Module 2. - Week I

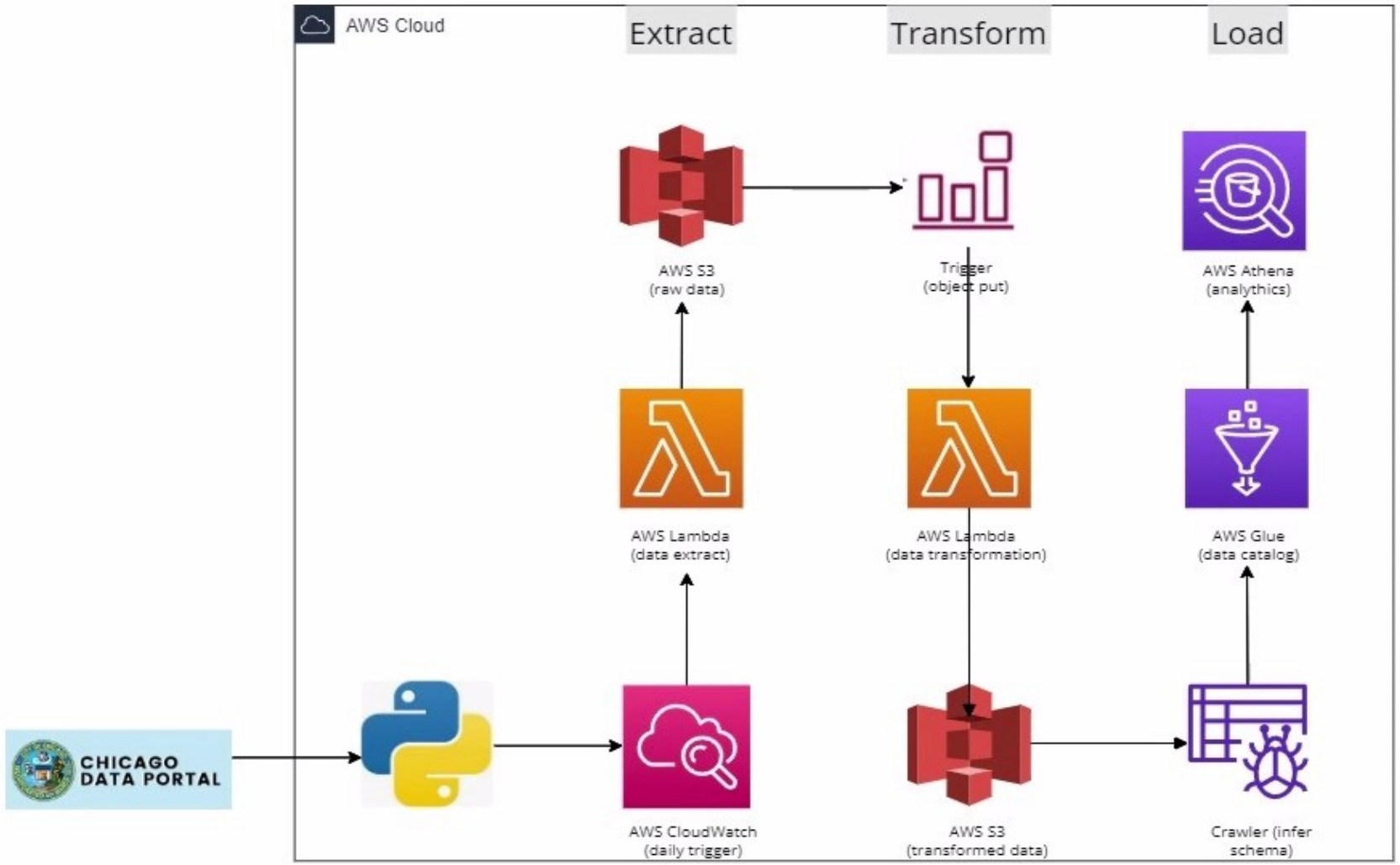
Chapter I.

Data Engineer

Trainer: Balazs Balogh

UBIX
INSTITUTE OF TECHNOLOGY

Course overview



CUBIX
INSTITUTE OF TECHNOLOGY

Course overview

What will you do?

- Get familiar with VSCode IDE (optional)
- Work with Amazon Web Services (AWS)
- Understand JSON handling in Python
- Utilize web scraping
- Use public APIs to get data and store it on AWS S3
- Create AWS Lambda functions to clean and transform data
- Use AWS Glue to crawl through data
- Use AWS Athena for querying
- Create visualisations on your computer from our full data

AWS usage costs

STORAGE

Free Tier — 12 MONTHS FREE

Amazon S3

5 GB

of standard storage

Secure, durable, and scalable object storage infrastructure.

5 GB of Standard Storage

20,000 Get Requests

2,000 Put Requests

COMPUTE

Free Tier — ALWAYS FREE

AWS Lambda

1 Million

free requests per month

Compute service that runs your code in response to events and automatically manages the compute resources.

1,000,000 free requests per month

Up to 3.2 million seconds of compute time per month

- S3 and Lambda is basically free for our project.
- If you exceed the 5GB limit, it will be $0.023 per GB onwards.
- One month’s data is around 800 Mb.

CUBIX
INSTITUTE OF TECHNOLOGY

AWS usage costs

Glue - Data Catalog

Storage:

- Free for the first million objects stored
- $1.00 per 100,000 objects stored above 1M, per month

Requests:

- Free for the first million requests per month
- $1.00 per million requests above 1M in a month

Glue - Crawler

- $0.44 per DPU-Hour, billed per second, with a 10-minute minimum per crawler run

- Glue is also free in terms of the Data Catalog.

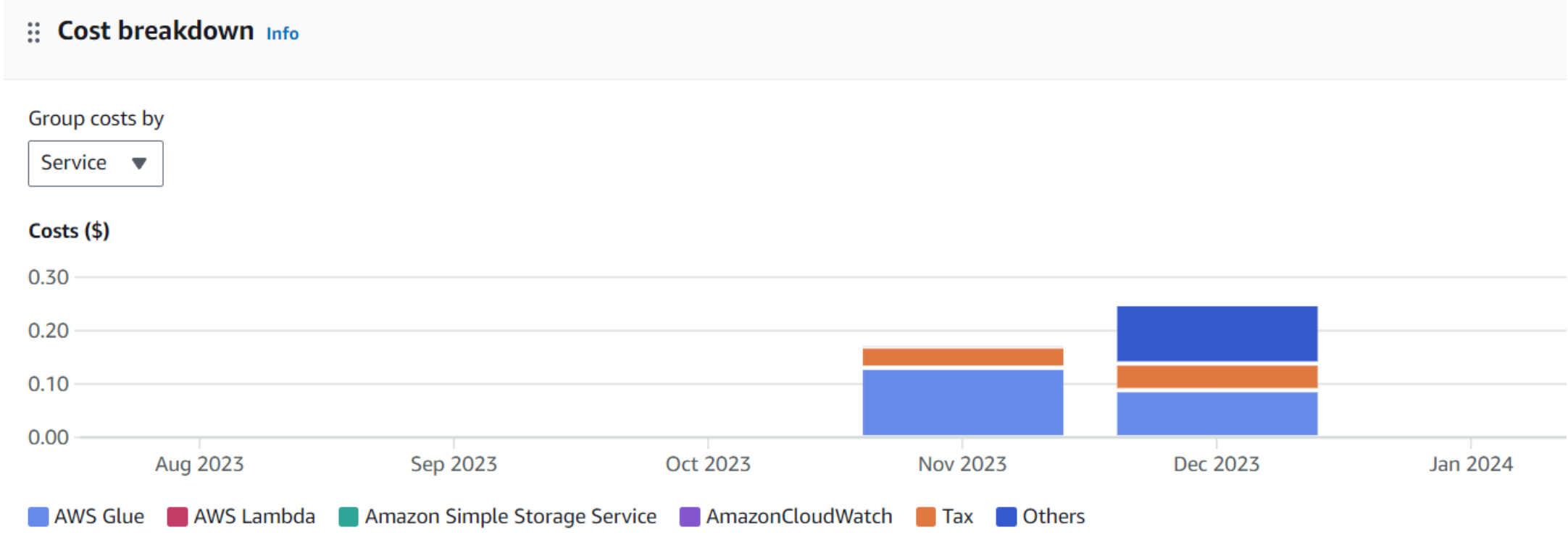
AWS usage costs

- Athena – SQL queries

SQL queries	$6.75 per TB of data scanned. Save up to 90% on per query and get better performance by compressing, partitioning, and converting your data into columnar formats.

AWS usage costs

- Sample spending for two full months.
- First month was 0.20$, second was 0.30$.



CUBIX
INSTITUTE OF TECHNOLOGY

VSCode – virtual environment

- The python virtual environment (venv) is crucial for managing dependencies, and isolating project-specific packages.
- It allows you to create a self-contained environment with its own set of packages, ensuring that your project runs with the correct versions and avoids conflicts with other projects.

VSCode – virtual environment

- For creating a venv, go into your project's folder in cmd and use the
  - python –m venv venv command
- You can name it as you want, but using "venv" is a common practice, especially when multiple people work on the same project, so you don't have to have different names for it in the gitignore file.
- When it is created, you have to activate it, with the
  - venv\Scripts\activate command
- For installing libraries, use
  - pip install library-name
- For deactivating the virtual environment use
  - deactivate

City of Chicago API

- https://data.cityofchicago.org/ - register here.
- Straight to the Taxi data:
  - https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew
- Currently 211.000.000 rows, and 23 columns.
- Each row is a trip, by default it’s coming in JSON format.

Python JSON handling

- For reading the file, use
  - with open(path, “r”) as file:

```
import json


file_path = "../../data/json_handling/spotify_playlist.json"

with open(file_path, "r") as json_file:
    data = json.load(json_file)
```

Python JSON handling

- Then check the data.
- This data is a sample from the Spotify Global Top 50 playlist, it consists tracks and artists.

```
data
```


```
{'status': True,
 'contents': {'totalCount': 50,
  'items': [{'type': 'track',
    'id': '01qFKNWq73UfEslI0GvumE',
    'name': '3D (feat. Jack Harlow)',
    'shareUrl': 'https://open.spotify.com/track/01qFKNWq73UfEslI0GvumE',
    'durationMs': 201812,
    'durationText': '03:21',
    'discNumber': 1,
    'trackNumber': 1,
    'playCount': 30569756,
    'artists': [{'type': 'artist',
      'id': '6HaGTQPmzraVmaVxvz6EUc',
      'name': 'Jung Kook',
      'shareUrl': 'https://open.spotify.com/artist/6HaGTQPmzraVmaVxvz6EUc'},
     {'type': 'artist',
      'id': '2LIk90788K0zvyj2JJVwkJ',
      'name': 'Jack Harlow',
      'shareUrl': 'https://open.spotify.com/artist/2LIk90788K0zvyj2JJVwkJ'}],
```


- In VSCode at the bottom of the cell's output you can open the result in a new tab.

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

CUBIX
INSTITUTE OF TECHNOLOGY

Python JSON handling

- Step inside the file like you step in a set of folders.
- The data's contents/items/name will give you the name of the song (check the previous slide).

```
    data["contents"]["items"]["name"]
```

```
'3D (feat. Jack Harlow)'
```

Python JSON handling

- Getting all the elements of the file, you have to use a for loop.
- Print out the song names and the play count.

```
    for track in data["contents"]["items"]:
        print(track["playCount"])
        print(track["name"])
        print()
```

```
30569756
3D (feat. Jack Harlow)

825957263
Seven (feat. Latto) (Explicit Ver.)

356592711
Paint The Town Red
```

Web scraping

- Web scraping is the process of collecting and parsing raw data from the Web.
- You will download the Community Areas of Chicago from Wikipedia
  - https://en.wikipedia.org/wiki/Community_areas_in_Chicago
  - You are aiming for the 01 – Rogers Park, 02 – West Ridge, etc. table
- You will use the beautifulsoup / pandas / requests python libraries, install it in your venv
  - pip install beautifulsoup4
  - pip install requests - Requests is a simple, yet elegant, HTTP library.

Chicago community areas by number, population, and area[8]

No.	Name	Population (2020)[10]	Area[9] (sq mi.)	Area[9] (km^2)	Density (/sq mi.)	Density (/km^2)
01	Rogers Park	55,628	1.84	4.77	30,232.61	11,672.81
02	West Ridge	77,122	3.53	9.14	21,847.59	8,435.36
03	Uptown	57,182	2.32	6.01	24,647.41	9,516.37
04	Lincoln Square	40,494	2.56	6.63	15,817.97	6,107.32
05	North Center	35,114	2.05	5.31	17,128.78	6,613.42
06	Lake View	103,050	3.12	8.08	33,028.85	12,752.44

CUBIX
INSTITUTE OF TECHNOLOGY

Web scraping

- To parse the Wikipedia page
  - first get the page with requests
  - then use Beatifulsoup() to

```
from bs4 import BeautifulSoup
import pandas as pd
import requests
```

```
response = requests.get(url="https://en.wikipedia.org/wiki/Community_areas_in_Chicago")

soup = BeautifulSoup(response.content, "html.parser")
```

- Now you can print the header for testing.

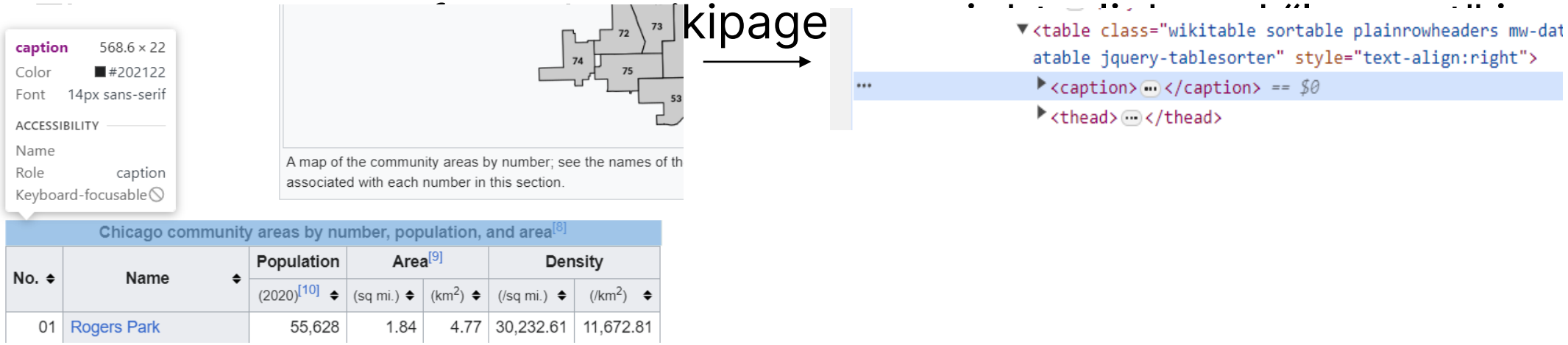
```
title = soup.find(id="firstHeading")
print(title.string)
```

```
Community areas in Chicago
```

CUBIX
INSTITUTE OF TECHNOLOGY

Web scraping

- With soup.find you can look for specific types. Here we are looking for a table, called “wikitable sortable plainrowheaders mw-datatable”.




```
table = soup.find("table", class_="wikitable sortable plainrowheaders mw-datatable")
table
```

CUBIX
INSTITUTE OF TECHNOLOGY

Web scraping

- You can iterate through the “table” variable, which holds the Community Areas, and gather the Area Code and the Community Name to a list of dictionaries.

```
data = []

for row in table.tbody.find_all("tr")[2:-1]:
    cells = row.find_all("td")
    header_cell = row.find("th")

    area_code = cells[0].get_text(strip=True)
    community_name = header_cell.a.get_text(strip=True)

    data.append({"area_code": area_code, "community_name": community_name})
```

- Output:

```
[{'area_code': '01', 'community_name': 'Rogers Park'},
 {'area_code': '02', 'community_name': 'West Ridge'},
 {'area_code': '03', 'community_name': 'Uptown'},
 {'area_code': '04', 'community_name': 'Lincoln Square'},
 {'area_code': '05', 'community_name': 'North Center'},
 {'area_code': '06', 'community_name': 'Lake View'},
 {'area_code': '07', 'community_name': 'Lincoln Park'},
 {'area_code': '08', 'community_name': 'Near North Side'},
 {'area_code': '09', 'community_name': 'Edison Park'},
 {'area_code': '10', 'community_name': 'Norwood Park'},
```

Web scraping

- Create a DataFrame out of it, and save it in a csv file.

```
community_areas = pd.DataFrame(data)

community_areas["area_code"] = community_areas["area_code"].astype("int")
```

```
community_areas.to_csv("community_areas_master.csv", index=False)
```