DATA FOR HEROES II

OLF 2025

Catherine Devlin

2025-12-06

SUPERVILLAINS

Nell Strongarm
The Astro-Thug

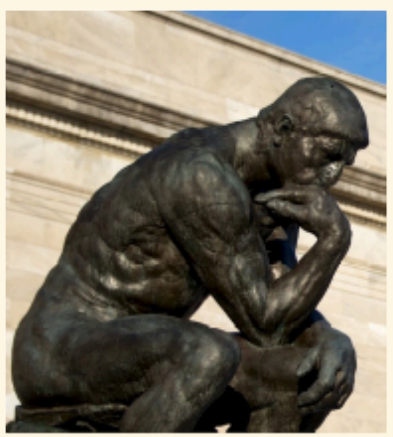
Henry Fnord
Mass-Producer
Of Mayhem

The Wrong Brothers
Fiends of Flight

The Trilobite

SUPERHEROES

Cardinal Virtue
The Redbird of
Righteousness



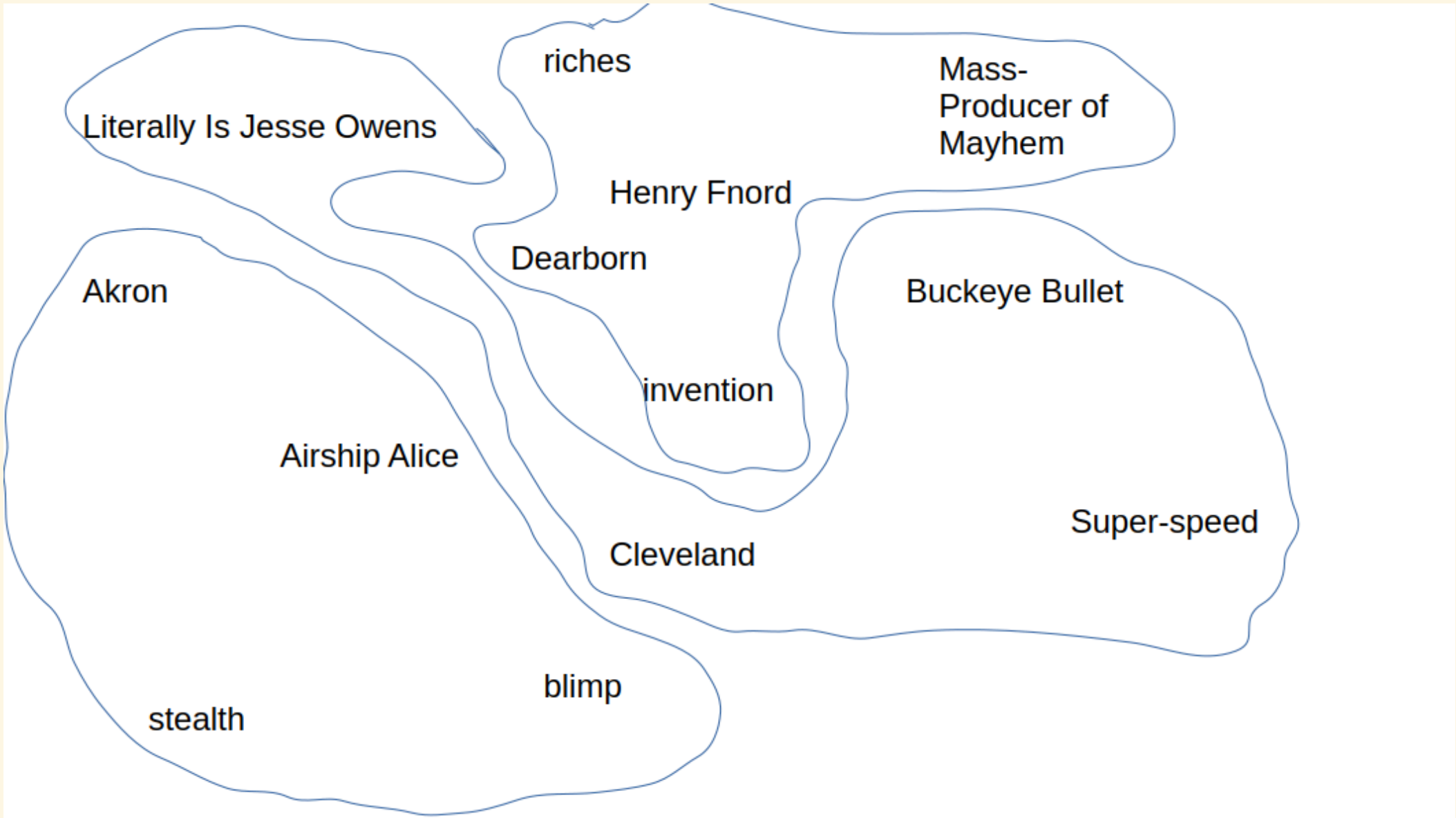
The Thinker

Captain Columbus
Protector of Ohio

Ohio Gosaimasu
Sandusky Samurai

INFORMATION?

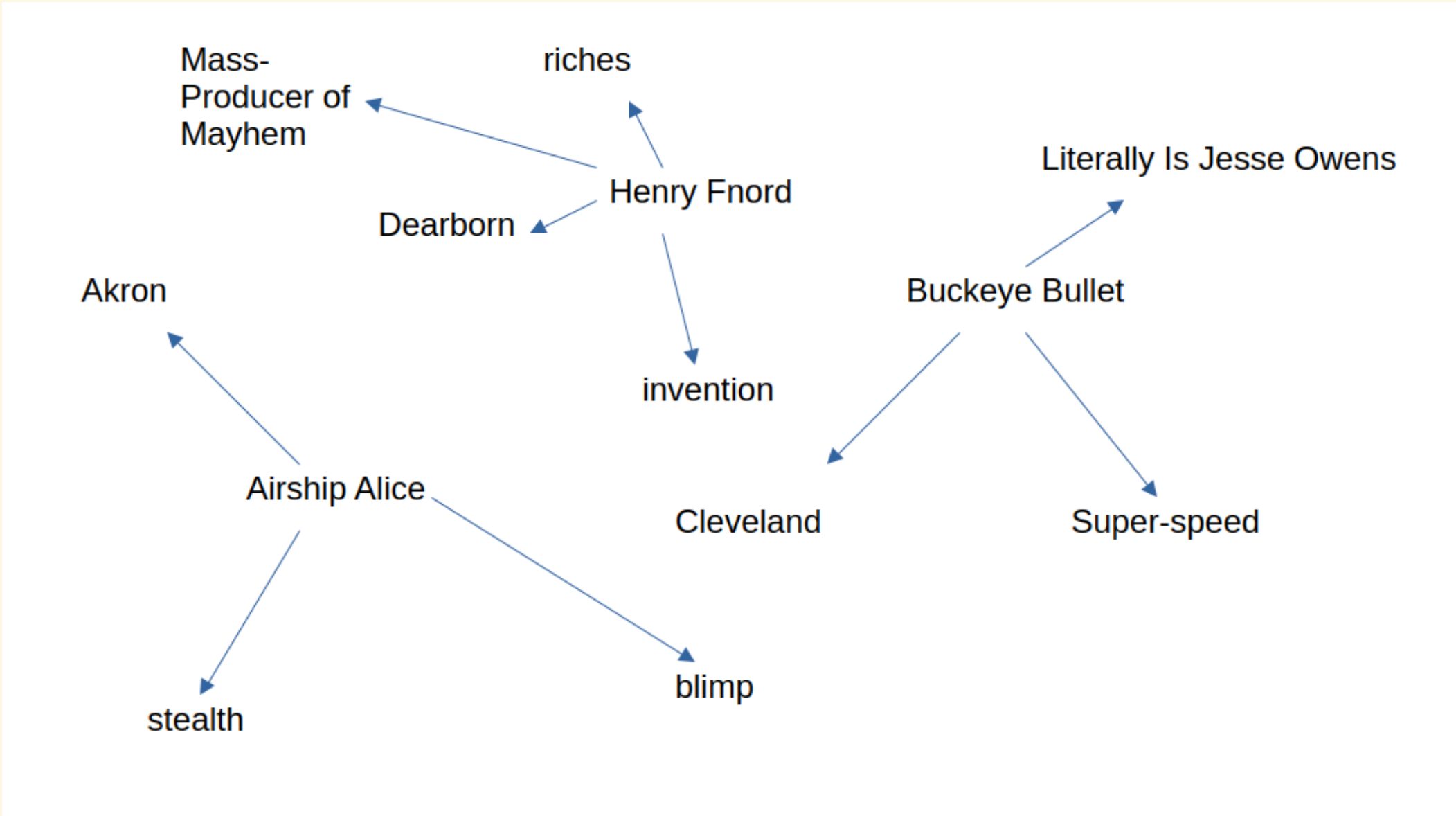
RELATIONSHIPS == INFORMATION



TABLES (RELATIONS)

Name	Tagline	Location	Power 1	Power 2
Buckeye Bullet	Literally Is Jesse Owens	Cleveland	Super-speed	
Henry Fnord	Mass-Producer of Mayhem	Dearborn	invention	riches
Airship Alice		Akron	blimp	stealth

GRAPH: NODES AND EDGES



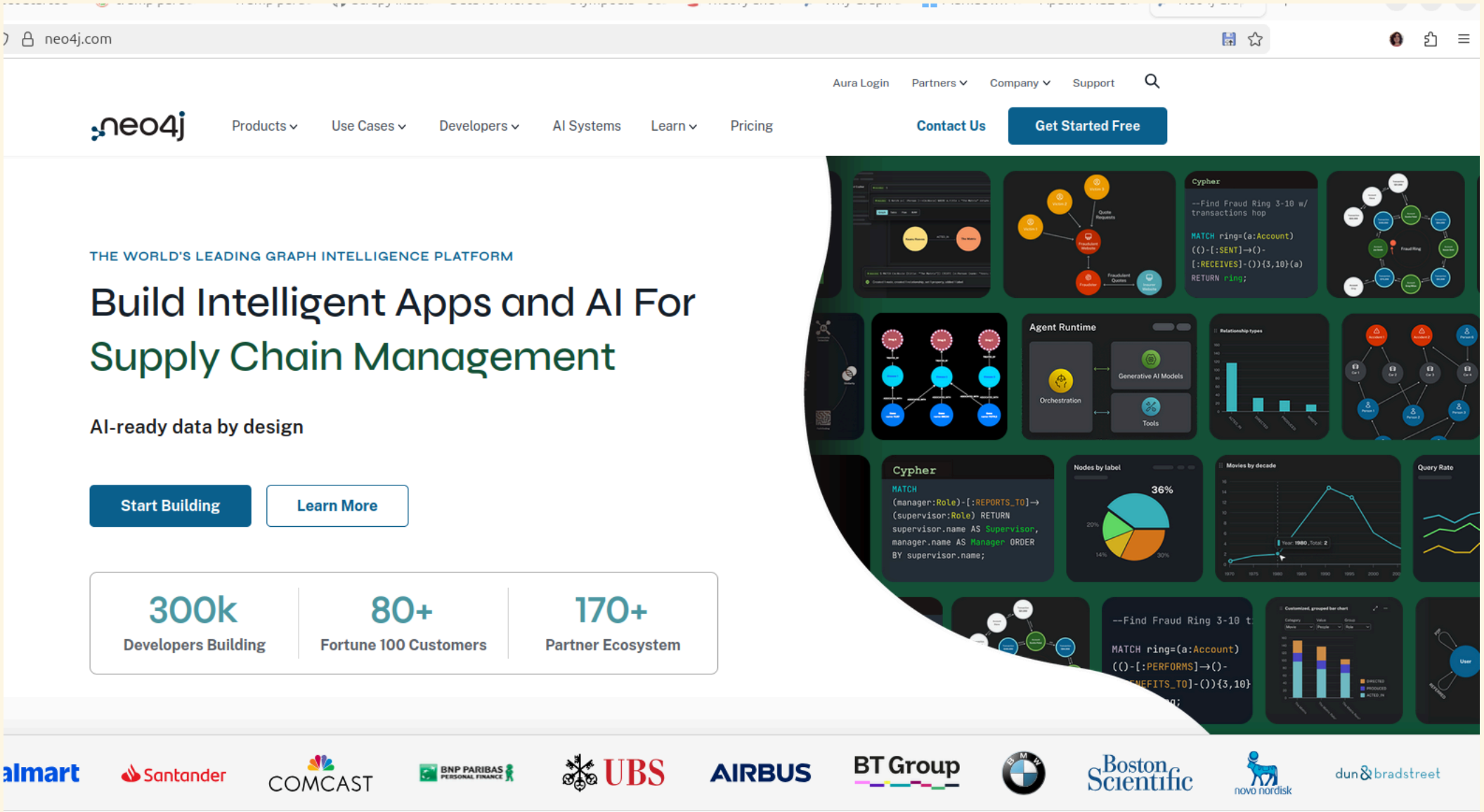
FLEXIBILITY

Name	Power 1	Power 2	Favorite food	Zodiac	Height (cm)
Buckeye Bullet	speed				178
Airship Alice	blimp	stealth	Sub		
Henry Fnord	invention	riches		Leo	

CHAINED QUERY

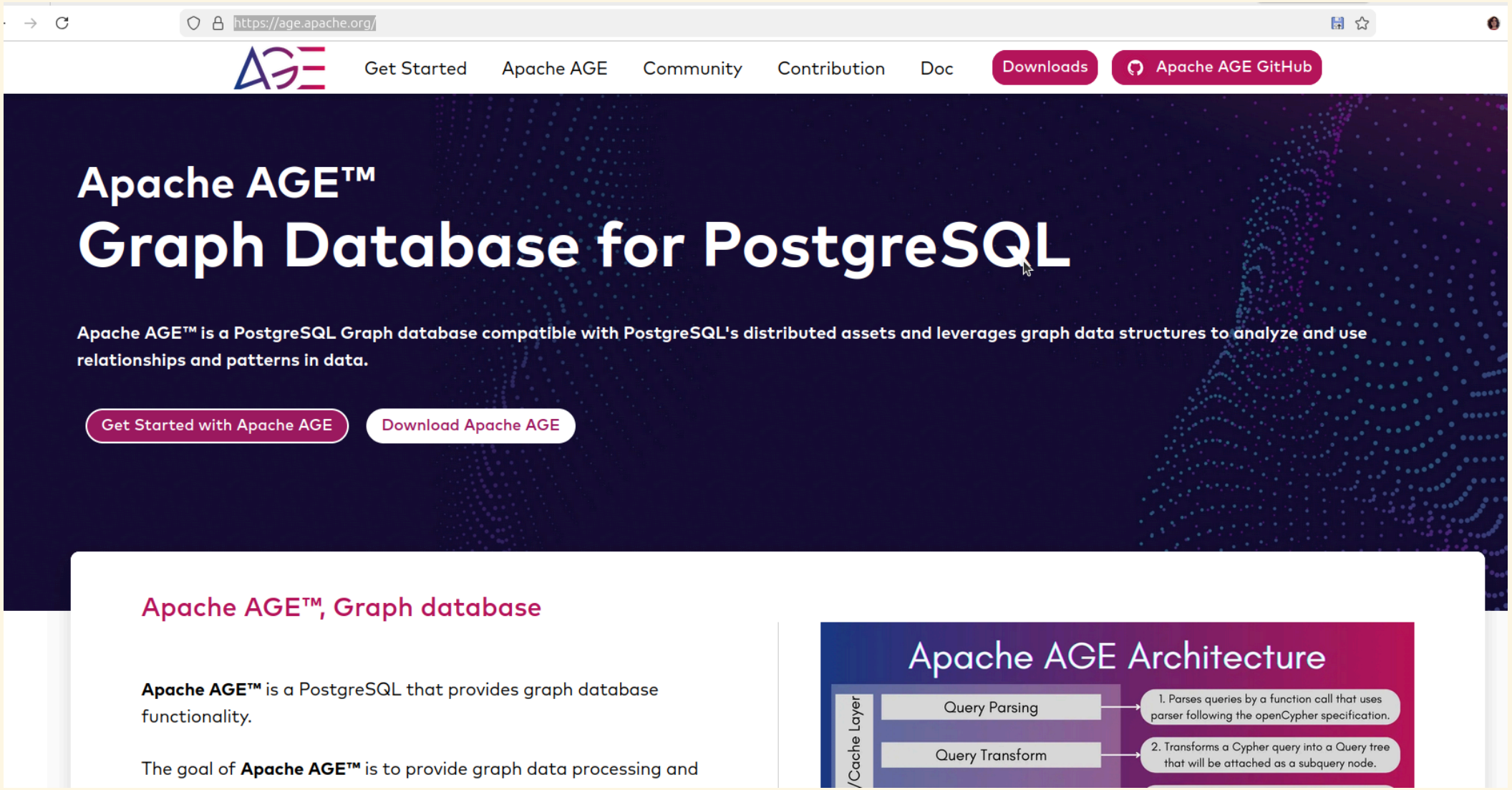
“Somebody who knows somebody who knows somebody who knows Henry Fnord”

NEO4J


neo4j.com
Aura Login
Partners
Company
Support
neo4j
Products
Use Cases
Developers
AI Systems
Learn
Pricing
Contact Us
Get Started Free
THE WORLD'S LEADING GRAPH INTELLIGENCE PLATFORM
Build Intelligent Apps and AI For
Supply Chain Management
AI-ready data by design
Start Building
Learn More
300k
Developers Building
80+
Fortune 100 Customers
170+
Partner Ecosystem
Cypher
--Find Fraud Ring 3-10 w/ transactions hop
MATCH ring=(a:Account)
((()-[:SENT]→()-
[:RECEIVES]-()){3,10}(a)
RETURN ring;
Agent Runtime
Orchestration
Generative AI Models
Tools
Relationship types
Cypher
MATCH
(manager:Role)-[:REPORTS_TO]→
(supervisor:Role) RETURN
supervisor.name AS Supervisor,
manager.name AS Manager ORDER
BY supervisor.name;
Nodes by label
36%
Movies by decade
Query Rate
--Find Fraud Ring 3-10 t
MATCH ring=(a:Account)
((()-[:PERFORMS]→()-
Customized, grouped bar chart
almart
Santander
COMCAST
BNP PARIBAS PERSONAL FINANCE
UBS
AIRBUS
BT Group
Boston Scientific
novo nordisk
dun & bradstreet

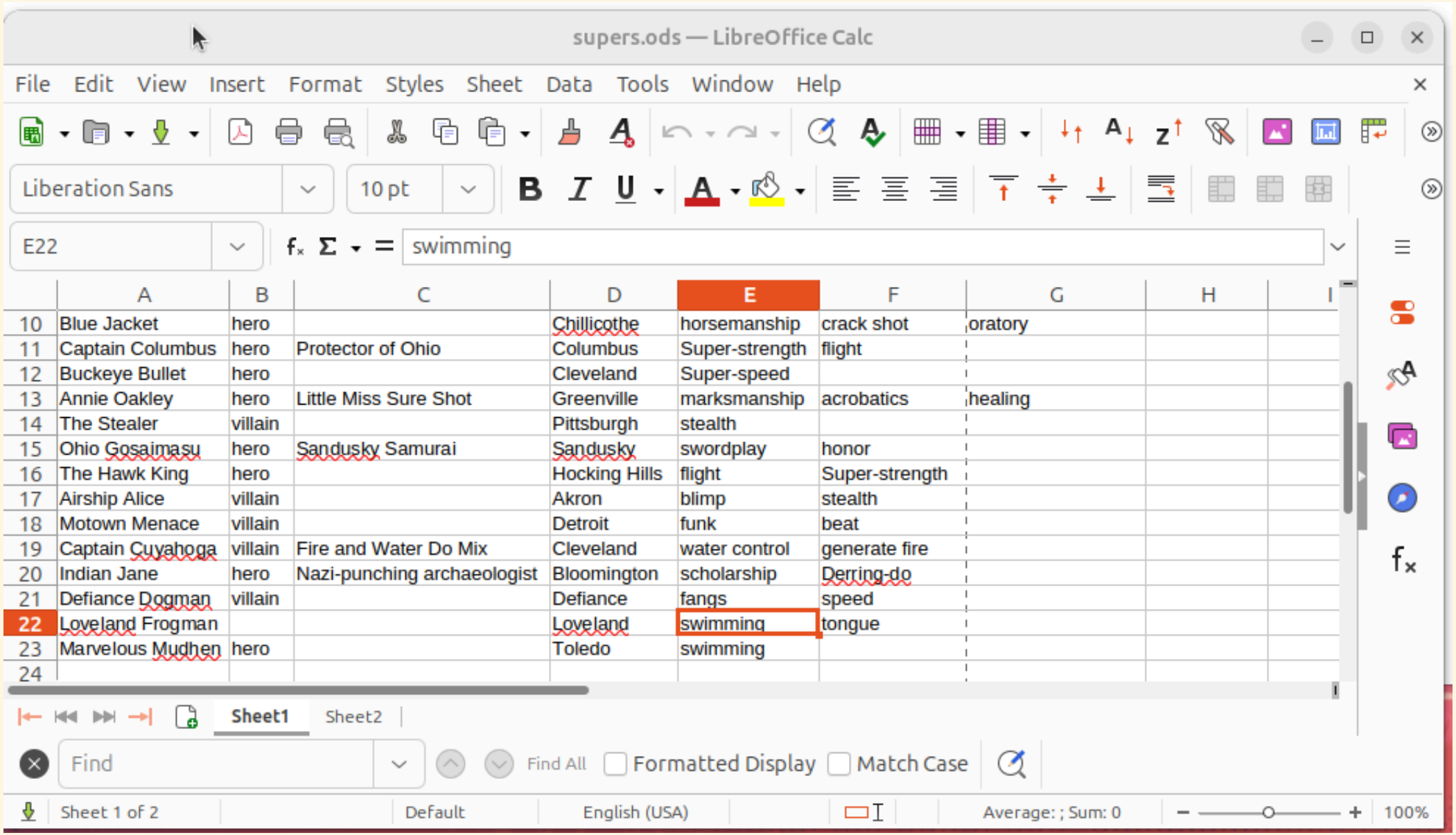
CHELNIK, THE POSTGRESQL ELEPHANT

APACHE AGE


https://age.apache.org/
Get Started
Apache AGE
Community
Contribution
Doc
Downloads
Apache AGE GitHub
Apache AGE™
Graph Database for PostgreSQL
Apache AGE™ is a PostgreSQL Graph database compatible with PostgreSQL's distributed assets and leverages graph data structures to analyze and use relationships and patterns in data.
Get Started with Apache AGE
Download Apache AGE
Apache AGE™, Graph database
Apache AGE™ is a PostgreSQL that provides graph database functionality.
The goal of Apache AGE™ is to provide graph data processing and
Apache AGE Architecture
Query Parsing
1. Parses queries by a function call that uses parser following the openCypher specification.
Query Transform
2. Transforms a Cypher query into a Query tree that will be attached as a subquery node.

GETTING INFORMATION

ASSUMED



	A	B	C	D	E	F	G	H	I
10	Blue Jacket	hero		Chillicothe	horsemanship	crack shot	oratory		
11	Captain Columbus	hero	Protector of Ohio	Columbus	Super-strength	flight			
12	Buckeye Bullet	hero		Cleveland	Super-speed				
13	Annie Oakley	hero	Little Miss Sure Shot	Greenville	marksmanship	acrobatics	healing		
14	The Stealer	villain		Pittsburgh	stealth				
15	Ohio Gosaimasu	hero	Sandusky Samurai	Sandusky	swordplay	honor			
16	The Hawk King	hero		Hocking Hills	flight	Super-strength			
17	Airship Alice	villain		Akron	blimp	stealth			
18	Motown Menace	villain		Detroit	funk	beat			
19	Captain Cuyahoga	villain	Fire and Water Do Mix	Cleveland	water control	generate fire			
20	Indian Jane	hero	Nazi-punching archaeologist	Bloomington	scholarship	Derring-do			
21	Defiance Dogman	villain		Defiance	fangs	speed			
22	Loveland Frogman			Loveland	swimming	tongue			
23	Marvelous Mudhen	hero		Toledo	swimming				
24									

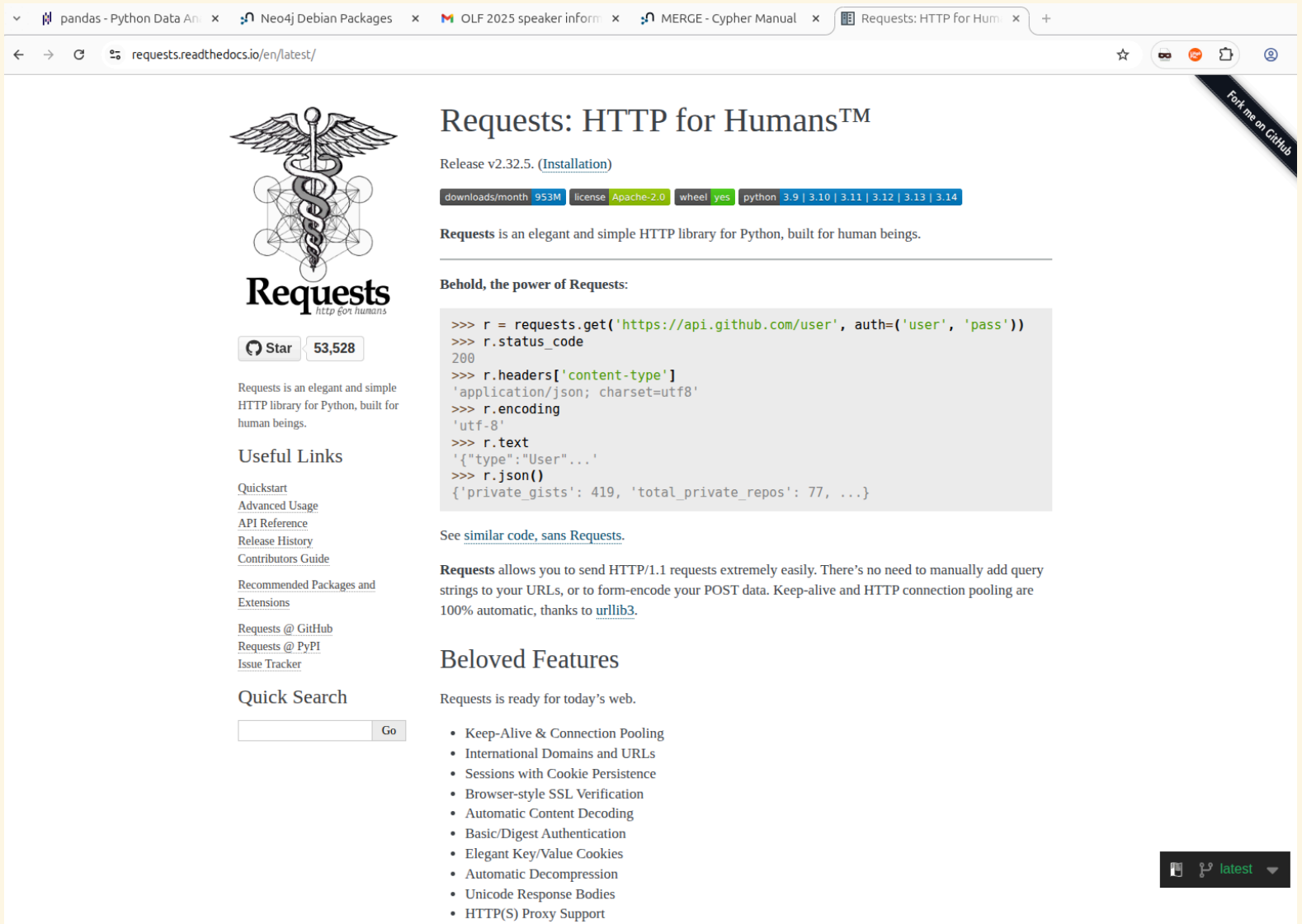
ALWAYS A SEQUEL

2023: Info was gathered, needed organization

2025: Gather the info

AUTOMATE!

REQUESTS


pandas - Python Data An
Neo4j Debian Packages
OLF 2025 speaker inform
MERGE - Cypher Manual
Requests: HTTP for Hum
requests.readthedocs.io/en/latest/
Requests
http for humans
Star 53,528
Requests is an elegant and simple HTTP library for Python, built for human beings.
Useful Links
Quickstart
Advanced Usage
API Reference
Release History
Contributors Guide
Recommended Packages and Extensions
Requests @ GitHub
Requests @ PyPI
Issue Tracker
Quick Search
Go
Requests: HTTP for Humans™
Release v2.32.5. (Installation)
downloads/month 953M
license Apache-2.0
wheel yes
python 3.9 | 3.10 | 3.11 | 3.12 | 3.13 | 3.14
Requests is an elegant and simple HTTP library for Python, built for human beings.
Behold, the power of Requests:
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type":"User"...'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
See similar code, sans Requests.
Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to urllib3.
Beloved Features
Requests is ready for today's web.
Keep-Alive & Connection Pooling
International Domains and URLs
Sessions with Cookie Persistence
Browser-style SSL Verification
Automatic Content Decoding
Basic/Digest Authentication
Elegant Key/Value Cookies
Automatic Decompression
Unicode Response Bodies
HTTP(S) Proxy Support
Fork me on GitHub
latest

GETTING A FILE: SUPERS.CSV

```
>>> import requests
>>> resp =
requests.get("http://localhost:8000/supers.csv")
>>> resp.text
'name,align,tagline,based_in,power_1,power_2,power_3\nT
Trilobite,,Caped
Cambrian,Silvania,armor,antennae,\nPutin
Bae,villain,Rasslinâ\x80\x99
Russophile,Mansfield,politics,wrestling,selective
blindness\nCardinal Virtue,hero,Redbird of
Righteousness,Cincinnati,flight,angry beak,moral purit
```

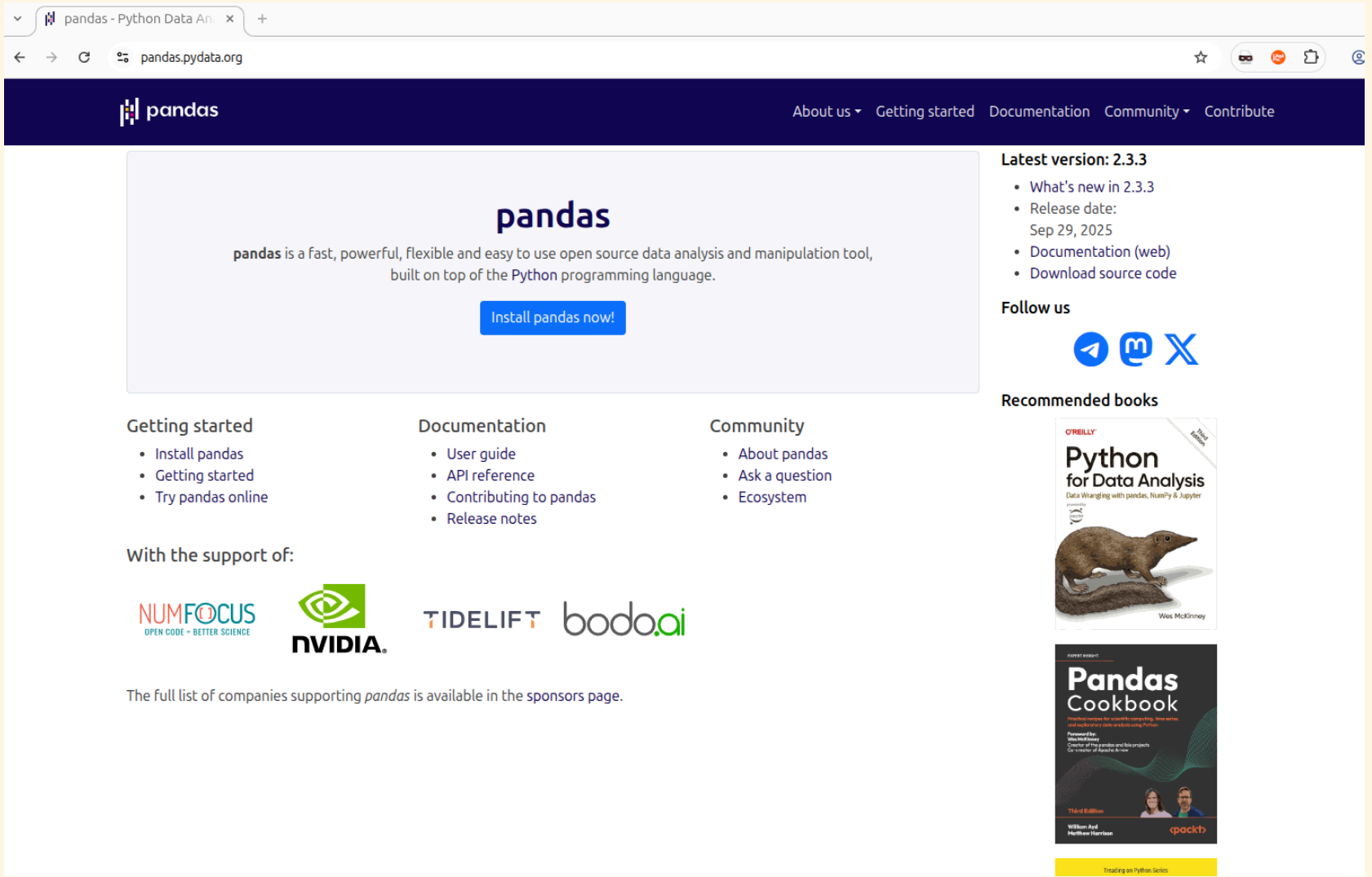
HTML TABLE

Wikipedia: List of Superpowered individuals in Ohio

```
<title>List of Superpowered individuals in Ohio -
Wikipedia</title>
<style>:root{--sf-img-1:
url("
-sf-img-2:
url("
-sf-img-3:
url("
-sf-img-4:
url("
-sf-img-5:
```

PANDAS

Dataframe processing library



DATA IMPORT FUNCTIONS

```
>>> import pandas as pd
>>> pd.read_
pd.read_clipboard(  pd.read_gbq(
pd.read_parquet(    pd.read_sql_query(
pd.read_csv(        pd.read_hdf(
pd.read_pickle(     pd.read_sql_table(
pd.read_excel(      pd.read_html(       pd.read_sas(
pd.read_stata(
pd.read_feather(    pd.read_json(       pd.read_spss(
pd.read_table(
pd.read_fwf(        pd.read_orc(        pd.read_sql(
```

`read_html()` READS *ALL* TABLES

```
>>> import pandas as pd

>>> URL = 'http://localhost:8000/ohio-supers.html'
>>> frames = pd.read_html(URL)
>>> len(frames)
3
```

PPRINT

```
>>> import pprint
>>> pprint.pprint(frames)

[    Unnamed: 0                Name Alignment
Tagline       Based in          Power 1           Power
2              Power 3
0             0     The Trilobite        NaN
Caped Cambrian       Silvania            armor
antennae                   NaN
1             1         Putin Bae   villain
Rasslin' Russophile      Mansfield         politics
```

DUMP

```
>>> df.to_
df.to_clipboard(  df.to_excel(      df.to_hdf(
df.to_latex(      df.to_orc(        df.to_pickle(
df.to_stata(      df.to_xarray()
df.to_csv(        df.to_feather(    df.to_html(
df.to_markdown(   df.to_parquet(    df.to_records(
df.to_string(     df.to_xml(
df.to_dict(       df.to_gbq(        df.to_json(
df.to_numpy(      df.to_period(     df.to_sql(
df.to_timestamp(        import pandas as pd
```

MORE SPECIFIC JSONL

```
import sys

import jsonlines
import pandas as pd

URL = "http://localhost:8000/ohio-supers.html"

def json_rows(row):
    if row["Based in"]:
        yield {
            "entity_1": {
```

OUTPUT

```
$ uv run get-html.py
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Place", "name":
"Silvania"}, "relationship": "based_in"}
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Power", "name":
"armor"}, "relationship": "has"}
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Power", "name":
"antennae"}, "relationship": "has"}
{"entity_1": {"type": "Person", "name": "Putin Bae"}
```

FROM JSONL TO NEO4J

GETTING NEO4J

neo4j.com -> cloud product

https://debian.neo4j.com/

```
wget -O -
https://debian.neo4j.com/neotechnology.gpg.key | sudo
apt-key add -
echo 'deb https://debian.neo4j.com stable latest' |
sudo tee /etc/apt/sources.list.d/neo4j.list
sudo apt-get update
sudo apt-get install neo4j
```

Docker container available

STARTING

```
sudo -i -u neo4j neo4j-admin dbms set-initial-password
4justice
sudo -i -u neo4j start
open http://localhost:7474
```

SHOW FULL GRAPH

```
MATCH (n) RETURN n;
```

INSERTING DATA

- CYPHER queries
- .csv files
- `neo4j` Python library

WRITE & EXECUTE CYPHER QUERIES

jsonl2neo4j.py

```
import sys
import jsonlines
from neo4j import GraphDatabase, RoutingControl

URI = "neo4j://localhost:7687"
AUTH = ("neo4j", "4justice")

def save(driver, row):
    relationship = row["relationship"].upper()
    type2 = row["entity_2"]["type"]
    qry = (
```

(THIS)-[:TO]->(THAT)

```
MERGE (entity_1:Person {name: "Captain Columbus"})
MERGE (entity_2:Power {name: "Super-strength"})
MERGE (entity_1)-[:HAS]->(entity_2)
```

FULL PIPELINE

```
uv run get-html.py > supers.jsonl
head supers.jsonl
uv run jsonl2neo4j.py supers.jsonl

{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Place", "name":
"Silvania"}, "relationship": "based_in"}
{"entity_1": {"type": "Person", "name": "The
Trilobite"}, "entity_2": {"type": "Power", "name":
"armor"}, "relationship": "has"}
{"entity_1": {"type": "Person", "name": "The
```

VIEW

```
MATCH (n) RETURN n;
```

OTHER DATA SOURCES

```
>>> pd.read_
pd.read_clipboard(  pd.read_gbq(
pd.read_parquet(    pd.read_sql_query(
pd.read_csv(        pd.read_hdf(
pd.read_pickle(     pd.read_sql_table(
pd.read_excel(      pd.read_html(       pd.read_sas(
pd.read_stata(
pd.read_feather(    pd.read_json(       pd.read_spss(
pd.read_table(
pd.read_fwf(        pd.read_orc(        pd.read_sql(
pd.read_xml(
```

UNCOMPILED DATA

INNOVATORS OF INIQUITY

Henry Fnord's Model T2: Judgement Day

WEB SCRAPING

Innovators of Iniquity

- Requests + BeautifulSoup
- Scrapy

GET SCRAPY

```
$ uv init --bare -p 3.13
Initialized project `ccbus`
$ uv add scrapy
Using CPython 3.14.0rc1 interpreter at:
/home/catherine/.local/bin/python3.14
Creating virtual environment at: .venv
Resolved 39 packages in 760ms
Prepared 2 packages in 407ms
Installed 37 packages in 222ms
+ attrs==25.4.0
+ automat==25.4.16
```

NEW PROJECT

```
$ uv run scrapy startproject ccbus_scraper
New Scrapy project 'ccbus_scraper', using template
directory
'/home/catherine/ccbus/.venv/lib/python3.14/site-
packages/scrapy/templates/project', created in:
    /home/catherine/ccbus/ccbus_scraper

You can start your first spider with:
    cd ccbus_scraper
    scrapy genspider example example.com
```

WHAT DO WE HAVE?

```
$ tree ccbus_scraper/
ccbus_scraper/
├── ccbus_scraper
│   ├── __init__.py
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       └── __init__.py
└── scrapy.cfg
```

EMPTY SPIDER

```
import scrapy

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
    ]

    def parse(self, response):
        if response.status == 200:
            self.log(f"Scraped {response.url}")
```

RUN

```
/ccbus_scraper$ uv run scrapy crawl ioi
2025-11-27 15:17:16 [scrapy.utils.log] INFO: Scrapy
2.13.4 started (bot: ccbus_scraper)
2025-11-27 15:17:16 [scrapy.utils.log] INFO: Versions:
{'lxml': '6.0.2',
'libxml2': '2.14.6',
'cssselect': '1.3.0',
'parsel': '1.10.0',
'w3lib': '2.3.1',
'Twisted': '25.5.0',
'Python': '3.14.0rc1 (main, Jul 23 2025, 00:37:53)
```

FIND ELEMENTS OF INTEREST

Developer Tools (Ctrl+Shift+I)

Right-click; Inspect

Right-click; Copy; CSS Path

COPY CSS PATH

```
#app > div.container.px-4.py-6.\32 xl\:px-0 >
div.flex-grid.md\:flex-wrap.md\:-mx-6.py-6 > div:nth-
child(1) > div > div.media-card__header > h2 > a
```

Less specific will do!

SCRAPY SHELL

```
~/ccbus/ccbus_scraper$ uv run scrapy shell
http://localhost:8000/pod.html
2025-11-27 15:47:19 [scrapy.utils.log] INFO: Scrapy
2.13.4 started (bot: ccbus_scraper)
...
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
>>>
```

EXPERIMENT TO FIND

```
>>> response.css("div.media-card__header")
[<Selector query="descendant-or-self::div[@class and
contains(concat(' ', normalize-space(@class), ' '), '
media-card__header ')]" data='<div class="media-
card__header">\n    ...'>, <Selector
query="descendant-or-self::div[@class and
contains(concat(' ', normalize-space(@class), ' '), '
media-card__header ')]" data='<div class="media-
card__header">\n    ...'>, <Selector
query="descendant-or-self::div[@class and
contains(concat(' ', normalize-space(@class), ' '), '
```

::TEXT

```
>>> response.css("div.media-card__header
a::text").getall()
['\n                          Holiday Villainy Season
2025\n                        ', '\n
Interview: The Wrong Brothers\n                        ',
'\n                         Interview: Putin Bae\n
', '\n                          Interview: The
Trilobite\n                          ']
```

IOI.PY

```
import scrapy

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
    ]

    def parse(self, response):
        for line in response.css("div.media-
card  header a::text").getall():
```

FEED EXPORTS

```
uv run scrapy crawl ioi -O ioi.jsonl
cat ioi.jsonl

{"person1": "Henry Fnord", "person2": "The Wrong
Brothers", "relationship": "allied"}
{"person1": "Henry Fnord", "person2": "Putin Bae",
"relationship": "allied"}
{"person1": "Henry Fnord", "person2": "The Trilobite",
"relationship": "allied"}
```

OK, BUT "SPIDER"?

~~This meeting could have been an email~~

This spider could have been `requests.get()` + `BeautifulSoup`

FINDING “NEXT”

```
uv run scrapy shell localhost:8000/pod.html

response.css('a.btn')
```

`::attr()` pseudo-element

```
response.css('a.btn::attr(href)').getall()

response.css('nav.justify-center
a.btn::attr(href)').getall()
```

YIELD ITEM (DATA) *OR* REQUEST

```
import scrapy

EP_TITLE = "div.media-card__header a::text"
EP_BODY = "div.media-card__body p::text"
NAV_BUTTON = "nav.justify-center a.btn::attr(href)"

class IOISpider(scrapy.Spider):
    name = "ioi"
    start_urls = [
        "http://localhost:8000/pod.html",
    ]
```

OH, NO, PREV!

Infinite loop?

Repetition protection

CARDINAL VIRTUE!?!


```
[
{"person1": "Henry Fnord", "person2": "The Wrong
Brothers", "relationship": "allied"},
{"person1": "Henry Fnord", "person2": "Putin Bae",
"relationship": "allied"},
{"person1": "Henry Fnord", "person2": "The Trilobite",
"relationship": "allied"},
{"person1": "Henry Fnord", "person2": "Motown Menace",
"relationship": "allied"},
{"person1": "Henry Fnord", "person2": "Cardinal
Virtue", "relationship": "allied"}
```

SENTIMENT ANALYSIS

Training a model could be a whole talk

VADER

MODEL DOWNLOAD

```
>>> import nltk
>>> nltk.download('vader_lexicon')
```

PASS BODY TO VADER

```
def parse(self, response):

    for card in response.css("div.media-card"):
        line = card.css(EP_TITLE).get()
        # `card` supports css paths like `response`
does
        pieces = line.split("Interview:")
        if len(pieces) == 2:
            body =
"\n".join(card.css(EP_BODY).getall())
            sentiment = analyzer.polarity_scores(body)
```

-1 TO +1


```
[
{"person1": "Henry Fnord", "person2": "The Wrong
Brothers", "sentiment": 0.7184},
{"person1": "Henry Fnord", "person2": "Putin Bae",
"sentiment": 0.69},
{"person1": "Henry Fnord", "person2": "The Trilobite",
"sentiment": -0.1306},
{"person1": "Henry Fnord", "person2": "Motown Menace",
"sentiment": 0.2481},
{"person1": "Henry Fnord", "person2": "Cardinal
Virtue", "sentiment": -0.8163}
```

PICK A THRESHOLD

```
relationship = "allied" if sentiment > -0.5 else
"opposed"
yield {
    "person1": "Henry Fnord",
    "person2": pieces[1].strip(),
    "relationship": relationship,
}
```

REFORMAT FOR JSONL

```
def parse(self, response):
    for card in response.css("div.media-card"):
        line = card.css(EP_TITLE).get()
        # `card` supports css paths like `response`
does
        piecestext= line.split("Interview:")
        if len(pieces) == 2:
            body =
"\n".join(card.css(EP_BODY).getall())
            sentiment = analyzer.polarity_scores(body)
["compound"]
```

IMPORT TO NEO4J

`-O` overwrites, `-o` appends

```
uv run get-html.py > supers.jsonl
cd ccbus_scraper
uv run scrapy crawl ioi -o ../supers.jsonl
cd ..
uv run jsonl2neo4j.py supers.jsonl

MATCH (n) RETURN (n)
```

PARSING PROSE

CLEVELAND.COM

Cardinal Virtue drives Nell Strongarm from skies over Findlay

CRAWL ALL THE ARTICLES

```
import scrapy

class ClevelandSpider(scrapy.Spider):
    name = "cleveland"
    start_urls = [
        "http://localhost:8000/strongarm.html",
    ]

    def parse(self, response):
        article_content = response.css("div.entry-
content")
```

```
$ uv run scrapy crawl cleveland -O cleveland.jsonl
$ cat cleveland.jsonl
{"url": "http://localhost:8000/strongarm.html"}
{"url": "http://localhost:8000/virtue-strongarm.html"}
{"url": "http://localhost:8000/stealer.html"}
{"url": "http://localhost:8000/virtue-jane.html"}
{"url": "http://localhost:8000/strongarm.html"}
```

GET LOCATION

```
import re
import scrapy

DATELINE = re.compile(r"\n([^\n]*?) \-\- ")

class ClevelandSpider(scrapy.Spider):
    name = "cleveland"
    start_urls = [
        "http://localhost:8000/strongarm.html",
    ]
```

```
$ uv run scrapy crawl cleveland -O cleveland.jsonl
$ cat cleveland.jsonl
{"url": "http://localhost:8000/strongarm.html",
"location": "Toledo"}
{"url": "http://localhost:8000/virtue-strongarm.html",
"location": "Findlay"}
{"url": "http://localhost:8000/stealer.html",
"location": "Georgetown"}
{"url": "http://localhost:8000/virtue-jane.html",
"location": "Indianapolis"}
{"url": "http://localhost:8000/strongarm.html"
```

ML: NAMED ENTITY RECOGNITION

Choosing flair

- simple
- no training

```
>>> from flair.data import Sentence
>>> from flair.nn import Classifier
>>> tagger = Classifier.load("ner")
2025-11-30 23:28:49,990 SequenceTagger predicts:
Dictionary with 20 tags: <unk>, O, S-ORG, S-MISC, B-
PER, E-PER, S-LOC, B-ORG, E-ORG, I-PER, S-PER, B-MISC,
I-MISC, E-MISC, I-ORG, B-LOC, E-LOC, I-LOC, <START>,
<STOP>
>>> text = Sentence("The fearless Captain Columbus
protects Ohio from Henry Fnord.")
>>> tagger.predict(text)
```

LABELLED TEXT

```
import itertools
import re

import scrapy
from flair.data import Sentence
from flair.nn import Classifier

DATELINE = re.compile(r"\n([^\n]*?) \-\- ")

NAV_BUTTON = "nav.justify-center a.btn::attr(href)"
tagger = Classifier.load("ner")
```

```
$ uv run scrapy crawl cleveland -O cleveland.jsonl
$ cat cleveland.jsonl
{"url": "http://localhost:8000/strongarm.html",
"entities": "['Span[0:2]: \"Nell Strongarm\"'/'PER'
(0.7241), 'Span[3:5]: \"Crime Capsule\"'/'MISC'
(0.8431), 'Span[7:8]: \"Toledo\"'/'LOC' (0.9944),
'Span[10:11]: \"TOLEDO\"'/'LOC' (0.5525),
'Span[14:15]: \"Toledo\"'/'LOC' (0.9999),
'Span[22:24]: \"Nell Strongarm\"'/'PER' (0.7919),
'Span[25:27]: \"Crime Capsule\"'/'MISC' (0.8985),
'Span[51:53]: \"Middlegrounds Metropark\"'/'LOC'
```

ACTUAL NEO4J-READY FORMAT

```
import itertools
import re

import scrapy
from flair.data import Sentence
from flair.nn import Classifier

DATELINE = re.compile(r"\n([^\n]*?) \-\- ")
NAV_BUTTON = "nav.justify-center a.btn::attr(href)"
tagger = Classifier.load("ner")
```

```
$ uv run scrapy crawl cleveland -O cleveland.jsonl
$ cat cleveland.jsonl
{"entity_1": {"type": "Person", "name": "Nell
Strongarm"}, "entity_2": {"type": "Place", "name":
"Toledo"}, "relationship": "active_in"}
{"entity_1": {"type": "Person", "name": "Virtue"},
"entity_2": {"type": "Place", "name": "Findlay"},
"relationship": "active_in"}
{"entity_1": {"type": "Person", "name": "Nell
Strongarm"}, "entity_2": {"type": "Place", "name":
"Findlay"}, "relationship": "active_in"}
```

LLM

OLLAMA

Locally run LLM models

- Can pick an open-source model
- Control power use
- No authorization

PYPI'S `ollama` MODULE

```
import ollama
response = ollama.generate(model="llama3.2",
prompt=PROMPT+text)
print(response.response)
```

READ AN ARTICLE

Specify the format we need

```
PROMPT = """
I'm going to give you the text of a newspaper article
that involves
one or more superpowered beings (heroes and/or
villains).  Please
briefly summarize the article in JSON format, with an
array of
objects (no more than 5) using a format following this
example.
entity_1 should always be a Person.  Relationship
should be one
```

```
import json

import ollama
import scrapy

class ClevelandLLMSpider(scrapy.Spider):
    name = "cleveland_llm"
    start_urls = [
        "http://localhost:8000/virtue-strongarm.html",
    ]
```

{"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "Bird", "name": "Cardinal Virtue"}, "relationship": "attacked"}
{"entity_1": {"type": "Astro-Thug", "name": "Nell Strongarm"}, "entity_2": {"type": "Crime Capsule", "name": "N/A"}, "relationship": "used"}
{"entity_1": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "entity_2": {"type": "Astro-Thug", "name": "Nell Strongarm"}, "relationship": "defended"}
{"entity_1": {"type": "Crime Capsule", "name": "N/A"}, "entity_2": {"type": "Astro-Thug", "name": "Nell

DID THIS WORK?

- NER… usually
- NER classification… sometimes
- Valid syntax… yes
- Desired format… no
- Accurate summary… sometimes

RECOGNIZED FORMAT

```
The output should be a list where each element adheres
to the
pydantic RelationshipListModel, as defined thus:

from pydantic import BaseModel, Field, constr
from typing import List

class EntityPerson(BaseModel):
    name: str
    type: str = Field("Person", const=True)
```

{"entity_1": {"type": "Person", "name": "Nell Strongarm"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "attacked"}
{"entity_1": {"type": "Nell Strongarm", "name": "Astro-Thug"}, "entity_2": {"type": "Cardinal Virtue", "name": "Cardinal Virtue"}, "relationship": "defended against"}
{"entity_1": {"type": "Person", "name": "Cardinal Virtue"}, "entity_2": {"type": "Person", "name": "Indiana Jane"}, "relationship": "spoke"}

IMPROVE THIS WITH

- Prompt engineering
- Choice of models

LET'S USE IT

ASSEMBLE

```
MATCH (n) DETACH DELETE n;

uv run get-html.py > supers.jsonl
cd ccbus_scraper
uv run scrapy crawl ioi -o ../supers.jsonl
uv run scrapy crawl cleveland -o ../supers.jsonl
cd ..
uv run jsonl2neo4j.py supers.jsonl ~/ccbus$

MATCH (n) RETURN n;
```

THANK YOU!

Catherine Devlin

https://github.com/catherinedevlin/data4heroes