

# 39. Get the median fare of survivors
print(data[data['Survived'] == 1]['Fare'].median())

# 40. Get the median fare of non-survivors
print(data[data['Survived'] == 0]['Fare'].median())
```

```
Sex
1    233
0    109
Name: count, dtype: int64
Sex
0    468
1     81
Name: count, dtype: int64
Embarked_S
True     219
False    123
Name: count, dtype: int64
Embarked_S
True     427
False    122
Name: count, dtype: int64
0.5398230088495575
0.36118251928020567
28.0
28.0
26.0
10.5
```

```python
# 41. Find the most common age
print(data['Age'].mode()[0])

# 42. Find the most common fare
print(data['Fare'].mode()[0])

# 43. Find the most common family size
print(data['FamilySize'].mode()[0])

# 44. Find the passenger with the maximum fare
print(data[data['Fare'] == data['Fare'].max()])

# 45. Find the passenger with the minimum fare
print(data[data['Fare'] == data['Fare'].min()])

# 46. Get the number of passengers from each port of embarkation
print(data['Embarked_S'].value_counts())
```

```
print(data['Embarked_S'].value_counts())

# 47. Get the mean family size of survivors
print(data[data['Survived'] == 1]['FamilySize'].mean())

# 48. Get the mean family size of non-survivors
print(data[data['Survived'] == 0]['FamilySize'].mean())

# 49. Get the number of passengers aged 30 or older
print(data[data['Age'] >= 30].shape[0])

# 50. Get the number of passengers younger than 30
print(data[data['Age'] < 30].shape[0])
```

```
28.0
8.05
0
     PassengerId  Survived  Pclass                                Name  Sex  \
258          259         1       1                   Ward, Miss. Anna    1
679          680         1       1  Cardeza, Mr. Thomas Drake Martinez    0
737          738         1       1              Lesurer, Mr. Gustave J    0

      Age  SibSp  Parch    Ticket      Fare  FamilySize  IsAlone  Embarked_Q  \
258  35.0      0      0  PC 17755  512.3292           0        1       False
679  36.0      0      1  PC 17755  512.3292           1        0       False
737  35.0      0      0  PC 17755  512.3292           0        1       False

     Embarked_S
258       False
679       False
737       False
     PassengerId  Survived  Pclass                                Name  Sex  \
179          180         0       3               Leonard, Mr. Lionel    0
263          264         0       1             Harrison, Mr. William    0
271          272         1       3      Tornquist, Mr. William Henry    0
277          278         0       2       Parkes, Mr. Francis "Frank"    0
302          303         0       3   Johnson, Mr. William Cahoone Jr    0
413          414         0       2    Cunningham, Mr. Alfred Fleming    0
466          467         0       2             Campbell, Mr. William    0
481          482         0       2  Frost, Mr. Anthony Wood "Archie"    0
597          598         0       3               Johnson, Mr. Alfred    0
633          634         0       1     Parr, Mr. William Henry Marsh    0
674          675         0       2        Watson, Mr. Ennis Hastings    0
732          733         0       2              Knight, Mr. Robert J    0
806          807         0       1            Andrews, Mr. Thomas Jr    0
815          816         0       1                 Fry, Mr. Richard    0
822          823         0       1   Reuchlin, Jonkheer. John George    0

      Age  SibSp  Parch  Ticket  Fare  FamilySize  IsAlone  Embarked_Q  \
179  36.0      0      0    LINE   0.0           0        1       False
263  40.0      0      0  112059   0.0           0        1       False
271  25.0      0      0    LINE   0.0           0        1       False
277  28.0      0      0  239853   0.0           0        1       False
302  19.0      0      0    LINE   0.0           0        1       False
413  28.0      0      0  239853   0.0           0        1       False
466  28.0      0      0  239853   0.0           0        1       False
481  28.0      0      0  239854   0.0           0        1       False
597  49.0      0      0    LINE   0.0           0        1       False
633  28.0      0      0  112052   0.0           0        1       False
674  28.0      0      0  239856   0.0           0        1       False
732  28.0      0      0  239855   0.0           0        1       False
806  39.0      0      0  112050   0.0           0        1       False
815  28.0      0      0  112058   0.0           0        1       False
```

```
     822  38.0      0      0   19972   0.0          0        1       False

          Embarked_S
     179        True
     263        True
     271        True
     277        True
     302        True
     413        True
```

Start coding or generate with AI.