```
import pandas as pd
import seaborn as ns
import matplotlib.pyplot

df_titanic = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')

print(df_titanic.info())

#Create a DataFrame called fare that contains only the Fare column of the Titanic dataset. Print the head of the result.
fare = df_titanic[['Fare']]
print(fare.head())

# Task 2: Create a DataFrame called 'class_age' containing 'Pclass' and 'Age' columns
class_head = df_titanic[['Pclass', 'Age']]
print(class_head.head())

# Task 3: Create a DataFrame called 'survived_gender' containing 'Survived' and 'Sex' columns
survived_gender = df_titanic[['Survived', 'Sex']]
print(survived_gender.head())
```

```
<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
      Fare
0   7.2500
1  71.2833
2   7.9250
3  53.1000
4   8.0500
   Pclass   Age
0       3  22.0
1       1  38.0
2       3  26.0
3       1  35.0
4       3  35.0
   Survived     Sex
0         0    male
1         1  female
2         1  female
3         1  female
4         0    male
```

```
import pandas as pd

titanic_df = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')

# Task 1: Filter for passengers with fare > 100
fare_gt_100 = titanic_df[titanic_df['Fare'] > 100]
print("Passenger with fare over 100:")
print(fare_gt_100)

# Task 2: Filter for passengers in Pclass = 1
first_class = titanic_df[titanic_df['Pclass'] == 1]
print("\nPassenger in first class:")
print(first_class)

# Task 3: Filter for passengers under 18 and female
female_under_18 = titanic_df[(titanic_df['Age'] < 18) & (titanic_df['Sex'] == 'female')]
print("\nFemale passengers under 18:")
print(female_under_18)

# Task 4: Filter for passengers whose Embarked port is "C" or "S"
```

```
embarked_c_or_s = titanic_df[titanic_df['Embarked'].isin(['C', 'S'])]
print("Passenger who embarked in C or S:")
print(embarked_c_or_s)

# Task 5: Filter for passengers in Pclass = 1 or 2
first_second_class = titanic_df[titanic_df['Pclass'].isin([1,2])]
print("\nPassenger in first or second class:")
print(first_second_class)
```

```
Passenger with fare over 100:
     PassengerId  Survived  Pclass  \
27            28         0       1
31            32         1       1
88            89         1       1
118          119         0       1
195          196         1       1
215          216         1       1
258          259         1       1
268          269         1       1
269          270         1       1
297          298         0       1
299          300         1       1
305          306         1       1
306          307         1       1
307          308         1       1
311          312         1       1
318          319         1       1
319          320         1       1
325          326         1       1
332          333         0       1
334          335         1       1
337          338         1       1
341          342         1       1
373          374         0       1
377          378         0       1
380          381         1       1
390          391         1       1
393          394         1       1
435          436         1       1
438          439         0       1
498          499         0       1
505          506         0       1
527          528         0       1
537          538         1       1
544          545         0       1
550          551         1       1
557          558         0       1
581          582         1       1
609          610         1       1
659          660         0       1
660          661         1       1
679          680         1       1
689          690         1       1
698          699         0       1
700          701         1       1
708          709         1       1
716          717         1       1
730          731         1       1
737          738         1       1
742          743         1       1
763          764         1       1
779          780         1       1
802          803         1       1
856          857         1       1

                                                Name     Sex    Age  SibSp  \
27                      Fortune, Mr. Charles Alexander    male  19.00      3
```

Which passenger had the highest fare paid relative to their age?

Generate randomly select 5 items from a list Close

```
import pandas as pd

titanic_df = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')


# Step 1: Handle missing values in the Age column by filling with the median
median_age = titanic_df['Age'].median()
titanic_df['Age'].fillna(median_age, inplace= True)
```

```
# Step 2: Add a column 'fare_per_year' containing Fare divided by Age
titanic_df['fare_per_age'] = titanic_df['Fare']/titanic_df['Age']


# Step 3: Subset rows where fare_per_year is higher than 5
high_fare_age = titanic_df[titanic_df['fare_per_age'] > 5]

# Step 4: Sort high_fare_age by descending fare_per_year
high_fare_age_srt = high_fare_age.sort_values(by='fare_per_age', ascending= False)

# Step 5: Select only the Name and fare_per_year columns
result = titanic_df[['Name', 'fare_per_age']]

# Step 6: Look at the result
print("Passenger(s) with the highest fare paid relative to their age (fare_per_year > 5):")
print(result)
```

```
Passenger(s) with the highest fare paid relative to their age (fare_per_year > 5):
                                                  Name  fare_per_age
0                              Braund, Mr. Owen Harris      0.329545
1    Cumings, Mrs. John Bradley (Florence Briggs Th...      1.875876
2                               Heikkinen, Miss. Laina      0.304808
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)      1.517143
4                             Allen, Mr. William Henry      0.230000
..                                                 ...           ...
886                              Montvila, Rev. Juozas      0.481481
887                       Graham, Miss. Margaret Edith      1.578947
888           Johnston, Miss. Catherine Helen "Carrie"      0.837500
889                              Behr, Mr. Karl Howell      1.153846
890                                Dooley, Mr. Patrick      0.242188

[891 rows x 2 columns]
<ipython-input-9-6aae7534f575>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  titanic_df['Age'].fillna(median_age, inplace= True)
```

Which adult male passenger (age ≥ 18 and Sex is 'male') paid the highest fare relative to their class?

```
import pandas as pd

titanic_df = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')

# Step 1: Add a column 'fare_per_class' containing Fare divided by Pclass
titanic_df['fare_per_class'] = titanic_df['Fare']/titanic_df['Pclass']

# Step 2: Subset rows where the passenger is male and an adult (Age >= 18)
# Handle missing Age values by filling them with the median
median_age = titanic_df['Age'].median()
titanic_df['Age'].fillna(median_age, inplace= True)
adult_males = titanic_df[(titanic_df['Age'] > 18) & (titanic_df['Sex'] == 'male')]

# Step 3: Sort adult males by descending fare_per_class
adult_male_srt = adult_males.sort_values(by='fare_per_class', ascending =  False)

# Step 4: Select only the Name, Age, and fare_per_class columns
result = adult_male_srt[['Name', 'Age', 'fare_per_class']]

# Step 6: Look at the result
print("Adult male(s) with the highest fare paid relative to their class:")
print(result)
```

```
Adult male(s) with the highest fare paid relative to their class:
                                  Name   Age  fare_per_class
679  Cardeza, Mr. Thomas Drake Martinez  36.0        512.3292
737              Lesurer, Mr. Gustave J  35.0        512.3292
438                   Fortune, Mr. Mark  64.0        263.0000
27       Fortune, Mr. Charles Alexander  19.0        263.0000
118            Baxter, Mr. Quigg Edmond  24.0        247.5208
..                                  ...   ...             ...
302     Johnson, Mr. William Cahoone Jr  19.0          0.0000
633       Parr, Mr. William Henry Marsh  28.0          0.0000
```

```
674          Watson, Mr. Ennis Hastings  28.0          0.0000
732                Knight, Mr. Robert J  28.0          0.0000
263               Harrison, Mr. William  40.0          0.0000

[506 rows x 3 columns]
<ipython-input-10-e23eae7c52d8>:11: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assign
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  titanic_df['Age'].fillna(median_age, inplace= True)
```

What percent of the total fare revenue came from each passenger class?

```
import pandas as pd

titanic_df = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')

# Step 1: Calculate the total fare paid across all passengers
total_fare = titanic_df['Fare'].sum()

# Step 2: Subset for first-class passengers and calculate their total fare
first_class_fare = titanic_df[titanic_df['Pclass'] == 1]['Fare'].sum()
second_class_fare = titanic_df[titanic_df['Pclass'] == 2]['Fare'].sum()
third_class_fare = titanic_df[titanic_df['Pclass'] == 3]['Fare'].sum()

# Step 5: Combine the fare totals into a list
class_fare_total = [first_class_fare, second_class_fare, third_class_fare]

# Step 6: Calculate the proportion of fare revenue by class
fare_revenue_percentage = [(fare/total_fare) * 100 for fare in class_fare_total]

# Display the results
print("Percentage of total fare revenue by passenger class:")
print(f"First calss: {fare_revenue_percentage[0]:.2f}%")
print(f"Second class: {fare_revenue_percentage[1]:.2f}%")
print(f"Third Class: {fare_revenue_percentage[2]:.2f}%")
```

```
Percentage of total fare revenue by passenger class:
First calss: 63.35%
Second class: 13.25%
Third Class: 23.40%
```

What percent of the total number of passengers on the Titanic belonged to each age group (e.g., child, adult, senior)?

```
import pandas as pd

titanic_df = pd.read_csv('/content/drive/MyDrive/Concept of AI -- week 3/Copy of Copy of Copy of Titanic-Dataset.csv')

# Step 1: Handle missing Age values and create the 'age_group' column
median_age = titanic_df['Age'].median()
titanic_df['Age'].fillna(median_age, inplace = True)

# Categorize passengers into age groups
def catogorize_age(age):
  if age < 18:
    return 'child'
  elif age < 65:
    return 'adult'
  else:
    return 'senior'

titanic_df['age_group'] = titanic_df['Age'].apply(catogorize_age)


# Step 2: Calculate the total number of passengers
total_passenger = len(titanic_df)

# Step 3: Count the number of passengers in each age group
age_group_counts = titanic_df['age_group'].value_counts()

# Step 4: Calculate the proportion of passengers in each age group
age_group_proportion = (age_group_counts/total_passenger) * 100
```

```
print("Percentage of passengers in each age group:")
for group, percentage in age_group_proportion.items():
  print(f"{group} : {percentage:.2f}%")
```

```
Percentage of passengers in each age group:
adult : 86.08%
child : 12.68%
senior : 1.23%
<ipython-input-15-90d0fa3dc1c2>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  titanic_df['Age'].fillna(median_age, inplace = True)
```

Start coding or generate with AI.