

Module 2: Exploratory Data Analysis

Part 2: Two Dimensional Data

The Data Frame

```
df = pd.DataFrame( <data>, <index>, <column_names> )
```

CSV

```
df = pd.read_csv( <file> )
```

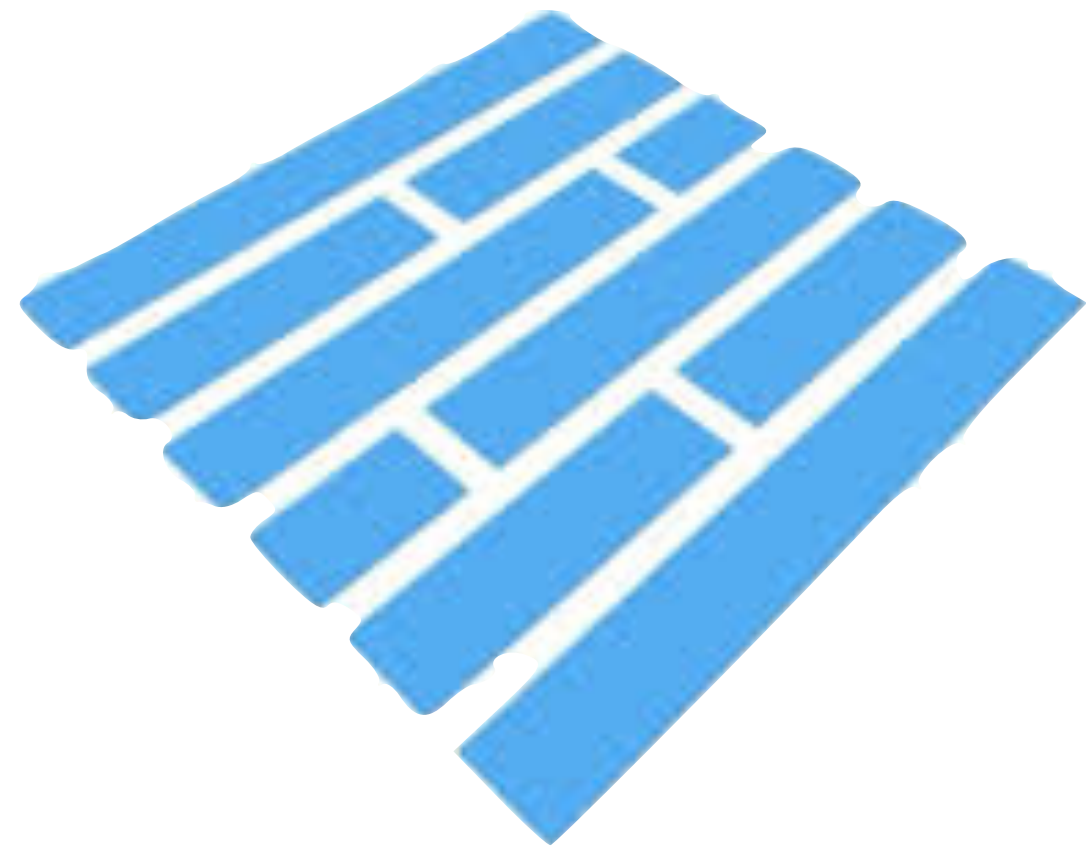
```
df = pd.read_csv( <url> )
```

Excel

```
df = pd.read_excel( <file>, sheetname=<sheetname> )
```

From a Database

```
df = pd.read_sql( <query>, <connection> )
```



Parquet

```
df = pd.read_parquet( <file> )
```

XML

```
import requests
```

```
user_agent_url = 'http://www.user-agents.org/allagents.xml'  
xml_data = requests.get(user_agent_url).content
```

<http://www.austintaylor.io/lxml/python/pandas/xml/dataframe/2016/07/08/convert-xml-to-pandas-dataframe/>

Web Server Logs



Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/  
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://  
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)  
Gecko/20100101 Firefox/35.0"
```

Web Server Logs

```
195.154.46.135 - - [25/Oct/2015:04:11:25 +0100] "GET /linux/  
doing-pxe-without-dhcp-control HTTP/1.1" 200 24323 "http://  
howto.basjes.nl/" "Mozilla/5.0 (Windows NT 5.1; rv:35.0)  
Gecko/20100101 Firefox/35.0"
```

```
from apachelogs import LogParser  
line_parser = LogParser("%h %l %u %t \"%r\" %>s %b  
\"{Referer}i\" \"%{User-agent}i\"")
```

Web Server Logs

request_first_line	request_header_referer	request_header_user_agent	request_http_ver	request_method	request_url
GET /linux/doing-pxe-without-dhcp-control HTTP/1.1	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	GET	/linux/doing-pxe-v
GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.0	GET	/join_form
POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 5.1; rv:35.0) Gecko/20...	1.1	POST	/join_form
GET /join_form HTTP/1.0	http://howto.basjes.nl/	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.0	GET	/join_form
POST /join_form HTTP/1.1	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	POST	/join_form
GET /acl_users/credentials_cookie_auth/require...	http://howto.basjes.nl/join_form	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:34.0) G...	1.1	GET	/acl_users/credent

Nested Data?



Nested Data?

```
{ "time": 1084443427.311224,  
  "timestamp": "2004-05-13T10:17:07.311224",  
  "IP": {  
    "version": 4,  
    "ttl": 128,  
    "proto": 6,  
    "options": [],  
    "len": 48,  
    "dst": "65.208.228.223",  
    "frag": 0,  
    "flags": 2, "src": "145.254.160.237",  
    "chksum": 37355  
  },  
  "Ethernet": { "src": "00:00:01:00:00:00", "type": 2048, "dst": "fe:ff:20:00:01:00" },  
  ...  
}
```

Nested Data

```
pd.read_json( 'nested_data.json' )
```

```
pd.read_json( 'nested_data.json' )
```

	DNS	Ethernet	IP	TCP	UDP	time	timestamp
0	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 48, 'version'...	{'window': 8760, 'chksum': 49932, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:07.311224
1	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 48, 'version'...	{'window': 5840, 'chksum': 23516, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
2	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 40, 'version'...	{'window': 9660, 'chksum': 31076, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
3	NaN	{'type': 2048, 'dst': 'fe:ff:20:00:01:00', 'sr...	{'dst': '65.208.228.223', 'len': 519, 'version'...	{'window': 9660, 'chksum': 43352, 'sport': 337...	NaN	1.084443e+09	2004-05-13 10:17:08.222534
4	NaN	{'type': 2048, 'dst': '00:00:01:00:00:00', 'sr...	{'dst': '145.254.160.237', 'len': 40, 'version'...	{'window': 6432, 'chksum': 33825, 'sport': 80,...	NaN	1.084443e+09	2004-05-13 10:17:08.783340

Nested Data

```
from pandas import json_normalize
import json
import pandas as pd

with open('nested.json') as data_file:
    pcap_data = json.load(data_file)

df = pd.DataFrame(json_normalize(pcap_data))
```

```
df = pd.DataFrame( json_normalize(pcap_data) )
```

...	TCP.seq	TCP.sport	TCP.urgptr	TCP.window	L
...	951057939.0	3372.0	0.0	8760.0	
...	290218379.0	80.0	0.0	5840.0	
...	951057940.0	3372.0	0.0	9660.0	
...	951057940.0	3372.0	0.0	9660.0	
...	290218380.0	80.0	0.0	6432.0	

Stop

Manipulating a DataFrame

Creating a new Column

```
df[ 'new_col' ] = df[ 'column1' ] + 3
```

Two Ways of Accessing Columns

```
my_series = df.loc[:, 'column1']
```



Don't use the dots!

```
series = df.column1
```

```
my_series = df.loc[:, 'column1']
```

Returns a **series**

```
df = df[['column1', 'column2', 'column3']]
```

Returns a DataFrame

Filtering a DataFrame

```
df[<boolean condition>]
```

```
df[['col1', 'col2']][df['col3'] > 5]
```

↑
Columns

↑
Filter

Questions?

In Class Exercise Back at xx:xx

Please complete Exercise 5 in Worksheet 2.1: Exploring Two Dimensional Data

Start on Worksheet 2.2

Get Oriented

```
#set the axis to apply the sum  
data.sum( axis='index' )
```

OR

```
data.sum( axis='columns' )
```

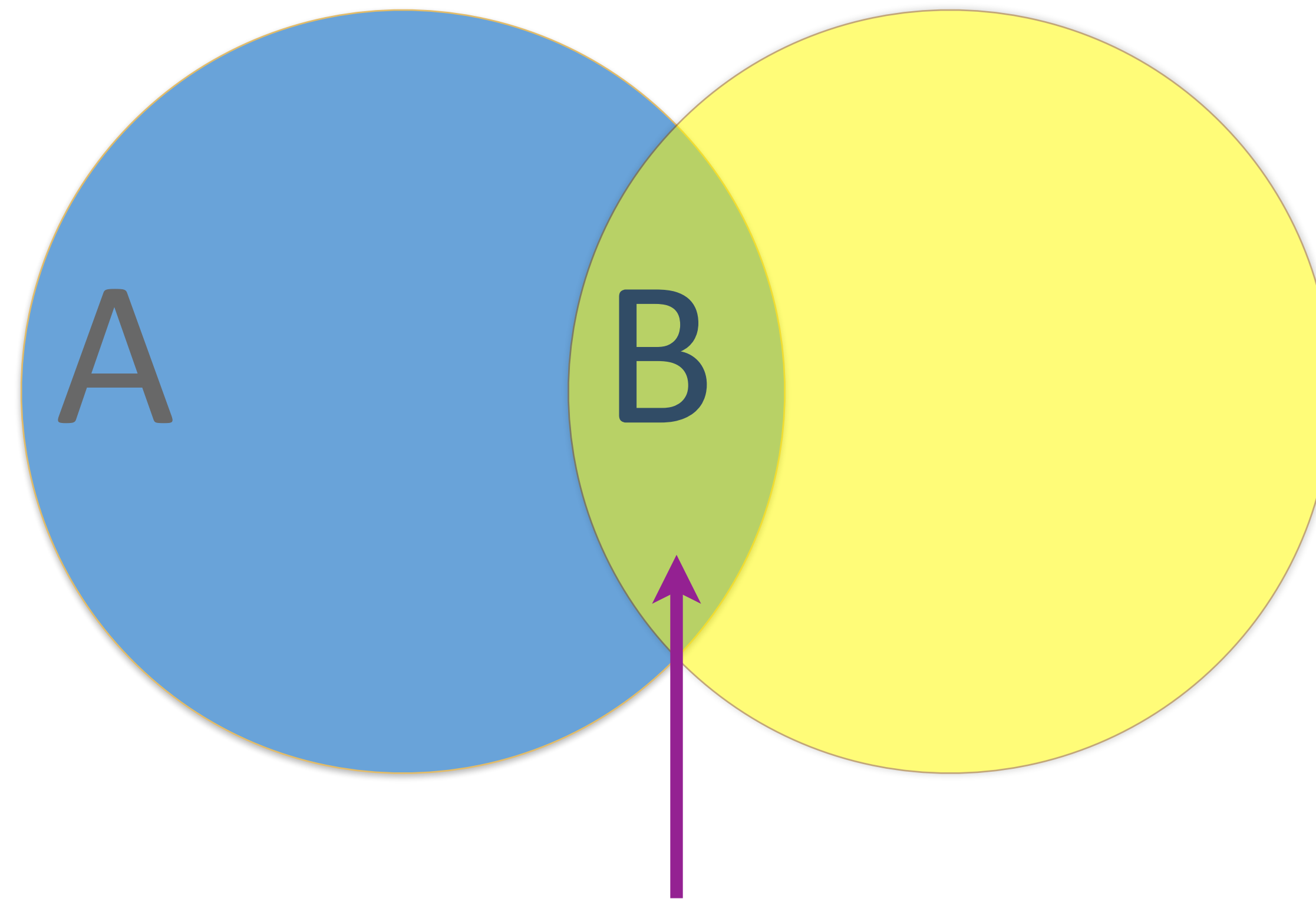
```
DataFrame.drop(labels,  
                axis=0,  
                level=None,  
                in_place=False,  
                errors='raise')
```

Merging Data Sets

```
combinedSeries = pd.concat(  
    [series1, series2])
```

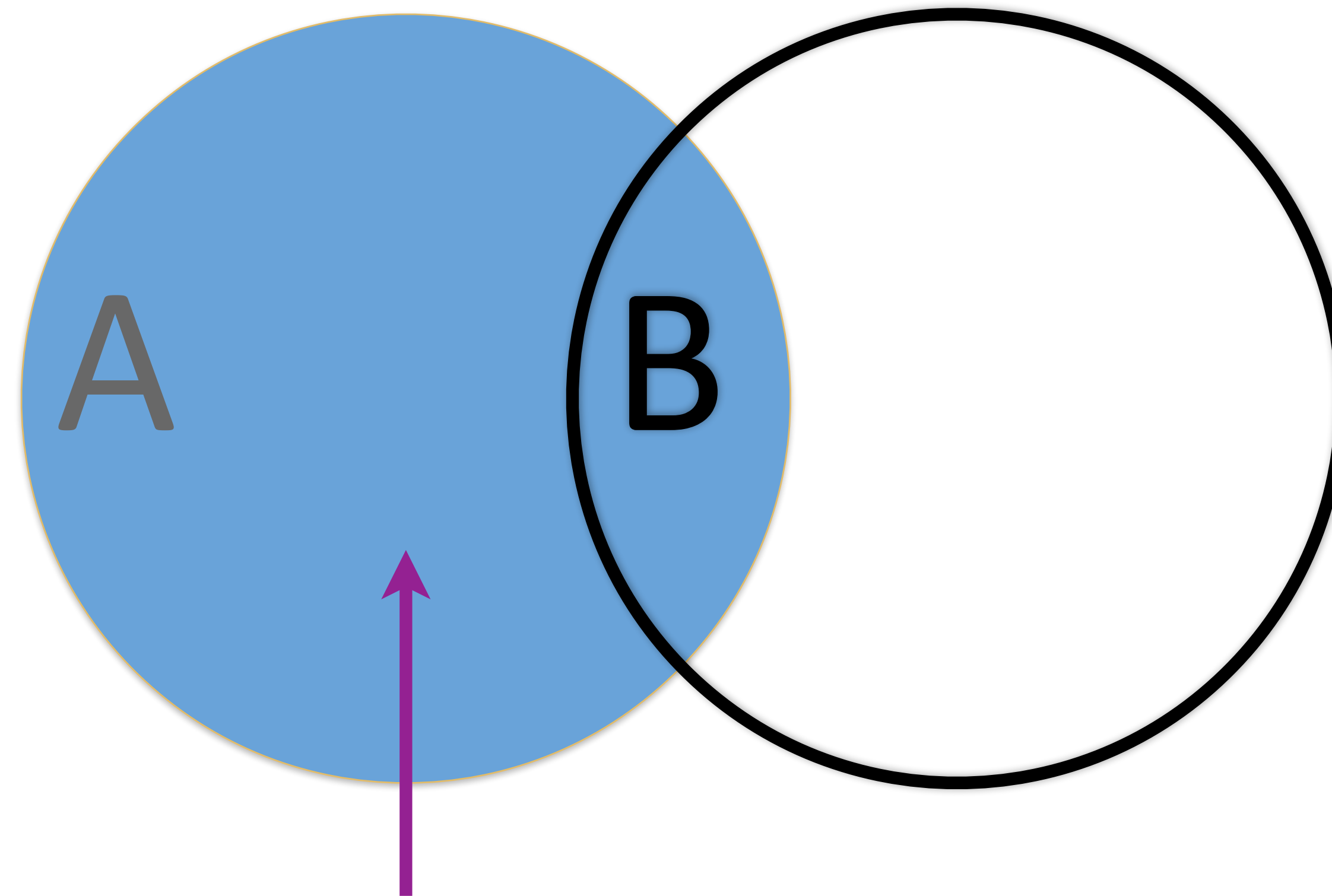

Joins

Inner Join (Intersection)



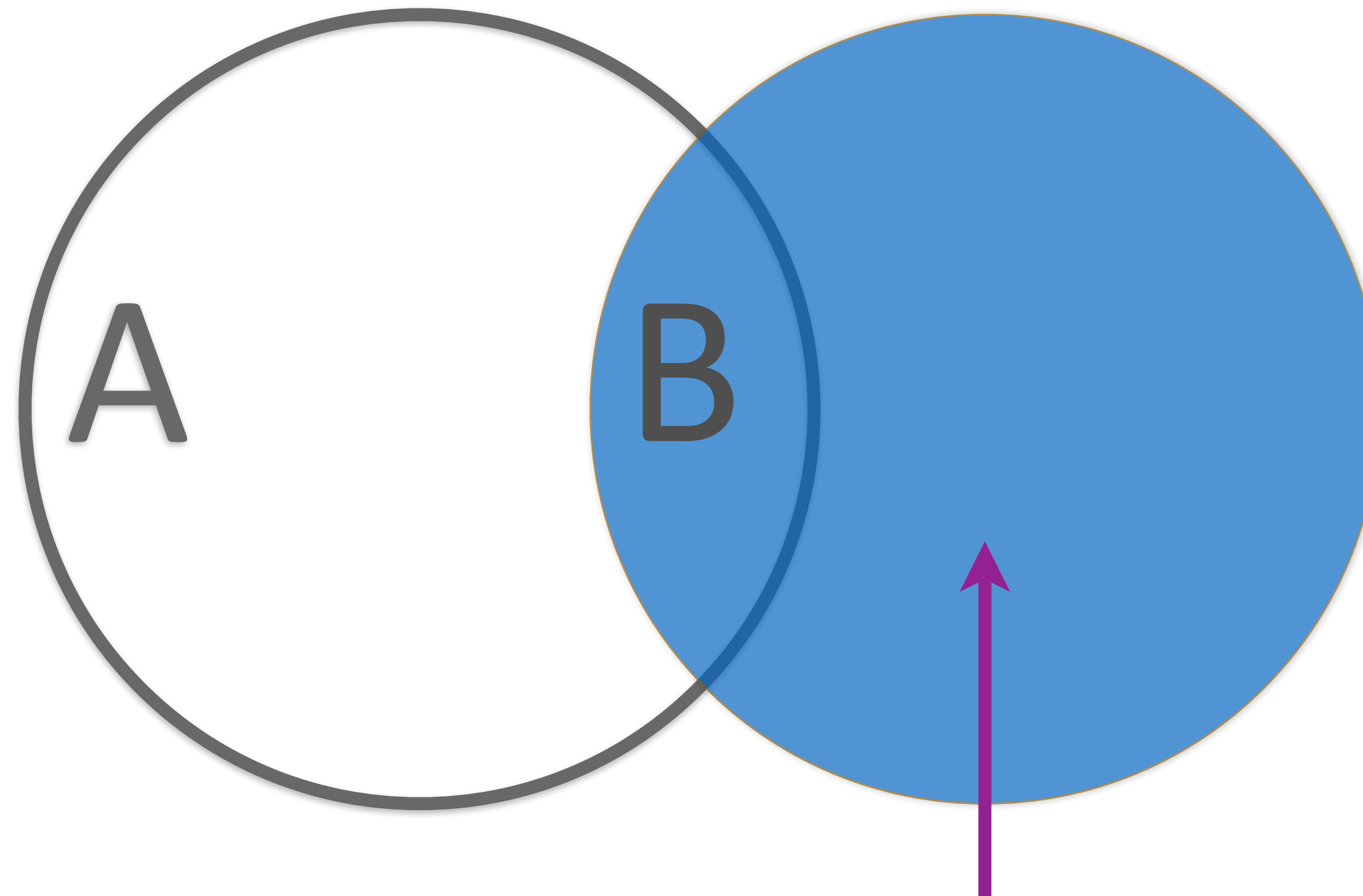
Inner Join

Left Join



Left Join

Right Join



Right Join

```
pd.merge( leftData, rightData )
```

```
pd.merge( leftData,  
          rightData,  
          how="<join type>" )
```

```
pd.merge( leftData, rightData,  
how="<join type>" )
```

Option	SQL Equivalent	Description
inner	INNER JOIN	Intersection
left	LEFT OUTER JOIN	Returns items in Set A, but not in Set B
right	RIGHT OUTER JOIN	Returns items in Set B, but not in Set A
outer	FULL OUTER JOIN	Returns the union of both sets

```
pd.merge( leftData, rightData,  
          how="<join type>",  
          on="<field list> )
```


Stop

In-Class Exercise

Back at xx:xx

Please complete Worksheet 2.2: Exploratory Data Analysis

Grouping and Aggregating Data

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

`df.groupby(field or list of fields)`

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby( 'src_ip' )[ 'port' ].count( )
```

src_ip	count
192.168.20.1	1
192.168.20.2	3

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby(['date', 'src_ip'])['port'].count()
```

Multi-Index

date	src_ip	
2018-06-21	192.168.20.1	1
	192.168.20.2	2
2018-06-22	192.168.20.2	1

Grouping and Aggregating Data

date	src_ip	dst_ip	port
2018-06-21	192.168.20.2	10.10.4.1	80
2018-06-21	192.168.20.1	10.10.4.2	443
2018-06-21	192.168.20.2	10.10.5.1	80
2018-06-22	192.168.20.2	10.10.4.1	80

```
df.groupby([ 'date' , 'src_ip' ], as_index=False)  
[ 'port' ].count()
```

	date	src_ip	port
0	2018-06-21	192.168.20.1	1
1	2018-06-21	192.168.20.2	2
2	2018-06-22	192.168.20.2	1

```
Series.dropna( )  
Series.fillna(value="<something>" )
```


In-Class Exercise

Back at xx:xx

Please complete Worksheet 2.2: Exploratory Data Analysis

Scaling Pandas

- Pandas is limited by the amount of memory on your machine. You will need 5-10x as much RAM as the size of your datasets.
- Operations on DataFrames creates copies of the data in memory.
- Pandas 2.0 has a big improvement in memory management.

Scaling Pandas

Several drop-in or near drop-in replacements for Pandas exist that run on GPUs or parallelize operations.

- **Dask:** A partial Pandas replacement geared towards running code on compute clusters.
- **cuDF:** A drop in partial replacement for Pandas for use on GPUs. Developed by NVIDIA
- **Ray:** A unified low level framework for scaling AI and Python applications.
- **RAPIDS:** A set of libraries and APIs that runs on NVIDIA CUDA GPUs. It offers GPU-enabled substitutes for popular Python libraries, such as cuDF for Pandas, cuML for scikit-learn, and cuGraph for NetworkX.
- **Polars:** A fast dataframe library implemented in Rust. (Runs in Python).
- **PySpark:** The Python API for Spark that lets you harness the simplicity of Python and the power of Apache Spark.

Apache Arrow

- Apache Arrow is a standardized framework for developing data analytics applications that use columnar data.
- Arrow can be used in python (PyArrow), Spark and many other languages.
- Arrow is designed as a compliment to parquet format.
- Main advantages are interoperability, speed and efficiency.

Pandas 2.0

- Pandas 2.0 was released in early 2023 and now has an integration with Apache Arrow
- The Arrow integration can help overcome the limitations of the earlier versions of Pandas.
- The Arrow backend is up to 35x faster.
- Pandas 2.0 has better support for analytic datatypes via Arrow.

```
pd.read_csv("data.csv", engine='pyarrow', dtype_backend='pyarrow')
```

Pandas 2.0: Copy on Write Optimization

- One of the big challenges of Pandas is that it creates multiple copies of your data frame when you chain methods together.
- Pandas 2.0 introduces a new copy on write option which, when set to True, returns views rather than copies, minimizing data duplication.
- When enabled, chained assignments will not work.

```
pd.options.mode.copy_on_write = True
```

Questions?