[1] ✓ 0s

```
import pandas as pd
import numpy as np
```

Load dataset(replace file name if needed)

[2] ✓ 1s

```
df=pd.read_csv("/content/drive/MyDrive/Concepts and technology of AI/Titanic-Dataset.csv")
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Next steps: Generate code with df | New interactive sheet

Next steps: Generate code with df | New interactive sheet

[19] ✓ 0s

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 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
 12  age_group    891 non-null    object
dtypes: float64(2), int64(5), object(6)
memory usage: 90.6+ KB
```

3.1

[4] ✓ 0s

```
fare = df[['Fare']]
fare.head()
```

	Fare
0	7.2500
1	71.2833
2	7.9250
3	53.1000
4	8.0500

3. Survived and Sex columns

```
survived_gender = df[['Survived', 'Sex']]
survived_gender.head()
```

	Survived	Sex
0	0	male
1	1	female
2	1	female
3	1	female
4	0	male

Next steps: Generate code with survived_gender | New interactive sheet

Problem 2 — Subsetting Rows

```
fare_gt_100 = df[df["Fare"] > 100]
fare_gt_100.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S	adult
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	146.5208	B78	C	senior
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	S	adult
118	119	0	1	Baxter, Mr. Quigg Edmond	male	24.0	0	1	PC 17558	247.5208	B58 B60	C	adult
195	196	1	1	Lurette, Miss. Elise	female	58.0	0	0	PC 17569	146.5208	B80	C	adult

2. Pclass = 1

```
first_class = df[df["Pclass"] == 1]
first_class.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	adult
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	adult
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	adult
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S	adult
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S	adult

Next steps: Generate code with first_class | New interactive sheet

3. Females under 18

```
female_under_18 = df[(df["Age"] < 18) & (df["Sex"] == "female")]
female_under_18.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	child
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S	child
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S	child
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q	child
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S	child

Next steps: Generate code with female_under_18 | New interactive sheet

Subsetting by Categorical

```
[24]
embarked_c_or_s = df[df["Embarked"].isin(["C", "S"])]
embarked_c_or_s.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	adult
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	adult
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	adult
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	adult
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	adult

Next steps: Generate code with embarked_c_or_s | New interactive sheet

2. Pclass in [1, 2]

```
[25]
first_second_class = df[df["Pclass"].isin([1, 2])]
first_second_class.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	adult
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	adult
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	adult
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	child
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S	adult

Next steps: Generate code with first_second_class | New interactive sheet

3.2 — EDA PRACTICE EXERCISE – 1

```
[26]
df["Age"].fillna(df["Age"].median(), inplace=True)
```

```
/tmp/ipython-input-2097741607.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df["Age"].fillna(df["Age"].median(), inplace=True)
```

```
[27]
df["fare_per_year"] = df["Fare"] / df["Age"]
```

Subset fare_per_year > 5

```
[28]
high_fare_age = df[df["fare_per_year"] > 5]
```

Sort descending by fare_per_year

```
[29]
high_fare_age_srt = high_fare_age.sort_values(by="fare_per_year", ascending=False)
```

Select name + fare_per_year

```
[30]
result = high_fare_age_srt[["Name", "fare_per_year"]]
result.head()
```

	Name	fare_per_year
305	Allison, Master. Hudson Trevor	164.728261
297	Allison, Miss. Helen Loraine	75.775000
386	Goodwin, Master. Sidney Leonard	46.900000
164	Panula, Master. Eino Viljami	39.687500
183	Becker, Master. Richard F	39.000000

(B) Adult male passenger with highest fare/class

Add Fare/Pclass column

```
[31] ✓ 0s
df["fare_per_class"] = df["Fare"] / df["Pclass"]
```

Subset adult males

```
[32] ✓ 0s
adult_males = df[(df["Sex"] == "male") & (df["Age"] >= 18)]
adult_males
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	age_group	fare_per_year	fare_per_class
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	adult	0.329545	2.416667
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	adult	0.230000	2.683333
5	6	0	3	Moran, Mr. James	male	28.0	0	0	330877	8.4583	NaN	Q	senior	0.302082	2.819433
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	adult	0.960417	51.862500
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S	adult	0.402500	2.683333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S	adult	0.375000	5.250000
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S	adult	0.282000	2.350000
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	adult	0.481481	6.500000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	adult	1.153846	30.000000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	adult	0.242188	2.583333

519 rows × 15 columns

3. Sort descending

```
[33] ✓ 0s
adult_males_srt = adult_males.sort_values(by="fare_per_class", ascending=False)

result_male = adult_males_srt[["Name", "Age", "fare_per_class"]]
result_male.head()
```

	Name	Age	fare_per_class
737	Lesurer, Mr. Gustave J	35.0	512.3292
679	Cardeza, Mr. Thomas Drake Martinez	36.0	512.3292
27	Fortune, Mr. Charles Alexander	19.0	263.0000
438	Fortune, Mr. Mark	64.0	263.0000
118	Baxter, Mr. Quigg Edmond	24.0	247.5208

Next steps: Generate code with result_male | New interactive sheet

3.3(A) % of total fare revenue by class

Total fare of all passengers

```
[16] ✓ 0s
total_fare = df["Fare"].sum()
```

Fare totals for each class

```
[10] ✓ 0s
fare_class1 = df[df["Pclass"] == 1]["Fare"].sum()
fare_class2 = df[df["Pclass"] == 2]["Fare"].sum()
fare_class3 = df[df["Pclass"] == 3]["Fare"].sum()

fare_list = [fare_class1, fare_class2, fare_class3]
fare_list
```

```
[np.float64(18177.4125), np.float64(3801.8417), np.float64(6714.6951)]
```

Percent from each class

[11] ✓ 0s

```
percentage = [f / total_fare * 100 for f in fare_list]
percentage
```

```
[np.float64(63.349287718996564),
 np.float64(13.24962855496507),
 np.float64(23.401083726038365)]
```

b.1. Create age group column

[12] ✓ 0s

```
def age_group(age):
    if age < 18:
        return "child"
    elif age < 65:
        return "adult"
    else:
        return "senior"

df["age_group"] = df["Age"].apply(age_group)
print(df[['Name','Age','age_group']].head())
```

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

total passenger

[13] ✓ 0s

```
total_passengers = len(df)
total_passengers
```

```
891
```

Count per group

[14]
✓ 0s

```
group_counts = df["age_group"].value_counts()
group_counts
```

age_group	count
adult	590
senior	188
child	113

dtype: int64

Percentage

[15]
✓ 0s

```
group_percent = (group_counts / total_passengers) * 100
group_percent
```

age_group	count
adult	66.217733
senior	21.099888
child	12.682379

dtype: float64