Introduction to ETL in Python

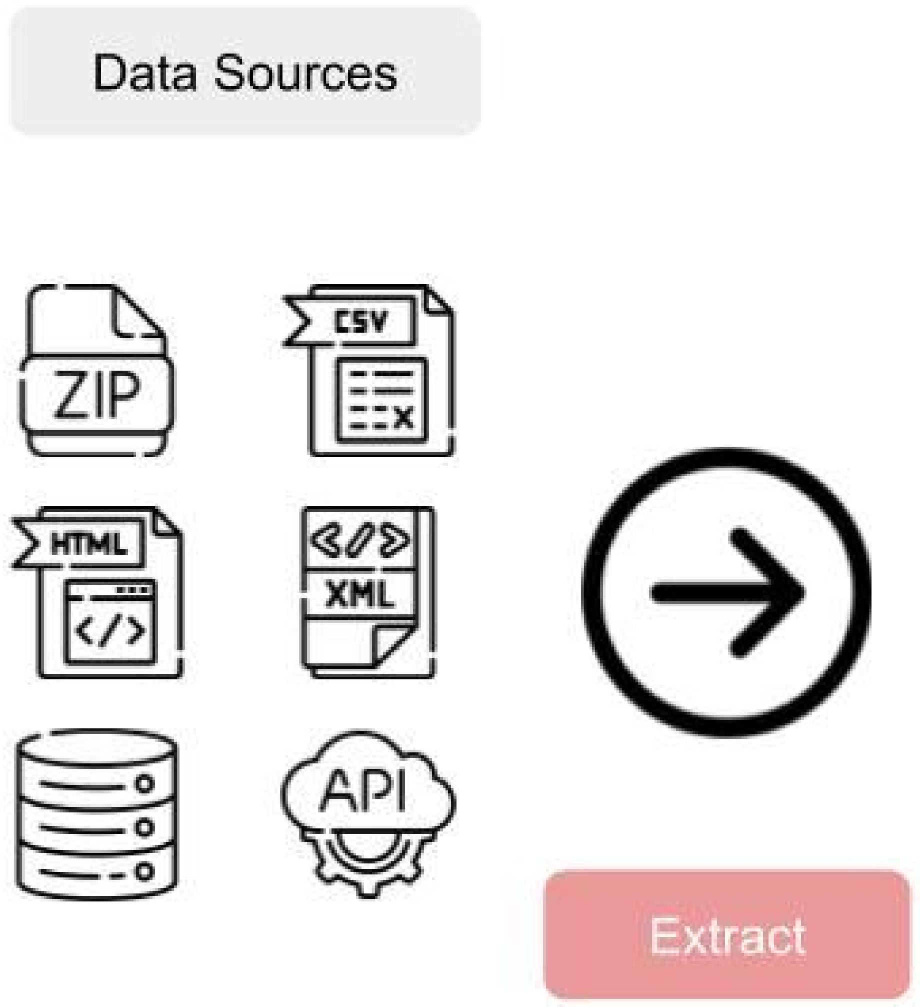
ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

datacamp

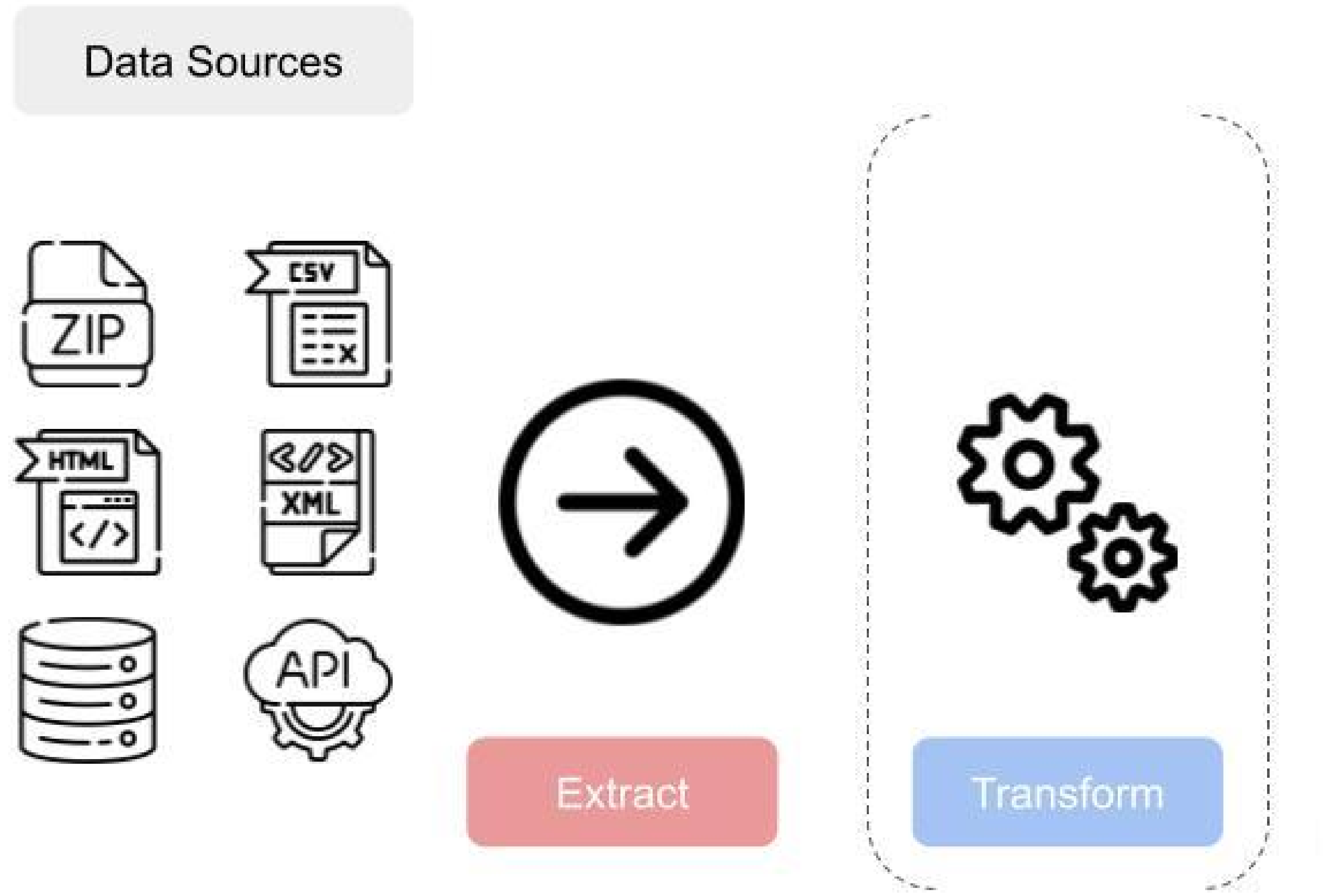
What is ETL?

Extract, Transform, Load



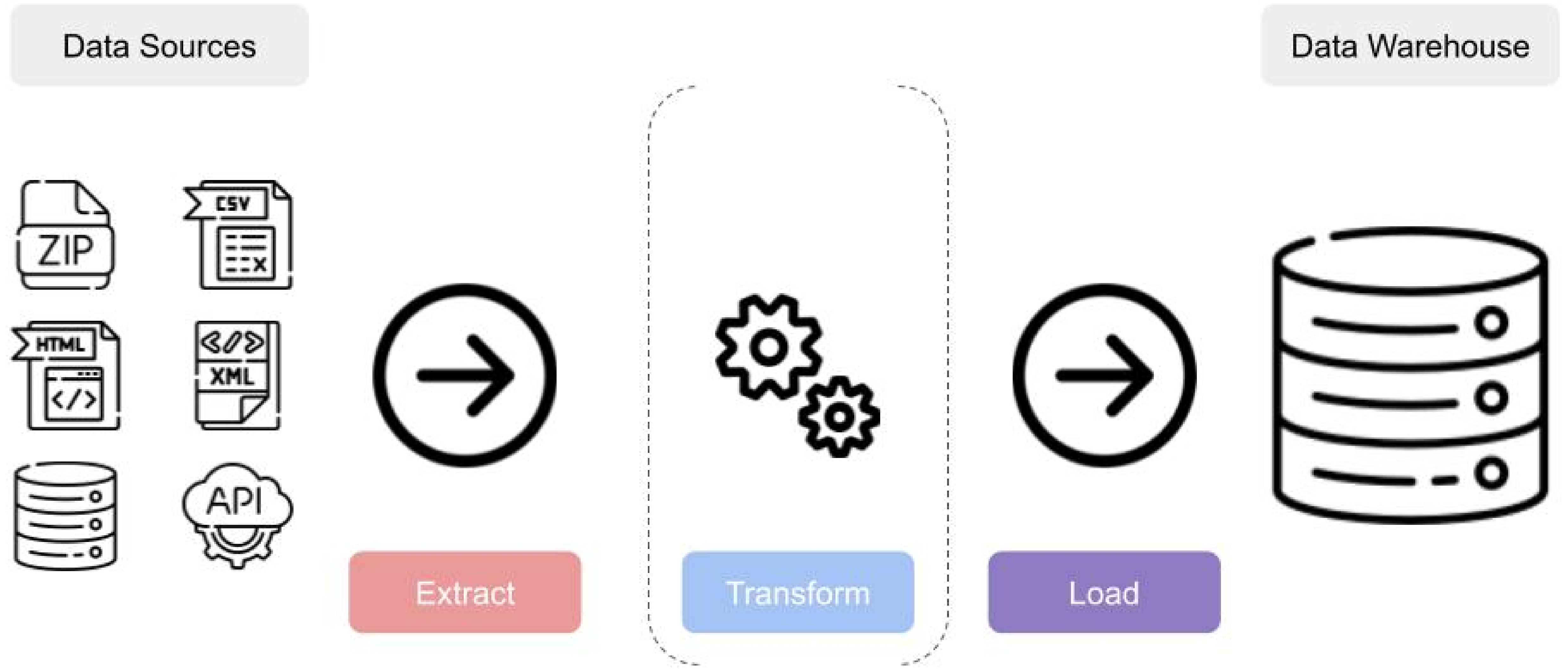
What is ETL?

Extract, Transform, Load



What is ETL?

Extract, Transform, Load

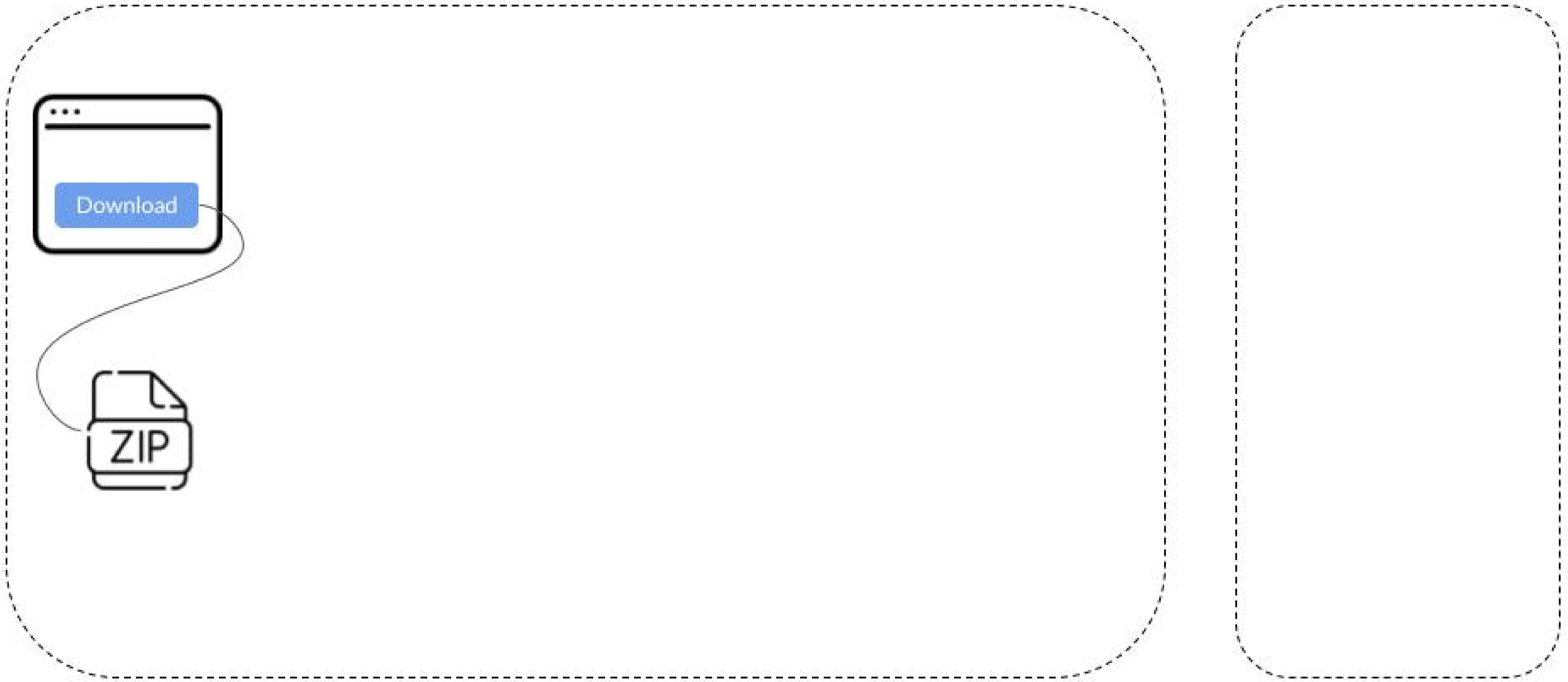


The scene

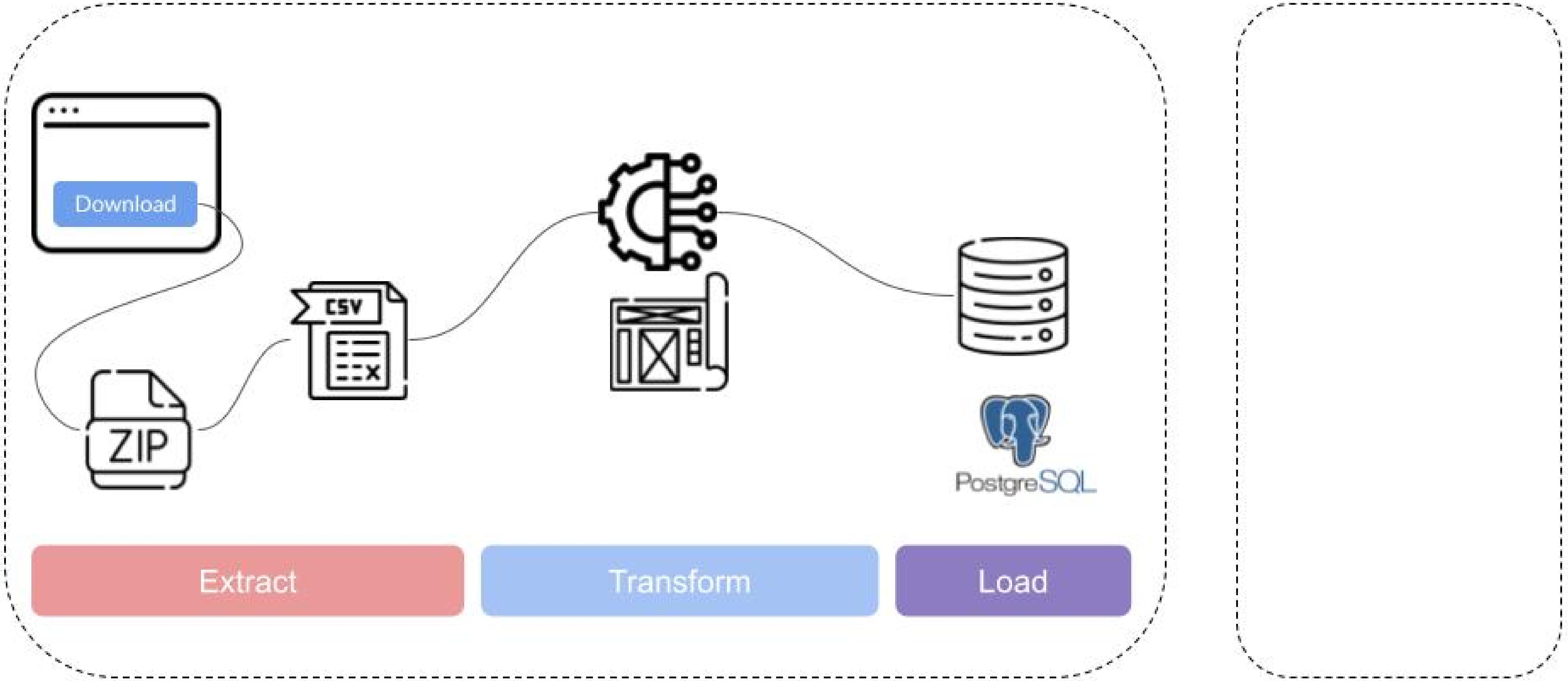
- **Private equity fund** called "DataCamp Capital Group" (DCG Capital)
- **Residential** assets
- **Monthly sales** insights
- In charge of the ETL **pipeline**
- **Stakeholder** is the business **analyst**

data
camp
capital.
GROUP

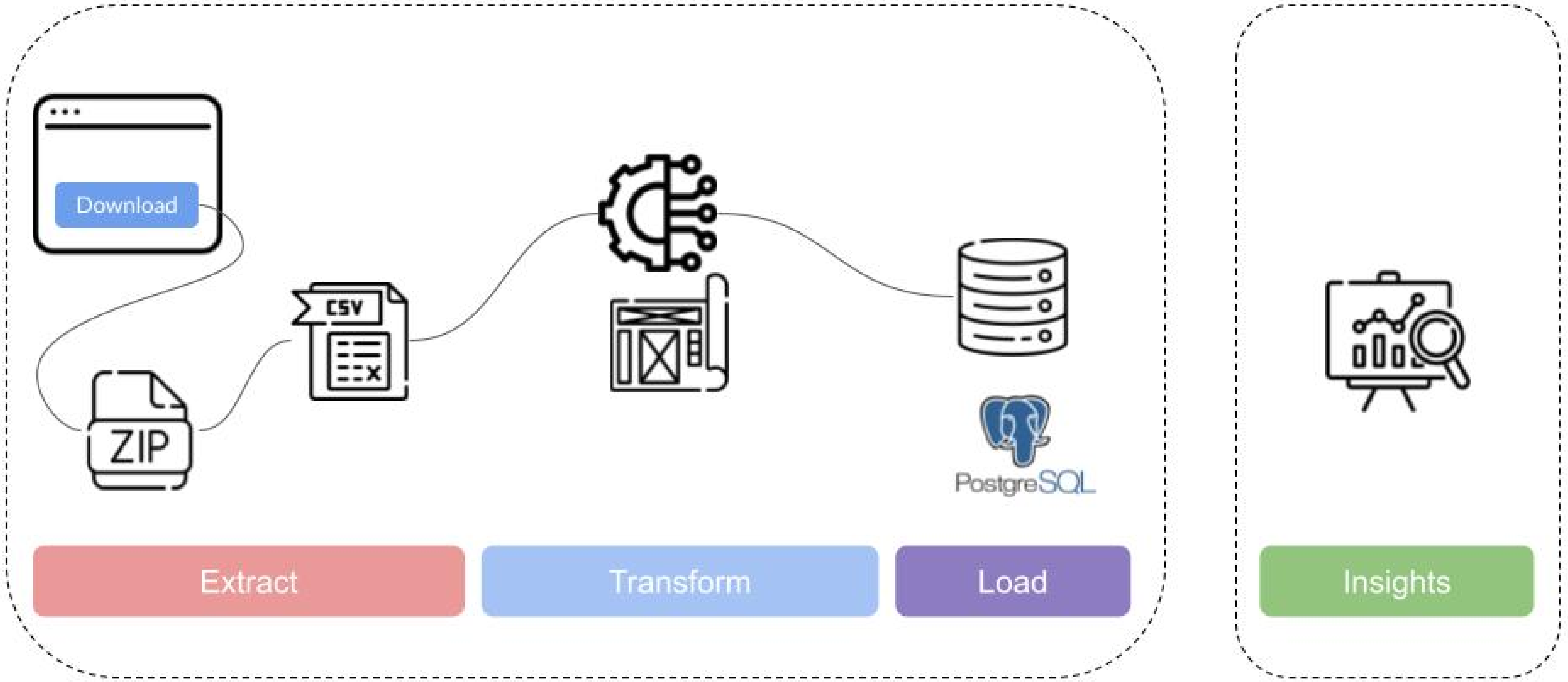
The pipeline



The pipeline

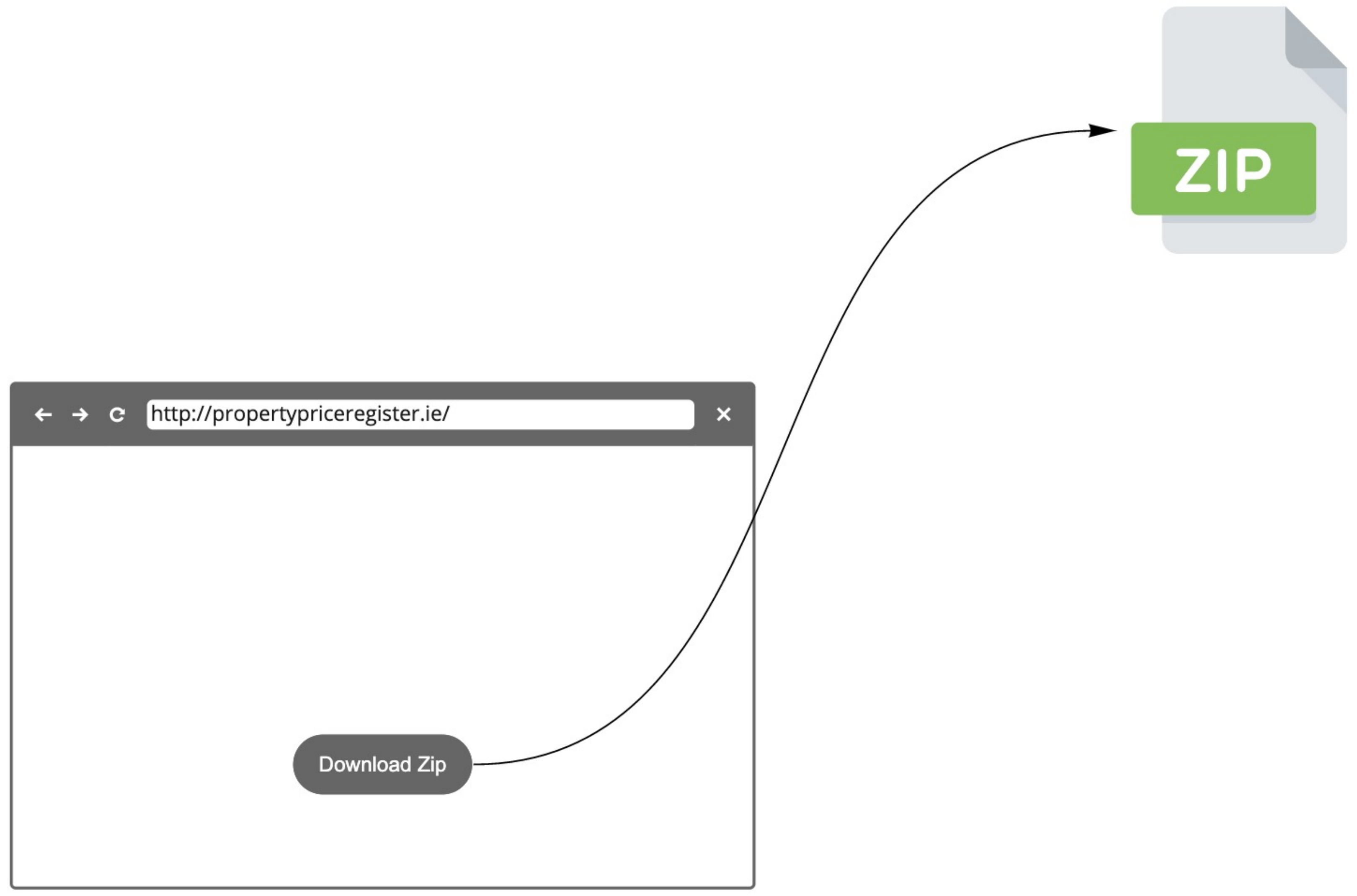


The pipeline

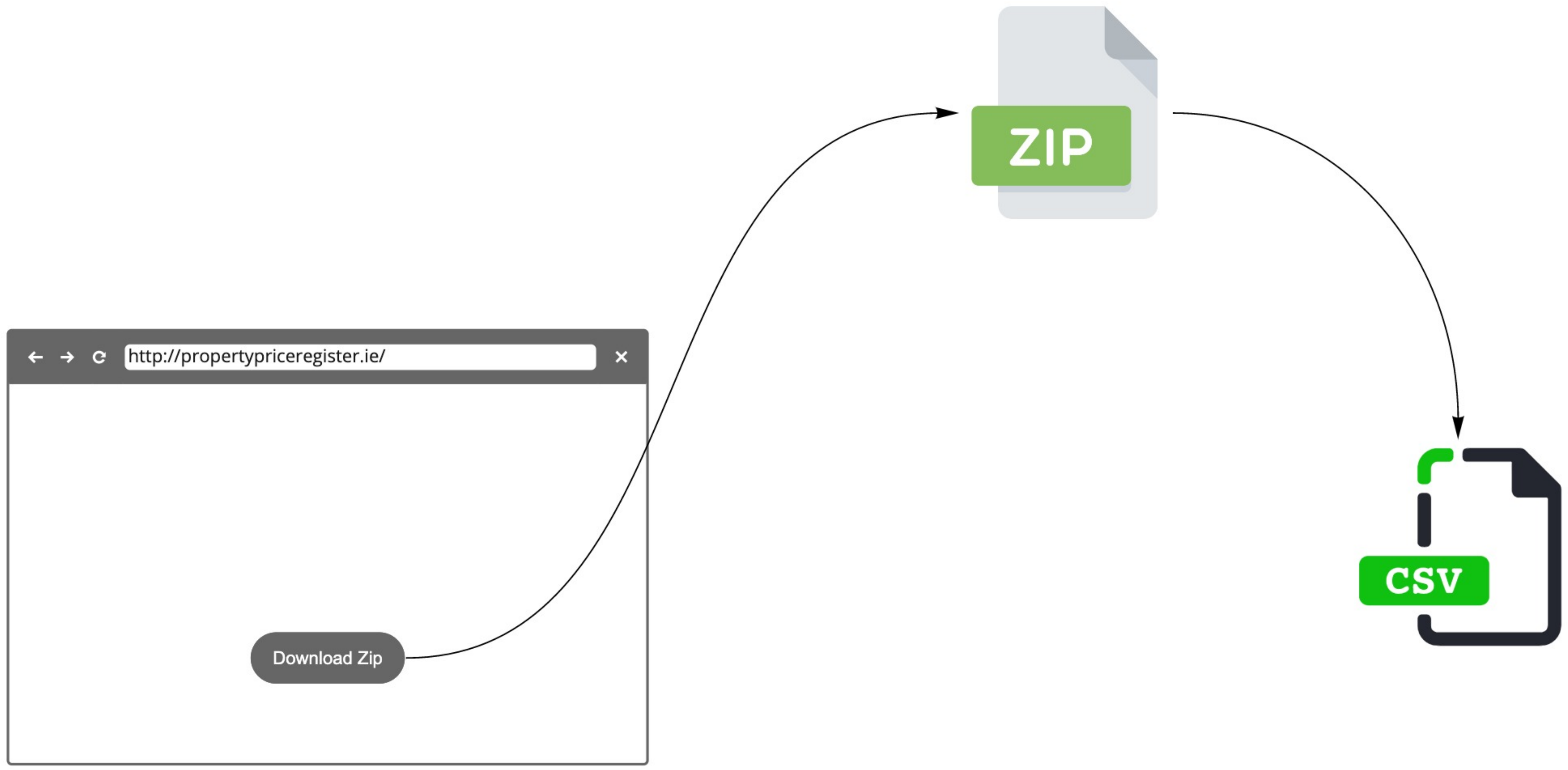


In this lesson

In this lesson



In this lesson



Requests

`GET`

- **request/fetch** data from a resource
- Social network GETs **contacts** content
- `response` : `requests.Response` object

- `requests.get('<url>')`

```python
get_url = 'https://example.com/file.zip'
response = requests.get(get_url)
```

`POST`

- **create/update** a resource
- Social network POSTs **user generated** content
- `response` : `requests.Response` object

- `requests.post('<url>', data={'key': 'value'})`

```python
post_url = 'https://example.com/page'
post_data = {'company': 'DCG Capital'}
response = requests.post(post_url,
                         data=post_data)
```

Common requests attributes

```
response = requests.get('https://example.com/ny-properties-onsale.csv')
```

```
city, address, price
New York, "441 W 37th St FLOOR 2, New York, NY 10018", "$1,700,000"
New York, "22 W 57th St #Q56, New York, NY 10019", "$3,895,000"
New York, "788 9th Ave APT 1B, New York, NY 10019", "$1,000,000"
```

Common requests attributes

Name	Output	Example
`response.content`	raw bytes response payload	b'city, address, price\nNew York...'
`response.text`	character encoded (e.g. UTF-8) string payload	'city, address, price\nNew York...'
`response.headers`	dictionary-like object which contains header payload as key-value	{ 'Date': 'Wed, 20 Oct 2021 18:49:30 GMT', 'Content-Length': '218'... }
`response.status_code`	status code returned by the external service	`200` means successful response

Zipfile

- `from zipfile import ZipFile` class
- Built-in `zipfile` function
- Commonly used with two arguments: `ZipFile(filepath, mode)`
- Read mode

```
with ZipFile(filepath, mode='r') as f:
    f.namelist()
    f.extract()
```

- `f.namelist()` returns the list of files inside the opened .zip file
- `f.extract(filename, path=None)` extracts a specific file to a specified directory

Zipfile: an example

```python
from zipfile import ZipFile


filepath = "/my/custom/path/example.zip"
with ZipFile(filepath, mode='r') as f:
    name_list = f.namelist()
    print("List of files:", name_list)
    extract_path = f.extract(name_list[0], path="/my/custom/path/")
    print("Extract Path:", extract_path)
```

```
List of files: ["example.csv"]
Extract path: "/my/custom/path/example.csv"
```

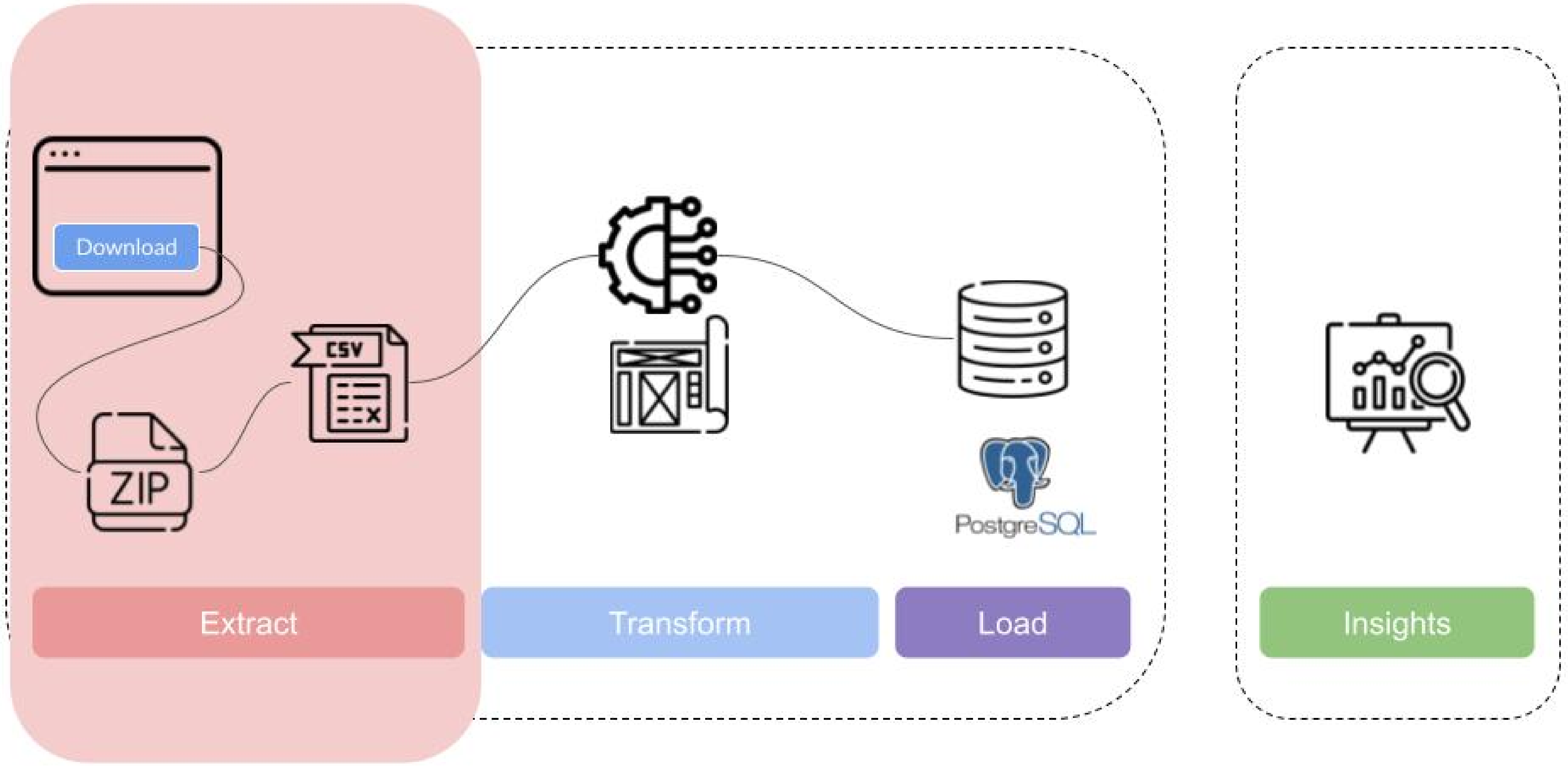
Let's practice!

Ask the right questions

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

Where we are in the pipeline



Dataset example

Date of Sale (dd/mm/yyyy)	Address	Postal Code	County	Price (€)	Description of Property
12/02/2021	123 WALKINSTOWN PARK, WALKINSTOWN, DUBLIN 12	Dublin 12	Dublin	€297,000.00	Second-Hand Dwelling house /Apartment
04/01/2021	12 Oileain Na Cranoige.Cranogue Isl, Balbutcher Lane, BALLYMUN	Dublin 11	Dublin	€192,951.00	New Dwelling house /Apartment

Open a file

- Built-in `open()` function
- Commonly used with 2 arguments: `open(filepath, mode)`
- Most common `mode` :

Character	Meaning
'r'	open for reading (default)
'w'	open for writing

- `encoding` argument example: `open("file.csv", mode="r", encoding="windows-1252")`

Open a file: example

Read mode

```python
with open('file.csv', mode="r", encoding="windows-1252"):
    # Code here
```

Write mode

```python
with open('file.csv', mode="w", encoding="windows-1252"):
    # Code here
```

CSV module

- `csv` implements classes to **read** and **write** tabular data in CSV format
- **Dictionary** form with `csv.DictReader()` and `csv.DictWriter()` functions
  - `csv.DictReader(file, fieldnames=None...)`
  - `csv.DictWriter(file, fieldnames, ...)`
- keys = column names
- values = row values

Read in action

Code

```python
with open("file.csv", mode="r") as csv_file:
    reader = csv.DictReader(csv_file)
    row = next(reader)
    print(row)
```

Output

```
OrderedDict([
('Date of Sale (dd/mm/yyyy)', '03/01/2021'),('Postal Code', 'Dublin 4'),
('Address', '16 BURLEIGH COURT, BURLINGTON ROAD, DUBLIN 4'),('County', 'Dublin'),
('Price (€)', '€450,000.00'), ...])
```

Write in action

Code

```python
with open("file.csv", mode="w") as csv_file:
    new_column_names = {"Date of Sale (dd/mm/yyyy)": "date_of_sale",
        "Address": "address", "Postal Code": "postal_code", "County": "county",
        "Price (€)": "price", "Description of Property": "description"}
    writer = csv.DictWriter(csv_file, fieldnames=new_column_names)
    # Write headers as first line
    writer.writeheader()
    # Write all rows in file
    for row in reader:
        writer.writerow(row)
```

Let's practice!

ETL IN PYTHON

Extracting

ETL IN PYTHON

Stefano Francavilla
CEO - Geowox

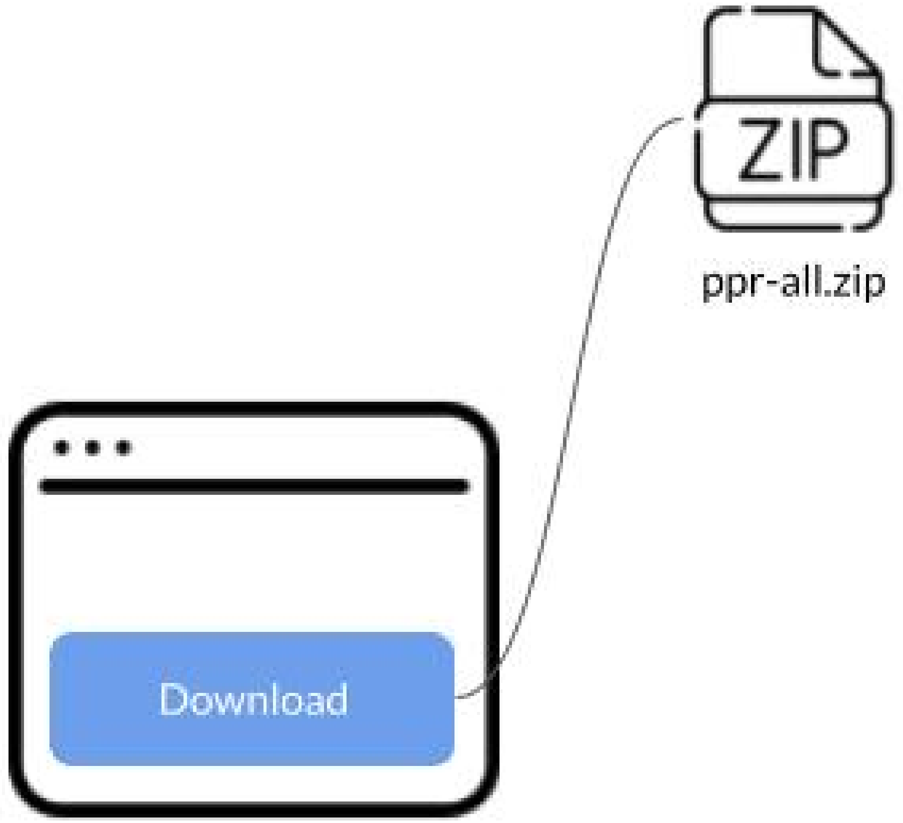
End goal

Automated pipeline

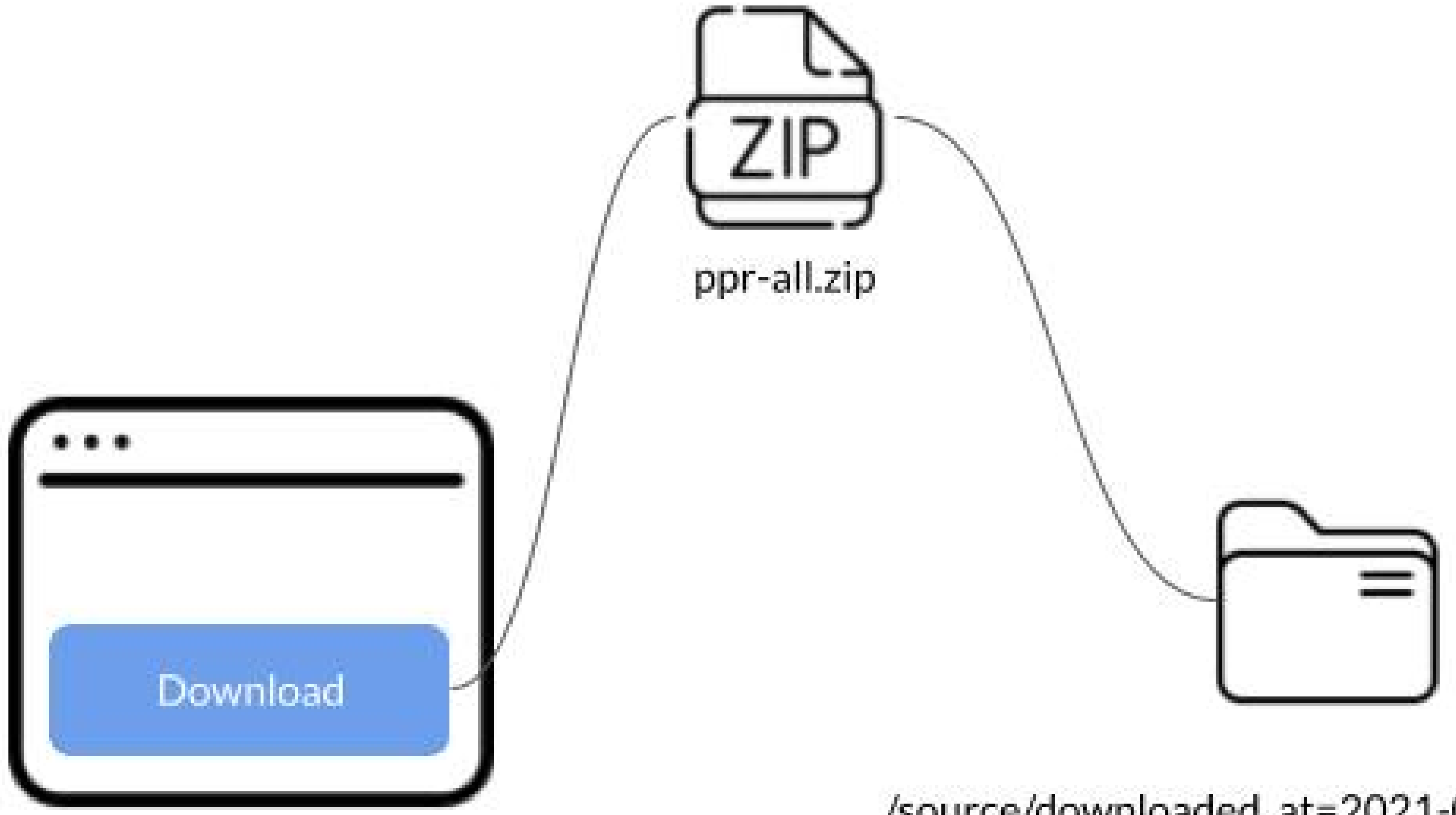
- `cron`
  - Command line utility used for scheduling
- `execute.py`
  1. `extract.py`
  2. `transform.py`
  3. `load.py`
- Download and process property transactions

E(xtract)TL

E(xtract)TL

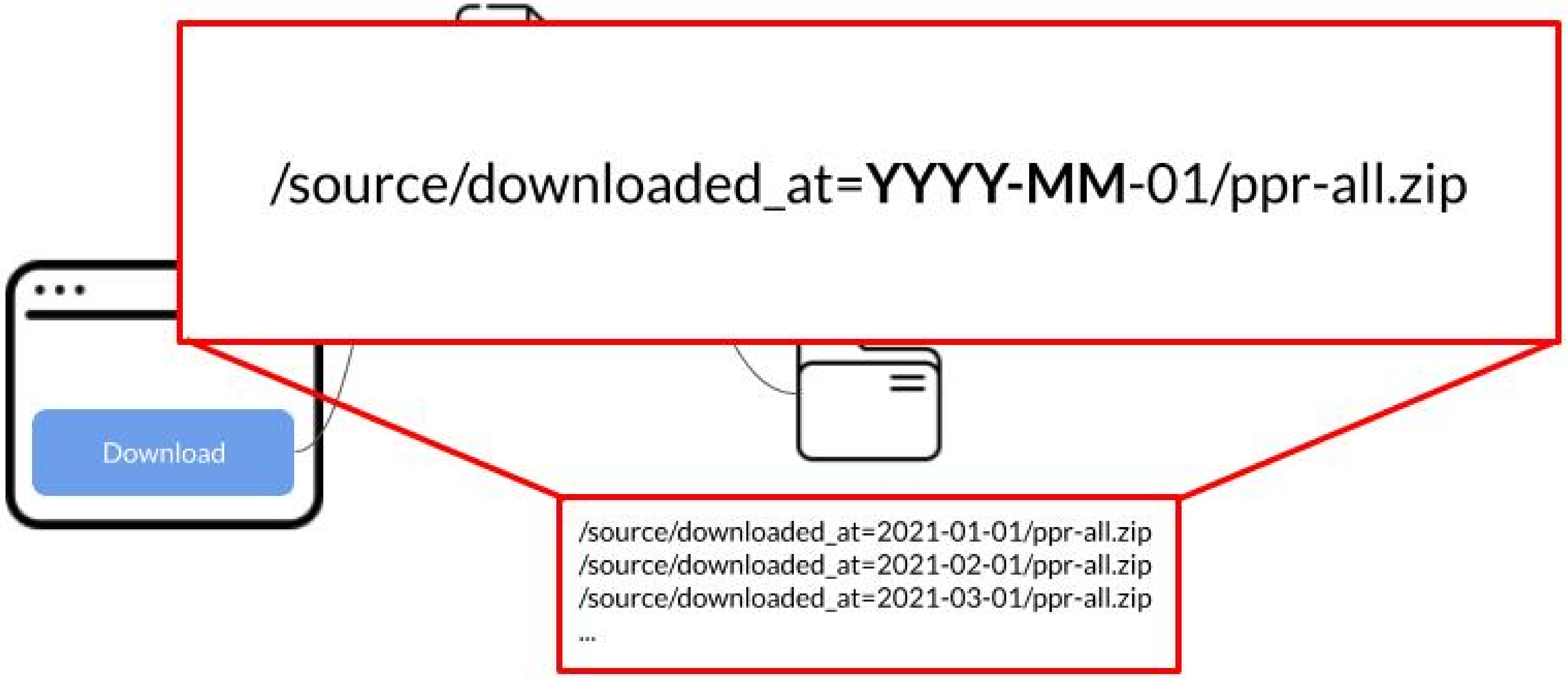


E(xtract)TL



/source/downloaded_at=2021-01-01/ppr-all.zip
/source/downloaded_at=2021-02-01/ppr-all.zip
/source/downloaded_at=2021-03-01/ppr-all.zip
...

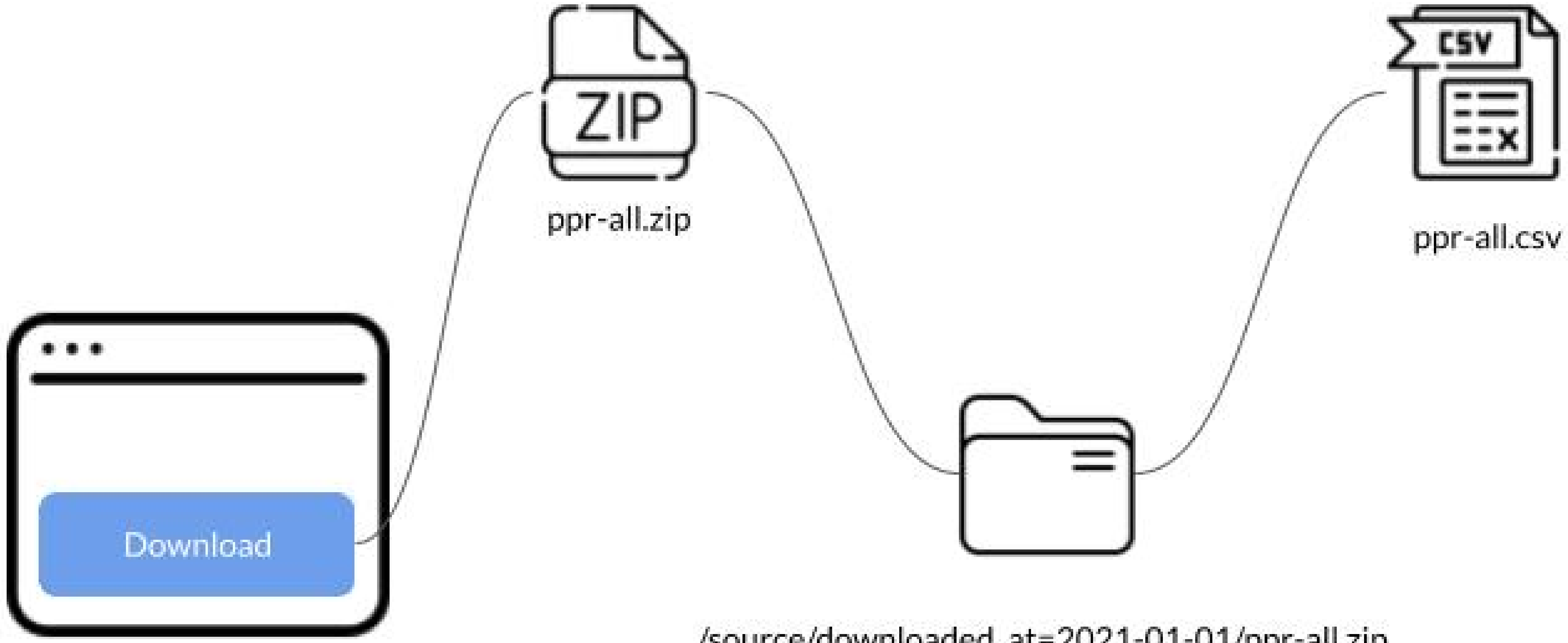
E(xtract)TL



/source/downloaded_at=**YYYY-MM**-01/ppr-all.zip

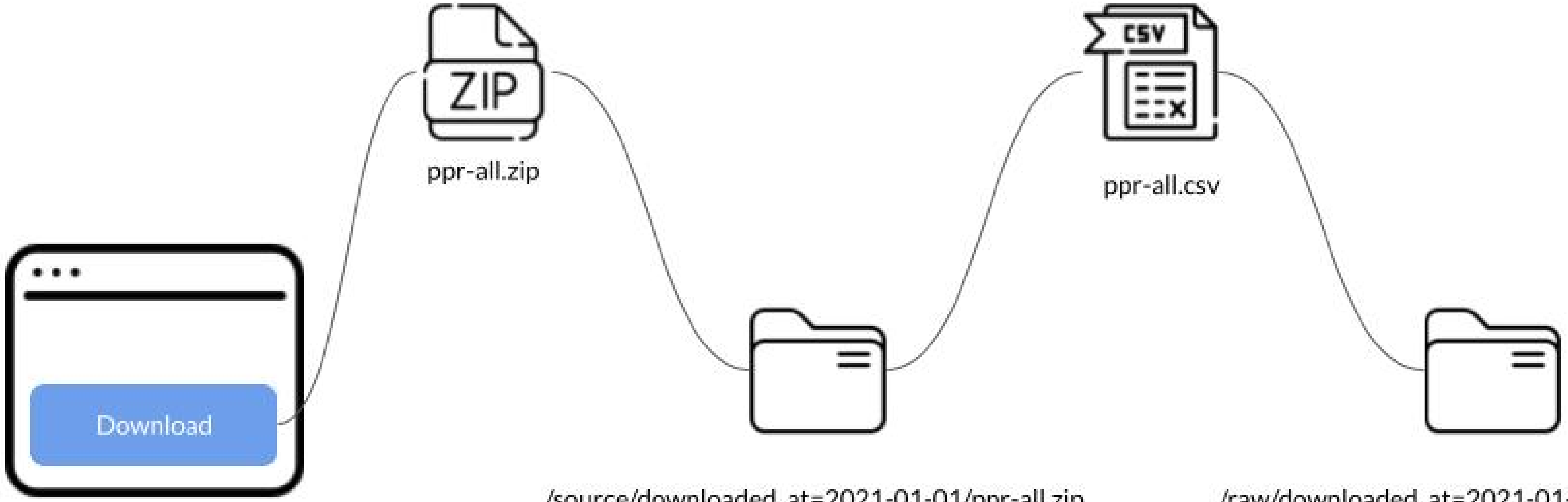
/source/downloaded_at=2021-01-01/ppr-all.zip
/source/downloaded_at=2021-02-01/ppr-all.zip
/source/downloaded_at=2021-03-01/ppr-all.zip
...

E(xtract)TL



/source/downloaded_at=2021-01-01/ppr-all.zip
/source/downloaded_at=2021-02-01/ppr-all.zip
/source/downloaded_at=2021-03-01/ppr-all.zip
...

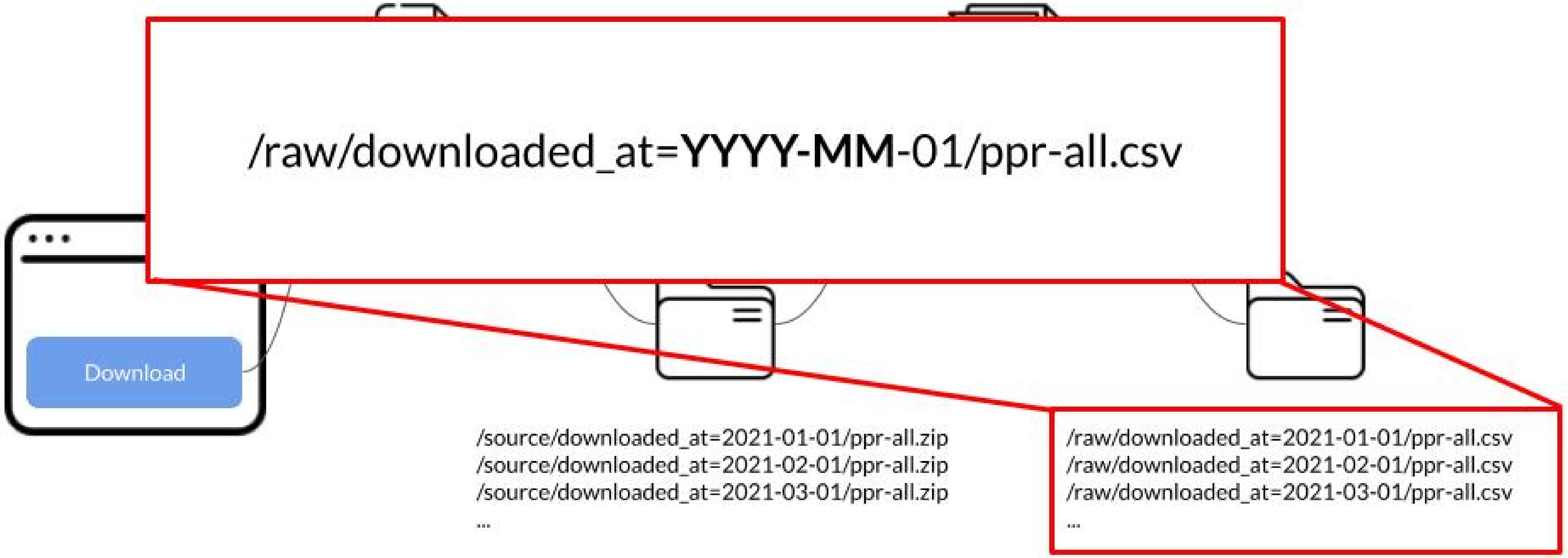
E(xtract)TL



/source/downloaded_at=2021-01-01/ppr-all.zip
/source/downloaded_at=2021-02-01/ppr-all.zip
/source/downloaded_at=2021-03-01/ppr-all.zip
...

/raw/downloaded_at=2021-01-01/ppr-all.csv
/raw/downloaded_at=2021-02-01/ppr-all.csv
/raw/downloaded_at=2021-03-01/ppr-all.csv
...

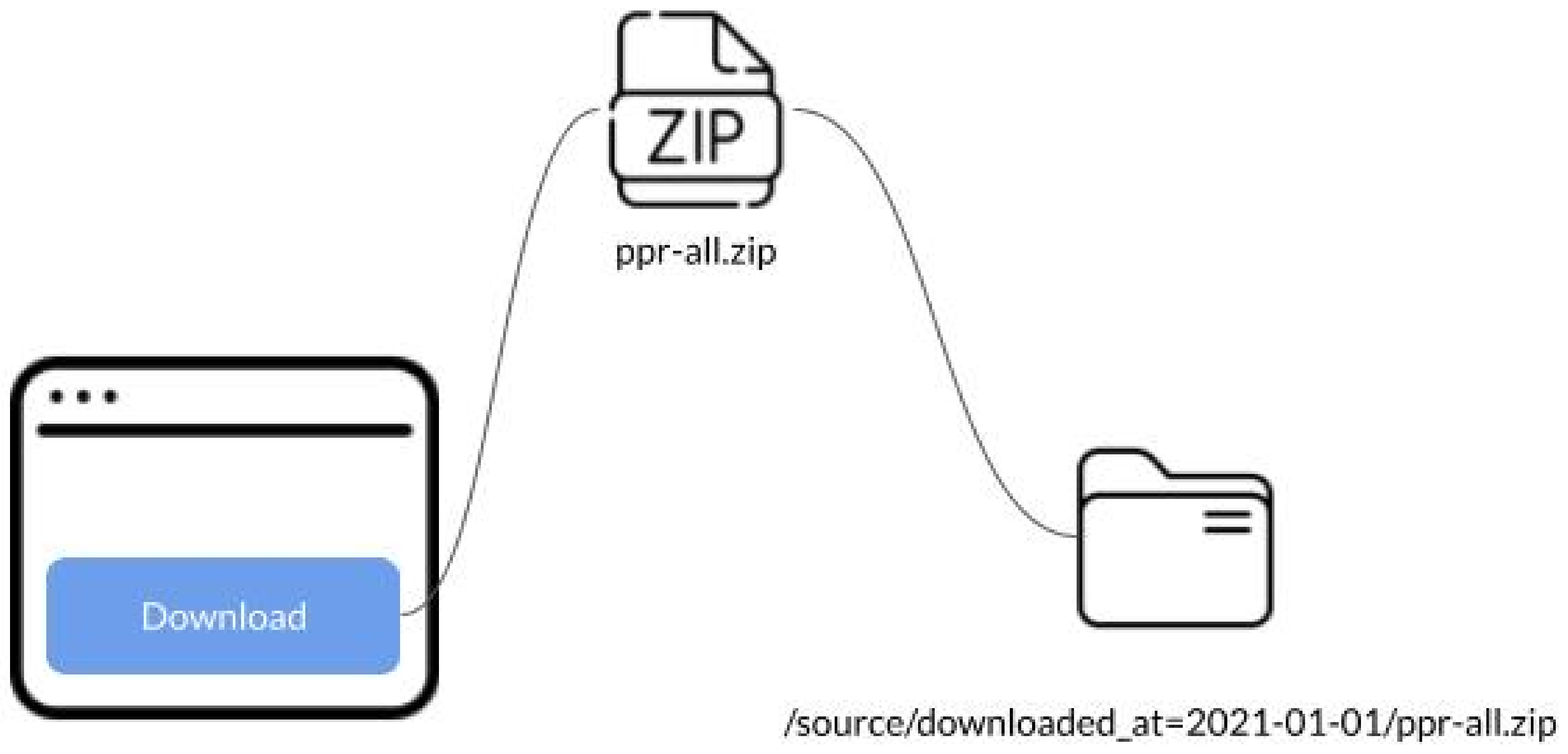
E(xtract)TL



/source/downloaded_at=2021-01-01/ppr-all.zip
/source/downloaded_at=2021-02-01/ppr-all.zip
/source/downloaded_at=2021-03-01/ppr-all.zip
...

/raw/downloaded_at=2021-01-01/ppr-all.csv
/raw/downloaded_at=2021-02-01/ppr-all.csv
/raw/downloaded_at=2021-03-01/ppr-all.csv
...

In this lesson: E(xtract)



Create a folder

- Make sure the `downloaded_at` folder exists
- `import os`
  - Allows Python to **interact** with the **operating system**
- `os.makedirs()` creates a folder **recursively**
  - **Missing** intermediate-level **directories are created** as well
- `os.makedirs(path, exist_ok=[True|False])`

Create a folder: an example

- January 1st, 2021
  - first time we run the cron job
- Save the `.zip` file in the current month directory: `<root>/source/downloaded_at=2021-02-01`
  - create `.../downloaded_at=2021-01-01` folder
  - but... `.../source` folder does not exist yet

Create a folder: an example

```
# Create <root>/source/downloaded_at=2021-01-01
path = "root/source/downloaded_at=2021-01-01"
os.makedirs(path, exist_ok=True)
# 1. Create source
# 2. Create downloaded_at=2021-01-01
```

```
/source/downloaded_at=2021-01-01/<zipfile_name>.zip
```

Save ZIP file locally

- `open()`
  - Commonly used with two arguments: `open(filepath, mode)`
- Text vs binary `mode` :

Character	Meaning
'w'	open for writing in **text** format
'wb'	open for writing in **binary** format

Write binary mode

```
with open('source/downloaded_at.../ppr-all.zip', mode="wb") as f:
    f.write(...)
```

Let's practice!

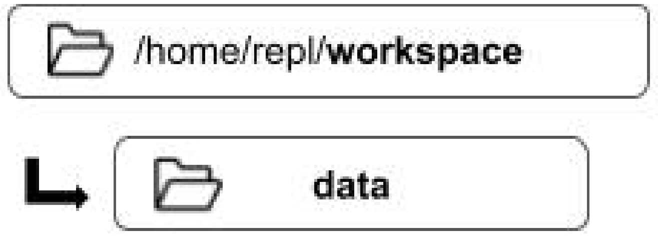
ETL IN PYTHON

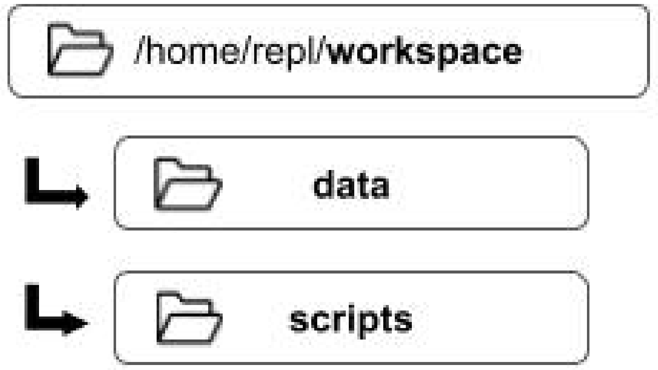
Project folder structure

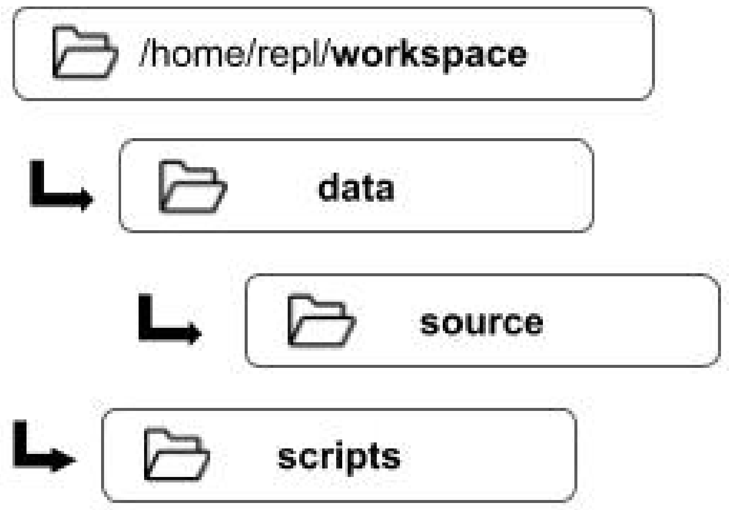
ETL IN PYTHON

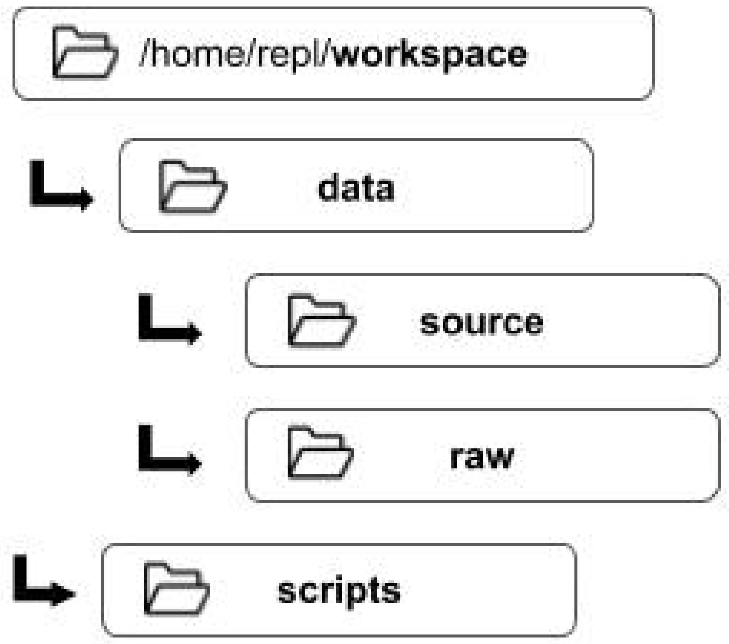
Stefano Francavilla
CEO Geowox

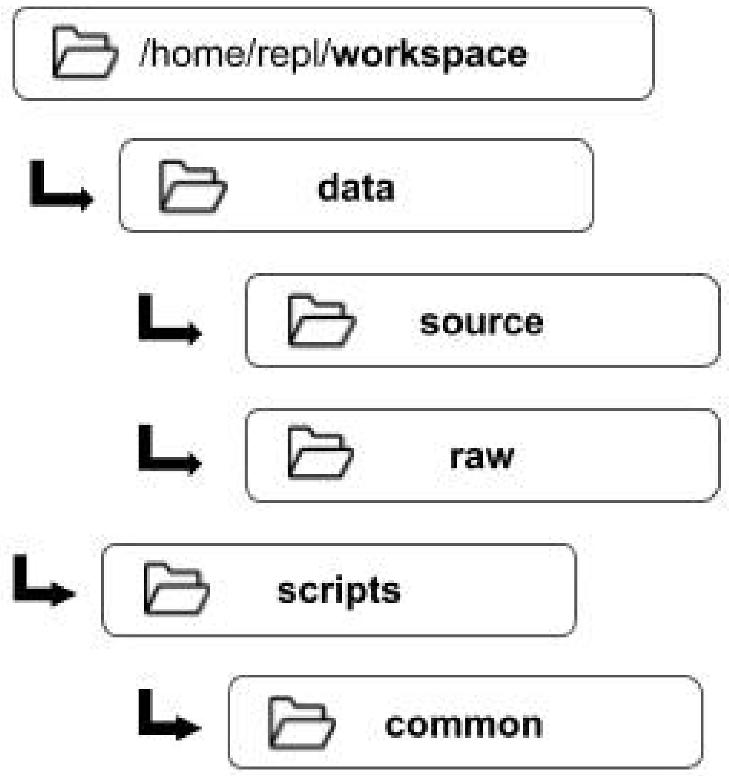
/home/repl/**workspace**

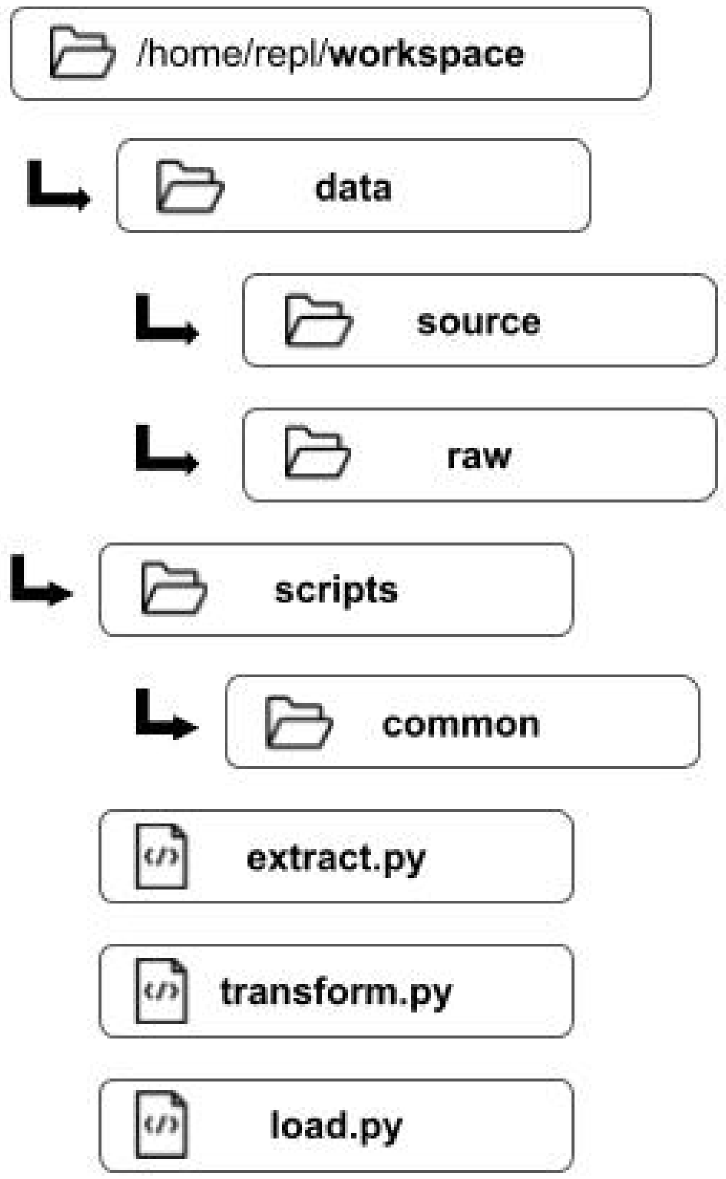

/home/repl/workspace
data

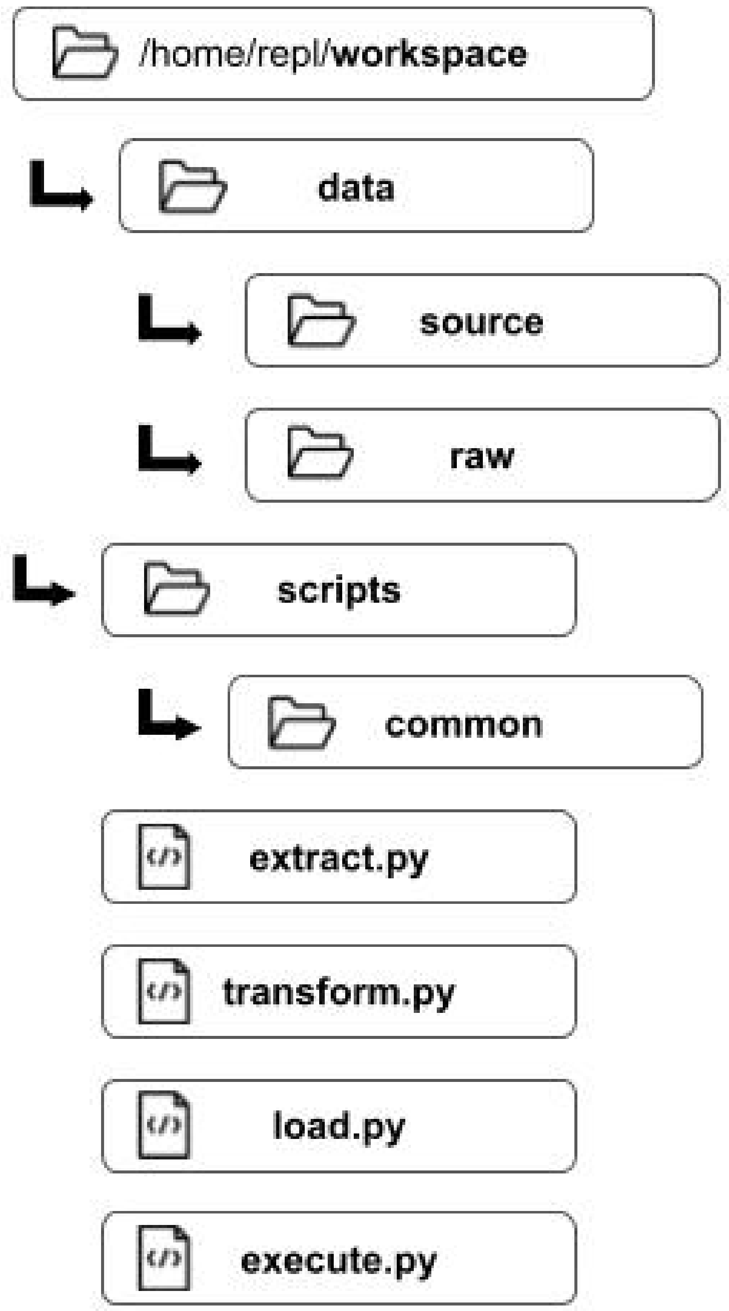

/home/repl/workspace
data
scripts

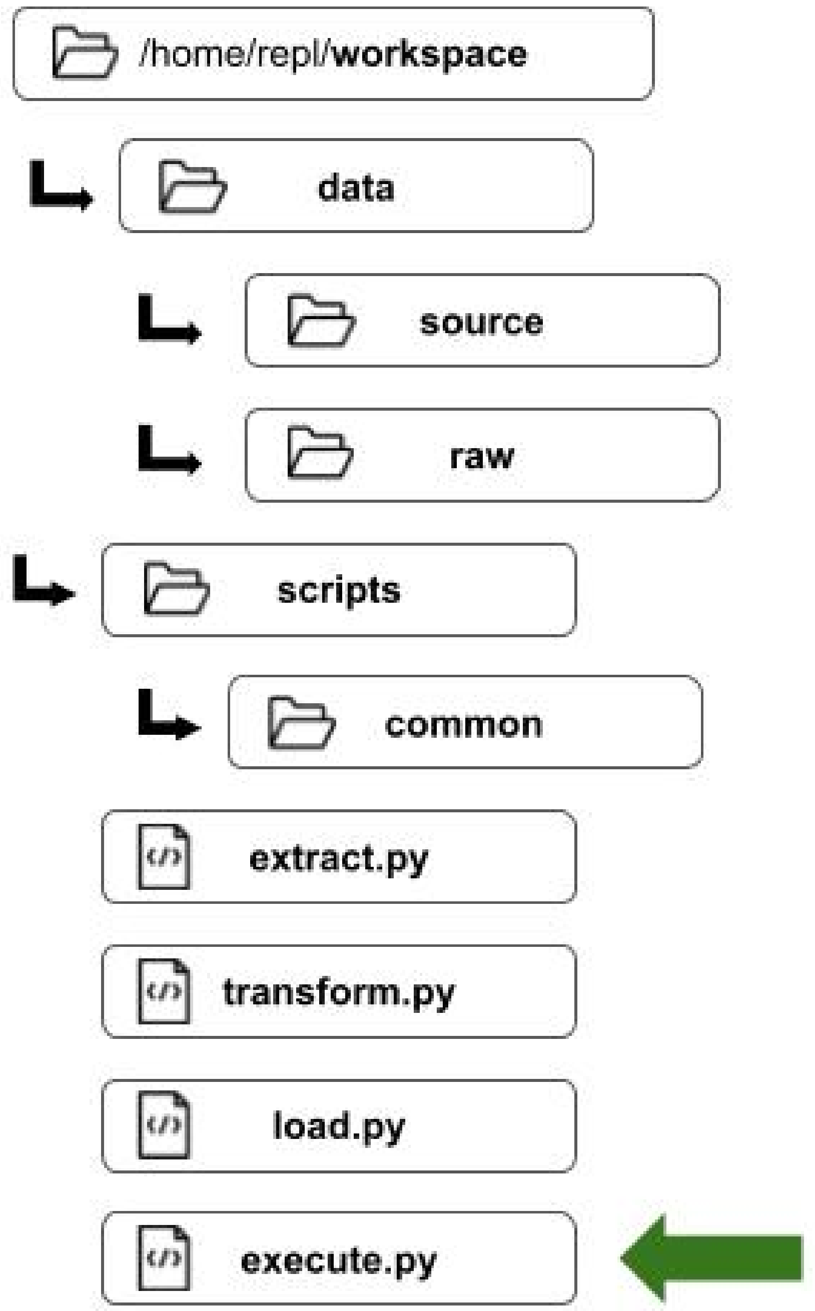

/home/repl/workspace
data
source
scripts

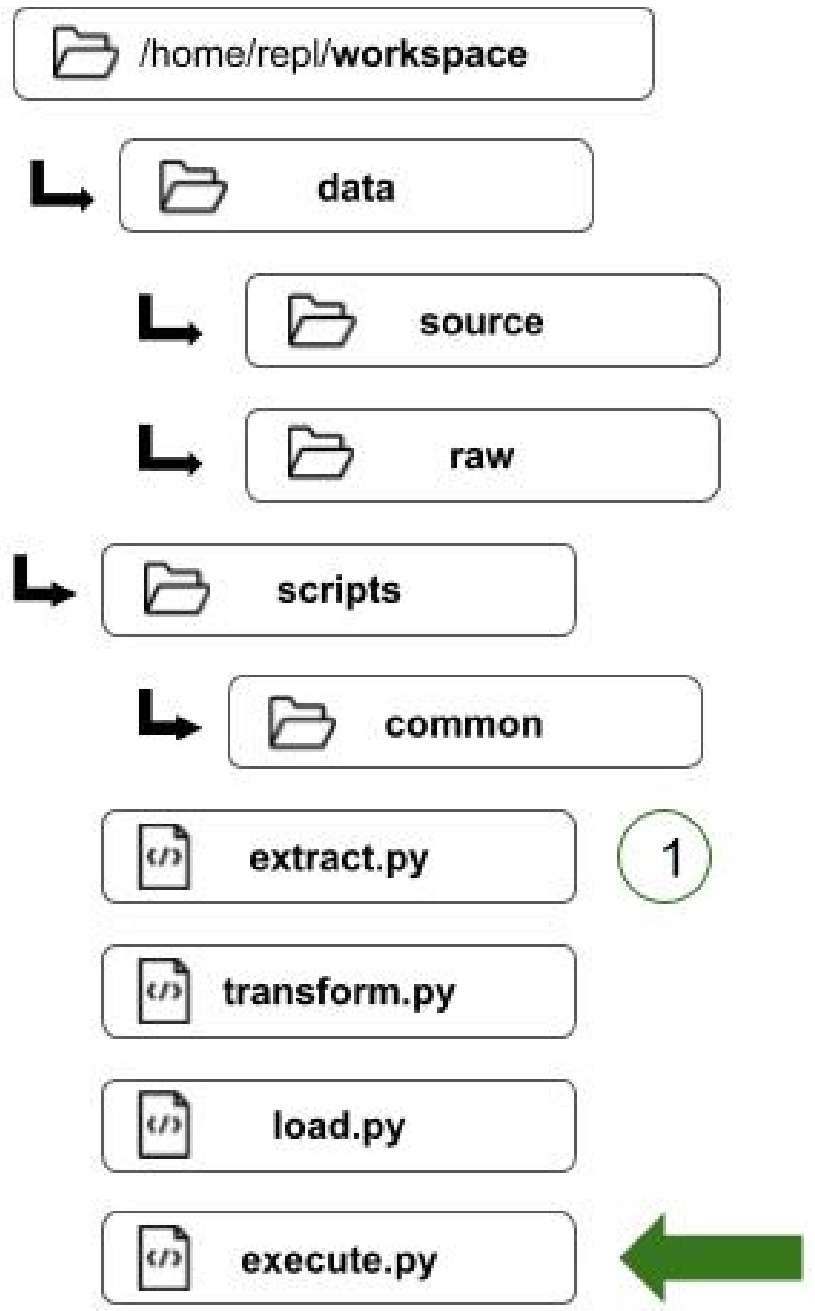

/home/repl/workspace
data
source
raw
scripts

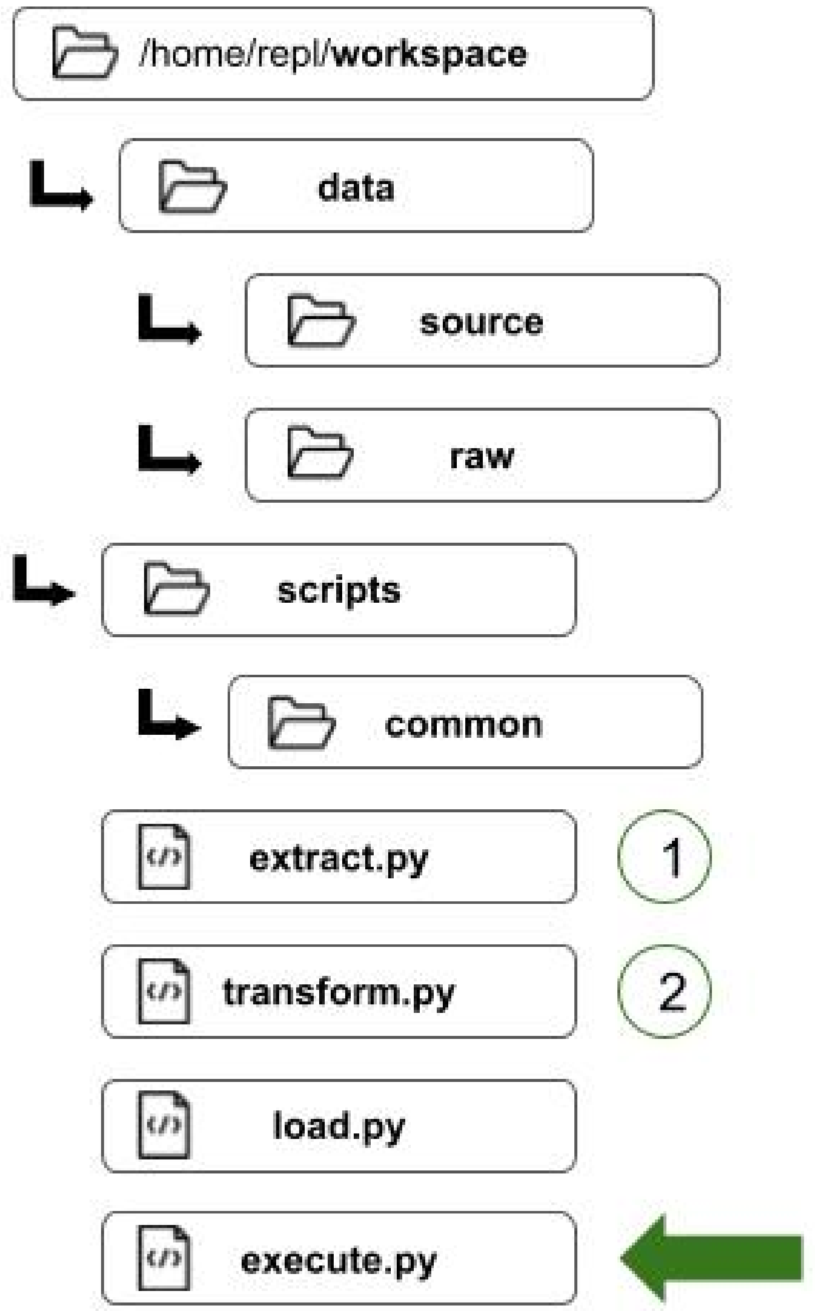

/home/repl/workspace
data
source
raw
scripts
common

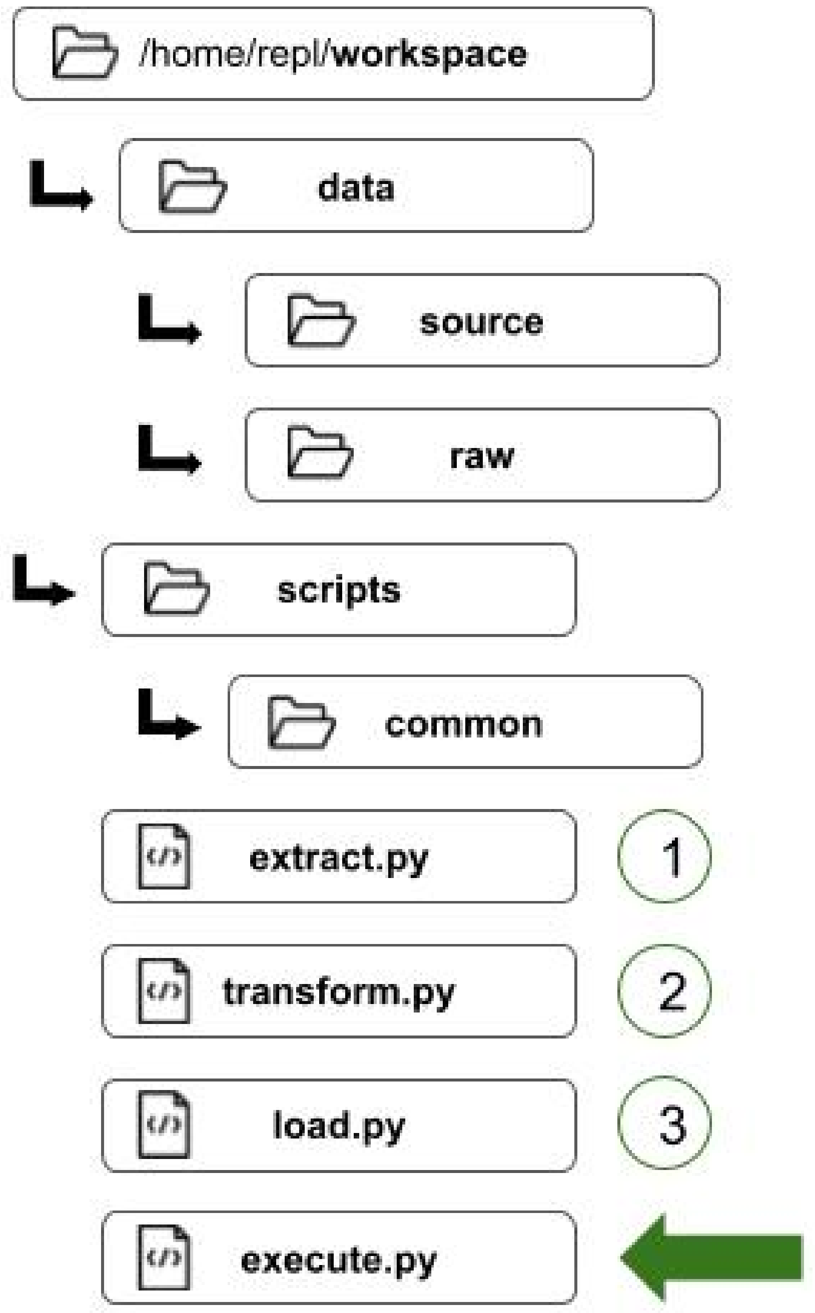

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py

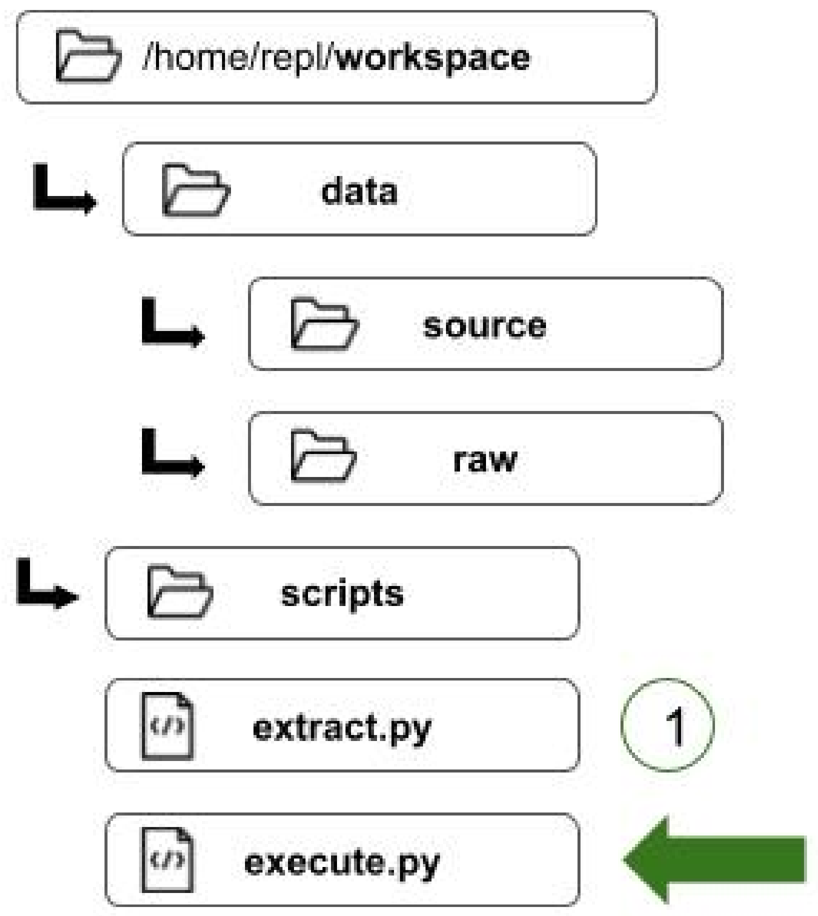

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py
execute.py

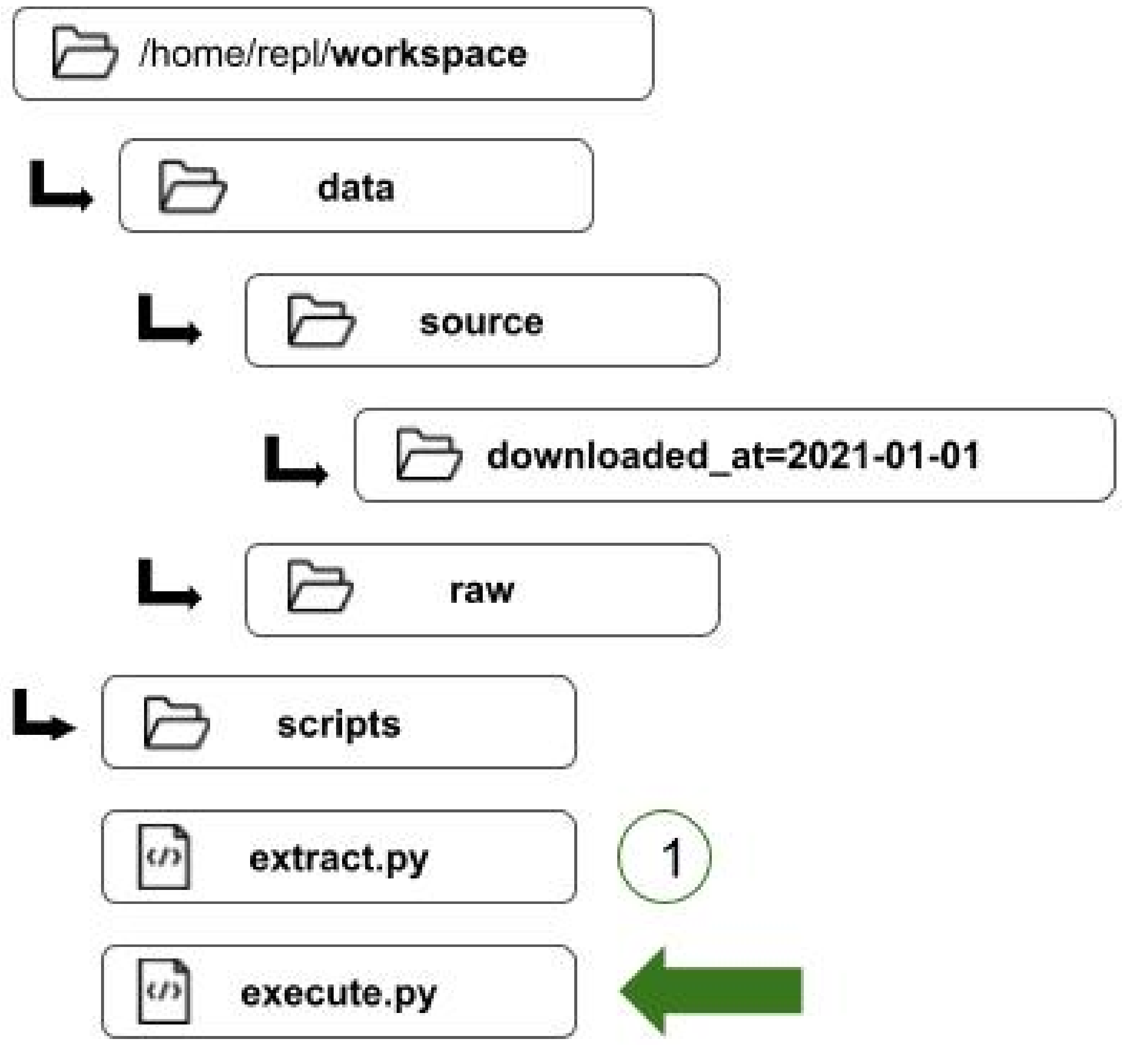

/home/repl/workspace
data
source
raw
scripts
common
extract.py
transform.py
load.py
execute.py

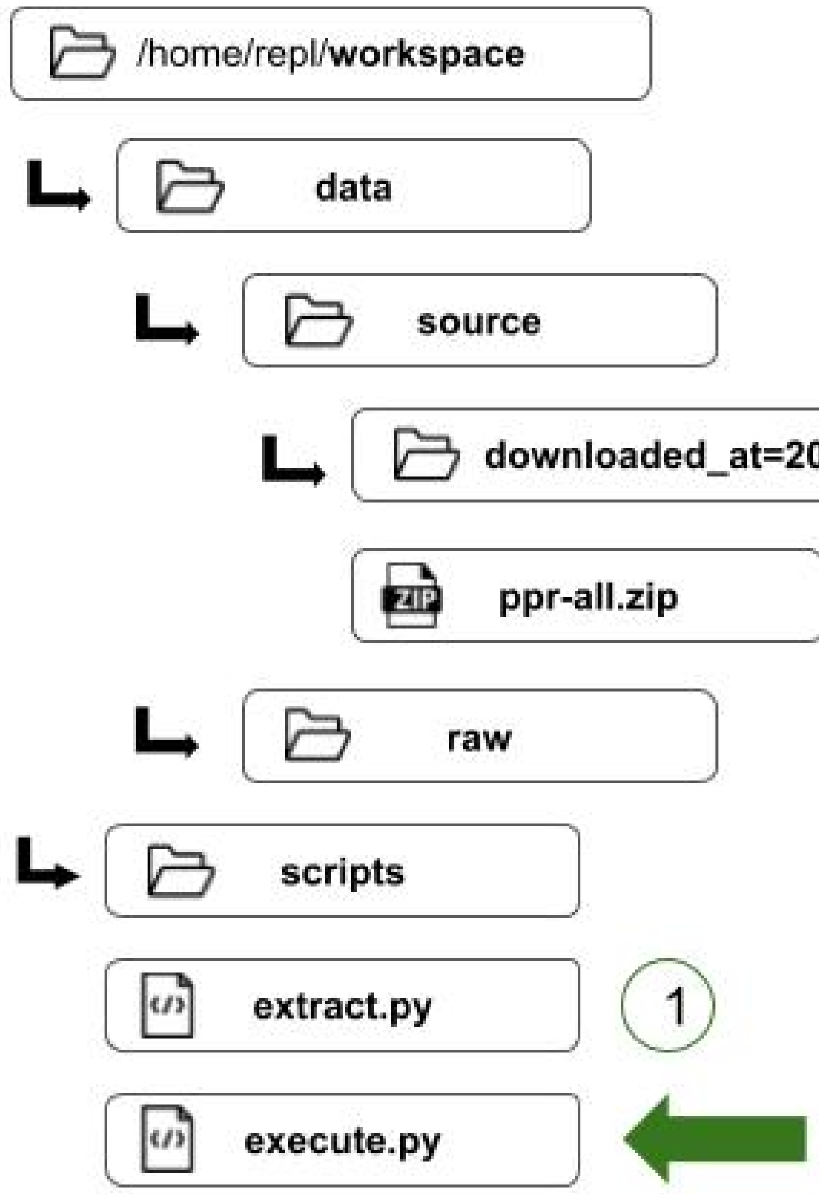

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
load.py
execute.py

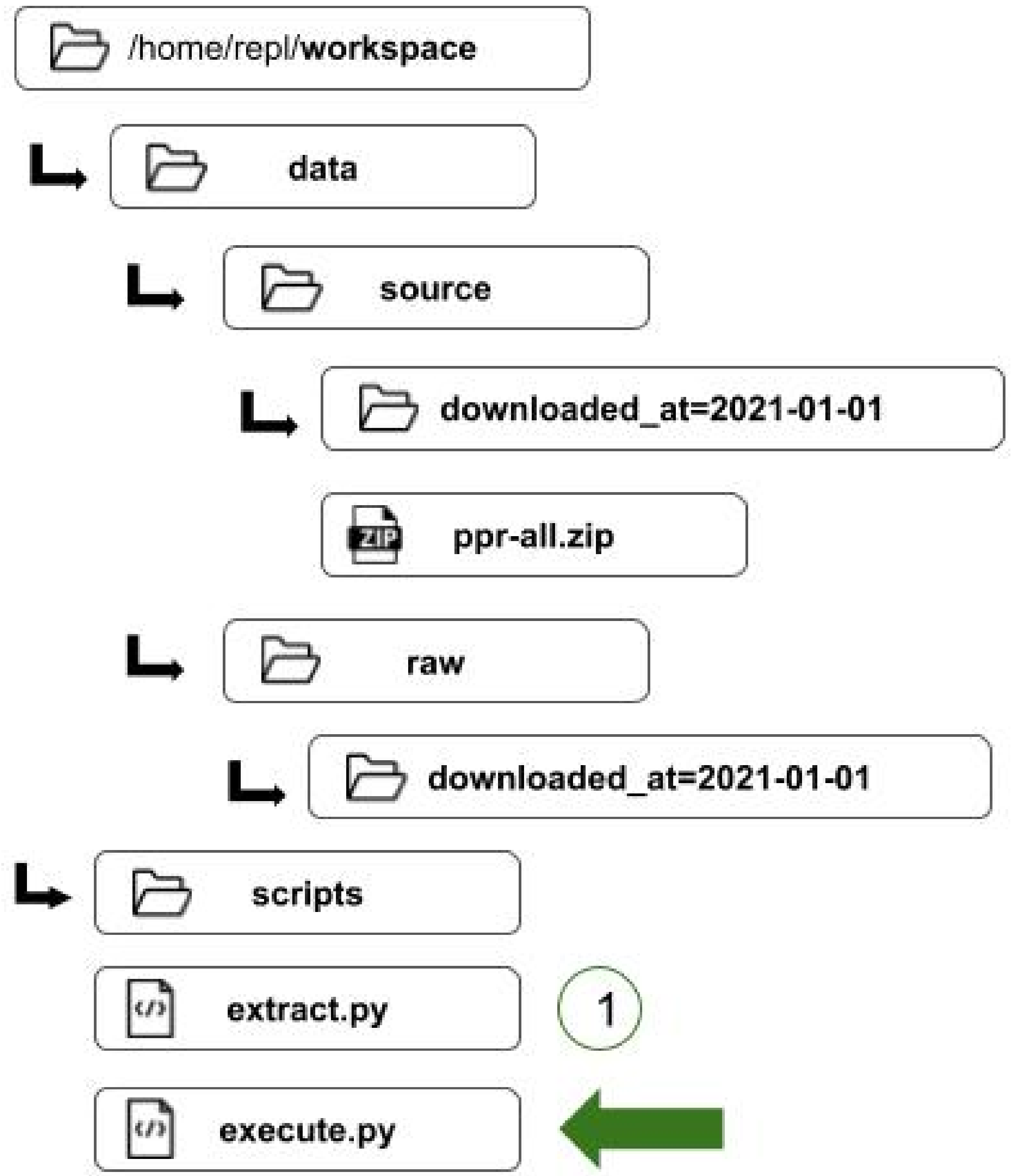

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
2
load.py
execute.py

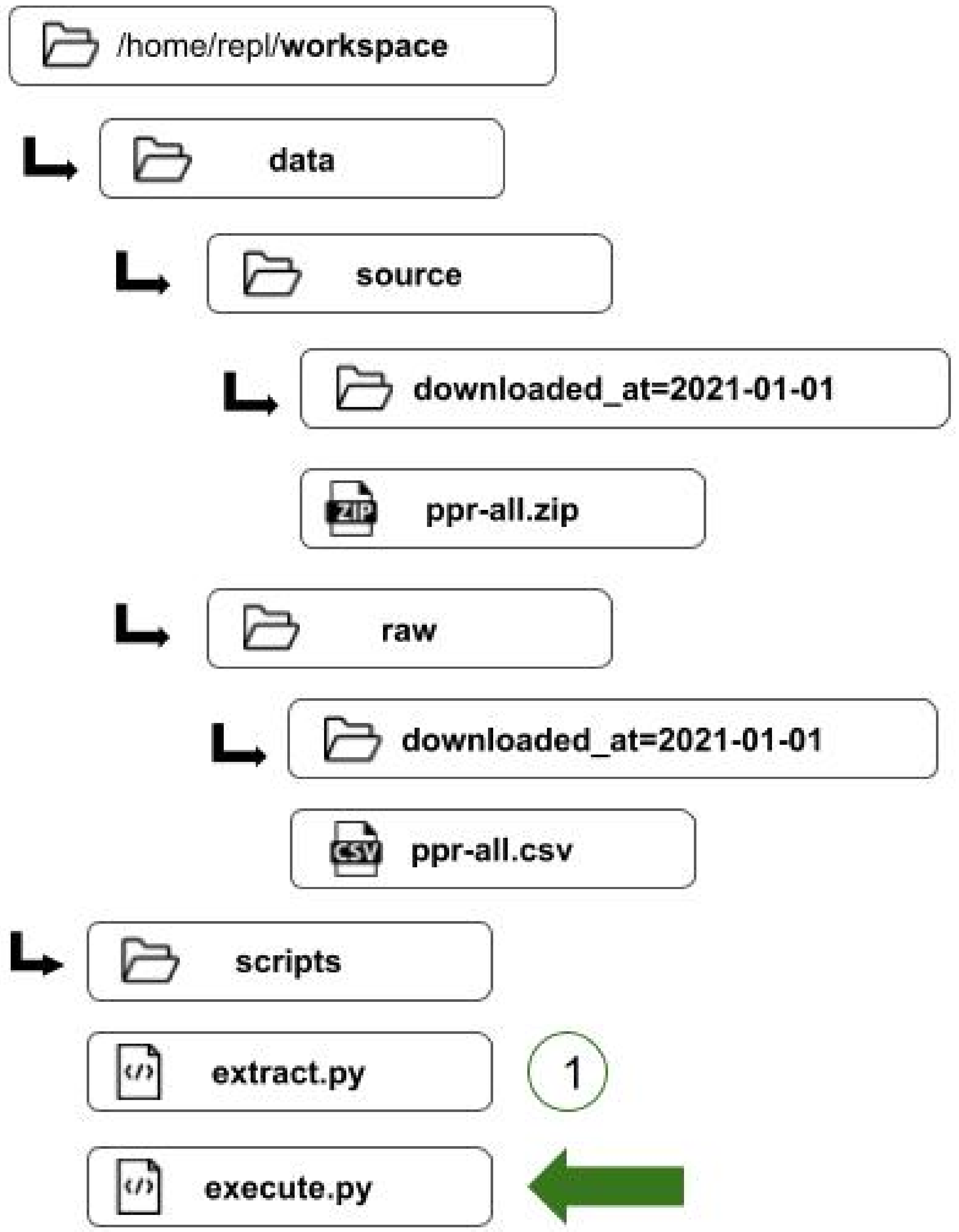

/home/repl/workspace
data
source
raw
scripts
common
extract.py
1
transform.py
2
load.py
3
execute.py


/home/repl/workspace
data
source
raw
scripts
extract.py
1
execute.py


/home/repl/workspace
data
source
downloaded_at=2021-01-01
raw
scripts
extract.py
1
execute.py


/home/repl/workspace
data
source
downloaded_at=2021-01-01
ppr-all.zip
raw
scripts
extract.py
1
execute.py


/home/repl/workspace
data
source
downloaded_at=2021-01-01
ppr-all.zip
raw
downloaded_at=2021-01-01
scripts
extract.py
1
execute.py


/home/repl/workspace
data
source
downloaded_at=2021-01-01
ppr-all.zip
raw
downloaded_at=2021-01-01
ppr-all.csv
scripts
extract.py
1
execute.py

Extract, Transform and Load

```
# Import libraries

def methodX():
    # Code here
    pass
def methodY():
    # Code here
    pass

def main():
     methodX()
     methodY()
```

Execute

```python
# Import extract, transform and load
import extract, transform, load

# Ensure execute.py can only be ran from bash
if __name__ == "__main__":
    # 1. Run Extract
    extract.main()
    # 2. Run Transform
    transform.main()
    # 3. Run Load
    load.main()
```

```
python execute.py
```

Let's practice!

ETL IN PYTHON