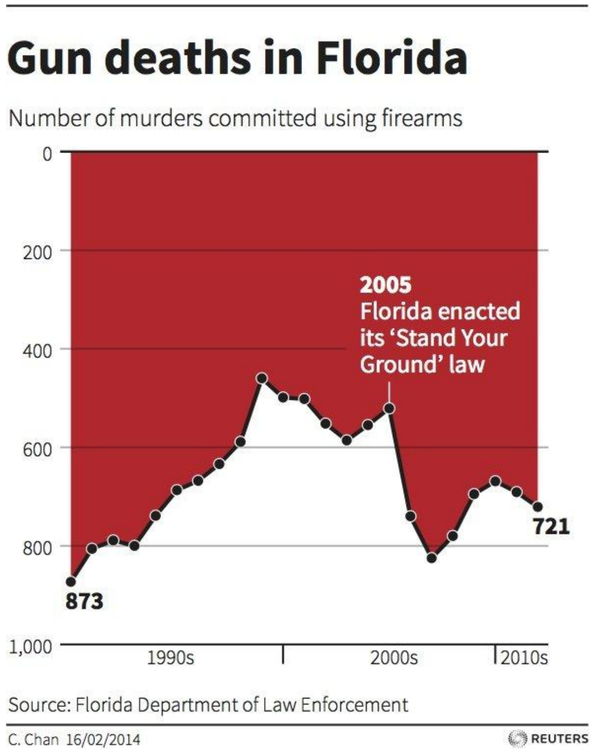
Data Reporting

THE KNOWLEDGE HOUSE

Agenda - Schedule

1. Leetcode Warm Up
2. Joining DataFrames
3. Data Reporting
4. Break
5. TLAB Work



Statistics is not immune from human-trickery

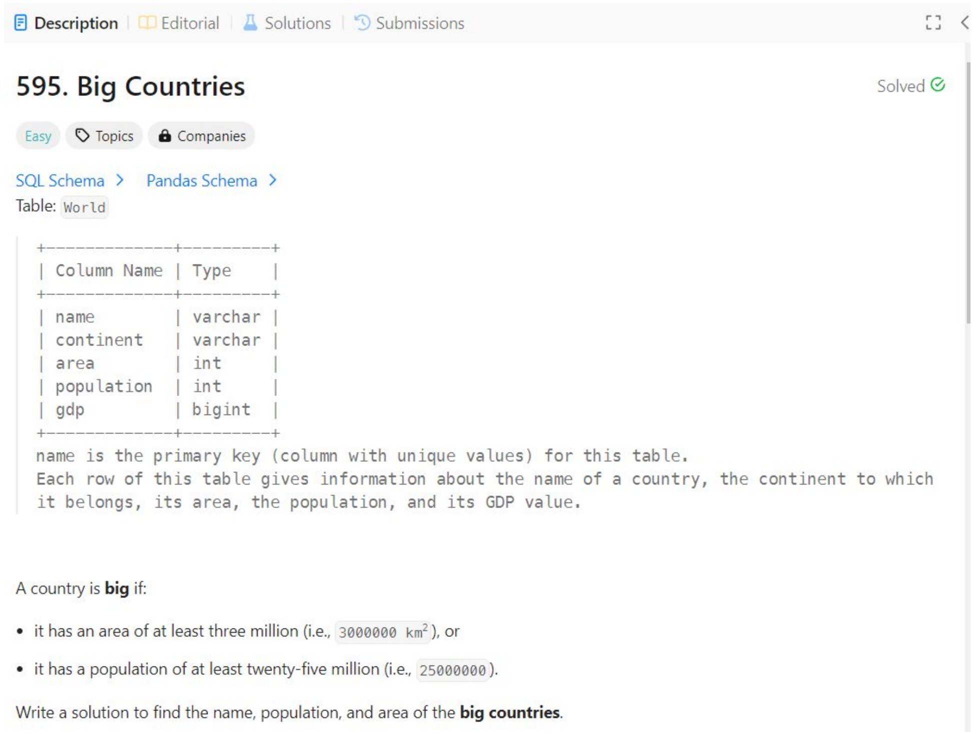
Agenda - Goals

- Load CSV data into pandas DataFrames.
- Concatenate datasets with similar structure using pd.concat().
- Merge datasets on a common key using pd.merge().
- Review foundational pandas methods to inspect your dataframes.
- Apply best practices when summarizing and reporting your findings.

Announcement(s)

- **Career Class on 4/17** (*this time for real*)
- **TLAB #2** due 4/21
- Farukh Group office hours open (*register via Canvas*)
    - **Cohort A** Group Office Hours **on Thursday**: *4:30 - 5:30*
    - **Cohort B** Group Office Hours **on Thursday**: *2:00 - 3:00*

Warm Up



595. Big Countries

Table: `World`

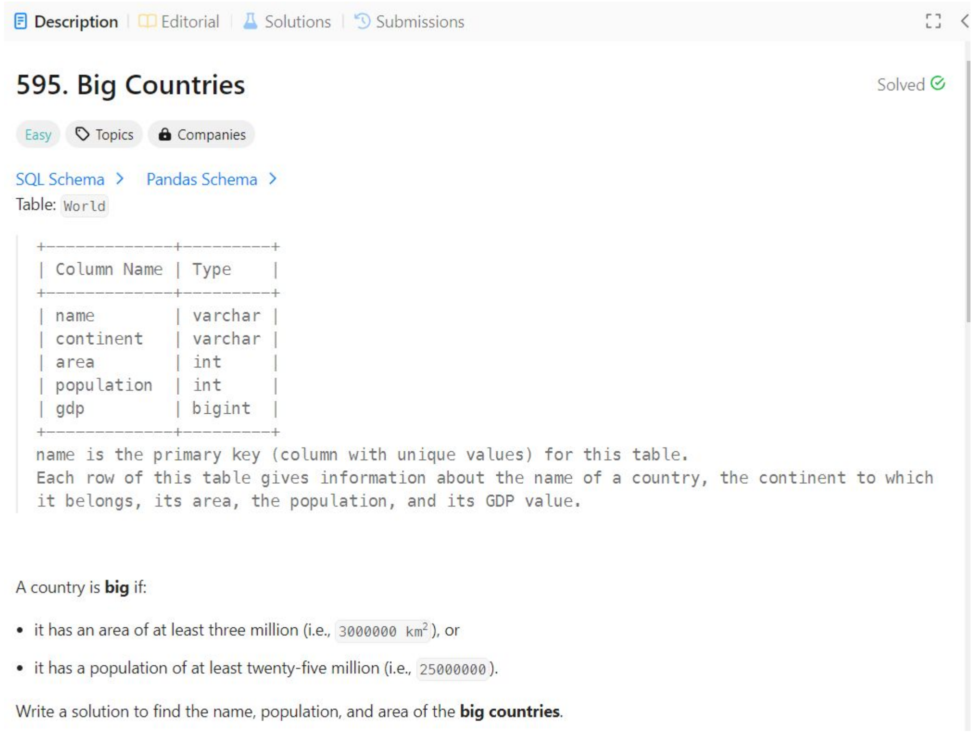
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
Each row of this table gives information about the name of a country, the continent to which
it belongs, its area, the population, and its GDP value.
```

A country is **big** if:

- it has an area of at least three million (i.e., `3000000 km`^{2}), or
- it has a population of at least twenty-five million (i.e., `25000000`).

Write a solution to find the name, population, and area of the **big countries**.

During your 1st or 2nd-round interview, you will most likely be asked to solve some sort of technical task to prove your competency in the tools you listed on your resume.



595. Big Countries

Table: `World`

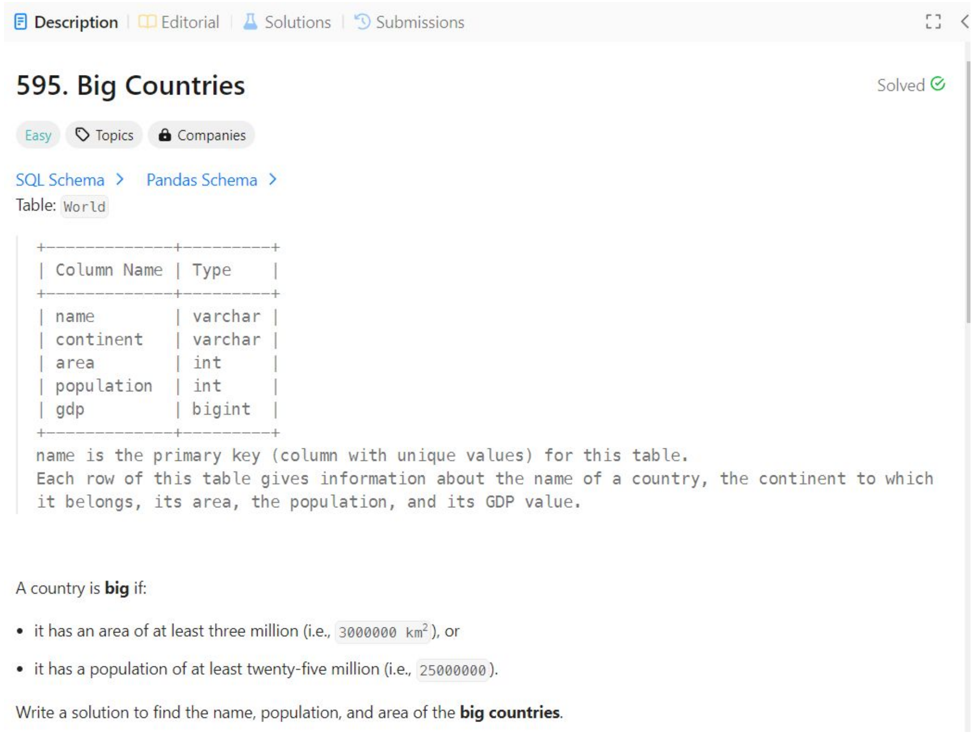
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
Each row of this table gives information about the name of a country, the continent to which
it belongs, its area, the population, and its GDP value.
```

A country is **big** if:

- it has an area of at least three million (i.e., `3000000 km`^{2}), or
- it has a population of at least twenty-five million (i.e., `25000000`).

Write a solution to find the name, population, and area of the **big countries**.

That's why going forward, we will start most of our classes off with a **Leetcode question**. Now while we will not be grading you on this...



595. Big Countries

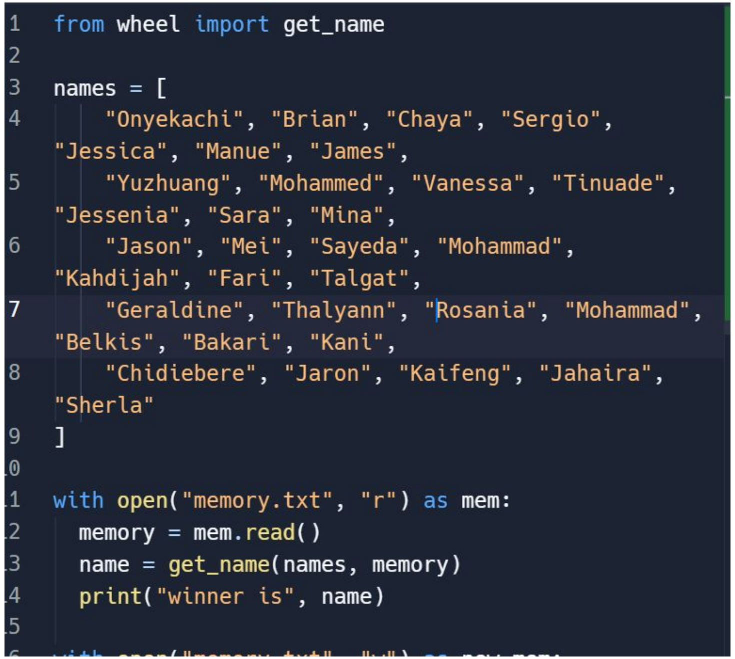
Table: `World`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
Each row of this table gives information about the name of a country, the continent to which
it belongs, its area, the population, and its GDP value.
```

A country is **big** if:

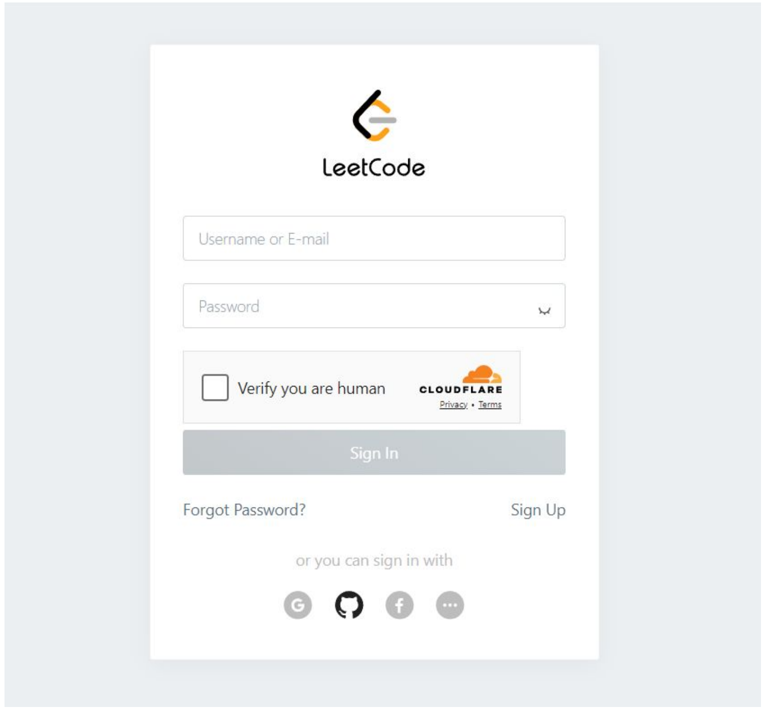
- it has an area of at least three million (i.e., `3000000 km²`), or
- it has a population of at least twenty-five million (i.e., `25000000`).

Write a solution to find the name, population, and area of the **big countries**.

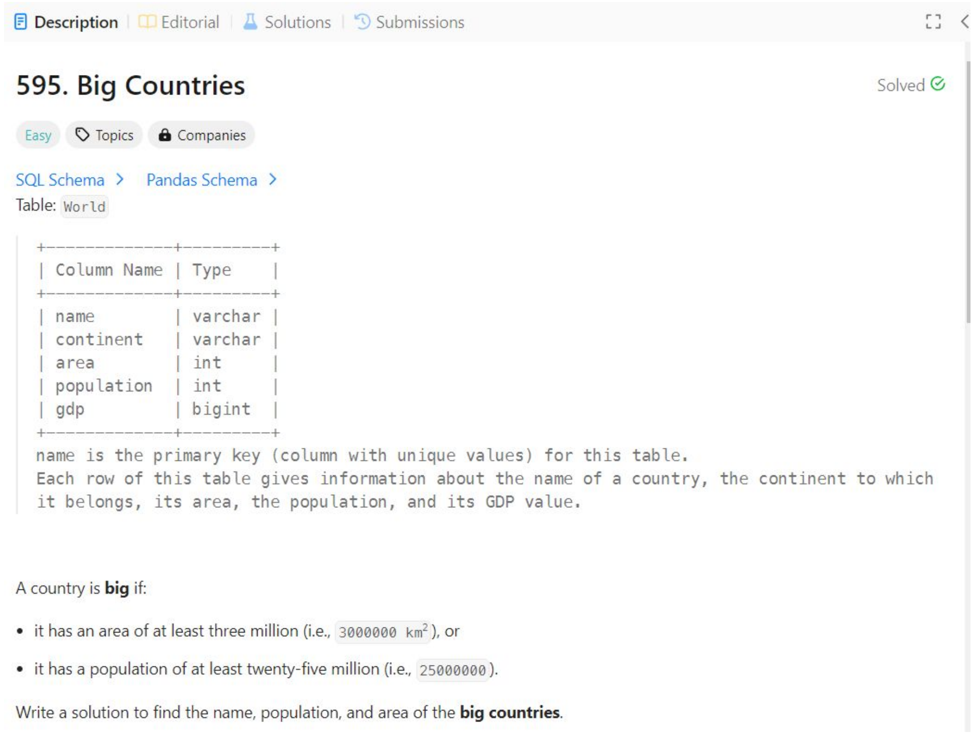


```python
from wheel import get_name

names = [
    "Onyekachi", "Brian", "Chaya", "Sergio", "Jessica", "Manue", "James",
    "Yuzhuang", "Mohammed", "Vanessa", "Tinuade", "Jessenia", "Sara", "Mina",
    "Jason", "Mei", "Sayeda", "Mohammad", "Kahdijah", "Fari", "Talgat",
    "Geraldine", "Thalyann", "Rosania", "Mohammad", "Belkis", "Bakari", "Kani",
    "Chidiebere", "Jaron", "Kaifeng", "Jahaira", "Sherla"
]

with open("memory.txt", "r") as mem:
  memory = mem.read()
  name = get_name(names, memory)
  print("winner is", name)

with open("memory.txt", "w") as new_mem:
```

After working on this problem for 10 minutes, we will be pulling from a uniform distribution (*with selective replacement*) of names to make sure everyone gets the **leetcode experience**.



To start, please create a Leetcode account using your GitHub **login information**.



595. Big Countries

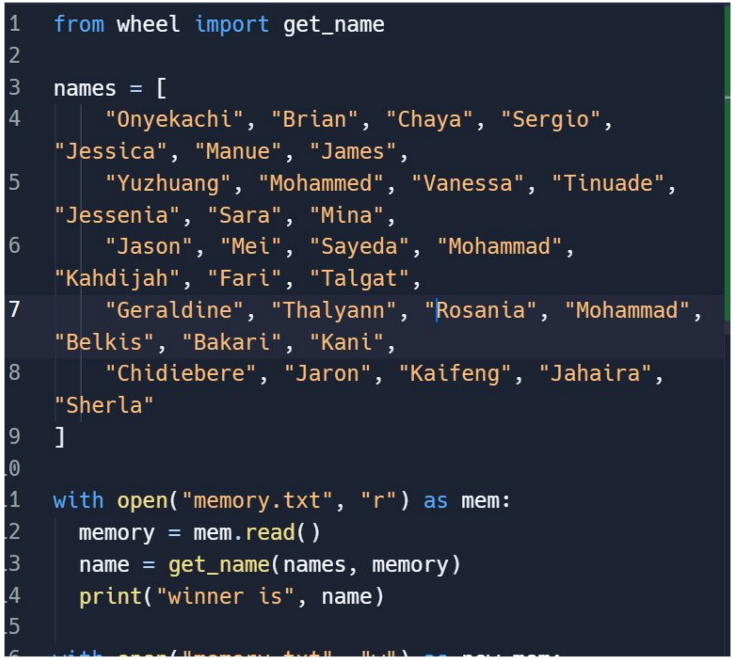
Table: `World`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
Each row of this table gives information about the name of a country, the continent to which
it belongs, its area, the population, and its GDP value.
```

A country is **big** if:

- it has an area of at least three million (i.e., `3000000 km²`), or
- it has a population of at least twenty-five million (i.e., `25000000`).

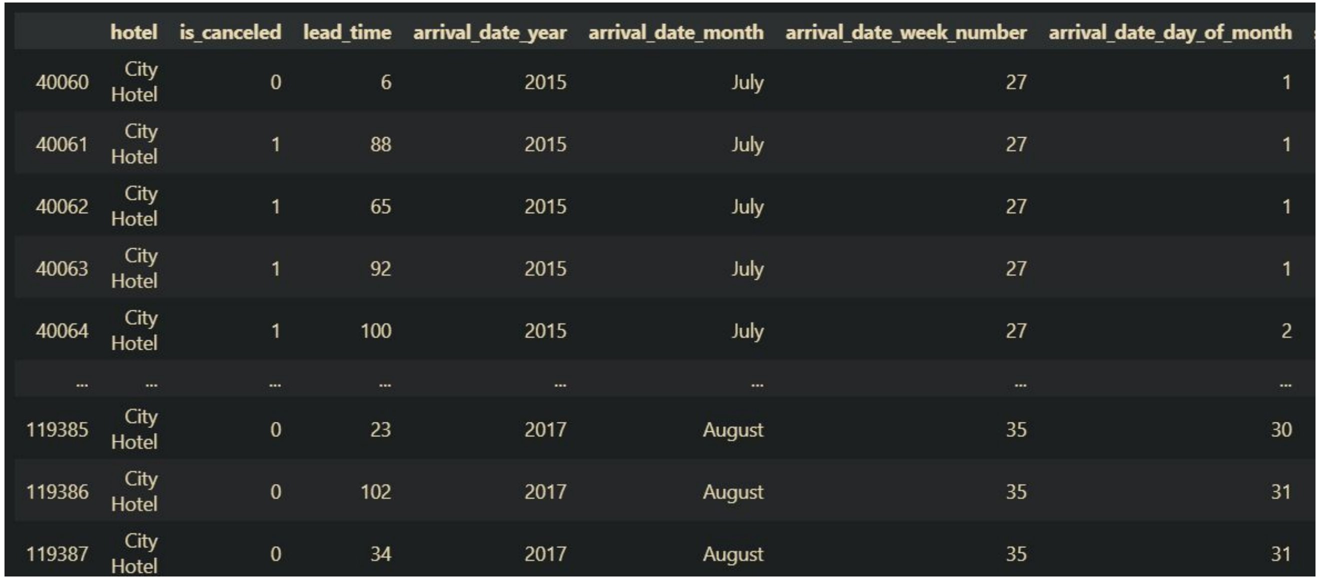
Write a solution to find the name, population, and area of the **big countries**.



```python
from wheel import get_name

names = [
    "Onyekachi", "Brian", "Chaya", "Sergio", "Jessica", "Manue", "James",
    "Yuzhuang", "Mohammed", "Vanessa", "Tinuade", "Jessenia", "Sara", "Mina",
    "Jason", "Mei", "Sayeda", "Mohammad", "Kahdijah", "Fari", "Talgat",
    "Geraldine", "Thalyann", "Rosania", "Mohammad", "Belkis", "Bakari", "Kani",
    "Chidiebere", "Jaron", "Kaifeng", "Jahaira", "Sherla"
]

with open("memory.txt", "r") as mem:
  memory = mem.read()
  name = get_name(names, memory)
  print("winner is", name)
```

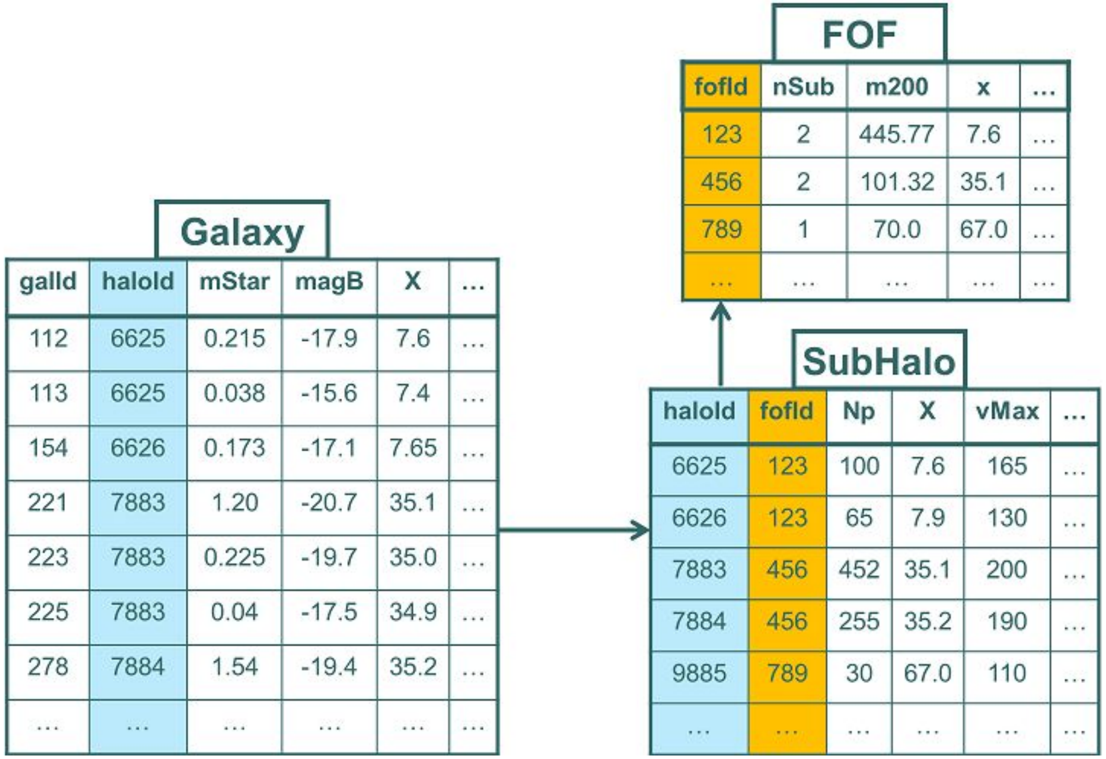
Take 10 minutes to work on the “Big Countries” Leetcode problem: https://leetcode.com/problems/big-countries/description/?envType=study-plan-v2&envId=30-days-of-pandas&lang=pythondata

Pandas Merging & Concatenation



	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month
40060	City Hotel	0	6	2015	July	27	1
40061	City Hotel	1	88	2015	July	27	1
40062	City Hotel	1	65	2015	July	27	1
40063	City Hotel	1	92	2015	July	27	1
40064	City Hotel	1	100	2015	July	27	2
...	...	...	...	...	...	...	...
119385	City Hotel	0	23	2017	August	35	30
119386	City Hotel	0	102	2017	August	35	31
119387	City Hotel	0	34	2017	August	35	31

In yesterday's lecture, we learned how to manipulate a single CSV file for some light descriptive analytics.



FOF

fofId	nSub	m200	x	...
123	2	445.77	7.6	...
456	2	101.32	35.1	...
789	1	70.0	67.0	...
...	...	...	...	...

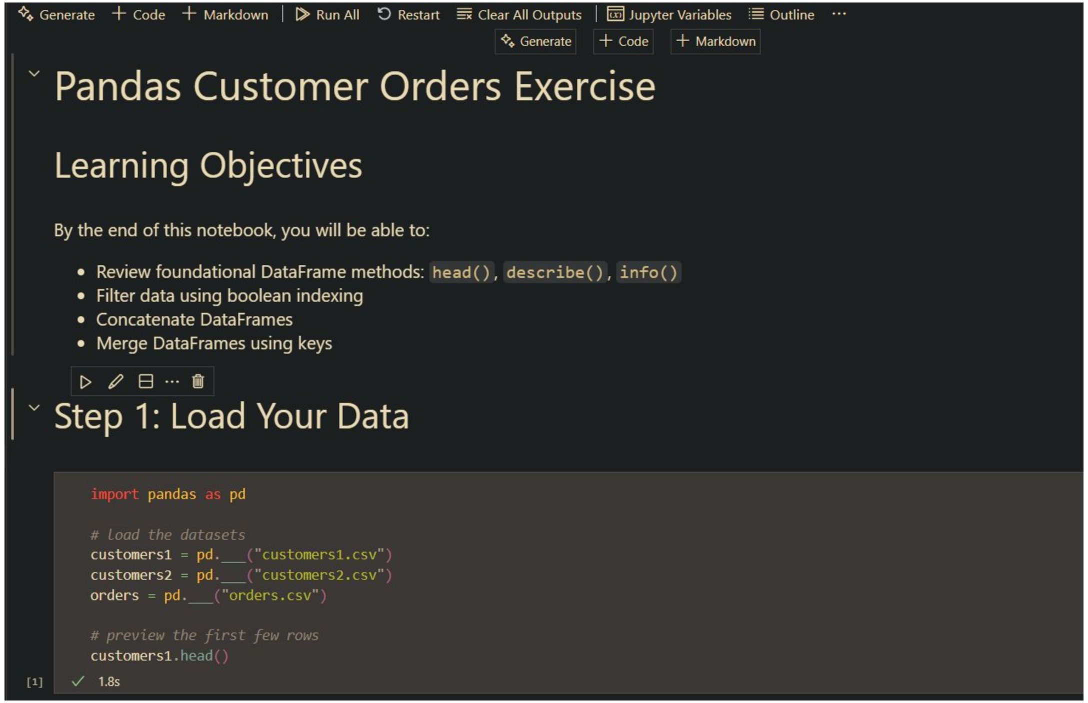
Galaxy

galId	haloId	mStar	magB	X	...
112	6625	0.215	-17.9	7.6	...
113	6625	0.038	-15.6	7.4	...
154	6626	0.173	-17.1	7.65	...
221	7883	1.20	-20.7	35.1	...
223	7883	0.225	-19.7	35.0	...
225	7883	0.04	-17.5	34.9	...
278	7884	1.54	-19.4	35.2	...
...	...	...	...	...	...

SubHalo

haloId	fofId	Np	X	vMax	...
6625	123	100	7.6	165	...
6626	123	65	7.9	130	...
7883	456	452	35.1	200	...
7884	456	255	35.2	190	...
9885	789	30	67.0	110	...
...	...	...	...	...	...

But most of the time, you will not only be dealing with one dataset that contains all **features**. The most likely case is that you will be dealing with **numerous data tables** that need to be joined together so that you can perform analysis.



Pandas Customer Orders Exercise

Learning Objectives

By the end of this notebook, you will be able to:

- Review foundational DataFrame methods: `head()`, `describe()`, `info()`
- Filter data using boolean indexing
- Concatenate DataFrames
- Merge DataFrames using keys

Step 1: Load Your Data

```
import pandas as pd

# load the datasets
customers1 = pd.___("customers1.csv")
customers2 = pd.___("customers2.csv")
orders = pd.___("orders.csv")

# preview the first few rows
customers1.head()
```

```
[1]  ✓ 1.8s
```

Just like yesterday, we will apply these general syntax on our code-blocks to demonstrate the **actual utility of this code**. Remember, these slides are secondary, the best way to understand the code is to write and run it yourself.

Pandas Review - Merging & Concatenating

Most of the time, we don’t have just **one** dataset but rather **multiple**. A few common join operations that we will write include:

- **Concatenate**: *appending two dataframes to one another*
- **Appending**: *appending one row to a database*
- **Joining**: *performing logical joins with two dataframes*

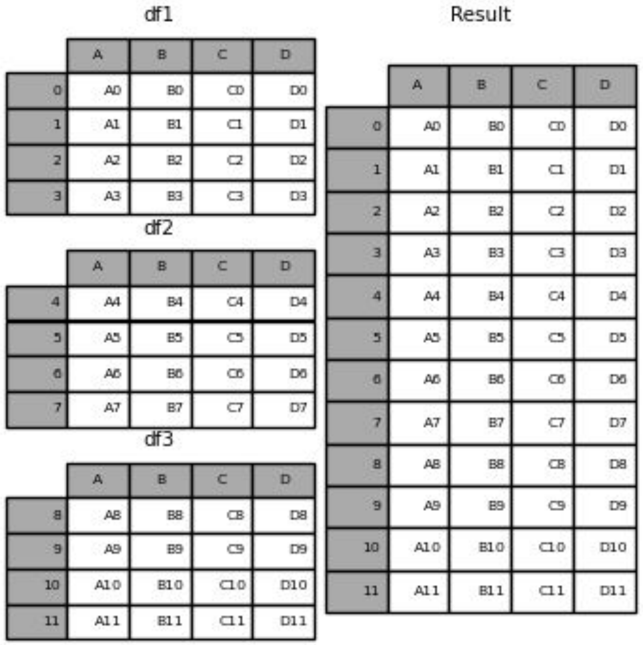
This is only a review, check out the full documentation for more info.

Pandas Review - Concatenating

“takes a list or dict of homogeneously-typed objects a
concatenates them with some configurable handling
of ‘what to do with the other axes’”

pd.concat([df1,df2,df3])

*This is the simplest case. The documentation expands more upon this.



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

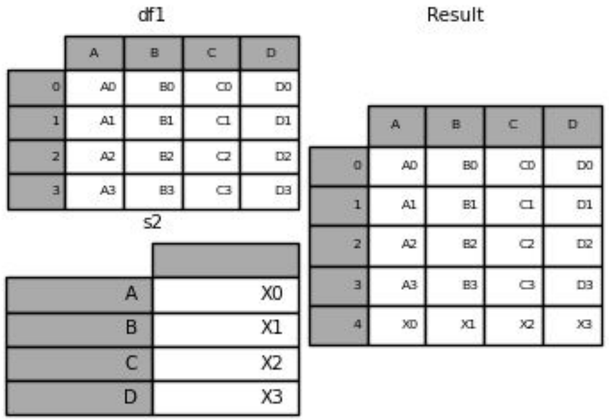
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Pandas Review - Appending

*“If you have a series that you want to **append as a single row to a DataFrame**, you can convert the row into a DataFrame and use concat.”*

```
pd.concat([df1,
s2.to_frame().T],ignore_index=True)
```

*Always ask yourself if it's necessary to do such a granular operation in pandas. More often than not, we shouldn’t be.



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

s2

A	X0
B	X1
C	X2
D	X3

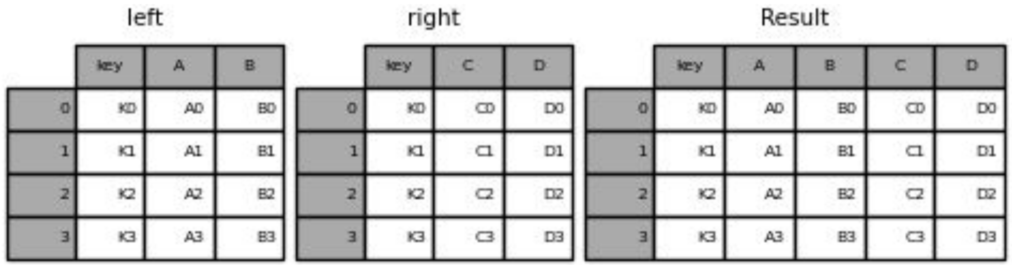
Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	X0	X1	X2	X3

Pandas Review - Joining

“high performance in-memory join operations ***idiomatically very similar to relational databases like SQL****.”*

```
pd.merge(left, right, on="key")
```



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

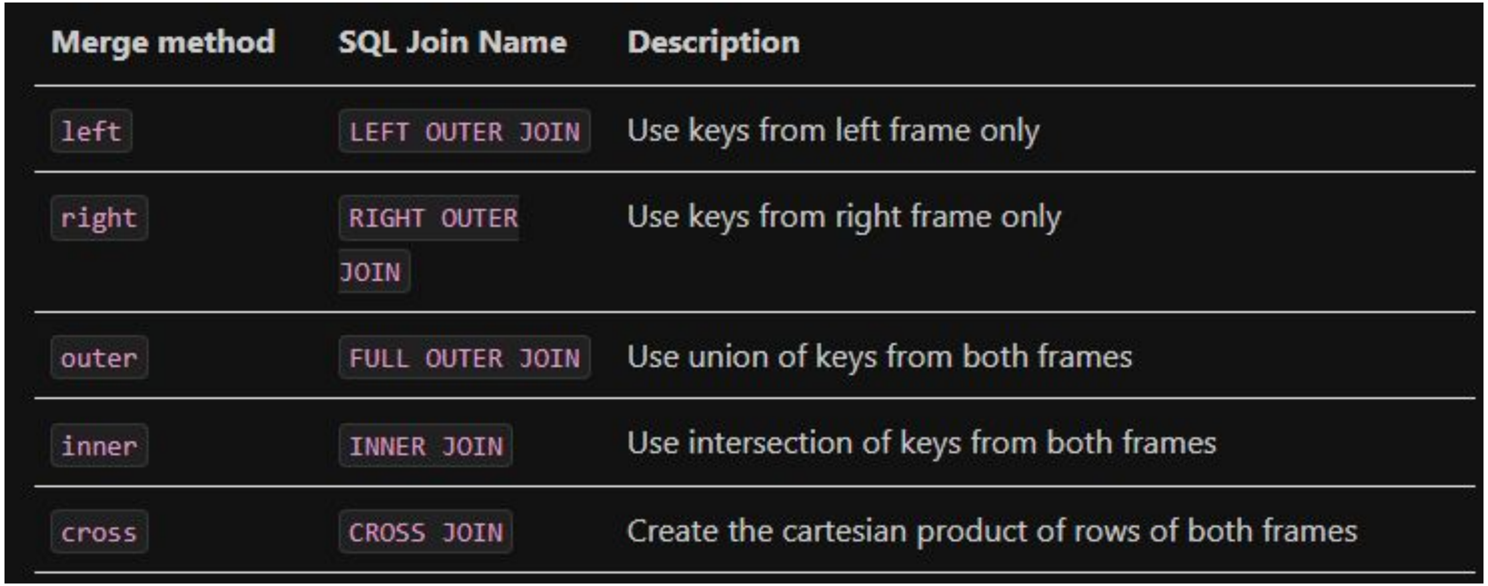
right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Result

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Notice that the merge occurs according to **similar key values**



Merge method	SQL Join Name	Description
`left`	`LEFT OUTER JOIN`	Use keys from left frame only
`right`	`RIGHT OUTER JOIN`	Use keys from right frame only
`outer`	`FULL OUTER JOIN`	Use union of keys from both frames
`inner`	`INNER JOIN`	Use intersection of keys from both frames
`cross`	`CROSS JOIN`	Create the cartesian product of rows of both frames

We can create these complex joins by using the simplicity of the pandas API. Just use the correct parameter as your merge method! (aka ‘how=...’)

Data Reporting

Data Reporting

Not only do we need to have the technical/mathematical competency to properly use tools…

But we also need to be able to communicate findings & methodology to both **high-context** (other technologists) and **low-context** (external team) crowds.

This applies to any analytical document you create

Data Reporting

Let's go over a few do's & don'ts.

- Be specific
- Use helpful visuals
- Appropriately discuss inferences
- Report methods

Data Reporting

To frame our exploration of data write-ups, let’s take a look at the 2013 paper titled ***The impact of free-ranging domestic cats on wildlife of the United States*** (https://www.nature.com/articles/ncomms2380)

Data Reporting - Be Specific

DO:

Mention specific values, such as mean, median, mode, or any other useful metric when describing your observations. Discuss specific magnitude and direction when discussing changes.

DON'T:

Make broad or general statements with no backing by data.

Free-ranging cats on islands have led to many extinctions of different mammals.

Bad Example: When describing your observations, we are ambiguous. What is “many?”

“Free-ranging cats on islands have caused or contributed to 33 (14%) of the modern bird, mammal and reptile extinctions…”

Good Example: When describing your observations, make reference to the descriptives you’ve calculated.

"We estimate that cats in the contiguous United States annually kill between 1.3 and 4.0 billion birds (median=2.4 billion) (Fig. 1a), with most mortality caused by un-owned cats."

Bad Example: When describing your observations, we are ambiguous.

“We estimate that cats in the contiguous United States annually kill between 1.3 and 4.0 billion birds (median=2.4 billion) (Fig. 1a), with ~69% of this mortality caused by un-owned cats.”

Good Example: When describing your observations, make reference to the descriptives you’ve calculated.

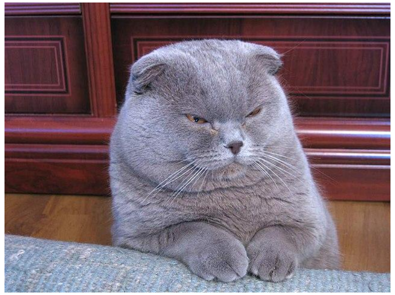
Data Reporting - Use Helpful Visualizations

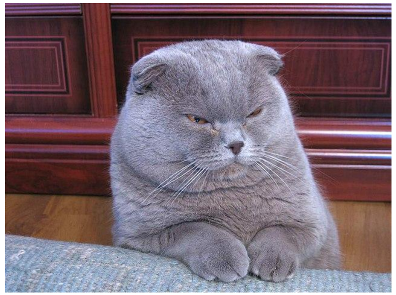
DO:

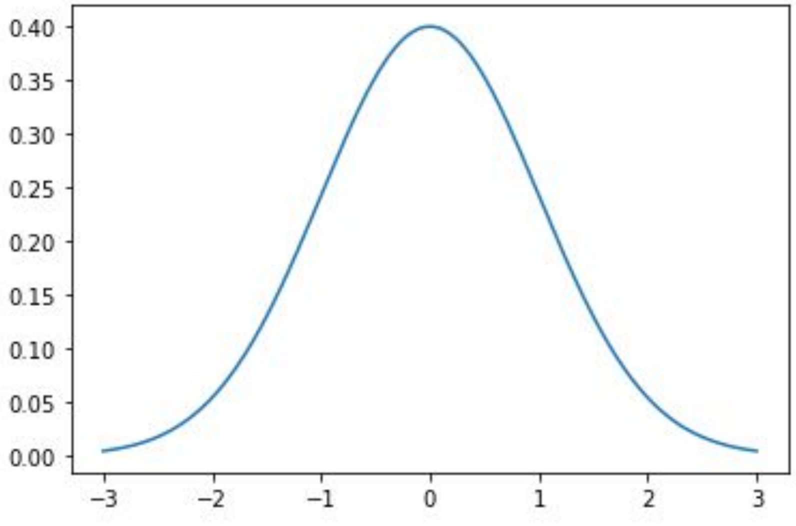
Use well-annotated visuals to represent your insight. Point to key leverage points in your visuals.

DON'T:

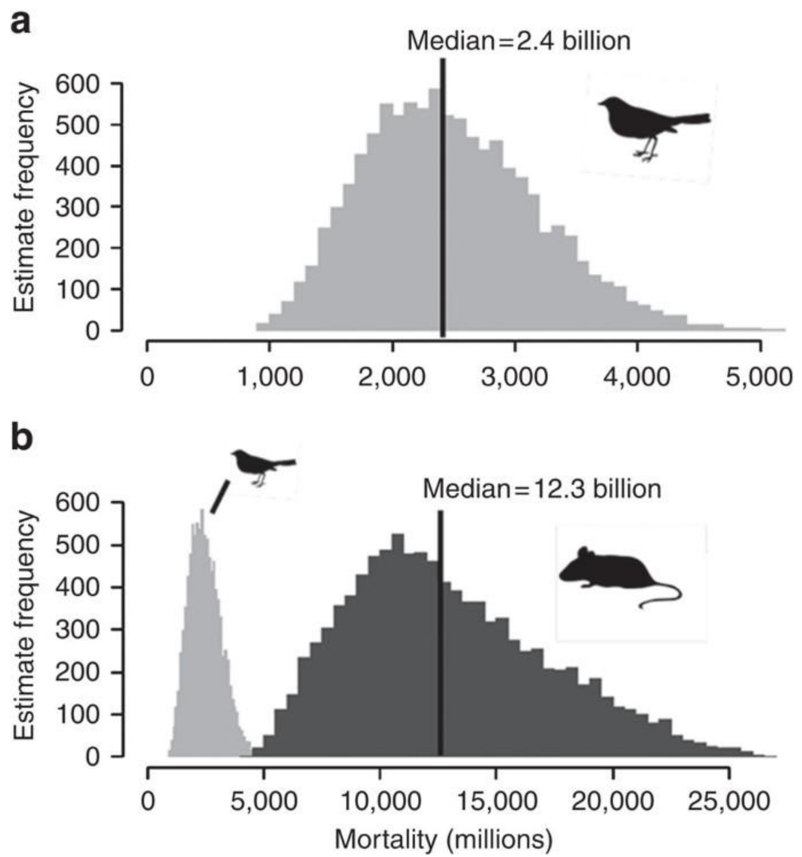
Rely on visuals without explanatory text.







Bad Example: An unmarked distribution, while telling, is not too useful to support your observations.



Good Example: Visuals utilize proper annotation to visually explain insights

Data Reporting - Appropriate Inferences

DO:

Use standard language when describing the output of hypothesis tests or other statistical methods. **(fail to reject, reject, significant)**

DON'T:

Use ambiguous or unclear language when discussing outputs of inferences.

The t-test showed that we can accept the alternative hypothesis. One teacher's learning methods were definitely better.

Bad Example: This is a misunderstanding of hypothesis testing.

“A Student’s independent samples t-test showed that this 5.4% difference was significant (t(31)=2.1, p<.05, CI95=[0.2,10.8]), suggesting that a genuine difference in learning outcomes has occurred.”

Good Example: Use standard language when describing the output of hypothesis tests or other statistical methods. The information in the parentheses is not needed unless sharing this info with other statisticians.

Data Reporting - Report Methods

DO:

Plainly state your methodology of collecting data. This includes source, time frame, and techniques used if applicable.

DON'T:

Leave your data collection a mystery to unravel.

"We searched the web to identify studies that document cat predation on birds and mammals."

Bad Example: Leave your data collection a mystery to unravel.

“We searched JSTOR, Google Scholar, and the Web of Science database (formerly ISI Web of Science) within the Web of Knowledge search engine published by Thomson Reuters to identify studies that document cat predation on birds and mammals.”

Good Example: Plainly state your methodology of collecting data.

“We got the count of data-adjacent job-postings”

Bad Example: Leave your data collection a mystery to unravel.

“We used regex via pandas to search for job-postings that contain the title ‘data’ in order to count the frequency of data-adjacent jobs in the industry.”

Good Example: Plainly state your methodology of collecting data.

“I tapped into my secret worldwide network of cat-related-news to identify studies that document cat predation on birds and mammals. My contacts: Whiskers, Mr.Mittens, and Tom introduced me to this hidden bastion of information.”

Entertaining Example

Applying to README

README’s

When a recruiter, looks at your public projects, they are not immediately diving into code.

Instead, they will look for evidence of relevant skills in your

README

Therefore we will require us all to have README’s before submitting

README’s

At the minimum your README should contain:

- *One-sentence project summary*
- *Your methodology*
- *Your conclusions with specific metrics*
- *Helpful/interesting visuals*
- *Your challenges & next-steps*

README’s

For the upcoming and final TLAB, you will write README’s so that a recruiter understands the **intent**, **analysis**, and **competencies** associated with our projects.

However, in the future you should fit your README to your target audience:

- Engineers → Inform on **software structure**
- Stakeholders → Inform on **value**
- Regulators → Inform on **compliance**

TLAB #2

Lab (Due 04/21)

Vancouver, Canada

You are a growth analyst at a Vancouver-based consulting firm called Monica Group. Your manager is spearheading the completion of a a new analytical tool which will automatically label if a review is positive, neutral, negative, or irrelevant.

You will be kicking off completion of this milestone by independently implementing a minimal-viable-product. **This will be a Python pipeline that ingests a text-file of review data and interfaces with the Open AI API in order to automatically label each review.**

We released API keys.

Thursday

No review session. Financial workshop

Jupyter: scratchpad of the data scientist

If you understand what you're doing, you're not learning anything. - Anonymous