Bit Project

Introduction to Pandas

Module 2

What is Pandas?

Pandas is an open source data analysis and manipulation tool
It is:

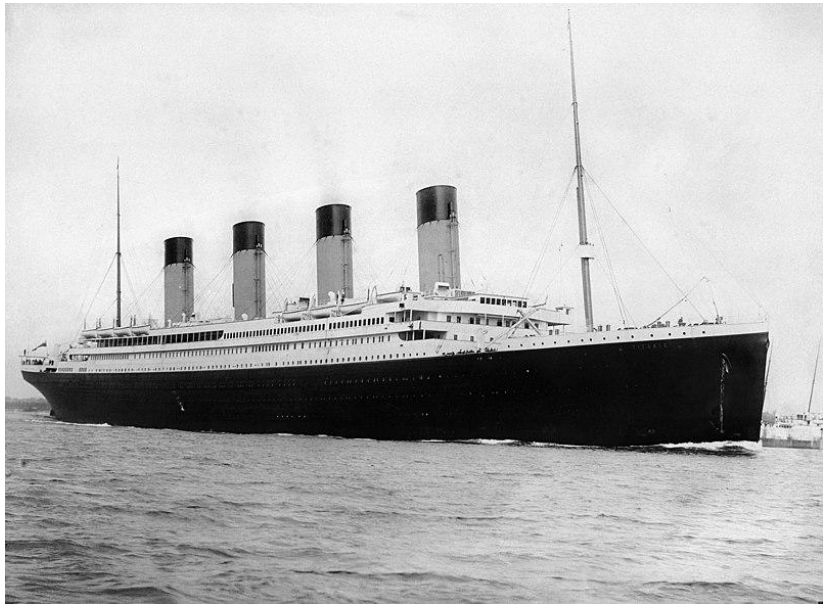
- Built on Python
- Used widely in both academia and industry
- Multipurpose

What is Pandas? pt.2

Within Pandas, there are 3 types of data structures
1: Series ~ 1-dimensional array
2: Dataframe ~ 2-dimensional data structure - basically your standard data table
3: Panel ~ 3-dimensional data structure

Day	Container A: Water Only	Container B: Water plus Fertilizer
1	2.0	2.0
2	2.2	2.3
3	2.3	2.8
4	2.5	3.2
5	2.6	3.8

The dataset in this module



Getting Started

Step 1: Import Pandas library

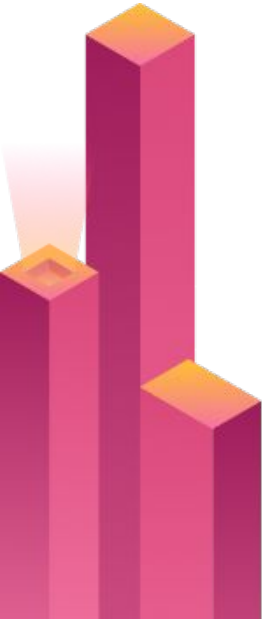
This lets us access Pandas using the name “pd”

```
import pandas as pd
```

Step 2: Load your dataset

```
url = 'https://exampleurl.com/blah'

df = pd.read_csv(url)
```



We assign the url of our dataset to “url”, then use pd.read_csv to store the data in a dataframe

Voila!

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

891 rows × 12 columns

Collecting Basic Info

head()/tail()

```
[ ] df.head()
```

	PassengerId	Survived	Pclass	
0	1	0	3	Braund, Mr. Owen
1	2	1	1	Cumings, Mrs. John Bradley (Florence
2	3	1	3	Heikkinen, Miss.
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May
4	5	0	3	

```
[ ] df.tail()
```

	PassengerId	Survived	Pclass	
886	887	0	2	Montvila, Rev. J
887	888	1	1	Graham, Miss. Margaret
888	889	0	3	Johnston, Miss. Catherine Helen "C
889	890	1	1	Behr, Mr. Karl H
890	891	0	3	Dooley, Mr. P

info()

```
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
```

WOW!

COOL!!!

describe()

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	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	20.125000	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	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

shape()

```
[ ]  df.shape

     (891, 12)
```

(# of rows, # of columns)

Working with Indexes

Setting an index

```
[ ]  df.set_index(keys='PassengerId', inplace=True)
```

```
[ ]  df
```

PassengerId	Survived	Pclass	Name	Sex	Ag
1	0	3	Braund, Mr. Owen Harris	male	22.
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.
3	1	3	Heikkinen, Miss. Laina	female	26.
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.
5	0	3	Allen, Mr. William Henry	male	35.
...	...	...	...	...	.
887	0	2	Montvila, Rev. Juozas	male	27.
888	1	1	Graham, Miss. Margaret Edith	female	19.
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	Nal
890	1	1	Behr, Mr. Karl Howell	male	26.
891	0	3	Dooley, Mr. Patrick	male	32.

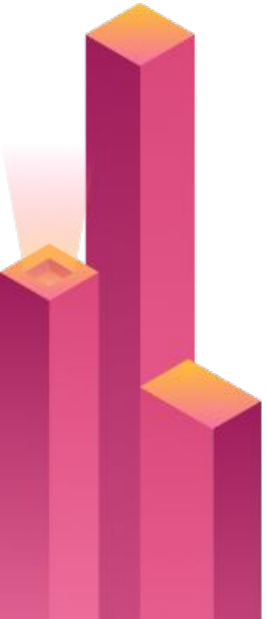
891 rows × 11 columns

Indexing

What is indexing?

Indexing refers to the selection of subsets of data

Think back to indexing from Module 1



There are two ways to do this:

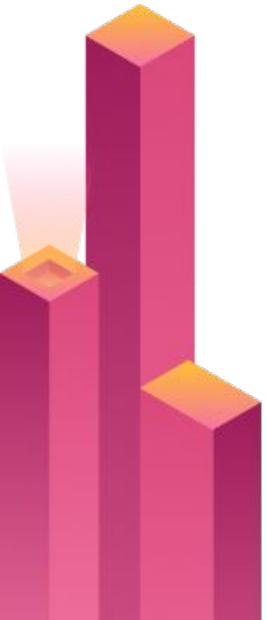
- 1. Selection by index numbers
- 2. Selection by column names

1. Selection by index #s

```
[ ]  df.iloc[: , [0,1,2]]
```

PassengerId	Survived	Pclass	Name
1	0	3	Braund, Mr. Owen Harris
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
3	1	3	Heikkinen, Miss. Laina
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
5	0	3	Allen, Mr. William Henry
...	...	...	...
887	0	2	Montvila, Rev. Juozas
888	1	1	Graham, Miss. Margaret Edith
889	0	3	Johnston, Miss. Catherine Helen "Carrie"
890	1	1	Behr, Mr. Karl Howell
891	0	3	Dooley, Mr. Patrick

891 rows × 3 columns

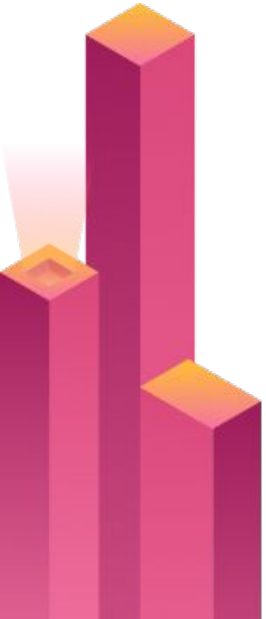


2. Selection by column names

```
[ ]  df[['Survived', 'Sex', 'Age']]
```

PassengerId	Survived	Sex	Age
1	0	male	22.0
2	1	female	38.0
3	1	female	26.0
4	1	female	35.0
5	0	male	35.0
...	...	...	...
887	0	male	27.0
888	1	female	19.0
889	0	female	NaN
890	1	male	26.0
891	0	male	32.0

891 rows × 3 columns



Dropping / Removing data

Dropping rows

Method 1: `df.drop(index #)`

```
[ ]  df.drop(3, inplace=True)
```

```
[ ]  df
```

PassengerId	Survived	Pclass	Name	Sex	Age	Sib
1	0	3	Braund, Mr. Owen Harris	male	22.0	
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	
5	0	3	Allen, Mr. William Henry	male	35.0	
6	0	3	Moran, Mr. James	male	NaN	

Dropping rows

Method 2:

```
df.drop(df[<some boolean condition>].index)
```

```
[7]  df.drop(df[df['Fare'] == 0].index)
```

PassengerId	Survived	Pclass	Name
1	0	3	Braund, Mr. Owen Harris
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th..
3	1	3	Heikkinen, Miss. Laina
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel
5	0	3	Allen, Mr. William Henry
...	...	...	..
887	0	2	Montvila, Rev. Juozas
888	1	1	Graham, Miss. Margaret Edith
889	0	3	Johnston, Miss. Catherine Helen "Carrie"
890	1	1	Behr, Mr. Karl Howell
891	0	3	Dooley, Mr. Patrick

876 rows × 11 columns

Dropping columns

Method 1:

```
df.drop(columns=[column_name])
```

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

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Q
...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

890 rows × 10 columns

Dropping columns

Method 2: `del df[column_name])`

```
[ ] del df['Cabin']
```

```
[ ] df
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Q
...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

890 rows × 10 columns

Dropping NaN values

NaN = null values
Dropping Nan values = dropping rows / data with missing information (aka null values) in one or more rows

3 ways:

- `df.dropna()`: drop the rows where at least one of the elements is missing.
- `df.dropna(how='all')`: drop the rows where all of the elements are missing.
- `df.dropna(subset=[columns])`: define which columns to look for missing values and then drop

Dropping NaN values

1

```
[ ]  df.dropna()
```

PassengerId	Survived	P
1	0	
2	1	
4	1	
5	0	
7	0	
...	...	
886	0	
887	0	
888	1	
890	1	
891	0	

711 rows × 10 columns

2

```
[ ]  df.dropna(how='all')
```

PassengerId	Survived	P
1	0	
2	1	
4	1	
5	0	
6	0	
...	...	
887	0	
888	1	
889	0	
890	1	
891	0	

890 rows × 10 columns

3

```
[ ]  df.dropna(subset=['Age'])
```

PassengerId	Survived	Pclass
1	0	3
2	1	1
4	1	1
5	0	3
7	0	1
...	...	...
886	0	3
887	0	2
888	1	1
890	1	1
891	0	3

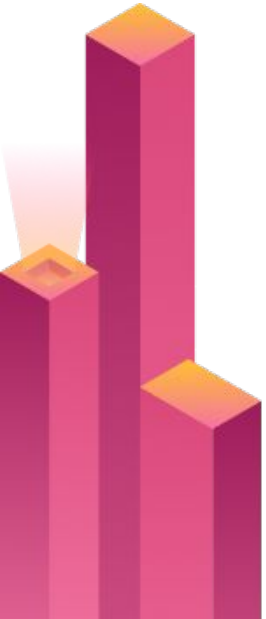
713 rows × 10 columns

Unique Values

Getting unique values

```
[ ]  df[['Parch']].drop_duplicates()
```

PassengerId	Parch
1	0
8	1
9	2
14	5
87	3
168	4
679	6

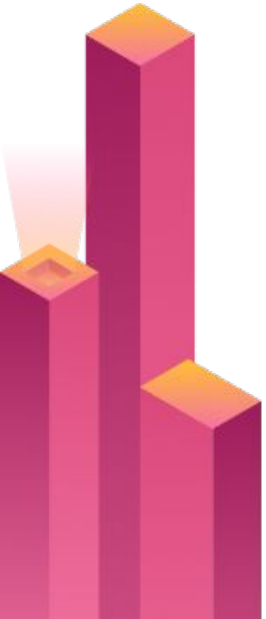


Shows the first instance of each unique value and range of “Parch”

Counting unique values

```
[ ]  df.groupby(['Embarked']).size()
```

```
Embarked
C    168
Q     77
S    643
dtype: int64
```



Use groupby() and size() functions to count number of instances for each unique value

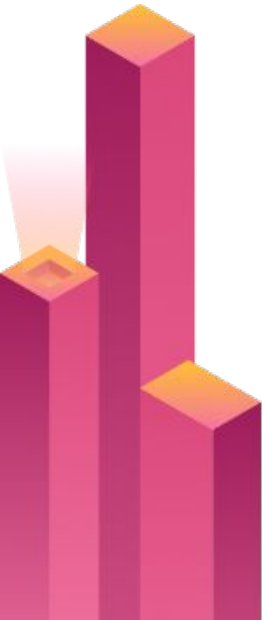
Also, can tack sort_values() on end like so:
`df.groupby(['Embarked']).size().sort_values()`
To have data sorted from least to greatest

Number of unique values

```
[ ]  df["Age"].nunique()
```

```
88
```

88 unique values in Age column!



Custom Selections

Selecting rows based on criteria

To select rows based on your own custom criteria, there are two methods you can use:

1. `df.where(condition)`
2. `df.loc[condition]`

First method fills all inapplicable rows with NaN
Second method just drops all inapplicable rows

Selecting rows based on criteria

Method 1: `df.where(condition)`

```
[ ] filter = df['Age'] >= 30
    older_than_30 = df.where(filter)
```

```
[ ] older_than_30
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1.0	1.0	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1.0	0.0	PC 17599	71.2833	C
4	1.0	1.0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1.0	0.0	113803	53.1000	S
5	0.0	3.0	Allen, Mr. William Henry	male	35.0	0.0	0.0	373450	8.0500	S
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
887	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
888	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
890	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
891	0.0	3.0	Dooley, Mr. Patrick	male	32.0	0.0	0.0	370376	7.7500	Q

890 rows × 10 columns

Selecting rows based on criteria

Method 2: `df.loc[condition]`

```
[ ]  filter = df['Age'] >= 30
     df.loc[filter]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	S
12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	S
...	...	...	...	...	...	...	...	...	...	...
874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	S
880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C
882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	S
886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	Q
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

330 rows × 10 columns

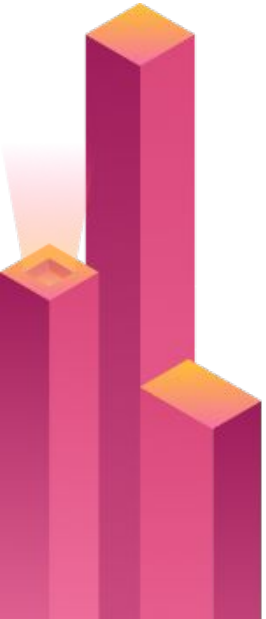
Basic Analysis

Aggregate Functions

To do basic analysis on the dataset, we will be using aggregate functions

These functions summarize and show us overall properties of the dataset

Ex: `sum()`, `count()`, `min()`, `max()`, etc.



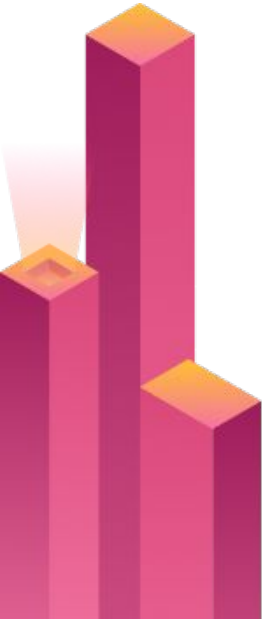
sum()

```
[ ]  df['Fare'].sum()
```

```
28686.024299999997
```

```
[ ]  total_fares = df['Fare'].sum()
     print(total_fares)
```

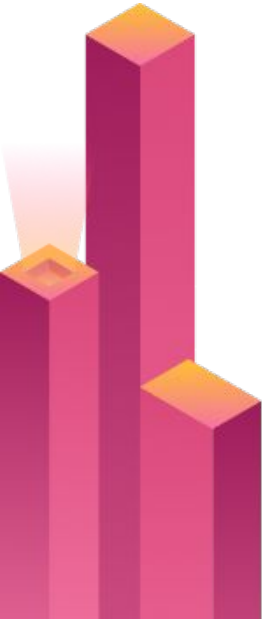
```
28686.024299999997
```



sum() example

```
[ ]  survived_passengers = df['Survived'].sum()
     survived_passengers
```

```
341
```



In this example, we use sum() to figure out that 341 passengers survived out of the dataset of 890 passengers.

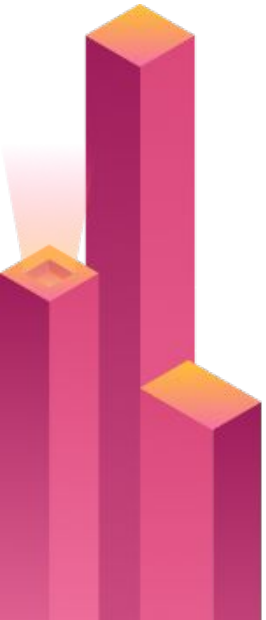
max() / min()

```
[ ]  df['Fare'].max()
```

```
512.3292
```

```
[ ]  df['Fare'].min()
```

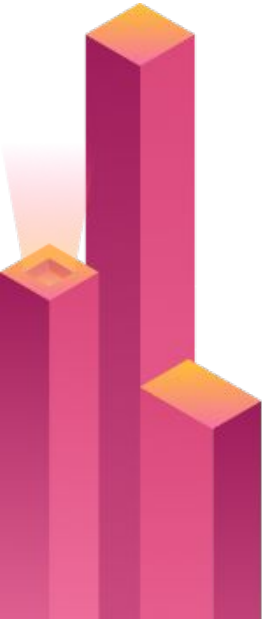
```
0.0
```



mean()

```
[ ]  df['Survived'].mean()
```

```
0.3831460674157303
```



Here, we used mean() to figure out the survival rate in the dataset, 38.3%.

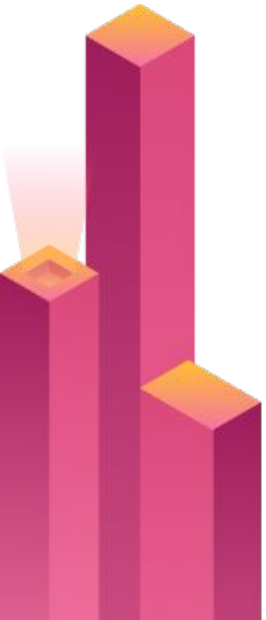
max() / min()

```
[ ]  df['Fare'].max()
```

```
512.3292
```

```
[ ]  df['Fare'].min()
```

```
0.0
```



Ex: Survival rate by age group

Let’s use skills we learned earlier and these aggregate functions to tackle a more complex problem.

Step 1: Filter / select dataset based on criteria

```
[ ]  filter_over_30 = df['Age'] >= 30
     filter_under_30 = df['Age'] < 30

     df_over_30yrs = df.loc[filter_over_30]
     df_under_30yrs = df.loc[filter_under_30]
```

Ex: Survival rate by age group

Step 2: Find survival rate for each new dataset

```
[ ]  df_over_30yrs['Survived'].mean()
```

```
0.406060606060607
```

```
[ ]  df_under_30yrs['Survived'].mean()
```

```
0.4046997389033943
```

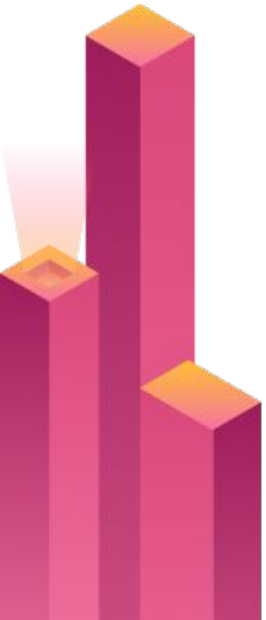
What can we conclude?

groupby()

Use groupby in this format:
`df.groupby(['column_name']).aggregate_function`
to use aggregate functions based on groups based on a chosen a column

```
[ ]  df.groupby(['Sex']).mean()
```

Sex	Survived	Pclass	Age	SibSp	Parch	Fare
female	0.741214	2.156550	27.923077	0.696486	0.651757	44.596606
male	0.188908	2.389948	30.726645	0.429809	0.235702	25.523893



Takeaways + Ending Notes

Takeaways

In this module, we learned to:

- Set indexes
- Index
- Drop specific rows / columns
- Drop NaN or null values
- Analyze unique values
- Select rows/data based on criteria
- Analyze using aggregate functions
-

Through all of these methods, we can analyze our data!

Additional Resources

Colab Notebook

Pandas Cheatsheet

Glossary

Notebook Walkthrough

Page w/ more helpful resources!

Also, as always, don’t hesitate to ask for help!

Check us out!

@bitprojectorg

@bitprj

@BitProject

@BitProject

Thank you!