In [ ]:
```
# pandas library for data manipulation and analysis in Python
import pandas as pd
```

Load the dataset into a pandas DataFrame

In [ ]:
```
# Load the dataset into a pandas DataFrame using read_csv
# pd.read_csv() reads a CSV file and loads it into a DataFrame for further analy
train_df = pd.read_csv(r'C:\Users\Bhanu Sri V\Bootcamp-1\Pandas\train.csv')
```

In [ ]:
```
# Display the entire DataFrame
# print() displays the entire DataFrame so you can see all the data loaded
print(train_df)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3   
1              2         1       1   
2              3         1       3   
3              4         1       1   
4              5         0       3   
..           ...       ...     ...   
886          887         0       2   
887          888         1       1   
888          889         0       3   
889          890         1       1   
890          891         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   
..                                                 ...     ...   ...    ...   
886                              Montvila, Rev. Juozas    male  27.0      0   
887                       Graham, Miss. Margaret Edith  female  19.0      0   
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1   
889                              Behr, Mr. Karl Howell    male  26.0      0   
890                                Dooley, Mr. Patrick    male  32.0      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  
..     ...               ...      ...   ...      ...  
886      0            211536  13.0000   NaN        S  
887      0            112053  30.0000   B42        S  
888      2        W./C. 6607  23.4500   NaN        S  
889      0            111369  30.0000  C148        C  
890      0            370376   7.7500   NaN        Q  

[891 rows x 12 columns]
```

Identify which columns have missing values and how many

```
In [ ]: # Show the count of missing values in each column using isnull().sum()
        # isnull().sum() shows the count of missing (NaN) values in each column
        print(train_df.isnull().sum())
```

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

Fill missing age values with the median age

```
In [ ]: # Fill missing values in the 'Age' column with the median age
        # fillna() replaces missing values in the 'Age' column with the median age
        train_df['Age'] = train_df['Age'].fillna(train_df['Age'].median())
```

Display the first 5 and last 5 rows

```
In [ ]: # Print the first 5 rows of the DataFrame using head()
        # head() returns the first 5 rows of the DataFrame for a quick preview
        print("First 5 rows:")
        print(train_df.head().to_string())
```

```
First 5 rows:
   PassengerId  Survived  Pclass                                                 
Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                               Braund, Mr. Owen Ha
rris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Tha
yer)  female  38.0      1      0          PC 17599  71.2833   C85        C
2            3         1       3                                Heikkinen, Miss. L
aina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3            4         1       1         Futrelle, Mrs. Jacques Heath (Lily May P
eel)  female  35.0      1      0            113803  53.1000  C123        S
4            5         0       3                              Allen, Mr. William H
enry    male  35.0      0      0            373450   8.0500   NaN        S
```

```
In [ ]: # tail() returns the last 5 rows of the DataFrame for a quick preview
        print("Last 5 rows:")
        print(train_df.tail().to_string())
```

```
Last 5 rows:
     PassengerId  Survived  Pclass                                      Name
Sex    Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886          887         0       2                     Montvila, Rev. Juozas    m
ale   27.0      0      0      211536  13.00   NaN        S
887          888         1       1              Graham, Miss. Margaret Edith  fem
ale   19.0      0      0      112053  30.00   B42        S
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"  fem
ale   28.0      1      2  W./C. 6607  23.45   NaN        S
889          890         1       1                     Behr, Mr. Karl Howell    m
ale   26.0      0      0      111369  30.00  C148        C
890          891         0       3                       Dooley, Mr. Patrick    m
ale   32.0      0      0      370376   7.75   NaN        Q
```

Check the dimensions of the dataset (number of rows and columns)

In [ ]:
```
# shape returns a tuple (rows, columns) representing the DataFrame's dimensions
print("Number of Rows and Columns:")
print(train_df.shape)
```

```
Number of Rows and Columns:
(891, 12)
```

Get the column names and data types

In [ ]:
```
# dtypes shows the data type of each column in the DataFrame
print("Column Names and Data Types:")
print(train_df.dtypes)
```

```
Column Names and Data Types:
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Generate descriptive statistics for numerical columns

In [ ]:
```
# describe() generates descriptive statistics for numerical columns
print("Description of Numeric Columns:")
print(train_df.describe().to_string())
```

```
Description of Numeric Columns:
       PassengerId    Survived      Pclass         Age       SibSp       Parch
Fare
count   891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  8
91.000000
mean    446.000000    0.383838    2.308642   29.361582    0.523008    0.381594
32.204208
std     257.353842    0.486592    0.836071   13.019697    1.102743    0.806057
49.693429
min       1.000000    0.000000    1.000000    0.420000    0.000000    0.000000
0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000    0.000000
7.910400
50%     446.000000    0.000000    3.000000   28.000000    0.000000    0.000000
14.454200
75%     668.500000    1.000000    3.000000   35.000000    1.000000    0.000000
31.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000    6.000000  5
12.329200
```

What is the survival rate overall?

```
In [ ]: # mean() calculates the average; here, it gives the overall survival rate as a p
        print("Overall Survival Rate:")
        print(train_df['Survived'].mean()*100)
```

```
Overall Survival Rate:
38.38383838383838
```

How many passengers were in each class (Pclass)?

```
In [ ]: # value_counts() counts the number of occurrences of each class in 'Pclass'; sor
        print("Passengers in each class:")
        print(train_df['Pclass'].value_counts().sort_index())
```

```
Passengers in each class:
Pclass
1    216
2    184
3    491
Name: count, dtype: int64
```

What is the distribution of genders onboard?

```
In [ ]: # value_counts() shows the count and percentage of each gender in the 'Sex' colu
        print("Gender Distribution:")
        print(train_df['Sex'].value_counts())
        print("Percentage of Gender Distribution:")
        print(train_df['Sex'].value_counts(normalize=True) * 100)
```

```
Gender Distribution:
Sex
male      577
female    314
Name: count, dtype: int64
Percentage of Gender Distribution:
Sex
male      64.758698
female    35.241302
Name: proportion, dtype: float64
```

What are the minimum, maximum, and average ages of passengers?

```
In [ ]: # min(), max(), and mean() find the minimum, maximum, and average values in the
        print("Minimum,Maximum and Average Ages of Passengers:")
        print("Minimum Age:", train_df['Age'].min())
        print("Maximum Age:", train_df['Age'].max())
        print("Average Age:", train_df['Age'].mean())
```

```
Minimum,Maximum and Average Ages of Passengers:
Minimum Age: 0.42
Maximum Age: 80.0
Average Age: 29.36158249158249
```

How many passengers embarked from each port (Embarked)?

```
In [ ]: # value_counts() counts the number of passengers from each embarkation port
        print("Passengers Embarked from each port:")
        print(train_df['Embarked'].value_counts())
```

```
Passengers Embarked from each port:
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

Select all female passengers

```
In [ ]: # Boolean indexing selects all rows where the 'Sex' column is 'female'
        print("All Female Passengers:")
        print(train_df[train_df['Sex']=='female'])
```

```
All Female Passengers:
     PassengerId  Survived  Pclass  \
1              2         1       1   
2              3         1       3   
3              4         1       1   
8              9         1       3   
9             10         1       2   
..           ...       ...     ...   
880          881         1       2   
882          883         0       3   
885          886         0       3   
887          888         1       1   
888          889         0       3   

                                                  Name     Sex   Age  SibSp  \
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   
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0   
9                  Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1   
..                                                 ...     ...   ...    ...   
880       Shelley, Mrs. William (Imanita Parrish Hall)  female  25.0      0   
882                       Dahlberg, Miss. Gerda Ulrika  female  22.0      0   
885               Rice, Mrs. William (Margaret Norton)  female  39.0      0   
887                       Graham, Miss. Margaret Edith  female  19.0      0   
888           Johnston, Miss. Catherine Helen "Carrie"  female  28.0      1   

     Parch            Ticket     Fare Cabin Embarked  
1        0          PC 17599  71.2833   C85        C  
2        0  STON/O2. 3101282   7.9250   NaN        S  
3        0            113803  53.1000  C123        S  
8        2            347742  11.1333   NaN        S  
9        0            237736  30.0708   NaN        C  
..     ...               ...      ...   ...      ...  
880      1            230433  26.0000   NaN        S  
882      0              7552  10.5167   NaN        S  
885      5            382652  29.1250   NaN        Q  
887      0            112053  30.0000   B42        S  
888      2        W./C. 6607  23.4500   NaN        S  

[314 rows x 12 columns]
```

Find all passengers who survived and were in first class

In [ ]:
```python
# Boolean indexing with multiple conditions to select survivors in first class
print("All Passengers who survived and were in first class:")
print(train_df[(train_df['Survived'] == 1) & (train_df['Pclass'] == 1)])
```

```
All Passengers who survived and were in first class:
     PassengerId  Survived  Pclass  \
1              2         1       1   
3              4         1       1   
11            12         1       1   
23            24         1       1   
31            32         1       1   
..           ...       ...     ...   
862          863         1       1   
871          872         1       1   
879          880         1       1   
887          888         1       1   
889          890         1       1   

                                                  Name     Sex   Age  SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1   
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1   
11                            Bonnell, Miss. Elizabeth  female  58.0      0   
23                        Sloper, Mr. William Thompson    male  28.0      0   
31      Spencer, Mrs. William Augustus (Marie Eugenie)  female  28.0      1   
..                                                 ...     ...   ...    ...   
862  Swift, Mrs. Frederick Joel (Margaret Welles Ba...  female  48.0      0   
871   Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0      1   
879      Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0      0   
887                       Graham, Miss. Margaret Edith  female  19.0      0   
889                              Behr, Mr. Karl Howell    male  26.0      0   

     Parch    Ticket      Fare Cabin Embarked  
1        0  PC 17599   71.2833   C85        C  
3        0    113803   53.1000  C123        S  
11       0    113783   26.5500  C103        S  
23       0    113788   35.5000    A6        S  
31       0  PC 17569  146.5208   B78        C  
..     ...       ...       ...   ...      ...  
862      0     17466   25.9292   D17        S  
871      1     11751   52.5542   D35        S  
879      1     11767   83.1583   C50        C  
887      0    112053   30.0000   B42        S  
889      0    111369   30.0000  C148        C  

[136 rows x 12 columns]
```

Find passengers aged between 20 and 30 years

In [18]:
```
# Boolean indexing to select passengers aged between 20 and 30
print("Passengers aged between 20 and 30:")
print(train_df[(train_df['Age'] >= 20) & (train_df['Age'] <= 30)])
```

```
Passengers aged between 20 and 30:
     PassengerId  Survived  Pclass  \
0              1         0       3
2              3         1       3
5              6         0       3
8              9         1       3
12            13         0       3
..           ...       ...     ...
883          884         0       2
884          885         0       3
886          887         0       2
888          889         0       3
889          890         1       1

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
5                                     Moran, Mr. James    male  28.0      0
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
12                      Saundercock, Mr. William Henry    male  20.0      0
..                                                 ...     ...   ...    ...
883                       Banfield, Mr. Frederick James    male  28.0      0
884                              Sutehall, Mr. Henry Jr    male  25.0      0
886                                Montvila, Rev. Juozas    male  27.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female  28.0      1
889                                Behr, Mr. Karl Howell    male  26.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
2        0  STON/O2. 3101282   7.9250   NaN        S
5        0            330877   8.4583   NaN        Q
8        2            347742  11.1333   NaN        S
12       0         A/5. 2151   8.0500   NaN        S
..     ...               ...      ...   ...      ...
883      0  C.A./SOTON 34068  10.5000   NaN        S
884      0   SOTON/OQ 392076   7.0500   NaN        S
886      0            211536  13.0000   NaN        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C

[422 rows x 12 columns]
```

Select only the Name, Age, and Survived columns for passengers over 60

In [ ]:
```
# Selects only the Name, Age, and Survived columns for passengers over 60
print("Name, Age and Survived columns of passengers over 60:")
print(train_df[train_df['Age']>=60][['Name', 'Age','Survived']])
```

```
Name, Age and Survived columns of passengers over 60:
                                          Name   Age  Survived
33                          Wheadon, Mr. Edward H  66.0         0
54                     Ostby, Mr. Engelhart Cornelius  65.0         0
96                         Goldschmidt, Mr. George B  71.0         0
116                           Connors, Mr. Patrick  70.5         0
170                         Van der hoef, Mr. Wyckoff  61.0         0
252                          Stead, Mr. William Thomas  62.0         0
275                  Andrews, Miss. Kornelia Theodosia  63.0         1
280                             Duane, Mr. Frank  65.0         0
326                         Nysveen, Mr. Johan Hansen  61.0         0
366  Warren, Mrs. Frank Manley (Anna Sophia Atkinson)  60.0         1
438                             Fortune, Mr. Mark  64.0         0
456                         Millet, Mr. Francis Davis  65.0         0
483                           Turkula, Mrs. (Hedwig)  63.0         1
493                          Artagaveytia, Mr. Ramon  71.0         0
545                      Nicholson, Mr. Arthur Ernest  64.0         0
555                            Wright, Mr. George  62.0         0
570                            Harris, Mr. George  62.0         1
587                    Frolicher-Stehli, Mr. Maxmillian  60.0         1
625                          Sutton, Mr. Frederick  61.0         0
630                 Barkworth, Mr. Algernon Henry Wilson  80.0         1
672                        Mitchell, Mr. Henry Michael  70.0         0
684                    Brown, Mr. Thomas William Solomon  60.0         0
694                              Weir, Col. John  60.0         0
745                      Crosby, Capt. Edward Gifford  70.0         0
829            Stone, Mrs. George Nelson (Martha Evelyn)  62.0         1
851                           Svensson, Mr. Johan  74.0         0
```

Create a new column 'FamilySize' by combining SibSp and Parch

In [ ]:
```python
# Create a new column 'FamilySize' by summing SibSp and Parch, then adding 1 (th
print("New column 'FamilySize':")
train_df['FamilySize'] = train_df['SibSp'] + train_df['Parch'] + 1
print(train_df[['Name', 'FamilySize']])
```

```
New column 'FamilySize':
                                               Name  FamilySize
0                           Braund, Mr. Owen Harris           2
1    Cumings, Mrs. John Bradley (Florence Briggs Th...           2
2                            Heikkinen, Miss. Laina           1
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)           2
4                          Allen, Mr. William Henry           1
..                                              ...         ...
886                           Montvila, Rev. Juozas           1
887                       Graham, Miss. Margaret Edith           1
888          Johnston, Miss. Catherine Helen "Carrie"           4
889                           Behr, Mr. Karl Howell           1
890                             Dooley, Mr. Patrick           1

[891 rows x 2 columns]
```

Create a categorical column 'AgeGroup' (child, adult, senior)

In [21]:
```python
# Define a function to categorize age and create a new 'AgeGroup' column
print("Age Group Column:")
def age_group(age):
    if age < 18:
        return 'Child'
    elif age < 60:
        return 'Adult'
    else:
        return 'Senior'
train_df['AgeGroup'] = train_df['Age'].apply(age_group)
print(train_df['AgeGroup'])
```

```
Age Group Column:
0      Adult
1      Adult
2      Adult
3      Adult
4      Adult
       ...
886    Adult
887    Adult
888    Adult
889    Adult
890    Adult
Name: AgeGroup, Length: 891, dtype: object
```

Convert the Sex column to numeric (1 for male, 0 for female)

In [22]:
```
# map() converts the 'Sex' column to numeric: 1 for male, 0 for female
print("Sex Column to Numeric:")
train_df['Sex'] = train_df['Sex'].map({'male': 1, 'female': 0})
print(train_df['Sex'])
print(train_df['Sex'].value_counts())
```

```
Sex Column to Numeric:
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 891, dtype: int64
Sex
1    577
0    314
Name: count, dtype: int64
```

Extract titles from names (Mr, Mrs, Miss, etc.) into a new column

In [23]:
```
# Extract titles from names using a custom function and create a new 'Title' col
print("Extracting Titles from Names:")
def extract_title(name):
    title = name.split(',')[1].split('.')[0]
    return title
train_df['Title'] = train_df['Name'].apply(extract_title)
print(train_df['Title'])
```

```
Extracting Titles from Names:
0          Mr
1         Mrs
2        Miss
3         Mrs
4          Mr
        ...
886       Rev
887      Miss
888      Miss
889        Mr
890        Mr
Name: Title, Length: 891, dtype: object
```

Calculate survival rates by passenger class

In [24]:
```
# groupby() groups data by 'Pclass' and calculates the mean survival rate for ea
print("Survival Rate by Passenger Class:")
survival_rate_by_class = train_df.groupby('Pclass')['Survived'].mean()*100
print(survival_rate_by_class)
```

```
Survival Rate by Passenger Class:
Pclass
1    62.962963
2    47.282609
3    24.236253
Name: Survived, dtype: float64
```

Find average age by passenger class and gender

In [ ]:
```
# groupby() groups by 'Pclass' and 'Sex', then mean() calculates the average age
print("Average age by passenger class and gender:")
average_age_by_class_gender = train_df.groupby(['Pclass','Sex'])['Age'].mean()
print(average_age_by_class_gender)
```

```
Average age by passenger class and gender:
Pclass  Sex
1       0      33.978723
        1      38.995246
2       0      28.703947
        1      30.512315
3       0      23.572917
        1      26.911873
Name: Age, dtype: float64
```

What was the survival rate by embarkation port?

In [ ]:
```
# groupby() groups by 'Embarked' and calculates the mean survival rate for each
print("Survival Rate by Embarked Port:")
survival_rate_by_embarked = train_df.groupby('Embarked')['Survived'].mean()*100
print(survival_rate_by_embarked)
```

```
Survival Rate by Embarked Port:
Embarked
C    55.357143
Q    38.961039
S    33.695652
Name: Survived, dtype: float64
```

Calculate family size distribution among survivors vs non-survivors

In [ ]:
```
# groupby() and value_counts() to analyze family size distribution among survivo
print("Family Size distribution among survivors and non-survivors:\n")
family_size_distribution = train_df.groupby( 'Survived')['FamilySize'].value_cou
print(family_size_distribution)
```

```
Family Size distribution among survivors and non-survivors:

FamilySize     1     2     3     4     5     6    7    8    11
Survived
0           374.0  72.0  43.0   8.0  12.0  19.0  8.0  6.0  7.0
1           163.0  89.0  59.0  21.0   3.0   3.0  4.0  NaN  NaN
```

Analyze survival rates by both class and gender

In [ ]:
```
# groupby() and mean() with unstack() to analyze survival rates by class and gen
print("Survival Rate by Class and Gender:")
survival_rate_by_class_gender = train_df.groupby(['Pclass','Sex'])['Survived'].m
print(survival_rate_by_class_gender)
```

```
Survival Rate by Class and Gender:
Sex            0         1
Pclass
1       0.968085  0.368852
2       0.921053  0.157407
3       0.500000  0.135447
```

Examine how age groups affected survival in different classes

In [ ]:
```
# groupby() and mean() with unstack() to analyze survival rates by class and age
print("Survival Rate by Class and Gender:")
survival_rate_by_class_age_group = train_df.groupby(['Pclass','AgeGroup'])['Surv
print(survival_rate_by_class_age_group)
```

```
Survival Rate by Class and Gender:
AgeGroup      Adult     Child    Senior
Pclass
1          0.641711  0.916667  0.294118
2          0.414013  0.913043  0.250000
3          0.218137  0.371795  0.200000
```

Explore how family size impacted survival rates

In [ ]:
```
# groupby() and mean() to analyze survival by family size
print("Survival by Family Size:")
survival_rate_by_family_size = train_df.groupby('FamilySize')['Survived'].mean()
print(survival_rate_by_family_size)
```

```
Survival by Family Size:
FamilySize
1     0.303538
2     0.552795
3     0.578431
4     0.724138
5     0.200000
6     0.136364
7     0.333333
8     0.000000
11    0.000000
Name: Survived, dtype: float64
```

Use pivot tables to analyze survival rates across multiple dimensions

In [ ]:
```
# pivot_table() summarizes survival rates by class, age group, and sex using mea
pivot_table = train_df.pivot_table(index=['Pclass','AgeGroup'],columns='Sex',val
print("Pivot Table of Survival by Class, Sex and AgeGroup:")
print(pivot_table)
```

```
Pivot Table of Survival by Class, Sex and AgeGroup:
Sex                     0         1
Pclass AgeGroup
1      Adult     0.975904  0.375000
       Child     0.875000  1.000000
       Senior    1.000000  0.142857
2      Adult     0.906250  0.075269
       Child     1.000000  0.818182
       Senior         NaN  0.250000
3      Adult     0.481481  0.123333
       Child     0.542857  0.232558
       Senior    1.000000  0.000000
```

In [ ]:
```
# Create a pivot table to show survival rates by passenger class and age group
# index='Pclass' sets the rows as passenger class
# columns='AgeGroup' sets the columns as age group (Child, Adult, Senior)
# values='Survived' uses the 'Survived' column to calculate the mean survival ra
# aggfunc='mean' calculates the average survival rate for each group
pivot_table = train_df.pivot_table(index='Pclass', columns='AgeGroup', values='S
print("Pivot Table of Average Survival Rate by Class and Age Group:")
print(pivot_table)
```

```
Pivot Table of Average Age by Class and Survival:
AgeGroup     Adult     Child    Senior
Pclass
1         0.641711  0.916667  0.294118
2         0.414013  0.913043  0.250000
3         0.218137  0.371795  0.200000
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```